Distributional Modelling and Control of Large-Scale Passive Swarm Robots

Seth J. Lim

June 3, 2024

Distributional Modelling and Control of Large-Scale Passive Swarm Robots

by

Seth J. Lim

A dissertation submitted to

The University of Sheffield

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

June 3, 2024

Seth J. Lim

Distributional Modelling and Control of Large-Scale Passive Swarm Robots

PhD Dissertation, June 3, 2024

Supervisors:	Prof Visakan Kadirkamanathan
	Dr Roderich Groß

The University of Sheffield

Department of Automatic Control and Systems Engineering

Amy Johnson Building

Portobello Street

Sheffield, S1 3JD

Acknowledgements

I would like to express my appreciation to a number of people who have supported me throughout this PhD project.

First and foremost, I would like to express my sincerest gratitude to my supervisor, Prof. Visakan Kadirkamanathan, for his active and patient guidance throughout the entirety of my project with his seemingly limitless knowledge and great passion towards his field.

I would also like to express my deepest appreciation to Dr. Yuanbo Nie, for his supportive guidance throughout the project and his valuable feedback.

Last but certainly not least, I would like to express my deepest regards to my dear family. To my kind and compassionate sister, Olivia, for her continual support and inspiration. To my always gracious mother, who unconditionally supports me regardless of what I do. And lastly, to my generous father, who has always impelled me to take on challenges and realise my dreams.

Abstract

The overall aim of this research is to address some of the challenges for modelling and controlling a large number of particle swarm robots. The inspiration of this research was the control of microrobots, in which because of their tiny hardware space, have limited computational capabilities and are reliant on external magnetic field, controlled by a central distributional controller, for movement to fulfil their objectives. The probability distribution to represent the behaviour of a large scale swarm of robots and a representation of the potential field are defined, which are parametrised through Gaussian basis functions. A swarm distributional states estimation algorithm, modified from the Expectation-Maximisation algorithm, is also presented to calculate the swarm distributional states. To represent the swarm robots as a probability distribution, a continuum representation of the advection-diffusion model is first considered and a reduced-order model of the advection-diffusion is derived using the Galerkin approach and parametrised with Gaussian functions. Finally, to control the robot swarm using this reduced model, a distributional controller was developed to control the trajectories of the robot swarm for open-loop and closed-loop control. The models and the control schemes were analysed using simulations for a swarm of robots to form a desired objective spatial distribution. The simulations show the potential for this novel modelling and control framework to be applicable to control the behaviour of a swarm of large number of robots.

Contents

List of Figures

List of Tables

List of Acronyms

BFGS	Broyden-Fletcher-Goldfarb-Shanno
DOC	Distributional optimal control
EM	Expectation-Maximisation
GRG	Generalised reduced gradient
KLD	Kullbach-Leibler divergence
MPC	Model predictive control
MRI	Magnetic Resonance Imaging
NLP	Nonlinear programming
OCP	Optimal control problem
PDE	Partial differential equation
PDF	Probability density function
SDSE	Swarm distributional states estimation

1 Introduction

Swarm robotics is the field of study where multiple robots are coordinated to collectively achieve a set objective, including aggregation, pattern formation, self-assembly, foraging, coordinated motion, and collective transportation. This collective behaviour is also known as self-organisation. By using a swarm of simple robots instead of a singular complex robot, the robot swarm has the advantages that it can tackle the set objective robustly, flexibly, and in a distributed manner [1]. Swarm robotics is a topic of intense research within the control and robotics communities because it has the potential to revolutionise many fields, including agriculture, military, surveillance, warehouse management, disaster relief, package delivery, and space debris cleanup [2].

1.1. Motivation

The motivation for the swarm application focused within the framework of this project is inspired by the use of microrobot swarms which are guided, controlled, and powered by magnetic forces for use in health applications such as targeted drug delivery, diagnostic imaging, and surgery. The advantages of using a swarm of magnetic microrobots, which each robot dimensions are within the millimetre and micrometer scale, is that it has the application potential for minimally invasive surgery and targeted treatment by accumulating microrobots to deliver the drug to the desired area of the body intravenously [3]. These microrobots use the magnetic field to be guided toward the targeted destination.

With the development and increasing popularity of using robots for assistance during surgical procedures, such as the da Vinci surgical robotic system [4], research has been conducted on using magnetic microrobots for use in minimally invasive procedures, such as targeted drug delivery and other health applications. Magnetic microrobots for use in medical applications use the magnetic resonance imaging (MRI) system for simultaneous actuation, tracking, powering, and controlling the microrobot swarm within the body. The first in vivo proof of concept of MRI-based actuation has been conducted in 2007 where a magnetic bead has navigated successfully along a preplanned path inside the carotid of a living swine [5]. Other applications for MRI-actuated microrobots include targeted drug delivery [6], magnetic particle imaging for guiding camera pills in the gastrointestinal tract [7], and localised treatment for cancer [8].

As the microrobots have a size restriction of the millimetre and micrometer size range to traverse within the human body, it is necessary to actuate, power, and control the robots indirectly and externally, such as by using an MRI system, because mechanically controlling and actuating the robots directly from outside the patient is impractical and dangerous. Other advantages of using magnetic microrobot swarms for surgery include requiring smaller incisions, resulting in faster recoveries, fewer complications for the patient, and shorter hospital stays than traditional surgeries [9].

Microrobots are characterised by the robots' relative simple capability as a consequence of their small size. Because of the microrobots' size restrictions, microrobots have the disadvantage of not having on-board actuation, power source, and controller. These limitations are necessary, however, for the microrobots to travel the internal natural pathways of the human body. Because microrobots may experience failures or be lost during operation, having a swarm of microrobots is necessary as swarms are generally characterised for being fault-tolerant and robust in large and sufficient numbers of robots, flexible and adaptable to handle a wide range of objectives, and scalable to be able to function at large swarm size.

So how can we represent a swarm of robots in a scalable manner, independent of the number of robots? This thesis seeks to address the development of a distributional controller for use with swarm microrobots [10, 11]. However, the scope and applicability of this project is wider than this particular application because the project's methods can equally be applied to a wide range of applications, including using an aerial swarm system for disaster relief, and using a swarm of robotic sailboats for oil spill cleanup [12], in which each robot may only have limited sensing and computational capabilities. This project addresses and models swarm robots that are completely manipulated by a centralised controller. Similarly, the environment's generated potential fields which are sensed by the robots can either be time-variant and dynamic or static.

1.2. Research Aims and Objectives

The overall aim of this research is to address some of the challenges of developing algorithms for functional deployment of large swarms of particle swarm robots. Specifically, it seeks to develop a distributional controller appropriate for a swarm system consisting of robots guided and informed by potential fields. The project labels these particle swarm robots which have limited capabilities as "passive" as they have no computational capabilities and are reliant on external magnetic fields to fulfil their objectives.

There are three broad questions addressed in this research. The first is on how to represent a large swarm of robots that can be controlled by an external field. The second is on how to develop a dynamic model that is amenable to analysis and control design. And finally, how can this model be utilised to control the swarm behaviour. These questions led to the following objectives:

- Development of an agent-based model of the swarm dynamics under field-oriented control. The key objectives of this part are:
  - Derive the equations of motion for the passive robot to be able to simulate its behaviour under a potential input field.
  - Develop a scalable spatial distribution representation for a large number of swarm robots.
  - Estimate the swarm distributional states given the robots' locations using a swarm distributional states estimation (SDSE) algorithm, a modified version of the expectation-maximisation (EM) algorithm.
- Develop a Galerkin-reduced advection-diffusion model through the finite dimensional parametrisation identified previously. The model should ensure that all constraints of the swarm dynamic characteristics are preserved.
- Given that the passive agents have limited computational or actuation capabilities, the

central controller is not able to direct each agent explicitly with control actions, but instead relies solely on potential functions for agent movement and manipulation. This requires the development of an distributional control framework to control the trajectories of the particle swarm for open-loop control and closed-loop control.

- The swarm representations, the reduced-order model and the central controller will be mathematical abstractions and derivations. In order to show that these result in potentially useful solutions, simulations are required to demonstrate the effectiveness of the methods.

1.3. Thesis Structure

The rest of the chapters is as follows:

- *Chapter 2* introduces the background of modelling approaches of swarm robots and the properties of modelling and controlling microrobot swarms, to which they have very limited computational and actuation capabilities.

- *Chapter 3* presents a model of the particle swarm behaviour in which its movements are influenced by a dynamic potential field. Afterwards, a description of the probability distribution to describe the swarm robots' behaviour under the external potential field is presented. Lastly, since the robot swarm is considered in a continuum model representation in which its probability density function is represented through weighted basis functions, a SDSE algorithm is introduced to derive the swarm distributional states, along with the choice of a symmetric Kullback-Leibler divergence (KLD) as a metric for the assessment on the quality of estimation.

- *Chapter 4* introduces the continuum representation of the advection-diffusion model that considers the swarm robots as a distribution. Since this representation is infinite dimensional, a reduced-order representation of this advection-diffusion model using the Galerkin method is presented. However, since this reduced-order model was not enough to satisfy the required constrains, a further modification was done to derive an error compensated reduced-order model.

- *Chapter 5* presents the development of a distributional control framework to control the trajectories of the particle swarm for open-loop and closed-loop control. This is based on the reduced-order representation of the advection-diffusion model and simulations were shown to analyse the performance under different bandwidths and system parameters.

- *Chapter 6* provides a summary of the results in this thesis and proposes possible routes for future work.

2

Background to Modelling Swarm Robot Systems

2.1. Background on Modelling Swarm Robot Systems

The field of swarm robotics is the study of using a multitude of robots capable of only simple capabilities to accomplish a variety of tasks, including aggregation, pattern formation, self-assembly, foraging, coordinated motion, and collective transportation. By using a swarm of simple robots instead of a singular complex robot, the swarm has the advantages that it can tackle the objective robustly, flexibly, and in a distributed manner. Swarm robotics is a topic of intense research within the control and robotics communities because it has the potential to revolutionize many fields, including agriculture, military, surveillance, warehouse management, disaster relief, package delivery, and space debris cleanup [2].

The essence of the field of swarm robotics is to investigate how to use a swarm of robots, in which each robot agent has limited computational capabilities, to fulfil a collective higher-level objective in which a single sophisticated robot would find it difficult to solve [1]. The advantages of using multiple computationally simple robots instead of a single sophisticated robot lie in the following swarm system properties:

- **Robustness**: The swarm system has the property of being robust amidst disturbances. A failure of a single agent in a swarm system is not critical due to the large number of agents in the system to still be able to collectively fulfil the objective.
- **Scalability**: The swarm system is capable of fulfilling the objective regardless of its group size [13].
- **Flexibility**: The swarm system is capable of being flexible in performing a variety of tasks through collaboration among the agents.

The swarm system is generally divided into two levels: the microscopic level and the macroscopic level. The microscopic level is the point of view of an individual robot, which is characterised by limited communication, incomplete knowledge of the environment and simple actions [1]. The macroscopic level is the point of view of the entire swarm where the features of the system can be seen and the collective objective of the swarm is defined. By the necessity to reduce the dimensionality of the swarm, the behaviour of the agents are simplified and abstracted. Thus, the challenge of swarm robotics systems is connecting the macroscopic level and the microscopic level. In practice, the microscopic level is where the controller is implemented to each agent but the performance of the swarm is calculated at the macroscopic level. The Eulerian approach to model the macroscopic and microscopic levels refer to describing the macroscopic level of the swarm robot system as a continuum and as probability distribution functions [14].

There are several approaches for modelling a swarm system depending on its application. One modelling approach is using the rate equation model to model state transitions for each agents and is as a result of only robot-to-robot interactions [15]. The rate equation approach is macroscopic, probabilistic, and non-spatial.

However, in order to model the swarm system in continuous-space, stochastic differential equations for the microscopic level and the partial differential equations for the macroscopic level are used. For example, the Langevin equation and the Fokker-Planck equation can be used to describe the microscopic level and the macroscopic level of the swarm system, respectively, in order to incorporate random and goal-oriented motion [16]. At the microscopic level, the Langevin equation, originally used to describe the motion of a particle in a fluid [17], can be used to describe the behaviour of an robot agent as a single Brownian agent:

$$\dot{\mathbf{R}} = -\mathbf{A}\left(\mathbf{R}(\mathbf{t}), t\right) + B\left(\mathbf{R}(t), t\right)\mathbf{F}(t), \tag{2.1}$$

where $\mathbf{F}$ are the random dislocation of forces upon the robot, $\mathbf{A}$ describes the deterministic motion based on the environment and B describes the random motion.

The macroscopic level, the Fokker-Planck equation can be used to describe the behaviour dynamics of the swarm as a whole, by modelling the time-evolution of the probability density function that describes the states of all robot particles in the environment:

$$\frac{\partial \rho(\mathbf{r},t)}{\partial t} = -\nabla\left(\mathbf{A}(\mathbf{r},t)\rho(\mathbf{r},t)\right) + \frac{1}{2}Q\nabla^2\left(B^2(\mathbf{r},t)\rho(\mathbf{r},t)\right), \tag{2.2}$$

where $\rho(\mathbf{r},t)$ is the probability of encountering another robot particle at position $\mathbf{r}$. The positions $\mathbf{R}$ used in the microscopic scale are for the positions of the individual robots, and the positions $\mathbf{r}$ are for locations in space irrespective to the individual robots' locations.

By coupling the Langevin equation and the Fokker-Planck equation to describe the microscopic and macroscopic levels of the robot swarm, respectively, we are able to capture and control the robot swarm's movements.

Many robot swarm modelling approaches are inspired by nature as well. The most well-known swarm algorithms include ant colony optimisation, particle swarm optimisation and the artificial bee colony algorithm [18]. The bacterial foraging optimisation algorithm is inspired by the group foraging behaviour of bacteria, such as E. coli, specifically for the chemotaxis behaviour which perceives the chemical gradients in the environment and move toward or away from specific signals [19]. This strategy would allow agents to stochastically and collectively swarm toward the optima.

2.2. Modelling and Control of Microrobot Systems

Microrobots in medical research settings have the potential to revolutionise applications such as drug delivery, minimally invasive surgery, and diagnostic surgery because these microrobots' small sizes allow for navigation in the natural pathways of a human body [20].

Magnetic microrobots have been used in conjunction with Magnetic Resonance Imagine (MRI) systems for actuation, control and power for applications and include research in treating age-related macular degeneration [21] and targeting tumours [22]. These microrobots can be considered as a robot swarm due to their limited computational capabilities given from the microrobots' size restrictions. They are only be able to perform simple actions but in which collectively, achieve a higher-level objective.

Microrobots are advantageous for use in health applications because their small size allows them to traverse the natural pathways of the human body, including the circulatory system, the urinary system, and the central nervous system [23]. Using microrobots instead of traditional surgery allow for minimal intervention and minimal trauma upon the human body during the operation.

Microrobots are generally restricted and characterised by their small size within the millimetre to micrometer scale. However, there are unique challenges and limitations of having microrobots at this size scale. For example, typically, the forces experienced by traditional centimetre-scaled robots are inertial forces, such as gravitational forces. However, at the micrometer scale, the microrobots are dominated by surface-related forces including surface tension, drag, viscous forces, and Brownian motion [24], [25]. Also, due to the microrobots' restrictive small size, the actuator, power source, and controller cannot be installed on-board. Instead, microrobots are actuated, powered, and controlled externally, typically using magnetic forces from an MRI system [23]. Furthermore, due to its size, the number of sensors installed in a microrobot is very limited and thus reliance on estimation techniques becomes greater.

In regards to influencing the behaviour of microrobots, magnetic actuation for robot swarm systems is achieved by translating magnetic field gradients into effectors with different concentrations [26]. Electromagnetic systems [9] and permanent magnetic systems [27] have already been shown to be able to move and control the swarm robots externally. Presently, the magnetic resonance imaging (MRI) system is commonly investigated for simultaneous magnetic actuation and magnetic resonance imaging to achieve robot tracking and control at the intravascular level [20], [28]. The use of MRI systems for navigating microrobots has many advantages such as enhanced tissue contrast and the lack of radiation during treatment. Furthermore, the MRI system is already in widespread use in clinical environments, making the technology accessible.

Microrobot navigation using an MRI system for actuation is achieved by inducing magnetic displacement forces from the three orthogonal slice selection and signal encoding gradient coils of a standard MRI system [5]. MRI systems provide instant feedback information to the controller allowing for real-time control and careful navigation along pre-planned paths in the blood vessels [29], [30].

The challenges for magnetic actuation systems and medical devices for clinical use include the generation of appropriate forces and torques for microrobots, accurate microrobot localisation, and minimisation of the magnetic actuation system footprint [20]. Control of a swarm of microrobots actuated by magnetic forces is presented in [31] and [32]. These experiments have focused on manipulating the external magnetic potential energy distribution so that collective motion and navigation can be achieved.

Applications of robots using MRI systems for actuation include directing magnetic macrophages into tumour sites in mice [6] and navigating a small ferromagnetic device along a preprogrammed path in a swine's carotid artery [33], [34]. MRI systems have also shown to be used for particle imaging for guiding catheters in a blood vessel and steering camera pills in the gastrointestinal tract [7]. The control of microrobots using MRI systems for endovascular navigation is presented in [35, 36], where the objective is to provide a feasible pathway within arterial networks.

2.3. Distributional Optimal Control Framework

When manipulating the movements a single robot, the traditional optimal control framework can be applied to control this single dynamical system in which its movements are governed by a small set of ordinary differential equations. However, when dealing with a large number of robots, it is necessary to be able to guide the robot swarm to desired locations and in desired distributions without assigning movement instructions to every single robot. In order to control large numbers of swarm robots, the swarm robots are directed to desired locations and distributions using probability density functions instead of assigning movement instructions to every single robot. This is accomplished using the distributional optimal control (DOC) method, which allows the control of robots' movements by using and manipulating probability density functions. The advantages of this method is that it does not need to decouple the microscopic agents' dynamics with the macroscopic behaviour of the swarm.

The distributional optimal control approach is commonly used for path planning purposes for a swarm of passive agents in an active environment and aims to optimize the macroscopic probability density function of the swarm and the microscopic control law for each agent, as described in the tutorial paper [10]. The advantage of this approach is that it does not need to decouple the microscopic agents' dynamics with the macroscopic behaviour of the swarm, and the probability density functions do not need to be pre-specified. In works presented in [37], [38], the advection-diffusion equation is used to describe the change of the probability density function of the swarm over time.

The distributional optimal control method is distributional because the agents' control actions are determined through a central supervisory controller. This is presented in the context where the robot swarm is represented as a probability distribution and whose collective swarm movement evolution is described by partial differential equations, while the individual robots' movements are described by a set of ordinary differential equations. This leads to the question how the design choices, such as the number of basis functions and the potential field bandwidths achieve the desired level of control of the swarm's behaviour within the distributional control framework, and whether a low bandwidth can be balanced with active control to improved the accuracy of distributional control.

2.3.1 Distributional Optimal Control Problem

Consider a continuous-time system of N agents, in which each of the agent's dynamics with the following stochastic differential equation:

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}[\mathbf{x}_i(t), \mathbf{u}_i(t), t] + \mathbf{G}[\mathbf{x}_i(t), t]\mathbf{w}_i(t), \quad \mathbf{x}_i(T_0) = \mathbf{x}_{i0}, \tag{2.3}$$

where i is an individual agent, $\mathbf{x}_{i0}$ is the initial state, and $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^n$ and $\mathbf{u}_i \in \mathcal{U} \subset \mathbb{R}^m$ are the microscopic state and control of the agent, respectively, and $\mathcal{X}$ and $\mathcal{U}$ are the microscopic state space and the admissible control inputs of the system, respectively. The agent dynamics $\mathbf{f}$ are described with a small set of ordinary differential equations and the disturbances $\mathbf{w}_i$ and $\mathbf{G}$ represent the Gaussian disturbance vector and constant matrix, respectively. All the agents of the swarm operate in state space $\mathcal{X}$ in time periods $(T_0, T_f]$.

The probability density function of the swarm p, which is also referred to as the restriction operator in [10], maps the microscopic states to the macroscopic-level behaviour and can be described as

$$p(\mathbf{x}_i, t) = g(\mathbf{x}_i, \mathbf{X}), \tag{2.4}$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$ is the matrix of the states of all the agents, $\mathbf{x}$ is a two-dimensional state vector which represents the position of the agent, and $g : \mathcal{X} \times \mathcal{X}^N \to \mathbb{R}$, where $\mathcal{X}^N$ represents a vector of the agents' positions. p is a non-negative scalar which also satisfies the normalization condition

$$\int_{\mathcal{X}} p(\mathbf{x}_i, t) d\mathbf{x}_i = 1. \tag{2.5}$$

The integral cost function used for the distributional optimal control approach is

$$J = \phi[p(\mathbf{x}_i, T_f)] + \int_{T_0}^{T_f} \int_{\mathcal{X}} \mathcal{L}[p(\mathbf{x}_i, t), \mathbf{u}_i, t] d\mathbf{x}_i dt, \tag{2.6}$$

where ϕ is the terminal cost, $\mathcal{L}$ represents the Lagrangian, and p is the probability density function of the agents. Using this integral cost function, the DOC approach calculates the macroscopic optimal agent distribution p^* and the optimal microscopic control law for each agent u_i^* which minimize the cost function J over a time interval $(T_0, T_f]$. This integral cost function is subject to the agents' dynamics, the normalization property (2.5), and the state constraints.

The purpose of the advection-diffusion equation is to describe the time rate change of the probability density function p, assuming that $\mathbf{x}_i(t) \in \mathcal{X}$ for all agents and in $(T_0, T_f]$. The advection-diffusion equation is

$$\begin{aligned}\frac{\partial p(\mathbf{x}_i, t)}{\partial t} &= -\nabla \cdot \{p[\mathbf{x}_i(t), t]\mathbf{v}_i(t)\} + \nabla \cdot \left\{(\mathbf{G}\mathbf{G}^T)\nabla p(\mathbf{x}_i, t)\right\} \\ &= -\nabla \cdot \{p[\mathbf{x}_i(t), t]\mathbf{f}[\mathbf{x}_i(t), \mathbf{u}_i(t), t]\} + (\mathbf{G}\mathbf{G}^T)\nabla^2 p(\mathbf{x}_i, t),\end{aligned} \tag{2.7}$$

where $\mathbf{v}_i(t) = \mathbf{f}[\mathbf{x}_i(t), \mathbf{u}_i(t), t]$ is the velocity field, ∇ represents the gradient with respect to elements of the state vector $\mathbf{x}$, and $(\cdot)$ is the dot product. From this equation, it can be seen that the advection-diffusion equation consists of the negative divergence of the advection vector $(p\mathbf{v})$ and the positive divergence of the diffusion vector $(\mathbf{G}\mathbf{G}^T\nabla p)$.

The initial condition of the optimal control problem, which for the DOC approach, is the initial agent distribution at the macroscopic level, is

$$p[\mathbf{x}_i(T_0), T_0] = p_0(\mathbf{x}_i), \tag{2.8}$$

where p_0 is the initial probability density function.

2.3.2 Numerical Solutions of the Distributional Optimal Control Problem

The DOC method calculates the optimal agent distribution p^* and the microscopic control law for each agent $\mathbf{u}_i^*$, which minimize the macroscopic cost function (2.6) subject to the advection-diffusion equation (2.7), the normalization condition (2.5), and the initial (2.8). To solve the for the optimal agent distribution and the microscopic control law for each agent two numerical methods are briefly presented: the indirect optimization method and the direct optimization method.

The direct optimisation method as shown in [11], [39], solves for the optimal agent distribution p^* and the agents' control law $\mathbf{u}_i^*$, by first assuming that the p^* can be approximated with a n-dimensional multivariate Gaussian mixture model parametrised over the state space domain $\mathcal{X}$. At any point during the system's run time $t \in (T_0, T_f]$, the optimal agent distribution can be estimated as

$$p(\mathbf{x}_i, t) = \sum_{j=1}^{z} w_j(t) f_j(\mathbf{x}_i, t), \tag{2.9}$$

where the weights are $0 \leq w_j \leq 1$ for all j and $\Sigma_{j=1}^{z} w_j = 1$, and

$$f_j(\mathbf{x}_i, t) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}_j|^{1/2}} e^{[-(1/2)(\mathbf{x}_i-\mu_j)^T \mathbf{\Sigma}_j^{-1}(\mathbf{x}_i-\mu_j)]}, \tag{2.10}$$

where $\mu_j \in \mathbb{R}$ is the time-varying mean vector, $\mathbf{\Sigma}_j \in \mathbb{R}^{n \times n}$ is the time-varying covariance matrix.

Indirect optimisation methods are applied to solve the DOC method iteratively to determine the optimal agent distribution p^*, in which then the agent control law $\mathbf{u}_i^* = \mathbf{c}[p^*(t), t]$ is calculated. One indirect method used to solve the nonlinear DOC optimality conditions [37] is the generalized reduced gradient (GRG) method [10], [38]. An advantage of using the GRG method is that the advection-diffusion equation (2.7), which describes the swarm's probability density function over time, is represented only as a function of $\mathbf{u}$. The partial differential equations can be solved by using the Galerkin method or a modified Galerkin method, known as the constraint integration method [40].

During every iteration of the GRG algorithm, the advection-diffusion equation is solved at every time step to obtain the approximation of p^*. Then, by holding the approximations of p^* in that approximation, $\mathbf{u}^*$ is updated to minimize the integral cost function (2.6). This process repeats until the gradient norm is lower than a predefined tolerance or after several iterations, any update to $\mathbf{u}^*$ results in an increase in the cost function J.

The distributional optimal control approach is the mathematical formulation of generating dynamic potential fields without decoupling the dynamics of the passive agents. The advantages of the DOC approach is that it optimizes the performance of the system measured by the cost function in the macroscopic level without the need to uncouple to the dynamics of the agents in the microscopic level. The computational complexity of the problem is reduced by using probability density functions and the advection-diffusion equation as the evolution equation.

2.4. Modelling of Swarm Robot Systems Using Partial Differential Equations

As the number of robots increase in the swarm, the computational time needed to analyse individual robot trajectories and designing a control system which accounts for every robot increases dramatically with their number, leading to long computational times and scalability issues. However, in order to eliminate this scalability challenge, a solution may be to represent the robot swarm in a continuum-representation in order to model and control the swarm as a probability density distribution. In order to guide the robot swarm as a distribution, a distributional optimal control approach can be used to achieve the desired robot swarm distribution. The behaviour of the swarm can be represented in partial-differential equation form, such as the advection-diffusion model [41], [42].

Additionally, applications with dynamics in the form of partial differential equations (PDEs) has been used in many industrial fields such as in thermal regulation [43], control of chemical process systems [44] and control of power converters [45].

However, developing a controller for an infinite-dimensional system is not trivial [46] as there are only a few special cases where analytical computable solutions can be derived [47]. Hence, in order to control the robot swarm behaviour, first it would be necessary to be able to represent this behaviour in a finite-dimensional space. As the probability density function is infinite-dimensional due to the scalar fields being functions of the spatial domain, this function must first be parametrised.

One way to parametrise the infinite dimensional approximate system representation to be reduced into finite dimensional space is by considering the projection of the system onto a set of finite number of basis functions, as in the Galerkin-type decomposition [48]. This Galerkin decomposition takes the infinite dimensional function and represents it into a sum of a finite number of basis functions, where the idea is if there are enough basis functions used, then the result will get as close to an approximation possible to the infinite-dimensional function result. The Galerkin approach has been used in many applications for approximating the infinite-dimensional PDEs, such as thermal shallow water equations [49] and electromagnetic current fields [50].

2.5. Summary

This chapter has presented a background on the properties and features of swarm robot systems and the general modelling approach of being represented at two levels: the microscopic level (as can be represented with the Langevin equation) and the macroscopic level (as can be represented with the Fokker-Planck equation). The microrobot swarms' properties, limitations and applications was presented along with the need for the microrobot swarm of an external potential field generated from an external field, such as the the MRI system, to be able to be controlled and guided to reach its desired locations and distributions. This swarm model representation and control can be achieved using the distributional optimal control framework and the general DOC approach. Lastly, we presented the need of the evolution equation to be parametrised into a finite-dimensional representation using the Galerkin approximation.

3

Agent-Based Model and Representation for Large Scale Swarms

As described in the previous chapter, we take our inspiration from microrobots in which due to its tiny size, there are very limited computational and actuation capabilities. To develop a model of the particle swarm behaviour in which its movements are influenced by a dynamic potential field, we first present a simple particle-based representation of the swarm robot. Then a description of the probability distribution used to describe the swarm of robots is defined. The aim of this chapter is to introduce the behaviour of agent-based particles model under the influence of a potential field and its representation for large scale swarms.

To describe and control robots with limited computation or actuation capabilities, we define these robots as passive agents as they are not able to move without external influences from a controller. Likewise, as we would be dealing with a large scale swarm, we assume that the central controller controlling the movements of these passive agents are not able to direct each agent explicitly with control actions. Instead, the central controller directs the robot swarm through an external dynamic input field.

3.1. Agent-Based Model of a Passive Robot

The agent model of a passive robot used here is of a simple particle based representation. The particles experience an external force that drives their dynamic behaviour under the laws of particle motion. This is illustrated in Figure 3.1 where the particles move up the gradient to the peak centre.

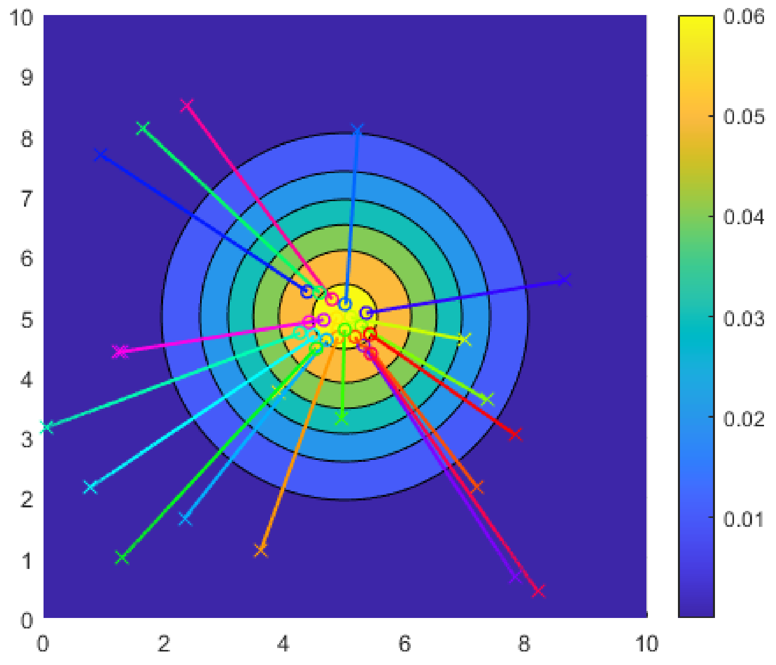


Figure 3.1.: The trajectories of passive robots moving up the gradient to the peak at the centre of the arena. The x markers denote the starting position of the robot and the o markers denote the end position of the robot.

Let the position $\mathbf{z} = [z_x, z_y]^\top \in \mathbb{R}^2$ of the agent at time t be denoted by $\mathbf{z}(t)$ and its velocity by $\dot{\mathbf{z}}(t)$. Let the forces per unit mass being applied to the particle at time t be denoted by $\mathbf{f}(t)$. Then, the dynamics of the agent in a two-dimensional space can be described by the equation,

$$\begin{pmatrix} \dot{\mathbf{z}}(t) \\ \ddot{\mathbf{z}}(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{z}(t) \\ \dot{\mathbf{z}}(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{f}(t) \tag{3.1}$$

The mathematical relation in 3.1 is deterministic so that the agent behaves in an ideal fashion. If imperfections and non-ideal representations have to be accounted for, then a stochastic disturbance term must be added. This results in,

$$\begin{pmatrix} \dot{\mathbf{z}}(t) \\ \ddot{\mathbf{z}}(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{z}(t) \\ \dot{\mathbf{z}}(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{f}(t) + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{w}(t) \tag{3.2}$$

where $\mathbf{w}(t)$ represents the random variations in the robot acceleration. The disturbance terms are typically assumed to be independent and identically distributed zero mean Gaussian distributed random quantities.

3.2. Representation of Field-driven Forces

For passive robots or agents, the forces they experience are externally driven, often through what is referred to in this thesis as *control fields*. The forces experienced by the passive robot for this control field can be modelled in a variety of ways. One popular approach to guide passive robots is the potential field method [51], a method used for moving agents to a desired location while avoiding obstacles.

The potential field $F(\mathbf{s}, t)$ is defined as a scalar field over the spatial domain $\mathbf{s} = [s_x, s_y]^\top \in \mathcal{S} \subseteq \mathbb{R}^2$ in which its negative gradient is a vector field of forces

$$\mathbf{f}(\mathbf{s}, t) = -\nabla F(\mathbf{s}, t) = \begin{bmatrix} -\frac{\partial F(\mathbf{s},t)}{\partial s_x} \\ -\frac{\partial F(\mathbf{s},t)}{\partial s_y} \end{bmatrix}. \tag{3.3}$$

Mathematical representations of the control scalar fields would be functions of the spatial domain and will therefore be infinite dimensional. For any practical scenario, they must be suitably parametrised in terms of a finite number of controllable parameters. This can be achieved via a basis function representation.

The potential field applied as the control signal to manipulate the swarm of robots is parametrised as,

$$F(\mathbf{s}, t) = \Psi(\mathbf{s})^\top \mathbf{u}(t), \tag{3.4}$$

where $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ is a vector of basis function coefficients and $\Psi(\mathbf{s}) = [\psi_1(\mathbf{s}) \ \psi_2(\mathbf{s}) \ \cdots \ \psi_{N_u}(\mathbf{s})]^\top$ is a vector of basis functions.

In this instance, the control signal is limited to the subspace induced by the bases $\Psi(\mathbf{s})$. The parametrisation will therefore have an impact on the ability to control the swarms. Again, without loss of generality, the basis functions can be assumed to satisfy the property:

$$\int_{\mathcal{S}} \psi_i(\mathbf{s})d\mathbf{s} = 1, \quad \forall\, i \qquad \Rightarrow \qquad \int_{\mathcal{S}} \Psi(\mathbf{s})d\mathbf{s} = \mathbf{1} \tag{3.5}$$

Any assumptions to be made regarding the coefficients $\mathbf{u}(t)$ will depend on the type of field control that can be exerted on the swarm of robots.

The basis functions chosen are the Gaussian functions for their following properties:

- Integration of Gaussian functions multiplied by polynomials has a closed-form solution akin to the statistical properties of Gaussian distributions.
- Multiplication of Gaussian functions results in Gaussian functions.
- Derivatives of Gaussian functions results in multiplication of Gaussians and polynomials.

Its functional form is given by,

$$\psi_j(\mu_j, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(\mathbf{s}-\mu_j)^2}{\sigma^2}\right) \tag{3.6}$$

where the basis function parameters are μ_j which is the location parameter and σ which is the scalar scale parameter.

The choice of the basis function parameters can be obtained from an analysis of the bandwidth of the function being approximated [52]. Typically, the basis functions are arranged such that the basis function location parameters are placed on a regular grid. The scale parameter of the Gaussian basis functions are calculated by

$$\nu = \frac{1}{\pi\sigma}\sqrt{\frac{\ln 2}{2}} \tag{3.7}$$

$$\Delta \leq \frac{1}{2\rho\nu} \tag{3.8}$$

where Δ is the regular grid distance of the basis function locations and ρ is the oversampling parameter.

