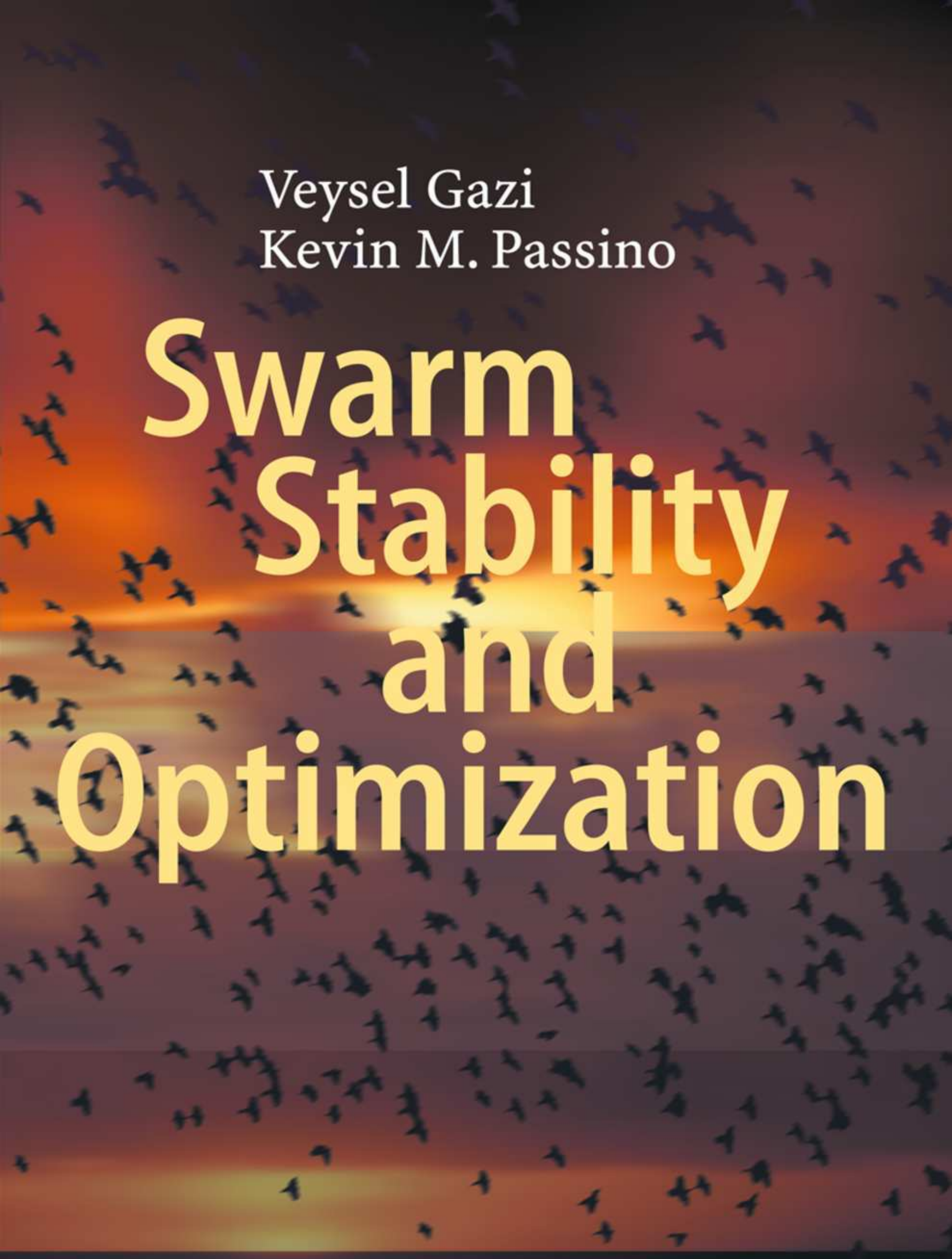


Veysel Gazi
Kevin M. Passino

Swarm Stability and Optimization

Springer

Swarm Stability and Optimization

Veysel Gazi and Kevin M. Passino

Swarm Stability and Optimization

Springer

Authors

Dr. Veysel Gazi
Firuzkoy Mahallesi,
Dervis Sokak No: 10,
Avcilar, Istanbul,
Turkey
E-mail: gazi.2@osu.edu

Prof. Kevin Michael Passino
The Ohio State University
Department of Electrical and
Computer Engineering
416 Dreese Laboratories
2015 Neil Ave
Columbus, OH 43210
USA
E-mail: passino.1@osu.edu

ISBN 978-3-642-18040-8 e-ISBN 978-3-642-18041-5

DOI 10.1007/978-3-642-18041-5

© Springer Science + Business Media B.V. 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Data supplied by the authors

Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To our parents ...

Preface

The beautiful and exotic collective behaviors of swarming species such as some bacteria, ant colonies, bee colonies, flocks of birds, schools of fish, and others, have fascinated and attracted the interest of researchers for many years. Collaborative swarming behavior that we observe in these groups provides survival advantages. The behavior results in what is sometimes called "collective intelligence" or "swarm intelligence" where groups of relatively simple and "unintelligent" individuals can accomplish very complex tasks using only limited local information and simple rules of behavior (e.g., coordinated group motion for predator avoidance). With the development of technology, including the technology on sensing, computation, information processing, power storage and others, it has become feasible to develop engineered autonomous multi-agent dynamic systems such as systems composed of multiple robots, satellites, or ground, air, surface, underwater or deep space vehicles (i.e., "vehicle swarms"). The studies on the dynamics and mechanisms of the behavior of swarms in nature can provide guidelines for developing principles for decentralized and cooperative coordination and control algorithms for engineering multi-agent dynamic systems. In other words, we can learn from the swarms in nature and utilize the resulting concepts and ideas in developing artificial engineering swarms (e.g., for vehicle swarms). This is the focus of this book.

Overview of the Book

In this book we have taken a systems theoretic perspective and investigate problems such as aggregation, social foraging, formation control, swarm tracking, distributed agreement, and engineering optimization inspired by swarm behavior. We have been concerned mostly with stability of the intended behavior, or controller development to achieve an intended behavior. We investigate the dynamic performance using analytic tools or simulations. We have used techniques from nonlinear control theory including Lyapunov stability, LaSalle's invariance principle, sliding mode control, output regulation, and from the literature on parallel and distributed computation. After introducing the above mentioned problems in Chapter 1 and Chapter 2 in Part I we have grouped the investigations in three main groups: (i) swarms composed of agents with dynamics evolving in continuous time (Part II); (ii) swarms composed of agents with dynamics evolving in discrete time (Part III); (iii) swarm based optimization techniques (Part IV).

Part II, which constitutes the part on continuous time swarms, consists of five chapters. The first chapter (Chapter 3) investigates swarms composed of agents with single integrator dynamics which is the simplest case. We develop a potential functions based approach in this chapter which serves as a proof of concept and paves the road for developing some of the results on swarms composed of agents with more complex dynamics. The second chapter on continuous time swarms (Chapter 4) investigates the stability and dynamics of swarms composed of double integrator agents under sensing uncertainties. Then in Chapter 5 we consider swarms composed of fully actuated robotic agents and build on the potential functions based results for swarms of single integrator agents. The agent dynamics are allowed to also contain model uncertainties and disturbances and the sliding mode control technique (known for its robustness characteristics) is used to suppress the uncertainties and achieve the desired behavior. In Chapter 6, on the other hand, swarms composed of robotic agents with non-holonomic dynamics are investigated under model uncertainties and disturbances and results are obtained using once more the sliding mode control technique. A completely different approach is taken in Chapter 7 where the formation control problem is investigated in swarms composed of agents with general nonlinear dynamics within the framework of output regulation in nonlinear systems.

Part III consists of three chapters. Chapter 8 investigates the stability and dynamics of a one dimensional swarm with nonlinear inter-agent interactions and time delays in synchronous and asynchronous settings. The analysis of the asynchronous case is done using tools from the literature on parallel and distributed computation. Chapter 9, on the other hand, extends the analysis to the problem of distributed agreement, which is referred to also as the consensus problem by some authors in the literature, and shows that the agents will achieve agreement despite only local unidirectional interactions, dynamically changing topology, time delays in information sensing, and asynchronous operation. Chapter 10 returns to the problem of formation control and develops a strategy for path (way-point) planning for autonomous agents to achieve a desired geometric formation in discrete time setting. The procedure developed is based on potential functions and Newton's iteration.

Part IV presents two swarm based optimization techniques which have been very popular in recent years. Chapter 11 presents the Bacteria Foraging Optimization algorithm, which is an optimization technique inspired by the foraging behavior of bacteria colonies enhanced with evolutionary events. Chapter 12, on the other hand, presents the Particle Swarm Optimization algorithm in which the particles search the space in parallel for the optimal solution based on their current momentum, past experience, and the experience of their neighbors. Substantial evidence has been collected in the literature showing the effectiveness of these optimization algorithms.

Intended Audience

Multi-agent dynamic systems have many potential commercial and defense applications including pollution clean up, search and rescue operations, firefighter assistance, surveillance, demining operations, and others. These applications range in many different areas such as agriculture (for cultivation or applying pesticides for protection), forestry (for surveillance and early detection of forest fires), in disaster areas (such as for search in areas with radioactive release after a nuclear disaster), border patrol and homeland security, search and coverage in warehouses under fire, fire distinction, health care, etc. There are enormous investments around the world in developing practical engineering autonomous multi-agent dynamic systems. These investments are both in the line of developing the related hardware for the autonomous agents and in the line of developing efficient, robust, flexible, and scalable decentralized coordination and control algorithms with guaranteed performance. This book presents results obtained within the efforts in this line of research. It can be used as textbook for a course on multi-agent dynamic systems. It can be used also as a supplementary book in courses on nonlinear control systems or optimization or discrete-time control systems. It can serve as a reference book for graduate students, researchers and engineers performing studies and or implementation work in the area of multi-agent dynamic systems from variety of fields including robotics, electrical and computer engineering, mechanical and mechatronics engineering, aerospace engineering, computer science, physics, biology, or even chemistry.

Acknowledgments

Financial support has been provided by various sources during the research reported in this book. First of all Veysel Gazi would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for awarding him with a nine month scholarship for studying towards PhD degree at the Ohio State University in Columbus, Ohio, USA. Other organizations which have provided funding include the National Institute of Standards and Technology (NIST) of USA through the “Real-Time Control Systems Software Library - Documentation, Implementation, Graduate level Laboratory Course Development, and Control and Optimization Routines Development” project, the Dayton Area Graduate Studies Institute (DAGSI) and the Air Force Research Laboratories (AFRL) at the Wright-Patterson Air Force Base in Dayton, Ohio, USA through the “Distributed Coordination

and Control of Autonomous Air Vehicles" project, and the Mixed Initiative Control of Autonoma-teams (MICA) program of the Defense Advanced Research Project Agency (DARPA) of USA in the context of the "Strategies for Human-Automaton Resource Entity Deployment (SHARED)" project. Moreover, financial support has been provided by TÜBİTAK in the context of the research projects "Robot Swarms: Coordination and Formation Control of Groups" and "Adaptive Swarm Control" and the CAREER project/award "Particle Swarm Optimization: Parallel and Asynchronous Implementation, Rigorous Convergence Analysis, and Implementation on a Multi-Robot System" awarded to Veysel Gazi. Further financial support has been provided also by the European Commission under the Specific Targeted Research Project (STREP) entitled "GUARDIANS: Group of Unmanned Assistant Robots Deployed In Aggregative Navigation supported by Scent detection." We would like to thank all these agencies and their staff for their positive attitude and support during the development/research phase of this book.

Throughout the research presented in this book we have collaborated and have done joint work with several researchers around the world. We would like to thank Dr. Yang Liu and Dr. Yanfei Liu who have contributed part of their work completed during their studies at the Ohio State University. Also we have collaborated with Dr. Ahmet Buğra Koku and our student Andaç Töre Şamiloğlu from Middle East Technical University Department of Mechanical Engineering (Ankara, Turkey), Dr. Raúl Ordóñez and his student Jingyi Yao from University of Dayton Department of Electrical and Computer Engineering (Dayton, Ohio, USA), Dr. Barış Fidan from University of Waterloo Department of Mechanical and Mechatronics Engineering (Waterloo, Canada), Dr. Nilay Sezer-Uzol from TOBB University of Economics and Technology Department of Mechanical Engineering, Dr. Bülent Tavlı from TOBB University of Economics and Technology Department of Computer Engineering, Dr. Vaios Lappas and Dr. Chakravarthini Manimekhala Saaj from University of Surrey - Surrey Space Centre (Guildford, UK). We would like to thank all the partners from the Guardians project and in particular the coordinator Dr. Jacques Penders and Dr. Lyuba Alboul from Sheffield Hallam University Materials and Engineering Research Institute, Dr. Lino Marques from University of Coimbra Institute for Systems and Robotics and his students, Dr. Enric Cervera and Dr. Raul Marin from University of Castellon Department of Computer Science, and Dr. Ulf Witkowski from University of Paderborn Heintz Nixdorf Institute. We would like to also thank Dr. Vadim I. Utkin and Dr. Andrea Serrani from the Ohio State University Department of Electrical and Computer Engineering (Columbus, Ohio, USA) and Dr. Manfredi Maggiore from University of Toronto Department of Electrical and Computer Engineering (Toronto, Ontario, Canada) for valuable discussions, some of which have inspired parts of this book.

Usually graduate students play a central role in performing scientific or engineering research and there are students who have directly or indirectly contributed to the book. Here we would like to tank Murat İlter Köksal, Yusuf Sinan Hanay, Halil Volkan Hünerli, Salih Burak Akat, Yunus Ataş, Ömer Çayırpunar, Mirbek Turduev, Engin Karataş, Esma Gül, Sabahat Duran, Özgür Pekçağlıyan, Abdel-Razzak Merheb, Şadi Çağatay Öztürk, Tuna Bulut, and Murat Kırtay who have been members of the Swarm Systems Research Laboratory at TOBB University of Economics and Technology (Ankara, Turkey).

We would like to also thank the publishers at Springer Verlag and in particular Dr. Thomas Ditzinger, for all their support in publishing this book. Finally, we would like to thank our families for their endless love and support during the research and publishing phases of this book.

Istanbul, Turkey, October 2010 Veysel Gazi
Columbus, Ohio, USA, October 2010 Kevin M. Passino

Contents

Part I: Basic Principles

1 Introduction 3
1.1 Swarms in Biology and Engineering 3
1.2 For Further Reading 9
1.2.1 Early Works in Biology 9
1.2.2 Early Works in Physics 11
1.2.3 Early Works in Engineering and Computer Science 12

2 Swarm Coordination and Control Problems 15
2.1 Aggregation 15
2.2 Social Foraging 17
2.3 Formation Control and Swarm Tracking 18
2.4 Distributed Agreement 20
2.5 Function Minimization 22
2.6 Related Problems and Behaviors and Further Reading 23
2.6.1 Formation Structure Breakdowns 23
2.6.2 Flocking 23
2.6.3 Rendezvous 24
2.6.4 Synchronization of Coupled Nonlinear Oscillators 24
2.6.5 Other Behaviors 25

Part II: Continuous Time Swarms

3 Swarms of Single Integrator Agents 29
- 3.1 Single Integrator Agent Model 29
- 3.2 Aggregation 30
  - 3.2.1 Potential Function Design 30
  - 3.2.2 Analysis of Swarm Motion 32
  - 3.2.3 Swarm Cohesion Analysis 35
  - 3.2.4 Agents with Finite Body Size 41
  - 3.2.5 Simulation Examples 44
- 3.3 Social Foraging 46
  - 3.3.1 Introduction 46
  - 3.3.2 Swarm Cohesion Analysis 48
  - 3.3.3 Swarm Motion in Various Resource Profiles 51
  - 3.3.4 Analysis of Individual Behavior in a Cohesive Swarm 59
  - 3.3.5 Simulation Examples 60
- 3.4 Formation Control 64
  - 3.4.1 Simulation Examples 66
- 3.5 Swarm Tracking 67
  - 3.5.1 Simulation Examples 71
- 3.6 Further Issues 72
  - 3.6.1 General Neighborhood Topology 72
  - 3.6.2 Non-reciprocal Agent Interactions 73
  - 3.6.3 For Further Reading 73

4 Swarms of Double Integrator Agents 75
- 4.1 Double Integrator Model 75
- 4.2 Stability Analysis of Swarm Cohesion Properties 76
  - 4.2.1 Controls and Error Dynamics 77
  - 4.2.2 Cohesive Social Foraging with Noise 81
  - 4.2.3 Special Case: Identical Agents 93
- 4.3 Stability Analysis of Swarm Trajectory Following 99
- 4.4 Simulation Examples 102
  - 4.4.1 No-Noise Case 102
  - 4.4.2 Noise Case 102
- 4.5 Further Issues 105
  - 4.5.1 Extensions and Generalizations 105
  - 4.5.2 For Further Reading 105

5 Swarms of Fully Actuated Agents with Model Uncertainty ... 107
5.1 Fully Actuated Agent Model with Uncertainty ... 107
5.2 Controller Development ... 108
5.2.1 Aggregation, Foraging, and Formation Control ... 108
5.2.2 Swarm Tracking ... 111
5.3 Potential Functions and Bounds ... 114
5.3.1 Aggregation ... 114
5.3.2 Social Foraging ... 115
5.3.3 Formation Control ... 117
5.3.4 Swarm Tracking ... 117
5.4 Simulation Examples ... 118
5.4.1 Aggregation ... 119
5.4.2 Social Foraging ... 120
5.4.3 Formation Control ... 121
5.4.4 Swarm Tracking ... 122
5.5 Further Issues ... 123
5.5.1 Extensions and Generalizations ... 123
5.5.2 For Further Reading ... 123

6 Swarms of Non-holonomic Unicycle Agents with Model Uncertainty ... 125
6.1 Non-holonomic Unicycle Agent Model with Uncertainty ... 125
6.2 Controller Development ... 126
6.2.1 Aggregation, Foraging, and Formation Control ... 126
6.2.2 Swarm Tracking ... 131
6.3 Potential Functions and Bounds ... 134
6.3.1 Aggregation ... 134
6.3.2 Social Foraging ... 137
6.3.3 Formation Control ... 139
6.3.4 Swarm Tracking ... 139
6.4 Simulation Examples ... 141
6.4.1 Aggregation ... 142
6.4.2 Social Foraging ... 144
6.4.3 Formation Control ... 144
6.4.4 Swarm Tracking ... 145
6.5 Further Issues ... 146
6.5.1 Model Equivalence ... 146
6.5.2 Extensions and Generalizations ... 148
6.5.3 For Further Reading ... 148

7 Formation Control Using Nonlinear Servomechanism ... 151
7.1 General Non-linear Agent Model ... 151
7.2 Nonlinear Servomechanism Based Controller Development ... 154

7.2.1 Full Information Controller 155
7.2.2 Error Feedback Controller 156
7.3 Formation Reconfiguration 158
7.3.1 Expansion/Contraction 160
7.3.2 Rotation 160
7.3.3 Topology Change 161
7.4 Illustrative Examples 161
7.5 Further Issues 170
7.5.1 Extensions and Generalizations 170
7.5.2 For Further Reading 172

Part III: Discrete Time Swarms

8 One-Dimensional Discrete-Time Swarm 177
8.1 The Swarm Model 177
8.1.1 Agent Model 177
8.1.2 One-Dimensional Swarm Model 178
8.2 The System under Total Synchronism 183
8.3 Asynchronous Swarm 189
8.4 Simulation Results 193
8.5 Further Issues 195
8.5.1 Extensions and Generalizations 195
8.5.2 For Further Reading 195

9 Asynchronous Distributed Agreement in Discrete Time Swarms 197
9.1 Model of the System 197
9.2 System under Total Synchronism 200
9.3 Asynchronous System 204
9.4 Simulation Examples 208
9.5 Further Issues 213
9.5.1 Application Areas of the Agreement Model 213
9.5.2 Extensions and Generalizations 214
9.5.3 For Further Reading 216

10 Formation Control with Potential Functions and Newton's Iteration 219
10.1 Path Planning for a Single Agent 219
10.2 Controller Development 223
10.3 Simulation Examples 227
10.4 Further Issues 230
10.4.1 Extensions and Generalizations 230
10.4.2 For Further Reading 230

Part IV: Swarm Based Optimization Methods

11 Bacteria Foraging Optimization 233
- 11.1 Bacterial Foraging by *E. coli* 233
  - 11.1.1 Swimming and Tumbling 233
  - 11.1.2 Chemotaxis and Climbing Nutrient Gradients 235
  - 11.1.3 Underlying Sensing and Decision-Making Mechanisms 236
  - 11.1.4 Elimination and Dispersal Events 237
- 11.2 *E. coli* Bacterial Foraging for Optimization 238
  - 11.2.1 An Optimization Model for *E. coli* Bacterial Foraging 238
  - 11.2.2 Bacterial Foraging Optimization Algorithm (BFOA) 242
  - 11.2.3 Guidelines for Algorithm Parameter Choices 244
  - 11.2.4 Relations to the Genetic Algorithm 245
- 11.3 Example: Function Optimization via *E. coli* Foraging 245
  - 11.3.1 Nutrient Hill-Climbing: No Swarming 246
  - 11.3.2 Swarming Effects 247
- 11.4 Further Issues 249
  - 11.4.1 Extensions and Generalizations 249
  - 11.4.2 For Further Reading 249

12 Particle Swarm Optimization 251
- 12.1 Basic PSO Iteration 251
- 12.2 Synchronous and Asynchronous PSO Implementations 253
  - 12.2.1 Traditional PSO Implementations 253
  - 12.2.2 Decentralized Asynchronous PSO Formulation 255
- 12.3 Particle Neighborhoods 257
  - 12.3.1 Static Neighborhood 257
  - 12.3.2 Dynamic Neighborhood 259
  - 12.3.3 Directed Graph Representation of Particle Neighborhoods 264
  - 12.3.4 Connectivity of the Swarm 266
- 12.4 Simulation Examples 270
- 12.5 Further Issues 275
  - 12.5.1 Extensions and Generalizations 277
  - 12.5.2 For Further Reading 278

References 281

Index 295

Part I

Basic Principles

1

Introduction

1.1 Swarms in Biology and Engineering

In engineering, the terminology of "swarms" has come to mean a set of agents possessing independent individual dynamics but exhibiting intimately coupled behaviors and collectively performing some task. Another terminology to describe such systems is the term "multi-agent dynamic systems." Examples of engineering swarms include autonomous ground, air, underwater or surface vehicles, satellites or deep space vehicles performing a cooperative task such as surveillance or monitoring of an area, extracting the map of an area, searching for an object, cultivating crop fields, cleaning mines, collectively carrying an object, etc. In biology, the terminology of swarms is reserved for certain species when they are in certain behavioral modes (e.g., honey bees after hive fission occurs and the swarm of bees is searching for, or flying to, a new home). There are also various synonymous expressions to swarms used in biology to describe various collective behavior of different species. Examples include "colonies" of ants or bees, "flocks" of birds, "schools" or "shoals" of fish, "herds" of antelopes, "packs" of wolf, etc. The coordinated behavior of such groups has fascinated and attracted the interest of researchers and scientists in biology and engineering for many years. From engineering perspective it is sometimes useful to use biological swarms as examples of behaviors that are achievable in multi-agent dynamic systems technologies (e.g., via a "bio-inspired design" approach). Moreover, operational principles from such biological systems can be used as guidelines in engineering for developing distributed cooperative control, coordination, and learning strategies for autonomous multi-agent dynamic systems. In other words, development of such highly automated systems is likely to benefit from biological principles including modeling of biological swarms, coordination strategy specification, and analysis to show that group dynamics achieve group goals. In contrast, from a biologist's perspective multi-agent (multi-vehicle) systems technologies have adjustable physical characteristics that can make them useful hardware simulation testbeds to emulate the dynamics of animal groups and hence help understand the mechanisms of animal group decision making. These close connections between biology and engineering call for an integrated view of swarms.

To develop a generic swarms perspective, one can first begin by defining agents (vehicles, animals, software agents) to have sensory capabilities (e.g., to sense position or velocity of other agents or sense environmental characteristics), processing ability (e.g., a brain or an on-board computer), the ability to communicate or exchange information (e.g., via direct communication or through the environment), and the ability to take actions via actuators (e.g., move to a location at a velocity or pick up some object and fix it). Sensor and actuator limitations (e.g., sensing range, bandwidth) and errors (e.g., sensor noise, sensing and/or communication delay), along with limited agent processing abilities (e.g., due to finite memory and computational throughput) make any agent error-prone. Physical agent characteristics (e.g., of a wheeled robot or a flying vehicle) along with agent motion dynamics (e.g., a fast or a slow agent) and its sensory and processing capabilities (e.g., comprehensive and long range sensing abilities and extensive computation capabilities or limited sensing and computation capabilities) constrain how the agent can move in its environment and the rates at which it can sense and act in spatially distributed areas and interact with the other agents. Note also that there might be various ways through which agents influence each other and exchange information or communicate (either directly or indirectly). In biology this may be via chemical communication (e.g., in bacteria or ants through pheromones) or signals (e.g., the waggle dance of the honey bee) or through the environment (e.g., by modifying the environment and therefore resulting in a change in the behavior of the other agents). In engineering it may be via an ad-hoc wireless network. Moreover, methodologies similar to those in biology can be developed as well. Regardless, it is useful to think of the agents as nodes, and arcs between nodes as representing abilities to sense or communicate with other agents. The existence of an arc may depend on the communication or sensing range of agents, the properties of the environment (e.g., walls may adversely effect sensing or communication, too many signals in the environment can result in signal collisions), communication network and link imperfections (e.g., noise and random delays), along with local agent abilities (e.g., an ability to only communicate with one other agent at a time) and goals (e.g., a desire to communicate with only the agents that can help it complete its task). One can view a multi-agent dynamic system as a set of such communicating agents that work collectively to solve a task.

By its nature, operation of swarm systems in biology is distributed and decentralized meaning that each agent/individual operates based on its own local (and limited) sensing and action rules without global knowledge or global planning. In other words, the sensing, computation, and cognitive capabilities of the individual agents are limited and they are unable to conceive neither the global emergent behavior of the swarm nor the large scale outcome of this behavior (e.g., the nest which is being built) and there is no global planner which dictates the individual agent behaviors. For example, it is known that insects such as ants or termites operate based on very simple rules which are guided by the chemicals (called pheromones) laid by the other individuals in the swarm/colony and the environmental cues (e.g., changes in the environment), and have limited capabilities. However, they can collectively build fascinating structures and achieve goals which individual insects are incapable of achieving alone. There is no centralized controller or global supervisor of the swarm. In engineering, in the case where the number of agents is very small it might be possible to develop centralized coordination strategies for a multi-agent system. However, centralized strategies result in decreased robustness and flexibility and increased computational complexity in the system. Moreover, as the number of agents increases, centralized algorithms become intractable. Therefore, similar to the case of swarms in nature, the common approach is to use decentralized coordination and control algorithms which operate based on local limited information.

Multi-agent robotic systems possess various potential advantages over single robot systems. First of all multi-robot systems are *more flexible* and they can re-adjust and re-organize (in space and time) based on the needs of the task under consideration, whereas single robotic systems do not have this capability. Multi-robot systems can *operate in parallel* (and different agents can concurrently perform different tasks) and therefore in a more efficient manner (provided that appropriate cooperation algorithms are developed), whereas in single agent robotic systems the robot usually has to finish (or interrupt) its current task before starting another task. Multi-agent robotic systems possess *improved robustness* properties since if one agent fails the other agents can continue (after re-organization and re-planning if needed) and complete the task, whereas for a single robotic system if the agent fails the task will fail as well. Moreover, since the robots in a multi-agent robotic system are usually simpler (in both hardware and algorithmic levels) they are *less prone to errors* such as development bugs (in both hardware and software). Furthermore, since the individual agents in a multi-robot system are usually very simple compared to a single complex and intelligent robot in a single robot system, they can be produced in mass at very *low-cost* and result in a cost-effective robotic system meaning that the total cost of a multi-robot system can be lower than the cost of a single complex and capable robot. Despite their simplicity, as in biological swarms, multi-agent robotic systems collectively are capable of performing complex tasks which are beyond the capabilities of any individual agent in the swarm. Moreover, multi-agent robotic systems can have *improved task/mission capabilities* compared to single robotic systems and perform tasks which are not achievable by a single (complex) robot. In other words, the set of tasks a multi-agent system can perform can be much larger than the set of tasks of a single agent. Moreover, the range of possible applications and areas of use of a given multi-agent system can be wider compared to those for a single robot system.

There have been several multi-agent system behaviors and task achievement goals that have been studied in the literature. For instance, coordinated motion has received significant recent attention in cooperative robotics (e.g., to make the agents to aggregate and stay in a tight group, achieve a spatial pattern or a geometric formation, or track a moving object) and biology (e.g., formation of fruiting bodies by bacteria, foraging behavior of ants, or cohesive flight of swarms of bees). Such problems are also closely related to a group reaching a consensus or agreement (since often preference can be linked to position). The problem of distributed agreement is another problem which has been popular in the multi-agent dynamic systems community. A substantial portion of this book is also concerned with coordinated motion or distributed agreement and develops various different techniques to achieve these goals. Coordinated motion and distributed agreement are, however, only particular multi-agent system objectives, and often are not the only possible objectives. There are other related problems such as task allocation or scheduling, executing spatially distributed tasks (e.g., cooperative mapping of a large area by multiple robots), or task execution under time or energy or other constraints. For example, task allocation often arises in multi-agent system problems where the "tasks" arise via interactions with the environment, and the tasks must be allocated across the agents and properly scheduled in time for efficient execution (e.g., via dynamic allocation of tasks to various robots). In this context, methods from distributed scheduling, load balancing, and assignment are integrated into coordinated motion methods to achieve a multi-objective method that must balance tight group cohesion with the pressing need to complete tasks (e.g., to search a large region). In problems where agents must be distributed across large regions to execute spatially distributed tasks (e.g., monitoring a large region with a small number of sensor-limited robots or providing particular service to customers distributed over a large region by a limited number of robots, or in biology when animals distribute themselves across spatially distributed food sources to maximize their feeding rates) similar strategies need to be deployed in order to achieve efficient coverage and the most beneficial outcome/performance. In such problems, the environmental conditions (such as the distribution/density of the food sources in biology or the frequency/density and spatial distribution of service requests in engineering) should be taken into account in the optimization process in addition to the agent dynamics. Moreover, time can play a critical role during or before a particular collective task. For example, in some applications a group of error-prone agents must work together to find and select, as fast as possible, the best task to perform. In such group choice problems there is a complex interplay between the need to search for more/better tasks and the need to come to agreement on which is the best of the discovered tasks. Generally, there is a speed-accuracy trade-off where if a choice is made fast then it is more error-prone. In contrast, if more time is allowed, agents can profitably combine their erroneous task quality estimates and fully search the space to ensure that a better task choice is made. However, the limited resources of the group may not allow for comprehensive search. There might be also applications in which in a heterogeneous group of autonomous agents with different sensing, computation, and actuation capabilities, an agent might search (through the communication network) for and request a service from another agent which has the capability to provide the needed service. Heterogeneous robot groups operating with efficient division of labor and service discovery techniques can result in effective multi-agent systems with high performance. In summary, in problems/situations such as these, significant attention must be given to achieving tasks that are not quantifiable using only standard inter-agent distance/velocity patterns as in conventional studies of coordinated motion. Task achievement demands that agent motion characteristics achieve the task and hence traditional inter-vehicle spacing and velocity objectives may need to be augmented (or replaced) with task oriented objectives.

Dynamics of multi-agent systems can exhibit rich and complex characteristics. Combining agent, sensing, and communication characteristics results in a multi-scale system with local spatio-temporal agent actions dynamically combining into an “emergent” global spatio-temporal pattern of group behavior. For a given set of error-prone agents and a task, there is a need to predict what behavioral pattern will exist for the multi-agent system to verify that the task will be achieved (especially in safety-critical applications). Moreover, for a given desired objective or behavior for the multi-agent system, there is a need to know how to design local agent characteristics in engineering (or how evolution adapted these in biology) so that the desired objective reliably emerges. These form two key theoretical research problems, and investigations on validation of correct behaviors of multi-agent systems are progressing rapidly for certain problems. For instance, methods from Lyapunov stability theory, which we use extensively in this book as well, have been quite useful to establish conditions on local agent dynamics so that appropriate coordinated motion and task allocation emerges. Statistical and simulation-based methods have also met with some success in spite of the complexity of the systems. On the other hand, there has been relatively little work on the establishment of an experimentally-validated multi-scale mathematical model of a biological swarm, one that can in some way also lead to analytical tractability and the subsequent elucidation of principles of species-generic swarm behavior in nature (e.g., how the mechanism of actions at the local level dynamically combine in agent-to-agent and agent-to-environment interactions for robust achievement of emergent behaviors in spite of error-prone agents).

There have been some studies in the swarm robotics community investigating behavior based and learning strategies which lead to emergence of cooperation in simple tasks such as pulling a stick from a hole in the ground, a task which individual agents cannot achieve on their own. The success of such initial studies provides stepping stones for more comprehensive investigations. It is possible to model and analyze swarm behavior from different perspectives. Two fundamentally different approaches which are worth mentioning are *spatial* and *non-spatial* approaches. The distinctive property of the spatial approach is that the space (the environment) is either explicitly or implicitly present in the model and the analysis. In contrast, in the non-spatial approaches the space (such as the position and surroundings of the agents) is not present in the model and the population level swarming dynamics are usually described in a non-spatial way in terms of frequency distributions of groups of various size. For example, one possible assumption is that groups of various sizes split or merge into other groups based on the inherent group dynamics (such as the density of the group), environmental conditions (which are also general and independent from the position/location in the space), and encounters of other groups.

The studies on modeling and analysis of swarm dynamics can be also classified from the perspective of representation of individual agents and overall swarm behavior and can be divided into two distinct frameworks which are *individual-based* (or Lagrangian) framework and *continuum* (or Eulerean) framework. In the individual-based models the basic description is the dynamic motion equation of each individual (or agent) and therefore it is a natural approach for modeling and analysis of complex social interactions and aggregations. Then, the resulting coordinated swarming behavior in its most basic form is due to the interplay between a long range attraction and a short range repulsion between the individuals (species or agents). Indeed, it is known that in nature for some species such as fish the mechanisms for aggregation and coordinated behavior can be described by such a model. However, there might be other factors affecting the coordinated motion of the agents such as the environment (which might affect the behavior of different agents in different manner) or task oriented goals which may also force different patterns or modes of behavior. In contrast, in the Eulerean framework the swarm dynamics are described using a *continuum model* of the *flux*, namely *concentration* or *population density* of the swarm and there is no distinct equation for the individual agent dynamics. In other words, in the Eulerean models individual agents and their dynamics are not important and the swarm is viewed as a continuum described by its density in one, two, or three dimensional space and its dynamics are described by spatio-temporal partial differential equations of its density. The basic equation of the Eulerean models is the *advection-diffusion-reaction* equation, where advection and diffusion are the joint outcome of individual behavior and environmental influences, and the reaction term is due to the population dynamics.

Each of the above approaches has its advantages and disadvantages and can represent different aspects of swarm behavior/dynamics. For example, in order to model and analyze the social foraging behavior it is useful to include both individual agent dynamics and the effect of the environment on the agent and the overall swarm dynamics. In contrast, in order to model the distributed agreement problem and the agreement mechanisms (which sometimes might be independent of the position of the agents and its absolute location in the environment) it might be sufficient to analyze only the dynamics of the agreement variable (or the relative differences of the agent “opinions” or “estimates” of the agreement variable) independent of the other variables/parameters. Moreover, if the behavior of every single individual is important or there is a need for independent controller development for every agent (which is usually the case in engineering) the individual based-models might be more suitable. In contrast, in the case in which only overall population dynamics and the swarm density are of primary interest (and the dynamics of the agents on individual level are unimportant/irrelevant) one can consider a continuum model. In this book we use individual/agent based swarm models. The effect of the environment is either present or not depending on the particular problem under consideration.

As mentioned above, the behavior and success of swarms in nature provides important inspiration for development of engineering multi-robot systems. Swarming behavior in nature can result from several different mechanisms. For example, individuals may respond directly to local physical cues such as concentration of nutrients or distribution of some chemicals (which may be laid by other individuals). This process is called *chemotaxis* and is used by organisms such as bacteria or social insects (e.g., by ants in trail following or by honey bees in cluster formation). As another example, individuals may respond directly to other individuals (rather than the cues they leave about their activities) as seen in some organisms such as fish, birds, and herds of mammals. Evolution of swarming behavior in nature is driven by the advantages of such collective and coordinated behavior for avoiding predators and increasing the chance of finding food. It has been demonstrated by biologists that social foragers perform chemotaxis more successfully as a group over noisy gradients than individually. In other words, individuals do much better collectively compared to the case when they forage on their own. In fact, for many organisms, swarming often occurs during "social foraging." Since foraging is in principle an optimization process, biomimicry of foraging behavior can sometimes result in powerful optimization algorithms. The bacteria foraging optimization and the particle swarm optimization presented in Part IV of this book as well as the ant colony optimization algorithm (which is not included here) are examples of optimization algorithms which have been inspired from the foraging behavior of various species and in particular, bacteria, birds, and ants, respectively.

The behavior of individuals (and therefore swarms) in nature usually is not deterministic. In other words, in addition to the error prone behavior, there is some randomness (a probabilistic component) in the operation of the swarms in nature. This results in multiple stable modes of behavior meaning that the emergent behavior of the swarm can converge to different persisting modes of operation. In addition to the current mostly deterministic studies, developing widely accepted and tractable models supported with corresponding analytical investigations for such swarm behavior will further enhance the literature on swarms.

Because of the interdisciplinary nature of the field, the studies on coordinated multi-agent dynamic systems have a moderately wide spectrum of perspectives. This book focuses on the system dynamics and control perspective. Noting that it by no means completely covers the topic, even from this particular perspective, the objective is to present important results on mathematical modeling, and coordination and control of multi-agent dynamical systems as well as swarm based optimization algorithms.

1.2 For Further Reading

There have been many relevant works in the field of swarms undertaken by researchers from different disciplines or ares of research including biology, physics, engineering, and computer science. Below we briefly point out to some of these works. A similar treatment can be found in [78, 87].

1.2.1 Early Works in Biology

Early works on understanding and modeling coordinated animal behavior as well as empirically verifying the developed/proposed models has been performed by biologists. The work in [105] classifies the work of biologists into the above mentioned *individual-based* (Lagrangian) and *continuum* (Eulerean) frameworks. Another work which presents a useful background and a review of the swarm modeling concepts and literature such as spatial and non-spatial models, individual-based versus continuum models and so on can be found in [181].

One of the early works within the *individual-based* framework is the work by Breder in [28], where the author suggested a simple model composed of a constant attraction term and a repulsion term which is inversely proportional to the square of the distance between two individuals. Similar work was performed by Warburton and Lazarus in [254] where the authors study also the effect of a family of attraction/repulsion functions on swarm cohesion. An example work within the *continuum* framework is the work in [172] where the authors present a swarm model which is based on non-local interactions of the individuals in the swarm. Their model consists of integro-differential advection-diffusion equations with convolution terms that describe attraction and repulsion. In [106], on the other hand, a general continuous model for animal group size distribution, which is a non-spatial patch model and constitutes an example work on non-spatial approaches, is presented. The authors consider a population with fixed size which is divided into groups of various dynamic sizes. For that purpose they introduce several assumptions about fusion and fission of groups of various sizes in order to describe and analyze the population dynamics.

Other works on model development for biological swarms by mathematical biologists include [102, 159]. The work by Grindrod in [102] is an effort to generate a model for (spatial) aggregation and clustering of species and consider its stability (i.e., its ability to preserve the swarm density). While [102] consider a continuum (in space) model of a swarm, the article in [159] describes a spatially discrete model. The authors show that their model can describe the swarming behavior, i.e., the aggregation of individuals in dense populations.

In [65] Durrett and Levin present a comparative study and compare four different approaches to modeling the dynamics of spatially distributed systems by using three different examples, each with different realistic biological assumptions. In particular they investigate the *mean field approaches* (in which every individual is assumed to have equal probability of interacting with every other individual), *patch models* (that group discrete individuals into patches without additional spatial structure), *reaction-diffusion equations* (in which infinitesimal individuals are distributed in space), and *interacting particle systems* (in which individuals are discrete and the space is treated explicitly). They show that the solutions of all the models do not always agree, and argue in favor of individual-based models that treat the space explicitly. In the work in [190] Parrish and her colleagues survey similarities and differences between different models of swarm aggregations and present preliminary results of efforts to unify all the models within a single framework. An important work to mention here is also the work by Grünbaum in [103, 104] where the author explains how social foragers perform chemotaxis over noisy gradients more successfully as a group than individually. In other words, the work shows that social foragers perform better than individual (selfish) foragers. Another related work is the article in [107] where the dynamic behavior of migrating herds is investigated by means of two dimensional discrete stochastic (or individual based) models. The authors use an individual based model with a hierarchical decision scheme involving short-range repulsion and long-range attraction together with density-independent and density-dependent decisions.

Comprehensive treatments of swarms in biology can be found in [189] and [34] and references therein. General useful references in biology are the books by Murray [178] and Edelshtein-Keshet [67]. The work in [226] presents foraging theory. Note that aggregation and collective behavior for many organisms usually occurs during social foraging. Moreover, foraging behavior of various species in biology constitutes inspiration for some biologically inspired optimization algorithms.

1.2.2 Early Works in Physics

There are related studies performed by physicists investigating swarming behavior. The general approach the physicists take is to model each individual as a particle, which they usually call a *self-driven* or *self-propelled particle*, and study the collective behavior due to their interaction. In particular, they analyze either the dynamic model of the density function or perform simulations based on a model for each individual particle. Some articles consider the Newton's equation of motion. However, this is not the only type of model they consider. Many researchers consider a discrete time model that assumes that particles are moving with constant absolute velocity and at each time step each one travels in the average direction of motion of the particles in its neighborhood with some random perturbation. Using such a model they try to study the effect of the noise on the collective behavior and to validate their models through extensive simulations.

In [200] Rauch et al. explored a simplified set of swarm models, which were driven by the collective motion of social insects such as ants. In their model the swarm members move in an energy field that models the nutrient or chemotactic profile in biology. They show that some interesting phenomena such as formation of stable lines of traffic flow emerge. In [239] Toner and Tu proposed a nonequilibrium continuum model for collective motion of large groups of biological organisms and later in [240] they develop a quantitative continuum theory of flocking. They show that their model predicts (models or represents) the existence of an *ordered phase* of flocks, in which all individuals in even arbitrarily large flocks move together.

In [41] a simple self-driven lattice-gas model for collective biological motion is introduced. The authors show the existence of a transition from individual random walks to collective migration. Similarly, Vicsek et al. in [251], which is a work which has caught attention of the engineering community in the recent years, introduce a simple simulation model for system of self-driven particles. They assume that particles are moving with constant absolute velocity and at each time step assume the average direction of motion of the particles in its neighborhood with some random perturbation. They show that high noise (and/or low particle density) leads to a no transport phase, where the average velocity is zero, whereas in low noise (and/or high particle density) the swarm is moving in a particular direction (that may depend on the initial conditions). They call this transition from a stationary state to a mobile state *kinetic phase transition*. Similarly, in [45] they present experimental results and mathematical model for forming bacterial colonies and collective motion of bacteria. The model is a simple self-propelled particle model that tries to capture the effect of nutrient diffusion, reproduction, extracellular slime deposition, chemoregulation, and inhomogeneous population. Other results in the same spirit include [44, 46, 47, 252]. In [46] a nonequilibrium model was compared to some equilibrium XY model in ferromagnets, in [44] the authors demonstrate similar results in one dimension, in [252] the effect of fluctuations on the collective motion of self-propelled particles is investigated, and in [47] the effect of noise and dimensionality on the scaling behavior of swarms of self-propelled particles is investigated.

Results of similar nature by different authors can be found in [146, 171, 224]. In [171] the authors consider a dynamic model of swarms of self-propelled particles with attractive long-range interactions. They show that the system can be found in either a coherent traveling state or an incoherent oscillatory state and that the increase in noise intensity leads to a transition from a coherent to oscillatory state. Similarly, in [224] the authors propose a model that represents several kinds of cluster motion observed in nature including collective rotation, chaos, and wandering. The article in [146] describes a model that exhibits coherent localized solutions in one and two dimensions. The solution of the model is of finite extent and the density drops sharply to zero at the edges of the swarm as in biological swarms. Moreover, they develop a continuum version of their discrete model and show that the two models agree.

1.2.3 Early Works in Engineering and Computer Science

The field of coordinated multi-agent dynamic systems has become popular in the past decade in the engineering community as well. One of the earliest works in the community is the work by Reynolds [206] on simulation of a flock of birds in flight using a behavioral model based on few simple rules and only local interactions. Since then the field has witnessed many developments some of which are presented in this book.

Early work on swarm stability is given by Beni and coworkers in [129] and [17]. In [129] they consider a synchronous distributed control method for discrete one and two dimensional swarm structures and prove stability in the presence of disturbances using Lyapunov methods. Then, in [17] they consider a *linear* swarm model and provide sufficient conditions for asynchronous convergence (without time delays) of the swarm to a synchronously achievable configuration.

In [201] Reif and Wang introduce the concept of *very large scale robotic* (VLSR) systems and consider a distributed control approach based on artificial force laws between individual robots and robot groups. The force laws are inverse-power or spring force laws incorporating both attraction and repulsion. The force laws can be distinct and to some degree they reflect the “social relations” among robots. Therefore, they call the method *social potential fields* method. Individual robot motion depends on the resultant artificial force imposed by the other robots and other components of the system such as obstacles. The approach is a distributed approach since each robot performs its own force and control calculations in a (possibly) asynchronous manner. It is one of the early works employing artificial potentials to specify inter-agent interactions. However, it is based only on simulations and no analytical investigations of the stability are performed.

Many of the early works on swarms have been limited to either one or two dimensional space. Note that in one dimension, the problem of collective motion without collisions is very similar to the problem of *platooning* of vehicles in *automated highway systems*. Automated highway systems and in particular platooning have been studied extensively in the literature. Example works include [16, 48, 231, 232]. Concepts related to swarm stability or formation stability from the literature on automated highway systems include *string stability* in one dimensions and *mesh stability* [188] in two dimensions.

As mentioned before search is an important task for multi-agent dynamic systems. Gelenbe et al. provide a survey of autonomous search strategies by robots and animals in [98]. They first review the literature on coordination and search by robots, then summarize the research in the field of animal search.

Coordinated motion and distributed formation control of robots are important problems in the multi-agent formation control literature. In systems under minimalistic assumptions it might be difficult to achieve even simple formations. In [230] the authors consider asynchronous distributed control and geometric pattern formation of multiple anonymous robots. The robots are anonymous in the sense that they all execute the same algorithm and they cannot be distinguished by their appearances. Moreover, the robots do not necessarily have a common coordinate system. The authors present an algorithm for moving the robots to a single point and also characterize the class of geometric patterns that the robots can form in terms of their initial configuration, and present some impossibility results.

There are many other important studies on cooperative control and coordination of swarms of agents and in particular formation control of autonomous air or land vehicles using various different approaches. Some examples include [14, 99, 196, 258]. In [258] the author considers cooperative control and coordination of a group of holonomic mobile robots to capture/enclose a target by making group formations. Results of a similar nature using behavior based strategy can be found also in [14], where the authors consider a strategy in which the formation behavior is integrated with other navigational behavior and present both simulation and implementation results for various types of formations and formation strategies. In [99], on the other hand, the authors describe formation control strategies for autonomous air vehicles. They use optimization and graph theory approach to find the best set of communication channels that will keep the aircraft in the desired formation. Moreover, they describe reconfiguration strategies in case of faults or loss of aircraft.

Other work on formation control and coordination of multi-agent (multi-robot) teams can be found in [13, 55, 56, 68, 145, 180]. In [55, 56] a feedback linearization technique using only local information for controller design to exponentially stabilize the relative distances of the robots in the formation is proposed. Similarly, in [68, 180], the concept of control Lyapunov functions together with formation constraints is used to develop a formation control strategy and prove stability of the formation (i.e., formation maintenance). The results in [145], on the other hand, are based on using virtual leaders and artificial potentials for robot interactions in a group of agents for maintenance of a predefined group geometry. By using the system kinetic energy and the artificial potential energy as a Lyapunov function closed loop stability is shown. Moreover, a dissipative term is employed in order to achieve asymptotic stability of the formation. In [13], the results in [145] are extended to the case in which the group is moving in a sampled gradient field. Similarly, the article in [131] investigates formation control in three dimensional space, whereas the article in [196] presents a cooperative search method for a group of agents using artificial potentials and based on the concept of rivaling force.

One should note here that not all types of formations may be possible. In other words, there may exist formations that may not be feasible given the system dynamics. The article in [233] describes a systematic framework for studying feasibility of formations for both undirected and directed type formations. Note also that many works utilize *artificial potential functions* to achieve coordinated motion. The concept of artificial potential functions is not new, and it has been used extensively for robot navigation and control [138, 139, 208]. The main difference of the present studies from the earlier studies such as those in [138, 139, 208] is that in the present studies artificial potentials are also employed to specify inter-agent interactions.

Developing models for swarming behavior is important in engineering not only because we can use them in developing swarms of autonomous agents, but also we can use these ideas in “controlling” natural flocks (herds, schools, swarms) of animals. An interesting example for this is the article in [249], where the authors develop a mobile robot that gathers a flock of ducks and maneuvers them safely to a specified goal position. They use a potential-field model of flocking behavior and using it investigate methods for generalized flock control (in simulation). Then they use the robot to control a real flock of ducks and show that the real world behavior of the ducks is similar to the expected one from simulations.

There are optimization methods inspired from the behavior of swarms in nature. Two of these methods, which are bacterial foraging optimization and particle swarm optimization, will be treated in more detail in this book. Another related and popular nature inspired optimization technique is “ant colony optimization” which is an optimization method based on foraging in ant colonies and is discussed in [25, 58, 60]. In the ant colony optimization algorithm the focus is on biomimicry for the solution of combinatorial optimization algorithms (e.g., shortest path algorithms).

Finally, we would like to mention that models of multi-agent systems with interacting particles may represent not only biological or engineering swarms, but also other systems and can be used in different engineering applications. The book [134], in addition to presenting the particle swarm optimization algorithm, discusses several different systems such as the operation of a brain that can be modeled as swarms of interacting agents.

2

Swarm Coordination and Control Problems

There are a number of different swarm coordination and control tasks that will be investigated in this book. In this chapter, we briefly introduce these problems. In particular, we present aggregation, social foraging, formation control, swarm tracking, and distributed agreement. We also present the problem of function minimization since we will approach it using biologically inspired direct search methods in the last part of this book. At the end of the chapter we also point out other tasks investigated in the multi-agent systems literature which are of immediate relevance to the tasks or behaviors considered in this book.

In order to state the problems we will assume a generic swarm and will denote the number of agents in the swarm with N. We will not specify the dynamic motion models of the agents (robots, vehicles, UAVs, etc.) in the swarm. However, we will assume that the dynamics of the variables of interest evolve in $\mathbb{R}^n$. The variables of interest, which we will denote with $x_i(t),\ i = 1,...,N$, may represent different variables based on the application under consideration. In other words, sometimes they will represent the complete state of the agents, whereas in some other cases they will represent the agent outputs which might be part of the agent states or some linear or nonlinear function of them. In order to be consistent with the notation in the control systems literature we will denote with $u_i(t),\ i = 1,...,N$, the control inputs of the agents. The problems or behaviors will be stated rather generically and not in detail and with the mathematical rigor they could be stated. The objective is to introduce the problems at high-level without any dependence on the low-level agent dynamics or other system/application dependent properties. Besides giving a conceptual idea about the tasks, this will allow their application to broader class of swarm systems.

2.1 Aggregation

Aggregation (or gathering together) is a basic behavior that many swarms in nature exhibit. Moreover, many of the collective behaviors seen in biological swarms and some behaviors to be possibly implemented in engineering multi-agent dynamic systems emerge in aggregated swarms. In other words, aggregation behavior is a basic behavior of multi-agent systems which facilitates achieving cooperative tasks. Therefore, developing mathematical models for swarm aggregations and studying the dynamics and properties of these models are important. The aggregation task can be stated as follows.

Problem 1. *(Aggregation) Consider a swarm of N agents with states (or outputs)* $x_i(t),\ i = 1,...,N$*, evolving in* $\mathbb{R}^n$*. Let the corresponding control inputs of the agents be denoted by* $u_i(t),\ i = 1,...,N$*. Design the control inputs* $u_i(t),\ i = 1,...,N$*, such that the agents converge to a close vicinity of each other, or basically as* $t \longrightarrow \infty$ *for all agents* $i = 1,...,N$ *and* $j = 1,...,N$*, we have*

$$\lim_{t \longrightarrow \infty} \|x_i(t) - x_j(t)\| \leq \varepsilon \tag{2.1}$$

for some $\varepsilon > 0$.

Note that this definition has some similarities with the control theoretic concept of uniform ultimate boundedness. The value of ε is a measure of the ultimate swarm size. Its value may depend on the number of agents in the swarm as well as the controller parameters. Note here that for the aggregation problem, implicitly it is assumed that the agents cannot occupy the same position in the space and such situations need to be avoided since they mean collisions between physical agents. In practical applications, the agents are not point agents and they have physical size. Therefore, in applications it may not be possible to achieve an arbitrarily small ε and its value may need to be scaled based on the number of agents in the swarm (hence the dependence on N). The case in which collisions between agents is not an issue and the swarm is desired to collapse to a single point is a different problem which will also be discussed below.

Another issue which is not explicitly specified in the problem definition above is the connectivity or interaction topology of the swarm (i.e., when we view agents as nodes and interactions as arcs of a graph). In fact, the problem might need to be solved under various interaction topologies which can be represented with a complete (strongly connected) graph, a connected graph, or time dependent uniformly connected graph. In this book we mostly utilize interaction topologies which can be represented with a strongly connected graph. However, the discussions are directly extendable to interaction topologies which can be represented with a connected or uniformly connected time varying graph. Note also that in some applications it might be more convenient to specify the distance of the agents to a common point such as the centroid

$$\bar{x}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t)$$

of the swarm instead of the inter-agent distances in (2.1). Therefore, the aggregation problem can be equivalently stated as follows.

Problem 2. *(Aggregation) Consider a swarm of N agents with states (or outputs)* $x_i(t),\ i = 1,...,N$*, evolving in* $\mathbb{R}^n$*. Let the corresponding control inputs of the agents be denoted by* $u_i(t),\ i = 1,...,N$*. Design the control inputs* $u_i(t),\ i = 1,...,N$*, such* *that the agents converge to a close vicinity of the swarm centroid* $\bar{x}(t)$ *or basically as* $t \rightarrow \infty$ *for all agents* $i = 1,...,N$, *we have*

$$\lim_{t \to \infty} \|x_i(t) - \bar{x}(t)\| \leq \varepsilon \tag{2.2}$$

for some $\varepsilon > 0$.

Note that the above two definitions are equivalent and can be used interchangeably.

2.2 Social Foraging

Aggregation in biological swarms usually occurs during *social foraging*. Social foraging has many advantages, one of which is increasing the probability of success for the individuals in the swarm. Therefore, social foraging is an important problem since swarm studies in engineering may benefit from similar advantages.

In social foraging the environment affects the motion or behavior of the agents. The environment may have favorable regions (representing food or nutrients in biological swarms or targets or goals in engineering applications) to which the agents may want or need to move and unfavorable regions (representing toxic or hazardous substances in biological swarms or threats or obstacles in engineering applications) which the agents may want or need to avoid. Therefore, the studies on social foraging usually incorporate determining strategies to move towards and achieve aggregation in the favorable regions while avoiding unfavorable ones.

Let us represent the environment with the function $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}$, which we call the *resource profile* in this book. Let us assume that regions or points with lower values of the resource profile are "favorable" to the agents in the swarm and they desire to move to the minimum points of the profile. For example, without loss of generality one can assume that at a point $y \in \mathbb{R}^n$ if $\sigma(y) < 0$ it is an attractant or nutrient rich point, if $\sigma(y) = 0$ it is a neutral point, and if $\sigma(y) > 0$ it is point which contains noxious substances. Moreover, for any two points $y_1, y_2 \in \mathbb{R}^n$ if $\sigma(y_1) < \sigma(y_2)$ then it means that y_1 is more favorable compared to y_2. Note that these constitute just a convention and the reverse case (i.e., the case in which $\sigma(y) < 0$ represents noxious, $\sigma(y) = 0$ represent neutral, and $\sigma(y) > 0$ represents attractant) and other cases can be equivalently handled. Under these assumptions the social foraging problem can be stated as follows.

Problem 3. *(Social Foraging) Consider a swarm of N agents with states (or outputs)* $x_i(t)$, $i = 1,...,N$, *evolving in* $\mathbb{R}^n$. *Assume that the swarm moves in an environment (a resource profile) represented by a function* $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}$. *Let the corresponding control inputs of the agents be denoted by* $u_i(t)$, $i = 1,...,N$. *Design the control inputs* $u_i(t)$, $i = 1,...,N$, *so that the agents converge to a close vicinity of local minimums* $c_{\sigma j}$ *of the profile* $\sigma(x)$ *(favorable regions), while simultaneously avoiding local maximums (unfavorable regions) and keeping cohesiveness or basically as* $t \rightarrow \infty$ *for all agents* $i = 1,...,N$, *we have*

$$\lim_{t \to \infty} \|x_i(t) - c_{\sigma j}\| \leq \varepsilon_j \tag{2.3}$$

for some $\varepsilon_j > 0$ *and* $c_{\sigma j} \in \mathbb{R}^n$ *such that* $f(c_{\sigma j}) \leq f(x)$ *holds for some* ε_{j2} *and for all* x *satisfying* $\|x - c_{\sigma j}\| \leq \varepsilon_{j2}$. *Moreover, it should be the case that* $\|x_i(t) - x_k(t)\| \leq \varepsilon$ *for all agents* $i = 1,\ldots,N$, *and* $k = 1,\ldots,N$, *and for some* $\varepsilon > 0$.

Note that the social foraging problem as defined above is basically aggregation in particular (desired) region(s) of the space corresponding to a favorable region(s) of the environment. As one would recall in the aggregation problem introduced in the preceding section the effect of the environment is not taken into account and therefore the location where the aggregation occurs is somehow unimportant, whereas in social foraging it is of primary interest. Note also that the problem is stated such that different individuals are allowed to converge to the vicinity of different local minima. If needed this can be further narrowed to require aggregation in the vicinity of one local minimum only.

Social foraging has some advantages over individual (selfish) foraging. It has been observed in some biological systems that social foragers perform better than individual foragers. Social foraging is, in particular, more advantageous when the resource profile is (or the measurements of it are) noisy. This is because in a noisy profile individual foragers can get stuck at locations which they may perceive as local minima due to the noise. In contrast, in the case of social foraging this can be avoided since the motions of the agents are affected by the other agents in the swarm as well and they can "pull out" the stuck agent out of such a trap. In other words, the overall motion of the swarm in social foraging can exhibit some averaging effects which can filter high frequency noise and improve the performance of the overall swarm and therefore the performance of individual agents as well.

2.3 Formation Control and Swarm Tracking

The formation control problem is basically the problem in which a group of agents are required to form and keep geometric configuration(s). It can be further categorized into various stages/tasks which are basically formation acquisition (or stabilization), formation maintenance, and formation reconfiguration (or switching).

In *formation stabilization*, the task is convergence of a group of agents which are initially at non-structured and possibly random positions to a particular geometrical configuration, and thereby construction of a structured formation. The geometric configuration usually has to match a pre-defined geometric pattern or shape. However, there are cases in which achieving uniform structure is more important than achieving particular shape. The convergence may be required to be asymptotic, exponential, or in finite time depending on the application under consideration.

Other terms used interchangeably with *formation stabilization* are *formation acquisition* or *formation achievement*. These last two terms are mostly used when in the *formation stabilization* task it is also required that the final shape of the formation matches a pre-defined geometric pattern. In a typical *formation acquisition* task, the scale of the pre-defined geometric pattern (i.e., the distances between the nodes in the geometric pattern indicating the desired final positions of agents relative to each other) may or may not be specified. A change in the scale or orientation of the formation constitute formation maneuvers which will be briefly discussed below.

In this book we mostly consider the problem of formation stabilization and achievement of a predefined geometric shape or pattern with predefined inter-agent distances and fixed scale. In other words, we consider formation control in the context of the following definition.

Problem 4. *(Formation Control - Formation Acquisition/Stabilization) Consider a swarm of N agents with states (or outputs)* $x_i(t),\ i = 1,...,N$*, evolving in* $\mathbb{R}^n$*. Let the corresponding control inputs of the agents be denoted by* $u_i(t),\ i = 1,...,N$*. Given a set of desired inter-agent distances* $\{d_{ij} | i, j \in \{1,\dots,N\}, i \neq j\}$*, where* d_{ij} *denotes the desired distance between agents i and j, design the control inputs* $u_i(t),\ i = 1,...,N$*, such that as* $t \longrightarrow \infty$ *for all agent pairs* $(i, j),\ i, j = 1,...,N$*, we have*

$$\lim_{t\longrightarrow\infty} \|x_i(t) - x_j(t)\| = d_{ij} \tag{2.4}$$

In the above definition the desired inter-agent distances d_{ij} are such that the configuration of the agents corresponds to a particular geometric shape such as line, triangle, tetrahedron, etc. Often after a formation is acquired (or sometimes simultaneously as the formation is acquired) the agents are required to move cohesively and possibly track a trajectory which brings the problem of *formation maintenance*. In other words, formation maintenance problems focus on maintenance of an achieved formation structure of a swarm during any continuous motion of the swarm. It is possible to define the formation such that flexibility in the shape is allowed meaning that the shape can be deformed if needed or it might be required that the formation is rigid during the entire motion. Deformation in shape might be allowed in the cases in which the shape is not of paramount importance and proper navigation in the environment possibly filled with obstacles is not possible with the initial shape. In contrast, rigidity is required when the geometrical shape of the formation is necessary for a collective task the swarm of robots is required to perform (such as carrying an object, for example) and deforming the formation may jeopardize achievement of the task. Sometimes the agents might need to catch and enclose (surround) a moving target, form a predefined formation around it and track it, or escort it with a given formation. In such cases, in contrast to the trajectory tracking problems not all information (such as velocity and/or acceleration) of the moving target might be known by the agents. This is a special type of formation maintenance which we will refer to as *swarm tracking*. The swarm tracking task can be stated more formally as follows.

Problem 5. *(Swarm Tracking) Consider a swarm of N agents with states (or outputs)* $x_i(t),\ i = 1,...,N$*, and target with state (or output)* $x_T(t)$ *all evolving in* $\mathbb{R}^n$*. Let the corresponding control inputs of the agents be denoted by* $u_i(t),\ i = 1,...,N$*. Given a set of desired inter-agent distances* $\{d_{ij} | i, j \in \{1,\dots,N\}, i \neq j\}$*, where* d_{ij} *denotes the desired distance between agents i and j, design the control inputs* $u_i(t),\ i = 1,...,N$*, such that as* $t \longrightarrow \infty$ *both the conditions*

$$\lim_{t\to\infty} x_T(t) \in \mathrm{conv}\{x_1(t),\ldots,x_N(t)\} \tag{2.5}$$

and

$$\lim_{t\to\infty} \Big| \|x_i(t) - x_j(t)\| - d_{ij} \Big| \le \varepsilon,\ \forall i \ne j \in \{1,\ldots,N\} \tag{2.6}$$

are satisfied where conv$\{x_1,\ldots,x_N\}$ *denotes the convex hull of* $x_1,\ldots,x_N$ *and* $\varepsilon > 0$ *is a small constant.*

The swarm tracking problem is the generalized (extended) counterpart to multi-agent systems of pursuit-evasion problems for single agents. It is assumed that the moving target is maneuvering in order to evade capture (or being enclosed) by the swarm of agents and the swarm does not have full information about the motion of the target. In some applications, in addition to stabilizing and keeping a formation it might be needed to perform various *formation maneuvers* without modifying the overall geometrical shape of the formation. Common formation maneuvers include translation, rotation, expansion, and contraction. It is possible to formally define these maneuvers as well. Simply stated, translation constitutes moving the formation from one position in space to another without modifying its orientation or relative distances between agents; rotation constitutes changing the orientation of the geometrical shape without modifying its position in the space or relative distances between agents; expansion or contraction constitutes modifying the relative distances between agents (either uniformly decreasing or uniformly increasing) without modifying its position or orientation in the space. All these maneuvers can be viewed under the broader concept of *formation reconfiguration*. Another form of formation reconfiguration is *formation switching*, or basically changes from one shape to another. Note that formation switching is a formation maneuver which requires change in both relative bearings and relative distances between the agents in the swarm. Such shape changes might be needed due to task requirements or might occur as a reaction to environmental changes. For example, a swarm moving in a triangular formation might need to switch to a line formation when passing through a narrow passage in which it is impossible to keep the initial formation, and to switch back after clearing the passage.

2.4 Distributed Agreement

An important and amazing feature of swarms in nature is that they usually have the capacity to achieve agreement on a common global variable in a distributed manner using only local limited information and local interactions. Examples of swarms exhibiting such behavior or phenomena include distributed synchronization of the flashing of fireflies in which thousands of fireflies flash in unison, distributed decision making in swarm of bees during nest site selection in which a swarm of bees after leaving the old hive chooses in a finite time the best new nest/hive among the available ones, reaching consensus in opinion in closed social networks, and others. It is scientifically interesting to study and uncover the mechanisms of this phenomena. Moreover, at different stages of swarm studies and different multi-agent dynamic systems applications there may arise situations in which the agents may need to agree on some information which could be agent position, velocity, oscillation phase, target to be handled or any other decision variable. Moreover, due to the limited resources and agent capacities this agreement might need to be done using only local limited information and available local interactions. Therefore, there is a need to develop mathematical models and corresponding control strategies which guarantee agreement.

The problems of developing distributed or decentralized control strategies for agreement are called *distributed agreement* problems. There are also studies which refer to the distributed agreement problem as *distributed consensus seeking*. Note that these two terms can be used interchangeably to describe the same problem. It is said that *agreement* or *consensus* is achieved if the corresponding variables of interest of all agents converge to the same value. The variable of interest can represent the whole state of the agents in which case the problem can be referred as a *state agreement* problem whereas on other case it can represent part of the agent states (such as heading of the agents) and the neighborhoods might be defined based on another part of the states (such as for example the relative positions of the agents) in which case one can refer to the problem as an *output agreement* problem. We state the distributed agreement problem as follows.

Problem 6. *(Distributed Agreement) Consider a swarm of N agents with states (or outputs)* $x_i(t),\ i = 1,...,N$*, evolving in* $\mathbb{R}^n$*. Let the corresponding control inputs of the agents be denoted by* $u_i(t),\ i = 1,...,N$*. Assume that at every time instant t each agent i can interact with (i.e., obtain information from) a possibly different (i.e., time varying) subset of agents* $\mathcal{N}_i(t)$ *called its neighbors. Design the control inputs* $u_i(t),\ i = 1,...,N$*, and determine conditions on the neighborhood (interaction) topology (or, if possible, appropriately design the interaction topology as well), so that as* $t \longrightarrow \infty$ *for all agents* $i = 1,...,N$*, we have*

$$\lim_{t \longrightarrow \infty} x_i(t) = x_c \tag{2.7}$$

for some $x_c \in \mathbb{R}^n$*.*

The state x_c in the above definition is referred to as the *consensus state*.

The distributed agreement problem has some similarities with the formation control and aggregation problems. In particular, one may have the impression that distributed agreement is a special case of formation control under the condition that $d_{ij} = 0$ for all agents i and j. However, re-examining carefully the above definition one can see that there are also important differences. First of all, the consensus state, i.e., the state (or output) to which all $x_i(t)$ converge, is emergent in the distributed agreement problem. Its value is usually dependent on the initial states (or outputs) $x_i(0)$ and the changes in the interaction (neighborhood) topology over time. Moreover, the connectivity properties of the neighborhood (information flow) topology are of paramount importance in distributed agreement problems. Also, since the interaction (and therefore the information flow) can be time varying the uniformity in the connectivity over time plays crucial role in achieving agreement. Moreover, in formation control or aggregation two agents occupying the same state is undesirable since it implies collision between agents, whereas in the distributed agreement problem it is the sole objective. Note also that the control strategies or update rules that lead to agreement are usually called *consensus protocols* in the literature. Although it is possible to use continuous-time agent dynamics and continuous-time protocols to model the agreement problem, here we will approach the problem using discrete-time techniques. It is possible also to further generalize the distributed agreement problem such that agreement is achieved over a common trajectory and not a fixed value. In other words, it can be generalized such that x_c in (2.7) is a time varying function $x_c(t)$. Note, however, that still $x_c(t)$ and its properties are emergent and not predefined.

2.5 Function Minimization

Function minimization is a fundamental problem in many engineering disciplines. Various classical methods for function minimization are available. In this book we will discuss alternative solution methods inspired from the behavior or operation of swarms in nature. In particular, we will present the bacterial foraging optimization (BFO) and the particle swarm optimization (PSO) algorithms. These methods are direct search methods which have been inspired by the foraging behavior of swarms in nature. In other words, they are based on mimicing (to some extend) the behavior of foraging behavior of swarms in nature. Beside their ease of implementation, empirical studies have demonstrated that the presented bioinspired optimization algorithms have effective performance. Moreover, they can be implemented on parallel and asynchronous computing systems. Below we briefly specify the basic optimization (function minimization) problem.

Problem 7. *(Function Minimization) Consider an objective function* $f : \mathbb{R}^n \to \mathbb{R}$ *to be minimized. Determine a point (or a set of points)* $x^* \in \mathbb{R}^n$ *such that for some neighborhood* $N(x^*)$ *of* x^* *and for all* $x \in N(x^*)$

$$f(x^*) \leq f(x)$$

is satisfied.

The above problem definition corresponds to unconstrained optimization. In other words, there are no restrictions over x. There are also problems in which, in addition to minimizing the objective function, the variable of interest has to satisfy certain conditions or constraints which might be expressed in forms of equalities or inequalities. Moreover, in the above definition no restriction on the objective function $f(x)$ has been specified. Usually continuity and/or smoothness requirements on $f(x)$ need to be imposed since otherwise the problem becomes much more difficult to solve. Note also that the foraging problem presented in the preceding sections has some similarities with the function minimization problem presented here.

2.6 Related Problems and Behaviors and Further Reading

There are also problems in the multi-agent dynamic systems literature which are very much related to the problems presented above but not discussed in this book. Some of these problems are formation structure breakdowns, flocking, rendezvous, and synchronization. Below we briefly mention these as well.

2.6.1 Formation Structure Breakdowns

During cohesive motion of swarms in the form of structured formations sometimes it might be inevitable to have structure break-downs or changes different from the formation maneuvers discussed above in the context of formation reconfiguration. Some relevant formation operations that may result in structure change include *merging*, *splitting*, and *agent loss*. Merging is combining of two formations via some interaction links in between to form a single formation. Splitting is the "reverse" of merging or basically division of a given formation into two or more smaller formations via breaking some of the interaction links. In agent loss, one or more agents together with the attached interaction links are lost and there is a need to reconfigure the interaction topology and/or the geometrical shape of the formation. Various properties such as *rigidity* or *persistency* might be needed to be preserved during such operations. Works investigating maintenance of rigidity and persistence during changes on the formation structure due to *merging*, *splitting*, or *agent loss* can be found in [8, 69, 263, 264].

2.6.2 Flocking

Flocking, in general, can be defined as collective motion behavior of a large number of interacting agents with a common group objective. Usually the common direction of motion is emergent from the interactions between the agents in the swarm. The interactions between agents are based on the so-called "nearest neighbor rule" (where the agents adjust their motion based only on their nearest neighbors). The pioneering work by Reynolds [206] proposed three simple rules to implement a flocking behavior, namely separation, alignment, and cohesion. These rules have been used to develop realistic computer simulations of the flocking behavior of animal swarms. The work in [251] based on the *self-propelled particle* systems can be also viewed as a simplified model of flocking which investigates the effects of noise on the overall system dynamics and phase transition from disordered to ordered behavior.

Flocking has many similarities with the formation control problem for the case in which the geometric shape of the formation is not specified a priori. Also, achieving common orientation in a large flock despite the fact that not all agents can interact with each other, is in a sense a distributed agreement problem. Mathematical analysis on a simplified model of achieving common orientation during the flocking behavior based on "nearest neighbor rules" is provided and some corresponding convergence results are established in [126]. Recent empirical studies in [42, 43] investigate the effect of neighborhood size and asynchronism [42] and turn angle restrictions (a type of non-holonomic constraint) [43] on the flocking behavior of the system in [251]. See also [211] for a more comprehensive treatment.

An important attempt to rigorously model and analyze flocking behavior can be found in [234, 235] for systems with point mass dynamics and in [236] for agents with non-holonomic unicycle dynamics. Similarly the work in [182] investigates stable flocking of agents with point mass dynamics. The agent interactions are based on "nearest neighbor rules" and potential functions are utilized for aggregation and alignment. Several algorithms are proposed and analyzed with and without group objectives and it is shown that under certain conditions flocking will be achieved and the flock will have a lattice-type structure. Three rules, which are flock centering, collision avoidance, and velocity matching, are used for achieving the flocking behavior.

2.6.3 Rendezvous

Rendezvous is the problem in which the task is meeting (or gathering) of agents at a point or a small region in the space in finite time [205]. It can be thought as a specific form of aggregation or flocking [9] or a specific form of distributed agreement (consensus seeking) and therefore also specific form of formation stabilization (to a single point). The main difference from these problems is that in rendezvous the gathering has to occur in finite time [74, 101, 148, 149], whereas aggregation or agreement (consensus) are allowed to be achieved asymptotically. The neighborhoods can be time dependent and based on nearest neighbors. The problem can be investigated in different settings such as synchronous [148] or asynchronous [149] or for agents under cyclic pursuit (i.e., swarms where each agent follows the preceding agent and the interaction topology forms a ring) [165, 166]. We will not consider rendezvous in this book.

2.6.4 Synchronization of Coupled Nonlinear Oscillators

As mentioned above a good example of distributed agreement in nature is the synchronization of the flashing of fireflies [228]. This phenomenon can be viewed also as distributed synchronization of coupled oscillators. Synchronization of coupled oscillators is usually mathematically represented by the so-called Kuramoto model [216, 227]. The Kuramoto model can be viewed as a special case of the single integrator aggregation model considered in the following sections. It can be viewed also as an example of a continuous-time consensus protocol. There are various studies in the literature which investigate continuous-time consensus protocols [72, 126, 184, 203, 205]. Alternative discrete-time consensus protocols are considered in [126, 176, 203, 205]. The discrete-time consensus protocols are usually based on taking convex combinations between the states (outputs) of the neighboring agents.

2.6.5 Other Behaviors

The possibilities for multi-agent dynamic systems are enormous. Although we are interested in particular aspect which is mostly coordinated motion, as mentioned earlier, there are other aspects as well. Another aspect in swarm coordination and control is cooperative operations to achieve a common or shared objective different than cooperative motion. Various examples of this, including surveillance, sweeping and coverage tasks, and others can also be found in the literature, e.g. [33, 49, 59, 85, 143, 177, 195, 209, 210]. An attempt to study emergence can be found in [142]. Studies on mobile sensor networks are also relevant to studies on swarms. A survey on sensor networks can be found in [6]. Recent related books on multi-agent dynamic systems include [32, 202, 221].

Part II

Continuous Time Swarms

3

Swarms of Single Integrator Agents

3.1 Single Integrator Agent Model

The simplest mathematical representation of agents (e.g., robots, satellites, unmanned ground or air vehicles) used for studying swarm behavior is the *single integrator* model. In this model the motion of an agent i, $i = 1,\dots,N$, is given by

$$\dot{x}_i = u_i, \tag{3.1}$$

where $x_i \in \mathbb{R}^n$ is the state of agent i and $u_i \in \mathbb{R}^n$ is its control input. We refer to this model as a *higher-level* or *kinematic* agent model since it ignores the lower-level vehicle dynamics of the individual agents. However, it is a relevant and useful model since it can be used for studying higher level algorithms, independent of detailed agent dynamics, and for obtaining "proof of concept" type results for swarm behavior. Moreover, in certain control tasks involving path planning, the trajectories generated using the *higher-level* agent model in (3.1) can be used as reference trajectories for the actual agents to track. Furthermore, (3.1) is a realistic simplified kinematic model for a class of *omni-directional* mobile robots with so-called *universal* (or *Swedish*) *wheels* [35, 111, 261]. In different contexts the state x_i can represent different entities. For example, for robotic swarms it may represent the position of agent i, whereas in distributed agreement problems it may represent the current estimate of the variable to be agreed upon (i.e., the "opinion") of agent i. Nevertheless, in implementations of the results obtained for (3.1) with real-life robotic agents it may be necessary to consider the effects of the actual low-level agent dynamics.

Given the agent dynamics in (3.1), in this chapter we will discuss developing control algorithms for obtaining several different swarm behaviors including aggregation, social foraging, formation control, and swarm tracking. Our approach to solving these problems will be a potential functions based approach.

We assume all individuals in the swarm move simultaneously and know the exact relative position of the other individuals. Let $x^\top = [x_1^\top, x_2^\top, \dots, x_N^\top] \in \mathbb{R}^{Nn}$ denote the vector of concatenated states of all the agents. In this chapter the control input u_i of individual i in (3.1) will have the form

$$u_i = -\nabla_{x_i} J(x) \tag{3.2}$$

where $J : \mathbb{R}^{Nn} \to \mathbb{R}$ is a potential function which represents the interaction (i.e., the attraction and/or repulsion relationship) between the individual agents and needs to be chosen by the designer based on the swarm application under consideration and the desired behavior from the swarm. We will discuss which properties the potential functions should satisfy for different problems and present results for some potential functions.

3.2 Aggregation

Aggregation is one of the most basic behaviors seen in swarms in nature (such as insect colonies) and is sometimes the initial phase in collective tasks performed by a swarm [25]. Below, we discuss how to achieve aggregation for the single integrator model in (3.1).

3.2.1 Potential Function Design

If only simple aggregation is desired from the swarm, then the potential function J can be selected as $J(x) = J_{aggregation}(x)$ where

$$J_{aggregation}(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[J_a\Big(\|x_i - x_j\|\Big) - J_r\Big(\|x_i - x_j\|\Big) \right]. \tag{3.3}$$

Here, $J_a : \mathbb{R}^+ \to \mathbb{R}$ represents the attraction component, whereas $J_r : \mathbb{R}^+ \to \mathbb{R}$ represents the repulsion component of the potential function. Although not the only choice, the above potential function is very intuitive since it represents an interplay between attraction and repulsion components. Note also that it is based only on the relative distances between the agents and not the absolute agent positions.

Given the above type of potential function, the control input of individual i, $i = 1, \ldots, N$, can be calculated as

$$u_i = -\sum_{j=1, j\neq i}^{N} \left[\nabla_{x_i} J_a\Big(\|x_i - x_j\|\Big) - \nabla_{x_i} J_r\Big(\|x_i - x_j\|\Big) \right]. \tag{3.4}$$

Note that since $\dot{x}_i = u_i$, the motion of the individual is along the negative gradient and leads to a decent motion towards a minimum of the potential function J. Moreover, since the functions $J_a(\|y\|)$ and $J_r(\|y\|)$ create a potential field of attraction and repulsion, respectively, around each individual, the above property restricts the motion of the individuals toward each other along the gradient of these potentials (i.e., along the combined gradient field of $J_a(\|y\|)$ and $J_r(\|y\|)$).

One can show that, because of the chain rule and the definition of the functions J_a and J_r, the equalities

$$\begin{aligned}\nabla_y J_a(\|y\|) &= y g_a(\|y\|)\\ \nabla_y J_r(\|y\|) &= y g_r(\|y\|)\end{aligned}$$

are always satisfied for some functions $g_a : \mathbb{R}^+ \longrightarrow \mathbb{R}$ and $g_r : \mathbb{R}^+ \longrightarrow \mathbb{R}$. Here $g_a : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ represents (in a sense the magnitude of) the attraction term, whereas $g_r : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ represents (in a sense the magnitude of) the repulsion term. Note also that the combined term $-yg_a(\|y\|)$ represents the actual attraction, whereas the combined term $yg_r(\|y\|)$ represents the actual repulsion, and they both act on the line connecting the two interacting individuals, but in opposite directions. The vector y determines the alignment (i.e., it guarantees that the interaction vector is along the line on which y is located) as well as affects the magnitude of the attraction and repulsion components. The terms $g_a(\|y\|)$ and $g_r(\|y\|)$, on the other hand, affect correspondingly only the magnitude of the attraction and repulsion, whereas their difference determines the direction of the interaction along vector y.

Let us define the function $g(\cdot)$ as

$$g(y) = -y\Big[g_a(\|y\|) - g_r(\|y\|)\Big]. \tag{3.5}$$

We call the function $g(\cdot)$ an attraction/repulsion function and assume that on large distances attraction dominates, that on short distances repulsion dominates, and that there is a *unique distance* at which the attraction and the repulsion balance. In other words, we assume that $g(\cdot)$ satisfies the following assumptions.

Assumption 1. *The function* $g(\cdot)$ *in* (3.5) *and the corresponding* $g_a(\cdot)$ *and* $g_r(\cdot)$ *are such that there exist a* unique distance δ *at which we have* $g_a(\delta) = g_r(\delta)$. *Moreover, we have* $g_a(\|y\|) > g_r(\|y\|)$ *for* $\|y\| > \delta$ *and* $g_r(\|y\|) > g_a(\|y\|)$ *for* $\|y\| < \delta$.

One issue to note here is that for the attraction/repulsion functions $g(\cdot)$ defined as above we have $g(y) = -g(-y)$. In other words, the above $g(\cdot)$ functions are *odd* (and therefore symmetric with respect to the origin). This is an important feature of the $g(\cdot)$ functions that leads to reciprocity in the inter-agent relations and interactions. In this chapter we will consider swarms with such reciprocal interactions.

In order for the above assumptions to be satisfied the potential function designer should choose the attraction and repulsion potentials such that the minimum of $J_a(\|x_i - x_j\|)$ occurs on or around $\|x_i - x_j\| = 0$, whereas the minimum of $-J_r(\|x_i - x_j\|)$ (or the maximum of $J_r(\|x_i - x_j\|)$) occurs on or around $\|x_i - x_j\| \to \infty$, and the minimum of the combined $J_a(\|x_i - x_j\|) - J_r(\|x_i - x_j\|)$ occurs at $\|x_i - x_j\| = \delta$. In other words, at $\|x_i - x_j\| = \delta$ the attraction/repulsion potential between two interacting individuals has a global minimum. Note, however, that when there are more than two individuals, the minimum of the combined potential does not necessarily occur at $\|x_i - x_j\| = \delta$ for all $j \neq i$. Moreover, there exists a family of minima. However, this does not constitute a problem since in the aggregation problem we are not concerned with which minimum (or relative configuration) the swarm converges to. Note also that one can view $J(x)$ as the potential energy of the swarm, whose value depends on the inter-individual distances (such that it is high when the agents are either far from each other or too close to each other) and the motion of the swarm in (3.2) (and therefore the one in (3.4)) is towards a minimum energy configuration (one that is not necessarily unique).

It has been observed in nature that there are attraction and repulsion forces (with attraction having longer range than repulsion) between individuals that lead to the swarming behavior. For example, for fish attraction is generally based on vision and has a long range, whereas repulsion is based on the pressure on the side of the fish and has a short range (but is stronger than attraction). Moreover, it has been observed that both attraction and repulsion are always "on" and the resulting behavior is due to the *interplay* between these two forces, and there is a distance (called the "equilibrium distance" in biology) at which attraction and repulsion between two individuals balance. Note that the above model captures these properties by having attraction and repulsion terms in the motion equation acting in opposite directions, and the "equilibrium distance" is the unique distance δ at which we have $g_a(\delta) = g_r(\delta)$.

One potential function which satisfies the above conditions, including Assumption 1, and has been used in the literature for swarm aggregations (see for example [92]) is

$$J(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{a}{2} \|x_i - x_j\|^2 + \frac{bc}{2} \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \right]. \tag{3.6}$$

Its corresponding attraction/repulsion function can be calculated as

$$g(y) = -y \left[a - b \exp\left(-\frac{\|y\|^2}{c} \right) \right]. \tag{3.7}$$

The equilibrium distance for this function can be easily calculated as $\delta = \sqrt{c \ln (b/a)}$.

Swarming in nature normally occurs in a distributed fashion. In other words, there is no leader (or boss) and each individual decides independently its direction of motion. The above potential function based interaction model captures this in its simplest form by having separate equations of motion for each individual that do not depend on an external variable (such as a command from a boss or another agent). In contrast, an individual's motion depends only on the position of the individual itself and its observation of the positions (or relative positions) of other individuals.

3.2.2 Analysis of Swarm Motion

In this section we will discuss the collective motion of the swarm without taking into consideration whether the swarm is cohesive (has aggregated/clustered) or not. Let us define the *centroid* of the swarm (to which sometimes we will also loosely refer as the swarm center) as

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i.$$

Note that since the functions $g(\cdot)$ are odd, and therefore symmetric with respect to the origin, the centroid $\bar{x}$ of the swarm is stationary for all t. This is stated formally in the following lemma.

Lemma 1. *The centroid* $\bar{x}$ *of the swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.4) *with an attraction/repulsion function* $g(\cdot)$ *which is odd and satisfies Assumption 1 is stationary for all* t.

Proof: The time derivative of the centroid is given by

$$\begin{aligned}\dot{\bar{x}} &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\left[g_a\Big(\|x_i-x_j\|\Big)-g_r\Big(\|x_i-x_j\|\Big)\right](x_i-x_j)\\ &= -\frac{1}{N}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left\{\left[g_a\Big(\|x_i-x_j\|\Big)-g_r(\|x_i-x_j\|)\right](x_i-x_j)\right.\\ &\qquad\left.+\left[g_a\Big(\|x_j-x_i\|\Big)-g_r\Big(\|x_j-x_i\|\Big)\right](x_j-x_i)\right\}=0.\end{aligned}$$

■

Let us denote the invariant set of equilibrium (or stationary) points for the swarm with

$$\Omega_e = \{x : \dot{x} = 0\}.$$

Note that $x \in \Omega_e$ implies that $\dot{x}_i = 0$ for all $i = 1,\ldots,N$, implying that all individuals are stationary (i.e., have stopped) whenever the state is within that set.

Theorem 1. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.4) *with an attraction/repulsion function* $g(\cdot)$ *which is odd and satisfies Assumption 1. For any* $x(0) \in \mathbb{R}^{Nn}$, *as* $t \to \infty$ *we have* $x(t) \to \Omega_e$.

Proof: The potential function $J(x)$ serves as a Lyapunov function for the system. Taking the gradient of $J(x)$ with respect to the state x_i of individual i we obtain

$$\nabla_{x_i}J(x) = \sum_{j=1,j\neq i}^{N}[\nabla_{x_i}J_a(\|x_i-x_j\|)-\nabla_{x_i}J_r(\|x_i-x_j\|)] = -\dot{x}_i, \tag{3.8}$$

which follows from (3.4).

Now, taking the time derivative of the Lyapunov function along the motion of the system we obtain

$$\dot{J}(x) = [\nabla_x J(x)]^\top \dot{x} = \sum_{i=1}^{N}[\nabla_{x_i}J(x)]^\top \dot{x}_i = \sum_{i=1}^{N}[-\dot{x}_i]^\top \dot{x}_i = -\sum_{i=1}^{N}\|\dot{x}_i\|^2 \leq 0,$$

for all t implying a decrease in $J(x)$ unless $\dot{x}_i = 0$ for all $i = 1,\ldots,N$. If the function $g(\cdot)$ is chosen such that the set defined as $\Omega_0 = \{x : J(x) \leq J(x(0))\}$ is compact, then using the LaSalle's Invariance Principle we can conclude that as $t \to \infty$ the state $x(t)$ converges to the largest invariant subset of the set defined as

$$\Omega_1 = \{x \in \Omega_0 : \dot{J}(x) = 0\} = \{x \in \Omega_0 : \dot{x} = 0\} \subset \Omega_e.$$

Note, however, that in general Ω_0 may not necessarily be compact for every $g(\cdot)$, which may happen if the corresponding $J(\cdot)$ is not radially unbounded. Therefore, the fact that $\dot{J}(x) \leq 0$ does not, in general, directly imply boundedness. In particular, for some potential functions the small increase in $J(x)$ due to one or more agents departing far away from the swarm might be compensated by the motion of the other agents resulting in overall decrease in $J(x)$. However, this case is not possible here since for every individual i we have $[\nabla_{x_i} J(x)]^\top \dot{x}_i = -\|\dot{x}_i\|^2 \leq 0$, which implies that every individual moves in a direction of decrease of $J(x)$. From the properties of the attraction/repulsion functions in Assumption 1 we know that attraction dominates on large distances and repulsion dominates on short distances. This, on the other hand, implies that on large distances decrease in $J(x)$ is due to agents moving towards each other, whereas on short distances decrease is due to agents moving away from each other. In other words, agents continuously departing from the rest of the swarm cannot result in decrease in $J(x)$. Therefore, the agent states are bounded and the set defined as $\Omega_x = \{x(t) : t \geq 0\} \subset \Omega_0$ is compact and we still can apply LaSalle's Invariance Principle arriving at the conclusion that as $t \to \infty$ the state $x(t)$ converges to the largest invariant subset of the set defined as

$$\Omega_2 = \{x \in \Omega_x : \dot{J}(x) = 0\} = \{x \in \Omega_x : \dot{x} = 0\} \subset \Omega_e.$$

Since in both of the above cases both Ω_1 and Ω_2 are invariant themselves and satisfy $\Omega_1 \subset \Omega_e$ and $\Omega_2 \subset \Omega_e$, we have $x(t) \to \Omega_e$ as $t \to \infty$ and this concludes the proof. ■

Note that in some engineering swarm applications such as *uninhabited air vehicles* (UAV's) individuals never stop. Therefore, the results here may seem not to be applicable. However, in some biological examples such as fruiting body formation by bacteria or engineering applications in which a group of agents are required to "gather together" to be loaded on a vehicle and transferred to a new area it is possible (or desirable) to have the agents aggregate and stop. Moreover, note also that the results here are based on relative inter-individual interactions and describe only aggregation. It is possible to extend them to mobile swarms by having a motion (or drift) term in the equation of motion together with the aggregation term described here. For example, consider the case in which given a reference trajectory $\{x_r, \dot{x}_r\}$, which is known (or estimated) by all the agents, the agents move based on

$$\dot{x}_i = u_i = \dot{x}_r - \nabla_{x_i} J(x) \tag{3.9}$$

Then, defining $\tilde{x}_i = x_i - x_r$ one obtains

$$\dot{\tilde{x}}_i = -\nabla_{x_i} J(x) = -\sum_{j=1, j\neq i}^{N} \left[g_a\Big(\|\tilde{x}_i - \tilde{x}_j\|\Big) - g_r\Big(\|\tilde{x}_i - \tilde{x}_j\|\Big) \right] (\tilde{x}_i - \tilde{x}_j)$$

which is exactly the model considered above. Then, for the swarm with motion dynamics given by (3.9), the result in Theorem 1 implies that the *relative dynamics* of the agents will eventually stop and all the agents will move with the velocity $\dot{x}_i = \dot{x}_r, i = 1, ..., N$ (somehow) along the reference trajectory $\{x_r, \dot{x}_r\}$ (with possibly a constant shift) as a single cohesive entity (in a minimum energy relative configuration). Further, if the agent controllers are chosen as

$$\dot{x}_i = u_i = \dot{x}_r - \alpha(x_i - x_r) - \nabla_{x_i} J(x) \tag{3.10}$$

then one can also show that as $t \longrightarrow \infty$ we will have $\bar{x} = x_r$ and $\dot{\bar{x}} = \dot{x}_r$ implying that the centroid $\bar{x}$ of the swarm will track the reference trajectory exactly (in addition to achieving a constant relative configuration).

We would like to emphasize that the above approach is distributed in a sense that in order to be able to implement their controllers the individuals do not have to know the global potential energy function $J(x)$ given in (3.3). Instead, it is sufficient if they know the local, or their internal potential energy function, given by

$$J_i(x) = \sum_{j=1, j\neq i}^{N} \left[J_a(\|x_i - x_j\|) - J_r(\|x_i - x_j\|) \right], \tag{3.11}$$

since

$$u_i = -\nabla_{x_i} J_i(x) = -\nabla_{x_i} J(x),$$

where the global potential $J(x)$ in (3.3) can be written as $J(x) = \frac{1}{2}\sum_{i=1}^{N} J_i(x)$ in terms of the local potentials $J_i(x)$ in (3.11).

The result in Theorem 1 is important. It proves that asymptotically the individuals will converge to a constant position and therefore to a constant relative arrangement. However, it does not specify any bound on the resulting size of the swarm. This issue will be investigated in the following section.

3.2.3 Swarm Cohesion Analysis

In this section, we will establish bounds on the ultimate "swarm size." To this end, we define the distance between the position x_i of individual i and the centroid $\bar{x}$ of the swarm as $e_i = x_i - \bar{x}$. The ultimate bound on the magnitude of e_i will quantify the size of the swarm. Taking the time derivative of e_i we have $\dot{e}_i = \dot{x}_i - \dot{\bar{x}} = \dot{x}_i$, since from Lemma 1 we have $\dot{\bar{x}} = 0$. Now, for each individual i choose $V_i = \frac{1}{2}\|e_i\|^2 = \frac{1}{2}e_i^\top e_i$ as the corresponding Lyapunov function. Taking the time derivative of V_i we obtain

$$\dot{V}_i = \dot{e}_i^\top e_i = -\sum_{j=1, j\neq i}^{N} \left[g_a\big(\|x_i - x_j\|\big) - g_r\big(\|x_i - x_j\|\big) \right] (x_i - x_j)^\top e_i. \tag{3.12}$$

Below, first we analyze the case in which we have a linear attraction and a bounded repulsion.

Linear Attraction and Bounded Repulsion Case

In this section we consider the special case in which the attraction part satisfies

$$g_a\big(\|y\|\big) = a \tag{3.13}$$

for some finite positive constant $a > 0$ and for all y, which corresponds to linear attraction since the actual attraction is given by $y g_a\big(\|y\|\big) = ay$, whereas the repulsion is constant or bounded as

$$g_r(\|x_i - x_j\|)\|x_i - x_j\| \leq b, \tag{3.14}$$

for some finite positive constant b. Note that the potential function in (3.7) satisfies both of these conditions.

Theorem 2. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.4) *with an attraction/repulsion function* $g(\cdot)$ *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). As time progresses all the members of the swarm will converge to a hyperball*

$$B_\varepsilon(\bar{x}) = \{x : \|x - \bar{x}\| \leq \varepsilon\},$$

where

$$\varepsilon = \frac{b}{a}.$$

Moreover, the convergence will occur in finite time bounded by

$$\bar{t} = \max_{i \in \{1,\ldots,N\}} \left\{ -\frac{1}{2a} \ln\left(\frac{\varepsilon^2}{2V_i(0)}\right) \right\}.$$

Proof: Choose any swarm member (agent) i. Incorporating the value of $g_a(\|x_i - x_j\|)$ in (3.12) we obtain

$$\dot{V}_i = -a \sum_{j=1, j\neq i}^{N} (x_i - x_j)^\top e_i + \sum_{j=1, j\neq i}^{N} g_r\big(\|x_i - x_j\|\big)(x_i - x_j)^\top e_i.$$

Now, note that

$$\sum_{j=1, j\neq i}^{N} (x_i - x_j) = \sum_{j=1}^{N} (x_i - x_j) = Nx_i - \sum_{j=1}^{N} x_j = Nx_i - N\bar{x} = Ne_i. \tag{3.15}$$

Substituting this in the $\dot{V}_i$ equation we obtain

$$\begin{aligned}\dot{V}_i &= -aN\|e_i\|^2 + \sum_{j=1, j\neq i}^{N} g_r(\|x_i - x_j\|)(x_i - x_j)^\top e_i \\ &\leq -aN\|e_i\| \left[\|e_i\| - \frac{1}{aN} \sum_{j=1, j\neq i}^{N} g_r(\|x_i - x_j\|)\|x_i - x_j\| \right],\end{aligned}$$

which implies that $\dot{V}_i < 0$ is satisfied as long as $\|e_i\| > \frac{1}{aN}\sum_{j=1,j\neq i}^{N} g_r(\|x_i - x_j\|)\|x_i - x_j\|$. This, on the other hand, implies that as $t \to \infty$ asymptotically the inequality

$$\|e_i\| \leq \frac{1}{aN} \sum_{j=1, j\neq i}^{N} g_r(\|x_i - x_j\|)\|x_i - x_j\|,$$

will be satisfied. Note that this inequality relation holds for any type of repulsion, provided that the attraction is linear. Then, using the bound on the repulsion in (3.14) we conclude that asymptotically for this case we will have

$$\|e_i\| \leq \frac{b(N-1)}{aN} < \frac{b}{a} \triangleq \varepsilon,$$

which provides a bound on the maximum ultimate swarm size. Now, notice that for $\|e_i\| \geq \varepsilon$ we have $\dot{V}_i \leq -a\|e_i\|^2 = -2aV_i$, which has a solution $V_i(t) \leq V_i(0)e^{-2at}$. Using this and solving for the equality $\frac{1}{2}\varepsilon^2 = V_i(0)e^{-2at}$ for t one can show that $\|e_i\| < \varepsilon$ will be reached for all i in a finite time bounded by

$$\bar{t} \triangleq \max_{i\in\{1,\ldots,N\}} \left\{ -\frac{1}{2a} \ln\left(\frac{\varepsilon^2}{2V_i(0)} \right) \right\}.$$

This concludes the proof. ■

Remark: One issue to notice here is that, if we had only attraction (i.e., if we had $g_r\Big(\|x_i - x_j\|\Big) \equiv 0$ for all i and j, $j \neq i$), then the above equation would imply that the swarm shrinks to a single point, which is the centroid $\bar{x}$. In contrast, if we had only repulsion (i.e., if we had $g_a\Big(\|x_i - x_j\|\Big) \equiv 0$ for all i and j, $j \neq i$), then the swarm would disperse in all directions away from the centroid $\bar{x}$ towards infinity. Having the attraction dominate at large distances prevents the swarm from dispersing, whereas having the repulsion dominate on short distances prevents it from collapsing to a single point, and the equilibrium is established in between. ■

Above we could establish an explicit bound on the swarm size for the case when the attraction is linear and the repulsion is bounded. Next, we will analyze the case for which we have an unbounded repulsion.

