BIFURCATIONS AND SYMMETRIES OF OPTIMAL SOLUTIONS FOR DISTRIBUTED ROBOTIC SYSTEMS

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Baoyang Deng,

Bill Goodwine, Director

Graduate Program in Aerospace and Mechanical Engineering

Notre Dame, Indiana

March 2011

BIFURCATIONS AND SYMMETRIES OF OPTIMAL SOLUTIONS FOR DISTRIBUTED ROBOTIC SYSTEMS

Abstract

by

Baoyang Deng

In this thesis, we consider the motion planning problem for a symmetric distributed system which consists of a group of autonomous mobile robots operating in a two-dimensional obstacle-free environment. Each robot has a predefined initial state and final state and the problem is to find the optimal path between two states for every robot. The path is optimized with respect to the control effort and the deviation from a desired formation. Due to scaling issues, it becomes more and more difficult and sometimes infeasible to numerically find solutions to the problem as the number of robots increases. One goal of this thesis is to exploit symmetries in distributed control systems to reduce the computational effort to determine solutions for optimal control of such systems. One way to characterize a distributed system is that it is a control system in which the state space is naturally decomposed into multiple subsystems, each of which typically only interacts with a limited subset of the other subsystems. A symmetric distributed system can be defined when the subsystems are diffeomorphically related. The optimal control problem for distributed systems may not scale well with the size of the overall system; hence, our efforts are directed toward exactly solving the optimization problem for large scale systems by working with a reduced order model that is determined by considering invariance properties with respect to certain group actions of the governing equations of the overall system.

Baoyang Deng

This thesis also studies bifurcations and multiple solutions of the optimal control problem for mobile robotic systems. While the existence of multiple local solutions to a nonlinear optimization problem is not unexpected, the nature of the solutions are such that a relatively rich and interesting structure is present, which potentially could be exploited for controls purposes. The bifurcation parameter is the relative weight given to penalizing the deviation from the desired formation versus control effort. Numerically it is shown that as this number varies, bifurcations of solutions are obtained. Theoretic results of this paper relate to the symmetric properties of these bifurcations and the number and existence of multiple solutions for large and small values of the bifurcation parameter. Understanding the existence and nature of multiple solutions for optimization problems of this type is also of practical importance due to the ubiquity of gradient-based optimization methods where the search method will typically converge to the nearest local optimum.

CONTENTS

FIGURES

ACKNOWLEDGMENTS

I would like to express my gratitude to those who made an important difference and played a vital role in the successful completion of my Doctoral studies at the University of Notre Dame. I am grateful to my advisor Professor Bill Goodwine whose help, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis. Professor Mihir Sen provided valuable advice and insight towards my work. I am very thankful for his opinion and guidance during the course of my thesis. I would like to thank Professor Panos Antsaklis, Professor John Renaud, Professor Mihir Sen and Professor Bill Goodwine for agreeing to be the members of my thesis committee and for taking the time to read my thesis and provide valuable suggestions.

I would like to thank my former and present colleagues, Brett McMickell, Neil Petroff, Dayu Lv, Jason Nightingale and Alice Nightingale for all their help, support, valuable hints.

I would like to give my special thanks to my husband whose patient love enabled me to complete this work. Thanks to my parents who gave me the love, education, skills, and constant support that enabled me to do this work.

CHAPTER 1

INTRODUCTION

Research on multi-robot systems originates in the late 1980s and increased substantially in the 1990s. Compared to a single-robot system, it may cost less in time and money to construct multiple simple robots and have those robots can work simultaneously to perform cooperative tasks. Also, the multi-robot system may be more robust and reliable than a single-robot system. Hence they have been the focus of increased research effort and attention in recent years. The application of distributed systems are everywhere: unmanned underwater vehicles [48], satellite clustering [36], electric power system [44], search and rescue operations [25] and so on.

Although each robot in a distributed system may possess a simple and tractable model and it may interact its neighbors in a very simple way, the resulting system often displays a rich and complex behavior when viewed as a whole. Since the robots in a distributed system work together to accomplish one task, coordination between robots are needed. Having multiple robots in a limited area may cause interference and collisions. Therefore, the control of a distributed system is more difficult than that of a single robotic system and it is a challenging topic in recent years. Especially when the number of the subsystems increases, the state space of the whole system becomes huge.

In this thesis, the problem addressed is to control a formation of robots moving along an optimal path between an initial configuration and a final configuration. The path is optimized with respect to a combination of the control effort and the deviation from a desired formation. Using standard methods from optimization, since each robot has its own predefined initial state and final state, the procedure to find the optimal path is to solve a boundary value problem for a set of second order ordinary differential equations. Since those equations are highly nonlinear, it is not feasible to determine closed form solutions. We develop a relaxation shooting method to solve them numerically. Theoretically, the method will work for a system with arbitrary large state space. But in fact, convergence issues prevent finding a solution when a system is too large. So one task here is to find a way that reduces the dimension of the state space of a large distributed system to a smaller one which is more manageable. The reduced state space should be "equivalent" to the larger state space in some sense. We will discuss the meaning of "equivalent" in later chapter. The idea behind this reduction is that a distributed system could have symmetric structure since the subsystems are identical to each other. We have the detailed definition of symmetric distributed system in this thesis. We exploit the symmetric properties of the distributed system to reduce it to a smaller one. Since the optimal control problem for distributed systems may not scale well with the size of the overall system, we work on this reduced order model to find the optimal trajectory for each robot in the reduced small system. Then, we can exactly solve the optimization problem for the original large scaled system by considering invariance properties with respect to certain group actions of the governing equations of the overall system reduced computational effort. Note that these results are general in that they will apply to any optimization problem for any type symmetric systems, not just for controls problems.

We defined the motion planning problem for a group of robots which result in solving a system nonlinear second order boundary value problems. The study for coordinated control of distributed systems has been developed for many years and overview of the literature appears in the next section. Also, the boundary value problem arises in a variety of different areas of applied mathematics and physics and the existence of nontrivial solutions has been paid much attention. These results are also outlined in the next section.

1.1 Motion Planning

The problem of motion planning is to deal with finding a feasible trajectory for a robotic system from a given initial configuration to a goal configuration, while satisfying some constraints.

Motion planning algorithms for single robot systems have been intensively discussed for years and the research on motion planning for mobile robots is vast. In the area of mobile robots, time optimal motion planning for a single car-like robot has been thoroughly studied. Dubins [14] proved the existence of shortest paths and provided a sufficient family of trajectories containing an optimal path to link any two configurations for a vehicle that can only move forward and is subject to curvature bounds. Paths in this family are at most three pieces of either arcs of circles with minimum radius or straight line segments. Reeds and Shepp [45] extended Dubins' results to a vehicle that can drive both forward and backward with a constant velocity. They proved that a shortest path in a free environment may always be one of 48 simple paths which consist of at most five pieces straight line segments or arcs of circle with minimum radius. In 1991, Sussmann and Tang [50] gave new proof of Dubins' results and Reeds and Shepp's results using Pontryagin's Maximunm Principle. They reduced the path family to 46 different paths and their result is an improvement of the Reeds and Shepp's. In 1994, Bui *et al.* [54] performed a complete optimal path synthesis for Dubins robots.

Although most mobile robotic systems involving a single robot can operate alone in its environment, many researchers have considered the problems and potential advantages involved in having an environment inhabited by a team of robots which cooperate in order to complete some required task. For some specific tasks such as search and rescue operations [25], cleaning up toxic waste [34] and pushing boxes [35], it would be more effective to send a number of smaller and simple robots to perform the task than sending one very complex and expensive robot. Since each individual robot is simple and cheap, using multiple robots can have several advantages such as the resulting system can be more economical and scalable and less susceptible to overall failure than a system with one robot. In order to complete one whole task, the robot must communicate with others and the coordination of the robots is very important. However, as the number of robots and degrees of freedom of the system become large, the control of the system becomes difficult which has been the focus of much research interest in recent years. On the other hand, broad applications in multiple robots systems call for practical and efficient motion planning strategies.

The multi-robot motion planning algorithms can be roughly grouped into two categories: centralized motion planning [18] and decentralized motion planning [24] according to the information handling structure among robots.

1.1.1 Centralized Planners

In a multi-agent system, a central planner designs the motion plan for all robots based on full knowledge about the environment. This approach fits better purely computational problems rather than tasks that rely on real-time feedback control. The obvious advantage is its conceptual simplicity. It allows the possibility of global optimization. The price for this convenience is the computationally intensity due to high dimensional configuration spaces. Most of the literature on centralized approach concentrates on decreasing the computational cost. This is typically achieved at the expense of completeness. In Kant and Zucker [26] the task is divided into two subtasks. Each robot's path is determined taking into account only stationary obstacles. With the paths fixed, velocities of all the robots are then adjusted avoiding collisions. Erdmann and Lozano-Perez [18] explore the motion planning problem for multiple moving robots. The approach assigns priorities to robots in advance, then plans motions one robot at a time. For each moving robot, the planner constructs a configuration space-time that represents the time varying constraints imposed on the moving robot by the other moving and stationary robots. This approach was demonstrated in two domains: one domain consisting of translating planar objects and the other one consisting of two link planar articulated arms.

One merit of the centralized planners is that they allow the possibility of completeness and global optimization. A drawback of most centralized planners is that they are computationally intensive due to high dimensional configuration spaces. This leads to search high dimensional configuration spaces quickly at the cost of losing optimality.

1.1.2 Decentralized Planners

Decentralized control has two obvious advantages. In principle, computational complexity of a decentralized system can be made independent of the number of agents in it, and it may be more stable and robust. Also a failure of one or few agents does not necessarily affect the whole system. On the negative side the decentralized control is less likely to deliver optimal performance since it might only use local information. Hence, many decentralized algorithms exist that search for near-optimal solutions.

In decentralized methods, each robot plans individually for itself by means of collecting information from other robots and environmental information around the robot. Belta and Kumar [6] propose a modern geometric approach to design trajectories for teams of robots. First, they consider the problem of generating minimum kinetic energy motion for a rigid body in a 3D environment. Then, they illustrate a procedure of optimal motion planning for groups of robots required to maintain a rigid formation. The overall procedure is invariant with respect to both the local coordinates and the choice of the inertial frame.

Guo and Parker [24] use altering velocity to produce a distributed planner that tries to optimize trajectories. First, each robot plans its own trajectory independently. Then a coordination diagram is constructed based on collision checks among all robot paths. This scheme was demonstrated both in simulations and on physical Nomad robots.

One of the fundamental problems in the coordinated control of distributed systems is consensus seeking among agents. In order for agents to coordinate their behaviors, they have to use some shared knowledge about variables such as position, speed etc. This shared information is a necessary condition for cooperation in multi-agent systems. Several consensus protocols have been proposed in the literature. Most consensus protocols operate in a synchronized fashion [21], and each agents's decisions must be synchronized to a common clock shared by all other agents in the group. This might not be natural in certain context, and Lei Fang, Panos Antsaklis [19, 20] proposed an asynchronous consensus protocol, where each agent updates on its own pace, and uses the most recently received information from other agents. It encompasses those synchronous ones with various communication patterns.

1.2 Formation Control

The formation problem in multi-robot systems is defined as the coordination of a team of robots to get into and maintain a formation with a certain shape. Current applications of formation control include unknown environments exploration [2], search and rescue operations [25], traffic control [1, 53], satellite clustering [36] and holding and transporting objects [11, 56].

Formation control is an important issues in coordinated control for a collection of robots. Many control approaches have been used to solve the problems in formation control, for example, leader-follower method [9, 10, 12, 13, 15, 28], behavior-based method [2, 3, 32, 33, 49], virtual structure method [6, 29, 51, 55] and so on.

1.2.1 Leader-following Method

In the leader-follower method, each robot has at least one designated leader. Leaders can be some robots in the group or virtual robots that represent pre-computed trajectory supplied by a higher level planner. The other robots are followers that try to maintain a specified relative configuration to their leaders. This method can control the team of robots behaviors if the leader's behavior is given. However there is no obvious feedback from the followers to their leaders, so the whole system is more susceptible to overall failure.

Desai, *et al.* [9, 10, 13] propose a graph theoretic framework for the control of a team of robots moving in an terrain with obstacles while maintaining a desired formation. The behaviors of robots in the formation are defined by using a control graph. This framework can handle transition between formations, *i.e.*, between control graphs, and they define the transition matrix to model transition from one control graph to another. It requires to enumerate and classify control graphs given in order to prove the mathematical results. When the number of robots increases, the computations for control graphs will increase. But these computations are decentralized, which allows the methods to be scalable to large number of robots.

In another paper, Desai, *et al.* [12] investigate feedback laws to control multiple robots moving in a formation and propose a method for controlling formations that uses only local sensor-based information. They assume that each robot has the ability to measure the relative position of other robots that are immediately adjacent to it. Once the motion for the lead robot is given, the remainder of the formation is governed by local control laws based on the relative dynamics of each of follower robots and the relative positions of the robots in the formation. These control laws can provide easily computable, real-time feedback control with provable performance for the entire system, and can be extended to control arbitrarily large numbers of robots moving in a formation. This paper proves that the zero dynamics of the system are asymptotically stable by using feedback linearizion to exponentially stabilize the relative distance and orientation of the followers. They demonstrate their result by applying it to simulate six robots moving around an obstacle.

Egerstedt and Hu [15] propose a model-independent coordination strategy for moving a group of robots in a desired formation over a given path. In the paper, the leader robot is a given, nonphysical point. The paths for the real robots are defined by a formation constraint in combination with the desired reference path for the virtual leader, which is specified by the planners. They applied the method to rigid body constrained motions. The paper shows that if the real robots track their reference points perfectly, or the tracking error of the robots are bounded, their method can stabilize the formation error.

Barfoot and Clark's paper [4] is similar to the leader-following work in that they use a reference trajectory and define the motion of each individual robot relative to this trajectory. But they did not use any particular feedback control to enable each robot to actually track its planning trajectory. The robot in the formation can not sense the locations of other robots but can sense its own location relative to a common global reference frame. They allow the distance between robots to change when the formation turns. The paper shows that a formation can be treated in the same way as a single robot is treated, which makes a great deal of single robot work relevant at the formation level. The method allows the geometry of a formation to be considered separately from the formation's overall trajectory by providing a way to combine these two components It was validated on the Stanford Micro-Autonomous RoverS (MARS) testbed.

In leader-following method, there is a explicit dependence of the motion of followers on their leaders, but the leaders motion is independent of their followers. This may cause some problem. For example, if a robot fails or slows down, the motion of the robots that are following it will be directly affected by its behavior, while its leaders are not affected and continue their tasks. This method is modified by Pereira, *et al.* [42], where a cooperative leader-follower method is introduced as a modification of the standard leader-following approach. In the new method, the motion of a robot is dependent on both its leaders and its followers. Thus, the system is more robust to failures.

1.2.2 Behavior-based Methods

Behavior-based methods draw inspiration from biology. In nature, each animal in a herd benefits by minimizing its encounters with predators. By grouping, they combine their sensors to maximize the chance to detecting predators or to more efficiently forage for food. Robotic researchers have developed formation behaviors for simulated robots inspired by animals behaviors. This approach can derive control strategies easily, but it can not define the whole team's behaviors, obviously. Therefore it is difficult to analyze the behaviors mathematically.

In a behavioral-based method, the behavior of each robot is prescribed and the final control is derived by weighting the relative importance of each behavior. A behavior-based architecture is exploited in [2] for multi-robot systems. In this study, each robot computes its proper position in the formation based on the locations of the other robots. Each robot is to simultaneously move to a goal position, avoid colliding with other robots and obstacles, and maintain a formation. In the paper, the authors present reactive behaviors for four formations (line, column, diamond and wedge) and three formation reference (unit-center-referenced, leader-referenced and neighbor-referenced). The behaviors were validated both in the laboratory on mobile robots and outdoors on non-holonomic 4-wheel-drive High Mobility Multipurpose Wheeled Vehicles (HMMWVs).

Balch and Hybinette [3] introduce a behavior-based approach to robot formation problem, which provides scalability, locality and flexibility to the system. This idea is inspired from the way molecules form crystals. In the formation, each robot has several local attachment sites that other robots may be attracted to. This type of attachment site geometry is similar to molecular covalent bonding. The robot formation shapes are influenced by the attachment site geometries used. This approach is scalable to large robot groups because global communication of robot position is not used and each robot only relies on the locations of nearby robots.

Mataric [32] proposes a bottom-up methodology that produced the desired system behavior as a result of the interaction dynamics between the robots and their environment and the biases and constraints introduced by the system designer. This approach is more flexible and robust than the top-down methodology. The paper [33] presents that the use of behaviors as the underlying control representation provides a useful encoding that both lends robustness to control and allows abstraction for handling scaling in learning, of key importance to robot systems.

Su and Lu [49] improve the behavior-based method by combining it with formation feedback. They design the main behaviors of the leader robot to avoid obstacles and move to a goal point. The leader plans its path according to the current environment, and then sends its formation through network to each follower robots. The follower generates its own behavior based on the information given by the leader. The current information, such as the positions of the robots and the shape of the formation is given to the leader as feedback. The leader plans its behavior according to this feedback.

1.2.3 Virtual Structure Methods

The virtual structure method involves the maintenance of a geometric configuration during robot movement using the idea that points in space should maintain fixed geometric relationship. If robots behaved in this way, they would be moving inside a virtual structure. The concept of virtual structure in the framework of cooperative robotics is introduced in [51]. Tan and Lewis [29, 51] develop a control method to force an ensemble of robots to behave as if they were particles embedded in a rigid body structure. Their method has many merits: it is capable of high-precision control, inherently fault tolerant, no leader election are required and it is reconfigurable for different kinds of virtual structures. This method had been tested both using simulation and experimentation with a group of three robots. Although it was only tested with robots moving on a plane, there is no limitation to three dimensional space.

Yamaguchi, *et al.* [55] develop a distributed control scheme of a team of robotic vehicles that guaranteed stability and controllability using only relative position feedback. Each robots in this scheme has its own coordinate system and it can sense its relative position and orientation to others. There is no supervisor and each robot moves based on feedback from itself and its neighbors. The robots interact with each other directly or indirectly through others by the relative position feedback. This scheme is validated by computer simulations.

The formation control used in this thesis is most similar to the virtual structure method in that we try to maintain the rigid body formation during the robots moving. But as the geometrical distances between robots vary slightly with time, our approach is flexible rather than rigid.

1.3 Multiple Solutions of Second Order Ordinary Differential Equations

The existence of multiple solutions to boundary value problems is a common feature of the types of problems considered in this thesis. This section reviews the literature related to multiple solutions of boundary value problems. Consider the equations of the form

$$u'' + a(t) f(u) = 0, \qquad 0 \le t \le 1, \qquad u(0) = u(1) = 0.$$

Define

$$f_0 = \lim_{u \to +0} \frac{f(u)}{u}, \qquad f_\infty = \lim_{u \to +\infty} \frac{f(u)}{u},$$

then, the properties of the solutions depend on the limiting behavior of the function $f(u)$. And, the fixed-point theorem of cone expansion/compression [23] is broadly used in this area. The theorem is stated as follows.

THEOREM 1.3.1 *Let E be a Banach space, and let $K \subset E$ be a cone in E. Assume Ω_1,*

Ω_2 *are open subsets of* E *with* $0 \in \Omega_1$, $\overline{\Omega}_1 \subset \Omega_2$, *and let*

$$A : K \cap \left(\overline{\Omega}_2 \setminus \Omega_1\right) \to K$$

be a completely continuous operator such that either

(i) $\| Au \| \ \leq \ \| u \|$, $u \in K \cap \partial\Omega_1$, *and* $\| Au \| \ \geq \ \| u \|$, $u \in K \cap \partial\Omega_2$; *or*

(ii) $\| Au \| \ \geq \ \| u \|$, $u \in K \cap \partial\Omega_1$, *and* $\| Au \| \ \leq \ \| u \|$, $u \in K \cap \partial\Omega_2$.

Then A has a fixed point in $K \cap \left(\overline{\Omega}_2 \setminus \Omega_1\right)$.

The proof of the fixed point theorem can be found in [23]. And the following application of the theorem can also be found in [23].

Example 1.3.2: Consider the two-point boundary value problem of an ordinary differential equation:

$$x'' + f(x) = 0, \qquad 0 \leq t \leq 1, \qquad x(0) = x(1) = 0,$$

where $f(x)$ is continuous and nonnegative for $x \geq 0$ and $f(x) = 0$. The above equation has at least one nontrivial solution $x(t) \in C^2[0,1]$ if $f_0 = 0, f_\infty = \infty$ or $f_0 = \infty, f_\infty = 0$.
Proof: Now, we apply Theorem 1.3.1 to prove it. Obviously $x(t) = 0$ is the trivial solution of the problem.
The Green's function of the differential operation $-x''$ with $x(0) = x(1) = 0$ is

$$G(t,s) = \begin{cases} t(1-s), & t \leq s \\ s(1-t), & t > s \end{cases}.$$

Let $Ax(t) = \int_0^1 G(t,s) f(x(s)) ds$, and P_ε be the cone in $E = C[0,1]$ given by

$P_\varepsilon = \{x(t) \in C[0,1] : x(t) > 0, \min_{\frac{1}{2}-\varepsilon \le t \le \frac{1}{2}+\varepsilon} x(t) \ge (\frac{1}{2} - \varepsilon) \parallel x \parallel\}$ where $0 \le \varepsilon \le \frac{1}{2}$,

$\parallel x \parallel = \sup_{[0,1]} |x|$.

When $\frac{1}{2} - \varepsilon \le t \le \frac{1}{2} + \varepsilon$

$$G(t,s) = \begin{cases} t(1-s) \ge \left(\frac{1}{2} - \varepsilon\right)(1-s), & t \le s \\ s(1-t) \ge s\left(\frac{1}{2} - \varepsilon\right), & t > s \end{cases}.$$

Therefore, $G(t,s) \ge \left(\frac{1}{2} - \varepsilon\right) s(1-s) = \left(\frac{1}{2} - \varepsilon\right) G(s,s), \forall \frac{1}{2} - \varepsilon \quad \le t \le \frac{1}{2} + \varepsilon$ and $0 \le s \le 1$.

Then

$$\begin{aligned} \min_{\frac{1}{2}-\varepsilon \le t \le \frac{1}{2}+\varepsilon} Ax &= \min_{\frac{1}{2}-\varepsilon \le t \le \frac{1}{2}+\varepsilon} \int_0^1 G(t,s) f(x(s)ds \\ &\ge \left(\frac{1}{2} - \varepsilon\right) \int_0^1 G(s,s) f(x(s)ds \\ &\ge \left(\frac{1}{2} - \varepsilon\right) \int_0^1 G(t,s) f(x(s)ds \\ &\ge \left(\frac{1}{2} - \varepsilon\right) \parallel Ax.) \parallel \end{aligned}$$

So $Ax \in P_\varepsilon$, and $A(P_\varepsilon) \subset P_\varepsilon, \forall \quad 0 < \varepsilon < \frac{1}{2}$.

Since $f_0 = 0$, we can choose $r_1 > 0$, such that $|f_0| \le \eta$, for $|x| < r_1$, where $\eta > 0$ satisfies $\eta \int_0^1 G(s,s)ds \le 1$

$$\begin{aligned} Ax &= \int_0^1 G(t,s) f(x(s))ds \\ &\le \int_0^1 G(t,s)|f(x(s))|ds \\ &\le \eta \parallel x \parallel \int_0^1 G(t,s)ds \\ &\le \parallel x \parallel, \end{aligned}$$

which means $\| Ax \| \leq \| x \|$, when $x \in P_\varepsilon \cap \partial\Omega_1$, where $\Omega_1 := \{x \in E : \| x \| < r_1\}$.

For $f_\infty = \infty$, choose $\hat{r}_2$, such that $|f_\infty| \geq \mu$, for $|x| \geq \hat{r}_2$, where $\mu > 0$ satisfies $\left(\frac{1}{2} - \varepsilon\right) \mu \int_{\frac{1}{2}-\varepsilon}^{\frac{1}{2}+\varepsilon} G(\frac{1}{2}, s) ds \geq 1$.

