ABSTRACT

Title of dissertation: BIOLOGICALLY-INSPIRED OPTIMAL CONTROL

Cheng Shao, Doctor of Philosophy, 2005

Dissertation directed by: Dr. Dimitrios Hristu-Varsakelis
Department of Mechanical Engineering
Institute for Systems Research

Inspired by the collective activities of ant colonies, and by their ability to gradually optimize their foraging trails, this dissertation investigates the cooperative solution of a broad class of trajectory optimization problems with various types of boundary conditions. A set of cooperative control algorithms are presented and proved to converge to an optimal solution by iteratively optimizing an initially feasible trajectory/control pair. The proposed algorithms organize a group of identical control systems by imposing a type of pair-wise interaction known as "local pursuit". The bio-inspired approach taken here requires only short-range, limited interactions between group members, avoids the need for a "global map" of the environment in which the group evolves, and solves an optimal control problem in "small" pieces, in a manner which is made precise. These features enable the application of the proposed algorithms in numerical optimization, leading to an increase of the permitting size of problems that can be solved, as well as a decrease of numerical errors incured in ill-conditioned problems. The algorithms' effectiveness is illustrated in a series of simulations and laboratory experiments.

BIOLOGICALLY-INSPIRED OPTIMAL CONTROL

by

Cheng Shao

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

DEDICATION

To my parents and grandparents

ACKNOWLEDGMENTS

I would like to express my sincere gratitude for all the people who have made this thesis possible and because of whom my life as a graduate student has been one that I will always cherish.

First and foremost, I am greatly appreciated to my advisor, Dr. Dimitrios Hristu-Varsakelis, for his guidance and support through past four years. Dr. Hristu provided me an invaluable opportunity to work on challenging and extremely interesting research projects, his dedication to excellence and integrity in science propelled me to learn the essence of research that makes this dissertation possible.

I would like to thank other members of my dissertation committees, Prof. Shapour Azarm, Prof. Balakumar Balachandran, Prof. Amr Baz, and Prof. P. S. Krishnaprasad, for kindly serving on the committee and their invaluable advice on how to improve the dissertation.

My thanks also go to my officemates, Mr. Lei Zhang, Mr. Kevin Roy Kefauver and Mr. Philip Yip. I will never forget their friendship and talents. I am also grateful to all my friends who make my life at Maryland a great journey, in particular Lei Zhang, Guojing Tang, Peng Lin, Haiying Liu, Hai Shi, Wei Xi, Fang Rong, Xue Mei and Wenjing Weng, to name a few.

I owe my deepest thanks to my parents for their love and encouragement in past 29 years. My gratitude to them cannot be expressed by words.

Last but not the least, I would like to thank my girlfriend, Gengsheng Lu. I would not have survived the doctorial program without her love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In nature, animal aggregation is a common phenomenon, seen in organisms that range in complexity from primitive zooplanktons to advanced mammals. Many species exhibit collective movement patterns which are highly organized when compared to the seemingly independent behaviors of their members. This is particularly spectacular in species whose individuals are of lower intelligence. For example, a school of fish can move together in a tight formation and respond almost as a single organism to evade approaching predators; worker honey bees can distribute themselves to different nectar sources in accordance with the profitability of each source; foraging ants can recruit their nest-mates to form efficient trails between the nest and a food source [13, 56].

The collective behaviors of these biological groups, especially social insects, suggest a variety of promising cooperative solutions to complex control and optimization tasks, such as formation control, resource allocation and trajectory optimization. It is therefore no surprise that in recent years, a growing number of researchers have focused on cooperation as a means of solving challenging systems and control problems, and that cooperative control systems have been emerging as attractive alternatives to their centralized counterparts. The rising interest in cooperative control also stems from the fact that it has recently become possible, and in some cases straightforward, to construct groups of communicating, low-cost, small-scale electromechanical systems that have the potential to operate collectively and outperform single-system solutions, much like animal groups can outperform individuals. For example, mobile exploration and information-gathering tasks could be accomplished cheaply and more reliably by swarms of small autonomous robots as opposed to a more sophisticated single vehicle (see Fig. 1.1). Other examples include satellite arrays that enable global communication, and teams of mobile robots that perform localization, estimation, and search tasks.

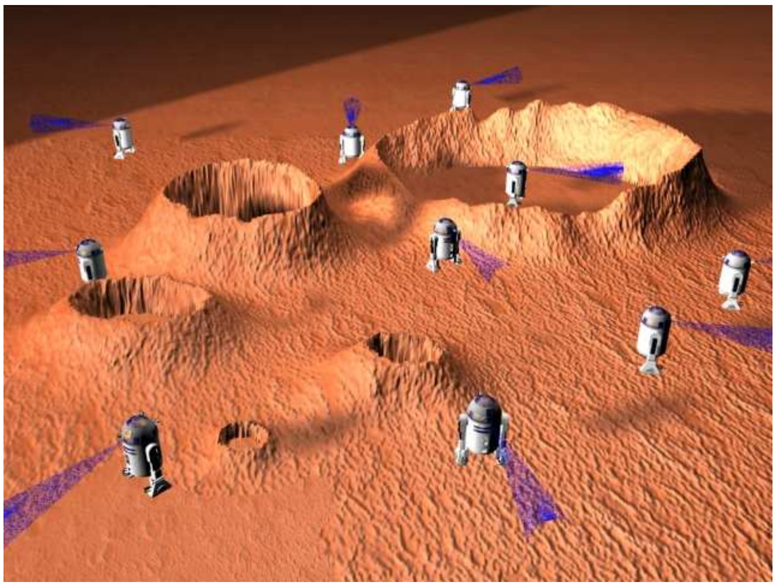

Figure 1.1: Illustration of a swarm of robots exploring an unknown planetary surface.

Besides solving problems that are interesting from an engineering point of view, some biological systems also operate under sensing, communication and computational constraints which are similar to those faced by members of artificial collectives. For instance, an ant can only see within a small region around itself, its communication with others is based on primitive methods, such as pheromonal secretions, and its cognitive ability is limited. Small-scale, low-cost units of a robotic team may be facing similar challenges regarding short-range sensing, communication and limited computing power. These considerations form the basic motivation behind what is known as the "bio-inspired" approach to systems in general, and cooperative control in particular: extracting effective ways to organize and control what would be broadly termed "engineered collectives", by studying the behavior of their biological counterparts.

The performance of a cooperative system (for example, a team of autonomous ground, aerial or underwater vehicles) depends on what could be termed its "organizing principle", taken to mean the set of rules or policies that govern the interactions among its members. An effective organizing principle can make a collective into "more than the sum of its parts", by endowing it with properties or abilities that are beyond those of individual members. At the same time, the rules of interaction must take into account the limitations of individuals and attempt to circumvent them by cooperation. In that regard, natural systems have developed specialized skills through natural selection, and may offer us important clues on how to proceed.

Finding effective organizing principles is generally a challenge because it requires solving a difficult inverse problem, namely decomposing a desired group behavior into individual behaviors. Except in degenerate cases, there is almost no hope that such a mapping – if it could be written down – would be linear. Thus one could not assume that the behavior of a group is somehow the "sum" of its members' behaviors. In many instances, the "forward" problem is more straightforward, namely postulating rules for individual behavior and predicting the collective pattern that will emerge.

To be useful, the forward approach requires one to "decode" the interactions within animal groups whose behavior is interesting or relevant, and to modify what is observed there to fit specific cooperative control problems. This modeling process requires making decisions about what kind of architecture is most appropriate for describing what is observed. Perhaps the most basic categorization that we are called to make is whether any proposed control policy should be centralized or decentralized, and whether limiting ourselves to one category leaves us with a sufficiently rich class in which to seek solutions. Certainly, if all group members are coordinated by a centralized "leader", that leader must have the capability to communicate with others and coordinate their behaviors. Leaders do exist in animal aggregates with a relatively small number of group members and advanced intelligence, e.g. wolf packs and primates. However, in groups with hundreds or thousands of – or more – members, e.g., fish schools or ant colonies, central coordination of the activities of all members seems to be out of the question, as it is beyond the capabilities of any member. The existence of centralized leaders in such aggregates would also make such groups "fragile" to possible loss or injury of those leaders. These considerations, combined with observations of the qualitatively similar behaviors of members in an insect aggregate, suggest that there are no such leaders in these groups, and this is supported by other research [13, 56, 31]. Similar concerns exist for engineered collectives. For example, it might be difficult or inefficient to coordinate the movement of a large team of autonomous vehicles by designating one of them as the "leader".

On the other hand, decentralized systems schemes have apparently been successful in nature. For instance, at the individual level, honey bees receive limited information (including the relative direction of nectar sources only) from other workmates and go to forage selected flowers. One may think that this type of behavior would lead to random distribution over the various sources, because the message each bee obtains does not convey accurate information about the profitability of nectar sources. At the group level, however, one observes that foragers are rationally dispatched over different flowers in accordance with the distribution of nectar over various sources [13]. Similar examples of useful collective behavior emerging from decentralized cooperation include trail formation by ant colonies [10], and swarming in fish and bird populations [35].

In addition to the biological evidence against centralized schemes, and technical difficulties of applying centralized control to large groups, it is significant that mathematical models of cooperating teams suggest that a few simple, myopic rules of interaction between members can often generate dramatic and complex behaviors at the group level [13, 10, 35]. Combined with empirical observations, this fact suggests that the mechanisms behind complex natural behaviors may indeed be "local", and at the very least, local interactions are a rich "language" in which to explore the problem of constructing useful cooperative control laws. This is precisely the approach taken in this thesis.

1.1 Research Objectives

The goal of this dissertation is to explore a class of cooperative control strategies that can be applied to groups of agents[1] with limited sensing and computing capabilities in order to solve difficult optimal control problems. We have argued that biological systems demonstrate many promising cooperative solutions that are similar (at least functionally) to what one might want to do with engineered collectives. What we are essentially interested in is trading off individual capabilities for cooperation in order to accomplish a complex task with less sophisticated equipment: low power, short sensing range and low communication burden. We will look to natural examples (specifically the foraging behavior of ant colonies) for successful prototypes.

Our approach will be to postulate organizing principles inspired by observations of individual behavior, and then investigate the "collective behavior" that emerges. We will obtain control policies for individual agents by modeling the individual behaviors that seem to explain the collective movement patterns of ant colonies. An effective model will allow us to capture some aspects of the "experience" accumulated through interaction within the colony. The rules we aim to extract are to be kept simple with respect to the computation and communication

[1]Throughout the document we will use "agent" to refer to a member of a group of control systems.

resources required to implement them, and are to be applied to cooperative control systems, including inexpensive autonomous robots.

1.2 Research Contributions

The main contributions of this thesis are:

- To propose a set of novel bio-inspired optimization algorithms that enable a group of control systems with limited communication, sensing and computation capabilities to solve complex optimal control problems under conditions that would be prohibitive for an individual system.

- To generalize earlier pursuit-based cooperative optimization algorithms to a broad class of optimal control problems that involve systems with non-trivial dynamics and include many of the well-studied situations in optimal control, such as the computation of geodesics, and optimal control in minimum-time and/or with partially-constrained final states.

- To explore the application of pursuit-based optimization as a numerical method and to illustrate the advantages of combining the proposed cooperative algorithms with existing numerical methods in order to address large scale optimization problems.

- To demonstrate the proposed optimal control methods through a series of simulations and laboratory experiments.

1.3 Dissertation Outline

The remainder of this thesis is organized as follows:

Chapter 2: We review previous research on cooperative and biologically-inspired control strategies as well as an early prototype of one of the algorithms to be presented in Chapter 3.

Chapter 3: We define the class of optimal control problems under consideration, and introduce a set of biologically inspired algorithms that achieve optimality via cooperation. The proposed algorithms are variants of a strategy known as "local pursuit" and mimic the interactions of ant colonies.

Chapter 4: We present the main results concerning the behavior of a group of control systems that evolve according to the proposed algorithms, and prove that local pursuit leads to a locally optimal solution.

Chapter 5: We discuss the application of local pursuit as a computational tool, when combined with current numerical optimization methods. In particular, we will show that the proposed algorithms cannot only extend the permitting size of numerical optimal control problems, but also decrease computational errors.

Chapter 6: We present a series of simulations and laboratory experiments that illustrate the performance of the proposed algorithms.

Chapter 7: We summarize the results of this research and discuss possible directions for future work.

Chapter 2

Literature Review

This work lies at the intersection of cooperative and biologically-inspired control. There is a broad variety of research directions in each of these two areas. In this chapter we review some of the works most relevant to our approach.

2.1 Cooperative Control Systems

Group Formations

Current research directions under the general theme of "cooperative control" include work on group formation control, virtual leader-based formations and group-based estimation, among others.

The work in [25] described a control framework that allows robots equipped with range sensors to coordinate their movements in order to accomplish search and rescue manipulations. The authors derived three formation control policies - "Separation-Bearing Control", "Separation-Separation Control" and "Separation

Distance-To-Obstacle Control" - with respect to relative position of neighboring robots or obstacles in the environment. Using the above formation controls, a higher-level "basic formation" policy was constructed and proved to stabilize the formation of a robot team.

A team of robots in formations were modeled as a triple $(g, r, \mathcal{H})$ in [17], where g represents the gross position and orientation of the lead robot, r is a set of shape variables describing the relative position of individual robots and $\mathcal{H}$ is a so-called "control graph" which describes the control strategy used by each robot and is modeled by a transition matrix. This framework enables the representation and enumeration of possible transition between two formations. The authors also developed a set of decentralized control laws that allow robots to maintain different kinds of formations and undergo shape changes.

Virtual Leader-Based Formations

Another important research direction in group formation is concerned with the construction of artificial potentials and virtual leaders, such as the coordination strategy for vehicle group maneuvers, including translation, rotation, expansion and contraction, as presented in [53]. The control applied on each vehicle is defined as the linear combination of the gradient of these potentials (V_I and V_h in the following equation) as well as a linear damping term:

$$u_i = -\sum_{j \neq i}^{N} \nabla_{x_i} V_I(x_{ij}) - \sum_{k \neq i}^{M} \nabla_{x_i} V_h(h_{ik}) - K\dot{x}_i,$$

where $u_i = \ddot{x}_i$ is the control, x_{ij} is the distance between the i^{th} vehicle and the j^{th} vehicle, and h_{ik} is the distance between the i^{th} vehicle and the virtual leader k. The artificial potentials V_I apply attraction to distant neighbors as well as repulsion for neighbors which are too close. The accomplishment of a desired mission is through controll of the direction of the virtual leaders, while the speed of the virtual leader is to ensure the convergence of the formation. The convergence property is proved by Lyapunov's method.

The work in [22] also proposed a general, platform-independent approach by letting the virtual leader follow a desired trajectory. By constructing a one-to-one formation constraint function $F(\mathbf{x}_1, \ldots, \mathbf{x}_m) : \mathbb{R}^n \times \cdots \times \mathbb{R}^n \to \mathbb{R}^+$, where $\mathbf{x}_i$ is the position of each vehicle, the solution of the desired trajectories is obtained by taking the steepest descent to desired formation. Under the assumption that the real robots track their respective reference trajectories perfectly, it holds that the virtual leader's trajectory converges to the desired trajectory.

Distributed Localization and Estimation

In [43, 44], the authors proposed a method called "Cooperative Positioning System (CPS)" to alleviate the weakness of traditional position identification techniques usually applied in robotics, including dead reckoning and landmark-based localization. In that work, a robot group is divided into two teams in order to provide "portable landmarks". At every instance, one team moves and the other stays static, acting as the landmark, then they exchange roles. Therefore, each team can benefit from accurate measurements by utilizing static landmarks, while at the same time, no prior placing of landmarks is required. The drawback is that at least one robot must stay stationary so that the overall speed of the algorithm is limited.

Another approach to simultaneous relative localization for a group of mobile robots was presented in [58]. Each robot measures its own motion using its proprioceptive sensors. When two robots x_i, x_j meet, they share information with one another, then the i^{th} robot updates the estimate of its own position with respect to that of the j^{th} robot's and the relative distance estimate between the two. The process is described by a set of Kalman filter equations. This method "distributes" what would be a centralized estimation process among a number of Kalman filters, each of them operating on a different robot.

2.2 Biologically-inspired Control Laws

Animal Group Pattern Modeling

Successful observations of biological groups have seeded a variety of bio-inspired research in engineering, from modeling of animal groups, to distributed collective covering and searching, swarm formation stabilization analysis and biologically-motivated optimization.

The work of [64] proposed a simple model concerning the movement of n autonomous agents moving at the same speed but with varying initial headings. If each agent of the group uses the "nearest neighbor rule" to update its heading, that is

$$\langle\theta_i(t)\rangle_r = \frac{1}{1+n_i(t)}\left(\theta_i(t) + \sum_{j\in\mathcal{N}_i(t)} \theta_j(t)\right),$$

where $\theta_i(t)$ is the heading of the i^{th} agent and $n_i(t)$ is the number of neighbors of the i^{th} agent at time t. All agents' headings are proved to converge to a common constant vector as time goes on. The theoretical explanation for the convergence described in the above model is provided in [35], along with several similar models inspired by [64], such as "leader following" showing that if there exists an agent acting as the "leader" in the group, all agents will evolve to point to the same heading as the leader . This "nearest neighbor rule" can cause all the members of a group to move towards the same direction despite the fact that there is no centralized coordination and that an agent's set of nearest neighbors might change as the system evolves. The successful models developed on [64, 35] have been used to explain how a group of birds or fish manage to move in tight formation as a single entity while there is no central coordinator.

A mathematical model for describing the honey bees' behaviors was constructed in [13]. Although individual honey bees do not have global information about the distribution of nectar sources, each one will comply with certain rules to determine where it will go to forage. This process is described by a flow diagram illustrated in Fig. 2.1. At the bifurcations on the diagram, honey bees make decisions on which nectar source to forage and whether to dance - the way honey bees transfer information - or not. The decision-making process is modeled by means of probabilities of proceeding with various actions. For example, P_X^A represents the

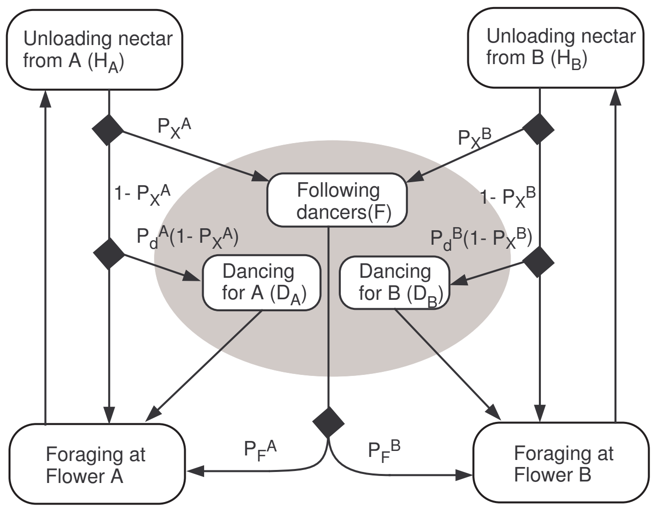

Figure 2.1: Flow diagram illustrating a model of how honey bees allocate their foragers.

probability for one bee to watch other dancers after it unloads the nectar collected from flower A, $P_d^A(1 - P_X^A)$ represents the probability of dancing to convey information on flower A, and P_F^A represents the probability of following other dancers to forage flower A. Noticing that honey bees make decisions only after receiving limited information from their workmates, [13] proposed a set of simple equations to describe these probabilities, e.g.,

$$P_F^A = \frac{D_A d_A}{D_A d_A + D_B d_B},$$

where D_A represent the number of dancers for flower A and d_A is the proportion of time that foragers actually dance. Other probabilities such as P_X^A can be assumed to be a constant. Simulations showed a collective result that is qualitatively similar to what is observed in real bee colonies.

The work of [16] presented a model of how ants select foraging areas. According to that model, each ant initially has a uniform distribution over all foraging areas, so that it is equally likely to forage everywhere. If at time t, the ant finds food in zone i, then the probability $P_i(t+1)$ of foraging at area i at time $(t+1)$ is given by $P_i(t+1) = P_i(t) + \min(P^+, 1 - P_i(t))$, where P^+ is a constant indicating the relative importance of "learning". If at time t there is no food found at area i, then the probability $P_i(t+1)$ will be decreased. By this mechanism, both an individual ant and its colony will evolve into an optimal spatial distribution over foraging areas – obtaining maximum food when the appearance of food at each zone is random and unknown to the ants.

Models of Ant-Trail Formation

An insect aggregation example of particular relevance to this thesis is the foraging activity of ant colonies, which includes discovering food, recruiting nest-mates and forming trails. When an ant finds food, it will recruit other ants around to convey that food back to the nest. These co-workers will rapidly form a well-defined trail between the nest and food even though they are homogeneously distributed at first. Finding an efficient path between the nest and food seems too complicated a problem for an individual ant to solve, especially if one considers the ant's tiny size relatively to obstacles in the environment, such as stones, stick and crevices. Nonetheless, a colony of ants seem quite adept at such tasks [13]. To explore the intrinsic mechanism that leads to the collective efficiency as opposed to individual clumsiness, several models concerning ant-trail formation have been proposed.

The work of [13] described a model about how ants utilize pheromonal secretions to choose ongoing pathways. According to this model, pheromonal secretions are laid along the paths by ants to keep a trace and recruit other nest-mates. At the same time, pheromonal secretions evaporate as time goes on. When an ant comes to a location where several traces cross, it follows the path with the highest concentration. As illustrated in Fig. 2.2, the probability of taking the left branch of

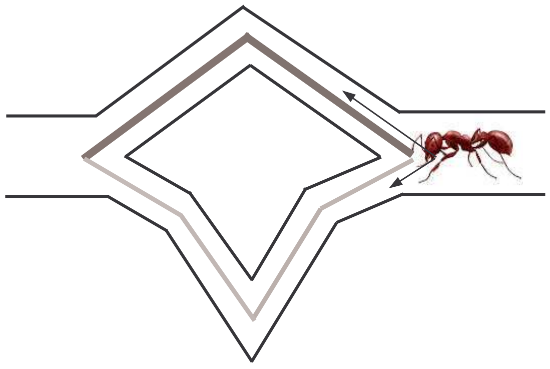

Figure 2.2: An ant chooses the path in accordance with pheromone concentrations

a “fork” in the terrain is quantified as

$$P_L = \frac{(k + C_L)^n}{(k + C_L)^n + (k + C_R)^n},$$

The parameters C_L and C_R represent the pheromone concentrations on the left and right branch, n and k are constants corresponding to the degree of nonlinearity of the choice and the attraction of an unmarked branch, respectively. The key point is that the pheromonal secretions play a “positive feedback” role. Although an individual ant knows little about the entire environment and the distribution of its co-workers, simulations suggest that the colony has the collective potential to find the shortest path.

Another model concerning ant-trail formation on a plane was explored in [10]. The basic rule in this model is that each ant "follows" one of its co-workers instead of measuring pheromonal secretions, as Fig. 2.3 illustrates. In [10], the pheromonal secretions laid by an ant are used to trace its own tail and find its way back to the nest but not to recruit its nest-mates. Paraphrasing [10], the path traveled by

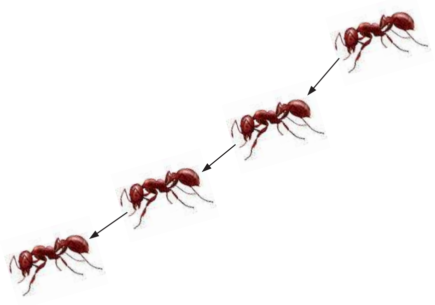

Figure 2.3: Ants find the shortest path joining two members

a single ant is a curve $x_k(t) : [0,T] \to \mathbb{R}^2$ with $\dot{x}_k = u$ ($u \in \mathbb{R}^2$). The boundary conditions for these systems are $x_0(0) = x_0$ and $x_0(T) = x_f$, which represent the starting point (nest) and the target point (food) respectively. Any ant can trace its own trajectory back to the nest so that we have a sequence virtually infinite of ants departing from x_0. Each ant moves with unit speed and there are Δ units of time between the departure time of successive ants. At every instance, each ant except the first one will follow its predecessor by pointing its speed vector in a straight line toward the predecessor. In short, for $k = 1, 2, 3\ldots$

$$\dot{x}_k(t) = \frac{x_{k-1}(t) - x_k(t)}{\|x_{k-1}(t) - x_k(t)\|},$$

with $x_k(0) = x_0$ for $t \leq k\Delta$ and $x_k(T) = x_f$ if x_k reaches the target x_f. For the case when $x_k(t) \in \mathbb{R}^2$, it has been shown that if the initial ant $x_0(t)$ follows a sub-optimal path from x_0 to x_f, then the trajectories $\{x_k\}$ will converge to a straight line connecting x_0 and x_f.

Distributed Covering and Searching

Inspired by the fact that ants and other insects use pheromones for various communication and coordination tasks, [65] developed robust adaptive algorithms to perform tasks requiring the traversal over an unknown region, such as cleaning the floor of an unmapped building. The region to be covered is described by a graph $G = (V, E)$, where every vertex represents an "atomic region" (tile). When agents deployed in the algorithms are traveling on G, they mark their trails by depositing a pheromone, which evaporates as time goes on. By this mechanism, the agents can assign each edge of the graph (which represents an adjacency neighborhood relation between two "atoms"), with a label of the time of the most recent traversal of that edge. An agent visiting vertex $u \in V(G)$ checks the labels on all edges emanating from u, and goes in the direction that was not visited for the longest time by choosing the smallest label. The time t_k needed to cover all edges of the graph by k agents under the "ANT-WALK-1" rule based on the above idea is bounded by

$$t_k \leq n\Delta \left(\rho(G) + \frac{(1+\alpha)n}{k} \right),$$

where Δ is the maximum vertex degree in G, $n = |V(G)|$, α is related to the measurement noise and $\rho(G)$ is the cut-resistant of G. In the same work, the "ANT-WALK-2" rule, a generalization of the famous Depth-First search algorithm, was developed for agents with limited memory. The time t_k for this rule is bounded by

$$t_k \leq (n\Delta/2) \left\lceil \frac{(1+\alpha)}{k} \right\rceil .$$

The work in [55] investigated the performance of cooperative strategies that control autonomous air vehicles searching a dynamic environment to gather information. The proposed framework considers two main components for each agent: distributed learning of the environment and distributed path planning based on the information gathered. The collective results based on a recursive q-step ahead as well as an interleaved planning technique illustrate that the cooperation among vehicles improves the performance. The authors also explored the feasibility of developing coordination control strategies inspired by the social foraging activities of *E. coli*, a common type of bacteria.

