# *Decentralized Motion Planning within an Artificial Potential Framework (APF) for Cooperative Payload Transport by Multi-Robot Collectives*

by

**Leng-Feng Lee**

December 2004

A thesis submitted to the Faculty of the Graduate School of the State University of New York at Buffalo in partial fulfillment of the requirements for the degree of

**Master of Science**

Department of Mechanical and Aerospace Engineering
State University of New York at Buffalo
Buffalo, New York 14260

*To my family and friends*
*Without whom, I am nothing*

# Abstract

Cooperative mobile robots can provide interesting functionalities beyond single mobile robot. For many of its capabilities, motion planning such robot collectives provides the fundamental framework for cooperative become possible. In this thesis, we examine decentralized motion planning within an artificial potential field framework for cooperative payload transport by multi robot collectives. Artificial potential field approach is the implicit motion planning approach where the robot configuration space is modeled with potentials: obtascles create repulsive potential and target exerts an attractive potential. The sum of these potentials, create a total force that drives the robot to the target while avoiding obstacles. However, limitations of this seemingly simple and elegant approach arise when superimpose of the repulsive potential and attractive potential creates a local minima. Since the local minima may trap the robot in the middle of the motion planning, different form of potential functions were proposed to solve this problem. In the first part of this thesis, we studied the characteristics of various form of potential functions and their limitations. In particular, the navigation function that provides a potential field with a unique minimum is selected as the testbed for motion planning for group of robots.

In the second part of the thesis, we discussed methods to formulate the dynamics of a group of robots used in the literature, and then formulate the dynamic for such system to be used within a potential framework. Group of robots forming a geometry shape for payload transport can be view as a mechanical system with holonomic

constraints. Thus, we formulated the dynamics of group of robots as a constrained mechanical system, and the motion planning for this group of robots become solving the forward dynamic problem for this system. Extensive literature provides ways to solve such constrained mechanical system. In particular, we solved such system using three methods: Method I, direct Lagrange elimination method, Method II, penalty formulation approach, and Method III, constraint manifold projection method. For cooperative payload transport, strict formation maintenance is critical. As such, various case studies were then performed to examine the performance for each of the three methods in their formation maintenance ability, computation time, and their decentralization ability.

Based on the simulations results from various case studies, we conclude the thesis by discuss the general characteristics for each of the three methods from different aspects and suggest future work for this research.

# Acknowledgment

I would like to thank my advisor Professor Venkat N. Krovi. His guidance and encouragement have been invaluable throughout my undergraduate and graduate study. He helped me passed through some of the tough time as a friend, and enlightened me in the methods of scientific research as a researcher.

I would also like to expresses my sincere thanks to Dr. Tarunraj Singh and Dr. John L. Crassidis for serving on my thesis committee.

Everyone had difficult time in life, and I am glad to have friends who doesn't give up on me even when I give up on myself. For that reason, I would like to thank my fellow good friends in UB, Chia Ken, Chin Pei, J-J, S.K., Ashwin, T.K., Wai Keong, and Ying-ying, who help me and encourage me when I most needed. Their friendship and accompanion led me through those difficult days. Special thanks go to Alex in Texas and I-Kun and Wei Kwong in Callifornia, for the encouragements you guys gave to me. Also for the friends in Malaysia, Mrs Gan, Choon Yun, Choon Mei, and Choon Wei, the encouragements that I received from you all are nothing comparable.

In memory of uncle Gan and uncle Ping-Jiao, who always been my moral role, help me and believe in me, I promise to be the man as you guys expected me to be. This work is dedicated to both of you.

I would also like to thank my colleagues at the ARM Lab- Pravin, Chetan, Rajan, Glenn, Mike, Tao, Kiran, Anu, and Ajay who make the lab so diverse and thus interesting to work in.

Thanks to my parents, for instilling in me a curiosity and appreciation for scientific endeavor. The supports you gave to me during my long absence from home are valuable.

Finally, I would like to thank Lye Theng, who constantly encourage me and believe in my ability even when there is only little hope, had make all these worthwhile.

Thank you all.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In recent years, the study of groups of multiple autonomous mobile robots exhibiting cooperative behavior has emerged as an active and challenging research area. Ongoing revolutions in computing effectiveness and miniaturization of processors/ sensors/ actuators in the past decade have facilitated the creation and the transition deployment of networked distributed collectives of mobile robots in numerous applications from reconnaissance, foraging, hearding to cooperative payload transport. Despite the overall increase in complexity, such systems of multiple mobile robots are of great interest for many reasons [1]:

(1) The task may be inherently too complex (or impossible) for a single mobile robot to accomplish;

(2) Significant overall performance benefits can be gained by using a group of robots;

(3) Constructing and delivering several simple small-size robots can be easier, cheaper, more flexible and more fault tolerant than having a single big-size powerful robot for each separate task;

(4) The robot team might be more robust and reliable: if one robot fails, the other robots can take over and continue the mission;

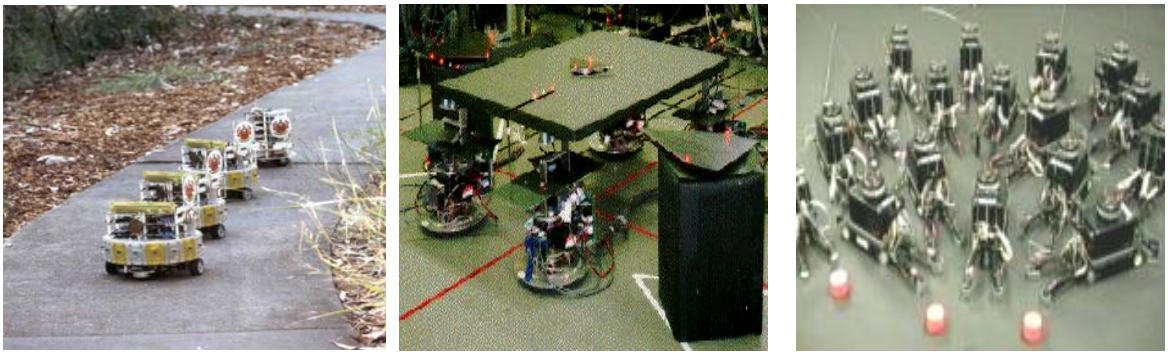
(5) The constructive, synthetic approach inherent in cooperative mobile robots system can possibly yield insights into fundamental problems in the social sciences (organization theory, economics, cognitive psychology), and life science (theoretical biology, animal athology).