Linearly Bounded from Below Attraction and Unbounded Repulsion

By linearly bounded from below attraction we mean the case in which we have

$$g_a(\|x_i - x_j\|) \geq a, \tag{3.16}$$

for some finite positive constant a and for all $\|x_i - x_j\|$. For the repulsion functions, on the other hand, we will consider the unbounded functions satisfying

$$g_r(\|x_i - x_j\|) \leq \frac{b}{\|x_i - x_j\|^2}. \tag{3.17}$$

The simplest example of attraction/repulsion function $g(\cdot)$ satisfying the above assumptions is the case in which the equalities are satisfied it is given by

$$g(x_i - x_j) = \left[a - \frac{b}{\|x_i - x_j\|^2}\right]\Big(x_i - x_j\Big)$$

which corresponds to the potential function

$$J(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{a}{2} \|x_i - x_j\|^2 - b \ln \Big(\|x_i - x_j\| \Big) \right].$$

Theorem 3. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.4) *with an attraction/repulsion function* $g(\cdot)$ *which is odd, satisfies Assumption 1, and has linearly bounded from below attraction satisfying* (3.16) *and unbounded repulsion satisfying* (3.17). *As time progresses the* root mean square *of the distances of the swarm members to the swarm centroid will satisfy*

$$e_{rms} \triangleq \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|e_i\|^2} \leq \sqrt{\frac{b}{2a}} \triangleq \varepsilon_{rms}.$$

Proof: First, we define the cumulative (or overall) Lyapunov function for the swarm motion dynamics as $V = \sum_{i=1}^{N} V_i$ and note that since at equilibrium $\dot{e}_i = \dot{x}_i = 0$ (which was shown in Theorem 1), we have also $\dot{V}_i = 0$ for all i and therefore $\dot{V} = 0$. In other words, at equilibrium we have

$$\begin{aligned}
\dot{V} &= -\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \left[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big) \right] (x_i - x_j)^\top e_i \\
&= -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left\{ \left[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big) \right] (x_i - x_j)^\top e_i \right. \\
&\qquad \left. + \left[g_a\Big(\|x_j - x_i\|\Big) - g_r\Big(\|x_j - x_i\|\Big) \right] (x_j - x_i)^\top e_j \right\} \\
&= -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big) \right] \|x_i - x_j\|^2 \\
&= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \left[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big) \right] \|x_i - x_j\|^2 = 0,
\end{aligned}$$

where the third line was obtained using the fact that for any $\alpha \in \mathbb{R}$ we have

$$\alpha(x_i - x_j)^\top e_i + \alpha(x_j - x_i)^\top e_j = \alpha \|x_i - x_j\|^2, \tag{3.18}$$

which is true since $x_i - x_j = e_i - e_j$. From the result on $\dot{V}$ in the above equation one obtains

$$\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} g_a(\|x_i - x_j\|) \|x_i - x_j\|^2 = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} g_r(\|x_i - x_j\|) \|x_i - x_j\|^2. \tag{3.19}$$

Note that since the actual attraction term is $y g_a(\|y\|)$, we have $g_a\Big(\|x_i - x_j\|\Big) \|x_i - x_j\| \geq a \|x_i - x_j\|$ for this case (and hence the name linearly bounded from below attraction). Using this fact, from (3.19) one obtains

$$a\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\|x_i-x_j\|^2 \leq \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} g_r(\|x_i-x_j\|)\|x_i-x_j\|^2.$$

Similarly, from the bound on $g_r\Big(\|x_i-x_j\|\Big)$ we know that $g_r\Big(\|x_i-x_j\|\Big)\|x_i-x_j\|^2 \leq b$ and obtain

$$\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} g_r\Big(\|x_i-x_j\|\Big)\|x_i-x_j\|^2 \leq bN(N-1).$$

Using the fact that $e_i = \frac{1}{N}\sum_{j=1}^{N}(x_i-x_j)$ (see equation (3.15)) and the equality in (3.18) for the sum of the squares of the error one can show that

$$\sum_{i=1}^{N}\|e_i\|^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}(x_i-x_j)^\top e_i = \frac{1}{N}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\|x_i-x_j\|^2 = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\|x_i-x_j\|^2$$

holds (where we again used the fact in (3.18) to obtain the second equality).

Combining these equations with (3.19) we obtain

$$2aN\sum_{i=1}^{N}\|e_i\|^2 \leq bN(N-1)$$

which implies that at equilibrium we have

$$\frac{1}{N-1}\sum_{i=1}^{N}\|e_i\|^2 \leq \frac{b}{2a}.$$

Then, for the *root mean square* of the error we have

$$e_{rms} \triangleq \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|e_i\|^2} \leq \sqrt{\frac{b}{2a}} \triangleq \varepsilon_{rms}, \tag{3.20}$$

which concludes the proof. ■

The inequality in (3.20) establishes a bound on the swarm size and implies that the swarm will be cohesive. Moreover, it shows that $\|e_i\|$ will be ultimately bounded by

$$\|e_i\| \leq \sqrt{\frac{bN}{2a}} = \varepsilon_{rms}\sqrt{N},$$

for all t. In other words, no agent in the swarm can diverge to infinity.

The equality in (3.19) states, in a sense, that at equilibrium the total attraction and the total repulsion in the swarm will balance. This is consistent with the earlier discussions on the motion of the swarm towards a minimum of $J(x)$ or basically to a minimum potential energy configuration. In fact, the result in (3.19) corresponds to a (local) minimum of the potential function $J(x)$.

Remark: The cumulative Lyapunov function V is only one way to quantify the cohesion/dispersion of the swarm. In other words, instead of V, we could equally well choose

$$\bar{V} = \frac{1}{2}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\|x_i - x_j\|^2,$$

which would quantify the inter-agent distances instead of the distances to the centroid. In some applications, where the centroid of the swarm is moving or the relative motion or relative positions of the agents to each other is more important than their relative motion to a predefined point such as their centroid, it may be better to use a function like $\bar{V}$. In fact, we arrive at the same conclusion using $\bar{V}$ since it can be shown that

$$\dot{\bar{V}} = -\frac{N}{2}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\left[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big)\right]\|x_i - x_j\|^2 = N\dot{V}.$$

■

Almost Constant Attraction and Unbounded Repulsion

In this section we discuss the attraction functions that satisfy $g_a(\|x_i - x_j\|) \to 0$ as $\|x_i - x_j\| \to \infty$. However, we assume also that

$$g_a(\|x_i - x_j\|) \geq \frac{a}{\|x_i - x_j\|}. \tag{3.21}$$

For the repulsion function we use the same type of functions as in the previous section, i.e., functions satisfying (3.17). An example of attraction/repulsion function $g(\cdot)$ satisfying the above assumptions could be stated as

$$g(x_i - x_j) = \left[\frac{a}{\|x_i - x_j\|} - \frac{b}{\|x_i - x_j\|^2}\right]\Big(x_i - x_j\Big)$$

which corresponds to the potential function

$$J(x) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left[a\|x_i - x_j\| - b\ln\Big(\|x_i - x_j\|\Big)\right]. \tag{3.22}$$

For this case we have the following theorem.

Theorem 4. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.4) *with an attraction/repulsion function* $g(\cdot)$ *which is odd, satisfies Assumption 1, and has almost constant attraction satisfying* (3.21) *and unbounded repulsion satisfying* (3.17). *As time progresses the* average *of the distances of the swarm members to the swarm centroid will satisfy*

$$e_{avg} \triangleq \frac{1}{N}\sum_{i=1}^{N}\|e_i\| \leq \frac{b}{a} \triangleq \varepsilon_{avg},$$

Proof: For this case we have

$$a\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\|x_i-x_j\|\leq\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}g_r(\|x_i-x_j\|)\|x_i-x_j\|^2.$$

Also, since

$$\|e_i\|=\frac{1}{N}\left\|\sum_{i=1}^{N}(x_i-x_j)\right\|\leq\frac{1}{N}\sum_{i=1}^{N}\|x_i-x_j\|,$$

holds (which is obtained using the equality in (3.15)) in a similar manner to earlier, one obtains

$$aN\sum_{i=1}^{N}\|e_i\|\leq bN(N-1)$$

which, on the other hand, implies that

$$\frac{1}{N-1}\sum_{i=1}^{N}\|e_i\|\leq\frac{b}{a}.$$

In other words, the *average* of the errors satisfies

$$e_{avg}\triangleq\frac{1}{N}\sum_{i=1}^{N}\|e_i\|\leq\frac{b}{a}\triangleq\varepsilon_{avg},$$

at equilibrium, implying cohesiveness of the swarm. ■

From the above result one can also deduce that $\|e_i\|$ will be ultimately bounded by the bound

$$\|e_i\|\leq\frac{Nb}{a}=N\varepsilon_{avg}.$$

We would like to emphasize that in the results obtained so far the bounds obtained (i.e., ε, ε_{rms}, and ε_{avg}) are independent of the number of individuals N. Therefore, it could be the case that as the number of individuals increase the density of the swarm may also increase. This might happen even for the case with unbounded repulsion, which guarantees that the individuals will not occupy the same point, but does not necessarily guarantee uniform swarm density. Such a behavior will not be consistent with real biological swarms. By creating a private or safety area for each agent, it is possible to account for the finite body sizes of the agents and also to achieve swarm size which increases with the number of agents implying, in a sense, more uniform swarm density. This will be discussed in the next section.

3.2.4 Agents with Finite Body Size

In this section, we will discuss how the potential functions could be chosen to handle finite body size or some private or safety area for the swarm members. In other words, the agents will be viewed as entities with finite body size instead of points without dimensions. In particular, we will consider individuals which are hyperspheres in the n-dimensional space.

Assume that all the agents have the same size and let η be the radius of the hypersphere representing the body size or private (safety) area of each agent. Let x_i be the center of the hypersphere for individual i. Then, in order for two individuals i and j not to collide $\|x_i - x_j\| > 2\eta$ needs to be satisfied. Note that this is not guaranteed to be the case given the attraction/repulsion functions considered in the preceding sections. The main reason for that is the fact that, even in the case of unbounded repulsion, for the repulsion function $g_r(\cdot)$ we have

$$\lim_{\|x_i - x_j\| \to 0^+} g_r(\|x_i - x_j\|)\|x_i - x_j\| = \infty.$$

Therefore, since as $\|x_i - x_j\| \to 0^+$ the repulsion becomes unbounded, the agents cannot occupy the same point, i.e., collisions between point-individuals are prevented. In the case in which the agents have finite body size, the attraction and repulsion can be taken from the outer surface of the body and the corresponding potential function modified accordingly. In particular, modifying the repulsion function to be of "hard-limiting" type satisfying

$$\lim_{\|x_i - x_j\| \to 2\eta^+} g_r(\|x_i - x_j\|)\|x_i - x_j\| = \infty,$$

will be sufficient, where η is the parameter representing the radius of the safety area, or the body size, of the individuals as mentioned above.

One repulsion function which satisfies the above condition is given by

$$g_r(\|x_i - x_j\|) = \frac{b}{(\|x_i - x_j\|^2 - 4\eta^2)^2}$$

for $\eta > 0$, which corresponds to the repulsive potential

$$J_r(\|x_i - x_j\|) = -\frac{b}{2(\|x_i - x_j\|^2 - 4\eta^2)}. \tag{3.23}$$

Note that with the assumption that initially all the agents are sufficiently far apart from each other, i.e., that we have $\|x_i(0) - x_j(0)\| > 2\eta$ for all $(i, j), j \neq i$, this type of repulsion function will guarantee that $\|x_i(t) - x_j(t)\| > 2\eta$ is satisfied for all t and all $(i, j), j \neq i$.

In contrast to the potential functions considered in the preceding section, the hard limiting repulsion function guarantees that the swarm will scale with the number of individuals leading to a more uniform density. To see this, first consider the two dimensional space $\mathbb{R}^2$. The private area of an individual is a disk with center x_i and radius η with occupation area equal to $A_i = \pi\eta^2$. Given the fact that $\|x_i(t) - x_j(t)\| > 2\eta$ for all t and all $(i, j), j \neq i$ and the safety areas of the swarm members are disjoint, the total area occupied by the swarm will be $A_{ts} = N\pi\eta^2$.

Assume that all the swarm members are "squeezed" cohesively as close as possible in an area (a disk) of radius r around the swarm center $\bar{x}$. Then, we have

$$\pi r^2 \geq N\pi\eta^2,$$

from which we obtain that a lower bound on the radius of the smallest circle which can enclose all the individuals is given by

$$r_{min} = \eta\sqrt{N}.$$

This, on the other hand, implies that in $\mathbb{R}^2$ the swarm will have a size which is always greater than $\eta\sqrt{N}$. This is important because it shows that the lower bound on the swarm size depends on N, implying that the swarm size will scale with the number of individuals. In particular, even the size of the smallest possible swarm will be greater than r_{min} because of the unoccupied area "lost" between the agents in the swarm. The most compact swarm is achieved when the individuals are located on a regular grid with the grid points as the edges of equilateral triangles with edge size equal to 2η and the total of $(N-2)$ triangles. Defining ρ as the density (the number of individuals per unit area/volume) of the swarm the above inequalities imply that

$$\rho \leq \frac{1}{\pi\eta^2}.$$

In other words, the density of the swarm is upper bounded implying that the swarm cannot become arbitrarily dense.

With similar analysis to above, one can show that on $\mathbb{R}^n$ for any n the lower bound on the swarm size is given by

$$r_{min} = \eta\sqrt[n]{N}.$$

This bound implies that as the dimension n of the state space gets larger, the relative effect of N on the swarm size gets smaller, which is an intuitively expected result. As in the two dimensional case, the smallest swarm occurs when the individuals are placed on a regular grid where each individual is located at the vertex of an equilateral shape (triangle in $\mathbb{R}^2$, tetrahedron in $\mathbb{R}^3$, etc.). Similar to the $n = 2$ case it can be shown that the density of the swarm is upper bounded by

$$\rho \leq \frac{\beta(n)}{\eta^n},$$

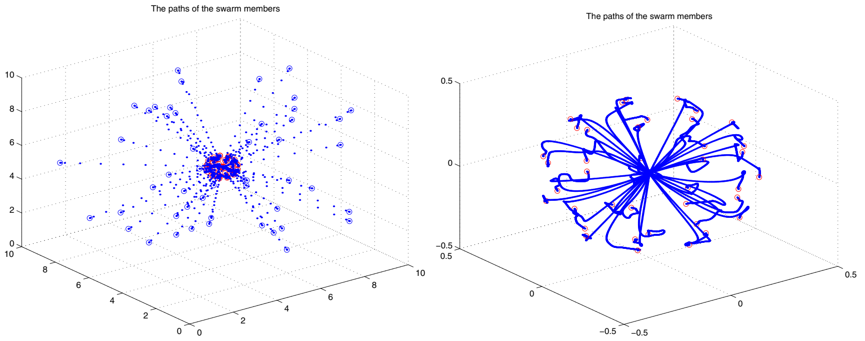
where $\beta(n)$ is a constant for a given n. In other words, it depends only on the dimension n of the state space.

Having relatively uniform swarm density is an important characteristic of real biological swarms and therefore, a desired characteristic of mathematical models of swarms. The discussion in this section shows that incorporation of hard-limiting type of repulsion functions will lead to a behavior which can be more consistent with biology. Moreover, in engineering applications creating a safety area around each individual might be more effective in avoiding collisions.

3.2.5 Simulation Examples

This section is devoted to illustrative numerical simulation examples that provide better insight into the dynamics of the swarm. Although the theory holds for arbitrary dimension n of the state space, in the examples below either $n = 2$ or $n = 3$ are used for easy visualization.

First, the case for linear attraction and bounded repulsion is considered. Figure 3.1(a) shows the paths of the agents in a swarm of $N = 51$ individuals with initial agent positions which are apart from each other, whereas Figure 3.1(b) illustrates the case in which the agents start very close to each other ($N = 60$ for this case). The initial and final positions of the agents are represented with circles while their paths are shown with dots. It is easily seen that, as expected, all the agents move toward each other in Figure 3.1(a) and away from each other in Figure 3.1(b) and form a cohesive swarm cluster with "comfortable" inter-agent distances. The center of the swarm is stationary for all time (although not shown in the plots).

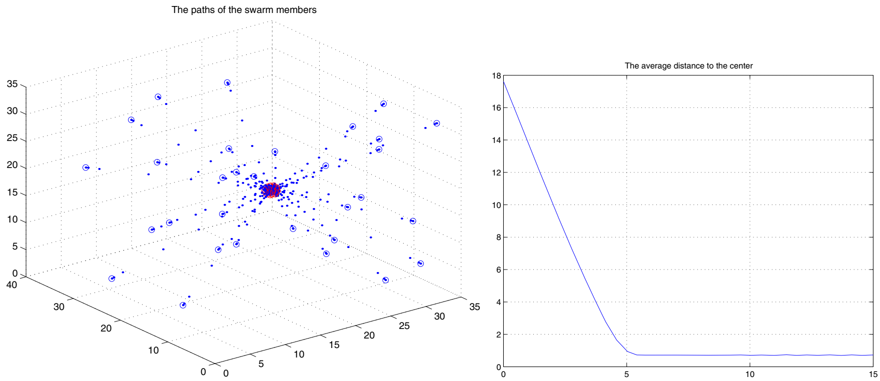


(a) Agents start apart from each other. (b) Agents start with close initial positions.

Fig. 3.1. Swarm with linear attraction and bounded repulsion between agents.

In these simulations, the potential function $J(x)$ in (3.6) with the corresponding attraction repulsion function $g(\cdot)$ in (3.7) with parameters $a = 1$, $b = 20$, and $c = 0.2$ was used. For these values of the parameters, the swarm members are expected to converge to a ball with radius $\varepsilon \approx 3.8$ around the centroid of the swarm. Note that the actual swarm size is much smaller than this since ε is a conservative bound, as discussed earlier.

Note that at their final positions, the distance between the agents in the swarm is less than the distance δ in Assumption 1 at which attraction and repulsion balance. This is expected since even though two agents are closer to each other than δ, they cannot push each other because the other members are pulling them in a direction opposite of their repulsion. Then the equilibrium occurs when the attraction and repulsion balance and this balance occurs on inter-agent distances less then δ. Similar results are obtained when different parameters are used in $g(\cdot)$. The main reason for this is the fact that all agents are affected by all other agents. If the model is modified such that agents are affected only by their neighbors, then inter-agent distances in the swarm would converge to δ.

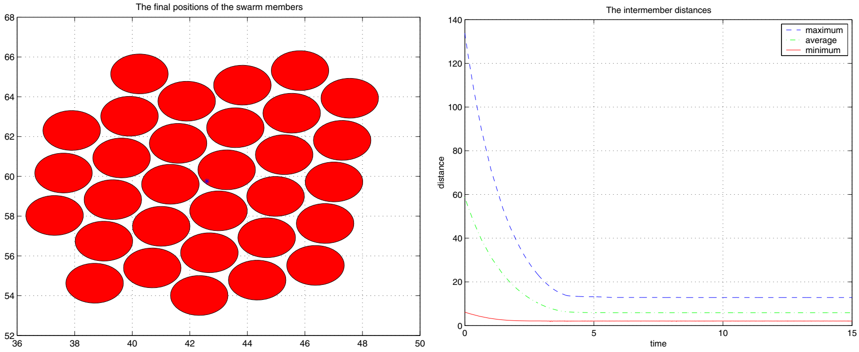


(a) Aggregation of the swarm.

(b) Average distance of the agents to $\bar{x}$.

Fig. 3.2. Almost constant attraction and unbounded repulsion ($N = 31$).

Next, the case of almost constant attraction and unbounded repulsion is considered with the potential function in (3.22) with parameters $a = b = 0.2$. The plot in Figure 3.2(a) shows the behavior of $N = 31$ agents with initial positions chosen at random. As one can see, the individuals form a cohesive cluster (around the centroid) as predicted by the theory. For this case, we have the bound $\varepsilon_{avg} = \frac{b}{a} = 1$ as the ultimate size of the swarm. Figure 3.2(b) shows the average e_{avg} of the distances of the individual positions to the swarm centroid. Note that the average converges to a value smaller than ε_{avg}, confirming the analytical derivations. The behavior of the swarm for the other two cases is similar.



(a) Final agent positions.

(b) Inter-agent distances.

Fig. 3.3. Hard-limiting repulsion ($N = 31$).

Finally, we consider the case of hard-limiting repulsion with potential function in (3.23) and $\eta = 1$. In other words, it is assumed that the agents have a safety area of radius $\eta = 1$ and therefore keep a distance of at least $2\eta = 2$ units apart from each other. Figure 3.3(a) shows the final positions of the agents for a swarm with $N = 31$ individuals in a two dimensional space. As can be seen from the figure, the agents do not collide with each other and are distributed in almost a grid like arrangement. Figure 3.3(b), on the other hand, shows the minimum, the average, and the maximum distances between the agents in the swarm. As one can see from the figure, the minimum distance between agents is greater than $2\eta = 2$. In contrast, although not shown in the current plots, as the number of individuals increases, while the minimum distance between agents continues to be greater than $2\eta = 2$, the maximum distance scales with the number of agents. This implies that the swarm size scales with the number of agents while the density of the swarm remains relatively constant (as expected from the discussions in the earlier sections). Having the swarm density nearly constant is an important feature of the real biological swarms and this shows that the swarm model with hard-limiting repulsion can represent that feature.

3.3 Social Foraging

3.3.1 Introduction

For the case of social foraging, in addition to the inter-agent interactions, the dynamics of the swarm will be affected by the environment. In order to incorporate the effect of the environment within the potential functions framework it (the environment) can also be represented by a potential function. Let $\sigma : \mathbb{R}^n \to \mathbb{R}$ be the potential which represents the environment. We will call this potential the "resource profile." It is a potential function/profile representing the resources in the environment which can be nutrients or some attractant or repellent substances (e.g., food/nutrients, pheromones laid by other individual, or toxic chemicals in biological swarms or goals, targets, obstacles or threads in engineering swarm applications). Assume that the areas that are minimum points are "favorable" to the agents in the swarm. In particular, for a point $y \in \mathbb{R}^n$ assume that $\sigma(y) < 0$ represents attractant or nutrient rich, $\sigma(y) = 0$ represents a neutral, and $\sigma(y) > 0$ represents a noxious environment at y. In the context of multi-agent (i.e., multi-robot) systems the profile $\sigma(\cdot)$ models the environment containing obstacles or threats to be avoided (analogous to toxic substances) and targets or goals to be moved towards (analogous to food). Note also that the resource profile $\sigma(\cdot)$ can be a combination of several sub-profiles.

Modeling the environment this way, the potential function J can be chosen as $J(x) = J_{foraging}(x)$ where

$$\begin{aligned} J_{foraging}(x) &= \sum_{i=1}^{N} \sigma(x_i) + J_{aggregation}(x) \\ &= \sum_{i=1}^{N} \sigma(x_i) + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[J_a\Big(\|x_i - x_j\|\Big) - J_r\Big(\|x_i - x_j\|\Big) \right]. \end{aligned} \tag{3.24}$$

Then, in light of equation (3.2) the control input for each individual $i = 1, \ldots, N$, can be calculated as

$$\begin{aligned} u_i &= -\nabla_{x_i}\sigma(x_i) - \sum_{j=1, j\neq i}^{N} \Big[\nabla_{x_i} J_a\Big(\|x_i - x_j\|\Big) - \nabla_{x_i} J_r\Big(\|x_i - x_j\|\Big)\Big] \\ &= -\nabla_{x_i}\sigma(x_i) - \sum_{j=1, j\neq i}^{N} \Big[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big)\Big](x_i - x_j). \qquad (3.25) \end{aligned}$$

Here, we assume that the aggregation potential $J_{aggregation}(x)$ and its related attraction/repulsion function $g(\cdot)$ satisfy the conditions stated in the preceding section. In particular, we assume that $g(\cdot)$ is odd and satisfies Assumption 1.

The term $-\nabla_{x_i}\sigma(x_i)$ represents the motion of the individuals towards regions with higher nutrient concentration in biological swarms and towards goals or targets in engineering swarms and away from regions with high concentration of toxic substances in biological swarms and away from threats or obstacles in engineering swarms. Note that the implicit assumption that the individuals know the gradient of the profile at their position is not very restrictive since it is known that some organisms such as bacteria are able to construct local approximations to gradients [193].

Similar to the case for aggregation in the preceding section, the objective here is to analyze the qualitative properties of the collective behavior (motions in n-space) of the individuals. Considering first the motion of the *centroid* of the swarm one can obtain

$$\begin{aligned} \dot{\bar{x}} &= -\frac{1}{N}\sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \Big[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big)\Big](x_i - x_j) \\ &= -\frac{1}{N}\sum_{i=1}^{N} \nabla_{x_i}\sigma(x_i). \qquad (3.26) \end{aligned}$$

This is because

$$\frac{1}{N}\sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \Big[g_a\Big(\|x_i - x_j\|\Big) - g_r\Big(\|x_i - x_j\|\Big)\Big](x_i - x_j) = 0,$$

holds due to the fact that inter-agent attraction/repulsion relationships are reciprocal (i.e., $g(\cdot)$ is an odd function of the form (3.5) we have $g(x_i - x_j) = -g(x_j - x_i)$ for all pairs (i, j)). The above equation implies that the centroid of the swarm moves along the *average* of the gradient of the resource profile evaluated at the current positions of the agents. However, this does not necessarily mean that it will converge to a minimum. Moreover, this does not imply that we can reach any conclusions about the motions of the individuals. In fact, the convergence properties of the swarm to minimum (or critical) points of the profile depends on the properties of the profile.

One can see that the collective behavior in (3.26) has a kind of averaging (filtering or smoothing) effect. This may be important if the resource profile $\sigma(\cdot)$ is a noisy function (or there is a measurement error or noise in the system as discussed in [103, 104]). In other words, if the resource profile $\sigma(\cdot)$ were a "noisy function" and the agents were moving individually (without inter-agent attraction/repulsion), then they could get stuck at a local minima, whereas if they swarm, since they are moving collectively, the other individuals will "pull" them out of such local minima. This in turn will lead to better performance by swarming agents compared to the performance of "selfish" agents as seen in some biological examples [103, 104].

For the foraging swarm we will consider only the case of attraction/repulsion functions which are continuous and have *linear attraction* as in (3.13) and *bounded repulsion* as in (3.14).

3.3.2 Swarm Cohesion Analysis

Before proceeding with analysis of the swarm behavior for different resource profiles, in this section we will analyze the cohesiveness of the swarm under some general conditions satisfied by several resource profiles. To this end, as before, let $e_i = x_i - \bar{x}$ denote the relative position between the position x_i of individual i and the centroid $\bar{x}$ of the swarm. The ultimate bound on the magnitude of e_i will again quantify the size of the swarm. Note that

$$\dot{e}_i = -\nabla_{x_i}\sigma(x_i) - \sum_{j=1,j\neq i}^{N}\left[a - g_r\Big(\|x_i - x_j\|\Big)\right](x_i - x_j) + \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j),$$

where $g_r(\cdot)$ is such that (3.14) is satisfied. Defining a Lyapunov function as $V_i = \frac{1}{2}\|e_i\|^2 = \frac{1}{2}e_i^\top e_i$, and since $e_i = \frac{1}{N}\sum_{j=1}^{N}(x_i - x_j)$ holds one can easily obtain

$$\begin{aligned}\dot{V}_i = {} & -aN\|e_i\|^2 + \sum_{j=1,j\neq i}^{N} g_r(\|x_i - x_j\|)(x_i - x_j)^\top e_i \\ & - \left[\nabla_{x_i}\sigma(x_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j)\right]^\top e_i. \end{aligned} \tag{3.27}$$

Now, if we can show that there is a constant ε such that for all $\|e_i\| > \varepsilon$ we have $\dot{V}_i < 0$, then we will guarantee that in that region $\|e_i\|$ is decreasing and eventually $\|e_i\| \leq \varepsilon$ will be achieved. With this in mind, we have two assumptions about the resource profile $\sigma(\cdot)$. Note that these two assumptions do not have to be satisfied simultaneously.

Assumption 2. *There exists a constant* $\bar{\sigma} > 0$ *such that*

$$\|\nabla_y \sigma(y)\| \leq \bar{\sigma}$$

for all y.

Assumption 3. *There exists a constant* $A_\sigma > -aN$ *such that*

$$\left[\nabla_{x_i}\sigma(x_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j)\right]^\top e_i \geq A_\sigma\|e_i\|^2$$

for all x_i *and* x_j.

Note that Assumption 2 requires only that the gradient of the resource profile $\sigma(\cdot)$ is bounded and is a very reasonable assumption that is satisfied with almost any realistic profile (e.g., plane and Gaussian profiles). In contrast, Assumption 3 is a more restrictive assumption. It requires the gradient of the profile at x_i to have a "large enough" component along e_i so that the effect of the profile does not prevent swarm cohesion. Therefore, it may be satisfied only by few profiles (e.g., a quadratic profile). With this in mind we state the following result.

Theorem 5. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*, or basically* $g_a(\|x_i - x_j\|) = a$ *and* $g_r(\|x_i - x_j\|)\|x_i - x_j\| \leq b$ *for some* $a > 0$, $b > 0$ *and all* $\|x_i - x_j\|$*). Then, as* $t \to \infty$ *we have* $x_i(t) \to B_\varepsilon(\bar{x}(t))$*, where*

$$B_\varepsilon(\bar{x}(t)) = \{y(t) : \|y(t) - \bar{x}(t)\| \leq \varepsilon\}$$

and

- *If Assumption 2 is satisfied, then*
$$\varepsilon = \varepsilon_1 = \frac{(N-1)}{aN}\left[b + \frac{2\bar{\sigma}}{N}\right],$$
- *If Assumption 3 is satisfied, then*
$$\varepsilon = \varepsilon_2 = \frac{b(N-1)}{aN + A_\sigma}.$$

Proof: **Case 1:** From the $\dot{V}_i$ equation in (3.27) we obtain

$$\begin{aligned}\dot{V}_i &\leq -aN\|e_i\|^2 + \sum_{j=1, j\neq i}^{N} g_r(\|x_i - x_j\|)\|x_i - x_j\|\|e_i\| \\ &\quad + \left\|\nabla_{x_i}\sigma(x_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j)\right\|\|e_i\| \\ &\leq -aN\|e_i\|\left[\|e_i\| - \frac{b(N-1)}{aN} - \frac{2\bar{\sigma}(N-1)}{aN^2}\right],\end{aligned}$$

which implies that as long as $\|e_i\| > \varepsilon_1$ we have $\dot{V}_i < 0$. Above, the inequality in the last line is obtained using the fact that $g_r(\|x_i - x_j\|)\|x_i - x_j\| \leq b$ and the inequality

$$\left\|\nabla_{x_i}\sigma(x_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j)\right\| \leq \frac{2\bar{\sigma}(N-1)}{N},$$

which follows from Assumption 2.

Case 2: Similarly, using Assumption 3, one can show that $\dot{V}_i$ satisfies

$$\dot{V}_i \leq -(aN + A_\sigma)\|e_i\|\left[\|e_i\| - \frac{b(N-1)}{aN + A_\sigma}\right].$$

Therefore, one can conclude that as long as $\|e_i\| > \varepsilon_2$ we have $\dot{V}_i < 0$. ■

This result is analogous to Theorem 2 in Section 3.2 and is important because it shows the cohesiveness of the swarm and provides a bound on the swarm size, defined as the radius of the hyperball centered at $\bar{x}(t)$ and containing all the individuals. Therefore, in order to analyze the collective behavior of the swarm we need to consider the motion of the centroid.

In species that engage in social foraging it has been observed that the individuals in swarms desire to be close, but not too close, to other individuals. In the mean time, they want to find more food. The balance between these desires determines the size of the swarm. The dynamics in (3.25) capture this by having an inter-individual attraction/repulsion term, and also a term due to the environment (or the nutrient profile) affecting their motion. In the results above, the resulting swarm sizes depend on the inter-individual attraction/repulsion parameters (a and b) and the parameters of the nutrient profile ($\bar{\sigma}$ and A_σ). Moreover, the dependence on these parameters makes intuitive sense. Larger attraction (larger a) leads to a smaller swarm size, larger repulsion (larger b) leads to a larger swarm size, larger $\bar{\sigma}$ (fast changing landscape) leads to a larger swarm. These concepts are found in foraging theory in biology and model the balance of the desire of the individuals to "stick together" with the desire to "get more food" that was created by evolutionary forces. Note that for Assumption 3 to be satisfied we have the condition $A_\sigma > -aN$. The threshold $A_\sigma = -aN$ is the point at which the inter-individual attraction is not guaranteed to "hold the swarm together" since it is counterbalanced by the repulsion from the resource profile. In other words, beyond that threshold, the repulsion from a critical point of the resource profile (i.e., toxic substances or an immediate danger) is so intense that the "desire to keep away from the critical point of the resource profile" dominates the "desire to stick together." Therefore, if this condition is not satisfied we cannot anymore guarantee cohesiveness of the swarm, i.e., it can happen that the swarm members move arbitrary far from each other. This helps to quantify the inherent balance between the sometimes conflicting desires for swarm cohesiveness and for following cues from the environment to find food. Such behavior can be seen in, for example, fish schools when a predator attacks the school. In that case the fish move very fast in all directions away from the predator [191] resulting in a scattering behavior sometimes called explosion.

Note that the desire of the individuals to "stick together" depends on the inter-individual attraction parameter a and the number of individuals N. This is somewhat consistent with *some* biological swarms, where it has been observed that individuals are attracted more to larger (or more crowded) swarms (even though that attraction may not be linearly proportional to the number of individuals).

One issue to note here is that as N gets large both ε_1 and ε_2 approach constant values. This implies that for a large N the individuals will form a cohesive swarm of a constant size independent of the number of the individuals and the characteristics of the profile. Unfortunately, as mentioned earlier for aggregating swarms, this is not biologically very realistic.

The above result is an asymptotic result, i.e., $x_i(t) \to B_\varepsilon(\bar{x}(t))$ as $t \to \infty$. However, from stability theory it is well known that for any $\varepsilon^* > \varepsilon$, $x_i(t)$ will enter $B_{\varepsilon^*}(\bar{x}(t))$ in a finite time. In other words, it can be shown that a swarm of any size ε^* a little larger than ε will be formed in a finite time.

In the following sections, we will analyze the behavior of the swarm on different resource profiles. In particular, we will discuss plane, quadratic, Gaussian, and multi-modal Gaussian profiles.

3.3.3 Swarm Motion in Various Resource Profiles

Motion along a Plane Resource Profile

In this section we assume that the resource profile is described by a *plane* equation of the form

$$\sigma(y) = a_\sigma^\top y + b_\sigma, \tag{3.28}$$

where $a_\sigma \in \mathbb{R}^n$ and $b_\sigma \in \mathbb{R}$. One can see that the gradient of the profile is given by

$$\nabla_y \sigma(y) = a_\sigma$$

and Assumption 2 holds with $\bar{\sigma} = \|a_\sigma\|$. However, note also that

$$\nabla_{x_i}\sigma(x_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla_{x_j}\sigma(x_j) = a_\sigma - \frac{1}{N}\sum_{j=1}^{N} a_\sigma = 0$$

for all i, implying that the last term in (3.27) vanishes. Therefore, from Theorem 5 (and also from Theorem 2 in Section 3.2) one can deduce that for this profile the bound on the swarm size is given by

$$\varepsilon = \varepsilon_p = \frac{b(N-1)}{aN} < \frac{b}{a}.$$

Note also that for this case we have

$$\dot{\bar{x}}(t) = -a_\sigma,$$

which implies that the centroid of the swarm will be moving with the constant velocity vector $-a_\sigma$ (and eventually will diverge towards infinity where the minimum of the profile occurs).

The motions in this section can be viewed as a model of a foraging swarm that moves in a constant direction (while keeping its cohesiveness) with a constant speed such as the one considered in [107]. Another view of the system in this section could be as a model of a multi-agent system in which the autonomous agents move in a formation with a constant speed. In fact, transforming the system to e_i coordinates we obtain

$$\dot{e}_i = \sum_{j=1, j\neq i}^{N} g(e_i - e_j), i = 1, \ldots, N,$$

which is exactly the model of an aggregating swarm discussed in Section 3.2. Therefore, all the results obtained in Section 3.2 for aggregation apply for e_i. In particular, we have $\dot{e}_i(t) \to 0$ as $t \to \infty$. In other words, the swarm converges to a constant configuration or a formation (i.e., constant relative positions) and moves with a constant speed in the direction of $-a_\sigma$.

Quadratic Resource Profiles

In this section, we will consider a *quadratic* resource profile given by

$$\sigma(y) = \frac{A_\sigma}{2} \|y - c_\sigma\|^2 + b_\sigma, \tag{3.29}$$

where $A_\sigma \in \mathbb{R}$, $b_\sigma \in \mathbb{R}$, and $c_\sigma \in \mathbb{R}^n$. Note that this profile has a global extremum (either a minimum or a maximum depending on the sign of A_σ) at $y = c_\sigma$. Its gradient at a point $y \in \mathbb{R}^n$ is given by

$$\nabla_y \sigma(y) = A_\sigma (y - c_\sigma).$$

Assume that $A_\sigma > -aN$. Then, with few manipulations one can show that for this profile Assumption 3 holds with strict equality. Therefore, the result of Theorem 5 holds with the bound

$$\varepsilon_q = \varepsilon_2 = \frac{b(N-1)}{aN + A_\sigma}.$$

Now, let us analyze the motion of the centroid $\bar{x}$. Substituting the gradient in the equation of motion of $\bar{x}$ given in Eq. (3.26) we obtain

$$\dot{\bar{x}} = -A_\sigma (\bar{x} - c_\sigma).$$

Defining the distance between the centroid $\bar{x}$ and the extremum point c_σ as $e_\sigma = \bar{x} - c_\sigma$, we have

$$\dot{e}_\sigma = -A_\sigma e_\sigma,$$

which implies that as $t \to \infty$ we have $e_\sigma(t) \to 0$ if $A_\sigma > 0$ and that $e_\sigma(t) \to \infty$ if $A_\sigma < 0$ and $e_\sigma(0) \neq 0$. Therefore, we have the following result.

Lemma 2. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is given by* (3.29). *As* $t \to \infty$ *we have*

- *If* $A_\sigma > 0$, *then* $\bar{x}(t) \to c_\sigma$ *(i.e., the centroid of the swarm converges to the global minimum* c_σ *of the profile), or*

- *If $A_\sigma < 0$ and $\bar{x}(0) \neq c_\sigma$, then $\bar{x}(t) \to \infty$ (i.e., the centroid of the swarm diverges from the global maximum c_σ of the profile).*

Note that this result holds for any A_σ (i.e., we do not need the assumption $A_\sigma > -aN$). Note also that for the case with $A_\sigma > 0$ for any finite $\varepsilon^* > 0$ (no matter how small) it can be shown that $\|\bar{x}(t) - c_\sigma\| < \varepsilon^*$ is satisfied in a finite time. In other words, $\|\bar{x}\|$ enters any ε^* neighborhood of c_σ in a finite time. In contrast, for the case with $A_\sigma < 0$ and $\bar{x}(0) \neq c_\sigma$ for any $D > 0$ (no matter how large) it can be shown that $\|\bar{x}(t) - c_\sigma\| > D$ is satisfied in a finite time, implying that $\|\bar{x}\|$ exits any bounded D-neighborhood of c_σ in a finite time. If $A_\sigma < 0$ and $\bar{x}(0) = c_\sigma$, on the other hand, then $\bar{x}(t) = c_\sigma$ for all t. In other words, for this case the swarm will be either "trapped" around the maximum point because of the inter-individual attraction (i.e., desire of the individuals to be close to each other) or will disperse in all directions if the inter-individual attraction is not strong enough (i.e., $A_\sigma < -aN$). Note, however, that even if they disperse, the centroid $\bar{x}$ will not move and will be stationary at c_σ. As mentioned above, such a dispersal behavior can be seen in fish schools when attacked by a predator [191]. In other words, the effect of the presence of a predator can be represented by a large intensity quadratic repellent resource profile. Nevertheless, we would like to also mention that the situation $\bar{x}(t) = c_\sigma, t \geq 0$ corresponds to an unstable equilibrium of the system and even very small perturbation which results in $\bar{x}(t) \neq c_\sigma$ will lead to $\bar{x}(t)$ diverging away from c_σ.

Here, we did not consider the $A_\sigma = 0$ case. This is because if $A_\sigma = 0$, then the profile is uniform everywhere and $\nabla_y \sigma(y) = 0$ for all $y \in \mathbb{R}^n$. Therefore, the existence of the profile does not affect the motion of the individuals and stability analysis is reduced to the analysis of simple aggregation presented in Section 3.2.

Combining the results of Theorem 5 and Lemma 2 together with the above observations gives us the following result.

Theorem 6. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function $g(\cdot)$ as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile $\sigma(\cdot)$ of the environment is given by* (3.29) *and that $A_\sigma > -aN$. Then, the following hold*

- *If $A_\sigma > 0$, then for any $\varepsilon^* > \varepsilon_q$ all individuals $i = 1, \ldots, N$, will enter $B_{\varepsilon^*}(c_\sigma)$ in a finite time,*
- *If $A_\sigma < 0$ and $\bar{x}(0) \neq c_\sigma$, then for any $D < \infty$ all individuals $i = 1, \ldots, N$, will exit $B_D(c_\sigma)$ in a finite time.*

This result is important because it gives finite time convergence to nutrient rich regions (targets, goals) of the resource profile or divergence from toxic regions (threads, obstacles) of the profile of *all* agents in the swarm.

Now, assume that instead of the quadratic resource profile in (3.29) we have a profile which is a sum of quadratic functions. In other words, assume that the resource profile is given by

$$\sigma(y) = \sum_{i=1}^{M} \frac{A_{\sigma i}}{2} \|y - c_{\sigma i}\|^2 + b_\sigma,$$

where $A_{\sigma i} \in \mathbb{R}$, and $c_{\sigma i} \in \mathbb{R}^n$ for all $i = 1, \ldots, M$, and $b_\sigma \in \mathbb{R}$. Its gradient at a point y is given by

$$\nabla_y \sigma(y) = \sum_{i=1}^{M} A_{\sigma i}(y - c_{\sigma i}).$$

Defining

$$A_\sigma = \sum_{i=1}^{M} A_{\sigma i}$$

and

$$c_\sigma = \frac{\sum_{i=1}^{M} A_{\sigma i} c_{\sigma i}}{\sum_{i=1}^{M} A_{\sigma i}}$$

we obtain

$$\nabla_y \sigma(y) = A_\sigma (y - c_\sigma),$$

which is exactly the same as above. The point c_σ is the point of the unique extremum of the combined resource profile function. Therefore, the above results will directly transfer without any modification. This also is true because it can be shown that

$$\sum_{i=1}^{M} \frac{A_{\sigma i}}{2} \|y - c_{\sigma i}\|^2 = \frac{A_\sigma}{2} \|y - c_\sigma\|^2 + C,$$

where C is a constant.

Quadratic resource profiles are rather simple profiles and the results in this section are intuitively expected. However, note also that more complicated resource profiles can be locally modeled (or approximated) as quadratic in regions near extremum points. In the following sections, we will discuss resource profiles which are not necessarily quadratic. Moreover, later we will allow the environmental resource profile to have multiple extremum points.

Gaussian Resource Profiles

In this section, we consider resource profiles that are described by a Gaussian-type of equation,

$$\sigma(y) = -\frac{A_\sigma}{2} \exp\left(-\frac{\|y - c_\sigma\|^2}{l_\sigma}\right) + b_\sigma, \tag{3.30}$$

where $A_\sigma \in \mathbb{R}$, $b_\sigma \in \mathbb{R}$, $l_\sigma \in \mathbb{R}^+$, and $c_\sigma \in \mathbb{R}^n$. Note that this profile also has the unique extremum (either a global minimum or a global maximum depending on the sign of A_σ) at $y = c_\sigma$. Its gradient is given by

$$\nabla_y \sigma(y) = \frac{A_\sigma}{l_\sigma}(y - c_\sigma) \exp\left(-\frac{\|y - c_\sigma\|^2}{l_\sigma}\right).$$

Calculating the time derivative of the centroid of the swarm by using (3.26) one can obtain

$$\dot{\bar{x}} = -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}(x^i - c_\sigma)\exp\left(-\frac{\|x^i - c_\sigma\|^2}{l_\sigma}\right).$$

Compared to the quadratic case, here we cannot write $\dot{\bar{x}}$ as a function of $e_\sigma = \bar{x} - c_\sigma$. This is because of the nonlinearity of the gradient of the profile. However, intuitively we would expect that we still should be able to get some results similar to the ones in the preceding section. To this end we note that Assumption 2 is satisfied with

$$\bar{\sigma} = \frac{|A_\sigma|}{\sqrt{2l_\sigma}}\exp\left(-\frac{1}{2}\right).$$

Therefore, Theorem 5 holds and we know that as $t \to \infty$ all the individuals will converge to (and stay within) the

$$\varepsilon_G = \varepsilon_1 = \frac{(N-1)}{aN}\left[b + \frac{|A_\sigma|}{N}\sqrt{\frac{2}{l_\sigma}}\exp\left(-\frac{1}{2}\right)\right]$$

neighborhood of the (mobile) centroid $\bar{x}$.

Now, we have to analyze the motion of $\bar{x}$ in order to determine the overall behavior of the swarm.

Lemma 3. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is given by* (3.30)*. Then, as* $t \to \infty$ *we have*

- *If* $A_\sigma > 0$*, then* $\|e_\sigma(t)\| \leq \max_{i=1,\ldots,N}\|e_i(t)\| \triangleq e_m(t)$*,*
- *If* $A_\sigma < 0$ *and* $\|e_\sigma(0)\| > e_m(0)$ *(here we assume that* $x_i(0) \neq x_j(0)$ *for all pairs of individuals* $(i,j), j \neq i, 1 \leq i,j \leq N$ *and therefore* $e_m(0) > 0$*), then* $\|e_\sigma\| \to \infty$*.*

Proof: To start with, let $V_\sigma = \frac{1}{2}e_\sigma^\top e_\sigma$. Then, its derivative along the motion of the swarm is given by

$$\begin{aligned}\dot{V}_\sigma &= -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i - c_\sigma\|^2}{l_\sigma}\right)(x_i - c_\sigma)^\top e_\sigma \\ &= -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i - c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|^2 - \frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i - c_\sigma\|^2}{l_\sigma}\right)e_i^\top e_\sigma,\end{aligned}$$

where we used the fact that $x_i - c_\sigma = e_i + e_\sigma$.

Case 1: $A_\sigma > 0$*:*

Bounding $\dot{V}_\sigma$ from above we obtain

$$\begin{aligned}
\dot{V}_\sigma &\leq -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|^2+\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_i\|\|e_\sigma\| \\
&\leq -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|\left[\|e_\sigma\|-\frac{\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_i\|}{\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)}\right] \\
&\leq -\frac{A_\sigma}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|\Big[\|e_\sigma\|-e_m\Big],
\end{aligned}$$

where $e_m=\max_{i=1,\ldots,N}\|e_i\|$. The above inequality implies that as long as $\|e_\sigma(t)\|>e_m(t)$, i.e., the minimum point c_σ is outside the swarm boundary, then the centroid of the swarm will be moving toward it. Therefore, as $t\longrightarrow\infty$ we will asymptotically have $\|e_\sigma(t)\|\leq e_m(t)$, i.e., c_σ will be within the swarm.

Case 2: $A_\sigma<0$:

With analysis similar to the case 1 above it can be shown that

$$\dot{V}_\sigma\geq\frac{|A_\sigma|}{Nl_\sigma}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|\Big[\|e_\sigma\|-e_m\Big],$$

which implies that we have $\dot{V}_\sigma>0$. In other words, if $\|e_\sigma\|>e_m$, then $\|e_\sigma\|$ will increase. From Theorem 5 we have that e_m is decreasing. Therefore, since by hypothesis $\|e_\sigma(0)\|>e_m(0)$ we have that $\dot{V}_\sigma>0$ holds. Now, given any large but fixed $D>0$ and $\|e_\sigma(t)\|\leq D$ we have

$$\exp\left(-\frac{\|x_i-c_\sigma\|^2}{l_\sigma}\right)\|e_\sigma\|\Big[\|e_\sigma\|-e_m\Big]\geq\exp\left(-\frac{(D^2+\varepsilon_3^2)}{l_\sigma}\right)D\Big[D-\varepsilon_3\Big]>0,$$

implying that

$$\dot{V}_\sigma\geq\frac{|A_\sigma|}{l_\sigma}\exp\left(-\frac{(D^2+\varepsilon_G^2)}{l_\sigma}\right)D\Big[D-\varepsilon_G\Big]>0$$

from which using (a corollary to) the Chetaev Theorem [136] we conclude that $\|e_\sigma\|$ will exit the D-neighborhood of c_σ. ■

Note that the result in Lemma 3 makes intuitive sense. If we have a hole (i.e., a minimum) it guarantees that the agents will "gather" around its lowest potential point (as expected). If we have a hill (i.e., a maximum) and all the agents are located on one side of the hill, it guarantees that the agents diverge from it (as expected). If there is a hill, but the individuals are spread around it, then we can conclude neither convergence nor divergence. This is because it can happen that the swarm may move to one side and diverge or the inter-individual attraction forces can be counterbalanced by the inter-individual repulsion combined with the repulsion from the hill so that the swarm does not move away from the hill.

The above result in Lemma 3, together with the result in Theorem 5, allow us to state the following.

Theorem 7. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is given by* (3.30)*. Then, as* $t \to \infty$ *we have*

- *If* $A_\sigma > 0$*, then all individuals* $i = 1, \ldots, N$*, will enter (and stay within)* $B_{2\varepsilon_G}(c_\sigma)$*,*
- *If* $A_\sigma < 0$ *and* $\|e_\sigma(0)\| \geq e_m(0)$*, then all individuals* $i = 1, \ldots, N$*, will exit* $B_D(c_\sigma)$ *for any fixed* $D > 0$*.*

For the case $A_\sigma > 0$, Theorem 5 states that the swarm will have a maximum size of ε_G, i.e., $\|e_\sigma\| \leq \varepsilon_G$ for all $i = 1, \ldots, N$. Lemma 3 states that the swarm centroid will converge to the e_m and therefore to the ε_G neighborhood of c_σ, i.e., $\|e_\sigma\| \leq e_m \leq \varepsilon_G$. Combining these two bounds we obtain the $2\varepsilon_G$ in the first case in Theorem 7.

Theorem 7 is a parallel of Theorem 6. However, here we have a weaker result since for the $A_\sigma > 0$ case we cannot guarantee that $\bar{x}(t) \to c_\sigma$. Moreover, the region around c_σ in which the individuals converge is larger ($2\varepsilon_G$) compared to the region in Theorem 6 (ε_q).

Multimodal Gaussian Resource Profiles

Now, we will consider a resource profile which is a combination of *Gaussian* resource profiles. In other words, we will consider the profiles given by

$$\sigma(y) = -\sum_{i=1}^{M} \frac{A_{\sigma i}}{2} \exp\left(-\frac{\|y - c_{\sigma i}\|^2}{l_{\sigma i}}\right) + b_\sigma, \tag{3.31}$$

where $c_{\sigma i} \in \mathbb{R}^n$, $l_{\sigma i} \in \mathbb{R}^+$, $A_{\sigma i} \in \mathbb{R}$ for all $i = 1, \ldots, M$, and $b_\sigma \in \mathbb{R}$. Note that since the $A_{\sigma i}$'s can be positive or negative there can be both hills and valleys leading to more irregular combined resource profiles.

The gradient of the profile at a point $y \in \mathbb{R}^n$ is given by

$$\nabla_y \sigma(y) = \sum_{i=1}^{M} \frac{A_{\sigma i}}{l_{\sigma i}} (y - c_{\sigma i}) \exp\left(-\frac{\|y - c_{\sigma i}\|^2}{l_{\sigma i}}\right).$$

Note that for this profile, Assumption 2 is satisfied with

$$\bar{\sigma} = \sum_{i=1}^{M} \frac{|A_{\sigma i}|}{\sqrt{2 l_{\sigma i}}} \exp\left(-\frac{1}{2}\right).$$

Therefore, from Theorem 5 we have

$$\varepsilon_{mG} = \varepsilon_1 = \frac{(N-1)}{aN} \left[b + \frac{1}{N} \sum_{i=1}^{M} |A_{\sigma i}| \sqrt{\frac{2}{l_{\sigma i}}} \exp\left(-\frac{1}{2}\right) \right]$$

as the bound on the swarm size. In other words, as $t \to \infty$ we will have $x_i(t) \to B_{\varepsilon_{mG}}(\bar{x}(t))$, where ε_{mG} is as given above.

Using the profile gradient equation we can write the equation of motion of the swarm centroid $\bar{x}$ as

$$\dot{\bar{x}} = -\frac{1}{N}\sum_{j=1}^{M}\frac{A_{\sigma j}}{l_{\sigma j}}\sum_{i=1}^{N}(x_i - c_{\sigma j})\exp\left(-\frac{\|x_i - c_{\sigma j}\|^2}{l_{\sigma j}}\right).$$

It is not obvious from this equation how the centroid $\bar{x}$ will move. Therefore, for this type of profile it is not easy to show convergence of the individuals to a minimum of the profile for the general case. However, under some conditions it is possible to show convergence to the vicinity of a particular $c_{\sigma j}$ (if $c_{\sigma j}$ is the center of a valley) or divergence from the neighborhood of a particular $c_{\sigma j}$ (if $c_{\sigma j}$ is the center of a hill).

Lemma 4. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is given by* (3.31)*. Moreover, assume that for some* $k, 1 \leq k \leq N$*, we have*

$$\|x_i(0) - c_{\sigma k}\| \leq h_k\sqrt{l_{\sigma k}}$$

for some h_k *and for all* $i = 1,\ldots,N$*, and that for all* $j = 1,\ldots,M, j \neq k$ *we have*

$$\|x_i(0) - c_{\sigma j}\| \geq h_j\sqrt{l_{\sigma j}}$$

for some $h_j, j = 1,\ldots,M, j \neq k$ *and for all* $i = 1,\ldots,N$*. (This means that the swarm is sufficiently close to* $c_{\sigma k}$ *and sufficiently far from other* $c_{\sigma j}, j \neq k$*.) Moreover, assume that*

$$\frac{A_{\sigma k}}{\sqrt{l_{\sigma k}}}h_k\exp\left(-h_k^2\right) > \frac{1}{\alpha}\sum_{j=1,j\neq k}^{M}\frac{|A_{\sigma j}|}{\sqrt{l_{\sigma j}}}h_j\exp\left(-h_j^2\right),$$

is satisfied for some $0 < \alpha < 1$*. Then, for* $e_{\sigma k} = \bar{x} - c_{\sigma k}$ *as* $t \rightarrow \infty$ *we will have*

- *If* $A_{\sigma k} > 0$*, then* $\|e_{\sigma k}(t)\| \leq \varepsilon_{mG} + \alpha h_k\sqrt{l_{\sigma k}}$
- *If* $A_{\sigma k} < 0$ *and* $\|e_{\sigma k}(0)\| \geq e_m(0) + \alpha h_k\sqrt{l_{\sigma k}}$*, then* $\|e_{\sigma k}(t)\| \geq \varepsilon_{mG} + \alpha h_k\sqrt{l_{\sigma k}}$*, where* $e_m = \max_{i=1,\ldots,N}\|e_i\|$.

Proof: Let $V_{\sigma k} = \frac{1}{2}e_{\sigma k}^\top e_{\sigma k}$ be the Lyapunov function.

Case 1: $A_{\sigma k} > 0$**:** Taking the derivative of $V_{\sigma k}$ along the motion of the swarm one can show that

$$\dot{V}_{\sigma k} \leq -\frac{A_{\sigma k}}{Nl_{\sigma k}}\sum_{i=1}^{N}\exp\left(-\frac{\|x_i - c_{\sigma k}\|^2}{l_{\sigma k}}\right)\|e_{\sigma k}\|\left[\|e_{\sigma k}\| - e_m - \alpha h_k\sqrt{l_{\sigma k}}\right],$$

which implies that we have $\dot{V}_{\sigma k} < 0$ as long as $\|e_{\sigma k}\| > e_m + \alpha h_k\sqrt{l_{\sigma k}}$, and from Theorem 5 we know that as $t \rightarrow \infty$ we have $e_m(t) \leq \varepsilon_{mG}$.

Case 2: $A_{\sigma k} < 0$**:** Similar to above, for this case it can be shown that

$$\dot{V}_{\sigma k} \geq \frac{|A_{\sigma k}|}{N l_{\sigma k}} \sum_{i=1}^{N} \exp\left(-\frac{\|x_i - c_{\sigma k}\|^2}{l_{\sigma k}}\right) \|e_{\sigma k}\| \left[\|e_{\sigma k}\| - e_m - \alpha h_k \sqrt{l_{\sigma k}}\right],$$

which implies that if $\|e_{\sigma k}\| > e_m + \alpha h_k \sqrt{l_{\sigma k}}$, then we have $\dot{V}_\sigma > 0$. In other words, $\|e_{\sigma k}\|$ will increase. From Theorem 5 we have that e_m is decreasing. Therefore, since by hypothesis $\|e_{\sigma k}(0)\| > e_m(0) + \alpha h_k \sqrt{l_{\sigma k}}$ we have that $\dot{V}_\sigma > 0$ holds at $t = 0$. Now, consider the boundary $\|e_{\sigma k}\| = \varepsilon_{mG} + h_k \sqrt{l_{\sigma k}}$. It can be shown that on the boundary we have

$$\dot{V}_\sigma \geq \frac{|A_{\sigma k}| h_k (1-\alpha) \left(\varepsilon_{mG} + h_k \sqrt{l_{\sigma k}}\right) \exp\left(-h_k^2\right)}{\sqrt{l_{\sigma k}}} > 0,$$

from which once again using (a corollary to) the Chetaev Theorem we conclude that $\|e_{\sigma k}\|$ will exit the $\varepsilon_{mG} + h_k \sqrt{l_{\sigma k}}$-neighborhood of $c_{\sigma k}$. ■

Now, as in the preceding section we can combine this result (i.e., Lemma 4) together with Theorem 5 to obtain the following theorem.

Theorem 8. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is given by* (3.31). *Assume that the conditions of Lemma 4 hold. Then, as* $t \to \infty$ *all individuals will*

- *Enter the hyperball* $B_{\varepsilon_5}(c_{\sigma k})$, *where* $\varepsilon_5 = 2\varepsilon_{mG} + \alpha h_k \sqrt{l_{\sigma k}}$, *if* $A_{\sigma k} > 0$, *or*
- *Leave the* $h_k \sqrt{l_{\sigma k}}$*-neighborhood of* $c_{\sigma k}$, *if* $A_{\sigma k} < 0$.

In order for the results in Theorem 8 to make sense, we need

$$2\varepsilon_{mG} + \alpha h_k \sqrt{l_{\sigma k}} < h_k \sqrt{l_{\sigma k}}.$$

This implies that

$$\varepsilon_{mG} < \left(\frac{1-\alpha}{2}\right) h_k \sqrt{l_{\sigma k}}$$

is needed which sometimes may not be easy to satisfy. However, one issue to note is that ε_{mG} is a very conservative bound which holds for extreme situations and pathological cases as well. In reality, the actual size of the swarm is typically much smaller than the bound. Therefore, effectively, ε_{mG} can be replaced with $e_m(\infty) < \varepsilon_{mG}$ and it may be easier to satisfy the above condition.

3.3.4 Analysis of Individual Behavior in a Cohesive Swarm

The results in the previous sections specify whether the swarm will diverge or converge, and if it converges they specify in which regions of the profile it will converge, together with bounds on the swarm size. The results above do not provide information about the ultimate behavior of the individuals. In other words, they do not specify whether the individuals will eventually stop moving or will end up in oscillatory motions within the specified regions. In this section, we will investigate the ultimate behavior of the agents. In particular, we will analyze the ultimate behavior of the agents in a quadratic profile with $A_\sigma > 0$, a Gaussian profile with $A_\sigma > 0$, and in a multimodal Gaussian profile with the conditions of Lemma 4 for the $A_{\sigma k} > 0$ case satisfied.

As was the case in aggregating swarms we will show that for the above mentioned cases as $t \longrightarrow \infty$ the state $x(t)$ converges to Ω_e, i.e., eventually all the individuals stop moving.

Theorem 9. *Consider a foraging swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.25) *with inter-agent attraction/repulsion function* $g(\cdot)$ *as given in* (3.5) *which is odd, satisfies Assumption 1, and has linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that the resource profile* $\sigma(\cdot)$ *of the environment is one of the following*

- *A quadratic profile in Eq. (3.29) with* $A_\sigma > 0$*, or*
- *A Gaussian profile in Eq. (3.30) with* $A_\sigma > 0$*, or*
- *A multimodal Gaussian profile in Eq. (3.31) with conditions of Lemma 4 for the* $A_{\sigma k} > 0$ *case satisfied.*

Then, as $t \longrightarrow \infty$ *we have the state* $x(t) \longrightarrow \Omega_e$*.*

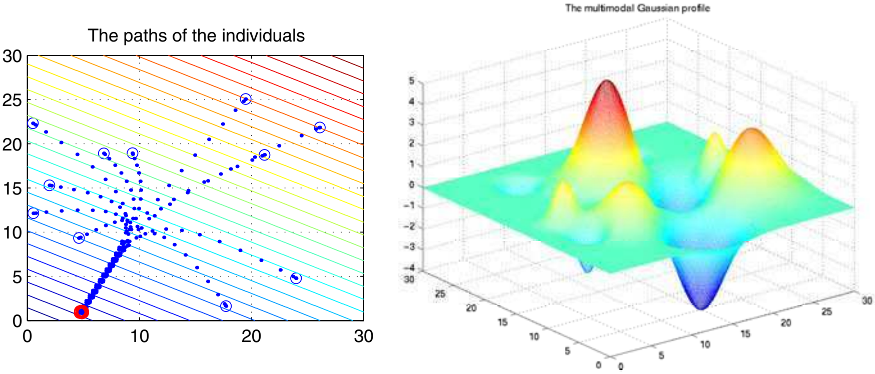
Proof: The proof is very similar to the proof of Theorem 1. ■

One issue to note here is that for the cases excluded in the above theorem, i.e., for the plane profile, quadratic profile with $A_\sigma < 0$, Gaussian profile with $A_\sigma < 0$, and the multi-modal Gaussian profile for $A_\sigma < 0$ case or not necessarily satisfying the conditions of Lemma 4, the set $\Omega_c = \{x : J(x) \leq J(x(0))\}$ may not be compact. Therefore, we cannot apply the LaSalle's Invariance Principle. Moreover, since they are (possibly) diverging, intuitively we do not expect them to stop their motion. Furthermore, note that for the plane profile we have $\Omega_e = \emptyset$. In other words, there is no equilibrium for the swarm moving in a plane profile.

3.3.5 Simulation Examples

This section provides illustrative numerical simulation examples of a foraging swarm moving in the profiles discussed in the preceding sections. The dimension of the state space is once more selected as $n = 2$ for ease of visualization of the results. In all the simulations, the region $[0,30] \times [0,30]$ in the space is used as the simulation region and the simulations are performed with $N = 11$ agents. The attraction/repulsion function $g(\cdot)$ with linear attraction and bounded repulsion in (3.7) is used in the simulations with parameters $a = 0.01$, $b = 0.4$, and $c = 0.01$ for most of the simulations, and $a = 0.1$ for some of them.

Figure 3.4(a) shows the swarm dynamics in a plane resource profile with $a_\sigma = [0.1, 0.2]^\top$. One easily can see that, as expected, the swarm moves along the negative gradient $-a_\sigma$ exiting the simulation region and heading toward unboundedness.



(a) Response for plane resource profile.
(b) Multimodal Gaussian profile.

Fig. 3.4. Plane profile result and multimodal Gaussian profile.

Note that initially some of the individuals move in a direction opposite to the negative gradient. This is because the inter-agent attraction is much stronger than the intensity of the resource profile.

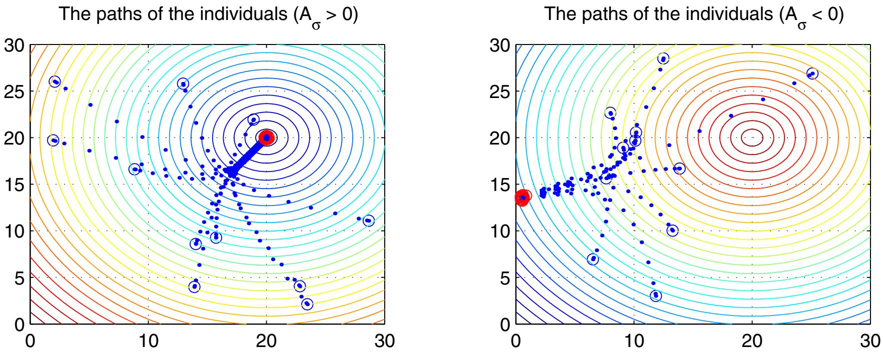


Fig. 3.5. Response for quadratic resource profile.

Figure 3.5 shows the dynamics of the swarm in a quadratic resource profile with extremum at $c_\sigma = [20,20]^\top$ and magnitude $A_\sigma = \pm 0.02$. The plot on the left of the figure shows the paths of the individuals and the centroid of the swarm for the case $A_\sigma > 0$, whereas the one on the right is for the case $A_\sigma < 0$. Once more, it is observed that the results obtained are in parallel with the expectations from the analysis performed in the preceding sections. Note also that the centroid $\bar{x}$ of the swarm converges to the minimum of the profile c_σ for the $A_\sigma > 0$ case, and diverges from the maximum for the $A_\sigma < 0$ case.

Results of similar nature are obtained also for the Gaussian resource profile as shown in Figure 3.6. The plot on the left of the figure shows the paths of the

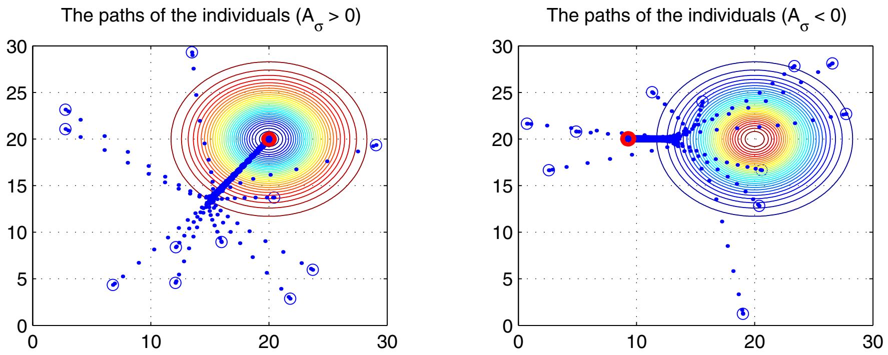


Fig. 3.6. Response for Gaussian resource profile.

individuals and the centroid of the swarm for the case $A_\sigma > 0$, whereas the one on the right is for the case $A_\sigma < 0$. The point $c_\sigma = [20,20]^\top$ is the extremum of the resource profile for these simulations as well. The other parameters of the profile are $A_\sigma = \pm 2$ and $l_\sigma = 20$. Note that for the $A_\sigma > 0$ case, even though in theory we could not prove that $\bar{x}(t) \rightarrow c_\sigma$, in simulations we observe that this is apparently the case. This was systematically observed in all the simulations performed with Gaussian resource profiles.

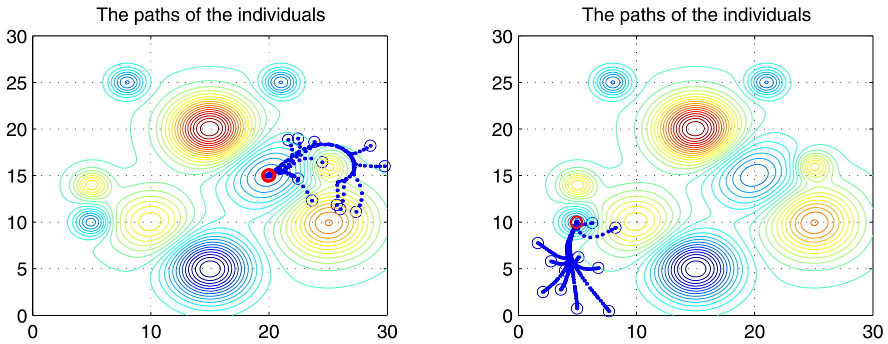


Fig. 3.7. The response for a multimodal Gaussian profile (initial positions close to a minimum).

The resource profile in Figure 3.4(b) is used in order to illustrate the dynamics of the swarm in a multimodal Gaussian profile. Note that it has several minima and maxima. Its global minimum is located at $[15,5]^\top$ with a “magnitude” of 4 and a spread of 10. The plot in Figure 3.7 shows two example runs with initial member positions nearby a local minimum, and it shows convergence of the entire swarm to that minimum. The attraction parameter a was chosen to be $a = 0.01$ for this case.

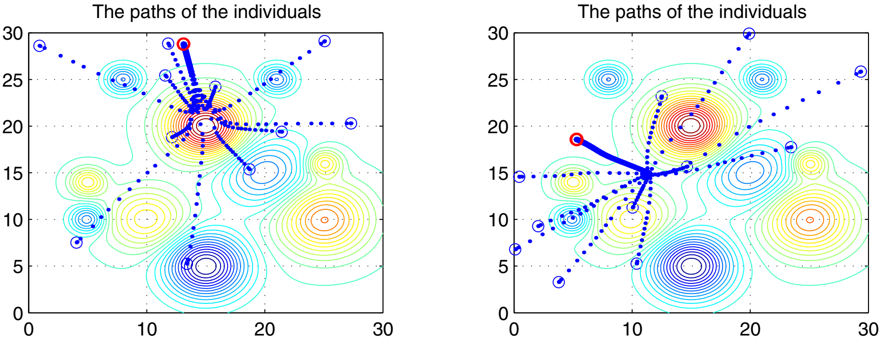


Fig. 3.8. The response for a multimodal Gaussian profile.

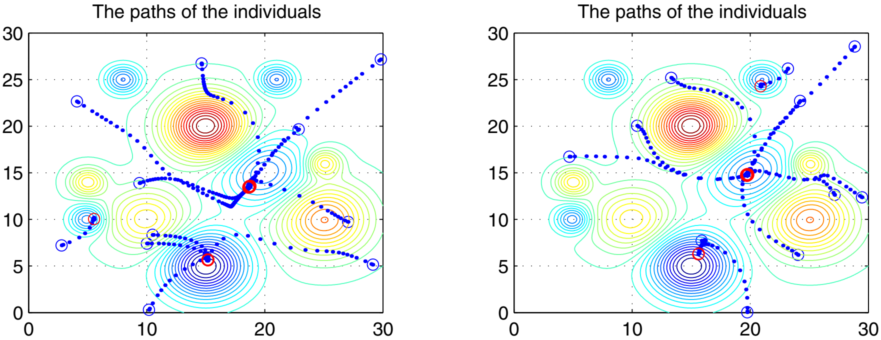


Fig. 3.9. The response for a multimodal Gaussian profile.

Figure 3.8, on the other hand, illustrates the case in which the value of the attraction parameter was increased to $a = 0.1$. One can see that the attraction is so strong that the individuals climb gradients to form a cohesive swarm. For this and similar cases, the manner in which the overall swarm will behave (where it will move) depends on the initial distribution of the agents in the swarm. For these two runs they happened to be located such that the swarm diverged. For some other simulation runs (not presented here) with different initial conditions the entire swarm converges to either a local or global minimum. Figure 3.9 shows two runs for which the value of the attraction parameter was decreased again to $a = 0.01$ and the positions of the agents are initialized all over the region. For both of the simulations one can see that the swarm fails to form a cohesive cluster since the initial positions of the individuals are such that they move to nearby local minima and the attraction is not strong enough to "pull them out" of these valleys. This causes formation of several groups or clusters of individuals at different locations of the space. For these reasons, the centroid $\bar{x}$ of the swarm does not converge to any minimum. Note, however, that this is expected since for this case the conditions of Theorem 8 are not satisfied. Nevertheless, the result of Theorem 5 still holds. The only issue is that ε_{mG} is large and contains all the region within which all the agents eventually converge and remain within. Finally, note also that during their motion to the groups, the agents try to avoid climbing gradients and this results in motions resembling the motion of individuals in real biological swarms.

3.4 Formation Control

The formation control problem is an important problem in multi-agent systems and robotics that has seen recent substantial progress. It can simply be described as the problem of choosing the control inputs for the agents such that they asymptotically form and maintain an arbitrary predefined geometric shape. Given the single integrator agent model in (3.1) and the control input in the form of (3.2) it is straightforward to design a potential function $J(x)$ (and therefore the corresponding attraction/repulsion function $g(\cdot)$) so that the formation control problem is solved at least locally. With this objective, we will assume that the attraction/repulsion functions $g(\cdot)$ are pair dependent. In other words, we will assume that the potential function $J(x)$ can be denoted as

$$J_{formation}(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \Big[J_{ija}\Big(\|x_i - x_j\| \Big) - J_{ija}\Big(\|x_i - x_j\| \Big) \Big] \tag{3.32}$$

and the corresponding control input is given by

$$\begin{aligned} u_i &= - \sum_{j=1, j\neq i}^{N} \Big[\nabla_{x_i} J_{ija}\Big(\|x_i - x_j\| \Big) - \nabla_{x_i} J_{ijr}\Big(\|x_i - x_j\| \Big) \Big] \\ &= - \sum_{j=1, j\neq i}^{N} \Big[g_{ija}\Big(\|x_i - x_j\| \Big) - g_{ijr}\Big(\|x_i - x_j\| \Big) \Big] (x_i - x_j). \end{aligned} \tag{3.33}$$

Note that for all pairs (i, j) it is still assumed that the pair dependent attraction/repulsion functions

$$g_{ij}(y) = y\Big[g_{ija}(y) - g_{ijr}(y) \Big] \tag{3.34}$$

are odd satisfying $g_{ij}(x_i - x_j) = -g_{ji}(x_j - x_i)$ and satisfy Assumption 1 possibly with pair dependent equilibrium distances δ_{ij}. Note, however, that $(x_i - x_j) = (x_k - x_l)$ does not necessarily imply $g_{ij}(x_i - x_j) = g_{kl}(x_k - x_l)$. In other words, in this new potential function, the attraction and repulsion functions, and therefore the equilibrium distance δ_{ij} for different pairs of individuals, can be different.

For the potential function in (3.6) the above simply mean that the parameters a, b, and c of the potential function will be pair dependent (i.e., they will be of the form a_{ij}, b_{ij}, and c_{ij}) as

$$J(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{a_{ij}}{2} \|x_i - x_j\|^2 + \frac{b_{ij} c_{ij}}{2} \exp\left(-\frac{\|x_i - x_j\|^2}{c_{ij}} \right) \right]. \tag{3.35}$$

Here the control input for agent i will be of the form

$$u_i = - \sum_{j=1, j \neq i}^{N} \left[a_{ij} - b_{ij} \exp\left(-\frac{\|x_i - x_j\|^2}{c_{ij}} \right) \right] (x_i - x_j). \tag{3.36}$$

Let us assume that the desired formation is *uniquely* specified a priori by *formation constraints* of the form

$$\|x_i - x_j\| = d_{ij},$$

for all $(i, j), j \neq i$ and it is required that the formation is achieved (i.e., stabilized) from any initial position of the agents. Here, the term "uniquely" is used somewhat loosely since with constraints of the above form (where only relative distances are used) a formation can be specified uniquely only in terms of relative inter-individual arrangement and rotation or translation of the whole swarm will not destroy the geometrical shape of the formation. If rotation and translation is not allowed, then different types of formation constraints need to be specified. In particular, leader based strategies, or constraints including desired global reference coordinates for the agents can be deployed. By allowing global reference coordinate points for the agents the local minima problem (inherent in the potential function based approaches) may also be resolved. However, in many realistic situations the agents may not possess global coordinate information. Finally, note also that in order for the problem to be solvable at all, the formation constraints should not be conflicting and the formation should be feasible.

With the above requirements in mind, the idea is to choose each of the potential functions and therefore the corresponding attraction/repulsion functions $g_{ij}(\cdot)$ such that $\delta_{ij} = d_{ij}$ for every pair of individuals (i, j). Here, δ_{ij} is the distance at which the attraction and repulsion between agents i and j balance (see Assumption 1). This, in turn, results in the fact that the potential function $J(x)$ for the swarm (3.32) has a minimum achieved at the desired formation.

Remark: Note that under the stated assumptions for the formation control case considered here, i.e., the swarm with dynamics given in (3.1) with control input given by (3.33), all the theorems stated for aggregating swarms in Section 3.2 still hold (some of them possibly with simple modifications). In other words, the centroid of the swarm $\bar{x}$ is stationary for all time ($\dot{\bar{x}} = 0$ for all t–see Lemma 1), and as time progresses all the agents in the swarm stop motion (as $t \to \infty$, $\dot{x}_i = 0$ for all i–see Theorem 1). Moreover, provided that the attraction is linear and the repulsion bounded (i.e., the conditions in (3.13) and (3.14) are satisfied as in Theorem 2) all the agents converge to a hyperball $B_\varepsilon(\bar{x}) = \{x : \|x - \bar{x}\| \leq \varepsilon\}$, where $\varepsilon = \frac{b_{max}}{a_{min}}$, $a_{min} = \min_{1 \leq i,j \leq N}\{a_{ij}\}$, and $b_{max} = \max_{1 \leq i,j \leq N}\{b_{ij}\}$. The same is true for the results with other types of attraction/repulsion functions (Theorems 3 and 4). ■

From the above remark, one can easily deduce the following result.

Corollary 1. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.33) *with pair dependent inter-agent attraction/repulsion functions* $g_{ij}(\cdot)$ *as given in* (3.34) *which are odd, satisfy Assumption 1, and have linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Assume that* $g_{ij}(\cdot)$ *are chosen such that the distance* δ_{ij} *at which the inter-agent attractions and repulsions balance satisfy* $\delta_{ij} = d_{ij}$*, where* d_{ij} *are the desired formation distances. Then, the equilibrium at the desired formation is locally asymptotically stable. Moreover, if* $g_{ij}(\cdot)$ *are such that* $J(x)$ *has unique minimum at the desired formation, then asymptotic stability holds globally.*

The above result is by nature local since, as mentioned before, the potential function based methods suffer from the local minima problem. In other words, there exist local minima of the potential function $J(x)$ which do not correspond to the desired formation and since the motion of the agents is along the gradient there is a chance that the system will converge to such a minimum instead of the global minimum corresponding to the desired formation. Convergence to the desired formation is only guaranteed if the initial positions of the agents are “sufficiently close” to that configuration. In other words, the desired formation has a region of attraction whose size may be different for different applications. If the potential function $J(x)$ is such that it has a single (unique) minimum at the desired formation, then Corollary 1 implies global asymptotic stability of the desired formation. $J(x)$ might have unique minimum for some simple formations (such as forming equilateral triangle by three agents) or can be designed intelligently such that to have unique minimum. However, in that case global desired position coordinates for the agents also need to be included in the potential function.

In the case when the swarm has to achieve the formation, and as a formation track a given reference trajectory $\{x_r, \dot{x}_r\}$ which is known (or estimated by all the agents), the control inputs of the agents can be chosen as in (3.9) or as in (3.10) and provided the other conditions of Corollary 1 are satisfied, then formation stabilization and maintenance as well as tracking of the desired trajectory will be guaranteed. In the next section, we will discuss a similar problem in which the agents do not know the velocity x_r of the reference trajectory.

3.4.1 Simulation Examples

In this section, the dynamics of the swarm model for the formation control will be investigated with a simple numerical simulation. Assume that there are six agents in the swarm which are required to create a formation of an equilateral triangle with three of the agents in the middle of each edge and distances between two neighboring agents equal to 1. For this case, the potential functions (or the corresponding attraction/repulsion functions) for each pair of individuals are chosen such that the global potential function a minimum at the desired formation. This is done by choosing $g_{ij}(\cdot)$'s such that the equilibrium distances are one of $\delta_{ij} = 1$, $\delta_{ij} = 2$, or $\delta_{ij} = \sqrt{3}$ for different pairs (i, j) of individuals depending on their relative location in the desired formation. Figure 3.10 shows the trajectories of the agents as well

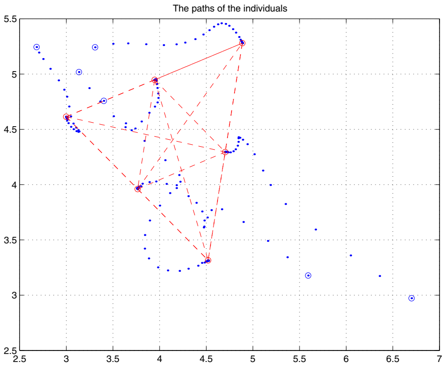


Fig. 3.10. Equilateral triangle formation of 6 agents.

as the formed geometric shape with initial positions chosen at random. As one can see the agents move and create the required formation while avoiding collisions in accordance with the expectations since we used unbounded repulsion.

3.5 Swarm Tracking

In this section, we will discuss the problem in which a swarm of agents with single integrator dynamics in (3.1) is required to catch or intercept a moving target and surround (or enclose or capture) it possibly forming a geometric formation around it. Let us denote with $x_t \in \mathbb{R}^n$ the position vector of the target, which is moving based on some motion equation of the form

$$\dot{x}_t = g_t(x_t, t).$$

If the target dynamics $g_t(x_t, t)$ and in particular its velocity $\dot{x}_t$ is available, then one can use the procedure discussed in the Section 3.4 in order to solve the problem. Therefore, in this section we will assume that the velocity $\dot{x}_t$ is unknown; however it is bounded $\|\dot{x}_t\| \leq \gamma_t$ for some known bound $\gamma_t > 0$. Note that such an assumption is very realistic since any realistic agent has a bounded velocity. We also assume that the agents know the relative position of the target x_t. This is not a restrictive assumption since there are sensors (such as laser scanners) for measuring relative positions.

In order to satisfy both the tracking and formation control specifications, we consider potential functions $J(x, x_t)$ which are composed of two parts–the inter-agent interactions (or formation control) part and the agent-target interaction (or tracking) part. In particular, we consider potential functions of the form

$$\begin{aligned} J(x,x_t) &= W_t \sum_{i=1}^{N} J_{it}\Big(\|x_i - x_t\|\Big) + W_f J_{formation}(x) \qquad (3.37)\\ &= W_t \sum_{i=1}^{N} J_{it}\Big(\|x_i - x_t\|\Big) + W_f \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \Big[J_{ija}\Big(\|x_i - x_j\|\Big) - J_{ija}\Big(\|x_i - x_j\|\Big)\Big] \end{aligned}$$

where $J_{it}(\|x_i - x_t\|)$ is the potential between agent i and the target. The second term in the potential is the same type of potential as considered in Section 3.4 for formation control. In particular, we choose pair dependent potential functions since the agents are required to form some kind of formation around the target (although this is not the primary objective and potentials similar to those used in Section 3.2 for aggregation will also work). With the above form for the potential function, each agent takes care of the tracking task by itself and keeps certain distances between itself and its neighbors. For the tracking part, it is required that $J_{it}(\|x_i - x_t\|)$ is attractive towards the target so that the agents catch up with the target, encircle/enclose it, and track (move together with) it. The formation part $J_{formation}(x)$, on the other hand, is selected so that it has a minimum at the desired formation and all the other assumptions for formation achievement discussed in Section 3.4 are satisfied.

Here, note that in general the functions $J_{it}(\|x_i - x_t\|)$ can be chosen differently for different agents (although we will use the same function for all agents i here). However, there are constraints that these functions should satisfy. First of all, note that since $J_{it}(\cdot)$ is a function of the distance $\|x_i - x_t\|$, due to the chain rule there always exists a corresponding function $h_{it} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ such that the gradient $\nabla_y J_{it}(\|y\|)$ can be expressed as

$$\nabla_y J_{it}(\|y\|) = y h^{it}(\|y\|). \qquad (3.38)$$

We have also the following assumption about $J_{it}(\|x_i - x_t\|)$.

Assumption 4. *The potential functions* $J_{it}(\|x_i - x_t\|)$ *are such that the corresponding functions* $h_{it}(\cdot)$ *(in* $\nabla_y J_{it}(\|y\|) = y h_{it}(\|y\|)$*) satisfy* $h_{it}(\|y\|) \geq 0$ *for all* y *and* $h_{it}(\|y\|) = 0$ *is satisfied only for* $\|y\| = 0$.

With the assumptions above, we choose the control laws so that each agent is moving based on the equation

$$u_i = -\alpha \nabla_{x_i} J(x,x_t) - \beta \operatorname{sign}(\nabla_{x_i} J(x,x_t)) \qquad (3.39)$$

for all $i = 1,...,N$ where $\alpha \geq 0$ and $\beta \geq \gamma_t$ are positive constants, and $\operatorname{sign}(\cdot)$ is the signum function that operates elementwise for a vector $y \in \mathbb{R}^n$. The gain β is chosen such that it satisfies $\beta \geq \gamma_t$, where γ_t is the known bound on the (unknown) velocity of the target.

Before starting to analyze the performance of the system we highlight a few useful properties of the potential functions. The gradient of the $J(x,x_t)$ with respect to the position x_i of agent i can be expressed as

$$\nabla_{x_i} J(x,x_t) = W_t (x_i - x_t) h_{it}(\|x_i - x_t\|) + W_f \sum_{j=1, j\neq i}^{N} (x_i - x_j) g_{ij}(\|x_i - x_j\|) \qquad (3.40)$$

whereas its gradient with respect to the position x_t of the target is given by

$$\nabla_{x_t} J(x,x_t) = -W_t \sum_{i=1}^{N} (x_i - x_t) h_{it}(\|x_i - x_t\|) \tag{3.41}$$

By observing the equalities in (3.40) and (3.41), the equality in (3.41) can be rewritten as

$$\nabla_{x_t} J(x,x_t) = -\sum_{i=1}^{N} \nabla_{x_i} J(x,x_t) + W_f \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} (x_i - x_j) g_{ij}(\|x_i - x_j\|) \tag{3.42}$$

Moreover, the second term is zero since the attraction/repulsion functions $(x_i - x_j)g_{ij}(\cdot)$ are odd (see Assumption 1) resulting in a the symmetry/reciprocity in the interaction between individuals (see the proof of Lemma 1 seen earlier in Section 3.2), which results in the equality

$$\nabla_{x_t} J(x,x_t) = -\sum_{i=1}^{N} \nabla_{x_i} J(x,x_t). \tag{3.43}$$

Now, we can state the following result.

Theorem 10. *Consider a swarm consisting of agents with dynamics in* (3.1) *and with control input in* (3.39) *with pair dependent inter-agent attraction/repulsion functions* $g_{ij}(\cdot)$ *as given in* (3.34) *which are odd, satisfy Assumption 1, and have linear attraction and bounded repulsion (i.e., satisfies the conditions* (3.13) *and* (3.14)*). Moreover, assume that the tracking potentials* $J_{it}(\|x_i - x_t\|)$ *satisfy Assumption 4. Then, as* $t \longrightarrow \infty$ *we will have*

$$x_t \in conv\{x_1, x_2, \ldots, x_N\}$$

where $conv\{x_1, x_2, \ldots, x_N\}$ *denotes the convex hull of the positions of the agents in the swarm (implying that the agents will enclose the target).*

Proof: The time derivative of potential function J is given by

$$\dot{J} = \sum_{i=1}^{N} [\nabla_{x_i} J(x,x_t)]^\top \dot{x}_i + [\nabla_{x_t} J(x,x_t)]^\top \dot{x}_t \tag{3.44}$$

Substituting the agent dynamics in (3.1) and the control input in (3.39) as well as the condition in (3.43) in the $\dot{J}$ equation, one obtains

$$\begin{aligned}\dot{J} &= -\sum_{i=1}^{N} [\nabla_{x_i} J(x,x_t)]^T [\alpha \nabla_{x_i} J(x,x_t) + \beta \text{sign}(\nabla_{x_i} J(x,x_t))] - \sum_{i=1}^{N} [\nabla_{x_i} J(x,x_t)]^\top \dot{x}_t \\ &\leq -\alpha \sum_{i=1}^{N} \|\nabla_{x_i} J(x,x_t)\|^2 - \beta \sum_{i=1}^{N} \|\nabla_x J(x,x_t)\| + \gamma_t \sum_{i=1}^{N} \|\nabla_{x_i} J(x,x_t)\| \end{aligned} \tag{3.45}$$

Since we have $\beta \geq \gamma_t$, the time derivative of J is bounded by

$$\dot{J} \leq -\alpha \sum_{i=1}^{N} \|\nabla_{x_i} J(x,x_t)\|^2 \tag{3.46}$$

This equation implies that as time tends to infinity, we have $\dot{J} \to 0$, which indicates that J converges to a constant value implying that the system converges to a constant configuration corresponding to a minimum of J. This is because (3.46) also implies that $\|\nabla_{x_i} J(x,x_t)\| \to 0$ for all i. Also, from (3.43) we see that we have $\|\nabla_{x_t} J(x,x_t)\| \to 0$. In other words, as $t \to \infty$ we have

$$(x,x_t) \to \Omega \subset \{(x,x_t) | \dot{J} = 0\}$$

where

$$\Omega = \{(x,x_t) | \nabla_{x_t} J(x,x_t) = 0, \nabla_{x_i} J(x,x_t) = 0, i = 1,...,N\}$$

Then, from (3.43) we have

$$-W_t \sum_{i=1}^{N} (x_i - x_t) h_{it}(\|x_i - x_t\|) = 0$$

Rearranging this equation, we obtain

$$\sum_{i=1}^{N} x_i h_{it}(\|x_i - x_t\|) = x_t \sum_{i=1}^{N} h_{it}(\|x_i - x_t\|) \tag{3.47}$$

which is guaranteed to be achieved as $t \to \infty$. This is an important observation because it provides a relation between the position of the target and the position of the agents at equilibrium. Then, since from Assumption 4 we have $h_{it}(\|x_i - x_t\|) \geq 0$ and from the fact that it cannot be the case that $h_{it}(\|x_i - x_t\|) = 0$ simultaneously (i.e., we have at least one i for which we have $h_{it}(\|x_i - x_t\|) > 0$) we have $\sum_{i=1}^{N} h_{it}(\|x_i - x_t\|) \neq 0$ using which one obtains

$$x_t = \frac{\sum_{i=1}^{N} x_i h_{it}(\|x_i - x_t\|)}{\sum_{i=1}^{N} h_{it}(\|x_i - x_t\|)}.$$

Defining

$$\eta_i = \frac{h_{it}(\|x_i - x_t\|)}{\sum_{i=1}^{N} h_{it}(\|x_i - x_t\|)}, i = 1, \ldots, N,$$

the position of the target can be represented as

$$x_t = \sum_{i=1}^{N} \eta_i x_i.$$

Here, note that for all y and for all agents i the corresponding η_i satisfy $0 \leq \eta_i \leq 1$ and their sum is given by $\sum_{i=1}^{N} \eta_i = 1$ implying that as $t \to \infty$ we will have $x_t \to conv\{x_1, x_2, \ldots, x_N\}$, where $conv\{x_1, x_2, \ldots, x_N\}$ is the convex hull of the positions of the agents. In other words, as $t \to \infty$ the agents will "surround" or "enclose" the target, and this concludes the proof. ■

3.5.1 Simulation Examples

This section presents illustrative simulations of the swarm tracking technique discussed above. It is assumed that the target moves in a $n = 2$ dimensional space with the (unknown) dynamics

$$\begin{bmatrix} \dot{x}_{t_1} \\ \dot{x}_{t_2} \end{bmatrix} = \begin{bmatrix} 0.25 + 0.3\sin(t) \\ 1.9\sin(0.25t) \end{bmatrix}.$$

In (3.39) $\alpha = 0.1$ and $\beta = 2$ were used as controller parameters in all the simulations.

The tracking potential $J_{it}\left(\|x_i - x_t\|\right)$ in (3.37) is chosen as

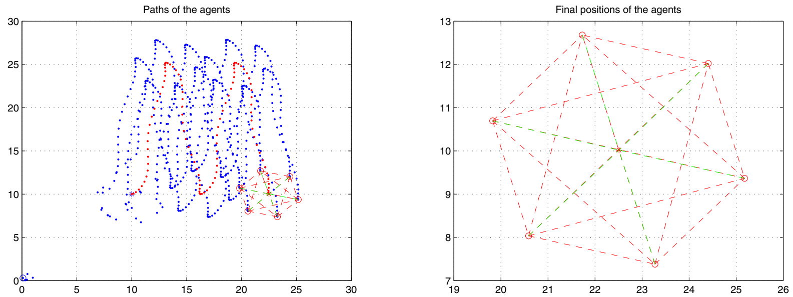
$$J_{it}\left(\|x_i - x_t\|\right) = \frac{1}{4}\|x_i - x_t\|^4$$

whose gradient is given by

$$\nabla_{x_i} J_{it}\left(\|x_i - x_t\|\right) = \|x_i - x_t\|^2 (x_i - x_t)$$

from which we have $h_{it}\left(\|x_i - x_t\|\right) = \|x_i - x_t\|^2$ which satisfies the conditions stated, including Assumption 4. The weights of the tracking potential and aggregatin/formation potential were chosen as $W_t = 0.6$ and $W_f = 1 - W_t = 0.4$.

Figure 3.11 shows the results for a simulation in which the agents were not required to form a predefined geometrical formation around the target. Instead, the objective was to have them aggregate around the target and follow it in this manner. For this simulation, for inter-agent interactions the aggregation potential in (3.6) with parameters $a = 1$, $b = 20$, and $c = 10$ were used. Figure 3.11(a) presents the trajectories of the agents and the target, whereas Figure 3.11(b) presents their positions at the instant at which the simulation was stopped. The position of the target



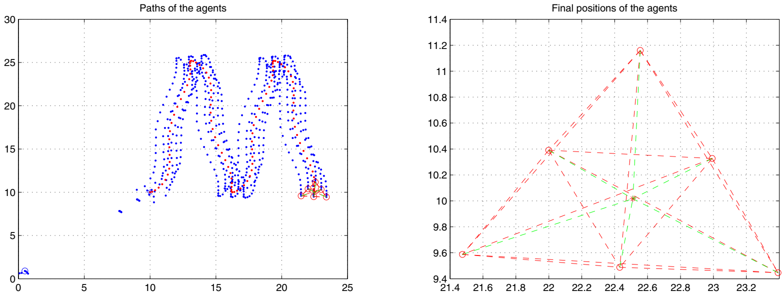
(a) Paths of target and agents.

(b) Final positions of target and agents.

Fig. 3.11. Enclose target (no formation required with $N = 6$).

is denoted by a star (located at the center of the swarm), while the positions of the agents are denoted with circles. As one can see, for this case, as predicted by theory, the agents enclose the target as it moves and forms a regular polygon (hexagon in this case since there are $N = 6$ agents in the swarm) around it.

Figure 3.12 shows the results for a simulation in which it was desired that a swarm of $N = 6$ agents forms an equilateral triangle around the target (although it is not the main objective–the main objective is to surround the target). For this simulation, for inter-agent interactions we use the aggregation potential in (3.35) with parameters $b_{ij} = 20$ and $c_{ij} = 10$, and a_{ij} calculated as $a_{ij} = b_{ij} \exp\left(-\frac{d_{ij}^2}{c_{ij}}\right)$, where d_{ij} is the desired inter-agent distance in the desired formation. Figure 3.12(a) shows the trajectories of the agents and the target, whereas Figure 3.12(b) shows their positions at the instant at which the simulation was stopped. The figures show that the objective of enclosing of the target by the agents was achieved, while the formation objective (which is not necessarily the primary objective) was achieved with a small error.



(a) Agent and target trajectories. (b) Final positions of target and agents.

Fig. 3.12. Enclose target (equilateral triangle formation with $N = 6$).

3.6 Further Issues

In this section we will discuss further issues that can be considered for swarms consisting of agents with dynamics which can be represented by the single integrator model in (3.1).

3.6.1 General Neighborhood Topology

In the swarms considered in this chapter (and in general in this book) the neighborhood topology of the agents can be represented as a complete graph. In other words, it was assumed that all the agents can sense the (relative) positions of all the other agents. This assumption could be relaxed and most of the results will hold with possibly simple modifications for the more general case in which the neighborhood in the swarm is represented with a bi-directional (reciprocal) strongly connected graph. In other words, most of the results will hold with possibly simple modifications (excluding possibly the formation control and swarm tracking cases) for the case in which every agent is not necessarily a neighbor of every other agent but there is path from every agent to every other agent in the swarm through intermediate agents. For the case of formation control, further issues need to be considered since usually having a strongly connected neighborhood topology may not be sufficient to uniquely specify a geometric formation.

3.6.2 Non-reciprocal Agent Interactions

The assumption that the attraction repulsion functions are odd leads to reciprocal interactions between agents which is a key for the fact that the swarms considered in this chapter are non-drifting unless there is an external influence as in the foraging swarm or the swarm tracking cases. Swarms with non-reciprocal interactions may drift even if there are no external influences in the system and express more complex behavior. Analyzing the dynamics of such systems for the agent dynamics in (3.1) within the potential functions framework in this chapter is outside the scope of this book. However, we will consider some non-reciprocal interactions for swarm models with discrete time state dynamics in the following parts of this book.

3.6.3 For Further Reading

This chapter is based on the results in [92–94]. Artificial potential functions have been extensively used for robot navigation and control - see for example [138, 208] for the pioneering works on the topic. One of the related works in specifying inter-individual interactions in a group of robotic agents is the work on *social potential fields* in [201]. Other related recent work on using potential function for multi-agent coordination can be found in [13, 68, 145, 152, 183, 258]. Earlier related work in the literature on mathematical biology on using potential functions to model group dynamics in swarms in nature can be found in [28], where the authors uses a model composed of a constant attraction term and a repulsion term which is inversely proportional to the square of the distance between two individuals and in [254] the authors study the effect of a family of potential functions on swarm cohesion. These works (i.e., those in [28, 254]) as well as the work in [105] discuss also the attraction and repulsion forces between individuals which lead to the swarming behavior in nature.

The related works in [212, 237, 258] employ the agent model in (3.1) to study multi-agent coordinations. The work in [184] also analyze distributed agreement (consensus) problems using the agent model in (3.1). They also use a consensus protocol which is very similar to the attraction potential used in this chapter. Distributed agreement (which will be treated in a later chapter in this book) is in a sense aggregation to a single point. The difference there is that the inter-agent interactions are local and possibly time-dependent. Work on swarms with general topology can be found in [147]. Swarms with non-reciprocal interactions are considered in [37].