Swarm Formation Stability Analysis

In [27, 29] the authors considered a continuous-time swarm model in an n-dimensional space base on artificial potential functions, which is inspired from the literature on swarming in mathematical biology. The motion of individual i is given by

$$\dot{x}^i = \sum_{j=1, j\neq i}^{M} g(x^i - x^j), \tag{2.1}$$

where $g(.)$ represents an attraction/repulsion function between the swarm members. Thus, the force acting on a particle in the swarm is determined by the sum of the attraction and repulsion forces from all other members. Then, if a swarm is described by Eq. (2.1), all members of the swarm will converge to a hyper-ball $B_\epsilon(\bar{x}) = x : \|x - \bar{x}\| < \epsilon$, where $\epsilon = \frac{b}{a}\sqrt{\frac{c}{2}}\exp(-\frac{1}{2})$ and $\bar{x}$, the center of the swarm, can be shown to be stationary.

The last potential function has also been extended to social foraging swarms in [28], where the motion of each individual is determined by

$$\dot{x}^i = -\nabla_{x^i}\sigma(x^i) + \sum_{j=1, j\neq i}^{M} g(x^i - x^j). \tag{2.2}$$

Here $\sigma : \mathbb{R}^n \to \mathbb{R}$ represents the attractant/repellent profile of some substances (for example, food/nutrients, or toxic chemicals), hence the term $-\nabla_{x^i}\sigma x^i$ represents the motion preference of the individual toward regions with higher nutrient concentration and away from regions with high concentration of toxic substances. The authors also proposed a cohesion analysis (swarm stability) for different kinds of attractant/repellent profile.

Biologically-Motivated Optimization

In [18] a search methodology was introduced based on the "distributed autocatalytic process", to solve a classical optimization problem - the Traveling Salesman Problem (TSP). Inspired from the fact that ants can use pheromonal secretions to find the shortest path when foraging, [18] utilized an ant team to travel through the towns in TSP. The transition probability from town i to town j for the k^{th} ant is defined as

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k\in\Omega_i}[\tau_{ik}(t)]^\alpha \cdot [\eta_{ik}]^\beta} & \text{if } j \in \Omega_i \\ 0 & \text{otherwise} \end{cases}, \tag{2.3}$$

where Ω_k is the set of towns reachable by k, and $\tau_{ij}(t)$ is the intensity of pheromonal trail on edge (i, j) at time t, which is laid by ants on the edge and evaporates as time goes on. The visibility of the path, η_{ij}, is defined as the reciprocal of the distance between the town i and town j, d_{ij}, i.e., $\eta_{ij} = 1/d_{ij}$. Lastly, α and β are parameters evaluating the relative importance of the trail and the visibility, respectively. Based on Eq. (2.3), [18] developed three algorithms: "ant-cycle","ant density" and "ant-quantity", each based on slightly different rules by which ants update $\tau_{ij}(t)$ along their trails. The trajectories of the ant team in each algorithm eventually converges to the optimal tour for the TSP.

A "probabilistic pursuit" algorithm for a group of agents moving on a planar grid was presented in [11]. Briefly, a sequence of agents $A_0, A_1, \ldots$ move from the origin at time $t = 0, \Delta, 2\Delta, \ldots$ to a destination. While moving on the grid, A_{n+1} "chases" A_n by making a random choice of a neighboring grid point and moving there. The probability distribution that defines the agent's choice is determined by its relative position to its predecessor, that is

$$A_{n+1}(t+1) = A_{n+1}(t) + \delta_{n+1}(t+1),$$

where $\delta_{n+1}(t+1) \in \{1, -1, j, -j\}$ and

$$\begin{aligned} \text{Prob}\{\delta_{n+1}(t+1) = \text{sign}(d_x)\} &= \frac{\|d_x\|}{d}, \\ \text{Prob}\{\delta_{n+1}(t+1) = j \cdot \text{sign}(d_y)\} &= \frac{\|d_y\|}{d}, \end{aligned}$$

where $A_n(t)$ is the position of the n^{th} agent at time t, $d = \|d_x\| + \|d_y\|$ and d_x, d_y are relative distances between A_n and A_{n+1} at the x and y directions, respectively. Analytical proofs showed that the average trajectories of agents converge to a straight line on the discretized plane.

Lastly, a biologically motivated algorithm, termed "particle swarm optimization", was presented in [37, 20]. Particle swarm optimization lies somewhere between the generic algorithms and evolutionary programming. The algorithm was initially inspired by the choreography of bird flocks. The algorithm seeks the minimum of a cost function by first setting up a swarm of particles with random positions and velocities. The velocity of each particle is changed towards a random linear combination of the best solution achieved by the particle and the best solution achieved by the swarm so far.

Chapter 3

Biologically Inspired Algorithms for Optimal Control

In this chapter we introduce a class of cooperative, optimal control algorithms inspired by ant-trail formation and discuss their potential advantages. The algorithms cover a broad set of trajectory optimization problems, including problems with fixed or partially-constrained final states, and fixed or free final time. Among the existing models of ant-trail formation which were discussed in Chapter 2, we are particularly interested in the simplicity of [10]. However, that work applies to a rather narrow domain, specifically, it is restricted to the discovery of shortest paths in the Euclidean plane, by a group of kinematic vehicles. Our goal is to exploit the same principle used in [10] but to expand its applicability to a much broader class of optimization problems. Before proceeding, we describe the precise problems we are concerned with.

3.1 Problem Statements and Notation

This dissertation explores the solution of optimal control problems using a group of cooperating "agents". The term "agent" here refers to a number of "copies" of a system with non-trivial dynamics, i.e., for $k = 0, 1, 2 \dots$

$$\dot{x}_k = f(x_k, u_k), \qquad x_k(t) \in \mathbb{R}^n, u_k(t) \in \Omega \subset \mathbb{R}^m, \tag{3.1}$$

Physically, each copy of Eq. (3.1) could stand for a robot, UAV or other autonomous system.

The systems evolving under Eq. (3.1) are with fixed initial states x_s. The final states could be either fixed or partially-constrained. Each curve $x_k(t) : [0, T] \to \mathbb{R}^n$ are the trajectory of the k^{th} agent. We begin with a class of optimal control problems with fixed final time and fixed final states. In all algorithms, we assume that we have available an initial feasible (but sub-optimal) control/trajectory pair $(u_{feas}(t), x_{feas}(t))$ for Eq. (3.1), obtained through a combination of priori knowledge about the problem and/or random exploration.

Problems with Fixed Final Time and Fixed Final States

Assume that there is a pair of states x_s and x_f which are equilibrium points[1] of Eq. (3.1) for $u = 0$.

Problem 3.1: *Find a trajectory $x^*(t)$ $(t \in [0, T], T$ fixed$)$ that minimizes*

$$J(x, \dot{x}, t_0, T) = \int_{t_0}^{t_0+T} g(x(t), \dot{x}(t))dt \tag{3.2}$$

[1]Without loss of generality we assume that $u = 0$ at those equilibria.

with $x(t_0) = x_s,\ x(t_0 + T) = x_f,\ \ g(\cdot, \cdot) \geq 0,$ *and* $x(t)$ *subject to Eq. (3.1).*

It will be convenient to define the following notation. Let $\mathbb{D} \subset \mathbb{R}^n$ be a domain containing states a and b. Assume $0 < \sigma \leq T$ and $t_0 \geq 0$. The optimal trajectory from a to b in fixed T units of time will be denoted by $x^*(t)\ \ (t \in [t_0, t_0 + T])$ and will satisfy:

$$J(x^*, \dot{x}^*, t_0, T) = \min_x J(x, \dot{x}, t_0, T), \tag{3.3}$$

subject to $x(t_0) = a,\ x(t_0 + T) = b$. We denote the cost of following the optimal trajectory from a to b for σ units of time by:

$$\eta(a, b, T, t_0, \sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x^*(t), \dot{x}^*(t))dt, \qquad \sigma \leq T, \tag{3.4}$$

where the optimal trajectory $x^*(t)$ is defined by Eq. (3.3).

For a generic trajectory $x(t)$ of Eq. (3.1), we define

$$C(x, t_0, \sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x(t), \dot{x}(t))dt \tag{3.5}$$

to be the cost incurred along $x(t)$ during $[t_0, t_0 + \sigma)$.

Problems with Free Final Time and Partially-Constrained Final States

We are also interested in problems with free-final time and partially-constrained final state:

Problem 3.2: *Find a trajectory* $x^*(t)$*, a final time* $\Gamma^* > 0$ *and a final state* $x^*(\Gamma^*)$ *that minimize*

$$J_Q(x, \dot{x}, t_0) = \int_{t_0}^{t_0+\Gamma} g(x(t), \dot{x}(t))dt + G(x(t_0 + \Gamma)) \tag{3.6}$$

subject to the constraints $x(t_0) = x_s$ *and* $Q(x(t_0 + \Gamma)) = 0$.

Here it is assumed that $Q(\cdot)$ is an algebraic function of the state, $g(x(t), \dot{x}(t)) \geq 0$ and

$$G(x) = \begin{cases} F(x) & \text{if } x \in S_Q \\ 0 & \text{if } x \notin S_Q \end{cases},$$

with $F(x) \geq 0, \forall x \in S_Q$ is a penalty cost for the final state. S_Q is the constraint set of final state:

Definition 3.1: *Given the final state constraint* $Q(x) = 0$, *the* constraint set *of* x *is*

$$S_Q \triangleq \{x | Q(x) = 0\}.$$

For any pair of fixed states $a, b \in \mathbb{D} \subset \mathbb{R}^n$, let $x^*(t)$ denote the optimal trajectory from a to b with free final time (minimizing J with respect to x and Γ only). The corresponding optimal final time is $\Gamma^*(a, b)$. The cost of following x^* is denoted by:

$$\begin{aligned} \eta_F(a, b, t_0) &\triangleq \int_{t_0}^{t_0+\Gamma^*} g(x^*(t), \dot{x^*}(t))dt + G(x^*(t_0 + \Gamma^*)) \\ &= \min_{x,\Gamma} J_Q(x, \dot{x}, t_0), \end{aligned} \tag{3.7}$$

subject to $x(t_0) = a, \ x(t_0 + \Gamma) = b$.

Now, let $x^*(t)$ be the optimal trajectory from an initial state a to the constraint set S_Q, and let $\Gamma^*_Q(a, S_Q)$ be the corresponding optimal final time from a to S_Q. The cost of following x^* is denoted by

$$\begin{aligned}\eta_Q(a,t_0) &\triangleq \int_{t_0}^{t_0+\Gamma_Q^*} g(x^*,\dot{x^*})dt + G(x^*(t_0+\Gamma_Q^*)) \\ &= \min_{x,\Gamma_Q} J_Q(x,\dot{x},t_0), \qquad (3.8)\end{aligned}$$

subject to $x(t_0) = a, Q(x(t_0+\Gamma_Q)) = 0$.

The cost of following a generic trajectory $x(t)$ of Eq. (3.1) during $[t_0, t_0+\sigma)$ is denoted by:

$$C_Q(x,t_0,\sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x(t),\dot{x}(t))dt + G(x(t_0+\sigma)). \qquad (3.9)$$

The following facts can be derived easily from the properties of optimal trajectories and will be helpful in the sequel.

Fact : *Let η, C, η_F as defined in Eq. (3.4),(3.5),(3.7); let $x_k(t)$ be a trajectory of Eq. (3.1) and $x^*(t)$ an optimal trajectory of Eq. (3.2) or Eq. (3.6). Then, the following holds:*

1. *$\eta(a,b,T,t_0,\sigma) \leq C(x_k,t_0,\sigma)$ with any $x_k(t_0) = x^*(t_0), x_k(t_0+\sigma) = x^*(t_0+\sigma)$ where $x^*(t)$ satisfies Eq. (3.3).*

2. *$\eta(a,c,T,t_0,T) \leq \eta(a,b,\sigma,t_0,\sigma) + \eta(b,c,T-\sigma,t_0+\sigma,T-\sigma)$.*

3. *$C(x_k,t_0,T) = C(x_k,t_0,\sigma) + C(x_k,t_0+\sigma,T-\sigma)$.*

4. *$\eta_F(a,b,t_0,\sigma) \leq \eta(a,b,T,t_0,\sigma)$.*

5. *$\eta(a,b,T,t_0,\sigma) = C(x^*,t_0,\sigma)$ where $x^*(t)$ satisfies Eq. (3.3).*

3.2 Difficulties Faced by a Single Agent

It may be difficult for a single system or a centralized system to solve instances of the optimal control problems stated above when the problems' boundary conditions are far apart (here "far" refers to intervals in space as well as time). As Fig. 3.1 illustrates, there are a number of considerations which highlight the challenges of finding an optimal trajectory over "long distances":

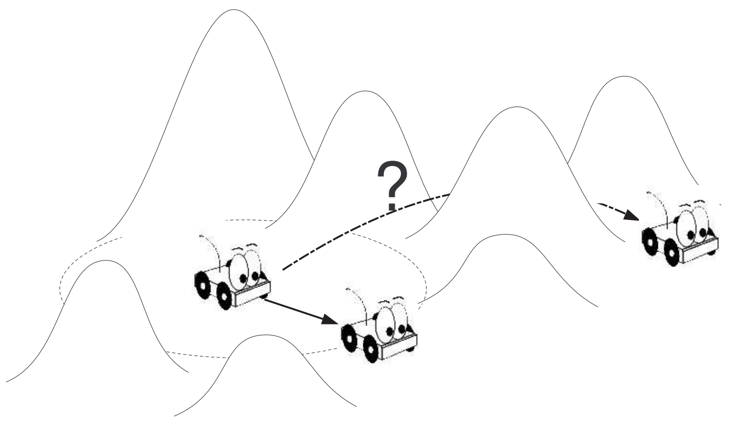

Figure 3.1: It is easier to solve an optimization problem within a "small" region.

1. Environment information. An individual must have access to a map of the terrain, or at least a partial map containing the optimal trajectory and the coordinates of the target state. Such a map is difficult to construct from sensor data of a single system because it requires the system to "discover" the global geometry of the space. The map usually covers more area than is necessary.

2. Sensing/communicating capabilities. It is much more costly to construct long-range sensors and communication devices. Thus it is more difficult to communicate with far-off members of the group than with nearby ones. Moreover,

unless group members agree on a common coordinate system, state and sensor data communicated from other members may need to be transformed appropriately by the receiver. This implies that the receiver must know how its own coordinate system is related to that of a far-off neighbor.

3. Information fusion. Even if a group of agents can be dispersed and are able to compose a map that describes a local "patch" around each of them, the composition of these patches is nontrivial. It is not guaranteed that the composition of these patches covers the whole environment, or at least covers the region containing the optimal trajectory. If enough such local patches have been collected, their fusion into a composite map requires a large amount of information communication, and sophisticated algorithms. That is because each agent could in principle use its own coordinate system when composing its map, so that the composite map would have to need additional one agent in the group has sufficient memory, communication bandwidth and computing ability to accomplish this.

4. Computational costs. Even if an effective global map could be obtained, solving optimization problems over a moderately complex environment, especially one that cannot be parametrized by a single coordinate system, requires large amounts of calculations. Oftentimes, the associated optimal control problems have no closed-form solution and one needs to make use of numerical methods. As we shall see in the sequel, the computing resources needed by numerical optimal control methods may be prohibitive when long trajectories are con-

sidered.

In light of these considerations and for the purposes of avoiding the high cost and low reliability associated with solving such problems with a single system, we now propose a class of cooperative control algorithm – inspired from foraging activities of ant colonies – which use a group of cooperative systems in a way that allows for demands on individual agents to be minimized.

3.3 A Class of Bio-Inspired Pursuit Algorithms

Considering the sensor and communication limitations of an individual autonomous vehicle (or indeed of an ant), it seems easier to aim at its leader on a manifold and move on a shortest path toward it if the two of them were close. Similarly, it is much easier for an agent to detect the state of its leader and compute an optimal trajectory to that state if the pair were closer. Fig. 3.1 illustrates this point. In the following we generalize the notion of "pursuit" as a process of seeking optimal trajectories locally, by combining the efforts of a group of agents to gradually optimize an initial solution.

Depending on the type of optimal control problem in Section 3.1, we have four variations of a basic pursuit-based algorithm – a control policy for cooperative optimization by the group. All are similarly structured and vary only in the manner in which agents adjust their own trajectories when proceeding. For simplicity, we start with the algorithms that apply to problems with fixed final states and fixed final time.

We consider the formation of an ordered sequence of agents, with each agent trying to reach its predecessor along an optimal trajectory. The sequence is initiated with the first agent following x_{feas} to the desired final state. The precise rules that govern the movement of each agent are:

Algorithm 3.1 (Sampled Local Pursuit): *Identify two states x_s and x_f on $\mathbb{D}$. Let $x_0(t)$ $(t \in [0,T])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_s, x_0(T) = x_f$. Choose $\Delta,\ \delta \in \mathbb{R}$ such that $0 < \delta < \Delta \leq T$. Then:*

1. *For $k = 1, 2, 3\ldots$, let $t_k = k\Delta$ be the starting time of the k^{th} agent, i.e., $u_k(t) = 0,\ x_k(t) = x_s$ for $0 \leq t \leq t_k$.*

2. *When $t = t_k + i\delta, i = 0, 1, 2, 3, \ldots$, calculate the control $u_t^*(\tau)$ that achieves:*
$$\begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta),\ \tau \in [t, t+\Delta] & \text{if } \Delta + i\delta < T \\ \eta(x_k(t), x_f, \lambda, t, \lambda),\ \tau \in [t, t_k + T] & \text{otherwise} \end{cases},$$
where $\lambda = t_k + T - t$.

3. *Apply $u_k(t) = u^*_{t_k+i\delta}(t - t_k - i\delta)$ to the k^{th} agent for $t \in [t_k + i\delta, t_k + (i+1)\delta)$ if $\Delta + i\delta < T$, or for $t \in [t_k + i\delta, t_k + T)$ otherwise.*

4. *Repeat from step 2 until the k^{th} agent reaches x_f.*

According to the Sampled Local Pursuit (SLP) algorithm, agents leave the starting state x_s one after another, every Δ units of time , so that the k^{th} agent leaves at time $t_k = k\Delta$. We assume that the number of agents in the group is sufficiently large and let $x_k(t)$ denote the k^{th} agent's trajectory[2]. Each agent moves to "pursue"

[2]By slight abuse of notation, we will sometimes utilize x_k to denote both the k^{th} agent and its trajectory.

its predecessor – to move along the optimal trajectory from the follower's state to that of the leader's. When considering two agents as a pursuit pair, we denote the $(k-1)^{th}$ agent as the "leader" and, the k^{th} agent as the "follower".

The two adjustable parameters in the SLP algorithm are the *pursuit interval* Δ, which denotes the frequency with which new agents depart from x_s, and the *updating interval* δ, which denotes the frequency with which an agent updates its own trajectory. At each $t = t_k^i$, the follower calculates the optimal control $u_t^*(\tau)$ ($\tau \in [t, t+\Delta)$) that steers it from $x_k(t)$ to $x_{k-1}(t)$ over Δ units of time, i.e., from its current state to the leader's current state. During $[t_k + i\delta, t_k + (i+1)\delta]$, the follower moves along the trajectory produced by $u_t^*(\tau)$, and the process repeats until the follower reaches x_f. Usually, we will take $0 < \delta < \Delta$ so that each agent only needs to solve a finite number of optimal control problems along its trajectory. If the problem in question can be solved efficiently, e.g., in closed form, one may choose to decrease δ.

Given $x_{k-1}(t)$, SLP yields a well-defined trajectory $x_k(t)$ on $[0, T]$ for the k^{th} agent. The resulting trajectory is continuous but not necessarily smooth on the interval $[t_k, t_k + T]$. A snapshot of this iterative updating processes is illustrated in Fig. 3.2.

For notational convenience, we define the *planned* trajectories, denoted by $\hat{x}_{k,i}(t)$, to be the trajectories along which the follower plans to move at $(t_k + i\delta)$ but may not do so because it will update its future trajectory at $t_k + (i+1)\delta$ (for notational convenience we will often neglect the i subscript and simply use $\hat{x}_k(t)$, when the value of i is clear from the discussion). In other words, the planned

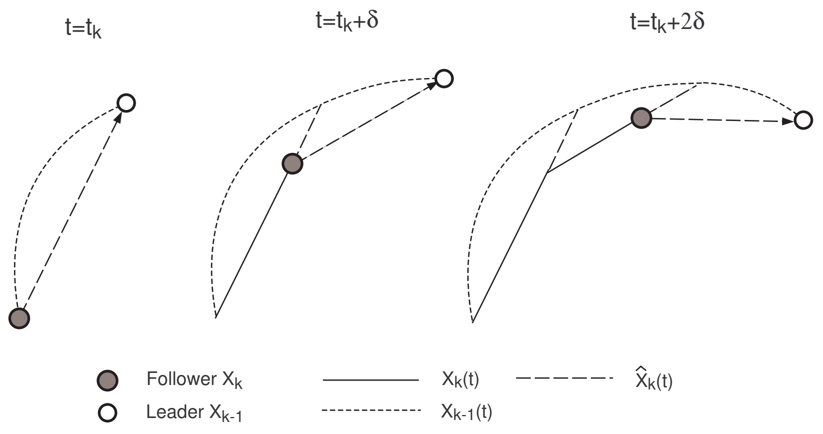

Figure 3.2: A snapshot of the updating processes executed by the k^{th} agent.

trajectories are the trajectories that would be produced by $u^*_{t_k+i\delta}(\tau)$ on $[t_k + (i + 1)\delta, t_k + i\delta + \Delta]$. Of course, in the next iteration of Step 2, $u^*_{t_k+(i+1)\delta}(\tau)$ will steer the agent along a different trajectory. The *realized* trajectories, defined as the trajectories along which the follower actually moves, are $x_k(t)$. Referring to Fig. 3.2, the planned trajectories and realized trajectories are represented by the dashed lines and solid lines, respectively.

SLP can also be altered to apply in problems with free final times. Doing so requires that at every updating step the calculated optimal final time Γ defined by Eq. (3.6) satisfies $\Gamma \leq \delta$. If that is not the case, then for some agent, there will be a trajectory segment which will remain “static” and will not be further optimized. To avoid that situation, we must select δ to be less than all Γ defined by Eq. (3.6) for all instances of the“local” optimal control problem in Step 2 of the SLP algorithm, the obvious choice being $\delta = 0$. As we shall see, $\delta = 0$ suggests a kind of continuous pursuit of each leader by the corresponding follower. We proceed to describe this continuous version of SLP, which we later show, can be obtained by having δ approach 0.

Algorithm 3.2 (Continuous Local Pursuit): *Identify two states x_s and x_f on $\mathbb{D}$. Let $x_0(t)$ $(t \in [0, T_0])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_s, x_0(T_0) = x_f$. Choose Δ such that $0 < \Delta \leq T$. Then:*

1. *For $k = 1, 2, 3 \ldots$, let $t_k = k\Delta$ be the starting time of the k^{th} agent, i.e., $u_k(t) = 0,\ x_k(t) = x_s$ for $0 \leq t \leq t_k$.*

2. *For all $t \geq t_k$, calculate $u_t^*(\tau)$ for all $t \in [t_k, t_k + T_k]$ such that $f(\hat{x}_k(\tau), u_t^*(\tau)) = \dot{\hat{x}}_t(\tau)$, and $\hat{x}_t(\tau)$ achieves:*
$$\begin{cases} \eta_F(x_k(t), x_{k-1}(t), t), \tau \in [t, t + \Gamma^*(x_k(t), x_{k-1}(t))], & \text{if } \ x_{k-1}(t) \neq x_f \\ \eta_F(x_k(t), x_f, t), \tau \in [t, t + \Gamma_Q^*(x_k(t), x_f], & \text{if } \ x_{k-1}(t) = x_f \end{cases}.$$

3. *Apply $u_k(t) = u_t^*(0)$ to the k^{th} agent.*

4. *Repeat from step 2, until the k^{th} agent reaches x_f.*

Besides being applicable to problems with free final time, CLP can be "extended" to settings with partially constrained final states.

Algorithm 3.3 (Modified Continuous Local Pursuit): *Identify the starting state x_s on $\mathbb{D}$ and the constraint set S_Q. Let $x_0(t)$ $(t \in [0, T_0])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_s,\ Q(x_0(T_0)) = 0$. Choose $0 < \Delta \leq T_0$.*

1. *For $k = 1, 2, 3 \ldots$, let $t_k = k\Delta$ be the starting time of k^{th} agent. Let $u_k(t) = 0, x_k(t) = x_s$ for $0 \leq t \leq t_k$.*

2. *For all* $t \geq t_k$, *calculate* $u_t^*(\tau)$ *for all* $t \in [t_k, t_k+T_k]$ *such that* $f(\hat{x}_k(\tau), u_t^*(\tau)) = \dot{\hat{x}}_t(\tau)$, *and* $\hat{x}_t(\tau)$ *achieves:*

$$\begin{cases} \eta_F(x_k(t), x_{k-1}(t), t), \tau \in [t, t+\Gamma^*(x_k(t), x_{k-1}(t))] & \text{if } x_{k-1}(t) \notin S_Q \\ \eta_Q(x_k(t), t), \tau \in [t, t+\Gamma_Q^*(x_k(t), S_Q) & \text{if } x_{k-1}(t) \in S_Q \end{cases}.$$

3. *Apply* $u_k(t) = u_t^*(0)$ *to the* k^{th} *agent.*

4. *Repeat from step 2, until the* k^{th} *agent reaches* S_Q.

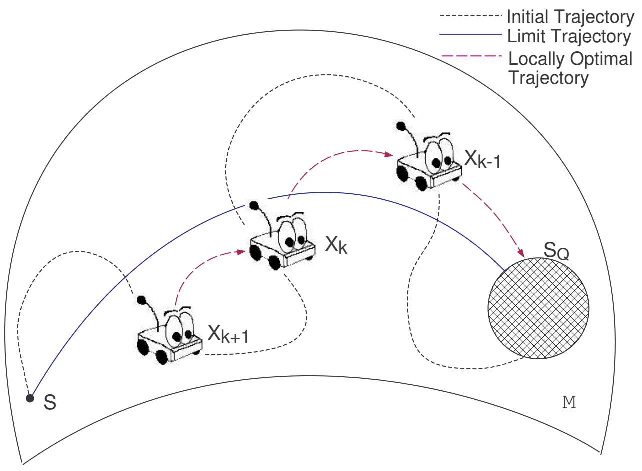

Figure 3.3: Illustration of local pursuit on a manifold. Each agent – except the first – is trying to "catch up" its predecessor before the predecessor reaches S_Q, and trying to reach S_Q via a locally optimal trajectory thereafter.

As Step 2 of the algorithm indicates, in modified continuous local pursuit (mCLP) there are two types of follower movements, "catching up" and "free running", depending on whether the leader has reached the final constraint set S_Q. The former lets agents "learn" from their leaders, while the "free running" stage enables them to find the optimal final state within S_Q once they are close enough to that set. Both stages will be essential in order for the group to solve Problem Eq. (3.2). The process of mCLP is illustrated in Fig. 3.3.