It is especially note worthy that biologists who study animal aggregations such as swarms, flocks, school and herds (see Figure 1-1) have observed the remarkable group-level characteristics such as: the ability to make very fast and efficient coordinated maneuvers, quickly process data and to cooperatively make decisions in many of these natural groups.

**(a)** **(b)** **(c)**

**Figure 1-1: Group behavior in nature: (a) Fish moving in group; (b) Bird flying in V-shape; and (c) Cooperative payload transports by group of ants.**

Such groups in nature appear to make use of distributed control architecture in which individuals not only respond to their sensed environment (with limited ranges), but are also respond to (or are constrained by) the behavior of their neighbors. Recent literature has identified the ability of relatively simple constraints such as: (1) attraction to neighbors up to a maximum distance, (2) repulsion from neighbors if too close, and (3) alignment or velocity matching with neighbors as playing the principal role in maintaining a group formation. Additionally, these constraints may also be employed to explain a 'high-level emergent' group behavior such as finding a food source, or move to higher temperature area, while avoiding obstacles. Thus, there is a significant interest within the robotic community to better understand the biological imperative and exploit the same by incorporating similar principles in artificial robot collectives.



**(a)** **(b)** **(c)**

**Figure 1-2: Examples of Cooperative robots: (a) Cooperative autonomous robots developed in Inteligent System Control Lab in Griffith University; (b) Cooperative payload transport by group of robots developed in Mechanical Engineering department in Shizuoka University, Japan; and (c) Group of robots used in University of Southern California to demonstrate cooperative behaviors.**

The many application arenas for such artificial robot collectives came with varying requirements for "cooperations". Consider a case where a network of robots is to use in a reconnaissance mission. Increasing the numbers of networked robots would permit the site to be mapped in a much shorter time span than possible using only a single robot. Similarly, cooperative payload transport by a team of robots would be very beneficial when the payload is too big for a single robot to carry.

In both cases, system of robots has advantages over deploying a single robot but their cooperation requirements are different: in the first case, the network of robots might want to change their formation like expanding, contracting, or reconfiguration within the network; in the cooperative payload transport case, tight formation is desired to carry a common payload without losing it. Considerable amount of research attention has been focus on the first case where only loose formation maintenance is required [2-4]. In this thesis, however, we would like to examine the latter case which has stricter/tighter requirements on formation design and maintenance.

## 1.1 Our Systems

In this thesis, we will consider varying fidelities of modeling for the constituent robots within each robot collective. We will restrict ourselves to systems operating solely in the x-y plane but extension to systems moving in 3D, such as formation flying aircraft or spacecraft, is also possible. In all cases, we will begin by modeling the kinematics of the individual robots and combine this together with various coupling constraints to develop the kinematics/dynamics of the entire system. These aspects are discussed in greater detail in subsequent subsections.

### 1.1.1 Single Point-Mass Robot

A point mass robot is the simplest model of real robot and is the most common model employed (for the constituents of a system of robots) during the study of various kinds of potential field approaches to motion planning. Each robot is assumed to exhibit the dynamics of a single point mass particle with two degrees of freedom in the x-y plane as shown in Figure 1-3.

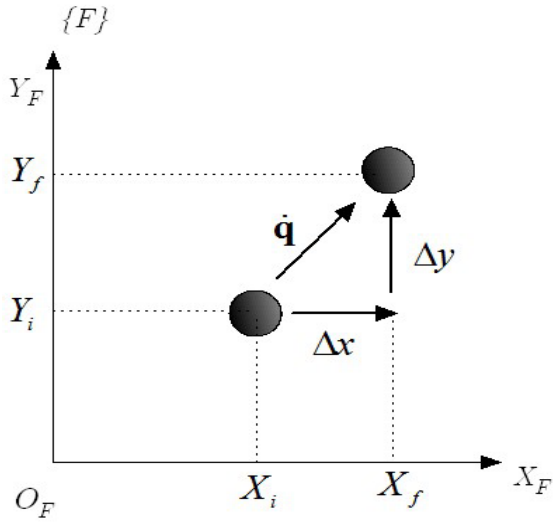



**Figure 1-3: A single point mass robot posses two degrees of freedom.**

### 1.1.2 Single Nonholonomic Wheeled Mobile Robot

A single differentially driven wheeled mobile robot (DD-WMR) is shown in Figure 1-4:

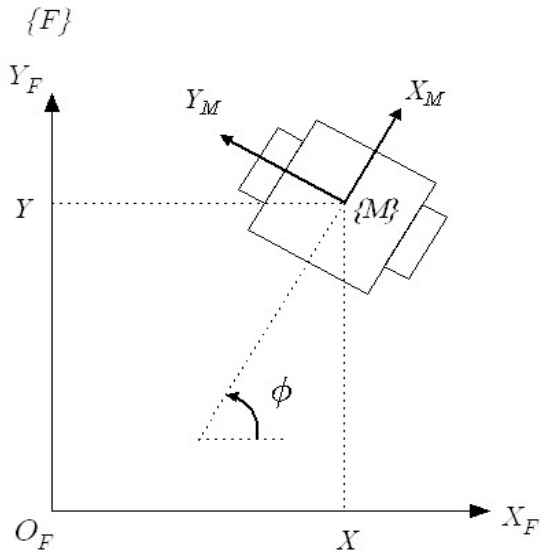



**Figure 1-4: Graphical representation of a nonholonomic wheeled mobile robot.**

Such a robot is also termed a nonholonomic wheeled mobile robot (NH-WMR) due to the nature of the constraints added by the individual disk-wheels. The nonholonomic constraints of the individual wheels combine with the differentially driven architecture to limit the possible motion of such robots. Any point on the robot lying along the axis between its two wheels cannot move in the direction of the axis. However, any point on the robot away from the axis is not bound to this constraint. In our thesis, we will primarily treat such WMR together with its nonholonomic constraints. However, on occasion we can reduce such WMR to an equivalent point mass model by selecting a point away from the axle. This is illustrated in Figure 1-5 and we will present more details of the formulation in Chapter 4.