4

Swarms of Double Integrator Agents

In this chapter we consider a double-integrator model of an agent. As in the last chapter, we characterize swarm cohesiveness as a stability property and use a Lyapunov approach to develop conditions under which local agent actions lead to cohesive foraging. The conditions here allow for the presence of "noise" characterized by uncertainty on sensing other agent's position and velocity, and in sensing nutrients (a resource profile) that each agent is foraging for. The results quantify claims in biology that social foraging is in a certain sense superior to individual foraging when noise is present, and provide clear connections between local agent-agent interactions and emergent group behavior. Moreover, the simulations at the end of the chapter show that very complicated but orderly group behaviors, reminiscent of those seen in biology, emerge in the presence of noise.

4.1 Double Integrator Model

We consider swarms composed of an interconnection of N agents, each with point mass dynamics given by

$$\begin{aligned} \dot{x}_i &= v_i \\ \dot{v}_i &= \frac{1}{M_i} u_i \end{aligned} \tag{4.1}$$

where $x_i \in \mathbb{R}^n$ is the position, $v_i \in \mathbb{R}^n$ is the velocity, M_i is the mass, and $u_i \in \mathbb{R}^n$ is the (force) control input for the i^{th} agent. The above equations imply that $u_i = M_i \ddot{x}_i$ (force is mass times acceleration). Integrating acceleration once we get velocity, twice and we get position; hence, the term "double integrator model." It is assumed that all agents know their own dynamics. For some organisms like bacteria that move in highly viscous environments it can be assumed that $M_i = 0$, and if a velocity damping term is used in u_i for this, we obtain the model studied in Chapter 3. In this chapter, we do not assume $M_i = 0$. Moreover, we will assume that each agent can sense information about the position and velocity of other agents, but possibly with some errors (what we will call "noise"), something not considered in Chapter 3.

Agent to agent interactions considered here are, like in Chapter 3, of the "attract-repel" type where each agent seeks to be in a position that is "comfortable" relative to its neighbors (and here all other agents are its neighbors). There are many ways to define attraction and repulsion, each of which can be represented by characteristics of how we define u_i for each agent. Attraction here will be represented by a term in u_i like

$$-k_p^i\,(x_i - x_j)$$

where $k_p^i > 0$ is a scalar that represents the strength of attraction. Then, if the agents are far apart, there is a large attraction between them, and if they are close, there is a small attraction. For repulsion, we use a term in u_i of the form

$$k_r^i \exp\left(\frac{-\frac{1}{2}\|x_i - x_j\|^2}{{r_s^i}^2}\right)(x_i - x_j) \tag{4.2}$$

where $k_r^i > 0$ is the magnitude of the repulsion, and $r_s^i > 0$ quantifies the region size around the agent from which it will repel its neighbors. When $\|x_i - x_j\|$ is big relative to r_s^i the whole term approaches zero so there is no repulsion. Note that these correspond to the attraction/repulsion function in (3.7) in Chapter 3 with the corresponding potential function in (3.6). In particular, this function satisfies the linear attraction and bounded repulsion property.

Here, similar to Chapter 3, we will model the agent's environment as a resource profile $\sigma(x)$, where $x \in \mathbb{R}^n$. We will assume that $\sigma(x)$ is continuous with finite slope at all points. Agents move in the direction of the negative gradient of $\sigma(x)$ (i.e., in the direction of $-\nabla\sigma(x) = -\frac{\partial\sigma}{\partial x}$) in order to move away from "bad" areas and into "good" areas of the environment (e.g., to avoid noxious substances and find nutrients). That is, they will use a term in their u_i that holds the negative gradient of $\sigma(x)$. Clearly, there are many possible shapes for $\sigma(x)$, including ones with many peaks and valleys like the ones we studied in Chapter 3. Here, we only consider the "plane" profile since it is simple yet representative. In this case, we have $\sigma(x) = \sigma_p(x)$ where

$$\sigma_p(x) = a_\sigma^\top x + b_\sigma$$

where $a_\sigma \in \mathbb{R}^n$ and b_σ is a scalar. Here, $\nabla\sigma_p(x) = a_\sigma$. Below, we will assume that each agent can sense the gradient, but only with some errors, which we will refer to as "noise." We will consider the case where the agents seek to follow different plane profiles. When we want to refer to agent i as following a possibly different profile than agent j, $j \neq i$, we will use σ_p^i, a_σ^i, and b_σ^i for the plane profiles.

4.2 Stability Analysis of Swarm Cohesion Properties

Cohesion and swarm dynamics will be quantified and analyzed using stability analysis of the swarm error dynamics that we derive next.

4.2.1 Controls and Error Dynamics

Given N agents, let

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

be the centroid of the swarm and

$$\bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_i$$

be the average velocity of the group of agents. The objective of each agent is to move so as to end up at $\bar{x}$ and $\bar{v}$. The problem is that since all the agents are moving at the same time, $\bar{x}$ and $\bar{v}$ are time-varying; hence, in order to study the stability of swarm cohesion we study the dynamics of an error system with

$$e_p^i = x_i - \bar{x}$$

and

$$e_v^i = v_i - \bar{v}$$

Alternative choices for the error system could be made. For instance,

$$\tilde{e}_p^i = \sum_{j=1}^{N}(x_i - x_j)$$

could be used. This corresponds to computing the errors to every other agent and then trying to get all those errors to go to zero. Note, however, that

$$\tilde{e}_p^i = N\left(x_i - \frac{1}{N}\sum_{j=1}^{N} x_j\right) = N(x_i - \bar{x}) = Ne_p^i$$

So this error definition is a scaled version of what we use.

The error dynamics are given by

$$\begin{aligned} \dot{e}_p^i &= e_v^i \\ \dot{e}_v^i &= \frac{1}{M_i}u_i - \frac{1}{N}\sum_{j=1}^{N}\frac{1}{M_j}u_j \end{aligned} \tag{4.3}$$

Some animal's senses and sensory processing may naturally provide $\bar{x}$ and $\bar{v}$, but not all the individual positions of the other agents that it could then use to compute $\bar{x}$ and $\bar{v}$. We could assume that each agent knows its own position and velocity and can sense $\bar{x}$ and $\bar{v}$. Here, however, we only assume that each agent i can sense directly the differences $x_i - \bar{x}$ and $v_i - \bar{v}$, and possibly with some noise. In particular, let $d_p^i \in \mathbb{R}^n$ and $d_v^i \in \mathbb{R}^n$ be these sensing errors for agent i, respectively. We assume that $d_p^i(t)$ and $d_v^i(t)$ are any trajectories that are sufficiently smooth and fixed a priori for all the time (but below we will study the stability for the case when the d_p^i and

d_v^i trajectories can be any of a certain class). We will refer to these terms somewhat colloquially as “noise” but clearly our framework is entirely deterministic. Thus, each agent actually senses

$$\begin{aligned}\hat{e}_p^i &= e_p^i - d_p^i \\ \hat{e}_v^i &= e_v^i - d_v^i\end{aligned}$$

and below we will also assume that it can sense its own velocity.

It is important to highlight our motivation for studying the addition of noise. On the one hand, it adds another element of realism to how a sensor for e_p^i and e_v^i might operate and the results below will help quantify the effects of the noise on cohesion. We also, however, view our approach as progress in the direction of not requiring that each agent can sense the variables of all the other agents or even the accurate values of $\bar{x}$ and $\bar{v}$. How? Here, as will be seen below, we will assume that our noise is bounded so that each agent actually only needs to be able to sense the swarm center and average velocity with a low degree of accuracy. In specific types of agents, this actually may *not* require that each agent senses position and velocity of every other agent (e.g., if for some organism it can compute an approximate center of gravity of the positions of all other agents via some type of averaging).

The i^{th} agent will also try to follow the plane nutrient profile σ_p^i defined earlier. We assume that it senses the gradient of σ_p^i, but with some sufficiently smooth error $d_f^i(t)$ that is fixed a priori for all the time (as with d_p^i and d_v^i we will allow below d_f^i to be any in a certain class of trajectories) so each agent actually senses

$$\nabla\sigma_p^i(x_i) - d_f^i$$

This can be viewed as either sensing error or “noise” (variations, ripples) on the resource profile.

Suppose the general form of the control input for each agent is

$$\begin{aligned}u_i = &- M_i k_p^i \hat{e}_p^i - M_i k_v^i \hat{e}_v^i - M_i k v_i + \\ &M_i k_r^i \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{{r_s^i}^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right) - M_i k_f^i\left(a_\sigma^i - d_f^i\right)\end{aligned} \tag{4.4}$$

The term M_i is used in each term so it will cancel with the $1/M_i$ term in Equation (4.3). We think of the scalars $k_p^i > 0$ and $k_v^i > 0$ as the “attraction gains” which indicate how aggressive each agent is in aggregating. The gain $k_r^i > 0$ is a “repulsion gain” which sets how much that agent wants to be away from others and r_s^i represents its repulsion range. Also note that in the repulsion term

$$\hat{e}_p^i - \hat{e}_p^j = \left((x_i - \bar{x}) - d_p^i\right) - \left((x_j - \bar{x}) - d_p^j\right) = (x_i - x_j) - \left(d_p^i - d_p^j\right)$$

Obviously, if $d_p^i = 0$ for all i, there is no sensing error on repulsion, and a repulsion term of the form explained in Equation (4.2) is obtained. The gain $k > 0$ works as a "velocity damping gain;" note that we use the same such gain for all agents. The last term in Equation (4.4) indicates that each agent wants to move along the negative gradient of the i^{th} resource profile with the gain $k_f^i > 0$ proportional to that agent's desire to follow its profile.

The sensing errors create the possibility that agents will try to move away from each other when they may not really need to, and they may move toward each other when they should not. Similarly, the attraction gains k_p^i and k_v^i dictate how the attraction forces operate, but the presence of the noise results in additive noise terms to u_i that are multiplied by k_p^i and k_v^i. Hence, raising the attraction gains also has a negative influence on u_i in that it results in more noise entering the control and hence possibly poor aggregating decisions by individuals (e.g., if $\|x_i - x_j\|$ is small but d_p^i is relatively large, noise will set the control value). Clearly, this complicates the situation for the whole swarm to achieve cohesiveness.

To study stability properties, we will substitute the above choice for u_i into the error dynamics in Equation (4.3). First, consider the $\dot{v}_i$ term of $\dot{e}_v^i = \dot{v}_i - \dot{\bar{v}}$ and note that

$$\begin{aligned}\dot{v}_i = \frac{1}{M_i}u_i &= -k_p^i e_p^i + k_p^i d_p^i - k_v^i e_v^i + k_v^i d_v^i - k v_i \\ &+ k_r^i \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{{r_s^i}^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right) - k_f^i\left(a_\sigma^i - d_f^i\right)\end{aligned} \tag{4.5}$$

Next, substituting u_i into $\dot{\bar{v}}$ and we have

$$\begin{aligned}\dot{\bar{v}} = &-\frac{1}{N}\sum_{j=1}^{N} k_p^j e_p^j + \frac{1}{N}\sum_{j=1}^{N} k_p^j d_p^j - \frac{1}{N}\sum_{j=1}^{N} k_v^j e_v^j + \frac{1}{N}\sum_{j=1}^{N} k_v^j d_v^j - \frac{1}{N}\sum_{j=1}^{N} k v_j + \\ &\frac{1}{N}\sum_{l=1}^{N} k_r^l \sum_{j=1, j\neq l}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^l - \hat{e}_p^j\right\|^2}{{r_s^l}^2}\right)\left(\hat{e}_p^l - \hat{e}_p^j\right) - \\ &\frac{1}{N}\sum_{j=1}^{N} k_f^j\left(a_\sigma^j - d_f^j\right)\end{aligned} \tag{4.6}$$

Define $\bar{k}_p = \frac{1}{N}\sum_{j=1}^{N} k_p^j$ and $\Delta k_p^j = k_p^j - \bar{k}_p$. Since

$$\sum_{j=1}^{N} e_p^j = \sum_{j=1}^{N} (x_j - \bar{x}) = N\bar{x} - \sum_{j=1}^{N} \bar{x} = 0$$

we have

$$\sum_{j=1}^{N} k_p^j e_p^j = \sum_{j=1}^{N} \Delta k_p^j e_p^j + \sum_{j=1}^{N} \bar{k}_p e_p^j = \sum_{j=1}^{N} \Delta k_p^j e_p^j$$

Similarly, define $\bar{k}_v$ and Δk_v^j, so we have $\sum_{j=1}^{N} k_v^j e_v^j = \sum_{j=1}^{N} \Delta k_v^j e_v^j$. So Equation (4.6) becomes

$$\dot{\bar{v}} = -\frac{1}{N}\sum_{j=1}^{N}\Delta k_p^j e_p^j + \frac{1}{N}\sum_{j=1}^{N} k_p^j d_p^j - \frac{1}{N}\sum_{j=1}^{N}\Delta k_v^j e_v^j + \frac{1}{N}\sum_{j=1}^{N} k_v^j d_v^j - \frac{1}{N}\sum_{j=1}^{N} k v_j + \\ \frac{1}{N}\sum_{l=1}^{N} k_r^l \sum_{j=1, j\neq l}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^l - \hat{e}_p^j\right\|^2}{{r_s^l}^2}\right)\left(\hat{e}_p^l - \hat{e}_p^j\right) - \\ \frac{1}{N}\sum_{j=1}^{N} k_f^j\left(a_\sigma^j - d_f^j\right) \tag{4.7}$$

Define $E^i = [{e_p^i}^\top, {e_v^i}^\top]^\top$ and $E = [{E^1}^\top, {E^2}^\top, \ldots, {E^N}^\top]^\top$. Since

$$k v_i - \frac{1}{N}\sum_{j=1}^{N} k v_j = k v_i - k\bar{v} = k e_v^i$$

from Equations (4.5) and (4.7) we have

$$\dot{e}_v^i = \dot{v}_i - \dot{\bar{v}} = -k_p^i e_p^i - \left(k_v^i + k\right) e_v^i + g^i + \phi(E) + \delta^i(E) \tag{4.8}$$

where

$$g^i = k_p^i d_p^i + k_v^i d_v^i + k_f^i d_f^i - k_f^i a_\sigma^i \tag{4.9}$$

$$\phi(E) = \frac{1}{N}\sum_{j=1}^{N}\Delta k_p^j e_p^j + \frac{1}{N}\sum_{j=1}^{N}\Delta k_v^j e_v^j - \frac{1}{N}\sum_{j=1}^{N} k_p^j d_p^j - \frac{1}{N}\sum_{j=1}^{N} k_v^j d_v^j + \\ \frac{1}{N}\sum_{j=1}^{N} k_f^j\left(a_\sigma^j - d_f^j\right) \tag{4.10}$$

$$\delta^i(E) = k_r^i \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{{r_s^i}^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right) - \\ \frac{1}{N}\sum_{l=1}^{N} k_r^l \sum_{j=1, j\neq l}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^l - \hat{e}_p^j\right\|^2}{{r_s^l}^2}\right)\left(\hat{e}_p^l - \hat{e}_p^j\right) \tag{4.11}$$

which is a nonlinear non-autonomous system. If I is an $n \times n$ identity matrix, the error dynamics of the i^{th} agent may be written as

$$\dot{E}^i = \overbrace{\begin{bmatrix} 0 & I \\ -k_p^i I & -\left(k_v^i + k\right) I \end{bmatrix}}^{A_i} E^i + \overbrace{\begin{bmatrix} 0 \\ I \end{bmatrix}}^{B}\left(g^i + \phi(E) + \delta^i(E)\right) \tag{4.12}$$

Note that any matrix

$$\begin{bmatrix} 0 & I \\ -k_1 I & -k_2 I \end{bmatrix}$$

with $k_1 > 0$ and $k_2 > 0$ has eigenvalues given by the roots of $(s^2 + k_2 s + k_1)^n$, which are in the strict left half plane. Thus, the matrix A_i above is Hurwitz if $k_p^i > 0$, $k_v^i > 0$ and $k > 0$.

4.2.2 Cohesive Social Foraging with Noise

Our analysis methodology involves viewing the error system in Equation (4.12) as generating $E^i(t)$ trajectories for a given $E^i(0)$ and the fixed sensing error trajectories $d_p^i(t)$, $d_v^i(t)$, an $d_f^i(t)$, $t \geq 0$. We do not consider, however, all possible sensing error trajectories. We only consider a class of ones that satisfy for all $t \geq 0$

$$\begin{aligned} \|d_f^i(t)\| &\leq D_f^i \\ \|d_p^i(t)\| &\leq D_{p_1}^i \left\|E^i(t)\right\| + D_{p_2}^i \\ \|d_v^i(t)\| &\leq D_{v_1}^i \left\|E^i(t)\right\| + D_{v_2}^i \end{aligned} \tag{4.13}$$

where $D_{p_1}^i$, $D_{p_2}^i$, $D_{v_1}^i$, $D_{v_2}^i$ and D_f^i are known non-negative constants for $i = 1,\ldots,N$. Hence, we assume for position and velocity the sensing errors have linear relationship with the magnitude of the state of the error system. Basically, the assumption means that when two agents are far away from each other, the sensing errors can increase. The noise d_f^i on the nutrient profile is unaffected by the position of an agent. By considering only this class of fixed sensing error trajectories we prune the set of possibilities for E^i trajectories and it is only that pruned set that our analysis holds for.

Uniform Ultimate Boundedness of Inter-agent Trajectories

We first establish that the inter-agent trajectories are uniformly ultimately bounded. Following this, we derive the ultimate bound on the inter-agent trajectories.

Theorem 11. *Consider the swarm described by the model in Equation (4.3) with control input u_i given in Equation (4.4). Assume that the nutrient profile for each agent is a plane defined by $\nabla\sigma_p^i(x) = a_\sigma^i$ and the noise satisfies Equation (4.13). Let*

$$\beta_1^i = \frac{\left(k_p^i + 1\right)^2 + \left(k_v^i + k\right)^2}{2k_p^i\left(k_v^i + k\right)} + \sqrt{\left(\frac{{k_p^i}^2 + (k_v^i + k)^2 - 1}{2k_p^i\left(k_v^i + k\right)}\right)^2 + \frac{1}{{k_p^i}^2}} \tag{4.14}$$

for $i = 1,\ldots,N$. If for all i we have

$$k_p^i D_{p_1}^i + k_v^i D_{v_1}^i < \frac{1}{\beta_1^i} \tag{4.15}$$

and the parameters are such that

$$\sum_{i=1}^{N} \frac{\beta_1^{i*}\left(k_p^i D_{p_1}^i + k_v^i D_{v_1}^i + \sqrt{\Delta {k_p^i}^2 + \Delta {k_v^i}^2}\right)}{N\left(1-\theta^i\right)\left(1 - \beta_1^i\left(k_p^i D_{p_1}^i + k_v^i D_{v_1}^i\right)\right)} < 1 \tag{4.16}$$

for some constants $0 < \theta^i < 1$*, where*

$$i^* = \arg\max_i \beta_1^i,\ i = 1, \ldots, N$$

then the trajectories of Equation (4.12) are uniformly ultimately bounded.

Proof: To study the stability of the error dynamics, it is convenient to choose Lyapunov functions for each agent as

$$V_i\left(E^i\right) = {E^i}^\top P_i E^i \tag{4.17}$$

with $P_i = P_i^\top$ a $2n \times 2n$ matrix and $P_i > 0$ (a positive definite matrix). Then we have

$$\dot{V}_i = {E^i}^\top P_i \dot{E}^i + {\dot{E}^i}{}^\top P_i E^i = {E^i}^\top \underbrace{\left(P_i A_i + A_i^\top P_i\right)}_{-Q_i} E^i + 2{E^i}^\top P_i B \left(g^i + \phi^i(E) + \delta^i(E)\right) \tag{4.18}$$

Note that when $Q_i = Q_i^\top$ and $Q_i > 0$, the unique solution P_i of $P_i A_i + A_i^\top P_i = -Q_i$ has $P_i = P_i^\top$ and $P_i > 0$ as needed.

Choose for the composite system

$$V(E) = \sum_{i=1}^{N} V_i(E^i)$$

where $V_i\left(E^i\right)$ is given in Equation (4.17). Since for any matrix $M > 0$ and vector X

$$\lambda_{min}(M) X^\top X \leq X^\top M X \leq \lambda_{max}(M) X^\top X$$

where $\lambda_{min}(M)$ and $\lambda_{max}(M)$ denote the minimum and maximum eigenvalue of M, respectively, from Equation (4.17) we have

$$\sum_{i=1}^{N} \left(\lambda_{min}(P_i) \left\|E^i\right\|^2\right) \leq V(E) \leq \sum_{i=1}^{N} \left(\lambda_{max}(P_i) \left\|E^i\right\|^2\right)$$

It can be shown that the function $F(\psi) = \exp\left(\frac{-\frac{1}{2}\|\psi\|^2}{{r_s^i}^2}\right)\|\psi\|$, with ψ any real vector, has a unique maximum value of $\exp(-\frac{1}{2}) r_s^i$ which is achieved when $\|\psi\| = r_s^i$ (a property used in Chapter 3). Defining

$$\Delta^i = k_r^i \exp\left(-\frac{1}{2}\right) \sum_{j=1, j\neq i}^{N} r_s^j + \frac{1}{N} \exp\left(-\frac{1}{2}\right) \sum_{l=1}^{N} k_r^l \sum_{j=1, j\neq l}^{N} r_s^j \tag{4.19}$$

we have $\left\|\delta^i(E)\right\| \leq \Delta^i$ for $i = 1, \ldots, N$. Define

$$\tilde{a}_\sigma = \frac{1}{N} \sum_{i=1}^{N} k_f^i a_\sigma^i$$

Then substituting Equation (4.12) into (4.18), and using Equation (4.19) and the fact that $\|B\| = 1$ we have

$$\begin{aligned}
\dot{V}(E) &= \sum_{i=1}^{N} \dot{V}_i(E^i) = \sum_{i=1}^{N} \left[-E^{i\top} Q_i E^i + 2E^{i\top} P_i B \left(g^i + \phi^i(E) + \delta^i(E) \right) \right] \\
&\leq \sum_{i=1}^{N} \Bigg[-\lambda_{min}(Q_i) \left\| E^i \right\|^2 + 2 \left\| E^i \right\| \lambda_{max}(P_i) \Bigg(k_p^i \left\| d_p^i \right\| + k_v^i \left\| d_v^i \right\| + k_f^i \left\| d_f^i \right\| + \\
&\qquad \left\| k_f^i a_\sigma^i - \tilde{a}_\sigma \right\| + \frac{1}{N} \sum_{j=1}^{N} \left\| \begin{bmatrix} \Delta k_p^j I & \Delta k_v^j I \end{bmatrix} \begin{bmatrix} e_p^j \\ e_v^j \end{bmatrix} \right\| + \\
&\qquad \frac{1}{N} \sum_{j=1}^{N} k_p^j \left\| d_p^j \right\| + \frac{1}{N} \sum_{j=1}^{N} k_v^j \left\| d_v^j \right\| + \frac{1}{N} \sum_{j=1}^{N} k_f^j \left\| d_f^j \right\| + \Delta^i \Bigg) \Bigg] \\
&\leq \sum_{i=1}^{N} \Bigg[-\lambda_{min}(Q_i) \left\| E^i \right\|^2 + 2 \left\| E^i \right\| \lambda_{max}(P_i) \Bigg(k_p^i D_{p_1}^i \left\| E^i \right\| + k_p^i D_{p_2}^i + \\
&\qquad k_v^i D_{v_1}^i \left\| E^i \right\| + k_v^i D_{v_2}^i + k_f^i D_f^i + \left\| k_f^i a_\sigma^i - \tilde{a}_\sigma \right\| + \\
&\qquad \frac{1}{N} \sum_{j=1}^{N} \sqrt{\Delta {k_p^j}^2 + \Delta {k_v^j}^2} \left\| E^j \right\| + \frac{1}{N} \sum_{j=1}^{N} k_p^j D_{p_1}^j \left\| E^j \right\| + \frac{1}{N} \sum_{j=1}^{N} k_p^j D_{p_2}^j + \\
&\qquad \frac{1}{N} \sum_{j=1}^{N} k_v^j D_{v_1}^j \left\| E^j \right\| + \frac{1}{N} \sum_{j=1}^{N} k_v^j D_{v_2}^j + \frac{1}{N} \sum_{j=1}^{N} k_f^j D_f^j + \Delta^i \Bigg) \Bigg] \\
&= \sum_{i=1}^{N} \left[-c_1^i \left\| E^i \right\|^2 + c_2^i \left\| E^i \right\| + \left\| E^i \right\| \sum_{j=1}^{N} \left(a^{ij} \left\| E^j \right\| \right) \right] \qquad (4.20)
\end{aligned}$$

with c_1^i, c_2^i and a^{ij} constants and

$$\begin{aligned}
c_1^i &= \lambda_{min}(Q_i) \left(1 - \frac{2\lambda_{max}(P_i)}{\lambda_{min}(Q_i)} \left(k_p^i D_{p_1}^i + k_v^i D_{v_1}^i \right) \right) \\
c_2^i &= 2\lambda_{max}(P_i) \Bigg(k_p^i D_{p_2}^i + k_v^i D_{v_2}^i + k_f^i D_f^i + \left\| k_f^i a_\sigma^i - \tilde{a}_\sigma \right\| + \frac{1}{N} \sum_{j=1}^{N} k_p^j D_{p_2}^j + \\
&\qquad \frac{1}{N} \sum_{j=1}^{N} k_v^j D_{v_2}^j + \frac{1}{N} \sum_{j=1}^{N} k_f^j D_f^j + \Delta^i \Bigg) \\
a^{ij} &= \frac{2}{N} \lambda_{max}(P_i) \left(k_p^j D_{p_1}^j + k_v^j D_{v_1}^j + \sqrt{\Delta {k_p^j}^2 + \Delta {k_v^j}^2} \right)
\end{aligned}$$

Obviously $c_2^i > 0$, $a^{ij} > 0$, and if we have

$$k_p^i D_{p_1}^i + k_v^i D_{v_1}^i < \frac{1}{\beta_0^i} \qquad (4.21)$$

where

$$\beta_0^i = \frac{2\lambda_{max}(P_i)}{\lambda_{min}(Q_i)}$$

then $c_1^i > 0$.

Before we proceed, note that in Equation (4.21), we want β_0^i to be as small as possible so that the system may tolerate noise with the largest possible bounds ($D_{p_1}^i$ and $D_{v_1}^i$) while keeping stability. Notice that we can influence the size of the β_0^i by the choice of $Q_i > 0$. Next, we explain why letting $\beta_0^i = \beta_1^i$ (defined in Equation (4.14)) renders Equation (4.21) less restrictive in terms of allowing as high of magnitude of noise as possible. First, β_0^i is minimized by letting $Q_i = k_q^i I$ with $k_q^i > 0$ a free parameter. Next, we find

$$\beta_1^i = \beta_0^i\big|_{Q_i = k_q^i I} = \min \beta_0^i$$

Specifically, it can be proven that the $2n \times 2n$ matrix P_i has n repeated values of the eigenvalues of $\tilde{P}_i$, where $\tilde{P}_i\tilde{A}_i + \tilde{A}_i^\top \tilde{P}_i = -\tilde{Q}_i$ with

$$\tilde{A}_i = \begin{bmatrix} 0 & 1 \\ -k_p^i & -\left(k_v^i + k\right) \end{bmatrix}, \; \tilde{Q}_i = \begin{bmatrix} k_q^i & 0 \\ 0 & k_q^i \end{bmatrix}$$

Then it can be shown that $\lambda_{1,2}(\tilde{P}_i)$, the two eigenvalues of $\tilde{P}_i$, are given by

$$\frac{1}{2}\left[\frac{\left(k_p^i + 1\right)^2 k_q^i + \left(k_v^i + k\right)^2 k_q^i}{2k_p^i\left(k_v^i + k\right)} \pm \sqrt{\left(\frac{{k_p^i}^2 k_q^i + \left(k_v^i + k\right)^2 k_q^i - k_q^i}{2k_p^i\left(k_v^i + k\right)}\right)^2 + \frac{{k_q^i}^2}{{k_p^i}^2}}\;\right]$$

where $\lambda_1(\tilde{P}_i)$ is defined by the "+" and $\lambda_2(\tilde{P}_i)$ by the "−". Here $\lambda_{max}(\tilde{P}_i) = \lambda_1(\tilde{P}_i)$ and $\lambda_{min}(\tilde{P}_i) = \lambda_2(\tilde{P}_i)$. Comparing the above equation with Equation (4.14), we have $\lambda_1(\tilde{P}_i) = \frac{1}{2}k_q^i\beta_1^i$. Since $\lambda_{max}(P_i) = \lambda_{max}(\tilde{P}_i)$ and $\lambda_{min}(\tilde{Q}_i) = k_q^i$,

$$\min \beta_0^i = \frac{2\lambda_{max}(P_i)}{\lambda_{min}(Q_i)}\bigg|_{Q_i = k_q^i I} = \frac{2\lambda_{max}(\tilde{P}_i)}{\lambda_{min}(\tilde{Q}_i)} = \frac{k_q^i\beta_1^i}{k_q^i} = \beta_1^i \tag{4.22}$$

Thus, from (4.21) and (4.22), if Equation (4.15) holds, then by choosing $Q_i = k_q^i I$, the corresponding constant c_1^i for (4.20) is positive.

Now for simplicity, we choose $Q_i = I$ for all i (so $k_q^i = 1$) and use Equation (4.22), so c_1^i, c_2^i and a^{ij} of (4.20) are simplified to become

$$c_1^i = 1 - \beta_1^i\left(k_p^i D_{p_1}^i + k_v^i D_{v_1}^i\right) \tag{4.23}$$

$$\begin{aligned} c_2^i = \beta_1^i \Bigg(& k_p^i D_{p_2}^i + k_v^i D_{v_2}^i + k_f^i D_f^i + \left\| k_f^i a_\sigma^i - \tilde{a}_\sigma \right\| + \frac{1}{N}\sum_{j=1}^N k_p^j D_{p_2}^j + \\ & \frac{1}{N}\sum_{j=1}^N k_v^j D_{v_2}^j + \frac{1}{N}\sum_{j=1}^N k_f^j D_f^j + \Delta^i \Bigg) \end{aligned} \tag{4.24}$$

$$a^{ij} = \frac{\beta_1^i}{N}\left(k_p^j D_{p_1}^j + k_v^j D_{v_1}^j + \sqrt{\Delta {k_p^j}^2 + \Delta {k_v^j}^2}\right) \tag{4.25}$$

Now, return to (4.20) and note that for any θ^i, $0 < \theta^i < 1$,

$$\begin{aligned} -c_1^i \left\|E^i\right\|^2 + c_2^i \left\|E^i\right\| &= -(1-\theta^i)c_1^i \left\|E^i\right\|^2 - \theta^i c_1^i \left\|E^i\right\|^2 + c_2^i \left\|E^i\right\| \\ &\leq -(1-\theta^i)c_1^i \left\|E^i\right\|^2, \ \forall \ \left\|E^i\right\| \geq r^i \\ &= \sigma^i \left\|E^i\right\|^2 \end{aligned} \tag{4.26}$$

where $r^i = \frac{c_2^i}{\theta^i c_1^i}$ and $\sigma^i = -(1-\theta^i)c_1^i < 0$. This implies that as long as $\|E^i\| \geq r^i$, the first two terms in (4.20) combined will give a negative contribution to $\dot{V}(E)$.

Next, we seek conditions under which $\dot{V}(E) < 0$. To do this, we consider the third term in (4.20) and combine it with the above results. First, note that the third term in (4.20) can be over-bounded by replacing a^{ij} by a^{i^*j} where

$$a^{i^*j} = \max_{1 \leq i \leq N} a^{ij} \tag{4.27}$$

which were defined in the statement of the theorem via Equation (4.25). Next, we consider the general situation where some of the E^i are such that $\|E^i\| < r^i$ and others are not. Accordingly, define sets

$$\Pi_O = \left\{ i : \left\|E^i\right\| \geq r^i, \ i \in 1,\ldots,N \right\} = \left\{ i_O^1, \ i_O^2, \ldots, \ i_O^{N_O} \right\}$$

and

$$\Pi_I = \left\{ i : \left\|E^i\right\| < r^i, \ i \in 1,\ldots,N \right\} = \left\{ i_I^1, \ i_I^2, \ldots, \ i_I^{N_I} \right\}$$

where N_O and N_I are the size of Π_O and Π_I, respectively, and $N_O + N_I = N$. Also, $\Pi_O \bigcup \Pi_I = \{1,\ldots,N\}$ and $\Pi_O \bigcap \Pi_I = \phi$. Of course, we do not know the explicit sets Π_O and Π_I; all we know is that they exist. The explicit values in the sets clearly depend on time, but we will allow that time to be arbitrary so the analysis below will be for all t. For now, we assume $N_O > 0$, that is, the set Π_O is non-empty. We will later discuss the $N_O = 0$ case. Then using analysis ideas from the theory of stability of interconnected systems, and Equations (4.20), (4.26) and (4.27), we have

$$\begin{aligned} \dot{V}(E) \leq & \sum_{i \in \Pi_O} \sigma^i \left\|E^i\right\|^2 + \sum_{i \in \Pi_O} \left(\left\|E^i\right\| \sum_{j \in \Pi_O} a^{i^*j} \left\|E^j\right\| \right) + \sum_{i \in \Pi_O} \left(\left\|E^i\right\| \sum_{j \in \Pi_I} a^{i^*j} \left\|E^j\right\| \right) \\ & + \sum_{i \in \Pi_I} \left(-c_1^i \left\|E^i\right\|^2 + c_2^i \left\|E^i\right\| \right) + \sum_{i \in \Pi_I} \left(\left\|E^i\right\| \sum_{j \in \Pi_O} a^{i^*j} \left\|E^j\right\| \right) \\ & + \sum_{i \in \Pi_I} \left(\left\|E^i\right\| \sum_{j \in \Pi_I} a^{i^*j} \left\|E^j\right\| \right) \end{aligned}$$

Note for each fixed N_O, there exist positive constants $K_1(N_O)$, $K_2(N_O)$, $K_3(N_O)$ and $K_4(N_O)$ such that,

$$K_1(N_O) \geq \sum_{j \in \Pi_I} a^{i^*j} \left\|E^j\right\|$$

$$K_2(N_O) \geq \sum_{i \in \Pi_I} \left(-c_1^i \left\|E^i\right\|^2 + c_2^i \left\|E^i\right\| \right) \tag{4.28}$$

$$K_3(N_O) \geq \sum_{i \in \Pi_I} \left\|E^i\right\|$$

$$K_4(N_O) \geq \sum_{i \in \Pi_I} \left(\left\|E^i\right\| \sum_{j \in \Pi_I} a^{i^*j} \left\|E^j\right\| \right)$$

Then, we have

$$\begin{aligned}
\dot{V}(E) \leq & \sum_{i \in \Pi_O} \sigma^i \left\|E^i\right\|^2 + \sum_{i \in \Pi_O} \left(\left\|E^i\right\| \sum_{j \in \Pi_O} a^{i^*j} \left\|E^j\right\| \right) + K_1 \sum_{i \in \Pi_O} \left\|E^i\right\| + K_2 + \\
& K_3 \sum_{j \in \Pi_O} a^{i^*j} \left\|E^j\right\| + K_4 \\
= & \sum_{i \in \Pi_O} \sigma^i \left\|E^i\right\|^2 + \sum_{i \in \Pi_O} \left(\left\|E^i\right\| \sum_{j \in \Pi_O} a^{i^*j} \left\|E^j\right\| \right) + \sum_{i \in \Pi_O} \left(K_1 + K_3 a^{i^*i} \right) \left\|E^i\right\| + \\
& K_2 + K_4
\end{aligned}$$

Let $w^\top = \left[\|E^{i_O^1}\|, \|E^{i_O^2}\|, \ldots, \|E^{i_O^{N_O}}\| \right]$ (the composition of this vector can be different at different times) and the $N_O \times N_O$ matrix $S = [s_{jn}]$ be specified by

$$s_{jn} = \begin{cases} -(\sigma^{i_O^j} + a^{i^*j}), & j = n \\ -a^{i^*j}, & j \neq n \end{cases} \tag{4.29}$$

(the $\sigma^{i_O^j}$ is a constant, not to be confused with the notation we use for the plane profile) so we have

$$\dot{V}(E) \leq -w^\top S w + \sum_{i \in \Pi_O} \left(K_1 + K_3 a^{i^*i} \right) \left\|E^i\right\| + K_2 + K_4$$

For now, assume that $S > 0$ in the above equation and thus, $\lambda_{min}(S) > 0$, then we have

$$\dot{V}(E) \leq -\lambda_{min}(S) \sum_{i \in \Pi_O} \left\|E^i\right\|^2 + \sum_{i \in \Pi_O} \left(K_1 + K_3 a^{i^*i} \right) \left\|E^i\right\| + K_2 + K_4$$

So when the $\left\|E^i\right\|$ for $i \in \Pi_O$ are sufficiently large, the sign of $\dot{V}(E)$ is determined by $-\lambda_{min}(S) \sum_{i \in \Pi_O} \left\|E^i\right\|^2$ and $\dot{V}(E) < 0$. This analysis is valid for any value of N_O, $1 \leq N_O \leq N$; hence for any $N_O \neq 0$ the system is uniformly ultimately bounded if $S > 0$, so we seek to prove that next.

A necessary and sufficient condition for $S > 0$ is that its successive principal minors are all positive. Define $|S_m|$ as the determinants of the principal minors of S, $m = 1, \ldots, N_O$. Then

$$
\begin{aligned}
|S_m| &= \begin{vmatrix} -(\sigma^{i_O^1} + a^{i^*1}) & -a^{i^*1} & \dots & -a^{i^*1} \\ -a^{i^*2} & -(\sigma^{i_O^2} + a^{i^*2}) & \dots & -a^{i^*2} \\ \vdots & \vdots & \ddots & \vdots \\ -a^{i^*m} & -a^{i^*m} & \dots & -(\sigma^{i_O^m} + a^{i^*m}) \end{vmatrix} \\
&= \begin{vmatrix} -\sigma^{i_O^1} & 0 & \dots & \frac{a^{i^*1}}{a^{i^*m}} \sigma^{i_O^m} \\ 0 & -\sigma^{i_O^2} & \dots & \frac{a^{i^*2}}{a^{i^*m}} \sigma^{i_O^m} \\ \vdots & \vdots & \ddots & \vdots \\ -a^{i^*m} & -a^{i^*m} & \dots & -(\sigma^{i_O^m} + a^{i^*m}) \end{vmatrix} \\
&= \begin{vmatrix} -\sigma^{i_O^1} & 0 & \dots & \frac{a^{i^*1}}{a^{i^*m}} \sigma^{i_O^m} \\ 0 & -\sigma^{i_O^2} & \dots & \frac{a^{i^*2}}{a^{i^*m}} \sigma^{i_O^m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -\sigma^{i_O^m} \left(1 + \sum_{j=1}^m \frac{a^{i^*j}}{\sigma^{i_O^j}}\right) \end{vmatrix} \\
&= \left(1 + \sum_{j=1}^m \frac{a^{i^*j}}{\sigma^{i_O^j}}\right) \prod_{k=1}^m \left(-\sigma^{i_O^k}\right)
\end{aligned}
$$

Since $-\sigma^{i_O^k} > 0$ for $k = 1, \dots, m$, to have all the above determinants positive we need

$$
\sum_{j=1}^m \frac{a^{i^*j}}{\sigma^{i_O^j}} > -1
$$

that is

$$
\sum_{j=1}^m \frac{\beta_1^{i^*} \left(k_p^j D_{p_1}^j + k_v^j D_{v_1}^j + \sqrt{\Delta {k_p^j}^2 + \Delta {k_v^j}^2}\right)}{N \left(1 - \theta^{i_O^j}\right) \left(1 - \beta_1^{i_O^j} \left(k_p^{i_O^j} D_{p_1}^{i_O^j} + k_v^{i_O^j} D_{v_1}^{i_O^j}\right)\right)} < 1
$$

for all $m = 1, \dots, N_O$. Since $1 \le m \le N_O \le N$, the equation above is satisfied when Equation (4.16) is satisfied and thus, $S > 0$ for all $N_O \ne 0$. Hence, when $\|E^i\|$ is sufficiently large, $\dot{V}(E) < 0$ and the uniform ultimate boundedness of the trajectories of the error system is achieved.

To complete the proof, we need to consider the case when $N_O = 0$. Note that when $N_O = 0$, $\|E^i\| < r^i$ for all i. If we have $N_O = 0$ persistently, then we could simply take $\max_i r^i$ as the uniform ultimate bound. If otherwise, at certain moment the system changes such that some $\|E^i\| \ge \max_i r^i$, then we have $N_O \ge 1$ immediately, then all the analysis above, which holds for any $1 \le N_O \le N$, applies. Thus, in either case we obtain the uniform ultimate boundedness. This concludes the proof. ∎

Uniform ultimate boundedness is obtained when Equations (4.15) and (4.16) are satisfied. Note that these conditions do not depend on k_r^i and r_s^i; these two parameters can affect the size of the ultimate bound, but it is the attraction gains k_p^i and k_v^i and damping gain k that determine if boundedness can be achieved for given parameters that quantify the size of the noise. The conditions also do not depend on $D^i_{p_2}$ and $D^i_{v_2}$, but these too will affect the size of the ultimate bound. The conditions do not depend on k^i_f, a^i_σ and D^i_f since our error system quantifies swarm cohesiveness, not how well the resource profile is followed.

From Equation (4.14), if both k^i_p and k^i_v are fixed, when k is sufficiently large, increasing k will increase β^i_1, which means $D^i_{p_1}$ and $D^i_{v_1}$ have to be decreased to satisfy Equation (4.15). This means that although we may expect a large k to dampen the error system faster, it could make the system be more vulnerable to noise. Note that when all other parameters are fixed, β^i_1 goes to infinity when k^i_p either goes to infinity or approaches zero. Thus, when k^i_p is the only free parameter, there exists some upper bound for $D^i_{p_1}$ beyond which Equation (4.15) can never hold whatever k^i_p is. This is because when $D^i_{p_1}$ is large enough, k^i_p has to be sufficiently small to decrease the product $k^i_p D^i_{p_1}$ in Equation (4.15), while the β^i_1 corresponding to this sufficiently-small k^i_p will be so large that Equation (4.15) cannot be satisfied. Basically, this means that if $D^i_{p_1}$ is too large and leads to potential instability, it cannot be remedied by merely tuning k^i_p. In comparison, if k^i_v is a free parameter with other parameters fixed and $D^i_{p_1}$ sufficiently small, then for any arbitrarily large $D^i_{v_1}$, we can always find some k^i_v such that (4.15) still holds. This is because k^i_v and k always appear together in β^i_1 and thus, for any large $D^i_{v_1}$, we are free to decrease k^i_v such that the product of $k^i_v D^i_{v_1}$ is small.

Finally, note that the smaller ${\Delta k^j_p}^2$ (${\Delta k^j_v}^2$) is, meaning k^i_p (k^i_v) and k^j_p (k^j_v) are closer to each other for all i and j, the easier it is to meet the condition specified by Equation (4.16). This means that better approximations of the agent parameters may facilitate the boundedness of the error system. In fact when all agents are identical, the sufficient condition (4.16) can be immediately simplified to

$$\frac{\beta_1 (k_p D_{p_1} + k_v D_{v_1})}{(1-\theta)(1-\beta_1 (k_p D_{p_1} + k_v D_{v_1}))} < 1$$

by letting $k^i_p = k_p$, $k^i_v = k_v$, $\beta^i_1 = \beta_1$, $D^i_{p_1} = D_{p_1}$ and $D^i_{v_1} = D_{v_1}$ for all i. Note the term $\sqrt{{\Delta k^i_p}^2 + {\Delta k^i_v}^2}$ is zero now since $k^i_p = \bar{k}_p$ and $k^i_v = \bar{k}_v$ for all i. Furthermore, when $D^i_{p_1} = D^i_{v_1} = 0$ for all i, the conditions (4.15) and (4.16) will always hold. This means when agents are identical and noise is constant, or with constant bound, the trajectories of the error system are always uniformly ultimately bounded. Also note that since the agents are in general *not* identical and have different parameters, the conditions stated by the theorem are quite conservative.

Ultimate Bound on Inter-Agent Trajectories

So far we have shown that the swarm error system is uniformly ultimately bounded when certain conditions are satisfied. We have shown that the bound exists but have not specified it as we now seek to do. If we define the bound as $R_b > 0$, then the set

$$\Omega_c = \left\{E : \left\|E^i\right\| \leq R_b, i = 1, \ldots, N\right\}$$

is attractive and compact. One such bound is given by the Corollary below. Before we state the Corollary, some new notation needs to be introduced. Notice that for each given N_O, $1 \leq N_O \leq N$, the set Π_O can have $N_{N_O} = C_N^{N_O}$ types of compositions, where $C_N^{N_O}$ is the number of combinations of choosing N_O members from a set with N members. (Note that the special case of $N_O = 0$ will be considered separately at the end of the proof for the Corollary.) Let $\Pi_O^{(k)}$ be the set Π_O corresponding to the k^{th} composition, $k = 1, \ldots, N_{N_O}$. Let the $N_O \times N_O$ matrix $S^{(k)}$ be specified in the same way as S, defined in Equation (4.29), but corresponding to the k^{th} composition, $k = 1, \ldots, N_{N_O}$. Define for $k = 1, \ldots, N_{N_O}$

$$\begin{aligned} a_d^k(N_O) &= \lambda_{min}(S^{(k)}_{N_O \times N_O}) \\ b_d(N_O) &= K_1(N_O) + K_3(N_O)\hat{a} \\ c_d(N_O) &= K_2(N_O) + K_4(N_O) \end{aligned}$$

where K_1, K_2, K_3 and K_4 are defined in (4.28), and

$$\hat{a} = \max_{1 \leq i \leq N,\, 1 \leq j \leq N} a^{ij} = \max_{1 \leq j \leq N} a^{i^* j}$$

with $a^{i^* j}$ defined in Equation (4.27). Note that in Theorem 11 we do *not* highlight the difference between compositions because it does not matter, while in the proof for the Corollary below it will make things more clear to do so. Also note that b_d and c_d are not affected by the composition because we may choose K_1, K_2, K_3 and K_4 in such a way that (4.28) always holds for *any* composition with $0 \leq N_I \leq N - 1$ (and thus, $1 \leq N_O \leq N$). In the following Corollary and proof, all notation is the same as in Theorem 11 unless otherwise specified.

Corollary 2. *Define* $r^* = \max_{1 \leq i \leq N} r^i$*, with* r^i *defined in Equation (4.26) via some set of* θ^i *that satisfy Equation (4.16). When the conditions in Theorem 11 are all satisfied, there exists some constant* $0 < \theta_d < 1$ *such that the uniform ultimate bound of the trajectories of the error system is*

$$R_b = \max\{r_b, r^*\}$$

where

$$r_b = \frac{b_d^* + \sqrt{N b_d^{*2} + 4 a_d^* c_d^*}}{2 a_d^* \theta_d} \tag{4.30}$$

with a_d^**,* b_d^**, and* c_d^* *are all constants and*

$$\begin{aligned} a_d^* &= min_{k,N_O} a_d^k(N_O) \\ b_d^* &= max_{N_O} b_d(N_O) \\ c_d^* &= max_{N_O} c_d(N_O) \end{aligned}$$

for $N_O = 1, \ldots, N$ *and* $k = 1, \ldots, N_{N_O}$.

Proof: Note that when the conditions of Theorem 11 are all satisfied, a set of constants θ^i exists and can be found. Also recall that both c_1^i and c_2^i are constants, then $r^i = \frac{c_2^i}{\theta^i c_1^i}$ and $\sigma^i = -(1-\theta^i)c_1^i$, defined in (4.26), are constants for all i and can be found. Thus, numeric values of a_d^*, b_d^*, and c_d^* can be found in terms of known parameters. Now we will first show that this Corollary applies to a fixed $N_O \neq 0$ with a particular composition k.

With $V(E)$ defined in Theorem 11, for $N_O \neq 0$ and $\Pi_O^{(k)}$, from (4.30) we have

$$\begin{aligned} \dot{V}(E) &\leq -a_d^k \sum_{i\in\Pi_O^{(k)}} \|E^i\|^2 + b_d \sum_{i\in\Pi_O^{(k)}} \|E^i\| + c_d \\ &= \sum_{i\in\Pi_O^{(k)}} \underbrace{\left[-a_d^k \left(\|E^i\| - \frac{b_d}{2a_d^k} \right)^2 + \frac{b_d^2}{4a_d^k} \right]}_{F^i} + c_d \end{aligned} \tag{4.31}$$

with a_d^k, b_d and c_d all positive constants. Notice we want to find a bound r_b' such that $\dot{V}(E) < 0$ so long as there exist some $\|E^i\| > r_b'$, $i \in \Pi_O^{(k)}$. Before we start to solve for this r_b', note that F^i in Equation (4.31) can be visualized by Figure 4.1, where F^i is a parabolic function with respect to $\|E^i\|$ and crosses the $\|E^i\|$ axis at two points r_A and r_B, respectively. To have $\dot{V}(E) < 0$, one possibility is such that $\|E^i\| > r_B$ and thus, $F^i < 0$ for all $i \in \Pi_O^{(k)}$. (Note that due to the nature of our problem, we do not consider the case of $\|E^i\| < r_A$, though this also results in $F^i < 0$.) We call this situation the "best situation." While in a more general case, we have some $\|E^i\| > r_B$ while all the other $\|E^i\| \leq r_B$, $i \in \Pi_O^{(k)}$ and we call such a situation the "normal situation." For the best situation case, each $\|E^i\|$ just needs to be *a little* bigger than r_B to achieve $\dot{V}(E) < 0$, which means r_b' is just a little bigger than r_B. In comparison, to get $\dot{V}(E) < 0$ for the normal situation, generally it means those $\|E^i\|$ that satisfy $\|E^i\| > r_B$ have to be further to the right on $\|E^i\|$ axis (i.e., *much* bigger than r_B) to counteract the "positive" effects brought in by those $\|E^i\|$ with $\|E^i\| \leq r_B$, meaning r_b' needs to be much bigger than r_B. With this idea, we can see that the worst r_b' happens when there is only *one* F^i, call it $i = i'$, free to change while all the other F^i, with $i = i_O^1, \ldots, i_O^{N_O}$ and $i \neq i'$, are fixed at their respective maximum (or most "positive" value). Basically this depicts a situation when there is only one agent having its norm of error $\|E^{i'}\|$ slide along $\|E^i\|$ axis (to the right) to bring $\dot{V}(E)$ down to negative value while all other agents in $\Pi_O^{(k)}$ stay in positions as bad as they can (in the sense of keeping stability). Note that by construction, when this $\|E^{i'}\|$ slides along $\|E^i\|$ axis, it always stay in the region of $\|E^{i'}\| \geq r^{i'}$. This seemingly trivial comment in fact plays a significant role in completing the proof, which will become clear soon.

From Equation (4.31) we can see that each F^i achieves its maximum of $\frac{b_d^2}{4a_d^k}$ with $\|E^i\| = \frac{b_d}{2a_d^k}$. Then based on the analysis above, we can solve for the r_b' by letting all $F^i = \frac{b_d^2}{4a_d^k}$ except for $i = i'$. From Equation (4.31), for some constant $0 < \theta_d < 1$

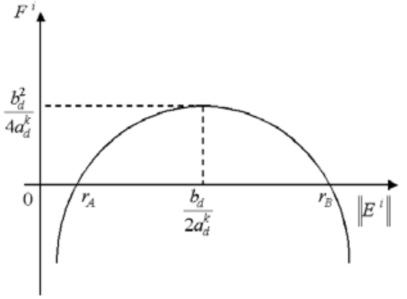


Fig. 4.1. F^i vs $\|E^i\|$.

we have

$$\begin{aligned}\dot{V}(E) &\leq \left[-a_d^k\left(\left\|E^{i'}\right\| - \frac{b_d}{2a_d^k}\right)^2 + \frac{b_d^2}{4a_d^k}\right] + \sum_{i\in\Pi_O^{(k)}, i\neq i'} \frac{b_d^2}{4a_d^k} + c_d \\ &= -a_d^k\left\|E^{i'}\right\|^2 + b_d\left\|E^{i'}\right\| + (N_O - 1)\frac{b_d^2}{4a_d^k} + c_d \\ &= -a_d^k(1-\theta_d)\left\|E^{i'}\right\|^2 - a_d^k\theta_d\left\|E^{i'}\right\|^2 + b_d\left\|E^{i'}\right\| + (N_O - 1)\frac{b_d^2}{4a_d^k} + c_d \\ &\leq -a_d^k(1-\theta_d)\left\|E^{i'}\right\|^2, \forall \left\|E^{i'}\right\| \geq r_b'\end{aligned}$$

where

$$r_b' = \frac{b_d + \sqrt{N_O b_d^2 + 4a_d^k c_d}}{2a_d^k\theta_d}$$

Note that in setting $\|E^i\| = \frac{b_d}{2a_d^k}$ to get $F^i = \frac{b_d^2}{4a_d^k}$, we may violate the prerequisite of $i \in \Pi_O^{(k)}$ since it may happen that $r^i > \frac{b_d}{2a_d^k}$ for some $i \in \Pi_O^{(k)}$. But this violation only adds more conservativeness to the resultant r_b' and does not nullify the fact that r_b' is a valid bound. Specifically, when $r^i > \frac{b_d}{2a_d^k}$ for some $i \in \Pi_O^{(k)}$, the corresponding F^i become "less" positive, as seen from Figure 4.1, and thus, the actual bound will be smaller than the r_b' obtained above. Hence, r_b' is still a valid (upper) bound.

Since $r_b \geq r_b'$ with r_b defined in Equation (4.30), and the positive constant $a_d^* \leq a_d^k$, we have $\dot{V}(E) \leq -a_d^*(1-\theta_d)\left\|E^i\right\|^2$ when $\left\|E^i\right\| \geq r_b$. Now note that the choice of N_O and composition k in the above proof are in fact arbitrary (except that $N_O \neq 0$) and can be time-varying. So we actually have $R_b \geq r_b \geq r_b'$ for any $1 \leq N_O \leq N$ and any composition at any time t. That is, the proof above is actually valid for the general case when N_O ($N_O \neq 0$) and the composition are time-varying.

To complete the proof, we need to show that R_b is also a valid bound for the case of $N_O = 0$, i.e., an empty set Π_O. Notice that $N_O = 0$ means $\|E^i\| < r^i$ for

$i = 1, \ldots, N$. Also notice that $r^i \leq \max_{1 \leq i \leq N} r^i \leq R_b$ for all i. Then by definition, as long as $N_O = 0$, the trajectories of the swarm error system stay within the bound R_b. This completes the proof. ■

The value of R_b is affected by two components: r_b and r^*. We will discuss r_b first. It is easy to see from Equation (4.30) that increasing a_d^* helps to decrease r_b. Notice that a_d^* is a function including many parameters, so it is difficult to provide clear relationships to their effects on r_b. But we may get some ideas based on intuition. Note that a_d^* is related to the minimum eigenvalue of matrix S defined in (4.29), where the components $-(\sigma_O^{i^k} + a^{i^* j})$ on the main diagonal reflect the stabilizing effect of the isolated parts of the composite system, while the cross terms $-a^{i^* j}$ reflect the destabilizing effect of the interconnection parts of the composite system. So, the larger *magnitude* those main diagonal components have (or relatively, the smaller *magnitude* those cross terms have), the "more stable" the composite system is and thus, we may expect the larger a_d^* is. Based on this analysis and Equation (4.23), since smaller $D_{p_1}^i$ and $D_{v_1}^i$ gives larger c_1^i and thus, larger $\sigma_O^{i^k}$ in magnitude (implying "better stability"), it may render larger a_d^* and thus, smaller r_b. When $\sigma_O^{i^k}$ are relatively big, from Equation (4.25), we may deduce that smaller N and thus, larger $a^{i^* j}$ (implying "worse stability") may lead to smaller a_d^* and thus, larger r_b.

Note that b_d^* is affected by $a^{i^* j}$ via $\hat{a}$. Larger $a^{i^* j}$ may lead to larger b_d^* and thus, a larger bound r_b. This is consistent with the analysis in the previous remark. Similar conclusions may be drawn by inspecting c_d^* since it includes K_4, which is affected by $a^{i^* j}$. It is interesting to note that from Equation (4.30), smaller N means smaller r_b, while in the previous remark we mention that smaller N leads to larger $a^{i^* j}$ and thus, larger r_b. These seemingly contradictive conclusions in fact make sense intuitively. Too large N does not help in reducing r_b because with each agent desiring to keep certain distance from others, large N leads to a large swarm radius. Too small N does not always help reducing r_b because the effect of noise becomes more significant when N is small. In other words, with smaller N, the swarm cannot "average" out the noise and thus, the bound on the trajectories is not reduced. This "noise-averaging" idea will become more clear in the following sections, when we deal with identical agents, and later in simulations.

The value of R_b is also affected by r^*. Note that r^* is determined by c_1^i and c_2^i for all i. Specifically, smaller c_2^i and larger c_1^i are helpful in decreasing r^*. Then from Equations (4.23) and (4.24), we can see that all the noise bounds ($D_{p_1}^i$, $D_{v_1}^i$, $D_{p_2}^i$, $D_{v_2}^i$, and D_f^i) affect r^*. Smaller noise bounds help decrease r^* and thus, may decrease R_b. So do smaller k_f^i, k_r^i and r_s^i.

Finally, similar to Theorem 11, lots of conservativeness is introduced into the deduction of Corollary 2. One example is that K_1, K_2, K_4, and thus, b_d^* and c_d^*, are actually functions of N_O. When N_O increases, b_d^* and c_d^* will generally decrease. This fact is not considered in the above deduction because of the complexity that originated in the use of both heterogeneous swarm agents and multiple resource profiles in the environment.

4.2.3 Special Case: Identical Agents

Here, we will study the stability of the system when all the agents are identical (i.e., with $k_p^i = k_p$, $k_v^i = k_v$, $k_r^i = k_r$, $r_s^i = r_s$, and $k_f^i = k_f$, for all i), but with different types of noise and nutrient profiles. Equation (4.4) becomes

$$\begin{aligned} u_i &= -M_i k_p \hat{e}_p^i - M_i k_v \hat{e}_v^i - M_i k v_i + \\ & M_i k_r \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\| \hat{e}_p^i - \hat{e}_p^j \right\|^2}{{r_s}^2} \right) \left(\hat{e}_p^i - \hat{e}_p^j \right) - M_i k_f \left(a_\sigma^i - d_f^i \right) \end{aligned} \quad (4.32)$$

Also $A_i = A$ in Equation (4.12) for all i. Note that with all agents being identical, we have $\Delta k_p^i = 0$ and $\Delta k_v^i = 0$ for all i. Also,

$$\frac{1}{N}\sum_{l=1}^{N} k_r \sum_{j=1, j\neq l}^{N} \exp\left(\frac{-\frac{1}{2}\left\| \hat{e}_p^l - \hat{e}_p^j \right\|^2}{{r_s^l}^2} \right) \left(\hat{e}_p^l - \hat{e}_p^j \right) = 0$$

Letting $\bar{d}_p = \frac{1}{N}\sum_{i=1}^{N} d_p^i$, $\bar{d}_v = \frac{1}{N}\sum_{i=1}^{N} d_v^i$, $\bar{d}_f = \frac{1}{N}\sum_{i=1}^{N} d_f^i$, and $\bar{a}_\sigma = \frac{1}{N}\sum_{i=1}^{N} a_\sigma^i$. Then, Equation (4.7) can be simplified to

$$\dot{\bar{v}} = -k\bar{v} + \underbrace{k_p \bar{d}_p + k_v \bar{d}_v + k_f \bar{d}_f - k_f \bar{a}_\sigma}_{z(t)} \quad (4.33)$$

and

$$\dot{e}_v^i = -k_p e_p^i - (k_v + k) e_v^i + g^i + \phi(E) + \delta^i(E) \quad (4.34)$$

where g^i, $\phi(E)$, and $\delta^i(E)$ are respectively

$$g^i = k_p d_p^i + k_v d_v^i + k_f d_f^i - k_f a_\sigma^i \quad (4.35)$$

$$\phi(E) = -k_p \bar{d}_p - k_v \bar{d}_v - k_f \bar{d}_f + k_f \bar{a}_\sigma \quad (4.36)$$

$$\delta^i(E) = k_r \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\| \hat{e}_p^i - \hat{e}_p^j \right\|^2}{{r_s}^2} \right) \left(\hat{e}_p^i - \hat{e}_p^j \right) \quad (4.37)$$

Using the idea of deriving Equation (4.19) we have

$$\left\| \delta^i(E) \right\| \leq k_r r_s (N-1) \exp\left(-\frac{1}{2} \right) \quad (4.38)$$

Noise with Constant Bounds

In this case, we assume that $d_p^i(t)$ and $d_v^i(t)$ are sufficiently smooth and bounded by some constants for all i,

$$\begin{aligned} \|d_p^i\| &\leq D_p \\ \|d_v^i\| &\leq D_v \end{aligned} \tag{4.39}$$

where $D_p \geq 0$ and $D_v \geq 0$ are known constants. The sensing error on the gradient of the nutrient profile is assumed to be sufficiently smooth and bounded by a known constant $D_f \geq 0$ such that for all i,

$$\|d_f^i\| \leq D_f \tag{4.40}$$

Theorem 12. *Consider the swarm described by the model in Equation (4.3) with control input u_i given as in Equation (4.32). Assume that the nutrient profile for each agent is a plane defined by $\nabla\sigma_p^i(x) = a_\sigma^i$. Also assume the noise satisfies Equations (4.39) and (4.40). Let $\|a_\sigma^*\| = \max_i \left\|a_\sigma^i - \bar{a}_\sigma\right\|$. Then, the trajectories of the swarm error system are uniformly ultimately bounded, and E^i for all i will converge to the set Ω_b, where*

$$\Omega_b = \left\{E : \left\|E^i\right\| \leq \beta_1\beta_2,\ i = 1,2,\ldots,N\right\} \tag{4.41}$$

is attractive and compact, with

$$\beta_1 = \frac{(k_p+1)^2 + (k_v+k)^2}{2k_p(k_v+k)} + \sqrt{\left(\frac{k_p^2 + (k_v+k)^2 - 1}{2k_p(k_v+k)}\right)^2 + \frac{1}{k_p^2}}$$

and

$$\beta_2 = 2k_pD_p + 2k_vD_v + 2k_fD_f + k_f\|a_\sigma^*\| + k_r r_s(N-1)\exp\left(-\frac{1}{2}\right)$$

Moreover, there exists some finite T and constant $0 < \theta < 1$ such that

$$\|\bar{v}(t)\| \leq \exp[-(1-\theta)kt]\,\|\bar{v}(0)\|,\ \forall\, 0 \leq t < T$$

and

$$\|\bar{v}(t)\| \leq \frac{\delta}{k\theta},\ \forall\, t \geq T$$

with $\delta = k_pD_p + k_vD_v + k_fD_f + k_f\|\bar{a}_\sigma\|$.

Proof: To find the set Ω_b note that by following the deduction in Theorem 11, we obtain an equation similar to (4.20), where c_1, c_2 and a^{ij} have the same form as in Equation (4.23), (4.24) and (4.25), except that now $D_{p_1}^i = D_{v_1}^i = 0$ since the noise has a constant bound, the norm of Δ^i in Equation (4.19) is simplified to $\|\delta^i(E)\|$ in (4.38), and the term of $\sqrt{\Delta k_p^{j^2} + \Delta k_v^{j^2}}$ in a^{ij} is zero. So after simplification, we have $a^{ij} = 0$, $c_1 = 1$, and

$$c_2 = \beta_1\left(2k_pD_p + 2k_vD_v + 2k_fD_f + k_f\,\|a_\sigma^*\| + k_r r_s(N-1)\exp\left(-\frac{1}{2}\right)\right) = \beta_1\beta_2$$

where D_p, D_v, D_f and β_1 are the counterparts of $D^i_{p_2}$, $D^i_{v_2}$, D^i_f and β^i_1 in Equation (4.24), respectively. From (4.20), we have

$$\dot{V}(E) \leq \sum_{i=1}^{N} \left(-\|E^i\|^2 + \beta_1\beta_2\|E^i\|\right)$$

and the uniform ultimate boundedness is specified in the discussion just after Theorem 11. Moreover, if

$$\left\|E^i\right\| > \beta_1\beta_2 \tag{4.42}$$

for all i, we have $\dot{V}(E) < 0$. So the set

$$\Omega_b = \left\{E : \left\|E^i\right\| \leq \beta_1\beta_2,\ i = 1,2,\ldots,N\right\}$$

is attractive and compact. Also we know that within a finite amount of time, $E^i \to \Omega_b$. This means that we can guarantee that if the swarm is not cohesive, it will seek to be cohesive, but only if it is a certain distance from cohesiveness as indicated by (4.42).

To study the boundedness of $\bar{v}(t)$, choose a Lyapunov function

$$V_{\bar{v}} = \frac{1}{2}\bar{v}^\top\bar{v}$$

defined on $D = \{\bar{v} \in \mathbb{R}^n \mid \|\bar{v}\| < r_v\}$ for some $r_v > 0$, and we have

$$\dot{V}_{\bar{v}} = \bar{v}^\top\dot{\bar{v}} = -k\bar{v}^\top\bar{v} + \bar{v}^\top z(t)$$

with $z(t)$ defined in Equation (4.33). Since $\left\|d^j_p\right\| \leq D_p$ for all j, we have $\left\|\bar{d}_p\right\| = \left\|\frac{1}{N}\sum_{j=1}^{N} d^j_p\right\| \leq D_p$. Similarly, $\left\|\bar{d}_v\right\| \leq D_v$ and $\left\|\bar{d}_f\right\| \leq D_f$. Thus, we have

$$\|z(t)\| \leq \left\|k_p\bar{d}_p\right\| + \left\|k_p\bar{d}_v\right\| + \left\|k_f\bar{d}_f\right\| + \left\|k_f\bar{a}_\sigma\right\| \leq \delta$$

If $\delta < k\theta r_v$ for all $t \geq 0$, all $\bar{v} \in D$ and some positive constant $\theta < 1$, then it can be proven that for all $\|\bar{v}(0)\| < r_v$ and some finite T we have

$$\|\bar{v}(t)\| \leq \exp\left[-(1-\theta)kt\right]\|\bar{v}(0)\|,\ \forall\, 0 \leq t < T$$

and

$$\|\bar{v}(t)\| \leq \frac{\delta}{k\theta},\ \forall\, t \geq T$$

Since this holds globally we can take $r_v \to \infty$ so these equations hold for all $\bar{v}(0)$. This completes the proof. ■

The size of Ω_b in Equation (4.41), which we denote by $|\Omega_b|$, is directly a function of several known parameters. If there are no sensing errors, i.e., $D_p = D_v = D_f = 0$, then Ω_b reduces to the set representing the no-noise case. For fixed values of N, k_p, k_v, k, and k_r, if we increase r_s, each agent has a larger region from which it will repel its neighbors so $|\Omega_b|$ is larger. For fixed k_r, k_p, k_v, k, and r_s if we let $N \to \infty$, then $|\Omega_b| \to \infty$ as we expect due to the repulsion.

It is interesting to note that in some swarms N is very large and when there is no biasing of sensing errors, we have $\bar{d_p} \approx \bar{d_v} \approx \bar{d_f} \approx 0$. This reduces the bound defined by β_2. Also when $\|a_\sigma^*\|$ is decreased, implying that a_σ^i is closer to a_σ^j for all i and j, then β_2 is smaller. In the special case when all a_σ^i are the same, we have $\|a_\sigma^*\| = 0$ and $|\Omega_b|$ is minimized with respect to resource profiles. This means when $\|a_\sigma^*\|$ is large, the agents pursue resource profiles that are far different from each other and the swarm is spread out, while when those profiles are equal to each other, all the agents move along the same profile and smaller swarm size is achieved.

If δ and θ are fixed, with increasing k we get that $\|\bar{v}(t)\|$ decreases faster for $0 \leq t < T$ and has smaller bound for $t \geq T$. If δ gets larger with k and θ fixed, $\|\bar{v}(t)\|$ has larger bound for $t \geq T$; hence if the magnitude of the noise increases, this increases δ and hence there can be larger magnitude changes in the ultimate average velocity of the swarm (e.g., the average velocity could oscillate). Note that if in Equation (4.33) $z(t) \approx 0$ (e.g., due to noise that destroys the directionality of the resource profile a_σ), then the above bound may be reduced but the swarm could be going in the wrong direction.

Regardless of the size of the bound it is interesting to note that while the noise destroys the ability of an individual agent to follow a gradient accurately, the average sensing errors of the group are what changes the direction of the group's movement relative to the direction of the gradient of $\sigma_p(x)$. In some cases when the swarm is large (N big), it can be that $\bar{d_p} \approx \bar{d_v} \approx \bar{d_f} \approx 0$ since the average sensing error is zero and the group will perfectly follow the proper direction for foraging (this may be a reason why for some organisms, large group size is favorable). In the case when $N = 1$ (i.e., single agent), there is no opportunity for a cancellation of the sensor errors; hence an individual may not be able to climb a noisy gradient as easily as a group.

Finally, note that there is an intimate relationship between sensor noise and observations of biological swarms that there is a type of "inertia" of a swarm. Note that for large swarms (high N) there can be regions where the average sensor noise is small so that agents in that region move in the right direction. In other regions there may be alignments of the errors and hence the agents may not be all moving in the right direction so they may get close to each other and impede each other's motion, having the effect of slowing down the whole group. With no noise, the group inertia effect is not found since each agent is moving in the right direction. The presence of sensor noise generally can make it more difficult to get the group moving in the proper direction. Large swarms can help move the group in the right direction, but at the expense of possibly slowing their movement initially in a transient period.

Constant Errors

In this case, we assume each agent senses the velocity and position of other members and the nutrient profile with some constant errors.

Theorem 13. *Consider the swarm described by the model in Equation (4.3) with control input u_i given as in Equation (4.32). Assume that the nutrient profile for* *each agent is a plane defined by* $\nabla \sigma_p^i(x) = a_\sigma^i$*. Also assume the noise* d_p^i*,* d_v^i*, and* d_f^i *are time-invariant for each agent so that* $\bar{d}_p = \frac{1}{N}\sum_{i=1}^N d_p^i$*,* $\bar{d}_v = \frac{1}{N}\sum_{i=1}^N d_v^i$*,* $\bar{d}_f = \frac{1}{N}\sum_{i=1}^N d_f^i$*, and* $\bar{a}_\sigma = \frac{1}{N}\sum_{i=1}^N a_\sigma^i$ *are constants. Then, the error dynamics of the swarm system are uniformly ultimately bounded and* E^i*,* $i = 1,\ldots,N$*, will converge to the attractive and compact set* Ω_t *defined by*

$$\Omega_t = \left\{E : \left\|E^i\right\| \leq \alpha^i\beta_1,\ i = 1,\ldots,N\right\} \tag{4.43}$$

where β_1 *is defined in Theorem 12 and*

$$\begin{aligned}\alpha^i = &\left\|k_p\left(d_p^i - \bar{d}_p\right) + k_v\left(d_v^i - \bar{d}_v\right) + k_f\left(d_f^i - \bar{d}_f\right) - k_f\left(a_\sigma^i - \bar{a}_\sigma\right)\right\| + \\ &k_r r_s(N-1)\exp\left(-\frac{1}{2}\right)\end{aligned}$$

Moreover, $e_v^i \to 0$ *and*

$$v_i(t) \to \frac{k_p\bar{d}_p + k_v\bar{d}_v + k_f\bar{d}_f - k_f\bar{a}_\sigma}{k} \tag{4.44}$$

for all i *as* $t \to \infty$*.*

Proof: From Equations (4.18) and (4.34) we have

$$\begin{aligned}\dot{V}_i &= -E^{i\top}Q_iE^i + 2E^{i\top}P_iB\left(g^i + \phi(E) + \delta^i(E)\right) \\ &\leq -\lambda_{min}(Q_i)\left\|E^i\right\|^2 + 2\left\|E^i\right\|\lambda_{max}(P_i)\left\|k_p\left(d_p^i - \bar{d}_p\right) + k_v\left(d_v^i - \bar{d}_v\right) + \right. \\ &\quad \left. k_f\left(d_f^i - \bar{d}_f\right) - k_f\left(a_\sigma^i - \bar{a}_\sigma\right)\right\| + 2\left\|E^i\right\|\lambda_{max}(P_i)k_r r_s(N-1)\exp\left(-\frac{1}{2}\right) \\ &= -\lambda_{min}(Q_i)\left\|E^i\right\|^2 + 2\left\|E^i\right\|\lambda_{max}(P_i)\alpha^i \\ &= -\lambda_{min}(Q_i)\left\|E^i\right\|\left(\left\|E^i\right\| - 2\frac{\lambda_{max}(P_i)}{\lambda_{min}(Q_i)}\alpha^i\right)\end{aligned}$$

Following the idea in Theorem 11, we have by letting $Q_i = I$, β_1 as the counterpart of β_1^i in Theorem 11. So we have

$$\dot{V}_i \leq -\left\|E^i\right\|\left(\left\|E^i\right\| - \alpha^i\beta_1\right)$$

That is, $\dot{V}_i < 0$ if $\left\|E^i\right\| > \alpha^i\beta_1$. So the set

$$\Omega_t = \left\{E : \left\|E^i\right\| \leq \alpha^i\beta_1,\ i = 1,2,\ldots,N\right\}$$

is attractive and compact. Also we know that within a finite amount of time, $E^i \to \Omega_t$.

Next, to find the ultimate velocity of each agent in the swarm, we consider Ω_t and a Lyapunov function $V^o(E) = \sum_{i=1}^N V_i^o\left(E^i\right)$ with

$$V_i^o\left(E^i\right) = \frac{1}{2}k_p\left(e_p^i - \frac{\gamma}{k_p}\right)^\top\left(e_p^i - \frac{\gamma}{k_p}\right) + \frac{1}{2}e_v^{i\top}e_v^i + k_r r_s^2 \sum_{j=1,j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{r_s^2}\right) \tag{4.45}$$

where the constant $\gamma = k_p\left(d_p^i - \bar{d}_p\right) + k_v\left(d_v^i - \bar{d}_v\right) + k_f\left(d_f^i - \bar{d}_f\right) - k_f\left(a_\sigma^i - \bar{a}_\sigma\right)$. Note that this Lyapunov function is not positive definite, but $V_i^o(E^i) > 0$. Here, we think of the swarm moving so as to *minimize* $V^o(E)$ with the i^{th} agent trying to minimize $V_i^o(E^i)$. Agents try to place themselves at positions to reduce the first term in Equation (4.45), achieve a velocity to reduce the second term, and move to a distance from each other to minimize repulsion quantified in the last term. There is a resulting type of balance that is sought between the conflicting objectives that each of the three terms represent.

Now we have

$$\begin{aligned}\nabla_{e_p^i}V_i^o &= k_p e_p^i - \gamma - k_r\sum_{j=1,j\neq i}^{N}\exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{r_s^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right)\\ \nabla_{e_v^i}V_i^o &= e_v^i\end{aligned}$$

so from this and Equation (4.34)

$$\begin{aligned}\dot{V}_i^o &= \left[\nabla_{e_p^i}V_i^{o\top}, \nabla_{e_v^i}V_i^{o\top}\right]\dot{E}_i\\ &= k_p e_p^{i\top}e_v^i - \gamma^\top e_v^i - k_r\sum_{j=1,j\neq i}^{N}\exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{r_s^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right)^\top e_v^i +\\ &\quad e_v^{i\top}\Bigg(-k_p e_p^i - k_v e_v^i - k e_v^i + k_p\left(d_p^i - \bar{d}_p\right) + k_v\left(d_v^i - \bar{d}_v\right) + k_f\left(d_f^i - \bar{d}_f\right) -\\ &\quad k_f\left(a_\sigma^i - \bar{a}_\sigma\right) + k_r\sum_{j=1,j\neq i}^{N}\exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{r_s^2}\right)\left(\hat{e}_p^i - \hat{e}_p^j\right)\Bigg)\\ &= -\left(k_v + k\right)e_v^{i\top}e_v^i\end{aligned}$$

Hence, $\dot{V}^o = -\left(k_v + k\right)\sum_{i=1}^{N}\|e_v^i\|^2 \leq 0$ on $E \in \Omega$ for any compact set Ω. Choose Ω so it is positively invariant, which is clearly possible, and so $\Omega_e \in \Omega$ where

$$\Omega_e = \{E :\ \dot{V}^o(E) = 0\} = \{E :\ e_v^i = 0,\ i = 1,2,\ldots,N\}$$

From LaSalle's Invariance Principle we know that if $E(0) \in \Omega$ then $E(t)$ will converge to the largest invariant subset of Ω_e. Hence $e_v^i(t) \rightarrow 0$ as $t \rightarrow \infty$. From Equation (4.33), we have

$$\bar{v}(t) \longrightarrow \frac{k_p\bar{d}_p + k_v\bar{d}_v + k_f\bar{d}_f - k_f\bar{a}_\sigma}{k}$$

as $t \longrightarrow \infty$ since in this case $z(t)$ is a constant with respect to time. Thus, $v_i(t)$ approaches this value also for all i as $t \longrightarrow \infty$. This completes the proof. ■

From (4.44) we see that all agents will ultimately be moving at the same velocity despite the existence of constant errors. Contrast this with the earlier more general cases where it is possible that $\bar{v}$ and v_i ultimately, for example, oscillate. Next, note that even if $a_\sigma^i = a_\sigma^j$ for all i and j, the presence of $\bar{d}_p$, $\bar{d}_v$, and $\bar{d}_f$ represent the effects of sensor errors and they can result in the swarm not properly following the direction of the profile even when they all intend to go in the same direction. In the case when we have $\bar{d}_p \approx \bar{d}_v \approx \bar{d}_f \approx 0$ with N large enough, then all those agents will be following the "averaged" profile $-\frac{k_f}{k}\bar{a}_\sigma$. That is, due to the desire to stay together, they each sacrifice following their own profile and compromise to follow the averaged profile. In the special case when $\bar{a}_\sigma = 0$ or $a_\sigma^i = 0$ for all i (no resource profile effect), and no sensor errors, both $\bar{v}(t)$ and $v_i(t)$ will go to zero as $t \longrightarrow \infty$, representing the aggregation of the group independent of the environment. The size of Ω_t in Equation (4.43), which we denote by $|\Omega_t|$, is directly a function of several known parameters. All the remarks for the previous case, i.e., noise with constant bounds, apply here.

4.3 Stability Analysis of Swarm Trajectory Following

In this section, we briefly analyze the stability of the swarm error system when each agent is trying to track their respective trajectories. This is done by applying the same approach as in Section 4.2 to a slightly reformulated system model. Specifically, redefine the errors as

$$\begin{aligned} e_p^i &= x_i - x_d^i \\ e_v^i &= v_i - v_d^i \end{aligned}$$

where x_d^i is a sufficiently smooth desired position trajectory for agent i, $i = 1, \ldots, N$, and $v_d^i = \dot{x}_d^i$. We assume that there exist known bounds for $\dot{x}_d^i$ and $\dot{v}_d^i$ such that

$$\begin{aligned} \|\dot{x}_d^i\| &\le D_{x_d^i} \\ \|\dot{v}_d^i\| &\le D_{v_d^i} \end{aligned}$$

where $D_{x_d^i}$ and $D_{v_d^i}$ are known positive constants. Assume the nutrient profile for each agent is a plane defined by $\nabla\sigma_p^i(x) = a_\sigma^i$. Also let $\dot{e}_p^i$, $\dot{e}_v^i$, $\hat{e}_p^i$, $\hat{e}_v^i$, u_i, and E^i be defined in the same form as in Section 4.2.1. Then the error dynamics of the i^{th} agent are

$$\dot{E}^i = \begin{bmatrix} 0 & I \\ -k_p^i I & -\left(k_v^i + k\right) I \end{bmatrix} E^i + \begin{bmatrix} 0 \\ I \end{bmatrix} \left(g^i + \delta^i(E)\right) \tag{4.46}$$

where

$$g^i = k_p^i d_p^i + k_v^i d_v^i + k_f^i d_f^i - k_f^i a_\sigma^i \tag{4.47}$$

$$\delta^i(E) = k_r^i \sum_{j=1, j\neq i}^{N} \exp\left(\frac{-\frac{1}{2}\left\| \hat{e}_p^i - \hat{e}_p^j \right\|^2}{{r_s^i}^2} \right) \left(\hat{e}_p^i - \hat{e}_p^j \right) - k v_d^i - \dot{v}_d^i \tag{4.48}$$

Let $d_f^i(t)$, $d_p^i(t)$, and $d_v^i(t)$ be specified in the same way as in Section 4.2.2. Then we have the following theorem.

Theorem 14. *Consider the swarm described by the model in Equation (4.3) with control input u_i given in Equation (4.4). Let β_1^i be defined in Theorem 11. If for all i we have*

$$k_p^i D_{p_1}^i + k_v^i D_{v_1}^i < \frac{1}{\beta_1^i} \tag{4.49}$$

then the trajectories of the error system, specified by Equation (4.46), are uniformly ultimately bounded. Furthermore, E^i for all i will converge to an attractive and compact set Ω_f defined as

$$\Omega_f = \left\{ E : \left\| E^i \right\| \leq \frac{\tilde{c}_2^i}{\tilde{c}_1^i}, i = 1, \ldots, N \right\} \tag{4.50}$$

where

$$\tilde{c}_1^i = 1 - \beta_1^i \left(k_p^i D_{p_1}^i + k_v^i D_{v_1}^i \right) \tag{4.51}$$

$$\tilde{c}_2^i = \beta_1^i \left(k_p^i D_{p_2}^i + k_v^i D_{v_2}^i + k_f^i D_f^i + k_f^i \left\| a_\sigma^i \right\| + k D_{x_d^i} + D_{v_d^i} + \hat{\delta}^i \right) \tag{4.52}$$

with $\hat{\delta}^i = k_r^i r_s^i (N-1) \exp\left(-\frac{1}{2}\right)$. Moreover, if we have for any i and j

$$\left\| x_d^i - x_d^j \right\| \leq D_x \tag{4.53}$$

where D_x is a known constant, then the swarm will stay cohesive and

$$\|x_i - \bar{x}\| \leq \frac{\tilde{c}_2^i}{\tilde{c}_1^i} + \frac{1}{N} \sum_{j=1}^{N} \frac{\tilde{c}_2^j}{\tilde{c}_1^j} + D_x \tag{4.54}$$

for all i.

Proof: Note that g^i and $\delta^i(E)$, defined in Equation (4.47) and (4.48), are bounded by $k_p^i D_{p_2}^i + k_v^i D_{v_2}^i + k_f^i D_f^i + k_f^i \left\| a_\sigma^i \right\|$ and $k D_{x_d^i} + D_{v_d^i} + \hat{\delta}^i$, respectively. By following exactly the same method in the proof of Theorem 11, we obtain

$$\dot{V}(E) \leq \sum_{i=1}^{N} \left(-\tilde{c}_1^i \left\| E^i \right\|^2 + \tilde{c}_2^i \left\| E^i \right\| \right) \tag{4.55}$$

where $V(E)$ is the Lyapunov function defined for the whole error system, as specified in the proof of Theorem 11. Then the uniform ultimate boundedness of the error system and the set Ω_f are easily obtained via Equation (4.55).

When Equation (4.53) is satisfied, we can show that the cohesiveness of the swarm is conserved. To see this, note that for arbitrary i and j with $i \neq j$,

$$\|x_i - x_j\| = \left\|\left(e_p^i + x_d^i\right) - \left(e_p^j + x_d^j\right)\right\| \le \|e_p^i - e_p^j\| + \left\|x_d^i - x_d^j\right\| \le \|E^i\| + \|E^j\| + D_x \tag{4.56}$$

so from Equation (4.50) and (4.56) we have

$$\|x_i - \bar{x}\| = \frac{1}{N}\left\|\sum_{j=1}^{N}(x_i - x_j)\right\| \le \frac{1}{N}\sum_{j=1}^{N}\|x_i - x_j\| \le \frac{\tilde{c}_2^i}{\tilde{c}_1^i} + \frac{1}{N}\sum_{j=1}^{N}\frac{\tilde{c}_2^j}{\tilde{c}_1^j} + D_x$$

This is Equation (4.54). This completes the proof. ■

Comparing Equation (4.20) with (4.55), we can see that the latter one does not include any cross term. This is because the errors for the swarm cohesion case are defined as the difference between an agent and the swarm centers ($\bar{x}$ and $\bar{v}$), which are affected by all the agents in the swarm, while the errors for the trajectory following case are defined as the difference between an agent and the given position and velocity trajectories, which are *not* affected by the behaviors of any agent. This absence of a cross term significantly simplifies the proof of the theorem.

By comparing Theorems 11 and 14, we can see that Theorem 14 will hold whenever Theorem 11 holds, as long as $D_{x_d^i}$ and $D_{v_d^i}$ exist. This means that a cohesion property of a swarm in a certain environment guarantees the stability of that swarm in following *any* bounded trajectory in the same environment. Similar to the case of Theorem 11, when Equation (4.49) holds, uniform ultimate boundedness is obtained. This condition only depends on k_p^i, k_v^i, D_{p_1}, and D_{v_1}. Although the remaining parameters, including k_r^i, r_s^i, k_f^i, D_{p_2}, D_{v_2}, D_f, and a_σ^i, do not affect the boundedness of the error system, they do affect the ultimate bound. In the special case when $D_{p_1}^i = D_{v_1}^i = 0$ for all i, Equation (4.49) always holds and thus, the swarm error system is always bounded. Smaller $\|a_\sigma^i\|$ may decrease the bound. Smaller magnitudes of the position and velocity trajectories also help in decreasing the ultimate bound. Our analysis includes the possibility that the resource profiles indicate that the agents should go in the opposite direction that is indicated by (x_d^i, v_d^i). If there is an alignment between where the resource profiles say to go and the (x_d^i, v_d^i), then the size of the bound decreases.

Finally, note that in the special case when v_d^i and all sensing errors are constant, we have that $e_v^i \to 0$ as $t \to \infty$ for all i, meaning v_i of each agent will be precisely following the given constant velocity trajectory ultimately in such a case. To see this, let $\gamma^i = k_p^i d_p^i + k_v^i d_v^i - k v_d^i + k_f^i d_f^i - k_f^i a_\sigma^i$ and construct for all i a Lyapunov function

$$V_i^o\left(E^i\right) = \frac{1}{2}k_p^i\left(e_p^i - \frac{\gamma^i}{k_p^i}\right)^\top\left(e_p^i - \frac{\gamma^i}{k_p^i}\right) + \frac{1}{2}e_v^{i\top}e_v^i + k_r^i r_s^{i2}\sum_{j=1, j\neq i}^{N}\exp\left(\frac{-\frac{1}{2}\left\|\hat{e}_p^i - \hat{e}_p^j\right\|^2}{r_s^2}\right) \tag{4.57}$$

Note that $\dot{v}_d^i = 0$ when v_d^i is constant. Then by following the method in the proof of Theorem 13, the above claim holds.

4.4 Simulation Examples

In this section, we will show some simulation results for both the no-noise and noise cases. Unless otherwise stated, in all the following simulations the parameters, which we refer to as "normal parameters," are: $N = 50$, $k_p^i = k_p = 1$, $k_v^i = k_v = 1$, $k = 0.1$, $k_f^i = k_f = 0.1$, $k_r^i = k_r = 10$, $r_s^i = r_s = 0.1$, and the three dimensional nutrient plane profile $\nabla\sigma_p^i(x) = a_\sigma^i = [1,\ 2,\ 3]^\top$ for all i.

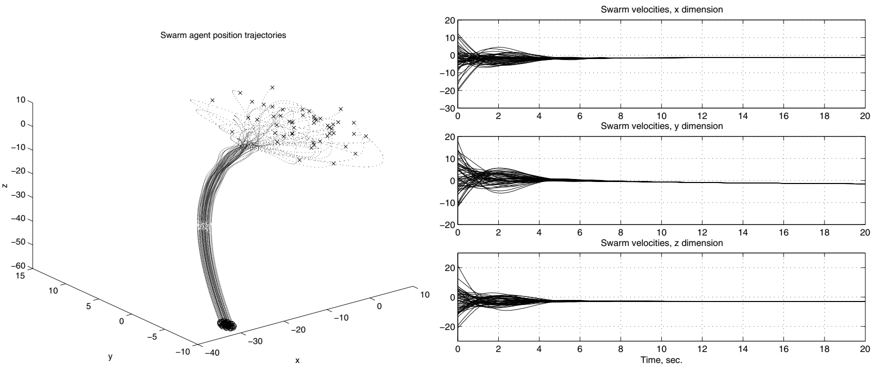
4.4.1 No-Noise Case

All the simulations in this case are run for 20 seconds. The position and velocity trajectories of the swarm agents with the normal parameters are shown in Figure 4.2. All the agents are assigned initial velocities and positions randomly. At the beginning of the simulation, they appear to move around erratically. But soon, they swarm together and continuously reorient themselves as a group to slide down the plane profile. Note how these agents gradually catch up with each other while still keeping mutual spacing. Recalling from the previous sections that for this case $v_i(t) \to -\frac{k_f}{k}a_\sigma$ for all i as $t \to \infty$, and this can be seen from Figure 4.2(b) since the final velocity of each swarm agent is indeed $-[1,\ 2,\ 3]^\top$.

Next, we change the values of some of the parameters to show their impact on the system behavior. Figures 4.3(a) and (b) show the results of keeping all normal parameters unchanged except for increases of k_r to 1000 and r_s to 1, respectively. Since both k_r and r_s are parameters affecting the repulsion range of each agent, as expected we find that the final swarm size becomes larger than in the previous case, while the swarm velocity and settling speed do not change much. Effects of other parameters are also as expected.

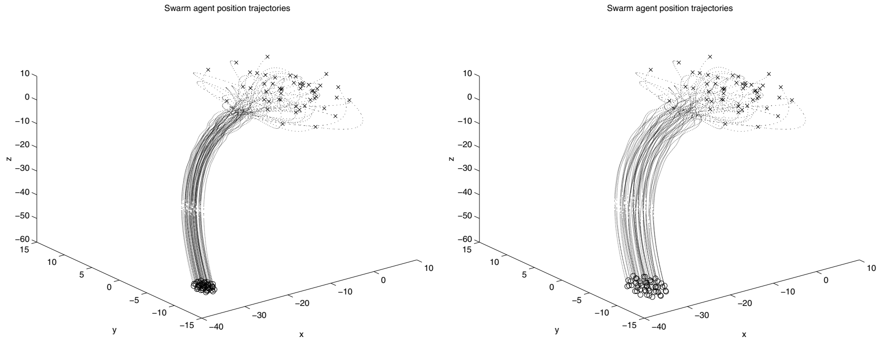
4.4.2 Noise Case

Now we will consider the case when noise exists. In our simulations, the solutions of Duffing's equation are used as "noise" so that the noise is guaranteed to be differentiable. Of course many other choices are possible, e.g., ones that lead to errors on a higher or lower frequency spectrum. Duffing's equation is in the form



(a) Position trajectories. (b) Velocity trajectories.

Fig. 4.2. No noise case with normal parameters.



(a) Position trajectories ($k_r = 1000$, $r_s = 0.1$). (b) Position trajectories ($r_s = 1$, $k_r = 10$).

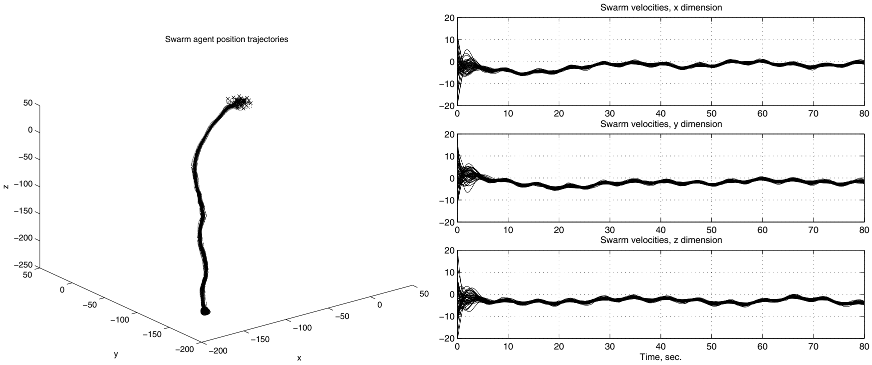
Fig. 4.3. No noise case with parameters changed.

$$\ddot{\vartheta} + \delta_D \dot{\vartheta} - \vartheta + \vartheta^3 = \gamma_D \cos(\omega_D t)$$

In the simulations we use $\delta_D = 0.25$, $\gamma_D = 0.30$ and $\omega_D = 1.0$ so that the solution ϑ of Duffing's equation demonstrates chaotic behavior. We will simulate many such equations to generate noise on position, velocity, and resource profile gradient sensing. We denote by ϑ_i the solution to the i^{th} Duffing's equation that we simulate. Note that the magnitude of ϑ is always bounded by a value of 1.5. Thus, we can easily set the noise bounds with some scaling factors. For example, in the case of noise with a linear bound, the position sensing noise is generated by $d_p^i = \frac{D_{p1}\vartheta_1^i}{1.5}\|E^i\| + \frac{D_{p2}\vartheta_2^i}{1.5}$ so that Equation (4.13) is satisfied.

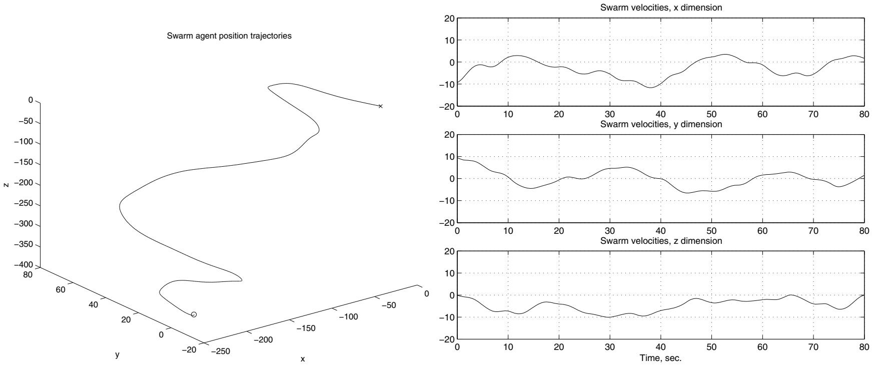
In this case, we run the simulations for 80 seconds. All the normal parameters used in the no-noise case are kept unchanged except the number of agents in the swarm in certain simulations, which is specified in the relevant figures. Figures 4.4 and 4.5 illustrate the case with linear noise bounds for a typical simulation run. The noise bounds are $D_{p_1} = D_{v_1} = 0.05$, $D_{p_2} = D_{v_2} = 1$, and $D_f = 10$, respectively.

Forming a swarm may help the agents go down the gradient of the nutrient profile without being significantly distracted by noise. Figure 4.4 shows that the existence of noise does affect the swarm's ability to follow the profile, which is indicated by the oscillation of the position and velocity trajectories. But with all the agents working together, especially when the agents number N is large, they are able to move in the right direction and thus, minimize the negative effects of noise. In comparison, Figure 4.5 shows the case when there is only one agent. Since the single agent cannot benefit from the averaging effects possible when there are many agents, the noise more adversely affects its performance in terms of accurately following the nutrient profile.



(a) Agent position trajectories. (b) Agent velocity trajectories.

Fig. 4.4. Linear noise bounds case ($N = 50$).



(a) Agent position trajectories. (b) Agent velocity trajectories.

Fig. 4.5. Linear noise bounds case ($N = 1$).

4.5 Further Issues

4.5.1 Extensions and Generalizations

In this chapter, we have derived conditions under which social foraging swarms consisting of agents with double-integrator dynamics maintain cohesiveness and follow a resource profile even in the presence of sensor errors and noise on the profile. We also studied the case where all agents are identical, and special types of noise. While we only studied the "plane profile," extensions to profiles with other shapes are possible. Moreover, even though we only studied one type of attraction and repulsion function the results can be extended to other classes using approaches such as the ones in Chapter 3. Our simulations illustrated advantages of social foraging in large groups relative to foraging alone since they show that a noisy resource profile can be more accurately tracked by a swarm than an individual. Moreover, the simulations produced agent trajectories that are curiously reminiscent of those seen in biology (e.g., by some insects).

It would be interesting to determine if the model here, with appropriately chosen parameters, is an acceptably accurate representation for some social organisms and whether the predictions of the analysis would also accurately represent their group-level behavior.

4.5.2 For Further Reading

This chapter is based on the results in [153]. For some key steps in our proofs we used the theory of stability of interconnected systems and the interested reader should see [170] for more details. The notion that animals can forage in noisy environments more efficiently as a group than individually has been studied for some organisms [104, 151]. Additional work on gradient climbing by engineering swarms, including work on climbing noisy gradients, is in [13, 187]. Swarms composed of agents with double-integrator dynamics but without sensing errors have been considered in [97, 127, 128].

5

Swarms of Fully Actuated Agents with Model Uncertainty

5.1 Fully Actuated Agent Model with Uncertainty

In this chapter, we consider a swarm of N agents whose dynamics evolve based on a more realistic agent dynamics model compared to the single integrator and the double integrator point mass models considered in the preceding chapters. In particular, we assume that the dynamics of the agents obey the *fully actuated model*

$$M_i(x_i)\ddot{x}_i + f_i(x_i,\dot{x}_i) = u_i, 1 \leq i \leq N \tag{5.1}$$

where x_i represents the position or configuration of agent i, $M_i(x_i) \in \mathbb{R}^{n\times n}$ is its mass or inertia matrix, $f_i(x_i,\dot{x}_i) \in \mathbb{R}^n$ represents the centripetal, Coriolis, gravitational effects, and additive disturbances. It is a realistic model for fully actuated omnidirectional mobile robots or for some fully actuated robotic manipulators [35, 111, 261]. What makes the model even more realistic is that it is assumed that (5.1) contains uncertainties and disturbances. In particular, it is assumed that

$$f_i(x_i,\dot{x}_i) = f_i^k(x_i,\dot{x}_i) + f_i^u(x_i,\dot{x}_i), 1 \leq i \leq N$$

where $f_i^k(\cdot,\cdot)$ represents the *known* part and $f_i^u(\cdot,\cdot)$ represents the *unknown* part. The unknown part is assumed to be bounded with a known bound, i.e.,

$$\|f_i^u(x_i,\dot{x}_i)\| \leq \bar{f}_i(x_i,\dot{x}_i), 1 \leq i \leq N$$

where $\bar{f}_i(x_i,\dot{x}_i)$ are known for all i. Moreover, besides the additive disturbances and uncertainties, it is assumed that for all i the mass/inertia matrix is unknown but is nonsingular and lower and upper bounded by known bounds. In other words, the matrices $M_i(x_i)$ satisfy

$$\underline{M}_i\|y\|^2 \leq y^\top M_i(x_i)y \leq \bar{M}_i\|y\|^2, 1 \leq i \leq N \tag{5.2}$$

where $y \in \mathbb{R}^n$ is arbitrary and $\underline{M}_i$ and $\bar{M}_i$ are known and satisfy $0 < \underline{M}_i < \bar{M}_i < \infty$. These uncertainties provide an opportunity for developing algorithms that are robust with respect to the above types of realistic uncertainties and disturbances.