Finally, if we fix the final time in mCLP, or set the final state in SLP to partially-constrained, we obtain the modified sampled local pursuit (mSLP) algorithm:

Algorithm 3.4 (Modified Sampled Local Pursuit): *Identify the starting state x_s on $\mathbb{D}$ and the constraint set S_Q. Let $x_0(t)$ $(t \in [0, T_0])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_s,\ Q(x_0(T_0)) = 0$. Choose the pursuit interval Δ and updating interval δ such that $0 < \delta < \Delta \leq T_0$.*

1. *For $k = 1, 2, 3\ldots$, let $t_k = k\Delta$ be the starting time of the k^{th} agent, i.e., $u_k(t) = 0,\ x_k(t) = x_s$ for $0 \leq t \leq t_k$.*

2. *When $t = t_k + i\delta, i = 0, 1, 2, 3, \ldots$, calculate the control $u^*_t(\tau)$ that achieves:*
$$\begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta), \tau \in [t, t+\Delta] & \textit{if } \ x_{k-1}(t) \notin S_Q \\ \eta_Q(x_k(t), t, T-(t-t_k)), \tau \in [t, t_k+T] & \textit{if } \ x_{k-1}(t) \in S_Q \end{cases}.$$

3. *Apply $u_k(t) = u^*_{t_k+i\delta}(t)$ to the k^{th} agent for $t \in [t_k+i\delta, t_k+(i+1)\delta)$ if $\Delta + i\delta < T$, or for $t \in [t_k + i\delta, t_k + T)$ otherwise.*

4. *Repeat from step 2 until the k^{th} agent reaches S_Q.*

The applicability of the four algorithms which we have so far defined is, summarized in Table 3.1. The selection of proper algorithm depends on the problems to be solved. The followings are properties which are common to all four versions of local pursuit.

Table 3.1: Variations of local pursuit according to problem type

	Fixed Final Time	**Free Final Time**
Fixed Final States	Sampled Local Pursuit	Continuous Local Pursuit
Partially-Constrained Final States	modified Sampled Local Pursuit	modified Continuous Local Pursuit

3.4 Algorithm Advantages

Recall that solving the optimal control problems we have posed by a single system requires significant sensing, communication and computing capabilities. However, the algorithms described above, require each agent that participates in local pursuit to calculate optimal trajectories from itself to its nearby leader. Meanwhile the "distance" separating them can be limited by choice of the following interval Δ. Therefore every agent only needs to sense the environment within a limited region when in pursuit processes. This is preferable to obtaining a global map via random exploration with limited sensor range. There is no requirement for agents to exchange or fuse local maps that they sense in local pursuit, either. Agents only have to communicate in limited ways, e.g., using vision to track one another or by communicating in primitive ways to signal their locations, e.g., sound or radio transmission. In local pursuit, agents do not even need to agree on a common coordinate system. In fact, they do not need to know the coordinates of the target state/set! All that is required is an initial set of instructions for getting there. Although this is tentatively an open loop signal $u(t)$, it could also be given in other forms, e.g., using landmarks to reach the goal state. Finally, local pursuit only requires computing optima within small regions so that fewer resources are needed.

As we will show in the next chapters, local pursuit provides a tool for obtaining locally optimal trajectories over long distances by solving the problem in many short "pieces". This is accomplished by cooperation among an ordered sequence of identical agents, and requires only local knowledge about the environment as well as calculation of optimal trajectories within small regions. The trade-off is that each agent must solve multiple instances of the optimal control problem (although each instance is expected to be easier to solve than the overall problem).

Chapter 4

Main Results

In this chapter we investigate the collective behavior of a group whose members satisfy Eq. (3.1) and evolve under local pursuit. Recall that each of the pursuit algorithms presented in Chapter 3 defines an ordered sequence of trajectories $\{x_k(t)\}$. We will first explore the convergence of that sequence involved under sampled local pursuit (SLP). The limiting trajectory will be proved to be locally optimal, yielding the collective property we are seeking to obtain. Similar results will be proven for modified continuous local pursuit (mCLP).

4.1 Evolution under Sampled Local Pursuit

We begin by investigating the properties of the limiting trajectory generated by the group, i.e., $x_k(t)$ as $k \to \infty$. The convergence of the trajectories' cost will be explored first, followed by the convergence of trajectories themselves, $\{x_k(t)\}$. We will then show that the limiting trajectory, denoted by $x_\infty(t)$, is locally optimal.

Lemma 4.1 (Convergence of Cost in SLP): *Assume a group of agents — copies of Eq. (3.1) —* $x_1, \ldots, x_k$ *evolve under SLP with starting state* x_s *and target state* x_f*. Suppose an initial control/trajectory pair,* $\{u_0(t), x_0(t)\}$ $(t \in [0, T])$*, satisfying* $x_0(t) = x_s$ *and* $x_0(T) = x_f$ *is given. If the SLP updating time satisfies* $0 < \delta \leq \Delta$*, then the cost of the iterated trajectories converges, i.e.,* $\lim_{k\to\infty} C(x_k, t_k, T)$ *exists.*

Proof: It is enough to show that the cost of the iterated trajectories is non-increasing with k. Consider the pursuit process between the $(k-1)^{th}$ and k^{th} agents. The dotted line in Fig. 4.1 indicates the leader's path, $x_{k-1}(t)$ on $[t_{k-1}, t_{k-1} + T]$. The solid lines, denoted by $x_k(t)$, are the trajectories of the "follower", and the dashed lines, denoted by $\hat{x}_k(t)$, are the follower's planned trajectories, as defined in Section 3.3. We also use $\tilde{x}(t)$ to denote the trajectory segment during which $\tilde{x}_k(t + \Delta) = x_{k-1}(t)$, i.e., the follower "copies" the leader's trajectory, with a delay of Δ.

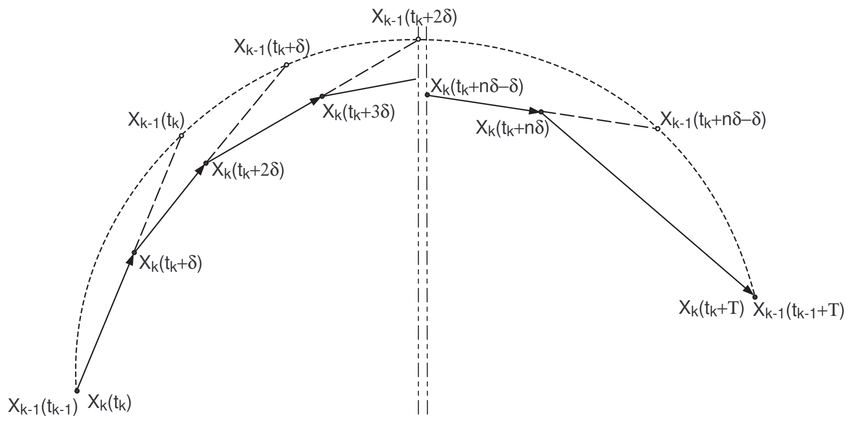

Figure 4.1: Illustrating Sampled Local Pursuit

The follower leaves the starting state at time t_k, while the leader leaves it at time t_{k-1}, where $t_k = t_{k-1} + \Delta$. For $t \in [t_k, t_k + \delta]$, the follower moves on an optimal trajectory from state $x_k(t_k)$ to $x_{k-1}(t_k)$ over Δ units of time. Thus from Fact 1:

$$\begin{aligned} \eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \Delta) &\leq C(\tilde{x}_k, t_k, \Delta) \\ &= C(x_{k-1}, t_{k-1}, \Delta). \end{aligned} \tag{4.1}$$

The right-hand side is the cost along the leader's path for the first Δ units of time wile the left-hand side is the optimal cost from $x_k(t_k)$ to $x_{k-1}(t_k)$.

At time $t_k+\delta$ the follower reaches the state $x_k(t_k+\delta)$. Recall that the trajectory driven by $u^*_{t_k}(\tau)$ is optimal from $x_k(t_k)$ to $x_{k-1}(t_k)$ and that from Fact 3, we can divide the cost into two parts, actual and planned[1], i.e.,

$$\begin{aligned} &\eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \Delta) \\ = \;& \eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) + \eta(x_k(t_k + \delta), x_{k-1}(t_k), \Delta - \delta, t_k + \delta, \Delta - \delta). \end{aligned} \tag{4.2}$$

From equations (4.1),(4.2), we obtain:

$$\begin{aligned} &\eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) \\ \leq \;& C(x_{k-1}, t_{k-1}, \Delta) - \eta(x_k(t_k + \delta), x_{k-1}(t_k), \Delta - \delta, t_k + \delta, \Delta - \delta). \end{aligned} \tag{4.3}$$

At time $t_k + \delta$, the follower updates its trajectory to intercept the leader at its new location $x_k(t_k + \delta)$. Because this trajectory is optimal from $x_k(t_k + \delta)$ to $x_{k-1}(t_k + \delta)$ over time Δ, any path $x_k(t)$ ($t \in [t_k + \delta, t_k + \delta + \Delta]$) from $x_k(t_k + \delta)$ to

[1]These two pieces are both optimal with respect to their corresponding end points.

$x_{k-1}(t_k+\delta)$ over time Δ passing through $x_{k-1}(t_k)$ at time $t_k+\Delta = t_k+\delta+\Delta-\delta$ has equal or greater cost. From Fact 2 it follows that:

$$\begin{aligned}
& \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \Delta) \\
\leq\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + \eta(x_{k-1}(t_k), x_{k-1}(t_k+\delta), \delta, t_k+\Delta, \delta) \\
\leq\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + C(\tilde{x}_k, t_k+\Delta, \delta) \\
=\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + C(x_{k-1}, t_k, \delta). \qquad (4.4)
\end{aligned}$$

We can also divide the cost $\eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \Delta)$ into a realized part and a planned one, i.e.,

$$\begin{aligned}
& \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \Delta) \\
=\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \delta) \\
& + \eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta). \qquad (4.5)
\end{aligned}$$

From Eq. (4.1)$\sim$(4.5), we obtain

$$\begin{aligned}
& C(x_k, t_k, 2\delta) \\
=\ & \eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) + \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \delta) \\
\leq\ & C(x_{k-1}, t_{k-1}, \Delta) + C(x_{k-1}, t_k, \delta) - \eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta) \\
=\ & C(x_{k-1}, t_{k-1}, \Delta+\delta) - C(\hat{x}_k, t_k+2\delta, \Delta-\delta), \qquad (4.6)
\end{aligned}$$

where $\eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta) = C(\hat{x}_k, t_k+2\delta, \Delta-\delta)$ is from the fact that the planned trajectory is optimal.

We repeat this procedure until $t = t_k+n\delta$ where $\Delta+(n-1)\delta < T$ and $\Delta+n\delta \geq$

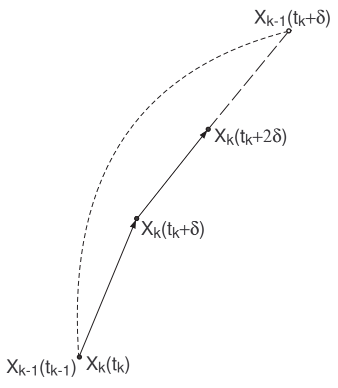

Figure 4.2: First two steps in sampled local pursuit

T. This choice of n means that the leader has not reached the final state, and

$$\begin{aligned} C(x_k, t_k, n\delta) &= \sum_{i=0}^{n-1} \eta(x_k(t_k + i\delta), x_{k-1}(t_k + i\delta), \Delta, t_k + i\delta, \delta). \\ &\leq C(x_{k-1}, t_{k-1}, \Delta + (n-1)\delta) - C(\hat{x}_k, t_k + n\delta, \Delta - \delta). \quad (4.7) \end{aligned}$$

When $t \in [t_k + n\delta, t_k + T]$, the leader reaches the final state and stays static. During this time period, no matter how many times the follower updates its movement, it will evolve on the same path that was determined at time $t = t_k + n\delta$. This path, which is indicated by the last solid line in Fig. 4.1, is locally optimal between the states $x_k(t_k + n\delta)$ and $x_k(t_k + T)$ over $T - n\delta$ units of time. Therefore

$$\begin{aligned} & C(x_k, t_k + n\delta, T - n\delta) \\ = \;& \eta(x_k(t_k + n\delta), x_{k-1}(t_{k-1} + T), T - n\delta, t_k + n\delta, T - n\delta) \\ \leq \;& C(\hat{x}_k, t_k + n\delta, \Delta - \delta) + C(x_{k-1}, t_k + (n-1)\delta, T - (n-1)\delta - \Delta). \quad (4.8) \end{aligned}$$

From Eq. (4.7)∼(4.8), we obtain

$$\begin{aligned} C(x_k, t_k, T) &\leq C(x_{k-1}, t_{k-1}, \Delta + (n-1)\delta) + C(x_{k-1}, t_k + (n-1)\delta, T - (n-1)\delta - \Delta) \\ &= C(x_{k-1}, t_{k-1}, T). \quad (4.9) \end{aligned}$$

We have shown that cost incurred by the follower is no greater than the leader's. Writing $C_k = C(x_k, t_k, T)$ for convenience, we can see that $C_k \leq C_{k-1}$. Obviously C_k is bounded below if there exits an optimal trajectory from the starting state to the target state. Hence we conclude that

$$\lim_{k \to \infty} C_k = C. \tag{4.10}$$

$\square$

Of course, convergence of the trajectories' cost does not automatically imply the convergence of the trajectories themselves. For example, if there exist multiple locally optimal trajectories connecting the leader and follower at the updating times, then the convergence of trajectories is not guaranteed, and Lemma 4.1 defines an equivalence class of trajectories with equal cost.

If we restrict the pursuit process to take place within a "small" region by selecting Δ sufficiently small, e.g., agents follow close to one another, there will exist a unique locally optimal trajectory from the follower to the leader at every updating time $t_k + i\delta$. Under that hypothesis, we obtain the following result:

Lemma 4.2 (Uniqueness of the Limiting Trajectory): *If at each updating time, the locally optimal trajectory obtained through SLP is unique, then the limiting trajectory $x_\infty(t)$ is also unique.*

Proof: Suppose there exist more than one limiting trajectories, and suppose $x_1(t)$ and $x_2(t)$ are two possibilities. Suppose also that $x_1(t)$ differs from $x_2(t)$ for $t \in$

$[t_1, t_2] \cup [t_3, t_4] \ldots$. From Lemma 4.1 these two trajectories must have the same cost.

Let the leader $x_{k-1}(t)$ travel along $x_1(t)$, while the follower $x_k(t)$ travels along $x_2(t)$. If no update occurs during $[t_1, t_2]$, then $x_2(t)$ has less cost during $[t_1, t_2]$ because the follower moves along $x_2(t)$ and the local optimum is unique. The same arguments applied to subsequent time periods leads to the fact that the total cost along $x_2(t)$ is less than that along $x_1(t)$ if no update occurs during $t \in [t_1, t_2] \cup [t_3, t_4] \ldots$, which contradicts to the fact that $x_1(t)$ and $x_2(t)$ have equal costs.

Next, assume that only one update occurs during $[t_1, t_2]$, as Fig. 4.3 indicates. Separate the curves during $[t_1, t_2]$ into several segments (the meaning of different

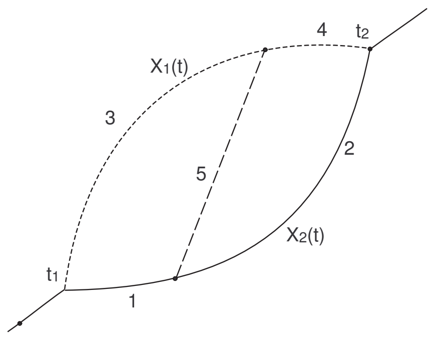

Figure 4.3: There is one update between two trajectories

curve style is the same as in Lemma 4.1), and indicate the cost along curve i by C_i. From the uniqueness of local optimum, we have $C_1 + C_5 < C_3$ and $C_2 < C_5 + C_4$. Therefore $C_1 + C_2 < C_3 + C_4$, which means that $x_2(t)$ has less cost than $x_1(t)$ during $[t_1, t_2]$.

If there are multiple updates during $[t_1, t_2]$, we can see that the updates do not change the fact that the cost along $x_2(t)$ is less than that along $x_1(t)$. Hence we still get the result that the cost along $x_2(t)$ is less than $x_1(t)$ for $t \in [t_1, t_2]$, no matter how many updates occur.

By applying the same argument on the subsequent periods we are led to the fact that the total cost along $x_2(t)$ must be less than that along $x_1(t)$. $\square$

Even if the locally optimal trajectories obtained at every updating time are smooth, e.g., solutions to a set of Euler-Lagrange equations, $x_k(t)$ is only known to be piecewise smooth. For example, in $\mathbb{R}^2$ with $\dot{x}_k = u_k$, if the locally optimal trajectories are straight lines, $x_k(t)$ is not smooth because there exists a "corner" at the joint of two segments. However, we can show that the limiting trajectory is smooth in the time interval $[0, T]$ if the locally optimal trajectories obtained at every updating time are smooth. The following definitions will be necessary for discussing the properties of the limiting trajectory.

Definition 4.1: *Let $\gamma_1(t)$ and $\gamma_2(t)$ be trajectories of Eq. (3.1), defined on time intervals I_1 and I_2, respectively, where $I_1 \cap I_2 \neq \emptyset$. We say that γ_1 and γ_2 **overlap** if $\gamma_1(t) = \gamma_2(t)$ for all $t \in I_1 \cap I_2$.*

Definition 4.2: *Let $\gamma_1(t)$ and $\gamma_2(t)$ be trajectories of Eq. (3.1), defined on a time interval I_1 and another time interval I_2, respectively, where $I_1 \cap I_2 \neq \emptyset$. The **composition** of $\gamma_1(t)$ and $\gamma_2(t)$ on the interval $I_1 \cup I_2$ is defined as*

$$\gamma_1 \circ \gamma_2 \triangleq \begin{cases} \gamma_1(t) & t \in I_1, t \notin I_2 - I_1 \cap I_2 \\ \gamma_2(t) & t \notin I_1, t \in I_2 - I_1 \cap I_2 \end{cases}.$$

Lemma 4.3 (Smoothness of Composition): *Suppose that in Lemma 4.1 the updating interval δ and following interval Δ satisfy $0 < \delta < \Delta$. Then, the planned trajectory $\hat{x}(t)$ and realized trajectory $x(t)$ of an agent whose leader evolves along the limiting trajectory overlap. Furthermore, if the locally optimal trajectories obtained at every updating time are smooth, the limiting trajectory is also smooth.*

Proof: Let a leader evolve along the limiting trajectory $x_\infty(t)$ and suppose it is the $(k-1)^{th}$ agent. From Lemma 4.2 we know that the limiting trajectory means that $x_{k-1}(t) = x_k(t+\Delta)$ for $\forall t \in [t_k, t_k + T]$.

At first we claim that in the time interval $[t_k+\delta, t_k+\Delta]$, the planned trajectory "agrees" with the realized one, i.e., $\hat{x}_k(t) = x_k(t), t \in [t_k+\delta, t_k+\Delta]$. To see why that has to be the case, suppose that $\hat{x}_k(t) \neq x_k(t)$ for some $t \in [t_k+\delta, t_k+\Delta]$. Because $\hat{x}(t)$ is optimal from $x_k(t_k+\delta)$ to $x_k(t_k+\delta+\Delta)$, the trajectory

$$\bar{x}(t) = \begin{cases} \hat{x}_k(t) & t \in [t_k+\delta, t_k+\Delta) \\ x_k(t) & t \in [t_k+\Delta, t_k+\delta+\Delta] \end{cases}$$

has a lower cost than $x_k(t)$ $(t \in [t_k+\delta, t_k+\delta+\Delta])$, which is updated by the follower at the time $t = t_k+\delta$ and is supposed to be optimal from $x_k(t_k+\delta)$ to $x_k(t_k+\delta+\Delta)$. This is a contradiction. Therefore, we obtain $\hat{x}_k(t) = x_k(t)$ for $\forall t \in [t_k+\delta, t_k+\Delta]$. The same argument applies to the subsequent time periods $[t_k+i\delta, t_k+i\delta+\Delta]$.

The trajectory $\bar{x}(t)$ is smooth for $t \in [t_k, t_k+\Delta]$ because the locally optimal trajectory is smooth. For the same reason, $x_k(t)$ is smooth for $t \in [t_k+\delta, t_k+\delta+\Delta]$ (second update step). We also know that $\hat{x}_k(t) = x_k(t)$ for $\forall t \in [t_k+\delta, t_k+\Delta]$. Thus the actual trajectory $x_k(t)(t \in [t_k, t_k+2\delta])$ is smooth. Repeating the argument for the next time intervals leads to the result that the entire trajectory $x_k(t)$ $(t \in$

$[t_k, t_k + T]$) is smooth. □

Before proceeding to the main theorem, we define the following Lipschitz-like condition about the optimal cost with respect to changes in the problem's boundary conditions:

Condition 4.1: *Assume that there exists an $\varepsilon > 0$ such that for all $a, b_1, b_2 \in \mathbb{D}$ and all $\Delta > 0$, the optimal cost $\eta(a, b_1, \Delta, 0, \Delta)$ from a to b_1 and $\eta(a, b_2, \Delta, 0, \Delta)$ from a to b_2 satisfy*

$$\|b_1 - b_2\| < \varepsilon \Rightarrow \|\eta(a, b_1, \Delta, 0, \Delta) - \eta(a, b_2, \Delta, 0, \Delta)\| < \mathcal{L}\Delta \tag{4.11}$$

for some constants $\mathcal{L}$ independent of Δ.

A piecewise locally optimal trajectory is not necessarily optimal. However, the composition of overlapping locally optimal trajectories is locally optimal if Condition 4.1 is satisfied.

Lemma 4.4 (Composition of Optimal Trajectories): *Let $\gamma_1(t)$ and $\gamma_2(t)$ be overlapping locally optimal trajectories defined on time intervals I_1 and I_2 respectively, where $I_1 \cap I_2 \neq \emptyset$. If Condition 4.1 is satisfied, then the composition $\gamma_1 \circ \gamma_2$ is locally optimal on $I_1 \cup I_2$.*

Proof: We will prove that that if $x^*(t)$ ($t \in [0, t_1 + \Delta_1]$) and $x^*(t)$ ($t \in [t_1, T]$) are two locally optimal trajectories, and Condition 4.1 is satisfied, where $0 < t_1 <$

$t_1 + \Delta_1 < T$, then the trajectory $x^*(t), t \in [0, T]$ is a local minimum.

We take $0 < \Delta \leq \Delta_1$. From the principle of optimality, we obtain that $x^*(t)(t \in [0, t_1 + \Delta])$ and $x^*(t)(t \in [t_1, T])$ are two locally optimal trajectories with respect to their corresponding end points.

Suppose that $x^*(t)(t \in [0, T])$ is not the local minimum, then there must exist an $\epsilon < \varepsilon$ and another optimum $x(t) \in \mathbb{D} \times [0, T]$ satisfying $\|x(t) - x^*(t)\|_\infty < \epsilon$ and $C(x(t), 0, T) < C(x^*(t), 0, T)$, as Fig. 4.4 shows.

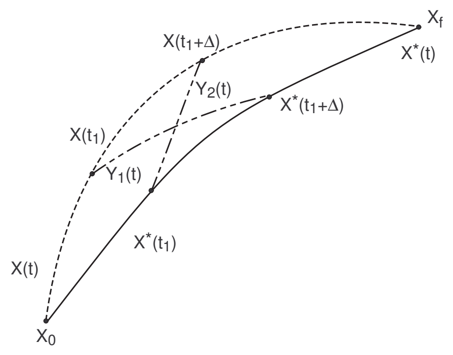

Figure 4.4: Overlapped local minimums lead to the local minimum overall

Construct two optimal trajectories $y_1(t), y_2(t), t \in [t_1, t_1 + \Delta]$ connecting $x(t)$ and $x^*(t)$ such that $x^*(t_1) = y_2(t_1), x^*(t_1+\Delta) = y_1(t_1+\Delta), x(t_1) = y_1(t_1), x(t_1+\Delta) = y_2(t_1 + \Delta)$. From the principle of optimality, $x^*(t)$ and $x(t)$ $(t \in [t_1, t_1 + \Delta])$ are both optimal trajectories with respect to their corresponding end points. Now using Eq. (4.11), we obtain

$$\begin{aligned} C(y_1(t), t_1, \Delta) &< C(x(t), t_1, \Delta) + \mathcal{L}\Delta, \\ C(y_2(t), t_1, \Delta) &< C(x^*(t), t_1, \Delta) + \mathcal{L}\Delta. \end{aligned} \tag{4.12}$$

For $x^*(t)$ $(t \in [0, t_1 + \Delta])$ and $x^*(t)$ $(t \in [t_1, T])$ are two unique local optimal trajectories, we have

$$\begin{aligned}
& C(x^*(t), 0, t_1) + C(x^*(t), t_1, \Delta) \\
< \;& C(x(t), 0, t_1) + C(y_1(t), t_1, \Delta), \\
& C(x^*(t), t_1, \Delta) + C(x^*(t), t_1 + \Delta, T - t_1 - \Delta) \\
< \;& C(x(t), t_1 + \Delta, T - t_1 - \Delta) + C(y_2(t), t_1, \Delta). \qquad (4.13)
\end{aligned}$$

Combining Eq. (4.12) and Eq. (4.13) leads to

$$C(x^*(t), 0, T) + C(x^*(t), t_1, \Delta) < C(x(t), 0, T) + C(x^*(t), t_1, \Delta) + 2\mathcal{L}\Delta,$$

which can be expressed as

$$C(x^*(t), 0, T) < C(x(t), 0, T) + 2\mathcal{L}\Delta. \qquad (4.14)$$

$C(x(t), 0, T)$ is assumed to be less than $C(x^*(t), 0, T)$, but if we take

$$0 < \Delta < \frac{C(x^*(t), 0, T) - C(x(t), 0, T)}{2\mathcal{L}},$$

we see that Eq. (4.14) cannot be true. Thus, there is a contradiction, because Δ could be set to be arbitrarily small. It follows that $x^*(t)$ ($t \in [0, T]$) must be the local minimum. □

The next theorem is an immediate consequence of the above lemmas.