Let $r_2 = \max\{2r_1, \frac{\hat{r}_2}{1/2-\varepsilon}\}$ and $\Omega_2 := \{x : |x| < r_2\}$. For $x \in P_\varepsilon \cap \partial\Omega_2$,

$$\min_{freq12-\varepsilon \leq t \leq \frac{1}{2}+\varepsilon} x(t) \geq \left(\frac{1}{2} - \varepsilon\right) \| x \| = \left(\frac{1}{2} - \varepsilon\right) r_2 \geq \hat{r}_2.$$

$$\begin{aligned} Ax\left(\frac{1}{2}\right) &= \int_0^1 G\left(\frac{1}{2}, s\right) f(x(s)) ds \\ &\geq \mu \int_0^1 G\left(\frac{1}{2}, s\right) x(s) ds \\ &\geq \mu \left(\frac{1}{2} - \varepsilon\right) \| x \| \int_0^1 G\left(\frac{1}{2}, s\right) ds \\ &\geq \| x \| . \end{aligned}$$

So $\| Ax \| \geq \| x \|$, when $x \in P_\varepsilon \cap \partial\Omega_2$, where $\Omega_2 := \{x \in E : \| x \| < r_2\}$.

The proof for sublinear case is similar to the suplinear case as stated. ■

Erbe and Wang [16] studied the existence of positive solutions of the equation with linear boundary conditions. They showed the existence of at least one positive solution in two cases, superlinearity ($f_0 = 0$, $f_\infty = \infty$) or sublinearity ($f_0 = \infty$, $f_\infty = 0$) by application of the fixed point theorem. A simple superlinearity exmaple is $f(s) = s^2$ and a simple sublinearity example is $f(s) = s^{1/2}$. In [31], Ma and Thompson extended the function f to be a continuous function satisfying $sf(s) > 0$. And proved that the problem had two solutions for superlinearity or sublinearity. And further, in [17], Erbe, Hu and Wang showed that there were at least two positive solutions in the case of superlinearity at one end (zero or infinity) and sublinearity at the other end.

Naito and Tanaka [40] and Ma and Thompson [30] established precise condition concerning the behavior of the ratio $f(s)/s$ for the existence and nonexistence of solutions. Their main results were that the boundary value problem had at least k solutions if the ratio $f(s)/s$ crossed the k eigenvalues of the associated eigenvalue problem.

1.4 Organization

In this thesis, we focus on solving the motion planning problem for a group of robots with the goal of optimizing a suitable cost function. We assume those robots move in a two dimensional obstacle-free environment. We encounter boundary value problems when solving equations of motion of the system, where there exists more than one solution for some cases. The main contributions of this thesis are finding the symmetric properties of the results, asymptotically analyzing the results for some special cases, numerically analyzing the bifurcation phenomena in multiple solutions and proving symmetry of bifurcations. The remainder of this thesis is organized as follows.

In Chapter 2, a brief mathematical background of differential geometry, group theory, graph theory and symmetric distributed systems is presented. Chapter 3 defines the optimal problem considered in this thesis and presents the properties of the cost function related to this problem and the algorithm to solve the problem. It also gives the results illustrating the properties of the optimal problem in a distributed system. Chapter 4 first presents numerical results illustrating bifurcations and multiple solutions of the boundary value problem associated with the optimal control problem. Then, it presents a theoretical result relating to the existence of multiple solutions in the limiting cases of small and large values of the bifurcation parameter. Finally, it proves the existence of symmetric solutions which guarantees that for any solution, a corresponding symmetric solution exists. The practical benefit of this result is that if a solution is found numerically, the symmetric solution can be computed from that algebraically. Finally, Chapter 5 presents conclusions and provides an outline of future work in this area.

CHAPTER 2

MATHEMATICAL BACKGROUND

This chapter provides an introduction to the mathematical terms used throughout this thesis. In section 2.1, we introduce some basic concepts from differential geometry, which is a wide area for study. Those who are interested in this area are referred to [7, 41, 47]. Some useful definitions of group theory and graph theory are presented in section 2.2 and section 2.3 respectively.

2.1 Differential Geometry

A manifold is the fundamental object in differential geometry. Roughly speaking, it is a set of points that locally “looks like” an open subset of Euclidean space. The formal definition of manifold is more challenging than the previous description and it is defined as follows.

Definition 2.1.1: [47] A m-dimensional manifold is a set M together with coordinate charts $(\mathscr{U}_a, \phi_a)$ with the property that $M = \cup_a \mathscr{U}_a$ and that, whenever $\mathscr{U}_a \cap \mathscr{U}_b \neq \emptyset$, we have

1. the overlap map $\phi_{ab} := \pi_b \circ \phi_a^{-1} | \phi_a(\mathscr{U}_a \cap \mathscr{U}_b)$ is a diffeomorphism from $\phi_a(\mathscr{U}_a \cap \mathscr{U}_b)$ to $\phi_b(\mathscr{U}_a \cap \mathscr{U}_b)$.
2. if $x \in U_a$, $y \in U_b$ are distinct points of M, then there exist open subset $W \subset R^m$, $V \subset R^m$, with $\phi_a(x) \in W$, $\phi_b(y) \in V$, satisfying $\phi_a^{-1}(W) \cup \phi_b^{-1}(V) = \emptyset$.

■

Example 2.1.2: [47] An *n*-sphere is a manifold. The unit sphere

$$S^n = \{x \in \mathbb{R}^{n+1} | \quad \|x\|_{\mathbb{R}^{n+1}} = 1\},$$

is a n-dimensional manifold realized as a surface in $\mathbb{R}^{n+1}$. Let U_1, U_2 be the subsets obtained by deleting the north and south poles respectively. Let

$$\phi_i : U_i \to \mathbb{R}^n, \qquad i = 1, 2,$$

be projections from the respective poles, so

$$\phi_1(x) = \left(\frac{x_1}{1 - x_{n+1}}, \frac{x_2}{1 - x_{n+1}}, \cdots \frac{x_n}{1 - x_{n+1}} \right),$$
$$\phi_2(x) = \left(\frac{x_1}{1 + x_{n+1}}, \frac{x_2}{1 + x_{n+1}}, \cdots \frac{x_n}{1 + x_{n+1}}, \right).$$

It can be easily checked that on the overlap $U_1 \cup U_2$,

$$\phi_{12} = \phi_1 \circ \phi_2^{-1} : \mathbb{R}^n \backslash \{0\} \to \mathbb{R}^n \backslash \{0\}$$

is a smooth diffeomorphism, given by the inversion

$$\phi_{12}(x) = \left(\frac{x_1}{1 - x_{n+1}^2}, \frac{x_2}{1 - x_{n+1}^2}, \cdots \frac{x_n}{1 - x_{n+1}^2} \right).$$

■

In this thesis, we always consider that the configuration of an object is in a manifold, so the trajectory of the object is a curve on the manifold. The velocity, which lies in the tangent space of the manifold, at a given time is a tangent vector to the curve evaluated at the same time. Now, we present the definitions of curve, tangent vector, tangent space and tangent bundle mathematically.

Definition 2.1.3: *A curve* on a n-dimensional manifold M is a map $c : I \rightarrow M$, where I is a subinterval of $\mathbb{R}$. At each point $x = c(t)$, the curve has *a tangent vector* $X = c'(t) = (\dot{x}_1, \dot{x}_2, \cdots, \dot{x}_n)$. The collection of tangent vectors to all possible curves passing through a given point p on M is a n-dimensional *tangent space*, denoted T_pM. The collection of tangent spaces for all points on M is the *tangent bundle* of M, denoted TM. ■

Definition 2.1.4: [47] Assume M is a manifold, *a vector field* X on M is a map $X : M \rightarrow TM$, such that $\pi \circ X = id^M$, *i.e.*, $\forall p \in M$, $X(p) \in T_pM$. Here π is the natural projection from TM onto M. *A smooth vector field* is a smooth map $X : M \rightarrow TM$, such that $\pi \circ X = id^M$. ■

Definition 2.1.5: [47] If V is a finite-dimensional $\mathbb{R}$-vector space, *the dual space* to V is the set $V^* = L(V;\mathbb{R})$ of linear maps from V to $\mathbb{R}$. If $(e_1, e_2, \cdots e_n)$ is a basis for V, then a basis for V^* can be denoted $(e^1, e^2, \cdots e^n)$, defined by $e^i(e_j) = \delta^i_j$. ■

2.2 Group Theory

In this section, we present some definitions and notations in group theory and they are from[41, 46].

Definition 2.2.1: A *group* G is a set together with a binary operation $(\cdot) : G \times G \mapsto G$, such that the following properties are satisfied:

1. Associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, for all $a, b, c \in G$.
2. Identity: $\exists$ an identity e such that $e \cdot a = a \cdot e = a$, for every element $a \in G$.
3. Inverse: For all $a \in G$, there exists an inverse a^{-1}, such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

$\blacksquare$

Example 2.2.2: The set of all invertible $n \times n$ matrices $GL(\mathbb{R}, n)$ with matrix multiplication is a group. $\blacksquare$

Definition 2.2.3: Let G be a group and S be a set. A *left action* of G on M is a map $\phi : G \times M \to M$ satisfying

- $\phi(e, x) = x$, where e is the identity element of G, for all $x \in M$.
- $\phi(g, \phi(h, x)) = \phi(g \cdot h, x)$, for all $g, h \in G$, $x \in M$.

Similarly, a *right action* of G on S is a map $\phi : S \times G \to M$ satisfying

- $\phi(x, e) = x$, where e is the identity element of G, for all $x \in M$.
- $\phi(\phi(x, g), h) = \phi(x, g \cdot h)$, for all $g, h \in G$, $x \in M$.

G is called a transformation group, and ϕ is called the left (resp. right) group action. $\blacksquare$

Definition 2.2.4: Let ϕ be a smooth group action of G on M, a function $f : M \to \mathbb{R}$ is *invariant under group action* ϕ if, for all $g \in G$, $\phi(f,g) = f$, if ϕ is a left action or $\phi(g,f) = f$, if ϕ is a right action. ■

Definition 2.2.5: A *permutation* of a set $S = \{1,2,\cdots,n\}$ is a one-to-one mapping of S into itself and is usually written as

$$P = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ i_1 & i_2 & i_3 & \cdots & i_n \end{pmatrix}.$$

which indicates that $1 \mapsto i_1$, $2 \mapsto i_2$, $\cdots$, $n \mapsto i_n$. A *permutation group* is a finite group G whose elements are permutations of a given set S and whose operation is composition of permutations in G. ■

The dihedral group D_n is an example of permutation groups. It is the symmetry group of an n-sided regular polygon centered about the origin in the plane for $n > 1$. The group order of D_n is $2n$. Dihedral groups D_n are non-Abelian permutation groups for $n > 2$. A reducible two-dimensional representation of dihedral group using real matrices has generators given by f and r, where f is a rotation by π radians about an axis passing through the center of a regular n-gon and one of its vertices and r is a rotation by $2\pi/n$ about the center of the n-gon, i.*e*.,

$$f = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad r = \begin{pmatrix} \cos\frac{2\pi}{n} & -\sin\frac{2\pi}{n} \\ \sin\frac{2\pi}{n} & \cos\frac{2\pi}{n} \end{pmatrix}.$$

Then, $D_n = \{e, r, r^2, \cdots, r^{n-1}, f, fr, fr^2, \cdots, fr^{n-1}\}$.

2.3 Graph Theory

Graph theory is a powerful tool to represent the structure of the distributed systems. In this section, we provide some definitions and notations in graph theory that allow us to identify symmetries in distributed systems. For more information about graph theory, the reader is referred to [22, 52].

Definition 2.3.1: A *graph* G consists of

1. A set V whose members are called vertices (also called "nodes" or simply "points").
2. A set E whose members are called edges.
3. A function (the endpoint function) which assigns to each edge e in E an unordered pair of vertices called the endpoints of e, *i.e.*, $f : E \to \{(V_i, V_j) | V_i, V_j \in V\}$.

We may write $G = (V, E, f)$ to represent graph G with the vertex set V, edge set E and the endpoint function f. ■

An edge is said to *connect* its two endpoints, and it is *incident* on each of its endpoints. A *loop* is an edge which joins a vertex to itself. Two edges which connect the same pair of endpoints are called *multiple edges* or *parallel edges*. A graph with no loops and no multiple edges is called a *simple graph*. Two vertices are *adjacent* if there is an edge connecting them. If both the vertex set and the edge set are finite, then the graph G is said to be *finite*. A *directed graph* or *digraph* is one where each edge has a specified direction. For a directed graph the edge-endpoint function assigns to each edge an ordered pair of vertices. The first member of the pair is called the initial vertex (or start) and the second is called the terminal vertex (or finish). Unless otherwise specified, the graphs discussed here are directed graph and are implicitly considered to be simple and finite.

2.4 Symmetric Distributed Systems

We adopt the definition of symmetric distributed system in [37, 38]. This definition is defined for a system without drift, then extended to the drift system. Consider a driftless system of the form

$$\Sigma : \qquad \dot{x} = \sum_{i=1}^{n} g_i(x) u_i. \tag{2.1}$$

We can use a graph to represent the system Σ. In the graph, each node represents a subsystem and the lines between nodes are interactions between subsystems. If the system Σ is symmetric, then there exists a graph symmetry of Σ, which defined as follows.

Definition 2.4.1: Let $G_\Sigma = (V, E, f)$ be the graph of a distributed system Σ given by Equation 2.1. A *graph symmetry* or *automorphism* of G_Σ is a permutation, σ, on the set of vertices V given by,

$$\sigma(V_i) = V_j, \qquad V_i, V_j \in V$$

that preserves adjacency:

$$\forall V_i, V_j \in V : (V_i, V_j) \in E \Leftrightarrow (\sigma(V_i), \sigma(V_j)) \in E$$

The group of all such permutations together with composition is called the automorphism group $\mathrm{Aut}(G)$ of the graph. ■

Since σ is a permutation of the vertices, we need to define the permutation on the states induced from σ. Let $\alpha(V_1, \ldots, V_n) = [x_1, \ldots, x_n]^T$ be the canonical map $\alpha : \mathbf{V} \mapsto M$. The action of a graph automorphism naturally induces a permutation, $\sigma_\sharp : M \mapsto M$, of states given by,

$$\sigma_\sharp(x) = \alpha(\sigma(\mathbf{V})) = \alpha \begin{pmatrix} V_1 & V_2, & \ldots, & V_n \\ V_{i_1} & V_{i_2}, & \ldots, & V_{i_n} \end{pmatrix} = \begin{pmatrix} x_1, & x_2, & \ldots, & x_n \\ x_{i_1}, & x_{i_2}, & \ldots, & x_{i_n} \end{pmatrix},$$

where $V_{i_1}, \ldots, V_{i_n}$ and $x_{i_1}, \ldots, x_{i_n}$ are rearrangements of the vertex set and state vector, respectively. Note, the elements x_i may be vectors if $\dim(M_i) > 1$. Vector fields are mapped under the induced permutation $\sigma_\sharp$ using the usual push-forward,

$$(\sigma_\sharp)_* g_i^j = T\sigma_\sharp \circ g_i^j \circ \sigma_\sharp^{-1}(x).$$

The induced permutation defines an equivalence relation between vector fields.

Definition 2.4.2: Two vector fields, $g_i(x)$ and $g_j(x)$ are *equivalent*, denoted $g_i \sim g_j$, if there exists a $\sigma \in \mathrm{Aut}(G)$ such that the corresponding induced permutation, $\sigma_\sharp$ is such that,

$$g_i(x) = (\sigma_\sharp)_* g_j(x). \tag{2.2}$$

■

Given an equivalence relation among vector fields, a symmetric nonlinear distributed system is defined as follows.

Definition 2.4.3: Let $G_\Sigma = (V, E, f)$ be the graph of the distributed system Σ given by Equation 2.1. The system, Σ is symmetric if there exists a graph symmetry,

$\sigma \in \mathrm{Aut}(G_\Sigma)$ other than the identity, such that if $\sigma(V_i) = V_j$, then

$$g_i \sim g_j.$$

■

The following example is to illustrate theses concepts.

Example 2.4.4: [37] Consider a four component system that can be represented by a graph shown in Figure 2.1 and described by the following equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \sin x_1 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix} u_1 + \begin{bmatrix} x_1 \\ \cos x_2 \\ x_2 + 1 \\ x_2 x_4 \end{bmatrix} u_2 + \begin{bmatrix} x_1 \\ x_3 x_2 \\ \cos x_3 \\ x_3 + 1 \end{bmatrix} u_3 + \begin{bmatrix} x_1 \\ x_4 + 1 \\ x_4 x_3 \\ \cos x_4 \end{bmatrix} u_4,$$

Choose a graph symmetry σ, which corresponds to a cyclic counter-clockwise permutation of the outer components of the graph shown in Figure 2.2. Then,

$$\sigma(V_1, V_2, V_3, V_4) = (V_1, V_4, V_2, V_3),$$

which creates an induced permutation given by

$$\sigma_\sharp(x_1, x_2, x_3, x_4) = (x_1, x_4, x_2, x_3)$$

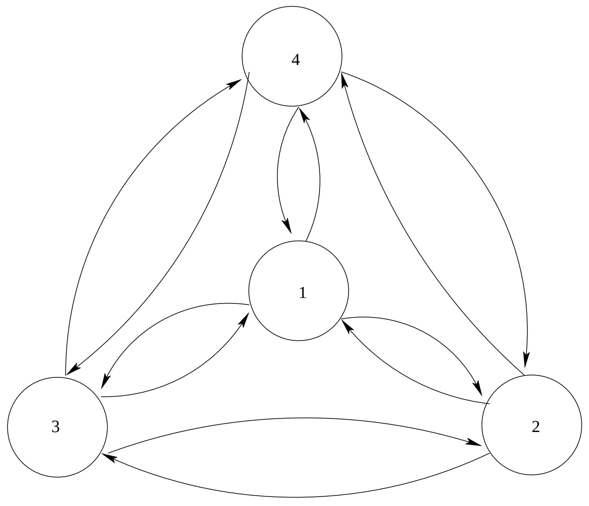

Figure 2.1. Graph of the four component system given in Example 3.4.4.

Consider the vector fields g_2 and g_3 in the given system,

$$\begin{aligned}
(\sigma_\sharp)_* g_2(x_1,x_2,x_3,x_4) &= T\sigma_\sharp \circ g_2 \circ \sigma_\sharp^{-1}(x_1,x_2,x_3,x_4) \\
&= T\sigma_\sharp \circ g_2(x_1,x_3,x_4,x_2) \\
&= T\sigma_\sharp \circ \begin{bmatrix} x_1 \\ \cos x_3 \\ x_3+1 \\ x_3x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \cos x_3 \\ x_3+1 \\ x_3x_2 \end{bmatrix} \\
&= \begin{bmatrix} x_1 \\ x_3x_2 \\ \cos x_3 \\ x_3+1 \end{bmatrix} \\
&= g_3(x_1,x_2,x_3,x_4).
\end{aligned}$$

So the vector fields g_2 and g_3 are equivalent and this result can be extended to other

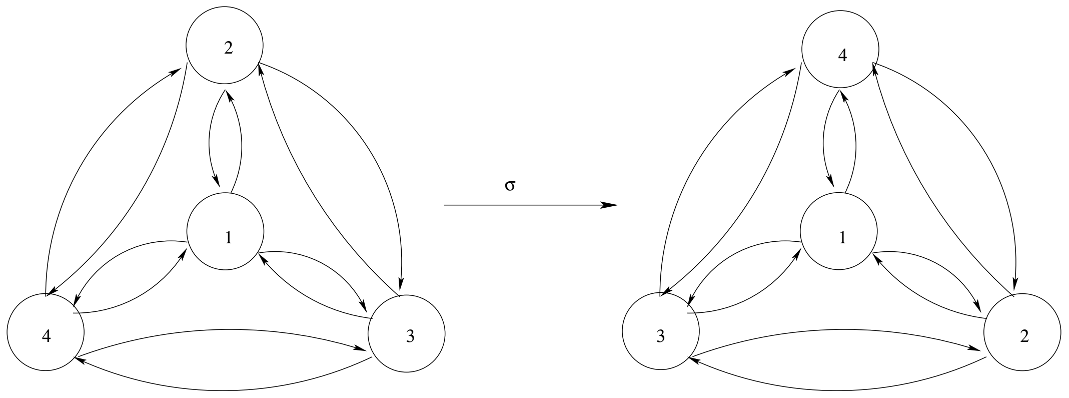

Figure 2.2. Graph symmetry σ acting on the four component system.

vector fields, the given system is symmetric. ■

Now, consider a general nonlinear system of the form,

$$\Sigma : \qquad \dot{x} = f(x) + \sum_{i=1}^{n} g_i(x) u_i \tag{2.3}$$

In order to define the symmetry in a distributed system with drift, we partitioning the drift vector field such that

$$f = \sum_{i=1}^{n} f_i(x),$$

where f_i is associated with the subsystem Σ_i. The kth component of the vector field f_i satisfies,

$$f_{i,k}(x_1, x_2, \ldots, x_i, \ldots, x_k, \ldots, x_n) = f_{i,k}(y_1, y_2, \ldots, x_i, \ldots, x_k, \ldots, y_n), \tag{2.4}$$

for all $\{x_p, y_q | p, q \in \{1, \ldots, n\}\}$. Then we can define the symmetric distributed system as follows.

Definition 2.4.5: Let $G_\Sigma = (V, E, f)$ be the graph of the distributed system Σ given by Equation 2.3. The system, Σ is symmetric if there exists a graph symmetry, $\sigma \in \mathrm{Aut}(G_\Sigma)$ other than the identity, such that if $\sigma(V_i) = V_j$, then

$$g_i \sim g_j, \qquad f_i \sim f_j.$$

■

Here is the example of symmetric distributed system with drift.

Example 2.4.6: Consider a four component system described by the following equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_1 + x_2 \\ x_1 + x_3 \\ x_1 + x_4 \end{bmatrix} + \begin{bmatrix} \sin x_1 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix} u_1 + \begin{bmatrix} x_1 \\ \cos x_2 \\ x_2 + 1 \\ x_2 x_4 \end{bmatrix} u_2 + \begin{bmatrix} x_1 \\ x_3 x_2 \\ \cos x_3 \\ x_3 + 1 \end{bmatrix} u_3 + \begin{bmatrix} x_1 \\ x_4 + 1 \\ x_4 x_3 \\ \cos x_4 \end{bmatrix} u_4,$$

where the vector field f can be partitioned as following:

$$f = \begin{bmatrix} x_1^2 \\ x_1 + x_2 \\ x_1 + x_3 \\ x_1 + x_4 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ x_1 + x_2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ x_1 + x_3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ x_1 + x_4 \end{bmatrix}.$$

We use the same graph symmetry Σ in example 3.4.4, then

$$\begin{aligned}
(\sigma_\sharp)_* f_2(x_1,x_2,x_3,x_4) &= T\sigma_\sharp \circ f_2 \circ \sigma_\sharp^{-1}(x_1,x_2,x_3,x_4) \\
&= T\sigma_\sharp \circ g_2(x_1,x_3,x_4,x_2) \\
&= T\sigma_\sharp \circ \begin{bmatrix} 0 \\ x_1+x_3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ x_1+x_3 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 0 \\ x_1+x_3 \\ 0 \end{bmatrix} \\
&= f_3(x_1,x_2,x_3,x_4).
\end{aligned}$$

This result can also be extended to other vector fields except f_1 and from example 3.4.4, we know $g_i \sim g_j$. So the given system in this example is a distributed system with drift. ■

CHAPTER 3

SYMMETRIC PROPERTIES OF A DISTRIBUTED SYSTEM MADE OF MICABOTs

This thesis considers large distributed systems made up of a group of identical robots which attain certain formations. We adopt two types of robotics: MICAbots [39] (see Figure 3.1) and unicycle-like autonomous mobile robots. In this chapter, we focus on the MICAbots. We will introduce the other prototypical model in next chapter.

3.1 Prototypical Model: MICAbots

The kinematics of MICAbot are described by

$$\begin{aligned} \dot{x} - \frac{r}{2}\cos\theta\,(u_1+u_2) &= 0 \\ \dot{y} - \frac{r}{2}\sin\theta\,(u_1+u_2) &= 0 \\ \dot{\theta} - \frac{r}{2b}(u_1-u_2) &= 0 \end{aligned} \tag{3.1}$$

where i is the index of the MICAbot, u_1 (u_2) is the angular velocity of the right (left) wheel of the robot, b and r are geometric parameters of the robot as illustrated in Figure 3.1.