Our objective is, given the agent dynamics in (5.1) together with all the stated assumptions, to develop appropriate controllers such that a desired behavior is achieved by the swarm. In other words, we would like to design each of the control inputs u_i so that the swarm coordination and control objectives, such as aggregation, foraging, formation control, swarm tracking, etc., are achieved. We will use sliding mode control techniques for that purpose.

5.2 Controller Development

5.2.1 Aggregation, Foraging, and Formation Control

We know from Chapter 3 that if the agents are forced to move based on the equation (see Equations (3.1) and (3.2))

$$\dot{x}_i = -\nabla_{x_i} J(x) \tag{5.3}$$

where $x^\top = [x_1^\top, x_2^\top, \ldots, x_N^\top] \in \mathbb{R}^{Nn}$ denotes the vector of concatenated positions of all the agents and $J : \mathbb{R}^{Nn} \to \mathbb{R}$ is an appropriately designed potential function which represents the inter-agent interactions and/or the effects of the environment (the resource profile) on the behavior of the agents, then the swarm will behave in a predictable manner. In other words, if we can force the agents with the dynamics in (5.1) to move based on (5.3) and choose the potential functions appropriately, then the results in Chapter 3 will be recovered (meaning we will achieve stable aggregation, foraging, formation stabilization, swarm tracking or other coordinated behavior based on the context) for swarms composed of agents with vehicle dynamics obeying (5.1). This is exactly the approach we will take in this chapter.

The sliding mode control technique has the property of reducing the motion (and the analysis) of a system to a lower dimensional space, which makes it very suitable for this application (since we want to enforce the system dynamics to obey (5.3), which constitutes only a part of the agents state in (5.1)). Moreover, it is a robust strategy which can suppress disturbances and system uncertainties. This is also an important desirable property since the agent dynamics contain disturbances and uncertainties.

With the objective to achieve satisfaction of (5.3) let us first define the n-dimensional sliding manifold for agent i as

$$s_i = \dot{x}_i + \nabla_{x_i} J(x) = 0, i = 1, \ldots, N \tag{5.4}$$

Our objective is to drive the agents to their corresponding sliding manifolds $s_i = 0$ since once all the agents reach their corresponding sliding manifolds $s_i = 0$ we have

$$\dot{x}_i = -\nabla_{x_i} J(x), i = 1, \ldots, N$$

satisfied which is exactly the motion equation in (5.3).

Given the vehicle dynamics in (5.1), the problem becomes to design the control inputs u_i such that to enforce occurrence of sliding mode. A sufficient condition for sliding mode to occur, referred to as the *sliding condition* in the literature, is given by [53]

$$s_i^\top \dot{s}_i < 0 \tag{5.5}$$

for all $i = 1, \dots, N$. This condition guarantees that sliding mode occurs and that the sliding manifold is asymptotically reached. For our case, however, we need the sliding mode to occur in all surfaces (i.e., for all agents) in finite time. Therefore, actually we need to enforce satisfaction of the inequality

$$s_i^\top \dot{s}_i < -\varepsilon_i \|s_i\| \tag{5.6}$$

for some $\varepsilon_i > 0$. The condition in (5.6) is called the *reaching condition* and guarantees that sliding mode occurs in finite time [53]. In fact using the comparison principle [136] one can show that the condition in (5.6) guarantees that the sliding surface $s_i = 0$ is reached in finite time bounded by

$$t_i \leq \frac{\|s_i(0)\|}{\varepsilon_i}$$

Then, sliding mode on all surfaces will occur in finite time bounded by

$$\bar{t}_{sm} = \max_{i=1,\dots,N} \{t_i\} = \max_{i=1,\dots,N} \left\{ \frac{\|s_i(0)\|}{\varepsilon_i} \right\} \tag{5.7}$$

In order to be able to satisfy the reaching condition we need to establish a connection between the control input u_i in equation (5.1) and the inequality in (5.6). Differentiating the sliding manifold equation with respect to time we obtain

$$\dot{s}_i = \ddot{x}_i + \frac{d}{dt}\left[\nabla_{x_i} J(x)\right]$$

From the vehicle dynamics of the agents in (5.1) we have

$$\ddot{x}_i = M_i^{-1}(x_i)\left[u_i - f_i(x_i, \dot{x}_i)\right]$$

where $M_i(x_i)$ is always invertible from the assumption in (5.2). Substituting this value in the equation of the derivative of s_i and evaluating the value of $s_i^\top \dot{s}_i$ one obtains

$$s_i^\top \dot{s}_i = s_i^\top \left[M_i^{-1}(x_i)u_i - M_i^{-1}(x_i) f_i(x_i, \dot{x}_i) + \frac{d}{dt}\left[\nabla_{x_i} J(x)\right] \right]$$

One issue to note here is that the potential function $J(x)$ is not static. It depends on the relative positions of the individuals. Therefore, uncertainties and disturbances (including those acting on the system dynamics) as well as the agent motions can affect the time derivative of $J(x)$. In order to make the analysis easier we will consider only a certain class of potential functions $J(x)$. In particular, we will consider the potential functions which satisfy the following assumption.

Assumption 5. *The potential function $J(x)$ satisfies*

$$\|\nabla_{x_i} J(x)\| \leq \alpha(x)$$

for all x_i and

$$\left\|\nabla_{x_j}\left[\nabla_{x_i} J(x)\right]\right\| \leq \beta(x)$$

for all x_i and x_j, where $\alpha(x)$ and $\beta(x)$ are known and finite and $\|\cdot\|$ denotes the Euclidean norm (the vector and the induced matrix norms, respectively).

The above assumption is in a sense a smoothness assumption since it requires bounds on the "first" and the "second" derivatives of $J(x)$. It is satisfied by many potential functions and certainly by most of the potential functions considered in Chapter 3 (excluding possibly only the potential functions with unbounded repulsion). For some $J(x)$ it is even possible to find constants $\bar{\alpha}$ and $\bar{\beta}$ such that $\alpha(x) \leq \bar{\alpha}$ and $\beta(x) \leq \bar{\beta}$ for the range of operating conditions. Later we will show an example of such a function.

Initially it may seem as if Assumption 5 is a restrictive assumption since $\alpha(x)$ and $\beta(x)$ must be known. However, note that in order to implement the potential functions based approach one already has to know the potential function $J(x)$ and once it is known finding $\alpha(x)$ and $\beta(x)$ is straightforward. Therefore, Assumption 5 is not a significant restriction. Moreover, the potential functions representing the inter-individual interactions are usually chosen by the system designer and therefore satisfaction of Assumption 5 can be guaranteed a priori. In the case of social foraging, on the other hand, the external resource profile contributes to the potential function $J(x)$. However, this does not bring extra restrictions since any realistic environmental profile has bounded gradients.

Below, we have one more reasonable assumption which states that all the agents are initially at rest.

Assumption 6. *At time $t = 0$ we have $\dot{x}_i(0) = 0$ for all $i = 1,\ldots,N$.*

Using Assumptions 5 and 6 one can establish a bound on $\left\|\frac{d}{dt}\left[\nabla_{x_i} J(x)\right]\right\|$ as

$$\begin{aligned}\left\|\frac{d}{dt}\left[\nabla_{x_i} J(x)\right]\right\| &= \left\|\left[\sum_{j=1}^{N} \nabla_{x_j}\left[\nabla_{x_i} J(x)\right]\right] \dot{x}_j\right\| \\ &= \left\|\left[\sum_{j=1}^{N} \nabla_{x_j}\left[\nabla_{x_i} J(x)\right]\right]\left[s_j - \nabla_{x_j} J(x)\right]\right\| \\ &\leq N\beta(x)\left[\alpha(x(0)) + \alpha(x)\right] \triangleq \bar{J}_i(x), \qquad (5.8)\end{aligned}$$

where the last inequality was established using

$$\|s_j(t)\| \leq \|s_j(0)\| = \|\nabla_{x_j} J(x(0))\| \leq \alpha(x(0)). \qquad (5.9)$$

The second inequality in this equation follows from Assumption 5, whereas the equality in the middle follows from Assumption 6. For now, assume that the first inequality holds; below we will show that it really does hold.

Given the bound in (5.8) we can choose u_i such that $s_i^\top \dot{s}_i < -\varepsilon_i \|s_i\|$ hold for all i and for all $t \geq 0$. In particular, by choosing

$$u_i = -K_i(x)\mathrm{sign}(s_i) + f_i^k(x_i, \dot{x}_i), \tag{5.10}$$

where the sign function is operated elementwise on s_i as $\mathrm{sign}(s_i) = [\mathrm{sign}(s_{i1}), \ldots, \mathrm{sign}(s_{in})]^\top$, we obtain

$$s_i^\top \dot{s}_i < -\|s_i\| \left[\frac{1}{\bar{M}_i} K_i(x) - \frac{1}{\underline{M}_i} \bar{f}_i(x_i, \dot{x}_i) - \bar{J}_i(x) \right]$$

Then, by choosing the gain $K_i(x)$ of the control input as

$$K_i(x) > \bar{M}_i \left(\frac{1}{\underline{M}_i} \bar{f}_i(x_i, \dot{x}_i) + \bar{J}_i(x) + \varepsilon_i \right) \tag{5.11}$$

for some $\varepsilon_i > 0$, one can guarantee that the reaching condition in equation (5.6) is satisfied and that sliding mode occurs in finite time. This equation also guarantees that the first inequality in (5.9) and therefore the bound in (5.8) hold for all $t \geq 0$.

The above discussion constitutes a constructive proof of a result which can be formally summarized as follows.

Theorem 15. *Consider a system of N agents with vehicle dynamics given by* (5.1). *Assume that the artificial potential function $J(x)$ satisfies Assumption 5 and that Assumption 6 holds. Let the controllers for the agents are given by (5.10) with gains as in (5.11). Then, sliding mode occurs in all the surfaces s_i and* (5.3) *is satisfied in a finite time bounded by the bound in (5.7).*

In the controller in (5.10), in addition to the switching sliding mode term, the known part $f_i^k(x_i, \dot{x}_i)$ of the vehicle dynamics was also utilized. If there are not known parts, then this portion of the controller can be set to zero. It is good to also emphasize that the exact value of the mass/inertia matrix $M_i(x_i)$ of the robot is not needed for implementation of the controller in (5.10)-(5.11) and, similar to the case with additive disturbances, its lower and upper bounds are sufficient.

5.2.2 Swarm Tracking

The swarm tracking problem is different from the aggregation, foraging, and formation control problems in the sense that instead of motion dynamics of the form (5.3), we require the agents to move based on

$$\dot{x}_i = -\eta \nabla_{x_i} J(x, x_t) - \lambda \mathrm{sign}(\nabla_{x_i} J(x, x_t)) \tag{5.12}$$

where $\eta > 0$ and $\lambda > 0$ are positive constants (see Equation (3.39)[1] in Chapter 3). Therefore, the n-dimensional sliding manifold for agent i is defined as

$$s_i = \dot{x}_i + \eta \nabla_{x_i} J(x) + \lambda \mathrm{sign}(\nabla_{x_i} J(x, x_t)) = 0, i = 1, \ldots, N \tag{5.13}$$

---

[1] We slightly changed notation here in order not to confuse with the bounds in Assumption 5.

in order to achieve the motion dynamics in (5.12). This results in

$$s_i^\top \dot{s}_i = s_i^\top \Bigg[M_i^{-1}(x_i)u_i - M_i^{-1}(x_i)f_i(x_i,\dot{x}_i) + \eta \frac{d}{dt}\left[\nabla_{x_i} J(x,x_t)\right] + \lambda \frac{d}{dt}\left[\mathrm{sign}(\nabla_{x_i} J(x,x_t))\right] \Bigg] \tag{5.14}$$

However, the term $\frac{d}{dt}\left[\mathrm{sign}(\nabla_{x_i} J(x,x_t))\right]$ is unbounded at the instants at which the gradient $\nabla_{x_i} J(x,x_t)$ switches sign. Therefore, the sliding mode control procedure discussed above is not directly applicable. To avoid this problem we introduce the low pass filters

$$\mu \dot{z}_i = -z_i + \lambda \mathrm{sign}(\nabla_{x_i} J(x,x_t)) \tag{5.15}$$

where μ is a small positive constant. With proper choice of the parameter μ one has

$$z_i \cong [\lambda \mathrm{sign}(\nabla_{x_i} J(x,x_t))]_{eq} \tag{5.16}$$

where the subscript *eq* denotes the *equivalent* (effective or average) value of the discontinuous signal. Therefore, although $\lambda \mathrm{sign}(\nabla_{x_i} J(x,x_t))$ is not differentiable, its approximation z_i is differentiable and can be used in the definition of the sliding manifold.

Using this, the new sliding manifold definition for the swarm tracking case becomes

$$s_i = \dot{x}_i + \eta \nabla_{x_i} J(x) + z_i = 0, i = 1,\ldots,N \tag{5.17}$$

which results in

$$s_i^\top \dot{s}_i = s_i^\top \left[M_i^{-1}(x_i)u_i - M_i^{-1}(x_i)f_i(x_i,\dot{x}_i) + \eta \frac{d}{dt}\left[\nabla_{x_i} J(x,x_t)\right] + \dot{z}_i \right]$$

Moreover, noting that

$$\left\| \frac{\partial}{\partial t}[\lambda \mathrm{sign}(\nabla_{x_i} J(x,x_t))]_{eq} \right\| = \|\dot{z}\| \leq \frac{2\lambda}{\mu} \triangleq \bar{J}_z \tag{5.18}$$

by choosing the control input as in equation (5.10) with gain $K_i(x)$ satisfying

$$K_i(x) > \bar{M}_i \left(\frac{1}{\underline{M}_i} \bar{f}_i(x_i,\dot{x}_i) + \eta \bar{J}_i(x) + \bar{J}_z + \varepsilon_i \right) \tag{5.19}$$

it is guaranteed that in finite time all the agents $i = 1,...,N$, will start moving based on the motion equations

$$\dot{x}_i = -\eta \nabla_{x_i} J(x,x_t) - z_i \cong -\eta \nabla_{x_i} J(x,x_t) - [\lambda \mathrm{sign}(\nabla_{x_i} J(x,x_t))]_{eq} \tag{5.20}$$

and the results on swarm tracking case obtained for swarms of single integrator agents in Chapter 3 will be recovered for swarms of agents with fully actuated dynamics as well.

The model for the motion dynamics of the agents in (5.1) is more general than the single integrator model considered in Chapter 3 and the double integrator model considered in Chapter 4. Moreover, it allows for possible system uncertainties and additive disturbances. Despite all these, due to the robustness properties of the sliding mode control algorithm, satisfaction of Equations (5.3) or (5.20) in finite time is guaranteed. This fact, on the other hand, implies that, provided that the stated assumptions are satisfied, the results derived for the single integrator model in Chapter 3 are recovered also for the swarms composed of agents with dynamics given in (5.1) as well.

We would like to emphasize here that besides providing important conclusions about swarms composed of agents with single integrator dynamics, the results in Chapter 3 serve also as *proof of concept* for swarm behavior and can be reproduced for a swarm composed of agents with other vehicle dynamics (as we do in this chapter). In other words, in engineering swarm applications with agents with particular motion dynamics one can use the results in Chapter 3 and develop corresponding control algorithms taking into account the agent dynamics to achieve the required swarming behavior. The control algorithm based on sliding mode control theory discussed in this section is one such method that could be applied if the agent dynamics are described by (5.1).

The design of the sliding mode surface considered here is similar to conventional sliding mode control problems. However, there is also small difference. In classical sliding mode control problems, the surface $s = 0$ is chosen such that on it the tracking error asymptotically decays to zero. Here, the surfaces $s_i = 0$ are chosen so that the system motion equation obeys certain dynamics. Even though $\dot{x}_i$ can be viewed as the output of the system and s_i as the output error and it can be argued that at $s_i = 0$ the output error becomes zero, there is still a difference since here the $s_i = 0$ surfaces are not constant surfaces and they can dynamically vary as the agents move.

Theorem 15 suggests that the results on swarm stability described in Chapter 3 for swarms composed of agents with single integrator dynamics will be recovered for swarms composed of agents obeying the dynamics in (5.1). However, in real applications usually it is not possible to achieve an ideal sliding mode due to actuator non-idealities, numerical errors, and the system uncertainties (unmodeled dynamics and disturbances) which may lead to the so called *chattering phenomena*. Therefore, in practical implementations it may not be possible to ideally recover all the stability results that could be obtained for the single integrator model. For example, the statement that $\bar{x}$ is stationary for all time, although in theory should hold after sliding mode occurs on all surfaces, may not necessarily hold in practice due to chattering affects and there may be small deviations of the center. Nevertheless, even in practical implementations despite the non-idealities (including the disturbances and unmodeled dynamics) the qualitative results about the overall swarm behavior (such as swarm cohesiveness, bounds on the swarm size, finite time convergence, etc.) will be recovered with only small perturbations.

In the sections below we will revisit the swarm control problems considered in Chapter 3, will show that Assumption 5 is satisfied, will derive the corresponding bounds and controller gains, and present simulation results.

5.3 Potential Functions and Bounds

5.3.1 Aggregation

In order to achieve aggregation we will use potential functions of the form considered in Chapter 3. In particular, we will consider potential functions in the form (3.3) and satisfying all the assumptions stated in Section 3.2. An example potential function which was used in Chapter 3 is the function in equation (3.6) which is given by

$$J(x) = J_{aggregation}(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{a}{2} \|x_i - x_j\|^2 + \frac{bc}{2} \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \right] \quad (5.21)$$

Here we will use the same function in order to be consistent with the results in Chapter 3. The negative gradient of this potential at x_i is given by

$$-\nabla_{x_i} J(x) = \sum_{j=1, j\neq i}^{N} g(x_i - x_j) = -\sum_{j=1, j\neq i}^{N} (x_i - x_j) \left[a - b\exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \right]$$

where $g(x_i - x_j)$ is the attraction/repulsion function in equation (3.7).

Note that this potential function also satisfies Assumption 5 and the bounds $\alpha(x)$ and $\beta(x)$ can be derived as shown below. Considering the norm of the gradient $\|\nabla_{x_i} J(x)\|$ we have

$$\begin{aligned} \|\nabla_{x_i} J(x)\| &\leq a \sum_{j=1, j\neq i}^{N} \|x_i - x_j\| + b \sum_{j=1, j\neq i}^{N} \|x_i - x_j\| \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \\ &\leq N \left[a \max_{1\leq j\leq N} \|x_i - x_j\| + b\sqrt{\frac{c}{2}} \exp\left(-\frac{1}{2} \right) \right] \triangleq \alpha(x) \end{aligned} \quad (5.22)$$

where we used the inequality $\|x_i - x_j\| \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \leq \sqrt{\frac{c}{2}} \exp\left(-\frac{1}{2} \right)$, the proof of which is straightforward. Similarly, considering $\left\|\nabla_{x_j} \left[\nabla_{x_i} J(x)\right]\right\|$ for $j \neq i$ we have

$$\begin{aligned} \left\|\nabla_{x_j} \left[\nabla_{x_i} J(x)\right]\right\| &= \left\| \nabla_{x_j} \left[\sum_{k=1, k\neq i}^{N} (x_i - x_k) \left[a - b\exp\left(-\frac{\|x_i - x_k\|^2}{c} \right) \right] \right] \right\| \\ &\leq a + b \left\| \nabla_{x_j} \left[(x_i - x_j) \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \right] \right\| \\ &\leq a + b \left\| \left(I + \frac{2}{c}(x_i - x_j)(x_i - x_j)^\top \right) \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \right\| \\ &\leq a + b \left(1 + \frac{2}{c} \|x_i - x_j\|^2 \right) \exp\left(-\frac{\|x_i - x_j\|^2}{c} \right) \\ &\leq a + 2b\exp\left(-\frac{1}{2} \right) \end{aligned}$$

When $j = i$, on the other hand, we have

$$\|\nabla_{x_i}[\nabla_{x_i}J(x)]\| \leq N\left[a+2b\exp\left(-\frac{1}{2}\right)\right] \triangleq \beta(x) \tag{5.23}$$

which also constitutes the bound $\beta(x)$. Using the above values of $\alpha(x)$ and $\beta(x)$ one can easily calculate the bound $\bar{J}_i(x)$ in equation (5.8) in Assumption 5. One issue to note, however, is that the above bounds $\alpha(x)$ and $\beta(x)$ are very conservative and result in large $\bar{J}_i(x)$ and the actual bounds can be much smaller. Therefore, in implementation the procedure will probably still perform satisfactorily with much smaller $\bar{J}_i(x)$.

Another issue to mention here is that for the considered potential $J(x)$ in (5.21) we have $\beta(x) = \bar{\beta}$, i.e., $\beta(x)$ is independent of x. Similarly, $\alpha(x)$ depends only on the relative distances $\|x_i - x_j\|$ between the individuals. From the results in Chapter 3 and those in the preceding section we know that for $J(x)$ in (5.21) once on the sliding mode surface the system will be well behaved (i.e., its states will be bounded and the position of all the agents will converge to a small hyperball around its center) and also that the sliding mode surface will be reached in a finite time. This implies that for any given initial position $x(0)$ there is a constant $\bar{\alpha}(x(0))$ such that $\alpha(x) \leq \bar{\alpha}(x(0))$ for all $t \geq 0$ implying that both of the bounds in Assumption 5 can be set as constants for the potential function $J(x)$ in (5.21).

5.3.2 Social Foraging

In this section we will consider the case of social foraging swarms, i.e., swarms moving in an environment modeled as a potential field (resource profile). Once again we consider a swarm the motion of which was determined by a potential function of the form

$$J(x) = J_{foraging}(x) = \sum_{i=1}^{N} \sigma(x_i) + J_{aggregation}(x)$$

where $J_{aggregation}(x)$ is the potential in (5.21) and determines the inter-agent interactions in the swarm, whereas the term $\sigma : \mathbb{R}^n \to \mathbb{R}$ represents a “resource profile” of attractant/repellent substances (e.g., nutrients and/or toxic substances in biology) or simply is a model of the environment. The gradient of the potential at the position x_i of agent i is given by

$$-\nabla_{x_i}J(x) = -\nabla_{x_i}\sigma(x_i) - \sum_{j=1, j\neq i}^{N} (x_i - x_j)\left[a - b\exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)\right]$$

As in Chapter 3, in this model it is assumed that the regions of lower values of $\sigma(\cdot)$ constitute more favorable regions. Then, note that the individuals try to move to more favorable regions along the negative gradient of the profile (due to the first term in the gradient equation), while trying to stay cohesive (due to the second term in the gradient equation).

In order to be able to apply the sliding mode control strategy discussed in this chapter one needs to derive the bounds $\alpha(x)$ and $\beta(x)$ in Assumption 5 depending on the resource profile under consideration. The derivation of these bounds is straightforward and their values for various resource profiles is as presented below.

Plane Resource Profiles

For a plane resource profile of the form

$$\sigma(y) = a_\sigma^\top y + b_\sigma$$

the values of $\alpha(x)$ and $\beta(x)$ in equation (5.8) in Assumption 5 are given by

$$\alpha(x) = \|a_\sigma\| + N\left[a \max_{1\le j\le N} \|x_i - x_j\| + b\sqrt{\frac{c}{2}}\exp\left(-\frac{1}{2}\right)\right]$$

which is the value in equation (5.22) with only $\|a_\sigma\|$ added and the expression of $\beta(x)$ is exactly the same as in equation (5.23).

Quadratic Resource Profiles

For a quadratic resource profile of the form

$$\sigma(y) = \frac{A_\sigma}{2}\|y - c_\sigma\|^2 + b_\sigma$$

the values of $\alpha(x)$ and $\beta(x)$ in equation (5.8) in Assumption 5 are given by

$$\alpha(x) = A_\sigma\|x_i - c_\sigma\| + N\left[a \max_{1\le j\le N} \|x_i - x_j\| + b\sqrt{\frac{c}{2}}\exp\left(-\frac{1}{2}\right)\right]$$

and

$$\beta(x) = |A_\sigma| + N\left[a + 2b\exp\left(-\frac{1}{2}\right)\right]$$

which can easily be derived as was done above.

Multimodal Gaussian Resource Profiles

For a multimodal Gaussian resource profile of the form

$$\sigma(y) = -\sum_{i=1}^{M}\frac{A_{\sigma i}}{2}\exp\left(-\frac{\|y - c_{\sigma i}\|^2}{l_{\sigma i}}\right) + b_\sigma \qquad (5.24)$$

the values of $\alpha(x)$ and $\beta(x)$ can be determined as

$$\alpha(x) = \sum_{i=1}^{M}|A_{\sigma i}|\sqrt{\frac{1}{2l_{\sigma i}}}\exp\left(-\frac{1}{2}\right) + N\left[a \max_{1\le j\le N} \|x_i - x_j\| + b\sqrt{\frac{c}{2}}\exp\left(-\frac{1}{2}\right)\right]$$

and the expression of $\beta(x)$ is once again exactly the same as in equation (5.23). Here we will consider only multimodal Gaussian profiles since the values of the corresponding bounds $\alpha(x)$ and $\beta(x)$ for a single Gaussian can easily be deduced from the above bounds.

5.3.3 Formation Control

In order to achieve the desired formation once more we will use the modified version of the aggregation potential in equation (5.21) such that it has a minimum at the desired formation. In particular, we will use the potential

$$J(x) = J_{formation}(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{a_{ij}}{2} \|x_i - x_j\|^2 + \frac{b_{ij}c_{ij}}{2} \exp\left(-\frac{\|x_i - x_j\|^2}{c_{ij}} \right) \right] \tag{5.25}$$

where the parameters a_{ij}, b_{ij}, and c_{ij} depend on the desired relative distances of the agents or basically formation constraints in the form of

$$\|x_i - x_j\| = d_{ij}, 1 \leq i, j \leq N$$

Note that $J_{formation}(x)$ is very similar to $J_{aggregation}(x)$ and the corresponding bounds $\alpha(x)$ and $\beta(x)$ can be expressed as

$$\alpha(x) = N \left[a_{max} \max_{1 \leq j \leq N} \|x_i - x_j\| + b_{max} \sqrt{\frac{c_{max}}{2}} \exp\left(-\frac{1}{2} \right) \right]$$

and

$$\beta(x) = N \left[a_{max} + 2b_{max} \exp\left(-\frac{1}{2} \right) \right]$$

where $a_{max} = \max_{1 \leq i,j \leq N}\{a_{ij}\}$, $b_{max} = \max_{1 \leq i,j \leq N}\{b_{ij}\}$, and $c_{max} = \max_{1 \leq i,j \leq N}\{c_{ij}\}$.

5.3.4 Swarm Tracking

The potential function for the swarm tracking problem is composed of two parts and has the form

$$J(x, x_t) = W_t \sum_{i=1}^{N} \frac{1}{4} \|x_i - x_t\|^4 + W_f J_{formation}(x) \tag{5.26}$$

where $J_{formation}(x)$ is the formation potential in (5.25) that specifies the geometrical shape to be acquired by the swarm around the target, whereas the first term specifies the requirement by the agents to track the moving target and to surround/enclose it.

The negative gradient of the potential at the position x_i of agent i is given by

$$\begin{aligned} -\nabla_{x_i} J(x, x_t) &= -W_t \|x_i - x_t\|^2 (x_i - x_t) \\ &\quad -W_f \sum_{j=1, j \neq i}^{N} (x_i - x_j) \left[a_{ij} - b_{ij} \exp\left(-\frac{\|x_i - x_j\|^2}{c_{ij}} \right) \right] \end{aligned} \tag{5.27}$$

Then, the bounds $\alpha(x)$ and $\beta(x)$ can be calculated as

$$\alpha(x) = W_t \|x_i - x_t\|^3 + W_f N \left[a_{max} \max_{1 \leq j \leq N} \|x_i - x_j\| + b_{max} \sqrt{\frac{c_{max}}{2}} \exp\left(-\frac{1}{2} \right) \right]$$

and

$$\beta(x) = \frac{W_t}{3}\|x_i - x_t\|^2 + W_f N \left[a_{max} + 2b_{max}\exp\left(-\frac{1}{2}\right)\right]$$

where a_{max}, b_{max}, and c_{max} are as defined above (for the formation control case).

The bounds $\alpha(x)$ and $\beta(x)$ derived above are needed in order to be able to calculate the value (or expression) of $\bar{J}_i(x)$ in equation (5.8), which, on the other hand, is needed in order to be able to calculate the gain $K_i(x)$ in equation (5.11) of the control input in equation (5.10). In other words, once $\bar{J}_i(x)$ is calculated the sliding mode controller developed in the preceding section can be implemented and the results for the single integrator case recovered. In the following section we will provide illustrative numerical simulation examples.

5.4 Simulation Examples

In this section we will provide simulation examples in order to illustrate the effectiveness of the sliding mode control method discussed in this chapter and to gain further insights on the dynamics of the swarm. As usual, for ease of plotting we use only $n = 2$ or $n = 3$; however, qualitatively the results will be the same for higher dimensions. We consider agents (robots) with point mass dynamics with unknown mass and additive sinosoidal disturbances. In other words, we consider the model

$$M_i\ddot{x}_i + f_i(x_i,\dot{x}_i) = u_i, 1 \le i \le N,$$

where M_i is the unknown mass which satisfies $\underline{M}_i \le M_i \le \bar{M}_i$ and the unknown additive uncertainty or disturbance in the system is given by

$$f_i(x_i,\dot{x}_i) = f_i^u(x_i,\dot{x}_i) = \sin(0.2t)\mathbf{1_v}$$

where $\mathbf{1_v}$ is a vector of ones in $\mathbb{R}^n$ (for $n = 2$ or $n = 3$). Since in principle it does not make any difference for the simulations, here we assumed that $f_i^k(x_i,\dot{x}_i) = 0$ (there is no known part in $f_i(x_i,\dot{x}_i)$). Note that $f_i^u(x_i,\dot{x}_i)$ satisfies the boundedness assumption

$$\|f_i^u(x_i,\dot{x}_i)\| = \left\|\sin(0.2t)\mathbf{1_v}\right\| \le \sqrt{n} \triangleq \bar{f}_i$$

Without loss of generality we assume unity mass $M_i = 1$ for all the agents. As controller parameters in the simulations below we choose $\underline{M}_i = 0.5$ and $\bar{M}_i = 1.5$, $\bar{f}_i = \sqrt{n}$ (for the simulations below either $n = 2$ or $n = 3$), and $\varepsilon_i = 1$. Moreover, in order to avoid numerical problems and to speed up the simulations we approximated the signum function with a hyperbolic tangent. In other words, instead of the $\mathrm{sign}(s_i)$ term in the controller we used $\tanh(\gamma s_i)$ with $\gamma = 10$. Besides helping with numerical problems similar to the boundary layer approaches this smooths the control action and reduces unwanted chattering due to the discontinuity in the controller. Figure 5.1 shows the similarity between the $\mathrm{sign}(s_i)$ and the $\tanh(\gamma s_i)$ functions and the effect of the parameter γ. As the value of γ increases the $\tanh(\gamma s_i)$ becomes closer to the $\mathrm{sign}(s_i)$ function. There are rigorous methods to prevent the chattering in the system (such as using a state observer [26] or using a boundary layer); however, they are outside of the scope of this chapter and book.

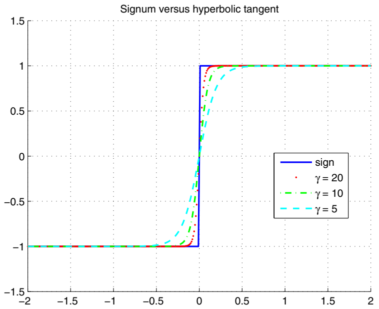
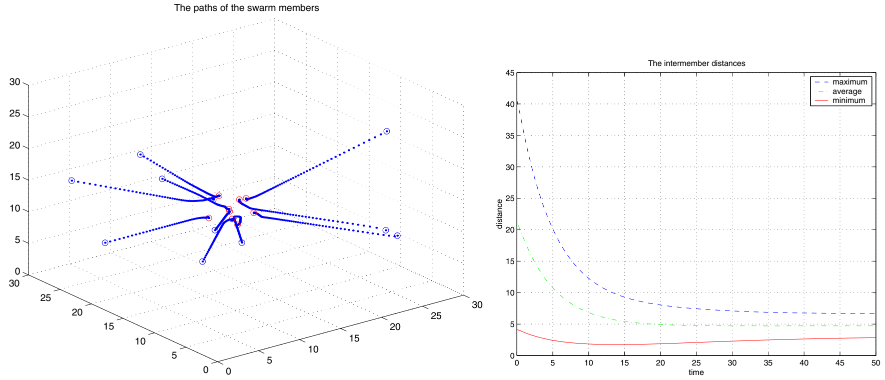


Fig. 5.1. Signum versus hyperbolic tangent.

5.4.1 Aggregation

In the simulations on swarm aggregation we used the parameters $a = 0.01$, $b = 20$, and $c = 1$ for the potential function in (5.21) and in the controller we used the bounds $\bar{J}_i(x) = 10N$, which is much smaller than the bound computed using the above $\alpha(x)$ and $\beta(x)$. Figure 5.2(a) shows in $\mathbb{R}^3$ the paths of the agents in a swarm composed of $N = 10$ individuals. The initial positions of the agents were chosen randomly and their initial velocities were set to zero (as is needed by Assumption 6). The initial positions of the agents are represented with circles and their paths with dots. Their positions after 40 seconds are also represented with circles. It is easily seen that all the agents move toward each other and form a cohesive swarm cluster reproducing the results obtained in Chapter 3 for swarms composed of agents with single integrator dynamics. The maximum, average, and the minimum inter-agent



(a) The paths of the agents in the swarm. Inter-agent distances in the swarm.

Fig. 5.2. Aggregating swarm of agents with fully actuated vehicle dynamics subject to external disturbances and model uncertainty.

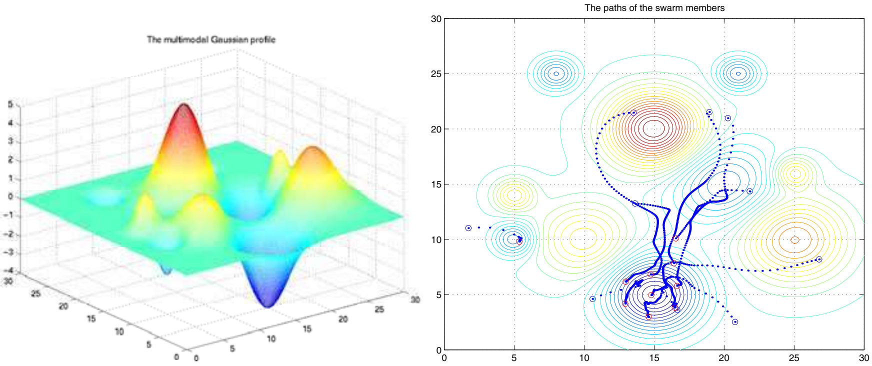
distances during the motion of the swarm are depicted in Figure 5.2(b). As can be seen from the figure, no collisions between agents occur and the final swarm size becomes about 3.8 units.

5.4.2 Social Foraging

For illustrating the case of social foraging we will investigate one case only. In particular, we will consider the case in which the environment potential is a multimodal Gaussian resource profile and for easy comparison we will use the same multimodal Gaussian resource profile we used in Chapter 3. Figure 5.3(a) shows the mentioned profile which is of the form in equation (5.24) with $M = 10$ Gaussians (which are either hills or valleys) and corresponding parameters

$$
\begin{aligned}
A_\sigma &\in \{5,-2,3,2,-2,-4,-2,-2,2,2\} \\
l_\sigma &\in \{10,12.5,12.5,10,2,10,2,2,2,2\} \\
c_\sigma &\in \left\{ \begin{bmatrix} 15 \\ 20 \end{bmatrix}, \begin{bmatrix} 20 \\ 15 \end{bmatrix}, \begin{bmatrix} 25 \\ 10 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 5 \\ 10 \end{bmatrix}, \begin{bmatrix} 15 \\ 5 \end{bmatrix}, \begin{bmatrix} 8 \\ 25 \end{bmatrix}, \begin{bmatrix} 21 \\ 25 \end{bmatrix}, \begin{bmatrix} 25 \\ 16 \end{bmatrix}, \begin{bmatrix} 5 \\ 14 \end{bmatrix} \right\}
\end{aligned}
$$

The hills in the profile represent unfavorable regions, while the valleys represent favorable ones. We would expect the swarm to avoid hills and to move towards the valleys (provided the parameters are appropriately chosen). For the attraction/repulsion parameters a, b, and c we use the same parameters as for the swarming case above.



(a) The environment resource profile.

(b) Paths of the agents moving.

Fig. 5.3. Motion swarm of agents with fully actuated vehicle dynamics in multi-modal gaussian environmental resource profile.

Figure 5.3(b) shows a simulation run for the model with the above parameters and randomly chosen initial positions. The initial velocities are once again set to zero. It is easily seen from the figure that the expectations are satisfied and the agents move towards valleys while avoiding hills. One can easily see the similarity of the result with the results obtained in Chapter 3.

5.4.3 Formation Control

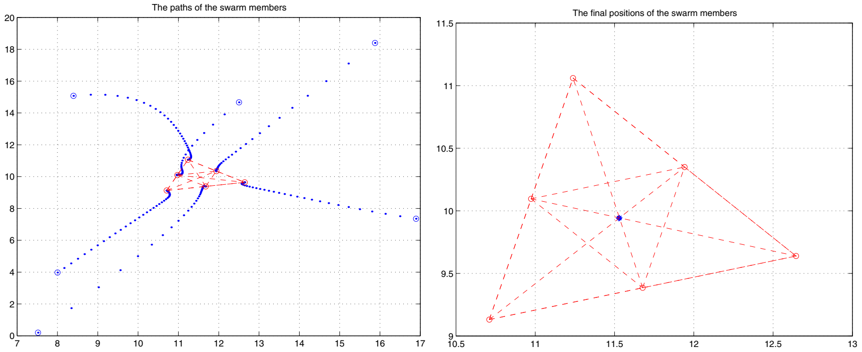
In this section, we consider the application of the procedure to the formation control (formation acquisition/stabilization) problem. In particular, we consider $N = 6$ agents which are required to form an equilateral triangle formation (with any orientation) in $\mathbb{R}^2$ with three of the agents at the corners of the triangle and three of the agents at the middle points of the vertices.

As parameters of the potential function $J_{formation}(x)$ in equation (5.25) we used $b_{ij} = 20, c_{ij} = 0.2$ for all i, j, and calculated a_{ij} using

$$a_{ij} = b_{ij} \exp\left(-\frac{d_{ij}^2}{c_{ij}}\right)$$

which, given the desired inter-agent distances $d_{ij} \in \{1, \sqrt{3}, 2\}$ based on the desired relative positions of the agents resulted in $a_{ij} \in \{0.1348, 6.1180 \times 10^{-6}, 4.1223 \times 10^{-8}\}$. We would like to remind here that the above type potentials may have local minima leading to only local results. If global convergence to the desired formation is desired then potentials with unique minimum (at the desired formation) should be chosen.

Figure 5.4 illustrates the application of the procedure to the formation control problem. The system is once more initialized at rest with random initial positions. Figure 5.4(a) shows the paths of the agents for 30 seconds, whereas Figure 5.4(b) depicts the positions of the swarm members after 30 seconds. As expected, as the time progresses the agents move to their required inter-individual distances and quickly create the desired formation. The centroid $\bar{x}$ of the agents is denoted by a star. During the transient, before occurrence of sliding mode, it is possible for the centroid to move. However, once all the agents reach their prospective sliding manifolds it is expected that the centroid will remain stationary after that point. Close



(a) Paths of the agents. (b) Final positions of agents.

Fig. 5.4. $N = 6$ acquiring an equilateral triangle formation.

examination of the centroid position (not shown here) shows that this expectation is satisfied and $\bar{x}$ is stationary. Note that even though we considered an example of a formation stabilization, as mentioned before, the procedure can easily be applied to the case in which the formation has to move (as a whole entity) and track some desired trajectory.

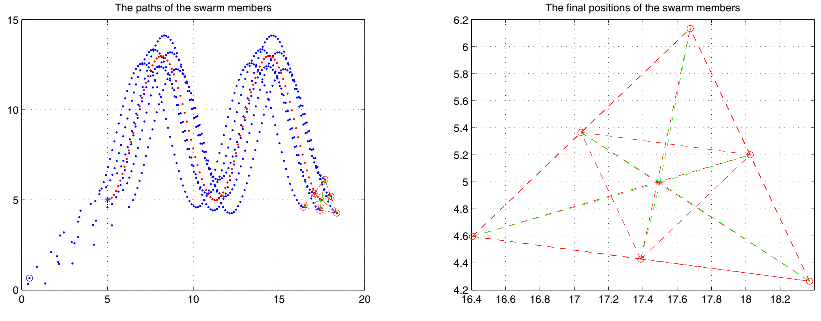
5.4.4 Swarm Tracking

Let us consider a target which moves based on the dynamics

$$\begin{aligned}\dot{x}_{t_1} &= 0.25\\ \dot{x}_{t_2} &= \sin(0.25t)\end{aligned}$$

For the controller parameters, we use $\eta = 0.1, \lambda = 2.0$, $\overline{f}_i = \sqrt{2}$, and $\varepsilon_i = 1$. For the filters, we chose $\mu = 0.02$. The parameters specifying the relative importance of the formation and tracking constraints are chosen as $W_t = 0.5$ and $W_f = 1 - W_t = 0.5$. As parameters of the potential function $J_{formation}(x)$ we used $b_{ij} = 20, c_{ij} = 20$ for all i, j, and calculated a_{ij} as in the formation control case in the preceding section.

Figure 5.5(a) shows the paths of the target and the pursuing agents. The target is initially located at point $[5,5]$ in the space while the agent positions are initialized randomly within $[0,2.5] \times [0,2.5]$. As one can see, the agents quickly catch-up with the target and enclose/surround it. They also form the desired geometrical shape around the target and continue their motion keeping in the formation as seen in Figure 5.5(b). Although it is not always guaranteed that the agents will form the desired formation (due to the local minima problem), they always enclose the target as expected.



(a) Paths of target and agents. (b) Final positions of agents and target.

Fig. 5.5. $N = 6$ acquiring an equilateral triangle formation while tracking and enclosing a moving target.

5.5 Further Issues

5.5.1 Extensions and Generalizations

In this chapter, we discussed a control algorithm based on the sliding mode control technique. Using this approach, a swarm consisting of agents with fully actuated dynamics will, depending on the problem, aggregate and form a cohesive group, cohesively forage and follow a resource profile, acquire a desired geometrical formation or track a moving target/trajectory even in the presence of model uncertainties and additive disturbances in the agent dynamics. In this aspect we showed that under the developed control strategy the results in Chapter 3 developed for swarms composed of agents with single integrator dynamics will be recovered also for agents with fully actuated agent dynamics with model uncertainties and disturbances. However, in contrast to the results in Chapter 4, in this chapter we did not consider measuring errors such as sensor errors for measuring inter-agent distances and/or noise on the resource profile. The results can easily be extended in that direction by making use of the robustness properties of the sliding mode control technique.

5.5.2 For Further Reading

This chapter is based mostly on the results in [81] and some on those in [259]. Sliding mode control for following gradients of potential fields, robot navigation and obstacle avoidance was first used in [109, 111, 112, 246] on which the development in this chapter is also based. Note that agents with point mass or fully actuated dynamics (but without uncertainties) are being considered in the literature (see for example [13, 145, 187, 265, 266]). Using a low pass filter to extract the equivalent value of a switching signal is very similar to the approach of sliding mode observers based on the equivalent control method [61, 62, 115]. Valuable tutorials on sliding mode control can be found in [244, 262]. Similarly, the books [245, 247] are good general references on sliding mode control.

6

Swarms of Non-holonomic Unicycle Agents with Model Uncertainty

6.1 Non-holonomic Unicycle Agent Model with Uncertainty

In this chapter, we consider a swarm composed of N agents whose dynamics have velocity constraints and evolve based on the equations

$$\begin{aligned} \dot{p}_{xi} &= v_i \cos(\theta_i), \\ \dot{p}_{yi} &= v_i \sin(\theta_i), \\ \dot{\theta}_i &= w_i, \\ \dot{v}_i &= \frac{1}{m_i}\left[u_{i1} + f_{v_i}\right], \\ \dot{w}_i &= \frac{1}{I_i}\left[u_{i2} + f_{w_i}\right], \end{aligned} \tag{6.1}$$

where $x_i(t) = [p_{xi}(t), p_{yi}(t)]^\top$ denotes the position of agent i at time instant t in cartesian coordinates, θ_i is the steering angle, v_i is the linear speed, and w_i is the angular speed of agent i. The quantities m_i and I_i are positive constants and represent the mass and the moment of inertia of agent i, respectively. The control inputs for agent i are the force input $u_{i1} = F_i$ and the torque input $u_{i2} = \tau_i$. The functions f_{v_i} and f_{w_i} represent unknown additive disturbances for agent i and are assumed to be bounded such that

$$|f_{v_i}| < f_{v_i}^+ \quad \text{and} \quad |f_{w_i}| < f_{w_i}^+$$

for known bounds $f_{v_i}^+$ and $f_{w_i}^+$. In addition to the unknown additive system disturbances it is assumed that the exact values of the mass m_i and the inertia I_i for agent i are unknown. However, they are assumed to be upper and lower bounded in the form

$$0 < \underline{M}_i < m_i < \overline{M}_i \quad \text{and} \quad 0 < \underline{I}_i < I_i < \overline{I}_i$$

where the bounds $\underline{M}_i, \overline{M}_i, \underline{I}_i$, and $\overline{I}_i$ are known. The model in Equation (6.1), which is also graphically illustrated in Figure 6.1, is sometimes called the *unicycle model*. It has non-holonomic constraints since the agents cannot move in the direction of the main axis connecting the two actuated/driving wheels. This can be mathematically represented with the (non-holonomic, velocity) constraint

$$\dot{p}_{xi}\sin(\theta_i) - \dot{p}_{yi}\cos(\theta_i) = 0$$

which basically states that the inner product of the agent velocity vector and the vector along the shaft/axel is always zero meaning that the motion of the agent is always perpendicular to the shaft axis. Many robots in the research laboratories around the world (for example differentially driven robots) have dynamics which obey the model in Equation (6.1).

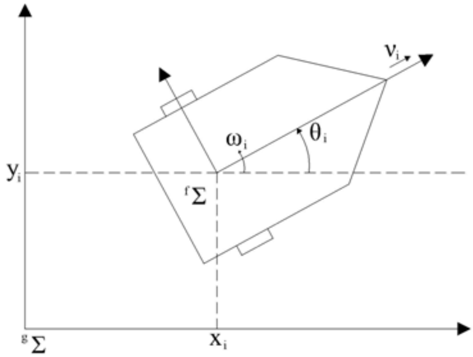


Fig. 6.1. Illustration of an agent with the non-holonomic unicycle dynamics.

In the next section we will discuss a potential functions and sliding mode control based approach for controller development for the model in (6.1). The approach will be similar to the approach in Chapter 5.

6.2 Controller Development

6.2.1 Aggregation, Foraging, and Formation Control

We know from Chapter 3 that if the agents are forced to move based on the equation (see Equations (3.1) and (3.2))

$$\dot{x}_i = -\nabla_{x_i} J(x) \tag{6.2}$$

where $x^\top = [x_1^\top, x_2^\top, \ldots, x_N^\top] \in \mathbb{R}^{Nn}$ denotes the lumped vector of concatenated positions $x_i \in \mathbb{R}^n$ of all agents and $J : \mathbb{R}^{Nn} \to \mathbb{R}$ is an appropriately defined potential function which represents the inter-agent interactions and/or the effects of the environment on the behavior of the agents, then the swarm will behave in a predictable manner. For the case in this chapter since $x_i(t) = [p_{xi}(t), p_{yi}(t)]^\top \in \mathbb{R}^2$ we have $n = 2$ or basically the motion of the agents is in a 2-dimensional space, although the fact that the internal dynamics of the agents in Equation (6.1) belong to $\mathbb{R}^5$. Therefore, similar to the case in Chapter 5, if we can force the agents with the dynamics in (6.1) to move based on (6.2) and choose the potential functions appropriately, then the results in Chapter 3 will be recovered for swarms composed of agents with unicycle vehicle dynamics with uncertainties obeying (6.1). This is the approach we will take in this chapter as well. However, because of some technical reasons we will develop the controller for the case

$$\|\nabla_{x_i} J(x)\| \geq \varepsilon \tag{6.3}$$

for some predefined small $\varepsilon > 0$. The consequence of this fact is that for the case here the results in Chapter 3 for swarms composed of agents with single-integrator dynamics will be recovered with some error/perturbation, the amount of which depends on the value of ε. Nevertheless, the overall qualitative behavior of the swarm here (i.e., swarms consisting of agents with the non-holonomic unicycle agent dynamics with uncertainties obeying the motion model in (6.1) under the condition in (6.3) and the controller discussed in the sections below) will be similar to the behavior of swarms consisting of agents with single-integrator dynamics discussed in Chapter 3. Note also that by choosing ε very small the error can be made very small as well. However, this will be achieved with the price that the bounds utilized in the sliding mode controller will become larger. The reader should note also that it is also possible to introduce a boundary layer and a corresponding controller with appropriate analysis.

As mentioned in Chapter 5, sliding mode control is a widely used technique in various application areas, mainly because of its suppressive and robust characteristics against uncertainties and disturbances in the system dynamics. In sliding mode control, a switching controller with high enough gain is applied to suppress the effects of modeling uncertainties and disturbances, and the agent dynamics are forced to move along a stabilizing manifold, which is also called a *sliding manifold*. The value of the gain is computed using the known bounds on the uncertainties and disturbances.

Below we design a sliding mode control scheme to force the motion of each individual agent along the negative gradient of the potential function $J(x)$, i.e. forcing each agent to obey Equation (6.2) where depending on the context (which can be aggregation, foraging, or formation control in this section) $J(x)$ is an appropriately defined artificial potential function. Let

$$-\nabla_{x_i} J(x) = \begin{bmatrix} -J_{p_{xi}}(x) \\ -J_{p_{yi}}(x) \end{bmatrix}$$

denote the gradient of the potential at x_i. In order to achieve satisfaction of (6.2) there is a need for the equality

$$-\nabla_{x_i} J(x) = \begin{bmatrix} -J_{p_{xi}}(x) \\ -J_{p_{yi}}(x) \end{bmatrix} = \begin{bmatrix} v_i \cos(\theta_i) \\ v_i \sin(\theta_i) \end{bmatrix} \tag{6.4}$$

to be satisfied. In other words, the translational speed v_i and the steering angle θ_i have to obey

$$v_i = \|\nabla_{x_i} J(x)\|, \;\; \theta_i = \angle\left(\left[-J_{p_{xi}}(x), -J_{p_{yi}}(x)\right]^\top\right) \tag{6.5}$$

where for an arbitrary vector $[x,y]^\top \in \mathbb{R}^2$ the expression $\angle\left([x,y]^\top\right)$ denotes the counter-clockwise angle form the cartesian coordinate x-axis to the vector $[x,y]^\top$.

Note that since the inputs in the agent model (6.1) are u_{i1} and u_{i2} implying that the above v_i and θ_i in (6.5) cannot be applied directly, the terms

$$v_{id} \triangleq \|\nabla_{x_i} J(x)\|, \quad \theta_{id} \triangleq \angle\left(\left[-J_{p_{xi}}(x), -J_{p_{yi}}(x)\right]^\top\right) \tag{6.6}$$

need to be considered as desired set-point values for v_i and θ_i, respectively.

In order to unify the controller development for all the cases/contexts considered in this section (which are aggregation, foraging, and formation control) and to make the development of the sliding mode controller possible we have the following assumption.

Assumption 7. *For all i there exist known and bounded functions $\alpha_i(x) > 0$ and $\beta_i(x) > 0$ such that*

$$|\dot{v}_{id}| = \left\|\frac{d}{dt}\nabla_{x_i} J(x)\right\| \leq \alpha_i(x)$$

and

$$|\ddot{\theta}_{id}| = \left\|\frac{d^2}{dt^2}\tan^{-1}\left(\frac{J_{p_{yi}}(x)}{J_{p_{xi}}(x)}\right)\right\| \leq \beta_i(x)$$

are satisfied.

The properties of the bounds $\alpha_i(x)$ and $\beta_i(x)$ in Assumption 7 depend on the properties of the potential function, which is chosen by the designer, as well as the translational speeds of the agents. In other words, the procedure requires that the agents know the linear speeds of the other agents in the swarm. Moreover, in order for the procedure discussed below to be implementable the artificial potential function needs to be chosen such that Assumption 7 is satisfied. We will derive such bounds for the potential functions considered in the following sections.

From the discussion above, and with the definition of the desired translational speed v_{id} and desired steering angle θ_{id} in (6.6), the objective is to design the control inputs u_{i1} and u_{i2} in (6.1) to force the motion of the agents so that the differences $(v_i - v_{id})$ and $(\theta_i - \theta_{id})$ converge to zero. With this objective in mind let us step back and define the sliding surfaces for the translational speed v_i and for the steering angle θ_i, respectively, as

$$s_{v_i} = v_i - v_{id} \tag{6.7}$$

$$s_{\theta_i} = c_{\theta_i}(\dot{\theta}_i - \dot{\theta}_{id}) + (\theta_i - \theta_{id}), \tag{6.8}$$

where $c_{\theta_i} > 0$ is a positive constant. With these definitions, the objective becomes to design the control inputs u_{i1} and u_{i2} so that $s_{v_i} \to 0$ and $s_{\theta_i} \to 0$, since if this is achieved we will have $v_i \to v_{id}$ and $\theta_i \to \theta_{id}$. Note here that the existence of the additional term $c_{\theta_i}(\dot{\theta}_i - \dot{\theta}_{id})$ in (6.8) is because of the double integrator relationship between θ_i and the applicable input u_{i2} as opposed to the single integrator relationship between v_i and u_{i1}.

In order to be able to compute the value of s_{θ_i} in Equation (6.8) one needs the time derivative of θ_{id}. Note that θ_{id} is in fact the inverse tangent of $\frac{J_{p_{yi}}}{J_{p_{xi}}}$. Taking its derivative with respect to time and with simple manipulation one obtains

$$\dot{\theta}_{id} = \frac{\frac{d}{dt}\left(\frac{J_{p_{yi}}}{J_{p_{xi}}}\right)}{1+\left(\frac{J_{p_{yi}}}{J_{p_{xi}}}\right)^2} = \frac{\frac{d}{dt}\left(J_{p_{yi}}\right)\cdot J_{p_{xi}} - \frac{d}{dt}\left(J_{p_{xi}}\right)\cdot J_{p_{yi}}}{\left(J_{p_{xi}}\right)^2+\left(J_{p_{yi}}\right)^2} \tag{6.9}$$

and using this one can calculate the value of $\dot{\theta}_{id}$ for a given potential $J(x)$. One issue to note here is that as $\|\nabla_{x_i}J(x)\| \to 0$ the value of $\dot{\theta}_{id}$ becomes unbounded. This, on the other hand, leads to unbounded $\beta_i(x)$. However, for the case in which (6.3) is satisfied $\dot{\theta}_{id}$ is bounded and this is the case for which we develop the sliding mode controller. Still, for the time intervals in which $\|\nabla_{x_i}J(x)\| < \varepsilon$ holds, to preserve bounded $\dot{\theta}_{id}$, which is needed for calculating s_{θ_i} in (6.8), we add a small positive constant $\bar{\varepsilon}$ to the denominator of the above equation and calculate $\dot{\theta}_{id}$ as

$$\dot{\theta}_{id} = \frac{\frac{d}{dt}\left(J_{p_{yi}}\right)\cdot J_{p_{xi}} - \frac{d}{dt}\left(J_{p_{xi}}\right)\cdot J_{p_{yi}}}{\left(J_{p_{xi}}\right)^2+\left(J_{p_{yi}}\right)^2+\bar{\varepsilon}} \tag{6.10}$$

This avoids unboundedness for the price of a small deviation from the sliding surface s_{θ_i} resulting in a small boundary layer around it.

As mentioned also in the preceding chapter, it is well known from the sliding mode control theory that if we have the reaching conditions

$$s_{v_i}\dot{s}_{v_i} \le -\varepsilon_{i1}|s_{v_i}| \tag{6.11}$$

$$s_{\theta_i}\dot{s}_{\theta_i} \le -\varepsilon_{i2}|s_{\theta_i}| \tag{6.12}$$

satisfied for some constants $\varepsilon_{i1} > 0$ and $\varepsilon_{i2} > 0$, then $s_{v_i} = 0$ and $s_{\theta_i} = 0$ will be achieved in finite time. In fact, provided that Equations (6.11) and (6.12) are satisfied and using the comparison principle one can show that $s_{v_i} = 0$ and $s_{\theta_i} = 0$ are achieved in finite time bounded by

$$t_{v_i} \le \frac{|s_{v_i}(0)|}{\varepsilon_{i1}} \quad \text{and} \quad t_{\theta_i} \le \frac{|s_{\theta_i}(0)|}{\varepsilon_{i2}}$$

respectively. Then, from (6.7) one can see that $v_i = v_{id}$ for $t \ge t_{v_i}$ and from (6.8) we have for $t \ge t_{\theta_i}$

$$\Big(\theta_i(t) - \theta_{id}(t)\Big) = \Big(\theta_i(t_{\theta_i}) - \theta_{id}(t_{\theta_i})\Big)e^{\frac{-(t-t_{\theta_i})}{c_{\theta_i}}} \tag{6.13}$$

implying exponential convergence of $\theta_i(t)$ to $\theta_{id}(t)$. The speed of convergence depends on the slope $-\frac{1}{c_{\theta_i}}$ of the sliding line. From above, the bound on the time for the occurrence of the sliding mode on all surfaces can be found as

$$\bar{t}_{sm} = \max_{i \in \{1,\ldots,N\}} \left\{ \max\{t_{v_i}, t_{\theta_i}\} \right\} \tag{6.14}$$

In order to achieve satisfaction of Equation (6.11) we choose the first control input u_{i1} as

$$u_{i1} = -K_{i1}(x)\mathrm{sign}(s_{v_i}) \tag{6.15}$$

so that the time derivative of s_{v_i} becomes

$$\dot{s}_{v_i} = -\frac{K_{i1}(x)}{m_i}\mathrm{sign}(s_{v_i}) + \frac{1}{m_i} f_{v_i} - \dot{v}_{id}$$

and we have

$$\begin{aligned} s_{v_i}\dot{s}_{v_i} &= s_{v_i}\left(-\frac{K_{i1}(x)}{m_i}\mathrm{sign}(s_{v_i}) + \frac{1}{m_i} f_{v_i} - \dot{v}_{id}\right) = -\frac{K_{i1}(x)}{m_i}|s_{v_i}| + \frac{1}{m_i} f_{v_i} s_{v_i} - \dot{v}_{id} s_{v_i} \\ &\leq -\left(\frac{K_{i1}(x)}{m_i} - \frac{1}{m_i}|f_{v_i}| - |\dot{v}_{id}|\right)|s_{v_i}| \end{aligned} \tag{6.16}$$

Then by choosing the gain $K_{i1}(x)$ such that the inequality

$$K_{i1}(x) \geq \frac{\overline{M}_i}{\underline{M}_i} f_{v_i}^{+} + \overline{M}_i \alpha_i(x) + \overline{M}_i \varepsilon_{i1} \tag{6.17}$$

is satisfied, where $\alpha_i(x)$ is the bound in Assumption 7 such that $|\dot{v}_{id}| \leq \alpha_i(x)$, one guarantees that the inequality in (6.11) holds and sliding mode occurs (i.e., $s_{v_i} = 0$ is achieved) in finite time.

Similarly, for the second sliding surface choosing the control input as

$$u_{i2} = -K_{i2}(x)\mathrm{sign}(s_{\theta_i}) \tag{6.18}$$

the time derivative of s_{θ_i} becomes

$$\dot{s}_{\theta_i} = -\frac{K_{i2}(x)c_{\theta_i}}{I_i}\mathrm{sign}(s_{\theta_i}) + \frac{c_{\theta_i}}{I_i} f_{w_i} - c_{\theta_i}\ddot{\theta}_{id} + w_i - \dot{\theta}_{id} \tag{6.19}$$

and we have

$$\begin{aligned} s_{\theta_i}\dot{s}_{\theta_i} &= s_{\theta_i}\left(-\frac{K_{i2}(x)c_{\theta_i}}{I_i}\mathrm{sign}(s_{\theta_i}) + \frac{c_{\theta_i}}{I_i} f_{w_i} - c_{\theta_i}\ddot{\theta}_{id} + w_i - \dot{\theta}_{id}\right) \\ &\leq -\left(\frac{K_{i2}(x)c_{\theta_i}}{I_i} - \frac{c_{\theta_i}}{I_i}|f_{w_i}| - c_{\theta_i}|\ddot{\theta}_{id}| - |w_i| - |\dot{\theta}_{id}|\right)|s_{\theta_i}| \end{aligned} \tag{6.20}$$

Then, by choosing the gain $K_{i2}(x)$ such that the inequality

$$K_{i2}(x) \geq \frac{\overline{I}_i}{\underline{I}_i} f_{w_i}^{+} + \overline{I}_i \beta_i(x) + \frac{\overline{I}_i}{c_{\theta_i}}|w_i| + \frac{\overline{I}_i}{c_{\theta_i}}|\dot{\theta}_{id}| + \frac{\overline{I}_i}{c_{\theta_i}}\varepsilon_{i2} \tag{6.21}$$

is satisfied, where $\beta_i(x)$ is the bound in Assumption 7 such that $|\ddot{\theta}_{id}| \leq \beta_i(x)$, one can guarantee that the inequality in (6.12) holds and the second sliding surface $s_{\theta_i} = 0$ in (6.8) will as well be reached in finite time.

The results in the above discussion can be formally summarized as follows.

Theorem 16. *Consider a system of N agents with vehicle dynamics given by* (6.1). *Assume that* (6.3) *holds and Assumption 7 is satisfied. Let the controllers u_{i1} and u_{i2} for the agents be given by* (6.15) *and* (6.18) *with gains satisfying* (6.17) *and* (6.21), *respectively. Then, sliding mode occurs in all surfaces s_{v_i} and s_{θ_i} in a finite time bounded by the bound in* (6.14).

One issue to emphasize here is that at the instance of occurrence of sliding mode $v_i = v_{id}$ is reached but it is not necessarily the case that $\theta_i = \theta_{id}$. In fact, as was mentioned above, after occurrence of sliding mode we have $\theta_i \to \theta_{id}$ exponentially fast (see Equation (6.13)) and the speed of convergence depends on the slope of the sliding surface $-\frac{1}{c_{\theta_i}}$. Therefore, one needs to choose c_{θ_i} as small as possible in order to achieve faster convergence. Note also that decreasing the parameter c_{θ_i} will require increasing the controller gain K_{i2}.

Once sliding mode occurs on all surfaces Equation (6.2) is satisfied exponentially fast, and from the results in Chapter 3 we know that the desired behavior will be exhibited by the swarm composed of agents with non-holonomic unicycle dynamics with uncertainties (6.1). However, note that this holds only for the case in which $\|\nabla_{x_i} J(x)\| \geq \varepsilon$ is satisfied. For the case in which $\|\nabla_{x_i} J(x)\| < \varepsilon$, to overcome unboundedness we will use the expression in (6.10) for the derivative $\dot{\theta}_{id}$ instead of the actual value in (6.9) resulting in a small error. Note that the effect of this on the cases for aggregation and foraging is only a small increase in the swarm size. For the case of formation control it leads to small formation error. In other words, the results in Chapter 3 obtained for swarms composed of agents with single integrator dynamics are recovered with small error for swarms composed of agents obeying (6.1).

6.2.2 Swarm Tracking

In this section we consider controller development for the swarm tracking problem. Recall that the swarm tracking problem is concerned with swarm of agents which are required to capture a moving target, enclose it and form a predefined geometrical formation around it. Let us assume that the target trajectory is specified by $\{x_t, \dot{x}_t, \ddot{x}_t\}$. We have the following assumption about the target dynamics.

Assumption 8. *The velocity of the target is known and bounded such that $\|\dot{x}_t\| \leq \gamma_{tv}$ for a known bound $\gamma_{tv} > 0$. Moreover, the unknown acceleration of the target is also bounded such that $\|\ddot{x}_t\| \leq \gamma_{ta}$ for a known bound $\gamma_{ta} > 0$.*

Recall from the previous chapters that in contrast to the motion in (6.2), which is used for aggregation, foraging, and formation control, for the swarm tracking problem we use agent dynamics of the form

$$\dot{x}_i = -\eta \nabla_{x_i} J(x, x_t) - \lambda \text{sign}(\nabla_{x_i} J(x, x_t)) \tag{6.22}$$

where $\eta > 0$ and $\lambda > 0$ are positive constants and $J(x, x_t)$ is an artificial potential function which specifies the tracking and formation constraints. Similar to the case of aggregation, foraging, and formation control considered in the preceding section, here we develop the controller for the case in which

$$\|\nabla_{x_i} J(x,x_t)\| \geq \varepsilon \tag{6.23}$$

is satisfied. In order to achieve the tracking and formation control objectives λ needs to be chosen such that $\lambda > \gamma_{tv}$ is satisfied, i.e., the value of λ must be larger than the upper bound on the target velocity. However, there is one problem in using the dynamics in Equation (6.22) which is the fact that the time derivative of the $\mathrm{sign}(\nabla_{x_i} J(x,x_t))$ function is unbounded at the instances when $\nabla_{x_i} J(x,x_t)$ switches sign. The time derivative of the right hand side of (6.22) is needed for the development of the sliding mode controller. In particular, it is needed for calculating the derivative of the desired steering angle $\dot{\theta}_{id}$ as well as the bounds $\alpha_i(x)$ and $\beta_i(x)$ which are used for calculating the gains of the sliding mode controller (see for example Equations (6.17) and (6.21) in the preceding section). In order to overcome a similar problem a low-pass filter was utilized in Chapter 5. In contrast, here a different strategy is taken and a continuous and differentiable approximation of the sign function is used. In particular, the motion dynamics of agent i are required to satisfy

$$\dot{x}_i = -\eta \nabla_{x_i} J(x,x_t) - \lambda h(\nabla_{x_i} J(x,x_t)) \tag{6.24}$$

instead of the dynamics in (6.22). Here the function $h : \mathbb{R}^2 \to \mathbb{R}^2$ is the elementwise version of the scalar function which for a scalar $y \in \mathbb{R}$ is defined as

$$h(y) = \begin{cases} -1, & y < -\varepsilon \\ \sin\left(\frac{\pi y}{2\varepsilon}\right), & |y| \leq \varepsilon \\ 1, & y > \varepsilon \end{cases} \tag{6.25}$$

where $\varepsilon > 0$ is a small constant. Figure 6.2 depicts an instance of the function for $\varepsilon = 1$. Note that this function overcomes the problems of discontinuity and unboundedness of the derivative of the signum function since both it and its derivative are continuous and bounded. In the mean time, it introduces a boundary layer and has the same behavior as the sign function outside the interval $[-\varepsilon, \varepsilon]$. This, on the other hand, implies that, when the components of the gradient $\nabla_{x_i} J(x,x_t)$ are outside

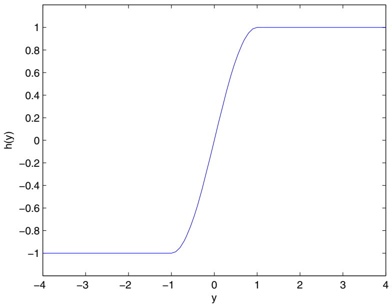


Fig. 6.2. Function $h(.)$ with $\varepsilon = 1$.

the interval $[-\varepsilon,\varepsilon]$, equations (6.22) and (6.24) are equivalent. Then, from the results in Chapter 3 we know that the potential $J(x,x_t)$ is decreasing. Note that for boundary layer here we utilize the same ε as in Equation (6.23) which implicitly specifies the bound on the allowed tracking error. Therefore, as long as the gradient is larger than the allowed $\varepsilon > 0$, decrease in the potential is guaranteed. Moreover, when the agents continue to move based on (6.24) and the components of the gradient $\nabla_{x_i}J(x,x_t)$ are within the interval $[-\varepsilon,\varepsilon]$ the time derivative $\dot{J}$ of the potential function becomes

$$\dot{J} \leq -\eta\|\nabla_{x_i}J(x,x_t)\|_2^2 - \lambda\nabla_{x_i}J^\top(x,x_t)\sin\left(\frac{\pi\nabla_{x_i}J(x,x_t)}{2\varepsilon}\right) + \gamma_{tv}\|\nabla_{x_i}J(x,x_t)\|_1.$$

The second term on the right hand side of the equation always satisfies

$$\lambda\nabla_{x_i}J^\top(x,x_t)\sin\left(\frac{\pi\nabla_{x_i}J(x,x_t)}{2\varepsilon}\right) \geq 0$$

which means that the potential function continues further to decrease within some region inside the interval $[-\varepsilon,\varepsilon]$.

Similar to the preceding section one needs to design the control inputs of the agents such that (6.24) is satisfied. In other words, the objective is to force the agents to move according to equation (6.24). Therefore,

$$-\eta\nabla_{x_i}J(x,x_t) - \lambda h(\nabla_{x_i}J(x,x_t)) = \begin{bmatrix} v_i\cos(\theta_i) \\ v_i\sin(\theta_i) \end{bmatrix} \tag{6.26}$$

needs to be satisfied. Defining Z_i as

$$Z_i \triangleq \eta\nabla_{x_i}J(x,x_t) + \lambda h(\nabla_{x_i}J(x,x_t)) \triangleq \begin{bmatrix} Z_{p_{xi}} \\ Z_{p_{yi}} \end{bmatrix} \tag{6.27}$$

analogous to the case in the preceding section, the desired linear speed and steering angle for agent i can be specified respectively as

$$v_{id} \triangleq \|Z_i\|,\ \theta_{id} \triangleq \angle\left(\left[-Z_{p_{xi}}, -Z_{p_{yi}}\right]^\top\right) \tag{6.28}$$

Here we have an assumption which is analogous to Assumption 7.

Assumption 9. *For all i there exist known and bounded functions $\alpha_i(x) > 0$ and $\beta_i(x) > 0$ such that*

$$|\dot{v}_{id}| = \left\|\dot{Z}_i\right\| = \left\|\eta\frac{d}{dt}\nabla_{x_i}J(x,x_t) + \lambda\frac{d}{dt}h(\nabla_{x_i}J(x,x_t))\right\| \leq \alpha_i(x)$$

and

$$|\ddot{\theta}_{id}| = \left\|\frac{d^2}{dt^2}\tan^{-1}\left(\frac{Z_{p_{yi}}(x,x_t)}{Z_{p_{xi}}(x,x_t)}\right)\right\| \leq \beta_i(x)$$

are satisfied.

Then, following the same steps as in the preceding section one can show that by choosing the sliding surfaces s_{v_i} and s_{θ_i} as in Equations (6.7) and (6.8) with the new definitions of the desired translational speed v_{id} and desired steering angle θ_{id} in (6.28) (instead of those in (6.5)) and the control inputs u_{i1} and u_{i2} as in (6.15) and (6.18) with corresponding gains K_{i1} and K_{i2} satisfying (6.17) and (6.21), respectively, with the new bounds $\alpha_i(x)$ and $\beta_i(x)$ in Assumption 9, sliding mode occurs on all surfaces in finite time bounded by the bound in (6.14). The only difference here is that the value of $\dot{\theta}_{id}$ is calculated based on

$$\dot{\theta}_{id} = \begin{cases} \frac{\frac{d}{dt}\left(Z_{p_{yi}}\right)\cdot Z_{p_{xi}} - \frac{d}{dt}\left(Z_{p_{xi}}\right)\cdot Z_{p_{yi}}}{\left(Z_{p_{xi}}\right)^2+\left(Z_{p_{yi}}\right)^2}, & \|\nabla_{x_i}J(x)\| \geq \varepsilon \\ \frac{\frac{d}{dt}\left(Z_{p_{yi}}\right)\cdot Z_{p_{xi}} - \frac{d}{dt}\left(Z_{p_{xi}}\right)\cdot Z_{p_{yi}}}{\left(Z_{p_{xi}}\right)^2+\left(Z_{p_{yi}}\right)^2+\bar{\varepsilon}}, & \|\nabla_{x_i}J(x)\| < \varepsilon \end{cases}$$

instead of using (6.9) or (6.10) and the bounds $\alpha_i(x)$ and $\beta_i(x)$ in Assumption 9 needed for determining the values of the gains K_{i1} and K_{i2} are calculated based on Z_i in (6.27). Note also that whenever Equation (6.23) is satisfied $\frac{d}{dt}h\left(\nabla_{x_i}J(x,x_t)\right) = 0$ and the value of the bound $\alpha_i(x)$ depends only on the value of $\frac{d}{dt}\nabla_{x_i}J(x,x_t)$.

6.3 Potential Functions and Bounds

6.3.1 Aggregation

Similar to the case in Chapter 5, in order to achieve aggregation we will use potential functions of the form considered in Chapter 3. In other words, once again we will consider potential functions in the form (3.3) and satisfying all the assumptions stated in Section 3.2. For convenience we repeat here the potential function in equation (3.6) which is given by

$$J(x) = J_{aggregation}(x) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left[\frac{a}{2}\|x_i - x_j\|^2 + \frac{bc}{2}\exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)\right] \quad (6.29)$$

In order to be consistent with the results in Chapter 3 and Chapter 5 we consider the same potential function here and derive the bounds for it. As was shown before, the negative gradient of this potential at x_i is given by

$$-\nabla_{x_i}J(x) = -\sum_{j=1, j\neq i}^{N}(x_i - x_j)\left[a - b\exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)\right]$$

Let us first derive a bound $\alpha_i(x)$ such that $|\dot{v}_{id}| \leq \alpha_i(x)$. From Equation (6.6) we have the desired velocity for agent i as $v_{id} \triangleq \|\nabla_{x_i}J(x)\|$. Then, using the fact that $|\dot{v}_{id}| \leq \left\|\frac{d}{dt}\nabla_{x_i}J(x)\right\|$ one can determine the bound $\alpha_i(x)$ by overbounding $\left\|\frac{d}{dt}\nabla_{x_i}J(x)\right\|$. Taking the time derivative of $\nabla_{x_i}J(x)$ one obtains

$$\frac{d}{dt}(\nabla_{x_i}J(x)) = \begin{bmatrix} \frac{d}{dt}(J_{p_{xi}}) \\ \frac{d}{dt}(J_{p_{yi}}) \end{bmatrix} = \sum_{j=1, j\neq i}^{N} G(x_i - x_j)(\dot{x}_i - \dot{x}_j) \tag{6.30}$$

where $G(x_i - x_j)$ is a 2×2 matrix (since in this chapter $n = 2$) given by

$$G(x_i - x_j) = aI + b\exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)\left(\frac{2}{c}(x_i - x_j)(x_i - x_j)^\top - I\right) \tag{6.31}$$

where I is the 2×2 identity matrix. Then, we have

$$\begin{aligned} |\dot{v}_{id}| &\leq \sum_{j=1, j\neq i}^{N} \|G(x_i - x_j)\|\|\dot{x}_i - \dot{x}_j\| \leq \sum_{j=1, j\neq i}^{N} \|G(x_i - x_j)\|\left(\|\dot{x}_i\| + \|\dot{x}_j\|\right) \\ &\leq 2\sum_{j=1, j\neq i}^{N} \|G(x_i - x_j)\| \max_{k\in\{1,\ldots,N\}} \|\dot{x}_k\| \end{aligned} \tag{6.32}$$

where $\|G\|$ denotes the induced 2-norm of the matrix G. From (6.31) one can compute

$$\|G(x_i - x_j)\| \leq a + 2b\exp(-1) + b \leq a + 2b$$

where we used the fact that the maximum of $\|x_i - x_j\|^2 \exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)$ occurs at $\|x\| = \sqrt{c}$ and the inequality $\|x_i - x_j\|^2 \exp\left(-\frac{\|x_i - x_j\|^2}{c}\right) \leq c\exp(-1)$ is always satisfied. Then the bound can be calculated as

$$|\dot{v}_{id}| \leq 2N(a + 2b) \max_{j\in\{1,\ldots,N\}} \{v_j\} \triangleq \alpha_i(x) \tag{6.33}$$

Note that for the implementation of the algorithm the agents need to know the speeds of their neighbors which are all the other agents in the particular setting here.

In order to determine $\beta_i(x) > 0$ such that $|\ddot{\theta}_{id}| \leq \beta_i(x)$ (which is needed in order to determine the controller gain K_{i2}) we differentiate $|\theta_{id}|$ which from Equation (6.9) can be expressed as

$$\dot{\theta}_{id} = \frac{\frac{d}{dt}\left[\nabla_{x_i}J(x)^\dagger\right]^\top \nabla_{x_i}J(x)}{\|\nabla_{x_i}J(x)\|^2}$$

where $\nabla_{x_i}J(x)^\dagger \triangleq [J_{p_{yi}}, -J_{p_{xi}}]^\top$. Differentiating $\dot{\theta}_{id}$ and overbounding the result one obtains the bound on $|\ddot{\theta}_{id}|$ as

$$|\ddot{\theta}_{id}| \leq \frac{\left\|\frac{d^2}{dt^2}\left[\nabla_{x_i}J(x)\right]\right\|}{\|\nabla_{x_i}J(x)\|} + 3\frac{\left\|\frac{d}{dt}\left[\nabla_{x_i}J(x)\right]\right\|^2}{\|\nabla_{x_i}J(x)\|^2}$$

In order to determine the value of $\dot{\theta}_{id}$ in (6.9) as well as to calculate the bound $\beta_i(x)$ one needs the values of $\frac{d}{dt}\left[\nabla_{x_i}J(x)\right]$ and $\frac{d^2}{dt^2}\left[\nabla_{x_i}J(x)\right]$. The value of $\frac{d}{dt}\left[\nabla_{x_i}J(x)\right]$ for the potential function in Equation (6.29) can be calculated from Equations (6.30) and (6.31). In fact, the bound on $\| \frac{d}{dt} [\nabla_{x_i} J(x)] \|$ is given by $\alpha_i(x)$ calculated in (6.33). From Equation (6.30) for $\frac{d^2}{dt^2} [\nabla_{x_i} J(x)]$ one obtains

$$\frac{d^2}{dt^2}(\nabla_{x_i} J(x)) = \sum_{j=1, j\neq i}^{N} \left[\frac{d}{dt}\Big(G(x_i - x_j)\Big)(\dot{x}_i - \dot{x}_j) + G(x_i - x_j)(\ddot{x}_i - \ddot{x}_j) \right] \tag{6.34}$$

where

$$\begin{aligned} \tfrac{d}{dt}\Big(G(x_i - x_j)\Big) = -\tfrac{2b}{c} \exp\left(-\tfrac{\|x_i - x_j\|^2}{c}\right) \Big[& (\dot{x}_i - \dot{x}_j)^\top (x_i - x_j) \left(\tfrac{2}{c}(x_i - x_j)(x_i - x_j)^\top - I\right) \\ & -(\dot{x}_i - \dot{x}_j)(x_i - x_j)^\top - (x_i - x_j)(\dot{x}_i - \dot{x}_j)^\top \Big] \end{aligned}$$

and $G(x_i - x_j)$ was defined in (6.31). Then, using the inequalities

$$\|x_i - x_j\| \exp\left(-\frac{\|x_i - x_j\|^2}{c}\right) \leq \sqrt{\frac{c}{2}} \exp\left(-\frac{1}{2}\right)$$

and

$$\|x_i - x_j\|^3 \exp\left(-\frac{\|x_i - x_j\|^2}{c}\right) \leq \frac{3c}{2}\sqrt{\frac{3c}{2}} \exp\left(-\frac{3}{2}\right)$$

it is easy to show that

$$\left\| \tfrac{d}{dt}\Big(G(x_i - x_j)\Big) \right\| \leq \tfrac{6b}{c} \left(\sqrt{\tfrac{3c}{2}} \exp\left(-\tfrac{3}{2}\right) + \sqrt{\tfrac{c}{2}} \exp\left(-\tfrac{1}{2}\right) \right) \|\dot{x}_i - \dot{x}_j\|$$

Defining

$$\bar{\beta} = \frac{6b}{c} \left(\sqrt{\frac{3c}{2}} \exp\left(-\frac{3}{2}\right) + \sqrt{\frac{c}{2}} \exp\left(-\frac{1}{2}\right) \right) \tag{6.35}$$

one has

$$\left\| \frac{d^2}{dt^2} [\nabla_{x_i} J(x)] \right\| \leq N \left[4\bar{\beta} \max_{1 \leq j \leq N} \{v_j^2\} + 2(a + 2b) \max_{1 \leq j \leq N} \left[\frac{K_{j1}(x) + f_{vj}^+}{\underline{M}_j} \right] \right]$$

Then, $\beta_i(x)$ can be defined as

$$\beta_i(x) \triangleq \frac{N}{\varepsilon} \left[4\bar{\beta} \max_{1 \leq j \leq N} \{v_j^2\} + 2(a + 2b) \max_{1 \leq j \leq N} \left[\frac{K_{j1}(x) + f_{vj}^+}{\underline{M}_j} \right] \right] + 3 \left(\frac{\alpha_i(x)}{\varepsilon} \right)^2 \tag{6.36}$$

where ε is the lower bound on $\|\nabla_{x_i} J(x)\|$ in Equation (6.3).

Note here that in order to calculate $\beta_i(x)$, in addition to the value of $\alpha_i(x)$ and the velocities of its neighbors, the agent needs the values of $K_{j1}(x)$, f_{vj}^+, and $\underline{M}_j$ from their neightbors as well. This brings a restriction on the control algorithm since these values need to be sensed or communicated. However, for many applications the values of parameters such as f_{vj}^+, $\overline{M}_j$, $\underline{M}_j$, and ε_{j1} might be independent from the agent identity and fixed variables with values known a priori by all agents and the values of K_{j1} (needed for calculating can be calculating $\beta_i(x)$) can easily be calculated from these and the corresponding values of $\alpha_j(x)$. This, together with the fact that the value of $\alpha_j(x)$ is the same for all agents (see Equation (6.33)), results in relaxing the requirement for exchanging the values of $K_{j1}(x)$, f_{vj}^{+}, and $\underline{M}_j$ between agents.

6.3.2 Social Foraging

The bounds for the case of social foraging swarms, i.e., swarms moving in an environment modeled as a potential field (called a resource profile), can be derived in a similar manner to those for aggregating swarms. Recall that for the social foraging swarms case the motion of the swarm is along the negative gradient of a potential function of the form

$$J(x) = J_{foraging}(x) = \sum_{i=1}^{N} \sigma(x_i) + J_{aggregation}(x)$$

where $J_{aggregation}(x)$ is the potential in (6.29) that determines the inter-agent interactions in the swarm, whereas the term $\sigma : \mathbb{R}^2 \to \mathbb{R}$ is a "resource profile" which represents the effect of the environment on the motion of the swarm. Once again the negative gradient of the potential at the position x_i of agent i is given by

$$-\nabla_{x_i} J(x) = -\nabla_{x_i}\sigma(x_i) - \sum_{j=1, j\neq i}^{N} (x_i - x_j)\left[a - b\exp\left(-\frac{\|x_i - x_j\|^2}{c}\right)\right]$$

Using these below we will describe how to determine the bounds $\alpha_i(x)$ and $\beta_i(x)$ for several resource profiles.

Plane Resource Profiles

Recall that a plane resource profile can be represented with equation of the form

$$\sigma(y) = a_\sigma^\top y + b_\sigma$$

for which we have

$$\frac{d}{dt}(\nabla_{x_i}\sigma(x)) = \frac{d}{dt}(a_\sigma) = 0$$

Therefore, for the plane profile the values of $\alpha_i(x)$ and $\beta_i(x)$ are the same as for the case of aggregating swarms discussed in the preceding section and are given by Equations (6.33) and (6.36), respectively.

Quadratic Resource Profiles

For a quadratic resource profile of the form

$$\sigma(y) = \frac{A_\sigma}{2}\|y - c_\sigma\|^2 + b_\sigma$$

we have

$$\frac{d}{dt}(\nabla_{x_i}\sigma(x)) = \frac{d}{dt}(A_\sigma(x_i - c_\sigma)) = A_\sigma \dot{x}_i \quad \text{and} \quad \frac{d^2}{dt^2}(\nabla_{x_i}\sigma(x)) = A_\sigma \ddot{x}_i$$

Therefore, for this case for agent i the bounds $\alpha_i(x)$ and $\beta_i(x)$ can be calculated as

$$\alpha_i(x) \triangleq |A_\sigma| v_i + 2N(a+2b) \max_{j\in\{1,\ldots,N\}} \{v_j\} \tag{6.37}$$

and

$$\begin{aligned}\beta_i(x) \triangleq\ & \frac{|A_\sigma|}{\varepsilon}\frac{K_{i1}(x) + f_{vi}^+}{\underline{M}_i} \\ & + \frac{N}{\varepsilon}\left[4\bar{\beta}\max_{1\le j\le N}\{v_j^2\} + 2(a+2b)\max_{1\le j\le N}\left[\frac{K_{j1}(x)+f_{vj}^+}{\underline{M}_j}\right]\right] + 3\left(\frac{\alpha_i(x)}{\varepsilon}\right)^2\end{aligned} \tag{6.38}$$

where $\bar{\beta}$ is defined in (6.35).

Multimodal Gaussian Resource Profiles

For a multimodal Gaussian resource profile of the form

$$\sigma(y) = -\sum_{j=1}^{M}\frac{A_{\sigma j}}{2}\exp\left(-\frac{\|y - c_{\sigma j}\|^2}{l_{\sigma j}}\right) + b_\sigma$$

we have

$$\frac{d}{dt}(\nabla_{x_i}\sigma(x)) = -\sum_{j=1}^{M}\frac{A_{\sigma j}}{l_{\sigma j}}\exp\left(-\frac{\|x_i - c_{\sigma j}\|^2}{l_{\sigma j}}\right)\left[1 + 2(x_i - c_{\sigma j})(x_i - c_{\sigma j})^\top\right]\dot{x}_i$$

Therefore, for this case for agent i we have

$$\alpha_i(x) \triangleq \sum_{j=1}^{M}\frac{|A_{\sigma j}|}{l_{\sigma j}}\left[1 + 2l_{\sigma j}\exp(-1)\right]v_i + 2N(a+2b)\max_{j\in\{1,\ldots,N\}}\{v_j\} \tag{6.39}$$

Similarly, calculating the value of $\frac{d^2}{dt^2}(\nabla_{x_i}\sigma(x))$ (which is not shown here), over-bounding it and calculating $\beta_i(x)$ one obtains

$$\beta_i(x) \triangleq \frac{\bar{\beta}_1}{\varepsilon} v_i^2 + \frac{\bar{\beta}_2}{\varepsilon}\left[\frac{K_{i1}(x) + f_{vi}^+}{\underline{M}_i}\right] + \frac{N}{\varepsilon}\left[4\bar{\beta}\max_{1\le j\le N}\{v_j^2\} + 2(a+2b)\max_{1\le j\le N}\left[\frac{K_{j1}(x)+f_{vj}^+}{\underline{M}_j}\right]\right] + 3\left(\frac{\alpha_i(x)}{\varepsilon}\right)^2 \tag{6.40}$$

where $\bar{\beta}$ is as defined in Equation (6.35) and

$$\bar{\beta}_1 = \sum_{j=1}^{M}\frac{|A_{\sigma j}|}{l_{\sigma j}}\left[\left(\frac{2}{l_{\sigma j}}+4\right)\sqrt{\frac{l_{\sigma j}}{2}}\exp\left(-\frac{1}{2}\right) + 3l_{\sigma j}\sqrt{\frac{3l_{\sigma j}}{2}}\exp\left(-\frac{3}{2}\right)\right]$$

and

$$\bar{\beta}_2 = \sum_{j=1}^{M}\frac{|A_{\sigma j}|}{l_{\sigma j}}\Big(1 + 2l_{\sigma j}\exp(-1)\Big)$$

6.3.3 Formation Control

Recall that in the preceding chapters for formation control we used a potential function of the form

$$J(x) = J_{formation}(x) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left[\frac{a_{ij}}{2}\|x_i - x_j\|^2 + \frac{b_{ij}c_{ij}}{2}\exp\left(-\frac{\|x_i-x_j\|^2}{c_{ij}}\right)\right]$$

where the parameters a_{ij}, b_{ij}, and c_{ij} depend on the desired relative distances of the agents d_{ij} in the desired formation. These formation constraints can be expressed in the form

$$\|x_i - x_j\| = d_{ij}, 1 \le i, j \le N$$

From the similarity of $J_{formation}(x)$ with $J_{aggregation}(x)$ one can easily deduce that the corresponding bounds $\alpha_i(x)$ and $\beta_i(x)$ can be expressed as

$$\alpha_i(x) \triangleq 2N(a_{max} + 2b_{max})\max_{j\in\{1,\ldots,N\}}\{v_j\}$$

and

$$\beta_i(x) \triangleq \frac{N}{\varepsilon}\left[4\bar{\beta}\max_{1\le j\le N}\{v_j^2\} + 2(a_{max}+2b_{max})\max_{1\le j\le N}\left[\frac{K_{j1}(x)+f_{vj}^+}{\underline{M}_j}\right]\right] + 3\left(\frac{\alpha_i(x)}{\varepsilon}\right)^2$$

where $a_{max} = \max_{1\le i,j\le N}\{a_{ij}\}$, $b_{max} = \max_{1\le i,j\le N}\{b_{ij}\}$, and $c_{max} = \max_{1\le i,j\le N}\{c_{ij}\}$, and $\bar{\beta}$ is defined in (6.35).

6.3.4 Swarm Tracking

For the swarm tracking problem again we consider a potential function of the same form considered in the preceding chapters which is composed of two parts and can be expressed as

$$J(x,x_t) = W_t \sum_{i=1}^{N} \frac{1}{4} \|x_i - x_t\|^4 + W_f J_{formation}(x)$$

where $J_{formation}(x)$ is the potential considered for the case of formation control that was discussed in the preceding section. Recall that it specifies the geometrical shape to be formed by the swarm around the target, whereas the first term specifies the requirement by the agents to track the moving target and to surround/enclose it. We would like to remind the reader here that the motion of the agents in the swarm tracking case is different from the previous cases and is given by (6.22) or (6.24) (and not by (6.2)).

The negative gradient of the potential at the position x_i of agent i can be calculated as

$$\begin{aligned} -\nabla_{x_i} J(x,x_t) &= -W_t \|x_i - x_t\|^2 (x_i - x_t) \\ &\quad -W_f \sum_{j=1, j\neq i}^{N} (x_i - x_j) \left[a_{ij} - b_{ij} \exp\left(-\frac{\|x_i - x_j\|^2}{c_{ij}} \right) \right] \end{aligned}$$

The time derivative of the first term in the above equation is given by

$$\frac{d}{dt}\left(-W_t \|x_i - x_t\|^2 (x_i - x_t)\right) = -W_t \left(2(x_i - x_t)^\top (\dot{x}_i - \dot{x}_t)(x_i - x_t) + \|x_i - x_t\|^2 (\dot{x}_i - \dot{x}_t) \right)$$

Similarly the time derivative of $h(y)$ in (6.25) for a $y \in \mathbb{R}$ can be calculated as

$$\frac{d}{dt} h(y) = \begin{cases} \left(\frac{\pi}{2\varepsilon}\right) \cos\left(\frac{\pi y}{2\varepsilon}\right) \dot{y}, & |y| \leq \varepsilon \\ 0, & |y| > \varepsilon \end{cases} \tag{6.41}$$

which always (for all finite $y \in \mathbb{R}$) satisfies

$$\left\| \frac{d}{dt} h(y) \right\| \leq \bar{\alpha}(y) \dot{y}$$

where $\bar{\alpha}(y)$ is defined as

$$\bar{\alpha}(y) = \begin{cases} \left(\frac{\pi}{2\varepsilon}\right), & |y| \leq \varepsilon \\ 0, & |y| > \varepsilon \end{cases} \tag{6.42}$$

Using the above, and the bound calculated in the preceding section for the formation control case, the bound $\alpha_i(x)$ can be expressed as

$$\alpha_i(x) \triangleq \left(\eta + \lambda \frac{\pi}{\varepsilon}\right) \left(3W_t \|x_i - x_t\|^2 (v_i + \gamma_{tv}) + 2W_f N (a_{max} + 2b_{max}) \max_{j \in \{1,\ldots,N\}} \{v_j\} \right)$$

where γ_{tv} is the bound on the velocity of the agent such that $\|\dot{x}_t\| \leq \gamma_{tv}$ given in Assumption 8 and a_{max}, b_{max}, and c_{max} are as defined above (for the formation control case).

Taking the second derivative of $h(y)$ in (6.25) for a $y \in \mathbb{R}$ and utilizing $\bar{\alpha}(y)$ in (6.42) one obtains

$$\frac{d^2}{dt^2}h(y) = \bar{\alpha}(y)\left(-\left(\frac{\pi}{2\varepsilon}\right)\sin\left(\frac{\pi y}{2\varepsilon}\right)\dot{y} + \cos\left(\frac{\pi y}{2\varepsilon}\right)\ddot{y}\right) \tag{6.43}$$

which with the second derivative of the potential function $J(x,x_t)$ (not shown here) the bound $\beta_i(x)$ can be calculated as

$$\begin{aligned}\beta_i(x) \triangleq {} & \left(\eta+\lambda\frac{\pi}{\varepsilon}\right)\left(\frac{W_t}{\varepsilon}\left[6\|x_i-x_t\|(v_i+\gamma_{tv})^2+3\|x_i-x_t\|^2\left(\frac{K_{i1}(x)+f_{vi}^{+}}{\underline{M}_i}+\gamma_{ta}\right)\right]\right.\\ & \left.+\frac{NW_f}{\varepsilon}\left[4\bar{\beta}\max_{1\le j\le N}\{v_j^2\}+2(a_{max}+2b_{max})\max_{1\le j\le N}\left[\frac{K_{j1}(x)+f_{vj}^{+}}{\underline{M}_j}\right]\right]\right)\\ & +\frac{1}{2}\lambda\left(\frac{\pi}{\varepsilon}\right)^2\left(\frac{\alpha_i(x)}{\varepsilon}\right)+3\left(\frac{\alpha_i(x)}{\varepsilon}\right)^2\end{aligned} \tag{6.44}$$

where γ_{ta} is the bound on the acceleration of the agent such that $\|\ddot{x}_t\| \le \gamma_{ta}$ given in Assumption 8.

Note that for calculating the values of the bounds $\beta_i(x)$ above the values of the corresponding $\alpha_i(x)$ are used. Then the values of these bounds $\alpha_i(x)$ and $\beta_i(x)$ are used for determining the controller gains $K_{i1}(x)$ and $K_{i2}(x)$ in equations (6.17) and (6.21), respectively. We would like to also emphasize that all bounds computed in this section are very conservative and that usually the procedure works satisfactorily with smaller values of the bounds $\alpha_i(x)$ and $\beta_i(x)$ and therefore smaller controller gains $K_{i1}(x)$ and $K_{i2}(x)$ as well.

6.4 Simulation Examples

In this section we present numerical simulation results to reproduce the swarm behavior obtained in Chapter 3 for the single integrator agent model and in Chapter 5 for the fully actuated agent model for swarms composed of agents with nonholonomic unicycle dynamics with model uncertainties in (6.1). In other words, we will test and verify the effectiveness of the control scheme discussed in the preceding sections of this chapter.

As in Chapter 5, in order to avoid numerical problems such as high frequency chattering which might occur due to the discontinuous characteristics of the sign function in the controller equations in (6.15) and (6.18) we used the function $\tanh(\gamma y)$ instead of the sign function. Here γ is a smoothness parameter which determines the slope of the function around $y=0$ and therefore the similarity between the sign and tanh functions. We used the value of $\gamma = 10$ as a smoothness parameter for all of the cases below. Moreover, we used $\varepsilon_{i1} = \varepsilon_{i2} = 1$ for calculating the controller gains.

The actual values of the mass and the inertia of agents are unknown and determined randomly at the beginning of the simulation by the program according to upper and lower bounds $\overline{M}_i = \overline{I}_i = 1.2$ and $\underline{M}_i = \underline{I}_i = 1.0$. The bounded unmodeled dynamics and disturbances are assumed to be

$$f_{vi}(t) = 1.2\sin(1.2t) \quad \text{and} \quad f_{wi}(t) = 1.2\cos(0.2t)$$

with the corresponding known bounds on them $f_{vi}^{+} = f_{wi}^{+} = 1.2$. The size of the boundary layer was chosen as $\varepsilon = 0.1$ and the value of $\bar{\varepsilon}$ which is added to the denominator in the equation of $\dot{\theta}_{id}$ within the boundary layer in order to prevent unboundedness was chosen as $\bar{\varepsilon} = 0.01$. The rest of the parameters are case dependent and hence specified in the corresponding sections below.

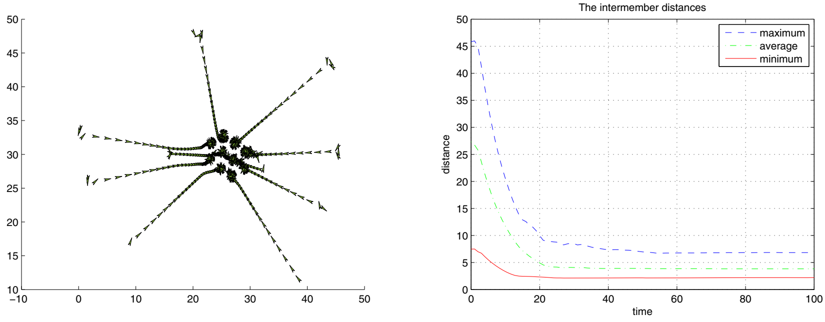
Before proceeding to results for the particular cases, note that for the simulations below all the steering angles θ_i and θ_{id} were projected within $[-\pi, \pi)$. Moreover, the relative angle differences were calculated (mod 2π) with $-\pi$ radians shift. In other words, $\theta_i - \theta_{id}$ were calculated using $[(\theta_i - \theta_{id} + \pi) (\text{mod } 2\pi) - \pi]$. This results in considering the smaller angle between the two orientations as the angle difference (and forces the agent to turn in that direction instead of the opposite direction).