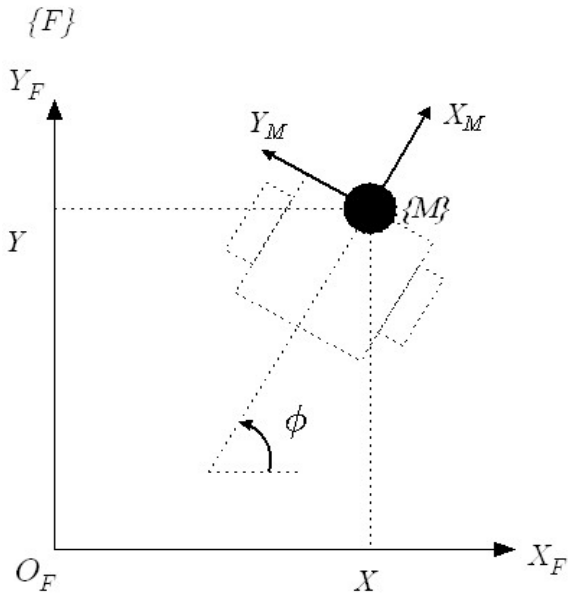



**Figure 1-5: Converting a nonholonomic wheel mobile robot to a point-mass robot representation.**

Other configurations for mobile robots are also possible such as the car-like robot [5], omni-directional mobile robot [6], etc but will not be considered in this thesis.

### 1.1.3 Team of Multiple Mobile Robots

These robots can now be assembled together to form collectives. While the overall goal is to develop the framework for motion planning of large ( $N > 100$ ) teams of robots, we will restrict our attention to teams of $N = 3$ and $N = 4$ in this thesis. A team of point-mass robots is shown in Figure 1-6 while a team of nonholonomic wheeled mobile robots is shown in Figure 1-7.

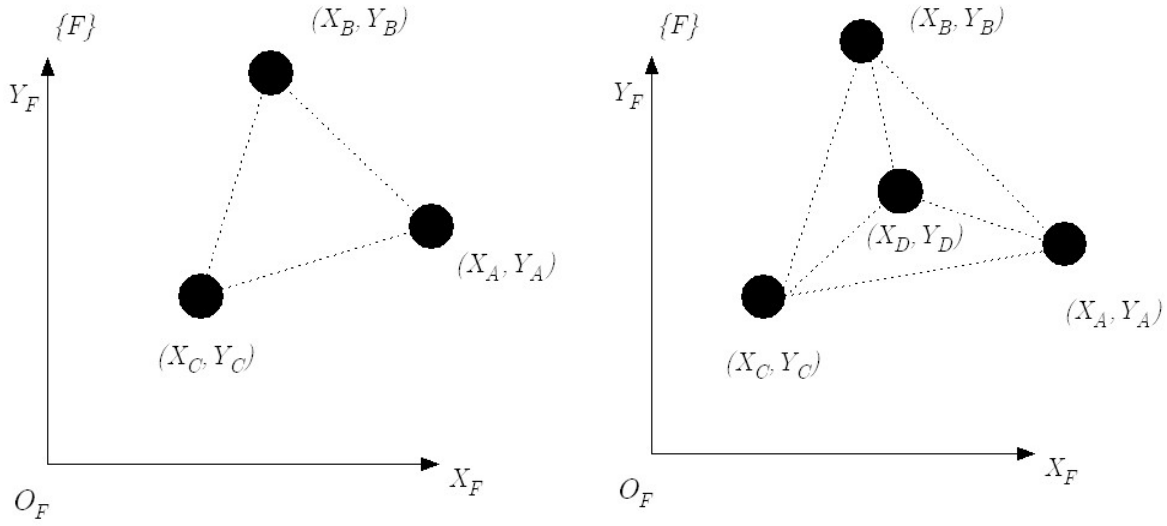



**Figure 1-6: A group of point-mass robot with (a) $N = 3$; and (b) $N = 4$.**

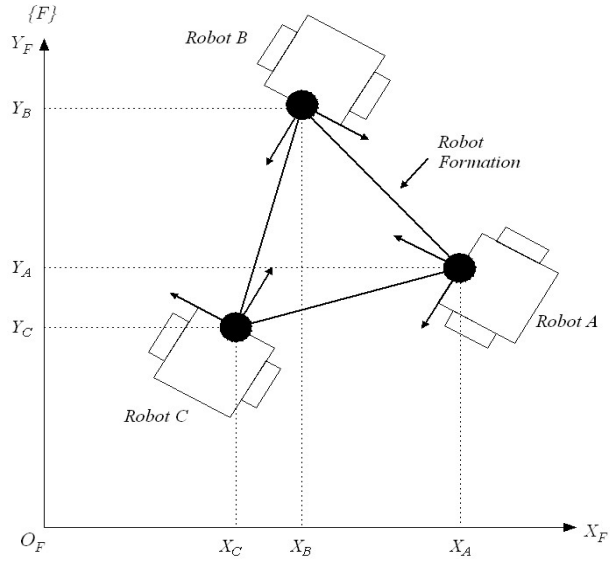



**Figure 1-7: A group of three nonholonomic wheel mobile robots in a triangular formation.**

## 1.2 Motion Planning

Motion planning algorithms can be classified into explicit motion planning and implicit motion planning [7], as will be discussed in greater detail in Chapter 2. Explicit motion planning approaches are by far the most common place approaches wherein the

motion planning task is decomposed into three steps: *path planning-trajectory planning-robot control*, as depicted in Figure 1-8.


**Figure 1-8: The traditional, explicit motion planning decomposed the motion planning task into three separate tasks.**

First, the path planning problem is solved using geometric data and the initial and final configurations to construct a collision-free curve connecting the robot's initial and final configurations. Such a curve determination is done based completely on the information of the configuration space (and without consideration of the robot's kinematics or dynamics). Second, the trajectory planning problem is solved by using the curve generated in the first step. The goal now is to find a time parameterization for the curve under certain constraints, to create the the reference trajectory. Finally, a control scheme is devised such that the physical robot follows the reference trajectory as closely as possible.

In implicit motion planning, the trajectory and the actuator inputs are not explicitly computed before the motion occurs. Instead, the motion plan implicitly specifies how the robot interacts with the environment and how it responds to sensory information and thereby specifies the desired dynamic behavior of the robot system. The drawback of implicit motion planning is that explicit performance guarantees can be difficult to achieve.

Nevertheless, implicit motion planning methods remain extreamly popular due to their ability to be implemented in an online implementation. Potential field approaches belong to this category of motion planning techniques and provides an intuitive way to model and analyze the behavior of groups of robot with many desirable characteristics. Typically, at group level, we model the workspace as a potential space where the global minimum is at the target, thus the group level motion become a gradient climbing problem. At an individual level, we model the potential field such that each individual is only affect by its neighboring robots up to a maximum distance, such that formation of the group can be maintained while achieving group motion.