Consider a formation when a group of n identical MICAbots form a n side regular polygon, but they are considered heterogeneous since each MICAbot has a unique

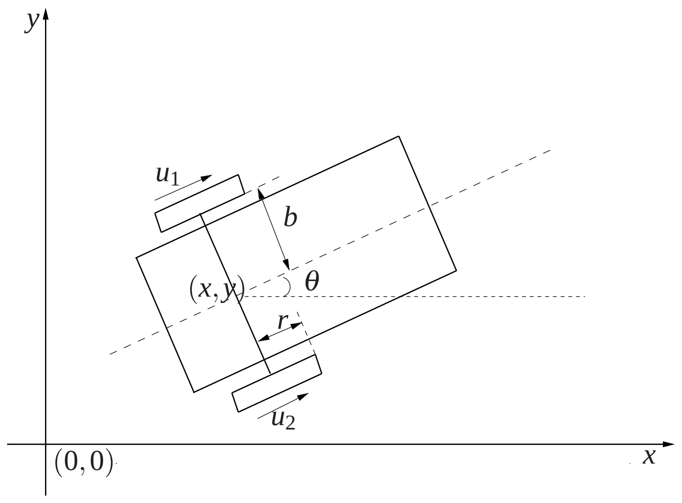

Figure 3.1. Sketch of MICAbot.

identification number (index). The graph theoretic representation of this system is illustrated in Figure 3.2.

Each vertex represents a MICAbot, and the edges between vertices represent the communications between the robots. For the distributed system containing n identical MICAbots, the configuration space Q is a submanifold of $\mathbb{R}^{3n}$,

$$Q = \{\underbrace{\mathbb{R}^2 \times S^1 \times \mathbb{R}^2 \times S^1 \times \cdots \times \mathbb{R}^2 \times S^1}_{\text{n copies}}\} \subset \mathbb{R}^{3n}.$$

The configuration space Q can be partitioned into a set of n submanifolds, $Q_i = \{\mathbb{R}^2 \times S^1\}$, where $i \in \{1, \ldots, n\}$, such that Q is the Cartesian product of the Q_i, *i.e.* $Q = \prod_{i=1}^{n} Q_i$. The submanifold, Q_i is the configuration space of the ith MICAbot, *i.e.* $Q_i = (x_i, y_i, \theta_i)$.

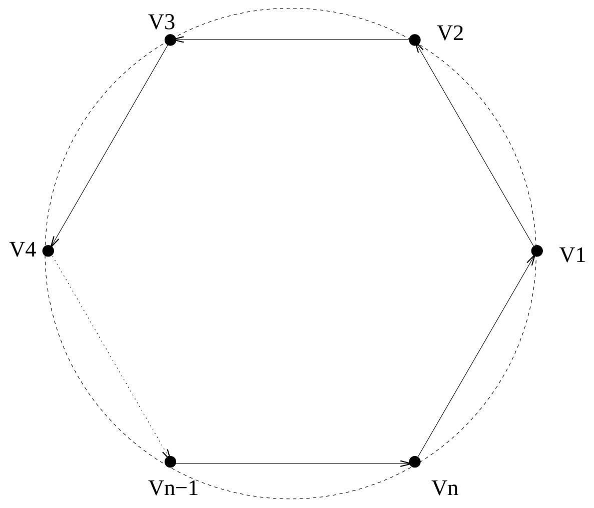

Figure 3.2. Graph theoretic representation of the distributed system.

3.2 Problem Statement

The optimal control problem we are investigating is defined as follows:

Find a sequence of controls $u_{1i}(t), u_{2i}(t)$ for each robot i which steer the formation from a predefined configuration q_0 to its goal configuration q_f, while maintaining a rigid body formation at the start and end points and minimizing the global performance index

$$J = \int_0^{t_f} \sum_{i=1}^{n} \left((u_{1i})^2 + (u_{2i})^2 \right) + \sum_{i=1}^{m} k \left(d_{i,i+1} - \overline{d} \right)^2 dt \tag{3.2}$$

subject to the robot kinematic constraints 3.1 where $n > 2$ is the number of robots, $d_{i,j} = \sqrt{((x_i - x_j)^2 + (y_i - y_j)^2)}$ is the Euclidean distance from ith to jth robots, $\overline{d}$ is the desired distance between two adjacent robots, and k is a non-negative weighting constant. For a closed formation such as a regular polygon, $m = n$; for an open formation such as a straight line, $m = n - 1$.

The definition of a rigid body formation is that the distances between chosen reference points on robots remain fixed. The relative orientation of each robot is not restricted in such a formation. The rigid body formation constraint is restrictive in many applications. We allow the robots to break formation in the middle of the trajectory, and to achieve a desired formation at the destination. The second summation in the cost function is the deviation from a desired formation. It is natural to minimize the the combination of the control effort (first summation) and the deviation in our case.

Remark 3.2.1: The cost function J for a regular polygon formation is invariant under the action of dihedral group D_n.

Proof: Let r, f be the two generators of the dihedral group, such that $r(V_i) = V_{i+1}$, $f(V_i) = V_{n+2-i}$, (note that $V_{n+1} = V_1$).

The cost function may be written as

$$J(V_1, V_2, \cdots V_n) = \int_0^1 \sum_{i=1}^{n} ({u_{1i}}^2 + {u_{2i}}^2) + \sum_{i=1}^{n} k(d_{i,i+1} - \overline{d})^2 dt$$

then

$$\begin{aligned} J \circ \phi_r(V_1, V_2, \cdots V_n) &= J(V_2, V_3, \cdots V_n, V_1) \\ &= \int_0^1 \sum_{i=2}^{n+1} ({u_{1i}}^2 + {u_{2i}}^2) + \sum_{i=2}^{n+1} k(d_{i,i+1} - \overline{d})^2 dt \\ &= \int_0^1 \sum_{j=1}^{n} ({u_{1j}}^2 + {u_{2j}}^2) + \sum_{j=1}^{n} k(d_{j,j+1} - \overline{d})^2 dt \\ &= J(V_1, V_2, \cdots V_n), \end{aligned}$$

and

$$\begin{aligned}
J \circ \phi_f(V_1, V_2, \cdots V_n) &= J(V_{n+1}, V_n, \cdots V_3, V_2) \\
&= \int_0^1 \sum_{i=n+1}^{2} ({u_{1i}}^2 + {u_{2i}}^2) + \sum_{i=n+1}^{2} k(d_{i,i+1} - \overline{d})^2 dt \\
&= \int_0^1 \sum_{i=2}^{n+1} ({u_{1i}}^2 + {u_{2i}}^2) + \sum_{i=2}^{n+1} k(d_{i-1,i} - \overline{d})^2 dt \\
&= \int_0^1 \sum_{j=1}^{n} ({u_{1j}}^2 + {u_{2j}}^2) + \sum_{j=1}^{n} k(d_{j,j+1} - \overline{d})^2 dt \\
&= J(V_1, V_2, \cdots V_n).
\end{aligned}$$

For each element $\sigma \in D_n$, σ can be written as $\sigma = f^p r^q$, where $p = 0, 1$ and $q = 0, 1, \cdots, n-1$.

If $p = 0, q = 0$, $\sigma = e$ and it is a trivial permutation.

If $p = 0, q > 0$, $\sigma = r^q$. In this case, we use induction method to prove that the cost function under the action of σ is invariant.

We already proved that $J \circ \phi_r(V_1, V_2, \cdots, V_n) = J(V_1, V_2, \cdots V_n)$. Now, we assume that $J \circ \phi_{r^s}(V_1, V_2, \cdots, V_n) = J(V_1, V_2, \cdots V_n)$, where s is a positive integer. Then

$$\begin{aligned}
J \circ \phi_{r^{s+1}}(V_1, V_2, \cdots V_n) &= J \circ r \circ r^s(V_1, V_2, \cdots V_n) \\
&= J \circ r^s(V_1, V_2, \cdots V_n \\
&= J(V_1, V_2, \cdots V_n).
\end{aligned}$$

If $p=1, q>0$, $\sigma = fr^q$. The cost function under the action of σ may be given by

$$\begin{aligned} J\circ\phi_{\sigma}(V_1,V_2,\cdots,V_n) &= J\circ\phi_f\circ\phi_{r^q}(V_1,V_2,\cdots,V_n) \\ &= J\circ\phi_{r^q}(V_1,V_2,\cdots,V_n) \\ &= J(V_1,V_2,\cdots V_n). \end{aligned}$$

In summary, for all $\sigma\in D_n$, the cost function does not change under the action of σ. ■

3.3 Equations of Motion

We adopt Pontryagin's Maximum Principle (PMP) to solve the optimal control problem stated in last section. The Hamiltonian associated with this problem is

$$\begin{aligned} H = \quad & \sum_{i=1}^{n}\Big\{-\big({u_{1i}}^2+{u_{2i}}^2+k(d_{i,i+1}-\overline{d})^2\big) \\ & + p_{1i}(\frac{r}{2}(u_{1i}+u_{2i})\cos\theta_i)+p_{2i}(\frac{r}{2}(u_{1i}+u_{2i})\sin\theta_i)+p_{3i}(\frac{r}{2b}(u_{1i}-u_{2i}))\Big\}, \end{aligned}$$

where $(\dot{x}_i(t),\dot{y}_i(t),\dot{\theta}_i(t))\in TQ_i$ and $(p_{1i},p_{2i},p_{3i})\in T^*Q_i$ are the system costates.

Remark 3.3.1: The Hamiltonian H is invariant under the action of dihedral group D_n. The proof of this remark is similar to that of the previous one.

According to PMP, we know the optimal u_1,u_2 should satisfy

$$(u_1,u_2)=\arg\max_{u_1,u_2} H.$$

Applying PMP to our problem, we have

$$\begin{aligned}(u_{1i}, u_{2i}) &= \arg\max_{u_{1i},u_{2i}} \Big\{ -\left({u_{1i}}^2 + {u_{2i}}^2\right) + p_{1i}(\frac{r}{2}(u_{1i}+u_{2i})\cos\theta_i) \\ &\quad + p_{2i}(\frac{r}{2}(u_{1i}+u_{2i})\sin\theta_i) + p_{3i}(\frac{r}{2b}(u_{1i}-u_{2i}))\Big\} \\ &= \arg\max_{u_{1i},u_{2i}} -\Big\{ \left(u_{1i} - (\frac{r}{4}p_{1i}\cos\theta_i + \frac{r}{4}p_{2i}\sin\theta_i + \frac{r}{4b}p_{3i})\right)^2 \\ &\quad + \left(u_{2i} - (\frac{r}{4}p_{1i}\cos\theta_i + \frac{r}{4}p_{2i}\sin\theta_i - \frac{r}{4b}p_{3i})\right)^2 \Big\},\end{aligned}$$

which gives

$$u_{1i} = \frac{r}{4}p_{1i}\cos\theta_i + \frac{r}{4}p_{2i}\sin\theta_i + \frac{r}{4b}p_{3i} \tag{3.3}$$

$$u_{2i} = \frac{r}{4}p_{1i}\cos\theta_i + \frac{r}{4}p_{2i}\sin\theta_i - \frac{r}{4b}p_{3i}. \tag{3.4}$$

The adjoint equation for $p(t)$ is

$$\dot{p}_{1i}(t) = -\frac{\partial H}{\partial x_i}, \qquad \dot{p}_{2i}(t) = -\frac{\partial H}{\partial y_i}, \qquad \dot{p}_{3i}(t) = -\frac{\partial H}{\partial \theta_i}.$$

The solutions to these equations together with state equations are

$$\begin{aligned}\dot{x}_i &= \frac{r^2}{4}\cos\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \dot{y}_i &= \frac{r^2}{4}\sin\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \dot{\theta}_i &= \frac{r^2}{4b^2}p3_i \\ \dot{p}_{1i} &= \frac{2k\left(x_i - x_{i-1}\right)\left(d_{i-1,i} - \overline{d}\right)}{d_{i-1}} + \frac{2k\left(x_i - x_{i+1}\right)\left(d_{i,i+1} - \overline{d}\right)}{d_{i,i+1}} \\ \dot{p}_{2i} &= \frac{2k\left(y_i - y_{i-1}\right)\left(d_{i-1,i} - \overline{d}\right)}{d_{i-1}} + \frac{2k\left(y_i - y_{i+1}\right)\left(d_{i,i+1} - \overline{d}\right)}{d_{i,i+1}} \\ \dot{p}_{3i} &= \frac{r^2}{4}\left(p_{1i}\sin\theta_i - p_{2i}\cos\theta_i\right)\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right),\end{aligned} \tag{3.5}$$

where $i = \{1, 2, \cdots, n\}$, and n is the number of robots. The initial conditions $x_i(0) = x_{i0},\ y_i(0) = y_{i0},\ \theta_i(0) = \theta_{i0}$, and final conditions, $x_i(t_f) = x_{if},\ y_i(t_f) = y_{if},\ \theta_i(t_f) = \theta_{if}$ are given.

Our goal here is to find curves $c_i(t) = (x_i(t), y_i(t), \theta_i(t)) \in Q_i$, $i = \{1, 2, \cdots, n\}$ satisfying the Equations 3.5 with $c_i(0) = (x_{i0}, y_{i0}, \theta_{i0})$ and $c_i(t_f) = (x_{if}, y_{if}, \theta_{if})$.

Remark 3.3.2: There exists a three parameters Lie group G, such that the differential equations 3.5 are invariant under the action of the Lie group. Any element $g \in G$ is defined as follows: for any $(x_i, y_i, \theta_i, p_{1i}, p_{2i}, p_{3i})^T \in M_i$ satisfying Equations 3.5, $(\hat{x}_i, \hat{y}_i, \hat{\theta}_i, \hat{p1}_i, \hat{p2}_i, \hat{p3}_i)^T = g(x_i, y_i, \theta_i, p_{1i}, p_{2i}, p_{3i})^T \in M_i$ and

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{\theta}_i \\ \hat{p}_{1i} \\ \hat{p}_{2i} \\ \hat{p}_{3i} \end{bmatrix} = g \begin{bmatrix} x_i \\ y_i \\ \theta_i \\ p_{1i} \\ p_{2i} \\ p_{3i} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\psi & -\sin\psi & 0 \\ 0 & 0 & 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ \theta_i \\ p_{1i} \\ p_{2i} \\ p_{3i} \end{bmatrix} + \begin{bmatrix} q_x \\ q_y \\ \psi \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where q_x, q_y, ψ are three parameters.

Proof:

$$\begin{aligned}
\dot{\hat{x}}_i &= \frac{d}{dt}(x_i\cos\psi - y_i\sin\psi + q_x)\\
&= \dot{x}_i\cos\psi - \dot{y}_i\sin\psi\\
&= \frac{r^2}{4}\cos\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right)\cos\psi - \frac{r^2}{4}\sin\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right)\sin\psi\\
&= \frac{r^2}{4}\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right)\cos(\theta_i+\psi)\\
&= \frac{r^2}{4}\left((p_{1i}\cos\psi - p_{2i}\sin\psi)\cos\hat{\theta}_i + (p_{1i}\sin\psi + p_{2i}\cos\psi)\sin\hat{\theta}_i\right)\cos\hat{\theta}_i\\
&= \frac{r^2}{4}\cos\hat{\theta}_i\left(\hat{p}_{1i}\cos\hat{\theta}_i + \hat{p}_{2i}\sin\hat{\theta}_i\right).
\end{aligned}$$

Similarly

$$\dot{\hat{y}}_i = \frac{r^2}{4}\sin\hat{\theta}_i\left(\hat{p}_{1i}\cos\hat{\theta}_i + \hat{p}_{2i}\sin\hat{\theta}_i\right).$$

$\dot{\hat{\theta}}_i$ is easy to verify:

$$\dot{\hat{\theta}}_i = \frac{d}{dt}(\theta_i + \psi) = \dot{\theta}_i = \frac{r^2}{4b^2}p_{3i} = \frac{r^2}{4b^2}\hat{p}_{3i},$$

$$\begin{aligned}
\hat{d}_{i-1,i} &= \sqrt{(\hat{x}_{i-1}-\hat{x}_i)^2 + (\hat{y}_{i-1}-\hat{y}_i)^2}\\
&= \sqrt{((x_{i-1}-x_i)\cos\psi - (y_{i-1}-y_i)\sin\psi)^2 + ((x_{i-1}-x_i)\sin\psi + (y_{i-1}-y_i)\cos\psi)^2}\\
&= \sqrt{(x_{i-1}-x_i)^2 + (y_{i-1}-y_i)^2} = d_{i-1,i}
\end{aligned}$$

$$
\begin{aligned}
\dot{\hat{p}}_{1i} &= \frac{d}{dt}(p_{1i}\cos\psi - p_{2i}\sin\psi) \\
&= \dot{p1}_i\cos\psi - \dot{p2}_i\sin\psi \\
&= \left(\frac{2k(x_i-x_{i-1})\left(d_{i-1,i}-\overline{d}\right)}{d_{i-1,i}} + \frac{2k(x_i-x_{i+1})\left(d_{i,i+1}-\overline{d}\right)}{d_{i,i+1}}\right)\cos\psi \\
&\quad - \left(\frac{2k(y_i-y_{i-1})\left(d_{i-1,i}-\overline{d}\right)}{d_{i-1,i}} + \frac{2k(y_i-y_{i+1})(d_{i,i+1}-d_n)}{d_{i,i+1}}\right)\sin\psi \\
&= \frac{2k((x_i\cos\psi - y_i\sin\psi)-(x_{i-1}\cos\psi - y_{i-1}\sin\psi))\left(\hat{d}_{i-1,i}-\overline{d}\right)}{\hat{d}_{i-1,i}} \\
&\quad + \frac{2k((x_i\cos\psi - y_i\sin\psi)-(x_{i+1}\cos\psi - y_{i+1}\sin\psi))\left(\hat{d}_{i,i+1}-\overline{d}\right)}{\hat{d}_{i,i+1}} \\
&= \frac{2k(\hat{x}_i-\hat{x}_{i-1})\left(\hat{d}_{i-1,i}-\overline{d}\right)}{\hat{d}_{i-1,i}} + \frac{2k(\hat{x}_i-\hat{x}_{i+1})\left(\hat{d}_{i,i+1}-\overline{d}\right)}{\hat{d}_{i,i+1}}.
\end{aligned}
$$

Similarly

$$
\dot{\hat{p}}_{2i} = \frac{2k(\hat{y}_i-\hat{y}_{i-1})\left(\hat{d}_{i-1,i}-\overline{d}\right)}{\hat{d}_{i-1,i}} + \frac{2k(\hat{y}_i-\hat{y}_{i+1})\left(\hat{d}_{i,i+1}-\overline{d}\right)}{\hat{d}_{i,i+1}},
$$

and

$$
\begin{aligned}
\dot{\hat{p}}_{3i} &= \dot{p}_i^3 = \frac{r^2}{4}(p_{1i}\sin\theta_i - p_{2i}\cos\theta_i)(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i) \\
&= \frac{r^2}{4}(p_{1i}\sin(\hat{\theta}_i-\psi) - p_{2i}\cos(\hat{\theta}_i-\psi))(p_{1i}\cos(\hat{\theta}_i-\psi) + p_{2i}\sin(\hat{\theta}_i-\psi)) \\
&= \frac{r^2}{4}(\hat{p1}_i\sin\hat{\theta}_i - \hat{p2}_i\cos\hat{\theta}_i)(\hat{p}_i\cos\hat{\theta}_i + \hat{p2}_i\sin\hat{\theta}_i).
\end{aligned}
$$

Since $(\hat{x}_i, \hat{y}_i, \hat{\theta}_i, \hat{p}_{1i}, \hat{p}_{2i}, \hat{p}_{3i})^T$ satisfy the motion equations 3.5, the equations are invariant under the specified Lie group, in other words, the system has Lie group symmetry, which illustrated in Figure 3.3. ■

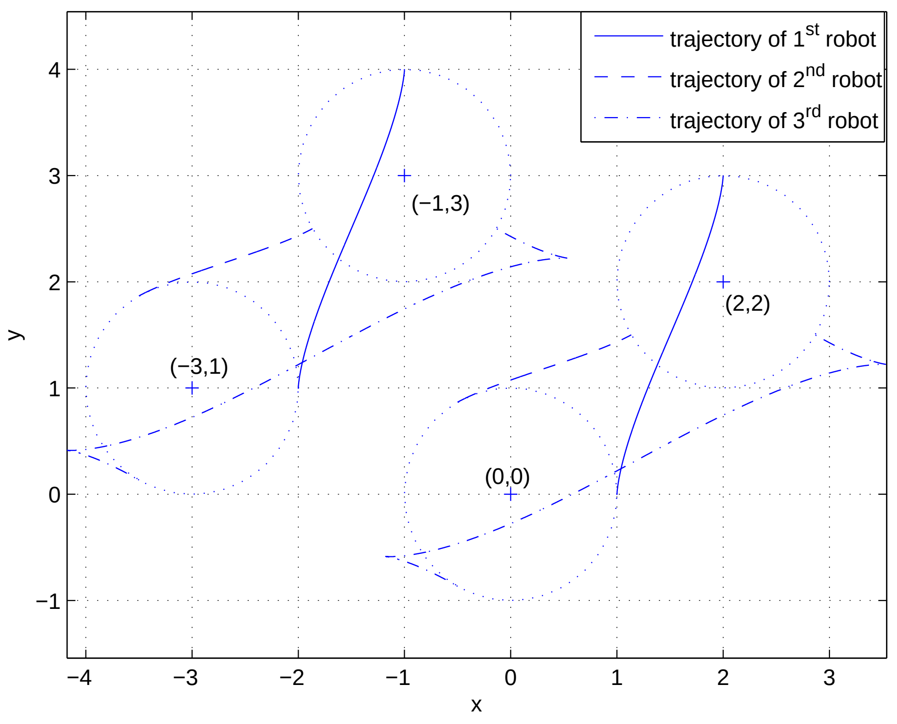

Figure 3.3. Lie symmetry.

3.4 Algorithm

We develop a relaxation shooting method to solve the equations of motion and obtain the trajectory of each robot. Let z represent the state and p stand for costate, the algorithm of this method is as follows.

1. Choose initial guess $p_i(0) = p_{i0}$ for the costate.

2. Numerically solve the equations of motion with the initial conditions $\{z_{i0},\ p_{i0}\}$. Obtain the final states z_{if}^0 based on the initial guess.

3. Compare z_{if}^0 with z_{if}. Divide the straight line segment from z_{if}^0 to z_{if} into several

small intervals. Let

$$Z_i^j = z_{if}^0 + (z_{if} - z_{if}^0)\frac{j}{m},$$

where $j = 0, 1, \cdots, m$ and m is the number of the intervals.

4. Treat z_{i0} and Z_i^0 as new initial and final boundary conditions of equations of motion. Use the shooting method [43] to solve the two points boundary value problems. We can obtain the proper initial values of the costates for this specified problem after solved the equations.

5. Use the initial value of the costate from fourth step as the new initial guess for the differential equations whose initial conditions are z_{i0} and final conditions are Z_i^1.

6. Repeat the fifth step until $j = m$. Then the whole problem is solved.

We give an example to illustrate the algorithm presented above. The initial q_0 and final configurations q_f (positions and orientations) are given for each robots in the system. We make initial guesses for all the costates, and we could get the final configurations q_f^0 and the trajectories of each robot could be found based on these initial guesses, which is shown as dashed lines in Figure 3.4. We divide the difference between the given final configurations q_f and calculated final configurations q_f^0 into several parts $q_f^0, q_f^1, q_f^2, \cdots, q_f$. Consider q_f^1 as the new desired final configurations, we use shooting method to get new trajectories, which are the closest lines to the dashed lines in Figure 3.4. Step by step, we can obtain the desired trajectories, thick lines in Figure 3.4.

For a distributed system, it becomes more and more difficult and sometimes infeasible to numerically find solutions to the problem as the number of robots increases.

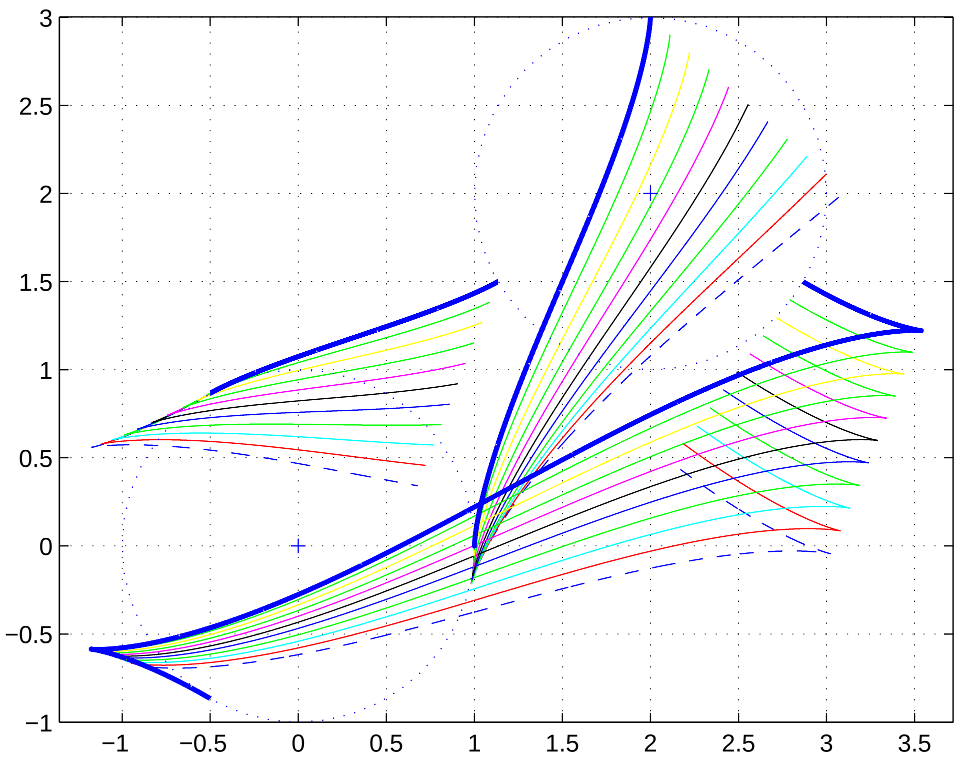

Figure 3.4. Illustration of the algorithm.