Theorem 4.1 (Sampled Local Pursuit): *Suppose that a group of agents* $\{x_k\}$ *evolve under SLP and that at each updating time* $t = t_k + i\delta$, *the locally optimal* *trajectory from $x_k(t)$ to $x_{k-1}(t)$ is unique. If the updating interval and following interval satisfy $0 < \delta < \Delta$ and Condition 4.1 is satisfied, then the trajectory sequence $x_k(t)$ converges to a unique local optimum. Furthermore, if the locally optimal trajectories at every updating time are smooth, the limiting trajectory is also smooth.*

Proof: From Lemma 4.2, the limiting trajectory is unique. We know that $x_\infty(t)$ $(t \in [0, \Delta))$ and $x_\infty(t)$ $(t \in [\delta, \delta + \Delta))$ are locally optimal for the realized trajectory and planned trajectories overlap (Lemma 4.3). The optimality of $x_\infty(t)$ $(t \in [0, \delta + \Delta))$ follows from Lemma 4.4. Repeating this argument on $[i\delta, i\delta + \Delta]$ $(i = 0, 1, 2 \ldots)$ leads to the result that $x_\infty(t)$ $(t \in [0, T])$ is locally optimal. The proof of smoothness follows from a similar argument. $\square$

4.2 Evolution under Modified Continuous Local Pursuit

This section explores the behavior of the group under modified continuous local pursuit (mCLP). Recall that mCLP differs from SLP in that it includes a "free running" stage and can handle problems with free final time and partially-constrained final states. Our first task is to show that a single SLP-like update on the leader's trajectory decreases the trajectory's cost.

Lemma 4.5 *Consider a leader-follower pair evolving under mCLP with a pursuit* *interval Δ. Let the leader's trajectory be $x_{k-1}(t)$ ($t \in [t_{k-1}, t_{k-1}+T_{k-1}]$) and fix $\lambda \in [0, T_{k-1})$. Suppose the follower updates its trajectory only once during $[t_k, t_k+T_k]$ in the manner described next:*

- *If $\lambda < T_{k-1} - \Delta$, the follower moves along the optimal trajectory (in the sense of Eq. (3.7)) joining $x_k(t_k+\lambda)$ and $x_{k-1}(t_k+\lambda)$ with optimal final time $\Gamma = \Gamma^*(x_k(t_k+\lambda), x_{k-1}(t_k+\lambda))$. During other times, the follower replicates the leader's trajectory, i.e.,*

$$\begin{cases} x_k(t) = x_{k-1}(t-\Delta) & t \in [t_k, t_k+\lambda] \\ x_k(t) = x_{k-1}(t-\Gamma) & t \in [t_k+\lambda+\Gamma, t_k+T_k] \end{cases},$$

- *If $\lambda \geq T_{k-1} - \Delta$, the follower evolves along the optimal trajectory from $x_k(t_k+\lambda)$ to the constraint set S_Q (in the sense of Eq. (3.8)). Similarly, during other times*

$$x_k(t) = x_{k-1}(t-\Delta) \quad t \in [t_k, t_k+\lambda],$$

Then the cost along the follower's trajectory will be no greater than the leader's.

Proof: First, choose $\lambda < T_{k-1} - \Delta$. Starting at time $t_k+\lambda$ and during $t \in [t_k+\lambda, t_k+\lambda+\Gamma]$, the follower moves on the locally optimal trajectory $x_k(t)$ (see Fig. 4.5). The cost along x_k is

$$\begin{aligned} & C_Q(x_k, t_k, T_k) \\ = \ & C_Q(x_k, t_k, \lambda) + C_Q(x_k, t_k+\lambda+\Gamma, T_k-\lambda-\Gamma) + \eta_F(x_k(t_k+\lambda), x_{k-1}(t_k+\lambda), t_k+\lambda) \\ \leq \ & C_Q(x_{k-1}, t_{k-1}, \lambda) + C_Q(x_{k-1}, t_{k-1}+\lambda, \Delta) + C_Q(x_{k-1}, t_{k-1}+\lambda+\Delta, T_{k-1}-\lambda-\Delta) \\ = \ & C_Q(x_{k-1}, t_{k-1}, T_{k-1}), \end{aligned} \tag{4.15}$$

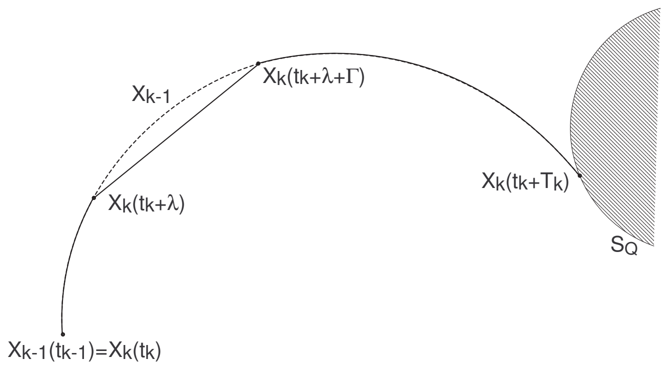

Figure 4.5: Illustration of the trajectory obtained by a single update when $\lambda < T_{k-1} - \Delta$.

where $\Gamma = \Gamma^*(x_k(t_k+\lambda), x_{k-1}(t_k+\lambda))$. If $\lambda \geq T_{k-1} - \Delta$ (see Fig. 4.6), the cost along

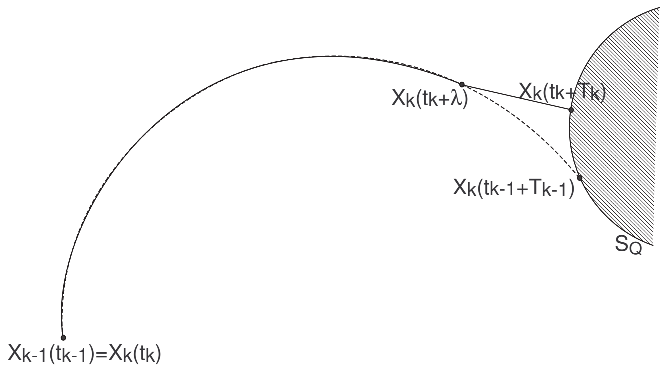

Figure 4.6: Illustration of the trajectory obtained by a single update when $\lambda \geq T_{k-1} - \Delta$.

x_k is

$$\begin{aligned} & C_Q(x_k, t_k, T_k) \\ = \; & C_Q(x_k, t_k, \lambda) + \eta_Q(x_k(t_k+\lambda), t_k+\lambda) \\ \leq \; & C_Q(x_{k-1}, t_{k-1}, \lambda) + C_Q(x_{k-1}, t_{k-1}+\lambda, T_{k-1}-\lambda) \\ = \; & C_Q(x_{k-1}, t_{k-1}, T_{k-1}). \end{aligned}$$

Therefore the cost along the follower's trajectory is no greater than the leader's. $\square$

Now, the cost of the iterative trajectories can be shown to converge under mCLP:

Lemma 4.6 (Convergence of Cost): *If the agents governed by Eq. (3.1) evolve under mCLP, then the cost of the iterated trajectories converges.*

Proof: Let C_{k-1} be the cost along the leader's trajectory $x_{k-1}(t)$ $(t \in [t_{k-1}, t_{k-1} + T_{k-1}])$. Define a trajectory sequence $x_k^i(t)$ $(t \in [t_k, t_k + T_k^i]), i = 0, 1, 2 \ldots,$ whose corresponding costs and final times are C_k^i and T_k^i, as follows: let $x_k^0(t) = x_{k-1}(t)$ (the trajectory of a "leader") and let x_k^i $(i > 0)$ be the trajectory of an agent that pursues x_k^{i-1} by performing only a *single trajectory update*, as described in Lemma 1, with $\lambda = (i-1)\delta,\ \delta > 0$ (see Fig. 4.7).

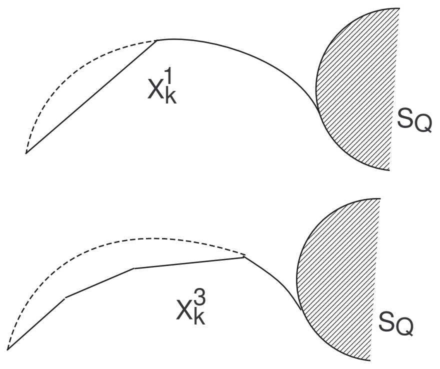

Figure 4.7: Illustration of the trajectory sequence $x_k^i(t)$. Each trajectory is obtained by a single update upon its predecessor.

From Lemma 4.5, the cost of each follower's trajectory will be no greater than the leader's. Also, the sequence C_k^i is bounded below for fixed k. Thus, $C_k^i \leq C_k^{i-1}$

and $\lim_{k\to\infty} C_k^i = C_k^\infty$ exists for each k. Consequently,

$$C_k^\infty \le C_k^0 = C_{k-1}.$$

Now, take $\delta = T_{k-1}/i$, so that $\delta \to 0$ as $i \to \infty$. At the limit, the trajectory $x_k^\infty(t)$ is precisely what would be obtained by an agent that pursues its leader x_{k-1}, using mCLP. Hence, the follower's cost is $C_k = C_k^\infty \le C_{k-1}$. Because the sequence $\{C_k\}$ is non-increasing and bounded below (there exists a minimum for Eq. (3.6)), it must converge to a limit. □

Now Condition 4.1 is modified to:

Condition 4.2: *Assume that for a generic trajectory $x_1(t)$ there exists $\varepsilon > 0$ such that for all $a, b_1, b_2 \in \mathbb{D}$ and all $\Delta > 0$, there exists a trajectory $x_2(t)$ such that the cost $C_Q(x_1, 0, \Delta)$ $(x_1(0) = a, x_1(T) = b_1)$ from a to b_1 and cost $C_Q(x_2, 0, \Delta)$ $(x_2(0) = a, x_2(T) = b_2)$ from a to b_2 satisfy*

$$\|b_1 - b_2\|_\infty < \varepsilon \Rightarrow \|C_Q(x_1, 0, \Delta) - C_Q(x_2, 0, \Delta)\|_\infty < \mathcal{L}\Delta$$

for some constant $\mathcal{L}$, independent of Δ.

Then the next lemma holds:

Lemma 4.7: *Let $x^*(t)$ be a trajectory of Eq. (3.1) such that:*

1. *$x^*(t)$ $(t \in [0, t_1 + \Delta_1])$ is optimal (in the sense of Eq. (3.7)) from $x^*(0)$ to $x^*(t_1 + \Delta_1)$.*
2. *$x^*(t)$ $(t \in [t_1, T^*])$ is optimal (in the sense of Eq. (3.8)) from $x^*(t_1)$ to the constraint set S_Q.*

Assume Condition 4.2 is satisfied and $0 < t_1 < t_1 + \Delta_1 < T^*$. *Then the trajectory* $x^*(t)$ $(t \in [0, T^*])$ *is a local minimum of Eq. (3.8) from* $x^*(0)$ *to* S_Q.

Proof: Choose $0 < \Delta \leq \Delta_1$. From the principle of optimality, $x^*(t)$ $(t \in [0, t_1 + \Delta])$ and $x^*(t)$ $(t \in [t_1, T^*])$ are locally optimal with respect to their corresponding end points. Suppose $\|x^*(t_1 + \Delta) - s\|_\infty \geq \varepsilon_1$ for any $s \in S_Q$ and that $x^*(t)$ $(t \in [0, T^*])$ is not a local minimum. There must exist $\epsilon < \min(\varepsilon, \varepsilon_1/2)$ (where ε is defined in Condition 1) and another optimum $x(t) \in \mathbb{D} \times [0, T]$ satisfying $\|x(t) - x^*(t)\|_\infty < \epsilon$ and $C_Q(x(t), 0, T) < C_Q(x^*(t), 0, T^*)$.

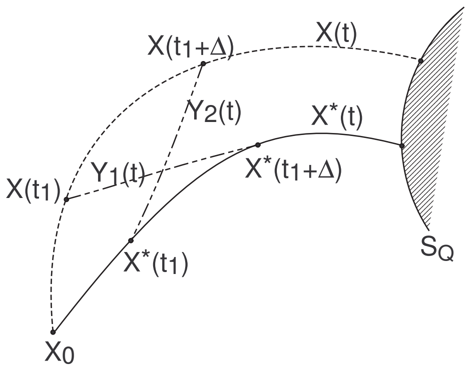

Figure 4.8: Illustrating the proof of Lemma 4.7: "overlapping" optimal trajectories form a locally optimal trajectory.

Notice that $\|x(t_1 + \Delta) - s\|_\infty \geq \epsilon$ for any $s \in S_Q$. Construct two trajectories $y_1(t), y_2(t)$ $(t \in [t_1, t_1 + \Delta])$ that connect $x(t)$ and $x^*(t)$ (see Fig. 4.8) and satisfy Condition 4.2 (with x^* or x playing the role of x_1, and y_1 or y_2 standing in for x_2). There must exist a Δ^* such that $y_1(t_1 + \Delta^*), y_2(t_1 + \Delta^*) \notin S_Q$ because $\|x(t_1 + \Delta) - s\|_\infty \geq \epsilon$ and $\|x^*(t_1 + \Delta) - s\|_\infty \geq \epsilon$ for any $s \in S_Q$. We pick $\Delta < \Delta^*$. In particular, let $y_1,\ y_2$ be such that $x^*(t_1) = y_2(t_1), x^*(t_1 + \Delta) = y_1(t_1 + \Delta), x(t_1) =$

$y_1(t_1), x(t_1+\Delta) = y_2(t_1+\Delta)$. Now ,Condition 4.2 implies that

$$\begin{aligned} C_Q(y_1(t), t_1, \Delta) &< C_Q(x(t), t_1, \Delta) + \mathcal{L}\Delta, \\ C_Q(y_2(t), t_1, \Delta) &< C_Q(x^*(t), t_1, \Delta) + \mathcal{L}\Delta. \end{aligned} \tag{4.16}$$

Because $x^*(t)$ $(t \in [0, t_1+\Delta])$ and $x^*(t)$ $(t \in [t_1, T^*])$ are each locally optimal, the following holds:

$$\begin{aligned} & C_Q(x^*(t), 0, t_1) + C_Q(x^*(t), t_1, \Delta) \\ < \; & C_Q(x(t), 0, t_1) + C_Q(y_1(t), t_1, \Delta), \end{aligned} \tag{4.17}$$

and

$$\begin{aligned} & C_Q(x^*(t), t_1, \Delta) + C_Q(x^*(t), t_1+\Delta, T^*-t_1-\Delta) \\ < \; & C_Q(x(t), t_1+\Delta, T-t_1-\Delta) + C_Q(y_2(t), t_1, \Delta). \end{aligned} \tag{4.18}$$

Combining Eq. (4.16) with Eq. (4.17,4.18) leads to

$$C_Q(x^*(t), 0, T) < C_Q(x(t), 0, T) + 2\mathcal{L}\Delta. \tag{4.19}$$

The cost $C(x(t), 0, T)$ is apparently less than $C(x^*(t), 0, T)$; but if Δ is chosen so that

$$0 < \Delta < \min\{\Delta_1, \Delta^*, \frac{C_Q(x^*(t), 0, T) - C_Q(x(t), 0, T)}{2\mathcal{L}}\},$$

then Eq. (4.19) cannot hold. This is a contradiction, because Δ could be chosen arbitrarily small. It follows that $x^*(t)$ $(t \in [0, T^*])$ must be a local minimum. □

Now let us assume that the locally optimal trajectory from the follower to the leader (or to S_Q) is unique at all times. Then, convergence of the trajectories' cost also implies convergence of the trajectories themselves:

Lemma 4.8: *If at all times during mCLP, the locally optimal trajectory from follower to leader (or to S_Q) is unique, then mCLP converges to a unique limiting trajectory $x_\infty(t)$.*

Proof: Suppose that the trajectories' cost converges but that there exist more than one limiting trajectory. Let $x_1(t)$ ($t \in [0, T_1]$) and $x_2(t)$ ($t \in [0, T_2]$) be two such possibilities. Let $t_1 \in [0, T_1]$ be the earliest time that $x_1(t)$ differs from $x_2(t)$. From Lemma 2, x_1 and x_2 must have the same cost, otherwise convergence of the cost is contradicted. Suppose that a leader $x_{k-1}(t)$ travels along $x_1(t)$, while a follower

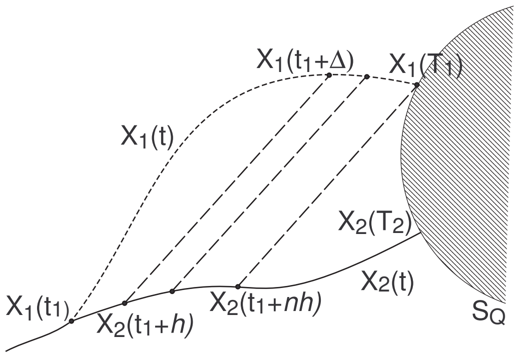

Figure 4.9: Illustrating the proof of Lemma 4: pursuit between agents moving on two supposed "limiting" equal-cost trajectories, leads to the conclusion that the cost along the follower's trajectory is less than that along the leader's.

$x_k(t)$ travels along $x_2(t)$. Choose $h > 0$ small, and that a series of sampled updates occur at $t_1 + ih$ ($i = 1, 2 \ldots, n = (T_1 - t_1 - \Delta)/h$), as Fig. 4.9 indicates.

Consider the update occurring at t_1, after which the follower moves on $x_2(t), t \in [t_1, t_1{+}h)$. This fact means that either i) the trajectory defined by $x_2(t), t \in [t_1, t_1{+}h)$ and the optimal trajectory from $x_2(t_1 + h)$ to $x_1(t_1 + \Delta)$ (as indicated by the left dashed line in Fig. 4.9) has less cost than $x_1(t), t \in [t_1, t_1 + \Delta)$, or ii) it has the same cost with $x_1(t), t \in [t_1, t_1 + \Delta)$. The latter contradicts the assumption that there exists a unique locally optimal trajectory from follower to leader at any time. Therefore the locally optimal trajectory that the follower actually calculated at t_1 has cost $C_Q(x_2, t_1, h) + \eta_F(x_2(t_1 + h), x_1(t_1 + \Delta), t_1 + h)$, and

$$\begin{aligned} & C_Q(x_2, t_1, h) + \eta_F(x_2(t_1 + h), x_1(t_1 + \Delta), t_1 + h) \\ < \; & C_Q(x_1, t_1, \Delta). \end{aligned} \tag{4.20}$$

Similarly, investigate the update occurring at $t_1 = ih$ $(i = 2, 3, \ldots, n)$, we obtain

$$\begin{aligned} & C_Q(x_2, t_1 + h, h) + \eta_F(x_2(t_1 + 2h), x_1(t_1 + \Delta + h), t_1 + 2h) \\ < \; & \eta_F(x_2(t_1 + h), x_1(t_1 + \Delta), t_1 + h) + C_Q(x_1, t_1 + \Delta, h), \\ \cdots \; & \cdots \\ & C(x_2, t_1 + (n-1)h) + \eta_F(x_2(t_1 + nh), x_1(T_1), t_1 + nh) \qquad (4.21) \\ < \; & \eta_F(x_2(t_1 + (n-1)h), x_1(T_1 - h), t_1 + (n-1)h) + C(x_1, T_1 - h, h). \end{aligned}$$

And at the last update step, the follower chooses the locally optimal trajectory from itself to S_Q:

$$\begin{aligned} & C_Q(x_2, t_1 + nh, T_2 - t_1 - nh) \\ < \; & \eta_F(x_2(t_1 + nh), x_1(T_1), t_1 + nh). \end{aligned} \tag{4.22}$$

Notice that the inequality does not depend on the size of h. No matter how small h is, it can always be concluded from Eq. (4.20)~(4.22) that

$$C_Q(x_2, t_1, T_2 - t_1) < C_Q(x_1, t_1, T_1 - t_1). \tag{4.23}$$

If we let $h \to 0$, the above sampled process will approach the continuous local pursuit. Because Eq. (4.23) holds regardless of the (decreasing) value of h, therefore the cost along $x_2(t)$ must be less than that for $x_1(t)$ under mCLP, which contradicts the convergence of the cost under mCLP. □

Lemma 4.9: *Along the limiting trajectory produced under mCLP, the planned trajectories $\hat{x}_k(t)$ and realized trajectories $x_k(t)$ overlap, i.e., $\hat{x}_k(t) = x_k(t)$. Furthermore, if the locally optimal trajectories obtained at every updating time are smooth, then the limiting trajectory is also smooth.*

Proof: Suppose also that a leader, x_{k-1} evolves along the limiting trajectory $x_\infty(t)$. Lemma 4 then implies that $x_{k-1}(t) = x_k(t + \Delta)$ for $\forall t \in [t_k, t_k + T_k]$.

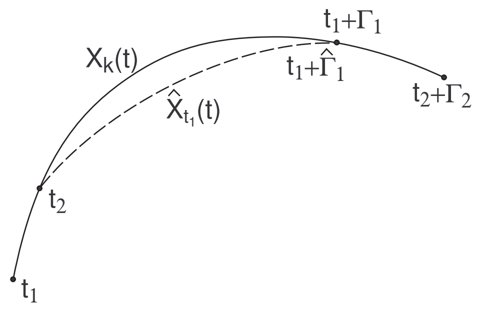

Figure 4.10: Differences between the planned and realized trajectories contradict the convergence of trajectories under mCLP.

Suppose that with the leader at $x_{k-1}(t_1 + \Gamma(t_1))$, where $\Gamma(t_1)$ is the best final time for update at t_1, and follower at $x_k(t_1)$, the planned trajectory $\hat{x}_{t_1}(t)$ ($t \in$

$[t_1, t_1 + \hat{\Gamma}(t_1))$ obtained at t_1 differs from $x_k(t)$ $(t \in [t_1, t_1 + \Gamma(t_1))$ starting at some time $t_2 \geq t_1$. Furthermore, let $\hat{x}_{t_1}(t_1 + \hat{\Gamma}(t_1)) = x(t_1 + \Gamma(t_1))$. Because the planned trajectory $\hat{x}_{t_1}(t)$ is unique (by assumption) and optimal,

$$C_Q(\hat{x}_{t_1}, t_2, \hat{\Gamma}(t_1) - (t_2 - t_1)) < C_Q(x_k, t_2, \Gamma(t_1) - (t_2 - t_1)).$$

Construct the trajectory

$$\bar{x}(t) = \begin{cases} \hat{x}_{t_1}(t) & t \in [t_2, t_1 + \hat{\Gamma}(t_1)) \\ x_k(t - \hat{\Gamma}(t_1) + \Gamma(t_1)) & t \in [t_1 + \hat{\Gamma}(t_1), t_2 + \Gamma(t_2)] \end{cases}.$$

Clearly, $\bar{x}$ has lower cost than $x_k(t)$ $(t \in [t_2, t_2 + \Gamma(t_2)])$ (See Fig. 4.10). Thus, under mCLP, the follower would have taken $\bar{x}$ (or another trajectory with even lower cost) over $x_k(t)$ $(t \in [t_2, t_2 + \Gamma(t_2)])$. This contradicts the convergence to a limiting trajectory. The same argument can be applied at any other updating time, so that it can be concluded that $\hat{x}(t) = x_k(t)$ $(t \in [0, T_k])$.

Finally, recall that $x_k(t)$ is smooth for $t \in [t_1, t_1 + \Gamma(t_1)]$, because the locally optimal trajectories linking follower and leader are smooth by assumption. Similarly, $x_k(t)$ is smooth for $t \in [t_2, t_2 + \Gamma(t_2)]$ for any $t_1 < t_2 < t_1 + \Gamma(t_1)$. Therefore, $x_k(t)(t \in [t_1, t_2 + \Gamma(t_2)])$ is smooth. Repeated applications of this argument lead to the conclusion that the entire trajectory $x_k(t)$ $(t \in [0, T_k])$ is smooth. $\square$

The next theorem is an immediate consequence of above Lemmas:

Theorem 4.2 (Modified Continuous Local Pursuit): *Suppose that the group of Eq. (3.1) evolves under mCLP and that at all times t, the locally optimal trajectories from follower to leader are unique. Then, the limiting trajectory is unique and locally optimal. It is also smooth, if the locally optimal trajectories calculated at every* *updating time are smooth.*

Proof: From Lemma 4.8, the limiting trajectory is unique. It follows that $x_{k-1}(t - \Delta) = x_k(t)$ if $x_{k-1}(t) = x_\infty(t - t_{k-1})$. Choose δ_1, δ_2 such that $0 < \delta_1 < \delta_2 < \Gamma$ for all optimal final times Γ of the planned trajectories $\hat{x}_k$ generated during mCLP. The limiting trajectory x_∞ is piecewise smooth and locally optimal for $t \in [t_k + i\delta_1, t_k + i\delta_1 + \delta_2], i = 0, 1, 2 \dots$ because it coincides with the planned trajectories $\hat{x}_k(t)$.

From Lemma 4.7 – in this case S_Q is a single point – it can be concluded that $x_k(t)$ $(t \in [t_k, t_k + \delta_1 + \delta_2])$ is optimal because it is the composition of two overlapping locally optimal trajectories, $x_k(t)$ $(t \in [t_k, t_k + \delta_2])$ and $x_k(t)$ $(t \in [t_k + \delta_1, t_k + \delta_1 + \delta_2])$. From successive applications of this argument $(i = 2, 3, \dots)$, we conclude that $x_\infty(t)$ is locally optimal. Smoothness of x_∞ is proved via a similar "piece by piece" argument. $\square$

Having shown how to obtain the results for SLP and mCLP, similar results for CLP and mSLP could be derived easily by fixing the final time or final states in mCLP.

4.3 Summary

Until now, we have seen that each of the two pursuit algorithms we have investigated (SLP, mCLP) will generate an interesting "collective behavior" - namely the local optimum for the optimal control problem of interest. Although each agent only solves the optimal control problem within a small region (limited by Δ), the trajectories generated by them are gradually optimized. Each agent "learns" from its predecessor and the limiting trajectory exhibits the collective "knowledge" of the group. Therefore, a complicated task (optimizing over long distances) is separated into small tasks requiring lesser capabilities in terms of sensing, communicating and computing.

Our algorithms fall into the category of "learning by repetition". Newton's method and gradient methods are well-known examples in this category, and are usually applied to solve extremal problems in finite dimensional vector spaces [15]. Extensions of such methods in function spaces also enable the development of trajectory optimization algorithms through repetition. For example, the work of [40] utilized a developed gradient method to iteratively optimize the control for a specified dynamic system . The control $u(t)$ is derived by

$$\frac{du}{dt} = -\alpha \frac{\partial}{\partial u} \left[\frac{\partial W(x,t)}{\partial x} X(x,u) \right], \tag{4.24}$$

where $X(x,u) = \dot{x}(t)$ are the system dynamics and $W(x,t)$ is the minimal cost of reaching the final state x_f provided with the initial state is $x(t_0) = x$. Eq. (4.24) converges to the optimal control $u^*(t)$ and $x^*(t)$ if the optimal control is smooth.

However, existing algorithms usually require the cost function and the control to be partial differentiable. To proceed with the above algorithm, one also needs to store and describe the entire x_k, in order to obtain x_{k+1}. Moreover, to obtain a smooth curve, infinitely small time increments are required so that laborious calculations are introduced. All these factors hinder the application of these algorithms in decentralized systems whose members are working cooperatively.