Combining these two equations, we can calculate the scale parameter,

$$\sigma = \frac{2\sqrt{0.5\ln(2)}\rho\Delta\kappa}{\pi} \tag{3.9}$$

where $\kappa \geq 1$ is a factor that turns the inequality in equation (3.8) into an equality condition.

An example is shown in Figure 3.2 where there is a potential field with four peaks at each corner and a grid of passive robots arrayed evenly in a grid throughout the arena that have not reacted to the potential field. Over time the particles move towards the nearest peaks respectively as shown in Figure 3.3.

The passive robots have dimensions and so care is needed to ensure that the robots do not overlap each other in any simulation. This is achieved by when the adjacent robots collide with each others' boundaries, the velocity becomes zero and the robots do not overlap. As can be seen in Figure 3.4 there is no overlap among the passive robots. When a particular time invariant $F(\mathbf{s}, t)$ is applied with the choice of $\mathbf{u}(t)$, these are the kind of trajectories the passive robots will follow.

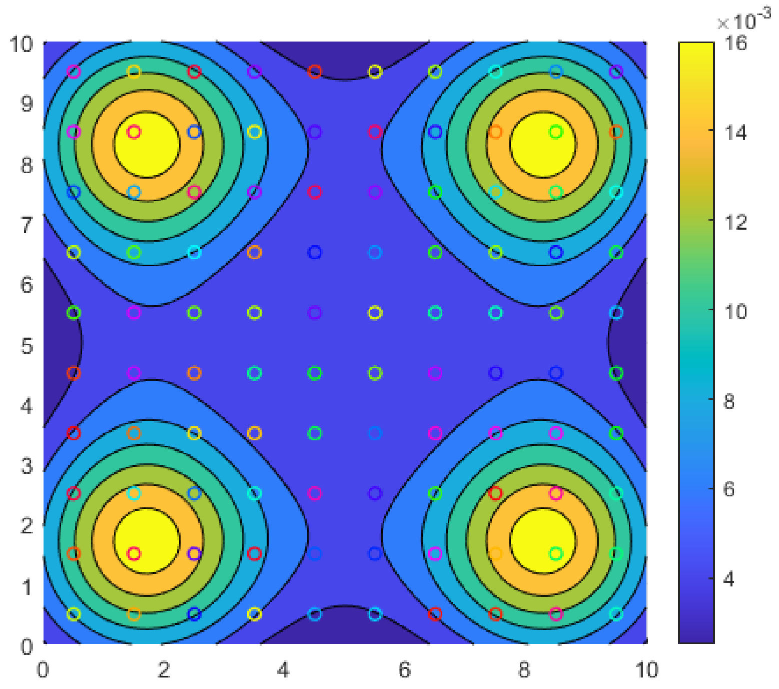


Figure 3.2.: Potential field with initial positions of particles.

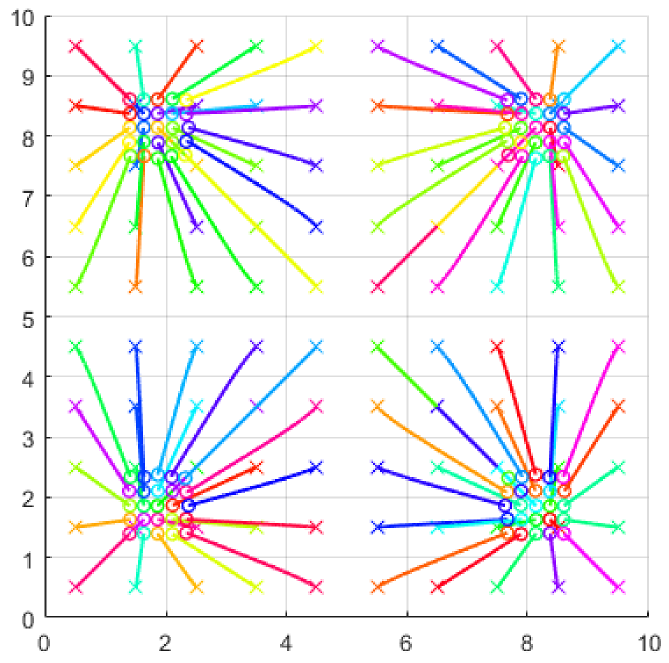


Figure 3.3.: Trajectory of particles. The 'x' markers denote the starting position of the robot and the 'o' markers denote the end position of the robot.

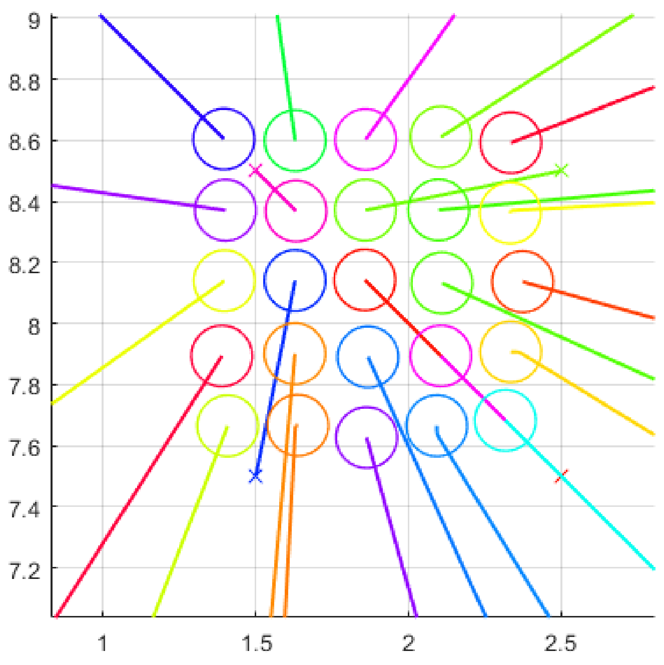


Figure 3.4.: Adjacent particles do not overlap each other.

3.3. Representation for Large Numbers of Swarm Robots

The probability density function $p(\mathbf{s}, t)$ is infinite dimensional as it is a function defined on $\mathcal{S}$. To obtain a finite dimensional representation, this function must be parametrised. This is achieved by using a Galerkin-type decomposition.

The probability density function (PDF) is represented by a finite dimensional decomposition approximation given by,

$$p(\mathbf{s}, t) = \Phi(\mathbf{s})^\top \mathbf{x}(t) \tag{3.10}$$

where $\mathbf{x}(t) \in \mathbb{R}^{N_x}$ is a vector of basis function coefficients which represents the swarm distributional states and $\Phi(\mathbf{s}) = [\phi_1(\mathbf{s}) \;\; \phi_2(\mathbf{s}) \;\; \cdots \;\; \phi_{N_x}(\mathbf{s})]^\top$ is a vector of basis functions defined over the space $\mathcal{S}$.

Parametric representations of probability distributions are typically based on standard classes of distributions such as multivariate Gaussian or Normal distributions and their mixture distribution extensions. While the decomposition in (3.10) can fit into a mixture-type representation, the basis functions themselves are time independent and so do not fit into the parametric probability distributions that would need to be time varying to represent a dynamic swarm of robots. In fact, the above decomposition would represent a semi-parametric probability distribution representation, as the mean and variance do not change.

The choice of this semi-parametric distribution representation requires that the following condition be additionally imposed on this approximation,

$$\int_{\mathcal{S}} p(\mathbf{s}, t) d\mathbf{s} = \int_{\mathcal{S}} \Phi(\mathbf{s})^\top \mathbf{x}(t) d\mathbf{s} = \left[\int_{\mathcal{S}} \Phi(\mathbf{s}) d\mathbf{s}\right]^\top \mathbf{x}(t) = 1 \tag{3.11}$$

Without loss of generality, the basis function can also be normalised as:

$$\int_{\mathcal{S}} \phi_i(\mathbf{s}) d\mathbf{s} = 1, \qquad \forall\, i \Rightarrow \qquad \int_{\mathcal{S}} \Phi(\mathbf{s}) d\mathbf{s} = \mathbf{1} \tag{3.12}$$

where $\mathbf{1} = [1 \;\; 1 \;\; \cdots \;\; 1]^\top$ is a vector of 1s.

With the normalised basis functions, the constraint (3.11) reduces to

$$\mathbf{1}^\top \mathbf{x}(t) = 1 \tag{3.13}$$

This equality constraint must be respected in any reduced advection-diffusion model representation derived from this basis function based parametrisation.

The choice for the class of basis functions, the number of parameters for representing the swarm distribution are important design choices. Gaussians functions are common, typically used as functions as Parzen density estimators, kernel density estimators, and Gaussian mixtures, and given by,

$$\phi_i(\mu_i, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(\mathbf{s}-\mu_i)^2}{\sigma^2}\right) \tag{3.14}$$

where the basis function parameters are μ_i which is the location parameter and σ which is the scale parameter.

3.4. Estimation of Swarm Distributional States

With the swarm of robots as particles being approximated as a continuum and as probability distribution functions, it is important to be able to determine the probability distribution of a swarm based on observed particle locations. Given that the probability distributions are represented using sums of weighted basis functions and that the basis function parameters are kept unchanged, the expectation-maximisation (EM) algorithm for estimating the coefficients of the basis functions has a simplified structure. This swarm distributional states estimation (SDSE) algorithm can be seen as a modified version of the EM algorithm for Gaussian mixtures.

3.4.1 Swarm Distributional States Estimation

The pseudo-mixture Gaussian distribution representation permits its estimation from data to be derived as a simpler version of the mixture Gaussian distribution estimation of the EM algorithm, known as the SDSE algorithm. The goal of this algorithm is to maximise the likelihood function with respect to the parameters of any model [53]. For the choice of the probability density function (PDF) of the swarms with *a priori* chosen basis functions, only the coefficients are the model parameters that need to be estimated from the observed data. This can be represented mathematically as,

$$\pi^* = \arg\max_{\pi} \left\{ \prod_n p(\mathbf{z}_n|\pi;\mu,\sigma) \right\} \tag{3.15}$$

where π represents the basis function weights, and the mixture Gaussian distribution

$$p(\mathbf{z}_n|\pi;\mu,\sigma) = \sum_i \pi_i \mathcal{N}(\mathbf{z}_n|\mu_i,\sigma_i) \tag{3.16}$$

is under the assumption that the observations $\mathbf{z}_n$ are identically and independently distributed with a mixture distribution. However, in our case, we will assume a pseudo-mixture representation by the use of Gaussian basis functions with pre-assigned mean and variance parameters leaving only the mixture coefficients to be estimated from the data as shown in the equation above.

With the choice of the basis functions being Gaussians, the probability density function resembles a mixture of Gaussians. The use of the popular EM algorithm for estimating a mixture of Gaussian distribution is therefore an appropriate choice here to use as the foundation. However, the location parameter of the basis functions resembling the means and the scale parameter resembling the standard deviation are fixed and need not be estimated. Hence, the following SDSE algorithm ignores these steps as shown in Algorithm 3.1.

Algorithm 3.1 *Swam distributional states estimation (SDSE) algorithm for the estimation of basis function weights that are also referred to as swarm distributional states.*

1. Initialise the basis function means (location parameter) μ_i, their covariance (scale parameter) σ and the basis function weights π_i.

2. **E step**: Calculate for the responsibilities γ_i for each of the ith basis function for each nth robot position $\mathbf{z}_n$,

$$\gamma_i(\mathbf{z}_n) = \frac{\pi_i \phi_i(\mathbf{z}_n|\mu_i, \sigma)}{\sum_{j=1}^{N_x} \pi_j \phi_j(\mathbf{z}_n|\mu_j, \sigma)} \tag{3.17}$$

3. **M step**: Recalculate the mixing coefficients π_i using the current responsibilities γ_i

$$\pi_i^{\text{new}} = \frac{N_i}{N} \tag{3.18}$$

   where

$$N_i = \sum_{n=1}^{N} \gamma_i(\mathbf{z}_n) \tag{3.19}$$

4. Check for convergence of the responsibilities γ and return to step 2 with π_i^{new} if convergence is not achieved. If converged,

$$x_i = \pi_i^{new} \tag{3.20}$$

A probability density function estimate using the SDSE algorithm as shown in Algorithm 3.1 and particle swarm locations at a particular time is presented in Figure 3.5. This example is

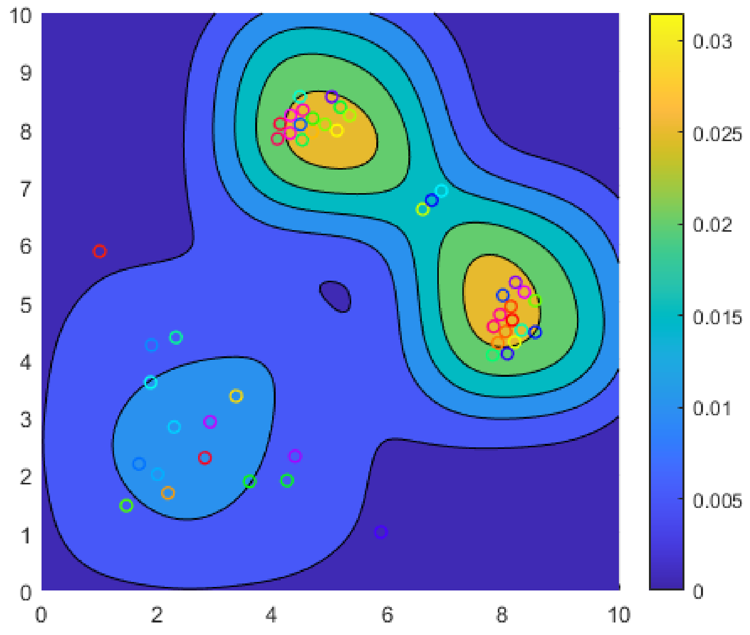


Figure 3.5.: Generated PDF with particles, in which their basis function locations are known.

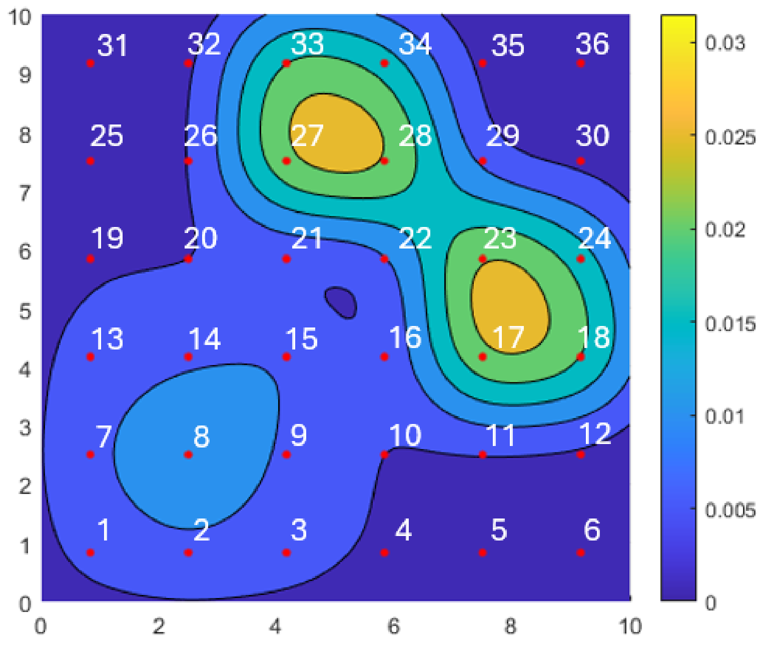


Figure 3.6.: Generated PDF showing the index of the basis functions.

chosen to illustrate the mismatch between the expected probability distribution of the particles and the SDSE estimated probability distribution of the swarm.

Using six by six grid of Gaussian basis functions with $\rho = 1.5$, the index of the Gaussians is shown in Figure 3.6. Figures 3.7 and 3.8 show the the SDSE algorithm with $\kappa = 1.0$ based on the particle locations shown in Figure 3.5. The swarm state estimation does not fully match the PDF model simulation. In peaks 16 to 18 and 26 to 29, for example, the model simulation shows two state value peaks but the state estimation shows only one peak at a large state value. This suggests that the SDSE algorithm does not capture the PDF of the particles accurately.

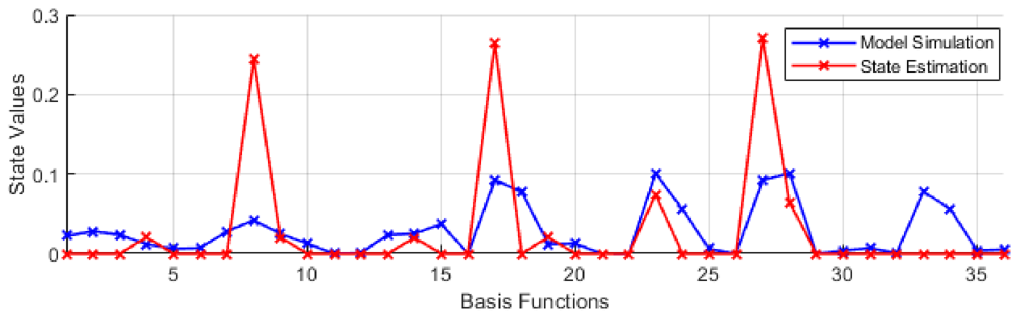


Figure 3.7.: Applied SDSE algorithm with $\kappa = 1.0$.

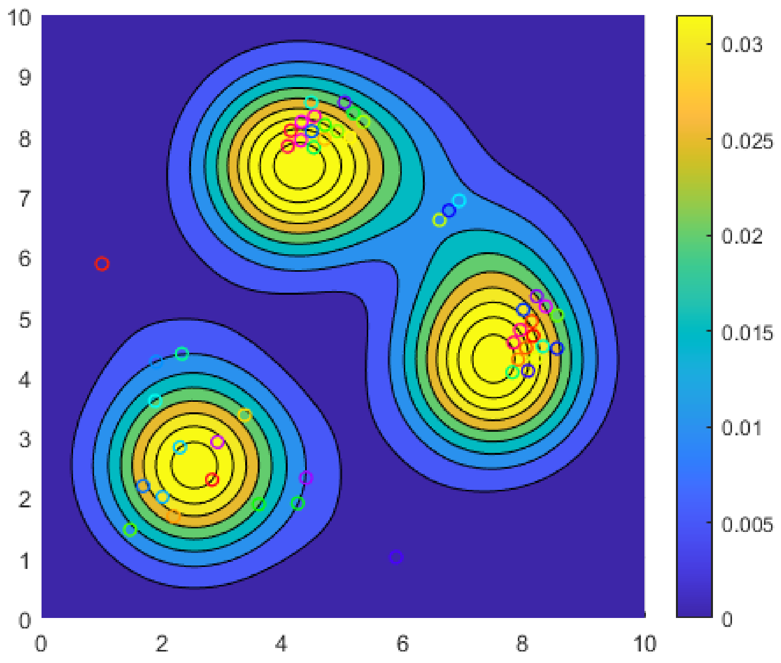


Figure 3.8.: Generated PDF with $\kappa = 1.0$.

κ values of 1.5 and 0.5 (even if this value does not obey equation (3.9)) are tested to see if there is an improvement. We can see in Figure 3.9 that the peaks do not fully match the model simulation even with lower or higher κ values.

The peaks appear to result from over concentration of the estimated weights of the basis function, reinforced through the iterations. What would be preferable is for the estimated weights to reflect the local density of the particles. To achieve this, an assumption can be

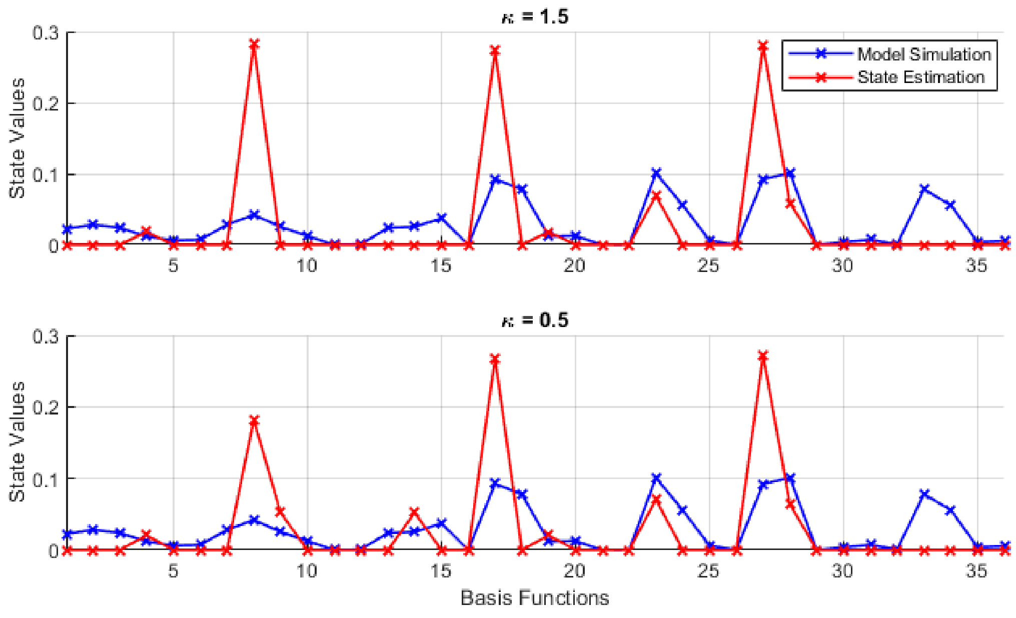


Figure 3.9.: Applied SDSE algorithm with $\kappa = 1.5$ and $\kappa = 0.5$, respectively.

made that the prior is a uniform distribution and that the posterior is therefore obtained in a single step, avoiding the SDSE iteration.

3.4.2 Modified Swarm Distributional State Estimation

From the analysis in the previous section, it is clear that the iteration appears to concentrate the estimate around fewer Gaussian functions. This over-fitting is likely to emerge from the estimation problem being addressed as a maximum likelihood estimation problem.

An alternative interpretation of the state estimation algorithm can be derived at if the estimate was made on the basis of the expected number of robots within the region of each basis function. This is equivalent to assuming that the prior used to derive the mixture coefficients are assumed equal. The result is an algorithm that only takes a single step instead of it being an iterated one. This algorithm is referred to as the modified SDSE algorithm as shown in Algorithm 3.2.

Algorithm 3.2 *Modified swarm distributional states estimation algorithm for pseudo-Gaussian mixtures*

1. Initialise the basis function means (location parameter) μ_i, their covariance (scale parameter) σ.

2. **E step**: Calculate for the responsibilities γ_i for each of the ith basis function for each nth robot position $\mathbf{z}_n$,

$$\gamma_i(\mathbf{z}_n) = \frac{\phi_i(\mathbf{z}_n|\mu_i, \sigma)}{\sum_{j=1}^{N_x} \phi_j(\mathbf{z}_n|\mu_j, \sigma)} \tag{3.21}$$

3. **M step**: Directly calculate the state vector $\mathbf{x}$ using the current responsibilities γ_i

$$x_i = \frac{N_i}{N} \tag{3.22}$$

where

$$N_i = \sum_{n=1}^{N} \gamma_i(\mathbf{z}_n) \tag{3.23}$$

We can see in Figures 3.10 and 3.11 that the model simulation and state estimation estimates are a closer match.

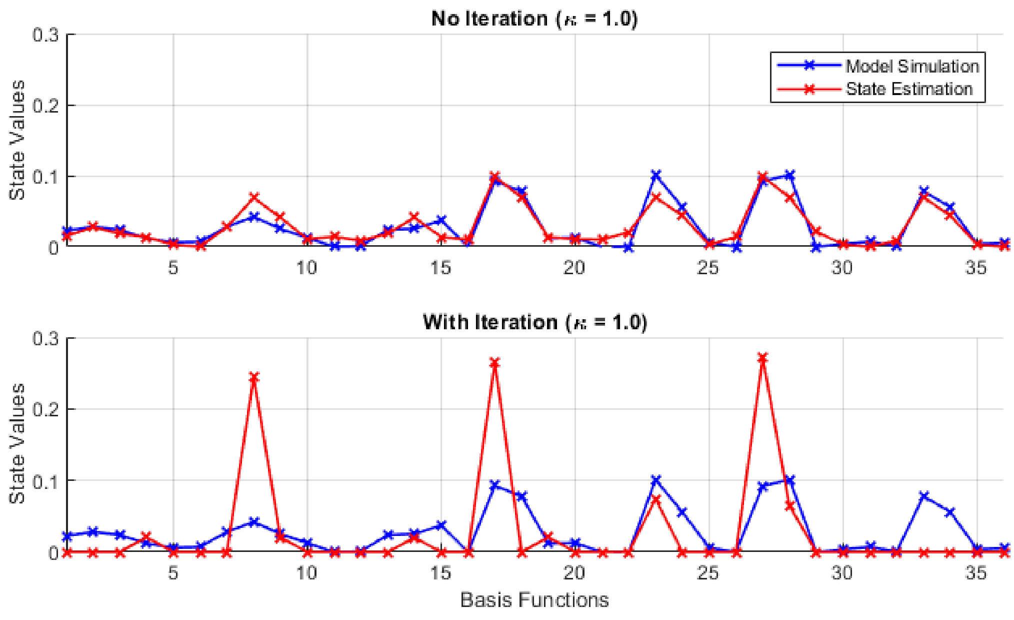


Figure 3.10.: Applied SDSE algorithm with no iterations and SDSE algorithm with iteration, respectively at $\kappa = 1.0$.

3.4.3 Symmetric Kullback-Leibler Divergence

In Section 3.4.2, we assessed the discrepancy between the estimated probability density and the empirical particle distributions to assess which of the two algorithms gave a more accurate

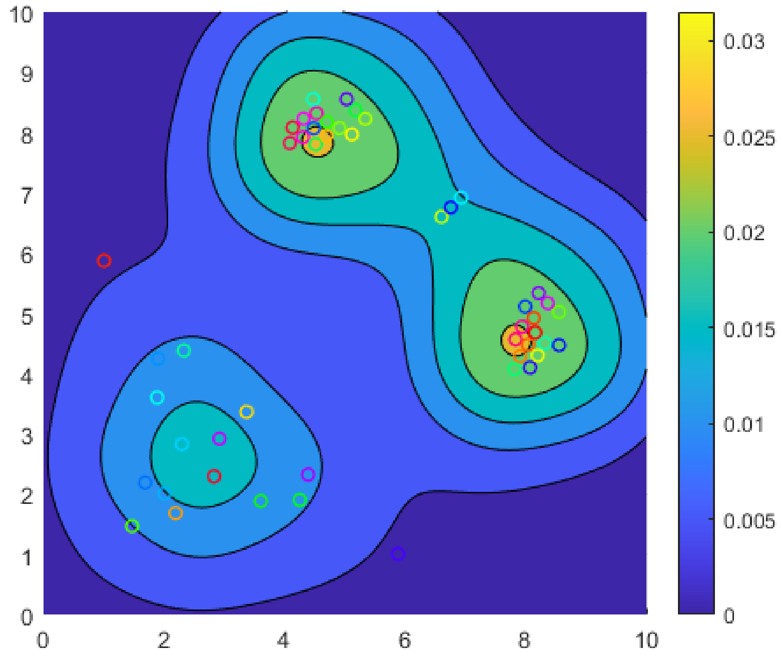


Figure 3.11.: Generated PDF with no iteration with $\kappa = 1.0$.

representation of the particle distribution. However, the assessments were based visually via plots, as these were sufficiently revealing. Preferably, these errors are quantified so that objective assessments can be made. If the estimated probability density function $\hat{p}$ and the empirical probability density function p^* differ, the symmetric Kullback-Leibler divergence (KLD) can be used as a metric of this difference [54],

$$\mathcal{J}(\hat{p}, p^*) = \frac{\mathcal{D}(\hat{p} \parallel p^*) + \mathcal{D}(p^* \parallel \hat{p})}{2} \tag{3.24}$$

where

$$D(p_0 \parallel p_1) = \int_{\mathcal{S}} p_0(\mathbf{s}, t) \log_2 \frac{p_0(\mathbf{s}, t)}{p_1(\mathbf{s}, t)} d\mathbf{s}, \tag{3.25}$$

The symmetric KLD is zero when $\hat{p}$ and p^* are equal.

For the example in the previous subsection, the symmetric KLD values between the empirical probability density function p^*, as shown in Figure 3.5, and the estimated probability density functions $\hat{p}$, as shown in Figures 3.7, 3.8, 3.9 3.10 and 3.11, are presented in Table 3.1:

Table 3.1.: KLD values from the modified SDSE algorithm example.

Simulation	KLD Value
κ = 1.0 (no iteration)	182.247
κ = 1.0 (with iteration)	383.998
κ = 1.5 (with iteration)	405.195
κ = 0.5 (with iteration)	362.466

In order to additionally show that the no iterations SDSE algorithm yields the most accurate representation of the particle distribution, Figures 3.12 and 3.13 are other examples with a generated PDFs and particle locations, in which Table 3.2 shows the symmetric KLD values.

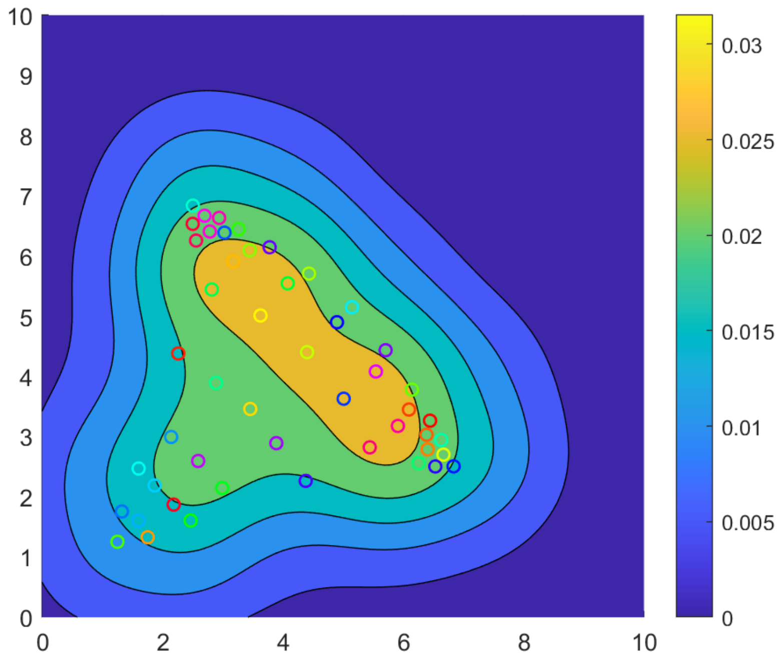


Figure 3.12.: Example of generated PDF showing the particle locations and in which their basis function locations are known.

Table 3.2.: KLD values from the modified EM state estimation example in which the second and third columns correspond to KLD values of the examples from Figure 3.12 and 3.13, respectively.

Simulation	KLD Value	
κ = 1.0 (no iteration)	738.783	66.185
κ = 1.0 (with iteration)	1280.877	401.015
κ = 1.5 (with iteration)	1265.242	399.012
κ = 0.5 (with iteration)	1300.935	277.798

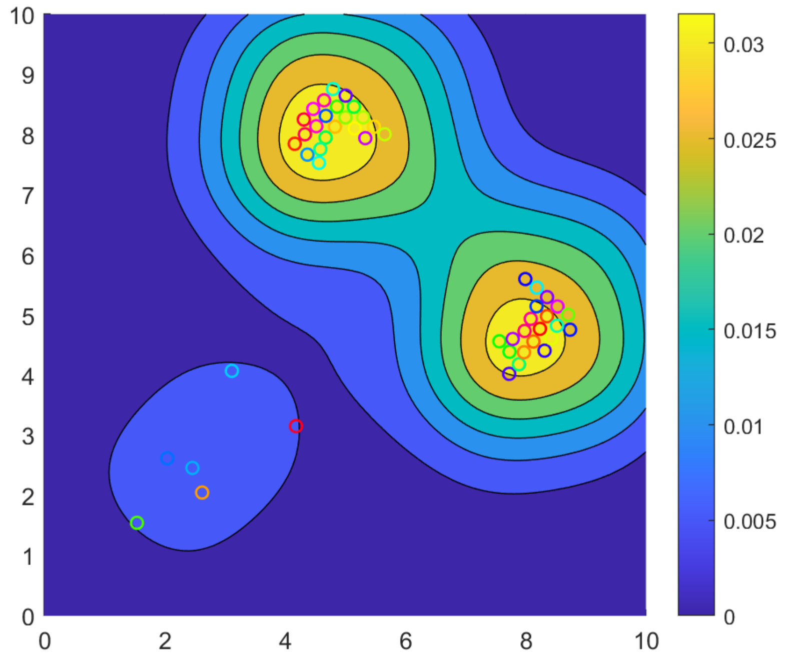


Figure 3.13.: Example of generated PDF showing the particle locations and in which their basis function locations are known.

3.5. Discussion

Analysis and control of large scale swarm systems require model representations that can scale with increasing number of agents within the swarm. This chapter has developed such representations for the agent behaviour under controlled conditions, and the controlling function, as well as the estimation of the swarm state for any feedback based swarm control.

The agent model was conceived as a simple motion model in 2-D with external forces that directly change the velocity of the individual robots in the swarm. Such models are used as standard and will be useful for the analysis of the swarm trajectories under any designed control mechanism. However, this scales linearly with the number of agent robots and is therefore not scalable to a large swarm. This motivated the development of using a probability density distribution to represent the swarm as a collective that is independent of the number of agents in the swarm.

The use of basis functions for representing field control as well as the probability distribution function was necessary to have finite dimensional representations that are amenable to analysis and computation. However, the choice of the basis functions can influence any resulting model equations to deviate from being a good approximation of a real robot swarm behaviour. The choice of Gaussian basis functions with design guidance from literature [52] was adopted owing to their potential for generating compact models, which are derived in the next chapter. The control field was similarly parametrised with Gaussian functions for the same reason.

Having obtained a Gaussian function based representation as the the probability density for the spatial distribution of the swarm, it was important to develop an estimation algorithm for the model states, based on the observation of individual robot agents. Given the difference of the basis function based probability representation to the mixture Gaussian density, the state estimation method had to be adapted from the well established EM algorithm [53]. Such an estimate is useful in any feedback control of the swarm, particularly when the robots are deemed passive.

One of the key aims for the control of swarms is to ensure that some desired spatial distribution of the robots is achieved, such as for example, splitting the robots towards two different locations. An evaluation metric that computes the deviation from the achievement of this task is necessary. This resulted in the choice of a symmetric KLD measure to represent this discrepancy which can also be used to check the accuracy of the estimation algorithm.

3.6. Summary

This chapter has presented the agent-based model of passive robot swarms as particle motion under external forces. The external forces themselves are derived from a potential field that is parametrised through Gaussian basis functions. The large numbers of swarm robots are then considered in a continuum model representation in which its probability density function is also represented through weighted basis functions. Estimation of such probability density functions are then carried out through a modified EM algorithm, along with the choice of a symmetric KLD as a metric for the assessment on the quality of estimation. Having developed the swarm and control field representations, as well as, the analyses required with them in this chapter, a model for the swarm dynamic behaviour is derived in the next chapter.

4

Reduced Model for Particle Swarm Behaviour

When swarms consist of a very large number of particles, analysing individual particle trajectories and design of a control system that applies equally to all particles scales computationally with their number. In the previous chapter we have defined a representation of the external forces from a potential field parametrised through Gaussian basis functions. Using this as its foundation, the aim of this chapter is to consider a continuum representation of the advection-diffusion model that considers and controls the collection of particles as a distribution. While this eliminates the scalability challenge in terms of particle numbers, the representation is infinite dimensional. One approach to the analysis and design to manipulate the swarms through control is to derive a reduced order representation of this advection-diffusion equation.