We may predict unknown trajectories of some robots based on the knowledge of other robots as stated in Proposition 3.4.1

Proposition 3.4.1: Consider a robotic system containing n robots and let C be the center of the regular polygon, which vertices represents MICAbots (see Figure 3.5). Suppose the trajectories of robot i and robot j are reflection backward in time about the perpendicular bisector to the straight line segment connecting initial point $C(X_0,Y_0)$ and final point $C(X_f,Y_f)$, which slope is a. Then,

$$\begin{bmatrix} x_i(t) \\ y_i(t) \\ \theta_i(t) \\ p_{1i}(t) \\ p_{2i}(t) \\ p_{3i}(t) \end{bmatrix} = \begin{bmatrix} \frac{a^2-1}{a^2+1} & -\frac{2a}{a^2+1} & 0 & 0 & 0 & 0 \\ -\frac{2a}{a^2+1} & -\frac{a^2-1}{a^2+1} & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{a^2-1}{a^2+1} & \frac{2a}{a^2+1} & 0 \\ 0 & 0 & 0 & \frac{2a}{a^2+1} & \frac{a^2-1}{a^2+1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j(t_f-t) \\ y_j(t_f-t) \\ \theta_j(t_f-t) \\ p1_j(t_f-t) \\ p2_j(t_f-t) \\ p3_j(t_f-t) \end{bmatrix} + \begin{bmatrix} X_f - X_0 \\ Y_f - Y_0 \\ 2\arctan a \\ 0 \\ 0 \\ 0 \end{bmatrix} . (3.6)$$

Proof: Let $\alpha = \arctan a$ and $\tau = t_f - t$, then

$$\sin(2\alpha) = \frac{2a}{1+a^2}, \quad \cos(2\alpha) = \frac{1-a^2}{1+a^2}.$$

After some simple but tedious computations, we obtain

$$p_{1j}(\tau)\cos\theta_j(\tau) + p_{2j}(\tau)\sin\theta_j(\tau) = p_{1i}(t)\cos\theta_i(t) + p_{2i}(t)\sin\theta_i(t),$$
$$d_{j-1,j}(\tau) = d_{i-1,i}(t),$$

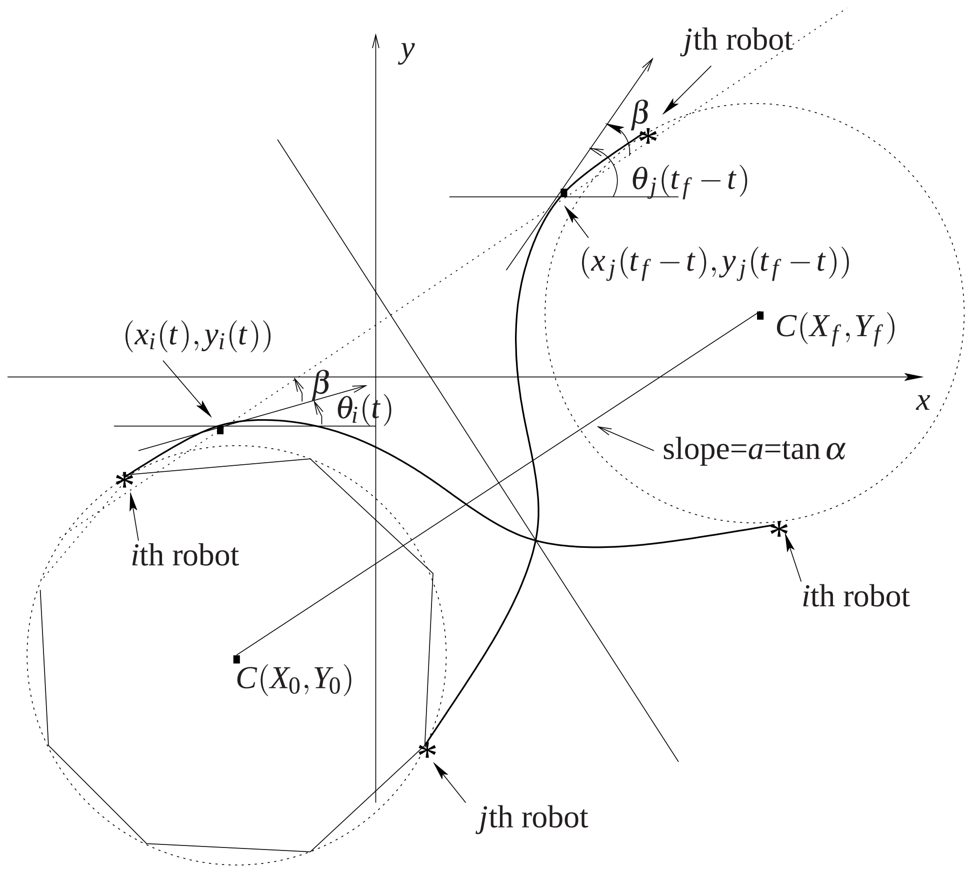

Figure 3.5. Illustration of symmetry trajectories.

$$
\begin{aligned}
\dot{x}_i &= \frac{d}{dt}\left(\frac{a^2-1}{a^2+1}x_j(\tau)-\frac{2a}{a^2+1}y_j(\tau)+(X_f-X_0)\right)\\
&= -\frac{a^2-1}{a^2+1}\frac{dx_j(\tau)}{d(\tau)}+\frac{2a}{a^2+1}\frac{dy_j(\tau)}{d(\tau)}\\
&= -\frac{a^2-1}{a^2+1}\left(\frac{r^2}{4}\cos\theta_j(\tau)\left(p_{1j}(\tau)\cos\theta_j(\tau)+p_{2j}(\tau)\sin\theta_j(\tau)\right)\right)\\
&\quad+\frac{2a}{a^2+1}\left(\frac{r^2}{4}\sin\theta_j(\tau)\left(p_{1j}(\tau)\cos\theta_j(\tau)+p_{2j}(\tau)\sin\theta_j(\tau)\right)\right)\\
&= \frac{r^2}{4}\left(p_{1i}(t)\cos\theta_i(t)+p_{2i}(t)\sin\theta_i(t)\right)\left(-\frac{a^2-1}{a^2+1}\cos\theta_j(\tau)+\frac{2a}{a^2+1}\sin\theta_j(\tau)\right)\\
&= \frac{r^2}{4}\cos(2\alpha-\theta_j(\tau))\left(p_{1i}(t)\cos\theta_i(t)+p_{2i}(t)\sin\theta_i(t)\right)\\
&= \frac{r^2}{4}\cos(\theta_i(t))\left(p_{1i}(t)\cos\theta_i(t)+p_{2i}(t)\sin\theta_i(t)\right).
\end{aligned}
$$

Similarly

$$
\begin{aligned}
\dot{y}_i &= \frac{d}{dt}\left(-\frac{2a}{a^2+1}x_j(\tau) - \frac{a^2-1}{a^2+1}y_j(\tau) + (Y_f - Y_0)\right) \\
&= \frac{r^2}{4}\left(p_{1i}(t)\cos\theta_i(t) + p_{2i}(t)\sin\theta_i(t)\right)\left(\frac{2a}{a^2+1}\cos\theta_j(\tau) + \frac{a^2-1}{a^2+1}\sin\theta_j(\tau)\right) \\
&= \frac{r^2}{4}\sin(2\alpha - \theta_j(\tau))\left(p_{1i}(t)\cos\theta_i(t) + p_{2i}(t)\sin\theta_i(t)\right) \\
&= \frac{r^2}{4}\sin(\theta_i(t))\left(p_{1i}(t)\cos\theta_i(t) + p_{2i}(t)\sin\theta_i(t)\right), \\
\dot{\theta}_i &= \frac{d}{dt}(-\theta_j(\tau) + 2\arctan a) = \frac{d(\theta_j(\tau))}{d(\tau)} = \frac{r^2}{4b^2}p_{3i}(\tau) = \frac{r^2}{4b^2}p_{3i}, \\
\dot{p}_{1i} &= \frac{d}{dt}\left(-\frac{a^2-1}{a^2+1}p_{1j}(\tau) + \frac{2a}{a^2+1}p_{2j}(\tau)\right) \\
&= \frac{a^2-1}{a^2+1}\frac{d(p_{1j}(\tau))}{d(\tau)} - \frac{2a}{a^2+1}\frac{d(p_{2j}(\tau))}{d(\tau)} \\
&= \frac{a^2-1}{a^2+1}\left(\frac{2k\left(x_j(\tau) - x_{j-1}(\tau)\right)\left(d_{i-1,i}(t) - \overline{d}\right)}{d_{i-1,i}(t)}\right. \\
&\quad \left. + \frac{2k\left(x_j(\tau) - x_{j+1}(\tau)\right)\left(d_{i,i+1}(t) - \overline{d}\right)}{d_{i,i+1}(t)}\right) \\
&\quad - \frac{2a}{a^2+1}\left(\frac{2k\left(y_j(\tau) - y_{j-1}(\tau)\right)\left(d_{i-1,i}(t) - \overline{d}\right)}{d_{i-1,i}(t)}\right. \\
&\quad \left. + \frac{2k\left(y_j(\tau) - y_{j+1}(\tau)\right)\left(d_{i,i+1}(t) - \overline{d}\right)}{d_{i,i+1}(t)}\right) \\
&= \frac{2k(d_{i-1,i}(t) - \overline{d})}{d_{i-1,i}(t)}\left(\frac{a^2-1}{a^2+1}\left(x_j - x_{j-1}\right) - \frac{2a}{a^2+1}\left(y_j - y_{j-1}\right)\right) \\
&\quad + \frac{2k(d_{i,i+1}(t) - \overline{d})}{d_{i,i+1}(t)}\left(\frac{a^2-1}{a^2+1}\left(x_j - x_{j+1}\right) - \frac{2a}{a^2+1}\left(y_j - y_{j+1}\right)\right) \\
&= \frac{2k\left(x_i - x_{i-1}\right)\left(d_{i-1,i} - \overline{d}\right)}{d_{i-1,i}} + \frac{2k\left(x_i - x_{j+1}\right)\left(d_{i,i+1} - \overline{d}\right)}{d_{i,i+1}}.
\end{aligned}
$$

The costate $\dot{p}_{2i}$ can be verified in very similar way and we skip the proof here. For the third costate

$$\begin{aligned}
\dot{p}_{3i} &= \frac{d(p_{3j}(\tau))}{d(\tau)} \\
&= -\frac{r^2}{4}(p_{1j}(\tau)\sin\theta_j(\tau) - p_{2j}(\tau)\cos\theta_j(\tau))(p_{1j}\cos\theta_j(\tau) + p_{2j}(\tau)\sin\theta_j(\tau)) \\
&= \frac{r^2}{4}(p_{1i}(t)\sin\theta_i(t) - p_{2i}(t)\cos\theta_i(t))(p_{1i}(t)\cos\theta_i(t) + p_{2i}(t)\sin\theta_i(t)).
\end{aligned}$$

■

Remark 3.4.2: Consider a robotic system containing n robots. If we give such initial and final configurations (position and orientation) for each robot i and there exists a robot j that the top three conditions of Equation 3.6 are satisfied. Then, if the whole system rotates by an angle $\phi = K\frac{2\pi}{n} - (\pi - 2\alpha)$, where K is an integer and $\alpha = \arctan\frac{Y_f - Y_0}{X_f - X_0}$, then $j = mod(n - K + 2 - i, n)$. Under such condition, if we want to know the trajectories of all robots in the system, we should at least know one of robot i and robot j's trajectory.

3.5 Symmetry in Distributed Systems

Our system is described by Equation 3.5. In this section, we are going to check if our system has symmetry by using the definition given in Chapter 2. The formation of the team of MICAbot is a an n-sided regular polygon, and the dihedral group D_n is the symmetry group of such graph. So the graph symmetry σ for our system is contained in the dihedral group D_n, which means $\sigma \in D_n = \{e, r, r^2, \cdots, r^{n-1}, f, fr, fr^2, \cdots, fr^{n-1}\}$.

We can partition the vector field in the system given by Equation 3.5 as

$$f(x) = \sum_{i=1}^{n} f_i$$

where

$$f_i = \begin{bmatrix} 0_{6\times 1} \\ 0_{6\times 1} \\ \vdots \\ \left\{ \begin{array}{c} \frac{r^2}{4}\cos\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \frac{r^2}{4}\sin\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \frac{r^2}{4b^2}p_{3i} \\ 0 \\ 0 \\ \frac{r^2}{4}\left(p_{1i}\sin\theta_i - p_{2i}\cos\theta_i\right)\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \end{array} \right\} \\ \vdots \\ 0_{6\times 1} \end{bmatrix}$$

the non-zero 6×1 blocks in the above matrix are in position i. For simplification, let

$$h_i = \left\{ \begin{array}{c} \frac{r^2}{4}\cos\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \frac{r^2}{4}\sin\theta_i\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \\ \frac{r^2}{4b^2}p_{3i} \\ 0 \\ 0 \\ \frac{r^2}{4}\left(p_{1i}\sin\theta_i - p_{2i}\cos\theta_i\right)\left(p_{1i}\cos\theta_i + p_{2i}\sin\theta_i\right) \end{array} \right\} \quad \text{and} \quad \underline{x}_i = \left\{ \begin{array}{c} x_i \\ y_i \\ \theta_i \\ p_{1i} \\ p_{2i} \\ p_{3i} \end{array} \right\},$$

then the Equation 3.5 can be rewritten as:

$$\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} = f(\underline{x}) = \sum_{i=1}^{n} f_i(\underline{x}) \quad f_i(\underline{x}) = \begin{bmatrix} \vdots \\ h_i \\ \vdots \end{bmatrix}$$

For each element $\sigma \in D_n$, σ can be written as $\sigma = f^p r^q$, where $p = 0, 1$ and $q = 0, 1, \cdots, n-1$.

- If $p = 0, q = 0$, $\sigma = e$ and it is a trivial permutation.
- If $p = 0, q > 0$, $\sigma = r^q$. Then $\sigma(V_i) = V_{i+q}$.

$$\begin{aligned} \sigma(V_1, \cdots, V_i, \cdots, V_n) &= (V_{1+q}, \cdots, V_{i+q}, \cdots, V_{n+q}), \\ \sigma_*(\underline{x}_1, \cdots, \underline{x}_i, \cdots, \underline{x}_n) &= (\underline{x}_{1+q}, \cdots, \underline{x}_{i+q}, \cdots, \underline{x}_{n+q}), \end{aligned}$$

$$\begin{aligned}
& (\sigma_\sharp)_* f_{i+q}(\underline{x}_1, \cdots, \underline{x}_i, \cdots, \underline{x}_n) \\
= \; & T\sigma_\sharp \circ f_{i+q} \circ \sigma_\sharp^{-1}(\underline{x}_1, \cdots, \underline{x}_{i+q}, \cdots, \underline{x}_n) \\
= \; & T\sigma_\sharp \circ f_{i+q}(\underline{x}_{1-q}, \cdots, \underline{x}_i, \cdots, \underline{x}_{n-q}) \\
= \; & \begin{bmatrix} 0 & \cdots & 0 & 1_{1,1+q} & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1_{2,2+q} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 1_{i,i+q} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1_{n,q} & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ h_i((i+q)\text{th row}) \\ \vdots \\ 0 \end{bmatrix} \\
= \; & \begin{bmatrix} 0 \\ \vdots \\ h_i(i\text{th row}) \\ \vdots \\ 0 \end{bmatrix} = f_i.
\end{aligned}$$

- If $p = 1, q > 0$, $\sigma = fr^q$. Then $\sigma(V_i) = V_{n+q+2-i}$.

$$\begin{aligned}
\sigma(V_1, \cdots, V_i, \cdots, V_n) &= (V_{n+q+1}, \cdots, V_{n+q+2-i}, \cdots, V_{q+2}), \\
\sigma_*(\underline{x}_1, \cdots, \underline{x}_i, \cdots, \underline{x}_n) &= (\underline{x}_{n+q+1}, \cdots, \underline{x}_{n+q+2-i}, \cdots, \underline{x}_{q+2}),
\end{aligned}$$

$$\begin{aligned}
& (\sigma_\sharp)_* f_{n+q+2-i}(\underline{x}_1, \cdots, \underline{x}_{n+q+2-i}, \cdots, \underline{x}_n) \\
= \; & T\sigma_\sharp \circ f_i \circ \sigma_\sharp^{-1}(\underline{x}_1, \cdots, \underline{x}_{n+q+2-i}, \cdots, \underline{x}_n) \\
= \; & T\sigma_\sharp \circ f_i(\underline{x}_{n+1-q}, \cdots, \underline{x}_i, \cdots, \underline{x}_{n+2-q}) \\
= \; & \begin{bmatrix} 0 & \cdots & 0 & 0 & 1_{1,q+1} & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & 1_{2,2+q} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1_{i-q,i} & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & & 1_{n,q+2} \cdots & 0 \end{bmatrix} \\
& \cdot \begin{bmatrix} 0 \\ \vdots \\ h_i((n+q+2-i)\text{th row}) \\ \vdots \\ 0 \end{bmatrix} \\
= \; & \begin{bmatrix} 0 \\ \vdots \\ h_i(i\text{th row}) \\ \vdots \\ 0 \end{bmatrix} = f_i.
\end{aligned}$$

To summarize, our distributed system is symmetric. Note, $1_{m,n}$ means the element 1 is in mth row, nth column.

We show how our approach works in simulation using several examples. Suppose that our systems are in an obstacle free environment. We first consider two examples which are not reduced to lower dimensional systems. We directly compute the optimal trajectories for all robots in the systems. These results can validate our Proposition 3.4.1 and Remark 3.4.2.

Example 3.5.1: We have five robots and they form a five-side regular polygon (see Figure 3.6). The robots are labeled by a unique integer number. The black point in the Figure 3.6 is the geometric center of the polygon. When we say the position of a formation, we mean the position of its center.

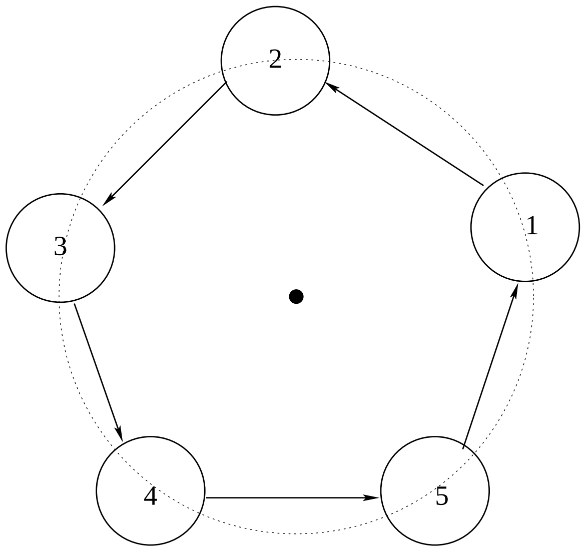

Figure 3.6. Five side regular polygon formation.

We solve the Equations 3.5 by numeric method for $n = 5$. At $t = 0$, the formation is at the point $(-1,-2)$. We choose $t_f = 1s$. The goal position is $(1,2)$. We try to find the optimal trajectories for all robots while the formation goes from $(-1,-2)$ to $(1,2)$ and rotates by an angle $\phi = -(\pi - 2\alpha)$ about its center and the cost function 3.2 is

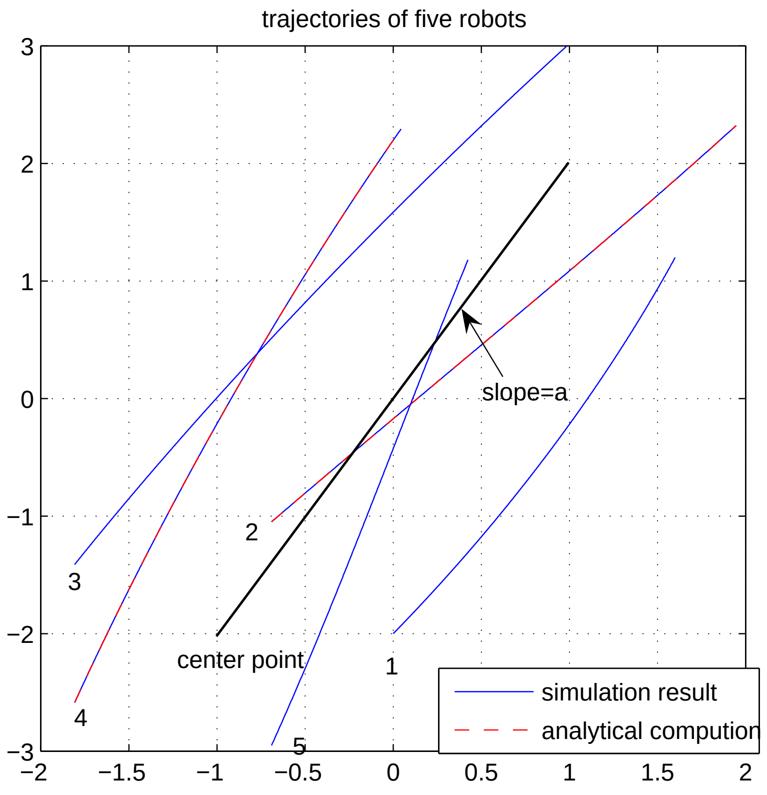

Figure 3.7. Trajectories of a team of five robots.

minimized. The trajectories are plotted in Figure 3.7. In Figure 3.7, the solid lines are the trajectories (costates) numerical solved from Equation 3.5. According the Remark 3.4.2, $\phi = -(\pi - 2\alpha) = k\frac{2\pi}{5} - (\pi - 2\alpha)$, *i.e.*, $k = 0$. Then, if we know the trajectories of the ith or $(7-i)$th, we can algebraically compute other robots' trajectories, which results in much computational savings. In this example, we pick $i = 1, 3, 5$. Using the information of these three robots to construct those of the second and fourth robots by using of Proposition 3.4.1. The dashed lines in the following figures represent these results.

$\blacksquare$

Example 3.5.2: In this example, our system contains six robots, which form a 6-side regular polygon. The solid and dashed lines represent the same things respectively as those in Figure 3.7. The coordinates of initial center of the formation is $(-1,-1)$ and the final position of the center is at $(1,1)$. The formation rotates by an angle $\phi = \frac{\pi}{2}$. According the remark 4.1.7, $\phi = \frac{\pi}{2} = k\frac{2\pi}{6} - (\pi - \frac{\pi}{2})$, *i.e.* $k = 3$. If we know the trajectories of robot i or robot $(5-i)$, we can algebraically compute the other robots' trajectories. Here, we pick $i = 1,3,5$. Since the trajectories and costates of the other three robots have the reflection relations with the known three robots respectively, we compute them and they are represented in dashed lines in Figure 3.8 and Figure 3.9.

■

The numerical results and the analytically computed results are exactly the same in the above two examples. These validated our Proposition 3.4.1 and Remark 3.4.2. 3.4.