While such cooperative systems can provide many advantages, the implementation of such systems also creates many challenges. The challenges lie in hierarchically combining these multiple levels of control within a common framework to realize truly decentralized yet scalable cooperative by such robot collectives.

## 1.3 Research Issues

### 1.3.1 Principal Issues

While many researchers may have looked at potential fields for individual robot or loose collectives of robots, in this thesis we will explore the use of potential field approaches for cooperative payload transport by multi robot collectives, which have more stringent requirements for formation maintenance. On this basis, this thesis can be separated into two parts. In the first part, we will study various variants of potential field approaches developed for use in motion planning; while the second part will focus on incorporating formation control within potential field framework for multi-robot motion

planning. Two principal research questions may be posed and the intimate coupling between these two parts is illustrated in the posing of these questions.

*Research Question 1: Which type of potential function is more suitable for use in motion planning for robot groups?*

Some very important criteria to search for the suitable potential field method to be used include computational time, convergence requirements, and the presence of local minima. Thus, in the first part of this thesis, we will study different types of potential fields that are widely used in the literature and discussed their salient characteristics. Considering local minima, for example, leads us to an investigation of the navigation function and other functions (such as the harmonic function) that do not have such local minima. After selecting a suitable potential field for the algorithm, the second research issue immediately becomes obvious:

*Research Question 2: How can we extend the potential field framework to help maintain formations? And how may this need to be further extended to realize the tight formation contact required of cooperative payload transport?*

Various variants of the Artifical Potential Field framework have been leveraged in motion-planning of robot collectives due to their seeming ease of formulation, decentralization and scalability. However, we note that while stability guarantees (typically asymptotic) may be obtained, potential field approaches are unable to

guarantee strict formation maintenance. Such strict formation maintenance is critical in application such as cooperative payload transport by collectives or in distributed sensor deployment application where the robots are to form some geometric pattern and maintain it while moving in the workspace.

We note that the group of independent mobile robots moving together in formation and coupled together by constraint dynamics can alternatively be viewed as a constrained mechanical system. The formulation and computation of motion plans for such collectives in a potential field may be treated as being equivalent to simulating the forward dynamics of a constrained multi-body mechanical system. By doing so, we can link and leverage the extensive literature on formulation and implementation of computational simulation of multibody systems [8-12].

Specifically, the constrained dynamics system may now be solve using three methods: (1) direct Lagrange multiplier elimination approach [13, 14]; (2) Penalty formulation approach [15]; or (3) Constraint manifold projection approach [16-19]. In particular, the penalty formulation approach is very closely analogous to the use of potential field approach to maintain formation found in [3, 4] – an aspect that will be closely investigated in our work.

### 1.3.2 Other Related Research Issues

Numerous others research questions may also be posed. These research questions are summarized pictorially in Figure 1-9 and many of the current efforts may be placed/ classified in the content of this picture.

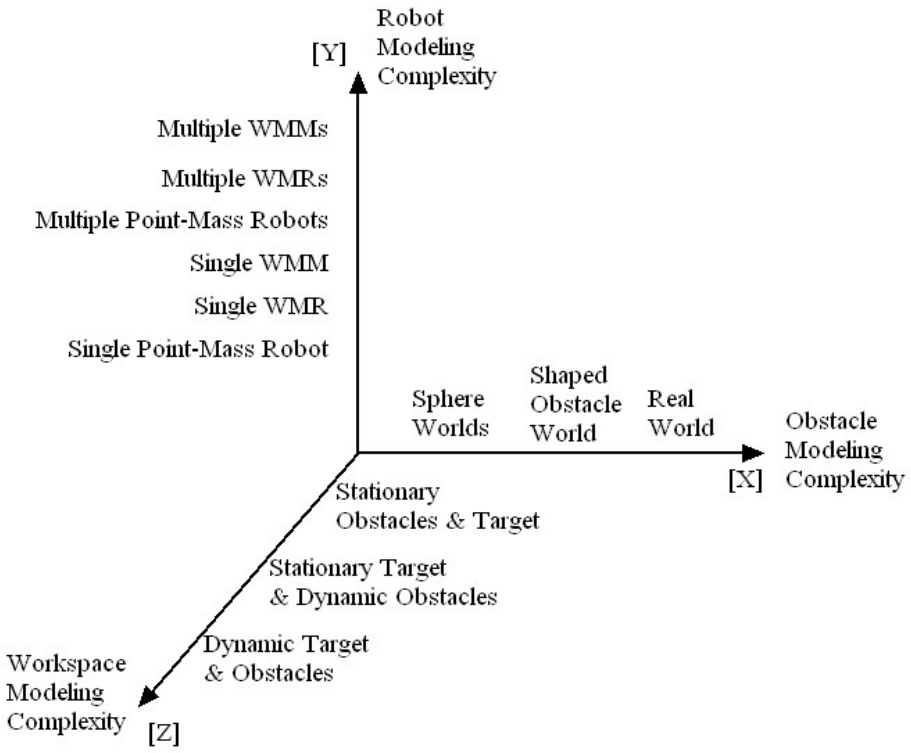



**Figure 1-9: Graph shows the hierachical difficulty in the motion planning problem.**

The $x$-axis is the *obstacle modeling* axis. The simplest case is the *sphere world* model, where all the obstacles are modeled as circular in shapes; with increasing complexity of modeling as one tries to capture real-world obstacles. Similarly we see that the $y$-axis is the *robot modeling axis*. Robot modeling techniques can range from The modeling of robot roughly range from: single point mass robot, single wheeled mobile robot (WMR), single wheeled mobile manipulator (WMM), multiple point mass robots, multiple WMRs, and multiple WMMs. A further classification into kinematic level modeling and dynamic level modeling is also possible. On the $z$-axis is the *workspace modeling* axis. While the simplest worskspaces only deal with stationary obstacles and targets, workspaces with dynamic obstacles and dynamic targets offer for greater challenges for motion planning.

In this thesis, we focus our attention on *sphere world* on the obstacle modeling axis; *multiple point mass robots* on the robot modeling axis, and *stationary obstacles and target* on the workspace modeling axis. While using our robot modeling and obstacle modeling, we can extend in the $z$-axis direction based on our formulation on the other two axes easily. Others constraints that arise due to particular implementation can add additional level of complexity to the motion planning problem. Nonholonomic constraints for differentially driven mobile platform and mixture of actuated and under-actuated n-link mobile robot are two examples.