6.4.1 Aggregation

First we consider the case of swarm aggregation. We performed several simulations with a different number of agents. However, since the simulations obtained for different parameters and agent numbers are in principle the same (do not differ qualitatively) here we show only the ones for $N = 10$ and potential function parameters $a = 0.01$, $b = 20$, and $c = 1$. As was mentioned in the preceding sections the bounds $\alpha_i(x)$ and $\beta_i(x)$, and therefore the controller gains $K_{i1}(x)$ and $K_{i2}(x)$, are very conservative and the procedure usually works for much smaller bounds. For this reason, in order to avoid numerical problems we saturated the controller gains of all the agents at $K_{1max} = 10$ and $K_{2max} = 20$. As a slope parameter of the orientation sliding line/surface we used $c_{\theta_i} = 0.1$ for all agents. The simulation results show that the agents aggregate as predicted by theory.

Figure 6.3(a) shows the motion (the trajectories) for random initial positions and orientations. The agents are plotted as polygons so that their orientations are explicitly shown. It is observed that the agents aggregate quickly and after aggregation they start to reorient themselves since there the variation of the time-varying potential function (which is due to the motion of the other agents in the group) is higher. Figure 6.3(b) depicts the inter-agent distances during the motion of the swarm. The curves specify the maximum, minimum, and average distances between the agents of the swarm. The distance decreases exponentially as expected and they converge to constant values similar to the results obtained before. For the above values of the parameters a, b, and c the distance at which the attraction and repulsion between two individuals balance is $\delta = 2.7570$. As can be seen from the figure, no collisions between agents occur and the final swarm size becomes approximately 6.8. An interesting observation here is that at their final positions the agents are distributed in almost a grid-like arrangement. Also, one should note that the distances between final positions of swarm members change for different values of attraction and repulsion parameters. For example, increasing the attraction parameter a or decreasing the repulsion parameter b results in a decrease in the inter-agent distances at the final positions.

The effect of disturbances results in the fact that even after aggregation the control inputs u_{i1} and u_{i2} do not converge to zero. Instead, they continue to counterbalance

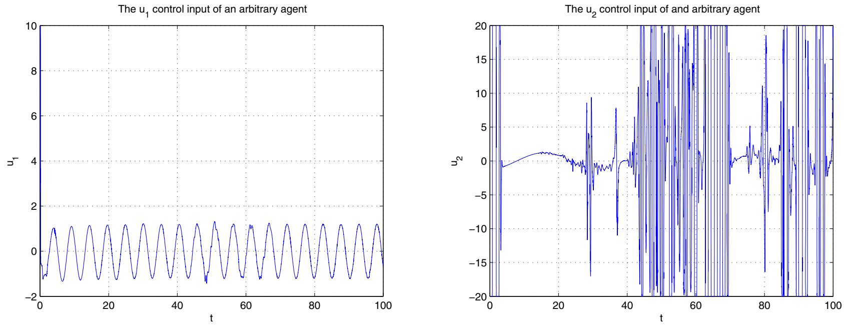


(a) The paths of the agents in the swarm.

(b) Inter-agent distances in the swarm.

Fig. 6.3. Aggregating swarm of agents with non-holonomic vehicle dynamics subject to external disturbances and model uncertainty.

disturbances to preserve cohesiveness. Figure 6.4 shows the u_{i1} and u_{i2} inputs for a randomly selected agent. Normally, because of the highly conservative values of the bounds, these inputs would become very large. However, as mentioned above to avoid numerical problems here we saturated them. It is interesting to note from Figure 6.4(a) that u_{i1} is saturated only at the beginning of the motion of the swarm and in a short period of time (only few seconds) it converges to a function such that to counterbalance the additive disturbance $f_{vi}(t)$. In contrast, in order to accommodate the abrupt changes in the direction of the gradient (which occurs after aggregation) the value of u_{i2} shown in Figure 6.4(b) is continued to be saturated with high frequency after aggregation. The reader can easily note that the behavior of the swarm is similar to the behavior of the swarms presented in Chapter 3 and Chapter 5.



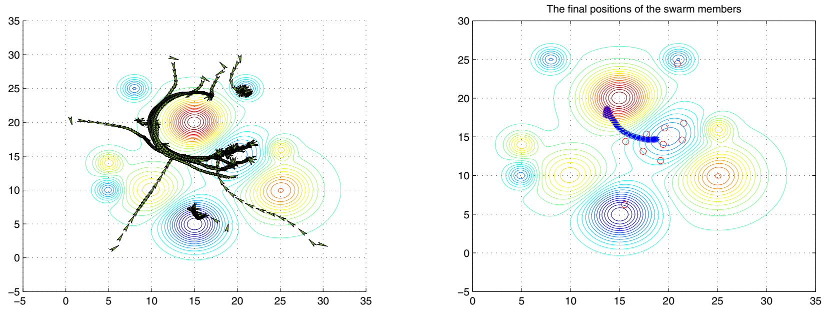
(a) The u_{i1} input.

(b) The u_{i2} input.

Fig. 6.4. The u_{i1} and u_{i2} control inputs of an arbitrary agent in the swarm.

6.4.2 Social Foraging

Figure 6.5(a) shows the paths of the agents for the case of a social foraging swarm with $N = 10$ agents and parameters of the inter-agent potential function as for the aggregation case or $a = 0.01$, $b = 20$, and $c = 1$. The environmental resource profile is the same multimodal Gaussian profile considered in the preceding chapters which is composed of $M = 10$ Gaussians (see for example Section 5.4.2). For the simulation presented in Figure 6.5 the control input gains were saturated at $K_{1max} = 20$ and $K_{2max} = 40$ and the slope parameter of the orientation sliding line/surface is chosen as $c_{\theta_i} = 0.05$. The contour curves in Figure 6.5(a) show the equipotential contours of the environmental potential. As can be seen from the figure the agents tend to move towards the minima of the profile while staying cohesive with close neighbors and avoiding maxima. Figure 6.5(b) shows the positions of the agents (indicated as circles) at the end of the simulation (i.e., after 100 seconds) as well as the motion of the centroid of the swarm (indicated as stars). As one can see, the agents have aggregated around respective local minima points of the resource profile as expected by the theory. The shape of the control inputs u_{i1} and u_{i2} of the agents is in principle similar to those obtained for the aggregation case above (except that the gains are saturated at different values) and therefore not shown here.

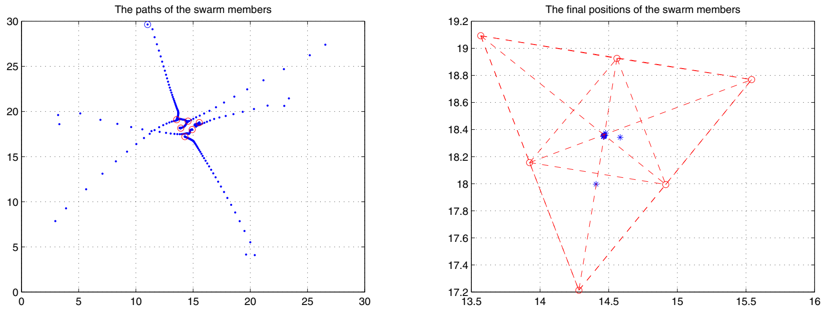


(a) The paths of the agents. (b) The final positions of the agents.

Fig. 6.5. Foraging swarm of agents with non-holonomic vehicle dynamics subject to external disturbances and model uncertainty.

6.4.3 Formation Control

Figure 6.6 illustrates a result of an application of the method to the formation control problem. Figure 6.6(a) shows the paths of the agents, whereas in Figure 6.6(b) the final positions of the agents marked as small circles, together with the movement of the swarm centroid marked with stars, are depicted. For this formation control simulation the control input gains are saturated at $K_{1max} = 50$ and $K_{2max} = 100$ and the slope parameter of the orientation sliding line/surface was chosen as $c_{\theta_i} = 0.05$.



(a) The paths of the agents.

(b) The final positions of the agents.

Fig. 6.6. A swarm of $N = 6$ agents with non-holonomic vehicle dynamics subject to external disturbances and model uncertainty forming an equilateral triangle formation.

In this simulation a group of six agents are required to form an equilateral triangle formation with lateral site equal to 2 units. For this reason the desired inter-agent distances d_{ij} are equal to $1, 2$, or $\sqrt{3}$ depending on the relative positions of the agents in the triangle. We used the values $b_{ij} = 20$ and $c_{ij} = 0.2$ for all i, j, as parameters of the potential function $J_{formation}(x)$ and calculated a_{ij} (again) using

$$a_{ij} = b_{ij} \exp\left(-\frac{d_{ij}^2}{c_{ij}}\right)$$

which, given the above mentioned desired inter-agent distances results in $a_{ij} \in \{0.1348, 6.1180 \times 10^{-6}, 4.1223 \times 10^{-8}\}$ depending on the inter-agent distances in the desired formation. As one can observe from the figure the objective is achieved despite the uncertainties and disturbances in the agent dynamics.

6.4.4 Swarm Tracking

In this subsection the simulation results for swarm tracking will be presented. The target is assumed to move with non-holonomic dynamics similar to the dynamics of the agents. The equations of motion of the target are given by

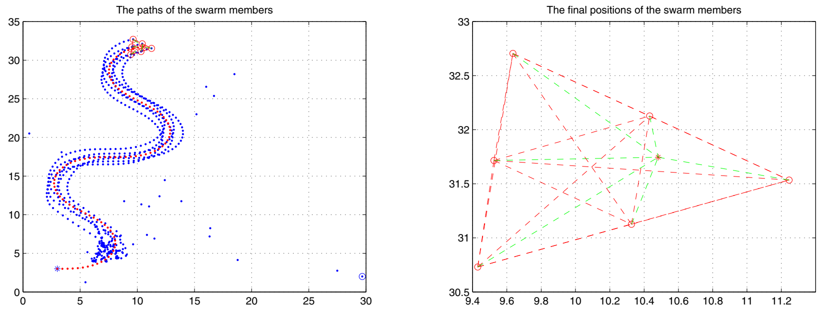
$$\begin{aligned} \dot{x}_T &= v_T \cos(\theta_T), \\ \dot{y}_T &= v_T \sin(\theta_T), \\ \dot{\theta}_T &= w_T, \end{aligned} \tag{6.45}$$

and the control inputs (the translational and angular speeds) of the target are assumed to be

$$\begin{aligned} v_T(t) &= 1.5, \\ w_T(t) &= 0.5 \sin(0.4t) \end{aligned}$$

We use $\gamma_{tv} = 1.5$ and $\gamma_{ta} = 0.75$ as bounds on the velocity and the acceleration of the target. The parameters of equation (6.24) are selected as $\eta = 0.1$ and $\lambda = 1.6$

($\lambda > \gamma_{tv}$). The approximation function of signum in (6.25) is used with $\varepsilon = 0.1$ which is also the value of the boundary layer. The parameters specifying the weights of the formation and tracking potentials in the artificial potential function are chosen as $W_t = 0.1$ and $W_f = 0.9$ giving higher importance to the formation. The geometrical shape for the formation is chosen to be the same equilateral triangle as in the preceding section with parameters $b_{ij} = 20$ and $c_{ij} = 20$ for all i, j, and the values of a_{ij} were calculated using the equation provided in the preceding section. The slope parameter of the orientation sliding line/surface was chosen as $c_{\theta_i} = 0.05$.



(a) The paths of agents and target. (b) The final positions of agents and target.

Fig. 6.7. A swarm of $N = 6$ agents with non-holonomic vehicle dynamics subject to external disturbances and model uncertainty tracking a moving target and forming an equilateral triangle formation.

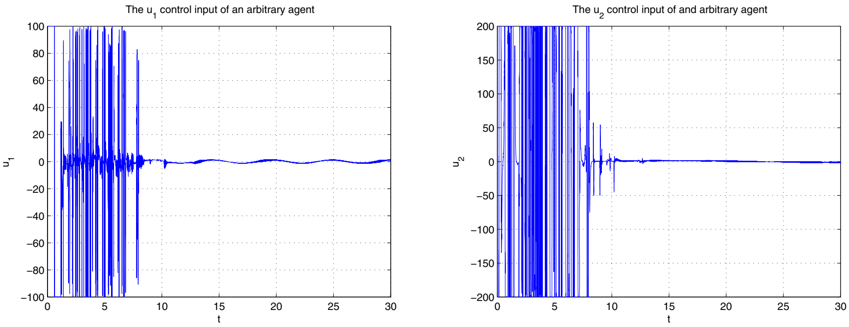
Figure 6.7 shows the results for a typical simulation run. The paths of the agents are shown in Figure 6.7(a), whereas Figure 6.7(b) presents the relative positions of the agents marked with circles and the target marked with a star at the end of the simulation. The initial configuration of the target is given by $[3, 3, 0]$ while the agent positions are initialized randomly within $[0, 30] \times [0, 30]$ with random orientations. As can be seen from the figure, the agents enclose/surround the target very fast, form the desired formation around it, and move together with it.

The plots of the u_{i1} and u_{i2} controller inputs for an arbitrary agent are shown in Figure 6.8. They were saturated at $K_{1max} = 100$ and $K_{2max} = 200$ as can be seen from the figure. However, as one can also notice after some transient their values settle to counterbalance the disturbances as the swarm moves in a formation surrounding the target.

6.5 Further Issues

6.5.1 Model Equivalence

Note that although in the model in (6.1) the control inputs are the force $u_{i1} = F_i$ and the torque $u_{i2} = \tau_i$, the model is equivalent to the case in which the control inputs



(a) The u_{i1} input.

(b) The u_{i2} input.

Fig. 6.8. The u_{i1} and u_{i2} control inputs of an arbitrary agent in the swarm.

are the voltages $\bar{u}_{i1} = V_{R_i}$ and $\bar{u}_{i2} = V_{L_i}$ to the motors driving the two wheels of a differentially driven robot depicted in Figure 6.1. To see this note that

$$v_i = \frac{r_i(w_{R_i} + w_{L_i})}{2} \quad \text{and} \quad w_i = \frac{r_i(w_{R_i} - w_{L_i})}{2L_i}$$

where w_{R_i} and w_{L_i} are the turning angular speeds of the right and the left motors, respectively, r_i is the radius of the wheels, and $2L_i$ is the length of the axel connecting the two wheels. The first order motor dynamics of the wheels can be represented as

$$\begin{aligned} \dot{w}_{R_i} &= f_{R_i}(\cdot) + b_{R_i} V_{R_i} \\ \dot{w}_{L_i} &= f_{L_i}(\cdot) + b_{L_i} V_{L_i} \end{aligned} \tag{6.46}$$

where V_{R_i} and V_{L_i} are the voltages applied to the right and the left motors, respectively, whereas $f_{R_i}(\cdot)$, $f_{L_i}(\cdot)$, b_{R_i}, and b_{L_i} represent respectively the system functions and parameters including uncertainties. Using these we have

$$\dot{v}_i = \frac{r_i}{2}\Big(f_{R_i}(\cdot) + f_{L_i}(\cdot)\Big) + \frac{r_i}{2}\Big(b_{R_i} V_{R_i} + b_{L_i} V_{L_i}\Big)$$

and

$$\dot{w}_i = \frac{r_i}{2L_i}\Big(f_{R_i}(\cdot) - f_{L_i}(\cdot)\Big) + \frac{r_i}{2L_i}\Big(b_{R_i} V_{R_i} - b_{L_i} V_{L_i}\Big)$$

Then, redefining

$$f_{v_i} = \frac{m_i r_i}{2}\Big(f_{R_i}(\cdot) + f_{L_i}(\cdot)\Big) \quad \text{and} \quad f_{w_i} = \frac{I_i r_i}{2L_i}\Big(f_{R_i}(\cdot) - f_{L_i}(\cdot)\Big)$$

and

$$F_i = \frac{m_i r_i}{2}\Big(b_{R_i} V_{R_i} + b_{L_i} V_{L_i}\Big) \quad \text{and} \quad \tau_i = \frac{I_i r_i}{2L_i}\Big(b_{R_i} V_{R_i} - b_{L_i} V_{L_i}\Big)$$

one obtains the model in (6.1). In other words, the relation between the model in (6.1) and an alternative agent model given by

$$\begin{aligned}\dot{p}_{xi} &= \tfrac{r_i(w_{R_i}+w_{L_i})}{2}\cos(\theta_i),\\ \dot{p}_{yi} &= \tfrac{r_i(w_{R_i}+w_{L_i})}{2}\sin(\theta_i),\\ \dot{\theta}_i &= \tfrac{r_i(w_{R_i}-w_{L_i})}{2L_i},\\ \dot{w}_{R_i} &= f_{R_i}(\cdot)+b_{R_i}V_{R_i}\\ \dot{w}_{L_i} &= f_{L_i}(\cdot)+b_{L_i}V_{L_i}\end{aligned} \tag{6.47}$$

is just a simple linear transformation. Therefore, the results in this chapter hold also for swarms composed of agents with dynamics represented with Equation (6.47).

6.5.2 Extensions and Generalizations

One restriction here is that to calculate the bounds $\alpha_i(x)$ and $\beta_i(x)$ and therefore implement the control algorithm for agent i one needs not only the position but also the velocity of its neighbors. In the particular setting here the neighbors of an agent are all the other agents. However, this is not necessarily required to be the case in general and the procedure can be extended to the case in which not all agents are neighbors of each other (but there is a strongly connected interaction graph). Moreover, the neighbor agent velocities can be estimated using adaptive methods or vision based techniques.

The shortcomings of the raw form of the sliding mode control scheme are the so-called chattering effect and possible generation of high-magnitude control signals. These shortcomings may possibly be avoided or relaxed via usage of observers, integration, and adaptive or some filtering techniques. Alternative methods can also be employed. A simple study in this direction for a target tracking problem using fully actuated agents and adaptive backstepping can be found in [73]. Also note that the effect of introducing of the tanh function instead of the sign function (which has in a sense a boundary layer effect) or using the desired orientation in (6.10) was not mathematically analyzed. It is also possible to introduce a boundary layer and rigorously analyze the behavior of the system within the layer.

6.5.3 For Further Reading

There are studies which consider only the kinematic part of the dynamics in (6.1) consisting of only the first three equations [56, 68, 89, 150, 166, 212, 216, 237, 266]

$$\begin{aligned}\dot{p}_{xi} &= v_i\cos(\theta_i),\\ \dot{p}_{yi} &= v_i\sin(\theta_i),\\ \dot{\theta}_i &= w_i,\end{aligned} \tag{6.48}$$

Some authors [236] have considered a four-state part of the model in (6.1) or the complete five state model [144] using alternative techniques. It is also possible to add one more integrator to the force input terminal of the model in (6.1) and extend it to a six state model which, under certain conditions, can be completely linearized via the feedback linearization method [78].

It is well known that for the dynamics in (6.1) the position $x_i = (p_{ix}, p_{iy})$ and the orientation θ_i of the robot cannot be simultaneously stabilized by a continuous static (time-invariant) feedback [29]. In order to avoid this problem one may, for some $d_i > 0$, define

$$z_i = \left[\, p_{ix} + d_i \cos(\theta_i),\; p_{iy} + d_i \sin(\theta_i) \,\right]^\top \tag{6.49}$$

as the output of the system and set the control objective based on the position of that output. This output may represent the position of a gripper at the end of a hand of length d_i or a sensor positioned in front of the robot. With respect to that output the system is input-output feedback linearizable with a well defined relative degree equal to two [144]. The drawback is that under the feedback linearizing controller the zero dynamics of the system are only marginally stable, which may lead to instability in the internal unobservable dynamics during tracking of certain trajectories [78, 144]. This may not be a problem in some applications since internal unobservable dynamics constitutes of the orientation angle θ_i. However, if having unstable θ_i is unacceptable one should not use a feedback linearizing controller and seek different techniques as was done in this chapter.

A simplified version of the kinematic model in (6.48) with all agents moving with a fixed translational speed $v_i = v$ for all i is called the *Dubins' vehicle model* and is widely used in the literature [63, 166, 213]. In some studies it is also represented in the form

$$\begin{aligned} \dot{p}_i &= e^{j\theta_i} \\ \dot{\theta}_i &= u_i \end{aligned} \tag{6.50}$$

where the vector $p_i \in C \approx \mathbb{R}^2$ denotes the position of agent/particle i in complex notation and $e^{j\theta_i} = \cos(\theta_i) + j\sin(\theta_i)$ (here $j = \sqrt{-1}$). This was the model considered in [216] where the authors studied the connection between oscillator synchronization and collective motions.

The Dubins' vehicle model is found useful in various studies where each vehicle of interest is required to move with a constant translational speed because of dynamic constraints (such as the ones for flight of certain unmanned aerial vehicles (UAVs) at a specified altitude) [213, 238] or optimality considerations (such as using the maximum available speed of each vehicle) [27, 213].

This chapter is based on the works in [88, 90, 163]. The primary gradient tracking tools were developed earlier in [110, 112, 113] and here were applied to the swarm related problems. For general references on sliding mode control one can see the tutorials [244, 262] and the books [245, 247]. Implementations of potential function based strategies on real mobile robots with unicycle dynamics can be found in [167, 168].

7

Formation Control Using Nonlinear Servomechanism

In this chapter we consider the formation control problem in the context of output regulation of nonlinear systems or basically nonlinear servomechanism, which is a completely different approach from the potential functions method discussed in the preceding chapters. First we formulate the problem for general nonlinear agent dynamics and present two different type of controllers which are namely the full information and error feedback controllers. Following that we discuss how various formation maneuvers can be achieved using the procedure presented. Finally, numerical simulation examples are presented in order to illustrate the technique.

7.1 General Non-linear Agent Model

Let us consider a multi-agent system which consists of N agents with motion dynamics described by the general non-linear model of the form

$$\begin{aligned}\dot{x}_i &= f_i(x_i, \mu_i, u_i),\\ y_i &= h_i(x_i, \mu_i), 1 \leq i \leq N,\end{aligned} \tag{7.1}$$

where $x_i \in \mathbb{R}^{n_i}$ represents the local state of agent i, $u_i \in \mathbb{R}^{m_i}$ represents its local control input, $\mu_i \in \mathbb{R}^{r_i}$ represents the local exogenous inputs (i.e., local reference inputs and possibly disturbances) affecting the dynamics of agent i and $y_i \in \mathbb{R}^{m_i}$ is its local output. The outputs y_i are assumed to be variables which are used to define the formation. For example, for vehicle formations x_i may represent the internal vehicle states, whereas y_i the position and/or orientation or basically the configuration of the vehicle. It is assumed that the functions f_i and h_i are *known* and *smooth* for all $i = 1, \ldots, N$.

The signals μ_i are assumed to be generated by the local *neutrally stable* systems

$$\dot{\mu}_i = g_{\mu_i}(\mu_i), i = 1, \ldots, N, \tag{7.2}$$

where g_{μ_i} are also *known* and *smooth*. Local reference inputs could be, for example, variables which determine the relative positions of the individuals in the formation.

Therefore, these local systems could be used to generate the reference trajectories for the agents during formation maneuvers such as expansion/contraction, rotation, topology change or reconfiguration of the formation or others. Moreover, they can model any local neutrally stable (i.e., mostly constant or periodic) disturbances acting on the agent dynamics.

In this chapter we consider only the formation control problem, i.e., the problem in which the group of agents moves in a predefined formation. However, the procedure can easily be extended to work for other behaviors such as aggregation and foraging as well. To begin with, we assume that the desired formation is uniquely determined by a set of constraints as defined below.

Assumption 10. *There are a set of predefined constraints*

$$\eta_{i,j}(y_i, y_j, t) = 0, 1 \leq i, j, \leq N, j \neq i, \tag{7.3}$$

which uniquely determine the formation.

We would like to mention here that the word *uniquely* in Assumption 10 is used in a loose sense, meaning uniquely in terms of geometrical shape and translation and rotation of the complete formation might be allowed. The time dependence of the constraints is due to the fact that it might be needed for the swarm to perform formation maneuvers as time progresses. Moreover, note that Equation (7.3) is defined as if there are constraints for every pair (i, j) of individuals. However, it is not absolutely necessary to have constraints for every pair of agents. It is sufficient to have only minimal number of constraints, which "uniquely" determine the desired formation. For example, in $\mathbb{R}^2$ for every agent it is sufficient to have constraints with respect to only two appropriately chosen preceding agents. It is possible to obtain such a set of minimal number of constraints, by using the concepts of rigid and unfoldable graphs from graph theory.

For many applications involving collective of autonomous robots, marine vehicles, uninhabited aerial vehicles (UAV's) or space vehicles such as satellites or inter-terrestrial vehicles the constraints in (7.3) may be just requirements in terms of relative distances of the form

$$\eta_{i,j}(y_i, y_j, t) = \|y_i - y_j\| - d_{ij}(t) = 0, 1 \leq i, j, \leq N, j \neq i, \tag{7.4}$$

where $d_{ij}(t)$ is the (possibly time varying) desired distance between agents i and j in the formation. Note, however, that the procedure is not limited to such constraints only.

In the framework here, we assume that there exists a *virtual leader* for the formation and the individuals are (i.e., the formation is) required to follow (track) that leader. In other words, the virtual leader generates the reference trajectories for the whole formation. Then, the objective is to design each of the local control inputs u_i such that the formation constraints in (7.3) are satisfied and the formation follows the trajectories of the virtual leader. We assume that the dynamics of the virtual leader are generated by the *neutrally stable* system

$$\begin{aligned}\dot{s} &= g_l(s),\\ y_l &= q_l(s),\end{aligned} \tag{7.5}$$

where $s \in \mathbb{R}^r$ and g_l is *known* and *smooth*. Note that in many applications we can choose (i.e., design) the virtual leader dynamics based on the mission requirements.

To achieve leader following, in addition to the formation constraints in Assumption 10, the agents in the swarm are required to satisfy a set of tracking constraints as defined below.

Assumption 11. *There are a set of predefined constraints*

$$\eta_{i,l}(y_i, y_l, t) = 0, 1 \leq i \leq N, \tag{7.6}$$

which need to be satisfied by the agents during motion.

As in the case of formation constraints, for many robotic, UAV, marine or space applications, the tracking constraints may be just requirements in terms of relative distances of the agents to the virtual leader and be of the form

$$\eta_{i,l}(y_i, y_l, t) = \|y_i - y_l\| - d_{il}(t) = 0, 1 \leq i \leq N, \tag{7.7}$$

where $d_{il}(t)$ is the (possibly time varying) desired distance between agent i and the virtual leader.

Note that for each application, depending on the application requirements, we may have different formation and tracking constraints. For example, for an application in which a swarm of agents is required to complete a search and rescue mission it may be desired that the agents move in a different formation compared to an application in which a swarm of agents is required to guard an object. Similarly, the formation needed by a group of satellites might be different from the formation needed by a group of military robots. However, it is also important to realize that the two sets of constraints, i.e., the formation constraints in Assumption 10 and the tracking constraints in Assumption 11, cannot be just arbitrary. In other words, in order for the problem to be solvable the formation constraints and the tracking constraints need to be nonconflicting with each other, i.e., simultaneous satisfaction of both of the constraints in (7.3) and (7.6) should be feasible.

One can easily note that since the required motion dynamics of the agents are tied to the dynamics of the virtual leader through the tracking constraints, it is possible to view the dynamics of the virtual leader as external inputs to the individual agent dynamics. With this in mind, for each agent i we define $s_i = [s^\top, \mu_i^\top]^\top$ and each of the local exosystems becomes

$$\dot{s}_i = g_i(s_i) = \begin{bmatrix} g_l(s) \\ g_{\mu_i}(\mu_i) \end{bmatrix}, \tag{7.8}$$

which are *neutrally stable* and the $g_i(s_i)$ functions are *known* and *smooth*. We would like to emphasize that the assumption that we know the dynamics g_i is not absolutely necessary. In fact, we only need to know the dynamics of an *internal model* which can generate the same output signals as the system in (7.8) (the output of that system will be defined later). If we use a state feedback controller with full information, then the internal model is the system itself. If we use an error feedback controller, on the other hand, the internal model is an immersion of the system.

In order to approach the problem from the perspective of output regulation it might be possible to view the constraints $\eta_{i,j}$ and $\eta_{i,l}$ as the new outputs of the system and develop the local controllers u_i to regulate these outputs to zero. However, here we will not take this approach. Instead, we have the following simplifying assumption.

Assumption 12. *There exist known smooth mappings $q_i(s_i), i = 1, \ldots, N$ such that*

$$\begin{aligned} \eta_{i,j}(q_i(s_i), q_j(s_j)) &= 0, 1 \leq i, j \leq N, j \neq i, \\ \eta_{i,l}(q_i(s_i), q_l(s)) &= 0, 1 \leq i \leq N. \end{aligned} \tag{7.9}$$

Assumption 12 constitutes, in a sense, a “feasibility assumption.” We say that the problem is feasible if the formation constraints (7.3) and the tracking constraints (7.6) can both be simultaneously satisfied. In fact, if this is the case, then there always exists mappings $q_i(s_i), i = 1, \ldots, N$, satisfying (7.9). Therefore, this part of the assumption is very realistic and not restrictive at all. The part that restricts the assumption a little bit is the requirement that the mappings $q_i(s_i), i = 1, \ldots, N$ are known since, in general, these mappings may not necessarily be always known. Nevertheless, in many practical formation control applications and in particular for applications with constraints in terms of relative distances between the agents and missions such as tracking constant or periodic trajectories it is possible to determine such mappings.

The mappings $q_i(s_i)$ constitute reference trajectories for the individual agents which, if tracked by the agents lead to simultaneous satisfaction of both the formation and tracking constraints. Note also that it is possible to view $q_i(s_i)$ as the output trajectories of virtual agents whose internal dynamics are given by the exogenous systems in (7.8). With this perspective, the control objective is to design the control input to each of the actual agents with vehicle dynamics in (7.1) so as to track the trajectory generated by the corresponding virtual agent.

Having defined the problem, in the next section we will discuss the nonlinear servomechanism based controller.

7.2 Nonlinear Servomechanism Based Controller Development

We start by noting that if we can force the output of each system to satisfy $y_i = q_i(s_i)$, then we will guarantee that both the formation constraints and the tracking constraints are simultaneously satisfied. With this objective in mind we can redefine the new (error) output of each agent as

$$e_i = \bar{h}_i(x_i, s_i) = y_i - q_i(s_i) = h_i(x_i, \mu_i) - q_i(s_i),$$

and rewrite the system to obtain

$$\begin{aligned} \dot{x}_i &= \bar{f}_i(x_i, s_i, u_i), \\ e_i &= \bar{h}_i(x_i, s_i), 1 \leq i \leq N, \end{aligned} \tag{7.10}$$

where we used the notation $\bar{f}_i$ since we used s_i instead of μ_i in f_i. The objective in this new formulation is to find the local control inputs u_i so that asymptotic convergence of the output error e_i to zero is achieved, i.e., so that

$$\lim_{t \longrightarrow \infty} e_i = 0 \tag{7.11}$$

for all $i = 1, ..., N$. Note that with this formulation the problem of formation control and trajectory following (i.e., both the formation constraints and the tracking constraints satisfied) in the presence of disturbances is equivalent to the problem of decentralized nonlinear output regulation (servomechanism) of the class of systems described with the dynamics in (7.10). Depending on the available information, two different approaches for controller design are possible, which are the full information (namely a state feedback) controller and the error feedback controller. Below we discuss these controllers. We start with the full information controller.

7.2.1 Full Information Controller

In this section we assume that the states s_i of the exosystems are known and briefly discuss conditions on the solvability of the problem. The problem is said to be solvable if there exist smooth controllers $u_i(t)$ such that (7.11) is satisfied or basically all the local output errors go to zero. Let us define the first approximation of the system around the origin as $A_i = \frac{\partial f_i}{\partial x_i}(0,0,0)$, $B_i = \frac{\partial f_i}{\partial u_i}(0,0,0)$, and $C_i = \frac{\partial h_i}{\partial x_i}(0,0)$. Then, based on the results on the nonlinear output regulation problem in [124, 125], necessary conditions for the solvability of the problem can be stated as

(i) Each of the pairs $(A_i, B_i), i = 1, ..., N$, is *controllable* and
(ii) There exist mappings $x_i = \pi_i(s_i)$ and $u_i = c_i(s_i)$ for all $i = 1, ..., N$, with $\pi_i(0) = 0$ and $c_i(0) = 0$, defined in a neighborhood S_i^o of the origin of $\mathbb{R}^{r_i + r}$, respectively, such that

$$\begin{aligned} \frac{\partial \pi_i(s_i)}{\partial s_i} g_i(s_i) &= \bar{f}_i(\pi_i(s_i), s_i, c_i(s_i)), \\ 0 &= h_i(\pi_i(s_i), s_i) - q_i(s_i), 1 \leq i \leq N, \end{aligned} \tag{7.12}$$

for all $s_i \in S_i^o$, respectively.

The equations in (7.12) are called regulator equations or Francis-Byrnes-Isidori (FBI) equations. The vector of the mappings $x_i = \pi_i(s_i)$ is the equation of a manifold on which the vector of the output errors e_i is zero and the vector of the control inputs $u_i = c_i(s_i)$, sometimes called a friend in the literature, is the control input which renders this manifold invariant. Therefore, for the solvability of the problem we need the existence of such a manifold and the corresponding controller.

Given that the states s_i of the exosystems are known and assuming that the nonlinear partial differential equations in Equation (7.12) are solvable for the mappings $\pi_i(s_i)$ and $c_i(s_i)$, the above two conditions are also sufficient for the solvability of the problem. In fact, a full information controller which locally asymptotically achieves $e_i(t) \to 0$ is given by

$$u_i = c_i(s_i) + K_i\big(x_i - \pi_i(s_i)\big), \tag{7.13}$$

where each of the matrices K_i is chosen such that the matrices $(A_i - B_iK_i)$ are Hurwitz for all i. Note that the controller in (7.13) renders the system matrix of the first approximation of the system in (7.10) Hurwitz. Then, by redefining $\tilde{x}_i = x_i - \pi_i(s_i)$ and analyzing the dynamics of the first approximation one can show that $\tilde{x}_i \to 0$ implying that $x_i \to \pi_i(s_i)$ and $u_i \to c_i(s_i)$ which from the relations in (7.12) lead to the satisfaction of the requirements in (7.11). From this respect the term $K_i\big(x_i - \pi_i(s_i)\big)$ in the full information controller in (7.13) renders the zero-error manifold $x_i = \pi_i(s_i)$ locally attractive, whereas the term $c_i(s_i)$ keeps the state x_i of the system on the manifold once it is reached, i.e., $c_i(s_i)$ renders the zero-error manifold $x_i = \pi_i(s_i)$ invariant. The fact that $e_i(t) \to 0$ for all i implies that $y_i = h_i(x_i, \mu_i) \to q_i(s_i)$ and that asymptotically the formation control and collective trajectory tracking objectives are achieved.

Note that the individuals do not need to have information about the other agents in the system. They only need to know the information about the virtual leader and their local inputs (specifying their desired relative position). Note, however, that inherently there is a drawback in this since it cannot guarantee avoidance of collisions between the agents. In order to guarantee avoidance of collisions there is a need to augment (or redesign) the stabilizing controller with a collision avoidance term. One possibility is to use a collision avoidance method based on artificial potential fields. However, discussing such an approach is outside of the scope of this chapter since it needs further investigations.

7.2.2 Error Feedback Controller

In the full information controller it was assumed that the exogeneous state s_i and the mappings $\pi_i(s_i)$ and $c_i(s_i)$ are known. However, sometimes it may not be possible to know s_i or to solve (7.12) for the mappings $\pi_i(s_i)$ and $c_i(s_i)$. For example, if s_i contains unmeasurable disturbances, then it may not be known. Despite that, if we would like to have $e_i(t) \to 0$ as $t \to \infty$, then one may need to develop a decentralized dynamic error feedback controller (i.e., a controller that uses only the local output information e_i) which still achieves the objective.

In order for the error feedback nonlinear servomechanism problem to be solvable we need a few more necessary and sufficient conditions to be satisfied in addition to the conditions (i) and (ii) above required for the solvability of the problem using a full information controller. These conditions can be stated as follows

(iii) For all $i = 1, \ldots, N$, the autonomous systems with outputs

$$\begin{aligned} \dot{s}_i &= g_i(s_i), \\ u_i &= c_i(s_i), \end{aligned}$$

are *immersed* into

$$\begin{aligned}\dot{\xi}_i &= \varphi_i(\xi_i),\\ u_i &= \gamma_i(\xi_i),\end{aligned} \tag{7.14}$$

defined on neighborhoods Ω_i of the origins of $\mathbb{R}^{p_i}$, respectively, in which $\varphi_i(0) = 0$ and $\gamma_i(0) = 0$.

(iv) For all $i = 1, ..., N$, the matrices

$$\Phi_i = \left[\frac{\partial \varphi_i}{\partial \xi_i}\right]_{\xi_i=0} \quad \text{and} \quad \Gamma_i = \left[\frac{\partial \gamma_i}{\partial \xi_i}\right]_{\xi_i=0}$$

are such that each of the pairs

$$\begin{bmatrix} A_i & 0 \\ M_iC_i & \Phi_i \end{bmatrix}, \begin{bmatrix} B_i \\ 0 \end{bmatrix}, \tag{7.15}$$

is *stabilizable* for some M_i and each of the pairs

$$\begin{bmatrix} C_i & 0 \end{bmatrix}, \begin{bmatrix} A_i & B_i\Gamma_i \\ 0 & \Phi_i \end{bmatrix}, \tag{7.16}$$

is *detectable*.

If these conditions are satisfied, the decentralized error feedback nonlinear servomechanism problem is solvable and the local controllers which solve the problem are given by

$$\begin{aligned}\dot{\xi}_i &= \varphi_i(\xi_i) + M_ie_i,\\ \dot{\chi}_i &= \Psi_i\chi_i + L_ie_i,\\ u_i &= \gamma_i(\xi_i) + G_i\chi_i, 1 \le i \le N,\end{aligned} \tag{7.17}$$

where the matrices Ψ_i, L_i, and G_i are chosen such that the matrices

$$\tilde{A}_i = \begin{bmatrix} \bar{A}_i & \bar{B}_iG_i \\ L_i\bar{C}_i & \Psi_i \end{bmatrix},$$

where

$$\bar{A}_i = \begin{bmatrix} A_i & B_i\Gamma_i \\ M_iC_i & \Phi_i \end{bmatrix}, \bar{B}_i = \begin{bmatrix} B_i \\ 0 \end{bmatrix}, \text{ and } \bar{C}_i = \begin{bmatrix} C_i & 0 \end{bmatrix},$$

are Hurwitz. Note that such a triple of matrices, i.e., Ψ_i, L_i, and G_i, always exists. In other words, the above conditions guarantee the existence of such a stabilizing controller. For example, one possible choice is an observer based controller given by [136]

$$\Psi_i = \bar{A}_i - \bar{B}_iK_i - H_i\bar{C}_i, \quad G_i = K_i, \quad L_i = -H_i, \tag{7.18}$$

where K_i and H_i are such that the matrices $(\bar{A}_i - \bar{B}_iK_i)$ and $(\bar{A}_i - H_i\bar{C}_i)$ are Hurwitz.

Note that the controller in (7.17) consists of two parts in parallel. The first part, which is due to the immersion, is called *internal model* or *servocompensator* and is able to asymptotically recover $c_i(s_i)$, whereas the second part is called *stabilizing compensator* and renders the interconnection of the system and the internal model asymptotically stable.

We would like to emphasize here once more that the results described above are local results for the general nonlinear agent/vehicle dynamics given in Equation (7.1). In other words, since they have been derived based on the first approximation of the system dynamics, they do not hold globally, i.e., they may not, in general, hold for all possible initial conditions. If, however, the vehicle dynamics for the agents are linear, then the results will hold globally.

7.3 Formation Reconfiguration

In addition to satisfaction of the formation and tracking constraints another requirement on the system might be to perform different formation maneuvers such as expansion/contraction, rotation, and reconfiguration during motion. To achieve these we use the local exogeneous inputs μ_i. With this objective we assume that the trajectory $q_i(s_i)$ to be tracked by the i^{th} agent can be written as

$$q_i(s_i) = q_l(s) + q_f(\mu_i)$$

where the functions $q_f(\mu_i)$ are appropriately defined and specify the reference trajectories to achieve the needed maneuvers. For example, in a two dimensional space it can be defined as

$$q_f(\mu_i) = \mu_{i1} R(\mu_{i2})$$

where μ_{i1} and μ_{i2} are the local variables (reference inputs or parameters) and $R : \mathbb{R} \longrightarrow \mathbb{R}^2$ is the unity "rotation vector"

$$R(\mu_{i2}) = \begin{bmatrix} \cos(\mu_{i2}) \\ \sin(\mu_{i2}) \end{bmatrix}$$

With this formulation μ_{i1} determines the relative distance of agent i to the virtual leader, whereas μ_{i2} determines the relative angle between its position and the position of the virtual leader in some global coordinate frame (located for example on the virtual leader). Here μ_{i1} can be used for expansion/contraction and topology change (formation reconfiguration) maneuvers, whereas μ_{i2} can be used for rotation and topology change maneuvers. In order to illustrate the basic idea, below we will stick to the two dimensional case keeping in mind that similar ideas hold for higher dimensional models as well.

If the desired formation is fixed, then μ_{i1} and μ_{i2} can be chosen as constants (parameters), i.e., for a fixed formation the dynamics of μ_{i1} and μ_{i2} can be stated as

$$\begin{aligned} \dot{\mu}_{i1} &= 0, \\ \dot{\mu}_{i2} &= 0, \quad i = 1, \ldots, N, \end{aligned} \tag{7.19}$$

and $\mu_{i1}(t) = \mu_{i1}(0)$ and $\mu_{i2}(t) = \mu_{i2}(0)$ for all $t \geq 0$ and for all $i = 1,...,N$. Otherwise they might need to be dynamically varied appropriately in order to achieve the desired maneuver. It is also possible to assume that they are fixed and design the controller accordingly. Then, occasionally when there is a need for formation maneuver switch their values. We will take this last approach and develop the controller with the assumption that $\mu_{i1} = \bar{\mu}_{i1}$ and $\mu_{i2} = \bar{\mu}_{i2}$, where $\bar{\mu}_{i1}$ and $\bar{\mu}_{i2}$ are constants and occasionally switch their values as desired. However, we will also briefly discuss how they can be dynamically varied to achieve the desired formation maneuvers.

Assuming that the tracking constraints are given by (7.7) in order for them to be satisfied we need to have

$$\|q_i(s_i) - q_l(s)\| = \|q_f(\mu_i)\| = d_{il},$$

which for the two dimensional case above is reduced to the form

$$\|\mu_{i1} R(\mu_{i2})\| = d_{il}.$$

This, on the other hand, results in the requirement

$$\mu_{i1} = \bar{\mu}_{i1} = d_{il}, \;\; i = 1, \ldots, N,$$

which is obtained using the fact that $\|R(\mu_{i2})\| = 1$. Similarly, in order for the formation constraints to be satisfied we need

$$\|q_i(s_i) - q_j(s_j)\| = \|q_f(\mu_i) - q_f(\mu_j)\| = d_{ij},$$

which for the two dimensional case above becomes

$$\|\mu_{i1} R(\mu_{i2}) - \mu_{j1} R(\mu_{j2})\| = d_{ij}.$$

Then, using the cosine theorem we have

$$\|d_{il} R(\mu_{i2}) - d_{jl} R(\mu_{j2})\| = \sqrt{d_{il}^2 - 2 d_{il} d_{jl} \cos(\mu_{i2} - \mu_{j2}) + d_{jl}^2} = d_{ij},$$

from which $\mu_{i2} - \mu_{j2}$ can be expressed as

$$\mu_{i2} - \mu_{j2} = \cos^{-1}\left(\frac{d_{il}^2 + d_{jl}^2 - d_{ij}^2}{2 d_{il} d_{jl}}\right).$$

If the constraints are feasible, then for fixed formations there exist constant values $\bar{\mu}_{i1} = d_{il}$ for all $i = 1, ..., N$, and $\bar{\mu}_{i2}$ and $\bar{\mu}_{j2}$ for all $i = 1, ..., N$, and $j = 1, ..., N$, such that the above conditions are satisfied. Since the cosine inverse has two possible solutions we need the relative angle and the constraints of at least two agents j and k in order to be able to uniquely determine the desired relative angle of agent i.

Below we will briefly discuss how to generate different formation maneuvers such as expansion/contraction, rotation, or topology change or basically reconfiguration of the formation. In these discussions we will use the two dimensional case above. However, one can easily develop counterparts for higher dimensional spaces as well.

7.3.1 Expansion/Contraction

Let us first consider the expansion/contraction maneuver while keeping constant relative angles with respect to the virtual leader. Assume that at some time t_1 an expansion or contraction maneuver is requested from the swarm (possibly by an external operator). In other words, assume that at time t_1 the desired inter-agent distances $d_{ij}(t)$ and $d_{il}(t)$ are reset as

$$\begin{aligned} d_{il}(t_1^+) &= \rho_1 d_{il}(t_1^-), \ \ \forall i, \\ d_{ij}(t_1^+) &= \rho_1 d_{ij}(t_1^-), \ \ \forall i, j, \end{aligned}$$

where $\rho_1 > 0$ is a positive constant determining the amount of expansion/contraction and satisfies $\rho_1 > 1$ for expansion and $\rho_1 < 1$ for contraction. Note that for proper expansion/contraction all the desired distances are reset by the same relative amount. Then, to achieve the maneuver, the easiest method is to reset the values of the "magnitude parameter" μ_{i1} accordingly as

$$\mu_{i1}(t_1^+) = \rho_1 \mu_{i1}(t_1^-), \ \ \forall i.$$

As was mentioned above for this case the controller is designed as if the value of μ_{i1} is constants, i.e., assuming that $\dot{\mu}_{i1} = 0$. Therefore, after each switch there will be a short transient during which the system will expand or contract to the new desired relative positions.

7.3.2 Rotation

Now let us consider the rotation maneuver while keeping constant relative distances with respect to the virtual leader, i.e., keeping constant formation size. Analogous to the expansion/contraction maneuver, assume that at some time t_3 a rotation maneuver is requested from the swarm. In other words, assume that at time t_3 the desired relative angles (with respect to some global coordinate system), say $\theta_{il}(t)$, are reset as

$$\theta_{il}(t_3^+) = \theta_{il}(t_3^-) + \rho_2, \ \ \forall i,$$

where ρ_2 is a rotation parameter which satisfies $\rho_2 > 0$ for rotation in the counter clockwise direction and $\rho_2 < 0$ for rotation in the clockwise direction. To achieve the desired rotation, again the simplest method is to reset the values of μ_{i2} as

$$\mu_{i2}(t_3^+) = \mu_{i1}(t_3^-) + \rho_2, \ \ \forall i.$$

Similar to the expansion/contraction maneuver, for this case there will be a short transient after each reset. However, the controller is simple since it is designed based on the assumption that μ_{i2} is constant.

7.3.3 Topology Change

We would like to briefly mention here also that with appropriate simultaneous resetting or variation of the values of both μ_{i1} and μ_{i2} for all the agents, it is possible to achieve formation reconfiguration (i.e., switching from one formation to another). For example, in a swarm of agents to switch from any (e.g., triangular) formation to a line formation we can reset μ_{i1} and μ_{i2} from their original values to $\mu_{i1} = id$ and $\mu_{i2} = \pi$ for all i, where d is the desired inter-agent distance in the line formation. In other words, as the system is moving in a given formation, similar to the previous cases we can just "reset" the reference trajectories $q_i(s_i)$ based on the next desired formation. Then, the nonlinear servomechanism based controller will guarantee that the new formation is achieved after a short transient.

Note that it is possible to vary the dynamics of the local inputs μ_i and achieve smoother transitions. In that case, in order to satisfy the stated assumptions the dynamics should be stable. Another option is to have piecewise constant inputs to the local exosystems in (7.2). We will not pursue these options here. However, the reader should be aware that such options exist.

7.4 Illustrative Examples

In this section we will provide few simulation examples in order to illustrate the effectiveness of the method discussed in the preceding sections. Assume that we have a system consisting of $N = 6$ agents, with point mass dynamics moving in a two dimensional space. In other words, we have

$$\begin{aligned}\dot{x}_i &= v_i,\\ \dot{v}_i &= \mu_{i3} + u_i,\\ y_i &= x_i,\end{aligned}$$

where $x_i \in \mathbb{R}^2$ is the position, $v_i \in \mathbb{R}^2$ is the velocity, $\mu_{i3} \in \mathbb{R}^2$ represents the local external disturbances acting on the system, and $u_i \in \mathbb{R}^2$ is the control (force) input (without loss of generality we assumed unity mass for all the agents). Note that we used the notation μ_{i3} for the local external disturbances since we reserved $\mu_{i1} \in \mathbb{R}$ and $\mu_{i2} \in \mathbb{R}$ for denoting the local external reference inputs to be used for generating different formation maneuvers. Point mass dynamics may seem to be simple. However, they are suitable for illustrating the procedure. Moreover, note that such dynamics are being used by many researchers for formation control studies. Let us assume that the agents are required to move in some predefined formation pattern. In particular, we will consider the problem of moving in an equilateral triangle formation of the form shown in Figure 7.2 along a circular trajectory. Note that the formation constraints for this type of formation are of type of (7.4) with desired inter-agent distances d_{ij} depending on the relative position of the agents in the formation as well as the size of the triangle. In particular, in the simulations below we require an equilateral triangle formation with side lengths equal to $2\sqrt{3}$. This, depending on the relative positions of the agents in the formation, results in desired inter-agent distances $d_{ij} = \sqrt{3}$, $d_{ij} = 3$, and $d_{ij} = 2\sqrt{3}$. Initially we illustrate the constant formation case, following which perform formation maneuvers as well. The virtual leader dynamics which can generate circular trajectory can be expressed by the linear system of the form

$$\begin{aligned} \dot{s} &= G_l s, \\ y_l &= C_l s, \end{aligned} \tag{7.20}$$

where $s \in \mathbb{R}^4$ and

$$G_l = \begin{bmatrix} 0 & -\beta & 0 & 0 \\ \beta & 0 & 0 & 0 \\ 0 & 0 & 0 & -\beta \\ 0 & 0 & \beta & 0 \end{bmatrix}, \; C_l = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Here β is a parameter which determines the frequency of rotation around the circle. Note that the dynamics of the virtual leader in the two dimensions of the output space are decoupled from each other. This will ease the development of the controller since one can design the controller for each dimension separately using the lower order decoupled sub-system. Assume that the parameter $\beta = 1$ and the initial state of the virtual leader is $s(0) = [10, 0, 10, 10]^\top$. Also, let us fix the position of the virtual leader at the center of the triangle. Then, from the above formation constraints the required (fixed) relative positions for the agents with respect to the leader can be determined as

$$\mu_{11} = \mu_{21} = \mu_{31} = 1, \quad \mu_{41} = \mu_{51} = \mu_{61} = 2,$$

and the corresponding relative angles can be set as

$$\mu_{12} = \frac{\pi}{2}, \; \mu_{22} = -\frac{\pi}{6}, \; \mu_{32} = \frac{7\pi}{6}, \; \mu_{42} = \frac{\pi}{6}, \; \mu_{52} = \frac{5\pi}{6}, \; \mu_{62} = \frac{3\pi}{2},$$

which result in

$$\begin{aligned} q_f(\mu_1) = \mu_{11} R(\mu_{12}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; q_f(\mu_2) = \mu_{21} R(\mu_{22}) = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ -\frac{1}{2} \end{bmatrix}, \\ q_f(\mu_3) = \mu_{31} R(\mu_{32}) = \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ -\frac{1}{2} \end{bmatrix}, \; q_f(\mu_4) = \mu_{41} R(\mu_{42}) = \begin{bmatrix} \sqrt{3} \\ 1 \end{bmatrix}, \\ q_f(\mu_5) = \mu_{51} R(\mu_{52}) = \begin{bmatrix} -\sqrt{3} \\ 1 \end{bmatrix}, \; \text{and} \; q_f(\mu_6) = \mu_{61} R(\mu_{62}) = \begin{bmatrix} 0 \\ -2 \end{bmatrix}. \end{aligned}$$

Using these the desired position trajectory of agent i is given by

$$q_i(s_i) = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} + \mu_{i1} R(\mu_{i2}).$$

As discussed before, one can assume the values of the local reference inputs μ_{i1} and μ_{i2} are constant and still achieve formation maneuvers by switching their values.

However, for that case during formation maneuvers after the instants at which the values of the local inputs switch there will be short transient. Nevertheless, it is expected that the new formation will be achieved in a short period of time. Therefore, we will design the controller assuming that the derivatives of the local reference inputs are zero or basically assuming that (7.19) is satisfied.

Under this assumption the manifold and the friend equations are simple and given by

$$\begin{aligned}\pi_{i1}(s_i) &= \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} + \mu_{i1} R(\mu_{i2}), \\ \pi_{i2}(s_i) &= \beta \begin{bmatrix} s_2 \\ s_4 \end{bmatrix}, \\ c_i(s_i) &= -\mu_{i3} - \beta^2 \begin{bmatrix} s_1 \\ s_3 \end{bmatrix}.\end{aligned}$$

Note also that since the functions $q_i(s_i)$ are independent from the external disturbances μ_{i3}, the controller developed using the procedure described will automatically achieve disturbance rejection.

Using the above definitions the full information controller is given by

$$u_i = c_i(s_i) - \alpha_1 (x_i - \pi_{i1}(s_i)) - \alpha_2 (v_i - \pi_{i2}(s_i))$$

which for the given values of the manifold equations is given by

$$u_i = -\mu_{i3} - \beta^2 \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} - \alpha_1 \left(x_i - \mu_{i1} R(\mu_{i2}) - \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} \right) - \alpha_2 \left(v_i - \beta \begin{bmatrix} s_2 \\ s_4 \end{bmatrix} \right).$$

For the stabilizing part of the controller we chose the parameters as $\alpha_1 = 2$ and $\alpha_2 = 3$, which lead to the poles at -1 and -2. Figure 7.1 shows the response of the system for about 10 seconds for the case of constant desired formation. For this simulation we assumed that the local external disturbances acting on the system are constant as well satisfying

$$\dot{\mu}_{i3} = 0$$

for all i. In particular, assuming that $\mu_{i3} = \bar{\mu}_{i3}$ is the constant value of the disturbance we used $\bar{\mu}_{i3} = [10, 5]^\top$. As one can see from the figure, initially the individuals are not in the required formation; however, they form the formation very fast (exponentially fast) and follow the required trajectory keeping the formation. This is achieved despite the disturbance present in the system. One drawback is that the full information controller assumes that the external inputs $s_i = [s, \mu_i]^\top$ are known including the disturbances μ_{i3} and uses this information. However, the disturbance may not necessarily be measurable. For that case, one can use the error feedback controller which does not require the knowledge of s_i and still guarantees satisfaction of the constraints.

Figure 7.2 shows the relative positions of the agents after 10 seconds. The agent positions are shown by small circles and are located at the desired locations on the sides and corners of the triangle. The virtual leader position is shown by a circle and a star and is located at the center of the triangle as desired.

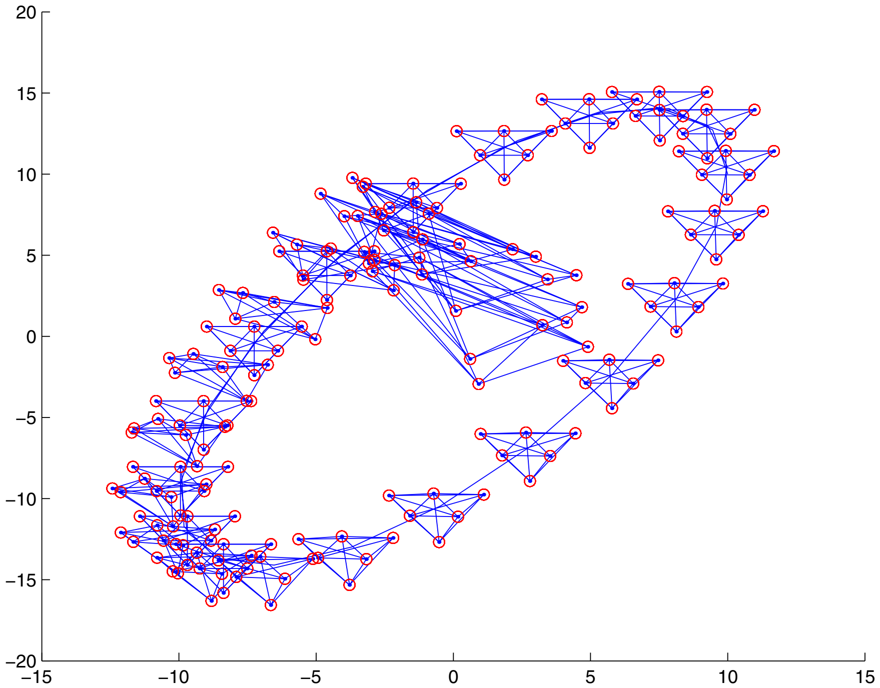


Fig. 7.1. The response for the full information controller with $\alpha_1 = 2$ and $\alpha_2 = 3$.

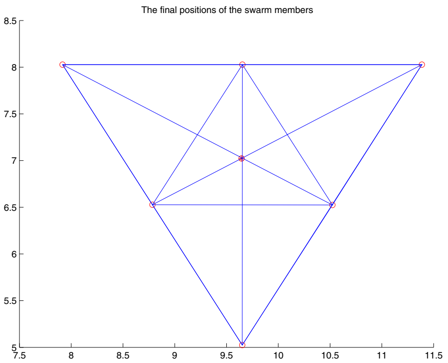
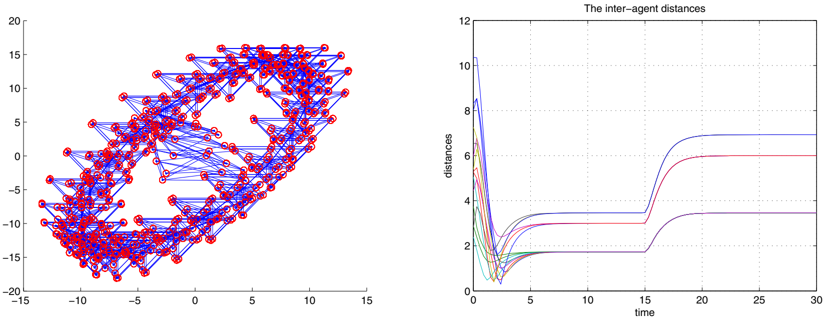


Fig. 7.2. The final positions of the agents in the formation.

Next, let us consider different formation maneuvers using the full information controller. Figure 7.3 shows the trajectories of the agents as well as the inter-agent distances during a formation expansion maneuver. The agents start from random initial positions as before and form the required triangle while tracking the circular trajectory of the virtual leader. Then at $t = 15s$ a request/command for expanding the formation to twice its size is received and as one can see from the figure the inter-agent distances are increased accordingly. The larger formation is achieved quickly.

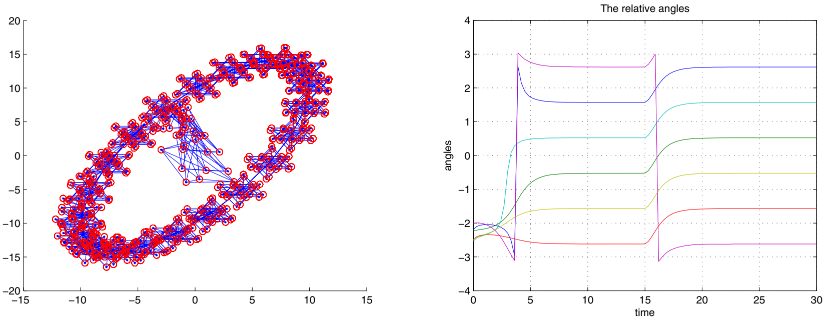


(a) Agent trajectories.

(b) Inter-agent distances.

Fig. 7.3. Agent trajectories and inter-agent distances during an expansion maneuver.

Figure 7.4 shows the trajectories of the agents and the relative angles of the agents with respect to the virtual leader during a formation rotation maneuver. Similarly to the expansion maneuver the agents start the rotation maneuver at time $t = 15s$. The desired amount of rotation is $\frac{\pi}{3}$ which results in the fact that every agent moves to the position of its preceding agent. As one can see from Figure 7.4(b) the maneuver is successfully achieved as expected.



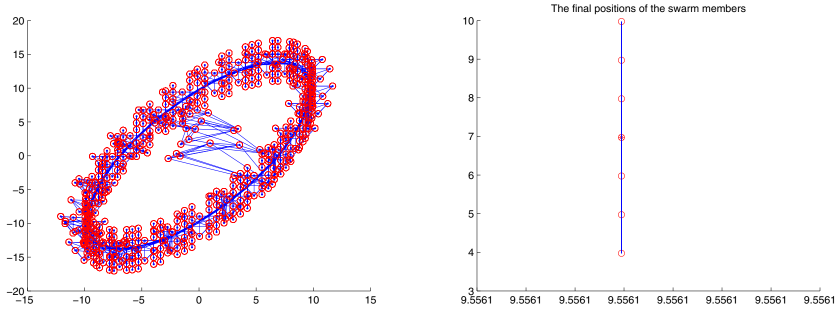
(a) Agent trajectories.

(b) Relative angles.

Fig. 7.4. Agent trajectories and relative angles of the agents to the virtual leader during a rotation maneuver.

Next, let us consider a formation reconfiguration maneuver. In particular let us assume that the agents are required to switch from the triangle formation to a line formation. Similarly to the above two cases the maneuvers starts at at time $t = 15s$.

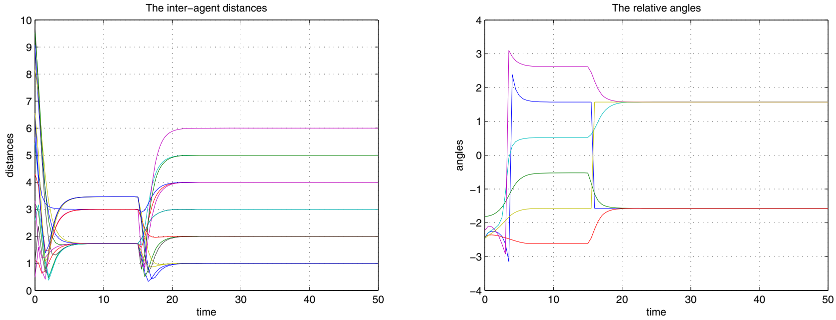
Figure 7.5 shows the agent trajectories and the final positions of the agents around $t = 50s$. The leader is at the center of the agents and shown with a circle and a star whereas the agents are spread on both sides of the leader with equal steps of 1 unit. Figure 7.6 shows the inter-agent distances and the relative angles of the agents to the virtual leader during the maneuver. As one can see from the figure the relative angles with respect to the leader frame converge to either $-\frac{\pi}{2}$ or $+\frac{\pi}{2}$ depending on whether the agent is on one side of the leader or the other. Similarly, the inter-agent distances converge to the expected/desired values in the line formation.



(a) Agent trajectories.

(b) Final relative positions.

Fig. 7.5. Agent trajectories and final relative positions of the agents during a reconfiguration maneuver.



(a) Inter-agent distances.

(b) Relative angles.

Fig. 7.6. Inter-agent distances and relative angles of the agents to the virtual leader during a reconfiguration maneuver.

Before developing the error feedback controller note once more that for this problem since all the local inputs are constant the dynamics of the system are decoupled for the two dimensions of the output space. Therefore, to ease the problem we will also decouple the problem of the servocontroller design. In other words, we will consider each dimension as a separate single-input-single-output system and design the controller for that dimension. Actually, we will design a controller for one dimension only and use the same structure for the other dimension too, since the dynamics in both dimensions have the same structure.

Considering only the single dimensional subsystem generating the controller (i.e., the friend) we have

$$\begin{aligned}
\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{\mu}_{i1} \\ \dot{\mu}_{i2} \\ \dot{\mu}_{i31} \end{bmatrix} &= \begin{bmatrix} 0 & -\beta & 0 & 0 & 0 \\ \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \mu_{i1} \\ \mu_{i2} \\ \mu_{i31} \end{bmatrix} \\
c_{i1}(s_i) &= -\mu_{i31} - \beta^2 s_1
\end{aligned}$$

where μ_{i31} is the first component of μ_{i3}. Using the transformation

$$\tau_i(s_i) = \begin{bmatrix} -\mu_{i1} - \beta^2 s_1 \\ \beta^3 s_2 \\ \beta^4 s_1 \end{bmatrix}$$

the system is immersed into

$$\begin{aligned}
\dot{\tau}_i(s_i) &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\beta^2 & 0 \end{bmatrix} \tau_i(s_i), \\
c_{i1}(s_i) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tau_i(s_i),
\end{aligned}$$

which is in observable canonical form. This system is the immersion in (7.14). Since the exosystem dynamics (i.e., the dynamics of the virtual leader and the local external inputs and disturbances) are linear, the immersion is also linear. Note that for this system taking again only the decoupled one dimension we have

$$\begin{aligned}
A_i &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C_i = \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\Phi_i &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\beta^2 & 0 \end{bmatrix}, \text{ and } \Gamma_i = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.
\end{aligned}$$

By choosing

$$M_i = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

the interconnection of the internal model (i.e., the immersion) and system is stabilizable for any nonzero constant β.

Let the vector P_1 denote the desired pole locations of the closed loop system and P_2 denote that of the observer in the observer based controller in (7.18). Then, the stabilizing controller matrices can be obtained by using the Matlab command `place` as

$$\begin{aligned} L_i &= place(\bar{A}_i, \bar{C}_i^\top, P_2)^\top, \\ G_i &= place(\bar{A}_i, \bar{B}_i, P_1), \\ \Psi_i &= \bar{A}_i - \bar{B}_i G_i - L_i \bar{C}_i, \end{aligned}$$

where $\bar{A}_i, \bar{B}_i$, and $\bar{C}_i$ are as defined in the preceding section. For the pole locations at $P_1 = P_2 = [-1, -1.5, -2, -2.5, -3]$ the stabilizing controller matrices are given by

$$\Psi_i = \begin{bmatrix} -10 & 1 & 0 & 0 & 0 \\ -61.25 & -10 & -22.5 & -27.5 & -62.5 \\ -65.25 & 0 & 0 & 1 & 0 \\ -62.5 & 0 & 0 & 0 & 1 \\ -36.75 & 0 & 0 & -1 & 0 \end{bmatrix}, \quad G_i = \begin{bmatrix} 37.75 \\ 10 \\ 23.5 \\ 27.5 \\ 62.5 \end{bmatrix}, \text{ and}$$
$$L_i = \begin{bmatrix} 10 \; 23.5 \; 65.25 \; 62.5 \; 37.75 \end{bmatrix}.$$

The plot in Figure 7.7 shows the motion of the system with the above error feedback controller for about 50 seconds. Note that compared to the full information case it takes more time for the system to converge to the desired formation and to follow the trajectory of the virtual leader. This is due to the fact that it takes some time for the observer states to converge and therefore to generate the appropriate control input. Nevertheless, it still converges in a short period of time. The plot in Figure 7.8 shows the trajectories of the six agents for the last 12.5 seconds of the above case. As one can see, they have converged to the desired formation and follow the desired trajectory. The final positions of the agents are exactly the same as shown in Figure 7.2 for the full information controller case and therefore are not shown here.

Next let us consider different formation maneuvers using the error feedback controller. Figure 7.9(a) shows the inter-agent distances during an expansion maneuver, whereas Figure 7.9(b) shows the relative angles during a rotation maneuver. In both cases the maneuver starts at $t = 60s$ some time after the initial formation is settled. For the expansion maneuver the formation is expected to expend to twice of initial size, whereas for the rotation maneuver it is required to rotate $\frac{\pi}{2}$ radians in the counter clockwise direction. As one can see from the figure after short transient the required maneuvers are achieved. The other plots are very similar to the ones presented above and therefore are not presented here.

For the formation reconfiguration (topology change) maneuver case we consider the same formation as in the full information controller. In other words, the swarm has to move from an equilateral triangle formation to a line formation. The maneuver again starts at $t = 60s$. Figure 7.10(a) shows the inter-agent distances, whereas Figure 7.10(b) shows the relative angles during the maneuver. As one can see from the figure, they converge to the expected values. In the new formation the relative distances between the agents vary from one unit to six units depending on their

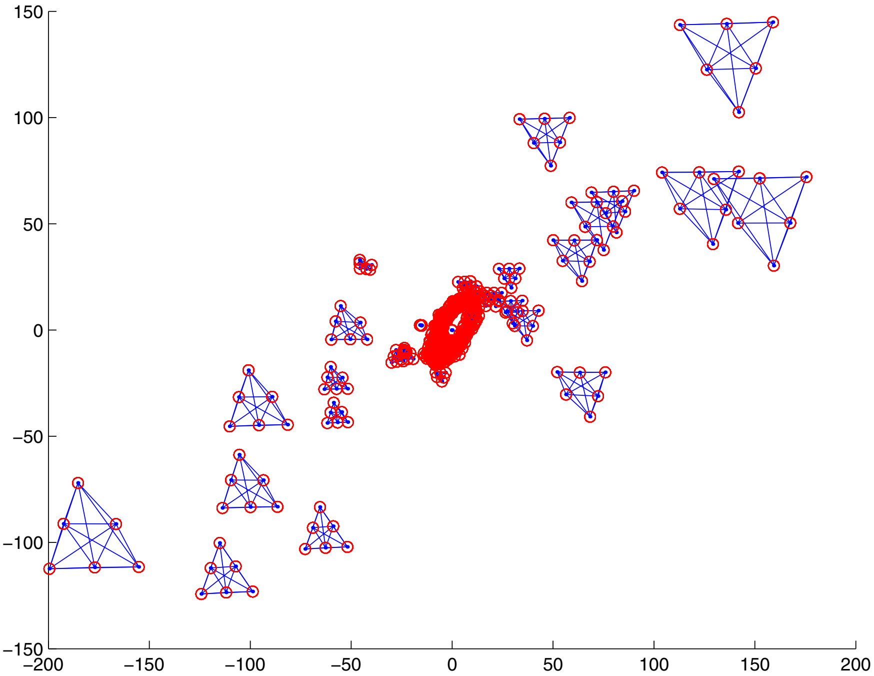


Fig. 7.7. The response for the error feedback controller for about 50 seconds.

relative location in the new formation, whereas the relative angles with respect to the leader converge to either $-\frac{\pi}{2}$ or $+\frac{\pi}{2}$ depending on which side of the leader they are located in. The other plots for the maneuver, i.e., the plots of the trajectories of the agents and the initial and the final formation as similar to those shown for the previous cases and therefore not shown here.

As one can notice the transient for the error feedback case is more severe compared to the full information controller case. The main reason for that is due to the observer based controller. One can see also from the plot in Figure 7.7 that collisions between agents might be possible during transient. As was mentioned before in its current form the procedure does not guarantee collision avoidance. This problem could be overcome by augmenting the controller with a potential functions based collision avoidance term, which is active only when the agents approach a collision and vanishes at steady state (when the agents have already formed the formation and the output error has converged to zero). This was done in [79] for formation control of nonholonomic mobile robots. Another approach could be to first design the collision avoidance controller (based on some appropriate method) and augment it to the system dynamics and then design the servomechanism based controller (discussed here) to the augmented dynamics. However, achieving rigorous conclusions on these issues may need careful consideration and further research.

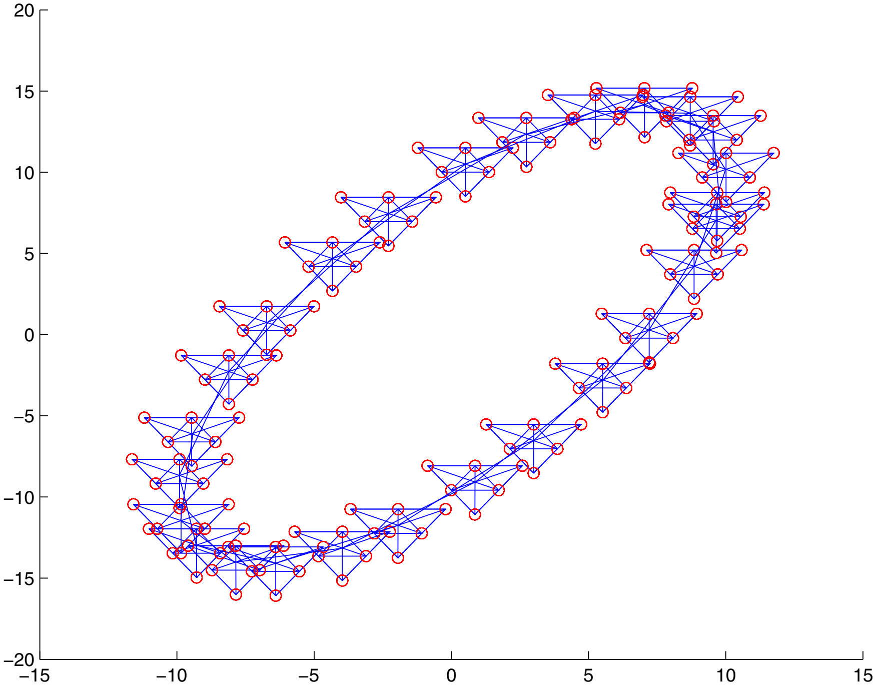
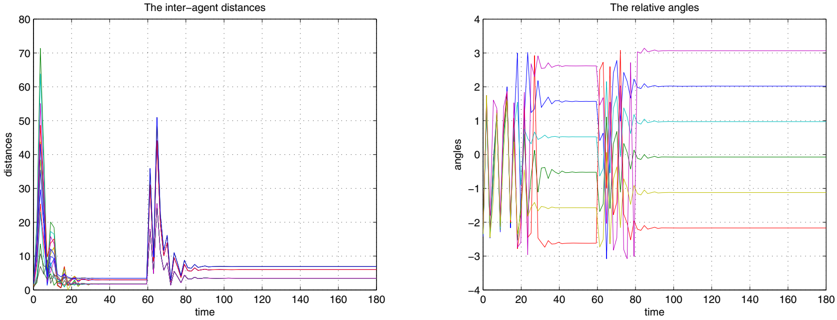


Fig. 7.8. The response for the error feedback controller after 37.5 seconds.



(a) Inter-agent distances (translation).

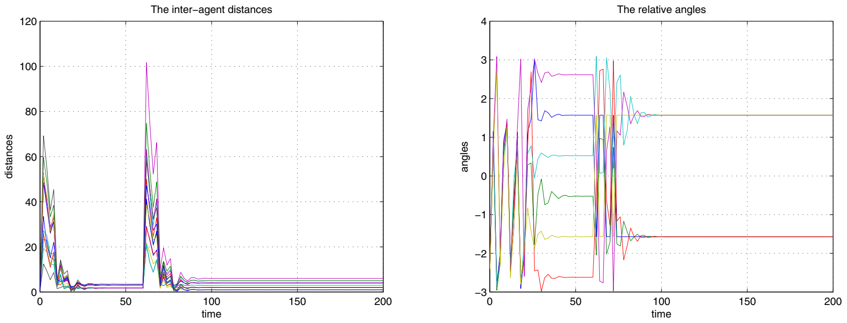
(b) Relative angles (rotation).

Fig. 7.9. Agent trajectories for a translation and relative angles for a rotation maneuvers.

7.5 Further Issues

7.5.1 Extensions and Generalizations

The reader should note that the results discussed in this chapter are local results for the general nonlinear vehicle (i.e., agent) dynamics given in (7.1). In other words,



(a) Inter-agent distances.

(b) Relative angles.

Fig. 7.10. Inter-agent distances and relative angles of the agents to the virtual leader during a reconfiguration maneuver.

they do not, in general, hold for all possible initial conditions. If, however, the vehicle dynamics for the agents are linear, then the results will hold globally. In the output regulation literature there are global or semiglobal results available for a class of nonlinear systems [137, 217–219]. If the agent/vehicle dynamics of the swarm members are of the form which could be transformed to one of the forms considered in one of [137, 217–219], then using the corresponding approach in the mentioned articles with any necessary modifications to fit within the formation control problem here one can design the controller which will achieve following the leader while keeping the formation globally or semiglobally.

The full information controller in Equation (7.13) is based on the assumption that s_i is measurable and we know the analytic expressions of $\pi_i(s_i)$ and $c_i(s_i)$. In order to find the analytic expressions for $\pi_i(s_i)$ and $c_i(s_i)$, one needs to solve the partial differential equations in (7.12), which may not be always possible. However, one can always approximate these mappings arbitrarily closely [38, 122]. If one cannot solve for $\pi_i(s_i)$ and $c_i(s_i)$, but still would like to use the full information controller in (7.13), then he or she can arbitrarily closely approximate them by using polynomials or neural networks (or any other universal approximators) as was done in [38, 122]. This, however, brings the drawback that convergence of the output errors $e_i(t)$ to zero is not guaranteed anymore. Nevertheless, as was shown in [38, 122] with appropriate choice of the approximator, regulation with arbitrary accuracy can be achieved. In other words, the output errors $e_i(t)$ are still guaranteed to be bounded with bounds being proportional to the approximation error. Moreover, better approximation will lead to a smaller tracking error and the tracking error can be also made arbitrarily small with the appropriate choice of the approximator. We believe that extension of this approach to the context of the formation control problem considered here is quite possible. However, analyzing the effects of the output errors may need careful further investigation.

The assumption that the system in (7.5) is neutrally stable limits the procedure to tracking the class of constant and periodic (e.g., sinusoidal) trajectories. However, note that it still covers many trajectories which may be of interest. Such reference trajectories can be found in many practical applications such as orbiting satellites around the earth, agents guarding an object, etc. Moreover, it is possible to generate and track more complicated trajectories. One approach to this is to apply appropriate switching in the exosystem generating the external dynamics and in the controller as was discussed in [80, 83]. Another approach is to include a piecewise constant input to the exosystem (i.e., virtual leader) dynamics in (7.5). To see this note that the output of a linear neutrally stable system with piecewise constant inputs (for which the results here directly follow) can generate many trajectories by appropriate choice of the input. In particular, note that the output of a system which is a chain of d integrators is a *spline* of degree d [141]. Given any smooth trajectory, it is possible to approximate it using splines [255]. Then, choosing an appropriate sequence of piecewise constant inputs to the virtual leader dynamics (chosen as a set of cascaded integrators) it might be possible to generate these splines (and therefore the desired trajectory). The only shortcoming for this case is that we need the switching between the values of the input to be slow enough (implying that the reference trajectory is smooth enough) so that tracking can be achieved. Here slow enough and smooth enough will depend on the time constants and the speed of the response of the agent dynamics in (7.1).

7.5.2 For Further Reading

The output regulation problem for linear systems was studied extensively in the 70's in [50–52, 75–77] and other related papers. Output regulation of nonlinear systems was first pursued by Huang and Rugh [121] for systems with constant exogenous signals and by Isidori and Byrnes [124, 125]. Other relevant work include [38, 122, 123, 253] which approximate the solutions of the regulator equations. Results on robust regional, semiglobal, and global regulation of nonlinear systems or regulation in systems with parameterized exosystems and adaptive internal models can be found in [137, 217, 219, 220]. Recent works on decentralized or switched output regulation can be found in [83, 96, 260].

In [96] the decentralized output regulation problem for a class of nonlinear systems including interconnected systems was analyzed in the framework of [124]. Note that the system here is a special case of the systems considered there. Therefore, the results in this chapter rely heavily on the results in [124]. One issue to be noted, however, is that despite the fact that the controller design presented here is based on decentralized output regulation results in [96], the existence of the virtual leader brings some kind of centralization to the controller. In the decentralized output regulation framework the controller of each subsystem of the overall system (which corresponds to each agent in the case here) uses only local information. Here, a similar procedure is used; however, there is coupling between the agents due to the virtual leader dynamics. This may not be desirable in some applications and may seem to be a shortcoming. Note, however, that many current approaches contain similar centralization (see for example [13]).

The notion of a virtual leader for formation control was introduced in [68]. The formation control problem discussed in this chapter is very similar to that considered in [68]. However, the approach here to the solution of the problem is completely different. There the authors use formation function (which is in a sense an artificial potential function) which attains unique global minimum at the desired formation and gradient descent (i.e., motion along the negative gradient of the formation function) to achieve the desired formation. In contrast, here we use techniques from nonlinear output regulation literature, which are fundamentally different. This also allows us not only to track the virtual leader and to form the desired formation, but also to reject constant or periodic disturbances, whereas no issue of disturbances is discussed in [68]. For a work on achieving formation maneuvers using a different method the reader can consult [187]. Recently, as an extension of the results in this chapter, adaptive internal models have been also employed for the formation control and reconfiguration problem. A work in this direction can be found in [108]. A work which considers the same problem of formation control and trajectory tracking using an alternative approach or basically adaptive fuzzy systems can be found in [64].

Part III

Discrete Time Swarms

8

One-Dimensional Discrete-Time Swarm

8.1 The Swarm Model

The simplest case which can be considered in a discrete-time setting is the case in which a swarm moves in one-dimensional space. In this chapter we will consider this case. First, we describe the model of a single agent. Then, we present the one-dimensional swarm model (i.e., when many agents are arranged next to each other on a line). The model we consider also allows for asynchronous operation and time delays in neighbor position sensing. Therefore, the analysis of the swarm dynamics is relatively difficult. For this reason, we first analyze the dynamics of the swarm under the assumption of synchronous operation and perfect sensing (without time delays). Then, we build on the results obtained for this case and investigate the dynamics of the asynchronous swarm with sensing delays.

8.1.1 Agent Model

Consider a swarm which moves in one-dimensional space and consists of agents represented with a schematic shown in Figure 8.1. Each agent is assumed to have a *driving device* for performing its movements and *neighbor position sensors* for sensing the position of the adjacent (left and right) neighbors. It is assumed that

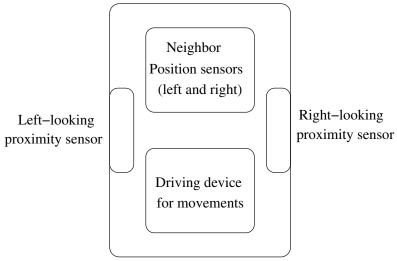


Fig. 8.1. Single agent representation.

there is no restriction on the range of these sensors. In other words, it is assumed that they can provide the accurate position of the neighbors of the given agent even if the neighbors are far away. In addition to the neighbor position sensors it is assumed that each agent also has two close *proximity sensors* on both sides (left and right). These sensors have sensing range of $\varepsilon > 0$ and can sense *instantaneously* in this proximity. Therefore, if another agent reaches an ε distance from it, then this will be *instantaneously* known by both of the agents. However, if the neighbors of the agent are out of the range of the proximity sensor, then it will return an infinite value (i.e., $-\infty$ for the left sensor and $+\infty$ for the right sensor) or some large number which will be ignored by the agent. The use of this sensor is to avoid collisions with the other agents in the swarm.

In the next section we describe the interaction model between the agents with these properties or basically the model of a swarm composed of agents described in this section arranged on a line.

8.1.2 One-Dimensional Swarm Model

Consider a discrete-time one-dimensional swarm described by the model

$$\begin{aligned} x_1(t+1) &= x_1(t),\ \forall t, \\ x_i(t+1) &= \max\Big\{x_{i-1}(t)+\varepsilon, \min\big\{p_i(t), x_{i+1}(t)-\varepsilon\big\}\Big\},\ \forall t \in T_i,\ i=2,\ldots,N-1, \\ x_N(t+1) &= \max\Big\{x_{N-1}(t)+\varepsilon, p_N(t)\Big\},\ \forall t \in T_N, \end{aligned} \tag{8.1}$$

where $x_i(t),\ i=1,\ldots,N$, represents the state (position) of agent i at time t and $T_i \subseteq T = \{0,1,2,3,\ldots\}$ is the set of time instants at which agent i updates its position. At the other time instants it is assumed that agent i is stationary. In other words, it is assumed that

$$x_i(t+1) = x_i(t),\ \forall t \notin T_i \text{ and } i=2,\ldots,N. \tag{8.2}$$

The variables $p_i(t),\ i=1,\ldots,N$, represent the *intended next positions* or *desired next way points* of the agents and are given by

$$p_i(t) = x_i(t) - g\big(x_i(t) - c_i(t)\big),\ i=2,\ldots,N, \tag{8.3}$$

where $g(\cdot)$ is an attraction/repulsion function to be discussed below. The objective of agent i is to move to position $p_i(t)$. If there are not collision situations, then individual i will move to $p_i(t)$, otherwise it will stop at the safe distance ε form its neighbor. Note that although the dynamics in (8.1) are discrete, it is assumed that the agents traverse the path between their current positions $x_i(t)$ and intended next positions $p_i(t)$ and therefore, they can detect collision situations and stop if there are any. This also prevents the agents from jumping to the other side of their neighbors. The quantities $c_i(t)$ in (8.3) represent the *perceived centers* or *perceived midpoints* of the adjacent neighbors of individual i. In other words, we have

$$c_i(t) = \frac{1}{2}\left[x_{i-1}(\tau^i_{i-1}(t)) + x_{i+1}(\tau^i_{i+1}(t))\right], \ \forall i = 2, \ldots, N-1,$$

and

$$c_N(t) = \frac{1}{2}\left[x_{N-1}(\tau^N_{N-1}(t)) + d + x_N(t)\right],$$

where the constant d represents the *comfortable inter-agent distance*. The variable $\tau^i_j(t), j = i-1, i+1$, is used to represent the time index at which agent i obtained position information of its neighbor j. It satisfies $0 \le \tau^i_j(t) \le t$ for $t \in T_i$, where $\tau^i_j(t) = 0$ means that agent i did not obtain any position information about agent j so far (it still has the initial position information), whereas $\tau^i_j(t) = t$ means that it has the current position information of agent j. The difference $(t - \tau^i_j(t)) \ge 0$ can be viewed as a sensing delay or a communication delay in obtaining the position information of agent j by agent i.

Note that in the swarm model in (8.1) it is implicitly assumed that $x_{i+1}(t) - x_{i-1}(t) > 2\varepsilon$. Later we will show that this always will be the case provided that $x_{i+1}(0) - x_{i-1}(0) > 2\varepsilon$ (which is satisfied by assumption). Also, notice that the first agent is always stationary at position $x_1(0)$. The other agents (except agent N), on the other hand, try to move to the position $c_i(t)$ which their current information tells them is the middle of their adjacent neighbors. However, they do not jump directly to $c_i(t)$. Instead, they take a step toward it through the attraction/repulsion function $g(\cdot)$ and move to $p_i(t)$. Moreover, note that due to the delays $c_i(t)$ may not be the midpoint between agents $i-1$ and $i+1$ at time t. The last agent (agent N), on the other hand, tries to move to the middle of its current position and the point it perceives to be a (comfortable) distance d from its left neighbor. In other words, it tries to move to the point half-way from the comfortable distance.

In the above setting only the N^{th} agent knows (or decides on) the value of the comfortable distance d. The advantage of this is that it does not require the achievement of agreement between the individuals (by negotiation or other means) on the value of d. The disadvantage is that the other agents do not have influence on the value of the comfortable inter-agent distance. Note also that the value of the comfortable inter-agent distance should be larger (usually much larger) than the collision distance. Therefore, we assume that $d \gg \varepsilon$, where the constant ε is the range of the proximity sensors as discussed in the preceding section.

The elements of T (and therefore of T_i) should be viewed as indices of the sequence of physical times at which the updates occur (similar to the times of events in discrete event systems), not as actual times. In other words, the elements of the sets T_i are integers that can be mapped to actual times. Note that this is consistent with the assumptions that the agents traverse the path from their current position $x_i(t)$ to their intended next position $p_i(t)$. The idle time in (8.2) can be thought of as the time during which the agent traverses the path and the sequence T_i represents the indexes of the times (events) at which the agent either arrives at $p_i(t)$ or encounters a neighbor. The sets T_i are independent from each other for different i. However, it is possible to have $T_i \cap T_j \neq \emptyset$ for $i \neq j$ (i.e., two or more agents may sometimes move simultaneously or in discrete event system terms one or more agents can simultaneously arrive at their respective destinations or simultaneously encounter neighbors).

The function $g(\cdot)$ describes the attractive and repelling relationships between an agent and its adjacent neighbors. It can be viewed as the gradient of a potential function as in the case of continuous-time swarm models considered in the preceding chapters. It is composed of only an attractive component whose minimum occurs when its argument is zero. Therefore, from (8.3) one can see that it results in attraction of agent i toward the perceived midpoint $c_i(t)$ of its neighbors. As already mentioned above, the attraction/repulsion function $g(\cdot)$ determines the step size that an agent will take toward the perceived middle of its neighbors (if it is not already there). We assume that $g(\cdot)$ is sector bounded

$$\underline{\alpha} y^2 \leq y g(y) \leq \bar{\alpha} y^2 \tag{8.4}$$

where $\underline{\alpha}$ and $\bar{\alpha}$ are two constants satisfying

$$0 < \underline{\alpha} < \bar{\alpha} < 1.$$

Figure 8.2 shows the plot of one such $g(\cdot)$. In the figure the plots of $\underline{\alpha} y(t)$ and $\bar{\alpha} y(t)$ for $\underline{\alpha} = 0.1$ and $\bar{\alpha} = 0.9$ are also shown to illustrate the sector boundedness.

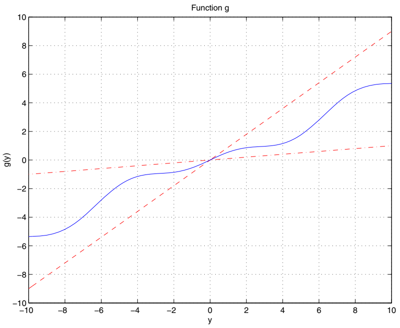


Fig. 8.2. An example $g(\cdot)$ function together with $0.1y(t)$ and $0.9y(t)$.

We would like to emphasize once more that the model in (8.1) is in a sense a discrete event model which does not allow for collisions between the agents or for jumps resulting in change of the order of the agents. This is because it is assumed that the agents do not directly jump to their next way points. Instead they traverse the path between their current position and the next way point and if during movement a given agent i suddenly finds itself within an ε range of one (or both) of its neighbors, it will restrain its movement toward that neighbor (or neighbors) according to equation (8.1).

We will use the notation $x(t) = [x_1(t), \ldots, x_N(t)]^\top$ to represent the position vector at time t of all agents in the swarm. Define the swarm *comfortable position* as

$$x^c = [x_1(0), x_1(0) + d, \ldots, x_1(0) + (N-1)d]^\top.$$

The problem is to analyze whether the swarm will converge to this position or not. In other words, we will analyze the stability of this position by considering the motions of the agents when they are initially located at positions different from x^c. We will consider two cases: synchronous operation with no delays and totally asynchronous operation. The underlying assumptions for these cases are described below.

Assumption 13. (Synchronism, No Delays) *For all agents i the sets* T_i *and the times* $\tau^i_j(t)$ *satisfy* $T_i = T$ *and* $\tau^i_j(t) = t$ *for both of its neighbors* $j = i-1, i+1$.

This assumption states that all the agents will move at the same time instants. Moreover, every agent will always have the current position information of its adjacent neighbors. Note that when this assumption holds there are no idle times and (8.2) becomes irrelevant.

The next assumption, on the other hand, allows the agents to move at totally independent time instants. Moreover, it allows the "delay" between two measurements performed by an individual to become arbitrarily large. However, it also states that there always will be a next time when the agent will perform a measurement.

Assumption 14. (Total Asynchronism) *For all agents i the sets* T_i *are infinite, and if* $\{t^i_\ell\}$ *is a sequence of elements of* T_i *that tends to infinity, then* $\lim_{\ell\to\infty}\tau^i_j(t^i_\ell) = \infty$ *for both of its neighbors* $j = i-1, i+1$.

The first question one can ask about the motion of a swarm described by (8.1) is whether the agent states are bounded or diverge from each other and become unbounded. Intuitively, since there is only attraction toward the perceived midpoint of the corresponding neighbors acting on the agents one would expect that the agent states and inter-agent distances are bounded. However, note also that it is not that obvious since there is asynchronism and time delays (which can become arbitrarily large) in sensing. Nevertheless, it can be shown that it is the case (i.e., the states are bounded) as is formally stated in the following lemma.

Lemma 5. *For the swarm described in* (8.1)-(8.2) *if Assumption 14 holds, then given any* $x(0)$, *there exists a constant* $\bar{b} = \bar{b}(x(0))$ *such that* $x_i(t) \le \bar{b}$, *for all t and all* $i, 1 \le i \le N$.

Proof: We prove this via contradiction. Assume that $x_{i+1}(t) \to \infty$ for some $i+1, 1 \le i \le N$. This implies that $x_j(t) \to \infty$ for all $j \ge i+1$. We will show that it must be the case that $x_i(t) \to \infty$ as well. Assume the contrary, i.e., assume that $x_{i+1}(t) \to \infty$, while $x_i(t) \le b < \infty$ for some b and for all t. Then we have $x_{i+1}(t) - x_i(t) \to \infty$, whereas $x_i(t) - x_{i-1}(t) < 2b_1$ for some $b_1 < b/2$. Let b_2 be a constant such that $\left(1+\frac{1}{\alpha}\right) b < b_2 < \infty$. Note that $b_2 > b_1$. From Assumption 14 there is always a time $t^i_1 \in T_i$ at which agent i performs position sensing of its neighbors and $\tau^i_{i-1}(t) \ge t^i_1$ and $\tau^i_{i+1}(t) \ge t^i_1$ for $t \ge t^i_1$ and

$$\begin{aligned} x_i(t) - x_{i-1}(\tau^i_{i-1}(t)) &< 2b_1, \\ x_{i+1}(\tau^i_{i+1}(t)) - x_i(t) &> 2b_2. \end{aligned}$$

This implies that we have $x_i(t) - c_i(t) < -(b_2 - b_1) < 0$. There exists also a time $t^i_2 \ge t^i_1$ at which agent i moves to the right and its new position satisfies

$$\begin{aligned} x_i(t_2^i+1) &= x_i(t_2^i) - g\big(x_i(t_2^i) - c_i(t_2^i)\big) \\ &\geq x_i(t_2^i) - \underline{\alpha}(x_i(t_2^i) - c_i(t_2^i)) \\ &> x_i(t_2^i) + \underline{\alpha}(b_2 - b_1) \\ &> x_i(t_2^i) + b > b. \end{aligned}$$

This contradicts the assumption that $x_i(t) \leq b$ for all t and implies that $x_i(t) \longrightarrow \infty$ as well. Repeating the argument for the other agents one obtains that $x_i(t) \longrightarrow \infty$ for all $i \neq 1$.

Since $x_1 = x_1(0)$ for all t the above implies that $x_2(t) - x_1(t) \longrightarrow \infty$ as $t \longrightarrow \infty$. Moreover, it must be the case that $x_i(t) - x_{i-1}(t) \longrightarrow \infty$ for all $i = 2, \ldots, N$. To see this assume that $x_i(t) - x_{i-1}(t) \longrightarrow \infty$, whereas $x_{i+1}(t) - x_i(t) < b_3$ for some b_3. There exist a time $t_3^i \in T_i$ at which agent i performs position sensing of its neighbors and $\tau_{i-1}^i(t) \geq t_3^i$ and $\tau_{i+1}^i(t) \geq t_3^i$ for $t \geq t_3^i$. Moreover, for $t \geq t_3^i$ we have

$$\begin{aligned} x_i(t) - x_{i-1}(\tau_{i-1}^i(t)) &> 2b_4, \\ x_{i+1}(\tau_{i+1}^i(t)) - x_i(t) &< 2b_3, \end{aligned}$$

where $b_4 > b_3$ and $x_i(t) < b$ for some $b > b_4$. This implies that we have $x_i(t) - c_i(t) > (b_4 - b_3) > 0$. There exists also a time $t_4^i \geq t_3^i$ at which agent i moves to the left and its new position satisfies

$$\begin{aligned} x_i(t_4^i+1) &= x_i(t_4^i) - g\big(x_i(t_4^i) - c_i(t_4^i)\big) \\ &\leq x_i(t_4^i) - \underline{\alpha}(x_i(t_4^i) - c_i(t_4^i)) \\ &< x_i(t_4^i) - \underline{\alpha}(b_4 - b_3) \\ &< x_i(t_4^i) < b. \end{aligned}$$

Note that as long as we have $x_i(t) - c_i(t) > (b_4 - b_3) > 0$ the agent will be moving to the left and $x_i(t)$ will always be bounded by b. Therefore, it cannot be the case that $x_{i+1}(t) - x_i(t) < b_3$ for some b_3, while $x_i(t) - x_{i-1}(t) \longrightarrow \infty$. Therefore, if $x_i(t) \longrightarrow \infty$ for some $i > 1$, then it must be the case that $x_i(t) \longrightarrow \infty$ and $x_i(t) - x_{i-1}(t) \longrightarrow \infty$ for all $i = 2, \ldots, N$.

Now let us consider individual N. From above we know that there exists a time t_5^N such that $x_N(t) - x_{N-1}(t) > d$. However, there is always a time $t_6^N > t_5^N$ such that agent N performs position sensing and $\tau_{N-1}^N(t) \geq t_6^N$ for $t \geq t_6^N$. Then, we have $x_N(t) - x_{N-1}(\tau_{N-1}^N(t)) > d$ and at some time $t_7^N > t_6^N$ the agent moves and

$$x_N(t_7^N+1) = x_N(t_7^N) - g\big(x_N(t_7^N) - c_N(t_7^N)\big) < x_N(t_7^N),$$

implying that it moves to the left. In fact, as long as we have $x_N(t) - x_{N-1}(\tau_{N-1}^N(t)) > d$ individual N moves to the left and it cannot diverge far away from its neighbor. Assuming that at time t_6^N we have $x_N(t) - x_{N-1}(t) < b$ for some $b > d$, this implies that $x_N(t) - x_{N-1}(t) < b$ for all $t > t_6^N$. Therefore, the $N'th$ individual cannot diverge leading to a contradiction. ■

Given the swarm model in (8.1)-(8.2) the above results states that both agent positions and inter-agent distances are bounded (implying that the swarm will not dissolve) despite the asynchronism in the agent motions and the (possibly arbitrarily large) time delays in neighbor position sensing. Therefore, the main question to be answered is whether the agent position vector $x(t)$ will converge to some constant value and if so is this constant position vector the comfortable position x^c. If the agent's positions do not converge to a constant position then there will be a need to analyze also whether the swarm dynamics will have periodic solutions or will exhibit chaotic behavior. As mentioned previously, we are concerned with the asymptotic stability of the comfortable position x^c and that is what we are going to analyze next. In the next section we will analyze the system in the case of synchronism with no delays. The result for the synchronous case will be used later in the proof of the system under total asynchronism.