4.1. Advection-Diffusion Model Representation

The exposition begins with the generic advection-diffusion model in the partial differential equation (PDE) form. A structure for the swarm distributions and the velocity fields are then imposed towards translating the partial differential equation into an ordinary differential equation.

4.1.1 Advection-Diffusion Equation

Controlling a swarm of robots with an external potential field with the aim of moving these robots to different regions with different number of robots in each demands that a model with control representation be developed. As discussed in the previous chapter, with a large number of robots in a swarm, a spatial distribution representation can be an advantage in the model development.

The swarm robots as a distribution driven by a velocity field are described by the following advection-diffusion equation [10]:

$$\frac{\partial p(\mathbf{s},t)}{\partial t} = \gamma\ \nabla^2 p(\mathbf{s},t) - \nabla^\top \left[p(\mathbf{s},t)\mathbf{f}(\mathbf{s},t)\right] \tag{4.1}$$

where $p(\mathbf{s},t)$ is the probability distribution of a swarm of robots at time instant $t \in \mathcal{T}$ over the n-dimensional spatial region $\mathbf{s} = [s_1 \ \ s_2 \ \ \cdots \ \ s_n]^\top \in \mathcal{S} \subseteq \mathbb{R}^n$; $\mathbf{f}(\mathbf{s},t) : \mathcal{S} \times \mathcal{T} \mapsto \mathbb{R}^n$ is the velocity field acting on the swarm distribution at spatial location $\mathbf{s}$ at time instant t; γ is the diffusion constant parameter.

The $\nabla(\cdot)$ and $\nabla^2(\cdot)$ operators acting on function $f(\mathbf{s})$ on the spatial domain $\mathcal{S}$ are defined as:

$$\nabla f(\mathbf{s}) = \frac{\partial f(\mathbf{s})}{\partial \mathbf{s}} = \left[\frac{\partial f(\mathbf{s})}{\partial s_1} \ \ \frac{\partial f(\mathbf{s})}{\partial s_2} \ \ \cdots \ \ \frac{\partial f(\mathbf{s})}{\partial s_n}\right] \tag{4.2}$$

$$\nabla^2 f(\mathbf{s}) = \left[\frac{\partial^2 f(\mathbf{s})}{\partial s_1^2} + \frac{\partial^2 f(\mathbf{s})}{\partial s_2^2} + \cdots + \frac{\partial^2 f(\mathbf{s})}{\partial s_n^2}\right] \tag{4.3}$$

The advection-diffusion equation is a partial differential equation that is infinite dimensional and therefore, cannot be used for any control design for the manipulation of the swarm behaviour.

4.1.2 Basis Decomposition of the Advection-Diffusion Equation

The basis decompositions of the previous section are now applied on the advection-diffusion equation (4.1) representing the dynamics of a swarm of robots. In this derivation, the assumption is made that the state distribution representation is error free ($z(t) = 0$).

To obtain this form, let us consider each term in turn. Substituting (3.10),

$$\frac{\partial p(\mathbf{s},t)}{\partial t} = \frac{\partial\left(\Phi(\mathbf{s})^\top \mathbf{x}(t)\right)}{\partial t} = \Phi(\mathbf{s})^\top \frac{\partial \mathbf{x}(t)}{\partial t} = \Phi(\mathbf{s})^\top \dot{\mathbf{x}}(t) \tag{4.4}$$

Similarly,

$$\nabla^2 p(\mathbf{s},t) = \nabla^2\left(\Phi(\mathbf{s})^\top \mathbf{x}(t)\right) = \left[\nabla^2\Phi(\mathbf{s})\right]^\top \mathbf{x}(t) = \mathbf{x}(t)^\top\left[\nabla^2\Phi(\mathbf{s})\right] \tag{4.5}$$

The final term additionally requires substitution of equations (3.4) and (3.10),

$$\begin{aligned}
\nabla^\top\left[p(\mathbf{s},t)\mathbf{f}(\mathbf{s},t)\right] &= \nabla^\top\left[p(\mathbf{s},t)\nabla F(\mathbf{s},t)\right] \\
&= \left[\nabla p(\mathbf{s},t)\right]^\top \nabla F(\mathbf{s},t) + p(\mathbf{s},t)\left[\nabla^2 F(\mathbf{s},t)\right] \\
&= \left\{\nabla\left[\mathbf{x}(t)^\top\Phi(\mathbf{s})\right]\right\}\left[\nabla\Psi(\mathbf{s})^\top\mathbf{u}(t)\right] + \mathbf{x}(t)^\top\Phi(\mathbf{s})\left\{\nabla^2\left[\Psi(\mathbf{s})^\top\mathbf{u}(t)\right]\right\} \\
&= \mathbf{x}(t)^\top\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\mathbf{u}(t) + \mathbf{x}(t)^\top\Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top\mathbf{u}(t) \\
&= \mathbf{x}(t)^\top\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top + \Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top\right]\mathbf{u}(t)
\end{aligned} \tag{4.6}$$

The last expansion details are given in the Appendix A. It would be convenient to write this as,

$$\nabla\left[p(\mathbf{s},t)\mathbf{f}(\mathbf{s},t)\right] = \mathbf{x}(t)^\top\left[D^{(1)}(\mathbf{s}) + D^{(2)}(\mathbf{s})\right]\mathbf{u}(t) \tag{4.7}$$

where

$$D^{(1)}(\mathbf{s}) = \nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top, \qquad D^{(2)}(\mathbf{s}) = \Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top \tag{4.8}$$

Putting each of (4.4), (4.5) and (4.6) into (4.1) results in,

$$\Phi(\mathbf{s})^\top\dot{\mathbf{x}}(t) = \gamma\left[\nabla^2\Phi(\mathbf{s})\right]^\top\mathbf{x}(t) - \mathbf{x}(t)^\top\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top + \Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top\right]\mathbf{u}(t) \tag{4.9}$$

This representation is still infinite dimensional with continuous functions of $\mathbf{s}$ being present in the equation. Thus, a further approximation is required to obtain a finite dimensional representation of the swarm dynamics.

4.2. Reduced Advection-Diffusion Model

The infinite dimensional approximate system representation can be reduced to finite dimensional space by considering the projection of the system onto a set of finite number of basis functions as in the Galerkin approach. One approach is to use the same basis functions as those used for the probability distribution function for this projection so that the required computations are made easier.

4.2.1 Reduced Affine in Input Model

In this subsection, the Galerkin approximation steps are followed resulting in the model being affine in the control input.

The projection operation in the basis function space is an integral operation, and when applied to equation (4.9), results in,

$$\begin{aligned}\int_{\mathbb{S}} \Phi(\mathbf{s})\Phi(\mathbf{s})^\top \dot{\mathbf{x}}(t)d\mathbf{s} = \int_{\mathbb{S}} \Phi(\mathbf{s})\gamma\left[\nabla^2\Phi(\mathbf{s})\right]^\top \mathbf{x}(t)d\mathbf{s}\\ - \int_{\mathbb{S}} \Phi(\mathbf{s})\mathbf{x}(t)^\top \left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top + \Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top\right]\mathbf{u}(t)d\mathbf{s}\end{aligned} \tag{4.10}$$

Taking each term one by one,

$$\int_{\mathbb{S}} \Phi(\mathbf{s})\Phi(\mathbf{s})^\top \dot{\mathbf{x}}(t)d\mathbf{s} = \left[\int_{\mathbb{S}} \Phi(\mathbf{s})\Phi(\mathbf{s})^\top d\mathbf{s}\right]\dot{\mathbf{x}}(t) = \mathbf{E}\dot{\mathbf{x}}(t) \tag{4.11}$$

where $\mathbf{E} \in \mathbb{R}^{N_x \times N_x}$ and

$$\mathbf{E} = \int_{\mathbb{S}} \Phi(\mathbf{s})\Phi(\mathbf{s})^\top d\mathbf{s} \tag{4.12}$$

with the $(i,j)^{th}$ term given by,

$$E_{ij} = \int_{\mathbb{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})d\mathbf{s} \tag{4.13}$$

The second term,

$$\int_{\mathbb{S}} \Phi(\mathbf{s})\gamma\left[\nabla^2\Phi(\mathbf{s})\right]^\top \mathbf{x}(t)d\mathbf{s} = \gamma\left[\int_{\mathbb{S}} \Phi(\mathbf{s})\nabla^2\Phi(\mathbf{s})^\top d\mathbf{s}\right]\mathbf{x}(t) = \gamma\mathbf{A}\mathbf{x}(t) \tag{4.14}$$

where $\mathbf{A} \in \mathbb{R}^{N_x \times N_x}$ and

$$\mathbf{A} = \int_{\mathbb{S}} \Phi(\mathbf{s})\nabla^2\Phi(\mathbf{s})^\top d\mathbf{s} \tag{4.15}$$

with the $(i,j)^{th}$ term given by,

$$A_{ij} = \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\phi_j(\mathbf{s})d\mathbf{s} \tag{4.16}$$

The third term can be split into two, with the first being (see Appendix for details),

$$\begin{aligned}\int_{\mathbb{S}} \Phi(\mathbf{s})\mathbf{x}(t)^\top\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\right]\mathbf{u}(t)d\mathbf{s} &= \int_{\mathbb{S}} \mathrm{vec}\left(\Phi(\mathbf{s})\mathbf{x}(t)^\top\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\right]\mathbf{u}(t)\right)d\mathbf{s}\\ &= \int_{\mathbb{S}} \left(\mathbf{x}(t)^\top \otimes \mathbf{I}_{N_x}\right)\left(\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\right]\otimes\Phi(\mathbf{s})\right)\mathbf{u}(t)d\mathbf{s}\\ &= \left(\mathbf{x}(t)^\top \otimes \mathbf{I}_{N_x}\right)\left[\int_{\mathbb{S}}\left(\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\right]\otimes\Phi(\mathbf{s})\right)d\mathbf{s}\right]\mathbf{u}(t)\\ &= \left(\mathbf{x}(t)^\top \otimes \mathbf{I}_{N_x}\right)\mathbf{B}^{(1)}\mathbf{u}(t)\end{aligned} \tag{4.17}$$

where $\mathbf{B}^{(1)} \in \mathbb{R}^{N_x^2 \times N_u}$

$$\mathbf{B}^{(1)} = \int_{\mathbb{S}} \left(\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top \right] \otimes \Phi(\mathbf{s}) \right) d\mathbf{s} \tag{4.18}$$

This can be represented on block form as,

$$\mathbf{B}^{(1)} = \begin{bmatrix} \int_{\mathbb{S}} \nabla\phi_1(\mathbf{s})\nabla\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_1(\mathbf{s})\nabla\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_1(\mathbf{s})\nabla\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_{\mathbb{S}} \nabla\phi_{N_x}(\mathbf{s})\nabla\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_{N_x}(\mathbf{s})\nabla\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \nabla\phi_{N_x}(\mathbf{s})\nabla\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \end{bmatrix} \tag{4.19}$$

with each of the block elements represented by (i, j) itself a vector of $N_x{\times}1$ as a consequence of the Kronecker product. This block can be written as,

$$\int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} = \begin{bmatrix} \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\phi_1(\mathbf{s})d\mathbf{s} \\ \vdots \\ \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\phi_k(\mathbf{s})d\mathbf{s} \\ \vdots \\ \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\phi_{N_x}(\mathbf{s})d\mathbf{s} \end{bmatrix} \tag{4.20}$$

Noting that there are effectively three functions that are being combined, it would be useful to consider a new term $\chi^{(1)}_{ijk}$ with the $(i, j, k)^{th}$ term given by,

$$\chi^{(1)}_{ijk} = \int_{\mathbb{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\phi_k(\mathbf{s})d\mathbf{s} \tag{4.21}$$

This is the basic term that would need evaluation upon the choice of basis functions for $\phi(\cdot)$ and $\psi(\cdot)$.

The remaining part of the third term can be similarly expanded,

$$\int_{\mathbb{S}} \Phi(\mathbf{s})\mathbf{x}(t)^\top \left[\Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top \right] \mathbf{u}(t)d\mathbf{s} = \left(\mathbf{x}(t)^\top \otimes \mathbf{I}_{N_x} \right) \mathbf{B}^{(2)}\mathbf{u}(t) \tag{4.22}$$

where $\mathbf{B}^{(2)} \in \mathbb{R}^{N_x^2 \times N_u}$

$$\mathbf{B}^{(2)} = \int_{\mathbb{S}} \left(\left[\Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top \right] \otimes \Phi(\mathbf{s}) \right) d\mathbf{s} \tag{4.23}$$

This can be represented on block form as,

$$\mathbf{B}^{(2)} = \begin{bmatrix} \int_{\mathbb{S}} \phi_1(\mathbf{s})\nabla^2\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_1(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_1(\mathbf{s})\nabla^2\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \int_{\mathbb{S}} \phi_{N_x}(\mathbf{s})\nabla^2\psi_1(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_{N_x}(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\,(\Phi(\mathbf{s}))\,d\mathbf{s} & \cdots & \int_{\mathbb{S}} \phi_{N_x}(\mathbf{s})\nabla^2\psi_{N_u}\,(\Phi(\mathbf{s}))\,d\mathbf{s} \end{bmatrix} \tag{4.24}$$

with each of the block elements represented by (i,j) itself a vector of $N_x{\times}1$ as a consequence of the Kronecker product. This block can be written as,

$$\int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\left(\Phi(\mathbf{s})\right)d\mathbf{s} = \begin{bmatrix} \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\phi_1(\mathbf{s})d\mathbf{s} \\ \vdots \\ \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\phi_k(\mathbf{s})d\mathbf{s} \\ \vdots \\ \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\phi_{N_x}(\mathbf{s})d\mathbf{s} \end{bmatrix} \tag{4.25}$$

Again, a new term $\chi^{(2)}_{ijk}$ is defined with the $(i,j,k)^{th}$ term given by,

$$\chi^{(2)}_{ijk} = \int_{\mathbb{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\phi_k(\mathbf{s})d\mathbf{s} \tag{4.26}$$

Putting all of the three terms together and by defining $\mathbf{B} = \mathbf{B}^{(1)} + \mathbf{B}^{(2)}$ into the approximate advection-diffusion model of (4.10) gives $\mathbf{G} = \mathbf{G}^{(1)} + \mathbf{G}^{(2)}$, and

$$\mathbf{E}\dot{\mathbf{x}}(t) = \gamma\mathbf{A}\mathbf{x}(t) - \left(\mathbf{x}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{B}\mathbf{u}(t) \tag{4.27}$$

The matrix $\mathbf{E}$ is invertible of the linearly independent basis function set, which then gives the final form of the state equation:

$$\dot{\mathbf{x}}(t) = \gamma\mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{x}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{B}\mathbf{u}(t) \tag{4.28}$$

The above form of the state space equation is useful for control design as the representation is affine in $\mathbf{u}(t)$.

4.2.2 Reduced Affine in States Model

However, an analysis of system behaviour under a control input would benefit from an affine in $\mathbf{x}(t)$ representation. The affine in $\mathbf{u}(t)$ representation above shows that this third term is bilinear in $\mathbf{u}(t)$ and $\mathbf{x}(t)$. Note that an alternative form of the Kronecker product algebra (details in Appendix) would give the third term to be written as,

$$\begin{aligned} \int_{\mathbb{S}} \Phi(\mathbf{s})\mathbf{x}(t)^{\top}\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^{\top}\right]\mathbf{u}(t)d\mathbf{s} &= \left(\mathbf{u}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{G}^{(1)}\mathbf{x}(t) \\ \int_{\mathbb{S}} \Phi(\mathbf{s})\mathbf{x}(t)^{\top}\left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^{\top}\right]\mathbf{u}(t)d\mathbf{s} &= \left(\mathbf{u}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{G}^{(2)}\mathbf{x}(t) \end{aligned} \tag{4.29}$$

where $\mathbf{G}^{(1)}, \mathbf{G}^{(2)} \in \mathbb{R}^{N_u N_x \times N_x}$ and given by,

$$\begin{aligned} \mathbf{G}^{(1)} &= \int_{\mathbb{S}} \left(\left[\nabla\Psi(\mathbf{s})\nabla\Phi(\mathbf{s})^{\top}\right] \otimes \Phi(\mathbf{s})\right)d\mathbf{s} \\ \mathbf{G}^{(2)} &= \int_{\mathbb{S}} \left(\left[\nabla^2\Psi(\mathbf{s})\Phi(\mathbf{s})^{\top}\right] \otimes \Phi(\mathbf{s})\right)d\mathbf{s} \end{aligned} \tag{4.30}$$

These can be represented in block forms as,

$$\mathbf{G}^{(1)} = \begin{bmatrix} \int_S \nabla\psi_1(\mathbf{s})\nabla\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_1(\mathbf{s})\nabla\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_1(\mathbf{s})\nabla\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_S \nabla\psi_i(\mathbf{s})\nabla\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_i(\mathbf{s})\nabla\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_i(\mathbf{s})\nabla\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_S \nabla\psi_{N_u}(\mathbf{s})\nabla\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_{N_u}(\mathbf{s})\nabla\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla\psi_{N_u}(\mathbf{s})\nabla\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \end{bmatrix} \tag{4.31}$$

and

$$\mathbf{G}^{(2)} = \begin{bmatrix} \int_S \nabla^2\psi_1(\mathbf{s})\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_1(\mathbf{s})\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_1(\mathbf{s})\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_S \nabla^2\psi_i(\mathbf{s})\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_i(\mathbf{s})\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_i(\mathbf{s})\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \int_S \nabla^2\psi_{N_u}(\mathbf{s})\phi_1(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_{N_u}(\mathbf{s})\phi_j(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} & \cdots & \int_S \nabla^2\psi_{N_u}(\mathbf{s})\phi_{N_x}(\mathbf{s})\Phi(\mathbf{s})\,d\mathbf{s} \end{bmatrix} \tag{4.32}$$

The individual elements of the above matrices are formed by the same elements as $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}$ but shaped differently into the appropriate sized matrices. The relationship between them are explored

The equivalent representation that is affine in $\mathbf{x}(t)$ and $\mathbf{G} = \mathbf{G}^{(1)} + \mathbf{G}^{(2)}$ is given by,

$$\dot{\mathbf{x}}(t) = \gamma\mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{u}(t)^\top \otimes \mathbf{I}_M\right)\mathbf{G}\mathbf{x}(t) = \mathbf{E}^{-1}\left[\gamma\mathbf{A} - \left(\mathbf{u}(t)^\top \otimes \mathbf{I}_{N_x}\right)\mathbf{G}\right]\mathbf{x}(t) \tag{4.33}$$

This form of the representation is convenient for the analysis and simulation of the system for known external potential field inputs.

4.2.3 Relationship Between B and G Matrices

In this section, we investigate the relationship between the $\mathbf{G}$ and $\mathbf{B}$ state matrices, in which $\mathbf{G} \in \mathbb{R}^{N_u N_x \times N_x}$ and $\mathbf{B} \in \mathbb{R}^{N_x 2 \times N_u}$.

From (4.21) and (4.26), note that the term $\chi^{(1)}_{ijk}$ and $\chi^{(2)}_{ijk}$ feature in both $\mathbf{G}$ and $\mathbf{B}$. By defining,

$$\tilde{\chi}^{(1)}_{ij} = \begin{bmatrix} \chi^{(1)}_{ij1} \\ \vdots \\ \chi^{(1)}_{ijk} \\ \vdots \\ \chi^{(1)}_{ijN_x} \end{bmatrix} \in \mathbb{R}^{N_x} \tag{4.34}$$

the term $\mathbf{B}^{(1)}$ from (4.19) can be written as,

$$\mathbf{B}^{(1)} = \begin{bmatrix} \tilde{\chi}_{11}^{(1)} & \cdots & \tilde{\chi}_{1j}^{(1)} & \cdots & \tilde{\chi}_{1N_u}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \tilde{\chi}_{i1}^{(1)} & \cdots & \tilde{\chi}_{ij}^{(1)} & \cdots & \tilde{\chi}_{iN_u}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \tilde{\chi}_{N_x1}^{(1)} & \cdots & \tilde{\chi}_{N_xj}^{(1)} & \cdots & \tilde{\chi}_{N_xN_u}^{(1)} \end{bmatrix} \tag{4.35}$$

Similarly, the term $\mathbf{G}^{(1)}$ from (4.31) can be written as,

$$\mathbf{G}^{(1)} = \begin{bmatrix} \tilde{\chi}_{11}^{(1)} & \cdots & \tilde{\chi}_{i1}^{(1)} & \cdots & \tilde{\chi}_{N_x1}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \tilde{\chi}_{1j}^{(1)} & \cdots & \tilde{\chi}_{ij}^{(1)} & \cdots & \tilde{\chi}_{N_xi}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \tilde{\chi}_{1N_u}^{(1)} & \cdots & \tilde{\chi}_{iN_u}^{(1)} & \cdots & \tilde{\chi}_{N_xN_u}^{(1)} \end{bmatrix} \tag{4.36}$$

The relationship between $\mathbf{B}^{(1)} \in \mathbb{R}^{N_x^2 \times N_u}$ and $\mathbf{G}^{(1)} \in \mathbb{R}^{N_xN_u \times N_x}$ looks like they are a transpose of each other. However, it should be noted that this operation is like a transpose of blocks of vectors which gives rise to their dimensions.

The pattern of relationship between $\mathbf{B}^{(2)}$ and $\mathbf{G}^{(2)}$ is similarly linked through a similar definition for $\tilde{\chi}_{ij}^{(2)} \in \mathbb{R}^{N_x}$ consisting of terms $\chi_{ijk}^{(2)}$.

It is to be noted that some of the states may be negative due to the boundary effect. Thus, we use the H matrix, although this may be a violation of reality, to respect the non-negativity constraint. Attributed the negative to other x.

4.2.4 Reduced-Order Model Simulations

The reduced order model was simulated for an example with 9 basis functions with N_x and N_u having a three by three grid, in order to see if the model has represents behaviours that are typical of swarms. The simulation of the continuous-time model was carried out using the Euler method, so that,

$$\mathbf{x}(k+1) = \mathbf{x}(k) + h\,[\dot{\mathbf{x}}(t)]_{t=kh} \tag{4.37}$$

where h is the step size parameter.

First, a simulation with zero external potential field input was carried out with the results shown in Figures 4.1 and 4.2 where trajectories of $\mathbf{x}(t)$ over time and the evolving probability density of the swarm are given respectively. The Figures shows that the states are stable as would be expected of a diffusion process but that the states all decay to zero. What this implies is that the sum of the states are no longer equal to unity, thus violating the constraint that the states are coefficients of a probability distribution.

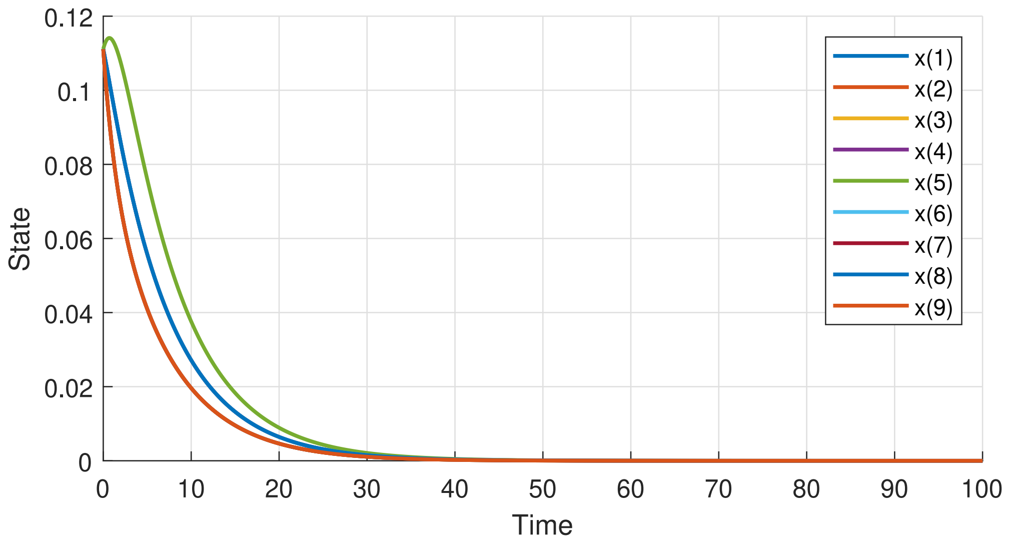


Figure 4.1.: Time series under zero input for grid size $N_x = 9$. Some state trajectories are exactly the same as others and so are overlapping and do not appear as separate curves.

Furthermore, when the external potential field input of a large enough magnitude was applied, the system showed unstable behaviour with the states growing unbounded over time. This result in shown in Figures 4.3. The analysis of the state constraints is carried out with the Figure 4.4 showing that the states do not sum to unity.

The lack of constraint satisfaction and potential system dynamics instability, both point to a deficiency in the model that needs further investigation. Recalling that the states should sum to unity suggests that the reduced order model eigenvalues should not only be negative and stable, but should also have an eigenvalue of 0.

An eigenvalue analysis was then carried out on the two separate part of the model equations as well as for the full model. These are given in Table 4.1.

The analysis reveals that the model is incapable of representing the swarm dynamics and so required further analysis. In particular, the reduced order model should not only have stable eigenvalues as would be expected of swarm dynamics with zero or constant inputs, but should also satisfy the state constraints of its sum being equal to unity.

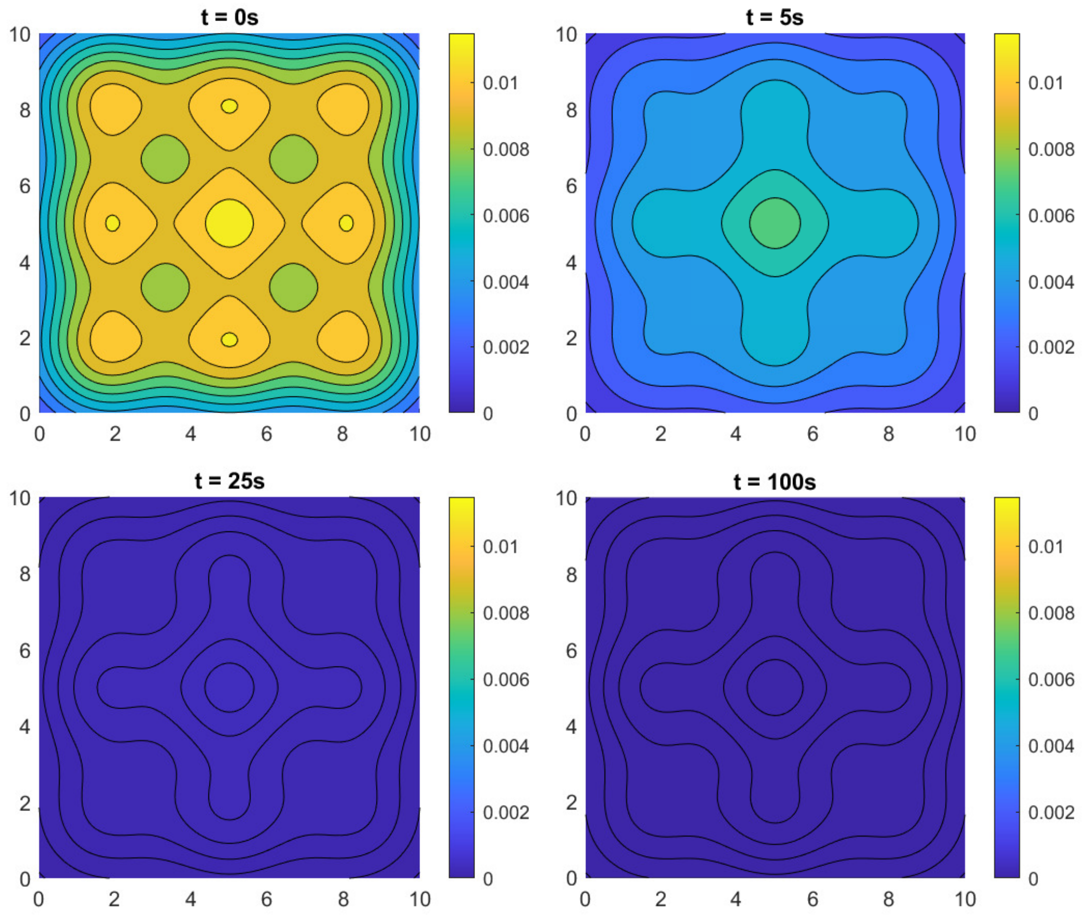


Figure 4.2.: Generated PDF under zero input for grid size $N_x = 9$.

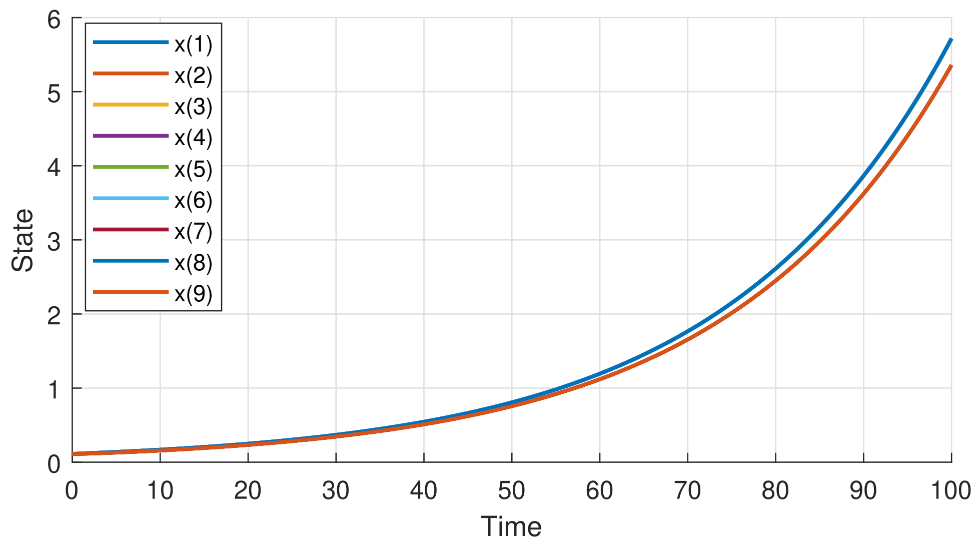


Figure 4.3.: Time series with all inputs elements being $u = 25$ showing a unstable output for grid size $N_x = 9$. Some state trajectories are exactly the same as others and so are overlapping and do not appear as separate curves.

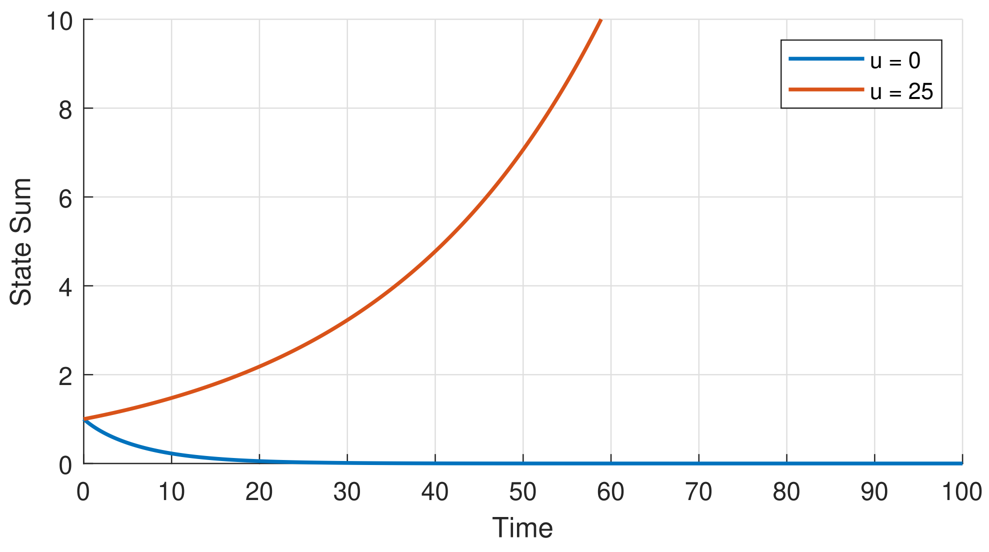


Figure 4.4.: States do not sum to one.

Table 4.1.: Eigenvalues of unstable inputs of $u = 25$.

Diffusion	Advection	Diffusion & Advection
-1.2684	-0.1081	-0.9203
-0.1430	0.4048	0.0392
-0.5505	-0.3703	-0.0838
-0.7057	-0.3363	-0.2143
-0.9095	-0.2356	-0.5665
-0.3468	-0.2725	-0.4507
-0.3468	-0.3703	-0.0838
-0.9095	-0.2731	-0.4326
-0.7057	-0.2356	-0.5665

4.3. Error Compensated Reduced-Order Model

The reduced order model suffers from the fact that the constraints of $\mathbf{x}$ are not satisfied in the dynamical state equation. In particular, even if the constraint of the probability distribution being satisfied such that $\mathbf{1}^\top \mathbf{x}(t) = 1$, the subsequent evolution may lead to $\mathbf{1}^\top \mathbf{x}(t^+) \neq 1$, thus violating the required constraint. This arises because, with

$$\dot{\mathbf{x}}(t) = \gamma \mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G}\mathbf{x}(t) \tag{4.38}$$

it cannot be guaranteed for any valid state $\mathbf{x}(t)$ nor for any valid inputs $\mathbf{u}(t)$ that

$$\mathbf{1}^\top \dot{\mathbf{x}}(t) = 0 \tag{4.39}$$

The analysis follows from the logic that if $\mathbf{1}^\top \mathbf{x}(t) = 1$ at time t and that $\mathbf{1}^\top \dot{\mathbf{x}}(t) = 0$ all subsequent state evolution will satisfy $\mathbf{1}^\top \mathbf{x}(t^+) = 1$.

4.3.1 Error Compensation

Let us first consider the zero input case $\mathbf{u}(t) = 0$. Then, the model equation is,

$$\dot{\mathbf{x}}(t) = \gamma \mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) \tag{4.40}$$

However, noting that $\mathbf{d}^\top = \mathbf{1}^\top \mathbf{E}^{-1}\mathbf{A}$ is a constant vector, there is no guarantee that the conditions,

$$\mathbf{d}^\top \mathbf{x}(t) = 0, \quad \mathbf{1}^\top \mathbf{x}(t) = 1 \tag{4.41}$$

can both be simultaneously satisfied by all valid state vectors with $n_x - 1$ degrees of freedom. Hence, the state equation requires a correction term.

In order to arrive at a compensation, a further assumption must be made. Let the errors be assumed to be proportional to the magnitude of the state vector and so the correction to be made to the state dynamics is proportional to the magnitude of the state vector.