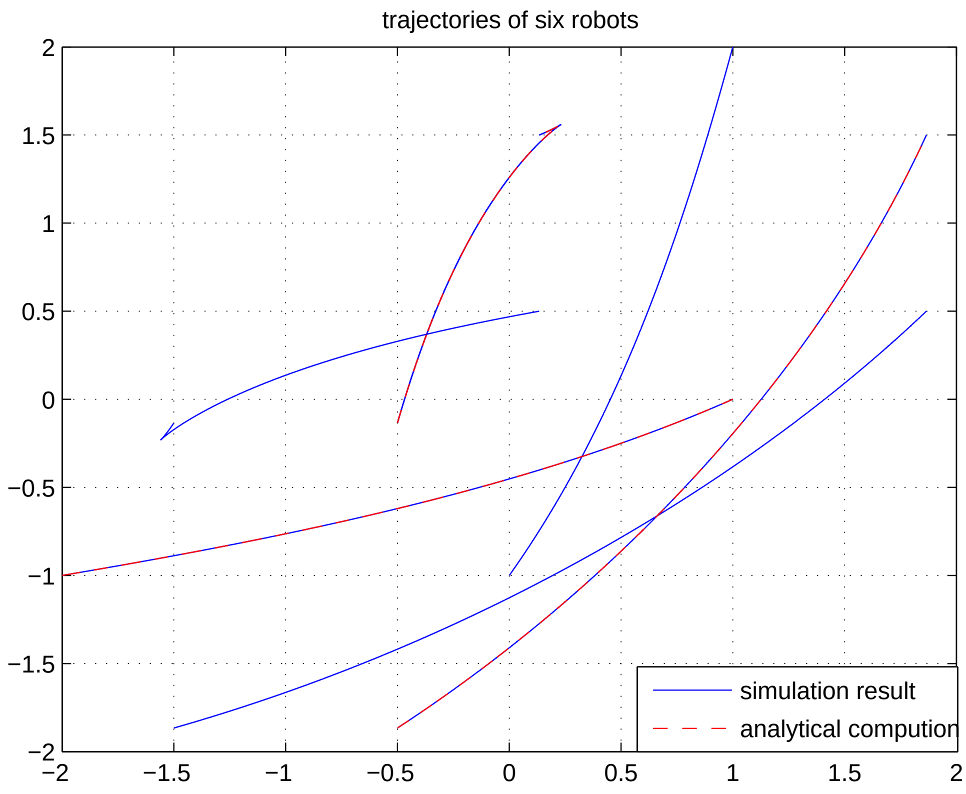

Figure 3.8. Trajectories of a team of six robots.

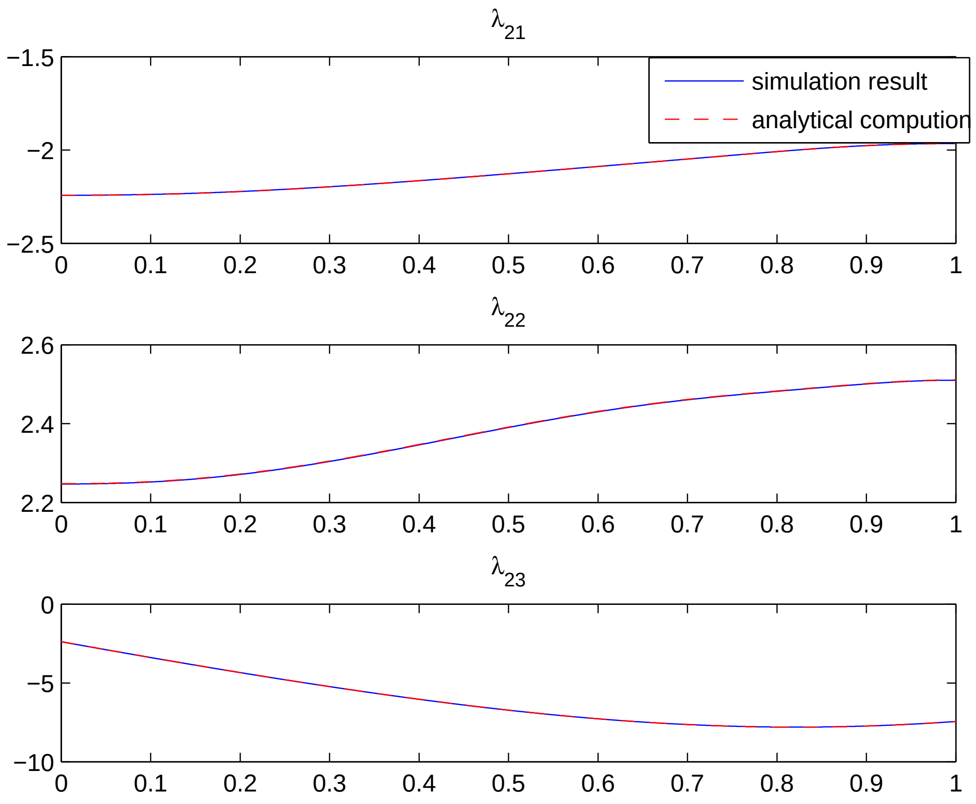

Figure 3.9. The optimal costates of the 2nd robot verse time for six robots system.

CHAPTER 4

BIFURCATION RESULTS OF A DISTRIBUTED SYSTEMS MADE OF UNICYCLES

The second prototypical model we adopt is a simplified version of the kinematic robotic unicycle. The kinematics of this kind of robot are described by

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2. \end{aligned} \tag{4.1}$$

In this chapter, the simplified unicycles we consider are arranged along a straight line. The optimal motion planning problem is to minimize the cost function 3.2 as defined in chapter 3, but subject to the constraints 4.1.

Similar procedure to the MICAbot, we obtain the optimal inputs

$$\begin{aligned} u_{1i} &= \frac{1}{2} p_{1i} \\ u_{2i} &= \frac{1}{2} p_{2i}, \end{aligned}$$

and equations of motion

$$
\begin{aligned}
\dot{x}_i &= \frac{1}{2}p_{1i} \\
\dot{y}_i &= \frac{1}{2}p_{2i} \\
\dot{p}_{1i} &= \frac{2k\left(x_i - x_{i-1}\right)\left(d_{i-1,i} - \overline{d}\right)}{d_{i-1,i}} + \frac{2k\left(x_i - x_{i+1}\right)\left(d_{i,i+1} - \overline{d}\right)}{d_{i,i+1}} \\
\dot{p}_{2i} &= \frac{2k\left(y_i - y_{i-1}\right)\left(d_{i-1,i} - \overline{d}\right)}{d_{i-1,i}} + \frac{2k\left(y_i - y_{i+1}\right)\left(d_{i,i+1} - \overline{d}\right)}{d_{i,i+1}},
\end{aligned}
\tag{4.2}
$$

where k is a non-negative weighting factor and $\overline{d}$ is the desired distance between two adjacent unicycles. Because they correspond to the robots at the end of the formation, the last two equations in Equation 4.2 only have the second term when $i = 1$ and they only have the first term when $i = n$.

The cases considered in this thesis are limited to the following boundary conditions

$$
\begin{aligned}
x_i(0) &= c + (i-1)\overline{d}, \\
x_i(1) &= 0, \\
y_i(0) &= 0, \\
y_i(1) &= c + (i-1)\overline{d},
\end{aligned}
\tag{4.3}
$$

where c is a constant. These boundary conditions correspond to an initial formation with the robots arranged along the x-axis starting with the first robot at $x = c$ with a distance $\overline{d}$ between each robot and a final formation with the robots arranged along the y-axis starting with the first robot at $y = c$ with a distance of $\overline{d}$ between each robot. It is important to note that if the initial and final formations are not parallel, then straight-line trajectories satisfying the boundary conditions will not, in general, maintain the desired distance between the robots.

For a distributed system containing n robots, when the weighting constant k is given, an optimal trajectory can be obtained numerically by solving the equations of motion given by Equation 4.2 using the relaxation shooting method. Since each robot has its own predefined initial state and final state, the procedure to find the optimal path is to solve a boundary value problem for a set of second order nonlinear ordinary differential equations. We show the solutions for three systems with different number of robots.

4.1 Bifurcation Results

Since k is a parameter in the differential equations, it will clearly affect the solutions. In fact, as k is varied, the nature and number of solutions changes. Section 4.3 shows that there is a unique solution when k is small and in the limit as k approaches infinity, the number of solutions also approaches infinity. In order to present the relationship between the number of solutions and k, we construct a bifurcation diagram as follows: since a straight line connecting end points is the optimal solution when $k = 0$, we will designate that as a nominal trajectory. One measure of the difference between solutions would be their deviation from the straight line nominal solution at some specified time. As long as the different solutions are not intersecting at that time, this would provide a measure of difference between different solutions. In all the bifurcation diagram illustrated subsequently, $t = 0.25$ is used. For different formations and different type of robots, a different value of t may be a better choice; however, for all the systems studied in this thesis, $t = 0.25$ appeared to adequately represent the relationship among the solutions. Also, alternative measures of differences between the solutions may, in general, be superior, this simple choice appears to suffice for all the cases considered in this thesis.

4.1.1 Solutions for a Five Robot System

Figure 4.1 illustrates three different solutions that satisfy the equations of motion in Equation 4.2 and boundary conditions in Equation 4.3 for $k = 24.5$, $c = 6$ and $\overline{d} = 2$ for a formation of five robots. Since the differences among these trajectories are difficult to distinguish on such a small graph, Figure 4.2 illustrates them for the third (middle) robot with the difference magnified by a factor of 10.

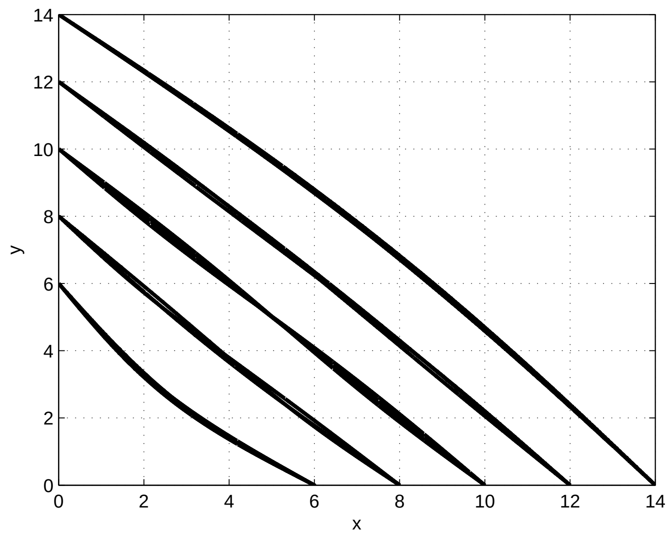

Figure 4.1. Optimal paths for the five robot system with $k = 24.5$.

The plots in Figure 4.3 through Figure 4.7 illustrate this measure of the difference between solutions for each robot in the five robot system as k is varied from 0 to 25. In these bifurcation diagrams, the first robot is the one with the shortest trajectory, the fifth robot is the one with the longest trajectory and they are ordered sequentially. The

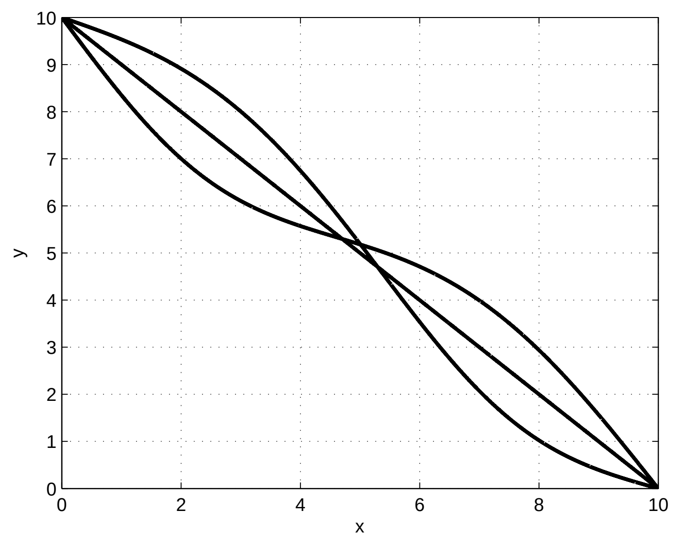

Figure 4.2. Difference among the optimal paths for robot three.

bifurcation occurs near $k = 16.5$. Observe that the bifurcation diagrams for robots 1 and 5 are symmetric to each other about $d = 0$ axis and the bifurcation diagrams for robots 2 and 4 are similarly symmetric (even though each follows a trajectory with a different length). Finally, the bifurcation diagram for robot 3 is symmetric to itself about $d = 0$ axis.

A close analysis of the actual trajectories that the robots follow illustrated in the figure on the right in Figure 4.1 reveals that the trajectories themselves are *not* symmetric (the two trajectories with pronounced curves intersect, but not at a point on the straight line solution). A measure that is based upon the deviation from the nominal solution appears to be necessary to determine the real symmetric nature of the solutions. Section 4.4 contains the analysis that these symmetries must, in fact, exist.

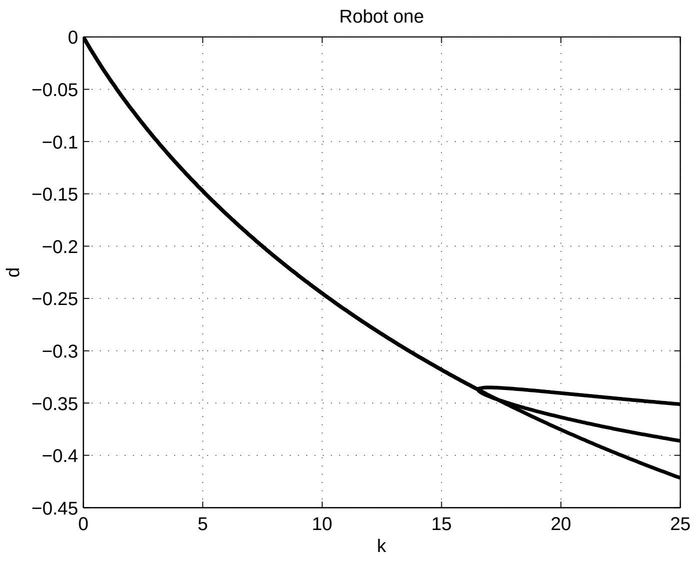

Figure 4.3. Bifurcation diagrams for robot one in a 5-robotic system.

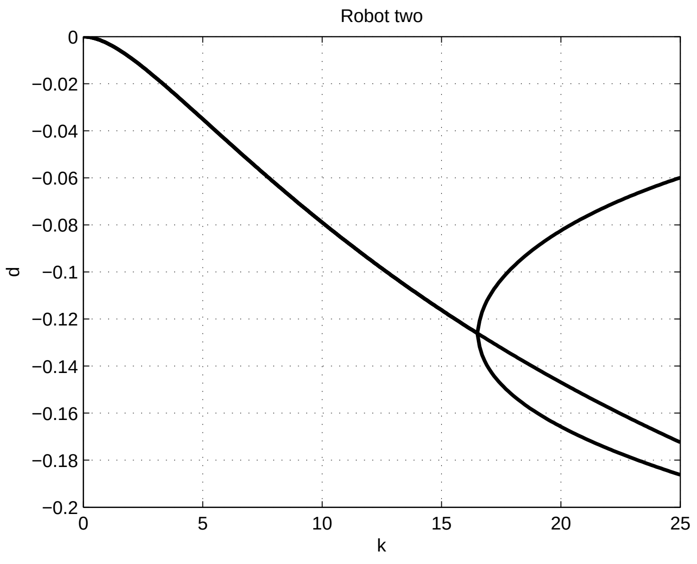

Figure 4.4. Bifurcation diagrams for robot two in a 5-robotic system.

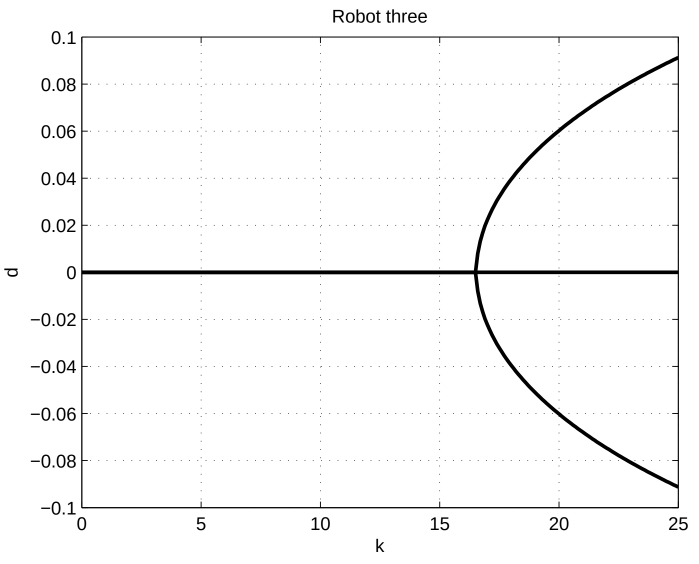

Figure 4.5. Bifurcation diagrams for robot three in a 5-robotic system.

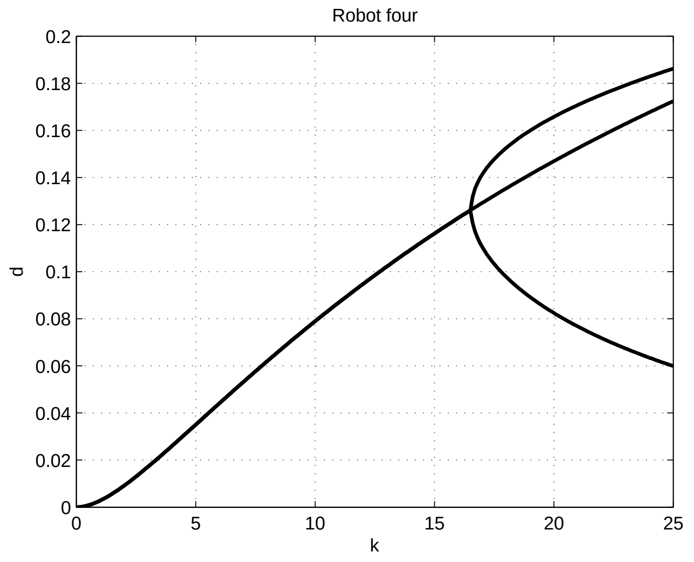

Figure 4.6. Bifurcation diagrams for robot four in a 5-robotic system.

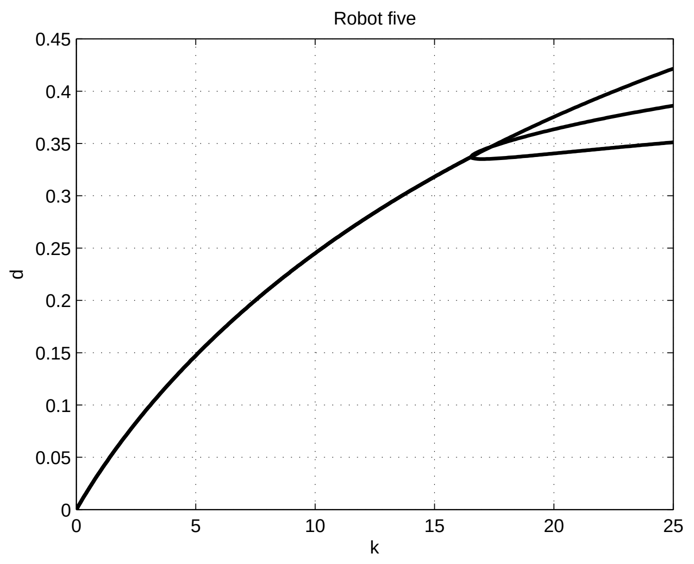

Figure 4.7. Bifurcation diagrams for robot five in a 5-robotic system.

4.1.2 Solutions for a Six Robot System

Figures 4.8 through 4.15 illustrate similar results for a six robot system. Figure 4.8 illustrates the trajectories when $k = 24.5$, $c = 4$ and $\overline{d} = 2$. Again, because the difference is hard to distinguish in this figure, Figure 4.9 illustrates the trajectory with the deviation from the nominal trajectory for the fifth robot magnified by a factor of five. Figure 4.18 through Figure 4.24 illustrate the bifurcation diagrams for the solutions versus k constructed in a manner identical to that of the system of five robots. The first bifurcation we found occurs near $k = 12.3$, the second occurs near $k = 16.1$ and the third occurs near $k = 23.2$. There might be other bifurcations we have not found due to the limitation of simulation. Observe that, different than the five robot case, there is no robot which bifurcation diagram is symmetric to itself about $d = 0$ axis. The bifurcation diagrams for robots 1 and 6 are symmetric to each other about $d = 0$ axis as is the bifurcation diagrams for robots 2 and 5 and robots 3 and 4.

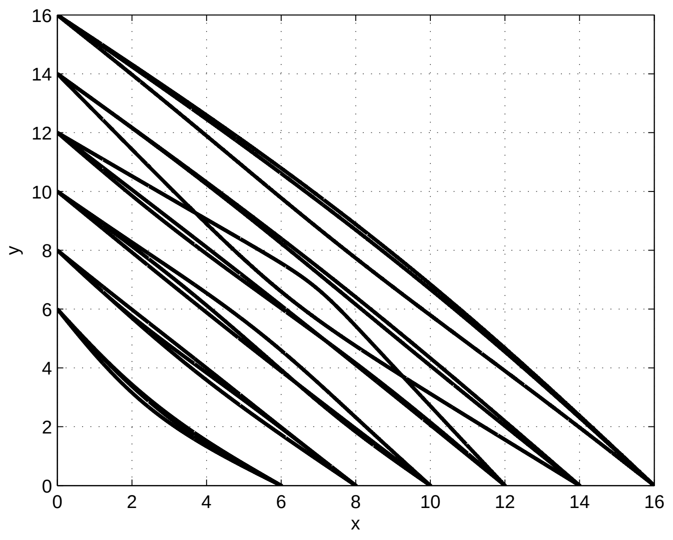

Figure 4.8. Optimal paths for a six robot system with $k = 24.5$.

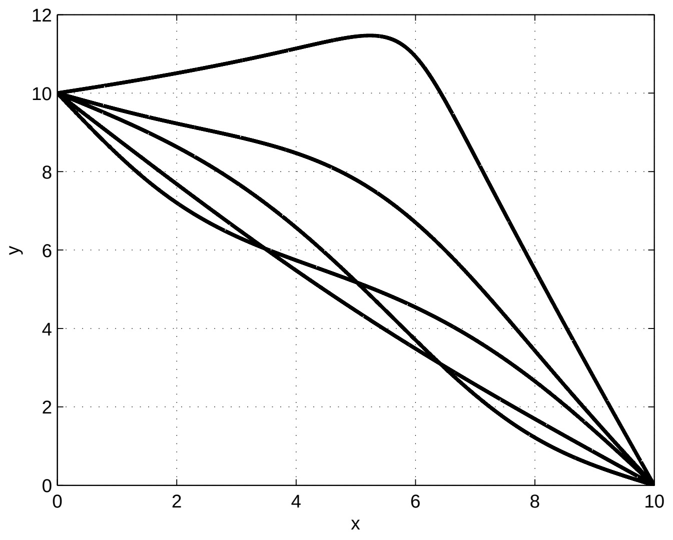

Figure 4.9. Difference among the optimal paths for robot three.

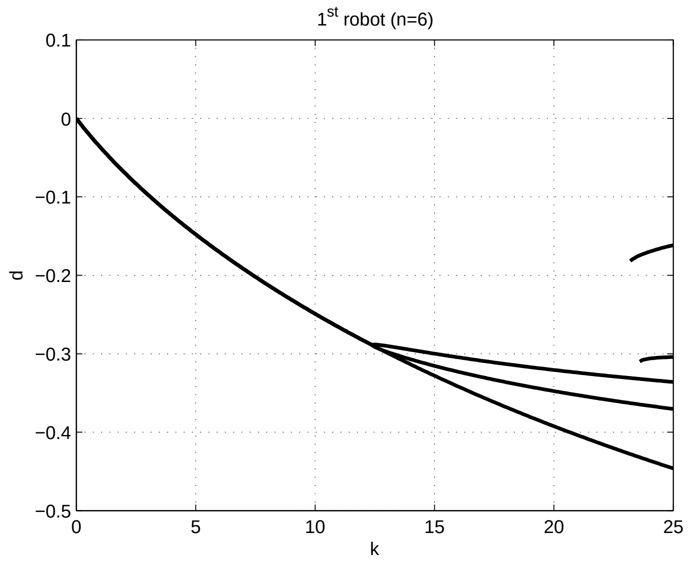

Figure 4.10. Bifurcation diagrams for robot one in a 6-robotic system.