In contrast, the proposed algorithms are suitable for a large class of optimization problems and do not suffer from the above drawbacks. For example, our algorithms could be applied in the situations where the control and trajectory are not smooth, such as bang-bang control. The computational requirements for each agent could be limited by defining an appropriate Δ. Furthermore, each agent only needs limited information from its predecessor so that multiple agents could work together to achieve the greatest effectiveness.

The idea of optimizing a trajectory by applying a repeatedly-updated sequence of controls is also present in model predictive control (MPC). However, in MPC one typically computes the optimal control from the current state to the terminal state x_f (see, for example, [19] and [47]), which in the present situation we would like to avoid. Instead, local pursuit substitutes a series of "shortened" versions of the original problem, involving only trajectories linking leader and follower.

4.4 Special Cases: Length and Time Minimization

For trajectory optimization problems that often involve length or time optimization, we have the following interesting results.

Theorem 4.3 : *If the time rate of the change of the cost along a trajectory is independent on $x_k(t)$ for all t, then the minimum cost from the follower to the leader with free final time is strictly decreasing under local pursuit, unless the leader moves along a locally optimal trajectory.*

Proof: Let $\rho(a, b) = J_F(x^*, \dot{x}^*, \tau)$ be the minimum cost to steer system from state a to another b. For the pursuit process shown in Fig. 4.11, We have that

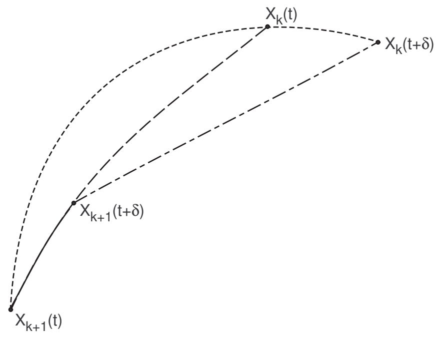

Figure 4.11: The minimum cost from $x_{k+1}(t))$ to $x_k(t)$ is decreasing if dC/dt is independent.

$$\begin{aligned}
\rho(x_{k+1}(t+\delta), x_k(t+\delta)) &\leq \rho(x_{k+1}(t+\delta), x_k(t)) + \rho(x_k(t), x_k(t+\delta)) \\
&\leq \rho(x_{k+1}(t+\delta), x_k(t)) + C(x_k(t), t, \delta) \\
&= \rho(x_{k+1}(t+\delta), x_k(t)) + C(x_{k+1}(t), t, \delta) \\
&= \rho(x_{k+1}(t), x_k(t)). \qquad (4.25)
\end{aligned}$$

If the equalities hold in Eq. (4.25) then $x_k(t)$ must be moving along an optimal trajectory. □

This result has a variety of applications, e.g., the minimum time control problem

$$J(x, \dot{x}, 0, T) = T, \qquad \|\ddot{x}\| \leq 1, \qquad (4.26)$$

whose solution could be obtained via the maximum principle; or the minimum path length problem with the condition that all agents are moving on unit speed

$$J(x, \dot{x}, 0, T) = T, \qquad \text{with } \|\dot{x}\| = 1, T \text{ is free.} \tag{4.27}$$

Chapter 5

Local Pursuit as a Computational Tool

This chapter discusses the use of local pursuit as a computational tool and how it can be used to complement existing numerical optimal control methods. As we shall see, local pursuit can not only extend the domain of applicability of existing numerical methods, but can also potentially reduce computational errors.

5.1 Motivation

As discussed in Chapter 2, local pursuit was originally conceived as a means of solving optimal control problems in settings where mapping, communication and sensing capabilities were severely limited. The algorithm manages to avoid the need for global information by breaking up the original problem into many pieces, each to be optimized by a leader-follower agent pair. Its key feature is a reduction in the range (measured by time, distance or other metric) over which computations must take place, by paying a price in terms of the number of agents that are necessary to carry out the algorithm. Up to now, we have focused on broadening the domain of applicability of local pursuit and on the limiting properties of the agents' trajectories. However, the algorithm's limited information requirements make it a potentially useful tool in numerical trajectory optimization, where there are often similar trade-offs to be made.

For many practical optimal control problems, exact (closed-form) solutions do not exist or are difficult to calculate. These include optimal control of nonlinear systems such as the well-known satellite transfer problem [49], minimum-time control for higher order systems [24], and initial value problems in ordinary differential equations [30], to name a few. In those settings, one must resort to numerical instead of analytical methods in order to compute optimal control policies. For continuous-time systems, the set of possible inputs is often infinite-dimensional. In order to pass to a finite-dimensional optimization problem, numerical optimal control methods typically generate a sequence of input samples that are to be optimized and applied at discrete times (meaning that the input signals are piecewise constant). We will provide a brief review of this process in the next section. Important considerations when applying numerical methods include the following:

1. Computing time. Numerical methods usually involve iterative computations, and may take numerous steps to converge to the desired result. Thus, a faster convergence rate can imply significant time savings.

2. Storage requirements. There usually exists an upper limit with respect to memory size for operating data – variables that are used to store the system's states or other data to be optimized. That limit is determined by various requirements for the solution, including smoothness, dynamics of system and desired accuracy.

3. Computational Accuracy. Iterative computations incur truncation errors at each step. These errors may accumulate and generate unacceptable results, as is the case, for example, when solving an ill-conditioned linear system of algebraic equations.

In light of the above considerations, there is a balance that must be struck between the number of trajectory samples and the size of the time intervals that separate them. On one hand, we would like to keep the number of segments small, so that the associated numerical optimization problem requires modest storage. However, if the number of trajectory segments is too small, then propagating the state vector from one sample point to the next may result in unacceptable approximation errors. At the same time, if the trajectory is sampled too densely, then there may not be adequate memory to run the numerical method of choice (e.g., nonlinear programming), which at the same time may be prone to a more significant accumulation of numerical error (we shall see an example of this in next chapter). In the following, we describe how local pursuit could be used in concert with existing numerical methods (specifically, multiple shooting), in order to relax this trade-off, and solve large-scale problems with limited storage, while maintaining small segment size.

5.2 Local Pursuit as a Computational Tool

The numerical optimization method known as multiple shooting (MS), together with its various improved forms, could be considered a workhorse of numerical optimization. However, the large storage requirements associated with MS limit the so-called "permitting size" of problems which can be solved on a digital computer [4]. In the following, we briefly review MS before showing how it can be improved when combined with SLP.

5.2.1 A Brief Review of Multiple Shooting

In principle, MS is a type of nonlinear programming (NLP) method. The development of NLP algorithms (and of MS) has followed the growth of the digital computer. The size of a typical application in the early 1960s was $n, m \approx 10$, while in the 1970s and early 1980s most application were of size as $n, m < 100$. With subsequent advances in linear algebra techniques, such as matrix sparsity, and ongoing progress in the semiconductor industry, the permitting size in late 1990s was $n, m \approx 10,000$ [4]. Suppose, for now, that we are interested in minimizing the scalar cost function $F(x) : \mathbb{R}^n \to \mathbb{R}$, subject to m ($m \leq n$) equality constraints $c(x) = 0$, where $c(x)$ is an $(m \times 1)$ vector. Using the notation in [4], we first introduce the Lagrangian:

$$L(x, \lambda) = F(x) + \lambda^T c(x). \tag{5.1}$$

The necessary conditions for the point (x^*, λ^*) to be an optimum are satisfied by the stationary points of the L which are the solutions of [1]:

$$\begin{aligned} \nabla_x L(x,\lambda) &= \nabla_x F(x) + \nabla_x c(x)^T \lambda = 0, && (5.2) \\ \nabla_\lambda L(x,\lambda) &= c(x) = 0. && (5.3) \end{aligned}$$

The equations (5.2)$\sim$(5.3) can be solved via Newton's method. Proceeding formally, we obtain the following Karush-Kuhn-Tucker system:

$$\begin{bmatrix} H_L & \nabla_x c(x)^T \\ \nabla_x c(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x F(x) - \nabla_x c(x)^T \lambda \\ -c(x) \end{bmatrix}, \tag{5.4}$$

where Δx is the "search-direction" and H_L is the Hessian of the Lagrangian

$$H_L = \nabla_x^2 F(x) + \sum_{i=1}^{m} \lambda_i \nabla_x^2 c_i. \tag{5.5}$$

For convenience, we define

$$H = \begin{bmatrix} H_L & \nabla_x c(x)^T \\ \nabla_x c(x) & 0 \end{bmatrix}. \tag{5.6}$$

Suppose now that we are interested in optimizing Eq. (3.6) subject to the dynamics

$$\dot{x} = f(x(t), u(t)). \tag{5.7}$$

[1]Here, we use the notation:

$$\nabla_x A(x) = \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_1}{\partial x_2} & \cdots & \frac{\partial a_1}{\partial x_n} \\ \frac{\partial a_2}{\partial x_1} & \frac{\partial a_2}{\partial x_2} & \cdots & \frac{\partial a_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial a_n}{\partial x_1} & \frac{\partial a_n}{\partial x_2} & \cdots & \frac{\partial a_n}{\partial x_n} \end{bmatrix}.$$

Doing so with MS (here we mainly refer to direct MS) involves breaking up the trajectory into "shorter" pieces [4] by partitioning the time domain $[t_0, t_f] = \cup_{i=1}^{N-1}[t_i, t_{i+1})$ $t_0 = t_1 < \cdots < t_N = t_f$. Each subinterval $[t_i, t_{i+1})$ is called a "segment".

Multiple shooting uses NLP to find the optimal trajectory (i.e., to minimize a scalar function along a trajectory), by defining the NLP variables $\nu = [\nu_1, \ldots, \nu_N]$ to be the arguments – usually they are concatenations of the states and the corresponding controls – at the sampling times $t_1, \ldots, t_N$ along a trajectory of the system. The stationary points $[\nu_1^*, \ldots, \nu_N^*]$, obtained via NLP, will approach points on the optimal trajectory. Without loss of generality, we may choose to sample the state and control vectors at the segment endpoints, and define the NLP variables

$$\nu = \{x_1, u_1, x_2, u_2, \ldots, x_N, u_N\}. \tag{5.8}$$

Note that the dimension of x_i and u_i, $i = 1, \ldots, N$ is M_x and M_u, respectively.

As the NLP variables are adjusted, one must ensure that they satisfy the system dynamics Eq. (5.7), and that sequential trajectory segments obey a matching condition at their boundaries. This means that the evolution of Eq. (5.7) from x_i at t_i with input u_i, should steer the system to x_{i+1} at t_{i+1}. This suggests that the following constraints are necessary:

$$c(x) = \begin{bmatrix} x_2 - \bar{x}_1 \\ x_3 - \bar{x}_2 \\ \vdots \\ x_N - \bar{x}_{N-1} \\ \phi_0(x_1, t_0) \\ \phi_T(x_N, t_f) \end{bmatrix} = 0, \tag{5.9}$$

where the functions $\phi_0(x_1, t_0)$ and $\phi_T(x_N, t_f)$ represent the initial and final condition constraints, respectively, and $\bar{x}_i$ are to be calculated by integrating the differential equation (5.7) from t_i to t_{i+1}.

In practice, one often computes $\bar{x}_i$ only approximately, instead of integrating the complete equations of motion. For example, Euler's method provides a first-order approximation to the integration of Eq. (5.7):

$$\bar{x}_{i+1} = x_i + hf(x_i, u_i), \tag{5.10}$$

where h is the time step of each segment. Other methods, e.g., Rung-Kutta methods, can provide higher order approcimation. The choice of approximation should be based on the following considerations: First, we have sampled the controls at the segment boundaries (the samples are to be optimized) instead of considering the continuous-time control on the intervals $[t_i, t_{i+1}]$; therefore, we cannot compute the precise state evolution by integration unless the controls are piece-wise constant upon the segments. Second, using linear approximation reduces the computational burden; on the other hand, higher-order approximations are likely to produce more precise results, but they will also give rise to more complicated equations in Eq. (5.2)$\sim$(5.9) and the resulting NLP problem will require more time to solve.

One shortcoming of linear approximation is that the time separation between neighboring points must be limited to a small time step h. If h is "too large", then $\bar{x}_{i+1}$ will not be a good estimate of the true state of Eq. (5.7) as it evolves from x_i to x_{i+1} under u_i, and NLP will produce erroneous results. The estimation error is of $o(h^n)$, where $n \in \mathbb{N}$ is the order of the approximation method [42]. It is clear that a smaller h results in better approximation. For that reason, keeping the segment size small is desirable and is a key feature of MS compared with other single-step shooting methods [4].

Let us assume that we can fix an acceptable h, i.e., one that is small enough to lead to convergence for the associated NLP. Then the permitting size of problems that can be solved by MS depends solely on the number of steps N required to cover the time span $[0, T]$, with $T = (N-1)h$. Thus, solving large-scale problems (with large time spans), requires a large number of segments $N = Th + 1$. For problems with time-varying dynamics and fixed time span, the requirement for large N could arise because nonlinearities in the dynamics make it necessary to use dense sampling to ensure that the estimates $\bar{x}_i$ are precise enough, and that the associated NLP will convergence. To summarize, for any given set of dynamics and choice of NLP method, the time step h is effectively upper bounded. The permitting size in MS is proportional to the number of segments N, if h is pre-determined.

Recall that the dimension of NLP variables for a multiple shooting method is $n_x = (M_x + M_u)N$, and the NLP has at least $M_x N$ constraints (the number of constraints could be larger if the final states are fixed). The Hessian in Eq. (5.5) is of dimension $n_x \times n_x = (M_x + M_u)^2 N^2$. To proceed with NLP, one must obtain the solution of Eq. (5.4), which requires finding the solution to a linear system of algebraic equations with dimension at least $(M_u + 2M_x)^2 N^2$. Here we have seen that the number of segments involved in the calculation affects the "degree of labor-consumption" with $O(N^2)$.

5.2.2 Numerical Optimization by Local Pursuit

As we have seen, the number of segments used in MS affects the dimension of the associated NLP variables as well as the computational complexity of the NLP problem to be solved. For that reason, it would be desirable to use fewer segments with the MS method. However, because the time step h is upper bounded for a fixed set of dynamics and NLP algorithm, decreasing the number of segments means that NLP process can only deal with trajectories spanning shorter time intervals. This limitation can be circumvented by combining local pursuit with direct MS, to obtain what we refer to as *pursuit-based multiple shooting* (PBMS). Specifically, we will introduce a sequence of simulated agents that pursue each other using the mSLP algorithm given in Chapter 3. Each agent will use MS to compute the optimal trajectory from its own state to that of its leader, giving rise to a series of smaller NLP problems, whose time span is limited by the pursuit interval Δ. Although there will be more of these NLP problems to solve, their lower dimension will make it possible to handle larger optimization problems overall.

Decreasing the Size of the NLP Problems During Computation

For simplicity, fix $h = T/(N-1)$ and select the pursuit interval $\Delta = (N_\Delta - 1)h,\ N_\Delta \in \mathbb{N}$, where N_Δ is the number of segments within Δ. Usually we will have $N_\Delta << N$. For convenience, we also choose the updating interval in the SLP algorithm to be an integer multiple of segment size, $\delta = N_\delta h,\quad N_\delta \in \mathbb{N}$, so that the agents are always updating their trajectories at the sampling times t_i which we chose for trajectory of Eq. (5.7).

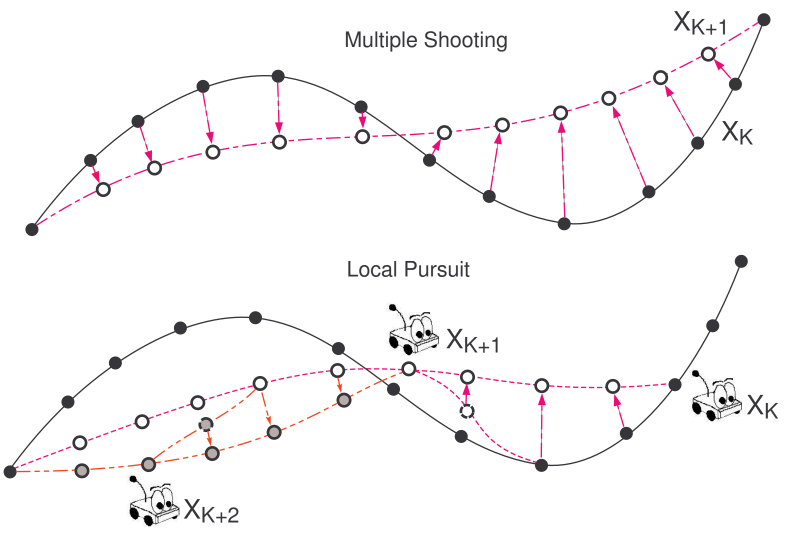

Figure 5.1: Local pursuit decreases the problem size at every updating step. In this figure, the number of variables involved in multiple shooting is $N = 13$, while the number of variables handled by each agent in local pursuit is $N_\Delta = 5$.

At each updating step $t = i\delta$, each agent is solving a NLP over N_Δ segments, with a time span of $(N_\Delta - 1)h$ instead of $(N - 1)h$, as illustrated in Fig. 5.1. Because the computational complexity of MS is related to the square of the number of segments, using $N_\Delta << N$ will significantly decrease the computational burden for each agent. Table 5.1 shows the dimensions of the NLP vector, ν, the constraints $c(x)$, and the matrix H in Eq. (5.6), respectively, for MS and PBMS. The dimension of the associated Karush-Kuhn-Tucker system is on the order of N^2 for MS, vs. N_Δ^2 for PBMS. Operating on large matrices consumes large amounts of memory, especially when using non-iterative methods, e.g., the memory of a typical desktop PC can be easily used up by a matrix with dimension of 5000×5000 in Matlab when using 64 bits of digital precision. However, under PBMS, the matrix size is scaled down by a factor of $(N_\Delta/N) \times (N_\Delta/N)$ and hardware requirements can be decreased significantly for any given problem.

Table 5.1: Comparing the dimensions of the NLP problem variables when using Multiple Shooting (MS) vs. Pursuit-based Multiple Shooting (PBMS). Typically, $N_\Delta << N$.

	MS	**PBMS**
$\dim(\nu)$	$(M_x + M_u)N \propto T$	$(M_x + M_u)N_\Delta \propto \Delta$
$\dim(c(x))$	$M_x N \propto T$	$M_x N_\Delta \propto \Delta$
$\dim(H)$	$((M_u + 2M_x)N)^2 \propto T^2$	$((M_u + 2M_x)N_\Delta)^2 \propto \Delta^2$

We note that under PBMS, each agent needs to solve $(N - N_\Delta)/N_\delta$ "smaller" MS problems in order to reach the target set S_Q from the initial state x_0, but the time needed to do so – denoted by T_a – will generally be less than the total iterative time in multiple shooting, which we will denote by T_{MS}. Of course, the total running time for PBMS – denoted by T_{LP} – may be greater than T_{MS} because local pursuit relies on multiple agents to converge to the optimum. Our experience with local pursuit and with various numerical experiments (including the example in next chapter) has been that, for well-conditioned problems, the convergence rate of PBMS is usually slower than that of MS. As expected, decreasing N_Δ leads to slower convergence for PBMS. However, the added running time comes with the benefit of lower memory requirements, allowing us to handle problems with larger state vectors and longer time horizons. At the same time, if N_Δ is decreased and the available storage is fixed, one can afford to also decrease the segment size h. Doing so has the effect of improving the state estimates $\bar{x}_i$ used to formulate the NLP, without making the problem ill-conditioned. This situation will be illustrated and discussed further in the next chapter.

Reducing Numerical Error When H is Ill-Conditioned

Besides maximum problem size, another important consideration in numerical optimal control, is the computational error introduced by the finite precision of digital computers and by algorithmic accuracy. It is possible that the error is too large to obtain useful results, e.g., solving a linear system with large condition number can result in unacceptable errors and non-convergence. In such settings, correction algorithms, such as Tikhonov regularization, can be applied [42, 51]; they are, however, time and storage consuming, and do not always succeed.

Consider, for example, using Gaussian elimination to solve Eq. (5.4) with limited digital precision. Every step of the elimination algorithm introduces some truncation error, so that the total accumulated error when solving a large linear system will generally be much larger than that associated with solving one of lower dimension, because the number of steps required by the algorithm is proportional to the system size. Furthermore, if H in Eq. (5.6) is ill-conditioned, then the numerical solution of Eq. (5.4) introduces significant errors which accumulate towards the final segment N. If the problem's time horizon is long, the accumulated error may lead to erroneous results, or prevent MS from converging.

PBMS can help reduce these numerical errors because the algorithm's simulated agents solve MS problems with a shorter time horizon, compared to that of the original problem. This implies that the dimension of Eq. (5.4) is reduced for each agent and there will be cases in which PBMS will succeed where MS failed to converge. Furthermore, if every locally optimal trajectory satisfies the convergence criteria of the numerical method used to solve the "short-range" MS problems between leader-follower pairs, then the convergence of the agents' trajectory sequence is guaranteed by the local pursuit algorithm itself.

Remarks

To summarize, the combination of MS with local pursuit can increase the permitting size of problems that can be handled with fixed storage, because at every updating step, each agent deals with a problem with "reduced size". However, when using a fixed time partition, PBMS involves solving problems of size $N_\Delta << N$ instead of N. Therefore, we can address much larger problems under the limits imposed by the hardware. Although it requires more running time, PBMS does provide a feasible solution when the traditional formulation exceeds those limits, making it impossible to proceed.

In cases where MS fails to converge because the errors introduced by the approximation to Eq. (5.7) are too great, PBMS may succeed by reducing the segment size h, thereby reducing the accumulated error over the trajectory of a single agent. In next section we will present an example of MS stagnancy caused by an ill-conditioned matrix H and how PBMS can avoid the problem.

Chapter 6

Simulations and Experiments

In this chapter we illustrate the application of local pursuit in a series of simulations and laboratory experiments. We do this through a progression of optimal control examples that make use of the four algorithms (SLP, CLP, mSLP and mCLP) which were discussed in Chapter 2.

6.1 Sampled Local Pursuit (Large δ)

In the first example, we illustrate the use of sampled local pursuit through an intuitive and straightforward example. We seek to minimize the path length

$$\int_0^1 \|x\| dt$$

on $\mathbb{R}^2$ subject to $\dot{x} = u$ and boundary conditions $x(0) = (0,0), x(1) = (1,1)$. Of course, the optimal trajectory is a straight line. We set $\delta = 0.25, \Delta = 0.5,$ and $T = 1$, and let the first agent move along a half circle to the final state. Then each subsequent agent departed from $(0,0)$ 0.5 seconds after its predecessor. At every updating time $t_k + i\delta, i = 0, 1, 2, \ldots$ each follower moved on a straight line towards its leader and changed its speed to Δ/d, where d is the length of the line segment connecting the leader and follower. Fig. 6.1 shows the trajectories of the first five agents. The fifth trajectory was close to a straight line.

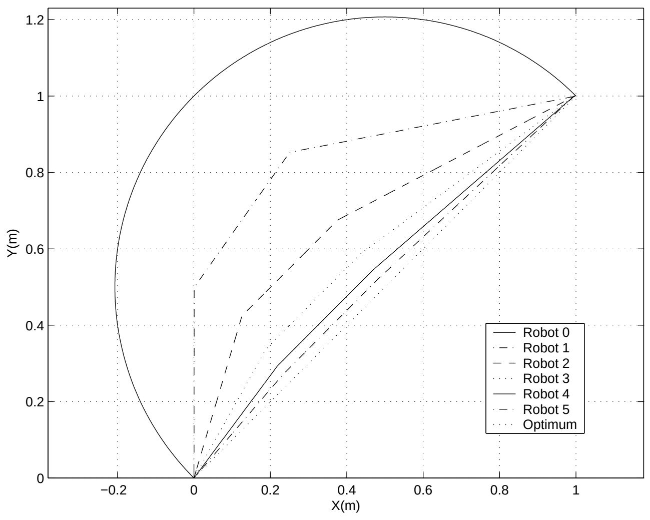

Figure 6.1: Iterated trajectories for the minimum length problem, using SLP on $\mathbb{R}^2$

6.2 An LQR Example

Suppose that we are to minimize the quantity

$$\int_0^1 \dot{x}_1(t)^2 + u(t)^2 dt, \tag{6.1}$$

subject to

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \tag{6.2}$$

and boundary conditions $x(0) = (0,0), x(1) = (1,0)$. We used CLP to solve this problem. We simulated a collection of agents whose dynamics were given by

Eq. (6.2). The pursuit interval Δ was set to 0.5. The first was steered to the final state along a sinusoidal trajectory. Each subsequent agent left $x(0)$ 0.5 second after its leader, and updated its trajectory continuously in an attempt to reach its leader's state with minimum cost, as defined by Eq. (6.1). In each case, the locally optimal trajectories linking a follower to its leader were obtained by solving the corresponding Euler-Lagrange equation:

$$\frac{d^4x_1}{dt^4} + \frac{d^2x_1}{dt^2} + x_1 \quad = \quad 0.$$

The trajectory sequence generated by the agents converged to the optimum, as Fig. 6.2 illustrates.

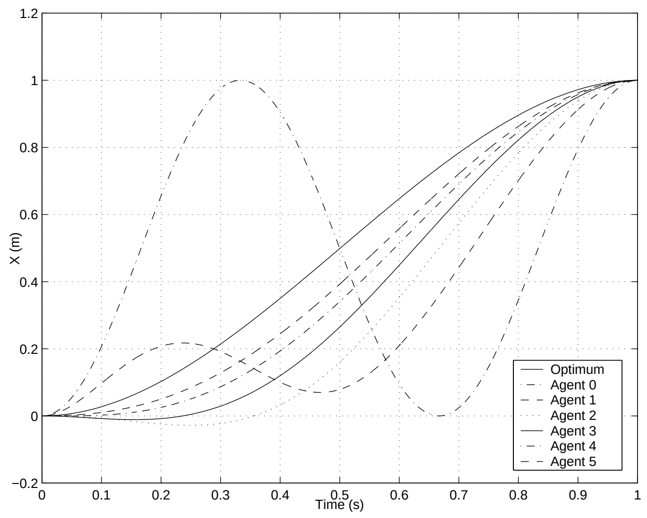

Figure 6.2: Iterated trajectories for the Lagrangian problem through SLP with $\Delta = 0.5, \delta \to 0$

6.3 Minimum Time Control

We want to minimize the final time

$$T = \int_0^T 1dt$$

subject to the the following system

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

with the boundary conditions of $x(0) = (\pi, 0), x(T) = (0, 0)$ and constraint $\|u\| \leq 1$. Because the final time was free, we applied mCLP with the following interval set to $\Delta = 0.3\pi$. The input for the first agent was arbitrarily selected (here the initial control input was chosen to a sine wave). Each agent continuously updated its trajectory after departure from $x(0)$, trying to reach the state of its leader within the minimum time T_L. From the maximum principle it was well-known that the optimal input for the locally optimal trajectory is the "bang-bang" control law:

$$u^*(t) = \begin{cases} -1 & \text{if } t \in [0, T_L/2) \\ 1 & \text{if } t \in [T_L/2, T_L] \end{cases} . \tag{6.3}$$

As Fig. 6.3 illustrates, the sixth agent evolved under the essentially optimal control, and the convergence was rapid. It is interesting to see that we reach the optimum with finite numbers of agents.