8.2 The System under Total Synchronism

In this section we will analyze the stability properties of the system under the condition that Assumption 13 holds (i.e., all agents move at the same time instants and they always have the current position information of their neighbors). First, we will show that x^c is the only equilibrium of the system.

Lemma 6. *For the swarm described in (8.1)-*(8.2) *assume that Assumption 13 holds (i.e., we have synchronism with no delays). If $x(t) \to \bar{x}$ as $t \to \infty$, where $\bar{x}$ is a constant vector, then $\bar{x} = x^c$.*

Proof: First of all, note that the inter-agent distances on all the states that the system can converge to are such that $\bar{x}_i - \bar{x}_{i-1} > \varepsilon$ for all i (i.e., it is impossible for the states to converge to positions that are very close to each other). To prove this, we assume that $\bar{x}_i - \bar{x}_{i-1} = \varepsilon$ for some i and $\bar{x}_j - \bar{x}_{j-1} > \varepsilon$ for all $j \neq i$ and seek to show a contradiction. In that case, $\bar{x}_{i+1} - \bar{x}_i > \varepsilon$ so

$$\bar{x}_i - \bar{c}_i = \bar{x}_i - \frac{\bar{x}_{i-1} + \bar{x}_{i+1}}{2} < 0$$

and we have from model constraints in (8.1) that

$$\bar{x}_{i-1} + \varepsilon < \bar{x}_i - g\left(\bar{x}_i - \frac{\bar{x}_{i-1} + \bar{x}_{i+1}}{2}\right) < \bar{x}_{i+1} - \varepsilon.$$

From (8.1) this implies that at the next time instant $t^i \in T_i$ agent i will move to the right toward agent $i+1$. Therefore, it must be the case that $\bar{x}_{i+1} - \bar{x}_i = \varepsilon$ since otherwise $\bar{x}_i - \bar{x}_{i-1} = \varepsilon$ also cannot hold. Continuing in this manner one can prove that all inter-agent distances must be equal to ε. However, in that case, from the last equality in (8.1) we have

$$\bar{x}_N - g\left(\frac{\varepsilon - d}{2}\right) = \bar{x}_{N-1} + \varepsilon$$

which implies that

$$g\left(\frac{\varepsilon-d}{2}\right)=0$$

must be satisfied. However, since $d \gg \varepsilon$ and from the properties of the function $g(\cdot)$ and in particular condition (8.4) this cannot hold. In fact, we have

$$\bar{x}_N - g\left(\frac{\varepsilon-d}{2}\right) > \bar{x}_{N-1} + \varepsilon - g\left(\frac{\varepsilon-d}{2}\right) > \bar{x}_{N-1} + \varepsilon$$

and this implies that on the next time instant $t^N \in T_N$ agent N will move to the right. Therefore, no inter-agent distance can converge to ε. For this reason, to find $\bar{x}$ we can drop the min and max operators and consider only the middle terms in (8.1).

Since $x(t) \to \bar{x}$ as $t \to \infty$ it should be the case that ultimately

$$\begin{aligned}
\bar{x}_1 &= \bar{x}_1 \\
\bar{x}_i &= \bar{x}_i - g\left(\bar{x}_i - \frac{\bar{x}_{i-1}+\bar{x}_{i+1}}{2}\right),\ i=1,\ldots,N-1 \\
\bar{x}_N &= \bar{x}_N - g\left(\bar{x}_N - \frac{\bar{x}_{N-1}+d+\bar{x}_N}{2}\right)
\end{aligned}$$

from which we obtain

$$\begin{aligned}
\bar{x}_1 &= x_1^c \\
2\bar{x}_i &= \bar{x}_{i-1}+\bar{x}_{i+1},\ i=1,\ldots,N-1 \\
\bar{x}_N &= \bar{x}_{N-1}+d.
\end{aligned} \tag{8.5}$$

Solving the second equation for $\bar{x}_{N-1}$ we have $2\bar{x}_{N-1} = \bar{x}_{N-2} + \bar{x}_N$ from which we obtain $\bar{x}_{N-1} = \bar{x}_{N-2} + d$. Continuing in this manner, we obtain

$$\bar{x}_i = \bar{x}_{i-1} + d,$$

for all $i = 1,\ldots,N-1$. Then since the first agent is stationary we have $\bar{x}_1 = x_1(t) = x_1(0) = x_1^c$ and this proves the result. ■

As mentioned above, the implication of this lemma is basically that x^c is the unique *fixed point* or *equilibrium point* of the system described by (8.1). Note that although the result was proven for the synchronous system without time delays, the same holds also for the asynchronous system. Moreover, as can be seen this fixed point corresponds to the arrangement with the comfortable inter-agent distances. The next objective is to analyze the stability of this fixed point. Toward that objective let us first analyze whether the agents move close to each other resulting in collision possibilities.

Lemma 7. *Consider the swarm described in (8.1)-*(8.2)*. Assume that* $x_i(0) - x_{i-1}(0) > \varepsilon$ *for all* $i = 2,\ldots,N$*. Moreover, assume that Assumption 13 holds (i.e., we have synchronism with no delays). Then,* $x_i(t) - x_{i-1}(t) > \varepsilon$ *for all* $i = 2,\ldots,N$*, and for all* t*.*

Proof: We will prove this by induction. By assumption for $t = 0$ we have $x_i(0) - x_{i-1}(0) > \varepsilon$ for all $i = 2,\ldots,N$. Assume that for some t we have $x_i(t) - x_{i-1}(t) > \varepsilon$ for all $i = 2,\ldots,N$. Then, with a simple manipulation one can show that at that time t we have

$$\begin{aligned} x_{i-1}(t) + \varepsilon &< x_i(t) < x_{i+1}(t) - \varepsilon \\ x_{i-1}(t) + \varepsilon &< c_i(t) < x_{i+1}(t) - \varepsilon \end{aligned} \tag{8.6}$$

and

$$c_{i+1}(t) - c_i(t) > \varepsilon, \tag{8.7}$$

for all $i = 2,\ldots,N$. Noting that it is possible to write the $g(\cdot)$ function as

$$g(y(t)) = \alpha(t)y(t),$$

where $0 < \underline{\alpha} \leq \alpha(t) \leq \bar{\alpha} < 1$, and using this in the swarm dynamics equation we have

$$\begin{aligned} x_i(t+1) &= x_i(t) - \alpha_i(t)\big(x_i(t) - c_i(t)\big) \\ &= \big(1 - \alpha_i(t)\big)x_i(t) + \alpha_i(t)c_i(t). \end{aligned}$$

Therefore, we have

$$\begin{aligned} x_i(t) < c_i(t) &\Rightarrow x_i(t) < x_i(t+1) < c_i(t) \\ x_i(t) > c_i(t) &\Rightarrow x_i(t) > x_i(t+1) > c_i(t). \end{aligned}$$

These inequalities together with (8.6) and (8.7) imply that

$$x_i(t+1) - x_{i-1}(t+1) > \varepsilon$$

and this completes the proof. ■

This lemma implies that for the synchronous case with no delays, provided that initially the agents are sufficiently apart from each other, the proximity sensors will not be used and that we can drop the min and max operations in (8.1) and the system can be represented as

$$\begin{aligned} x_1(t+1) &= x_1(t) \\ x_i(t+1) &= x_i(t) - g\left(x_i(t) - \frac{x_{i-1}(t) + x_{i+1}(t)}{2}\right),\ i = 2,\ldots,N-1, \\ x_N(t+1) &= x_N(t) - g\left(x_N(t) - \frac{x_{N-1}(t) + d + x_N(t)}{2}\right). \end{aligned}$$

Define the following change of coordinates

$$\begin{aligned} e_1(t) &= x_1(t) - x_1^c \\ e_i(t) &= x_i(t) - (x_{i-1}(t) + d),\ i = 2,\ldots,N. \end{aligned}$$

Then, one obtains the following representation of the system

$$\begin{aligned}
e_1(t+1) &= e_1(t) = 0,\\
e_2(t+1) &= e_2(t) - g\left(\tfrac{e_2(t)-e_3(t)}{2}\right),\\
e_i(t+1) &= e_i(t) - g\left(\tfrac{e_i(t)-e_{i+1}(t)}{2}\right) + g\left(\tfrac{e_{i-1}(t)-e_i(t)}{2}\right),\ i=3,\ldots,N-1,\\
e_N(t+1) &= e_N(t) - g\left(\tfrac{e_N(t)}{2}\right) + g\left(\tfrac{e_{N-1}(t)-e_N(t)}{2}\right).
\end{aligned}$$

Using again the fact that $g(y(t)) = \alpha(t)y(t)$ for $0 < \underline{\alpha} < \alpha(t) < \bar{\alpha} < 1$ we can represent the system with

$$\begin{aligned}
e_2(t+1) &= \left(1 - \tfrac{\alpha_2(t)}{2}\right) e_2(t) + \tfrac{\alpha_2(t)}{2} e_3(t),\\
e_i(t+1) &= \left(1 - \tfrac{\alpha_i(t)}{2} - \tfrac{\alpha_{i-1}(t)}{2}\right) e_i(t) + \tfrac{\alpha_{i-1}(t)}{2} e_{i-1}(t) + \tfrac{\alpha_i(t)}{2} e_{i+1}(t),\ i=3,\ldots,N-1,\\
e_N(t+1) &= \left(1 - \tfrac{\alpha_N(t)}{2} - \tfrac{\alpha_{N-1}(t)}{2}\right) e_N(t) + \tfrac{\alpha_{N-1}(t)}{2} e_{N-1}(t),
\end{aligned}$$

where we dropped $e_1(t)$ since it is zero for all t. Defining $e(t) = [e_2(t),\ldots,e_N(t)]^\top$ it can be seen that our system is, in a sense, a linear time varying system, which can be represented in a matrix form as

$$e(t+1) = A(t)e(t), \tag{8.8}$$

where $A(t)$ is a symmetric tridiagonal matrix of the form

$$A(t) = \begin{bmatrix}
b_2(t) & a_2(t) & 0 & \ldots & 0\\
a_2(t) & b_3(t) & a_3(t) & \ddots & \vdots\\
0 & a_3(t) & \ddots & \ddots & 0\\
\vdots & \ddots & \ddots & b_{N-1}(t) & a_{N-1}(t)\\
0 & \ldots & 0 & a_{N-1}(t) & b_N(t)
\end{bmatrix}$$

with diagonal elements given by

$$\begin{aligned}
\{b_2(t),\ldots,b_N(t)\} = \Bigg\{ &\left(1 - \frac{\alpha_2(t)}{2}\right), \left(1 - \frac{\alpha_3(t)}{2} - \frac{\alpha_2(t)}{2}\right), \ldots,\\
&\left(1 - \frac{\alpha_{N-1}(t)}{2} - \frac{\alpha_{N-2}(t)}{2}\right), \left(1 - \frac{\alpha_N(t)}{2} - \frac{\alpha_{N-1}(t)}{2}\right)\Bigg\}
\end{aligned}$$

and off-diagonal elements equal to

$$\{a_2(t),\ldots,a_{N-1}(t)\} = \left\{\frac{\alpha_2(t)}{2}, \ldots, \frac{\alpha_{N-1}(t)}{2}\right\}.$$

First, we will investigate the properties of the matrix $A(t)$ which will be useful later in deriving our stability result. In particular, we will show that all the eigenvalues of $A(t)$ lie strictly within the unit circle.

Lemma 8. *The spectrum of the matrix* $A(t)$ *in* (8.8), $\rho(A(t))$ *satisfies*

$$\rho(A(t)) < 1$$

for all t*. In other words, all the eigenvalues of* $A(t)$ *lie strictly within the unit circle.*

Proof: First, note that for the given $A(t)$ we have

$$\|A(t)\|_1 = \|A(t)\|_\infty = 1$$

for all t. On the other hand, for any given matrix $A(t)$ it is well known that the two norm satisfies

$$\|A(t)\|_2 \leq \sqrt{\|A(t)\|_1 \|A(t)\|_\infty}.$$

Hence, since because of the symmetry of $A(t)$ we have $\rho(A(t)) = \|A(t)\|_2$ and also due to the general properties of the spectral radius we obtain

$$\rho(A(t)) \leq 1$$

for all t. Note that this can be deduced from the Gerschgorin's theorem as well. Let us define

$$Z_2(t) = \{x : |x - b_2(t)| \leq a_2(t)\}$$

$$Z_k(t) = \{x : |x - b_k(t)| \leq a_{k-1}(t) + a_k(t)\},\ k = 3, ..., N-1$$

$$Z_N(t) = \{x : |x - b_N(t)| \leq a_{N-1}(t)\}$$

which are called Gerschgorin's disks. Then, from the Gerschgorin's theorem one can deduce that all the eigenvalues of $A(t)$ will be located in the region

$$Z(t) = \cup_{k=2}^{N} Z_k(t)$$

Moreover, since $A(t)$ is a symmetric matrix, all of its eigenvalues are real and the eigenvalues of $A(t)$ are located on the intersection of $Z(t)$ with the real line, which is located within the unit circle. However, this does not exclude the case $\lambda(A(t)) = 1$ or basically that $\rho(A(t)) = 1$. Therefore, for the sake of contradiction let us assume that there is an eigenvalue such that $\lambda(A(t)) = 1$ and that $\rho(A(t)) = 1$. Then, from the definition of the eigenvalues one can deduce that the matrix $(A(t) - I)$ must be a singular matrix. Defining the matrix

$$P = \begin{bmatrix} 1 & \dots & \dots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{bmatrix}$$

one can show that the matrix multiplication $P^\top (A(t) - I)P$ results in

$$P^\top(A(t)-I)P = \begin{bmatrix} -\frac{\alpha_2(t)}{2} & 0 & 0 & \cdots & 0 \\ 0 & -\frac{\alpha_3(t)}{2} & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -\frac{\alpha_{N-1}(t)}{2} & 0 \\ 0 & \cdots & 0 & 0 & -\frac{\alpha_N(t)}{2} \end{bmatrix}$$

which is nonsingular since

$$\underline{\alpha} \le \alpha_i(t) \le \bar{\alpha}$$

is satisfied for all t and $i = 2,\ldots,N$. Then, since the matrix P is nonsingular it must be the case that $(A(t)-I)$ is also nonsingular resulting in a contradiction. Therefore, $(A(t)-I)$ cannot be a singular matrix and hence $\lambda = 1$ cannot be an eigenvalue of $A(t)$ resulting in the fact that

$$\rho(A(t)) < 1$$

for each t. ■

This lemma basically states that the eigenvalues of $A(t)$ (which are all real numbers since $A(t)$ is symmetric) do not depend on the time index (or iteration number) t and lie strictly within the unit circle for each t. Before proceeding let us define

$$\bar{\rho} = \sup_{\underline{\alpha}\le\alpha_i\le\bar{\alpha}, i=2\ldots N} \{\rho(A)\}.$$

Then, from the above result we have

$$\bar{\rho} < 1.$$

Using this, one can state the following result for the system under total synchronism.

Theorem 17. *Consider the swarm described in (8.1)-(8.2) composed of N agents with $g(\cdot)$ as given in (8.4), if Assumption 13 holds and $x_{i+1}(0) - x_i(0) > \varepsilon$, $i = 1,\ldots,N-1$, then we have $x(t) \to x^c$ as $t \to \infty$.*

Proof: The result directly follows from Lemma 8. Since the eigenvalues of $A(t)$ lie strictly within the unit disk for all t, for every t given the matrix $A(t)$ there exist corresponding positive definite and symmetric matrices $P(t) = P(t)^\top > 0$ and $Q(t) = Q(t)^\top > 0$ such that the Lyapunov equation

$$A(t)^\top P(t) A(t) - P(t) = -Q(t)$$

is satisfied. In fact, at any instant t given the matrix $A(t)$ and any positive definite and symmetric matrix $Q(t) = Q(t)^\top > 0$ this equation has a unique positive definite and symmetric solution $P(t) = P(t)^\top > 0$. Choose, $Q(t) = \gamma I$ for some $\gamma > 0$ and for all t and solve

$$A(t)^\top P(t) A(t) - P(t) = -\gamma I$$

for $P(t)$. Note that existence of solutions which are positive definite (strictly bounded away from singularity) and symmetric for all t is guaranteed from the fact that the eigenvalues of $A(t)$ lie strictly within the unit disk for all t. Choose the common time varying Lyapunov function for the system in (8.8) as

$$V(t) = e(t)^\top P(t)e(t)$$

Note that this function is positive definite, radially unbounded and decrescent since the $P(t)$ matrices are strictly bounded away from singularity. In other words, defining

$$\lambda_{min} = \min_t\{\lambda(P(t))\} \quad \text{and} \quad \lambda_{max} = \max_t\{\lambda(P(t))\}$$

one can see that

$$\lambda_{min}\|e(t)\|^2 \le V(t) \le \lambda_{max}\|e(t)\|^2$$

is satisfied. Then, calculating the difference between two instances of $V(t)$ one obtains

$$V(t+1) - V(t) = -\gamma\|e(t)\|^2$$

which implies that as $t \to \infty$ we have $e(t) \to 0$. Then, the result follows from the definition of the relative inter-agent dynamics $e(t)$. ■

This result implies that (8.8) defines a contraction. In fact, taking the norm of both sides of (8.8) one can see that

$$\|e(t+1)\|_2 = \|A(t)e(t)\|_2 \le \|A(t)\|_2\|e(t)\|_2 \le \bar{\rho}\|e(t)\|_2 < \|e(t)\|_2$$

which is an alternative way of looking at the dynamics of the relative inter-agent distances.

The above theorem implies that the agent positions will asymptotically converge to the comfortable position x^c. It is an important result; however, it is derived only under the assumption that the agents move synchronously and there are no time delays in neighbor position sensing.

Our objective is to prove that the same type of convergence will be achieved for the totally asynchronous case. We will investigate that case in the following section and the result of Theorem 17 will be useful in the analysis there.

8.3 Asynchronous Swarm

In this section we return to the totally asynchronous case. In other words, we investigate the system under Assumption 14. To prove convergence to the comfortable position x^c we will use the result from the synchronous case and a result from [22]. For convenience we present this result here.

Consider a function $f : X \to X$, where $X = X_1 \times \ldots \times X_n$, and $x = [x_1, \ldots, x_n]^\top$ with $x_i \in X_i$. The function f is composed of functions $f_i : X \to X_i$ in the form $f = [f_1, \ldots, f_n]^\top$ for all $x \in X$. Consider the problem of finding the point x^* such that

$$x^* = f(x^*)$$

using an asynchronous algorithm. In other words, use an algorithm in which

$$x_i(t+1) = f_i\big(x_1(\tau_1^i(t)), \ldots, x_n(\tau_n^i(t))\big), \ \forall t \in T_i, \tag{8.9}$$

where $\tau_j^i(t)$ are times satisfying $0 \leq \tau_j^i(t) \leq t$, $\forall t \in T$. For all the other times $t \notin T_i$, x_i is left unchanged. In other words, we have

$$x_i(t+1) = x_i(t), \ \forall t \notin T_i. \tag{8.10}$$

Consider the following assumption.

Assumption 15. *([22]) There is a sequence of nonempty sets* $\{X(t)\}$ *with*

$$\cdots \subset X(t+1) \subset X(t) \subset \cdots \subset X,$$

satisfying the following two conditions:

1. Synchronous Convergence Condition (SCC): *We have*

   $$f(x) \in X(t+1), \forall t \text{ and } x \in X(t).$$

   Furthermore, if $\{y_t | y_t = f(y_{t-1})\}$ *is a sequence such that* $y_t \in X(t)$ *for every* t*, then every limit point of* $\{y_t\}$ *is a fixed point of* f.
2. Box Condition (BC): *For every* t*, there exist sets* $X_i(t) \subset X_i$ *such that*

   $$X(t) = X_1(t) \times X_2(t) \times \ldots X_n(t).$$

The synchronous convergence condition implies that the limit points of the sequences generated by the *synchronous iteration* $x(t+1) = f(x(t))$ are fixed points of f. The box condition, on the other hand, implies that combining components of vectors in $X(t)$ results in a vector in $X(t)$. In other words, if $x \in X(t)$ and $\bar{x} \in X(t)$, then replacing i^{th} component of x with the i^{th} component of $\bar{x}$ results in a vector in $X(t)$. An example when the box condition holds is when $X(t)$ is a sphere in $\mathbb{R}^n$ with respect to some weighted maximum norm.

Assumption 15 is about the convergence of the synchronous iteration (i.e. iteration (8.9) under total synchronism). The result below shows that if the synchronous algorithm is convergent, the asynchronous algorithm will also converge provided that Assumption 14 is satisfied.

Theorem 18. Asynchronous Convergence Theorem [22]: *Consider the asynchronous iteration in* (8.9)-(8.10)*. If the Synchronous Convergence Condition and Box Condition of Assumption 15 hold together with Assumption 14, and the initial solution estimate* $x(0) = [x_1(0), \ldots, x_n(0)]^\top$ *belongs to the set* $X(0)$*, then every limit point of* $\{x(t)\}$ *is a fixed point of* f.

This is a powerful result that can be applied to many different problems. The main idea behind its proof is as follows. Assume there is a time instant t_1 such that $x_j(\tau_j^i(t_1)) \in X_j(t)$ for all j and all i, which implies that *the perceived* $x(t_1)$ is in the set $X(t)$. Then Assumption 14 (the total asynchronism assumption) guarantees that there is another time $t_2 > t_1$ at which all the processors (agents) will have performed an update, i.e., for all i there are time instants $t^i \in T_i$ such that $t_1 \leq t^i \leq t_2$. Then, since $x_j(\tau^i_j(t_1)) \in X_j(t)$ for all $t \geq t_1$ the synchronous convergence condition in Assumption 15 together with the iteration in (8.9) guarantee that $x(t_2+1) \in X(t+1)$. In fact, it is guaranteed that $x(t) \in X(t+1)$ for all $t \geq t_2+1$. In addition, the box condition implies that each $x_j(t_2+1) \in X_j(t+1)$. Then, once again due to the total asynchronism assumption (Assumption 14) there will be always another time instant $t_3 > t_2$ such that all processors (agents) will have obtained information from (performed sensing of) their neighbors (i.e., $\tau^i_j(t_3) > t_2+1$ for all i and j) and $x_j(\tau^i_j(t_3)) \in X_j(t+1)$ for all j and all i and this completes the induction step. Then, since initially we have $x_j(\tau^i_j(0)) = x_j(0) \in X_j(0)$ and the above reasoning shows that the iteration has the same contraction properties as its synchronous counterpart convergence of the asynchronous iteration is guaranteed from the convergence of the synchronous iteration.

Note that the swarm model in (8.1)-(8.2) is in agreement with the general iteration model in (8.9)-(8.10). Therefore, the above theorem can be used to prove the stability of x^c for the asynchronous swarm as is stated in the following result.

Theorem 19. *For the N-agent swarm modeled in (8.1)-*(8.2) *with* $g(\cdot)$ *as given in (8.4), if Assumption 14 holds and* $x_{i+1}(0) - x_i(0) > \varepsilon$, $i = 1,\ldots,N-1$, *then the agent positions will converge asymptotically to the comfortable position* x^c.

Proof: In order to prove this result we once again consider the synchronous case. Recall that for this case the system can be described by

$$e(t+1) = A(t)e(t).$$

In the previous section it was shown that for the synchronous case we have $\lambda(A(t)) \leq \bar{\rho} < 1$ for all t and that $e(t) \longrightarrow 0$ as $t \longrightarrow \infty$. This implies that $A(t)$ is a maximum norm contraction mapping for all t. Define the sets

$$E(t) = \{e \in \mathbb{R}^{N-1} : \|e\|_\infty \leq \bar{\rho}^t \|e(0)\|_\infty\}.$$

Then since $A(t)$ is a maximum norm contraction mapping for all t we have $e(t) \in E(t)$ for all t and

$$\ldots \subset E(t+1) \subset E(t) \subset \ldots \subset E = \mathbb{R}^{N-1}.$$

Moreover, each $E(t)$ can be expressed as

$$E(t) = E_2(t) \times E_3(t) \times \ldots E_N(t)$$

where $E_j(t)$ corresponds to the j^{th} dimension (inter-agent distance). Since the position with comfortable inter-agent distance $e = 0$ (i.e., $x = x^c$) is the unique fixed point of the system and the synchronous swarm converges to it, it is implied that Assumption 15 above is satisfied. Applying the Asynchronous Convergence Theorem the result is obtained. ■

This result is important because it states that the stability of the system will be preserved (i.e., the system will converge to the comfortable distance) even though we have totally asynchronous motions and imperfect information due to the time delays. Note that the fact that in the asynchronous case in (8.1) the min and max operations are preserved does not change the result in Theorem 19 since the stability properties of the synchronous system is preserved even with them (i.e., the min and max operations) present in the model.

A relevant issue to mention here is the speed of convergence of the algorithm. Theorem 19 guarantees that the agent positions will asymptotically converge to the comfortable position x^c; however, it does not provide information about the speed of convergence. In fact, it can be seen from Assumption 14 that it is not possible to establish a lower bound on the speed of convergence due to the asynchronism (and the fact that there is no restriction on the sets T_i and therefore the update times) and the possibility of arbitrarily large time delays. Therefore, the convergence speed may vary depending on the properties of the sets T_i and the time delays $\tau^i_j(t)$ and sometimes may take very long. If a faster convergence is desired restrictions on the sets T_i such as restricting it to contain one instant from every interval $\{t-B+1, t-B+2, ..., t-1, t\}$ and a bound on the time delay of the form $t-B+1 \leq \tau^i_j(t) \leq t$ for some $B > 0$ need to be imposed. Such a uniformity in the updates and sensing will bring uniformity in the convergence. Of course the convergence properties of the synchronous iteration due to the properties of the function $g(\cdot)$ have also important role in the convergence of the asynchronous one. An asynchronous algorithm which satisfies the above type of uniformity restrictions is referred to as a *partially asynchronous* algorithm.

Note also that the swarm equation in (8.1) is naturally distributed, where each individual i performs the computation only of its next position $x_i(t+1)$ based on the perceived (measured or obtained by communication) position of its two neighbors. Moreover, it performs this computation only at the times it is “awake” (i.e., only at $t \in T_i$). Therefore, the computational load of the individuals is minimal.

A direct consequence of Theorem 19 is the stability of a swarm in which one agent in the middle is stationary, whereas all the other middle agents try to move similar to the middle agents in the model in (8.1) and both of the edge agents try to move to a distance d from their neighbors. In other words, suppose the swarm is described by

$$\begin{aligned}
x_1(t+1) &= \min\Big\{p_1(t), x_2(t)-\varepsilon\Big\}, \ \forall t \in T_1,\\
x_j(t+1) &= x_j(t), \ \forall t \text{ and for some } j, 1 \leq j \leq N,\\
x_i(t+1) &= \max\Big\{x_{i-1}(t)+\varepsilon, \min\{p_i(t), x_{i+1}(t)-\varepsilon\}\Big\}, \forall t \in T_i, i=2,..,N-1, i \neq j,\\
x_N(t+1) &= \max\Big\{x_{N-1}(t)+\varepsilon, p_N(t)\Big\}, \ \forall t \in T_N, \qquad (8.11)
\end{aligned}$$

where $p_i(t)$'s are as defined before and the perceived center for the first agent is

$$c_1(t) = \frac{1}{2}[x_2(\tau^1_2(t)) - d + x_1(t)].$$

Here again it is implicitly assumed that $x_{i+1}(t) - x_{i-1}(t) > 2\varepsilon$, as was the case in the preceding section. Recall that this is always the case provided that $x_{i+1}(0) -$

$x_{i-1}(0) > 2\varepsilon$. In this case we have the following corollary as a direct consequence of Theorem 19.

Corollary 3. *For the N-agent swarm modeled in (8.11)-*(8.2) *with* $g(\cdot)$ *as given in (8.4), if Assumption (14) holds and* $x_{i+1}(0) - x_i(0) > \varepsilon, i = 1, \ldots, N-1$, *then the agent positions will converge asymptotically to* x^c, *where* x^c *is defined such that* $x_j^c = x_j(0)$ *and* $x_i^c = x_j(0) + (i-j)d$, *for all* $i \neq j$.

The usefulness of this result is for systems in which the "leader" of the swarm is not the first (or the last) agent, but an agent in the middle. It is also worth mentioning here that if the first and the last agents in (8.11) employ two different desired inter-agent distances d_1 and d_N, $d_1 \neq d_N$, then these will be the inter-agent distances on the corresponding sides of the stationary agent j. This result is also directly implied by Theorem 19 and can be stated as another corollary. The proofs follow from considering each side of the stationary agent separately and applying Theorem 19.

8.4 Simulation Results

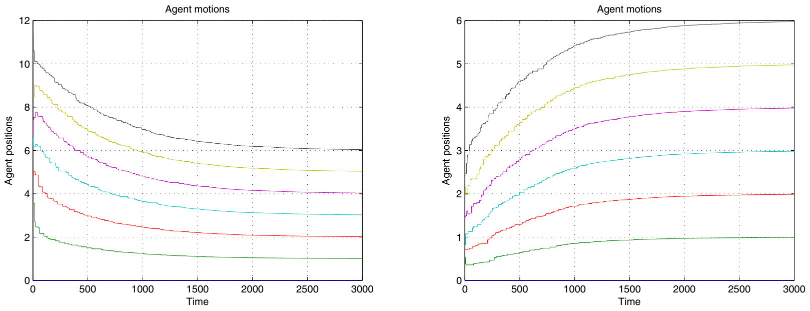
In this section we provide numerical simulation examples. We chose $N = 7$ agents and $d = 1$ as the desired comfortable distance. As an attraction/repulsion function we used the function

$$g(y) = \alpha y + \beta \sin(y)$$

with $\alpha = 0.5$ and $\beta = 0.4$. To achieve asynchronism and time delays at each time step the agents are set up to sense their neighbor positions (independently for each neighbor) and to update their own position with some probability. In particular, we defined two threshold probabilities $0 < \bar{p}_{sense} < 1$ and $0 < \bar{p}_{move} < 1$. At each time instant t for each agent i three random numbers $0 < p^i_{sense_{left}}(t) < 1$, $0 < p^i_{sense_{right}}(t) < 1$, and $0 < p^i_{move}(t) < 1$ are generated with uniform probability density. If $p^i_{sense_{left}}(t) > \bar{p}_{sense}$ agent i performs position sensing on its left (i.e., obtains the current position of its left neighbor). Otherwise, it keeps the old position information of its left neighbor. Similarly, $p^i_{sense_{right}}(t) > \bar{p}_{sense}$ agent i senses the position of its right neighbor. Furthermore, if $p^i_{move}(t) > \bar{p}_{move}$, then agent i updates its position according to (8.1). Otherwise, it keeps its current position according to (8.2).

We performed several simulations with different parameters. Figure 8.3(a) shows a simulation of a contracting swarm (i.e., a swarm in which the agents are far apart from each other initially). The parameters for this simulation are chosen as $\bar{p}_{sense} = 0.9$ and $\bar{p}_{move} = 0.9$ which mean 0.1 probability of sense and move. The agents move to the comfortable position ($d = 1$ apart from each other) as time progresses, as expected.

Figure 8.3(b) shows a simulation of an expanding swarm (i.e., a swarm in which the agents are close to each other initially). In this simulation the results are also as the theory predicts and the swarm converges to the desired comfortable relative position. The parameters for this simulation were chosen the same as those for the simulation in Figure 8.3(a).



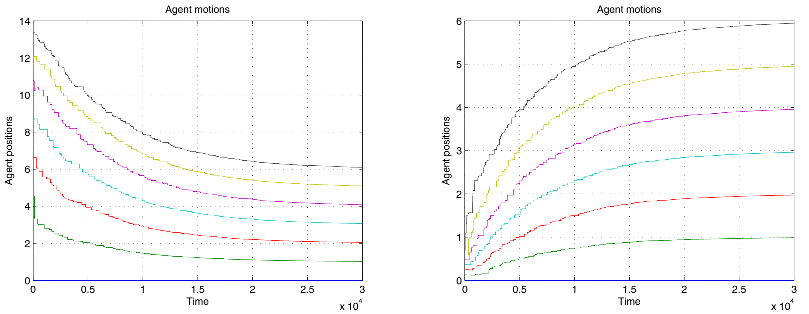
(a) Contracting swarm.

(b) Expanding swarm.

Fig. 8.3. Swarm motion ($\bar{p}_{sense} = 0.9$, and $\bar{p}_{move} = 0.9$).

Note that the values of $\bar{p}_{sense}$ and $\bar{p}_{move}$ effectively specify the amount of asynchronism and the time delays in the system. In particular, larger values $\bar{p}_{sense}$ and $\bar{p}_{move}$ mean larger delays in neighbor position sensing and less motion (more idle) time for the agents and therefore slower convergence. In contrast, decreasing the values of $\bar{p}_{sense}$ and $\bar{p}_{move}$ result in more frequent moves and more current information (or basically less asynchronous swarm) and therefore in faster convergence.

Figure 8.4 shows another simulation in which sense and move probability parameters were changed to $\bar{p}_{sense} = 0.99$ and $\bar{p}_{move} = 0.99$, resulting in 0.01 probability of sense and move. As can be seen from the figure the results are similar to those obtained in Figure 8.3. The only difference is that it takes much longer time to converge in the second case. In particular, while the iterations in Figure 8.3 are up to 3.000, in Figure 8.4 they are up to 30.000. These results are in parallel with the expectations. Cross varying the simulation parameters also results in expected results (simulations not shown).



(a) Contracting swarm.

(b) Expanding swarm.

Fig. 8.4. Swarm motion ($\bar{p}_{sense} = 0.99$, and $\bar{p}_{move} = 0.99$).

8.5 Further Issues

8.5.1 Extensions and Generalizations

The results in this chapter can be extended to the case in which there is no stationary agent and the agents on both sides employ the same value d for the comfortable inter-agent distances. In fact, a swarm model with similar properties will be discussed in the context of distributed agreement in the next chapter of this book.

Another interesting extension which can be investigated is to consider a swarm with no stationary agent and the agents on both sides employ different values d_1 and d_2 for the desired (comfortable) inter-agent distances. For that case one would expect that there will not be an equilibrium for the system and the swarm will either move in one direction or exhibit periodic (or chaotic) behavior.

A further extension could be to consider swarms which move in higher dimensional spaces and investigate problems such as formation control (i.e., achieving and maintaining a predefined geometrical shape).

8.5.2 For Further Reading

This chapter is based on the works in [91, 95]. Note that here we corrected a small error present in [91, 95]. Moreover, new proofs for some of the results were presented here. The single agent model described in this chapter was first used in [158] and is taken from there. In fact, the work in [155, 158] is very much related to the content of this chapter. However, different tools are used for analysis in there. Other related work are the articles in [154, 156, 157] where the authors considered moving swarms and proved cohesiveness results assuming *partial asynchronism* and the existence of a bound on the maximum step size. The analysis tools and some of the concepts and assumptions used in this chapter are based on the work on parallel and distributed computation in [22].

9

Asynchronous Distributed Agreement in Discrete Time Swarms

9.1 Model of the System

In this chapter we consider a substantially different problem from the problems considered so far in this book. It is the problem of distributed agreement in a swarm of agents using only local interactions. This problem has been considered in the literature under various names including synchronization and consensus. However, we believe that the term distributed agreement is the most appropriate terminology for the problem under consideration.

Let us consider a multi-agent system (a swarm) consisting of N individuals with states denoted by $x_i \in \mathbb{R}^n$ which could represent position, orientation, synchronization frequency or some other physical variable depending on the problem. It could also represent some other information (e.g. cognitive variables) to be agreed upon in a distributed manner by the agents. By "agreement" here we mean the situation in which all agents reach the same value. By "distributed manner," on the other hand, we mean using only local information, i.e., each agent uses information from only its immediate neighbors and does not have information about the global picture. With that purpose let us assume that each agent can communicate only with a fixed or time-dependent subset of the swarm. We will refer to this subset as the neighbors of the given agent. Given agent i we denote with $\mathcal{N}_i(t)$ the set of its neighbors and with $N_i(t) = |\mathcal{N}_i|$ the number of its neighbors at time t. In other words, $N_i(t)$ denotes the number of elements in the set $\mathcal{N}_i(t)$. In applications this subset representing the neighbors of a given agent may be determined based on the distance between the agents (due to, for example, the finite range of communication or sensing), based on physical layout or topology of the environment (walls may cause agents to be out of sight etc.) or by some other (e.g., heuristic, probabilistic or ad-hoc) means. Under these conditions, intuitively one would think that the best way to achieve agreement in a distributed fashion is to use averaging. Guided by this intuition let us consider a swarm model in which each agent i updates its state by

$$x_i(t+1) = \frac{1}{w_i(t)} \left[w_{ii}(t)x_i(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)x_j(\tau_{ij}(t)) \right], \forall t \in T_i, \tag{9.1}$$

where $x_i(t), i = 1, \ldots, N$, represents the state of agent i at time t, the variables $w_{ij}(t), 1 \leq i, j \leq N$ are weighting factors representing the importance of the state of a neighbor agent j for agent i, and the variables $w_i(t)$ are given by

$$w_i(t) = w_{ii}(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t), i = 1, \ldots, N.$$

similar to the case in Chapter 8, the set $T_i \subseteq T = \{0, 1, 2, \ldots\}$ is the set of time indices at which agent i updates its state. At the other time instants agent i is stationary, i.e., it does not perform update and

$$x_i(t+1) = x_i(t), \forall t \notin T_i \tag{9.2}$$

holds. It is assumed that the weighting factors $w_{ij}(t)$ satisfy $w_{min} \leq w_{ij}(t) \leq w_{max}$ for some $w_{min} > 0$ and $w_{max} < \infty$ for all i, j and all t. The variables $\tau_{ij}(t), j \in \mathcal{N}_i(t), i = 1, \ldots, N$, are used to represent the time index of the state information of $j \in \mathcal{N}_i(t)$ to which agent i has access to. They satisfy $0 \leq \tau_{ij}(t) \leq t$ for $t \in T_i$, where $\tau_{ij}(t) = 0$ means that agent i has not yet obtained any information about agent j (it still has the initial state information), whereas $\tau_{ij}(t) = t$ means that it has the current state information of agent j. The difference $(t - \tau_{ij}(t)) \geq 0$ can be viewed as a sensing delay or a communication delay in obtaining information about agent j by agent i. Note that this definition can represent both of the following cases: (i) the agents are memoryless and when at time t agent j becomes a neighbor of agent i it performs (relative) position sensing of agent j but possibly gets an old distance due to the time delay in sensing; (ii) the agents have memory and keep record of all its past neighbors and when at time t agent j becomes a neighbor of agent i it may perform sensing of the state of agent j with some probability or use the old recorded information about it. The first case may arise in, for example, robot gathering algorithms, whereas the second may, for example, serve as a crude representation of interactions in social networks and adaptation of attitudes or beliefs.

The reader may have already noticed that the model in Equation (9.1) is a weighted averaging model which takes convex combinations of the available information from a subset of the agents, which are the neighbors of a given agent, with possible time delays. Note that $j \in \mathcal{N}_i(t)$ does not necessarily mean that $i \in \mathcal{N}_j(t)$ as well. In other words, communication is unidirectional. Moreover, even if $j \in \mathcal{N}_i(t)$ and $i \in \mathcal{N}_j(t)$ hold simultaneously, this does not imply $\tau_{ij}(t) = \tau_{ji}(t)$. In other words, even if two agents i and j are mutually neighbors of each other at a given time instant, it does not mean that they have the current or equally outdated information about each other implying that they do not necessarily communicate information to each other simultaneously. In fact, it is possible that they might obtain information about each other as well as update their states at totally independent time instants.

The elements of the set T (and therefore of the sets T_i) should not be viewed as actual times but as indices of the sequence of ordered physical times $\mathcal{T} = \{t_0, t_1, t_2, \ldots\}$

at which the updates generated by all the agents occur (similar to the times of events in discrete event systems). By "ordered" here it is meant that for all k we have $t_k < t_{k+1}$. In other words, elements of the set T are integers that can be mapped into the actual times (i.e., the times of the events) $\{t_k | t_k < t_{k+1}\}$ and the physical time intervals $(t_{k+1} - t_k)$ between subsequent indices (events) are not necessarily uniform. The sets T_i are independent from each other for different i. However, it is possible to have $T_i \cap T_j \neq \emptyset$ for $i \neq j$ (i.e., it may happen that sometimes two or more agents update their states simultaneously). Note that the set T is needed only for analysis purposes only; it is not required for the agents to know it.[1] Similarly, the agents do not need to know neither the sets T_i nor the set of physical times $\mathcal{T}$. One can view these sets as global times viewed/observed by an external observer, while the agents operate on their local independent clocks. In other words, there is no need for a global clock or means for synchronization for implementing (9.1).

The swarm model in (9.1) has the properties that: (i) agents use local information; (ii) sensing/interaction/information exchange is unidirectional; (iii) agents update their states in asynchronous manner; (iv) agents do not necessarily have the exact information about the states of the other agents in the swarm. Therefore, it is very suitable for describing operation of distributed multi-agent systems since the above features are natural properties of such systems. Synchronous operation is often not possible in nature and is difficult to implement in artificial multi-agent systems such as swarms of robots since it requires a global clock to which all the agents must be subjected to undermining the distributed (decentralized) nature of the system. Moreover, local sensing/interaction and information exchange and possible delays are also very realistic properties. The problem of distributed agreement can be stated more formally as follows.

Definition 1. *It is said that the swarm asymptotically reaches agreement if the states of all agents converge to a common value or basically if as $t \longrightarrow \infty$*

$$\lim_{t \longrightarrow \infty} x_i(t) = x_a, \tag{9.3}$$

is satisfied for some constant vector $x_a \in \mathbb{R}^n$ and for all i.

The question here is whether this equation will be satisfied despite the local interactions (dynamic neighborhood), the asynchronism in the updates, and the time delays in the sensing/information flow.

It is common to use a directed graph to represent the interaction (information flow) topology. Let $\mathcal{G}(t) = (\mathcal{N}, \mathcal{A}(t))$ denote the information flow or interaction graph of the system/swarm at time t, where $\mathcal{N} = \{1, 2, \ldots, N\}$ is the fixed set of nodes and $\mathcal{A}(t) \subset \mathcal{N} \times \mathcal{N}$ denotes the set of directed arcs (or information flow links) at time t. Agent $i \in \mathcal{N}$ denotes the i^{th} node or vertex of the graph, whereas the arc $(i, j) \in \mathcal{A}(t)$ represents a directed information flow link from agent i to agent j at time t. In other words, if $(i, j) \in \mathcal{A}(t)$, then agent j can receive or obtain information

---

[1] The set T is also needed in artificial implementations of the iteration in (9.1). In practice in natural or artificial multi-agent systems whose operation obeys (9.1) it is an emergent set.

from agent i at instant t implying that $i \in \mathcal{N}_j(t)$. Note once more that the information flow is unidirectional meaning that $(i,j) \in \mathcal{A}(t)$ does not imply that $(j,i) \in \mathcal{A}(t)$.

Agent i is said to be connected to agent j if there is a directed path from i to j. In other words, there is a sequence of arcs $(i_1,i_2),(i_2,i_3),\ldots,(i_{p-1},i_p)$ such that $i_1 = i$ and $i_p = j$. If there is a (directed) path from every i to every j, then the graph is said to be strongly connected. A directed tree is a directed graph in which every node, except the root, has exactly one incoming edge (arc). If the tree connects all the vertices of the graph, then it is called a spanning tree. Note that if a graph has a spanning tree, then there is at least one node (agent) which is connected to all other nodes (agents).

As we mentioned above, in this chapter we assume that the communication topology can be time-dependent. Let us denote with $\bar{\mathcal{G}} = \{\mathcal{G}_1,\ldots,\mathcal{G}_M\} = \{\mathcal{G}_p | p = 1,\ldots,M\}$ the set of all possible interaction graphs. Note that $\bar{\mathcal{G}}$ is finite and for each t we have $\mathcal{G}(t) \in \bar{\mathcal{G}}$. The union of a set of graphs $\{\mathcal{G}_i = (\mathcal{N},\mathcal{A}_i)\} \subset \bar{\mathcal{G}}$ with the same vertex set is the graph defined as $\cup \mathcal{G}_i = (\mathcal{N}, \cup \mathcal{A}_i)$. It is said that a sequence of graphs $\{\mathcal{G}(t)\}$ has a spanning tree over an interval $\mathfrak{I}$ if the graph $\cup_{t \in I} \mathcal{G}(t)$ has a spanning tree. Let $\mathcal{P} = \{1,\ldots,M\}$ and $\sigma : T \to \mathcal{P}$ denote the switching sequence of the communication/interaction graphs. Also, given a switching sequence $\sigma(t)$ denote with $\{\mathcal{G}_{\sigma(t)}\} = \{\mathcal{G}_p(t) = (\mathcal{N},\mathcal{A}_p(t))\}$ the corresponding sequence of communication graphs. Now, we have the following assumption.

Assumption 16. *The switching sequence $\sigma(t)$ is such that there exists a constant $I \geq 0$ such that for every interval $\mathfrak{I}$ of length I the corresponding sequence of communication graphs $\{\mathcal{G}_{\sigma(t)}\}$ has a spanning tree.*

Assumption 16 is a minimal necessary assumption for achieving convergence of the agent's states to a common value. In fact, if it is not satisfied it would mean that there exists at least one agent which is never connected to part of the agents in the swarm and cannot receive information from them either directly or indirectly through other agents. This, on the other hand, will prevent the system states from converging to the same value. As will be discussed below in the case of synchronous motion and perfect information Assumption 16 is also sufficient for achieving agreement. However, in the case of asynchronism and time delays there is a need for extra conditions imposed on the information updates or the delay time.

As was done in Chapter 8, the easiest approach to analyze the system behavior is first to consider the system under synchronous motion and no time delays and later on with the intuition gained build on the results for the synchronous system to analyze the convergence properties of the asynchronous system in (9.1)-(9.2). This is the approach taken here.

9.2 System under Total Synchronism

We start with the following assumption (which is similar to Assumption 17 in Chapter 8).

Assumption 17. (Synchronism, No Delays) *The sets* T_i *and the times* $\tau_{ij}(t)$ *satisfy* $T_i = T$ *for all* i *and* $\tau_{ij}(t) = t$ *for all* i *and* $j \in \mathcal{N}_i(t)$.

This assumption states that all the agents in the swarm will move at the same time instants. Moreover, every agent will always have the current state information of its neighbors. Under this assumption the motion dynamics of the system become

$$x_i(t+1) = \frac{1}{w_i(t)} \left[w_{ii}(t)x_i(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)x_j(t) \right] \tag{9.4}$$

for all $t \in T$ and all i. Now the objective is to show that under Assumption 16 the condition in Equation (9.3) will be satisfied for the synchronous system in Equation (9.4). With this objective let us first define

$$m(t) = \min_{i=1,\ldots,N} \{x_i(t)\} \quad \text{and} \quad M(t) = \max_{i=1,\ldots,N} \{x_i(t)\}. \tag{9.5}$$

Here $m(t)$ and $M(t)$ are vectors of dimension n and the minimum and the maximum operators are elementwise. We would like to emphasize that the values of $m(t)$ and $M(t)$ depend on the initial configuration $x(0)$ of the system as well as the switching sequence $\sigma(t)$. Here $x(t)$ denotes the concatenation of the states of all agents with $x(t) = \left(x_1^\top(t), x_2^\top(t), \ldots, x_N^\top(t)\right)^\top \in \mathbb{R}^{N \times n}$.

It is easy to see that $\{m(t)\}$ is non-decreasing and $\{M(t)\}$ is non-increasing along the solutions of (9.4). In other words, we have $m(t+1) \geq m(t)$ and $M(t+1) \leq M(t)$ for all t. This is because of the convexity property of the weighted averaging in (9.4). By taking convex combination between a set of numbers/points the minimum value cannot decrease and the maximum value cannot increase. Since $m(t)$ and $M(t)$ are monotonic and bounded their limits exist and as we will see below they are equal. In other words, we have

$$\lim_{t \to \infty} m(t) = m = M = \lim_{t \to \infty} M(t).$$

This final value is the agreement value defined as x_a in (9.3). Note that since there is no stochastisity in the system the value of x_a is uniquely determined by the initial agent states $x(0)$ and the switching sequence $\sigma(t)$.[2] The agents or an external observer do not know the agreement value x_a a priori since they either do not have global information or do not know the switching sequence a priori. In fact, if the agreement value x_a was known a priori to the agents the agreement problem would become much simpler.

Besides the fact that $m(t)$ is non-decreasing and $M(t)$ is non-increasing it is guaranteed that an increase in $m(t)$ or a decrease in $M(t)$ will occur in at most a finite number of time steps, the number of which is bounded by the maximal information flow path in the communication graph (which is always less than the number of agents N) and the uniformity in connectivity parameter I in Assumption 16. To see

---

[2] This statement holds provided that the switching sequence $\sigma(t)$ is given. In fact there could be some stochasticity in the switching sequence itself.

this consider the following. Denote at time $t = t_k$ with $\mathcal{N}_M$ the set of agents j such that $x_j(t_k) = M(t_k)$ and with $\mathcal{N}_m$ the set of agents j such that $x_j(t_k) = m(t_k)$.[3] Based on Assumption 16 in at most I time steps, say time $t = t_k + I$ there will be a directed edge from the rest of the agents towards either an agent in $\mathcal{N}_M$ or $\mathcal{N}_m$ or between agents in the two sets. Then, from the dynamics in (9.4) and the properties of convex combinations we know that at time $t = t_k + I + 1$ the value of the state of the agent which had the directed in-link will change and therefore the number of agents in $\mathcal{N}_M \cup \mathcal{N}_m$ will decrease by at least one. Repeating the argument and noting that the number of agents in $\mathcal{N}_M \cup \mathcal{N}_m$ can be at most N one can see that at time $t = t_k + I \times N + 1$ we have $\mathcal{N}_M \cup \mathcal{N}_m = \emptyset$ meaning that both the minimum and the maximum values will change. In fact, at time $t = t_k + I \times (N-1) + 1$ it will be guaranteed that there is at most one agent in $\mathcal{N}_M \cup \mathcal{N}_m$ and that either the minimum $m(t_k)$ has increased or the maximum $M(t_k)$ has decreased. In other words, it is guaranteed that for all t either $m(t + I \times (N-1) + 1) > m(t)$ or $M(t + I \times (N-1) + 1) < M(t)$. Note that in practice, based on the switching sequence $\sigma(t)$, usually it may happen that $m(t+\eta) > m(t)$ or $M(t+\eta) < M(t)$ for some $\eta < I \times (N-1) + 1$ and $I \times (N-1) + 1$ is the length of the worst case interval for which the above condition is guaranteed to hold for any switching sequence satisfying Assumption 16. These observations intuitively imply that agreement occurs. To show that this is the case in more mathematical manner let us define the metric

$$\rho(x,y) = \sum_{j=1}^{n} \max_{i=1,\ldots,N} \left\{ |x_{ij} - y_{ij}| \right\}$$

for vectors $x, y \in \mathbb{R}^{N \times n}$ such that $x = \left(x_1^\top, x_2^\top, \ldots, x_N^\top\right)^\top$ and $y = \left(y_1^\top, y_2^\top, \ldots, y_N^\top\right)^\top$. Here $x_i, y_i \in \mathbb{R}^n$ are sub-components of x, y and $x_{ij}, y_{ij} \in \mathbb{R}$ denote the j'th elements of the vectors x_i, y_i. Similarly, the agreement set $X_a \subset \mathbb{R}^{N \times n}$ can be defined as

$$X_a = \left\{ x \in \mathbb{R}^{N \times n} : x_i = x_j, \forall i, j \right\} \tag{9.6}$$

where the partition of the vectors is as defined above. The generalized distance of a given $x \in \mathbb{R}^{N \times n}$ to the agreement set X_a is given by

$$\rho(x, X_a) = \inf_{y \in X_a} \left\{ \rho(x,y) \right\}$$

With the above definition $\left\{\rho, \mathbb{R}^{N \times n}\right\}$ is a metric space. Now, let us define the Lyapunov function candidate as

$$V(x(t)) = \sum_{j=1}^{n} \left(M_j(t) - m_j(t) \right) \tag{9.7}$$

where $m(t)$ and $M(t)$ are the minimum and maximum defined in (9.5) and $m_j(t)$ and $M_j(t)$ are their j'th components, respectively. Note that $V(x(t)) \geq 0$ for all $x(t) \in \mathbb{R}^{N \times n}$. Moreover, the distance of a given $x(t)$ to X_a satisfies

---

[3] Here the analysis is performed as if $x_i(t), M(t), m(t) \in \mathbb{R}$. However, the analysis holds for $x_i(t), M(t), m(t) \in \mathbb{R}^n$ by considering elementwise operations.

$$\rho(x(t), X_a) \geq \frac{1}{2}\sum_{j=1}^{n}\Big(M_j(t) - m_j(t)\Big)$$

and

$$\rho(x(t), X_a) \leq \sum_{j=1}^{n}\Big(M_j(t) - m_j(t)\Big)$$

implying that

$$\rho(x(t), X_a) \leq V(x(t)) \leq 2\rho(x(t), X_a)$$

is satisfied for all t. This, on the other hand, implies that $V(x(t))$ can serve as a Lyapunov function for the system. Taking the time difference one obtains

$$\begin{aligned} V(x(t+1)) - V(x(t)) &= \sum_{j=1}^{n}(M_j(t+1) - m_j(t+1)) - \sum_{j=1}^{n}(M_j(t) - m_j(t)) \\ &= \sum_{j=1}^{n}(M_j(t+1) - M_j(t)) - \sum_{j=1}^{n}(m_j(t+1) - m_j(t)) \leq 0 \end{aligned}$$

which follows from the facts that $M(t)$ is non-increasing and $m(t)$ is non-decreasing. This implies that the system is stable in the sense of Lyapunov. In order to show asymptotic convergence let us define $V(k) = V(x(k \times (I \times (N-1)+1)))$, where $k = 0, 1, 2, ...$ is the index representing the times at which contraction is guaranteed. From the discussion above we know that such contraction is always guaranteed for any switching sequence satisfying Assumption 16.[4] Then, we have

$$V(k+1) - V(k) < 0$$

which implies that as $k \to \infty$ we have $V(k) \to 0$. This, on the other hand, due to the properties of $V(x(t))$ also implies that as $t \to \infty$ we have $V(x(t)) \to 0$ and the set of agreement states X_a is asymptotically stable. Therefore, we have proved the following result.

Theorem 20. *Consider a swarm composed of N agents with motion dynamics given by* (9.4) *with dynamic (time-dependent) interconnection topology (interaction graph). If Assumption 16 and Assumption 17 hold, then as* $t \to \infty$ *we have* $x(t) \to X_a$ *and* (9.3) *is satisfied, i.e., the states of the agents in the swarm will asymptotically converge to a common value* x_a.

This is an important result which basically states that despite the local unidirectional inter-agent interactions the swarm will reach agreement provided that the uniformity in the connectivity assumption (Assumption 16) is satisfied. Therefore, the mapping consisting of concatenation of all (9.4) for all i is a contraction mapping in $\mathbb{R}^{N \times n}$. Defining the set $Y(t) \subset \mathbb{R}^n$ as

---

[4] Another possible way to define $V(k)$ could be to take $V(k) = V(t_k)$ where $\{t_k\} \subset T$ is a subsequence such that either $m(t_{k+1}) > m(t_k)$ or $M(t_{k+1}) < M(t_k)$ holds and contraction occurs.

$$Y(t) = \{x \in \mathbb{R}^n | m(t) \leq x \leq M(t)\}$$

from above we know that $Y(t+1) \subseteq Y(t)$ for all t. Moreover, we have $Y(t+I \times (N-1)+1) \subset Y(t)$ since contraction is guaranteed every $I \times (N-1)+1$ steps. Let us define $\bar{Y}(k) = Y(k \times (I \times (N-1)+1))$, where $k = 0,1,2,...$ is the index representing the times at which contraction is guaranteed. Then $\bar{Y}(k+1) \subset \bar{Y}(k)$ is satisfied for all k. Also let us define

$$X(k) = \underbrace{\bar{Y}(k) \times \bar{Y}(k) \times \cdots \times \bar{Y}(k)}_{N \text{ copies of } \bar{Y}(k)} \tag{9.8}$$

and let $\bar{x}_a = x_a \times x_a \times \cdots \times x_a \in X_a$. Then, from the discussion above we have

$$\bar{x}_a \subset \cdots \subset X(k+1) \subset X(k) \subset \cdots \subset X(0) \subset \mathbb{R}^{N \times n}.$$

In other words, since (9.3) holds, under the dynamics in (9.4) the above defined sets $X(k)$ uniformly contract until the state of the system converges to a value $\bar{x}_a \in X_a$ (for some $x_a \in \mathbb{R}^n$). In particular, any sequence $\{x_k\}$ such that $x_k \in X(k)$ converges to $\bar{x}_a$. For any switching sequence $\sigma(t)$ satisfying Assumption 16, there is always such a corresponding sequence of contracting sets $X(k)$ and corresponding agreement state $\bar{x}_a \in X_a$ (whose value also depends on $x(0)$). In other words, under Assumption 16 all the limit points of the synchronous iteration in 9.4 are fixed points of the iteration which are also agreement states and within the set X_a.

Having shown that the synchronous system with no delays will converge we return to the asynchronous system in the next section.

9.3 Asynchronous System

In this section we return to the asynchronous system in Equations (9.1)-(9.2). We start with an assumption which allows the agents to move at totally independent time instants. However, it also guarantees that the agents will perform measurement/communication with their neighbors and will update their state (will move) in at most B time steps for some finite B. In other words, there is uniformity in the measurement/communication as well as the update/move times, or basically the time delay and the times between two moves is uniformly bounded. The value of the bound B does not need to be known by the agents. It is needed for analysis purposes and it is sufficient for it just to exist.

Assumption 18. *There exists a finite positive constant B such that for every agent i and for all $t \geq 0$ the following conditions hold:*

- *At least one of the elements of $\{t, t+1, ..., t+B-1\}$ belongs to T_i.*
- *Given the switching sequence $\sigma(t)$ for every $j \in \mathcal{N}_i(t)$ we have $t - B < \tau_{ij}(t) \leq t$.*

Assumption 18, which is borrowed from [22], basically states that any agent performs a move in at most B time steps and that the information about the neighbors (used by the agent during determination of its next state/way-point) is outdated by at most B time steps.[5] Assuming such bounds is very reasonable since if there are agents which do not perform update/move for unbounded amount of time or do not perform position sensing of their neighbors they are not effectively part of the swarm.

Note that since the sets T_i are infinite and there are only a finite number of agents in the swarm, some of them may become neighbors of i only finitely many times, while others become its neighbor infinitely many times as $t \longrightarrow \infty$. The second part of Assumption 18 can be relaxed to state that given the switching sequence $\sigma(t)$ the agent i regularly updates its (perceived) information about members j which become its neighbors, i.e., $j \in \mathcal{N}_i(t)$, infinitely often as time goes to infinity (and therefore its state affects the update in (9.1) as time goes to infinity). Still even if some of the agents do not become neighbors at all, under Assumption 16 information flow in the swarm is guaranteed and at least one of these two agents is able to obtain information about the state of the other agent indirectly through other intermediate agents.

Under Assumption 18 using the result for the synchronous case, the box condition in (9.8), and Theorem 18 in Chapter 8 (the asynchronous convergence theorem from [22]) one can show that as $t \longrightarrow \infty$ we have $x(t) \longrightarrow X_a$, where X_a is the agreement set in (9.6), implying that (9.3) is satisfied. Still, here we will provide also an alternative proof based on Lyapunov argument. We state the result first.

Theorem 21. *Consider a swarm composed of N agents with motion dynamics given by* (9.1)-(9.2) *with dynamic (time-dependent) interconnection topology (interaction graph). If Assumption 16 and Assumption 18 hold, then as* $t \longrightarrow \infty$ *we have* $x(t) \longrightarrow X_a$ *and* (9.3) *is satisfied, i.e., the states of the agents in the swarm will asymptotically converge to a common value* x_a.

Proof: Let us once more denote the concatenation of the states of all agents with $x(t) = \left(x_1^\top(t), x_2^\top(t), \ldots, x_N^\top(t)\right)^\top \in \mathbb{R}^{N \times n}$. Initially at $t = 0$ we have $x(0) \in X(0)$ by hypothesis, where $X(k), k = 0, 1, 2, ...$, is given in Equation (9.8). Given that $x(t) \in X(k)$ at some t_k and for all $t \geq t_k$ we will show that there exist a time t_{k+1} such that $x(t) \in X(k+1)$ for all $t \geq t_{k+1}$. Then, in the light of the discussion for the synchronous case in the preceding section and application of the asynchronous convergence theorem in [22] the result follows. Still we will show that a Lyapunov argument can be used to deduce the same result as well. Let

$$x_{pi}(t) = (x_1(\tau_{i1}(t)), x_2(\tau_{i2}(t)), \ldots, x_N(\tau_{iN}(t))) \tag{9.9}$$

denote the "perceived" system state by agent i. We use this notation for convenience. Here, if $j \in \mathcal{N}_i(t)$, then $x_j(\tau_{ij}(t))$ is the perceived state of neighbor j, otherwise if

---

[5] Assumption 18 is similar to Assumption 14 in Chapter 8 but also has some differences. Systems whose operation obeys Assumption 14 in Chapter 8 can be referred to as totally asynchronous systems, whereas those whose operation obeys Assumption 18 can be referred to as partially asynchronous systems [22].

$j \notin \mathcal{N}_i(t)$ we take $x_j(\tau_{ij}(t)) = x_j(t)$ since it does not affect the state update of agent i in (9.1).

By Assumption 18 since for every i and $j \in \mathcal{N}_i(t)$ we have $t - B < \tau_{ij}(t) \leq t$, it is guaranteed that after time $\bar{t}_k = t_k + B$ we have $\tau_{ij}(t) \geq t_k$ for all $t \geq \bar{t}_k = t_k + B$ for all agents i and for all $j \in \mathcal{N}_i(t)$. In other words, all agents perform sensing of (or communication with) all of their neighbors (arising due to the switching sequence $\sigma(t)$) by time $\bar{t}_k$ or in at most B steps after time t_k; this is guaranteed by Assumption 18. Note that the perceived minimum by individual i

$$m_{pi}(t) = \min_{j=1,\ldots,N} \{x_j(\tau_{ij}(t))\} \leq m(t)$$

cannot decrease and the perceived maximum

$$M_{pi}(t) = \max_{j=1,\ldots,N} \{x_j(\tau_{ij}(t))\} \geq M(t)$$

cannot increase. Here $m(t)$ and $M(t)$ are the minimum and maximum defined in (9.5). Note that the asynchronism and the time delay in the iteration do not change the properties of these sequences. Also the inequalities $m_{pi}(t) \leq m(t)$ and $M_{pi}(t) \geq M(t)$ hold due to the facts that $m(t)$ is non-decreasing ($m(t-1) \leq m(t)$ for all t) and $M(t)$ is non-increasing ($M(t-1) \geq M(t)$ for all t) and the facts that the perceived $m_{pi}(t)$ and $M_{pi}(t)$ correspond to possibly older information. Therefore, since $x(t) \in X(k)$ for $t \geq t_k$ and $\tau_{ij}(t) \geq t_k$ for $t \geq \bar{t}_k = t_k + B$ and for all $j \in \mathcal{N}_i(t)$, we have $m_{pi}(t) \geq m(t_k)$ and $M_{pi}(t) \leq M(t_k)$ for all $t \geq \bar{t}_k = t_k + B$ implying that

$$x_{pi}(t) \in X(k), \forall t \geq \bar{t}_k = t_k + B, \forall i$$

In other words, for $t \geq \bar{t}_k = t_k + B$ the agents "perceive" that the state of the system belongs to $X(k)$. Then, as in the synchronous case defining the sets $\mathcal{N}_M$ as the set of agents j such that $x_j(t_k) = M(t_k)$ and $\mathcal{N}_m$ as the set of agents j such that $x_j(t_k) = m(t_k)$, we know that for $t \geq \bar{t}_k = t_k + B$ this information will also be perceived by all agents.[6] From Assumption 18 we know that every agent will perform a move in at most B time steps. Similarly from Assumption 16 we know that there will be a spanning tree every interval of length I. Then, with a similar reasoning to the synchronous case after $I \times B$ time steps it is guaranteed that the number of agents in $\mathcal{N}_m \cup \mathcal{N}_M$ will decrease by at least one. Repeating the argument as in the synchronous case we obtain that after $I \times B \times (N-1) + 1$ either an increase in $m(t)$ or a decrease in $M(t)$ will occur. Then, for the function in (9.7) defining $V(k) = V(x(k(B + I \times B \times (N-1) + 1)))$ we once again obtain

$$V(k+1) - V(k) < 0 \tag{9.10}$$

which implies that as $t \to \infty$ we have $V(x(t)) \to 0$ and that $x(t) \in X(k+1)$ for all $t \geq t_{k+1} = t_k + B + I \times B \times (N-1) + 1$. This completes the induction step and

---

[6] Here again the analysis is performed as if the variables are one-dimensional. However, as before it holds for vector-type variables with elementwise operations.

convergence can be deduced from the asynchronous convergence theorem. Looking at the system from Lyapunov perspective one can see that the inequality in (9.10) is not sufficient to provide an alternative argument. This is because there is a delay in the system and the delayed states need also be included into the system state. Therefore, let us define the extended state of agent i as $\bar{x}_i(t) = (x_i(t), x_i(t-1), \ldots, x_i(t-B+1)) = (\bar{x}_i^1(t), \bar{x}_i^2(t), \ldots, \bar{x}_i^B(t))$ and the extended state of the system as $\bar{x}(t) = (\bar{x}_1^\top(t), \bar{x}_2^\top(t-1), \ldots, \bar{x}_N^\top(t-B+1))^\top$. Similarly, the extended agreement set can be defined as

$$\bar{X}_a = \left\{ \bar{x} \in \mathbb{R}^{N\times n\times B} : \bar{x}_i^k = \bar{x}_j^k, \forall i,j \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,B\} \right\} \tag{9.11}$$

Also define the metric in $\mathbb{R}^{N\times n\times B}$ as

$$\rho(\bar{x},\bar{y}) = \sum_{j=1}^{n} \max_{k=1,\ldots B} \left\{ \max_{i=1,\ldots,N} \left\{ |\bar{x}_{ij}^k - \bar{y}_{ij}^k| \right\} \right\}$$

where $\bar{x},\bar{y} \in \mathbb{R}^{N\times n\times B}$ such that $\bar{x}_{ij}^k$ represents the j'th element of the k'th subcomponents of the vector $\bar{x}_i$ and similarly for $\bar{y}_{ij}^k$. Then, the generalized distance of a given $\bar{x} \in \mathbb{R}^{N\times n\times B}$ to the extended agreement set $\bar{X}_a$ can be calculated as

$$\rho(\bar{x},\bar{X}_a) = \inf_{\bar{y}\in\bar{X}_a} \left\{ \rho(\bar{x},\bar{y}) \right\}$$

Let us also define

$$\bar{m}(t) = \min_{k=t,\ldots,t-B+1} \left\{ \min_{i=1,\ldots,N} \{x_i(k)\} \right\} \tag{9.12}$$

and

$$\bar{M}(t) = \max_{k=t,\ldots,t-B+1} \left\{ \max_{i=1,\ldots,N} \{x_i(k)\} \right\} \tag{9.13}$$

as the minimum and the maximum of the extended system state. Note that due to the fact that the minimum is nondecreasing and the maximum is nonincreasing we have

$$m(t-B+1) \le \bar{m}(t) \le m(t) \le M(t) \le \bar{M}(t) \le M(t-B+1)$$

for all t. Defining the Lyapunov function as

$$\bar{V}(\bar{x}(t)) = \sum_{j=1}^{n} \left(\bar{M}_j(t) - \bar{m}_j(t) \right) \tag{9.14}$$

one can see that the relation

$$0 \le V(x(t)) \le \bar{V}(\bar{x}(t)) \le V(x(t-B+1))$$

is satisfied for all t. Careful consideration also shows that

$$\rho(\bar{x}(t),\bar{X}_a) \le \bar{V}(\bar{x}(t)) \le 2\rho(\bar{x}(t),\bar{X}_a)$$

is also satisfied which implies that $\bar{V}(\bar{x}(t))$ can serve as a Lyapunov function for the system. Then, from the fact that $V(x(t)) \longrightarrow 0$ as $t \longrightarrow \infty$ one can conclude that $\bar{V}(\bar{x}(t)) \longrightarrow 0$ as $t \longrightarrow \infty$ and that the set of agreement states $\bar{X}_a$ is asymptotically stable and this completes the proof. ■

This result is important because it states that the stability of the system will be preserved (i.e., all agent states will converge to a common value) even though we have asynchronous state update mechanism and imperfect information due to time delays in sensing the states of the agent neighbors on top of time varying or switching communication topology. The main arguments of the proof are based on a convexity-type condition and the contraction properties of the iteration (due to the averaging in the agent motion dynamics). The asynchronism and the time delays slow down the convergence but do not change the main stability properties of the motion. The speed of convergence is bounded below by a constant which depends on the value of $\mu = B + I \times B \times (N-1) + 1$ since it is guaranteed that contraction will occur in at most μ time steps.

A special case of this result occurs when the communication topology is fixed. If this is the case Assumption 16 basically becomes the following.

Assumption 19. *The interaction graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ has a spanning tree.*

The corresponding result can be stated as follows.

Corollary 4. *Consider a swarm composed of N agents with motion dynamics given by* (9.1)-(9.2) *with fixed (static) interconnection topology (interaction graph). If Assumption 19 and Assumption 18 hold, then as $t \longrightarrow \infty$ we have $x(t) \longrightarrow X_a$ and* (9.3) *is satisfied, i.e., the states of the agents in the swarm will asymptotically converge to a common value x_a.*

9.4 Simulation Examples

In this section numerical simulation examples for both fixed and dynamic neighborhood topology cases of asynchronous swarm agreement are presented. The dimension of the state space is chosen as $n = 3$, i.e., the agent dynamics evolve in $\mathbb{R}^3$. To achieve asynchronism artificially at each time step the agents are set up to sense their neighbor states and to update their own state with some probability. In particular, two threshold probabilities $0 < \bar{p}_{sense} < 1$ and $0 < \bar{p}_{move} < 1$ are defined. At each time instant t, for each individual i, total of $(N_i(t) + 1)$ random numbers, which include $p^{ij}_{sense}(t), j = 1, \ldots, N_i(t)$ and one $p^{i}_{move}(t)$, are generated with uniform probability density in the interval $[0, 1]$. If at time t we have $p^{ij}_{sense}(t) > \bar{p}_{sense}$, then agent i receives the current state of its neighbor $j \in \mathcal{N}_i(t)$. Otherwise, it keeps the old state information of agent j. Similarly, if at step t we have $p^{i}_{move}(t) > \bar{p}_{move}$, then individual i updates its state according to the dynamics in (9.1). Otherwise, it keeps its current state according to (9.2). The pseudo-code in Table 9.1 shows the logic based on which the agents move. This implementation is not a real discrete event based asynchronous system. Instead it mimics such systems and is sufficient

Table 9.1. Pseudo-code of Agent Dynamics

```
initialize $\bar{p}_{sense}$ and $\bar{p}_{move}$
initialize the positions of the agents (randomly)
determine the neighborhoods $\mathcal{N}_i, i = 1, ..., N$ (fixed topology only)
for t=1:final_time do
 for each agent $i$ do
  determine the set of its neighbors $\mathcal{N}_i(t)$ (dynamic topology only)
  for each agent $j \in \mathcal{N}_i(t)$
   generate $p^{ij}_{sense}(t)$
   if $p^{ij}_{sense}(t) > \bar{p}_{sense}$
    obtain the current state information of agent $j$
   end
  end
  generate $p^i_{move}(t)$
  if $p^i_{move}(t) > \bar{p}_{move}$,
   update state using (9.1)
  else
   keep current state using (9.2)
  end
 end
end
```

for illustrating the theoretical results presented in this chapter. Another issue here is how to determine the set of neighbors of the agents at each step. In fact, there could be various procedures for determining the neighbors of the agents both in the fixed and the dynamic topologies. These might include strategies based on the nearest neighbors in the motion space of the form

$$\mathcal{N}_i(t) = \{j \mid j \neq i, \ \|x_i(t) - x_j(t)\| \leq \delta_i\}$$

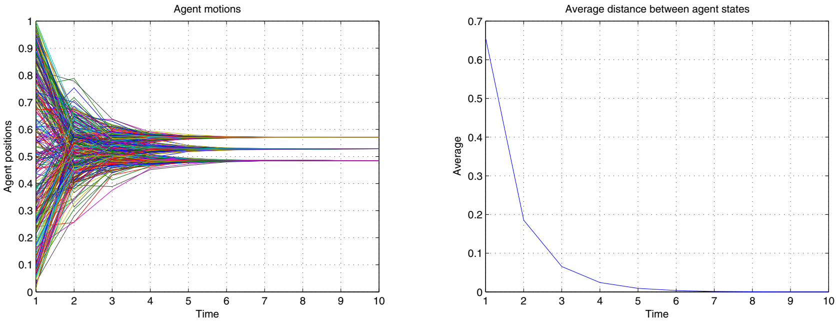
where the agents within δ_i distance from agent i (which can be viewed as the sensing range of agent i) are its neighbors. Other alternatives could be nearest neighbors in some other space not related to the states that are being agreed upon (such as for example orientation agreement in multi-agent dynamic systems where the nearest neighbors are determined based on the positions and not the orientations), nearest neighbors based on the performance with respect to some evaluation function, random determination procedure, and others. For many applications the neighborhoods will just emerge as the system dynamics progress. For the illustrative simulations here we assign the neighbors randomly. The pseudo-code for determining the neighborhoods randomly is shown in Table 9.2. In particular, for the fixed topology case before the main loop, whereas for the dynamics topology case at each time step t, a total of $N \times (N-1)$ random numbers are generated (i.e., different random numbers $p^{ij}_{nbr}(t)$ for every possible pairs $(i, j), j \neq i$). Also, a constant $\bar{p}_{nbr}$ was defined and if at time t for a pair $(i, j), j \neq i$ the condition $p^{ij}_{nbr}(t) > \bar{p}_{nbr}$ is satisfied, then agent j is assigned as a neighbor of agent i for time t, i.e., $j \in \mathcal{N}_i(t)$. Since $p^{ij}_{nbr}(t)$ is

Table 9.2. Pseudo-code For Random Neighborhood

```
initialize $\bar{p}_{nbr}$
for each agent $i$ do
  for each agent $j \neq i$ do
    generate $p^{ij}_{nbr}(t)$
    if $p^{ij}_{nbr}(t) > \bar{p}_{nbr}$
      $j \in \mathcal{N}_i(t)$
    end
  end
end
```

independent and possibly different from $p^{ji}_{nbr}(t)$, $j \in \mathcal{N}_i(t)$ does not imply $i \in \mathcal{N}_j(t)$ or vice versa.

Figure 9.1 shows the result for a simulation of a system with fixed communication topology for $N = 100$ agents. The initial states of the agents are chosen randomly in the interval $[0,1]$. Moreover, the (fixed) neighbors of the agents also assigned randomly with probability of 0.1 (i.e., $\bar{p}_{nbr} = 0.9$). Also for these simulations the parameters $\bar{p}_{sense} = 0.5$ and $\bar{p}_{move} = 0.5$ were used. As is seen from Figure 9.1(a) all agent states converge to the same value. The three distinct lines in the figure are due to the fact that the agent states evolve in $\mathbb{R}^3$ (i.e., $x_i \in \mathbb{R}^3$) and all three coordinates are plotted on the same plot. Figure 9.1(b) shows the plot of the average of the distances between the states of the agents given by



(a) States of agents.

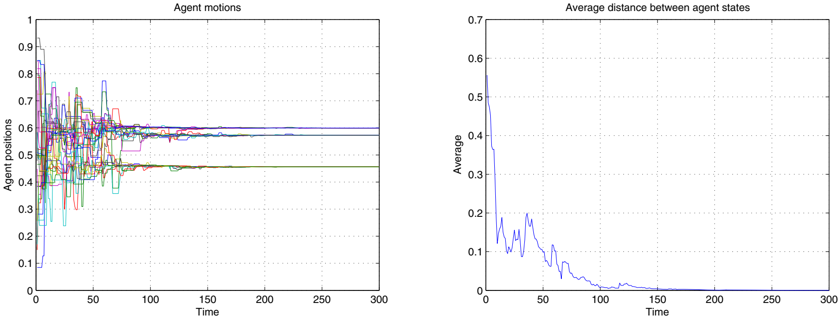
(b) Distance between agent states.

Fig. 9.1. Simulation for fixed neighborhood topology.

$$e(t) = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} ||x_i(t) - x_j(t)||$$

with respect to time. As predicted by the analysis it is seen to converge to zero. Experimenting with other fixed neighborhood topologies such as the fully connected and the cyclic (ring) neighborhood topologies shows that similar results are obtained (results not shown here). The fully connected neighborhood topology converges very fast. In the ring topology, on the other hand, the convergence is slower since the length of the path in the spanning tree there is much longer. The only point here is that the spanning tree assumption (Assumption 16) should be satisfied. For the fully connected and the ring topologies it is always satisfied, however for the random neighborhood for large values of $\bar{p}_{nbr}$ it might not be satisfied and for these cases convergence may not be guaranteed in general.

For the dynamic topology case, in addition to the asynchronism and time delays at each time step the neighbors of the agents were re-assigned randomly based on the procedure shown in Table 9.2. The plots in Figure 9.2 show respectively the evolution of the agent states and the sum of the distances between them with respect to time. For these simulations again the same probability values $\bar{p}_{sense} = 0.5$, $\bar{p}_{move} = 0.5$, and $\bar{p}_{nbr} = 0.9$ were used. As one can see, once more the states of all agents converge to a common value. For this case, however, convergence is slower compared to the corresponding fixed topology case.

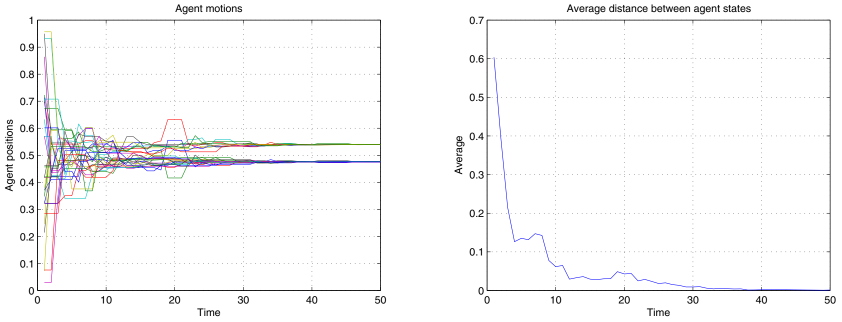


(a) States of agents.

(b) Distance between agent states.

Fig. 9.2. Simulation for dynamic neighborhood topology ($\bar{p}_{sense} = 0.5$, $\bar{p}_{move} = 0.5$, and $\bar{p}_{nbr} = 0.9$).

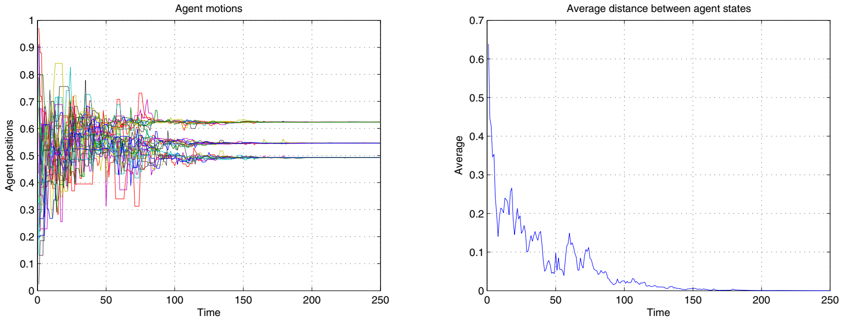
Although the fact that a bound B on the time delay and the time between two subsequent moves (recall Assumption 18) was not explicitly imposed, effectively such a bound exists in the above simulations despite the fact that with the above type of implementation theoretically infinite delays are also possible. In particular, in the simulations by choosing the values of $\bar{p}_{sense}$, $\bar{p}_{move}$, and $\bar{p}_{nbr}$ one can change the speed of convergence (of the implemented simulation algorithm). In fact, decreasing $\bar{p}_{sense}$, $\bar{p}_{move}$ or $\bar{p}_{nbr}$ leads to faster convergence (since it results in higher probabilities to sense, move, and become neighbors), whereas increasing $\bar{p}_{sense}$, $\bar{p}_{move}$ or $\bar{p}_{nbr}$ leads to slower convergence. This is because decreasing $\bar{p}_{sense}$ and $\bar{p}_{move}$ implies that the agents will move and sense the states of their neighbors more often (implying that effectively the bound B in Assumption 18 will decrease). In other words, the values of these parameters determine the resulting effective value of the bound B which, on the other hand, affects the speed of convergence. Decreasing $\bar{p}_{nbr}$, on the other hand, leads to a more connected communication topology (implying that effectively the bound I in Assumption 16 will decrease). Note from the proof of Theorem 21 that both B and I affect the worst case time for the set to contract and to guarantee decrease in the Lyapunov function. The plots in Figures 9.3, 9.4, and 9.5 investigate these effects. For the simulation in Figure 9.3 the probability threshold for the agents becoming neighbors was decreased from $\bar{p}_{nbr} = 0.9$ to $\bar{p}_{nbr} = 0.5$ (while keeping $\bar{p}_{sense} = 0.5$ and $\bar{p}_{move} = 0.5$). As one can see from the figure the system converges (i.e. agreement is achieved) much faster as expected. For the simulation in Figure 9.4 the probability threshold for the agents to move was decreased to



(a) States of agents.

(b) Distance between agent states.

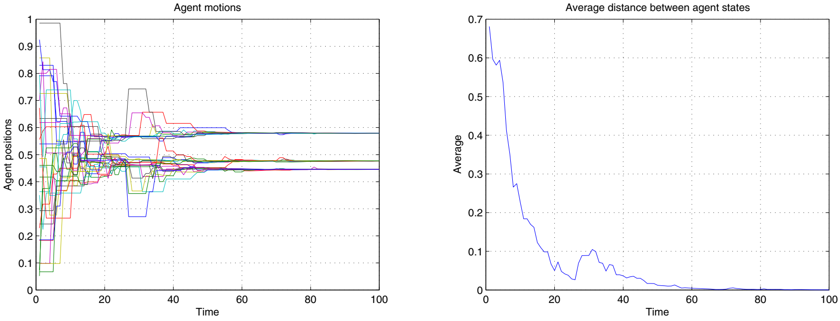
Fig. 9.3. Simulation for dynamic neighborhood topology ($\bar{p}_{sense} = 0.5$, $\bar{p}_{move} = 0.5$, and $\bar{p}_{nbr} = 0.5$).



(a) States of agents.

(b) Distance between agent states.

Fig. 9.4. Simulation for dynamic neighborhood topology ($\bar{p}_{sense} = 0.5$, $\bar{p}_{move} = 0.2$, and $\bar{p}_{nbr} = 0.9$).



(a) States of agents. (b) Distance between agent states.

Fig. 9.5. Simulation for dynamic neighborhood topology ($\bar{p}_{sense} = 0.2$, $\bar{p}_{move} = 0.5$, and $\bar{p}_{nbr} = 0.9$).

$\bar{p}_{move} = 0.2$ (while keeping $\bar{p}_{sense} = 0.5$ and $\bar{p}_{nbr} = 0.9$). This resulted in only a very slight difference compared to the simulation in Figure 9.2 and substantial increase in convergence speed is not achieved. This is probably because for the current implementation even though the agents move more often they still use the same amount of outdated information which prevents the achievement of a substantial increase in the convergence speed. For the simulation in Figure 9.5 the probability threshold for the agents to sense was decreased to $\bar{p}_{sense} = 0.2$ (while keeping $\bar{p}_{move} = 0.5$ and $\bar{p}_{nbr} = 0.9$). This also resulted in a faster convergence compared to the case in Figure 9.2. Note that the synchronous case corresponds to the case with $\bar{p}_{move} = 0$ and $\bar{p}_{sense} = 0$ (move at each step with always the current information) which both for the fixed and dynamic topology cases converges much faster compared to the corresponding asynchronous case with time delays considered here. Note also that the speed of convergence is not affected by the dimension of the state space n (since actually each dimension is independent and they do not effect each other). In contrast, the number of agents can effect the convergence rate as is expected from the bound obtained in the proof of Theorem 21. In particular, higher number of agents N combined with low connectivity properties of the interconnection graph may result in slower convergence since the length of the information flow paths in the spanning three can be large.

9.5 Further Issues

9.5.1 Application Areas of the Agreement Model

The distributed agreement model discussed in this chapter can be used for representing different applications/systems including distributed orientation alignment in schools of fish or bird flocks, distributed synchronization in fireflies, distributed decision making as in deciding on a new nest site by a swarm of honeybees and others. It can represent also the operation of self-propelled particle systems studied by physicists such as those considered in [46]. Moreover, it may serve as a crude representation of interactions in social networks (and adaptation of attitudes or beliefs, for example). More detailed information on social networks can be found in [1].

Robot gathering algorithms such as those considered in [74, 101, 149] can also be represented by the model in (9.1). In other words, (9.1) can be used for high-level representation of the rendezvous or gathering problem in asynchronous multi-vehicle systems. In particular, consider a networked system of agents which have continuous-time vehicle dynamics and have sensing/communication, computation, and motion capabilities. Assume that they operate on the (infinite) sequence of behaviors *wait-sense-compute-move*. Let at time index t agent i be located at $x_i(t)$. After performing a sensing of its current neighbors, during its *compute* state/behavior it computes using (9.1) the new *way point* (*position point* or *path point*) $x_i(t+1)$. Then, it moves (using some local control) in some (unspecified but bounded) amount of time to its new position (the *move* behavior), waits for some amount of time and performs new neighbor position sensing. Note that during the *wait*, *sense*, and *move* behaviors no new way points are computed which corresponds to (9.2). The agent does not need to know at which time instant the other agents move and there is no need for a global clock. The asynchronism in the system comes from the fact that the times for the completion of the behaviors and in particular of the *move* behavior by different agents is not necessarily uniform. The model in (9.1) provides a high-level view of such systems and the results developed in this chapter will hold for such systems as well. However, this representation is not concerned with the low-level vehicle dynamics and control of the agents. The only requirement is that appropriate low-level control/navigation algorithms should be designed such that the agents move to their next computed way points in some finite (unspecified but bounded) amount of time. In other words, in order for Assumption 18 to be satisfied the sequence *wait-sense-compute-move* must be completed in a finite amount of time.

Another issue to be emphasized here is that although the model in (9.1) seems to use global/actual positions/states of the agents it is suitable also for representing applications in which the relative positions/states can be measured (instead of gloabl/actual positions/states). To see this note that, by taking $w_{ij}(t) = 1, 1 \leq i, j \leq N$ and by rearranging, the model in (9.1) can be rewritten as

$$x_i(t+1) = x_i(t) + \frac{1}{N_i(t)+1} \sum_{j \in \mathcal{N}_i(t)} \left[x_j(\tau_{ij}(t)) - x_i(t) \right], \forall t \in T_i.$$

In some applications it might be easier to measure relative states instead of global states. Then one can use this relative information based model instead of (9.1) since they are equivalent under the assumption that $w_{ij}(t) = 1, 1 \leq i, j \leq N$.

9.5.2 Extensions and Generalizations

It is possible to extend the results discussed in this chapter in several ways. First of all the uniformity in the connectivity (occurrence of a spanning tree) in Assumption 16 can be removed if communication is bidirectional or if uniformity in the occurring cycles is assumed. A synchronous system under these conditions was analyzed in [175, 176] and the results there can be extended to the asynchronous system discussed in this chapter.

Once convergence of the synchronous system is established and the box condition in (9.8) is satisfied, then convergence of the corresponding asynchronous iteration follows from the asynchronous convergence theorem in [22]. (For more definitions and discussions see [22].) The asynchronous convergence theorem is a general result which is not limited to linear iterations and can be applied to nonlinear iterations as well. In fact, the two main arguments of the proof of convergence in the linear case are a convexity-type condition and the contraction properties (for more information on contraction mappings see [21]) of the iteration. Therefore, as long as these properties are preserved together with the box condition the results will still hold. Consider the nonlinear system in which individual i moves according to

$$x_i(t+1) = f_i\left(t, x_1(\tau_{i1}(t)), \ldots, x_N(\tau_{iN}(t))\right), \forall t \in T_i, \tag{9.15}$$

and it is stationary otherwise. This iteration can be also represented by $f_i(t, x_{pi}(t))$ where $x_{pi}(t) \in X \subset \mathbb{R}^{N \times n}$ (which is defined in (9.9)) is the perceived state of the system by agent i at time t and the time dependence in f_i is due to the time-dependent communication topology. For simplicity it is assumed that $f_i : \mathbb{N} \times X \to Y$ are continuous for all i. Also, assume that Assumption 16 is satisfied. Then, under certain convexity conditions it is possible to show that agreement will be achieved. In [176] Moreau considered a synchronous version of the nonlinear system in (9.15) with corresponding convexity type assumption and proved asymptotic agreement. With some manipulation and application of the asynchronous convergence theorem in [22] the result there can be extended to the asynchronous case. More concrete ideas on this can be found in [86].

The synchronous system in Equation (9.4) can be represented as a time varying discrete time system. To see this one can define the $(i,j)'th$ entry of a matrix $A(\mathcal{G},t)$ at time t as

$$[A(\mathcal{G},t)]_{ij} = \begin{cases} \frac{w_{ij}(t)}{w_i(t)}, & \text{if } (i,j) \in \mathcal{A}(t) \text{ or } i = j, \\ 0, & \text{otherwise,} \end{cases} \tag{9.16}$$

and the system can be represented as

$$x(t+1) = A(\mathcal{G},t)x(t), \tag{9.17}$$

where $x(t) = [x_1(t), \ldots, x_N(t)]^\top \in \mathbb{R}^{N \times n}$ represents the state of all agents at time t. (Note that the definition of $x(t)$ is different from before and here it is a matrix not a vector.) This is a discrete time system whose stability properties depend on the properties of the matrix $A(\mathcal{G},t)$. The matrix $A(\mathcal{G},t)$ is a stochastic matrix for all t. This directly follows from the definition of its elements in (9.16). A (row) stochastic matrix is a matrix with non-negative entries which are also less than one and with sum of each row equal to one. Stochastic matrices have the property that one of their eigenvalues is $\lambda = 1$ and all the other eigenvalues lie within the unit circle. Moreover, multiplication of stochastic matrices is also stochastic. (More information on stochastic matrices can be found in [118].) Using these properties and under

Assumption 16 for the synchronous system with the dynamics in (9.4), alternatively to the proof presented in this chapter, it is possible to show that the condition (9.3) is satisfied. This is the approach taken in [126, 203].

9.5.3 For Further Reading

This chapter is based on the work in [86]. The first work on distributed agreement was performed in the 80's in the context of parallel and distributed computation with the corresponding results collected in [22]. In this chapter we closely follow the notation in [22]. Moreover, the result for the asynchronous case can alternatively be proved using the Asynchronous Convergence Theorem in [22] (this was the approach taken in [86]). Similar work was done also on load balancing in computer networks or in flexible manufacturing systems in the framework of discrete event systems (DES) [192]. The analysis in [192] uses metric spaces and Lyapunov approach and the proofs in this chapter are inspired by the treatment there. Moreover, the book [169] is a good reference for stability analysis of general dynamical systems in metric spaces. Directed graphs are commonly used to represent interaction topologies in multi-agent dynamical systems (as we did here). More information on graph theory can be found in [100] (or any other text on graph theory).

Recent studies on distributed agreement or consensus or closely related problems can be found in [10, 24, 71, 72, 126, 175, 176, 184, 203, 216]. In [126] Jadbabaie and coworkers considered a simple model of n interacting particles with time-dependent bidirectional communication links and showed that the one-dimensional system state (the heading in their case) will converge to the same value provided that the union of the communication graphs is uniformly jointly connected. Ren and Beard [203] extended the results to unidirectional communication and relaxed the connectivity assumption to the assumption that the union of the communication graphs has a spanning tree. The works in [126, 203] have some common aspects with those in [22] but also have some differences. Independently, Moreau [176] considered a more general nonlinear interaction model and showed that under unidirectional communication agreement will be achieved if for any uniformly bounded time interval there is an agent which is connected to all other agents (equivalent to the spanning tree assumption of [203]), whereas for bidirectional communication the same will be achieved without uniformity in connectedness. Later in [175] the same author relaxed the uniformity in connectedness assumption for the unidirectional case as well, but assumed uniformity in the communication cycles in the graph. The results in [175, 176] are based on convexity analysis and are more general than those in both [126] and [203]. Similarly, in [10] Angeli and Bliman provide an extension of the result by Moreau [176] by relaxing the convexity assumption and allowing for a known and bounded time-delay. Another relevant reference is the work in [150], where the authors consider a group of unicycles and show that the rendezvous problem is solvable (i.e., a controller achieving stabilization to a point exits) if and only if the communication graph has a globally reachable node or basically a spanning tree (in contrast to the work in [126] and [203] where the analysis is based on a specific control law). However, they consider only the fixed communication topology case. The article in [71] briefly discusses asynchronous protocols, poses some open questions, and shows some simulation based preliminary results on asynchronous protocols using a custom Java based simulator. The article in [24], besides discussing the current results in the literature, presents some new results for synchronous systems/protocols with delays as well. More detailed treatment of consensus algorithms can be found also in the book [204].

In [72] the authors emphasize the role of information flow and graph Laplacians and derive Nyquist-like criterion for stabilizing vehicle formations. In [184] Olfati-Saber and Murray describe consensus protocols for networks of dynamic agents with fixed and switching communication topologies and show that connectivity of the network is key in reaching consensus. They determine a class of directed communication graphs which guarantee reaching average consensus and they establish connection with the Fiedler eigenvalue of the graph Laplacian and the speed of convergence. Moreover, they also consider time delays and channel filtering affects. In [216] Sepulchre and coworkers study connections between models of coupled phase oscillators and kinematic models of swarms (groups of self-propelled particles) and design control laws for stabilizing collective motions of groups.

The distributed agreement model in (9.1) has distinguishing properties: (i) the agents update their states in asynchronous manner; (ii) they do not necessarily have the exact information about the states of the other agents. Therefore, it is more realistic and more suitable for describing distributed agreement in multi-agent dynamic systems since these features are natural properties of such systems.

10

Formation Control with Potential Functions and Newton's Iteration

10.1 Path Planning for a Single Agent

In this chapter, we consider again the formation control problem. However, we take a different approach for solving the problem. In particular, we consider a discrete-time model for agent motion dynamics. Such a model does not necessarily represent dynamics of physical agents. However, one can view the model as a high-level representation which generates way-points for the physical agents. In other words, it serves as an iterative path planner for the physical agents. In that case, each agent should poses a low-level controller which should guarantee that the agent will move to the next way-point in a finite time. We will first describe the procedure for a single agent to move from its current position to one of the desired target locations in an environment with obstacles. Following that, we will discuss how the procedure can be modified to solve the formation control problem. The main idea is to define polynomial type potential functions such that the positions of the targets and obstacles constitute the zeros of these functions. Then using an iterative strategy which generates the way-points based on the negative gradient of the target function, and the positive gradient of the obstacle function, solve for the zeros of the target function (i.e., move towards the target positions) and avoid the zeros of the obstacle function (i.e., avoid the obstacles).

As was mentioned above, polynomial type artificial potential functions are used to define the positions of the targets and obstacles in the state space. We will now describe how these functions are being generated. Consider an n-dimensional position space and let $y = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ denote any vector in $\mathbb{R}^n$. Assume that there are m targets in the space and represent this set of targets with

$$H_T(y) = \{t_i \in \mathbb{R}^n | i = 1, \ldots, m\}$$

where $t_i = [t_{i1}, \ldots, t_{in}]^\top \in \mathbb{R}^n$ represents the position of the i'th target. Define the corresponding target function $F_T : \mathbb{R}^n \to \mathbb{R}^n$ as

$$F_T(y) = \begin{Bmatrix} f_1(y) \\ f_2(y) \\ \vdots \\ f_n(y) \end{Bmatrix} \tag{10.1}$$

such that for every $t_i \in H_T(y)$ we have $F_T(t_i) = 0$. In other words, given a set of targets $H_T(y)$ the corresponding nonlinear target function $F_T(y)$ is defined such that every $t_i \in H_T(y)$ is a root of $F_T(y)$. To this end, the first components t_{i1} of all target vectors t_i are used to form the function $f_1(y)$ in the form

$$f_1(y) = y_1 y_2 - \left[y_1 P_2^{m-1}(y_1) - (y_1 - t_{11})(y_1 - t_{21}) \dots (y_1 - t_{m1}) \right] \tag{10.2}$$

Similarly, the j'th and $(j-1)$'th components of the t_i vectors are used to create the function $f_j(y)$. In other words, the j'th and $(j-1)$'th components of the vectors t_i are taken into account such that the zeros of the function $f_j(y)$ are given by

$$\left[(t_{1(j-1)}, t_{1(j)}), (t_{2(j-1)}, t_{2(j)}), \dots, (t_{m(j-1)}, t_{m(j)}) \right] \tag{10.3}$$

This can be achieved by defining $P_j^{m-1}(y_{j-1})$ as an $(m-1)$'th order polynomial for which we have $P_j^{m-1}(t_{i(j-1)}) = t_{i(j)}$ for all targets $i = 1, \dots, m$ and components $j = 2, \dots, n$ and choosing $f_j(y)$ as

$$f_j(y) = y_j - P_j^{m-1}(y_{j-1})$$

Then, the target function constructed this way can be written as

$$F_T(y) = \begin{Bmatrix} f_1(y_1, y_2) \\ f_2(y_1, y_2) \\ f_3(y_2, y_3) \\ \vdots \\ f_n(y_{n-1}, y_n) \end{Bmatrix} = \begin{Bmatrix} y_1 y_2 - P_1^m(y_1) \\ y_2 - P_2^{m-1}(y_1) \\ y_3 - P_3^{m-1}(y_2) \\ \vdots \\ y_n - P_n^{m-1}(y_{n-1}) \end{Bmatrix} \tag{10.4}$$

This target function is a special type of artificial potential function that has zeros at the target points. For a better understanding of the procedure, we provide a numerical example. Let the dimension of the space be $n = 2$, the number of targets be $m = 3$ and the positions of the targets be given as $t_1 = [1, 1], t_2 = [2, 3], t_3 = [3, 8]$. In this case, the $P_2^{m-1}(y_1) = P_2^2(y_1)$ polynomial becomes a second order polynomial in the form of

$$P_2^2(y_1) = c_1(y_1 - t_{11})(y_1 - t_{21}) + c_2(y_1 - t_{21})(y_1 - t_{31}) + c_3(y_1 - t_{11})(y_1 - t_{31}) \tag{10.5}$$

where c_1, c_2, and c_3 are parameters which are calculated such that the condition in (10.3) is satisfied. In other words, the values of c_1, c_2, and c_3 are calculated using the conditions

$$P_2^2(t_{j1}) = c_1(y_1 - t_{11})(y_1 - t_{21}) + c_2(y_1 - t_{21})(y_1 - t_{31}) + c_3(y_1 - t_{11})(y_1 - t_{31}) = t_{j2} \tag{10.6}$$

for all targets $j = 1,2,3$. This means that $(y_2 - P_2^2)$ has zeros at t_1, t_2, and t_3. Solving equations (10.6) for the above targets one can obtain the values $c_1 = 4$, $c_2 = 0.5$, and $c_3 = -3$ for the parameters in equation (10.5).