The compensation term for the diffusion part be written as $\Delta_x \mathbf{x}(t)$. With this compensation scheme, the error corrected dynamics with zero input becomes,

$$\dot{\mathbf{x}}(t) = \gamma \mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) + \Delta_x \mathbf{x}(t) = \left[\gamma \mathbf{E}^{-1}\mathbf{A} + \Delta_x\right]\mathbf{x}(t) \tag{4.42}$$

The choice of Δ_x should be such that the constraint $\mathbf{1}^\top \dot{\mathbf{x}}(t) = 0$ is satisfied for any valid $\mathbf{x}(t)$. Considering the term $\mathbf{1}^\top \dot{\mathbf{x}}(t)$,

$$\left[\mathbf{1}^\top \gamma \mathbf{E}^{-1}\mathbf{A} + \mathbf{1}^\top \Delta_x\right]\mathbf{x}(t) = 0 \tag{4.43}$$

Since this should be satisfied for any valid $\mathbf{x}(t)$, a choice for Δ_x can be made to ensure that the following condition is always true:

$$\mathbf{1}^\top \gamma \mathbf{E}^{-1}\mathbf{A} + \mathbf{1}^\top \Delta_x = 0 \tag{4.44}$$

This has N_x number of constraints and so one possible choice is $\Delta_x = \text{diag}[\mathbf{d}_x]$ where $\mathbf{d}_x \in \mathbb{R}^{N_x}$. By noting that $\mathbf{1}^\top \Delta_x = \mathbf{d}_x^\top$, this choice reduces the above equation to,

$$\mathbf{1}^\top \gamma \mathbf{E}^{-1}\mathbf{A} + \mathbf{d}_x^\top = 0 \tag{4.45}$$

This results in the solution that

$$\mathbf{d}_x = -\gamma \mathbf{A}^\top \mathbf{E}^{-1}\mathbf{1} \tag{4.46}$$

Considering the full model in the original form,

$$\dot{\mathbf{x}}(t) = -\gamma \mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{u}^\top \otimes \mathbf{I}_{N_x}\right)\mathbf{G}\mathbf{x}(t) \tag{4.47}$$

note that the term $\mathbf{1}^\top \dot{\mathbf{x}}(t)$ of the advection part of the equation for some valid $\mathbf{u}(t)$ and $\mathbf{x}(t)$,

$$\mathbf{1}^\top \dot{\mathbf{x}}(t) = \mathbf{1}^\top \mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G}\mathbf{x}(t) \neq 0 \tag{4.48}$$

Hence, a compensation term Δ_u should also be incorporated, along with Δ_x. This leads to the compensated model of the form,

$$\dot{\mathbf{x}}(t) = \left[\gamma \mathbf{E}^{-1}\mathbf{A} + \Delta_x\right]\mathbf{x}(t) - \left[\mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G} + \Delta_u\right]\mathbf{x}(t) \tag{4.49}$$

Now, considering $\mathbf{1}^\top \dot{\mathbf{x}}(t)$ for this compensated model,

$$\mathbf{1}^\top \dot{\mathbf{x}}(t) = \mathbf{1}^\top \left(\gamma \mathbf{E}^{-1}\mathbf{A} + \Delta_x\right)\mathbf{x}(t) - \mathbf{1}^\top \left[\mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G} + \Delta_u\right]\mathbf{x}(t) \tag{4.50}$$

Likewise as with the steps in determining Δ_x, the choice of Δ_u is made such that the constraint $\mathbf{1}^\top \dot{\mathbf{x}}(t) = 0$ is satisfied with any valid $\mathbf{x}(t)$. Noting the result in equation (4.44), the above condition reduces to,

$$-\left[\mathbf{1}^\top \mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G} + \mathbf{1}^\top \Delta_u\right]\mathbf{x}(t) = 0 \tag{4.51}$$

which can be guaranteed for any $\mathbf{x}(t)$, if

$$-\mathbf{1}^\top \mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G} - \mathbf{1}^\top \Delta_u = 0. \tag{4.52}$$

By noting that $\mathbf{1}^\top \Delta_u = \mathbf{d}_u^\top$, if $\Delta_u = \text{diag}[\mathbf{d}_u]$, the above equation reduces to,

$$-\mathbf{1}^\top \mathbf{E}^{-1}\left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right)\mathbf{G} - \mathbf{d}_u^\top = 0, \tag{4.53}$$

resulting in

$$\mathbf{d}_u = -\mathbf{G}^\top \left(\mathbf{u}(t) \otimes \mathbf{I}_{N_x}\right) \mathbf{E}^{-1}\mathbf{1}. \tag{4.54}$$

Putting all of the analysis together, the state equation of the error compensated reduced order model is given by,

$$\dot{\mathbf{x}}(t) = \left[\gamma \mathbf{E}^{-1}\mathbf{A} + \Delta_x\right] \mathbf{x}(t) - \left[\mathbf{E}^{-1} \left(\mathbf{u}^\top(t) \otimes \mathbf{I}_{N_x}\right) \mathbf{G} + \Delta_u\right] \mathbf{x}(t) \tag{4.55}$$

where

$$\Delta_x = \text{diag}[\mathbf{d}_x], \quad \mathbf{d}_x = -\gamma \mathbf{A}^\top \mathbf{E}^{-1}\mathbf{1} \tag{4.56}$$

and

$$\Delta_u = \text{diag}[\mathbf{d}_u], \quad \mathbf{d}_u = -\mathbf{G}^\top \left(\mathbf{u}(t) \otimes \mathbf{I}_{N_x}\right) \mathbf{E}^{-1}\mathbf{1} \tag{4.57}$$

With this compensation, the state constraints and the state evolution constraints can be guaranteed.

4.3.2 Compensated Reduced-Order Model Simulations

Simulations of the same example as in subsection 4.2.4 was carried out now with the compensated reduced-order model. Simulations with zero external potential field input are shown in Figures 4.5 and 4.6. The state trajectories are now not only stable but also sum to unity. The probability density evolution is also stable because of the changes from its initial configuration being small.

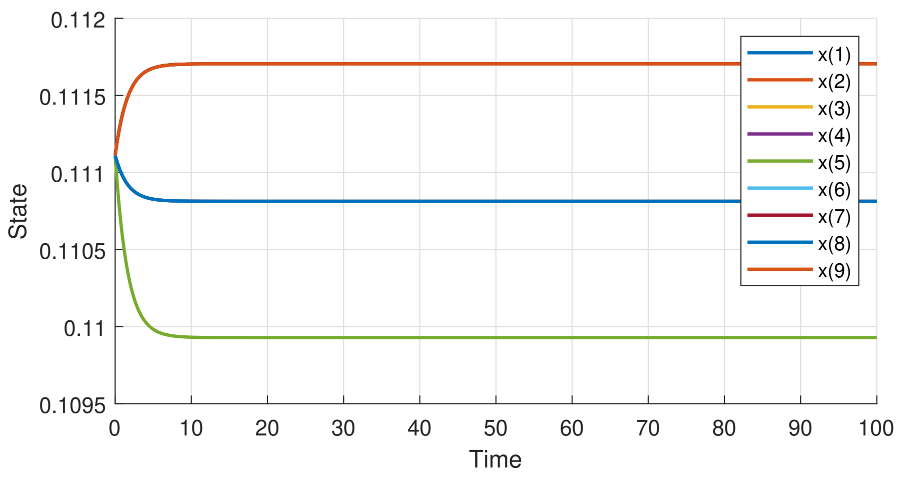


Figure 4.5.: Time series under zero input for grid size $N_x = 9$. Some state trajectories are exactly the same as others and so are overlapping and do not appear as separate curves.

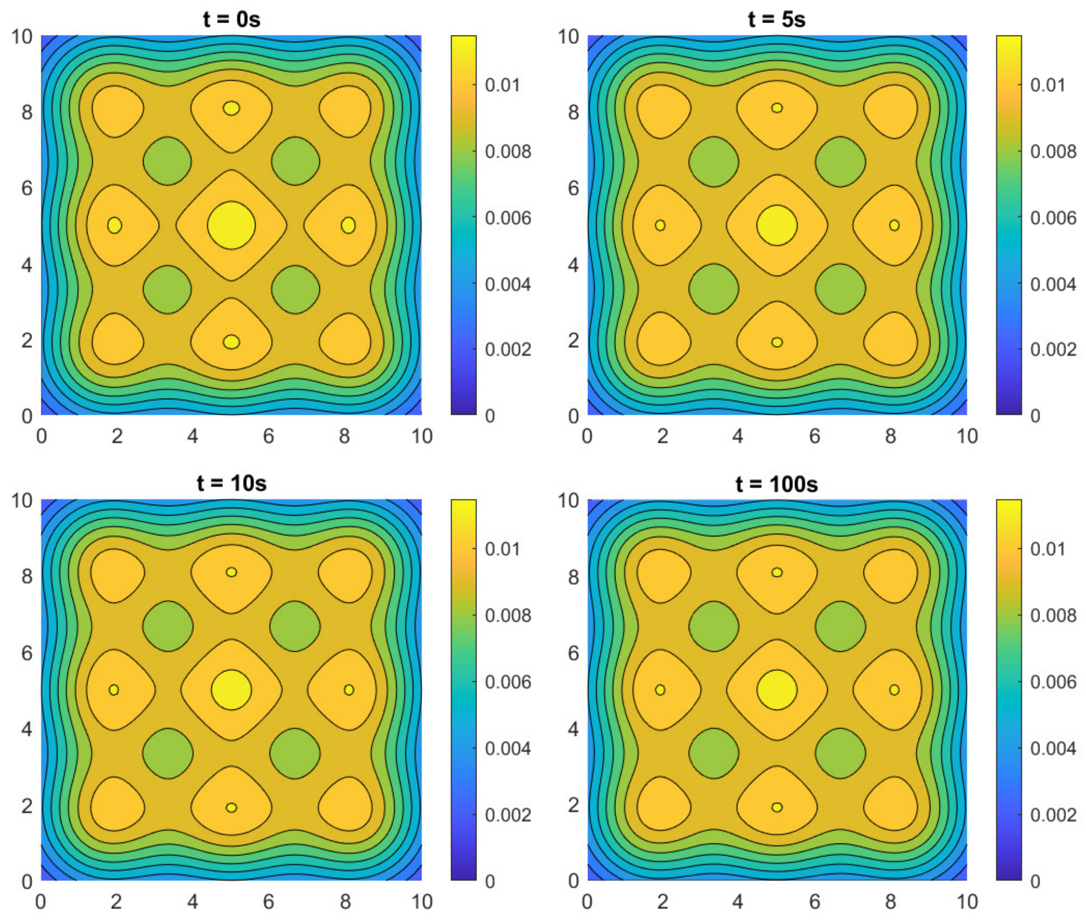


Figure 4.6.: Generated PDF under zero input for grid size $N_x = 9$.

Simulation with a constant high potential field input is shown in Figures 4.7 and 4.8. Again, a stable response is seen in the state trajectories and the state constraints are also satisfied. The probability distribution of the swarm clearly show their concentration around where the potential field external inputs are applied. This confirms that the compensated reduced order model shows expected swarm behaviour.

However, when the constant potential field input was increased further, unstable trajectories were observed as seen in Figures 4.9.

In order to understand why the unstable trajectories were observed, an eigenvalue analysis was carried out. The Table 4.2 shows the eigenvalues for stable input and unstable input, examples used in the previous analysis, respectively. It shows the eigenvalues for the diffusion and drift terms for the error compensated model for the stable and unstable input cases.

This dichotomous behaviour is attributed to the Euler method of simulation of the continuous-time system that can be improved upon either by a reducing the step size or utilising other

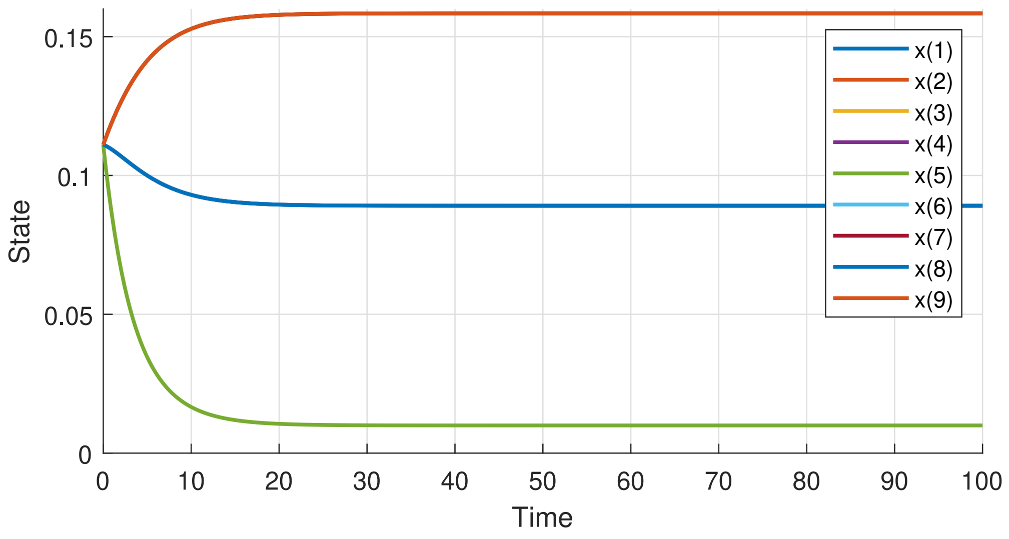


Figure 4.7.: Time series with all inputs elements being $u = 100$ showing a stable output for grid size $N_x = 9$. Some state trajectories are exactly the same as others and so are overlapping and do not appear as separate curves.

simulation approaches. Given that this instability is seen only for very high inputs, all subsequent analysis will ensure that a valid input limit is applied so that the simulation remains stable for the compensated reduced-order model.

As for respecting the equality constraint where the states sum to unity, both the zero and stable input cases are presented in Figure 4.10. It is only in the case of the unstable input case where the states do not sum to one due to the errors stemming from the Euler method simulation.

4.4. Discussion

In order to control a swarm of robots which consist of a very large number of particles, scalability becomes an issue as analysing individual trajectories and design of a control system that applies equally to all particles scales computationally with their number. Thus, in this chapter, a continuum representation of the advection-diffusion model is considered to represent the swarm as a distribution, which eliminates the scalability challenge. However, in order to use this model for control design for the manipulation of the swarm behaviour, this infinite-dimensional representation must be first parametrised to make this model finite-dimensional via projecting the system onto a set of finite number of basis functions, as in the Galerkin approximation approach. As this model represents a probability density function,

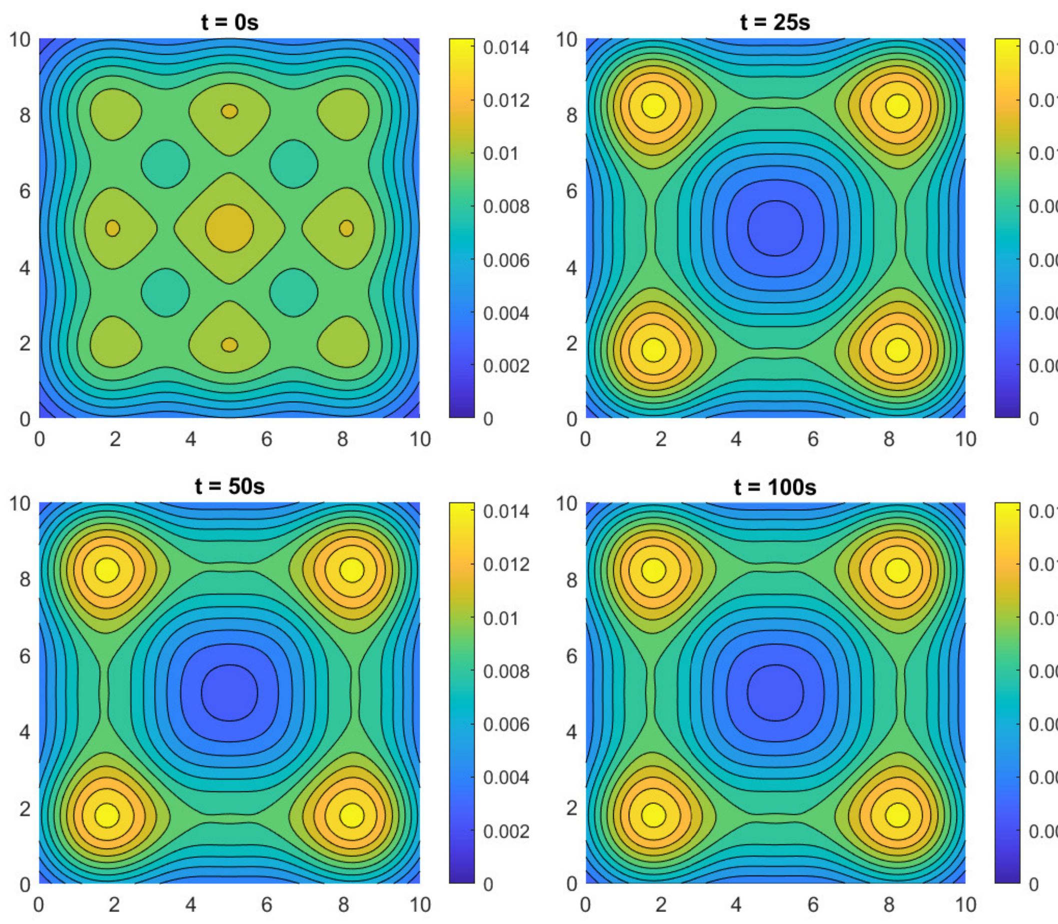


Figure 4.8.: Generated PDF with all inputs elements being $u = 100$ showing a stable output for grid size $N_x = 9$.

the states must also be non-negative and sum to one.

Simulations were performed on the non-error-compensated reduced-order advection-diffusion model via the Euler method, which resulted in the states summing to unity. However, no matter what input was used used, the sum of the states always dropped either to zero or grew unbounded greater than one over time. Another factor due to this is the choice of step size used when performing the simulation using the Euler method. This lack of constraint satisfaction both point to a deficiency of this model representing the swarm dynamics. required us to develop an error-compensated reduced-order model to obey the constraints.

In order to satisfy the equality constraint of the states summing to unity, compensation terms were included, in which these terms assume the error to be proportional to the magnitude of the state vector. These compensation terms, which are calculated for both the diffusion and advection parts, have guaranteed obeying the state constraints as shown in the simulations via the Euler method and eigenvalue analysis, regardless of the choice of input magnitudes or step size of the Euler method. These results have supported the assumption that the error

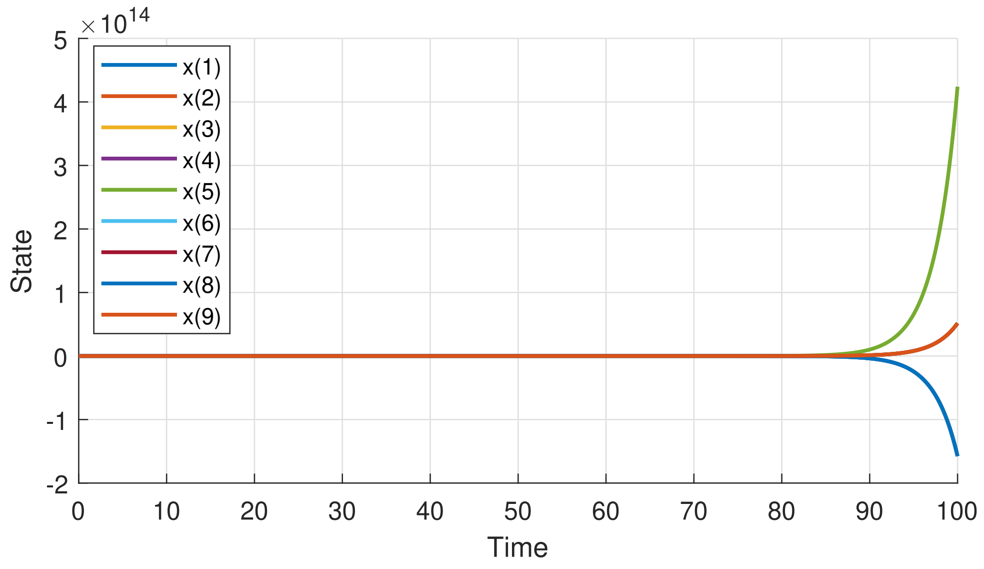


Figure 4.9.: Time series with all inputs elements being $u = 200$ showing an unstable output for grid size $N_x = 9$. Some state trajectories are exactly the same as others and so are overlapping and do not appear as separate curves.

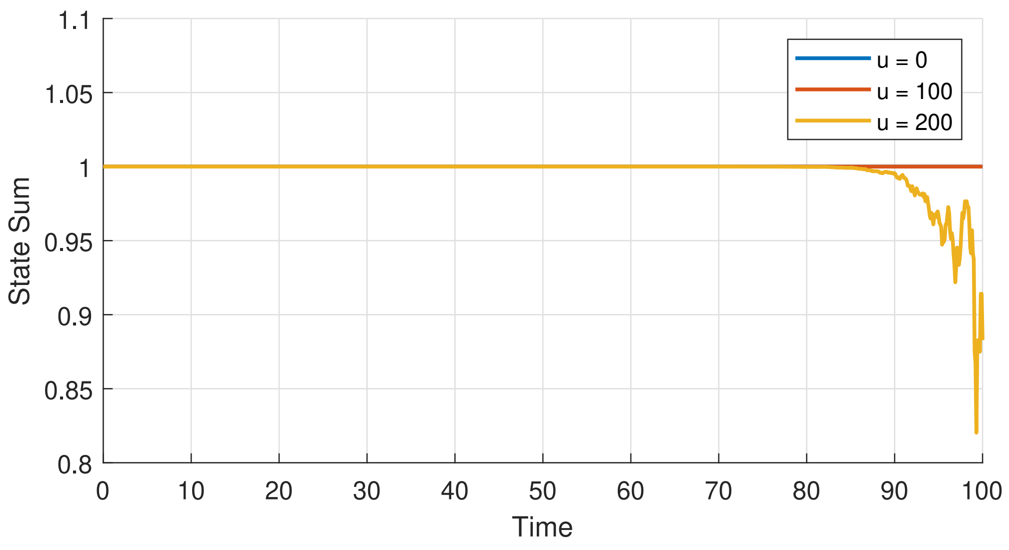


Figure 4.10.: Sum of the states for zero input, stable input and unstable input.

to be proportional to the magnitude of the state vector was the correct choice.

4.5. Summary

This chapter derived a reduced-order advection-diffusion model using the Galerkin approach and Gaussian basis functions. By choosing the external potential field input to be also para-

Table 4.2.: Eigenvalues of Stable (υ = 100) and Unstable (υ = 200) Inputs for Grid Size N_x = 9.

Stable			Unstable		
Diffusion	Advection	Diffusion & Advection	Diffusion	Advection	Diffusion & Advection
0	0	0	0	0	0
-1.1951	-0.7973	-0.3849	-1.1951	-1.5946	0.3803
-0.2504	-0.0911	-0.0821	-0.2504	-0.1822	0.0863
-0.1252	-0.1684	-0.2328	-0.1252	-0.3367	0.1770
-0.1252	-0.3777	-0.0364	-0.1252	-0.7554	0.0639
-0.7228	-0.4669	-0.0364	-0.7228	-0.9337	0.0639
-0.7228	-0.4669	-0.1943	-0.7228	-0.9337	0.2089
-0.5976	-0.4033	-0.2536	-0.5976	-0.8065	0.2040
-0.5976	-0.0911	-0.2536	-0.5976	-0.1822	0.2040

metrised with Gaussian basis functions, the model terms could be expressed analytically. The advantage of such an analytically derived reduced-order model is the ability to use model-based control designs to manipulate swarm behaviour.

The reduced-order model however proved to have deficiencies including not satisfying the required constraints of the swarm distribution. An analysis of the errors and the eigenvalues of the model revealed the need for modification to the model. This leads to the derivation of an error compensated reduced-order model that satisfied the required conditions.

5 Control of Particle Swarm Dynamics

Given that the passive agents have limited computational or actuation capabilities, the central controller is not able to direct each agent explicitly with control actions, but instead relies solely on potential functions for agent movement and manipulation. In more challenging cases where the passive particles may not end up in the desired location, the potential field may be amplified but this would not be the most optimal or cost-effective way. In Chapter 3, simulations were presented where having a static potential field could bring the passive robots to the desired locations and distributions and in the previous chapter, we have presented the reduced-order model to be used in the development of a distributional control framework. This control framework is then used to control the trajectories of the particle swarm for open-loop control and closed-loop control with simulations of various particle swarm objectives under different parameters and basis function bandwidths.

5.1. Trajectory Optimisation and Model Predictive Control

A general continuous-time optimal control problem (OCP) can be expressed in the so called Bolza form [55, 56],

$$\min_{\substack{x(\cdot),u(\cdot)\\,t_0,t_f}} V_M(x(t_0),x(t_f),u(t_0),u(t_f),t_0,t_f)+\int_{t_0}^{t_f}\ell(\dot{x}(t),x(t),u(t),t)dt \tag{5.1}$$

subject to

$$g_E(\dot{x}(t),x(t),u(t),t)=0,\quad \forall t\in[t_0,t_f], \tag{5.2a}$$

$$g_N(\dot{x}(t),x(t),u(t),t)\leq 0,\quad \forall t\in[t_0,t_f], \tag{5.2b}$$

$$\psi_E(x(t_0),x(t_f),u(t_0),u(t_f),t_0,t_f)=0, \tag{5.2c}$$

$$\psi_I(x(t_0),x(t_f),u(t_0),u(t_f),t_0,t_f)\leq 0 \tag{5.2d}$$

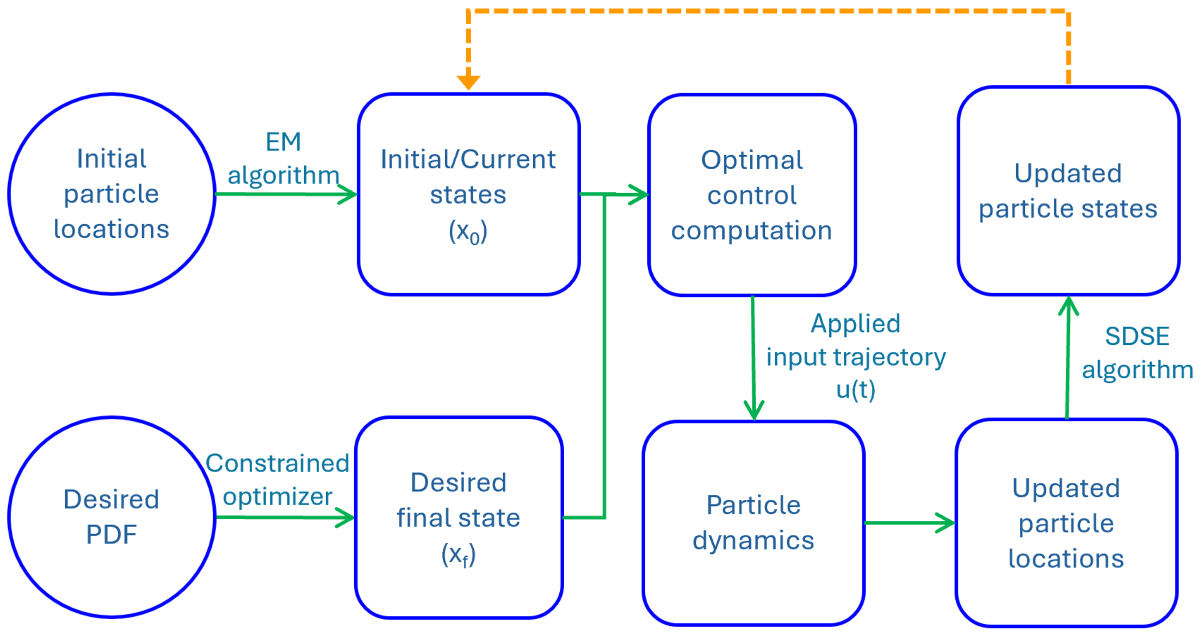
where $x:\mathbb{R}\longrightarrow\mathbb{R}^{N_n}$ is the state trajectory, $u:\mathbb{R}\longrightarrow\mathbb{R}^{N_m}$ is the input trajectory and t is the time variable with $t_0\in\mathbb{R}$ and $t_f\in\mathbb{R}$ the initial and final time.

In terms of the constraints, the system dynamics are enforced in the form of ordinary differential equations or differential algebraic equations with $f:\mathbb{R}^{N_n}\times\mathbb{R}^{N_n}\times\mathbb{R}^{N_m}\times\mathbb{R}\longrightarrow\mathbb{R}^{N_f}$. The inequality path constraint are expressed in $g:\mathbb{R}^{N_n}\times\mathbb{R}^{N_n}\times\mathbb{R}^{N_m}\times\mathbb{R}\longrightarrow\mathbb{R}^{N_g}$. $\psi_E:\mathbb{R}^{N_n}\times\mathbb{R}^{N_n}\times\mathbb{R}^{N_m}\times\mathbb{R}^{N_m}\times\mathbb{R}\times\mathbb{R}\longrightarrow\mathbb{R}^{n_E}$ and $\psi_I:\mathbb{R}^{N_n}\times\mathbb{R}^{N_n}\times\mathbb{R}^{N_m}\times\mathbb{R}^{N_m}\times\mathbb{R}\times\mathbb{R}\longrightarrow\mathbb{R}^{n_I}$ are the equality and inequality boundary constraints, respectively. Most practical OCPs arising from real-world engineering applications need to be solved numerically. There are many literature reviewing different numerical schemes that may be used, e.g. the overview provided in [56]. In this work, the Matlab Toolbox ICLOCS2 [57] is used to transcribe the OCPs with the direct collocation approach [58] into nonlinear programming problems (NLPs), which are subsequently solved by the interior-point method-based NLP solver IPOPT [59].

The key idea behind model predictive control (MPC) is to solve and implement OCP solutions in a closed-loop setting, to better handle uncertainties that, for example, arise from model mismatches and external disturbances. With MPC, only the first segment of the OCP solution will be implemented. As new measurements arrive, the OCPs will be repetitively solved to update the control action, effectively function as an implicit feedback law. In this work, with directly compare the results obtained via open-loop implementation of the OCP solutions and the closed-loop implementation with a MPC framework.

5.2. Optimal Control Problem Description

To bring the passive robot swarm from the initial locations in the arena to the desired final objective PDF, the modified swarm distributional states estimation (SDSE) algorithm, as presented in Section 3.4.2, is used to calculate the initial distributional swarm states $\mathbf{x}_0$ can be calculated based on the robot locations. Likewise, a constrained optimiser can be used to translate the desired PDF to the desired final state $\mathbf{x}_f$. What is needed is to find a solution of a sequence of input trajectories driving the initial states $\mathbf{x}_0$ to the final desired states $\mathbf{x}_f$. In order to do this, there is a balance between the the error and the input effort. This process is shown in the diagram in Figure 5.1.



Initial particle locations
EM algorithm
Initial/Current states (x_0)
Optimal control computation
Updated particle states
Applied input trajectory u(t)
SDSE algorithm
Desired PDF
Constrained optimizer
Desired final state (x_f)
Particle dynamics
Updated particle locations

Figure 5.1.: Diagram of the closed-loop optimal control problem.

5.2.1 Optimal Control Problem Formulation

The optimal control problem is defined as,

$$\min_{\mathbf{x},\mathbf{u}} \int_{t_0}^{t_f} (\mathbf{x}_d - \mathbf{x}(t))^2 + \xi \mathbf{u}(t)^2 dt \tag{5.3a}$$

$$\text{such that} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \forall t \in [t_0, t_f] \tag{5.3b}$$

$$\mathbf{1}^\top \mathbf{x}(t) = 1, \forall t \in [t_0, t_f] \tag{5.3c}$$

$$0 \le \mathbf{x}(t) \le 1, \forall t \in [t_0, t_f] \tag{5.3d}$$

$$0 \le \mathbf{u}(t) \le 1, \forall t \in [t_0, t_f] \tag{5.3e}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{5.3f}$$

with $\xi = 0.0001$ a choice of weight for the penalties on the control input. The dynamics function $\mathbf{f}$ represents the error compensated reduced-order advection-diffusion model introduced in Chapter 4. For the constraints, the states must be non-negative and sum to one since they are probability distributions, as shown in (3.11). This equality constraint is automatically applied as the dynamics are forward integrated. The input bounds are set between 0 and 1.

The error compensation scheme introduces discontinuities in the dynamics equation (5.3b), whereas convergence proofs for direct collocation [60, 61] typically requires smooth functions. To best fit the requirements of the optimal control framework, the following OCP is solved instead

$$\min_{\mathbf{x},\mathbf{u}} \int_{t_0}^{t_f} (\mathbf{x}_d - \mathbf{x}(t))^2 + \xi \mathbf{u}(t)^2 dt \tag{5.4a}$$

$$\text{such that} \quad \dot{\mathbf{x}}(t) = \mathbf{f}_{AD}(\mathbf{x}(t), \mathbf{u}(t), t), \forall t \in [t_0, t_f] \tag{5.4b}$$

$$0 \le \mathbf{u}(t) \le 1, \forall t \in [t_0, t_f] \tag{5.4c}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{5.4d}$$

$$0 \le \mathbf{x}(t_f) \le 1, \tag{5.4e}$$

with $\mathbf{f}_{AD}$ representing the reduced-order advection-diffusion model (4.33) without the error compensation. Because of this change, the state path constraint (5.3d) may not always be feasible hence is only enforced at the boundary. In addition, it is found that the dynamics equations (5.4b) would automatically preserve the condition (5.3c), in a manner similar to quaternions, where the rotational dynamics automatically preserve the condition requiring the norm of the quaternions states to equal to 1. Therefore (5.3c) is omitted from the OCP formulation as well. These changes can be viewed as additional (although practically negligible) model mismatches between the internal model of the system used in the optimal control setup, and the swarm robot dynamics in the simulation environment.

5.2.2 Derivative Information for Optimal Control Problem

As a derivative based NLP solver, the interior point method-based IPOPT requires the derivative information in the form of the gradient of the cost function, the Jacobian of the constraint equations, and the Hessian of the Lagrange function [58]. As the computation of derivatives, especially Hessian, can be expensive and time-consuming, many applications use Hessian approximations schemes such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) updates. For this advection-diffusion problem, the use of BFGS can often leads to a significant increase in the required number of NLP iterations, hence the exact Hessian is computed and supplied to the solver together with the gradient of the cost function and the Jacobian of the constraint equations.