6.4 Finding Geodesics on Uneven Terrain

We now present a geodesic discovery example that involves complex calculations when solved in its "global" version, but which becomes straightforward when solved

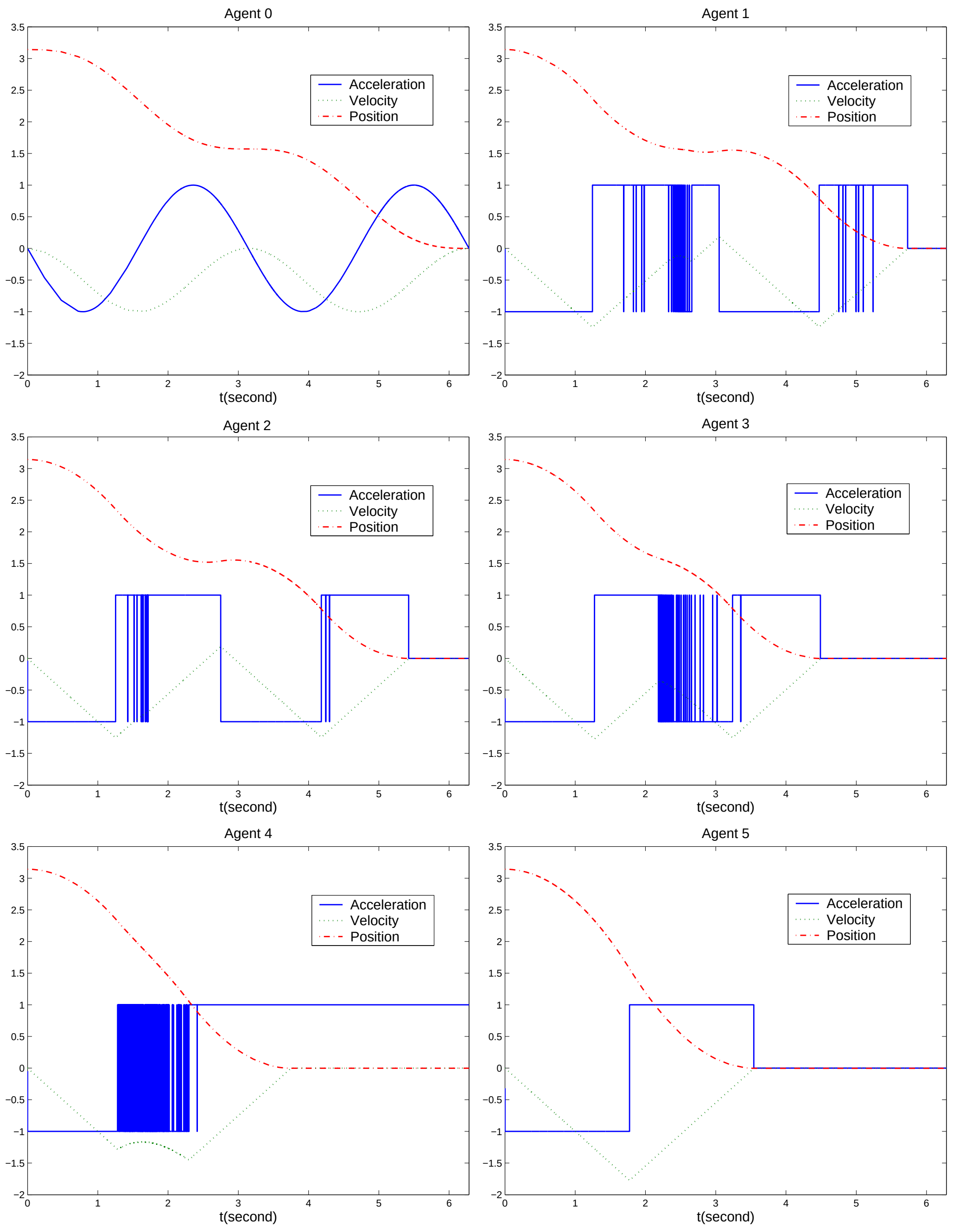

Figure 6.3: Iterated trajectories for the minimum time control problem via CLP with $\Delta = 0.3\pi$

over local patches. Consider the problem of finding shortest paths in an environment consisting of a plane with two right cones; a partial view is shown in Fig. 6.4. The radii of the cones were 800 and 1000 units of length, respectively. The agents were

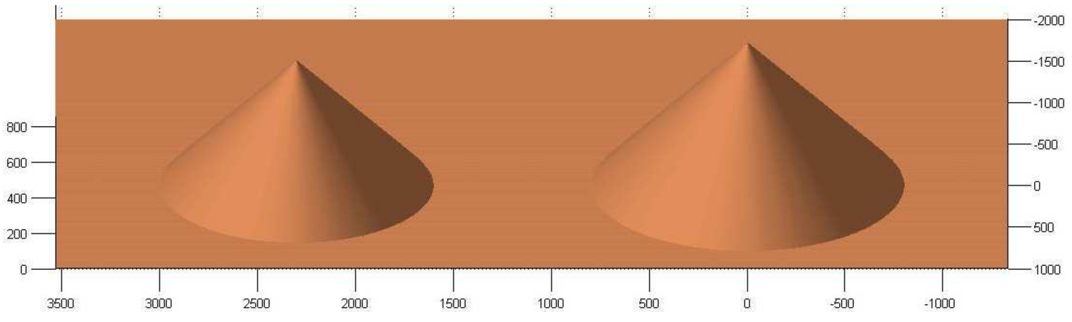

Figure 6.4: Uneven terrain consisting of two cones and a plane.

governed by $\dot{x}_k = u_k$, $\|u_k\| = 1$ and were required to travel from $x_0 = (3500, 0, 0)$ to $(-1300, 0, 0)$.

The first agent moved on a trajectory that followed along the border of the cones. In this environment, long geodesics were difficult to compute because doing so demands knowledge of the entire map and because one has to take into account four coordinate switches that occur along the optimal path. However, if we set $\Delta = 0.2T$ in CLP, the follower was required to compute optimal trajectories that crossed over at most one coordinate patch. When the leader and follower were both on the plane or on the same cone, the computation of optimal trajectories was straightforward. In other cases, agents had to optimize trajectories that crossed at most two coordinate patches (plane-to-cone or cone-to-plane), and did so by selecting from a one-parameter family of curves joining leader and follower. On the other hand, computing the globally optimal trajectory at once would have required searching over a four-parameter family of curves (there are a total of four "crossings" between coordinate patches). The iterated trajectories converged to the optimum, as Fig. 6.5 illustrates.

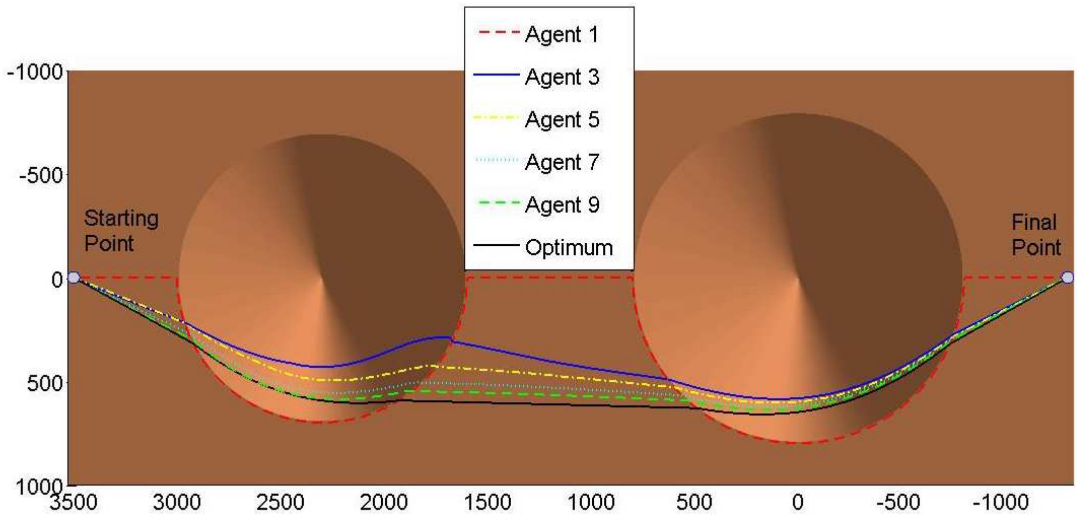

Figure 6.5: Iterated trajectories for the geodesic discovery problem using mCLP with $\Delta = 0.2T$.

6.5 A Trail Optimization Problem with Free Final State

Consider the problem of finding shortest paths in the same environment described in the last section, where the agents were required to travel from $x_0 = (3500, 0, 0)$ to the second cone with constant speed. Suppose that now we would like to minimize the path length (proportional to the final time T) necessary to reach the second cone. Because this is a problem with free final time and partially-constrained final state, mCLP was applied.

Fig. 6.6 shows the iterated trajectories generated when the agents implemented the mCLP policy with $T_0 = 3499$, $\Delta = 0.2T_0$. For the computation of the

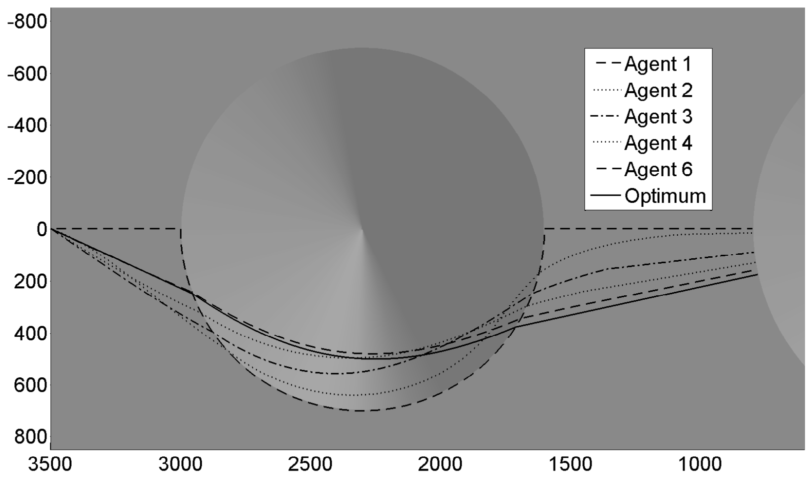

Figure 6.6: Continuous local pursuit in a complex environment. The initial trajectory (along the borders of the cones) was easy to describe (for example, “move along the cone boundaries”) but far from optimal. The locally optimal trajectories were easier to compute than the global optimum because of the limited pursuit distance ($\Delta = 0.2T_0$). The iterated trajectories converged to the optimum.

optimal trajectory, each agent had to solve its own version of the optimal control problem which was simpler than the “global” problem, for the same reasons that were outlined in the previous example. The only difference between this and the previous example was that when a follower detected that its leader had already reached the cone boundary, it ignored the leader and optimized its trajectory to the cone boundary, as described by mCLP.

6.6 Minimum-time Control with Speed and Acceleration Constraints

Next, consider the second-order system

$$\dot{x} \triangleq \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \tag{6.4}$$

where we seek to minimize

$$J(x, \dot{x}, 0) = \int_0^T 1dt = T$$

subject to $|u| \leq 30$, $|x_2| \leq 8$ and with boundary conditions $\dot{x}_1(0) = \dot{x}_1(T) = x_1(0) = 0$, and $x_1(T)$ fixed. Here, the constraint set S_Q was a single point in the state space. The optimal control policy for this problem is the well-known "bang-off-bang" control: the control u switched at most once between 30 and -30, with $u = 0$ when the maximum or minimum speed x_2 had been reached.

To apply CLP, we simulated a sequence of agents-copies of Eq. (6.4). The initial agent was driven by a suboptimal input (Agent 1 in Fig. 6.7) which alternated between the maximum and minimum available acceleration. Subsequent agents departed from the initial state with $\Delta = 1.3$ sec. Each agent chose the (bang-off-bang) control input that would allow it to intercept its leader's state in minimum

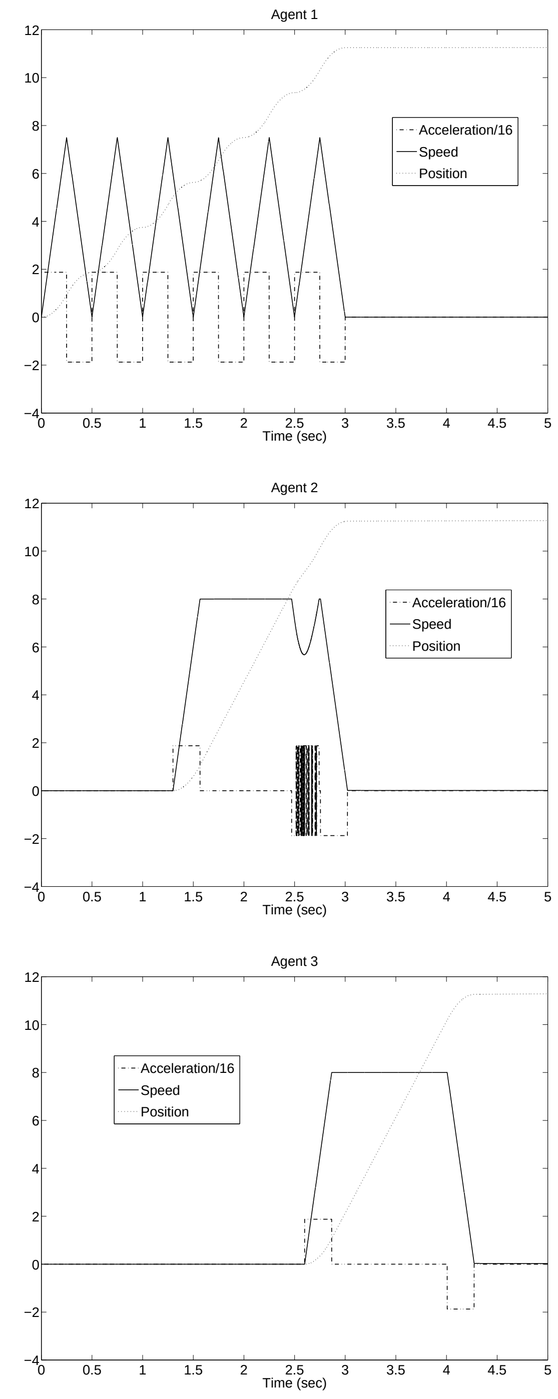

Figure 6.7: Iterative trajectories for minimum control with limited acceleration and speed. The simulated control loop ran at a frequency of $2000Hz$ so that the control policy could be regarded as approximately CLP. The pursuit interval was $\Delta = 1.3$.

time, and continuously updated that choice until it reached the final state. The third agent's trajectory was optimal (see Fig. 6.7). Notice that after $t > 2.7$ sec the second agent intercepted the first and subsequently moved along the same trajectory. It is also interesting to note that in this case, optimality was achieved after a finite number of iterations.

6.6.1 An Experiment in Minimum-time Control

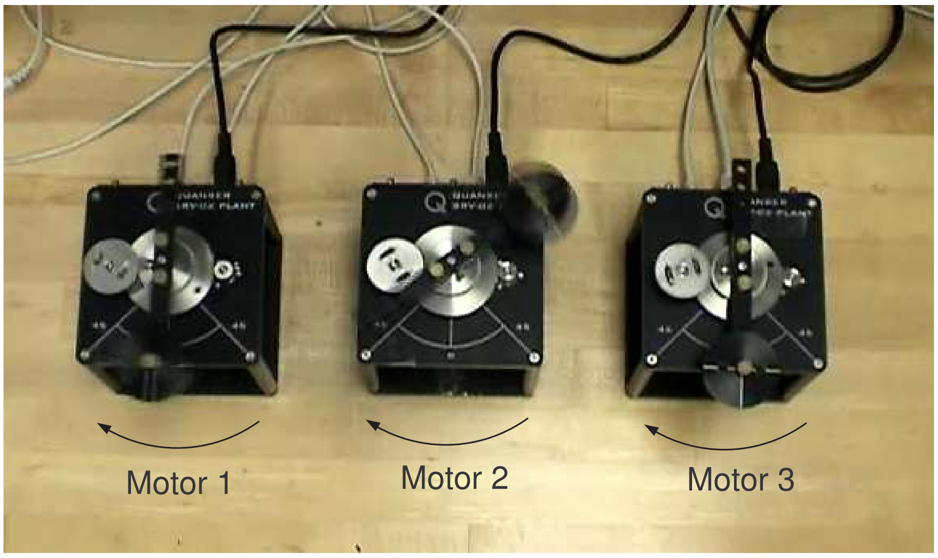

Figure 6.8: Applying local pursuit with a trio of motors to obtain minimum-time control with limited acceleration and speed.

We implemented the example of last section using a collection of three motors, pictured in Fig. 6.8. Each motor was equipped with position and speed sensors, which were sampled by a PC-based controller at a rate of $2000Hz$. The goal was to rotate the motors to a fixed final position in minimum time. Motor acceleration and speed were limited to 30 rad/sec^2 and 8 rad/sec, respectively.

The input to the first motor was a rectangular pulse with amplitude equal to the maximum acceleration (same as in the simulation of Sec. 2.4). Each of the remaining two motors tried to "catch up" with its predecessor by measuring the predecessor's state and applying a control to reach that state in minimum time.

The trajectories of all three motors with $\Delta = 1.3$ sec are shown in Fig. 6.9. We saw that the third motor evolved under essentially optimal control, and the second motor "intercepted" the first after $t \approx 2.3$ sec. We noted that because of unmodeled friction, the final position $\theta(T)$ was less than the nominal value (see $x(T)$ in the last simulation). The presence of friction also caused the motors to decelerate when a zero input was applied (once the motors had reached maximum speed). That deceleration in turn caused the CLP policy to try and "catch up" by introducing a positive control input, resulting in chatter observed in the velocity and acceleration curves of motors 2 and 3 in Fig. 6.9.

6.7 Numerical Optimal Control via PBMS: An Orbit Transfer Problem

In this section we illustrate the performance of PBMS via an optimal control example that requires the use of numerical methods. Consider an idealized spacecraft which must be transfered from one stable orbit[1] to another, within some fixed time T. For

[1]The term "stable orbit" means that the spacecraft will remain on this orbit if no external force (other than gravity) is applied, i.e., $\dot{\theta} = \sqrt{u_E/r^3}$.

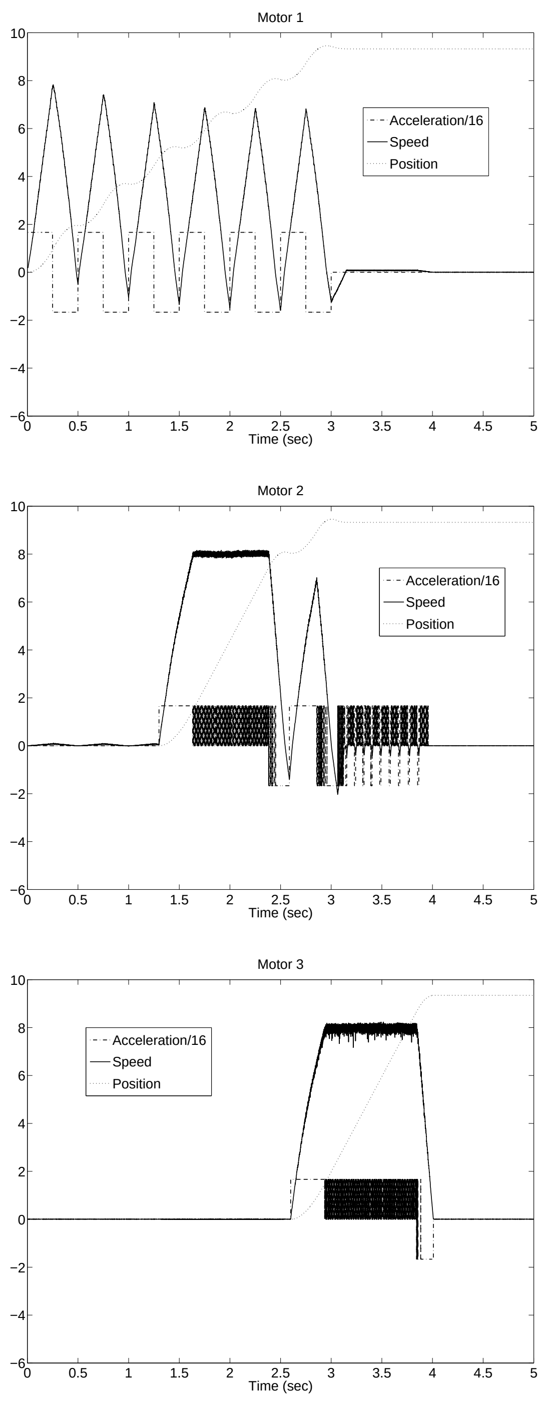

Figure 6.9: Iterative trajectories of motors when applying local pursuit to attain minimum-time control with limited acceleration and speed. The pursuit interval was $\Delta = 1.3$. The third motor evolved under essentially optimal control.

simplicity, we only considered the effect of the Earth and restricted the problem to a plane, as illustrated in in Fig. 6.10.

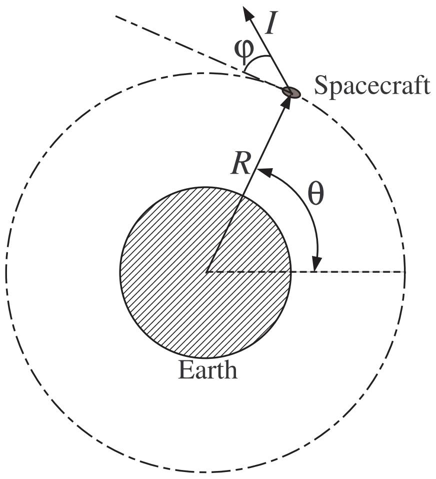

Figure 6.10: A planar spacecraft in orbit around the Earth.

The dynamics of this system were

$$
\begin{aligned}
\ddot{r} &= \frac{\dot{\theta}^2}{r} - \frac{u_E}{r^2} + \frac{P u_1}{mg}, \\
\ddot{\theta} &= -\frac{2\dot{\theta}\dot{r}}{r} + \frac{P u_2}{mgr}, \\
\dot{m} &= -\frac{P(u_1^2 + u_2^2)}{g^2}, \qquad (6.5) \\
u_1 &= I\sin(\varphi), \\
u_2 &= I\cos(\varphi),
\end{aligned}
$$

where r was the distance between the spacecraft and the center of Earth, θ was its longitude with respect to the horizontal line, m was the mass of the spacecraft, u_E was the gravitational parameter of Earth, P was a constant concerning engine power, and g was the acceleration of gravity at sea level [63]. The control inputs, u_1

and u_2, were functions of I and φ, the thrust force and thrust steering angle with respect to the tangent of the local orbit.

We would like to minimize the fuel consumed (equivalently, to maximize the final mass $m(T)$) while steering the system (6.5) from the initial condition $r(0) = R_0, \dot{r}(0) = 0, \dot{\theta}(0) = \sqrt{u_E/R_0^3}, m(0) = M_0$ to the final condition $r(T) = R_T, \dot{r}(T) = 0, \dot{\theta}(T) = \sqrt{u_E/R_T^3}$, where T was fixed. For that reason, we chose

$$J = \int_0^T u_1(t)^2 + u_2(t)^2 dt.$$

For simplicity, we assumed that there was no upper bound of the thrust force, thus there were no restrictions on the controls u_1 and u_2.

6.7.1 A Comparison of MS vs. PBMS

To solve the problem by MS, we first wrote the cost to be optimized as a function of the NLP variable:

$$J = \sum_0^N (u_1(i)^2 + u_2(i)^2),$$

where $u_1(i), u_2(i)$ are the controls at the i^{th} segment grid. We also used Euler's method to (approximately) integrate the dynamics of Eq. (6.5) and imposed the following continuity constraints:

$$\begin{aligned}
0 &= r_{j+1} - \bar{r}_j \\
&= r_{j+1} - (r_j + h_j \dot{r}_j), \\
0 &= \dot{r}_{j+1} - \bar{\dot{r}}_j \\
&= \dot{r}_{j+1} - (\dot{r}_j + h_j(\frac{\dot{\theta}_j^2}{r_j} - \frac{u_E}{r_j^2} + \frac{Pu_{1j}}{m_j g})), \\
0 &= \dot{\theta}_{j+1} - \bar{\dot{\theta}}_j \\
&= \dot{\theta}_{j+1} - (\dot{\theta}_j + h_j(-\frac{2\dot{\theta}_j \dot{r}_j}{r_j} + \frac{Pu_{2j}}{m_j g r_j})), \\
0 &= m_{j+1} - \bar{m}_j \\
&= m_{j+1} - (m_j - h_j \frac{P(u_1^2 + u_2^2)}{g^2}), \qquad (6.6)
\end{aligned}$$

for $j = 1, 2, 3 \ldots, N-1$, and

$$\begin{aligned}
0 &= r_1 - R_0, \\
0 &= \dot{r}_1 - \dot{R}_0, \\
0 &= \dot{\theta}_1 - \dot{\theta}_0, \\
0 &= m_1 - M_0, \\
0 &= r_T - R_T, \\
0 &= \dot{r}_T - \dot{R}_T, \\
0 &= \dot{\theta}_T - \dot{\theta}_T, \qquad (6.7)
\end{aligned}$$

where $r_i = r((i-1)h), \dot{r}_i = \dot{r}((i-1)h), \ldots$ and $h = T/(N-1)$ was the time step of the integration. The dimension of constraints $c(x)$ was $(4 \times (N-1) + 7)$. The NLP variable

$$\begin{aligned}\nu &= \{r_1, \ldots, r_N, \dot{r}_1, \ldots, \dot{r}_N, \dot{\theta}_1, \ldots, \dot{\theta}_N, m_1, \ldots, m_N \\ &\quad , u_{11}, \ldots, u_{1N-1}, u_{21}, \ldots, u_{2N-1}\} \qquad (6.8)\end{aligned}$$

was of dimension $(6 \times N - 2)$.

When applying local pursuit in PBMS, we simulated a sequence of control systems-copies of Eq. (6.5), that pursued one another in pairs, using mSLP. The first of those systems reached the target orbit along a feasible but suboptimal trajectory which was produced by an input similar to a bang-bang control law. During pursuit, each agent used multiple shooting to find the optimal (minimum fuel) control that steered it from its current position to that of its leader. The number of segments N (under MS) was now replaced by N_Δ in Eq. (6.8), and the boundary conditions were adjusted to correspond to the states of each leader-follower pair.