Given the above $P_2^2(y_1)$, the $P_1^3(y_1)$ polynomial becomes a third order polynomial of the form

$$P_1^3(y_1) = y_1 P_2^2(y_1) - (y_1 - t_{11})(y_1 - t_{21})(y_1 - t_{31}) \tag{10.7}$$

Using these, the corresponding target function $F_T(y)$ can be calculated as

$$F_T(y) = \left\{ \begin{array}{l} y_1 y_2 - (0.5y_1^3 + 3.5y_1^2 - 9y_1 + 6) \\ y_2 - 1.5y_1^2 + 2.5y_1 - 2 \end{array} \right\} \tag{10.8}$$

Thus, one can see that $F_T(y)$ has zeros at t_1, t_2, and t_3, i.e., $F_T(t_j) = 0$ for all targets $j = 1,2,3$. Then, given any initial position y, moving to a desired target position is analogous to finding (solving for) a zero of the function $F_T(y) = 0$. Moreover, by using an iterative method for solving the problem one can also simultaneously generate the path from the current position to the target position. There are various methods for solving this type of problem; here we use Newton's method. We would like to emphasize that other methods can be used as well.

Having constructed the target function $F_T(y)$, the next step is to solve the equation $F_T(y) = 0$ by an iterative method, and as we mentioned above, we will use a Newton iteration. Given that the agent is located at position $y(k)$ at iteration k its position can be updated based on the Newton's method as

$$y(k+1) = y(k) + \lambda \Delta y_a(k) \tag{10.9}$$

where λ is a step size parameter and the step vector is given by

$$\Delta y_a(k) = -\Big[\nabla F_T(y(k))\Big]^{-1} F_T(y(k)). \tag{10.10}$$

Note here that the step $\Delta y_a(k)$ is an attraction step towards one of the targets. The agent moves towards the target within whose basin of attraction it is located (which is usually the closest target). The parameter $\lambda > 0$ is a step size parameter which determines the coarseness of the steps. It is needed because adding $\Delta x_a(k)$ directly to the current position may result in large step size due to large relative difference between the agent and the targets and this may lead to convergence problems. Another issue to mention here is that the above implementation does not guarantee avoidance of collisions with possible obstacles in the space. In order to add collision avoidance one can use an approach that is in principle similar to moving towards the targets, with the difference of moving in the opposite direction (in order to avoid the obstacles). In other words, given a set of r obstacles in the space

$$H_O(y) = \{o_i \in \mathbb{R}^n | i = 1, \ldots, r\}$$

where $o_i = [o_{i1}, \ldots, o_{in}]^\top \in \mathbb{R}^n$ represents the position of the i'th obstacle, one can define the corresponding obstacle function as $F_O : \mathbb{R}^n \to \mathbb{R}^n$ such that $F_O(o_i) = 0$ for all $o_i \in H_O(y)$. Then, the motion of the agent can be modified as

$$y(k+1) = y(k) + \lambda\Big(\Delta y_a(k) + \Delta y_r(k)\Big) \tag{10.11}$$

where $\Delta y_a(k)$ is as defined in (10.10) and

$$\Delta y_r(k) = \begin{cases} \frac{1}{\left(1+\left(\frac{\|R(k)\|}{Cr}\right)^{\mu}\right)\|R(k)\|^3} R(k) - \frac{\varepsilon_r}{\left(1+\left(\frac{\varepsilon_r}{Cr}\right)^{\mu}\right)\varepsilon_r^3}\mathbf{1}, & \rho(y(k), H_O(y)) < \varepsilon_r, \\ 0, & \rho(y(k), H_O(y)) \geq \varepsilon_r. \end{cases} \tag{10.12}$$

Here, $R(k)$ constitutes a repulsion vector at step k and is calculated as

$$R(k) = +\Big[\nabla F_O(y(k))\Big]^{-1} F_O(y(k)) \tag{10.13}$$

$\rho(y(k), H_O(y))$ is the distance between $y(k)$ and the set $H_O(y)$, and $\mathbf{1} = [1, \ldots, 1]^\top \in \mathbb{R}^n$ is a vector of ones. Once again the step size parameter λ serves as a scaling factor of the step to achieve better accuracy and convergence. An issue to note here is that with the above formulation, the repulsion is only invoked if the agent is a very close distance ε_r to an obstacle. Note also that in order to guarantee avoidance of collisions in close vicinity of obstacles the repulsion force dominates since it grows unbounded as the agent gets closer to an obstacle as shown in Figure 10.1. The parameters C_r and μ determine the shape and the steepness of the repulsion function as can be seen in Figure 10.1. In particular, decreasing μ or increasing C_r increases its steepness and results in faster growth of the repulsion with respect to the decrease in the value of $R(k)$.

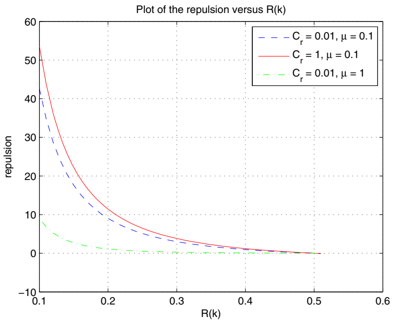


Fig. 10.1. Plot of the value of the repulsion versus R(k) for different values of C_r and μ (in one-dimensional space).

Having set the framework for a single agent in this section, in the next section we will return to the main problem under consideration–which is the formation control problem–and discuss how we can use the above framework and ideas to solve the problem.

10.2 Controller Development

Consider a swarm consisting of N agents which are moving in an n-dimensional space. Let $x_i = [x_{i1}, \ldots, x_{in}]^\top \in \mathbb{R}^n$ denote the position vector of agent i. Since we are mainly concerned with the formation control problem, the objective of every agent is to achieve a predefined geometrical shape or a formation pattern in coordination with other agents. The approach is based on the strategy discussed in the preceding section. Therefore, the method is iterative and the agents update their motion plans at each step utilizing the new position information of other agents.

The approach consists of three steps, the first of which is to define for every agent its desired relative positions with respect to the other agents as mobile targets and the other agents themselves as mobile obstacles. The second step is to generate the corresponding target and obstacle functions (which are basically polynomial potential functions as was discussed in the preceding section) based on these defined target/obstacle positions. The third step is to update the positions of the agents based on their time varying target and obstacle functions. We utilize Newton's update rule for updating the positions of the agents.

In order to achieve the desired formation (i.e., the desired predefined geometric shape), given any initial positions, the agents should move and locate themselves at the desired inter-agent distances. Assume that the desired formation is defined by the desired inter-agent distances or formation constraints

$$\|x_i - x_j\| = d_{ij}$$

for all $i, j = 1, ..., N,\ j \neq i$. Here $d_{ij} > 0$ represents the inter-agent distance between agents i and j in the desired formation. In order for the problem to be solvable these distances should be non-conflicting.

Guided by the discussion for the single agent case presented in the preceding section, in order to develop a strategy which moves the agents such that to achieve the desired formation, we define the target set for each agent as the set of points that consists of the points defined at the desired distances from the other agents. In other words, given an agent i, the set of its targets $j = 1, ..., N,\ j \neq i$ at time k are defined as

$$H_T^i(k) = \left\{ t_j(k) = x_j(k) + \frac{d_{ij}}{\|x_i - x_j\|}\Big(x_i(k) - x_j(k)\Big) | j = 1, \ldots, N,\ j \neq i \right\}.$$

Similarly, in order to avoid collisions between agents, they are viewed as obstacles to each other. In other words, the obstacle set for agent i at time k is defined as

$$H_O^i(k) = \{ o_j(k) = x_j(k) | j = 1, \ldots, N,\ j \neq i \}.$$

In order to further illustrate the procedure let us assume that there are three agents which are required to form an equilateral triangle with edge lengths d as shown in Figure 10.2. For this case, there exist two non-stationary (moving) target points for every agent at every step and the target points for a given agent are located on the lines that connect the agents to the other agents and at a distance d from these

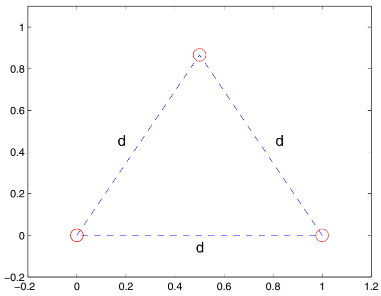


Fig. 10.2. Example equilateral triangle formation.

agents as shown in Figure 10.3. For example, for the agent in the lower left corner in Figure 10.3 the target set consists of the points labeled as stars. Similarly for the agent located on the upper left corner the target set consists of the points labeled as triangles. Note that at each step–after the motion of the agents–the positions of these points change and therefore, the target sets are time-varying.

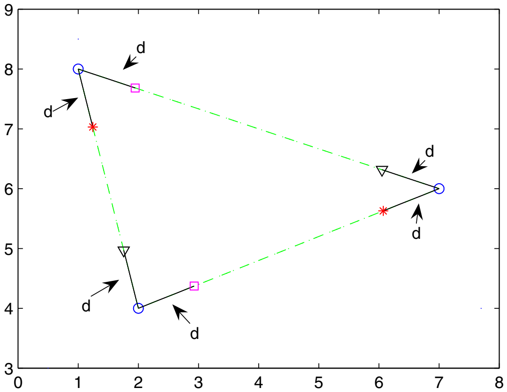


Fig. 10.3. The positions of the targets for each agent.

It is obvious that there are $(N-1)$ target points in the set $H_T^i(k)$. These targets are defined by the desired distances d_{ij} of the agent to the other agents in the desired formation. Note that here the problem is specified assuming that there is a predefined desired distance d_{ij} between all possible agent pairs (i,j) and the agents should try to simultaneously keep their distance to all other agents $j \neq i$ (i.e., for simplicity it is assumed that the interaction/constraints graph is fully connected or complete). However, we would like to emphasize here as well that it is possible to define a unique formation by using graph-theoretic concepts, and in particular the concept of rigid graphs with minimum connections, and choosing a smaller number of desired inter-agent distances and therefore fewer targets for each agent. To avoid collisions, the set of obstacles should still in principle include all the other agents. However, practically for a given agent the set of obstacles consists only of the other agents which are very close to it. In other words, the repulsion between agents is invoked only if agents get to a pre-specified repulsion distance to each other and there is no need to maintain constant complete connection graph for repulsion.

In order to reduce computational complexity, we will further redefine the target and obstacle sets. In particular, for each agent i we will redefine the target and obstacle sets as follows

$$H_T^i(k) = \bigcup_{j=1,\dots N, j\neq i} H_T^{ij}(k) \quad \text{and} \quad H_O^i(k) = \bigcup_{j=1,\dots N, j\neq i} H_O^{ij}(k)$$

where

$$H_T^{ij}(k) = \left\{ t_j(k) = x_j(k) + \frac{d_{ij}}{\|x_i - x_j\|}\Big(x_i(k) - x_j(k)\Big),\ j \neq i \right\}$$

and

$$H_O^{ij}(k) = \left\{ o_j(k) = x_j(k),\ j \neq i \right\}$$

consist of only one target and one obstacle, respectively. In other words, $H_T^{ij}(k)$ contains the target for agent i based on only agent $j \neq i$. Similarly, $H_O^{ij}(k)$ contains the obstacle for agent i based on only agent $j \neq i$. Then, given agent i we generate separate target and obstacle functions $F_T^{ij}(k)$ and $F_O^{ij}(k)$ for the j'th agent ($j \neq i$) instead of single target and obstacle functions $F_T^i(k)$ and $F_O^i(k)$ that cover all of the targets and all of the obstacles, respectively. Then the attraction step of agent i is generated based on

$$\Delta x_{ai}(k) = \sum_{j=1, j\neq i}^{N} \frac{1}{\|A_{ij}(k)\|} A_{ij}(k) \tag{10.14}$$

where

$$A_{ij}(k) = -\Big[\nabla F_T^{ij}(k)\Big]^{-1} F_T^{ij}(k). \tag{10.15}$$

Similarly, its repulsion step is generated as follows

$$\Delta x_{ri}(k) = \sum_{j=1, j\neq i}^{N} \bar{R}_{ij}(k) \tag{10.16}$$

where

$$\bar{R}_{ij}(k) = \begin{cases} \frac{1}{\left(1+\left(\frac{\|R_{ij}(k)\|}{C_r}\right)^{\mu}\right)\|R_{ij}(k)\|^3} R_{ij}(k) - \frac{\varepsilon_r}{\left(1+\left(\frac{\varepsilon_r}{C_r}\right)^{\mu}\right)\varepsilon_r^3}, & \|x_i(k) - x_j(k)\| < \varepsilon_r, \\ 0, & \|x_i(k) - x_j(k)\| \geq \varepsilon_r, \end{cases} \tag{10.17}$$

and

$$R_{ij}(k) = +\Big[\nabla F_O^{ij}(k)\Big]^{-1} F_O^{ij}(k). \tag{10.18}$$

Note that with this type of definition of the target and obstacle sets one needs to define $2N(N-1)$ target/obstacle functions for N agents. However, this results in polynomials which are only first order which leads to a significant decrease in computational complexity.

In this case, for each agent i we have $N-1$ number of target sets $H_T^{ij}(k)$ and equivalently for each of them there exists $n \times 1$ dimensional target function $F_T^{ij}(k)$. The $P_1^m(k)$, $P_2^{m-1}(k)$, and $P_n^{m-1}(k)$ polynomials in equation (10.4) which are used to form the target and obstacle functions simplify significantly. In particular, the $P_2^{m-1}(k) = P_2^0(k)$ to $P_n^{m-1}(k) = P_n^0(k)$ polynomials become constant numbers (i.e., polynomials of degree zero) and $P_1^m(k) = P_1^1$ turns into a first order polynomial. In other words, for agent i we have

$$P_2^{ij} = t_{j2}(k), \ldots, P_n^{ij} = t_{jn}(k)$$

and

$$P_1^{ij} = x_{i1}(k)t_{j2}(k) - (x_{i1}(k) - t_{j1}(k))$$

Here $t_{j1}(k)$ to $t_{jn}(k)$ are the components of the target vectors $t_j(k)$. Note that the value of $t_j(k)$ can be different at each step since the agents move.

As was the case in the preceding section, we use a step size parameter here as well and update the position of agent i based on

$$x_i(k+1) = x_i(k) + \lambda_i \Delta x_i(k) \tag{10.19}$$

where $\lambda_i > 0$ is the step size parameter to be determined by the designer and $\Delta x_i(k)$ is the unit step vector determining the direction of motion which is calculated as

$$\Delta x_i(k) = \frac{\Delta x_{ai}(k) + \Delta x_{ri}(k)}{\|\Delta x_{ai}(k) + \Delta x_{ri}(k)\|}. \tag{10.20}$$

Here we assume that each agent might have different step size parameter λ_i.

Since the method we consider here is discrete, once the robots achieve the formation they start to oscillate around their target point with a deviation which is proportional to the step size. In other words, the step size parameter λ determines also the final accuracy in the formation. In order to further improve the formation accuracy one can employ an adaptive step size such that adaptation is invoked whenever the formation error is sensed to be small. In particular, one possible straightforward implementation of adaptive step size could be to decrease the step size once the formation error for a given agent is less than twice its current step size. In other words, for any agent i basically set

$$\text{if } \xi_{ij} < 2\lambda_i \ \ \forall j \neq i \ \ \text{then} \ \ \lambda_i = \alpha_i \lambda_i \tag{10.21}$$

where $0 < \alpha_i < 1$ is a scaling down parameter, ξ_{ij} is given by $\xi_{ij} = |\delta_{ij} - d_{ij}|$, and $\delta_{ij} = \|x_i - x_j\|$ and d_{ij} are, respectively, the current and the desired distance between agents i and j.

Finally, we would like to also stress here that although the procedure was described considering the formation control problem, it is very easy to accommodate also the problem of distributed agreement within this framework. The only modification which is needed for that purpose is to set $d_{ij} = 0$ for all pars (i, j) and to remove inter-agent collision avoidance. Similarly, it can be modified to handle the aggregation, social foraging, and swarm tracking problems as well.

10.3 Simulation Examples

In this section we present illustrative numerical simulation examples. We perform simulations for $n = 2$ and $n = 3$ dimensional spaces. For $n = 2$ there are $N = 6$ agents in the swarm which are required to form an equilateral triangle formation with edge lengths equal to $d = 6$ and to arrange themselves such that three of the agents are located at the corners of the triangle, whereas the other three agents are located at the intermediate points of the edges. Therefore, depending on the relative position in the desired formation, the distances between agents is either $d = 6$, $d/2 = 3$, or $(d/2)\sqrt{3} = 3\sqrt{3}$. For the case of $n = 3$ there are $N = 8$ agents in the swarm which are required to form a tetrahedron shape with edge length equal to $d = 6$. Here again four of the agents are required to locate themselves at the corners of the tetrahedron, while the remaining four are required to locate at the center points of its side faces. Therefore, depending on the relative position in the desired formation, there are four possible desired inter-agent distances which are $d = 6$, $d/3 = 2$, $d/\sqrt{3} = 2\sqrt{3}$, and $d\sqrt{2/3} = 2\sqrt{6}$. We use an adaptive step size which is initialized as $\lambda_i = 0.1$ for all agents $i = 1, ..., N$, and the adaptation is performed as in equation (10.21) with $\alpha_i = 0.5$ for all i. The other parameters we have used in the simulations below are the repulsion range $\varepsilon_r = 0.5$, and the repulsion function parameters $C_r = 0.01$ and $\mu = 0.1$. The simulations were set to stop either if the number of steps/iterations reached the predefined number of maximum iterations $N_{max} = 400$ or if the average formation error becomes smaller than $\varepsilon_{stop} = 0.001$.

Figure 10.4 shows the trajectories of the agents during a simulation for $n = 2$ in which $N = 6$ agents form an equilateral triangle formation. Final relative positions and the formed geometric shape are also shown in the figure. As can be seen from the figure, the agents arrange themselves in the desired formation. The simulation lasts for about 163 steps and stops once the formation error decreases below $\varepsilon_{stop} = 0.001$. Checking the step size of the agents after stopping one can see that adaptation has been employed and the step size has been decreased to $\lambda_i < 0.001$ for all i.

Figure 10.5 shows the plots of the inter-agent distances and the average distance error (the formation error)

$$e(k) = \frac{1}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i}^{N} \left| \|x_i(k) - x_j(k)\| - d_{ij} \right|.$$

As can be seen from Figure 10.5(a) the inter-agent distances converge to the desired values of $d = 6$, $d/2 = 3$, and $(d/2)\sqrt{3} = 3\sqrt{3}$. Similarly, one can observe in Figure 10.5(b) that the formation error converges to zero (actually it converges to a value less than $\varepsilon_{stop} = 0.001$).

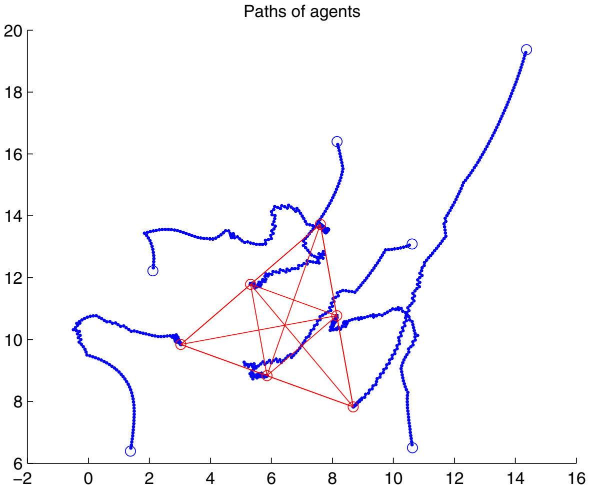
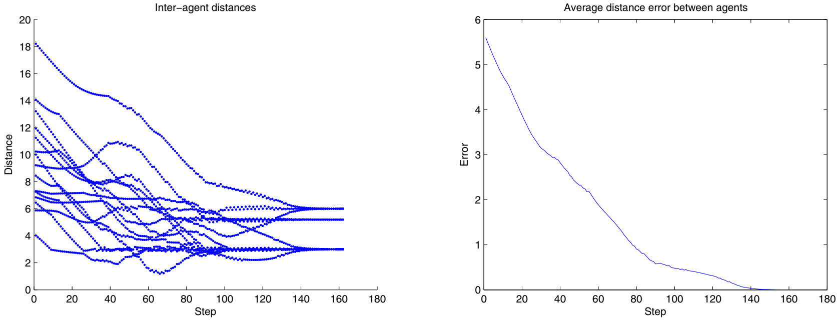


Fig. 10.4. The paths of $N = 6$ agents forming an equilateral triangle formation.



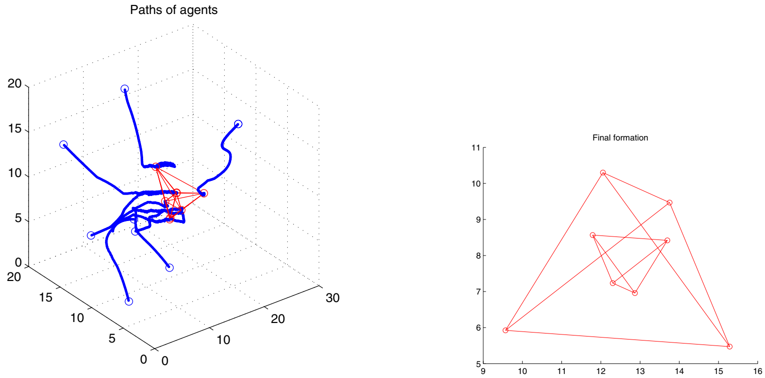
(a) Inter-agent distances.

(b) Average distance error.

Fig. 10.5. Equilateral triangle formation ($n = 2$, $N = 6$): distance between agents and average inter-agent distance error.

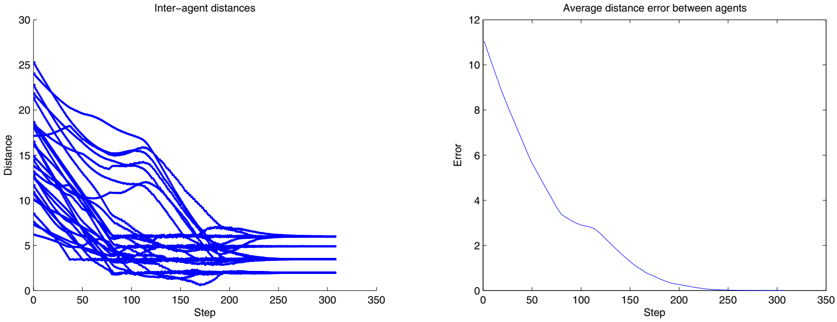
Figure 10.6 shows simulation results for $n = 3$ in which $N = 8$ agents are required to form a tetrahedron formation. The paths of the agents as well as the formation achieved can be seen in Figure 10.6(a). The final formation is also depicted in Figure 10.6(b) for better visualization. To make it even clearer, some of the inter-agent connection lines are omitted. In other words, although the simulation was performed using a complete graph connection topology some of these connection lines are not depicted in Figure 10.6(b) to prevent confusion in the shape of the formation.

This simulation lasted for about 309 steps. Adaptive step size was employed again and it was observed once more that after stopping, the step size for all agents i has been decreased to $\lambda_i < 0.001$. The plots of the inter-agent distances and the



(a) Agent paths. (b) Final formation.

Fig. 10.6. Tetrahedron formation ($n = 3$, $N = 8$): agent paths and the final formation.



(a) Inter-agent distances. (b) Average distance error.

Fig. 10.7. Tetrahedron formation ($n = 3$, $N = 8$): distance between agents and average inter-agent distance error.

formation error with respect to the step k are shown in Figure 10.7. As can be seen from the figure, the formation error converges to zero and the inter-agent distances converge to the desired distances of $d = 6$, $d/3 = 2$, $d/\sqrt{3} = 2\sqrt{3}$, and $d\sqrt{2/3} = 2\sqrt{6}$.

We would like to also mention here that as all the potential function methods, and also due to the properties of the Newton's method, the agents might sometimes get stuck on a local minimum. Therefore, the objective of forming the formation might be unsuccessful. To prevent or overcome such situations one can carefully re-assign the neighborhood desired distance requirements so that the current agent positions are "close enough" to the final formation. In other words, one can carefully re-assign the neighbor distance requirements such that the current position of the agents is close to their relative position in the desired formation so that to avoid local minima. Other procedures for avoiding local minima can also be employed.

10.4 Further Issues

10.4.1 Extensions and Generalizations

As was already mentioned in the preceding sections the method described in this chapter is based on artificial potential functions and Newton's iteration. It is high-level path planning procedure which is independent of low-level dynamics of the agents and therefore can be applied to different types of agents or systems of heterogeneous agents. As this is a big advantage, it requires also that the agents possess low-level controllers which will guarantee that they move from their current way-points to their next way-points in finite time.

One shortcoming of the procedure described is that similar to the other potential functions based strategies it can suffer from the so-called local minima problem. Therefore, strategies to avoid local minima might be incorporated in conjunction to the method described. The method can suffer also from the limitations of the Newton's iteration. Therefore, alternative iteration strategies can also be considered.

As the reader is already aware, the present strategy can easily be adapted to solve the problem of distributed agreement. The only modification which is needed for that purpose is to set $d_{ij} = 0$ for all pairs (i, j) and to remove inter-agent collision avoidance. The other main difference from the framework here could be that the neighborhood relations for the distributed agreement case might be dynamic. It can also be modified to handle the aggregation, social foraging, and swarm tracking problems as well.

10.4.2 For Further Reading

This chapter is based on the work in [114]. The work there has been also inspired by earlier results obtained by Erkmen and coworkers in [70]. In order to solve for the zeros of the target function (and to set the paths for the agents) we used Newton's iteration method in this chapter. However, although we have not tested them, we believe that other possible iterative methods solving the zeros of a function can be equally used. For alternative strategies one can consult any book on numerical methods.

Part IV

Swarm Based Optimization Methods

11

Bacteria Foraging Optimization

Natural selection tends to eliminate animals with poor "foraging strategies" (methods for locating, handling, and ingesting food) and favor the propagation of genes of those animals that have successful foraging strategies since they are more likely to enjoy reproductive success (they obtain enough food to enable them to reproduce). After many generations, poor foraging strategies are either eliminated or shaped into good ones. Such evolutionary principles have led scientists to hypothesize that it is appropriate to model the activity of foraging as an optimization process. In this chapter, we first explain the biology and physics underlying the chemotactic (foraging) behavior of *E. coli* bacteria. Next, we introduce an algorithmic optimization model of *E. coli* foraging behavior. Finally, we show that this algorithm can perform optimization for a multiple-extremum function minimization problem.

11.1 Bacterial Foraging by *E. coli*

The *E. coli* bacterium has a plasma membrane, cell wall, and capsule that contain, for instance, the cytoplasm and nucleoid. Up to six flagella are used for locomotion. The cell is about $2\mu m$ in length and weighs about 1 picogram. When *E. coli* grows, it gets longer, then divides in the middle into two "daughters." Given sufficient food and held at the temperature of the human gut (one place where they live) of 37 deg. C, *E. coli* can synthesize and replicate everything it needs to make a copy of itself in about 20 min. Mutations in *E. coli* occur at a rate of about 10^{-7} per gene, per generation.

11.1.1 Swimming and Tumbling

E. coli locomotion is achieved via a set of relatively rigid flagella that enable it to "swim" via each of them rotating in the same direction at about $100 - 200$ revolutions per second. Each flagellum is a left-handed helix configured so that as the base of the flagellum (i.e., where it is connected to the cell) rotates counterclockwise, as viewed from the free end of the flagellum looking towards the cell, it produces a force against the bacterium, so it pushes the cell. You may think of each flagellum as a type of propeller. If a flagellum rotates clockwise, then it will pull at the cell. From an engineering perspective, the rotating shaft at the base of the flagellum is quite an interesting contraption that seems to use what biologists call a "universal joint" (so the rigid flagellum can "point" in different directions, relative to the cell). In addition, the mechanism that creates the rotational forces to spin the flagellum in either direction is described by biologists as being a biological "motor."

An *E. coli* bacterium can move in two different ways: it can "run" (swim for a period of time) or it can "tumble," and it alternates between these two modes of operation its entire lifetime (i.e., it is rare that the flagella will stop rotating). First, we explain each of these two modes of operation. Following that, we will explain how it decides how long to swim before it tumbles, which will enable it to perform "nutrient hill climbing."

If the flagella rotate clockwise, each flagellum pulls on the cell and the net effect is that each flagellum operates relatively independent of the others and so the bacterium "tumbles" about (i.e., the bacterium does not have a set direction of movement and there is little displacement). See Figure 11.1(a). To tumble after a run, the cell slows down or stops first. Since bacteria are so small they experience almost no inertia, only viscosity, so that when a bacterium stops swimming, it stops within the diameter of a proton. Call the time interval during which a tumble occurs a "tumble interval." Under certain experimental conditions (an isotropic, homogeneous medium—one with no nutrient or noxious substance gradients) for a "wild type" cell (one found in nature), the mean tumble interval is about 0.14 ± 0.19 sec. (mean ± standard deviation, and it is exponentially distributed) [19, 20]. After a tumble, the cell will generally be pointed in a random direction, but there is a slight bias toward being placed in the direction it was traveling before the tumble.

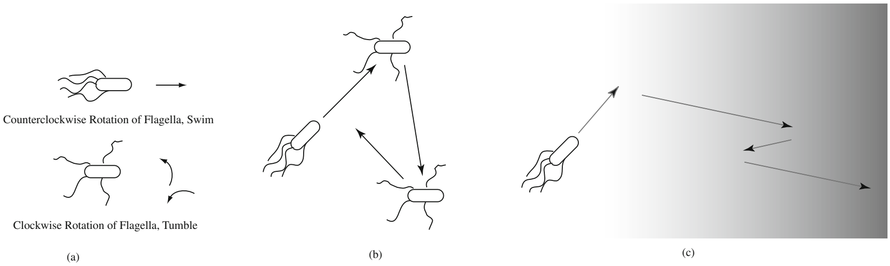

Counterclockwise Rotation of Flagella, Swim
Clockwise Rotation of Flagella, Tumble
(a)
(b)
(c)

Fig. 11.1. Bundling phenomenon of flagella shown in (a), swimming and tumbling behavior of the *E. coli* bacterium is shown in (b) in a neutral medium and in (c) where there is a nutrient concentration gradient, with darker shades indicating higher concentrations of the nutrient. (Note: Relative sizes of the bacteria and lengths of runs are not to scale.)

If the flagella move counterclockwise, their effects accumulate by forming a "bundle" (it is thought that the bundle is formed due to the viscous drag of the medium) and hence, they essentially make a "composite propeller" and push the bacterium so that it runs (swims) in one direction (see Figure 11.1(a)). On a run, bacteria swim at a rate of about $10 - 20$ μmeters/sec., but in a rich medium they can swim even faster [162]. Call the time interval during which a run occurs the "run interval." Under certain experimental conditions (an isotropic, homogeneous medium—the same as the one mentioned above) for a wild type cell, the mean run interval is about 0.86 ± 1.18 sec. (and it is exponentially distributed) [19, 20]. Also, under these conditions, the mean speed is $14.2 \pm 3.4\, \mu m/sec$. Runs are not perfectly straight since the cell is subject to Brownian movement that causes it to wander off course by about 30 deg. in 1 sec. in one type of medium, so this is how much it typically can deviate on a run. In a certain medium, after about 10 sec. it drifts off course more than 90 deg. and hence, essentially forgets the direction it was moving [19].

11.1.2 Chemotaxis and Climbing Nutrient Gradients

The motion patterns (called "taxes") that the bacteria will generate in the presence of chemical attractants and repellents are called "chemotaxes." For *E. coli*, encounters with serine or aspartate result in attractant responses, while repellent responses result from the metal ions Ni and Co, changes in pH, amino acids like leucine, and organic acids like acetate. What is the resulting emergent pattern of behavior for a whole group of *E. coli* bacteria? Generally, as a group they will try to find food and avoid harmful phenomena, and when viewed under a microscope, you will get a sense that a type of intelligent behavior has emerged, since they will seem to intentionally move as a group.

To explain how chemotaxis motions are generated, we simply must explain how the *E. coli* decides how long to run since, from the above discussion, we know what happens during a tumble or run. First, note that if an *E. coli* is in some substance that is neutral, in the sense that it does not have food or noxious substances, and if it is in this medium for a long period of time (e.g., more than one minute), then the flagella will simultaneously alternate between moving clockwise and counterclockwise so that the bacterium will alternately tumble and run. This alternation between the two modes will move the bacterium, but in random directions, and this enables it to "search" for nutrients (see Figure 11.1(b)). For instance, in the isotropic homogeneous environment described above, the bacteria alternately tumble and run with the mean tumble and run lengths given above, and at the speed that was given. If the bacteria are placed in a homogeneous concentration of serine (i.e., one with a nutrient but no gradients), then a variety of changes occur in the characteristics of their motile behavior. For instance, mean run length and mean speed increase and mean tumble time decreases. They do, however, still produce a basic type of searching behavior; even though it has some food, it persistently searches for more. As an example of tumbles and runs in the isotropic homogeneous medium described above, in one trial motility experiment lasting 29.5 sec., there were 26 runs, the maximum run length was 3.6 sec., and the mean speed was about $21\, \mu m/sec$. [19, 20].

Next, suppose that the bacterium happens to encounter a nutrient gradient (e.g., serine) as shown in Figure 11.1(c). The *change* in the concentration of the nutrient triggers a reaction such that the bacterium will spend more time swimming and less time tumbling. As long as it travels on a positive concentration gradient (i.e., so that it moves towards increasing nutrient concentrations) it will tend to lengthen the time it spends swimming (i.e., it runs farther). The directions of movement are "biased" towards increasing nutrient gradients. The cell does not change its *direction* on a run due to changes in the gradient—the tumbles basically determine the direction of the run, aside from the Brownian influences mentioned above.

On the other hand, typically if the bacterium happens to swim down a concentration gradient (or into a positive gradient of noxious substances), it will return to its baseline behavior so that essentially it tries to search for a way to climb back up the gradient (or down the noxious substance gradient). For instance, under certain conditions, for a wild-type cell swimming up serine gradients, the mean run length is 2.19 ± 3.43 sec., but if it swims down a serine gradient, mean run length is 1.40 ± 1.88 sec. [20]. Hence, when it moves up the gradient, it lengthens its runs. The mean run length for swimming down the gradient is the one that is expected, considering that the bacteria are in this particular type of medium; they act basically the same as in a homogeneous medium so that they are engaging their search/avoidance behavior to try to climb back up the gradient.

Finally, suppose that the concentration of the nutrient is constant for the region it is in, after it has been on a positive gradient for some time. In this case, after a period of time (not immediately), the bacterium will return to the same proportion of swimming and tumbling as when it was in the neutral substance so that it returns to its standard searching behavior. It is never satisfied with the amount of surrounding food; it always seeks higher concentrations. Actually, under certain experimental conditions, the cell will compare the concentration observed over the past 1 sec. with the concentration observed over the 3 sec. before that and it responds to the difference [19]. Hence, it uses the past 4 sec. of nutrient concentration data to decide how long to run [215]. Considering the deviations in direction due to Brownian movement discussed above, the bacterium basically uses as much time as it can in making decisions about climbing gradients [18]. In effect, the run length results from how much climbing it has done recently. If it has made lots of progress and hence, has just had a long run, then even if for a little while it is observing a homogeneous medium (without gradients), it will take a longer run. After a certain time period, it will recover and return to its standard behavior in a homogeneous medium.

Basically, the bacterium is trying to swim from places with low concentrations of nutrients to places with high concentrations. An opposite type of behavior is used when it encounters noxious substances. If the various concentrations move with time, then the bacteria will try to "chase" after the more favorable environments and run from harmful ones. Clearly, nutrient and noxious substance diffusion and motion will affect the motion patterns of a group of bacteria in complex ways.

11.1.3 Underlying Sensing and Decision-Making Mechanisms

The *E. coli* bacterium's sensors are receptor proteins, which are signaled directly by external substances (e.g., amino acids) or via the periplasmic substrate-binding proteins. The "sensor" is very sensitive, in some cases requiring less than 10 molecules of attractant to trigger a reaction, and attractants can trigger a swimming reaction in less than 200 ms. You can then think of the bacterium as having a "high gain" with a small attractant detection threshold (detection of only a small number of molecules can trigger a doubling or tripling of the run length). On the other hand, the corresponding threshold for encountering a homogeneous medium after being in a nutrient rich one is larger. Also, there is a type of time-averaging that is occurring in the sensing process. The receptor proteins then affect signaling molecules inside the bacterium. Also, there is in effect an "adding machine" and an ability to compare values and to arrive at an overall decision about which mode the flagella should operate in; essentially, the different sensors add and subtract their effects, and the more active or numerous have a greater influence on the final decision.

It is interesting to note that the "decision-making system" in the *E. coli* bacterium must have some ability to sense a *derivative*, and hence, it has a type of memory. At first glance it may seem possible that the bacterium senses concentrations at both ends of the cell and finds a simple difference to recognize a concentration gradient (a spatial derivative); however, this is not the case. Experiments have shown that it performs a type of sampling, and roughly speaking, it remembers the concentration a moment ago, compares it with a current one, and makes decisions based on the difference (i.e., it computes something like an Euler approximation to a time derivative).

In summary, we see that with memory, a type of addition mechanism, an ability to make comparisons, a few simple internal "control rules," and its chemical sensing and locomotion capabilities, the bacterium is able to achieve a complex type of searching and avoidance behavior. Evolution has designed this control system. It is robust and clearly very successful at meeting its goals of survival when viewed from a population perspective.

11.1.4 Elimination and Dispersal Events

It is possible that the local environment where a population of bacteria lives changes either gradually (e.g., via consumption of nutrients) or suddenly due to some other influence. There can be events such that all the bacteria in a region are killed or a group is dispersed into a new part of the environment. For example, local significant increases in heat can kill a population of bacteria that are currently in a region with a high concentration of nutrients (you can think of heat as a type of noxious influence). Or, it may be that water or some animal will move populations of bacteria from one place to another in the environment. Over long periods of time, such events have spread various types of bacteria into virtually every part of our environment.

What is the effect of elimination and dispersal events on chemotaxis? It has the effect of possibly destroying chemotactic progress, but it also has the effect of assisting in chemotaxis since dispersal may place bacteria near good food sources. From a broad perspective, elimination and dispersal is part of the *population-level motile behavior*.

11.2 *E. coli* Bacterial Foraging for Optimization

Suppose that we want to find the minimum of $J(x)$, $x \in \mathbb{R}^p$, where we do not have measurements, or an analytical description, of the gradient $\nabla J(x)$. Here, we use ideas from bacterial foraging to solve this "nongradient" optimization problem. First, suppose that x is the position of a bacterium and $J(x)$ represents the combined effects of attractants and repellents from the environment, with, for example, $J(x) < 0$, $J(x) = 0$, and $J(x) > 0$ representing that the bacterium at location x is in nutrient-rich, neutral, and noxious environments, respectively. Basically, chemotaxis is a foraging behavior that implements a type of optimization where bacteria try to climb up the nutrient concentration (find lower and lower values of $J(x)$) and avoid noxious substances and search for ways out of neutral media (avoid being at positions x where $J(x) \geq 0$).

11.2.1 An Optimization Model for *E. coli* Bacterial Foraging

To define our optimization model of *E. coli* bacterial foraging, we need to define a population (set) of bacteria, and then model how they execute chemotaxis, swarming, reproduction, and elimination/dispersal. After doing this, we will highlight the limitations (inaccuracies) in our model.

Population and Chemotaxis

Define a chemotactic step to be a tumble followed by a tumble, or a tumble followed by a run. Let j be the index for the chemotactic step. Let k be the index for the reproduction step. Let ℓ be the index of the elimination-dispersal event. Let

$$P(j,k,\ell) = \{x_i(j,k,\ell) | i = 1,2,\ldots,S\}$$

represent the positions of each member in the population of the S bacteria at the j^{th} chemotactic step, k^{th} reproduction step, and ℓ^{th} elimination-dispersal event. Here, let $J(i,j,k,\ell)$ denote the cost at the location of the i^{th} bacterium $x_i(j,k,\ell) \in \mathbb{R}^p$ (sometimes we drop the indices and refer to the i^{th} bacterium position as x_i). Note that we will interchangeably refer to J as being a "cost" (using terminology from optimization theory) and as being a nutrient surface (in reference to the biological connections). For actual bacterial populations, S can be very large (e.g., $S = 10^9$), but $p = 3$. In our computer simulations, we will use much smaller population sizes and will keep the population size fixed. We will allow $p > 3$, so we can apply the method to higher dimensional optimization problems.

Let N_c be the length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life. Let $C(i) > 0$, $i = 1,2,\ldots,S$, denote a basic chemotactic step size that we will use to define the lengths of steps during runs. To represent a tumble, a unit length random direction, say $\phi(j)$, is generated; this will be used to define the direction of movement after a tumble. In particular, we let

$$x_i(j+1,k,\ell) = x_i(j,k,\ell) + C(i)\phi(j)$$

so that $C(i)$ is the size of the step taken in the random direction specified by the tumble. If at $x_i(j+1,k,\ell)$ the cost $J(i,j+1,k,\ell)$ is better (lower) than at $x_i(j,k,\ell)$, then another step of size $C(i)$ in this same direction will be taken, and again, if that step resulted in a position with a better cost value than at the previous step, another step is taken. This swim is continued as long as it continues to reduce the cost, but only up to a maximum number of steps, N_s. This represents that the cell will tend to keep moving if it is headed in the direction of increasingly favorable environments.

Swarming Mechanisms

The above discussion was for the case where no cell-released attractants are used to signal other cells that they should swarm together. Here, we will also have cell-to-cell signaling via an attractant and will represent that with $J^i_{cc}(x, x_i(j,k,\ell))$, $i = 1,2,\ldots,S$, for the i^{th} bacterium. Let

$$d_{attract} = 0.1$$

be the depth of the attractant released by the cell (a quantification of how much attractant is released) and

$$w_{attract} = 0.2$$

be a measure of the width of the attractant signal (a quantification of the diffusion rate of the chemical). The cell also repels a nearby cell in the sense that it consumes nearby nutrients and it is not physically possible to have two cells at the same location. To model this, we let

$$h_{repellent} = d_{attract}$$

be the height of the repellent effect (magnitude of its effect) and

$$w_{repellent} = 10$$

be a measure of the width of the repellent. The values for these parameters are simply chosen to illustrate general bacterial behaviors, not to represent a particular bacterial chemical signaling scheme. The particular values of the parameters were chosen with the nutrient profile in mind, which we will use later in Figure 11.3. For instance, the depth and width of the attractant is small relative to the nutrient concentrations represented in Figure 11.3. Let

$$\begin{aligned} J_{cc}(x, P(j,k,\ell)) &= \sum_{i=1}^{S} J^i_{cc}(x, x_i(j,k,\ell)) \\ &= \sum_{i=1}^{S} \left[-d_{attract} \exp\left(-w_{attract} \sum_{m=1}^{p} (x^m - x_i^m)^2 \right) \right] \\ &\quad + \sum_{i=1}^{S} \left[h_{repellent} \exp\left(-w_{repellent} \sum_{m=1}^{p} (x^m - x_i^m)^2 \right) \right] \end{aligned}$$

denote the combined cell-to-cell attraction and repelling effects, where $x = [x^1, \dots, x^p]^\top$ is a point on the optimization domain and x_i^m is the m^{th} component of the i^{th} bacterium position x_i (for convenience, we omit some of the indices). An example for the case of $S = 2$ and the above parameter values is shown in Figure 11.2. Here, note that the two sharp peaks represent the cell locations, and as you move radially away from the cell, the function decreases and then increases (to model the fact that cells far away will tend not to be attracted, whereas cells close by will tend to try to climb down the cell-to-cell nutrient gradient towards each other and hence try to swarm). Note that as each cell moves, so does its $J_{cc}^i(x, x_i(j,k,\ell))$ function, and this represents that it will release chemicals as it moves. Due to the movements of all the cells, the $J_{cc}(x, P(j,k,\ell))$ function is *time-varying* in that, if many cells come close together, there will be a high amount of attractant and hence, an increasing likelihood that other cells will move towards the group. This produces the swarming effect. When we want to study swarming, the i^{th} bacterium, $i = 1, 2, \dots, S$, will hill-climb on

$$J(i,j,k,\ell) + J_{cc}(x,P)$$

(rather than the $J(i,j,k,\ell)$ defined above) so that the cells will try to find nutrients, avoid noxious substances, and at the same time try to move towards other cells, but not too close to them. The $J_{cc}(x,P)$ function dynamically deforms the search landscape as the cells move to represent the desire to swarm (i.e., we model mechanisms of swarming as a minimization process).

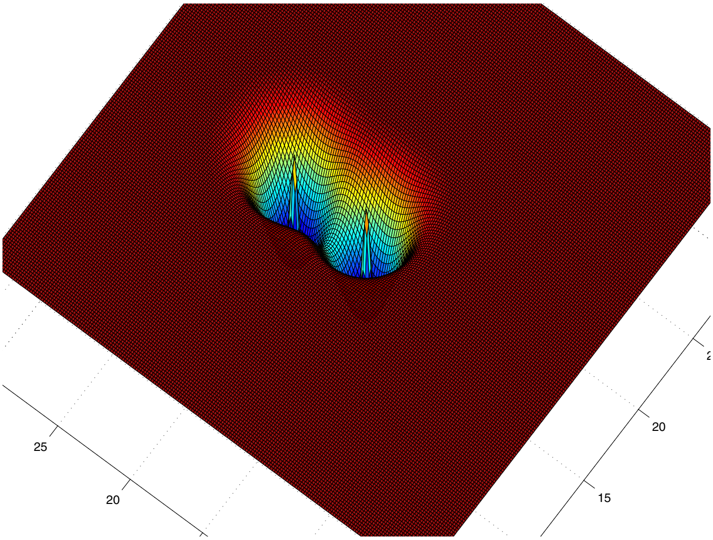


Fig. 11.2. Cell-to-cell chemical attractant model, $S = 2$.

Reproduction and Elimination/Dispersal

After N_c chemotactic steps, a reproduction step is taken. Let N_{re} be the number of reproduction steps to be taken. For convenience, we assume that S is a positive even integer. Let

$$S_r = \frac{S}{2} \tag{11.1}$$

be the number of population members who have had sufficient nutrients so that they will reproduce (split in two) with no mutations. For reproduction, the population is sorted in order of ascending accumulated cost (higher accumulated cost represents that it did not get as many nutrients during its lifetime of foraging and hence, is not as "healthy" and thus unlikely to reproduce); then the S_r least healthy bacteria die and the other S_r healthiest bacteria each split into two bacteria, which are placed at the same location. Other fractions or approaches could be used in place of Equation (11.1); this method rewards bacteria that have encountered a lot of nutrients, and allows us to keep a constant population size, which is convenient in coding the algorithm.

Let N_{ed} be the number of elimination-dispersal events, and for each such event event, each bacterium in the population is subjected to elimination-dispersal with probability p_{ed}. We assume that the frequency of chemotactic steps is greater than the frequency of reproduction steps, which is in turn greater in frequency than elimination-dispersal events (e.g., a bacterium will take many chemotactic steps before reproduction, and several generations may take place before an elimination-dispersal event).

Foraging Model Limitations

Clearly, we are ignoring many characteristics of the actual biological optimization process in favor of simplicity and capturing the gross characteristics of chemotactic hill-climbing and swarming. For instance, we assume that consumption does not affect the nutrient surface (e.g., while a bacterium is in a nutrient-rich environment, we do not increase the value of J near where it has consumed nutrients) where clearly in nature, bacteria modify the nutrient concentrations via consumption. A tumble does not result in a perfectly random new direction for movement; however, here we assume that it does. Brownian effects buffet the cell, so that after moving a small distance, it is within a pie-shaped region of its start point at the tip of the piece of pie. Basically, we assume that swims are straight, whereas in nature they are not. Tumble and run lengths are exponentially distributed random variables, not constant, as we assume. Run-length decisions are actually based on the past 4 sec. of concentrations, whereas here we assume that at each tumble, older information about nutrient concentrations is lost. Although naturally asynchronous, we force synchronicity by requiring, for instance, chemotactic steps of different bacteria to occur at the same time, all bacteria to reproduce at the same time instant, and all bacteria that are subjected to elimination and dispersal to do so at the same time. We assume a constant population size, even if there are many nutrients and generations. We assume that the cells respond to nutrients in the environment in the same way that they respond to ones released by other cells for the purpose of signaling the desire to swarm. (A more biologically accurate model of the swarming behavior of certain bacteria is given in [256].) Clearly, other choices for the criterion of which bacteria should split could be used (e.g., based only on the concentration at the end of a cell's lifetime, or on the quantity of noxious substances that were encountered). We are also ignoring conjugation and other evolutionary characteristics. For instance, we assume that $C(i)$, N_s, and N_c remain the same for each generation. In nature it seems likely that these parameters could evolve for different environments to maximize population growth rates.

11.2.2 Bacterial Foraging Optimization Algorithm (BFOA)

For initialization, you must choose p, S, N_c, N_s, N_{re}, N_{ed}, p_{ed}, and the $C(i)$, $i = 1,2,\ldots,S$. If you use swarming, you will also have to pick the parameters of the cell-to-cell attractant functions; here we will use the parameters given above. Also, initial values for the x_i, $i = 1,2,\ldots,S$, must be chosen. Choosing these to be in areas where an optimum value is likely to exist is a good choice. Alternatively, you may want to simply randomly distribute them across the domain of the optimization problem. The algorithm that models bacterial population chemotaxis, swarming, reproduction, elimination, and dispersal is given below (initially, $j = k = \ell = 0$). For the algorithm, note that updates to the x_i automatically result in updates to P. Clearly, we could have added a more sophisticated termination test than simply specifying a maximum number of iterations.

1. Elimination-dispersal loop: $\ell = \ell + 1$
2. Reproduction loop: $k = k + 1$
3. Chemotaxis loop: $j = j + 1$
   a) For $i = 1,2,\ldots,S$, take a chemotactic step for bacterium i as follows.
   b) Compute $J(i,j,k,\ell)$. Let

$$J(i,j,k,\ell) = J(i,j,k,\ell) + J_{cc}(x_i(j,k,\ell), P(j,k,\ell))$$

   (i.e., add on the cell-to-cell attractant effect to the nutrient concentration).
   c) Let $J_{last} = J(i,j,k,\ell)$ to save this value, since we may find a better cost via a run.
   d) Tumble: generate a random vector $\Delta(i) \in \mathbb{R}^p$ with each element $\Delta_m(i)$, $m = 1,2,\ldots,p$, a random number on $[-1,1]$.
   e) Move: let

$$x_i(j+1,k,\ell) = x_i(j,k,\ell) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^\top(i)\Delta(i)}}$$

   This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.
   f) Compute $J(i,j+1,k,\ell)$, and then let $J(i,j+1,k,\ell) = J(i,j+1,k,\ell) + J_{cc}(x_i(j+1,k,\ell), P(j+1,k,\ell))$.

g) Swim (note that we use an approximation, since we decide swimming behavior of each cell as if the bacteria numbered $\{1,2,\ldots,i\}$ have moved, and $\{i+1,i+2,\ldots,S\}$ have not; this is much simpler to simulate than simultaneous decisions about swimming and tumbling by all bacteria at the same time):
   i. Let $m=0$ (counter for swim length).
   ii. While $m<N_s$ (if have not climbed down too long)
      - Let $m=m+1$.
      - If $J(i,j+1,k,\ell)<J_{last}$ (if doing better), let $J_{last}=J(i,j+1,k,\ell)$ and let

$$x_i(j+1,k,\ell)=x_i(j+1,k,\ell)+C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\top}(i)\Delta(i)}}$$

        and use this $x_i(j+1,k,\ell)$ to compute the *new* $J(i,j+1,k,\ell)$ as we did in (f) above.
      - Else, let $m=N_s$. This is the end of the while statement.

h) Go to next bacterium $(i+1)$ if $i\neq S$ (i.e., go to (b) above to process the next bacterium).

4. If $j<N_c$, go to step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.
5. Reproduction:
   a) For the given k and ℓ, and for each $i=1,2,\ldots,S$, let

$$J^i_{health}=\sum_{j=1}^{N_c+1}J(i,j,k,\ell)$$

   be the health of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameters $C(i)$ in order of ascending cost J_{health} (higher cost means lower health).

   b) The S_r bacteria with the highest J_{health} values die and the other S_r bacteria with the best values split (and the copies that are made are placed at the same location as their mother).
6. If $k<N_{re}$, go to step 2. In this case, we have not reached the number of specified reproduction steps, so we start the next generation in the chemotactic loop.
7. Elimination-dispersal: for $i=1,2,\ldots,S$, with probability p_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if you eliminate a bacterium, simply disperse one to a random location on the optimization domain.
8. If $\ell<N_{ed}$, then go to step 1; otherwise end.

Matlab code for this can be obtained at:

`http://www.ece.osu.edu/~passino`

11.2.3 Guidelines for Algorithm Parameter Choices

The bacterial foraging optimization algorithm requires specification of a variety of parameters. First, you can pick the size of the population, S. Clearly, increasing the size of S can significantly increase the computational complexity of the algorithm. However, for larger values of S, if you choose to randomly distribute the initial population, it is more likely that you will start at least some bacterium near an optimum point, and over time, it is then more likely that many bacterium will be in that region, due to either chemotaxis or reproduction.

What should the values of the $C(i)$, $i = 1,2,\ldots,S$, be? You can choose a biologically motivated value; however, such values may not be the best for an engineering application. If the $C(i)$ values are too large, then if the optimum value lies in a valley with steep edges, it will tend to jump out of the valley, or it may simply miss possible local minima by swimming through them without stopping. On the other hand, if the $C(i)$ values are too small, then convergence can be slow, but if it finds a local minimum, it will typically not deviate too far from it. You should think of the $C(i)$ as a type of "step size" for the optimization algorithm.

The size of the values of the parameters that define the cell-to-cell attractant functions J^i_{cc} will define the characteristics of swarming. If the attractant width is high and very deep, the cells will have a strong tendency to swarm (they may even avoid going after nutrients and favor swarming). On the other hand, if the attractant width is small, and the depth shallow, there will be little tendency to swarm and each cell will search on its own. Social versus independent foraging is then dictated by the balance between the strengths of the cell-to-cell attractant signals and nutrient concentrations.

Next, large values for N_c result in many chemotactic steps, and, hopefully, more optimization progress, but of course, more computational complexity. If the size of N_c is chosen to be too short, the algorithm will generally rely more on luck and reproduction, and in some cases, it could more easily get trapped in a local minimum ("premature convergence"). You should think of N_s as creating a bias in the random walk (which would not occur if $N_s = 0$), with large values tending to bias the walk more in the direction of climbing down the hill.

If N_c is large enough, the value of N_{re} affects how the algorithm ignores bad regions and focuses on good ones, since bacteria in relatively nutrient-poor regions die (this models, with a fixed population size, the characteristic where bacteria will tend to reproduce at higher rates in favorable environments). If N_{re} is too small, the algorithm may converge prematurely; however, larger values of N_{re} clearly increase computational complexity.

A low value for N_{ed} dictates that the algorithm will not rely on random elimination-dispersal events to try to find favorable regions. A high value increases computational complexity but allows the bacteria to look in more regions to find good nutrient concentrations. Clearly, if p_{ed} is large, the algorithm can degrade to random exhaustive search. If, however, it is chosen appropriately, it can help the algorithm jump out of local optima and into a global optimum.

11.2.4 Relations to the Genetic Algorithm

There are *algorithmic analogies* between the genetic algorithm and the above optimization model for foraging. There are analogies between the fitness function and the nutrient concentration function (both a type of "landscape"), selection and bacterial reproduction (bacteria in the most favorable environments gain a selective advantage for reproduction), crossover and bacterial splitting (the children are at the same concentration, whereas with crossover they generally end up in a region around their parents on the fitness landscape), and mutation and elimination and dispersal. However, the algorithms are not equivalent, and neither is a special case of the other. Each has its own distinguishing features. The fitness function and nutrient concentration functions are *not* the same (one represents likelihood of survival for given phenotypic characteristics, whereas the other represents nutrient/noxious substance concentrations, or for other foragers predator/prey characteristics). Crossover represents mating and resulting differences in offspring, something we ignore in the bacterial foraging algorithm (we could, however, have made less than perfect copies of the bacteria to represent their splitting). Moreover, mutation represents gene mutation and the resulting phenotypical changes, not physical dispersal in an environment.

From one perspective, note that all the typical features of genetic algorithms could augment the bacterial foraging algorithm by representing evolutionary characteristics of a forager in their environment. From another perspective, foraging algorithms can be integrated into evolutionary algorithms and thereby model some key survival activities that occur during the lifetime of the population that is evolving (i.e., foraging success can help define fitness, mating characteristics, etc.). For the bacteria studied here, foraging happens to entail hill-climbing via a type of biased random walk, and hence, the foraging algorithm can be viewed as a method to integrate a type of approximate stochastic gradient search (where only an approximation to the gradient is used, not analytical gradient information) into evolutionary algorithms. Of course, standard gradient methods, quasi-Newton methods, etc., depend on the use of an explicit analytical representation of the gradient, something that is not needed by a foraging or genetic algorithm. Lack of dependence on analytical gradient information can be viewed as an advantage (fewer assumptions), or a disadvantage (e.g., since, if gradient information is available, then the foraging or genetic algorithm may not exploit it properly).

11.3 Example: Function Optimization via *E. coli* Foraging

As a simple illustrative example, we use the algorithm to try to find the minimum of the function in Figure 11.3 (note that the point $[15,5]^\top$ is the global minimum point). We assume that this surface can be sampled, but that the (analytical) gradient is not known.

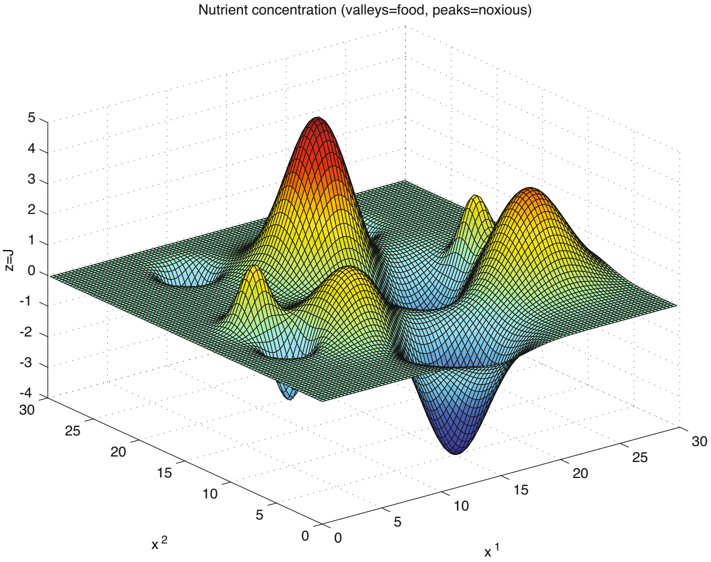


Fig. 11.3. Nutrient landscape.

11.3.1 Nutrient Hill-Climbing: No Swarming

According to the above guidelines, choose $S = 50$, $N_c = 100$, $N_s = 4$, $N_{re} = 4$, $N_{ed} = 2$, $p_{ed} = 0.25$, and the $C(i) = 0.1$, $i = 1,2,\ldots,S$. The bacteria are initially spread randomly over the optimization domain. The results of the simulation are illustrated by motion trajectories of the bacteria on the contour plot of Figure 11.3, as shown in Figure 11.4. In the first generation, starting from their random initial positions, searching is occurring in many parts of the optimization domain, and you can see the chemotactic motions of the bacteria as the black trajectories where the peaks are avoided and the valleys are pursued. Reproduction picks the 25 healthiest bacteria and copies them, and then, as shown in Figure 11.4 in generation 2, all the chemotactic steps are in five local minima. This again happens in going to generations 3 and 4, but bacteria die in some of the local minima (due essentially to our requirement that the population size stay constant), so that in generation 3, there are four groups of bacteria in four local minima, whereas in generation 4, there are two groups in two local minima.

Next, with the above choice of parameters, there is an elimination-dispersal event, and we get the *next* four generations shown in Figure 11.5. Notice that elimination and dispersal shifts the locations of several of the bacteria and thereby the algorithm explores other regions of the optimization domain. However, qualitatively we find a similar pattern to the previous four generations where chemotaxis and reproduction work together to find the global minimum; this time, however, due to the large number of bacteria that were placed near the global minimum, after one reproduction

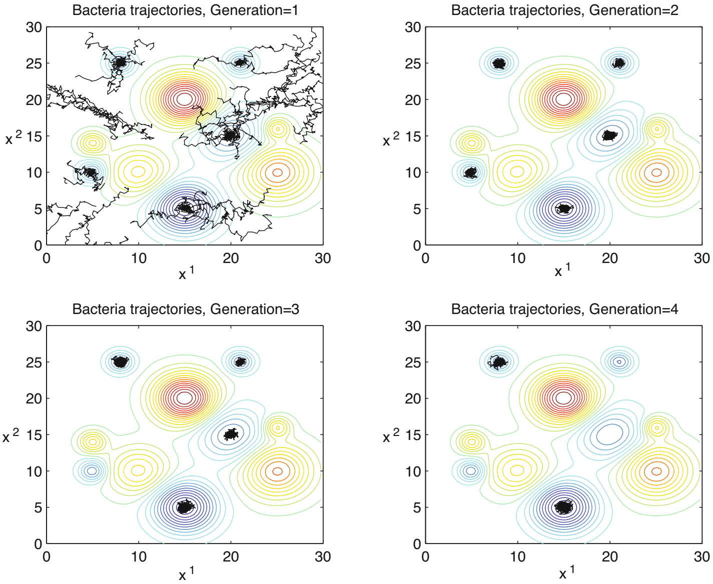


Fig. 11.4. Bacterial motion trajectories, generations 1–4, on contour plots.

step, all the bacteria are close to it (and remain this way). In this way, the bacterial population has found the global minimum.

11.3.2 Swarming Effects

Here we use the parameters defined earlier to define the cell-to-cell attraction function. Also, we choose $S = 50$, $N_c = 100$, $N_s = 4$, $N_{re} = 4$, $N_{ed} = 1$, $p_{ed} = 0.25$, and the $C(i) = 0.1$, $i = 1,2,\ldots,S$. We will first consider swarming effects on the nutrient concentration function with contour map shown on Figure 11.6 which has a zero value at $[15,15]^\top$ and decreases to successively more negative values as you move away from that point; hence, the cells should tend to swim away from the peak. We will initialize the bacterial positions by placing all the cells at the peak $[15,15]^\top$. Using these conditions, we get the result in Figure 11.6. Notice that in the first generation, the cells swim radially outward, and then in the second and third generations, swarms are formed in a concentric pattern of groups. Notice that with our simple method of simulating health of the bacteria and reproduction, some of the swarms are destroyed by the fourth generation. This simulation bears produces a qualitative behavior similar to the one seen in [31]. We omit additional simulations that show the behavior of the swarm on the surface in Figure 11.3, since qualitatively the behavior is as one would expect from the above simulations.

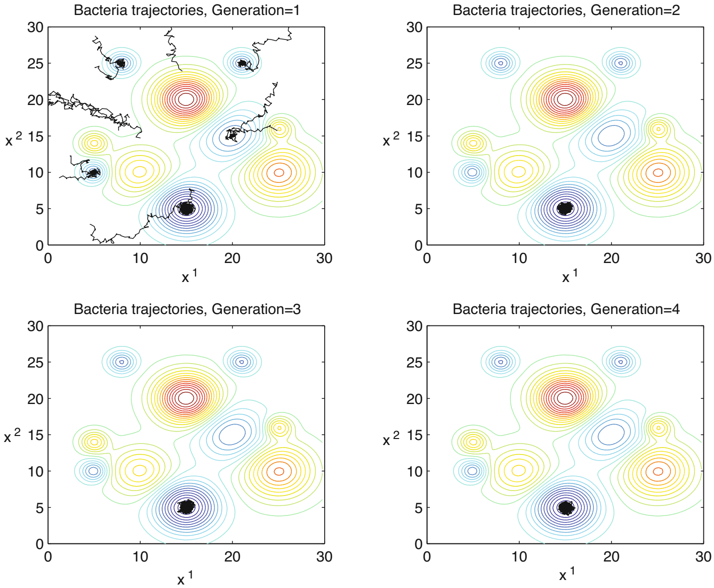


Fig. 11.5. Bacterial motion trajectories, generations 1–4, on contour plots, after an elimination-dispersal event.

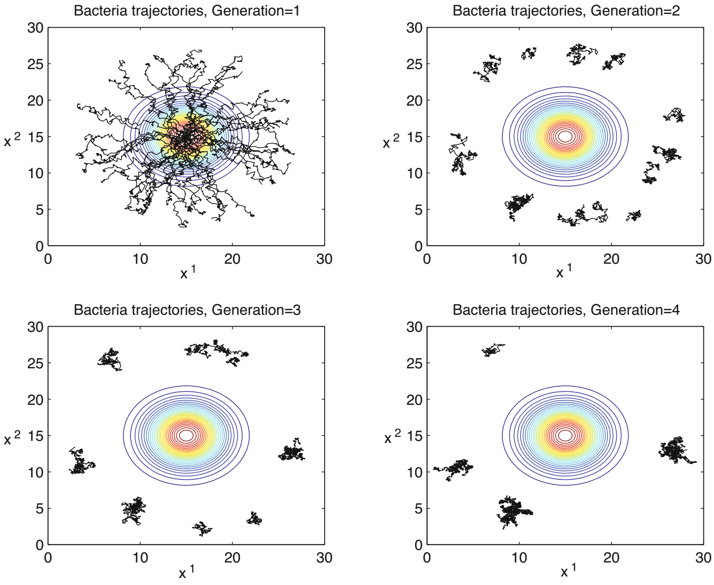


Fig. 11.6. Swarm behavior of *E. coli* on a test function.

11.4 Further Issues

11.4.1 Extensions and Generalizations

The *E. coli* foraging model we created had several limitations that were outlined in the chapter. Removal of these limitations would allow for a number of extensions and generalizations of the bacterial foraging optimization method. For instance, the modeling of Brownian effects so that the bacteria cannot swim straight could be incorporated. The tumble and run lengths could be made exponentially distributed random variables consistent with what has been found in nature. Run-length decisions could be based on the past 4 sec. of concentrations. The model and optimization algorithm could be specified to be asynchronous. A time-varying population size could be used. Other criteria by which bacteria split could be incorporated. Finally, one could add the effects of conjugation and other evolutionary characteristics (e.g., evolve $C(i)$, N_s, and N_c).

There are many species of bacteria that evidently perform some type of optimization during their motile behavior. Some optimize their position based on other chemicals, and others based on non-chemcial stimuli (e.g. light, magnetism, or heat). Each of these holds the potential for the creation of a bioinspired optimization method. In this chapter, we simply introduced one type of bacterial chemotactic behavior by one species for certain types of chemicals. We stuck to this one case as it is simple, yet representative. It could be, however, that other forms of bioinspired methods could work as well or better than the one developed here for specific types of optimization problems.

11.4.2 For Further Reading

This chapter is based on [193, 194]. An application of bacterial foraging optimization to adaptive control can be found in [193]. Similarly, a work on using bacterial foraging optimization for search and chemical concentration map building by multiple mobile robots can be found in [243]. Many other applications of the method (often called "BFO" or "BFOA") can be found via a search on the internet. Our presentation of bacterial motile behaviors was based on the work in [7, 11, 12, 18–20, 23, 30, 31, 162, 179, 215, 256]. While in this chapter we outlined the relations between bacterial foraging optimization and the genetic algorithm, relations to other non-gradient methods are outlined in [194].

12

Particle Swarm Optimization

In this chapter we consider the Particle Swarm Optimization (PSO) algorithm, which is another biologically inspired optimization algorithm. Consider again the problem in which we want to find the minimum of a function $J(x)$, $x \in \mathbb{R}^n$. Assume that measurements or an analytical expression of the gradient $\nabla J(x)$ are not available. Moreover, even if they are available, assume the function is very non-uniform or noisy so that this information is not useful. The PSO algorithm is another population based optimization algorithm which can be used to solve such problems. It is a direct search (non-gradient) algorithm where a population of particles "search" in parallel for the minimum of a given function in a multi-dimensional (n-dimensional) space (or region/domain) without using gradient information. Below we describe the basic PSO iteration. Then we discuss a modified decentralized and asynchronous version better suited for parallel and distributed implementations. Moreover, we discuss various neighborhood strategies including static and dynamic (i.e., time-varying) neighborhoods.

12.1 Basic PSO Iteration

The PSO algorithm has been inspired by the foraging behavior of species like birds in nature. In the basic iteration a population of particles, which constitute potential solutions, in search for better solutions update their positions (i.e., estimates) based on three components. At every iteration, every particle determines its velocity using

- its previous velocity,
- its best previous position, and
- the best previous position of its neighborhood.

The previous velocity constitutes the *momentum component*, the previous best position constitutes the *cognitive component*, and the best previous position of the neighborhood constitutes the *social component* of the iteration. These concepts have been inspired from the operation of the swarms in nature. There are different variations of the algorithm in the literature. The variation which is most widely used is the version

$$\begin{aligned} v_i(t+1) &= w_i(t)v_i(t) + \varphi_{i1}(t)(p_i(t) - x_i(t)) + \varphi_{i2}(t)(g_i(t) - x_i(t)) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned} \tag{12.1}$$

Here $x_i(t) \in \mathbb{R}^n$ represents the position of the i'th particle at time t (i.e., the estimate of this particle about the minimum point of the function being optimized at time t), $p_i(t) \in \mathbb{R}^n$ represents the best position obtained by the i'th particle until time t (i.e., the point with the lowest function value among all the points the particle has visited so far and refereed to as the *local best*), $g_i(t) \in \mathbb{R}^n$ represents the best position of the neighborhood of the i'th particle until time t (referred to as the *neighborhood best* or sometimes as the *global best*. Assuming that there are N particles in the swarm the index i varies from 1 to N. Note that for particle i the terms in the velocity equation (the first equation) in (12.1) constitute the momentum, cognitive, and social components, respectively. The parameters/variabiles

$$\varphi_{i1}(t) \in [0, \bar{\varphi}_1]^n \text{ and } \varphi_{i2}(t) \in [0, \bar{\varphi}_2]^n$$

which are referred to as the *learning coefficients*, are n-dimensional random vectors drawn from a uniform distribution. These random vectors determine the relative significance/weight of the cognitive and social components, respectively. The parameter $w_i(t) > 0$ is the *inertia weight* which determines the importance of the previous velocity (inertia) of the particle for the current update. It can be set as a constant or varied with time in order to improve the convergence properties of the algorithm. One can see that a swarm/particle with large inertia weight tends to perform more coarse global search, whereas a swarm/particle with small inertia weight tends to perform finer local search. One approach to vary the inertia weight is to start with a large inertia weight to favor search over larger regions at the beginning and gradually decrease the value of the weight to favor finer search close to the end.

The above PSO implementation may sometimes exhibit the so called *explosion behavior*, which is basically a scattering of the particles all over the search space (which is instability from control theoretic view point), and may not converge. In order to prevent such behavior, a bound V_{max} on the velocity $v_i(t+1)$ of the particles is imposed and the particle positions are projected within a search region $[x_{min}, x_{max}]^n \in \mathbb{R}^n$. Adaptive algorithms which vary the value of $w_i(t)$ may also be used for that purpose.

In addition to the model in (12.1) there are other possible PSO variations involving slightly different parameters. One such variation is PSO with a *constriction factor* which is proposed by Clerc and Kennedy in [40]. This version of the algorithm prevents the above mentioned scattering of particles by appropriate choice of the algorithm parameters without using the velocity bound V_{max}. In this implementation, the update equation for every particle $i = 1, \ldots, N$, can be written as

$$\begin{aligned} v_i(t+1) &= \chi\Big[v_i(t) + \varphi_{i1}(t)(p_i(t) - x_i(t)) + \varphi_{i2}(t)(g_i(t) - x_i(t))\Big] \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned} \tag{12.2}$$

Here, the constant parameter $\chi > 0$ is the constriction parameter which, provided that it is chosen appropriately, prevents the explosion behavior mentioned above without a need for bounding the velocity of the particles and projecting the states. Different choices of the parameters $\bar{\varphi}_1$ and $\bar{\varphi}_2$ (which determine the interval from which $\varphi_{i1}(t)$ and $\varphi_{i2}(t)$ are drawn) and the constriction factor $\chi > 0$ in (12.2) lead to different behavior of the swarm. One choice which leads to satisfactory performance and prevention of the explosion behavior is to select the components of the learning coefficient vectors $\varphi_{i1}(t)$ and $\varphi_{i2}(t)$ from the intervals for which the sum of the upper bounds satisfy the relation

$$\varphi = \bar{\varphi}_1 + \bar{\varphi}_2 > 4$$

and the constriction parameter $\chi > 0$ is calculated using the relation (refer to [40])

$$\chi = \begin{cases} \frac{2\kappa}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}}, & \text{if } \varphi > 4, \\ \kappa, & \text{otherwise.} \end{cases} \tag{12.3}$$

Here κ is a constant which is chosen from $\kappa \in [0,1]$. Choosing $\bar{\varphi}_1 = 2.05$ and $\bar{\varphi}_2 = 2.05$ and the coefficient $\kappa = 1$, the constriction parameter χ can be calculated as $\chi = 0.7298$.

The main concepts of this chapter can be illustrated equally well on any of the above (i.e., (12.1) and (12.2)) or other PSO variations. We will mostly use the model in (12.2) throughout the chapter and in the simulations without a particular reason for that.

12.2 Synchronous and Asynchronous PSO Implementations

12.2.1 Traditional PSO Implementations

Synchronous PSO

A large portion of the studies on PSO have been performed using synchronous operation. In such implementations all particles update their estimates $x_i(t)$ using the basic iteration formula, such as the one in (12.1) or (12.2), after which neighbors exchange information and update the value/position of the neightborhood/global best $g_i(t)$ at the same time. In this manner, all particles update their neighborhood bests "synchronously" (or concurrently) using the same current information. Such an iteration can easily be implemented on a single computer in a sequential manner. It is suitable for optimization problems which are not computationally intensive. However, in optimization problems which require intensive computations, such as problems in which the evaluation of the function $J(x)$ being optimized is computationally expensive or is based on some kind of measurement from the external environment and time consuming (compared to the extra communication burden), sequential implementation might be inefficient. To solve this problem it is possible to implement the algorithm in a parallel manner distributed over several computers. Since the algorithm is population based, in general, it is easy to distribute it over several processors. For that purpose the swarm can be divided to several groups of particles and the iteration of every group can be assigned to a different processor. Then, at the end of each iteration the processors can be synchronized to exchange their function evaluation information to determine the global/neighborhood best value and its corresponding position $g_i(t)$. In such a synchronous parallel and distributed implementation, in addition to the extra cost of synchronization, processors which finish their iterations early have to wait for the other processors. In heterogeneous systems this might be disadvantageous, although there might be some advantages rising from synchronizing the operation such as all particles having the current/same information. Therefore, given the possible disadvantages such parallel and synchronous implementations are only feasible if the benefit of parallelization outweighs the costs.

As is obvious from the above discussion a common assumption in the synchronous implementations of the PSO algorithm is that the particles have access to the current information about the achievements of their neighbors (i.e., the particles know the current best position of their neighbors) and use this information for calculating the current neighborhood best $g_i(t)$ which is used as the social component in (12.1) or (12.2). Such an assumption is very realistic when the algorithm is running on a single computer. However, as was mentioned above, enforcing it in parallel implementations requires achieving synchronization between particle updates and information exchange.

Sequential Asynchronous PSO

There are different possible asynchronous implementations of the algorithm with different levels of asynchronism. Here we will briefly mention some of them and in the next section discuss the decentralized asynchronous implementation in more detail. The most commonly used "asynchronous" PSO implementation is the sequential asynchronous implementation. It is obtained when the iteration of all particles are performed on a single computer in a sequential manner. However, as a difference from the sequential synchronous implementation, each particle updates the neighborhood/global best $g_i(t)$ just before updating its estimate (not necessarily simultaneously/concurrently with the other particles/agents). In this manner a given particle uses the updated information from the particles which have completed their iteration before it (i.e., the particles earlier in the sequence of the ongoing iteration), whereas the not-updated-yet information from the particles which are later in the sequence. Therefore, the value of the neighborhood/global best $g_i(t)$ for every particle is based on information which is a mixture of the achievements of the particles in the current and the previous iterations. If a particle is early in the sequence it uses more from the results of the previous iteration and less from the results of the current iteration. In contrast, if a particle is performing an update late in the sequence, then it uses more updated information from the current iteration and less information from the previous iteration. Note that such an implementation is not a truly asynchronous implementation in the sense that particles are still unable to perform independent iterations from each other.

Centralized Parallel Asynchronous PSO

Another asynchronous PSO implementation is the centralized parallel asynchronous implementation. In such an implementation the information of the global best value is collected and the updates are performed on a central single computer (a master computer), whereas the function evaluations are performed in parallel by the other computers (slave computers). The master computer passes the values of $x_i(t)$ and other necessary information to the slave computers and assigns them function evaluations. Each slave computer which finishes its iterations (note that the computers may finish their iterations at different time instants due to, for example, heterogeneity in the system) passes the value of the function to the master. The master updates the global best $g(t)$ and the particle estimate $x_i(t)$ for that particle using the update formula in (12.1) or (12.2) and immediately assigns new function evaluation (possibly new particle j) to that slave processor (without waiting for the other processors to finish) in order to prevent idle time and increase speed. Only one slave processor can communicate with the master at a given time. Note once more that in this set-up all iterations are performed on the master computer while only the function evaluations are performed on the slave computers. This implementation is very similar in principle to the sequential asynchronous implementation except that it increases the speed of the algorithm in problems with time consuming function evaluations. Still it has the disadvantage of being centralized and failure of the master processor (or the communication links with it) might lead to failure of the optimization process.

12.2.2 Decentralized Asynchronous PSO Formulation

In this section we discuss a decentralized asynchronous formulation of the PSO algorithm which also allows for time delays. Such a realization is much more suitable for parallel and distributed implementations of the algorithm. Since PSO is a population based algorithm it is by its nature suitable for parallel and distributed implementations unless synchronization is enforced explicitly. The current formulation is inspired from the concepts we used in the distributed agreement chapter (Chapter 9). In the version presented here the update of each of the particles can be performed on a different processor/computer without a need for synchronization or centralization. Note that the information exchange is needed (and therefore coupling between particles occurs) only for calculation of the neighborhood best $g_i(t)$. Therefore, the new formulation should allow the calculation of $g_i(t)$ to be performed totally asynchronously by each of the processors.

Given particle i let us denote with $\mathcal{N}_i(t)$ the set of its neighbors at time t. Let each particle calculate its neighborhood best $g_i(t)$ at time t by

$$g_i(t) = \operatorname{argmin}\{J_{N_i}(t), J(p_i(t)), J(g_i(t-1))\} \tag{12.4}$$

where

$$J_{N_i}(t) = \begin{cases} \min_{j \in N_i(t)}\{J(p_j(\tau_{ji}(t)))\}, & \forall t \in T_{i1}, \\ J_{N_i}(t-1), & \forall t \notin T_{i1}, \end{cases} \tag{12.5}$$

and $p_j(\cdot)$ is the best estimate of neighbor $j \in \mathcal{N}_i(t)$. Equation (12.4) basically states that particle i determines its current neighborhood best $g_i(t)$ as the minimum of the previous neighborhood best $g_i(t-1)$, its own current best $p_i(t)$, and the "perceived best" of its current neighbors $p_j(\tau_{ji}(t)), j \in \mathcal{N}_i(t)$ (which are the particles it could obtain information from). The set $T_{i1} \subseteq \{0,1,2,\ldots\}$ in (12.5) denotes the set of time instants at which particle i receives information from one or more of its neighbors (and uses this information to update its neighborhood best $g_i(t)$). At the other time instants $t \notin T_{i1}$ processor i does not obtain information from the other processors. This does not mean that processor i does not perform position (estimate) updates during these times (i.e., during $t \notin T_{i1}$). In fact, as can be seen from (12.4)-(12.5), at every time instant t the particle has an estimate of the global best $g_i(t)$ and the particle may be continuing updating its estimate $x_i(t)$. In the case that no information is obtained from the neighbors, the value of $g_i(t)$ is determined based on its old value $g_i(t-1)$ and the best $p_i(t)$ of the particle.

Similar to the case of distributed agreement in Chapter 9, the variables $\tau_{ij}(t), j \in \mathcal{N}_i(t), i = 1,\ldots,N$, in (12.5) are used to represent the time index at which a given particle i obtained information from its neighbor $j \in \mathcal{N}_i(t)$. They satisfy $0 \le \tau_{ij}(t) \le t$ for $t \in T_{i1}$, where $\tau_{ij}(t) = 0$ would mean that particle i did not obtain any information from a neighbor particle j so far (it still has the initial $p_j(0)$ and $J(p_j(0))$), whereas $\tau_{ij}(t) = t$ means that it has the current best information of particle j (i.e., it has $p_j(t)$ and $J(p_j(t))$). The difference $(t - \tau_{ij}(t)) \ge 0$ can be viewed as a communication delay in obtaining information from particle/processor j by particle/processor i. In other words, $p_j(\tau_{ji}(t))$ is the (possibly outdated) position of the best value of particle $j \in \mathcal{N}_i(t)$ currently perceived by particle i. The delay might be occurring due to, for example, congestion in the network traffic or some other reasons. This also models the implementations in which each processor updates its estimate for some time and obtains information from its neighbors once in a while (which could be a strategy with the objective to decrease the communication burden).

Note that a particle i can obtain information from only those particles which are its neighbors (i.e., it can obtain information from only $j \in \mathcal{N}_i(t)$). In our formulation we allow also for unidirectional communication implying that $j \in \mathcal{N}_i(t)$ does not necessarily mean that $i \in \mathcal{N}_j(t)$. In other words, the neighborhood definitions are not necessarily mutual or reciprocal. Moreover, even if $j \in \mathcal{N}_i(t)$ and $i \in \mathcal{N}_j(t)$ hold simultaneously, this does not imply $\tau_{ij}(t) = \tau_{ji}(t)$. In other words, even if two particles i and j are neighbors of each other at a given time instant, it does not mean that they have the current or equally outdated information about each other's best estimate implying that they do not necessarily communicate with each other simultaneously. Furthermore, we take one step further and assume that the processors may not necessarily be dedicated processors and they might be allowed to perform other jobs as well. To be able to represent such a system, we assume that a given processor i updates its estimate using (12.2) (or (12.1)) at the time instances $t \in T_{i2} \subseteq \{0,1,2,\ldots\}$ and at the other time instances $t \notin T_{i2}$ its update is "freezed"

$$\begin{aligned} v_i(t+1) &= v_i(t), \\ x_i(t+1) &= x_i(t), \end{aligned} \qquad (12.6)$$

for all $t \notin T_{i2}$. In other words, the processors update their estimates based on

$$\begin{pmatrix} v_i(t+1) \\ x_i(t+1) \end{pmatrix} = \begin{cases} \text{update using (12.2), } \forall t \in T_{i2}, \\ \text{update using (12.6), } \forall t \notin T_{i2}. \end{cases} \tag{12.7}$$

Note that this is just a more formal representation of the fact that processors might perform PSO updates at some instants, while might be busy with other jobs at other times.

Besides allowing the particles to be able to sometimes update their estimates and sometimes to freeze them, it is assumed that the processors can update their estimates at totally independent time instants as was the case with communicating/obtaining information to/from their neighbors. In other words, the sets T_{i1} and T_{i2} are assumed to be independent from each other for a given i. Moreover, they (i.e., the sets T_{i1} and T_{i2}) are independent from T_{j1} and T_{j2} for $j \neq i$ as well. However, it is possible to have $T_{i\ell_1} \cap T_{j\ell_2} \neq \emptyset$ for $i, j = 1, \ldots, N$, and $\ell_1, \ell_2 = 1, 2$ (i.e., there might be time instances at which two or more processors may update their neighborhood information and/or their estimates simultaneously). The elements of the sets $T_{i\ell}, i = 1, \ldots, N, \ell = 1, 2$ should be viewed as indices of the sequence of physical times at which the updates occur (similar to the times of events in discrete event systems), not as actual times. In other words, they are integers that can be mapped to actual times and the physical times between subsequent indices are not necessarily uniform. From this point of view the set of neighbors of a given particle i at time t (i.e., the set $\mathcal{N}_i(t)$) may represent the set of particles from which particle i obtained information between time instants/indexes $(t-1)$ and t. Note also that many of these concepts were present in the distributed agreement problem in Chapter 9.

The pseudocode of the iteration of a particle in the decentralized asynchronous PSO algorithm discussed above is shown in Table 12.1. The discussed algorithm has important differences from the PSO implementations presented in the preceding section including the standard synchronous PSO, the sequential asynchronous PSO, and centralized parallel synchronous and asynchronous PSO. First of all, it is completely decentralized and does not require existence of a central master processor to collect all the data and to perform the iterations. Note that a centralized architecture may be more prone to failures, whereas a decentralized one is more robust. Therefore, the decentralized asynchronous PSO formulation is more suitable for parallel implementations as well as implementations using concepts from the multi-agent systems field. Moreover, it may allow PSO to be implemented on a large number of computers over a large area much easily compared to its other versions.

In the next section we will discuss various static and dynamic particle neighborhood topologies that can be employed in PSO.

12.3 Particle Neighborhoods

12.3.1 Static Neighborhood

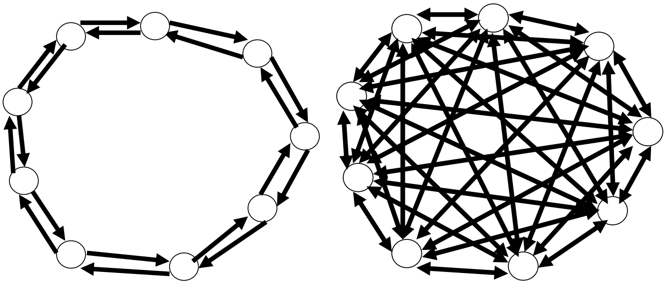
In most of the PSO implementations particle neighborhoods are fixed a priori and kept unchanged throughout the optimization process. In other words, given a

Table 12.1. Pseudocode of the Iteration of Particle i in the Decentralized Asynchronous PSO Algorithm

```
Initialize the parameters of the algorithm
Initialize randomly the particle estimate $x_i(0)$ and velocity $v_i(0)$
Calculate the initial function value $f(x_i(0))$
Initialize the particle best $p_i(0) = x_i(0)$ and $J_{pi}(0) = J(x_i(0))$
Initialize randomly the neighborhood best $g_i(0)$
Calculate the best function value $J_{gi}(0) = J(g_i(0))$
while (Stopping criteria has not reached) do
    Listen for information from neighbors
    if (Information from neighbors available (implying $t \in T_{i1}$)) then
        Update the neighborhood (global) best $g_i(t)$ using the received information
    end if
    if (Ready to do calculation (implying $t \in T_{i2}$)) then
        Calculate the new function value $J(x_i(t))$
        if ($J(x_i(t)) < J_{pi}(t-1)$) then
            Set the particle best $p_i(t) = x_i(t)$ and $J_{pi}(t) = J(x_i(t))$
            Broadcast $p_i(t)$ and $J_{pi}(t)$
        end if
        if $J(x_i(t)) < J_{gi}(t)$ then
            Set the neighborhood (global) best $g_i(t) = x_i(t)$ and $J_{gi}(t) = J(x_i(t))$
        end if
        Calculate the next estimate using (12.1) or (12.2)
    else
        Do not calculate a new estimate (see (12.6))
    end if
end while
```

particle i the time-invariant set of particles $\mathcal{N}_i$ from which it can obtain information from throughout the optimization process is predefined a priori. There are various neighborhood definition strategies. One possible strategy is to define the swarm neighborhood as a ring such that each particle has only two neighbors (one on its virtual right and one on its virtual left) and the neighborhood connections of the system form a ring as shown in Figure 12.1(a). In sequential implementations this can be realized easily so that a particle obtains information from the particles which are just before and after the given particle in the iteration sequence, i.e., particles $(i-1)$ and $(i+1)$ are set as neighbors of particle i. Another possible strategy is to define a global neighborhood in which every particle is a neighbor of every other particle in the swarm as shown in Figure 12.1(b). A third strategy could be to set the neighborhood relations randomly. These are not the only neighborhood definition strategies and there are other possible strategies as well. It is up to the user to choose the neighborhood which best suits their application. Sometimes physical constraints might be needed to be taken into account in neighborhood definitions as well.

Note that a neighborhood in which a particle is a neighbor of only a subset of the particles in the swarm (such as in the ring topology in Figure 12.1(a)) constitutes a local neighborhood in which every particle maintains its own neighborhood



(a) Ring topology. (b) Fully connected topology.

Fig. 12.1. Example neighborhood topologies.

best $g_i(t)$, whereas in the global neighborhood (i.e., the fully connected neighborhood topology in Figure 12.1(b)) there is only one global best (since there is only one overall neighborhood) and for all i we have $g_i(t) = g(t)$. It is known that, in general, PSO with the global neighborhood topology converges faster than PSO with local neighborhood topologies.[1] However, it is also known that PSO with the global neighborhood topology is more prone to premature convergence in a sense that the point to which the particles converge might not be a satisfactory point. Note also that in practice the neighborhood relations are taken as reciprocal. However, in principle, as we will see in the formulations below, this is not mandatory and they do not have to be reciprocal.

12.3.2 Dynamic Neighborhood

In some applications it might be more advantageous to have particle neighborhoods be dynamic or basically to change as the optimization process evolves. For example, one approach can be to start with local neighborhoods with low number of neighbors (such as the ring neighborhood topology) and gradually increase the number of neighbors of the particles and finish the iteration with a global neighborhood. For the ring topology the easiest way to accomplish this is to gradually increase the number of neighbors of each particle on their virtual right and virtual left. This type of dynamic neighborhood is pre-set and does not depend on external conditions or information such as the values of the function measurements $J(x_i(t))$. In some applications, particle neighborhoods might emerge as dynamic. For example, consider a PSO inspired robot search application in which at the high-level the robots act as PSO particles for path planning and way point generation in a search for a chemical source or some other substance. Given the fact that the communication ranges of the robots are limited and that they can exchange information only if they are within each other's communication range, as the robots move they will enter or

---

[1] By convergence here it is meant that all particles converge to a common point and the iteration is ended. There is not an implication that the point to which the particles have converged is a minimum point of the function being optimized $J(x)$.

exit each other's communication ranges and therefore the neighborhoods will dynamically emerge as the search process progresses. Similarly, consider the case in which the PSO algorithm is run in parallel as a batch process in a general purpose computer laboratory. Moreover, assume that the group of processors in the lab is heterogeneous meaning that not all have the same computational power. Furthermore, assume that no synchronization is enforced explicitly and a particle/processor which finishes its iteration broadcasts its information, reads the information obtained from the other processors, and continues its operation with its next iteration. In such a case, since the processors will be finishing their iterations at different time instants, the set of computers from which a given computer receives information might be changing at every iteration resulting in an emergent dynamic neighborhood.

In this section we will present three models for neighborhood representation. As one can infer from the discussion above, in real applications the dynamically changing subset of neighbors may be determined based on the physical layout or topology of the optimization/computing facility, the communication/internet traffic at the time of execution of the PSO algorithm, the distance between the particles in either the function or variable spaces or by some other means. It might depend also on the optimization problem under consideration. The neighborhood representations described can to some extent describe the above cases and have been inspired by the multi-robot search application of the algorithm as well as the parallel decentralized asynchronous PSO implementation described in the preceding section. They can also be used explicitly as dynamic neighborhood determination rules if desired.

Nearest Neighbors in the Search Space

One possible strategy to determine particle neighborhoods is to exploit the nearest neighbors rule which is based on the distance between particles in the search space. It can serve also as representation of the particle neighborhoods in the PSO inspired robot search application mentioned above or similar applications where the distance in the search space is a communication/interaction constraint. In this strategy, every particle is assumed to have a perception area and it is assumed that it can communicate only with the particles which are within that area (see Figure 12.2 where representative perception areas or neighborhood sizes for some particles are shown with circles). Assuming that the size/radius of the perception area of particle i is given by $\delta_i > 0$, a mathematical expression for representing its neighborhood at time t (given the nearest neighbors rule) is given by

$$\mathcal{N}_i(t) = \{j, j \neq i \mid \|x_i(t) - x_j(t)\| \leq \delta_i\} \tag{12.8}$$

Here $\mathcal{N}_i(t)$ denotes the set of neighbors of particle i at time t given the size of its perception area is $\delta_i > 0$. Note that this set can change with time. Moreover, note also that, in general, the perception areas of the particles can be different. If the perception areas of all particles are equal, then $\delta_i = \delta$ for some $\delta > 0$. In case of equal perception areas of particles, neighborhoods become reciprocal, meaning that if particle i is a neighbor of particle j at time t, then particle j is a neighbor of particle i at time t as well. Otherwise, the neighborhood relations may not be reciprocal leading to the fact that particle i is a neighbor of particle j at time t does not necessarily imply that particle j is also a neighbor of particle i at time t. For

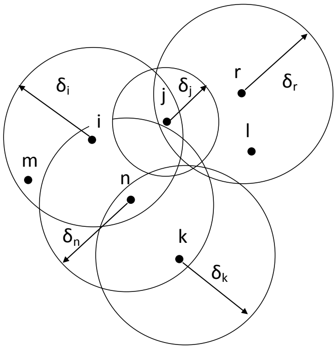


Fig. 12.2. Representation of nearest neighbors in the search space.

example, in Figure 12.2 the sizes of the perception areas of the particles are different and while particle j is a neighbor of particles i and r, the reverse is not true and particles i and r are not neighbors of particle j.

Nearest Neighbors in the Function Space

The neighborhood discussed above is based on distances in the search space. The neighborhood relations could also be determined based on the distances in function space as

$$\mathcal{N}_i(t) = \{j, j \neq i \mid \|J(x_i(t)) - J(x_j(t))\| \leq \delta_i\} \tag{12.9}$$

Here $J(\cdot)$ represents the function being optimized and the difference between function values are used to determine the neighborhood of the particles. In particular, neighbors of a particle i with neighborhood size δ_i are all particles j whose function values $J(x_j(t))$ are at most δ_i apart from its function value $J(x_i(t))$. Figure 12.3 illustrates this situation where, for example, particle j is a neighbor of particle k even though they are far from each other in the search space. In contrast, although particle n is very close to particle k in the search space it is not its neighbor. Here again δ_i can be different for different particles allowing for non-reciprocal neighborhood relations. If all the particles have the same neighborhood radius, then once more $\delta_i = \delta$ for all i and the neighborhoods become reciprocal. Note that this type of neighborhood links particles with similar performance/estimates irrespective of their distance in the search space. This strategy is similar to the elitist crossover strategies in the Genetic Algorithm (GA). In other words, it is similar to strategies in GA in which individuals with high fitness values are only allowed to crossover among each other and not with individuals with low fitness values.

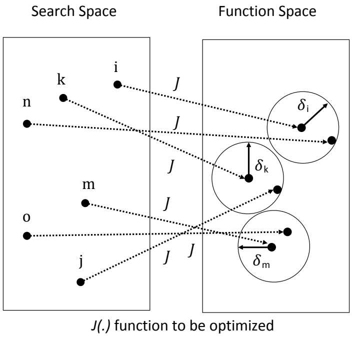


Fig. 12.3. Representation of nearest neighbors in the function space.

Random Neighborhood

Random neighborhoods can also be a model for representing dynamic neighborhoods in some applications (such as the parallel and decentralized asynchronous PSO implementation discussed above). It can be also intentionally used as a strategy for dynamic neighborhood determination. For the second case one can define a probability threshold $0 < \varepsilon < 1$ before the PSO iteration is applied and at every instance t, for every pair (i, j) generate a random number $\varepsilon_{ij} \in [0, 1]$ with uniform probability density. In this manner, and provided that ε_{ij} for pair (i, j) is generated independently from ε_{ji} for pair (j, i), at each step a total of

$$2 \times \binom{N}{2} = N(N-1)$$

uniformly distributed random numbers are generated, where N represents the number of particles in the swarm. Then the neighborhood of particle i is determined based on

$$\mathcal{N}_i(t) = \{j, j \neq i \mid \varepsilon_{ij} \leq \varepsilon\} \tag{12.10}$$

In other words, if $\varepsilon_{ij} \leq \varepsilon$ then particle j is a neighbor of particle i at time t implying that particle i can obtain information from particle j at time t. Here again, since ε_{ij} and ε_{ji} are independent, the neighborhood relations do not have to be reciprocal, i.e., given that particle i is a neighbor of particle j, the inverse of the statement–particle j is also a neighbor of particle i–may not be true. In order for the reverse to hold, (i.e., in order for particle j to be a neighbor of particle i at time t as well) the randomly generated number ε_{ji} must also satisfy the condition $\varepsilon_{ji} \leq \varepsilon$. If reciprocal neighborhood relations are desired for the random neighborhood, then one can generate

$$\binom{N}{2} = \frac{N(N-1)}{2}$$

random numbers (instead of $N(N-1)$) and assign $\varepsilon_{ji} = \varepsilon_{ij}$ for all pairs (i, j) and (j, i).

There might be other possibilities to determine the particle neighbors randomly or via other means. For example, one other option could be to have different thresholds ε_i for the particles instead of a single global threshold ε leading to different average number of neighbors for different particles. This, on the other hand, can in a sense represent "more social" and "less social" types of particles. Neighborhood determination strategies based on a combination or a hybrid of the above or other strategies may also be possible. It is up to application designer to choose the neighborhood which suits their application and optimization process best.