First the reduced advection-diffusion equations affine in $\mathbf{x}(t)$ and $\mathbf{u}(t)$ are reintroduced, as presented previously in equations (4.33) and (4.28),

$$\dot{\mathbf{x}}(t) = \mathbf{E}^{-1}\left[\gamma\mathbf{A} - \left(\mathbf{u}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{G}\right]\mathbf{x}(t) \tag{5.5}$$

$$\dot{\mathbf{x}}(t) = \gamma\mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{x}(t)^{\top} \otimes \mathbf{I}_{N_x}\right)\mathbf{B}\mathbf{u}(t) \tag{5.6}$$

The Jacobian matrix is defined as,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} \cdots \frac{\partial \mathbf{f}}{\partial x_n} & \frac{\partial \mathbf{f}}{\partial u_1} \cdots \frac{\partial \mathbf{f}}{\partial u_n} \end{bmatrix} \tag{5.7}$$

in which the first partial derivative with respect to $\mathbf{x}$ (in $\mathbb{R}^{N_x \times N_x}$) can be found using equation (5.5):

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \mathbf{E}^{-1}\left[\gamma\mathbf{A} - (\mathbf{u}^{\top} \otimes \mathbf{I}_{N_x})\mathbf{G}\right] \tag{5.8}$$

and first partial derivative with respect to $\mathbf{u}$ (in $\mathbb{R}^{N_x \times N_u}$) can be found using equation (5.6):

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \mathbf{E}^{-1}(\mathbf{x}^{\top} \otimes \mathbf{I}_{N_x})\mathbf{B} \tag{5.9}$$

Secondly, the Hessian tensor is defined as:

$$
\mathbf{H} = \begin{bmatrix}
\frac{\partial \mathbf{f}}{\partial x_1^2} & \frac{\partial \mathbf{f}}{\partial x_1 \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_1 \partial x_n} & \frac{\partial \mathbf{f}}{\partial x_1 \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_1 \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_1 \partial u_n} \\
\frac{\partial \mathbf{f}}{\partial x_2 \partial x_1} & \frac{\partial \mathbf{f}}{\partial x_2^2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_2 \partial x_n} & \frac{\partial \mathbf{f}}{\partial x_2 \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_2 \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_2 \partial u_n} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial \mathbf{f}}{\partial x_n \partial x_1} & \frac{\partial \mathbf{f}}{\partial x_n \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n^2} & \frac{\partial \mathbf{f}}{\partial x_n \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_n \partial u_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n \partial u_n} \\
\frac{\partial \mathbf{f}}{\partial u_1 \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_1 \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_1 \partial x_n} & \frac{\partial \mathbf{f}}{\partial u_1^2} & \frac{\partial \mathbf{f}}{\partial u_1 \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_1 \partial u_n} \\
\frac{\partial \mathbf{f}}{\partial u_2 \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_2 \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_n \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_2 \partial u_1} & \frac{\partial \mathbf{f}}{\partial u_2^2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_2 \partial u_n} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial \mathbf{f}}{\partial u_n \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_n \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_n \partial x_n} & \frac{\partial \mathbf{f}}{\partial u_n \partial u_1} & \frac{\partial \mathbf{f}}{\partial u_n \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_n^2}
\end{bmatrix} \tag{5.10}
$$

The second partial derivative with respect to $\mathbf{x}$ (in $\mathbb{R}^{N_x \times N_x}$) $\mathbf{u}$ (in $\mathbb{R}^{N_x \times N_u}$) and using equations (5.8) and (5.9), respectively,

$$
\frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}^2} = \mathbf{0} \qquad \text{and} \qquad \frac{\partial^2 \mathbf{f}}{\partial \mathbf{u}^2} = \mathbf{0} \tag{5.11}
$$

The second partial derivative with respect to $\mathbf{x}$ and $\mathbf{u}$ (in $\mathbb{R}^{N_x \times N_u}$) using equation (5.9) is:

$$
\begin{aligned}
\frac{\partial^2 \mathbf{f}}{\partial \mathbf{u} \partial \mathbf{x}} &= \frac{\partial}{\partial \mathbf{u}} \left[\cancelto{0}{\gamma \mathbf{E}^{-1} \mathbf{A}} - \mathbf{E}^{-1} (\mathbf{u}^\top \otimes \mathbf{I}) \mathbf{G} \right] \\
&= -\mathbf{E}^{-1} \left[\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^\top \otimes \mathbf{I}) \right] \mathbf{G} \\
&= -\mathbf{E}^{-1} \left[\frac{\partial}{\partial \mathbf{u}} \left(\mathbf{u}^\top \right) \otimes \mathbf{I} \right] \mathbf{G} + \cancelto{0}{\mathbf{E}^{-1} \left[\mathbf{u}^\top \otimes \frac{\partial}{\partial \mathbf{u}} (\mathbf{I}) \right]} \mathbf{G} \\
&= -\mathbf{E}^{-1} \left[\mathbf{1}^\top \otimes \mathbf{I} \right] \mathbf{G}
\end{aligned} \tag{5.12}
$$

The resulting Hessian is,

$$
\mathbf{H} = \left[\begin{array}{cc}
\mathbf{0} & \begin{matrix}
\frac{\partial \mathbf{f}}{\partial x_1 \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_1 \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_1 \partial u_n} \\
\frac{\partial \mathbf{f}}{\partial x_2 \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_2 \partial u_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_2 \partial u_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial \mathbf{f}}{\partial x_n \partial u_1} & \frac{\partial \mathbf{f}}{\partial x_n \partial u_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n \partial u_n}
\end{matrix} \\
\begin{matrix}
\frac{\partial \mathbf{f}}{\partial u_1 \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_1 \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_1 \partial x_n} \\
\frac{\partial \mathbf{f}}{\partial u_2 \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_2 \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_n \partial x_1} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial \mathbf{f}}{\partial u_n \partial x_1} & \frac{\partial \mathbf{f}}{\partial u_n \partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial u_n \partial x_n}
\end{matrix} & \mathbf{0}
\end{array} \right] \tag{5.13}
$$

With this the derivatives of the model for the optimal control problem have been defined.

Table 5.1.: Methods used for ICLOCS2

Transcription Method	Direct collocation
Discretisation Method	Hermite-Simpson collocation
Jacobian Calculation	User-defined (sec. 5.2.2)
Hessian Approximation	User-defined (sec. 5.2.2)

5.2.3 Optimisation Environment

The toolbox used to solve the nonlinear optimal control problems is ICLOCS2 [57], which can be used for offline trajectory optimisation and real-time optimisation-based control methods such as model predictive control. Table 5.1 shows the settings and methods used within the ICLOCS2 toolbox.

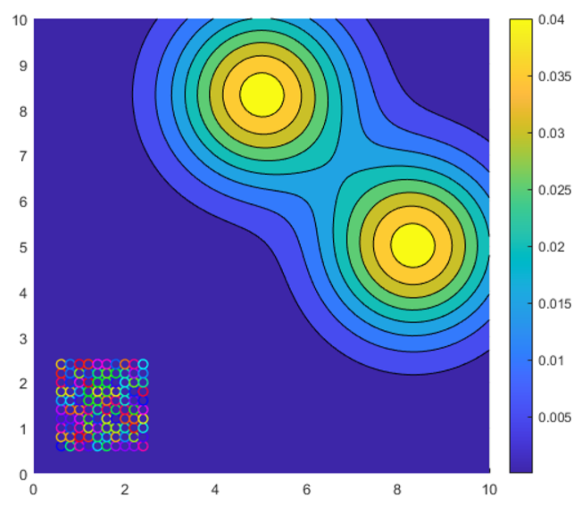
5.3. Simulation of Distributional Controller in Open-Loop and Closed-Loop

In this section, a simulation objective is shown in order to highlight the performance of the distributional controller in open-loop and closed-loop control. These simulations are shown under various parameters, including the number of basis functions, various oversampling parameters, differing peak magnitudes and the number of robots.

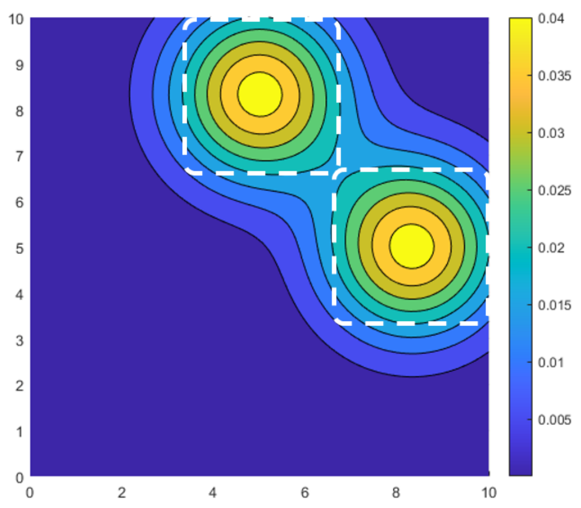
5.3.1 Simulation Objective

The objective of the controller is to direct a group of particles to two peaks in the 10 by 10 arena. There are 100 swarm robots initially arrayed in a 10 by 10 square grid formation centred at $(1.5, 1.5)$ and each has a body of radius of 0.1, as shown in 5.2(a). A swarm robot is set to have reached its goal if it reaches one of the two square areas centred centred at the middle-top $(5, 0.833)$ and right-middle $(8.333, 5)$. The middle-top square and right-middle square have corner coordinates, respectively, of $(3.333, 6.666)$—$(6.666, 6.666)$ —$(6.666, 10)$—$(3.333, 10)$ and $(6.666, 3.333)$—$(10, 3.333)$ —$(10, 6.666)$—$(6.666, 6.666)$, as shown in Figure 5.2(b).

The open-loop and closed-loop simulations are run for a total of 250 seconds with an ODE step of 0.5 seconds and use a diffusion coefficient of $\gamma = 0.1$. The prediction horizon of the



(a) Objective PDF with peaks at $(5, 0.833)$ and $(8.333, 5)$ and a robot swarm arrayed in grid formation with its centre at $(1.5, 1.5)$.



(b) The white square boundaries in the arena show the areas in which a swarm robot is marked to have to achieved its objective if it moves inside one of these squares.

Figure 5.2.: Objective PDF and robot swarm initial locations.

closed-loop controller is set to be 5 seconds.

5.3.2 Evolution of Particle Locations Over Time

To show how the particles are directed towards their objective peaks under open-loop and closed-loop controllers, time snapshots at 0, 25, 50, 100, 150 and 250 seconds are presented. These simulations are conducted with 36 basis functions as a 6 by 6 grid for N_x and N_u with the oversampling parameter set to $\rho = 1.5$. The time snapshots of the open-loop and closed-loop simulations are shown in Figures 5.3 and 5.4, respectively.

The Figures have a common pattern in which the particles are drawn towards the objective PDF with a leading front as seen at $t = 25$. A noticeable difference is that for the open-loop control, the particles are more narrowly positioned and fewer are left behind in their initial positions. The open-loop control solution appears to drag the particles towards the objective locations and then split them up into the two modes of the objective PDFs. Whereas, the closed-loop control scheme seems to split up the particles sooner towards the two mode objective PDF. The evolution of the PDF and the final particle positions also show that the open-loop control was more accurate than the closed-loop control scheme.

Figures 5.5 and 5.6 show the dynamic potential fields at times aligned to the particle locations.

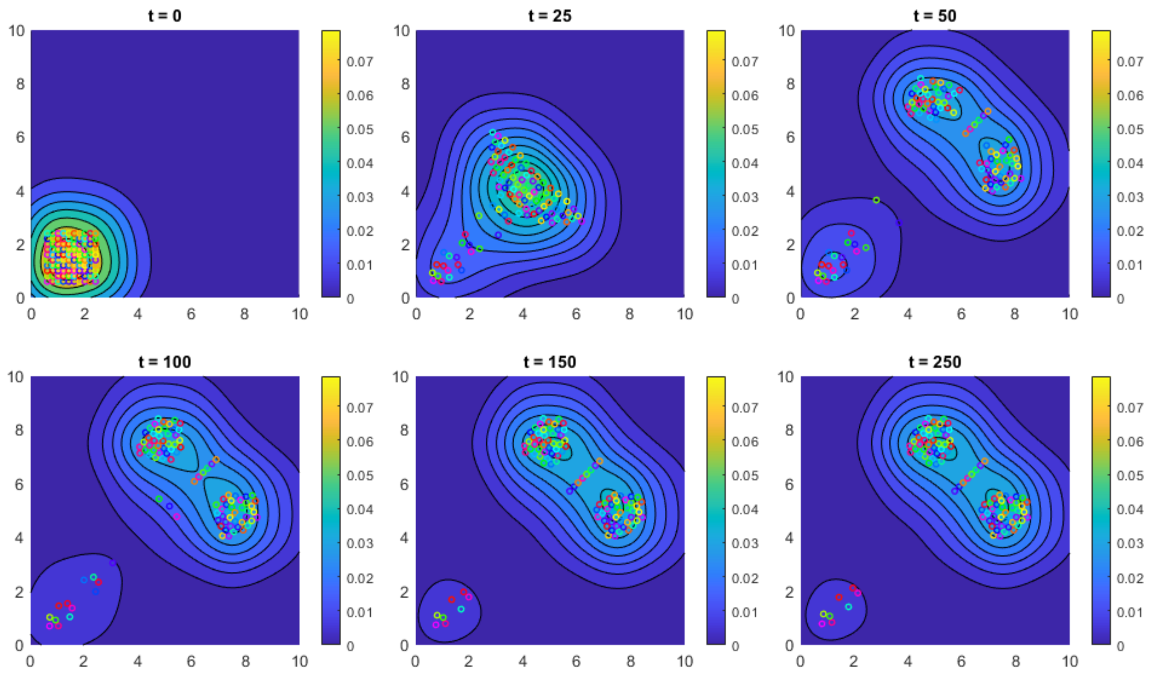


Figure 5.3.: Snapshots of the particle locations and modified swarm distributional states estimated PDFs for open-loop controller simulation with $N_x = 36$ (6 by 6 grid), $N_u = 36$ (6 by 6 grid), $\rho = 1.5$, $\gamma = 0.1$ and 100 particles at specific times.

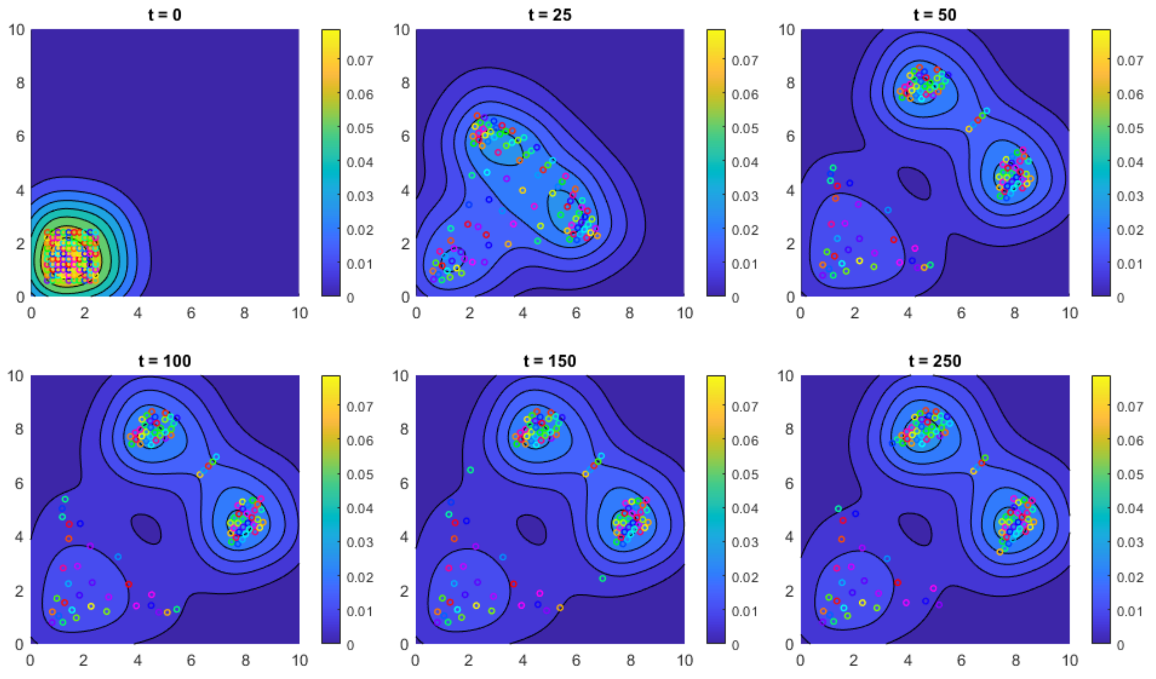


Figure 5.4.: Snapshots of the particle locations and modified swarm distributional states estimated PDFs for closed-loop controller simulation with $N_x = 36$ (6 by 6 grid), $N_u = 36$ (6 by 6 grid), $\rho = 1.5$, $\gamma = 0.1$ and 100 particles at specific times.

The potential field of the closed-loop control scheme as seen at $t = 25$, has gradients that are more likely to separate the particles towards the modes of the objective PDF than the open-loop control scheme.

Figure 5.7 shows the time series of the input energy for the open-loop and closed-loop simulations. The total input energy is lower for the closed-loop than the open-loop case. This also supports the observation that less overall control effort will be required if the particles are split up sooner.

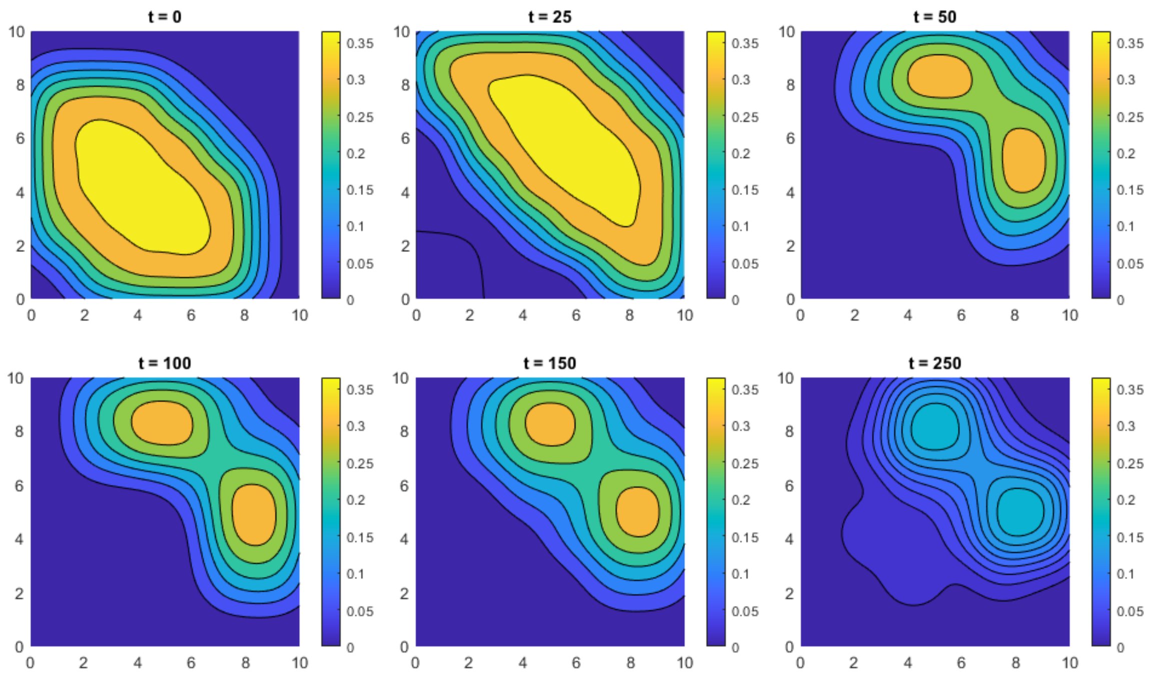


Figure 5.5.: Snapshots of the potential input field for open-loop control at specific times.

However, the KLD value at the end of the simulation, as shown in Figure 5.8 is lower for the open-loop simulation and consequently, more robots have reached their objectives at the arena peaks for the open-loop case, as shown in Figure 5.9. In these simulations, it shows that although the open-loop case has spent more input energy than the closed-loop case, more robots in the open-loop case have reached their objectives.

It should also be noted that having a constant input at the objective peaks fail to move the particles from their initial location, as shown in Figure 5.10. This is just to demonstrate that dynamic control is required to achieve some desired swarm distribution that is different from their original configuration.

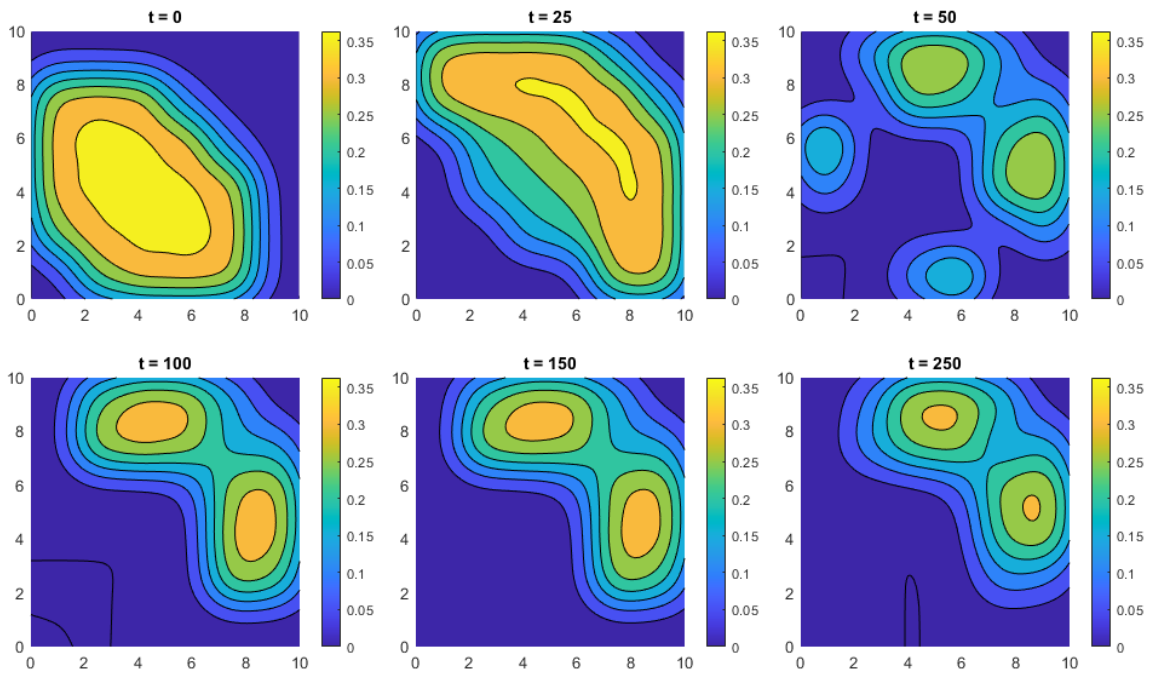


Figure 5.6.: Snapshots of the potential input field for closed-loop control at specific times.

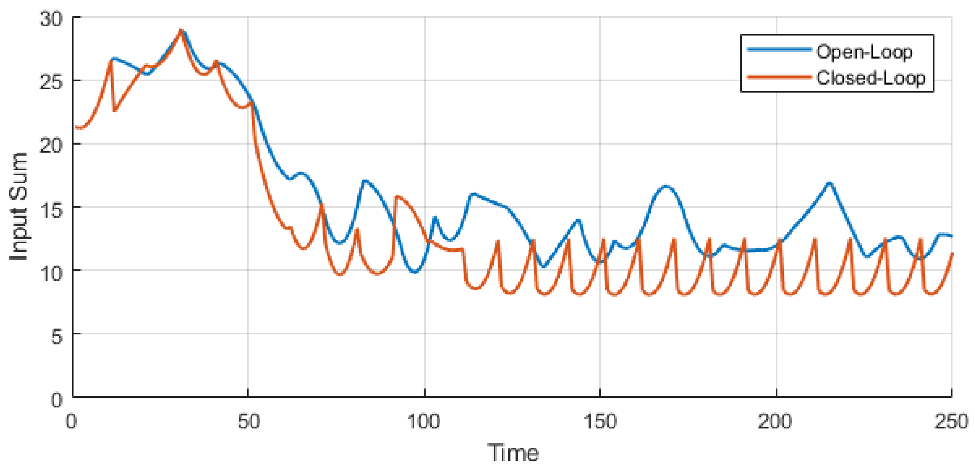


Figure 5.7.: Time series of input energy for open-loop and closed-loop simulations.

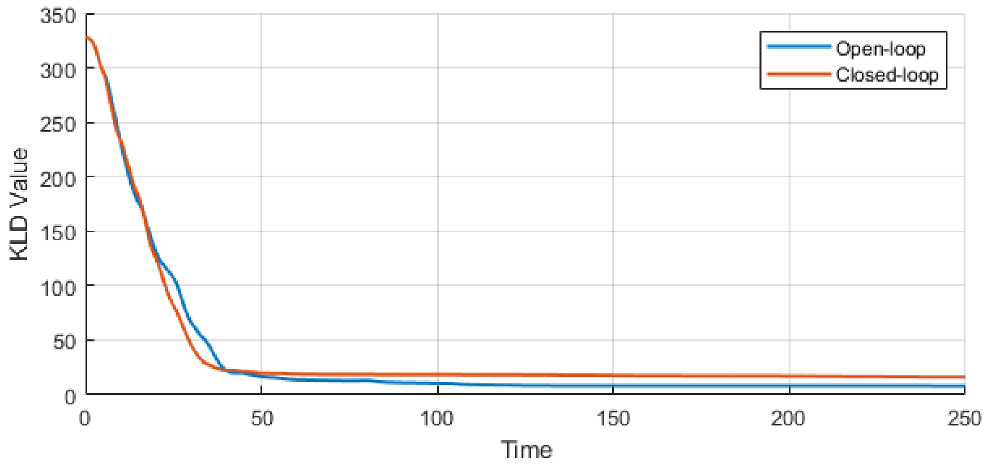


Figure 5.8.: Time series of KLD values for open-loop and closed-loop simulations.

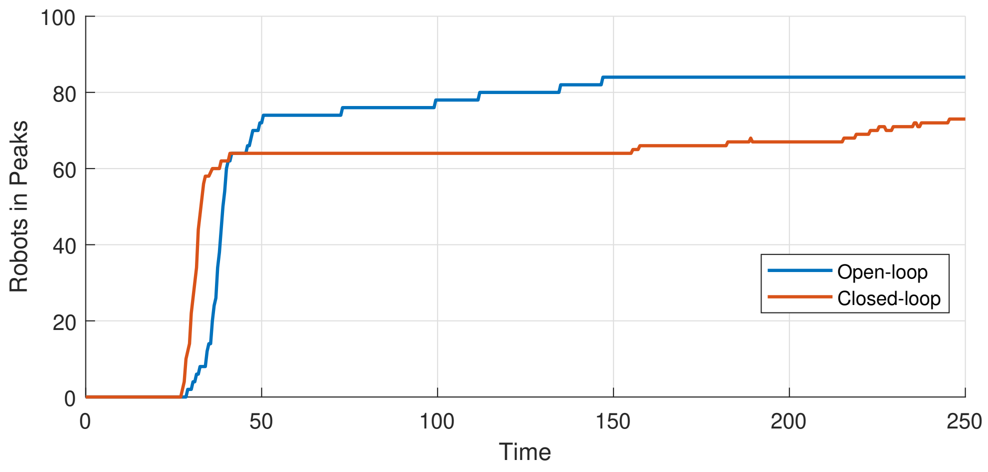


Figure 5.9.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks.

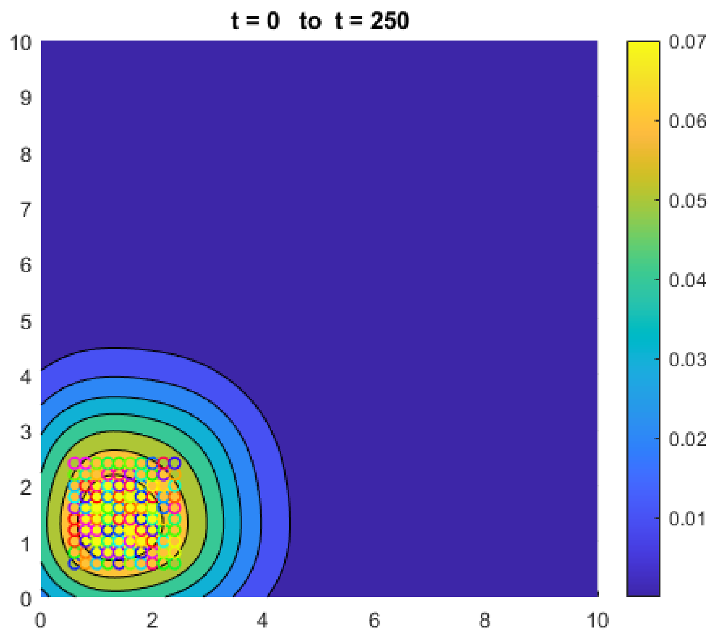


Figure 5.10.: Particle locations and modified swarm distributional states estimated PDFs for static input at the objectives' peaks. The constant input fails to move the particles from their initial location.

5.3.3 Performance Based On Number of Basis Functions

For high accuracy, the model representations and the control field should have a fine resolution which will result in a large number of basis functions, and hence a large dimension for both the particle swarm state and the coefficients of the dynamic potential field. However, the increased dimensions result in vastly increased computational time for simulations and controller calculations. Hence, a trade-off between accuracy and computation time has to be made in practice.

In these simulations, how the number of basis functions used affect the performances of the open-loop and closed-loop controllers are presented. Figures 5.11 and 5.12 show the particle locations and the modified swarm distributional states estimated PDF in open-loop and closed-loop simulations, respectively, under a low and high number of basis functions. These simulations are conducted with the oversampling parameter $\rho = 1.5$. The time series of the KLD values for varying numbers of basis function for open-loop and closed-loop simulations are shown in Figure 5.13 and a bar chart showing how many of the 100 robots have reached their objective at the arena peaks for the open-loop and closed-loop cases are shown in Figure 5.14.

From these figures, it is shown that the performance differences for the open-loop and closed-loop controllers are notable when using a limited number of basis functions are used. In the 3 by 3 case, none of the robots in the open-loop case reached their objective while 38 robots have reached their objective in the closed-loop case. This is to be expected since the closed-loop control scheme uses feedback to compensate for the model inaccuracies. As the number of basis functions chosen increase, the performances of both the open-loop and the closed-loop control improves. The KLD figures also show a pattern of consistent improvement not only in the speed of convergence but also in the final value, indicating that the desired objective PDF is reached faster and with more accuracy.

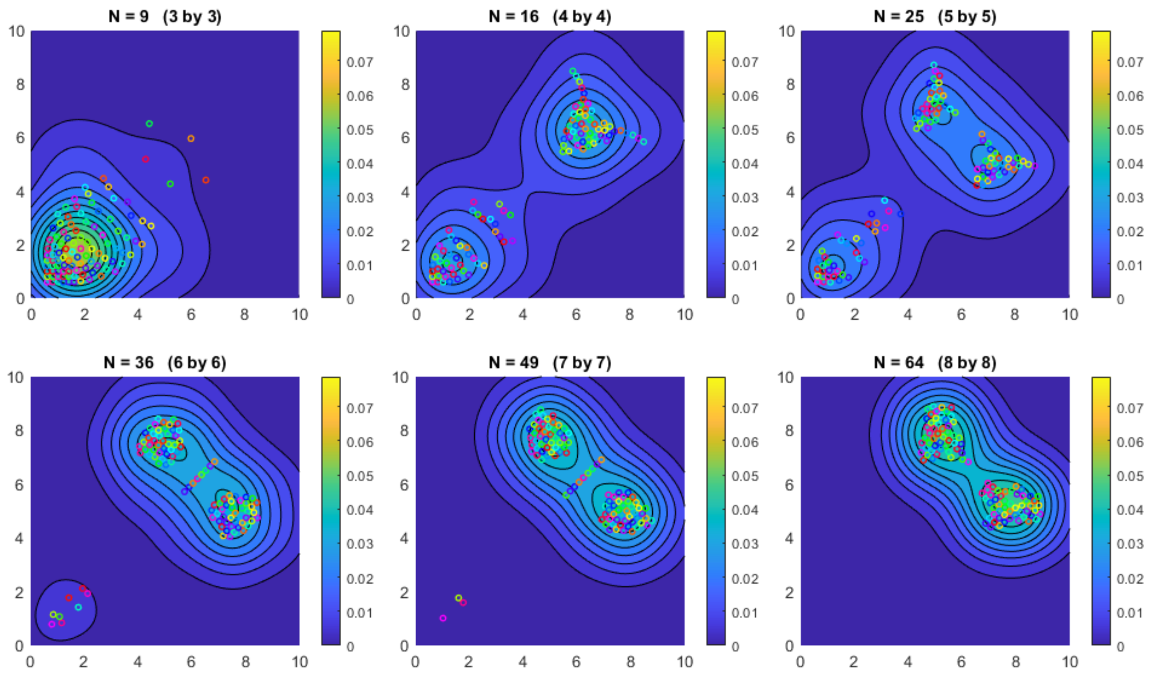


Figure 5.11.: Particle locations and modified swarm distributional states estimated PDFs for open-loop controller simulations with varying numbers of basis functions at $t = 250$.

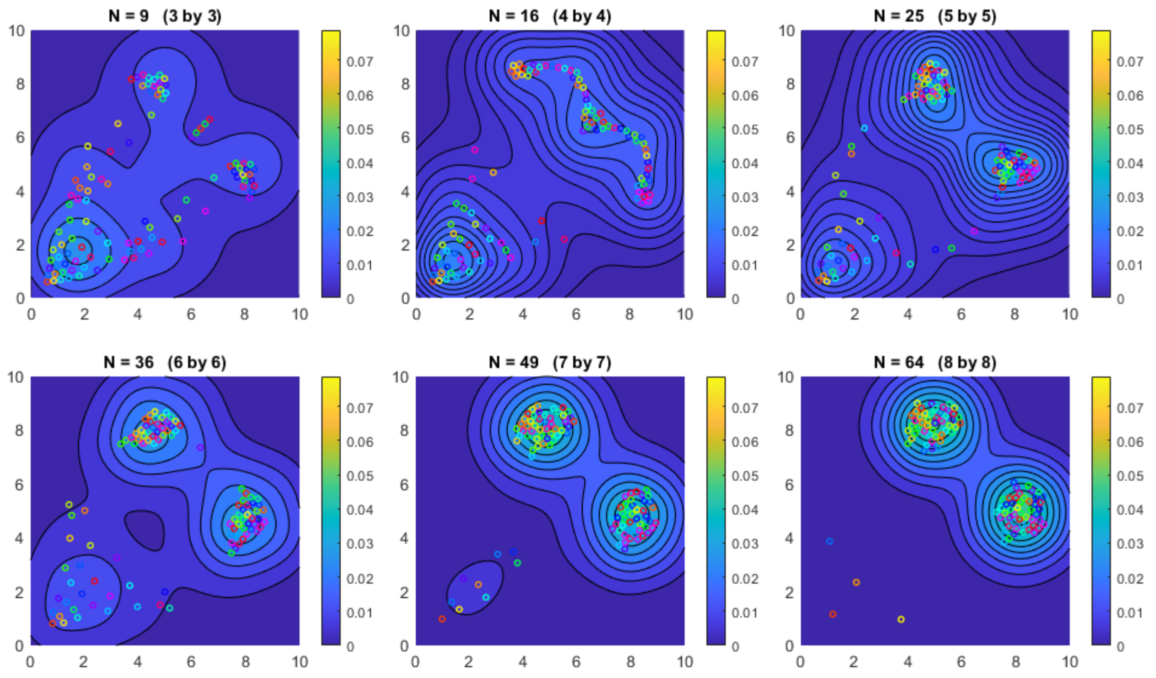


Figure 5.12.: Particle locations and modified swarm distributional states estimated PDFs for closed-loop controller simulations with varying numbers of basis functions at $t = 250$.

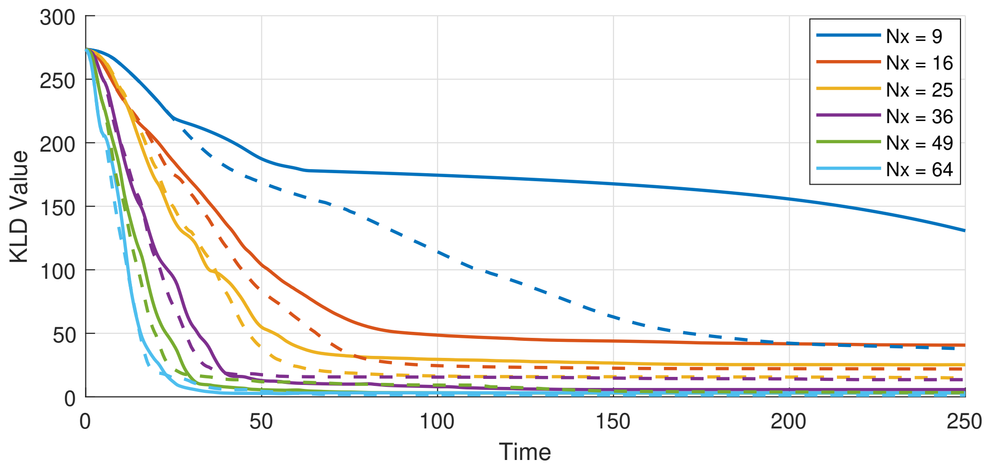


Figure 5.13.: Time series of KLD values for open-loop and closed-loop simulations under varying numbers of basis functions. The solid lines and dashed lines represent the open-loop and closed-loop simulations, respectively.