6.7.2 Results

We solved the orbit transfer problem both by PBMS and by standard MS. The criteria for convergence were $\|m(T)_{i+1} - m(T)_i\| \leq 10^{-8}$ (to guarantee little improvement with future iterations) and $\|c(x)\| \leq (10^{-15}) \times m$, where m was the dimension of constraints (to guarantee that the trajectory satisfied the system dynamics). For the convenience of verifying our experimental results, the total time T was fixed to the same amount (300 minutes) in all cases so that we could compare results using different number of segments. The numerical "behavior" of the problem – and the performance of the two methods – depended on the selection of the total number of segments N.

6.7.3 Well-conditioned Case

With $N = 101$, the matrix H in Eq. (5.6) was well conditioned (this fact could be verified by checking its condition number and was suggested by the fast convergence rate for MS, as shown below). The performance of MS and PBMS are summarized in Table 6.1, where c(H) denotes the condition number of matrix H. The "iterations"

Table 6.1: Comparison between Multiple Shooting and Local Pursuit in well-conditioned case

N=101	**Multiple Shooting**	**SLP** $(N_\Delta = 30, N_\delta = 16)$	**SLP** $(N_\Delta = 60, N_\delta = 32)$
Computing Time	66.8594	793.8594	430.2344
Iterations	14	277	41
$m(T)$	0.524246124	0.524245714	0.524246100
$\Vert c(x)\Vert$	3.7144E-14	9.2499E-15	9.9170E-15
Ave(c(H))	1.5931E+6	4.8919E+5	5.3612E+5

row lists the number of iterations required for convergence in multiple shooting, and number of agents needed for local pursuit to converge, respectively.

We see that both methods were successful and that the convergence rate of PBMS was slower than that of MS. Increasing N_Δ resulted in increased storage requirements and decreased running time. When $N_\Delta = N$, PBMS reduced to MS.

The final trajectories obtained by both methods are shown in Fig. 6.11.

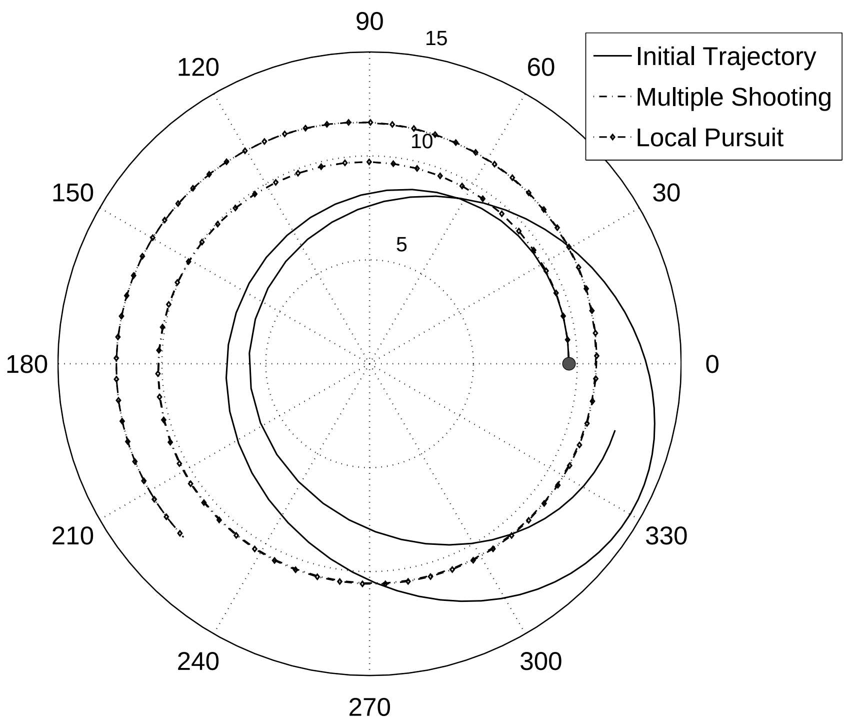

Figure 6.11: Trajectories generated by MS and PBMS for a well-conditioned case with $N = 101, N_{\Delta} = 30, N_{\delta} = 16$. The trajectories obtained from both methods were virtually identical. The symbol "•" indicated the starting point of the trajectories.

6.7.4 Ill-conditioned Case

When the segment size was halved, i.e., $N \geq 201$, MS was stagnant because the matrix H in Eq. (5.6) became ill-conditioned. The error generated by solving Karush-

Kuhn-Tucker system became so large that the NLP algorithm was not able to converge when using Newton's method. The large condition number of H (our criterion of ill-condition) was due to the large number of segments into which the trajectory was broken down. In fact H's condition number increased with the size of segments (see Table 6.2). For PBMS, the reduction in the number of segments for the sub-problems solved by pursuing agents meant a corresponding reduction in the condition number of H when using small N_Δ.

The results from both methods are summarized in Table 6.2. In this case, MS could not converge (the values of the constraint residues $c(x)$ did not become sufficiently small). On the other hand, PBMS was effective in producing the optimal trajectory. The final trajectories of both methods are shown in Fig. 6.12. By comparing to the trajectories generated in the well-conditioned case, it is clear that the trajectory obtained from multiple shooting was sub-optimal.

6.7.5 Large Number of Segments

When $N \geq 600$, the orbit transfer problem could not be solved a PC with 1Gb of RAM using MS[2]. On the other hand, PBMS's lower memory requirements meant that the algorithm was able to operate and converge to the optimum. Here we used $N_\Delta = 60, N_\delta = 32$; the final trajectory is shown in Fig. 6.13.

[2]This included the memory requirements of the function used to compute condition numbers. If we did not want to record condition numbers and only used Gaussian elimination method to solve the linear system, then N could be increased a bit further.

Table 6.2: Comparison between Multiple Shooting and Local Pursuit in ill-conditioned case

N=201	**Multiple Shooting**	**SLP** $(N_\Delta = 30, N_\delta = 16)$	**SLP** $(N_\Delta = 60, N_\delta = 32)$
Time	$\geq$100000	32911.8750	10676.9844
Iteration	$\geq$3000	7223	603
$m(T)$	0.523948746	0.524560874	0.524571236
$\lVert c(x) \rVert$	0.01709873	6.0790E-14	1.4019E-14
Ave(c(H))	1.8263E+10	5.0576E+5	4.7881E+5
Max(c(H))	4.3377E+12	6.1142E+5	5.6897E+5
N=301	**Multiple Shooting**	**SLP** $(N_\Delta = 30, N_\delta = 16)$	**SLP** $(N_\Delta = 60, N_\delta = 32)$
Time	$\geq$360000	239601.4219	81264.3906
Iteration	$\geq$3200	30722	2845
$m(T)$	0.527347984	0.524643889	0.524700485
$\lVert c(x) \rVert$	0.04135162	2.6051E-13	1.2922E-13
Ave(c(H))	4.5106E+10	5.3032E+5	4.8184E+5
Max(c(H))	7.3201E+10	6.5692E+5	6.5833E+5

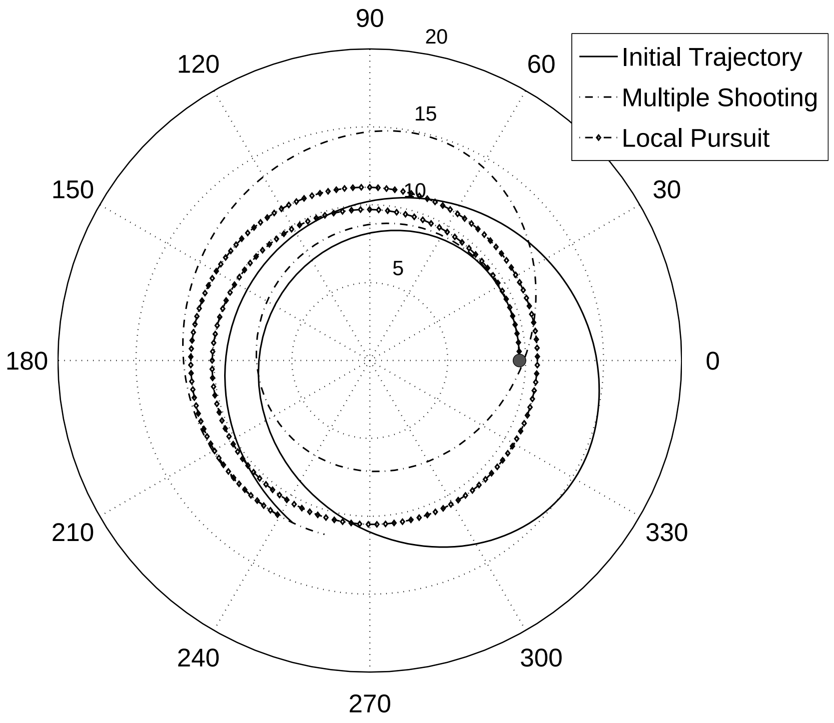

Figure 6.12: Satellite trajectories for an ill-conditioned case with $N = 201, N_\Delta = 30, N_\delta = 16$. The trajectory obtained via local pursuit was essentially optimal; multiple shooting failed to converge. The symbol "●" indicated the starting point of the trajectories.

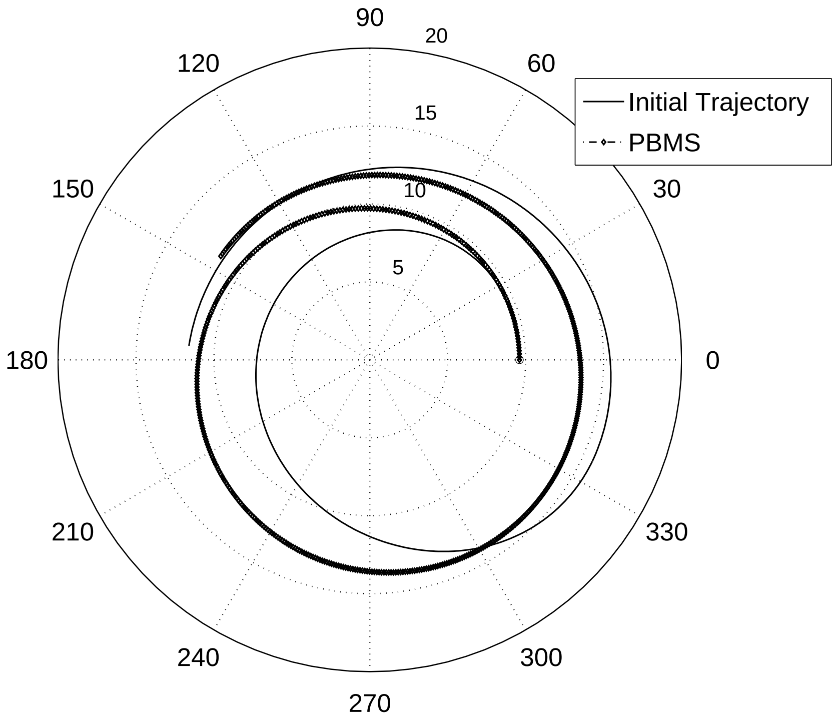

Figure 6.13: Spacecraft trajectory in the case of large number of segments, $N = 601, N_\Delta = 60, N_\delta = 32$. The symbol "•" indicates the starting point of the trajectories.

6.8 Quadratic Optimal Control on a Sphere

We now consider a quadratic optimal control problem involving a massive particle that moves on S^2 under the influence of external forces. We begin by deriving the dynamics of the particle in spherical coordinates; we will use the notion of parallel transport to maintain the correct direction on tangent space along which control forces are applied. Finally, we show how optimal controls can be computed via SLP.

6.8.1 System Dynamics on a Sphere

We described the particle's position (we do not consider any rotations of the particle in this example) on the unit sphere via the usual parametrization:

$$\mathbf{x}(\theta, \phi) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \tag{6.9}$$

We assumed that there were two thrusters fixed on the particle, which produce the forces F_{R1}, F_{R2}. Those forces act along a set of orthogonal directions. In a *body-fixed* coordinate frame, we have $F_{R1} = [u_1, 0]^T$and $F_{R2} = [0, u_2]^T$, where u_1 and u_2 were the magnitudes of the two forces.

Suppose that we want to minimize

$$J = \int_0^T F_{R1}^2(t) + F_{R2}^2(t)dt \tag{6.10}$$

with fixed T and with $\theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta}_0, \phi(0) = \phi_0, \dot{\phi}(0) = \dot{\phi}_0$, $\theta(T) = \theta_T, \dot{\theta}(T) = \dot{\theta}_T, \phi(T) = \phi_T, \dot{\phi}(T) = \dot{\phi}_T$. F_{R1} and F_{R2} are the forces applied on the massive particle along the direction body-fixed coordinates.

The particle's acceleration can be shown to be given by (see Appendix A for derivation):

$$\begin{aligned}\ddot{\mathbf{x}}(\theta,\phi) &= \mathbf{x}_{\theta\theta}\dot{\theta}^2+\mathbf{x}_{\theta}\ddot{\theta}+2\mathbf{x}_{\theta\phi}\dot{\theta}\dot{\phi}+\mathbf{x}_{\phi}\ddot{\phi}+\mathbf{x}_{\phi\phi}\dot{\phi}^2\\ &= (-\sin\theta\cos\phi,-\sin\theta\sin\phi,-\cos\theta)\dot{\theta}^2+(\cos\theta\cos\phi,\cos\theta\sin\phi,-\sin\theta)\ddot{\theta}\\ &\quad +2(-\cos\theta\sin\phi,\cos\theta\cos\phi,0)\dot{\theta}\dot{\phi}\\ &\quad +(-\sin\theta\sin\phi,\sin\theta\cos\phi,0)\ddot{\phi}+(-\sin\theta\cos\phi,-\sin\theta\sin\phi,0)\ddot{\phi}. \qquad (6.11)\end{aligned}$$

We defined a pair of orthonormal vectors $\bar{\mathbf{x}}_\theta, \bar{\mathbf{x}}_\phi$ on the tangent space TS^2 such that they were parallel to the tangent vectors x_θ and x_ϕ. These vectors formed an orthonormal basis for TS^2; we referred to the associated coordinate system as *sphere coordinates.* In particular

$$\bar{\mathbf{x}}_\theta = \mathbf{x}_\theta, \ \ \bar{\mathbf{x}}_\phi = \mathbf{x}_\phi/\sin\theta,$$

By expressing the particle's acceleration in the frame $(\bar{\mathbf{x}}_\theta, \bar{\mathbf{x}}_\phi)$ we obtained (see Appendix A) :

$$\begin{aligned}\ddot{\theta} &= \dot{\phi}^2\sin\theta\cos\theta+F_{S1},\\ \ddot{\phi} &= 1/\sin\theta(-2\dot{\theta}\dot{\phi}\cos\theta+F_{S2}), \qquad (6.12)\end{aligned}$$

where the F_{S1} and F_{S2} were forces applied to the particle along the directions $\bar{x}_\theta$ and $\bar{x}_\phi$, respectively.

Eq. (6.12) represents the dynamics of the particle on the sphere. Of course, the directions on TS^2 along which u_1 and u_2 were applied will generally differ from $\bar{\mathbf{x}}_\theta$ and $\bar{\mathbf{x}}_\phi$, as illustrated in Fig. 6.14. We must therefore calculate the relationship between the body-fixed and sphere coordinate systems (equivalently, we must

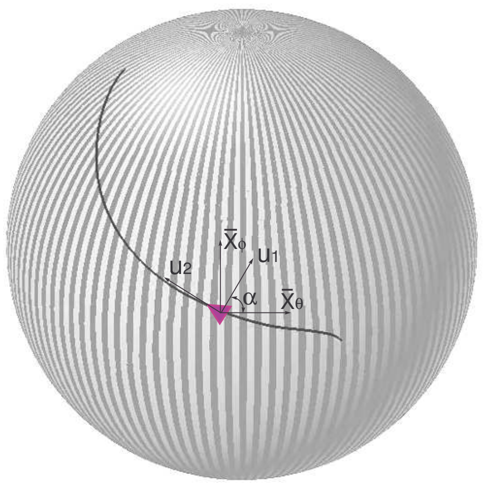

Figure 6.14: The angle between the body-fixed coordinate and sphere coordinate frames varies differ along the particle's trajectory. The relationship between them the two frames can be obtained via parallel transport.

describe F_{Si} in body-fixed coordinates) along the particle's trajectory, so that we could know how the control inputs u_1 and u_2 affected the particle's trajectory in sphere coordinates. This was accomplished by performing parallel transport of the body-fixed coordinate frame along the particle's trajectory.

Because F_{R1}, F_{R2} are the thrust forces expressed in bodyfixed coordinates, we obtained:

$$\begin{aligned} F_{R1}(t) &= F_{S1}(t)\cos\alpha(t) + F_{S2}(t)\sin\alpha(t), \\ F_{R2}(t) &= F_{S2}(t)\cos\alpha(t) - F_{S1}(t)\sin\alpha(t), \end{aligned} \tag{6.13}$$

where

$$\alpha = \arctan\left(\frac{b(t)\sin\theta(t)}{a(t)}\right)$$

is the relative orientation between the bodyfixed and sphere coordinate frames, and $a(t), b(t)$ are the components of the bodyfixed x-axis (along which F_{R1} acts) expressed in sphere coordinates:

$$\begin{aligned} \frac{da(t)}{dt} &= \sin\theta(t)\cos\theta(t)\phi(t)'b(t), \\ \frac{db(t)}{dt} &= -\frac{\cos\theta(t)}{\sin\theta(t)}(\phi(t)'a(t)+\theta(t)'b(t)). \end{aligned} \tag{6.14}$$

6.8.2 Applying the Maximum Principle

To apply the maximum principle, we set up the Hamiltonian

$$H = F_{S1}^2 + F_{S2}^2 + p_1\dot{\theta} + p_2\dot{\phi} + p_3(\dot{\phi}^2\sin\theta\cos\theta + F_{S1}) + p_4(-2\cos\theta\dot{\theta}\dot{\phi} + F_{S2})/\sin\theta.$$

The optimal control pair (F^*, ω^*) is

$$\begin{aligned} (F_{S1}^*, F_{S2}^*) &= \text{argmin}_{F_{S1},F_{S2}\in\mathcal{F}} H^* \\ &= \text{argmin }_{F_{S1},F_{S2}\in\mathcal{F}}[F_{S1}^2 + F_{S2}^2 + p_1^*\dot{\theta}^* + p_2^*\dot{\phi}^* \\ &\quad + p_3^*(\dot{\phi^*}^2\sin\theta^*\cos\theta^* + F_{S1}) + p_4^*(-2\cos\theta^*\dot{\theta^*}\dot{\phi^*} + F_{S2})/\sin\theta^*] \\ &= \text{argmin }_{F_{S1},F_{S2}\in\mathcal{F}}[F_{S1}^2 + F_{S2}^2 + p_3^*F_{S1} + p_4^*F_{S2}/\sin\theta^*]. \end{aligned} \tag{6.15}$$

Suppose that there is no restriction on the magnitude of the force ($F_{S1}, F_{S2} \in \mathbb{R}$); Eq. (6.15) leads to

$$F_{S1} = -p_3^*/2, \qquad F_{S2} = -p_4^*/(2\sin\theta^*),$$

where

$$\begin{aligned}
\dot{p_1^*} &= \frac{\partial H^*}{\partial \theta^*} = -p_3^*\dot{\phi}^*\dot{\phi}^* \cos 2\theta^* - 2p_4\dot{\theta}^*\dot{\phi}^*/\sin^2\theta^*, \\
\dot{p_2^*} &= \frac{\partial H^*}{\partial \phi^*} = 0, \\
\dot{p_3^*} &= -\frac{\partial H^*}{\partial \dot{\theta}^*} = -p_1^* + 2p_4^*\dot{\phi}^* \cos\theta^*/\sin\theta^*, \\
\dot{p_4^*} &= -\frac{\partial H^*}{\partial \dot{\phi}^*} = -p_2^* - p_3^* \sin 2\theta^* \dot{\phi}^* + 2p_4^*\dot{\theta}^* \cos\theta^*/\sin\theta^*. \qquad (6.16)
\end{aligned}$$

This is a set of coupled partial differential equations with unknown initial conditions (and final conditions) for the co-states. Considering the complexity involved in obtaining an analytical solution to Eq. (6.16), we decided to use numerical methods to solve this problem, specifically PBMS.

6.8.3 Applying Multiple Shooting

To apply the multiple shooting method, we defined a node sequence $\{1, 2, \ldots, N\}$, where at each node the states were

$$\{\theta_1, \ldots, \theta_N, \dot{\theta}_1, \ldots, \dot{\theta}_N, \phi_1, \ldots, \phi_N, \dot{\phi}_1, \ldots, \dot{\phi}_N\},$$

and the controls were

$$\{F_{S1}(1), \ldots, F_{S1}(N-1), F_{S2}(1), \ldots, F_{S2}(N-1)\}.$$

The cost function we want to minimize was

$$J = \sum_{i=1}^{N-1} (F_{S1}(i)^2 + F_{S2}(i)^2)\Delta t, \qquad (6.17)$$

where Δt was the length of the time segment. The NLP variable was

$$\begin{aligned}\nu &= \{\theta_1, \ldots, \theta_N, \dot{\theta}_1, \ldots, \dot{\theta}_N, \phi_1, \ldots, \phi_N, \dot{\phi}_1, \ldots, \dot{\phi}_N, \\ &\quad F_{S1}(1), \ldots, F_{S1}(N-1), F_{S2}(1), \ldots, F_{S2}(N-1)\}. \qquad (6.18)\end{aligned}$$

The controls for the initial trajectory are shown in Fig. 6.15. This was a problem

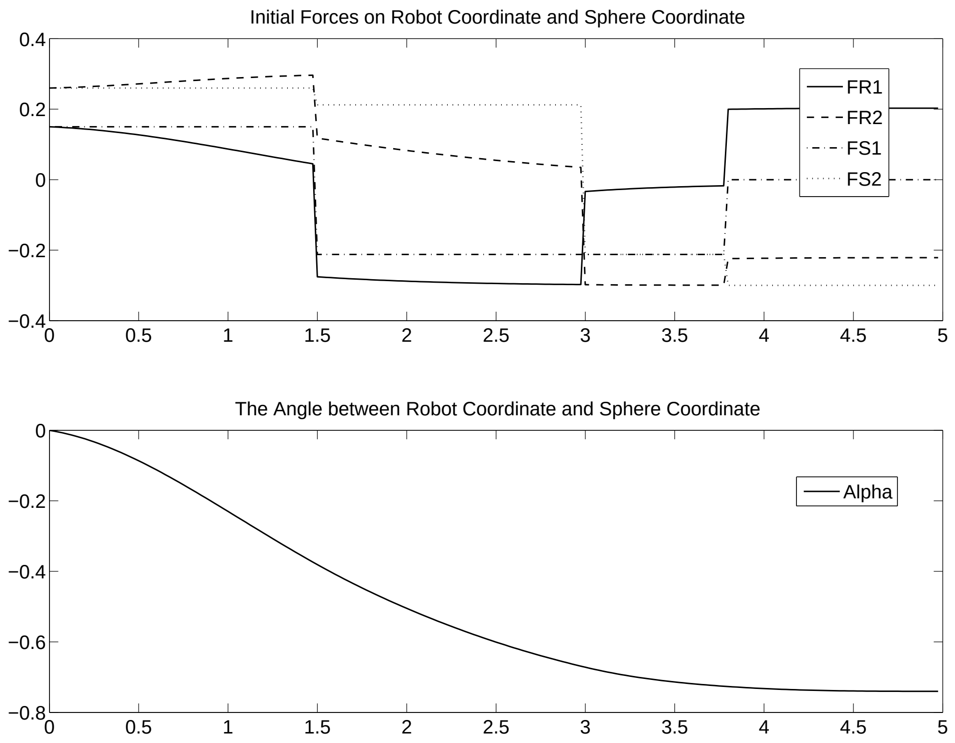

Figure 6.15: The controls for the initial trajectory in bodyfixed coordinate and sphere coordinate

with fixed final time and fixed final states, so we applied PBMS with $N = 201$, $\Delta = 60$, $\delta = 32$. For convenience, we decided to compute the optimal inputs in sphere coordinates first (F_{S1} and F_{S2}), and then transform those to their bodyfixed values F_{R1} and F_{R2} by means of Eq. (6.13). The resulting trajectories generated by agents are shown in Fig. 6.16, and the optimal controls are shown in Fig. 6.17. It took 810

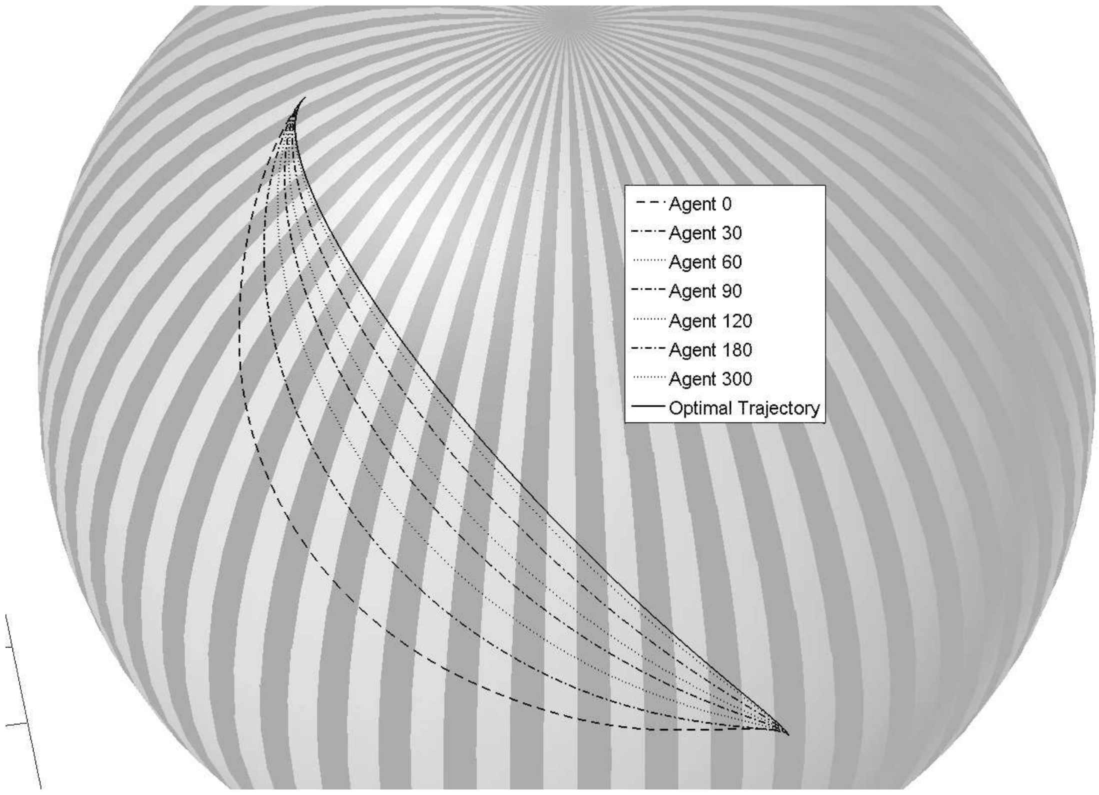

Figure 6.16: Trajectories generated by PBMS for this problem. The parameters selected were $N = 201, \Delta = 60, \delta = 32$.

agents to converge to the optimal trajectory with convergence criteria $\|J_{i+1} - J_i\| \leq 10^{-10}$ and $\|c(x)\| \leq 10^{-15}$.