We would like to emphasize once more that in many applications (such as for example the decentralized, distributed, and parallel asynchronous PSO implementation) particle neighborhoods may not depend on any predefined rule but might just turn out as dynamic. The neighborhood representations presented here might serve as models for representing neighborhood relations in such applications.

Discussion on Neighborhoods

The PSO optimization problem, from some perspectives, has similarities with aggregation and more specifically with the social foraging in multi-agent systems (such as swarms in nature). Dynamic neighborhoods allow a particle to be in different groups at different times (like in swarms in nature) and adjust it's prediction (or opinion) about the position of the best value/condition of the environment. Generally, there exists no bidirectional communication in swarms in nature like flocks of birds, which inspired the PSO algorithm, and communication in such systems is unidirectional in general. Moreover, as mentioned above, unidirectional and variable neighborhood realization is more appropriate for distributed and asynchronous implementations of the PSO algorithm and for applications such as PSO-inspired search for multi-robot systems. Given the facts that most swarm robot systems work using sampled data and application of gradient based approaches in sampled data systems may be difficult, algorithms such as PSO might be a more effective strategy in such applications.

Neighborhoods based on the distance in the function space is an implementation similar to some elitist crossover strategies in the genetic algorithms. It brings, in a sense, a caste system or a hierarchy in which only particles with "similar performance" interact or exchange information with each other. It might expedite convergence in some problems but also might lead to separation of the swarm to several sub-swarms (which is not necessarily undesired) if the neighborhood parameters are not set properly. Actually such partitioning or division of the swarm is possible in all the dynamic strategies presented here, unless the conditions discussed below are satisfied. In optimization of convex functions, or convex functions with some additive noise, the neighborhood in function space may be equivalent to the neighborhood in the search space. However, for more general problems they can be also significantly different.

The random neighborhood topology brings one more randomness aspect to the PSO iteration (in addition to the steps of random length in the direction of the local best and the global best). This is in contrast to the two distance-based neighborhood determination strategies discussed, in which particle neighborhoods are determined in a deterministic manner (i.e., no randomness aspect to determine neighborhood). This might enhance the performance of the algorithm since allowing each particle at each step to exchange information with a random (and therefore different) subset of the other particles may result in probing different (and possibly linearly independent) search directions in the space which may lead to better performance in some problems.

As was already mentioned above, in some PSO implementations dynamic neighborhoods are an emergent property and do not bring any computational overhead to the algorithm. However, the above neighborhood topologies can also be intentionally used as neighborhood determination strategies. In that case, they bring extra computational burden on the algorithm. This computational burden depends on the number of particles in the swarm. In particular, for the nearest neighbors in the search space at every step an extra $N(N-1)/2$ distance calculations in $\mathbb{R}^n$ and $N(N-1)$ comparison operations, for the nearest neighbors in the function space at every step extra $N(N-1)/2$ distance calculations in $\mathbb{R}$ and $N(N-1)$ comparison operations, and for the random neighborhoods an extra $N(N-1)$ random number generations and the same number of comparisons have to be performed. This might be a disadvantage especially in time constrained applications. Therefore, one needs to take these into consideration as well in choosing the neighborhood topology.

12.3.3 Directed Graph Representation of Particle Neighborhoods

A directed graph can be used to represent all the neighborhoods (information flow topology in the swarm). Such a representation helps to visualize the interaction links in the swarm and also allows us to exploit definitions from the graph theory literature in specifying the connectivity properties of the interaction topology of the swarm. For this purpose, let us represent with a graph $\mathcal{G}(t) = (\mathcal{N}, \mathcal{A}(t))$ the neighborhood (or information flow) topology of the particle swarm at time t (see Figure 12.4). Here $\mathcal{N} = \{1,2,\ldots,N\}$ represents a constant node/particle set, $\mathcal{A}(t) \subset \mathcal{N} \times \mathcal{N}$ represents the directed arcs at time t. In other words, in this representation the i'th particle is represented as a node $i \in \mathcal{N}$, whereas the arc $(i,j) \in \mathcal{A}(t)$ represents the information flow link directed from particle i to particle j at time t. In other words, if $(i,j) \in \mathcal{A}(t)$, then it means that particle j is able to receive information from particle i at time t and particle i is a neighbor of particle j at time t meaning that $i \in \mathcal{N}_j(t)$. We would like to emphasize once more that, in general, the information flow can be unidirectional and $(i,j) \in \mathcal{A}(t)$ does not necessarily mean that $(j,i) \in \mathcal{A}(t)$ as well. For example in Figure 12.4 particle i is a neighbor of particles o, l, and m. However, the reverse is not true. In order to have bidirectional information flow, the related conditions (discussed above) for determining the neighbors must be satisfied.

If the neighborhood topology is stationary, the communication graph $\mathcal{G}(t)$ is time-independent and during all iterations $\mathcal{G}(t) = \mathcal{G}$ for some predetermined static topology/graph $\mathcal{G}$ and for all time steps $t \geq 0$. As was the case for distributed agreement in Chapter 9, the connection properties of the graph affect the information flow in the system and therefore may also affect the performance of the PSO algorithm. Recall that usually the fully connected swarm converges faster than swarms

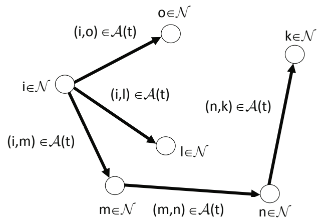


Fig. 12.4. Directed graph representation of particle neighborhoods.

with other neighborhood topologies. Moreover, consider the case of a swarm with a static neighborhood topology in which there are more than one group of particles and there is no connection (i.e., there is no communication link) between the groups. This effectively results in the fact that there are more than one sub-swarms which perform the optimization independently. This may lead to the result that the sub-swarms may converge to different points in the search space. Similar issues arise for the case of dynamic neighborhood as well. One important issue in PSO with time-varying neighborhoods is that certain connectivity conditions should be preserved in order for all particles to be able to converge to the same point. Otherwise, the swarm may act as if there are more than one sub-swarms in the system, each of which performs updates mutually independently. For instance, if $\mathcal{G}(t)$ is not strongly connected, then the particle swarm could be separated into two (or more) sub-swarms and these sub-swarms could continue to search independently and could converge to different points in the space. Therefore, in order to avoid (or at least classify) such situations, conditions on the connectivity properties of the interaction (i.e., communication) graph/topology need to be imposed.[2] Before that, we recall a few more useful concepts (definitions) form the graph theory literature.

In a graph, if there is a directed path from node i to node j, then node i is said to be connected to node j. In other words, if there exists a series of arcs from node i to node j such that $i = i_1$, $j = i_p$, and $(i_1, i_2), (i_2, i_3), \ldots, (i_{p-1}, i_p)$, then node i is connected to node j (see Figure 12.5). For example, in Figure 12.5 node i is connected to node k since there exist the links (i, o) and (o, k) (or also the links (i, l) and (l, k)). If there exists a path from every node i to every other node j, then the graph is said to be a strongly connected graph. It is important to note that it is not necessary that there exists a direct arc from every node i to every node j (as is the case in the fully connected topology, i.e., the global neighborhood) for the graph to be strongly connected and it is sufficient to have a *path* from every node i to every other node j consisting of a sequence of arcs through intermediate nodes. The graph in Figure 12.5 is strongly connected since it satisfies this condition. However, if in Figure 12.5 the link (l, i) was not present, the graph would not be strongly connected since node i would not be able to get information from any other node.

---

[2] Note here that we do not mean that separation of the swarm into sub-swarms is a bad property. In fact, in some applications it might even be useful.

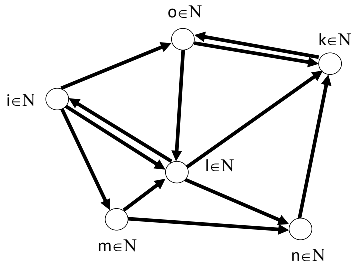


Fig. 12.5. Strongly connected graph.

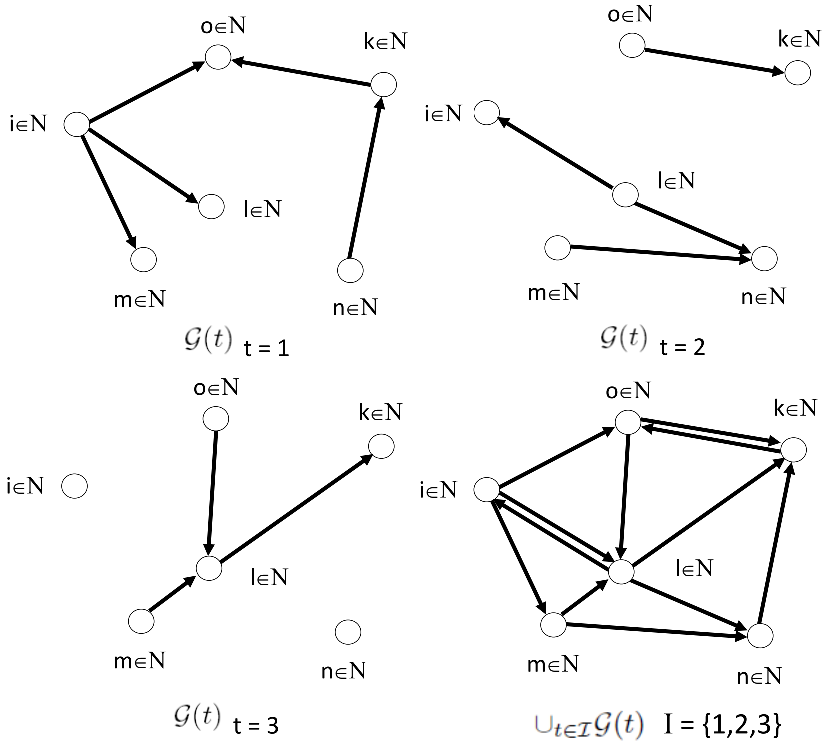
Due to the finite number of particles (number of nodes in the graph), the possible number of neighborhood (communication) graphs is finite. Given a particle swarm consisting of N particles, let the set $\bar{\mathcal{G}} = \{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ represent all the possible neighborhood topologies in the particle swarm. In this case any valid interconnection graph $\mathcal{G}(t)$ (whether static or dynamic) is within $\bar{\mathcal{G}}$, i.e., $\mathcal{G}(t) \in \bar{\mathcal{G}}$ for every t. In the static topology case the properties of the graph $\mathcal{G}$ are important for the performance and behavior of the PSO algorithm. In the dynamic topology case, on the other hand, the properties of the set of the sequence of communication/neighborhood graphs, $\{\mathcal{G}(t)\}$ is more significant than the properties of the instantaneous communication/neighborhood graphs $\mathcal{G}(t)$. In particular, the properties of the union of the sequence of graphs $\{\mathcal{G}(t)\}$ plays a significant role for the connectivity and information flow in the swarm. The union of graphs which have the same node set $\{\mathcal{G}_i = (\mathcal{N}, \mathcal{A}_i)\} \subset \bar{\mathcal{G}}$ is defined as $\cup \mathcal{G}_i = (\mathcal{N}, \cup \mathcal{A}_i)$. For a time interval $\mathfrak{I}$, if the union $\cup_{t \in \mathfrak{I}} \mathcal{G}(t)$ is strongly connected, then the sequence of graphs $\{\mathcal{G}(t)\}$ is said to be strongly connected over interval $\mathfrak{I}$. Notice that in this case, over interval $\mathfrak{I}$, all nodes are connected to each other and information from any node/particle could flow through the swarm of particles.

12.3.4 Connectivity of the Swarm

As was mentioned earlier, the change of particle neighbors with respect to time may result in separation of the swarm into a number of sub-swarms. The assumptions below provides sufficient conditions in order to ensure continuation of information flow throughout the entire swarm.

Assumption 20. *There exist a constant* $\mathrm{I} > 0$ *such that for every interval* $\mathfrak{I}$ *of length* I *the sequence of interaction/communication graphs* $\{\mathcal{G}(t) = (\mathcal{N}, \mathcal{A}(t))\}$ *is strongly connected.*

The assumption above could be referred to as the uniformity of connections in the communication graph. In other words, it basically states that the sequence of communication/interaction graphs is jointly uniformly strongly connected. This is explained with a simple representative schematic in Figure 12.6, where although the



$\mathcal{G}(t)$ t = 1

$\mathcal{G}(t)$ t = 2

$\mathcal{G}(t)$ t = 3

$\cup_{t\in\mathcal{I}}\mathcal{G}(t)$ I = {1,2,3}

Fig. 12.6. Directed graph representation of Assumption 20.

individual graphs at times $t = 1$, $t = 2$, and $t = 3$ are not strongly connected, the union of the graphs over the interval $\mathfrak{I} = \{1,2,3\}$ is strongly connected.

If the communication is bidirectional, the strong connectivity is equivalent to the spanning tree assumption which was needed in the distributed agreement case in Chapter 9. Strong connectivity is required in order to ensure that every particle eventually has access to the experience of every other particle and in this manner the swarm is able to improve its estimates. Note that even though here the swarm converges to a common point as well, the problem is completely different from the distributed problem in Chapter 9. This is because the objective here is not only to reach agreement at a point, but this point also has to be a minimum of the function $J(x)$ being optimized. For that reason bidirectional flow of information is needed for improved performance while spanning tree assumption (instead of strong connectivity) could be sufficient for achieving convergence of all particles to a single point, it can undermine the broader objective to minimize $J(x)$.

There are various parameters and properties that affect the performance of the three neighborhood determination strategies discussed in the preceding section, which also affect whether Assumption 20 is satisfied or not for a specific neighborhood determination strategy. For instance, the performance of the particle swarm exchanging information based on nearest neighbors in the search space in addition to the sizes of the perception areas depends on the number of particles performing search and the size of the search space, i.e., density of particles in the search space. If the density of particles in the search space is low and the sizes of the perception areas of the particles are small, the possibility of existence of strong connectedness in the particle swarm is small. As the particle density or the size of the perception areas increase, the number of particles that can exchange information among each other increases which also increases the possibility of existence of strong connectedness in the particle swarm and therefore satisfaction of Assumption 20. On the other hand, the performance of the swarm operating with neighborhood determination based on nearest neighbors in function space is also affected by the function being optimized in addition to the number of particles and the size of the perception ranges. The difference between the minimum and maximum values of the function being minimized determines the effective size of the space and the density of particles. In other words, for this case the density of the particles in the function space together with the sizes of the perception areas are the primary parameters for the connectedness of the communication graph. Since initially particles might be scattered, and as time progresses they get close to each other, for the cases in which distance based neighborhood is enforced artificially the perception areas might be set large at the beginning and slowly decreased as time progresses in order to ensure connectivity.

The performance of the swarm with a random neighborhood solely depends on the number of particles in the swarm and the probability threshold ε for two particles being neighbors. These two parameters are significant to satisfy Assumption 20. Recall that for this case particle j becomes a neighbor of a given particle i with some probability and the threshold ε represents the probability of being neighbors. Therefore, a larger value of ε or larger number of particles in the swarm will on average result in higher number of neighborhood relations/connections and a higher possibility of existence of strong connectedness in particle swarm leading to satisfaction of Assumption 20.

For the stationary comunication/neighborhood topologies, Assumption 20 can be stated as below in order to preserve the global connectivity property of the swarm.

Assumption 21. *The static interaction graph* $\{\mathcal{G} = (\mathcal{N}, \mathcal{A})\}$ *is strongly connected.*

Satisfaction of Assumption 21 ensures connectivity in the swarm. If there is no strong connectivity in the neighborhood topology of the swarm with fixed neighborhood, then it means that at least one particle is not connected to the other particles in the swarm and this particle cannot obtain information directly or indirectly from the (at least part of) the other particles. This fact may prevent the convergence of all particles to a common point. Even though it is typically not stated explicitly in the works on PSO in the literature, the particle neighborhoods are generally chosen such that Assumption 21 is satisfied. Assumption 20 and Assumption 21 provide sufficient conditions for continuation of information flow between particles in the swarm.

Note here that it is not claimed that separation of the swarm to several sub-swarms is always a disadvantageous property. There might be cases in which it is advantageous since it may allow the sub-swarms to search/explore concurrently different regions of the space thus identifying more than one potential minima/solution. This, on the other hand, may allow the user to freely choose between these minima/solutions based on other conditions/considerations which may not be directly coded within the function to be minimized and therefore into the PSO iteration. Given these perspectives, the above assumptions provide sufficient conditions which need to be satisfied in case it is desired (for one reason or another) that the swarm not to separate into sub-swarms.

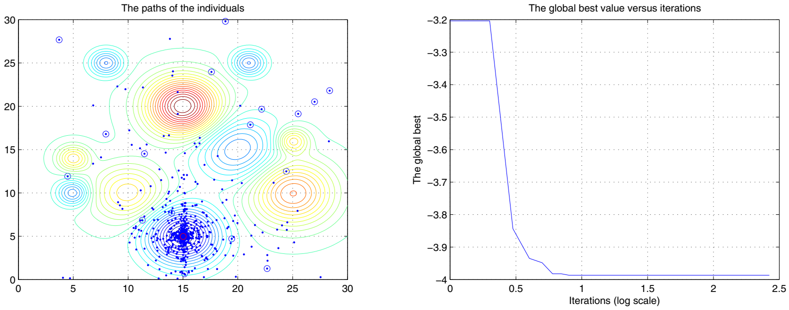
The particles (the agents) in the decentralized asynchronous PSO algorithm with dynamic neighborhood and time delays are fully autonomous and independent and they can join or exit neighborhood groups or even the optimization process without destructing the overall iteration. Therefore, the number of agents does not have to be fixed a priori in that version of the algorithm. This shows that this version of the algorithm is easily scalable. Moreover, as was mentioned earlier, particle/agent and/or communication failures do not necessarily result in the failure of the optimization process since the active (unfailed) agents/particles can still continue their iterations and complete the process. Besides these advantages, it is suitable for applications in which function evaluations (i.e., the determination of the function values $J(x_i(t))$) are obtained by some kind of measurement from the external environment instead of just pure computation.

An important issue in PSO, as is in other optimization algorithms, is how to determine the stopping criteria for the optimization process. In other words, the question is how do we know when to stop since a “good enough” solution has been reached. There might be various criteria which can be used for that purpose. One such criterion could be to have fixed number of iterations. In other words, the particles/agents/processors are given a fixed number of iterations a priori and each stops after finishing these iterations. Another criterion could be that agents stop if they are unable to achieve sufficient decrease/increase for a certain period of time or number of iterations. A third criterion could be the distance between the particles and the iterations are stopped when the swarm has almost collapsed to a single point. In the decentralized implementation, stopping criteria based on some kind of negotiation between the particles can also be considered in which a particle/processor decides to stop and passes such a request to the other particles/processors (its neighbors) and they negotiate about stopping or not. A combination of the above or other stopping criteria are also possible. Moreover, in the decentralized asynchronous PSO particle-specific stopping criteria are also possible, i.e., different particles can have different stopping criteria. For example, one particle might be set to stop after 500 iterations, while another particle might be required to perform 1000 iterations.

From the fact that in PSO the dynamics/iterations of the particles are relatively independent, it is also easy to set different parameter values (of, for example, the constriction factor χ and/or the learning coefficients φ_1^i and φ_2^i) for different particles. In this manner, it might be possible to obtain particles with different search tendencies/characteristics such as particles performing more coarse search (e.g., particles having large $\bar{\varphi}_1$ and $\bar{\varphi}_2$ values) and those performing finer search (e.g., particles having small $\bar{\varphi}_1$ and $\bar{\varphi}_2$ values). Such a variation might be even easier in its decentralized asynchronous and distributed implementations.

12.4 Simulation Examples

In this section we test the performance of the PSO algorithm in several optimization problems. The first function we consider is the function considered for social foraging in earlier chapters as well as in Chapter 11 for bacterial foraging optimization. Note that for this function the dimension of the state space is $n = 2$. For this case, the number of particles in the swarm we used is $N = 20$. The performance of a synchronous implementation of PSO with constriction factor in (12.2) with fully connected static neighborhood topology for randomly generated initial particle positions and zero initial velocities can be seen in Figure 12.7. Figure 12.7(a) shows the paths of the particles over the contour plot of the function, whereas Figure 12.7(b) shows the value of the global best with respect to the iterations. For this simulation, the bounds on the learning coefficients $\bar{\varphi}_1$ and $\bar{\varphi}_2$ are selected as 2.05 (i.e., the φ^i_{1j} and φ^i_{2j} are uniformly distributed random variables satisfying $\varphi^i_{1j} \in [0, 2.05]$ and $\varphi^i_{2j} \in [0, 2.05]$) and as a constriction factor $\chi = 0.7298$ was used. The minimum of the function is found to be $x^* = [15.0162, 4.9837]$ and has the value $J(x^*) = -3.9867$. As can be seen, PSO was able to locate it in less than 300 iterations (note that the horizontal axis in Figure 12.7(b) is in log scale). Recall that for the global neighborhood there is one single global minimum in contrast to the neighborhood/local minimums for the other topology cases. As can be seen from the figure and as expected it is constantly decreasing throughout the iterations. The stopping criteria for this case was set as



(a) Paths of particles.

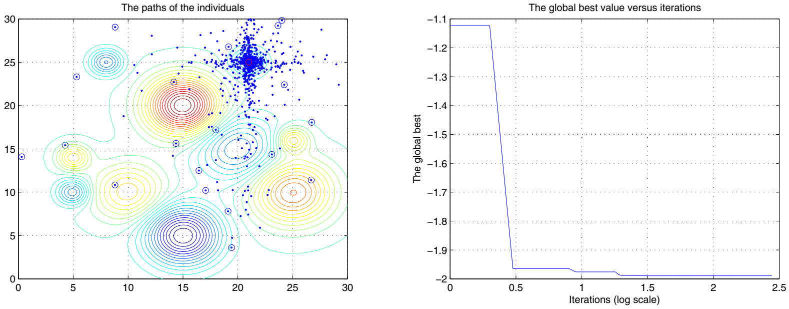
(b) Global best value.

Fig. 12.7. Synchronous PSO with fully connected neighborhood ($N = 20$ particles).

$$e(t) = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \|x_i(t) - x_j(t)\| \leq \eta \qquad (12.11)$$

with $\eta = 10^{-8}$. In other words, when the average distance between particles in the swarm becomes less than $\eta = 10^{-8}$ the optimization process is terminated since the swarm has collapsed to almost a single point. Note that it is possible to use alternative stopping criteria as well.

Another simulation with the same set of parameters but for different initial conditions can be seen in Figure 12.8. As can be seen from the figure, this time the swarm converges to a local minimum of the function because of the bias of the initial positions of the particles. In order to avoid such performance it is better to have a higher number of particles. Moreover, running the algorithm several times and choosing the best found value might be useful as well. Also, trying different neighborhood topologies might also enhance performance. As was mentioned before, PSO with full neighborhood converges fast but is more prone to converge to a local minimum. Our experience is that PSO with a random dynamic neighborhood might do better in problems such as the one considered here.



(a) Paths of particles.

(b) Global best value.

Fig. 12.8. Synchronous PSO with fully connected neighborhood ($N = 20$ particles) converges to a local minimum.

As another set of simulations we consider a set of benchmark functions commonly used in the PSO literature. The names and expressions of the functions considered are given in Table 12.2. Note that the global minimum for the DeJongF4, Griewank, and Rastrigin functions is located at the origin ($x^* = [0,...,0]^\top$), whereas for the Rosenbrock function it is located at $x^* = [1,...,1]^\top$.

Table 12.2. Benchmark Functions

Function	Expression
DeJongF4	$\sum_{i=1}^{n} ix_i^4$
Griewank	$\sum_{i=1}^{n}\left(\frac{x_i^2}{4000}\right)-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
Rastrigin	$\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right)$
Rosenbrock	$\sum_{i=1}^{n-1}\left((1-x_i)^2+100(x_{i+1}-x_i^2)^2\right)$

For these simulations we use the decentralized asynchronous PSO implementation discussed in this chapter with a dynamic neighborhood. The neighborhood determination strategy used is the random neighborhood strategy in (12.10) with neighborhood probability threshold $\varepsilon = 0.02$. The algorithm was run on a single computer in a sequential manner and the asynchronism in information exchange and estimate updates were introduced artificially using probabilistic methods. The pseudo-code of the algorithm used is shown in Table 12.3 and described in more detail below.

Table 12.3. Pseudocode of the PSO Algorithm Used

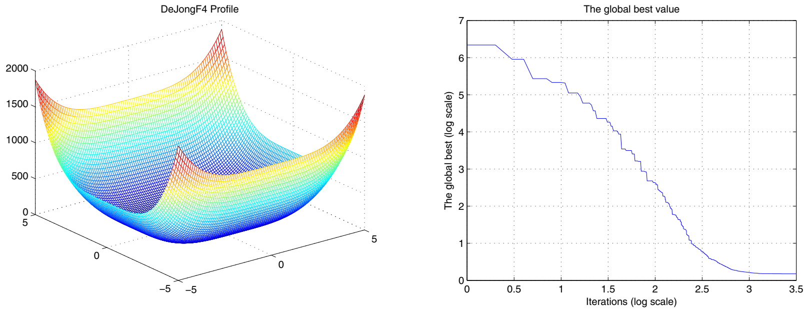
```
Initialize the parameters of the algorithm
Initialize randomly initial particle estimates $x_i(0)$ and zero velocities $v_i(0)$
Calculate the initial function values $J(x_i(0))$
Initialize the particle bests $p_i(0) = x_i(0)$ and $J_{pi}(0) = J(x_i(0))$
Initialize the neighborhood bests $g_i(0) = x_i(0)$ and $J_{gi}(0) = J(x_i(0))$
Initialize $\bar{p}_{sense}$, $\bar{p}_{move}$ and $\varepsilon$
while (Stopping criteria is not met) do
   for each agent $i$ do
      Determine the set of its neighbors $\mathcal{N}_i(t)$ using (12.10)
      for each particle $j \in \mathcal{N}_i(t)$ do
         Generate $p^{ij}_{sense}(t)$
         if $p^{ij}_{sense}(t) > \bar{p}_{sense}$ then
            Obtain the current best information of neighbor $j \in \mathcal{N}_i(t)$
         else
            Use the old saved information of neighbor $j \in \mathcal{N}_i(t)$
         end if
         Update the neighborhood (global) best $g_i(t)$ using $p_j(\tau_{ij}(t))$
      end for
      Generate $p^i_{move}(t)$
      if $p^i_{move}(t) > \bar{p}_{move}$ then
         Calculate the new function value $J(x_i(t))$
         if $(J(x_i(t)) < J_{pi}(t-1))$ then
            Set the particle best $p_i(t) = x_i(t)$ and $J_{pi}(t) = J(x_i(t))$
         end if
         if $J(x_i(t)) < J_{gi}(t)$ then
            Set the neighborhood (global) best $g_i(t) = x_i(t)$ and $J_{gi}(t) = J(x_i(t))$
         end if
         Calculate particle next estimate using (12.2)
      else
         Do not calculate a new estimate and velocity (see (12.6))
      end if
   end for
end while
```

As can be seen from Table 12.3, in order to achieve asynchronism at each time instant, particles are set to sense their neighbors state and to update their own states using the iteration in (12.2) with some probability. For this purpose, two thresholds $0 < \bar{p}_{sense} < 1$ and $0 < \bar{p}_{move} < 1$ are defined. At each time step t, particle i generates a total of $(N_i(t) + 1)$ uniformly distributed random numbers in the interval $[0, 1]$, namely p^i_{move} and $p^{ij}_{sense}(t), j = 1, \ldots, N_i(t)$. Here $N_i(t) = |\mathcal{N}_i(t)|$ is the number of neighbors of particle i, or basically the number of elements of the set $\mathcal{N}_i(t)$. The numbers $p^{ij}_{sense}(t), j = 1,\ldots,N_i(t)$ determine whether particle i can obtain information from a neighbor $j \in \mathcal{N}_i(t)$ or not. If $p^{ij}_{sense}(t) > \bar{p}_{sense}$, then particle i is able obtain the current information from its neighbor particle j and use the information acquired from particle j for determining the global/neighborhood best value. Otherwise, it uses old information from agent $j \in \mathcal{N}_i(t)$. This introduces, in a sense, a delay in the information flow. Similarly, if $p^{i}_{move}(t) > \bar{p}_{move}$, then particle i updates its state estimate using (12.2), implying that the particle is at its "awake" state. Otherwise, it keeps its current estimate using (12.6), implying that the particle is at its "asleep" state. We would like to point out that this kind of implementation is not a real decentralized asynchronous implementation of the algorithm but it mimics such properties.

For the simulations using the functions in Table 12.2 the dimension of the state space is chosen as $n = 20$ (i.e., the search space is a subset of $\mathbb{R}^{20}$) for all the functions considered. Moreover, we used a swarm consisting of $N = 100$ particles. The initial particle positions were generated randomly and the initial velocities of the particles were set to zero. The ranges for the learning coefficients and the value of the constriction factor were the same as in the previous simulation and were basically satisfying $\varphi^i_{1j} \in [0, 2.05]$, $\varphi^i_{2j} \in [0, 2.05]$, and $\chi = 0.7298$. As a difference from the previous case the stopping criteria was set as predefined constant number of iterations $N_{iterations} = 3162 \simeq 10^{3.5}$.

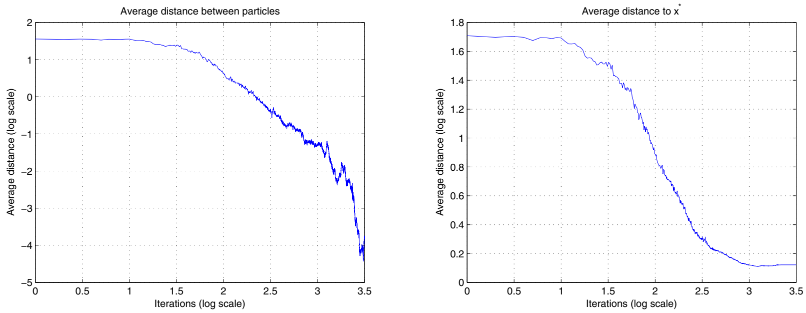
Figure 12.9 shows the results for the DeJongF4 function. The function itself for $n = 2$ is shown in Figure 12.9(a). As mentioned above, its global minimum is located at the origin. However, the region around the minimum is relatively flat, which might create algorithmic difficulties in some cases. For this simulation the initial particle positions were generated from within $[-20, 20]^n$. Figure 12.9(b) shows the plot of the global best value versus the iterations. The value plotted in the figure is minimum among the local/neighborhood bests $g(t) = \min\{g_i(t)\}$. Note that both the vertical and horizontal axes are in logarithmic scale. As can be seen from the figure, the value of the global best is constantly decreasing and at the end of the iteration it is equal to $g = \min\{g_i\} = 1.5113$. The plot in Figure 12.9(c) shows the evolution of the average distance between particles or basically $e(t)$ in (12.11) with respect to the iteration number. As can be seen from the figure, at the end of the iterations its value is in the order of 10^{-4}. This basically means that the swarm has sufficiently collapsed although there might be a little more room for further improvement. Figure 12.9(d) shows the average distance of the particles to the global best position (the origin) again in logarithmic scales. As one can see, the distance to the origin is less than 1.5.

Figure 12.10 shows the results for the Griewank function. Figure 12.10(a) shows the shape of the function in a two dimensional space. It is a function with multiple hills and valleys which might be misleading for the optimization process. The initial particle positions were generated from within $[-20, 20]^n$ for this simulation as well. Figure 12.10(b) depicts the value of the global best versus the iteration number. It can be seen that after $N_{iterations} = 3162$ iterations the global best value obtained is 0.7455 which is close to the global minimum located at the origin and equal to zero. It is easy to see from Figure 12.10(c) that the particles have again converged



(a) DeJongF4 function for $n = 2$.

(b) The best function value.



(c) Average distance between particles.

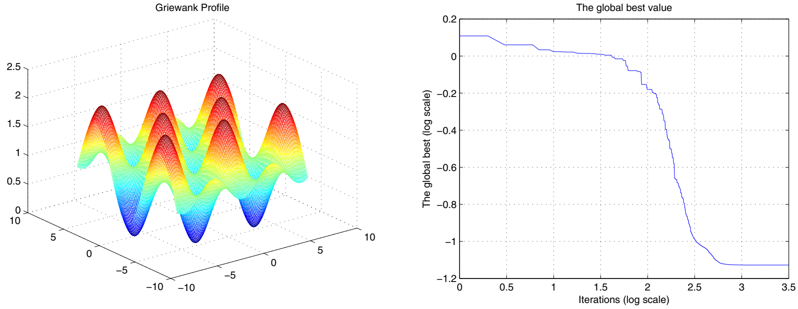
(d) Average distance to the minimum.

Fig. 12.9. Simulations for the DeJongF4 benchmark function.

very close to each other and the average distance between them is of order 10^{-3}. Similarly, the average distance of the particles to the global minimum is less than 10 as can be seen from Figure 12.10(d).

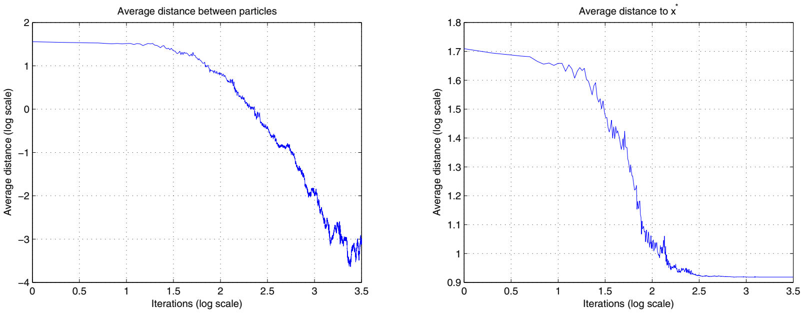
The results for the Rastrigin function are shown in Figure 12.11. As can be seen from Figure 12.11(a) the Rastrigin function is somehow similar to the Griewank function but is more hilly and more noisy which makes it a difficult function to minimize. For this function initial particle positions were initialized within the region $[-6,6]^n$. The decentralized asynchronous PSO with dynamic neighborhood shows similar performance for this function as well. As can be seen from Figure 12.11(c), at the end of the $N_{iterations} = 3162$ iterations the average distance between particles is again in the order of 10^{-4}. The average distance to the minimum is around 3.8 and the lowest function value achieved is $g = 22.4382$. Note that given the number of dimensions in the search space, the initial search range, and the difficulty of the function this is not unimpressive performance.

The results for the Rosenbrock function are shown in Figure 12.12. Note that different from the previous cases its global minimum is located at $x^* = [1,...,1]^\top$. Its shape in $n = 2$ dimensions can be seen in Figure 12.12(a). It seems a fairly smooth function. However, it is also difficult to minimize. For this function initial particle positions were initialized within the region $[-4,4]^n$. One can see from



(a) Griewank function for $n = 2$.

(b) The best function value.



(c) Average distance between particles.

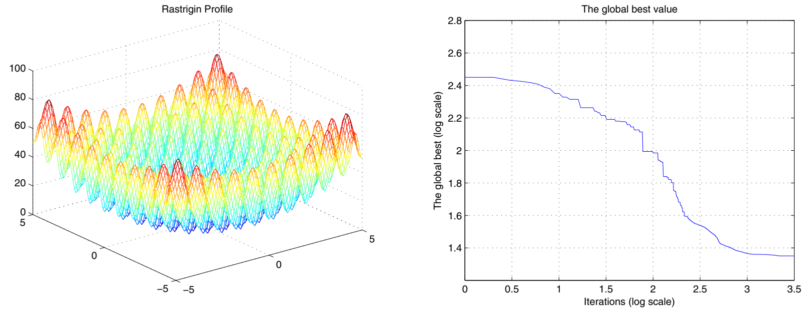
(d) Average distance to the minimum.

Fig. 12.10. Simulations for the Griewank benchmark function.

Figure 12.12(c) that after $N_{iterations} = 3162$ iterations the inter-particle distances have reached again the order of 10^{-4}. The average distance to the minimum is around 1.18 and the best function value reached is $g = 19.5670$. This might be seen unsatisfactory. However, note that for a higher number of particles, and provided that the swarm was allowed to run for longer period of time, it would probably reach a much better function value. Also, since there is some stochasticity in the algorithm the performance may change from run to run.

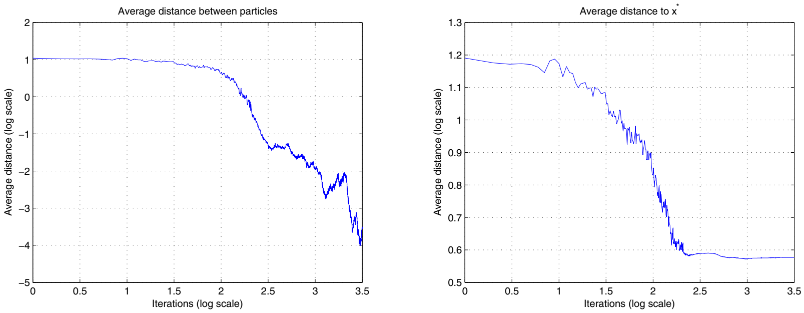
12.5 Further Issues

The problem of PSO optimization has some similarities with the problem of consensus seeking or distributed agreement in multi-agent systems and the dynamic neighborhood allows a particle to be part of different neighborhoods at different time instants and to adjust its estimate (its opinion) based on the information in these neighborhoods (just as in real biological systems in which there are no fixed neighborhoods). In natural swarms like flocks of birds or human social networks (on which the original PSO was inspired) communication (which does not necessarily mean direct communication) or sensing is rarely bidirectional and this is also present



(a) Rastrigin function for $n = 2$.

(b) The best function value.

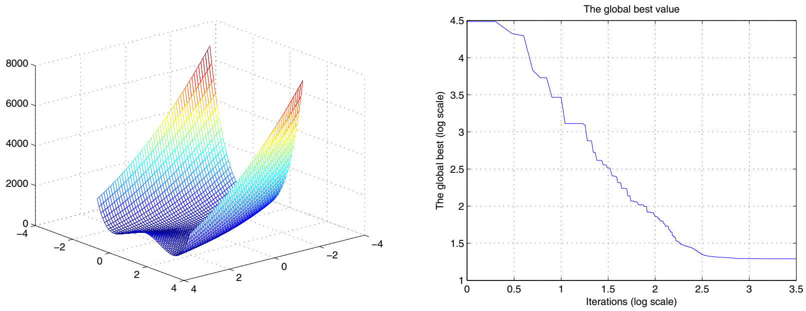


(c) Average distance between particles.

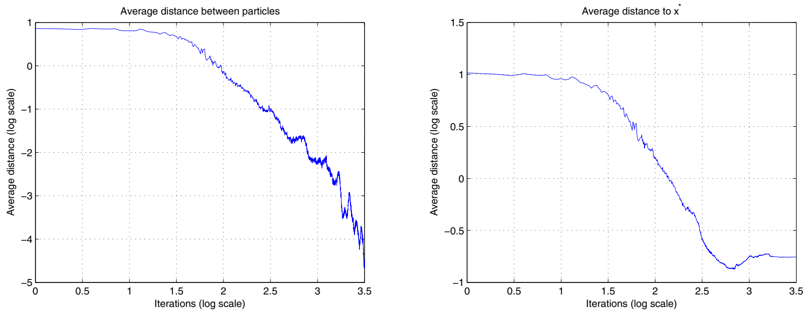
(d) Average distance to the minimum.

Fig. 12.11. Simulations for the Rastrigin benchmark function.

in the model/formulation considered in this chapter. Moreover, in natural swarms the information sensing and motion by the agents are usually done in a decentralized and asynchronous manner and the formulation in this chapter reflects that property as well. Furthermore, the model allows for uncertainties due to possible delays in the information flow or other reasons. All these allow for a distributed agent-based implementation of the algorithm in which each particle is programmed as a different software agent which runs on a possibly different processor/computer (similar to the case in real biological swarms) and occasionally communicates with some of the other agents in its environment/neighborhood. There is no need for a global clock for synchronization and the delays in the model account for communication/network delays which could be present in the system. The agents may fail, join or leave the group without leading to catastrophical failure of the overall operation. For these reasons we believe that decentralized and distributed asynchronous PSO formulation with time delays presented in this chapter has better philosophical connections with the operation of natural distributed multi-agent systems (such as schools of fish, flocks of birds, swarms of honeybees, social networks, etc.). Moreover, it allows for much more flexible implementation of the PSO algorithm. Furthermore, its current formulation has the prospect of creating new potential application areas of the algorithm. The PSO inspired search by multiple robots in a sampled environment



(a) Rosenbrock function for $n = 2$. (b) The best function value.



(c) Average distance between particles. (d) Average distance to the minimum.

Fig. 12.12. Simulations for the Rosenbrock benchmark function.

example (which was mentioned several times throughout the chapter) is one such application. Since implementing gradient based methods might be difficult in a sampled environment, algorithms like PSO might be more efficient in such applications. Synchronous operation is difficult to implement in such artificial multi-agent systems since it requires a global clock to which all the agents must be subjected to, therefore undermining the distributed (decentralized) nature of the problem.

The PSO algorithm has become a very popular algorithm in recent years. It is very easy to implement and seems fairly affective given the cost of implementation. These seem to be the main reasons behind its popularity. The performance of the PSO algorithm has been studied from many different perspectives in the literature and it has been applied on many optimization problems. Below, we highlight some areas of possible further study on the algorithm as well as briefly mention some of the studies that have already been done in the literature.

12.5.1 Extensions and Generalizations

As we mentioned above, the performance of the PSO algorithm with respect to the values of the algorithm parameters such as the learning, constriction, inertia coefficients, neighborhood structure and others have been investigated in the literature.

There are also studies investigating the convergence of the swarm. These studies look at the problem from the perspective of a single particle and investigate whether the particle will eventually stop or not. Note that this is not really a convergence analysis from the perspective of the optimization function. In function minimization the main objective is to reach a point which is a (possibly local) minimum of the function. Usually an algorithm which guarantees that a minimum will be reached is considered as good or trusted algorithm. Therefore, there is a need to rigorously prove (or disprove) that PSO guarantees convergence to a critical (minimum) point. In other words, given a function $J : \mathbb{R}^n \to \mathbb{R}$, the question is to find out whether PSO guarantees always to find a x^* such that

$$J(x^*) \leq J(x)$$

for all $x \in \mathcal{B}(x^*) \subset \mathbb{R}^n$, in some neighborhood $\mathcal{B}(x^*)$ of x^*. The study can be directed to show convergence to points satisfying

$$\nabla J(x^*) = 0$$

which will include all critical points. This is an open question and rigorously proving or disproving such a convergence would be a very nice contribution to the field. It would also significantly effect the popularity of the algorithm. In such formal studies, in addition to the mathematical formulation presented in this chapter, the graph representation of the neighborhood structure of the PSO algorithm may also be useful since it might be possible to utilize already available results from the graph theory literature in conjunction with possibly convex analysis or other techniques. We believe that although the fact that PSO does not use gradient information it is somehow gradient related. However, as can be seen also from the simulation results above, it may not be always guaranteed that it will converge to a critical point unless some extra conditions are imposed.

12.5.2 For Further Reading

The Particle Swarm Optimization algorithm was proposed in 1995 by Kennedy and Eberhart [66, 133] and since then has gained a lot of momentum. There has been also a book published on the topic [134]. The decentralized asynchronous PSO implementation with dynamic neighborhoods, which was mostly emphasized in this chapter, was proposed and studied in [2, 3, 82, 84] and most of this chapter is based on the work there. Note once more that although the main update/iteration formula of the decentralized asynchronous particle swarm optimization algorithm is the same as the classical implementations, the calculation of the neighborhood best $g_i(t)$ is based on a completely different philosophy.

There are studies which deal with analyzing particle trajectories or the convergence of the particles including [40, 130, 185, 186, 241, 248]. Note, however, that these works do not consider convergence to a critical point. Instead they deal with the convergence of the particles to some point or not. There are also studies, such as the ones in [15, 36], which investigate the effects of parameters on the performance of the algorithm and try to determine a good or standard set of parameters, for the best performance possible.

The studies in [132, 135] examine the performance of PSO with various neighborhood topologies, or basically the effect of the neighborhood topology on the performance of the algorithm. In addition, besides the works in [3, 84], there have also been other recent studies on PSO involving dynamic particle neighborhoods from different aspects [119, 173, 229]. Also, representative studies on parallel implementation of PSO can be found in [140, 214, 250], where in [214] a parallel synchronous implementation is adapted, whereas in [140, 250] a centralized parallel asynchronous implementation is used.

Some studies focus on improving the performance of PSO by combining it with other methods. Examples of such studies include [160, 161, 225], where evolutionary computation operators such as selection, crossover, and mutation are incorporated into the PSO algorithm, and [39, 54, 207], where other non-evolutionary approaches are used. There are also attempts to combine the Kalman Filter with PSO [174]. Adaptation of the parameters is also another approach considered in the literature [120, 222, 223, 257].

Examples of work on PSO inspired robotic search can be found in [4, 5, 57, 116, 117, 164, 197–199, 242], where [4, 5, 198, 242] use dynamic neighborhood topology. The distance in the search space based neighborhood topology in (12.8) (i.e., the nearest neighbor rule) allows one to represent the interactions in PSO-inspired multi-robot search applications.

References

1. Abelson, R.P., Aronson, E., McGuire, W.J., Newcomb, T.M., Rosenberg, M.J., Tannenbaum, P.H. (eds.): Theories of Cognitive Consistency: A Sourcebook. Rand-McNally, Chicago (1968)
2. Akat, S.B., Gazi, V.: Decentralized asynchronous particle swarm optimization. In: IEEE Swarm Intelligence Symposium (SIS 2008), St. Louis, Missouri (September 2008)
3. Akat, S.B., Gazi, V.: Particle swarm optimization with dynamic neighborhood topology: Three neighborhood strategies and preliminary results. In: IEEE Swarm Intelligence Symposium (SIS 2008), St. Louis, Missouri (September 2008)
4. Akat, S.B., Gazi, V., Marques, L.: Asynchronous particle swarm optimization based search with a multi-robot system. In: Workshop and Summer School on Evolutionary Computing (WSSEC 2008), Londonderry, Northern Ireland, pp. 64–67 (August 2008)
5. Akat, S.B., Gazi, V., Marques, L.: Asynchronous particle swarm optimization based search with a multi-robot system: Simulation and implementation on a real robotic system. Turkish Journal of Electrical Engineering and Computer Sciences (ELEKTRIK) 8(5), 749–764 (2010)
6. Akyildiz, I.F., Su, W., Sankarasubramniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Commununications Magazine 40(8), 102–114 (2002)
7. Alam, M., Claviez, M., Oesterhelt, D., Kessel, M.: Flagella and motility behavior of square bacteria. EMBO Journal 13(12), 2899–2903 (1984)
8. Anderson, B., Yu, C., Fidan, B., Hendrickx, J.: Control and information architectures for formations. In: Proc. IEEE Conference on Control Applications (Joint CCA/CACSD/ISIC), pp. 1127–1138 (October 2006)
9. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Transactions on Robotics and Automation 15(5), 818–828 (1999)
10. Angeli, D., Bliman, P.-A.: Stability of leaderless multi-agent systems. extension of a result by moreau, arXiv:math.OC/0411338 v1 (November 2004)
11. Armitage, J.: Bacterial tactic responses. Advances in Microbial Physiology 41, 229–290 (1999)
12. Audesirk, T., Audesirk, G.: Biology: Life on Earth, 5th edn. Prentice Hall, Englewood Cliffs (1999)
13. Bachmayer, R., Leonard, N.E.: Vehicle networks for gradient descent in a sampled environment. In: Proc. Conf. Decision Contr., Las Vegas, Nevada, pp. 112–117 (December 2002)

14. Balch, T., Arkin, R.C.: Behavior-based formation control for multirobot teams. IEEE Trans. on Robotics and Automation 14(6), 926–939 (1998)
15. Batton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)
16. Bender, J., Fenton, R.: On the flow capacity of automated highways. Transport. Sci. 4, 52–63 (1970)
17. Beni, G., Liang, P.: Pattern reconfiguration in swarms—convergence of a distributed asynchronous and bounded iterative algorithm. IEEE Trans. on Robotics and Automation 12(3), 485–490 (1996)
18. Berg, H.: Random Walks in Biology, New Expanded Edition. Princeton Univ. Press, Princeton (1993)
19. Berg, H.: Motile behavior of bacteria. Physics Today, pp. 24–29 (January 2000)
20. Berg, H., Brown, D.: Chemotaxis in *escherichia coli* analysed by three-dimensional tracking. Nature 239, 500–504 (1972)
21. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmond (1995)
22. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Athena Scientific, Belmont (1997)
23. Blat, Y., Eisenbach, M.: Tar-dependent and independent pattern formation by *salmonella typhimurium*. J. Bacteriology 177, 1683–1691 (1995)
24. Blondel, V.D., Hendrickx, J.M., Olshevsky, A., Tsitsiklis, J.: Convergence in multiagent coordination, consensus, and flocking. In: Proc. Conf. Decision Contr. and Proc. European Control Conf., Sevile, Spain, pp. 2996–3000 (December 2005)
25. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
26. Bondarev, A.G., Bondarev, S.A., Kostyleva, N.E., Utkin, V.I.: Sliding modes in systems with asymptotic state observers. Automation and Remote Control 46(6), 679–684 (1985)
27. Boscain, U., Piccoli, B.: Optimal Syntheses for Control Systems on 2-D Manifolds. Springer, New York (2004)
28. Breder, C.M.: Equations descriptive of fish schools and other animal aggregations. Ecology 35(3), 361–370 (1954)
29. Brockett, R.W.: Asymptotic stability and feedback stabilization. In: Millman, R.S., Sussmann, H.J. (eds.) Differential Geometric Control Theory, pp. 181–191. Birkhauser, Basel (1983)
30. Budrene, E., Berg, H.: Complex patterns formed by motile cells of *escherichia coli*. Nature 349, 630–633 (1991)
31. Budrene, E., Berg, H.: Dynamics of formation of symmetrical patterns by chemotactic bacteria. Nature 376, 49–53 (1995)
32. Bullo, F., Corts, J., Martnez, S.: Distributed Control of Robotic Networks. Series in Applied Mathematics. Princeton University Press, Princeton (2009)
33. Butenko, S., Murphey, R., Pardalos, P. (eds.): Cooperative Control: Models, Applications and Algorithms. Kluwer Academic, Dordrecht (2003)
34. Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, New Jersey (April 2001)
35. Campion, G., Bastin, G., Dandrea-Novel, B.: Structural properties and classification of kinematic and dynamicmodels of wheeled mobile robots. IEEE Tr. on Robotics and Automation 12(1), 47–62 (1996)

36. Carlisle, A., Dozier, G.: An off-the shelf pso. In: Proceedings of Workshop on Particle Swarm Optimization (2001)
37. Chu, T., Wang, L., Chen, T.: Self-organized motion in anisotropic swarms. Journal of Control Theory and Applications 1, 77–81 (2003)
38. Chu, Y.-C., Huang, J.: A neural-network method for the nonlinear servomechanism problem. IEEE Trans. on Neural Networks 10(6), 1412–1423 (1999)
39. Clerc, M.: The swarm and the queen towards a deterministic and adaptive particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1951–1957 (1999)
40. Clerc, M., Kennedy, J.: The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. on Evolutionary Computation 6(1), 58–73 (2002)
41. Csahok, Z., Vicsek, T.: Lattice-gas model for collective bilogical motion. Physical Review E 52(5), 5297–5303 (1995)
42. Şamiloglu, A.T., Gazi, V., Koku, A.B.: Effects of asynchronism and neighborhood size on clustering in self-propelled particle systems. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 665–676. Springer, Heidelberg (2006)
43. Şamiloglu, A.T., Gazi, V., Koku, A.B.: An empirical study on the motion of self-propelled particles with turn angle restrictions. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433. Springer, Heidelberg (2007)
44. Czirok, A., Barabasi, A.-L., Vicsek, T.: Collective motion of self-propelled particles: Kinetic phase transition in one dimension. Physical Review Letters 82(1), 209–212 (1999)
45. Czirok, A., Ben-Jacob, E., Cohen, I., Vicsek, T.: Formation of complex bacterial colonies via self-generated vortices. Physical Review E 54(2), 1791–1801 (1996)
46. Czirok, A., Stanley, H.E., Vicsek, T.: Spontaneously ordered motion of self-propelled partciles. Journal of Physics A: Mathematical, Nuclear and General 30, 1375–1385 (1997)
47. Czirok, A., Vicsek, T.: Collective behavior of interacting self-propelled particles. Physica A 281, 17–29 (2000)
48. Darbha, S., Rajagopal, K.R.: Intelligent cruise control systems and traffic flow stability. Transportation Research Part C 7, 329–352 (1999)
49. Das, A., Fierro, R., Kumar, V., Ostrowski, J.: A vision-based formation control framework. IEEE Trans. on Robotics and Automation 18(5), 813–825 (2002)
50. Davison, E.J.: The robust control of a servomechanism problem for linear time-invariant multivariable systems. IEEE Trans. on Automatic Control AC-21(1), 25–34 (1976)
51. Davison, E.J.: The robust decentralized control of a general servomechanism problem. IEEE Trans. on Automatic Control AC-21(1), 14–24 (1976)
52. Davison, E.J., Goldenberg, A.: Robust decentralized control of a general servomechanism problem: The servocompensator. Automatica 11, 461–471 (1975)
53. DeCarlo, R.A., Zak, S.H., Matthews, G.P.: Variable structure control of nonlinear multivariable systems: A tutorial. Proceedings of the IEEE 76(3), 212–232 (1988)
54. Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation 8, 225–239 (2004)
55. Desai, J.P., Ostrowski, J., Kumar, V.: Controlling formations of multiple mobile robots. In: Proc. of IEEE International Conference on Robotics and Automation, Leuven, Belgium, pp. 2864–2869 (May 1998)

56. Desai, J.P., Ostrowski, J., Kumar, V.: Modeling and control of formations of nonholonomic mobile robots. IEEE Trans. on Robotics and Automation 17(6), 905–908 (2001)
57. Doctor, S., Venayagamoorthy, G., Gudise, A.: Optimal PSO for collective robotic search applications. In: IEEE Congress on Evolutionary Computation (CEC 2004), Portland, USA, vol. 2, pp. 1390–1395 (2004)
58. Dorigo, M., Birattari, M., Sttzle, T.: Ant colony optimization artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1(4), 28–39 (2006)
59. Dorigo, M., Sahin, E. (eds.): Special Issue on Swarm Robotics. Autonomous Robots, vol. 17(2-3) (September 2004)
60. Dorigo, M., Sttzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
61. Drakunov, S.V.: Sliding-mode observers based on equivalent control method. In: Proc. Conf. Decision Contr., Tucson, AZ, USA, pp. 2368–2369 (December 1992)
62. Drakunov, S.V., Utkin, V.I.: Sliding mode observers: Tutorial. In: Proc. Conf. Decision Contr., New Orleans, LA, USA, pp. 3376–3378 (December 1995)
63. Dubins, L.E.: On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79(3), 497–516 (1957)
64. Duran, S., Gazi, V.: Adaptive formation control and target tracking in a class of multiagent systems. In: Proc. American Control Conf., Baltimore, MD, USA, pp. 75–80 (June-July 2010)
65. Durrett, R., Levin, S.: The importance of being discrete (and spatial). Theoretical Population Biology 46, 363–394 (1994)
66. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Sixth International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
67. Edelstein-Keshet, L.: Mathematical Models in Biology. Brikhäuser Mathematics Series. The Random House, New York (1989)
68. Egerstedt, M., Hu, X.: Formation constrained multi-agent control. IEEE Trans. on Robotics and Automation 17(6), 947–951 (2001)
69. Eren, T., Anderson, B., Morse, A., Whiteley, W., Belhumeur, P.: Operations on rigid formations of autonomous agents. Communications in Information and Systems 3(4), 223–258 (2004)
70. Erkmen, A.M., Yegenoglu, F., Stephanou, H.E.: Entropy driven on-line control of autonomous robots. Journal of Intelligent and Robotic Systems 2, 109–121 (1989)
71. Fang, L., Antsaklis, P.J., Tzimas, A.: Asynchronous consensus protocols: Preliminary results, simulations and open questions. In: Proc. Conf. Decision Contr. and Proc. European Control Conf., Sevile, Spain, pp. 2194–2199 (December 2005)
72. Fax, J.A., Murray, R.M.: Information flow and cooperative control of vehicle formations. IEEE Trans. on Automatic Control 49(9), 1465–1476 (2004)
73. Fidan, F., Gazi, V.: Target tracking using adaptive gain backstepping control. In: IFAC Workshop on Adaptation and Learning in Control and Signal Processing, Antalya, Turkey (August 2010)
74. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous oblivious robots with limited visibility. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 247–258. Springer, Heidelberg (2001)
75. Francis, B.A.: The linear multivariable regulator problem. SIAM Journal on Control and Optimization 15(3), 486–505 (1977)
76. Francis, B.A., Wonham, W.M.: The internal model principle for linear multivariable regulators. Appl. Math. Optimiz. 2, 170–195 (1975)

77. Francis, B.A., Wonham, W.M.: The internal model principle of control theory. Automatica 12, 457–465 (1976)
78. Gazi, V.: Stability analysis of swarms. Ph.D. dissertation, The Ohio State University (August. 2002)
79. Gazi, V.: Formation control of mobile robots using decentralized nonlinear servomechanism. In: 12'th Meditteranean Conference on Control and Automation, Kusadasi, Turkey (June 2004)
80. Gazi, V.: Formation control of a multi-agent system using nonlinear servomechanism. Int. J. Control 78(8), 554–565 (2005)
81. Gazi, V.: Swarm aggregations using artificial potentials and sliding mode control. IEEE Trans. on Robotics 21(6), 1208–1214 (2005)
82. Gazi, V.: Asynchronous particle swarm optimization. In: IEEE 15th Signal Processing and Communication Applications Conference, Eskisehir, Turkey (June 2007) (in Turkish)
83. Gazi, V.: Output regulation of a class of linear systems with switched exosystems. International Journal of Control 80(10), 1665–1675 (2007)
84. Gazi, V.: Particle swarm optimization with dynamic neighborhood. In: IEEE 15th Signal Processing and Communication Applications Conference, Eskisehir, Turkey (June 2007) (in Turkish)
85. Gazi, V. (ed.): Special Issue on Swarm Robotics. Turkish Journal of Electrical Engineering and Computer Sciences, vol. 15(2) (July 2007)
86. Gazi, V.: Stability of a discrete-time asynchronous swarm with time-dependent communication links. IEEE Trans. on Systems, Man, and Cybernetics: Part B 38(1), 267–274 (2008)
87. Gazi, V., Fidan, B.: Coordination and control of multi-agent dynamic systems: Models and approaches. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433, pp. 71–102. Springer, Heidelberg (2007)
88. Gazi, V., Fidan, B., Hanay, Y.S., Köksal, M.I.: Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques. Turkish Journal of Electrical Engineering and Computer Sciences 15(2) (July 2007)
89. Gazi, V., Fidan, B., Zhai, S.: Single view depth estimation based formation control of robotic swarms: Implementation using realistic robot simulator. In: 11'th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR), Coimbra, Portugal, pp. 1079–1089 (September 2008)
90. Gazi, V., Köksal, M.I., Fidan, B.: Aggregation in a swarm of non-holonomic agents using artificial potentials and sliding mode control. In: Proc. European Control Conf., Kos, Greece, pp. 1485–1491 (July 2007) (to appear)
91. Gazi, V., Passino, K.M.: Stability of a one-dimensional discrete-time asynchronous swarm. In: Proc. of the Joint IEEE Int. Symp. on Intelligent Control/IEEE Conf. on Control Applications, Mexico City, Mexico, pp. 19–24 (September 2001)
92. Gazi, V., Passino, K.M.: Stability analysis of swarms. IEEE Trans. on Automatic Control 48(4), 692–697 (2003)
93. Gazi, V., Passino, K.M.: A class of attraction/repulsion functions for stable swarm aggregations. Int. J. Control 77(18), 1567–1579 (2004)
94. Gazi, V., Passino, K.M.: Stability analysis of social foraging swarms. IEEE Trans. on Systems, Man, and Cybernetics: Part B 34(1), 539–557 (2004)
95. Gazi, V., Passino, K.M.: Stability of a one-dimensional discrete-time asynchronous swarm. IEEE Trans. on Systems, Man, and Cybernetics: Part B 35(4), 834–841 (2005)

96. Gazi, V., Passino, K.M.: Decentralized output regulation of a class of nonlinear systems. Int. J. Control 79(12), 1512–1522 (2006)
97. Gazi, V., Passino, K.M.: Swarm stability. In: Levine, W.S. (ed.) The Control Handbook, 2nd edn., ch. 9. CRC Press, Boca Raton (2010)
98. Gelenbe, E., Schmajuk, N., Staddon, J., Reif, J.: Autonomous search by robots and animals: A survey. Robotics and Autonomous Systems 22, 23–34 (1997)
99. Giulietti, F., Pollini, L., Innocenti, M.: Autonomous formation flight. IEEE Control Systems Magazine 20(6), 34–44 (2000)
100. Godsil, C., Royle, G.: Algebraic Graph Theory. Graduate Texts in Mathematics, vol. 207. Springer, New York (2001)
101. Gordon, N., Wagner, I.A., Bruckstein, A.M.: Gathering multiple robotic a(ge)nts with limited sensing capabilities. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 142–153. Springer, Heidelberg (2004)
102. Grindrod, P.: Models of individual aggregation or clustering in single and multi-species communities. Journal of Mathematical Biology 26, 651–660 (1988)
103. Grünbaum, D.: Schooling as a strategy for taxis in a noisy environment. In: Parrish, J.K., Hamner, W.M. (eds.) Animal Groups in Three Dimensions, pp. 257–281. Cambridge Iniversity Press, New York (1997)
104. Grünbaum, D.: Schooling as a strategy for taxis in a noisy environment. Evolutionary Ecology 12, 503–522 (1998)
105. Grünbaum, D., Okubo, A.: Modeling social animal aggregations. In: Frontiers in Theoretical Biology. Lecture Notes in Biomathematics, vol. 100, pp. 296–325. Springer, New York (1994)
106. Gueron, S., Levin, S.A.: The dynamics of group formation. Mathematical Biosciences 128, 243–264 (1995)
107. Gueron, S., Levin, S.A., Rubenstein, D.I.: The dynamics of herds: From individuals to aggregations. Journal of Theoretical Biology 182, 85–98 (1996)
108. Gül, E., Gazi, V.: Adaptive internal model based formation control of a class of multi-agent systems. In: Proc. American Control Conf., Baltimore, MD, USA, pp. 4800–4805 (June-July 2010)
109. Guldner, J., Utkin, V.I.: Sliding mode control for an obstacle avoidance strategy based on an harmonic potential field. In: Proc. Conf. Decision Contr., San Antonio, Texas, pp. 424–429 (December 1993)
110. Guldner, J., Utkin, V.I.: Stabilization on non-holonimic mobile robots using lyapunov functions for navigation and sliding mode control. In: Proc. Conf. Decision Contr., Lake Buena Vista, Florida, pp. 2967–2972 (December 1994)
111. Guldner, J., Utkin, V.I.: Sliding mode control for gradient tracking and robot navigation using artificial potential fields. IEEE Trans. on Robotics and Automation 11(2), 247–254 (1995)
112. Guldner, J., Utkin, V.I.: Tracking the gradient of artificial potential fields: Sliding mode control for mobile robots. Int. J. Control 63(3), 417–432 (1996)
113. Guldner, J., Utkin, V.I., Hashimoto, H., Harashima, F.: Tracking gradients of artificial potential fields with non-holonomic mobile robots. In: Proc. American Control Conf., Seattle, Washington, pp. 2803–2804 (June 1995)
114. Hanay, Y.S., Hünerli, H.V., Köksal, M.I., Şamiloğlu, A.T., Gazi, V.: Formation control with potential functions and newton's iteration. In: European Control Conference, Kos, Greece, pp. 4584–4590 (July 2007)

115. Haskara, I., Özgüner, Ü., Utkin, V.I.: On sliding mode observers via equivalent control approach. Int. J. Control 71(6), 1051–1067 (1998)
116. Hereford, J.: A distributed particle swarm algorithm for swarm robotic applications. In: IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, Canada, vol. 2, pp. 1678–1685 (2006)
117. Hereford, J., Siebold, M., Nichols, S.: Using the particle swarm optimization algorithm for robotic search applications. In: IEEE Symposium on Swarm Intelligence (SIS 2007), Honolulu, USA, pp. 53–59 (2007)
118. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
119. Hu, X., Eberhart, R.C.: Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: IEEE Congress on Evolutionary Computation, pp. 1677–1681 (2002)
120. Hu, X., Eberhart, R.: Adaptive particle swarm optimization: detection and response to dynamic systems. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, pp. 1666–1670 (2002)
121. Huang, J., Rugh, W.J.: On a nonlinear multivariable servomechanism problem. Automatica 26, 963–972 (1990)
122. Huang, J., Rugh, W.J.: An approximate method for the nonlinear servomechanism problem. IEEE Trans. on Automatic Control 37(9), 1395–1398 (1992)
123. Huang, J., Rugh, W.J.: Stabilization on zero-error manifolds and the nonlinear servomechanism problem. IEEE Trans. on Automatic Control 37(7), 1009–1013 (1992)
124. Isidori, A.: Nonlinear Control Systems, 3rd edn. Springer, Heidelberg (1995)
125. Isidori, A., Byrnes, C.I.: Output regulation of nonlinear systems. IEEE Trans. on Automatic Control 35(2), 131–140 (1990)
126. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. on Automatic Control 48(6), 988–1001 (2003)
127. Jin, D., Gao, L.: Stability analysis of swarm based on double integrator model. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) ICIC 2006. LNCS (LNBI), vol. 4115, pp. 201–210. Springer, Heidelberg (2006)
128. Jin, D., Gao, L.: Stability analysis of a double integrator swarm model related to position and velocity. Transactions of the Institute of Measurement and Control 30(3), 275–293 (2008)
129. Jin, K., Liang, P., Beni, G.: Stability of synchronized distributed control of discrete swarm structures. In: Proc. of IEEE International Conference on Robotics and Automation, San Diego, California, pp. 1033–1038 (May 1994)
130. Kadirkamanathan, V., Selvarajah, K., Fleming, P.: Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Transactions on Evolutionary Computation, 245–255 (2006)
131. Kang, W., Xi, N., Sparks, A.: Formation control of autonomous agents in 3d space. In: IEEEICRA, San Francisco, CA, pp. 1755–1760 (April 2000)
132. Kennedy, J.: Small worlds and mega minds: Effects of neighborhood topology on particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999), vol. 3, pp. 1931–1938 (1999)
133. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
134. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publisher, San Francisco (2001)

135. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, vol. 2, pp. 1671–1676 (2002)
136. Khalil, H.K.: Nonlinear Systems, 2nd edn. Prentice Hall, Upper Saddle River (1996)
137. Khalil, H.K.: On the design of robust servomechanisms for minimum phase nonlinear systems. Int. J. Robust and Nonlinear Control 10, 339–361 (2000)
138. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 5(1), 90–98 (1986)
139. Koditschek, D.E.: An approach to autonomous robot assembly. Robotica 12, 137–155 (1994)
140. Koh, B., George, A., Haftka, R., Fregly, B.: Parallel asynchronous particle swarm optimization. International Journal for Numerical Methods in Engineering 67, 578–595 (2004)
141. Krener, A.J.: The construction of optimal linear and nonlinear regulators. In: Isidori, A., Tarn, T.J. (eds.) Systems, Models, and Feedback: Theory and Applications, pp. 301–322. Birkhauser, Basel (1992)
142. Kubik, A.: Towards a formalization of emergence. Artificial Life 9, 41–65 (2003)
143. Kumar, V.J., Leonard, N.E., Morse, A.S. (eds.): Cooperative Control: 2003 Block Island Workshop on Cooperative Control. Lecture Notes in Control and Information Sciences, vol. 309. Springer, Heidelberg (2003)
144. Lawton, J.R.T., Beard, R.W., Young, B.J.: A decentralized approach to formation maneuvers. IEEE Trans. on Robotics and Automation 19(6), 933–941 (2003)
145. Leonard, N.E., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. In: Proc. Conf. Decision Contr., Orlando, FL, pp. 2968–2973 (December 2001)
146. Levine, H., Rappel, W.-J.: Self-organization in systems of self-propelled particles. Physical Review E 63(1), 017 101-1–017 101-4 (2001)
147. Li, W.: Stability analysis of swarms with general topology. IEEE Trans. on Systems, Man, and Cybernetics: Part B 38(4), 1084–1097 (2008)
148. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem. In: Proc. Conf. Decision Contr., Maui, Hawaii, USA, pp. 1508–1513 (December 2003)
149. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem - the asynchronous case. In: Proc. Conf. Decision Contr., Atlantis, Paradise Island, Bahamas, pp. 1926–1931 (December 2004)
150. Lin, Z., Francis, B., Maggiore, M.: Necessary and sufficient graphial conditions for formation control of unicycles. IEEE Trans. on Automatic Control 50(1), 121–127 (2005)
151. Liu, Y., Passino, K.M.: Biomimicry of social foraging behavior for distributed optimization: Models, principles, and emergent behaviors. Journal of Optimization Theory and Applications 115(3), 603–628 (2002)
152. Liu, Y., Passino, K.M.: Stable social foraging swarms in a noisy environment. IEEE Transactions on Automatic Control 49(1), 30–44 (2004)
153. Liu, Y., Passino, K.M.: Stable social foraging swarms in a noisy environment. IEEE Transactions on Automatic Control 49(1), 30–44 (2004)
154. Liu, Y., Passino, K.M., Polycarpou, M.: Stability analysis of one-dimensional asynchronous mobile swarms. In: Proc. Conf. Decision Contr., Orlando, FL, pp. 1077–1082 (December 2001)
155. Liu, Y., Passino, K.M., Polycarpou, M.: Stability analysis of one-dimensional asynchronous swarms. In: Proc. American Control Conf., Arlington, VA, pp. 716–721 (June 2001)

156. Liu, Y., Passino, K.M., Polycarpou, M.M.: Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. In: Proc. American Control Conf., Anchorage, Alaska, pp. 1278–1283 (May 2002)
157. Liu, Y., Passino, K.M., Polycarpou, M.M.: Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. IEEE Trans. on Automatic Control 48(1), 76–95 (2003)
158. Liu, Y., Passino, K.M., Polycarpou, M.M.: Stability analysis of one-dimensional asynchronous swarms. IEEE Trans. on Automatic Control 48(10), 1848–1854 (2003)
159. Lizana, M., Padron, V.: A specially discrete model for aggregating populations. Journal of Mathematical Biology 38, 79–102 (1999)
160. Løvbjerg, M., Rasmussen, T., Krink, T.: Hybrid particle swarm optimizer with breeding and subpopulations. In: Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO 2001), vol. 1, pp. 469–476 (2001)
161. Løvbjerg, M., Krink, T.: Extending particle swarms with self-organized critically. In: Proceedings of Fourth Congress on Evolutionary Computation (CEC 2002), vol. 2, pp. 1588–1593 (2002)
162. Lowe, G., Meister, M., Berg, H.: Rapid rotation of flagellar bundles in swimming bacteria. Nature 325, 637–640 (1987)
163. Köksal, M.I., Gazi, V., Fidan, B., Ordonez, R.: Tracking a Maneuvering Target with a Non-holonomic Agent Using Artificial Potentials and Sliding Mode Control. In: Mediterranean Conference on Control and Automation, Ajaccio, Corsica, France, pp. 1174–1179 (June 2008)
164. Marques, L., Nunes, U., de Almedia, A.: Particle swarm-based olfactory guided search. Autonomous Robots 20(3), 277–287 (2006)
165. Marshall, J.A., Broucke, M.E., Francis, B.A.: Pursuit formations of unicycles. Automatica 42(1), 3–12 (2006)
166. Marshall, J., Broucke, M., Francis, B.: Formations of vehicles in cyclic pursuit. IEEE Trans. on Automatic Control 49(11), 1963–1974 (2004)
167. Merheb, A.-R., Gazi, V., Sezer-Uzol, N.: Experimental study of robot formation control and navigation using potential functions and panel method. In: The Joint 41st International Symposium on Robotics (ISR 2010) and the 6th German Conference on Robotics (ROBOTIK 2010), Munich, Germany, pp. 586–593 (June 2010)
168. Merheb, A.-R., Gazi, V., Sezer-Uzol, N.: Implementation of robot formation control and navigation using real-time panel method. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan, pp. 5001–5006 (October 2010)
169. Michel, A.N., Hou, L., Liu, D.: Stability of Dynamical Systesm: Continuous, Discontinuous, and Discrete Systems. Systems and Control: Foundations and Applications. Birkhauser, Boston (2008)
170. Michel, A.N., Miller, R.K.: Qualitative Analysis of Large Scale Dynamical Systems. Academic Press, New York (1977)
171. Mikhailov, A.S., Zanette, D.H.: Noise-induced breakdown of coherent collective motion in swarms. Physical Review E 60(4), 4571–4575 (1999)
172. Mogilner, A., Edelstein-Keshet, L.: A non-local model for a swarm. Journal of Mathematical Biology 38, 534–570 (1999)
173. Mohais, A., Mendes, R., Ward, C., Posthoff, C.: Neighborhood re-structuring in particle swarm optimization. In: Australian Conference on Artificial Intelligence, pp. 776–785 (2005)

174. Monson, C.K., Seppi, K.D.: The kalman swarm: A new approach to particle motion in swarm optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 140–150. Springer, Heidelberg (2004)
175. Moreau, L.: A note on leaderless coordination via bidirectional and unidirectional time-dependent commnication. In: Int. Symp. on Mathematical Theory of Networks and Systems (2004)
176. Moreau, L.: Stability of multiagent systems with time-dependent communication links. IEEE Trans. on Automatic Control 50(2), 169–182 (2005)
177. Murphey, R., Pardalos, P. (eds.): Cooperative Control and Optimization. Kluwer Academic, Dordrecht (2002)
178. Murray, J.D.: Mathematical Biology. Springer, New York (1989)
179. Neidhardt, F., Ingraham, J., Schaechter, M.: Physiology of the Bacterial Cell: A Molecular Approach. Sinauer Associates, Inc., Pub., Massachusetts (1990)
180. Ögren, P., Egerstedt, M., Hu, X.: A control Lyapunov function approach to multi-agent coordination. In: Proc. Conf. Decision Contr., Orlando, FL, pp. 1150–1155 (December 2001)
181. Okubo, A.: Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. Advances in Biophysics 22, 1–94 (1986)
182. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Trans. on Automatic Control 51(3), 401–420 (2006)
183. Olfati-Saber, R., Murray, R.M.: Distributed cooperative control of multiple vehicle formations using structural potential functions. In: Proc. IFAC World Congress, Barcelona, Spain (June 2002)
184. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. on Automatic Control 49(9), 1520–1533 (2004)
185. Ozcan, E., Mohan, C.K.: Particle swarm optimization: Surfing the waves. In: IEEE Congress on Evolutionary Computation, pp. 1939–1944 (1999)
186. Ozcan, E., Mohan, C.: Analysis of a simple particle of a simple particle swarm optimization system. In: Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE 1998), pp. 253–258 (1998)
187. Ögren, P., Fiorelli, E., Leonard, N.E.: Formations with a mission: Stable coordination of vehicle group maneuvers. In: Symposium on Mathematical Theory of Networks and Systems (August 2002)
188. Pand, A., Seiler, P., Hedrick, K.: Mesh stability of look-ahead interconnected systems. IEEE Trans. on Automatic Control 47(2), 403–407 (2002)
189. Parrish, J.K., Hamner, W.M. (eds.): Animal Groups in Three Dimensions. Cambridge University Press, Cambridge (1997)
190. Parrish, J.K., Viscido, S.V., Grünbaum, D.: Self-organized fish school: An examination of emergent properties. Biol. Bull. 202, 296–305 (2002)
191. Partridge, B.L.: The structure and function of fish schools. Scientific American 245, 114–123 (1982)
192. Passino, K.M., Burgess, K.L.: Stability Analysis of Discrete Event Systems. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley and Sons, New York (1998)
193. Passino, K.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine 22(3), 52–67 (2002)
194. Passino, K.: Biomimicry for Optimization, Control, and Automation. Springer, London (2005)

195. Pettersen, K.Y., Gravdahl, J.T., Nijmeijer, H. (eds.): Group Coordination and Cooperative Control. Lecture Notes in Control and Information Sciences, vol. 336. Springer, Heidelberg (2006)
196. Polycarpou, M.M., Yang, Y., Passino, K.M.: Cooperative control of distributed multi-agent systems. to appear in IEEE Control Systems Magazine
197. Pugh, J., Martinoli, A.: Inspiring and modelling multi-robot search with particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002), vol. 2, pp. 1666–1670 (2002)
198. Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with particle swarm optimization. In: IEEE Swarm Intelligence Symposium (SIS 2007), pp. 332–339 (April 2007)
199. Pugh, J., Segapelli, L., Martinoli, A.: Applying aspects of multi-robot search to particle swarm optimization. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 506–507. Springer, Heidelberg (2006)
200. Rauch, E.M., Millonas, M.M., Chialvo, D.R.: Pattern formation and functionality in swarm models. Physics Letters A 207, 185–193 (1995)
201. Reif, J.H., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. Robotics and Autonomous Systems 27, 171–194 (1999)
202. Ren, W., Beard, R.: Distributed Consensus in Multi-vehicle Cooperative Control. Communication and Control Engineering Series. Springer, New York (2007)
203. Ren, W., Beard, R.W.: Consensus seeking in multi-agent systems under dynamically changing interaction topologies. IEEE Trans. on Automatic Control 50(5), 655–661 (2005)
204. Ren, W., Beard, R.W.: Distributed Consensus in Multi-vehicle Cooperative Control. Communications and Control Engineering. Springer, UK (2008)
205. Ren, W., Beard, R.W., Atkins, E.M.: A survey of consensus problems in multi-agent coordination. In: Proc. American Control Conf., Portland, OR, USA, pp. 1859–1864 (June 2005)
206. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Comp. Graph. 21(4), 25–34 (1987)
207. Riget, J., Vesterstrøm, J.: A diversity-guided particle swarm optimizer- the arpso. Department of Computer Science, University of Aarhus, Tech. Rep., eVALife Technical Report (2002)
208. Rimon, E., Koditschek, D.E.: Exact robot navigation using artificial potential functions. IEEE Trans. on Robotics and Automation 8(5), 501–518 (1992)
209. Sahin, E., Spears, W.M. (eds.): SAB 2004. LNCS, vol. 3342. Springer, Heidelberg (2005)
210. Sahin, E., Spears, W.M., Winfield, A.F.T. (eds.): SAB 2006. LNCS, vol. 4433. Springer, Heidelberg (2007)
211. Samiloglu, A.T., Gazi, V., Koku, A.B.: Comparison of three orientation agreement strategies in self-propelled particle systems with turn angle restrictions in synchronous and asynchronous settings. Asian Journal of Control 10(2), 212–232 (2008)
212. Sandeep, S., Fidan, B., Yu, C.: Decentralized cohesive motion control of multi-agent formations. In: Proc. 14th Mediterranean Conference on Control and Automation, Ancona, Italy (June 2006)
213. Savla, K., Bullo, F., Frazzoli, E.: On traveling salesperson problems for Dubins' vehicle: stochastic and dynamic environments. In: Proc. 44th IEEE Conference on Decision and Control and the European Control Conference 2005, pp. 4530–4535 (December 2005)

214. Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.T.: Parallel global optimization with the particle swarm algorithm. International Journal for Numerical Methods in Engineering 61, 2296–2315 (2004)
215. Segall, J., Block, S., Berg, H.: Temporal comparisons in bacterial chemotaxis. Proc. of the National Academy of Sciences 83, 8987–8991 (1986)
216. Sepulchre, R., Palay, D., Leonard, N.E.: Collective motion and oscillator synchronization. In: Kumar, V.J., Leonard, N.E., Morse, A.S. (eds.) Cooperative Control: 2003 Block Island Workshop on Cooperative Control. LNCIS, vol. 309. Springer, Heidelberg (2005)
217. Serrani, A., Isidori, A.: Global robust output regulation for a class of nonlinear systems. Systems and Control Letters 39(2), 133–139 (2000)
218. Serrani, A., Isidori, A.: Semiglobal nonlinear output regulation with adaptive internal model. In: Proc. Conf. Decision Contr., Sydney, Australia, pp. 1649–1654 (December 2000)
219. Serrani, A., Isidori, A., Marconi, L.: Semiglobal robust output regulation of minimum-phase nonlinear systems. Int. J. Robust and Nonlinear Control 10, 379–396 (2000)
220. Serrani, A., Isidori, A., Marconi, L.: Semiglobal nonlinear output regulation with adaptive internal model. IEEE Trans. on Automatic Control 46(8), 1178–1194 (2001)
221. Shamma, J. (ed.): Cooperative Control of Distributed Multi-Agent Systems. Wiley-Interscience, Hoboken (2008)
222. Shi, Y., Eberhart, R.: Fuzzy adaptive particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2001), vol. 1, pp. 101–106 (2001)
223. Shi, Y., Eberhart, R.: Particle swarm optimization with fuzzy adaptive inertia weight. In: Proceedings of the Workshop on Particle Swarm Optimization, vol. 1, pp. 101–106 (2001)
224. Shimoyama, N., Sugawa, K., Mizuguchi, T., Hayakawa, Y., Sano, M.: Collective motion in a system of motile elements. Physical Review Letters 76(20), 3870–3873 (1996)
225. Stacey, T., Jancic, M., Grundy, I.: Particle swarm optimization with mutation. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2003), vol. 2, pp. 1425–1430 (2003)
226. Stephens, D., Krebs, J.: Foraging Theory. Princeton Univ. Press, Princeton (1986)
227. Strogatz, S.H., Mirollo, R.E., Matthews, P.C.: Coupled nonlinear oscillators below the synchronization threshhold: Relaxation by generalized landau damping. Physical Review Letters 68(18), 2730–2733 (1992)
228. Strogatz, S.H., Stewart, I.: Coupled oscillators and biological synchronization. Scientific American, 102–109 (December 1993)
229. Suganthan, N.: Particle swarm optimiser with neighborhood operator. In: IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1945–1950 (1999)
230. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing 28(4), 1347–1363 (1999)
231. Swaroop, D.: String stability of interconnected systems: An application to platooning in automated highway systems. Ph.D. dissertation, Department of Mechanical Engineering, University of California, Berkeley (1995)
232. Swaroop, D., Hedrick, J.K., Chien, C.C., Ioannou, P.: A comparison of spacing and headway control laws for automatically controlled vehicles. Vehicle System Dynamics 23, 597–625 (1994)
233. Tabuada, P., Pappas, G.J., Lima, P.: Feasable formations of multi-agent systems. In: Proc. American Control Conf., Arlington, VA, pp. 56–61 (June 2001)

234. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Stable flocking of mobile agents, part i: Fixed topology. In: Proc. Conf. Decision Contr., Maui, Hawaii, pp. 2010–2015 (December 2003)
235. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Stable flocking of mobile agents, part ii: Dynamic topology. In: Proc. Conf. Decision Contr., Maui, Hawaii, pp. 2016–2021 (December 2003)
236. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Flocking in teams of nonholonomic agents. In: Kumar, V., Leonard, N., Morse, A. (eds.) Cooperative Control. LNCIS, vol. 309, pp. 229–239. Springer, Heidelberg (2005)
237. Tanner, H., Pappas, G.J., Kumar, V.: Leader-to-formation stability. IEEE Trans. on Robotics and Automation 20(3), 443–455 (2004)
238. Tomlin, C., Mitchell, I., Ghosh, R.: Safety verification of conflict resolution manoeuvres. IEEE Tr. on Intelligent Transportation Systems 2(2), 110–120 (2001)
239. Toner, J., Tu, Y.: Long-range order in a two-dimensional dynamical *xy* model: How birds fly together. Physical Review Letters 75(23), 4326–4329 (1995)
240. Toner, J., Tu, Y.: Flocks, herds, and schools: A quantitative theory of flocking. Physical Review E 58(4), 4828–4858 (1998)
241. Trelea, I.C.: The particle swarm optimization algorithm: Convergence analysis and parameter selection. Information Processing Letters 85, 317–325 (2003)
242. Tuduev, M., Ataş, Y., Souza, P., Gazi, V., Marques, L.: Cooperative chemical concentration map building using decentralized asynchronous particle swarm optimization based search algorithm by mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan, pp. 4175–4180 (October 2010)
243. Tuduev, M., Kırtay, M., Souza, P., Gazi, V., Marques, L.: Chemical concentration map building through bacterial foraging optimization based search algorithm by mobile robots. In: IEEE Conference on Systems, Man and Cybernetics (SMC 2010), Istanbul, Turkey, pp. 3242–3249 (October 2010)
244. Utkin, V.I.: Variable structure systems with sliding modes. IEEE Trans. on Automatic Control AC-22(2), 212–222 (1977)
245. Utkin, V.I.: Sliding Modes in Control and Optimization. Springer, Heidelberg (1992)
246. Utkin, V.I., Drakunov, S.V., Hashimoto, H., Harashima, F.: Robot path obstacle avoidance control via sliding mode approach. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems, Osaka, Japan, pp. 1287–1290 (November 1991)
247. Utkin, V.I., Guldner, J., Shi, J.: Sliding Mode Control in Electromechanical Systems. CRC Press, Boca Raton (1999)
248. van Den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information Sciences (176), 937–971 (February 2006)
249. Vaughan, R., Sumpter, N., Henderson, J., Frost, A., Cameron, S.: Experiments in automatic flock control. Robotics and Autonomous Systems 31, 109–117 (2000)
250. Venter, G., Sobieski, J.S.: A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. In: 6'th World Congress of Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil, May 30-June 03 (2005)
251. Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., Shochet, O.: Novel type of phase transition in a system of self-driven particles. Physical Review Letters 75(6), 1226–1229 (1995)
252. Vicsek, T., Czirok, A., Farkas, I.J., Helbing, D.: Application of statistical mechanics to collective motion in biology. Physica A 274, 182–189 (1999)

253. Wang, J., Huang, J., Yau, S.T.: Approximate nonlinear output regulation based on the universal approximation theorem. Int. J. Robust and Nonlinear Control 10, 439–456 (2000)
254. Warburton, K., Lazarus, J.: Tendency-distance models of social cohesion in animal groups. Journal of Theoretical Biology 150, 473–488 (1991)
255. Watson, G.A.: Approximation theory and numerical methods. John Wiley, New York (1980)
256. Woodward, D., Tyson, R., Myerscough, M., Murray, J., Budrene, E., Berg, H.: Spatio-temporal patterns generated by salmonella typhimurium. Biophysi. J. 68, 2181–2189 (1995)
257. Xie, X., Zhang, W., Yang, Z.: Adaptive particle swarm optimization on individual level. In: Proceedings of 6th International Conference on Signal Processing (ICSP 2002), vol. 2, pp. 1215–1218 (2002)
258. Yamaguchi, H.: A cooperative hunting behavior by mobile-robot troops. The International Journal of Robotics Research 18(8), 931–940 (1999)
259. Yao, J., Ordonez, R., Gazi, V.: Swarm tracking using artificial potentials and sliding mode control. ASME J. Dyn. Syst., Meas., Contr. 129(5), 749–754 (2007)
260. Ye, X., Huang, J.: Decentralized adaptive output regulation for a class of large-scale nonlinear systems. IEEE Trans. on Automatic Control 48(2), 276–281 (2003)
261. Yi, B.-J., Kim, W.: The kinematics for redundantly actuated omnidirectional mobile robots. Journal of Robotic Systems 19(6), 255–267 (2002)
262. Young, K.D., Utkin, V.I., Özgüner, Ü.: A control engineer's guide to sliding mode control. IEEE Trans. on Control Systems Technology 7(3), 328–342 (1999)
263. Yu, C., Fidan, B., Anderson, B.: Persistence acquisition and maintenance for autonomous formations. In: Proc. 2nd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 379–384 (December 2005)
264. Yu, C., Fidan, B., Anderson, B.: Principles to control autonomous formation merging. In: Proc. American Control Conference, pp. 762–768 (June 2006)
265. Zhai, S., Fidan, B.: Single view depth estimation based formation control of robotic swarms: Fundamental design and analysis. In: Mediterranean Conference on Control and Automation, Ajaccio, Corsica, France, pp. 1156–1161 (June 2008)
266. Zhai, S., Fidan, B., Öztürk, Ş.Ç., Gazi, V.: Single view depth estimation based formation control of robotic swarms: Obstacle avoidance, simulation, and practical issues. In: Mediterranean Conference on Control and Automation, Ajaccio, Corsica, France, pp. 1162–1167 (June 2008)

Index

adaptive step size 226, 227
advantages of multi-agent dynamic systems 5
advection-diffusion-reaction equation 8
agent loss 23
agent model in 1-D discrete swarm 177
agent neighborhood 197
agents with finite body size 41
aggregation 15, 30, 108, 114, 119, 134, 142, 263
aggregation potential 30
aggregation potential bound 114
agreement 21, 197, 199
agreement set 202
almost constant attraction 40, 45
artificial potential function 220
artificial potential functions 219
asynchronism 177, 181, 197, 204
asynchronous agreement 208
asynchronous convergence theorem 190
asynchronous operation 177, 181, 197, 204
asynchronous PSO 253
attraction gain 78
attraction step 221, 225
attraction vector 221
attraction/repulsion function 31, 178, 180, 193
automated highway systems 13
bacteria foraging optimization 233, 242
bacterial foraging 233, 238, 245
benchmark functions 271
BFOA 233, 242
bidirectional communication 214, 216, 267
bidirectional interaction 267
biological motor 234
boundary layer 127, 132, 148
bounded repulsion 35, 44, 48, 76
bounded sequence 201
bounded swarm dynamics 181
bounded time-delay 216
box condition 190

cell-to-cell attraction/repulsion 239, 242, 247
centralized parallel asynchronous PSO 255
centroid 32, 77
chattering phenomena 113, 148
chemotactic behavior of *E. coli* 233
chemotaxis 8, 235, 238, 242
cognitive component 251
cohesiveness of aggregating swarm 35
cohesiveness of foraging swarm 48
collision avoidance 221
comfortable distance 179
comfortable position 180
communication cycles 216
communication delay 179, 198, 256
connected agent 200
connectivity 266

consensus 21, 197, 216, 275
consensus protocols 22
consensus state 21
constriction factor 252, 270, 273
continuum framework 7, 9
continuum model 8
contraction maneuver 160
contraction mapping 191, 203
contradiction 188, 189
controlling natural flocks 14
convex hull 69
cost function 238
cyclic neighborhood topology 211

decentralized asynchronous PSO 255, 269, 271
decision making by *E. coli* 236
DeJongF4 function 271, 273
direct search 251
directed graph 199, 264
directed path 265
directed tree 200
discrete event model 180
discrete event systems 179, 199, 208, 216, 257
discrete-time agent model 219
discrete-time swarm model 177
distributed agreement 21, 197, 208, 216, 275
distributed consensus seeking 21
distributed decision making 213
distributed orientation alignment 213
distributed synchronization 213
distributed synchronization of coupled oscillators 24
double integrator agent model 75
Duffing's equation 102
dynamic communication topology 200, 208
dynamic neighborhood 199, 211, 259, 265, 269, 271

elimination and dispersal 237, 238, 241, 242
equilibrium distance 32
equilibrium point 184
equivalent value 112
error feedback controller 155, 156, 167
expansion maneuver 160, 168
explosion behavior 252
extended agreement set 207

Fiedler eigenvalue 217
finite body size 41
finite time convergence 36
fitness function 245
fixed communication topology 208, 210
fixed neighborhood 257, 270
fixed neighborhood topology 265
fixed point 184
flexible manufacturing systems 216
flocking 23
foraging behavior 238
foraging behavior of *E. coli* 233
foraging potential 46
foraging potential bound 115
foraging strategies 233
formation achievement 18
formation acquisition 18
formation constraints 65, 117, 139, 152, 223
formation control 18, 64, 108, 117, 121, 139, 144, 151, 152, 219, 223, 227
formation error 227
formation expansion 164
formation maintenance 19
formation maneuvers 20, 158, 164, 168
formation merging 23
formation potential bound 117
formation reconfiguration 20, 158, 161, 165, 168
formation rotation 165
formation splitting 23
formation stabilization 18
formation switching 20
Francis-Byrnes-Isidori equations 155
friend 155, 163
full information controller 155, 163
fully actuated agent model 107
fully connected neighborhood topology 259
fully connected topology 211, 224, 265, 270
function minimization 22, 233
function optimization 245

gathering 214
Gaussian resource profile 54, 61
generalized distance 207
genetic algorithm 245, 261
Gerschgorin's disks 187
Gerschgorin's theorem 187
global best 252
global neighborhood 258, 265
gradient climbing by *E. coli* 236
graph 199
graph Laplacians 217
Griewank function 271, 273

hard-limiting repulsion 42, 46
higher-level agent model 29
hill climbing 234
hill-climbing 246

immersed 157
immersion 157
individual-based framework 7, 9
inertia weight 252
information flow topology 199, 264
interacting particle systems 10
interaction topology 199, 264
internal model 153, 157
internal potential energy function 35
interplay between attraction and repulsion 32

jointly uniformly strongly connected 266

kinematic agent model 29
kinetic phase transition 11
Kuramoto model 24

LaSalle's Invariance Principle 34, 98
learning coefficients 252, 273
linear attraction 35, 44, 48, 76
linearly bounded from below attraction 37
local best 252
local minima 48, 63, 65, 66, 121, 122, 229
local potential function 35

matrix spectrum 186
maximum norm contraction mapping 191
mean field approaches 10
mesh stability 13
metric 202
metric space 202
model uncertainty 107
momentum component 251
monotone sequence 201
motile behavior 235
multimodal Gaussian resource profile 57, 62, 116, 120, 138, 144

nearest neighbors in function space 261, 263
nearest neighbors in search space 260, 268
nearest neighbors rule 209, 260
neighborhood best 252
neighborhood definition strategies 258
neighborhood of an agent 197
neighborhood topology 264
neighborhood types 209
neutrally stable 151–153
Newton's iteration 219, 221, 223
Newton's method 219, 221
non-decreasing sequence 201
non-gradient 251
non-holonomic agent model 125
non-holonomic constraint 125
non-increasing sequence 201
non-reciprocal neighborhood 261, 262
non-special approach 7
nongradient optimization 238
nonlinear servomechanism 151
nutrient function 238, 245
nutrient hill climbing 234
nutrient hill-climbing 246
nutrient surface 238
Nyquist-like criterion 217

observer based controller 157
obstacle avoidance 221
obstacle function 221
obstacle set 221, 223, 225
omni-directional mobile robots 29
one dimensional swarm model 177
optimization 238
orientation alignment 213

output agreement 21
output regulation 151

pair dependent attraction/repulsion function 64
pair dependent potential function 64
parallel and distributed PSO 255
parallel asynchronous PSO 255
parallel synchronous PSO 253
parameter selection for BFOA 244
partial asynchronism 204
partially asynchronous 192
particle density 268
particle neighborhoods 257, 264
particle swarm optimization 251
patch models 10
path planner 219
perceived best 256
perceived maximum 206
perceived minimum 206
perceived state 205
persistency 23
plane resource profile 51, 60, 76, 116, 137
platooning 13
polynomial type potential functions 219
population density 8
potential function 30, 220
potential function for aggregations 30
potential functions 219
premature convergence 259
private area of an agent 42
properties of multi-agent dynamic systems 5

quadratic resource profile 52, 61, 116, 138

random neighborhood 209, 211, 258, 262, 263, 271
Rastrigin function 271, 274
reaching condition 109
reaching conditions 129
reaction-diffusion equations 10
reciprocal neighborhood 260–262
regulator equations 155
relative dynamics 34
relative sensing 214
rendezvous 24, 214, 216
reproduction 238, 241, 242
repulsion gain 78
repulsion step 225
repulsion vector 222
resource profile 17, 46, 115, 137
rigid graphs 225
rigidity 23
ring neighborhood topology 211, 258
Rosenbrock function 271, 274
rotation maneuver 160, 168

safety area of an agent 42
scalable 269
search by *E. coli* 235
self-driven particles 11
self-propelled particles 11
self-propelled particle systems 213
sensing by *E. coli* 236
sensing delay 179, 198
sensing errors 77
sequential asynchronous PSO 254
servocompensator 157
servomechanism 151
set of equilibrium points 33
set of neighbors of an agent 197
set of stationary points 33
single integrator agent model 29
sliding condition 109
sliding manifold 108, 112, 127, 128
sliding surface 128
social component 251
social foraging 17, 46, 76, 81, 108, 115, 120, 137, 144, 263
social networks 214, 275
social potential fields 12
spanning tree 200, 216, 267
special approach 7
spectral radius 187
spectrum of a matrix 186
stabilizing compensator 158
state agreement 21
static graph 264
static neighborhood 257, 270
static neighborhood topology 265
static topology 264
step size 221, 226
step vector 221
stochastic matrix 215

stopping criteria 269, 270, 273
string stability 13
strong connectivity 267
strongly connected 200, 265, 266, 268
sub-swarm 268
swarm center 32
swarm centroid 16, 32
swarm connectivity 266
swarm density 41, 43, 46, 268
swarm explosion 50, 53
swarming 242, 247
swarming of *E. coli* 239
swarms in biology 3
swarms in nature 275
swarm size 16, 35
swarm tracking 18, 19, 67, 111, 117, 122, 131, 139, 145
swarm tracking potential bound 117
swarm trajectory following 99
switching communication topology 200, 208
switching sequence 200
synchronism 181, 183, 200
synchronization 197, 213
synchronous convergence condition 190
synchronous operation 181, 183, 200
synchronous PSO 253

target function 219–221
target set 219, 223, 225
taxes 235
time-dependent communication topology 200, 208
time-dependent neighborhood 199, 259
time-varying neighborhood 265
time delay 177, 255, 269
time varying communication topology 200, 208
topology change 161, 165, 168
total asynchronism 181
tracking constraints 153
tridiagonal matrix 186
tumble 234

ultimate bound 101
ultimate bound on inter-agent trajectories 88
ultimate swarm size 35
unbounded repulsion 37, 40, 45
unicycle agent model 125
unidirectional communication 198, 200, 216, 256
unidirectional interaction 198, 200
uniform connectedness 216
uniform connectivity 204
uniformly bounded delay 204
uniformly cyclic topology 215
uniformly strongly connected 266
uniform swarm density 43
uniform ultimate boundedness 101
uniform ultimate boundedness of inter-agent trajectories 81
uniform update 204
union of communication graphs 216
union of set of graphs 200, 266

velocity damping gain 79
virtual leader 152

zero-error controller 155, 163
zero-error manifold 155, 163