## 1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview on motion planning for mobile robots, noting the distinctions with respect to control. We will however restrict ourselves to focus to the motion planning problem and explore some of the differences between explicit motion planning and implicit motion planning.

Within the class of implicit motion planning approaches, we will introduce some of the most commonly used local potential functions in Chapter 3. We will also highlight the benefits and limitations of such local potential field approaches prior to the discussion of navigation function in Chapter 4.

Chapter 4 focuses on navigation functions which are a class of potential fields that may be constructed with a unique global minimum. We begin by reinvestigating the steps to create a potential field that has only a unique minimum at the target position for a workspace with arbitary shaped obstacles. Although theoretically we are able to convert workspace of any shapes into potential field of with a unique minimum point with

appropriate diffeomorphisms, we limit ourselves to the *sphere worlds*- i.e., both workspace and obstacles are assumed to be circular in shape.

Chapter 5 presents the kinematic and dynamic formulations for single robot module and robot collective without formation constraints, which are then used to generate motion plans for the robot together with the potential field approaches described in Chapter 3 and Chapter 4. Various simulations are performed to study those potential field approaches, and gain valuable insights, prior to employing these approaches for group of robots in Chapter 6.

Chapter 6 introduces the concept of constraint dynamics and formulates the dynamics for groups of robots with formation constraint within a potential framework. Three methods were introduced to solve the constrained dynamics formulations – i.e. Method I, *direct Lagrange multiplier elimination method*, Method II, *penalty formulation method*, and Method III, *constraint manifold projection method*. We systematically specialize and demonstrate these steps to formulate the dynamics equations for a system of three mobile robots using the above three methods.

Chapter 7 presents simulation results for various interesting cases studies using the dynamic equation formulated in Chapter 6. In particular, four case studies were performed. The first two case studies pertain to the performance of each of the three methods in maintaining rigid formation, and the other two case studies evaluate their proformance when the robots undergo formation changes. The general characteristics of each method are also discussed base on the simulation results we obtained from these case studies

Chapter 8 summarizes the contributions in this work concludes with providing suggestions for future research.

# 2 Background

## 2.1 Motion Planning

Robots, which include mobile robots, manipulators, or the combinations of the two, mobile manipulators, are widely used for task such as material handling, spot and welding, spray painting, mechanical and electronic assembly, material removal and payload transport, etc. To realize all the above functionalities, the primary, or the fundamental problem is getting a robot to move from one initial configuration to a desired final configuration without collides with any obstacles. Except for some degenerate cases, there are usually infinitely many possible motions for performing such task: i.e., robots are usually redundant in degree of freedom (DOF) than it is necessary to perform a required task [20]. Even in workspaces that involve obstacles, where the interaction of the robot with the environment may impose additional constraints on the motion, the set of all possible motions is still typically very large. Moreover, for a given motion, there might be many different inputs that will produce the desired motion. Thus, the term *motion planning* in the context of robotic can be defined as: The process of selecting a motion and the associated set of input forces and torques from the set of all possible motions and inputs while ensuring that all constraints are satisfied.

The motion of a robot system can be described in three different spaces. First, the task is specified in the so-called *task space* or *Cartesian space* using same suitable

parameterization of rigid body position and orientations. However, for *n* degree-of-freedom robot, it may be necessary to specify the robot motion in the *joint space*, by specifying the motion of *n* independent robot joints. The joint space is the Cartesian product of the intervals describing the allowable range of motion for each degree of freedom. Often, it is a subset of $\mathbb{R}^n$. Finally, because the robot is a mechanical system governed by the equations of motion derived from physics, for each motion that is consistent with the kinematic and dynamic constraints, there must exist at least one set of input forces and torques that produce the motion. The actuator forces or torques that achieves all possible motions define the *actuator space*. Since actuator forces or torques are real-valued, the actuator space is a subset of $\mathbb{R}^m$, where *m* is the number of actuators. Based on this classification, it is easy to see that it is possible to define a motion planning problem in the *task space*, the *joint space*, or in the *actuator space*, given a particular problem at hand.

For each mobile robot, we consider the task space to be all possible motions in the plane. For point mass robots it is $\mathbb{R}^2$ while for the differentially driven nonholonomic wheeled mobile robots this space is the Lie group $SE(2)$, the special Euclidean group in two dimensions. While the task-space can potentially serve as a configuration space, for many articulated systems, the joint space is a more suitable candidate for the configuration space. The principal reason is that joint-based configuration spaces are endowed with well defined biinvariant metrices which may not always be the case for task spaces (such as $SE(2)$ and $SE(3)$).

The principal theme of any developed motion plan is to find a collision free trajectory between a starting location and a final destination. Within the configuration

space, an obstacle $Obs_i$ maps to a set of configuration space, $C_{Obs_i}$ in which the robot touches or penetrates the obstacle. Most of the algorithms fall into one of these two main categories: (1) Explicit motion planning, and (2) Implicit motion planning. Our discussion of these two algorithms biased towards implicit motion planning. In particular, the potential field approach is discussed in detail since this is the method that we used in this thesis.

## 2.2 Motion Planning vs Control

Traditionally, *motion planning* is the process of selecting a motion and the associated set of input forces and torques from the set of all possible motions while ensuring all the constraints are satisfied. This process can be viewed as a set of computations typically performed offline that provide sub goals or set points for robot control. The computations and the resulting motion plans are based on a suitable model of the robot and its environment.

On the other hand, *control* can be defined as the task of getting the robot to follow the planned motion. Robot control, involves taking the planned motion in the motion planning phase as the desired motion and the planned actuators inputs as the nominal inputs, measuring the actual motion of the system and controlling the actuators to produce the appropriate input forces or torques to follow the desired motion. Generally, the inputs required for robot control will deviate from the nominal inputs to compensate for modeling errors and non ideal actuators and sensors.

However, since motion planning is viewed as process that generates set points for control, which in recent times can be performed in an online manner, often the distinction between planning and control blurs. However, in this thesis, we will limit ourselves to

motion planning problems and if necessary, some issues pertinent to control will also discussed.

## 2.3 Classification of Motion Planning Algorithm

*Explicit motion planning* methods involve explicit computation of the trajectory of the system (and in some cases the associated inputs in an appropriate space) in order to produce a set of sub-goals or set points for a suitable low-level controller. The notion of treating motion planning as a set of computations that produce a set of sub-goal or set points for a lower level controller involve explicit computations of the trajectory of the system, and in some cases the associated inputs in the appropriate space. Such motion plans are called *explicit motion planning*.