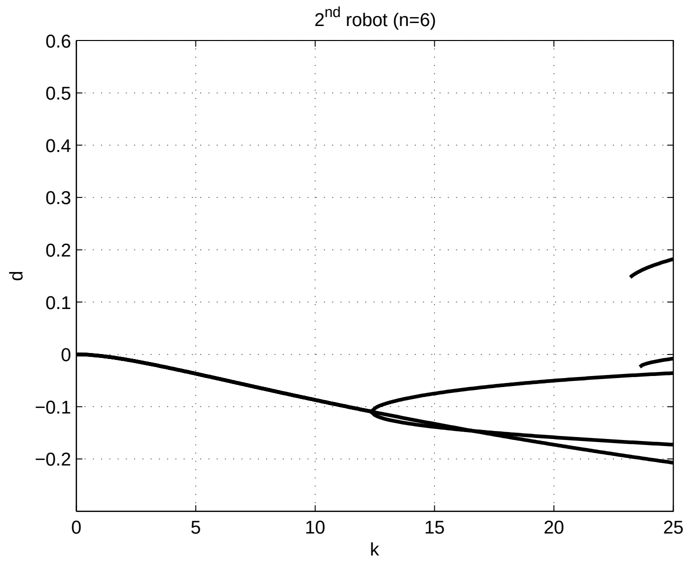

Figure 4.11. Bifurcation diagrams for robot two in a 6-robotic system.

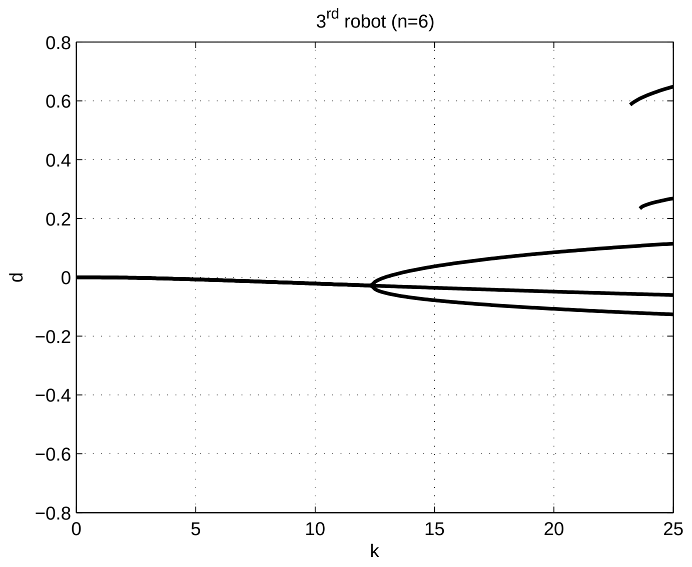

Figure 4.12. Bifurcation diagrams for robot three in a 6-robotic system.

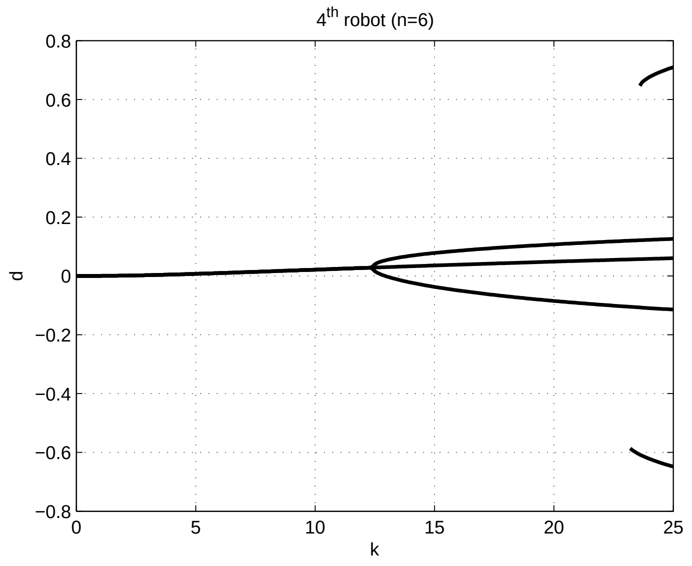

Figure 4.13. Bifurcation diagrams for robot four in a 6-robotic system.

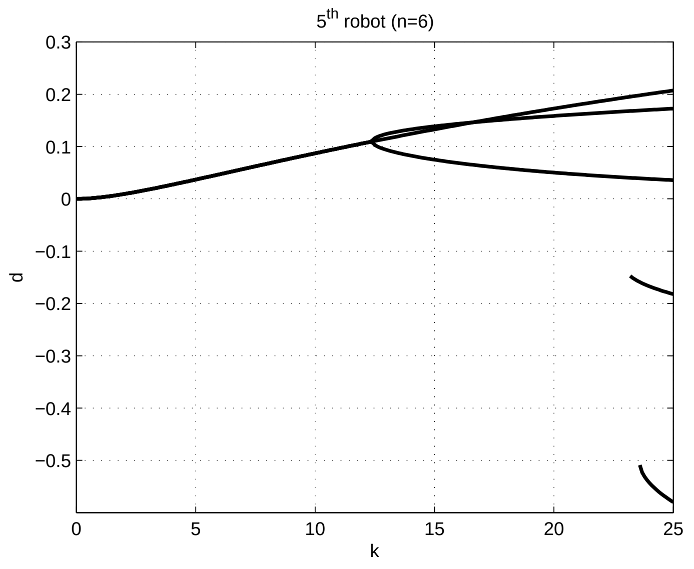

Figure 4.14. Bifurcation diagrams for robot five in a 6-robotic system.

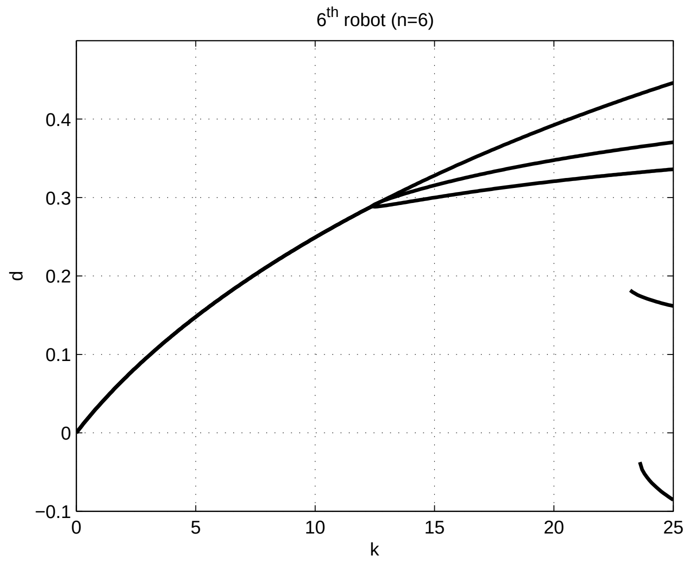

Figure 4.15. Bifurcation diagrams for robot six in a 6-robotic system.

4.1.3 Solutions for a Seven Robot System

Figures 4.16 through 4.24 illustrate similar results for a seven robot system. Figure 4.16 illustrates the trajectories when $k = 24.5$, $c = 4$ and $\overline{d} = 2$. Again, because the difference is hard to distinguish in this figure, Figure 4.17 illustrates the trajectory with the deviation from the nominal trajectory for the fifth robot magnified by a factor of five. Figure 4.18 through Figure 4.24 illustrate the bifurcation diagrams for the solutions versus k constructed in a manner identical to that of the system of five robots. The first bifurcation occurs near $k = 10.8$, the second occurs near $k = 16.1$ and the third occurs near $k = 20.6$. Observe that, similar to the five robot case, the bifurcation diagrams for robots 1 and 7 are symmetric to each other about $d = 0$ axis as is the bifurcation diagrams for robots 2 and 6 and robots 3 and 5, and the bifurcation diagram for robot 4 is symmetric to itself about $d = 0$ axis.

From the examples given above, we can find that when the number of robots in a distributed system is odd, the number of solutions is odd. A straight line connecting end points is one of trajectories of the robot located at the center of the system. Any other remaining trajectories of this robot has a symmetric one to the straight line solution. In one set of solution, two robots which have the same distance from the center one have symmetric trajectories about the trajectory of the center robot. When the number of robots in a distributed system is even, the number of solutions is even. No robot has straight line solution, but two robots which have the same distance from the center of the system still have symmetric solutions.

4.2 Optimization Software iSIGHT

In my Curricular Practical Training from October 2008 to April 2009, I utilized iSIGHT software to solve some optimization problem. It is a Multi-Disciplinary Opti-

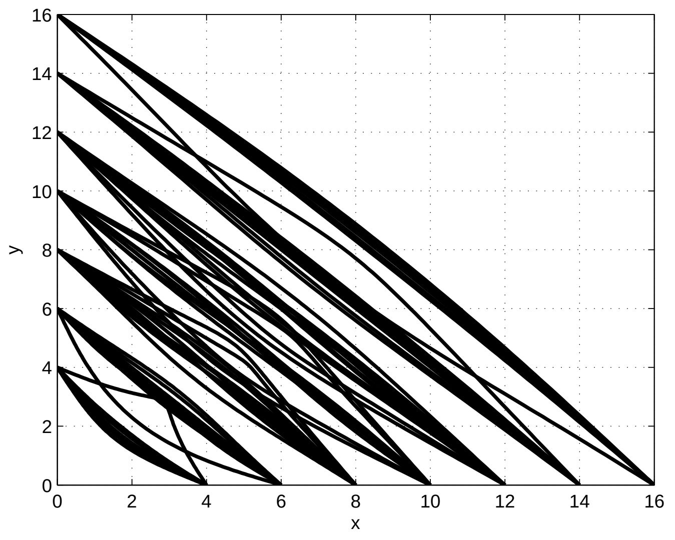

Figure 4.16. Optimal paths for a seven robot system with $k = 23$.

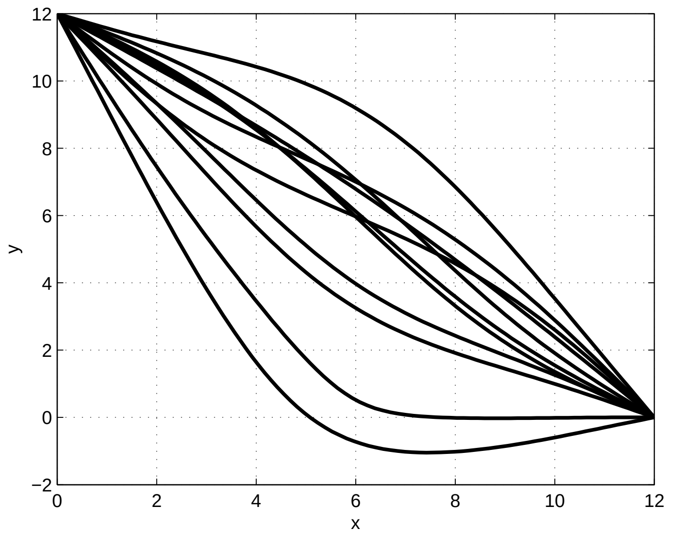

Figure 4.17. Difference among the optimal paths for robot four magnified by a factor of 5.

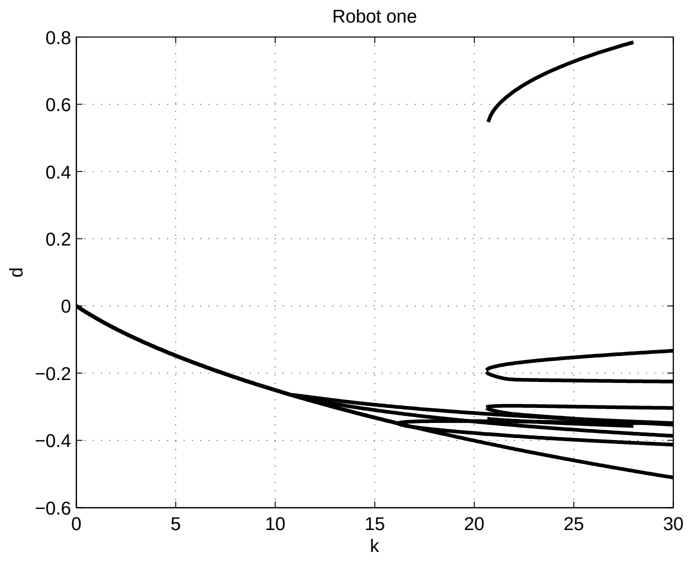

Figure 4.18. Bifurcation diagrams for robot one in a 7-robotic system.

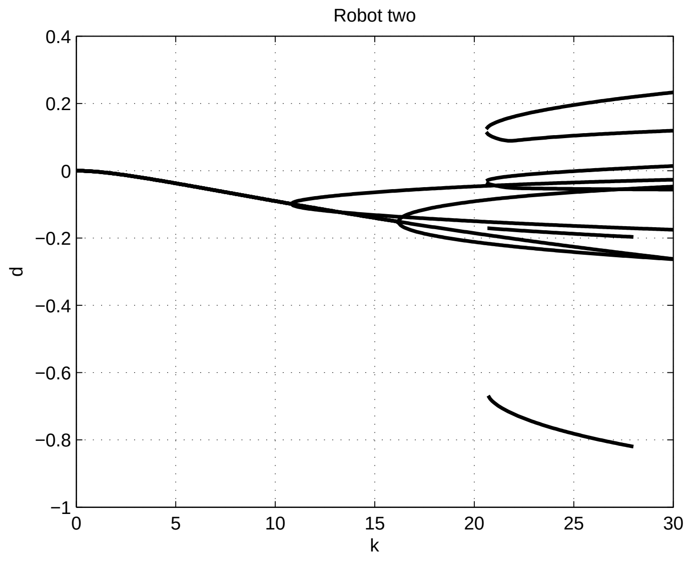

Figure 4.19. Bifurcation diagrams for robot two in a 7-robotic system.

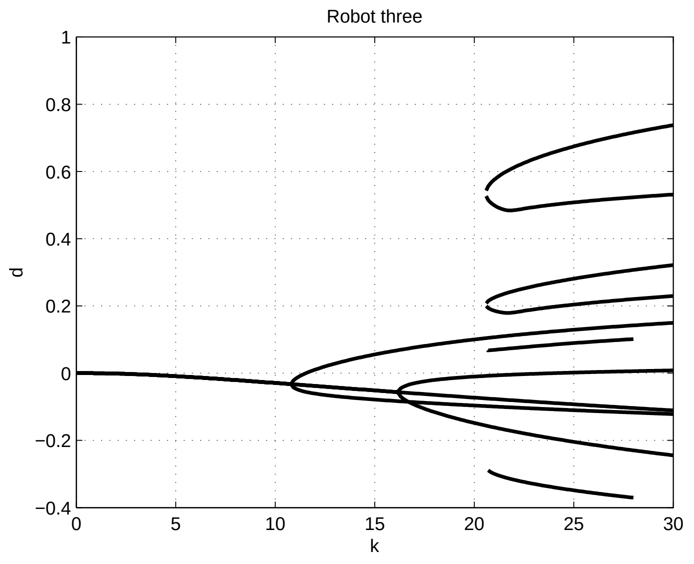

Figure 4.20. Bifurcation diagrams for robot three in a 7-robotic system.

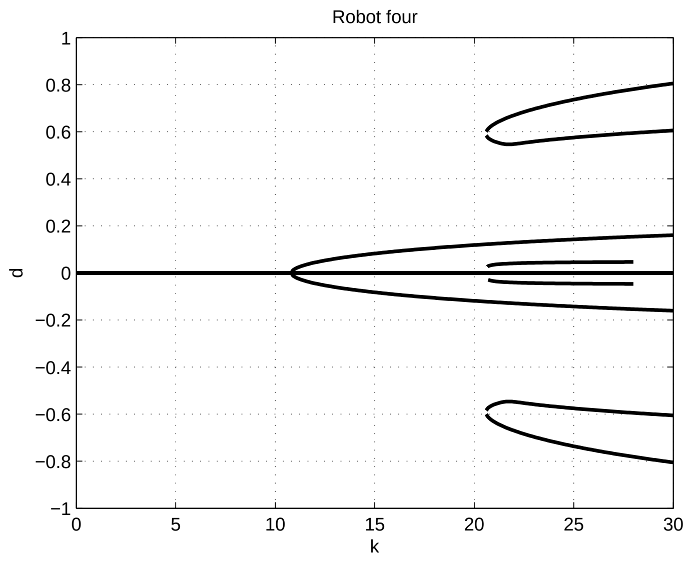

Figure 4.21. Bifurcation diagrams for robot four in a 7-robotic system.

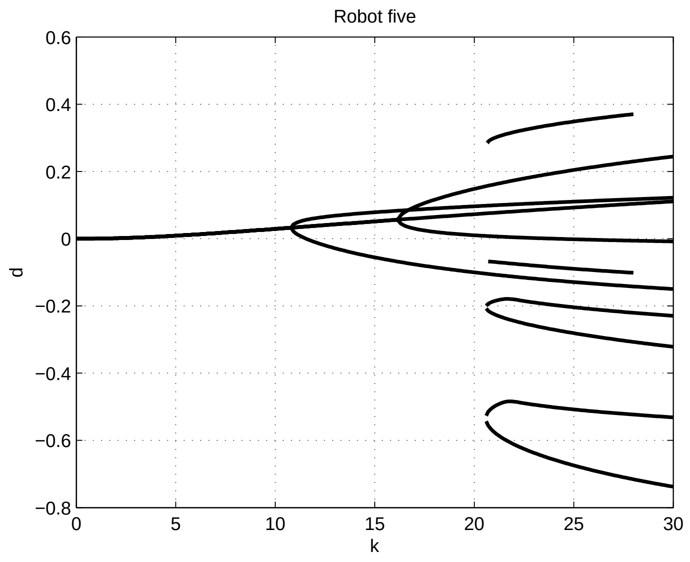

Figure 4.22. Bifurcation diagrams for robot five in a 7-robotic system.

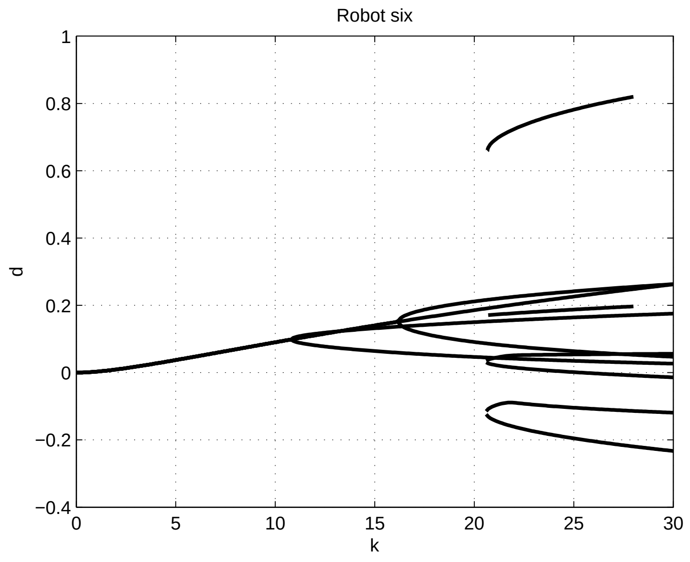

Figure 4.23. Bifurcation diagrams for robot six in a 7-robotic system.

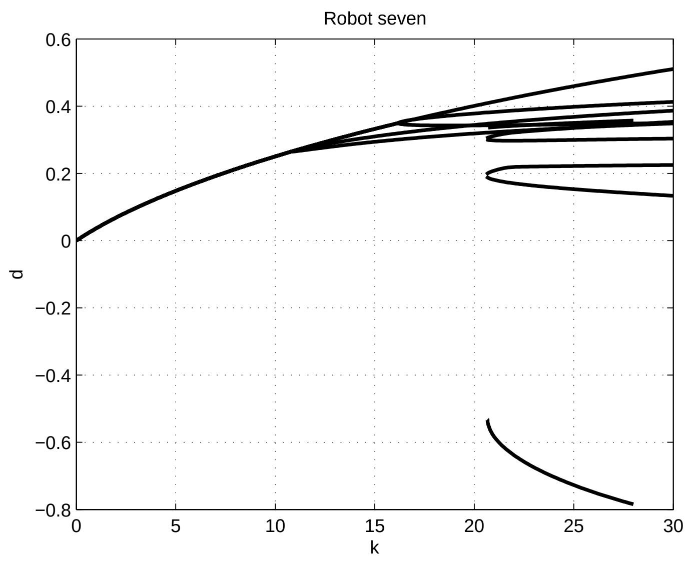

Figure 4.24. Bifurcation diagrams for robot seven in a 7-robotic system.

mization software developed by Engineous Software Inc. The key functions of iSIGHT are Automation, Integration and Optimization. We built the model (see Figure 4.25) by utilizing iSIGHT integrating MATLAB code to solve our optimization problem.

The Optimization contains design variables, constraints and objectives. Input parameters in the problem formulation is used as design variables. Lower/upper bounds specified on output parameters in the Problem Formulation will be used as lower/upper bounds for constraints. Any parameter that has an objective defined in the Problem Formulation will automatically be defined as an objective with the objective direction (minimize/maximize/target) as specified in the Problem Formulation. In our problem, the pre-defined initial and final positions of the robots, initial guesses of costates are considered to be design variables. But the positions are fixed, the values of initial guesses of costates are tunable. No constraint is defined for this problem, and objective is the distance between the calculated final position and the pre-defined final position.

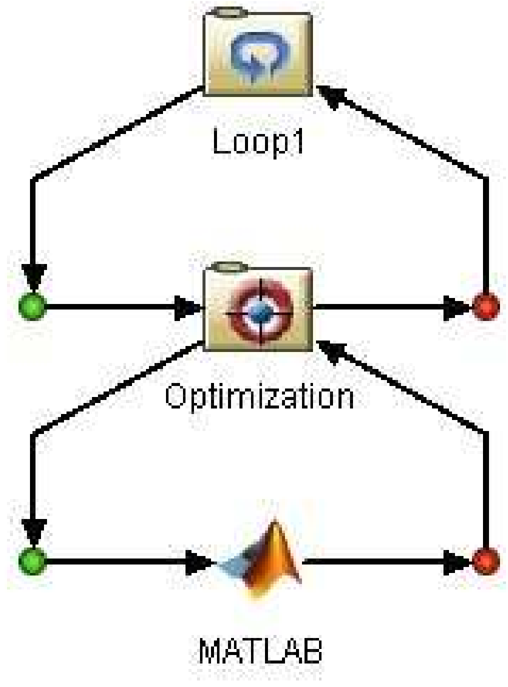

Figure 4.25. iSIGHT model layout.

The Loop component is a process component capable of executing subflows based on conditions. We use ”For” loop to execute subflows while continuously incrementing the value 0.1 of the parameter k from 0 to 30. MATLAB code is used to solve the motion equations. When the initial guesses are made for the costates, the boundary value problem is converted to initial value problem. And it is easy to solve numerically. MATLAB calculates the value of the objective. Optimization component will tune the initial guesses of the costates based on the chosen optimization method. It will stop until the optimal objective is found. The ”For” loop will let the optimization component repeat the process for different value of k.

We have an example for solving a five robot system. For $k = 24.5$, $c = 6$ and $\overline{d} = 2$, the result is illustrated in Figure 4.26

4.3 Asymptotic Analysis

In the two cases of very small k and very large k, we may use an asymptotic expansion to investigate the effect of k on the number of solutions to the boundary value problem. As will be shown, this analysis is consistent with the existence of a unique solution for small values of k and many solutions for very large k, which is the pattern indicated in the numerical bifurcation results that show an increased number of bifurcations and an increased number of solutions as k gets large.

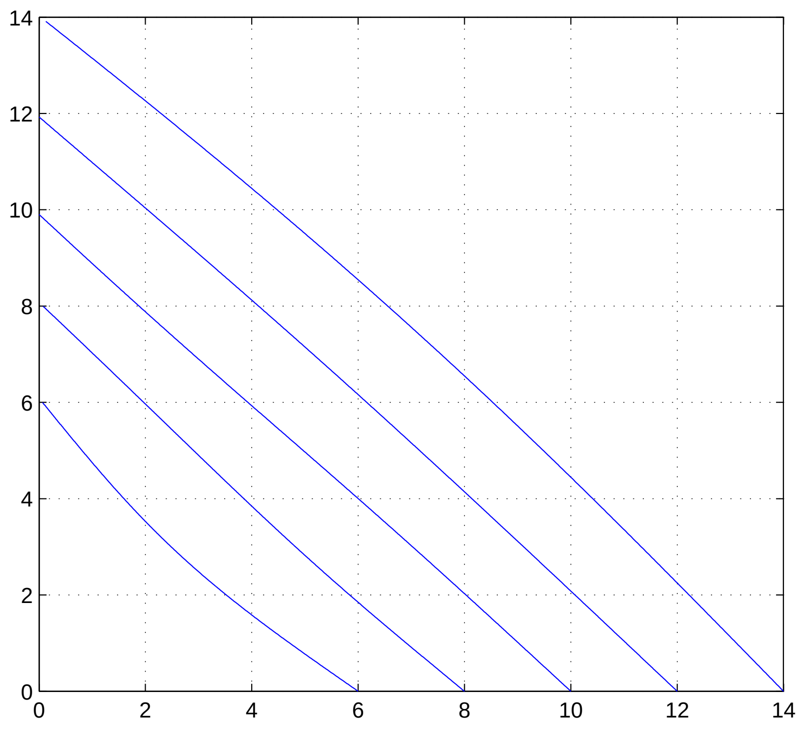

Figure 4.26. Optimal trajectories solved by iSIGHT.