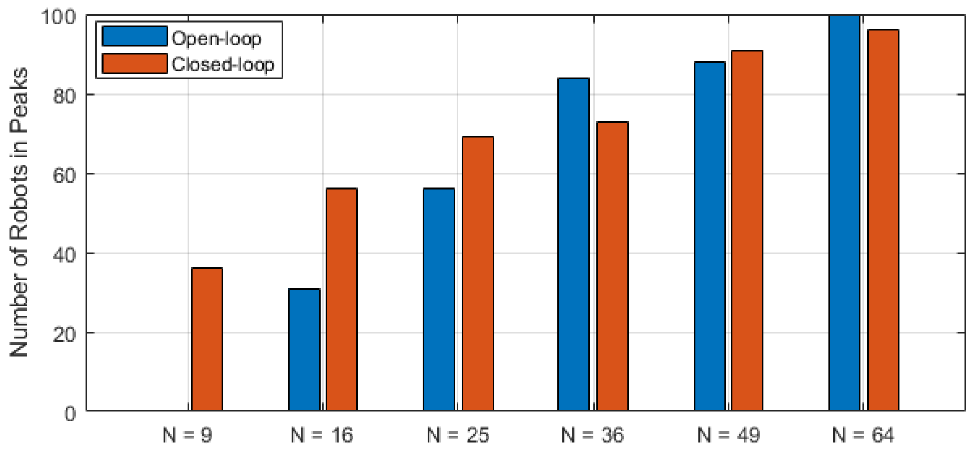


Figure 5.14.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks under varying numbers of basis functions. The N_x and N_u have the same number of basis functions.

5.3.4 Performance Based On Varying Numbers of State Basis Functions

Understanding the difference in model accuracy alone without the variations in the degree of controllability is investigated next. This is achieved by varying only the number of particle swarm states but the control potential field basis functions are kept unchanged.

In these simulations, the effect of the state basis functions are presented. Figure 5.15 shows the particle locations and the modified swarm distributional states estimated PDF in open-loop and closed-loop simulations, respectively, under a low and high number of state basis functions. These simulations are conducted with the oversampling parameter $\rho = 1.5$. The time series of the KLD values for varying numbers of state basis function for open-loop and closed-loop simulations are shown in Figure 5.16 and a bar chart showing how many of the 100 robots have reached their objective at the arena peaks for the open-loop and closed-loop cases are shown in Figure 5.17. This is consistent with the fact that improving model accuracy improves the controller performance.

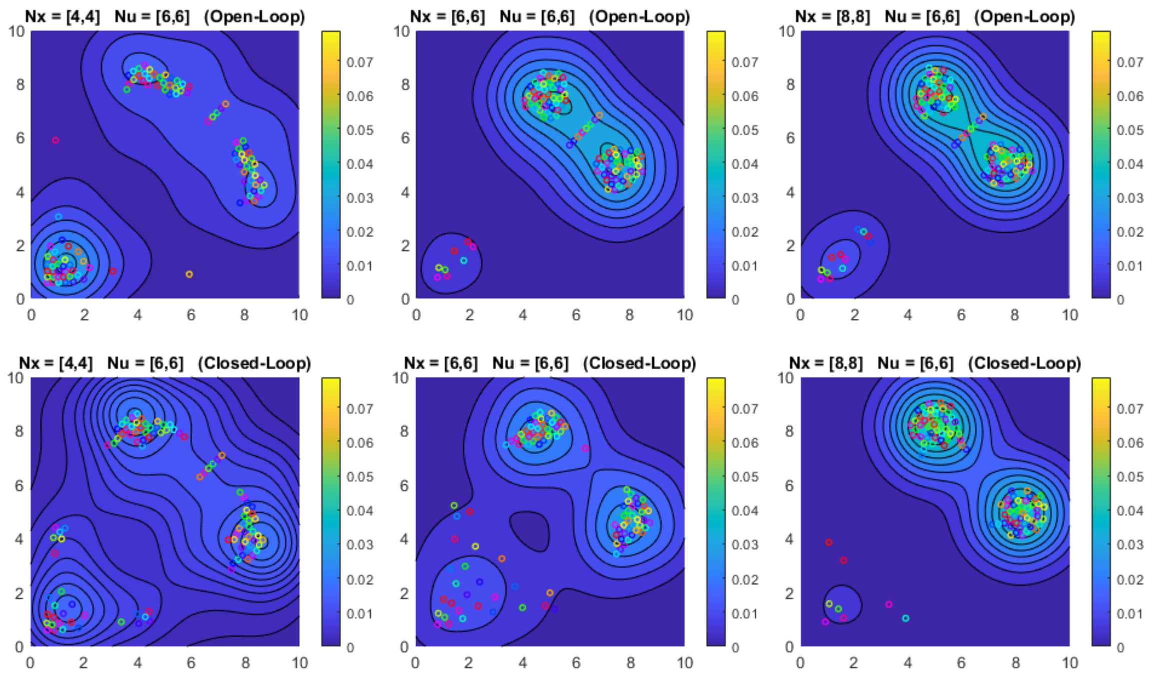


Figure 5.15.: Particle locations and modified swarm distributional states estimated PDFs for open-loop and closed-loop controller simulations with varying number of state basis functions at $t = 250$.

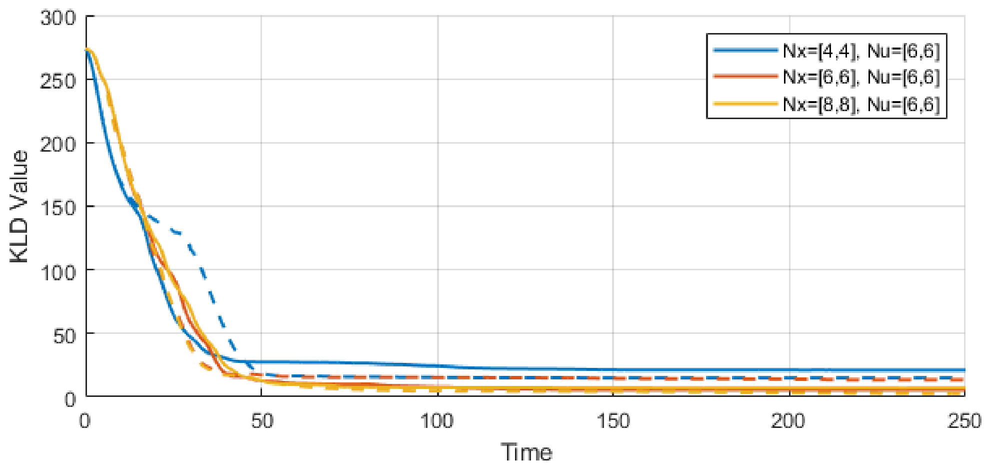


Figure 5.16.: Time series of KLD values for open-loop and closed-loop simulations with varying number of state basis functions. The solid lines and dashed lines represent the open-loop and closed-loop simulations, respectively.

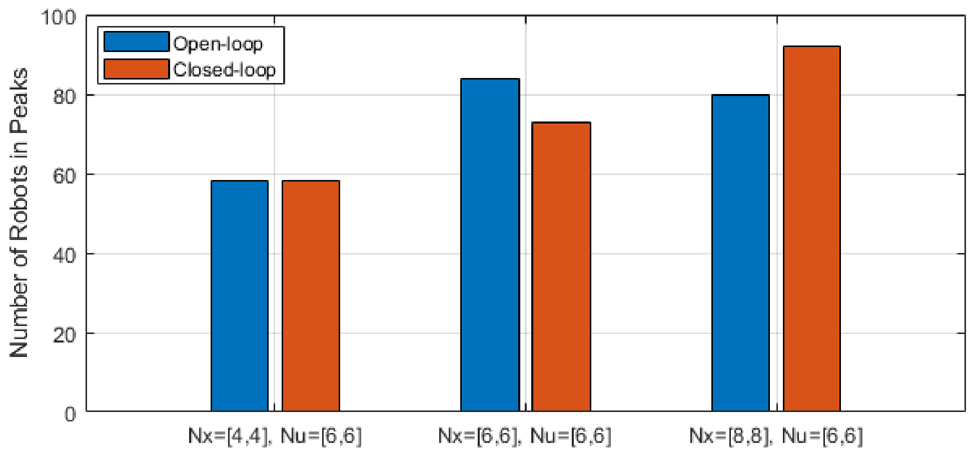


Figure 5.17.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks with varying number of state basis functions.

5.3.5 Performance Based On Varying Number of Input Basis Functions

Understanding the impact of better controllability with fixed model accuracy is also useful for the design of swarm PDF based control. This is achieved by keeping the particle swarm state dimension to be fixed while the control potential field basis functions are changed.

In these simulations, the effect of the state basis functions are presented. Figure 5.18 shows the particle locations and the modified swarm distributional states estimated PDF in open-loop and closed-loop simulations, respectively, under a low and high number of input basis functions. These simulations are conducted with the oversampling parameter $\rho = 1.5$. The time series of the KLD values for varying numbers of input basis function for open-loop and closed-loop simulations are shown in Figure 5.19 and a bar chart showing how many of the 100 robots have reached their objective at the arena peaks for the open-loop and closed-loop cases are shown in Figure 5.20.

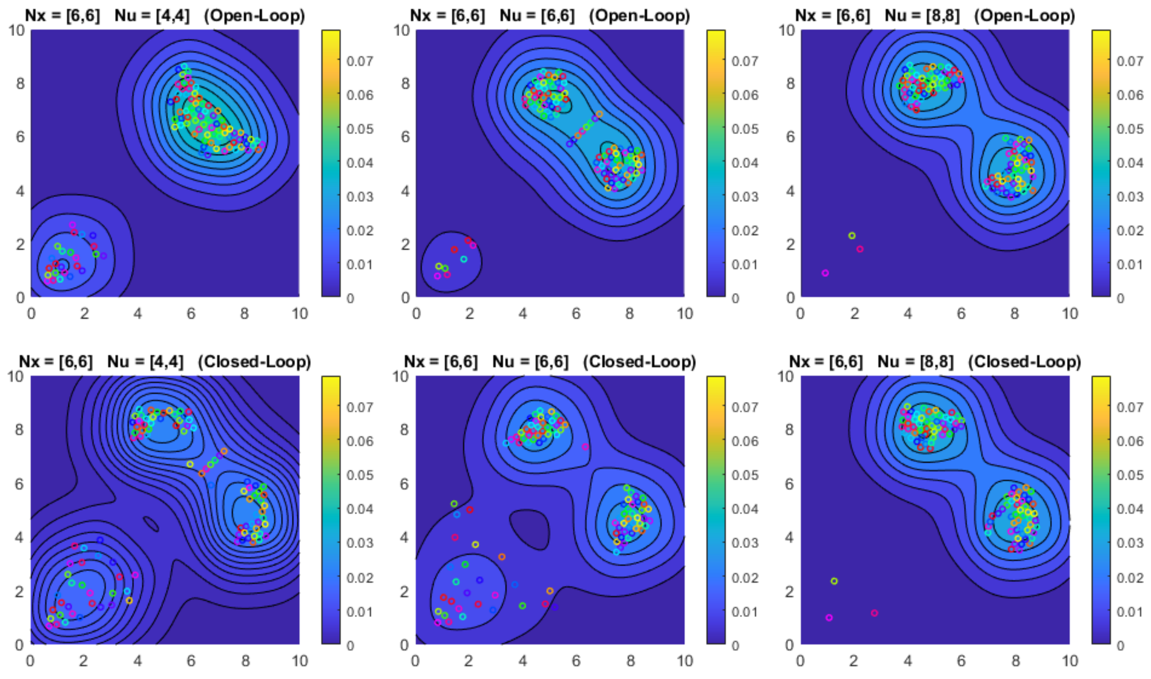


Figure 5.18.: Particle locations and modified swarm distributional states estimated PDFs for open-loop and closed-loop controller simulations with varying number of input basis functions at $t = 250$.

What is interesting in this analysis is that with improved controllability, the controllers are able to move the swarm quickly to the desired objective PDF as seen by the different slopes by which the KLD decreases in Figure 5.19. As expected, the improved controllability also improves the final goal achievement.

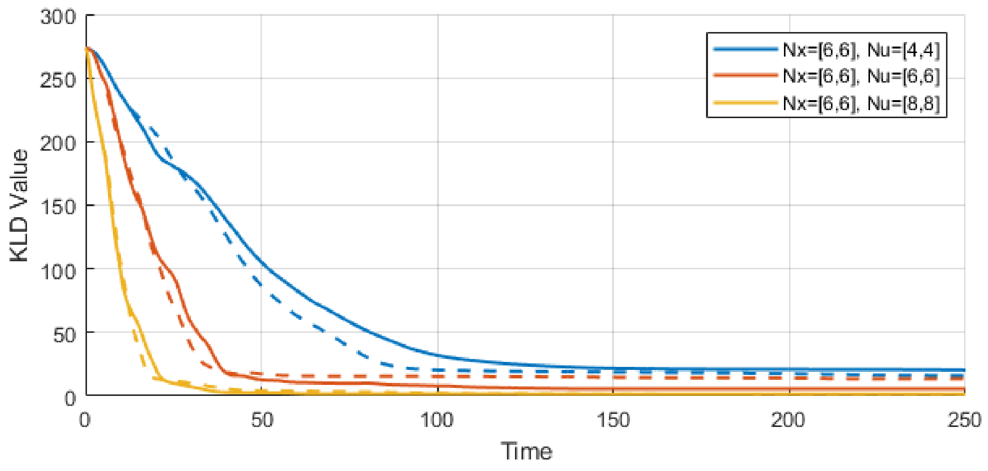


Figure 5.19.: Time series of KLD values for open-loop and closed-loop simulations with varying number of input basis functions. The solid lines and dashed lines represent the open-loop and closed-loop simulations, respectively.

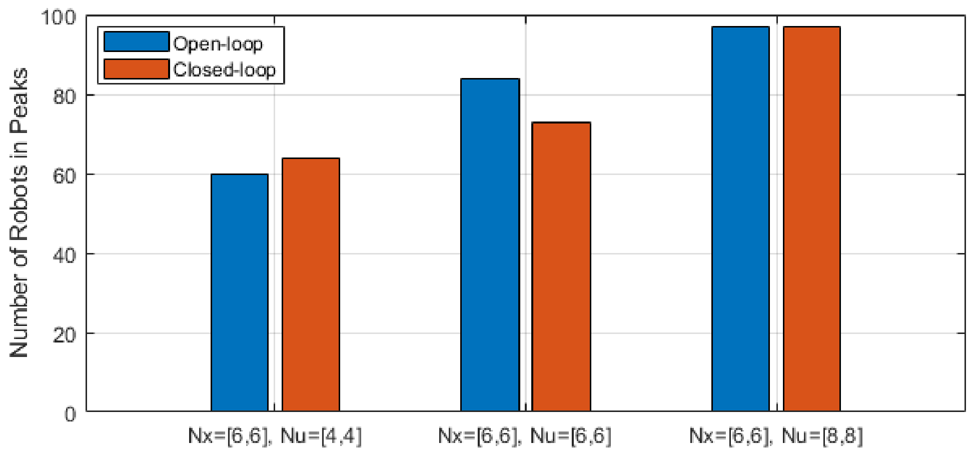


Figure 5.20.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks with varying number of input basis functions.

5.3.6 Performance Based on Oversampling Parameter Values

As described in Chapter 3, the oversampling parameter ρ determines the amount of overlap between the basis functions. For a fixed number of basis functions arrayed on a grid, a low ρ value has narrower basis functions and little overlap with neighbouring basis functions and high ρ value has wider basis functions and larger overlap with neighbouring basis functions. This will have an impact on the final model accuracy as well as the controllability of the dynamic potential field input, and hence on the performance of any model-based control scheme.

In order to show how the oversampling parameter ρ impacts the performances of the open-loop and closed-loop controllers, the open-loop and closed-loop simulations using varying ρ values are shown in Figure 5.21. These simulations are conducted with 36 basis functions as a 6 by 6 grid for N_x and N_u. Figure 5.22 shows the KLD values over the simulation under varying rho values for open-loop and closed-loop cases and Figure 5.23 shows how many of the 100 robots have reached their objective.

It can be seen that for the low ρ value of 0.5 the open-loop controller is not able to bring the robots to the objectives, while the feedback of the closed-loop controller is able to. However, in the case of the high ρ value of 2.5, even if the open-loop simulation is able to bring the robots to the top-right corner of the arena, it is not able to separate them into robot cluster into the two peaks at the middle-top and right-middle.

When $\rho = 0.5$, with little overlap between basis functions and hence its narrower shape, the ability to move particles from nearby locations is very limited. This is what leads to the inability of the particles to disperse from their initial configuration. The closed-loop control scheme shows a little more improvement but could only come from applying large potential field at specific locations, having observed where to do so. At the other end of high $\rho = 2.5$, the model accuracy suffers as well as the control becoming coarser. This points to a difficulty in being able to move the particles to the separate modes as seen in the open-loop control scheme. Again, the closed-loop control uses the feedback information to identify the required control to achieve its task.

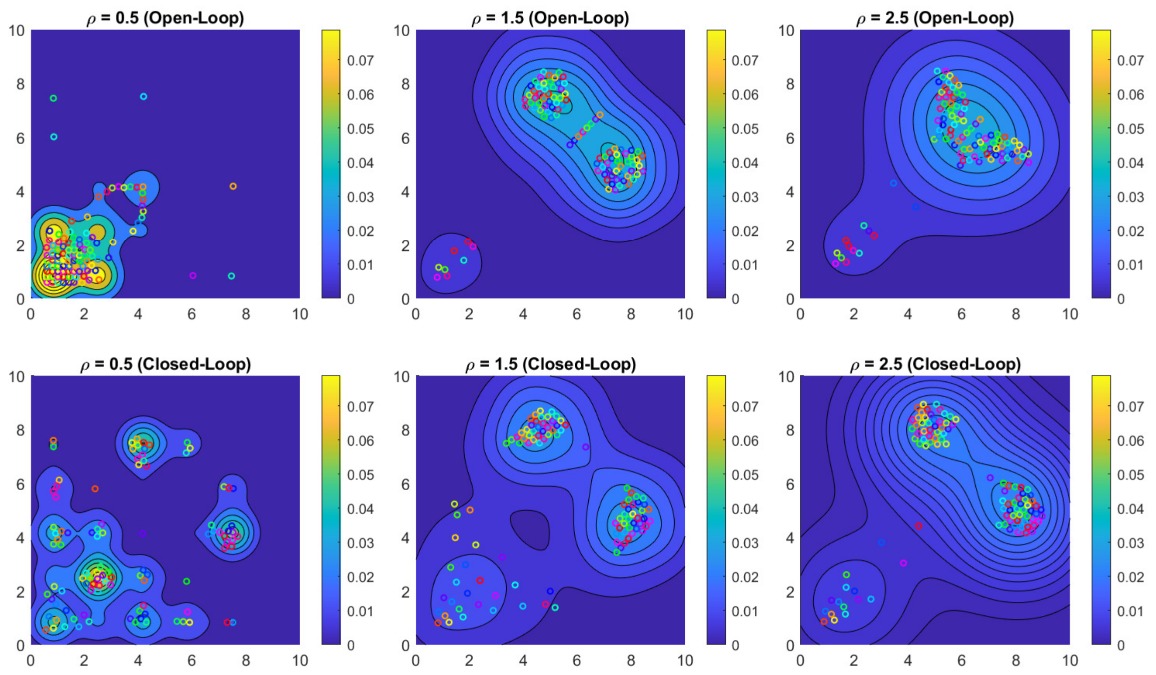


Figure 5.21.: Particle locations and modified swarm distributional states estimated PDFs for open-loop and closed-loop controller simulations with varying ρ values at $t = 250$.

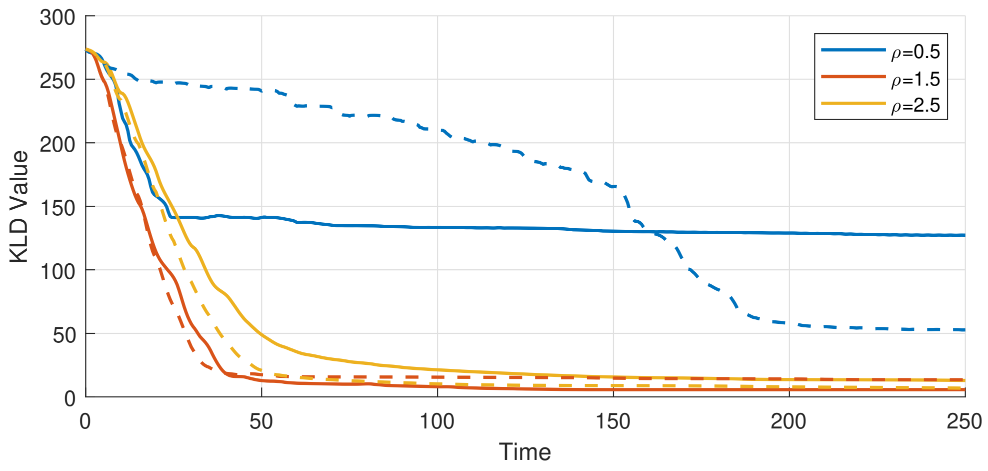


Figure 5.22.: Time series of KLD values for open-loop and closed-loop simulations under varying ρ values. The solid lines and dashed lines represent the open-loop and closed-loop simulations, respectively.

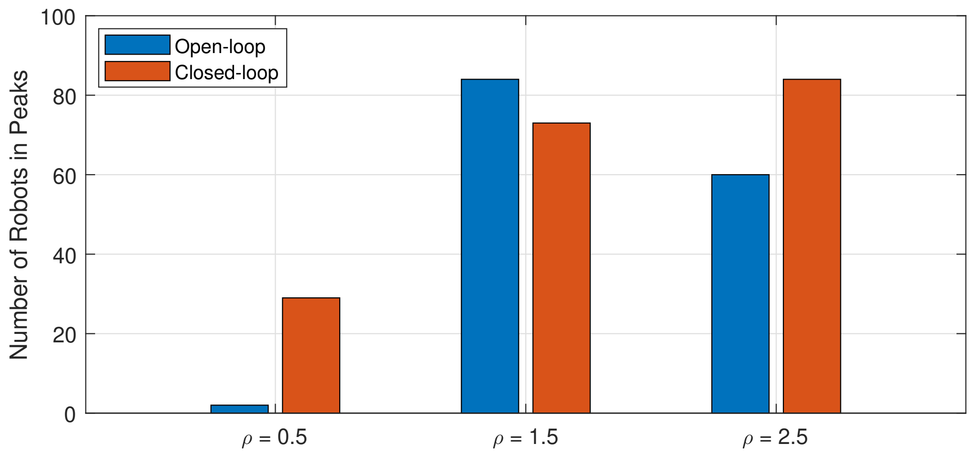


Figure 5.23.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks under varying ρ values.

5.3.7 Performance Based on Varying Peak Magnitude Ratios

Thus far, the analysis has restricted itself to a desired objective PDF with some symmetry, in terms requiring same number of robots to be located under the two modes. This was seen in the dynamic control potential fields being determined by the controllers to have symmetric patterns. Swarm robot tasks may require uneven distribution of robots to be placed in different locations and hence an investigation to understand the effect of such asymmetry is required.

In the cases where the objective is to have a different ratio of swarm robots in each of the peaks, the magnitudes of the peaks have been set to different percentage value splits. Figure 5.24 shows three cases where the objectives are to have 50%-50%, 60%-40% and 70%-30% splits of the number of robots at the two peaks. These simulations are conducted with 36 basis functions as a 6 by 6 grid for N_x and N_u with the oversampling parameter $\rho = 1.5$. Table 5.2 shows the number of the robots that have reached each of the two peaks and Table 5.3 shows the percentage of the robots at each peak of the robots that have reached one of the two peaks. It can be seen that for the 60%-40% and 70%-30% split cases the closed-loop controller is more effective at splitting at the desired ratio.

This difference in characteristics of the open-loop and closed-loop control performance may be attributed to the fact that the former appeared to favour moving the particles as a front first before attempting a split. Without having finer control, this task can be challenging and made more difficult with reduced model accuracy. Closed-loop control on the other uses the feedback information effectively to compensate for the model accuracy deficiency, even if the control is coarser.

Table 5.2.: Number of robots that have reached their objective at each peak.

	50/50		60/40		70/30	
Open-Loop	42	42	59	14	65	0
Closed-Loop	36	37	47	27	55	18

Table 5.3.: Percentage of particles distribution of the robots at each of the objectives' peaks.

	50/50		60/40		70/30	
Open-Loop	50%	50%	80.8%	19.2%	100%	0%
Closed-Loop	50.7%	49.3%	63.5%	36.5%	75.3%	24.7%

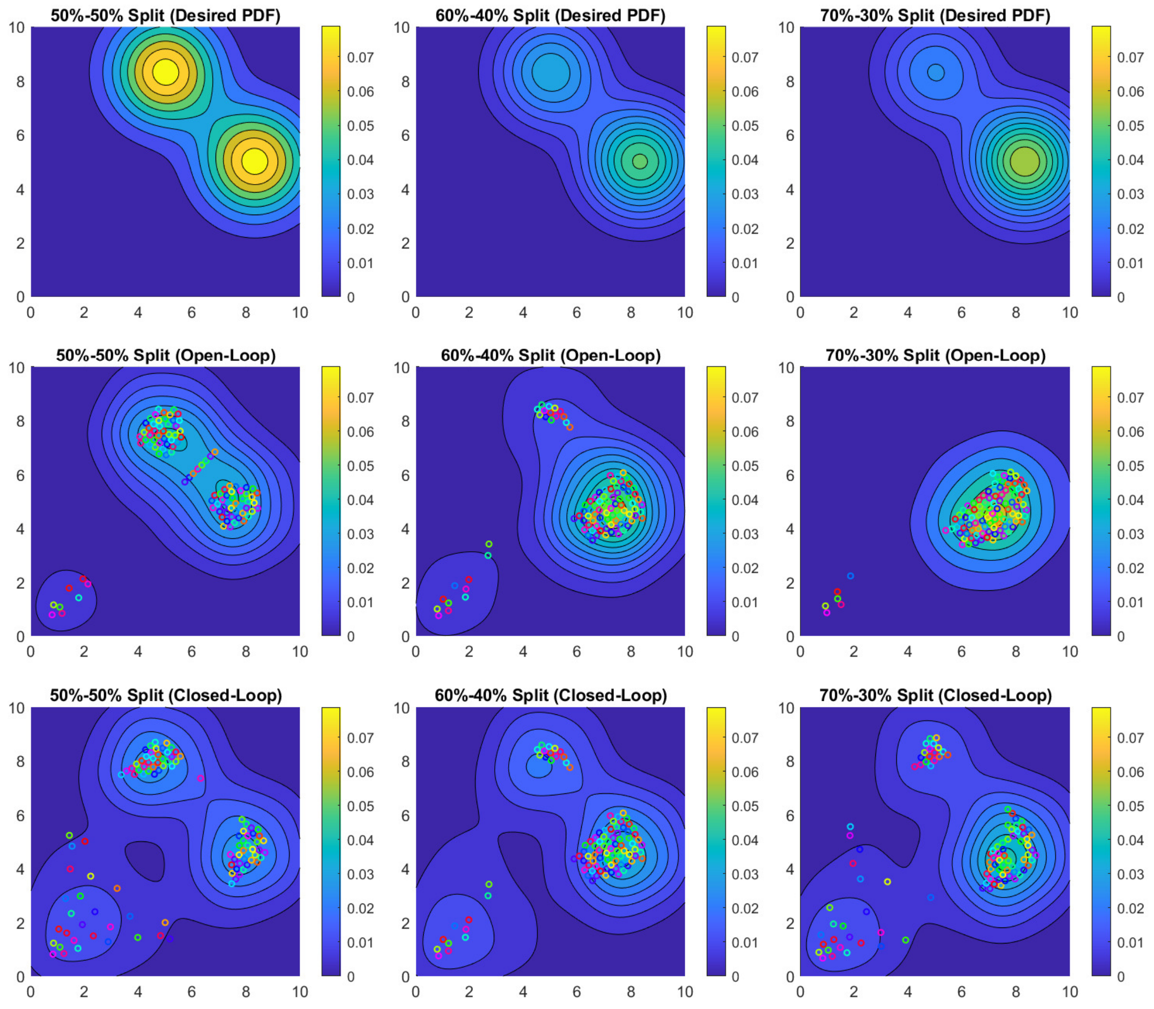


Figure 5.24.: Particle locations and modified swarm distributional states estimated PDFs for open-loop and closed-loop controller simulations with varying peak magnitudes at $t = 250$. The top row shows the objectives' desired PDF, the middle row shows the open-loop simulations, and the bottom row shows the closed-loop simulations.

5.3.8 Performance Based on Number of Robots

Lastly, with the focus of the thesis being on the development of modelling and control of a swarm of large number of robots, understanding when such approximations are sufficient is important. If the number of robots are small, then representing these as PDFs may lead to model inaccuracies and the controller will then find it challenging to achieve its desired objective.

Simulations are shown to see how the performance vary in the open-loop and closed-loop cases for a varying numbers of robots. Figure 5.25 show the particle locations and modified swarm distributional states estimated PDFs for 49, 100 and 225 particles. These simulations are conducted with 36 basis functions as a 6 by 6 grid for N_x and N_u with the oversampling parameter $\rho = 1.5$. For the 225 particle case, it can be seen that the controller is not able to direct the particles at the very bottom-left of the robot cluster. This would be the case where having a finer resolution grid (i.e. having a larger number of basis functions) would be beneficial in order to have more controllability.

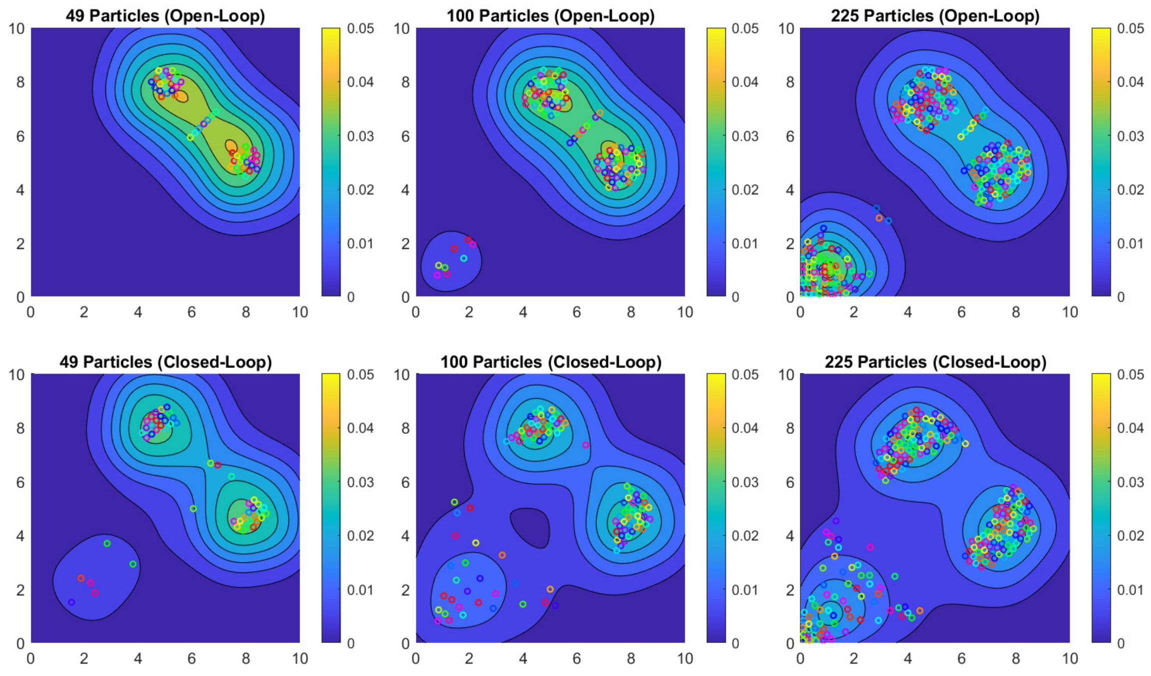


Figure 5.25.: Particle locations and modified swarm distributional states estimated PDFs for open-loop and closed-loop controller simulations with varying numbers of robots at $t = 250$.

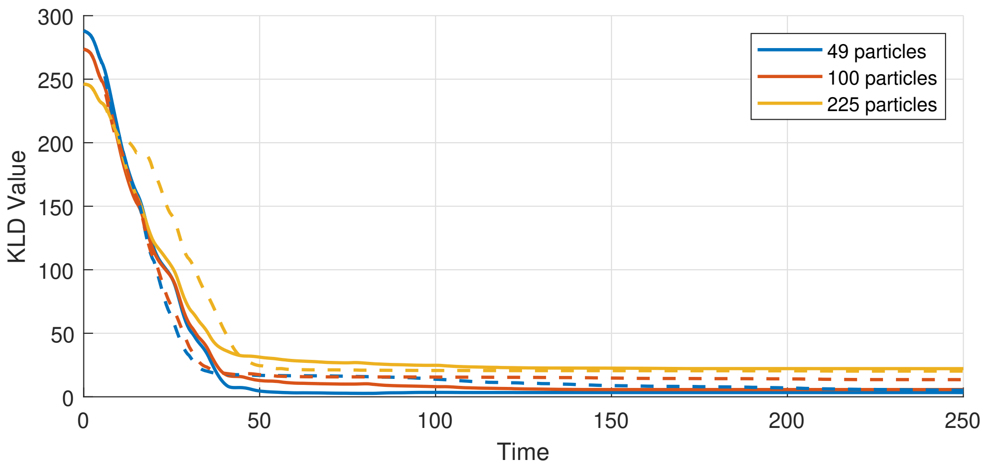


Figure 5.26.: Time series of KLD values for open-loop and closed-loop simulations with varying numbers of robots. The solid lines and dashed lines represent the open-loop and closed-loop simulations, respectively.

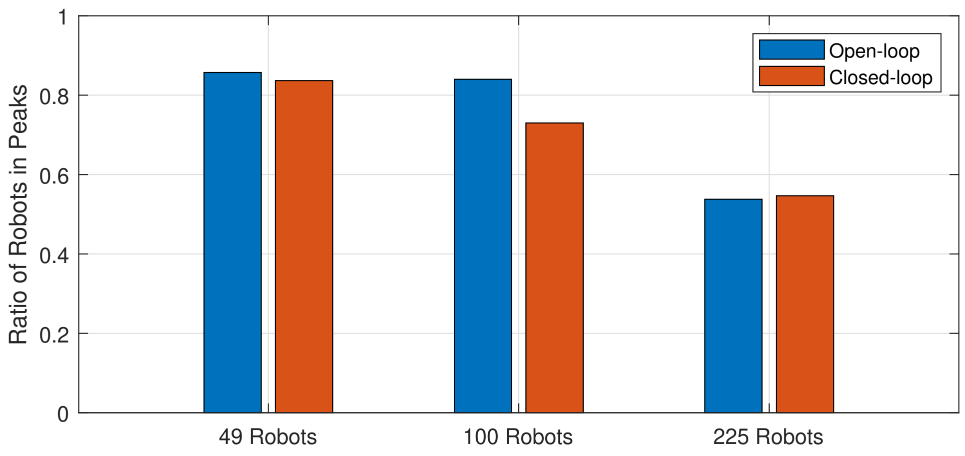


Figure 5.27.: Time series showing how many of the 100 robots have reached their objectives at the arena peaks with varying numbers of robots.