Explicit method can either be discrete or continuous. Discrete approaches [21-23] focus on geometric constraints, which generally introduced by the presence of obstacles, and the problem of finding a set of discrete configurations between the start and the goal configuration that are free from collisions with obstacles. Examples of discrete approaches are road map [21, 22] and cell decomposition methods [23]. On the other hand, the explicit, continuous methods try to find feed forward or open loop control laws for the control of the dynamic system [24-28]. In fact, explicit, continuous motion plans can be viewed as open loop control laws. They consist of the reference trajectory for the system and the associated actuator force, and they are completely devoid of sensory feedback.

In contrast to explicit motion planning, in *implicit motion planning*, we have implicit motion plans in which the trajectory and the actuator inputs are not explicitly computed *before the motion occurs*. Instead, the motion plan specified how the robot

interacts with the environment and how it responds to sensory information. One of the easiest examples of this kind of approach is the potential field algorithm that developed by Khatib and later Krogh [29, 30]. The key idea is to construct an artificial potential field which permits the robot attracted to the goal while at the same time being repelled by obstacles in the environment. The motion of the robot is determined by the artificial potential field.

Similary, in contrast to open-loop control law analogy for explicit, continuous motion plan, closed loop control laws can be viewed as implicit plans that characterize the deviation of the dynamic system from the nominal trajectory that is specified, possibly by an explicit motion plan. Generally, such implicit methods are always continuous in flavor.

In the subsequent paragraphs, we will discuss potential field approaches (which are a form of implicit motion planning) in greater detail.

## 2.4 Potential Field Method

In the past decades, *potential field methods* (PFM) have gained popularity among robotic researchers especially in the mobile robot arena [1, 3, 6, 20, 30-34] due to its mathematical simplicity and elegance. Such approaches have been variously termed as "*Potential Field Approaches*", "*Potential Field Methods*", "*Artificial Potential Approaches*", "*Virtual Potential Approaches*", "*Potential Based Approaches*", or "*Potential Field-based Navigation Methods*". In its simplest form, such potential field method can be implemented quickly and provide acceptable results without requiring

many refinements. In fact, often time this method is suitable for real-time implementation requiring only local gradient information and without requiring global information.

The first example of such a potential field method, was proposed by Khatib in early 80's [29, 30]. In this approach, a potential field is defined in the configuration space such that it has a minimum at the goal configuration. While the target is ideally at the minimum, all obstacle, or walls, are assumed to created a high potential hill. Thus, in such a potential field, the robot will be attracted to the target while at the same time repelled by obstacles in the workspace. The sum of all forces, determines the direction and speed of travel for the mobile robot. Motion planning then utilize the gradient information of an artificial field created using potential functions as an input force in the system's dynamics equations that drives the robot to its desired destination while avoid collision with obstacles.

Typically, the attractive potential field and the repulsive potential fields are formulated separately, and the total potential field of the workspace is obtained by linear superposition of the two fields. Examples of potential function designed with this idea are Krogh's GPF function [29], Khatib's FIRAS function [30], superquadric potential function [35], Ge and Cui's new potential function [33, 36], harmonic potential function [32, 37-39], and Beard's attractor and repulsor potential [40]. Since only local gradient information is needed to formulate these potential functions, these are called the *local potential approaches*. Such approaches are very attractive from a computational point of view since no processing is required prior to motion. Further, it is easy to specify a dynamic behavior that tends to avoid obstacles. In fact, there is a school of thought that believes human motor control and planning is organized in a similar fashion [41]. We

will discuss the characteristics and formulations of various types of local potential functions and also discussed some of the limitations commonly found in these potential functions in Chapter 3.

The main drawback of the potential field approach is that when obstacles are present, the potential field may not be convex and local minima that can 'trap' the robot may exist at points away from the target. This local minimum is the result of unpredictable shape of total potential field after the superposition of attractive and repulsive potential fields. The second disadvantage of this approach is that it is difficult to predict the actual trajectory. In practice, one has to carefully pick the constants in the potential function in order to guarantee obstacle avoidance. Furthermore, the generated trajectories are usually far from being optimal by any measure. Many of later approaches were developed to help overcome some of these limitations. Volpe and Khosla [42] introduced a *Superquadric Artificial Potential Functions* [35, 43], which model a wide range of shapes range from rectangles to ellipse using superquadric formulas [44, 45]. *Harmonic Potential Functions*, were introduced by Kim and Khosla [32, 37], and separately by Connolly[38], based on the *Laplace Potential* introduced by S. Akishita [46]. This function, which based on the numeric solution to Laplace's heat equation, provides a potential field without local minima that most potential field methods suffered from. All of the above potential field functions do not take into consideration of moving obstacles and moving target. S.S. Ge and Y.J Cui, proposed a new potential field function that is both function of positions and velocities [6, 33, 36] of the robot, obstacles, and target, solved this situation. Another common problem found in most potential field methods, the *goals nonreachable with obstacle nearby*, or the GNRON problem,

identified by S.S. Ge and Y.J Cui [36] and R. Volpe and P. Khosla [35, 43] , can be handle using their proposed potential fields.

While the linear superposition capability is one the desirable benefits of potential field approach, it can often generate undesirable local minima. To overcome the local minima problem, many different forms of the individual potential functions have been proposed in the literature [6, 29, 32, 33, 36-39, 43, 46]. Table 2-1 summarizes the principal approaches in tabular form and these will be discussed in greater detail in subsequent sections.

| | Types of Potential Functions | | | | | |
|---|---|---|---|---|---|---|
| | FIRAS Function | GPF Function | Superquadric Potential | Harmonic Function. | Navigation Function | G&C Function |
| No Local Minima - single obstacle | No | No | Yes | Yes | Yes | No |
| No Local Minima- multiple obstacles | No | No | No | Yes | Yes | No |
| Handle GNRON problem | No | Yes | Yes | Yes | Yes | Yes |
| Only Position level information needed. | Yes | No | Yes | Yes | Yes | Yes |
| References | [30, 47] | [29] | [29, 35, 43] | [38, 39, 48-51] | [52-55] | [6, 33, 36] |

**Table 2-1 : Examples of Potential Functions and situations where they are applicable.**

There is a class of global potential field approaches exemplified by navigation function [34, 40, 55-58] which were developed to overcome some of these limitations and provide a global minimum. The so-called *navigation function* can be explicitly designed to posses a unique global minimum at the target configuration. All other

equilibrium points (if they exist) are saddle-points and lie in a set of measure zero. We note that constructing such a navigation function requires the complete knowledge of the topology of the environment, and additionally also entails lost of much of the simplicity, and computational advantages inherent in the local potential field approaches. However, the navigation function provides us a standardized potential field for us to formulate and test the motion planning strategy for multiple robots. We will discuss many of these aspects in Chapter 4.