4.3.1 Small k

We use a standard perturbation method (see [27]) to solve Equations 4.2 for $k \ll 1$. If we let

$$\begin{aligned} x_i &= x_{i,0} + kx_{i,1} + k^2 x_{i,2} + k^3 x_{i,3} + \cdots + k^j x_{i,j} + \cdots, \\ y_i &= y_{i,0} + ky_{i,1} + k^2 y_{i,2} + k^3 y_{i,3} + \cdots + k^j y_{i,j} + \cdots, \\ p_{1i} &= p_{1i,0} + kp_{1i,1} + k^2 p_{1i,2} + k^3 p_{1i,3} + \cdots + k_j p_{1i,j} + \cdots, \\ p_{2i} &= p_{2i,0} + kp_{2i,1} + k^2 p_{2i,2} + k^3 p_{2i,3} + \cdots + k_j p_{2i,j} + \cdots, \end{aligned}$$

and substitute into the equations of motion (Equation 4.2), a set of linear differential equations is obtained for each power of the expansion parameter k and we can consider it term-by-term in powers of k.

Specifically, if z represents either x or y, then the following table illustrates the resulting recursive structure of the equations. Any entry that is zero corresponds to a variable that is identically zero. Furthermore, as is the typical case in an asymptotic expansion, any variable only depends on lower order ones, which in this table correspond to variables to the left of it. Specifically, we have

$z_{i,0}$	$z_{i,1}$	$z_{i,2}$	$\cdots$	$z_{i,m-1}$	$z_{i,m}$	$\cdots$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$	$\cdots$	$z_{1,m-1}$	$z_{1,m}$	$\cdots$
$z_{2,0}$	0	$z_{2,2}$	$\cdots$	$z_{2,m-1}$	$z_{2,m}$	$\cdots$
$z_{3,0}$	0	0	$\cdots$	$z_{3,m-1}$	$z_{3,m}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\cdots$
$z_{m,0}$	0	0	$\cdots$	0	$z_{m,m}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\cdots$
$z_{n-2,0}$	0	0	$\cdots$	$-z_{3,m-1}$	$-z_{3,m}$	$\cdots$
$z_{n-1,0}$	0	$-z_{2,2}$	$\cdots$	$-z_{2,m-1}$	$-z_{2,m}$	$\cdots$
$z_{n,0}$	$-z_{1,1}$	$-z_{1,2}$	$\cdots$	$-z_{1,m-1}$	$-z_{1,m}$	$\cdots$

where m is the smallest integer larger than or equal to $\frac{n}{2}$. So, if $z_{i,i}$ is known and since $z_{i,j}$ ($j > i$) depends on $z_{i-1,j-2}, z_{i-1,j-1}, z_{i,j-2}, z_{i,j-1}, z_{i+1,j-2}, z_{i+1,j-1}$, we can solve them in the order of $j = i+1, i+2, \cdots$.

In detail, the $j = 0$ (k^0) terms gives the set of linear equations

$$\begin{aligned}
\dot{x}_{i,0} &= \frac{1}{2} p_{1i,0}, \\
\dot{y}_{i,0} &= \frac{1}{2} p_{2i,0}, \\
\dot{p}_{1i,0} &= 0, \\
\dot{p}_{2i,0} &= 0,
\end{aligned}$$

with boundary conditions

$$\begin{aligned} x_{i0}(0) &= x_{10}(0)+(i-1)\overline{d} \\ y_{i0}(0) &= 0 \\ x_{i0}(1) &= 0 \\ y_{i0}(1) &= y_{10}(1)+(i-1)\overline{d}, \end{aligned}$$

which have solutions

$$\begin{aligned} x_{i,0} &= -x_{i,0}(0)t+x_{i,0}(0), \\ y_{i,0} &= y_{i,0}(1)t, \\ p_{1i,0} &= -2x_{i,0}(0), \\ p_{2i,0} &= 2y_{i,0}(1). \end{aligned}$$

Naturally, these are straight lines, which is expected when the only component of the cost function is the control effort and the 0th order solution does not contain k.

In all cases (all powers of k and all robots), an analysis of the resulting expansion shows that $x_{i,j} = -x_{n+1-i,j}$ and $y_{i,j} = -y_{n+1-i,j}$. Also, for $1 \le j < i \le \frac{n+1}{2}$, $x_{i,j} = 0$ and $y_{i,j} = 0$ (the higher order terms for the "outer" robots are zero up to a certain order. Hence we only need to consider the cases where $1 \le i \le \frac{n}{2}$ and $i \le j$.

In the case where $j = i = 1$,

$$\begin{aligned}
\dot{x}_{1,1} &= \frac{1}{2} p_{11,1} \\
\dot{y}_{1,1} &= \frac{1}{2} p_{21,1} \\
\dot{p}_{11,1} &= 2\overline{d}(t-1)\left(1 - \frac{1}{\sqrt{2t^2 - 2t + 1}}\right) \\
\dot{p}_{21,1} &= -2\overline{d}t\left(1 - \frac{1}{\sqrt{2t^2 - 2t + 1}}\right).
\end{aligned}$$

Since the right hand sides of the last two equations are continuous and bounded functions of t on the interval $\mathscr{I} = [0,1]$, they are integrable and the integrals are differentiable (see [8]), which indicates the integrals are continuous. Hence $x_{1,1}$, $y_{1,1}$ exist and are unique since the right hand side of the p equations may be directly integrated twice to obtain the x and y solutions. Since we integrate twice, there are two undetermined constants, which can be determined by the two zero boundary conditions.

When $i = j$ and $j > 1$,

$$\begin{aligned}
\dot{x}_{i,j} &= \frac{1}{2} p_{1i,j} \\
\dot{y}_{i,j} &= \frac{1}{2} p_{2i,j} \\
\dot{p}_{1i,j} &= -2x_{i-1,j-1} + \frac{2t\left(-y_{i-1,j-1} + t(x_{i-1,j-1} + y_{i-1,j-1})\right)}{(2t^2 - 2t + 1)^{3/2}} \\
\dot{p}_{2i,j} &= 2y_{i-1,j-1} + \frac{2(t-1)\left(-y_{i-1,j-1} + t(x_{i-1,j-1} + y_{i-1,j-1})\right)}{(2t^2 - 2t + 1)^{3/2}}.
\end{aligned}$$

The right hand sides of the last two equations are the sum of integrable functions or product of them, so they are differentiable (see [8]). Similar to the argument for $x_{1,1}$ and $y_{1,1}$, $x_{i,i}$ and $y_{i,i}$ therefore exist and unique.

The off-diagonal terms have the same essential structure that the right hand side of the co-state equations is a linear combination of the lower order solutions in the expansion. Since all the lower order solutions are continuous and bounded functions of t, they may be directly integrated to compute the actual solution.

Since all the terms in the expansion may be solved by direction integration of functions that are continuous and bounded, a solution for each term exists. Hence, for $k \ll 1$, this asymptotic analysis give a computable construction for the solutions, and also indicates that the solution is unique. In other words, for small k, only one solution exists.

4.3.2 Large k

For large k ($\frac{1}{k} \ll 1$), a similar asymptotic expansion is used to solve Equations 4.1 but instead of k, $\varepsilon = \frac{1}{k}$ is used as the expansion parameter. Let

$$\begin{aligned}
x_i &= x_{i,0} + \varepsilon x_{i,1} + \varepsilon^2 x_{i,2} + \varepsilon^3 x_{i,3} + \cdots + \varepsilon^j x_{i,j} + \cdots, \\
y_i &= y_{i,0} + \varepsilon y_{i,1} + \varepsilon^2 y_{i,2} + \varepsilon^3 y_{i,3} + \cdots + \varepsilon^j y_{i,j} + \cdots, \\
p_{1i} &= p_{1i,0} + \varepsilon p_{1i,1} + \varepsilon^2 p_{1i,2} + \varepsilon^3 p_{1i,3} + \cdots + \varepsilon_j p_{1i,j} + \cdots, \\
p_{2i} &= p_{2i,0} + \varepsilon p_{2i,1} + \varepsilon^2 p_{2i,2} + \varepsilon^3 p_{2i,3} + \cdots + \varepsilon_j p_{2i,j} + \cdots.
\end{aligned}$$

We obtain the following equation for leading order of ε,

$$\begin{aligned}
\dot{x}_{i,0} &= \frac{1}{2} p_{1i,0} \\
\dot{y}_{i,0} &= \frac{1}{2} p_{2i,0} \\
0 &= \frac{2k\left(x_{i,0} - x_{i-1,0}\right)\left(d_{i-1,0} - \overline{d}\right)}{d_{i-1,0}} + \frac{2k\left(x_{i,0} - x_{i+1,0}\right)\left(d_{i,0} - \overline{d}\right)}{d_{i,0}} \\
0 &= \frac{2k\left(y_{i,0} - y_{i-1,0}\right)\left(d_{i-1,0} - \overline{d}\right)}{d_{i-1,0}} + \frac{2k\left(y_{i,0} - y_{i+1,0}\right)\left(d_{i,0} - \overline{d}\right)}{d_{i0}}.
\end{aligned}$$

The last two equations may be simplified to

$$(x_{i,0} - x_{i-1,0})^2 + (y_{i,0} - y_{i-1,0})^2 = \overline{d}^2, \qquad (4.4)$$

which transparently shows that the limit for large k simply requires that the distance constraint be exactly maintained.

Since the third and fourth equations are algebraic (as is Equation 4.4), then the costates, p are unconstrained and hence *any* path that maintains the desired distance between the robots and satisfies the boundary conditions is a solution. This makes intuitive sense: in the limit as $k \to \infty$, the control effort becomes negligible relative to the distance constraint. Hence, in the limit of very large k, the asymptotic analysis indicates that there is an infinite number of solutions. As long as the separation distance is maintained and the boundary conditions are satisfied, any path is optimal.

4.4 Symmetries in the Bifurcation Diagrams

This section proves that the symmetries found in the numerically-constructed bifurcation diagrams must be present. This is of practical value because it reduces the computation time necessary in a search over multiple solutions since a second solution can always be found from any solution that is obtained (unless the solution is symmetric with itself).

Suppose $(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n)$ is a solution of Equation 4.1 with the boundary conditions in Equation 4.3, and let

$$\begin{aligned} x_i &= x_{s_i} + x_{d_i}, \\ y_i &= y_{s_i} + y_{d_i}, \end{aligned}$$

where

$$\begin{aligned} x_{s_i} &= (c+(i-1)\overline{d})(1-t), \\ y_{s_i} &= (c+(i-1)\overline{d})t. \end{aligned}$$

The subscripts s indicate a “straight-line” solution and the subscripts d indicate the component of the solution that is a “deviation” from the straight line. If $v(t) = (x_{d_1}, y_{d_1}, \cdots, x_{d_n}, y_{d_n})$, then x_{d_i}, y_{d_i}, $i = 1, 2, \cdots, n$, satisfy the following equations with homogeneous boundary conditions:

$$\begin{aligned} -\ddot{x}_{d_i}(t) &= f_i(v(t)), \\ -\ddot{y}_{d_i}(t) &= g_i(v(t)), \end{aligned} \tag{4.5}$$

where $f_1 = h_1$, $g_1 = l_1$, $f_n = -h_{n-1}$, $g_n = -l_{n-1}$, and for $i = (2, 3, \cdots, n-1)$

$$\begin{aligned} f_i &= h_i - h_{i-1}, \\ g_i &= l_i - l_{i-1} \end{aligned}$$

where, for all $i = (1, 2, \cdots, n)$

$$\begin{aligned} h_i &= \left(\frac{\overline{d}}{d_i} - 1\right)\left(-\overline{d} + \overline{d}t + x_{d_i} - x_{d_{i+1}}\right), \\ l_i &= \left(\frac{\overline{d}}{d_i} - 1\right)\left(-\overline{d}t + y_{d_i} - y_{d_{i+1}}\right), \\ d_i &= \left(\left(-\overline{d} + \overline{d}t + x_{d_i} - x_{d_{i+1}}\right)^2 + \left(-\overline{d}t + y_{d_i} - y_{d_{i+1}}\right)^2\right)^{\frac{1}{2}} \end{aligned}$$

The system (4.5), is equivalent to the system of integral equations

$$\begin{aligned} x_{d_i} &= \int_0^1 G(t,s) f_i(v(s))ds, \\ y_{d_i} &= \int_0^1 G(t,s) g_i(v(s))ds, \end{aligned} \tag{4.6}$$

where $G(t,s)$ is the Green's function of the differential operator $-\ddot{u} = 0$ with homogeneous boundary conditions, where $u = x_{d_i}$ or $u = y_{d_i}$, and

$$G(t,s) = \begin{cases} t(1-s), & t \leq s \\ s(1-t), & t > s \end{cases}.$$

If A_i, B_i and F are maps such that

$$\begin{aligned} A_i v(t) &= k\int_0^1 G(t,s) f_i(v(s))ds, \\ B_i v(t) &= k\int_0^1 G(t,s) g_i(v(s))ds, \\ Fv(t) &= (A_1(v)(t), B_1(v)(t), \cdots, A_n(v)(t), B_n(v)(t), \end{aligned}$$

then determining a solution to Equation (4.6) is equivalent to finding a fixed point to equation

$$Fv(t) = v(t). \tag{4.7}$$

The following proposition proves that if a solution is known, then the "opposite" deviation from the straight-line solution is also a solution for the robot on the other side of the formation.

Proposition 4.4.1: Suppose $v(t)$ is a fixed point of Equation 4.7. Let

$$\begin{aligned}\hat{x}_{d_{n+1-i}} &= -x_{d_i} \\ \hat{y}_{d_{n+1-i}} &= -y_{d_i}\end{aligned} \tag{4.8}$$

and $\hat{v}(t) = (\hat{x}_{d_1}, \hat{y}_{d_1}, \cdots, \hat{x}_{d_n}, \hat{y}_{d_n})$, then $\hat{v}(t)$ is also a fixed point of Equation 4.7

Proof: The proof is by direct substitution. Substituting for the definition of the hat terms for each gives:

$$\begin{aligned}
d_i &= \sqrt{\left(-\overline{d}+\overline{d}t+x_{d_i}-x_{d_{i+1}}\right)^2+\left(-\overline{d}t+y_{d_i}-y_{d_{i+1}}\right)^2} \\
&= \sqrt{\left(-\overline{d}+\overline{d}t-\hat{x}_{d_{n+1-i}}+\hat{x}_{d_{n-i}}\right)^2+\left(-\overline{d}t-\hat{y}_{d_{n+1-i}}+\hat{y}_{d_{n-i}}\right)^2} \\
&= \sqrt{\left(-\overline{d}+\overline{d}t+\hat{x}_{d_{n-i}}-\hat{x}_{d_{n-i+1}}\right)^2+\left(-\overline{d}t+\hat{y}_{d_{n-i}}-\hat{y}_{d_{n-i+1}}\right)^2} \\
&= \hat{d}_{n-i} \\
h_i &= \left(\frac{\overline{d}}{d_i}-1\right)\left(-\overline{d}+\overline{d}t+x_{d_i}-x_{d_{i+1}}\right) \\
&= \left(\frac{\overline{d}}{\hat{d}_{n-i}}-1\right)\left(-\overline{d}+\overline{d}t-\hat{x}_{d_{n+1-i}}+\hat{x}_{d_{n-i}}\right) \\
&= \left(\frac{\overline{d}}{\hat{d}_{n-i}}-1\right)\left(-\overline{d}+\overline{d}t+\hat{x}_{d_{n-i}}-\hat{x}_{d_{n-i+1}}\right) \\
&= \hat{h}_{n-i} \\
l_i &= \left(\frac{\overline{d}}{d_i}-1\right)\left(-\overline{d}t+(y_d)_i-(y_d)_{i+1}\right) \\
&= \left(\frac{\overline{d}}{\hat{d}_{n-i}}-1\right)\left(-\overline{d}t-\hat{y}_{d_{n+1-i}}+\hat{y}_{d_{n-i}}\right) \\
&= \left(\frac{\overline{d}}{\hat{d}_{n-i}}-1\right)\left(-\overline{d}t+\hat{y}_{d_{n-i}}-\hat{y}_{d_{n-i+1}}\right) \\
&= \hat{l}_{n-i}
\end{aligned}$$

and

$$\begin{aligned}
f_1 &= h_1 = \hat{h}_{n-1} = -\hat{f}_n \\
g_1 &= l_1 = \hat{l}_{n-1} = -\hat{g}_n \\
f_i &= h_i - h_{i-1} = \hat{h}_{n-i} - \hat{h}_{n+1-i} = -\hat{f}_{n+1-i} \\
g_i &= l_i - l_{i-1} = \hat{l}_{n-i} - \hat{h}_{n+1-i} = -\hat{g}_{n+1-i} \\
f_n &= -h_{n-1} = -\hat{h}_1 = -\hat{f}_1 \\
g_n &= -l_{n-1} = -\hat{l}_1 = -\hat{g}_1
\end{aligned}$$

which give us

$$f_i = -\hat{f}_{n+1-i},$$
$$g_i = -\hat{g}_{n+1-i}.$$

for all *i* from 0 to *n*. Then

$$\hat{x}_{d_i} = -x_{d_{n+1-i}} = -\int_0^1 G(t,s) f_{n+1-i} ds = \int_0^1 G(t,s) \hat{f}_i ds$$
$$\hat{y}_{d_i} = -y_{d_{n+1-i}} = -\int_0^1 G(t,s) g_{n+1-i} ds = \int_0^1 G(t,s) \hat{g}_i ds.$$

Hence $\hat{v}(t) = (\hat{x}_{d_1}, \hat{y}_{d_1}, \cdots, \hat{x}_{d_n}, \hat{y}_{d_n})$ is a solution of Equation 4.7.

Equation 4.8 gives an algebraic expression for the symmetric solutions, which is useful because the theorem proves they satisfy the boundary value problems and hence reduces the computational burden of determining additional solutions. Note that the relationship is not simply the opposite deviation from the straight line solution, but is the opposite deviation from the straight line for a different robot.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This research considers the optimal control problem for a group of autonomous mobile robots operating in a 2-dimensional obstacle-free environment. The trajectory of each robot is optimized with respect to a combination of the control effort and the deviation from a desired formation, which in this thesis is simply a formation that maintains a specified distance between adjacent robots.

The first part of the results is for a system consisting of multiple MICAbots which are forming a regular polygon. We first present the symmetric properties of the cost function related to this problem, and then prove the symmetry in the optimal trajectories of the robots. The second part is for a group of unicycle-like autonomous mobiles in a straight line formation. We present numerical results illustrating the structure of bifurcations and multiple solutions of the second order nonlinear boundary value problem associated with the optimal control problem. Then we show that an asymptotic analysis indicates that there is a unique solution when k is small and in the limit as k approaches infinity, the number of solutions also approaches infinity. Then, it presents a theoretical result relating to the existence of symmetric solutions. It guarantees that for any solution, a corresponding symmetric solution exists. The practical benefit is that if a solution found numerically, the symmetric solution can be computed from that algebraically. Also, if a gradient-based search method is used, understanding of the structure of the relationship among multiple solutions is necessary to find the desired result. Finding multiple solutions may be desirable if the cost function does not include all the optimization criteria; for example, if obstacles are present but not accounted for in the cost function.

Since common numerical solution techniques such as the shooting method are local in nature and hence are difficult to use to find multiple solutions, an alternative formulation of the problem will be used in the future that can be solved through homotopy methods for polynomial systems. The key point of this method is to rewrite the differential equations of motion into finite difference formulation, and finally convert the differential equations to polynomials. A solver, Bertini [5], could be used to solve those polynomials. Bertini uses homotopy continuation methods to find the roots of polynomial system and also is capable of preprocessing the systems to reduce the possible number of solutions based on system symmetries. This approach might find branches that were were perhaps simply not found in our search using the shooting method or solutions that are not numerically stable using the shooting method.

From our bifurcation diagrams, we found that the bifurcation happened early when the number of robots in a system increased and we plotted the value of weighting parameter k at the first bifurcation point versus the number of robots in a system (see Figure 5.1). It is difficult to construct bifurcation diagram for high dimensional system, and I did not find any bifurcation point for a system containing less than five robots for a limited range of k (≤ 25). It will be a interested topic to investigate the relationship between the value of k at the first bifurcation point with the number of robots in a system.

An additional focus of future efforts is to generalize the results. The results presented in thesis thesis that are specific to the system studied are likely to be much more general than the particular case presented in the thesis. Determining the most general

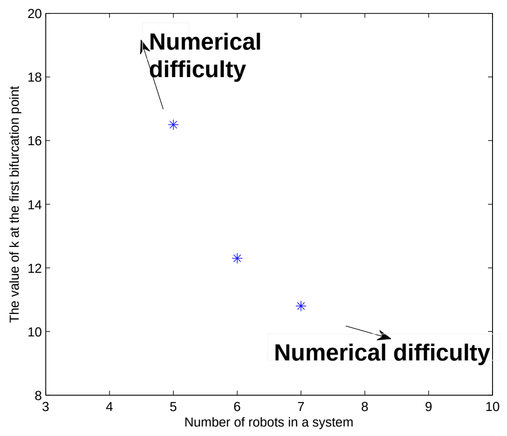

Figure 5.1. First bifurcation point versus the number of robots in a system.

classes of robots and formations that maintain the symmetry properties of the results and similar bifurcation structure is of interest. Also, the asymptotic analysis is only of any use for the limiting values for k. Determining conditions for the existence of a bifurcation for any value of k, similar to that for initial value problems, would be useful.