5.4. Discussion

To control the behaviour of the swarm particles, in which the particles have limited computational or actuation capabilities, a distributional control framework was developed in which control the trajectories of the particle swarm for open-loop and closed-loop control. In order to do so, the optimisation toolbox, ICLOCS2, has been used on the non-error compensated reduced-order model from the previous chapter, as the solver takes into account the required equality and inequality constraints of the model, in that states must sum to one and be non-negative, separately. Additionally, in order to reduce computational time for simulations and controller calculations, it required us to derive apply the Jacobian and the Hessian tensor to the solver.

A common theme in the simulations performed in this chapter is that all the cases would result in better performance with a finer or high-resolution grid. However, this was infeasible due to computer hardware limitations to run the simulations and much higher computational time needed. A possible remedy for this is to develop and implement a sparse representation of the reduced-order model, as most of the states remain close to zero for large amounts of time.

In the cases with low resolution coarse grids, low oversampling parameter values (higher overlap between basis functions) and differing desired peak magnitudes, closed-loop simulations better than their open-loop simulations as these simulations benefit from the feedback information to compensate for the model's accuracy deficiency. Also, the particles in the closed-loop simulations split off sooner than in the open-loop simulations, supporting the observation that less control effort is required, as seen in Figure 5.7. Additionally, closed-loop control helps in compensating the limitations from the particle agent model. When a particle collides with another particle, the particle's velocity becomes zero and experiences no momentum.

5.5. Summary

This chapter has presented the development of a distributional control framework to control the trajectories of the particle swarm for open-loop and closed-loop control. The optimal control problem framework was introduced based on the reduced-order model and the required constraints introduced in Chapter 4 and simulations were shown for varying parameters, such as the varying resolution of the basis function grid, varying the number of basis functions used for states and inputs, the effect of different oversampling parameters, adjusting the magnitude of the objectives' peaks for various split ratios, and the effect of the performance for simulating various particle numbers.

This study has showed that increasing the model accuracy through the dimension of particle swarm state can improve the performance of the controller. Similarly, improving the controllability through the increase of the dynamic potential field basis functions will also improve the system's results. Further, choosing a suitable set of parameters for the basis functions was shown to affect both the model accuracy and controllability and hence the overall performance of the system. Finally, investigations of asymmetric desired objective PDF also showed challenges for open-loop control scheme. While there was no consistently improved performance with closed-loop control, it showed better results than the open-loop control particularly under challenging tasks or design choices.

6 Conclusion

This chapter concludes the thesis by summarising the main points presented in the preceding chapters and proposes potential directions for future work.

6.1. Summary

The aim of this thesis was to address some of the challenges for modelling and controlling a large number of particle swarm robots. The inspiration of this research was the control of microrobots, in which because of their tiny hardware space, they have limited computational capabilities and are reliant on external magnetic field, as a potential input field, for movement to fulfil their objectives. Thus, they rely on a central distributional controller in which, it is not able to direct each agent explicitly with control actions, but through controlling the external dynamic potential field.

To first develop a model of the particle swarm movements which are influenced by an external dynamic potential field, a simple particle-based representation of the swarm robot is presented. Then, a description of the probability distribution to represent the behaviour of a large scale swarm of robots is defined and a representation of the potential field, which are parametrised through Gaussian basis functions, is introduced. Since the swarm robots are considered in a continuum model representation, its probability density is represented by weighted basis functions. This estimation, when presented with information about robot positions, is performed using the modified SDSE algorithm which yields the swarm distributional states.

To represent the swarm robots as a probability distribution, a continuum representation of the advection-diffusion model is first considered. Since this model is infinite-dimensional, a reduced-order model of the advection-diffusion is derived using the Galerkin approach and parametrised with Gaussian functions. The potential field input is also similarly parametrised with Gaussian basis functions, which with these both, allow for model-based control designs to manipulate the swarm behaviour. However, this reduced-order model resulted in not satisfying the required constraints. Thus, an analysis of errors and eigenvalues was performed and led to a further derivation of an error compensated reduced-order model which satisfied the required constraints.

Given the reduced-order advection-diffusion model, a distributional controller was developed to control the trajectories of the robot swarm for open-loop and closed-loop control. The optimal control problem formulation was presented using a derivative-based NLP solver IPOPT and the ICLOCS2 toolbox. To test the reduced-order model and the distributional controller, simulations were performed under different basis function bandwidths and model parameters, such as basis function grid bandwidth, varying the number of basis functions used for states and inputs, effect of various ρ values, differing the magnitudes of the objectives' peaks.

By developing a modelling and control framework for passive swarm robots that scale to high numbers, this thesis has laid the groundwork for future developments to advance the paradigm further.

6.2. Future Work

In this section, further extensions to this work are outlined. Three such ideas are presented.

Sparse Representation of the Reduced-Order Model

Currently, a simulation of a maximum of a 8 by 8 resolution for the states and inputs can be performed, due to limited computational power. In the simulations for a large resolution grid, however, most of the states remain close to zero for large amounts of time and the dynamic potential field control will also be limited to these spatial regions. The computational time could be vastly improved by exploiting the sparsity through a sparse dynamic representation of the model, in which only the states having non-zero elements are recruited. This would be helpful in running at a much higher resolution grid (i.e. 100 by 100 basis function grid, or 10,000 states).

Switching Tasks Under Dynamic Potential Fields

The controller framework studied in this thesis only considers reaching a static objective PDF, considered to be the final desired spatial distribution of the swarm. However, the control problem may be solved more easily and be more efficient if the parallels are made with the control of navigation problem. For this, the formulation can be extended by having a dynamic objective PDF, which would then lead to modifications to the controller. This modification may involve a supervisory controller that seeks to optimise the dynamic objective PDF in terms of minimal time to destination for instance.

Autonomous Agents Under a Dynamic Input Field

As the inspiration for our research was to simulate the computationally-limited microrobots, we have limited the study to swarm robots that are passive and are solely guided by the external potential input field. However, provided that the swarm robots' autonomy or in computation and actuation is not limited, how would the "active" robot swarm be complemented with a limited bandwidth input field? This extension of the problem class is more challenging with potential trade-off between the passive and active elements of control leading to finer control of swarms.

Appendices

A

Useful Relationshiops on Kronecker Product Algebra

Let $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{B} \in \mathbb{R}^{N \times P}$ and $\mathbf{C} \in \mathbb{R}^{P \times Q}$. Then,

$$\text{vec}(\mathbf{AB}) = (\mathbf{I}_P \otimes \mathbf{A})\text{vec}(\mathbf{B}) = (\mathbf{B}^\top \otimes \mathbf{I}_M)\text{vec}(\mathbf{A}) \tag{A.1}$$

and

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^\top \otimes \mathbf{A})\text{vec}(\mathbf{B}) = (\mathbf{I}_Q \otimes \mathbf{AB})\text{vec}(\mathbf{C}) \tag{A.2}$$

The reduction in Chapter 4 requires the manipulation of a term of the form,

$$\mathbf{d} = \mathbf{ab}^\top\mathbf{Cc}, \quad \text{where} \quad \mathbf{d}, \mathbf{a}, \mathbf{b} \in \mathbb{R}^M, \quad \mathbf{C} \in \mathbb{R}^{M \times N}, \quad \mathbf{c} \in \mathbb{R}^N \tag{A.3}$$

Applying the rules above,

$$\begin{aligned} \mathbf{d} &= \text{vec}(\mathbf{d}) = \text{vec}(\mathbf{ab}^\top\mathbf{Cc}) = (\mathbf{c}^\top \otimes \mathbf{I}_M)\text{vec}(\mathbf{ab}^\top\mathbf{C}) \\ &= (\mathbf{c}^\top \otimes \mathbf{I}_M)(\mathbf{C}^\top \otimes \mathbf{a})\text{vec}(\mathbf{b}^\top) = (\mathbf{c}^\top \otimes \mathbf{I}_M)(\mathbf{C}^\top \otimes \mathbf{a})\mathbf{b} \end{aligned} \tag{A.4}$$

Noting that the scalar term $\mathbf{b}^\top\mathbf{Cc} = \mathbf{c}^\top\mathbf{Cb}$, an equivalent expression can be written as,

$$\mathbf{d} = \text{vec}(\mathbf{d}) = \text{vec}(\mathbf{ac}^\top\mathbf{Cb}) = (\mathbf{b}^\top \otimes \mathbf{I}_M)(\mathbf{C} \otimes \mathbf{a})\mathbf{c} \tag{A.5}$$

Even though these expressions appear to have increased the dimensions of the matrices involved in the multiplications, the important reason for the manipulation is to write these equations with $\mathbf{a}$ and $\mathbf{c}$ appearing at the front and back. This will be required for subsequent analysis and design with the reduced model.

B

Derivations of the State Matrices Reduced Order Model in 2-D Canonical Form

The following details the derivations of the state matrices of the Galerkin-reduced advection-diffusion equation in two dimensions. To show this, we first present the properties of the multivariate Gaussian functions in parametric representation, its gradient and Laplace operators, and its integrations. Afterwards, the canonical representation of the multivariate Gaussian functions are presented. The advantage of using the canonical representation is its ease for deriving products of the multivariate Gaussian functions [62]. Using this canonical notation, we show the derivations of the state matrices of the reduced advection-diffusion equation.

Multivariate Gaussian Functions

The multivariate Gaussian probability density function in its traditional parametric representation is expressed in the terms of the parameters $\mu \in \mathbb{R}^2$, the mean vector, and $\mathbf{\Sigma} \in \mathbb{R}^{2\times 2}$, the positive-definite covariance matrix. The multivariate Gaussian PDF in parametric representation is defined as

$$\phi(\mathbf{s}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi\,|\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{s}-\boldsymbol{\mu})\right]. \tag{B.1}$$

The multivariate Gaussian PDF can be simplified as

$$\phi(\mathbf{s}) = \mathbf{k}\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right], \tag{B.2}$$

where

$$\mathbf{k} = \left[\sqrt{2\pi\,|\boldsymbol{\Sigma}|}\right]^{-1} \qquad \text{and} \qquad \mathbf{g}(\mathbf{s}) = (\mathbf{s}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{s}-\boldsymbol{\mu}). \tag{B.3}$$

To express the reduced advection-diffusion equation in two dimensions, let $\mathbf{s} = [\mathrm{s}_1 \quad \mathrm{s}_2]^\top$, $\boldsymbol{\mu} = [\mu_1 \quad \mu_2]^\top$, and $\boldsymbol{\Sigma} = [\rho_{11} \quad \rho_{12}; \quad \rho_{12} \quad \rho_{22}]$. With these parameters, $g(\mathbf{s})$ is expressed as

$$\begin{aligned}\mathbf{g}(\mathbf{s}) &= [\mathrm{s}_1-\mu_1 \quad \mathrm{s}_2-\mu_2]\begin{bmatrix}\rho_{11} & \rho_{12}\\ \rho_{12} & \rho_{22}\end{bmatrix}\begin{bmatrix}\mathrm{s}_1-\mu_1\\ \mathrm{s}_2-\mu_2\end{bmatrix}\\ &= \rho_{11}(\mathrm{s}_1-\mu_1)^2 + 2\rho_{12}(\mathrm{s}_1-\mu_1)(\mathrm{s}_2-\mu_2) + \rho_{22}(\mathrm{s}_2-\mu_2)^2.\end{aligned} \tag{B.4}$$

Gradient

To derive the gradient of the multivariate Gaussian function, we first calculate the partial derivatives of the term $g(\mathbf{s})$. The partial derivative of $g(\mathbf{s})$ in the s_1 direction is

$$\begin{aligned}\frac{\partial\mathbf{g}(\mathbf{s})}{\partial\mathrm{s}_1} &= 2\rho_{11}(\mathrm{s}_1-\mu_1) + 2\rho_{12}(\mathrm{s}_2-\mu_2)\\ &= 2[\mathrm{s}_1-\mu_1 \quad \mathrm{s}_2-\mu_2]\begin{bmatrix}\rho_{11}\\ \rho_{12}\end{bmatrix}.\end{aligned} \tag{B.5}$$

Similarly, partial derivative of $g(\mathbf{s})$ in the s_2 direction is

$$\begin{aligned}\frac{\partial\mathbf{g}(\mathbf{s})}{\partial\mathrm{s}_2} &= 2\rho_{12}(\mathrm{s}_1-\mu_1) + 2\rho_{22}(\mathrm{s}_2-\mu_2)\\ &= 2[\mathrm{s}_1-\mu_1 \quad \mathrm{s}_2-\mu_2]\begin{bmatrix}\rho_{12}\\ \rho_{22}\end{bmatrix}.\end{aligned} \tag{B.6}$$

Combining the partial derivatives of $g(\mathbf{s})$ above, we have

$$\begin{aligned} \frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}} &= \begin{bmatrix} \frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}_1} & \frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}_2} \end{bmatrix} \\ &= 2[\mathbf{s}_1 - \mu_1 \quad \mathbf{s}_2 - \mu_2] \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{12} & \rho_{22} \end{bmatrix} \\ &= 2(\mathbf{s} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}. \end{aligned} \tag{B.7}$$

Using these results, we can derive the gradient of the Gaussian function. The partial derivative of the Gaussian function in the $\mathbf{s}_1$ direction is

$$\begin{aligned} \frac{\partial \phi(\mathbf{s})}{\partial \mathbf{s}_1} &= \frac{\partial}{\partial \mathbf{s}_1} \left\{ \mathbf{k} \exp\left[-\frac{1}{2}\, \mathbf{g}(\mathbf{s}) \right] \right\} \\ &= \mathbf{k} \exp\left[-\frac{1}{2}\, \mathbf{g}(\mathbf{s}) \right] \cdot \left(-\frac{1}{2} \right) \frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}_1} \\ &= \phi(\mathbf{s}) \cdot \left(-\frac{1}{2} \right) 2[\mathbf{s}_1 - \mu_1 \quad \mathbf{s}_2 - \mu_2] \begin{bmatrix} \rho_{11} \\ \rho_{12} \end{bmatrix} \\ &= -\phi(\mathbf{s}) \cdot [\mathbf{s}_1 - \mu_1 \quad \mathbf{s}_2 - \mu_2] \begin{bmatrix} \rho_{11} \\ \rho_{12} \end{bmatrix} \\ &= -\phi(\mathbf{s}) \cdot [\rho_{11}(\mathbf{s}_1 - \mu_1) + \rho_{12}(\mathbf{s}_2 - \mu_2)]. \end{aligned} \tag{B.8}$$

Similarly, the partial derivative of the Gaussian function in the $\mathbf{s}_2$ direction is

$$\begin{aligned} \frac{\partial \phi(\mathbf{s})}{\partial \mathbf{s}_2} &= -\phi(\mathbf{s}) \cdot [\mathbf{s}_1 - \mu_1 \quad \mathbf{s}_2 - \mu_2] \begin{bmatrix} \rho_{12} \\ \rho_{22} \end{bmatrix} \\ &= -\phi(\mathbf{s}) \cdot [\rho_{12}(\mathbf{s}_1 - \mu_1) + \rho_{22}(\mathbf{s}_2 - \mu_2)] \end{aligned} \tag{B.9}$$

Combining both terms, the gradient of the Gaussian function is

$$\begin{aligned} \nabla \phi(\mathbf{s}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\partial \phi(\mathbf{s})}{\partial \mathbf{s}} &= \begin{bmatrix} \frac{\partial \phi(\mathbf{s}_1, \mathbf{s}_2)}{\partial \mathbf{s}_1} & \frac{\partial \phi(\mathbf{s}_2, \mathbf{s}_2)}{\partial \mathbf{s}_1} \end{bmatrix} \\ &= -\phi(\mathbf{s}_1, \mathbf{s}_2) \cdot [\mathbf{s}_1 - \mu_1 \quad \mathbf{s}_2 - \mu_2] \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{12} & \rho_{22} \end{bmatrix} \\ &= -\phi(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}. \end{aligned} \tag{B.10}$$

Laplace Operator

To derive the Laplace operator of the multivariate Gaussian function, we first derive the second derivative of the Gaussian function in the $\mathbf{s}_1$ direction

$$\begin{aligned}\frac{\partial^2 \phi(\mathbf{s}|\mu, \mathbf{\Sigma})}{\partial \mathbf{s}_1^2} &= \frac{\partial}{\partial \mathbf{s}_1}\left\{-\phi(\mathbf{s})[\rho_{11}(\mathbf{s}_1 - \mu_1) + \rho_{12}(\mathbf{s}_2 - \mu_2)]\right\} \\ &= -\frac{\partial}{\partial \mathbf{s}_1}\left\{\phi(\mathbf{s})\rho_{11}(\mathbf{s}_1 - \mu_1) + \phi(\mathbf{s})\rho_{12}(\mathbf{s}_2 - \mu_2)]\right\} .\end{aligned} \tag{B.11}$$

The first term of this equation is derived as

$$\begin{aligned}\frac{\partial}{\partial \mathbf{s}_1}\left\{\phi(\mathbf{s})\rho_{11}(\mathbf{s}_1 - \mu_1)\right\} &= \frac{\partial}{\partial \mathbf{s}_1}\left\{\mathbf{k}\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right]\rho_{11}(\mathbf{s}_1 - \mu_1)\right\} \\ &= \mathbf{k}\rho_{11}\frac{\partial}{\partial \mathbf{s}_1}\left\{(\mathbf{s}_1 - \mu_1)\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right]\right\} \\ &= \mathbf{k}\rho_{11}\left\{\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right] + (\mathbf{s}_1 - \mu_1)\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right]\cdot\left(-\frac{1}{2}\right)\frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}_1}\right\} \\ &\text{(via product rule)} \\ &= \left(\mathbf{k}\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right]\right)\rho_{11}\left\{1 - \frac{1}{2}(\mathbf{s}_1 - \mu_1)\frac{\partial \mathbf{g}(\mathbf{s})}{\partial \mathbf{s}_1}\right\} \\ &= \phi(\mathbf{s})\rho_{11}\left\{1 - \frac{1}{2}(\mathbf{s}_1 - \mu_1)\left[2\rho_{11}(\mathbf{s}_1 - \mu_1) + 2\rho_{12}(\mathbf{s}_2 - \mu_2)\right]\right\} \\ &= \phi(\mathbf{s})\rho_{11}\left\{1 - \rho_{11}(\mathbf{s}_1 - \mu_1)^2 - \rho_{12}(\mathbf{s}_1 - \mu_1)(\mathbf{s}_2 - \mu_2)\right\} \\ &= \phi(\mathbf{s})\left\{\rho_{11} - \rho_{11}^2(\mathbf{s}_1 - \mu_1)^2 - \rho_{11}\rho_{12}(\mathbf{s}_1 - \mu_1)(\mathbf{s}_2 - \mu_2)\right\} .\end{aligned} \tag{B.12}$$

The second term is derived as

$$\begin{aligned}\frac{\partial}{\partial \mathbf{s}_1}\left\{\phi(\mathbf{s})\rho_{12}(\mathbf{s}_2 - \mu_2)\right\} &= \frac{\partial}{\partial \mathbf{s}_1}\left\{\mathbf{k}\exp\left[-\frac{1}{2}\,\mathbf{g}(\mathbf{s})\right]\rho_{12}(\mathbf{s}_2 - \mu_2)\right\} \\ &= \phi(\mathbf{s})\rho_{12}\left\{1 - \frac{1}{2}(\mathbf{s}_2 - \mu_2)\left[2\rho_{11}(\mathbf{s}_1 - \mu_1) + 2\rho_{12}(\mathbf{s}_2 - \mu_2)\right]\right\} \\ &= \phi(\mathbf{s})\rho_{12}\left\{1 - \rho_{11}(\mathbf{s}_1 - \mu_1)(\mathbf{s}_2 - \mu_2) - \rho_{12}(\mathbf{s}_2 - \mu_2)^2\right\} \\ &= \phi(\mathbf{s})\left\{\rho_{12} - \rho_{11}\rho_{12}(\mathbf{s}_1 - \mu_1)(\mathbf{s}_2 - \mu_2) - \rho_{12}^2(\mathbf{s}_2 - \mu_2)^2\right\} .\end{aligned} \tag{B.13}$$

Combining these two terms,

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{s}_1}\left\{\phi(\mathbf{s})\rho_{11}(\mathbf{s}_1-\mu_1)+\phi(\mathbf{s})\rho_{12}(\mathbf{s}_2-\mu_2)]\right\} \\
&\quad=\phi(\mathbf{s})\left\{(\rho_{11}+\rho_{12})-\rho_{11}^2(\mathbf{s}_1-\mu_1)^2-2\rho_{11}\rho_{12}(\mathbf{s}_1-\mu_1)(\mathbf{s}_2-\mu_2)-\rho_{12}^2(\mathbf{s}_2-\mu_2)^2\right\} \\
&\quad=\phi(\mathbf{s})\left\{(\rho_{11}+\rho_{12})-\left[\rho_{11}(\mathbf{s}_1-\mu_1)+\rho_{12}(\mathbf{s}_2-\mu_2)\right]^2\right\}.
\end{aligned} \tag{B.14}$$

Therefore, the second derivative of the Gaussian function in the $\mathbf{s}_1$ direction is

$$\begin{aligned}
\frac{\partial^2\phi(\mathbf{s})}{\partial \mathbf{s}_1^2}&=-\frac{\partial}{\partial \mathbf{s}_1}\left\{\phi(\mathbf{s})\rho_{11}(\mathbf{s}_1-\mu_1)+\phi(\mathbf{s})\rho_{12}(\mathbf{s}_2-\mu_2)]\right\} \\
&=\phi(\mathbf{s})\left\{\left[\rho_{11}(\mathbf{s}_1-\mu_1)+\rho_{12}(\mathbf{s}_2-\mu_2)\right]^2-(\rho_{11}+\rho_{12})\right\}.
\end{aligned} \tag{B.15}$$

Similarly, the second derivative of the Gaussian function in the $\mathbf{s}_2$ direction is

$$\begin{aligned}
\frac{\partial^2\phi(\mathbf{s})}{\partial \mathbf{s}_2^2}&=-\frac{\partial}{\partial \mathbf{s}_2}\left\{\phi(\mathbf{s})\rho_{12}(\mathbf{s}_1-\mu_1)+\phi(\mathbf{s})\rho_{22}(\mathbf{s}_2-\mu_2)]\right\} \\
&=\phi(\mathbf{s})\left\{\left[\rho_{12}(\mathbf{s}_1-\mu_1)+\rho_{22}(\mathbf{s}_2-\mu_2)\right]^2-(\rho_{12}+\rho_{22})\right\}.
\end{aligned} \tag{B.16}$$

Combining the terms above, the Laplace operator is

$$\begin{aligned}
\nabla^2\phi(\mathbf{s}|\mu,\boldsymbol{\Sigma})&=\frac{\partial^2\phi(\mathbf{s})}{\partial \mathbf{s}_1^2}+\frac{\partial^2\phi(\mathbf{s})}{\partial \mathbf{s}_2^2} \\
&=\phi(\mathbf{s})\{\left[\rho_{11}(\mathbf{s}_1-\mu_1)+\rho_{12}(\mathbf{s}_2-\mu_2)\right]^2+\left[\rho_{12}(\mathbf{s}_1-\mu_1)+\rho_{22}(\mathbf{s}_2-\mu_2)\right]^2 \\
&\qquad-(\rho_{11}+2\rho_{12}+\rho_{22})]\} \\
&=\phi(\mathbf{s})\,\mathbf{h}(\mathbf{s}),
\end{aligned} \tag{B.17}$$

where

$$\begin{aligned}
\mathbf{h}(\mathbf{s}) &= \left[\rho_{11}(\mathbf{s}_1-\boldsymbol{\mu}_1)+\rho_{12}(\mathbf{s}_2-\boldsymbol{\mu}_2)\right]^2+\left[\rho_{12}(\mathbf{s}_1-\boldsymbol{\mu}_1)+\rho_{22}(\mathbf{s}_2-\boldsymbol{\mu}_2)\right]^2-(\rho_{11}+2\rho_{12}+\rho_{22})\\
&= \rho_{11}^2(\mathbf{s}_1-\boldsymbol{\mu}_1)^2+2\rho_{11}\rho_{12}(\mathbf{s}_1-\boldsymbol{\mu}_1)(\mathbf{s}_2-\boldsymbol{\mu}_2)+\rho_{12}(\mathbf{s}_2-\boldsymbol{\mu}_2)\\
&\quad+\rho_{12}^2(\mathbf{s}_1-\boldsymbol{\mu}_1)^2+2\rho_{12}\rho_{22}(\mathbf{s}_1-\boldsymbol{\mu}_1)(\mathbf{s}_2-\boldsymbol{\mu}_2)+\rho_{22}(\mathbf{s}_2-\boldsymbol{\mu}_2)-(\rho_{11}+2\rho_{12}+\rho_{22})\\
&= (\rho_{11}^2+\rho_{12}^2)(\mathbf{s}_1-\boldsymbol{\mu}_1)^2+2(\rho_{11}\rho_{12}+\rho_{12}\rho_{22})(\mathbf{s}_1-\boldsymbol{\mu}_1)(\mathbf{s}_2-\boldsymbol{\mu}_2)+(\rho_{12}^2+\rho_{22}^2)(\mathbf{s}_2-\boldsymbol{\mu}_2)^2\\
&\quad-(\rho_{11}+2\rho_{12}+\rho_{22})\\
&= [\mathbf{s}_1-\boldsymbol{\mu}_1 \quad \mathbf{s}_2-\boldsymbol{\mu}_2]\begin{bmatrix}\rho_{11}^2+\rho_{12}^2 & \rho_{11}\rho_{12}+\rho_{12}\rho_{22}\\ \rho_{11}\rho_{12}+\rho_{12}\rho_{22} & \rho_{12}^2+\rho_{22}^2\end{bmatrix}\begin{bmatrix}\mathbf{s}_1-\boldsymbol{\mu}_1\\ \mathbf{s}_2-\boldsymbol{\mu}_2\end{bmatrix}-(\rho_{11}+2\rho_{12}+\rho_{22})\\
&= [\mathbf{s}_1-\boldsymbol{\mu}_1 \quad \mathbf{s}_2-\boldsymbol{\mu}_2]\begin{bmatrix}\rho_{11} & \rho_{12}\\ \rho_{12} & \rho_{22}\end{bmatrix}^2\begin{bmatrix}\mathbf{s}_1-\boldsymbol{\mu}_1\\ \mathbf{s}_2-\boldsymbol{\mu}_2\end{bmatrix}-(\rho_{11}+2\rho_{12}+\rho_{22})\\
&= (\mathbf{s}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-2}(\mathbf{s}-\boldsymbol{\mu})-\mathbf{1}^\top\boldsymbol{\Sigma}^{-1}\mathbf{1}.
\end{aligned} \tag{B.18}$$

Integral Identities

The following are the integral identities of the multivariate Gaussian function:

- $\int_{\mathcal{S}} \phi_i(\mathbf{s})\, d\mathbf{s} = 1$
- $\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)\, d\mathbf{s} = 0$
- $\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}\, d\mathbf{s} = \boldsymbol{\mu}_i$
- $\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)(\mathbf{s}-\boldsymbol{\mu}_i)^\top d\mathbf{s} = \int \phi_i(\mathbf{s})\left[\mathbf{s}\mathbf{s}^\top-\boldsymbol{\mu}_i\mathbf{s}^\top-\mathbf{s}\boldsymbol{\mu}_i^\top+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top\right] d\mathbf{s} = \boldsymbol{\Sigma}_i$
- $\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}\mathbf{s}^\top d\mathbf{s} = \boldsymbol{\Sigma}_i+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top$

Derivation of above term:

$$\begin{aligned}
\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}\mathbf{s}^\top d\mathbf{s} &= \boldsymbol{\Sigma}_i+\int_{\mathcal{S}}\phi_i(\mathbf{s})\boldsymbol{\mu}_i\mathbf{s}^\top d\mathbf{s}+\int_{\mathcal{S}}\phi_i(\mathbf{s})\mathbf{s}\boldsymbol{\mu}_i^\top d\mathbf{s}-\int_{\mathcal{S}}\phi_i(\mathbf{s})\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top d\mathbf{s}\\
&= \boldsymbol{\Sigma}_i+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top-\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top\\
&= \boldsymbol{\Sigma}_i+\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top.
\end{aligned} \tag{B.19}$$

- $\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}^\top\mathbf{s}\, d\mathbf{s} = \mathrm{tr}(\boldsymbol{\Sigma}_i)+\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i$

Derivation of above term:

$$\begin{aligned}\int_{\mathcal{S}} \phi_i(\mathbf{s})\,(\mathbf{s}^\top\mathbf{s})\,d\mathbf{s} &= \int \phi_i(\mathbf{s})\ \mathrm{tr}(\mathbf{s}\mathbf{s}^\top)\,d\mathbf{s} = \mathrm{tr}\left[\int \phi_i(\mathbf{s})\,(\mathbf{s}\mathbf{s}^\top)\,d\mathbf{s}\right] \\ &= \mathrm{tr}\left[\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top\right] \\ &= \mathrm{tr}(\boldsymbol{\Sigma}_i) + \boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i.\end{aligned} \tag{B.20}$$

- $\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}^\top\mathbf{A}\mathbf{s})\,d\mathbf{s} = \mathrm{tr}[\mathbf{A}\boldsymbol{\Sigma}_i] + \boldsymbol{\mu}_i^\top\mathbf{A}\boldsymbol{\mu}_i$

Derivation of above term:

$$\begin{aligned}\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}^\top\mathbf{A}\mathbf{s})\,d\mathbf{s} &= \int \phi_i(\mathbf{s})\,\mathrm{tr}(\mathbf{s}^\top\mathbf{A}\mathbf{s})\,d\mathbf{s} = \int \phi_i(\mathbf{s})\,\mathrm{tr}(\mathbf{A}\mathbf{s}\mathbf{s}^\top)\,d\mathbf{s} \\ &= \mathrm{tr}\left[\int \phi_i(\mathbf{s})(\mathbf{A}\mathbf{s}\mathbf{s}^\top)\,d\mathbf{s}\right] = \mathrm{tr}\left[\mathbf{A}\int \phi_i(\mathbf{s})(\mathbf{s}\mathbf{s}^\top)\,d\mathbf{s}\right] \\ &= \mathrm{tr}\left[\mathbf{A}(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top)\right] = \mathrm{tr}[\mathbf{A}\boldsymbol{\Sigma}_i] + \mathrm{tr}[\mathbf{A}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top] \\ &= \mathrm{tr}[\mathbf{A}\boldsymbol{\Sigma}_i] + \boldsymbol{\mu}_i^\top\mathbf{A}\boldsymbol{\mu}_i.\end{aligned} \tag{B.21}$$

- $\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_j)(\mathbf{s} - \boldsymbol{\mu}_j)^\top\,d\mathbf{s} = \boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top$

Derivation of above term:

$$\begin{aligned}\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_j)(\mathbf{s} - \boldsymbol{\mu}_j)^\top\,d\mathbf{s} &= \int_{\mathcal{S}} \phi_i(\mathbf{s})\left[(\mathbf{s} - \boldsymbol{\mu}_i) + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\right]\left[(\mathbf{s} - \boldsymbol{\mu}_i) + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\right]^\top\,d\mathbf{s} \\ &= \int_{\mathcal{S}} \phi_i(\mathbf{s})[(\mathbf{s} - \boldsymbol{\mu}_i)(\mathbf{s} - \boldsymbol{\mu}_i)^\top + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\mathbf{s} - \boldsymbol{\mu}_i)^\top \\ &\qquad + (\mathbf{s} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top]\,d\mathbf{s} \\ &= \left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_i)(\mathbf{s} - \boldsymbol{\mu}_i)^\top\,d\mathbf{s}\right] + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_i)^\top\,d\mathbf{s}\right] \\ &\qquad + \left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_i)\,d\mathbf{s}\right](\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top \\ &\qquad + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top\left[\int_{\mathcal{S}} \phi_i(\mathbf{s})\,d\mathbf{s}\right] \\ &= \boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top\end{aligned} \tag{B.22}$$

- $\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s} - \boldsymbol{\mu}_j)(\mathbf{s} - \boldsymbol{\mu}_k)^\top\,d\mathbf{s} = \boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_k)^\top$

Derivation of above term:

$$
\begin{aligned}
\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_j)(\mathbf{s}-\boldsymbol{\mu}_k)^\top d\mathbf{s} &= \int_{\mathcal{S}} \phi_i(\mathbf{s}) \left[(\mathbf{s}-\boldsymbol{\mu}_i)+(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)\right] \left[(\mathbf{s}-\boldsymbol{\mu}_i)+(\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)\right]^\top d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})[(\mathbf{s}-\boldsymbol{\mu}_i)(\mathbf{s}-\boldsymbol{\mu}_i)^\top + (\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)(\mathbf{s}-\boldsymbol{\mu}_i)^\top \\
&\quad + (\mathbf{s}-\boldsymbol{\mu}_i)(\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)^\top + (\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)(\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)^\top]\, d\mathbf{s} \\
&= \left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)(\mathbf{s}-\boldsymbol{\mu}_i)^\top d\mathbf{s}\right] + (\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)\left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)^\top d\mathbf{s}\right] \\
&\quad + \left[\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)\, d\mathbf{s}\right](\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)^\top \\
&\quad + (\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)(\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)^\top \left[\int_{\mathcal{S}} \phi_i(\mathbf{s})\, d\mathbf{s}\right] \\
&= \boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)(\boldsymbol{\mu}_i-\boldsymbol{\mu}_k)^\top
\end{aligned}
\tag{B.23}
$$

Canonical Notation

The multivariate Gaussian function can alternatively be represented in canonical notation. The primary advantage of expressing the multivariate Gaussians in canonical notation is in its easy of deriving products. The multivariate Gaussian function in canonical notation is

$$\phi_i(\mathbf{s}|\boldsymbol{\eta}_i, \boldsymbol{\Lambda}_i) = \exp\left[\zeta_i + \boldsymbol{\eta}_i^\top \mathbf{s} - \frac{1}{2}\mathbf{s}^\top \boldsymbol{\Lambda}_i \mathbf{s}\right], \tag{B.24}$$

where

$$\boldsymbol{\Lambda}_i = \boldsymbol{\Sigma}_i^{-1}, \quad \boldsymbol{\eta}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i, \quad \text{and} \quad \zeta_i = -\frac{1}{2}(2\log 2\pi - \log|\boldsymbol{\Lambda}_i| + \boldsymbol{\eta}_i^\top \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i) \tag{B.25}$$