Table 2-2 quickly summarizes some of the principal characteristics, benefits, and limitations of local vs global potential field approaches.

| Two Formulation Approaches | | |
|---|---|---|
| | Local Potential Approaches | Global Potential Approaches |
| Characteristics | Attractive potential and repulsive potential are formulated separately and then added together to obtain the total potential field. | Target and obstacles information are used in a unified framework to create the total potential field. |
| Advantages | 1. Not required to know the complete information of the workspace.<br>2. Adding an additional obstacle to the workspace by superimpose of a repulsive potential to the total potential. | Provide a potential field with only global minimum. |
| Disadvantages | Local minimum exist. | 1. Required to know the complete information of the workspace.<br>2. Adding an additional obstacle required reconstructing the total potential. |
| Examples | 1. Krogh's GPF function [29];<br>2. Khatib's FIRAS function [30];<br>3. Superquadric potential [35];<br>4. G&C potential functions [6, 33, 36];<br>5. Harmonic potential function [32, 37-39];<br>6. Attractor and repulsor potential [40]. | Navigation Functions [34, 40, 55-58]. |

**Table 2-2: Comparison table for Local potential field approach and Global potential field approach.**

### 2.4.1 Formation Control for Group of Robots

While considerable amount of literature exists for motion planning of individual mobile agent, the renewed challenges lies in creating motion plans for the entire team while incorporating notion such as cooperation. The formation paradigm has emerged as a convenient mechanism for abstraction and coordination with approaches ranging from leader-following [59, 60], virtual structures [40, 61, 62], and virtual leaders [3, 4, 31, 63]. The group control problem now reduces to a well-known single agent control problem from which the other agents derive their control laws but requires commuinication of some coordination information. Early implementations involved the kinematic specification of the followers' motion-plan as a 'prescribe motion' relative to a team-leader without the ability to affect the dynamics of the leader. Subsequent approaches have incorporated some form of 'formation feedback' from the members to the overall group using natural or aritificially introduced dynamics within the constraints. The formation paradigm has evolved to allow prescription of parametrized formation maneuvers [63, 64] and allow group feedback [31, 61, 63-68].

From these seemingly disparate approaches, a dynamic system-theoritic perpestive has emerged for examing the decentralized multi-agent 'behevioral control' in the context of 'formation'. 'Behavioral' control laws, derived implicitly as gradients of limited-range artificial potentials [2-4, 31, 64], can be implemented in a decentralized manner while permitting a Lyapunov-baed analysis of formation performance. The approach we used in this thesis will be discussed in detail in Chapter 6.

# 3 Local Potential Field Approaches

In this chapter, we examine some of the local potential field functions discussed in section 2.4 in greater detail using several numerical case studies. First, we will introduce variants of the attractive and repulsive functions. We will then use linear combinations of these candidate variants to develop the total potential field and use it to illustrate the advantages and their limitations.

The fundamental idea behind potential field approaches is to treat the target position as a attractive well, where the minimum is at the target; and to treat obstacles as high potential hill that create a repulsive force. The overall potential is the sum of these two types of potentials and can be written as:

$$U_{Total}(\mathbf{q}) = U_{Att}(\mathbf{q}) + U_{Rep}(\mathbf{q}) \tag{3.1}$$

Where $U_{Total}(\mathbf{q})$ denote the *total artificial potential field*; $U_{Att}(\mathbf{q})$ denote the *attractive potential field*; and $U_{Rep}(\mathbf{q})$ is the *repulsive potential field*. All of them are function of position $\mathbf{q}$ only; where $\mathbf{q} = [x, y]^T$ in two dimensional workspace and $\mathbf{q} = [x, y, z]^T$ in a three dimensional workspace. With this idea in mind, if the robot is design to follow the negative gradient in the total potential field, it will finally converge to the target position since that is the lowest point in the potential field.

### 3.1.1 Attractive Potential Field

In general, the attractive potential field has the form of Figure 3-1, where in every point of the workspace, the negative gradient flows pointed towards the target.