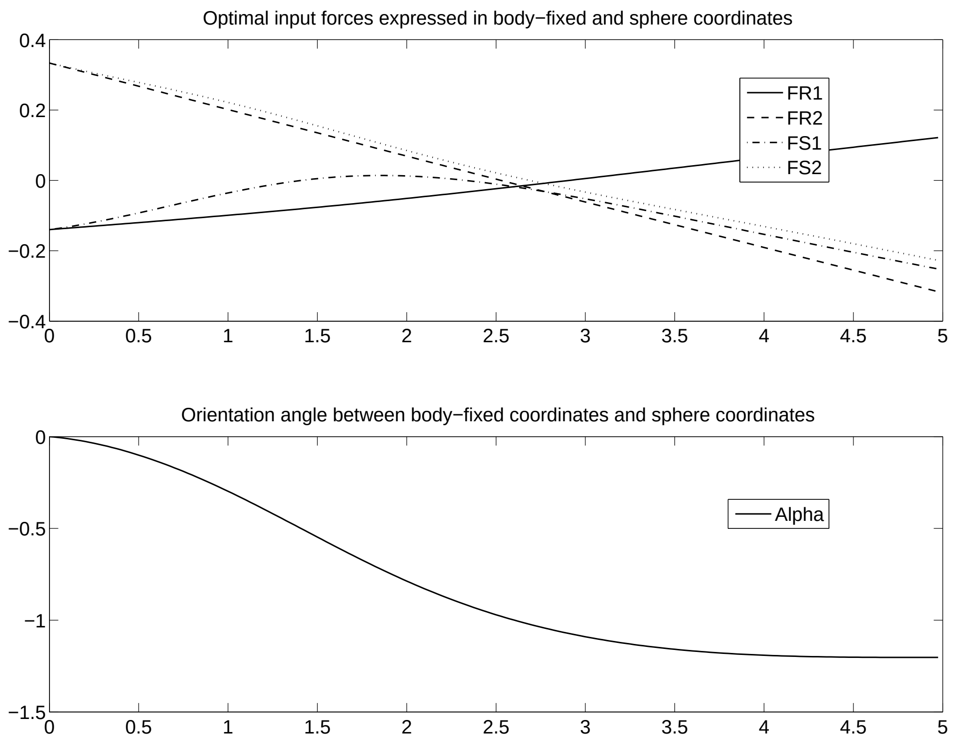

Figure 6.17: Top: optimal controls, expressed in bodyfixed and sphere coordinates. Bottom: The orientation angle $\alpha(t)$ of the bodyfixed coordinate frame in sphere coordinates.

Chapter 7

Conclusions

Inspired by the foraging activities of ant colonies that have the capability to form shortest paths between their nest and a food resource on uneven terrain, we developed a set of cooperative, pursuit-based algorithms that can be used to solve a broad class of optimal control problems. These algorithms differ according to the different boundary conditions that apply to the problems, namely fixed or free final time, and fixed versus free final state.

Our pursuit-based algorithms enable a group of cooperating agents – identical copies of a control system – to solve an optimal control problem by separating it into simpler instances, each of which can be solved by a pair of agents (designated as “leader” and “follower”) with limited capabilities. The first agent in the group evolves along a known feasible (but suboptimal) trajectory; subsequent members learn from their predecessors, successively improving the initial solution. The collective behavior of the group thus leads to a decrease of the cost function to be optimized. The learning process encoded in the proposed algorithms requires limited, short-range information exchanges to locate the leader, and it relies on an agent's sensing and computing capabilities to update its own trajectory. The trajectory sequence generated by the agents converges to a local optimum, as is the case with most analytical or numerical optimization methods that are based on necessary conditions for optimality. This optimum may be difficult for a single agent to obtain when considering the information constraints it must operate under. The proposed algorithms enable the group to act as "more than the sum of its parts" and are useful in a variety of settings, from robotic exploration of unknown regions with short-range sensors and low-band width communications, to numerical optimization for large-scale problems that involve nonlinear dynamical systems (as, for example, in the space science).

When applied as a numerical optimization method, local pursuit increases the permitting size of problems by breaking up the long trajectory into "smaller pieces", and solving a more manageable version of the problem over each piece. This property makes it possible to solve large-scale problems in computers with limited memory. In addition, local pursuit may improve the performance of existing numerical methods by introducing smaller accumulated errors and by enabling convergence in cases where traditional approaches fail.

7.1 Opportunities for Future Research

There are several research directions building upon on results of this thesis. One concern in local pursuit is that it may require an infinite number of agents to converge to the optimum. In practical settings, only a finite number of agents will be available. However, from the examples in Section 6.3, 6.6, one can see that a finite number of agents was sufficient when the inputs took on values in a finite set, e.g., bang-bang control. It would be of great practical importance to investigate the most general set of conditions under which the convergence generated by finite agents occurs.

Furthermore, it would also be interesting to investigate the performance of local pursuit under noisy sensors and state measurements. One feasible approach might be to first set up a generic model describing how a follower behaves when it pursues an optimally evolving leader (this kind of "local" analysis of the optimum under noisy measurements might be a good starting point), and then investigate the evolution of each leader/follower pair under noisy measurements.

Besides ants, other social insects, e.g. worker honey bees, demonstrate interesting and potentially useful coordinated behaviors, whose "mechanism" has been partly revealed by previous studies of bee activities. One may consider ways of "borrowing" the rules that govern the behaviors of bees and other insects, to develop additional biologically inspired algorithms for engineering. Potential topics include:

- The foraging activities of worker honey bees [13] could provide us with clues on solving resource allocating problems, including routing of pickup and delivery vehicles, and cooperative manufacturing by a limited number of robots.

- Further modeling of ant activities in [16] may help engineers to develop more effective adaptive control strategies. For example, one might consider searching for objects in a number of zones by a limited number of mobile robots. If each zones has the “unknown distribution of generating objects of interest”, one can attempt to devise ways in which the robots can learn the distribution of each zone, and develop decentralized rules for searching and convoying.

Appendix A

The Dynamics of a Massive Particle on S^2

From Eq. (6.9), we obtain that

$$\begin{aligned} \mathbf{x}_\theta(\theta,\phi) &= (-\cos\theta\cos\phi, -\cos\theta\sin\phi, \sin\theta), \\ \mathbf{x}_\phi(\theta,\phi) &= (-\sin\theta\sin\phi, -\sin\theta\cos\phi, 0) \end{aligned} \tag{A.1}$$

are a set of basis vectors for TS^2_x (but not normalized). The particle's velocity is

$$\begin{aligned} \dot{\mathbf{x}}(\theta,\phi) &= (\cos\theta\cos\phi, \cos\theta\sin\phi, -\sin\theta)\dot{\theta} + (-\sin\theta\sin\phi, \sin\theta\cos\phi, 0)\dot{\phi} \\ &= \mathbf{x}_\theta\dot{\theta} + \mathbf{x}_\phi\dot{\phi}. \end{aligned} \tag{A.2}$$

Taking derivative of Eq. (A.2) yields Eq. (6.11).

To find the relationship between the terms $\mathbf{x}_{\theta\theta}, \mathbf{x}_\theta, \mathbf{x}_{\theta\phi}, \mathbf{x}_\phi, \mathbf{x}_{\phi\phi}$, we calculate their inner products (using the metric "inherited" from Euclidean 3-space) and their norms

$$\begin{aligned} \|\mathbf{x}_\theta\| &= \|\mathbf{x}_{\theta\theta}\| = 1, \\ \|\mathbf{x}_{\phi\phi}\| &= |\sin\theta|\|\mathbf{x}_\phi\| = |\sin\theta|, \\ \|\mathbf{x}_{\theta\phi}\| &= |\cos\theta|, \end{aligned}$$

and

$$\begin{aligned} < \mathbf{x}_{\theta\theta}, \mathbf{x}_\theta > &= < \mathbf{x}_{\theta\theta}, \mathbf{x}_\phi >=< \mathbf{x}_{\theta\theta}, \mathbf{x}_{\theta\phi} >=< \mathbf{x}_{\phi\phi}, \mathbf{x}_\phi >=< \mathbf{x}_{\theta\phi}, \mathbf{x}_\theta >= 0, \\ < \mathbf{x}_{\theta\phi}, \mathbf{x}_\phi > &= - < \mathbf{x}_{\phi\phi}, \mathbf{x}_\theta >= \sin\theta\cos\theta, \\ < \mathbf{x}_{\theta\theta}, \mathbf{x}_{\theta\theta} > &= \sin^2\theta. \end{aligned}$$

Therefore we can conclude that

$$\bar{\mathbf{x}}_{\phi\phi} = \mathbf{x}_{\phi\phi}/\sin\theta, \quad \bar{\mathbf{x}}_{\theta\phi} = \mathbf{x}_{\theta\phi}/\cos\theta, \quad \bar{\mathbf{x}}_{\theta\theta} = \mathbf{x}_{\theta\theta} \tag{A.3}$$

and

$$\bar{\mathbf{x}}_{\theta\theta} \perp \bar{\mathbf{x}}_\theta, \quad \bar{\mathbf{x}}_{\theta\theta} \perp \bar{\mathbf{x}}_\phi, \quad \bar{\mathbf{x}}_{\theta\phi} = \bar{\mathbf{x}}_\phi, \quad \bar{\mathbf{x}}_{\phi\phi} = -\cos\theta\bar{\mathbf{x}}_\theta + \sin\theta\bar{\mathbf{x}}_{\theta\theta}. \tag{A.4}$$

Thus

$$\begin{aligned} \ddot{\mathbf{x}}(\theta,\phi) &= \dot\theta^2\bar{\mathbf{x}}_{\theta\theta} + \ddot\theta\bar{\mathbf{x}}_\theta + 2\dot\theta\dot\phi\cos\theta\bar{\mathbf{x}}_{\theta\phi} + \ddot\phi\sin\theta\bar{\mathbf{x}}_\phi + \dot\phi^2\sin\theta\bar{\mathbf{x}}_{\phi\phi} \\ &= [\ddot\theta + \dot\phi^2\sin\theta(-\cos\theta)]\bar{\mathbf{x}}_\theta + [\ddot\phi\sin\theta + 2\dot\theta\dot\phi\cos\theta]\bar{\mathbf{x}}_\phi + [\dot\theta^2 + \dot\phi^2\sin^2\theta]\bar{\mathbf{x}}_{\theta\theta} \end{aligned} \tag{A.5}$$

is the acceleration of the particle when no external forces are applied. Now consider applying an external force F that lies in the tangent plane at the particle's current position and makes an angle of ω with the direction of $\bar{\mathbf{x}}_\theta$. Newton's second law then implies:

$$\begin{aligned}
\ddot{\theta} + \dot{\phi}^2 \sin\theta(-\cos\theta) &= F_{S1} = F\cos\omega, \\
\ddot{\phi}\sin\theta + 2\dot{\theta}\dot{\phi}\cos\theta &= F_{S2} = F\sin\omega,
\end{aligned}$$

or, equivalently we could obtain Eq. (6.12).

To derive the parallel transport equations, first we calculate

$$\begin{aligned}
E &= < \mathbf{x}_\theta(\theta,\phi), \mathbf{x}_\theta(\theta,\phi) > \\
&= \cos^2\theta\cos^2\phi + \cos^2\theta\sin^2\phi + \sin^2\theta \\
&= 1, \\
F &= < \mathbf{x}_\theta(\theta,\phi), \mathbf{x}_\phi(\theta,\phi) > \\
&= \sin\theta\cos\theta\sin\phi\cos\phi - \sin\theta\cos\theta\sin\phi\cos\phi \\
&= 0, \\
G &= < \mathbf{x}_\phi(\theta,\phi), \mathbf{x}_\phi(\theta,\phi) > \\
&= \sin^2\theta\sin^2\phi + \sin^2\theta\cos^2\phi \\
&= \sin^2\theta, \qquad \text{(A.6)}
\end{aligned}$$

which are the components of the first fundamental form [14]. Also

$$E_\theta = E_\phi = F_\theta = F_\phi = G_\phi = 0, \;\; G_\theta = 2\sin\theta\cos\theta.$$

Therefore, the Christoffel Symbols $\Gamma^k_{ij}(i,j,k \in \{1,2\})$ satisfy

$$\begin{aligned}
\Gamma^1_{11}E + \Gamma^2_{22}F &= \frac{1}{2}E_\theta, \\
\Gamma^1_{11}F + \Gamma^2_{22}G &= F_\theta - \frac{1}{2}E_\phi, \\
\Gamma^1_{12}E + \Gamma^2_{12}F &= \frac{1}{2}E_\phi, \\
\Gamma^1_{12}F + \Gamma^2_{12}G &= \frac{1}{2}G_\theta, \\
\Gamma^1_{22}F + \Gamma^2_{22}G &= F_\phi - \frac{1}{2}G_\theta, \\
\Gamma^1_{22}F + \Gamma^2_{22}G &= \frac{1}{2}G_\phi. \qquad \text{(A.7)}
\end{aligned}$$

Solving Eq. (A.7) yields

$$\begin{aligned}
\Gamma^1_{11} &= \Gamma^2_{11} = \Gamma^1_{12} = \Gamma^2_{22} = 0, \\
\Gamma^2_{12} &= \frac{\cos\theta}{\sin\theta}, \\
\Gamma^1_{22} &= -\sin\theta\cos\theta. \qquad \text{(A.8)}
\end{aligned}$$

Suppose now that a curve on the sphere is parameterized by $(\theta(t), \phi(t))$. Let $w(t)$ be the unit vector parallel to the particle's u_1 axis, expressed in sphere coordinates. Because TS^2 is two-dimensional, $w(t)$ (equivalently, the angle it forms with the $\mathbf{x}_\theta$ axis) is sufficient to describe the orientation of the body-fixed coordinate system in sphere coordinates. We write

$$w(t) = a(t)\mathbf{x}_\theta + b(t)\mathbf{x}_\phi. \qquad \text{(A.9)}$$

Let $w_0 = a_0\mathbf{x}_\theta + b_0\mathbf{x}_\phi$ be the initial value of the particle's orientation. We compute Dw/dt, the tangent component of dw/dt, by taking w's derivative and dropping the normal component:

$$\begin{aligned}\frac{Dw}{dt} &= (a' + \Gamma^1_{11}a\theta' + \Gamma^1_{12}a\phi' + \Gamma^1_{12}b\theta' + \Gamma^1_{22}b\phi')\mathbf{x}_\theta \\ &\quad +(b' + \Gamma^2_{11}a\theta' + \Gamma^2_{12}a\phi' + \Gamma^2_{12}b\theta' + \Gamma^2_{22}b\phi')\mathbf{x}_\phi, \qquad (A.10)\end{aligned}$$

where the prime means the derivative with respect to t.

To obtain the parallel transport of the initial vector w_0, we must solve on-line the following coupled differential equations

$$\begin{aligned}\frac{da(t)}{dt} &= \sin\theta(t)\cos\theta(t)\phi(t)'b(t), \\ \frac{db(t)}{dt} &= -\frac{\cos\theta(t)}{\sin\theta(t)}(\phi(t)'a(t) + \theta(t)'b(t)).\end{aligned}$$

Having obtained the orientation $w(t)$, we can calculate the orientation angle $\alpha(t)$ between $w(t)$ and the $\bar{x}_\theta$ as

$$\alpha = \arctan\left(\frac{b(t)\sin\theta(t)}{a(t)}\right).$$

We can now find the components of the forces F_{R1}, F_{R2} in the sphere coordinate frame:

$$\begin{aligned}F_{R1}(t) &= F_{S1}(t)\cos\alpha(t) + F_{S2}(t)\sin\alpha(t), \\ F_{R2}(t) &= F_{S2}(t)\cos\alpha(t) - F_{S1}(t)\sin\alpha(t).\end{aligned}$$

Bibliography

[1] C. Ashcraft, R. G. Grimes, and J. G. Lewis. Accurate symmetric indefinite linear equation solvers. *SIAM Journal of Matrix Analysis & Application*, 40(3):636–666, Sep. 1998.

[2] N.S. Bakhvalov. On the optimization of numerical algorithms. In B. Bojanov and H. Woźniakowski, editors, *Optimal Recovery*, pages 1–58. Nova Science Publishers Inc., 1992.

[3] D.P. Bertsekas. *Dynamic programming and optimal control*, volume I. Athena Scientific, Belmont, Massachusetts, 2000.

[4] J.T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control and Dynamics*, 21(2):193–207, Mar.–Apr. 1998.

[5] J.T. Betts and W. P. Huffman. Trajectory optimization on a parallel processor. *Journal of Guidance, Control and Dynamics*, 31(10):955–958, Oct. 1986.

[6] A. Billard, A. J. Ijspeert, and A. Martinoli. A multi-robot system for adaptive exploration of a fast-changing environment: Probabilistic modeling and experimental study. *Connection Science*, 11:359–379, 1999.

[7] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Pergamon, 1984.

[8] R. A. Brooks. Artificial life and real robots. In *Proceeding of the First European Conference on Artificial Life*, pages 3–10, Cambridge, 1992. MIT Press/Bradford Books.

[9] R.A. Brooks and A.M Flynn. Fast, cheap and out of control: a robot invasion of the solar system. *Journal of the British Interplanetary Society*, 42:478–485, 1989.

[10] A.M. Bruckstein. Why the ant trails look so straight and nice. *The Mathematical Intelligencer*, 15(2):59–62, 1993.

[11] A.M. Bruckstein, C.L. Mallows, and I. A. Wagner. Probabilistic pursuits on the grid. *The American Mathematical Monthly*, 104(4):323–343, April 1997.

[12] A.E. Bryson, Jr., and Y.C. Ho. *Applied optimal control: optimization, estimation and control.* Hemisphere Pub. Corp., Washington, 1975.

[13] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems.* Princeton University Press, Princeton, New Jersey 08540, 2001.

[14] M. D. Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliff, NJ, 1976.

[15] E.K.P. Chong and S.H. Żak. *An introduction to optimization*. Wiley, New York, 1996.

[16] J-L. Deneubourg, S. Goss, J.M. Pasteels, D. Fresneau, and J-P. Lachaud. Self-organization mechanisms in ant societies (ii): learning in foraging and division of labor. In J.M. Pasteels and J-L. Deneubourg, editors, *From individual to collective behavior in social insects*, pages 177–196. Les Treilles Workshop, Basel, Boston, 1987.

[17] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. Ant systems: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 26(1):29–41, 1996.

[19] W. B. Dunbar and R. M. Murray. Model predictive control of coordinated multi-vehicle formation. In *Proceedings of the 41st Conference on Decision and Control*, pages 4631–4636, Las Vegas, Nevada USA, December 2002.

[20] R. C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings Congress on Evolutionary Computation*, pages 81–86, Hawaii, 2001.

[21] L. Edelstein-Keshet. Trail following as adaptable mechanism for popular behavior. In R. Murphey and P.M. Pardalos, editors, *Animal groups in three dimensions*, pages 282–300. Cambridge University Press, Cambridge, U.K., 1997.

[22] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001.

[23] P. J. Enright and B. A. Conway. Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. *Journal of Guidance, Control and Dynamics*, 15(4):994–1002, Jul.–Aug. 1992.

[24] D. Feng and B. H. Krogh. Acceleration-constrained time-optimal control in n dimensions. *IEEE Transactions on Automatic Control*, 14(2):431–439, Mar.–Apr. 1991.

[25] R. Fierro, P. Song, A. Das, and V. Kumar. Cooperative control of robot formation. In R. Murphey and P.M. Pardalos, editors, *Cooperative control and optimization*, pages 73–93. Kluwer Academic Publishers, 2002.

[26] G.D. Forney and Jr. The viterbi algorithm. In *Proceedings of the IEEE*, volume 61 of *3*, pages 268–278, March 1973.

[27] V. Gazi and K. M. Passino. A class of attraction/repulsion functions for stable swarm aggregations. In *Proceedings of Conference on Decision and Control*, pages 2842–2847, Las Vegas, Nevada, December 2002.

[28] V. Gazi and K. M. Passino. Stability analysis of social foraging swarms. *IEEE Transaction on Systems, Man and Cybernetics – Part B: Cybernetics*, 34(1):539–557, February 2003.

[29] V. Gazi and K. M. Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48(4):692–697, April 2003.

[30] C. W. Gear. *Numerical initial value problems with ordinary differential equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[31] D.M. Gordon. *Ants at work*. The Free Press, New York, 1999.

[32] W. W. Hager. Iterative methods for nearly singular linear systems. *SIAM Journal on Scientific Computing*, 22(2):747–766, 2000.

[33] D. Hristu-Varsakelis. Robot formations: Learning minimum-length paths on uneven terrain. In *Proceedings of the 8th IEEE Mediterranean Conference on control and Automation*, 2000.

[34] D. Hristu-Varsakelis, P. Krishnaprasad, S. Andersson, F. Zhang, P. Sodre, and L. D'Anna. The mdle engine: a software tool for hybrid motion control. Technical Report TR2000-54, Institute for systems Research, University of Maryland, College Park, MD 20742, Oct. 2000.

[35] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6), June 2003.

[36] H.B. Keller. *Numerical methods for two-point boundary value Problems.* Blaisdell Publishing Company, Waltham, Massachusetts, 1968.

[37] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE Int'l Conference on Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995.

[38] Y. Ketema and G. Balas. Agent-localized conditions for formation maintenance. In *Proceeding of the 42nd IEEE Conference on Decision and Control*, pages 1012–1016, Maui, Hawaii USA, December 2003.

[39] H.K. Khalil. *Nonlinear Systems.* Prentice Hall, New jersey, 2002.

[40] E.M. Khazen. Searching for optimal trajectory with learning. *IEEE Transaction on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31(6), 2001.

[41] D. Kincaid and W. Cheney. *Numerical Analysis.* Brooks/Cole Publishing Company, Pacific Grove, California, 1991.

[42] R. Kress. *Numerical Analysis.* Springer-Verlag New York Inc, New York, 1998.

[43] R. Kurazume and S. Hirose. Study on cooperative positioning system: optimum moving strategies for cps-iii. In *Proceeding of the 1998 IEEE International Conference in Robotics and Automation*, volume 4, pages 2896–2903, Leuven, Belgium, May 1998.

[44] R. Kurazume, S. Hirose, S. Nagata, and N. Sashida. Study on cooperative positioning system(basic principle and measurement experiment). In *Proceed-*

ing of the 1996 IEEE International Conference in Robotics and Automation, volume 2, pages 1421–1426, Minneapolis, MN, April 1996.

[45] N.E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 2968–2973, Orlando, Florida, December 2001.

[46] Y. Liu, K.M. Passino, and M. Polycarpou. Stability analysis of one-dimensional asynchronous swarms. *IEEE Transaction on Automatic control*, 48(10):1848–1854, 2003.

[47] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.

[48] R. Murphey and P.M. Pardalos. *Cooperative control and optimization.* Kluwer Academic Publishers, 2002.

[49] R.S. Nah, S.R. Vadali, and E. Braden. Fuel-optimal, low-thrust, three-dimensional earth-mars trajectories. *Journal of Guidance, Control, and Dynamics*, 24(6):1100–1107, Nov.–Dec. 2001.

[50] G.L. Nemhauser. *Introduction to dynamic programming.* John Wiley and Sons Inc., New York, 1966.

[51] A. Neumaier. Solving ill-conditioned and singular linear systems: a tutorial on regularization. *SIAM Review*, 40(3):636–666, Sep. 1998.

[52] P. Ögren, E. Fiorelli, and N.E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. submitted to IEEE Transactions on Automatic Control.

[53] P. Ögren, E. Fiorelli, and N.E. Leonard. Formation with a mission: Stable coordination of vehicle group maneuvers. In *Proceedings Fifteenth International Symposium on Mathematical Theory of Networks and Systems*, Notre Dame, IL, August 2002.

[54] J.K. Parrish and W.M. Hammer. *Animal groups in three dimensions.* Cambridge University Press, Cambridge, U.K, 1997.

[55] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint, and M. Baum. Cooperative control for autonomous air vehicles. In R. Murphey and P.M. Pardalos, editors, *Cooperative control and optimization*, pages 233–272. Kluwer Academic Publishers, 2002.

[56] J.M. Pasteels and J-L. Deneubourg. *From individual to collective behavior in social insects.* Les Treilles Workshop, Basel, Boston, 1987.

[57] D. G. Rajnarayan and D. Chose. Multiple agent team theoretic decision-making for searching unknown environment. In *Proceeding of the 42nd IEEE Conference on Decision and Control*, pages 2543–2548, Maui, Hawaii USA, December 2003.

[58] S.I. Roumeliotis and G.A. Bekey. Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795, 2002.

[59] H. Seywald. Trajectory optimization based on differential inclusion. *Journal of Guidance, Control and Dynamics*, 17(3):480–487, May–Jun. 1994.

[60] C. Shao and D. Hristu-Varsakelis. Biologically inspired algorithms for optimal control. Technical Report TR2004-29, Institute for Systems Research, University of Maryland, College Park, MD 20742, 2004.

[61] C. Shao and D. Hristu-Varsakelis. Biologically-inspired optimal control via intermittent cooperation. In *Proceedings of the 24th American Control Conference*, pages 1060–1065, Portland, OR, June 2005.

[62] H. Sussmann and J.C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, June 1997.

[63] S.R. Vadali and R. Nah. Fuel-optimal planar earth-mars trajectories using low-thrust exhaust-modulated propulsion. *Journal of Guidance, Control, and Dynamics*, 23(3):476–482, May–Jun. 2000.

[64] T. Vicsek, A. Czirok, E.B. Jacob, I. Cohen, and O. Schochet. Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, 75:1226–1229, 1995.

[65] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

[66] D.J. Wilde. *Optimum seeking methods*. Prentice-Hall, Englewood Cliffs, N.J., 1964.

[67] H. Yamaguchi. A cooperative hunting behavior by mobile-robot troops. *The International Journal of Robotics Research*, 18(8):931–940, September 1999.

[68] H. Yamaguchi and J.W. Burdick. Asymptotic stabilization of multiple nonholonomic mobile robots forming group formations. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3573–3580, 1998.