APPENDIX A

RELAXATION SHOOTING METHOD PROGRAM

This appendix provides the program used to solve the highly nonlinear equations of motion of the system.

```
#include "robot.h"


int main()
{double maxvalue(double array[], int);
 FILE *fid, *fid1;
 char filename[400];
 int i,j,ip,im,J,n,k,row,column,ii;
 double r=RADIUS, b=WIDTH, T_0=0.0, T_f=TFINAL, h, d_bar;
 double K1=1,K2,Psi;
 double  t[Nstep+1];
 double **A, **Phi,**phi, **inv_phi,**product_ptr, **d_x_f;
 double d_p_0[3*N];
 /* initial conditions --- starting point */
 double x_0[N], y_0[N], theta_0[N];
 /* initial guesses of Lagrange multipliers */
 double p1_0[N]={1,0.5,1},p2_0[N]={1,0.5,1},p3_0[N]={0.5,0.5,0.5};
```

```
/* end point */

double x_f[N],y_f[N],theta_f[N];

double X[N],Y[N],Theta[N],P1[N],P2[N],P3[N],D[N];

double X_inspace[N][M+1],Y_inspace[N][M+1],Theta_inspace[N][M+1];

double X_f[N][M+1],Y_f[N][M+1],Theta_f[N][M+1];

double x[N][Nstep+1],y[N][Nstep+1],theta[N][Nstep+1];

double p1[N][Nstep+1],p2[N][Nstep+1],p3[N][Nstep+1],d[N][Nstep+1];

double aa[N], ac[N], cc[N];

double err=1E-9,max;

double x6[6][Nstep+1],y6[6][Nstep+1],theta6[6][Nstep+1];

double p61[6][Nstep+1],p62[6][Nstep+1],p63[6][Nstep+1];

double d6[6][Nstep+1];

double intx,inty,intxy,inttheta;




  A=(double **)malloc((unsigned)(6*N)*sizeof(double));

  Phi=(double **)malloc((unsigned)(6*N)*sizeof(double));

  phi=(double **)malloc((unsigned)(3*N)*sizeof(double));

  inv_phi=(double **)malloc((unsigned)(3*N)*sizeof(double));

  d_x_f=(double **)malloc((unsigned)(3*N)*sizeof(double));

  product_ptr=(double **)malloc((unsigned)6*N*sizeof(double));


  for(i=0;i<6*N;i++)

     product_ptr[i]=(double *)malloc((unsigned)
                    6*N*sizeof(double));


  for(i=0;i< (6*N);i++) {
    A[i]=double *)malloc((unsigned)(6*N)*sizeof(double));
    Phi[i]=(double *)malloc((unsigned)(6*N)*sizeof(double));
  }
  for(i=0;i<(3*N);i++) {
    phi[i]=(double *)malloc((unsigned)(3*N)*sizeof(double));
    inv_phi[i]=(double *)malloc((unsigned)(3*N)*sizeof(double));
    d_x_f[i]=(double *)malloc((unsigned) 1*sizeof(double));
  }


 h=(T_f-T_0)/Nstep;
 for(i=0;i<=Nstep;i++)
    t[i]=T_0+h*i;
 d_bar=sqrt(pow(2*R*sin(PI/N),2));
 Psi=PI/2.0;


 for(i=0;i<N;i++){
     x_0[i]=XX0+R*cos(2*PI*i/N);
     y_0[i]=YY0+R*sin(2*PI*i/N);
     theta_0[i]=atan2(Psi*(x_0[i]-XX0)+(YYf-YY0),
                     -Psi*(y_0[i]-YY0)+(XXf-XX0));
     theta_0[i]=fmod(theta_0[i],2*PI);
```

```
      x_f[i]=XXf+R*cos(2*PI*i/N+Psi);

      y_f[i]=YYf+R*sin(2*PI*i/N+Psi);

      theta_f[i]=atan2(Psi*(x_0[i]-XX0)*cos(Psi)

                -Psi*(y_0[i]-YY0)*sin(Psi)+(YYf-YY0),

                -Psi*(x_0[i]-XX0)*sin(Psi)

                -Psi*(y_0[i]-YY0)*cos(Psi)+(XXf-XX0));

      theta_f[i]=fmod(theta_f[i],2*PI);

 }


/*Search the 1st point I shoot:*/

 for(i=0;i<N;i++){

     X[i]=x_0[i];

     Y[i]=y_0[i];

     Theta[i]=theta_0[i];

     Lambda1[i]=lambda1_0[i];

     Lambda2[i]=lambda2_0[i];

     Lambda3[i]=lambda3_0[i];}

 for(i=0;i<N;i++)

     D[i]=sqrt(pow(2*R*sin(PI/N),2));


 for(n=0;n<Nstep;n++) {

    for(i=0;i<N;i++){

        X[i]+=h*(r*r/4/K1*(Lambda1[i]*cos(Theta[i])

                 +Lambda2[i]*sin(Theta[i]))*cos(Theta[i]));

        Y[i]+=h*(r*r/4/K1*(Lambda1[i]*cos(Theta[i])
                 +Lambda2[i]*sin(Theta[i]))*sin(Theta[i]));
        Theta[i]+=h*(r*r/(4*b*b*K1))*Lambda3[i];
        Theta[i]=fmod(Theta[i],2*PI);
        if(i==0)
          im=N-1;
        else
          im=i-1;
        if(i==N-1)
          ip=0;
        else
          ip=i+1;
        Lambda1[i]+=h*2*K2*((X[i]-X[im])*(D[im]-d_bar)/D[im]
                    +(X[i]-X[ip])*(D[i]-d_bar)/D[i]);
        Lambda2[i]+=h*2*K2*((Y[i]-Y[im])*(D[im]-d_bar)/D[im]
                    +(Y[i]-Y[ip])*(D[i]-d_bar)/D[i]);
        Lambda3[i]+=h*(r*r/4/K1*(Lambda1[i]*cos(Theta[i])
                    +Lambda2[i]*sin(Theta[i]))
                    *(Lambda1[i]*sin(Theta[i])
                    -Lambda2[i]*cos(Theta[i])));
     }

    for(i=0;i<N;i++) {
        if (i==N-1)  ip=0;
        else ip=i+1;
        D[i]=sqrt(pow(X[i]-X[ip],2)+pow(Y[i]-Y[ip],2));
    }
  }

for(i=0;i<N;i++){
   for(j=0;j<=M;j++){
       X_inspace[i][j]=X[i]+(x_f[i]-X[i])*j/M;
       Y_inspace[i][j]=Y[i]+(y_f[i]-Y[i])*j/M;
         Theta[i]=fmod(Theta[i],2*PI);
       if (sqrt(pow(Theta[i]-theta_f[i],2))<=PI)
          Theta_inspace[i][j]=Theta[i]
                              +(theta_f[i]-Theta[i])*j/M;
       else if (Theta[i]>theta_f[i])
          Theta_inspace[i][j]=Theta[i]
                              +(theta_f[i]+2*PI-Theta[i])*j/M;
       else
          Theta_inspace[i][j]=Theta[i]
                              +(theta_f[i]-2*PI-Theta[i])*j/M;
   }
}

for(j=0;j<=M;j++){
   for(i=0;i<N;i++){
       X_f[i][j]=X_inspace[i][j];
       Y_f[i][j]=Y_inspace[i][j];
       Theta_f[i][j]=Theta_inspace[i][j];
   }
}

for(k=0;k<=M;k++){
   sprintf(filename, "file%d.dat",k);
   fid=fopen(filename,"w");
   for(i=0;i<N;i++){
      x_f[i]=X_f[i][k];
      y_f[i]=Y_f[i][k];
      theta_f[i]=Theta_f[i][k];
   }



/* create these variables in advance in case of
*/breaking down in condition judging
   d_lambda_0[0]=1;
   max=0.1;
   J=0;

   while(max>err){
       system("date");
       for(i=0;i< (6*N);i++) {
          for(j=0;j< (6*N);j++){
             A[i][j]=0.0;
          if(i==j)
             Phi[i][j]=1.0;
          else
             Phi[i][j]=0.0;}
          }

         for(i=0;i<N;i++){
             x[i][0]=x_0[i];
             y[i][0]=y_0[i];
             theta[i][0]=theta_0[i];
             lambda1[i][0]=lambda1_0[i];
             lambda2[i][0]=lambda2_0[i];
             lambda3[i][0]=lambda3_0[i];
         }

     for(i=0;i<N;i++){
        if (i==N-1)
           ip=0;
        else
           ip=i+1;
        d[i][0]=R;
         }

/*  Integration process */
    for(n=1;n<=Nstep;n++) {
```

```
for(j=0;j<N;j++){
   if (j==N-1)
      ip=0;
   else
      ip=j+1;
   aa[j]=2*K2*(d[j][n-1]-d_bar)/d[j][n-1]
          +2*K2*pow(x[j][n-1]-x[ip][n-1],2)
          *d_bar/pow(d[j][n-1],3);
    ac[j]=2*K2*(x[j][n-1]-x[ip][n-1])
          *(y[j][n-1]-y[ip][n-1])
          *d_bar/pow(d[j][n-1],3);
    cc[j]=2*K2*(d[j][n-1]-d_bar)/d[j][n-1]
          +2*K2*pow(y[j][n-1]-y[ip][n-1],2)
                         *d_bar/pow(d[j][n-1],3);
 }

 for(i=0;i<N;i++){
    A[i][2*N+i]=r*r/4/K1*(-lambda1[i][n-1]
                *sin(2*theta[i][n-1])
                +lambda2[i][n-1]*cos(2*theta[i][n-1]));
    A[i][3*N+i]=r*r/4/K1*pow(cos(theta[i][n-1]),2);
    A[i][4*N+i]=r*r/4/K1*sin(theta[i][n-1])
                *cos(theta[i][n-1]);
    A[N+i][2*N+i]=r*r/4/K1*(lambda1[i][n-1]
                  *cos(2*theta[i][n-1])
```

```
              +lambda2[i][n-1]*sin(2*theta[i][n-1]));
A[N+i][3*N+i]=r*r/4/K1*sin(theta[i][n-1])
              *cos(theta[i][n-1]);
A[N+i][4*N+i]=r*r/4/K1*pow(sin(theta[i][n-1]),2);
A[2*N+i][5*N+i]=r*r/(4*b*b*K1);

if(i==0)
   im=N-1;
else
   im=i-1;
if(i==N-1)
   ip=0;
else
   ip=i+1;

A[3*N+i][im]=-aa[im];
A[3*N+i][i]=aa[im]+aa[i];
A[3*N+i][ip]=-aa[i];
A[3*N+i][N+im]=-ac[im];
A[3*N+i][N+i]=ac[im]+ac[i];
A[3*N+i][N+ip]=-ac[i];
A[4*N+i][im]=-ac[im];
A[4*N+i][i]=ac[im]+ac[i];
A[4*N+i][ip]=-ac[i];
A[4*N+i][N+im]=-cc[im];
```

```
    A[4*N+i][N+i]=cc[im]+cc[i];

    A[4*N+i][N+ip]=-cc[i];

    A[5*N+i][2*N+i]=r*r/4/K1*(pow(lambda1[i][n-1]

       *cos(theta[i][n-1])+lambda2[i][n-1]

       *sin(theta[i][n-1]),2)-pow(lambda1[i][n-1]

       *sin(theta[i][n-1])-lambda2[i][n-1]

       *cos(theta[i][n-1]),2));

    A[5*N+i][3*N+i]=r*r/4/K1*(lambda1[i][n-1]

       *sin(2*theta[i][n-1])

       -lambda2[i][n-1]*cos(2*theta[i][n-1]));

    A[5*N+i][4*N+i]=-r*r/4/K1*(lambda1[i][n-1]

        *cos(2*theta[i][n-1])

        +lambda2[i][n-1]*sin(2*theta[i][n-1]));

 }


for(i=0;i<N;i++){

    x[i][n]=x[i][n-1]+h*(r*r/4/K1*(lambda1[i][n-1]

        *cos(theta[i][n-1])+lambda2[i][n-1]

        *sin(theta[i][n-1]))*cos(theta[i][n-1]));

    y[i][n]=y[i][n-1]+h*(r*r/4/K1*(lambda1[i][n-1]

        *cos(theta[i][n-1])+lambda2[i][n-1]

        *sin(theta[i][n-1]))*sin(theta[i][n-1]));

    theta[i][n]=theta[i][n-1]+h*(r*r/(4*b*b*K1))

        *lambda3[i][n-1];

    theta[i][n]=fmod(theta[i][n],2*PI);
    if(i==0)
       im=N-1;
    else
       im=i-1;
    if(i==N-1)
       ip=0;
    else
       ip=i+1;

    lambda1[i][n]=lambda1[i][n-1]+h*2*K2*((x[i][n-1]
          -x[im][n-1])*(d[im][n-1]-d_bar)/d[im][n-1]
          +(x[i][n-1]-x[ip][n-1])
          *(d[i][n-1]-d_bar)/d[i][n-1]);
    lambda2[i][n]=lambda2[i][n-1]+h*2*K2*((y[i][n-1]
          -y[im][n-1])*(d[im][n-1]-d_bar)/d[im][n-1]
          +(y[i][n-1]-y[ip][n-1])
          *(d[i][n-1]-d_bar)/d[i][n-1]);
    lambda3[i][n]=lambda3[i][n-1]+h*(r*r/4/K1
          *(lambda1[i][n-1]*cos(theta[i][n-1])
          +lambda2[i][n-1]*sin(theta[i][n-1]))
          *(lambda1[i][n-1]*sin(theta[i][n-1])
          -lambda2[i][n-1]*cos(theta[i][n-1])));
     }

for(i=0;i<N;i++) {
     if (i==N-1)
        ip=0;
     else
        ip=i+1;
     d[i][n]=sqrt(pow(x[i][n-1]-x[ip][n-1],2)
             +pow(y[i][n-1]-y[ip][n-1],2));
   }

   product_ptr=matrixmultiply(A,6*N,6*N,Phi,6*N,
               6*N,product_ptr);

   for(row=0;row<6*N;row++){
     for(column=0;column<6*N;column++){
           Phi[row][column]+=h*product_ptr[row][column];
     }
   }
 }  // n loop

 for(row=0;row<3*N;row++){
  for(column=0;column<3*N;column++){
      phi[row][column]=Phi[row][column+3*N];
   }
 }

for(i=0;i<N;i++){
    d_x_f[i][0]=x_f[i]-x[i][Nstep];
    d_x_f[N+i][0]=y_f[i]-y[i][Nstep];
    d_x_f[2*N+i][0]=theta_f[i]-theta[i][Nstep];
}

matrixinverse(phi,inv_phi,3*N);
product_ptr=matrixmultiply(inv_phi,3*N,3*N,d_x_f,3*N,
            1,product_ptr);

for(row=0;row<3*N;row++)
   d_lambda_0[row]=product_ptr[row][0];

for(i=0;i<N;i++){
    lambda1_0[i]+=d_lambda_0[i];
    lambda2_0[i]+=d_lambda_0[i+N];
    lambda3_0[i]+=d_lambda_0[i+2*N];
}

J+=1;
max=0.0;
for(i=0;i<N;i++){
   max+=sqrt(pow(x[i][Nstep]-X_f[i][k],2)
        +pow(y[i][Nstep]-Y_f[i][k],2)
        +pow(theta[i][Nstep]-Theta_f[i][k],2));
}
```

```
   } // while


    for(i=0;i<=Nstep;i++){
      fprintf(fid,"%13.9f ",t[i]);
      for(j=0;j<N;j++) {
         fprintf(fid,"%18.13f  %18.13f  %18.13f  %18.13f %18.13f
              %18.13f %18.13f",x[j][i],y[j][i],theta[j][i],
              lambda1[j][i],lambda2[j][i],lambda3[j][i],d[j][i]);}
         fprintf(fid,"\n");
         }
     fclose(fid);
  } // k loop
  return(0);
}

double maxvalue(double array[ ],int n) { int j;
  double  max;
  max=0;
  for(j=0;j<n;j++)
     if(sqrt(pow(array[j],2))>max) max=sqrt(pow(array[j],2));

  return(max);
}
```

```
/* This file is "matrixinverse.c".
 *
 * This program computes the inverse of a matrix using Gauss-Jordan
 * elimination.  Row shifting is only utilized if a diagonal element
 * of the original matrix to be inverted has a magnitude less than
 * DIAGONAL_EPS, which is set in "inversekinematics.h".
 *
 * The inverse matrix is stored in y[][], and a pointer to y is
 * returned.
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */

#include "robot.h"

double **matrixinverse(double **a, double **y, int n) {
  double temp,coef;
  double  max;
  int max_row;
  int i,j,k;

  /* Initialize y[][] to be the identity element. */
  for(i=0;i<n;i++) {
    for(j=0;j<n;j++) {
```

```
    if(i==j)
      y[i][j] = 1;
    else
      y[i][j] = 0;
  }
}

/* Gauss-Jordan elimination with selective initial pivoting */

/* Check the magnitude of the diagonal elements, and if one
 * is less that DIAGONAL_EPS, the search for an element lower
*/ in the same column with a larger magnitude.

for(i=0;i<n;i++) {
  if(fabs(a[i][i]) < DIAGONAL_EPS) {
    max = a[i][i];
    max_row = i;
    for(j=i;j<n;j++) {
  if(fabs(a[j][i]) > max) {
    max = fabs(a[j][i]);
    max_row = j;
  }
    }

    if(max < DIAGONAL_EPS) {
      printf("Ill-conditioned matrix encountered. Exiting...\n");
      exit(1);
    }

    /* This loop switches rows if needed. */
    for(k=0;k<n;k++) {
      temp = a[max_row][k];
      a[max_row][k] = a[i][k];
      a[i][k] = temp;

      temp = y[max_row][k];
      y[max_row][k] = y[i][k];
      y[i][k] = temp;
    }
  }
}

/* This is the forward reduction. */
for(i=0;i<n;i++) {

  coef = a[i][i];
  for(j=n-1;j>=0;j--) {
    y[i][j] /= coef;
    a[i][j] /= coef;
  }
  for(k=i+1;k<n;k++) {

    coef = a[k][i]/a[i][i];

    for(j=n-1;j>=0;j--) {

  y[k][j] -= coef*y[i][j];

  a[k][j] -= coef*a[i][j];

    }

  }

}



/* This is the back substitution. */

for(i=n-1;i>=0;i--) {


  for(k=i-1;k>=0;k--) {

    coef = a[k][i]/a[i][i];

    for(j=0;j<n;j++) {

  y[k][j] -= coef*y[i][j];

  a[k][j] -= coef*a[i][j];

    }

  }

}


// printf("%f\n",y);

return y;
```

```
}

#include "robot.h"

double **matrixmultiply(double** left, int row_left, int
column_left,
            double** right, int row_right, int column_right,
            double** product_ptr) {
  int i,j,k;

  if ( column_left != row_right ) {
    printf("\n The matrices cannot be multiplied! Exiting...\n");
    exit(1);
  }

  for(i=0;i<row_left;i++)
    for(j=0;j<column_right;j++)
      for(k=0,product_ptr[i][j]=0.0;k<column_left;k++)
    product_ptr[i][j] += left[i][k]*right[k][j];

  return product_ptr;
}
```

The following C-code is named ”robot.h” which defines the constants and declares functions used by the main code.

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>


#define RADIUS 3.0

#define WIDTH  1.0

#define GAIN 0.01

#define TFINAL 1

#define N 3

#define M 10

#define Nstep  5000

#define PI 4*atan(1.0)


#define R 1.

#define XX0 -2.

#define YY0 0.   //Initial coordinate of the center

#define XXf 0.

#define YYf 2.    //Final coordinate of the center



#define MAX_ITERATIONS 1000

#define EPS 0.0000000001

#define PERTURBATION 0.001

#define DIAGONAL_EPS 0.0001
```

```
double** matrixinverse(double **J, double **Jinv, int n); double**
matrixmultiply(double** left,int row_left,int column_left,
            double** right,int row_right,int column_right,
            double** product_ptr);
```

BIBLIOGRAPHY

1. L. Aguilar, R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Ten autonomous mobile robots (and even more) in a route network like environment. In *In Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 260–267, 1995.

2. T. Balch and R. Arkin. *IEEE Transactions on Robotics and Automation*, 14(6): 926–934, 1998.

3. T. Balch and M. Hybinette. Behavior-based coordination of large-scale robot formations. In *Fourth International Conference on MultiAgent Systems*, 2000.

4. T. D. Barfoot and C. M. Clark. *Robotics and Autonomous Systems*, 46(2), 2004.

5. D. Bates, J. Hauenstein, A. Sommese, and C. Wampler. `http://www.nd.edu/~sommese/bertini/`.

6. C. Belta and V. Kumar. *Geometric Methods for Multi-Robot optimal Motion Planning, Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics*. Springer–Verlag, Berlin, 2005.

7. F. Bullo and A. D. Lewis. *Geometric Control of Mechanical Systems*. Springer, 2005.

8. R. Carlson. *A Concrete Introduction to Real Analysis*. CRC, Boca Raton, 2006.

9. J. P. Desai. *Journal of Robotic Systems*, 19(11):511–525, 2002.

10. J. P. Desai. Modeling multiple teams of mobile robots: a graph theoretic approach. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots ans Systems*, pages 381–386, 2001.

11. J. P. Desai and V. Kumar. *Journal of Robotic Systems*, 10:557–579, 1999.

12. J. P. Desai, J. Ostrowski, and V. Kumar. Controlling formation of multiple mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 16–21, 1998.

13. J. P. Desai, J. P. Ostrowski, and V. Kumar. *IEEE Transzctons on Robotics and Automation*, 17(6), 2001.

14. L. E. Dubins. *American Journal of Mathematics*, 79:497–516, 1957.

15. M. Egerstedt and X. Hu. *IEEE Transactions on Robotics and Automation*, 17(6): 947–951, 2001.

16. L. Erbe and H. Wang. On the existence of positive solutions of ordinary differential equations. In *preceedings of the American Mathematical Society*, pages 743–748, 1994.

17. L. Erbe, S. Hu, and H. Wang. *Mathematical Analysis and Applications*, 184:640–648, 1994.

18. M. Erdmann and I. Lozano-Perez. On multiple moving objects. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1419–1424, 1986.

19. L. Fang and P. J. Antsaklis. Information consensus of asynchronous discrete-time multi-agent systems. In *Proceedings of ACC*, pages 1883–1888, 2005.

20. L. Fang, P. J. Antsaklis, and A. Tzimas. Asynchronous consensus protocols: preliminary results, simulations and open questions. In *Proceedings of 44th IEEE Conference on Decision and Control*, pages 2194–2199, 2005.

21. A. Frommer and D. B. Szyld. *Journel of Computational and Applied Math*, 123: 201–206, 2000.

22. T. Gross, Jonathan L. and T. W. *Topological Graph Theory*. Wiley Interscience series in Discrete Mathematics and Optimization, 1987.

23. D. Guo and V. Lakshmikantham. *Nonlinear Problems in Abstract Cones*. Academic Press, Orlando, FL, 1998.

24. Y. Guo and L. E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Proceedings of the 2002 IEEE Internatioanl Conference on Robotics and Automation*, pages 2612–2619, 2002.

25. J. Jennings, G. Whelan, and W. Evans. Cooperative search and rescue with a team of mobile robots. In *IEEE International Conference on Advanced Robotics*, pages 193–200, 1997.

26. K. Kant and S. Zucker. *International Journal of Robotics Research*, 5(3):72–89, 1986.

27. J. Kevorkian. *Perturbation Methods in Applied Mathematics*. Springer–Verlag, New York, 1981.

28. N. E. Leonard and E. Fiorelli. *In 40th IEEE Conference on Decision and Control*, pages 2968–2973, 2001.

29. M. A. Lewis and K. H. Tan. *Autonomous Robots*, 4:387–403, 1997.

30. R. Ma and B. Thompson. *Applied Mathematics Letters*, 18(5):587–595, 2005.

31. R. Ma and B. Thompson. *Nonlinear Analysis: Theory, Methods and Applications*, 303(2):726–735, 2005.

32. M. J. Mataric. *IEEE Intelligent Systems*, pages 6–8, 1998.

33. M. J. Mataric. *Cognitive Systems Research*, 2(1):81–93, 2001.

34. M. J. Mataric. *Autonomous Robots*, 4:73–83, 1997.

35. M. J. Mataric, M. Nilsson, and K. T. Simsarin. Cooperative multi-robot box-pushing. In *International Conference on Intelligent Robots and Systems*, pages 556–561, 1997.

36. C. R. McInnes. *AIAA Journal of Guidance Control and Dynamics*, 18(5):1215–1217, 1995.

37. M. McMickell. *Reduction and control of nonlinear symmetric distributed robotic systems*. PhD thesis, University of Notre Dame, 2003.

38. M. McMickell and B. Goodwine. *IEEE International Conference on Robotics and Automation*, pages 4228–4233, 2003.

39. M. McMickell, B. Goodwine, and L. Montestruque. *IEEE International Conference on Robotics and Automation*, 2:1600–1605, 2003.

40. Y. Naito and S. Tanaka. *Nonlinear Analysis*, 56(4):919–935, 2004.

41. P. J. Olver. *Applications of Lie Groups to Differential Equations*. Springer-Verlag, second edition, 1993.

42. G. Pereira, A. Das, V. Kumar, and M. Campos. Formation control with configuration space constraints. In *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems*, pages 2755–2760, 2003.

43. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*, chapter 17. Cambridge University, 1992.

44. H. Puttgen, P. MacGrego, and F. Lambert. *IEEE Power and Energy Magazine*, 1: 22–29, 2003.

45. J. A. Reeds and R. A. Shepp. *Pacific Journal of Mathematics*, 1990.

46. S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer, 1999.

47. R. W. Sharpe. *Differential Geometry*. Springer, 1997.

48. T. R. Smith, H. Hanssmann, and N. E. Leonard. *In 40th IEEE Conference on Decision and Control*, pages 4598–4603, 2001.

49. Z. Su and J. Lu. *Jounal of Beijing Institute of Technology*, 13(2):190–193, 2004.

50. H. J. Sussmann and W. Tang. Shortest paths for the reeds-shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control. Technical Report SYCON-91-10, Rutgers, 1991.

51. K.-H. Tan and M. A. Lewis. *International Conference on Intelligent Robots and Systems*, pages 132–139, 1996.

52. W. Tutte. *Graph Theory*. Cambridge University Press, 2001.

53. N. Utamaphethai and S. Ghosh. *IEEE Transactions on Intelligent Transportation Systems*, 31(3), 1998.

54. J.-D. B. X-N. Bui, P. Souéres and J.-P. Laumon. The shortest paths synthesis for nonholonomic robots moving forwards. In *IEEE International Conference on Robotics and Automation*, 1993.

55. H. Yamaguchi, T. Arai, and G. Beni. *Robotics and Autonomous Systems*, 36:125–147, 2001.

56. A. Yamashita, T. Arai, J. Ota, and H. Asama. *IEEE Transactions on Robotics and Automation*, 19(2):223–237, 2003.

This document was prepared & typeset with LATEX 2$_\varepsilon$, and formatted with NDdiss2$_\varepsilon$ *classfile (v3.0[2005/07/27]) provided by Sameer Vijay.*