Using this representation, the product of n Gaussian functions is defined as

$$\prod_{a=1}^{n} f_a(\mathbf{s}) = \exp(\zeta_{a=1\ldots n} - \zeta_a)\exp\left[\zeta_a + \boldsymbol{\eta}_a^\top \mathbf{s} - \frac{1}{2}\mathbf{s}^\top \boldsymbol{\Lambda}_a \mathbf{s}\right], \tag{B.26}$$

where

$$\zeta_{a=1\ldots n} = \sum_{a=1}^{n} \zeta_a = -\frac{1}{2}\left(2n\log 2\pi - \sum_{a=1}^{n}\log|\boldsymbol{\Lambda}_a| + \sum_{a=1}^{n}\boldsymbol{\eta}_a^\top \boldsymbol{\Lambda}_a^{-1}\boldsymbol{\eta}_a\right), \tag{B.27}$$

$$\boldsymbol{\Lambda}_n = \sum_{a=1}^{n}\boldsymbol{\Lambda}_a, \quad \boldsymbol{\eta}_n = \sum_{a=1}^{n}\boldsymbol{\eta}_a, \quad \text{and} \quad \zeta_n = -\frac{1}{2}(2\log 2\pi - \log|\boldsymbol{\Lambda}_n| + \boldsymbol{\eta}_n^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\eta}_n) \tag{B.28}$$

Product of 2 Multivariate Gaussians

The product of two multivariate Gaussian functions is

$$\phi_i(\mathbf{s}|\boldsymbol{\eta}_i, \boldsymbol{\Lambda}_i) \cdot \phi_j(\mathbf{s}|\boldsymbol{\eta}_j, \boldsymbol{\Lambda}_j) = \underbrace{\exp(\zeta_{a=1\ldots2} - \zeta_2)}_{\mathbf{S}_{ij}} \underbrace{\exp\left[\zeta_2 + \boldsymbol{\eta}_{ij}^\top \mathbf{s} - \frac{1}{2}\mathbf{s}^\top \boldsymbol{\Lambda}_{ij}\mathbf{s}\right]}_{\phi_{ij}} \tag{B.29}$$

$$= \mathbf{S}_{ij}\ \phi_{ij}(\mathbf{s})$$

where

$$\zeta_{a=1\ldots2} = -\frac{1}{2}\left[4\log 2\pi - (\log|\boldsymbol{\Lambda}_i| + \log|\boldsymbol{\Lambda}_j|) + (\boldsymbol{\eta}_i^\top \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i + \boldsymbol{\eta}_j^\top \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right], \tag{B.30}$$

$$\zeta_2 = -\frac{1}{2}\left[2\log 2\pi - \log|\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j| + (\boldsymbol{\eta}_i + \boldsymbol{\eta}_j)^\top(\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j)^{-1}(\boldsymbol{\eta}_i + \boldsymbol{\eta}_j)\right], \tag{B.31}$$

$$\boldsymbol{\eta}_{ij} = \boldsymbol{\eta}_i + \boldsymbol{\eta}_j \qquad \text{and} \qquad \boldsymbol{\Lambda}_{ij} = \boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j. \tag{B.32}$$

Product of 3 Multivariate Gaussians

The product of three multivariate Gaussian functions is

$$\phi_i(\mathbf{s}|\boldsymbol{\eta}_i, \boldsymbol{\Lambda}_i) \cdot \phi_j(\mathbf{s}|\boldsymbol{\eta}_j, \boldsymbol{\Lambda}_j) \cdot \phi_k(\mathbf{s}|\boldsymbol{\eta}_k, \boldsymbol{\Lambda}_k) = \underbrace{\exp(\zeta_{a=1\ldots3} - \zeta_3)}_{\mathbf{S}_{ijk}} \underbrace{\exp\left[\zeta_3 + \boldsymbol{\eta}_{ijk}^\top \mathbf{s} - \frac{1}{2}\mathbf{s}^\top \boldsymbol{\Lambda}_{ijk}\mathbf{s}\right]}_{\phi_{ijk}}$$

$$= \mathbf{S}_{ijk}\ \phi_{ijk}(\mathbf{s}), \tag{B.33}$$

where

$$\zeta_{a=1\ldots3} = -\frac{1}{2}\left[6\log 2\pi - (\log|\boldsymbol{\Lambda}_i| + \log|\boldsymbol{\Lambda}_j| + \log|\boldsymbol{\Lambda}_k|) + (\boldsymbol{\eta}_i^\top \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i + \boldsymbol{\eta}_j^\top \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j + \boldsymbol{\eta}_k^\top \boldsymbol{\Lambda}_k^{-1}\boldsymbol{\eta}_k)\right], \tag{B.34}$$

$$\zeta_3 = -\frac{1}{2}\left[2\log 2\pi - \log|\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j + \boldsymbol{\Lambda}_k| + (\boldsymbol{\eta}_i + \boldsymbol{\eta}_j + \boldsymbol{\eta}_k)^\top(\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j + \boldsymbol{\Lambda}_k)^{-1}(\boldsymbol{\eta}_i + \boldsymbol{\eta}_j + \boldsymbol{\eta}_k)\right], \tag{B.35}$$

$$\boldsymbol{\eta}_{ijk} = \boldsymbol{\eta}_i + \boldsymbol{\eta}_j + \boldsymbol{\eta}_k \qquad \text{and} \qquad \boldsymbol{\Lambda}_{ijk} = \boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j + \boldsymbol{\Lambda}_k \tag{B.36}$$

Gradient and Laplace operator

Using the gradient expression in parametric representation from the previous section, the gradient in canonical notation is

$$\begin{aligned}\nabla\phi_i(\mathbf{s}) &= -\phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\mu}_i)^\top\boldsymbol{\Sigma}_i^{-1}\\ &= -\phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\boldsymbol{\Lambda}_i\end{aligned} \tag{B.37}$$

Similarly, the Laplace operator in canonical notation is

$$\begin{aligned}\nabla^2\phi_i(\mathbf{s}) &= \phi_i(\mathbf{s})\left[(\mathbf{s}-\boldsymbol{\mu}_i)^\top\boldsymbol{\Sigma}_i^{-2}(\mathbf{s}-\boldsymbol{\mu}_i) - \mathbf{1}^\top\boldsymbol{\Sigma}_i^{-1}\mathbf{1}\right]\\ &= \phi_i(\mathbf{s})\left[(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\boldsymbol{\Lambda}_i^2(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i) - \mathbf{1}^\top\boldsymbol{\Lambda}_i\mathbf{1}\right]\end{aligned} \tag{B.38}$$

Integral Identities

Using the integral identities in parametric representation from the previous section, the integral identities in canonical notation are

- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})\, d\mathbf{s} = 1$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\, d\mathbf{s} = 0$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}\, d\mathbf{s} = \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\mathbf{s}-\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\, d\mathbf{s} = \boldsymbol{\Lambda}_i^{-1}$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}\mathbf{s}^\top\, d\mathbf{s} = \boldsymbol{\Lambda}_i^{-1} + (\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})\,\mathbf{s}^\top\mathbf{s}\, d\mathbf{s} = \mathrm{tr}(\boldsymbol{\Lambda}_i^{-1}) + (\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}^\top\mathbf{A}\mathbf{s})\, d\mathbf{s} = \mathrm{tr}\left[\mathbf{A}\boldsymbol{\Lambda}_i^{-1}\right] + (\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\mathbf{A}(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\, d\mathbf{s} = \boldsymbol{\Lambda}_i^{-1} + (\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top$
- $\displaystyle\int_{\mathcal{S}} \phi_i(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\boldsymbol{\Lambda}_k^{-1}\boldsymbol{\eta}_k)^\top\, d\mathbf{s} = \boldsymbol{\Lambda}_i^{-1} + (\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i - \boldsymbol{\Lambda}_k^{-1}\boldsymbol{\eta}_k)^\top$

Reduced Advection-Diffusion Equation

The Galerkin-reduced advection-diffusion can be expressed as

$$\dot{\mathbf{x}}(t) = \gamma \mathbf{E}^{-1}\mathbf{A}\mathbf{x}(t) - \mathbf{E}^{-1}\left(\mathbf{x}(t)^\top \otimes \mathbf{I}_{N_x}\right)\left(\mathbf{B}^{(1)} + \mathbf{B}^{(2)}\right)\mathbf{u}(t) \tag{B.39}$$

where

$$\mathbf{E} = \int_{\mathcal{S}} \boldsymbol{\phi}(\mathbf{s})\boldsymbol{\phi}(\mathbf{s})^\top \, d\mathbf{s} \quad \text{and} \quad \mathbf{E}_{ij} = \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\, d\mathbf{s}, \tag{B.40}$$

$$\mathbf{A} = \int_{\mathcal{S}} \boldsymbol{\phi}(\mathbf{s})\nabla^2\boldsymbol{\phi}(\mathbf{s})^\top \, d\mathbf{s} \quad \text{and} \quad \mathbf{A}_{ij} = \int_{\mathcal{S}} \phi_i(\mathbf{s})\nabla^2\phi_j(\mathbf{s})\, d\mathbf{s}, \tag{B.41}$$

$$\mathbf{B}^{(1)} = \int_{\mathcal{S}} \left[\nabla\Phi(\mathbf{s})\nabla\Psi(\mathbf{s})^\top\right] \otimes \Phi(\mathbf{s})\, d\mathbf{s} \quad \text{and} \quad \mathbf{B}^{(1)}_{ijk} = \int_{\mathcal{S}} \nabla\phi_i(\mathbf{s})\nabla\psi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s}, \tag{B.42}$$

$$\mathbf{B}^{(2)} = \int_{\mathcal{S}} \left[\Phi(\mathbf{s})\nabla^2\Psi(\mathbf{s})^\top\right] \otimes \Phi(\mathbf{s})\, d\mathbf{s} \quad \text{and} \quad \mathbf{B}^{(2)}_{ijk} = \int_{\mathcal{S}} \phi_i(\mathbf{s})\nabla^2\psi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s}. \tag{B.43}$$

Alternatively, the state space equation can be written affine in $\mathbf{x}(t)$:

$$\dot{\mathbf{x}}(t) = \mathbf{E}^{-1}\left[\gamma\mathbf{A} + (\mathbf{u}^\top \otimes \mathbf{I}_{N_x})(\mathbf{G}^{(1)} + \mathbf{G}^{(2)})\right]\mathbf{x}(t) \tag{B.44}$$

where

$$\mathbf{G}^{(1)} = \int_{\mathcal{S}} \left[\nabla\Psi(\mathbf{s})\nabla\Phi(\mathbf{s})^\top\right] \otimes \Phi(\mathbf{s})\, d\mathbf{s} \quad \text{and} \quad \mathbf{G}^{(1)}_{ijk} = \int_{\mathcal{S}} \nabla\psi_i(\mathbf{s})\nabla\phi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s}, \tag{B.45}$$

$$\mathbf{G}^{(2)} = \int_{\mathcal{S}} \left[\nabla^2\Psi(\mathbf{s})\Phi(\mathbf{s})^\top\right] \otimes \Phi(\mathbf{s})\, d\mathbf{s} \quad \text{and} \quad \mathbf{G}^{(2)}_{ijk} = \int_{\mathcal{S}} \nabla^2\psi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s}. \tag{B.46}$$

E Matrix

The derivation of the $\mathbf{E}$ matrix in canonical notation is as follows:

$$\begin{aligned} \mathbf{E}_{ij} &= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\, d\mathbf{s} \\ &= \int_{\mathcal{S}} \mathbf{S}_{ij}\ \phi_{ij}(\mathbf{s})\, d\mathbf{s} = \mathbf{S}_{ij}\int_{\mathcal{S}} \phi_{ij}(\mathbf{s})\, d\mathbf{s} \\ &= \mathbf{S}_{ij} \end{aligned} \tag{B.47}$$

$\mathbf{A}$ Matrix

The derivation of the $\mathbf{A}$ matrix in canonical notation is as follows:

$$\begin{aligned}
\mathbf{A}_{ij} &= \int_{\mathcal{S}} \phi_i(\mathbf{s})\nabla^2\phi_j(\mathbf{s})\,d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s}) \left\{\phi_j(\mathbf{s}) \left[(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right]\right\}\,d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s}) \left[(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right]\,d\mathbf{s} - \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s}) \left[\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right]\,d\mathbf{s} \\
&= \int_{\mathcal{S}} \mathbf{S}_{ij}\;\phi_{ij}(\mathbf{s}) \left[(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right]\,d\mathbf{s} - \int_{\mathcal{S}} \mathbf{S}_{ij}\;\phi_{ij}(\mathbf{s}) \left[\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right]\,d\mathbf{s} \\
&= \mathbf{S}_{ij}\int_{\mathcal{S}} \phi_{ij}(\mathbf{s})\,\mathrm{tr}\left[(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right]\,d\mathbf{s} - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right)\int_{\mathcal{S}} \phi_{ij}(\mathbf{s})\,d\mathbf{s} \\
&= \mathbf{S}_{ij}\int_{\mathcal{S}} \phi_{ij}(\mathbf{s})\,\mathrm{tr}\left[\boldsymbol{\Lambda}_j^2(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right]\,d\mathbf{s} - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right) \\
&= \mathbf{S}_{ij}\,\mathrm{tr}\left[\boldsymbol{\Lambda}_j^2\int_{\mathcal{S}} \phi_{ij}(\mathbf{s})(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\,d\mathbf{s}\right] - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right) \\
&= \mathbf{S}_{ij}\,\mathrm{tr}\left\{\boldsymbol{\Lambda}_j^2\left[\boldsymbol{\Lambda}_{ij}^{-1} + (\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right]\right\} - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right) \\
&= \mathbf{S}_{ij}\,\mathrm{tr}\left(\boldsymbol{\Lambda}_j^2\boldsymbol{\Lambda}_{ij}^{-1}\right) + \mathbf{S}_{ij}\,\mathrm{tr}\left[\boldsymbol{\Lambda}_j^2(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right] - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right) \\
&= \mathbf{S}_{ij}\,\mathrm{tr}\left(\boldsymbol{\Lambda}_j^2\boldsymbol{\Lambda}_{ij}^{-1}\right) + \mathbf{S}_{ij}(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \mathbf{S}_{ij}\left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right) \\
&= \mathbf{S}_{ij}\left[\mathrm{tr}\left(\boldsymbol{\Lambda}_j^2\boldsymbol{\Lambda}_{ij}^{-1}\right) + (\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\boldsymbol{\Lambda}_j^2(\boldsymbol{\Lambda}_{ij}^{-1}\boldsymbol{\eta}_{ij}-\boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \left(\mathbf{1}^\top\boldsymbol{\Lambda}_j\mathbf{1}\right)\right].
\end{aligned} \tag{B.48}$$

$\mathbf{B}^{(1)}$ Matrix

The derivation of the $\mathbf{B}^{(1)}$ matrix in canonical notation is as follows:

$$\begin{aligned}
\mathbf{B}^{(1)}_{ijk} &= \int_{\mathcal{S}} \nabla\phi_i(\mathbf{s})\nabla\phi_j(\mathbf{s})^\top \phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \left\{\phi_i(\mathbf{s}) \left[(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i\right]\right\} \left\{\phi_j(\mathbf{s}) \left[(\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j\right]\right\}^\top \phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i\right] \left[\mathbf{\Lambda}_j^\top (\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \int_{\mathcal{S}} \mathbf{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top (\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr} \left[(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top (\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr} \left[(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk}\, \mathrm{tr} \left[(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top) \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})(\mathbf{s} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk}\, \mathrm{tr} \left\{(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top) \left[\mathbf{\Lambda}_{ijk}^{-1} + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right]\right\} d\mathbf{s} \\
&= \mathbf{S}_{ijk}\, \mathrm{tr} \left(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathbf{S}_{ijk}\, \mathrm{tr} \left[(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] \\
&= \mathbf{S}_{ijk}\, \mathrm{tr} \left(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathbf{S}_{ijk}(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i) \\
&= \mathbf{S}_{ijk} \left[\mathrm{tr} \left(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top\mathbf{\Lambda}_{ijk}^{-1}\right) + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top(\mathbf{\Lambda}_i\mathbf{\Lambda}_j^\top)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\right].
\end{aligned} \tag{B.49}$$

$\mathbf{B}^{(2)}$ Matrix

The derivation of the $\mathbf{B}^{(2)}$ matrix in canonical notation is as follows:

$$\begin{aligned}
\mathbf{B}^{(2)}_{ijk} &= \int_{\mathcal{S}} \phi_i(\mathbf{s})\nabla^2\phi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s}) \left\{\phi_j(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right]\right\} \phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right] d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} - \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right] d\mathbf{s} \\
&= \int_{\mathcal{S}} \mathrm{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} - \int_{\mathcal{S}} \mathrm{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right] d\mathbf{s} \\
&= \mathrm{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr}\left[(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, d\mathbf{s} \\
&= \mathrm{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr}\left[\mathbf{\Lambda}_j^2(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right] d\mathbf{s} - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left[\mathbf{\Lambda}_j^2 \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s}-\mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top d\mathbf{s}\right] - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left\{\mathbf{\Lambda}_j^2 \left[\mathbf{\Lambda}_{ijk}^{-1} + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right]\right\} - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left(\mathbf{\Lambda}_j^2\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathrm{S}_{ijk}\, \mathrm{tr}\left[\mathbf{\Lambda}_j^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top\right] - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left(\mathbf{\Lambda}_j^2\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathrm{S}_{ijk}(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \mathrm{S}_{ijk} \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right) \\
&= \mathrm{S}_{ijk} \left[\mathrm{tr}\left(\mathbf{\Lambda}_j^2\mathbf{\Lambda}_{ijk}^{-1}\right) + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \mathbf{\Lambda}_j^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_j^{-1}\boldsymbol{\eta}_j) - \left(\mathbf{1}^\top\mathbf{\Lambda}_j\mathbf{1}\right)\right].
\end{aligned} \tag{B.50}$$

$\mathbf{G}^{(1)}$ Matrix

The derivation of the $\mathbf{G}^{(1)}$ matrix in canonical notation is as follows:

$$
\begin{aligned}
\mathbf{G}^{(1)}_{ijk} &= \int_{\mathcal{S}} \nabla\phi_i(\mathbf{s})\nabla\phi_j(\mathbf{s})^\top \phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \left\{\phi_i(\mathbf{s}) \left[(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \boldsymbol{\Lambda}_i\right]\right\} \left\{\phi_j(\mathbf{s}) \left[(\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top \boldsymbol{\Lambda}_j\right]\right\}^\top \phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \boldsymbol{\Lambda}_i\right] \left[\boldsymbol{\Lambda}_j^\top (\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \int_{\mathcal{S}} \mathbf{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top (\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s}) \operatorname{tr}\left[(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top (\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s}) \operatorname{tr}\left[(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \operatorname{tr}\left[(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top) \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})(\mathbf{s} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\mathbf{s} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] d\mathbf{s} \\
&= \mathbf{S}_{ijk} \operatorname{tr}\left\{(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top) \left[\boldsymbol{\Lambda}_{ijk}^{-1} + (\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right]\right\} d\mathbf{s} \\
&= \mathbf{S}_{ijk} \operatorname{tr}\left(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top\boldsymbol{\Lambda}_{ijk}^{-1}\right) + \mathbf{S}_{ijk} \operatorname{tr}\left[(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] \\
&= \mathbf{S}_{ijk} \operatorname{tr}\left(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top\boldsymbol{\Lambda}_{ijk}^{-1}\right) + \mathbf{S}_{ijk}(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i) \\
&= \mathbf{S}_{ijk} \left[\operatorname{tr}\left(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top\boldsymbol{\Lambda}_{ijk}^{-1}\right) + (\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\eta}_j)^\top(\boldsymbol{\Lambda}_i\boldsymbol{\Lambda}_j^\top)(\boldsymbol{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\right].
\end{aligned}
\tag{B.51}
$$

$\mathbf{G}^{(2)}$ Matrix

The derivation of the $\mathbf{G}^{(2)}$ matrix in canonical notation is as follows:

$$
\begin{aligned}
\mathbf{G}^{(2)}_{ijk} &= \int_{\mathcal{S}} \nabla^2 \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \left\{\phi_i(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i) - \mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right]\right\} \phi_j(\mathbf{s})\phi_k(\mathbf{s})\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i) - \mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right]\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\right]\, d\mathbf{s} - \int_{\mathcal{S}} \phi_i(\mathbf{s})\phi_j(\mathbf{s})\phi_k(\mathbf{s}) \left[\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right]\, d\mathbf{s} \\
&= \int_{\mathcal{S}} \mathrm{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\right]\, d\mathbf{s} - \int_{\mathcal{S}} \mathrm{S}_{ijk}\ \phi_{ijk}(\mathbf{s}) \left[\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right]\, d\mathbf{s} \\
&= \mathrm{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr}\left[(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top \mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)\right]\, d\mathbf{s} - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, d\mathbf{s} \\
&= \mathrm{S}_{ijk} \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})\, \mathrm{tr}\left[\mathbf{\Lambda}_i^2(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right]\, d\mathbf{s} - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left[\mathbf{\Lambda}_i^2 \int_{\mathcal{S}} \phi_{ijk}(\mathbf{s})(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\mathbf{s}-\mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\, d\mathbf{s}\right] - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left\{\mathbf{\Lambda}_i^2\left[\mathbf{\Lambda}_{ijk}^{-1} + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right]\right\} - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left(\mathbf{\Lambda}_i^2\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathrm{S}_{ijk}\, \mathrm{tr}\left[\mathbf{\Lambda}_i^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\right] - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\, \mathrm{tr}\left(\mathbf{\Lambda}_i^2\mathbf{\Lambda}_{ijk}^{-1}\right) + \mathrm{S}_{ijk}(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\mathbf{\Lambda}_i^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i) - \mathrm{S}_{ijk}\left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right) \\
&= \mathrm{S}_{ijk}\left[\mathrm{tr}\left(\mathbf{\Lambda}_i^2\mathbf{\Lambda}_{ijk}^{-1}\right) + (\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i)^\top\mathbf{\Lambda}_i^2(\mathbf{\Lambda}_{ijk}^{-1}\boldsymbol{\eta}_{ijk} - \mathbf{\Lambda}_i^{-1}\boldsymbol{\eta}_i) - \left(\mathbf{1}^\top\mathbf{\Lambda}_i\mathbf{1}\right)\right].
\end{aligned}
\tag{B.52}
$$

References

[1] Heiko Hamann. *Swarm Robotics: A Formal Approach*. Cham, Switzerland: Springer International Publishing, 2018 (cit. on pp. 1, 7, 8).

[2] M. Gupta, D. Saxena, S. Kumari and D. Kaur. 'Issues and applications of swarm robotics'. In: *International Journal of Research in Engineering, Technology and Science* 4 (2016) (cit. on pp. 1, 7).

[3] MIT News. *Nanoparticles take a fantastic, magnetic voyage*. 2019 (cit. on p. 2).

[4] J. Bodner, F. Augustin, H. Wykypiel et al. 'The da Vinci robotic system for general surgical applications: a critical interim appraisal'. In: *Swiss Medical Weekly* 135 (2005), pp. 674–678 (cit. on p. 2).

[5] S. Martel, J-B. Mathieu, O. Felfoul et al. 'Automatic navigation of an untethered device in the artery of a living animal using a conventional clinical magnetic resonance imaging system.' In: *Applied Physics Letters* 90.114105 (2007), pp. 1–3 (cit. on pp. 2, 10).

[6] M. Muthana, A.J. Kennerley, R. Hughes et al. 'Directing cell therapy to anatomic target sites in vivo with magnetic resonance targeting'. In: *Nature Communications* 6.8009 (2015) (cit. on pp. 2, 11).

[7] N. Nothnagel, J. Rahmer, B. Gleich et al. 'Steering of magnetic devices with a magnetic particle imaging system'. In: *IEEE Transactions on Biomedical Engineering* 63.11 (2016) (cit. on pp. 2, 11).

[8] J.A. Hubbell and A. Chilkoti. 'Nanomaterials for drug delivery'. In: *Science* 337.6092 (2012), pp. 303–305 (cit. on p. 2).

[9] H. Ren and H. Banerjee. 'A preface in electromagnetic robotic actuation and sensing in medicine'. In: *Electromagnetic actuation and sensing in mediocal robotics* (2018), pp. 1–10 (cit. on pp. 2, 10).

[10] S. Ferrari, G. Foderaro, P. Zhu and T. Wettergren. ‘Distributed optimal control of multiscale dynamical systems: a tutorial’. In: *IEEE Control Systems Magazine* (2016) (cit. on pp. 2, 11, 12, 14, 36).

[11] G. Foderaro, P. Zhu, H. Wei, T.A. Wettergren and S. Ferrari. ‘Distributed optimal control of sensor networks for dynamic target tracking’. In: *IEEE Transactions on Control of Network Systems* 5.1 (2018), pp. 142–153 (cit. on pp. 2, 13).

[12] A. Pasternack. *To fight oil spills, an open-source swarm of robotic sailboats*. 2011 (cit. on p. 2).

[13] E. Sahin. ‘Swarm robotics: from sources of inspiration to domains of application’. In: *Swarm Robotics: Lecture Notes in Computer Science* 3342 (2005), pp. 10–20 (cit. on p. 8).

[14] J.K. Parrish and L. Edelstein-Keshet. ‘Complexity, pattern, and evolutionary trade-offs in animal aggregation’. In: *Science* 284.5411 (1999), pp. 99–101 (cit. on p. 8).

[15] A. Martinoli, K. Easton and W. Agassounon. ‘Modeling swarm robotic systems: a case study in collaborative distributed manipulation’. In: *International Journal of Robotics Research* 23.4 (2004), pp. 415–436 (cit. on p. 8).

[16] H. Hamann and H. Wörn. ‘A framework of space-time continuous models for algorithm design in swarm robotics’. In: *Swarm Intelligence* 2 (2008), pp. 209–239 (cit. on p. 8).

[17] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo. ‘Swarm robotics: a review from the swarm engineering perspective’. In: *Swarm Intelligence* 7 (2013), pp. 1–41 (cit. on p. 8).

[18] O. Zedadra, A. Guerrieri, N. Jouandeau et al. ‘Swarm intelligence-based algorithms within IoT-based systems: a review’. In: *Journal of Parallel and Distributed Computing* 122 (2018), pp. 173–187 (cit. on p. 9).

[19] K.M. Passino. ‘Biomimicry of bacterial foraging’. In: *IEEE Control Systems Magazine* 22.3 (2002), pp. 52–67 (cit. on p. 9).

[20] P.E. Dupont, P. Vartholomeos and C. Bergeles. *Magnetically actuated multiscale medical robots*. 2012 (cit. on pp. 9, 10).

[21] C. Bergeles, M.P. Kummer, B.E. Kratochvil, C. Framme and B.J. Nelson. ‘Steerable intravitreal inserts for drug delivery: in vitro and ex vivo mobility experiments’. In: *MICCAI* (2011) (cit. on p. 9).

[22] P. Pouponneau, J-C. Leroux, G. Soulez, L. Gaboury and S. Martel. ‘Co-encapsulation of magnetic nanoparticles and doxorubicin into biodegradable microcarriers for deep tiessue targeting by vascular MRI navigation’. In: *Biomaterials* 32 (2011), pp. 3481–3486 (cit. on p. 9).

[23] B.J. Nelson, I.K. Kaliakatsos and J.J. Abbott. ‘Microrobots for minimally invasive medicine’. In: *Annual Review of Biomedical Engineering* 12 (2010), pp. 55–85 (cit. on pp. 9, 10).

[24] S. Chowdhury, W. Jing and D.J. Cappelleri. ‘Controlling multiple microrobots: recent progress and future challenges’. In: *Journal of Micro-Bio Robot* (2015) (cit. on p. 10).

[25] J.J. Abbott, Z. Nagy, F. Beyeler and B.J. Nelson. ‘Robotics in the Small, Part 1: Microrobotics’. In: *IEEE Robotics and Automation Magazine* 14 (2007), pp. 92–103 (cit. on p. 10).

[26] S. Hauert, J.H. Lo, A.D. Warren and S.N. Bhatia. ‘Magnetic actuation in the crowd-sourcing of swarm treatments’. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2012 (cit. on p. 10).

[27] C. Park, J. Kim, S-J. Kim and J. Yoo. ‘Development of a permanent magnet type microrobot actuated by external electromagnetic system’. In: *Microsystem Technology* 21 (2015), pp. 1257–1265 (cit. on p. 10).

[28] A. Becker, O. Felfoul and P.E. Dupont. ‘Simultaneously powering and controlling many actuators with a clinical MRI scanner’. In: *IROS*. 2014 (cit. on p. 10).

[29] S. Martel, M. Mohammadi, O. Felfoul, Z. Lu and P. Pouponneau. ‘Flagellated magnetotactic bacteria as controlled MRI-trackable propulsion and steering systems for medical nanorobots operating in the human microvasculature’. In: *International Journal of Robotics Research* 28.4 (2009), pp. 571–582 (cit. on p. 10).

[30] S. Martel, O. Felfoul, J-B. Mathieru et al. ‘MRI-based medical nanorobotic platform for the control of magnetic nanoparticles and flagellated bacteria for target interventions in human capillaries’. In: *International Journal of Robotics Research* 28.9 (2009), pp. 1169–1182 (cit. on p. 10).

[31] X. Dong and M. Sitti. ‘Collective formation and cooperative function of a magnetic microrobot swarm’. In: *Robotics: Science and Systems*. 2019 (cit. on p. 10).

[32] A. Becker. ‘Magnetic techniques to control many robots simultaneously’. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2012 (cit. on p. 10).

[33] A. Chanu and S. Martel. 'Co-encapsulation of magnetic nanoparticles and doxorubicin into biodegradable microcarriers for deep tissue targeting real-time software platform design for in-vivo navigation of a small ferromagnetic device in a swine carotid artery using a magnetic resonance imaging system'. In: *Conference of IEEE Engineering in Medicine and Biology Society*. 2007 (cit. on p. 11).

[34] S. Martel. 'Beyond imaging: macro- and microscale medical robots actuated by clinical MRI scanners'. In: *Science Robotics* 2 (2017) (cit. on p. 11).

[35] K. Belharet, D. Folio and A. Ferreira. 'MRI-based microrobotic system for the propulsion and navigation of ferromagnetic materials'. In: *Minimally Invasive Therapy* 19 (2010), pp. 157–169 (cit. on p. 11).

[36] K. Belharet, D. Folio and A. Ferreira. 'Simulation and planning of a magnetically actuated microrobot navigating in the arteries'. In: *IEEE Transactions on Biomedical Engineering* 60.4 (2013), pp. 994–1001 (cit. on p. 11).

[37] G. Foderaro and S. Ferrari. 'Necessary conditions for optimality for a distributed optimal control problem'. In: *49th IEEE Conference on Decision and Control*. 2010 (cit. on pp. 11, 14).

[38] K. Rudd, G. Foderaro and S. Ferrari. 'A generalized reduced gradient method for the optimal control of multiscale dynamical systems'. In: *52nd IEEE Conference on Decision and Control*. 2013 (cit. on pp. 11, 14).

[39] G. Foderaro, S. Ferrari and T. Wettergren. 'Distributed optimal control for multi-agent trajectory optimization'. In: *Automatica* (2014) (cit. on p. 13).

[40] K. Rudd and S. Ferrari. 'A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks'. In: *Neurocomputing* 155 (2015), pp. 277–285 (cit. on p. 14).

[41] C. Sinigaglia, A. Manzoni, F. Braghin and S. Berman. 'Indired optimal control of advection-diffusion fields through distributed robotic swarms'. In: *IFAC* (2022), pp. 299–304 (cit. on p. 15).

[42] K. Elamvazhuthi and S. Berman. 'Optimal control of stochastic coverage strategies for robotic swarms'. In: *IEEE International Conference on Robotics and Automation* (2015) (cit. on p. 15).

[43] S. Nouwens, M. Paulides and W. Heemels. 'Accelerating soft-constrained MPC for linear systems through online constraint removal'. In: *IEEE International Conference on Decision and Control* (2023), pp. 4273–4278 (cit. on p. 15).

[44] J. Rawlings and C. Maravelias. 'Bringing new technologies and approaches to the operation and control of chemical process systems'. In: *AIChE Journal* 65.6 (2019) (cit. on p. 15).

[45] S. Vazquez, J. Rodriguez, M. Rivera, L. Franquelo and M. Norambuena. 'Model predictive control for power converters and drives: advances and trends'. In: *IEEE Transactions on Industrial Electronics* 64.2 (2017) (cit. on p. 15).

[46] P. Kergus. 'Data-driven control of infinite dimensional systems: application to a continuous crystallizer'. In: *IEEE Control System Letters* 5 (2020) (cit. on p. 15).

[47] R. Curtain and H. Zwart. *Introduction to Infinite-Dimensional Systems Theory*. Cham, Switzerland: Springer International Publishing, 2020 (cit. on p. 15).

[48] T. Rabczuk and K. Bathe. *Machine Learning in Modeling and Simulation : Methods and Applications*. Cham, Switzerland: Springer International Publishing, 2023 (cit. on p. 15).

[49] K. Ricardo, K. Duru and D. Lee. 'An entropy stable discontinuous Galerkin method for the spherical thermal shallow water equations'. In: *SIAM Journal on Scientific Computing* 46 (2024) (cit. on p. 15).

[50] Z. Liang, H. Gao, X. Xin, S. Wang and Z. Peng. 'A mixed discretization scheme for discontinuous Galerkin domain decomposition method applied to surface integral equations'. In: *IEEE Antennas and Wireless Propagation Letters* (2024) (cit. on p. 15).

[51] Y. Koren and J. Borenstein. 'Potential field methods and their inherent limitations for mobile robot navigation'. In: *IEEE International Conference on Robotics and Automation* (1991) (cit. on p. 19).

[52] P. Aram, V. Kadirkamanathan and S. R. Anderson. 'Spatiotemporal system identification with Continuous-spatial-maps and sparse estimation'. In: *IEEE Transactions on Neural Networks and Learning Systems* 26.11 (2015), pp. 2978–2983 (cit. on pp. 20, 33).

[53] C. Bishop. *Pattern Recognition and Machine Learning*. Cham, Switzerland: Springer International Publishing, 2006 (cit. on pp. 24, 33).

[54] D. Johnson and S. Sinanovic. 'Symmetrizing the Kullback-Leibler distance'. In: *IEEE Transactions on Information Theory* (2001) (cit. on p. 30).

[55] M. Kelly. 'An introduction to trajectory optimization: how to do your own direct collocation'. In: *SIAM Review* 59.4 (2017), pp. 894–904 (cit. on p. 56).

[56] E. Kerrigan, Y. Nie, O. Faqir et al. 'Direct transcription for dynamic optimization: a tutorial with a case study on dual-patient ventilation during the COVID-19 pandemic'. In: *IEEE International Conference on Decision and Control* (2020) (cit. on p. 56).

[57] Y. Nie, O. Faqir and E. Kerrigan. 'ICLOCS2: Try this optimal control problem solver before you try the rest'. In: *2018 UKACC 12th International Conference on Control (CONTROL)* (2018) (cit. on pp. 56, 61).

[58] J. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. SIAM, 2010 (cit. on pp. 56, 59).

[59] A. Wachter and L. Biegler. 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming'. In: *Mathematical Programming* 106 (2006), pp. 25–27 (cit. on p. 56).

[60] W. Hager, H. Hou, S. Mohapatra, A. Rao and X. Wang. 'Convergence rate for a radau hp collocation method applied to constrained optimal control'. In: *Computational Optimization and Applications* 74 (2019), pp. 275–314 (cit. on p. 58).

[61] S. Kameswaran and L. Biegler. 'Convergence rates for direct transcription of optimal control problems using collocation at Radau points'. In: *Computational Optimization and Applications* 41 (2008), pp. 81–126 (cit. on p. 58).

[62] P. Bromiley. 'Products and convolutions of Gaussian probability density functions'. In: *School of Medicine, University of Manchester* (2014) (cit. on p. 93).