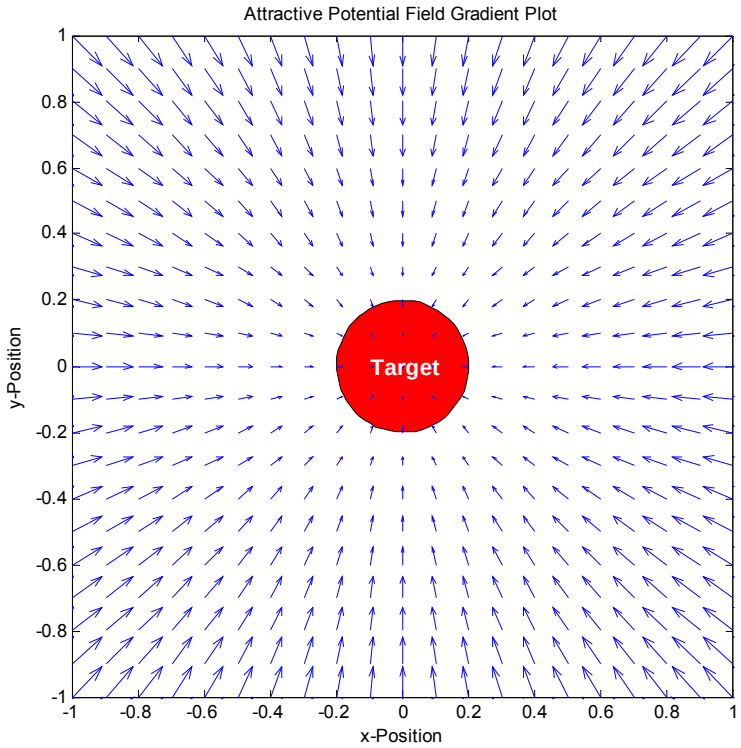



**Figure 3-1: Gradient plot of a general attractive potential field.**

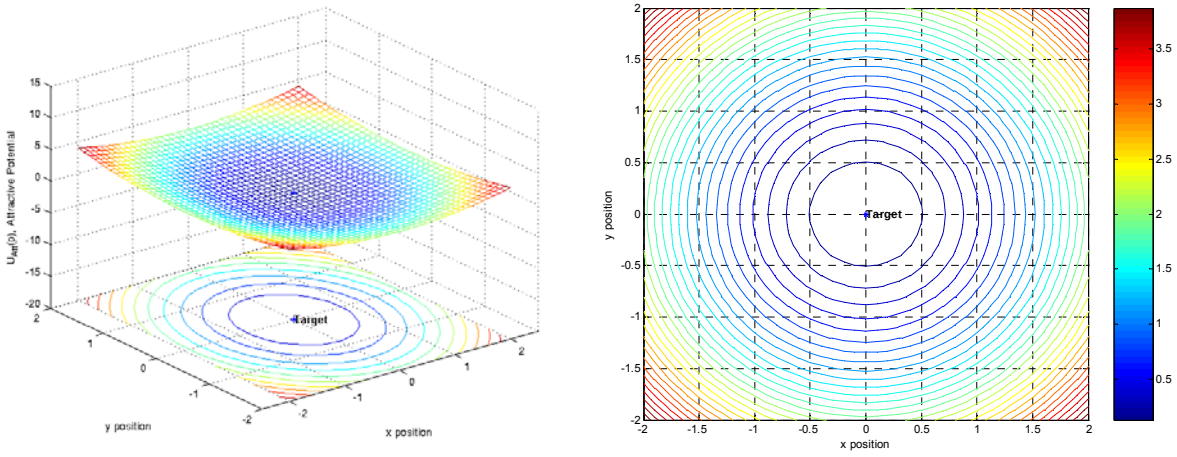
While many different *attractive potential function* have been proposed in the literature, the most commonly used attractive potential field takes the form of

$$U_{Att}(p) = \frac{1}{2}\xi \left\| \mathbf{q}_{Tar} - \mathbf{q}_{Rob} \right\|^{m} \tag{3.2}$$

where $\xi$ is a positive scaling factor, $\mathbf{q}_{Tar}$ and $\mathbf{q}_{Rob}$ denote the position for the target and the robot, respectively. $\left\| \mathbf{q}_{Tar} - \mathbf{q}_{Rob} \right\|$ is the Euclidean distance between the robot and the target, which is only a function of position, and $m$ is any positive number greater than zero.

For $0 < m \leq 1$, the Attractive Potential Field is conic in shape and the resulting attractive force has constant amplitude except at the goal, where $U_{Att}(\mathbf{q})$ is singular. Thus, it is common to use $m \geq 2$ to provide a minimum attractive potential value at the target; it is also possible to use different value for $m$ to avoid local minima in the presence of obstacles, as shown by Ge and Cui [33].

Figure 3-2 shows the attractive potential field created using Equation(3.2).




(a) (b)

**Figure 3-2: (a) Attractive Potential Field in 3D view and (b) in contour plot, with $m = 2$, $\xi = 1$ and target position at (0, 0).**

From Equation(3.2), we can modify the shape of the Attractive Potential Field by changing the value of $m$, and we can modify the effect of the Attractive Potential Field by modifying the value of $\xi$. Also, for $m \geq 2$, we could easily get the gradient information by:

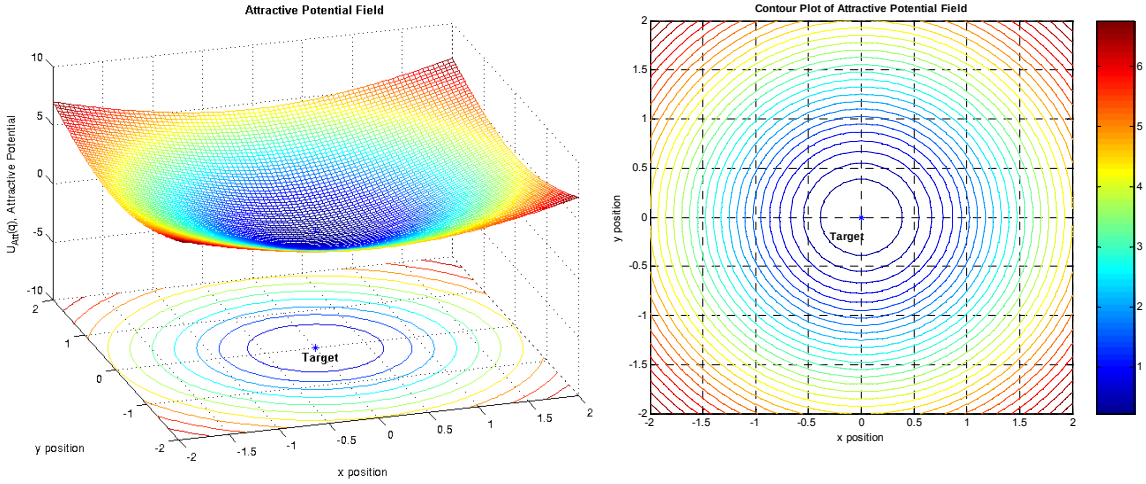
$$\frac{\partial U_{Att}(\mathbf{q})}{\partial \mathbf{q}_{Rob}} \tag{3.3}$$

In order to provide greater flexibility, composite functions can also be created [26], where the function is *quadratic* within a given range and then increases *linearly*:

$$U_{Att}(p)=\begin{cases}\xi\left\|\mathbf{q}_{Tar}-\mathbf{q}_{Rob}\right\|^2 & , \quad \left\|\mathbf{q}_{Tar}-\mathbf{q}_{Rob}\right\|<s \\ 2ks\left\|\mathbf{q}_{Tar}-\mathbf{q}_{Rob}\right\|-ks^2, & \left\|\mathbf{q}_{Tar}-\mathbf{q}_{Rob}\right\|\geq s\end{cases} \tag{3.4}$$

where $\xi$ is a positive scaling factor, $\left\|\mathbf{q}_{Tar}-\mathbf{q}_{Rob}\right\|$ is the Euclidean distance between the robot and the target. Thus, such potential field takes the form of Equation (3.2), with $m=2$, for smaller distances, and transformed into a constant magnitude, centrally attractive force field for distances larger than $s$.

In Figure 3-3, an attractive potential field was created using Equation(3.4) over a workspace of $-2<x<2$ and $-2<y<2$. The parameters setting are $s=1$, $k=1.5$ and $\xi=3$. The target is positioned at $(0,\ 0)$. Note however, that the value of $k$ and $\xi$ must be properly select such that it will not create a sudden drop at the boundary define by $s$. Often to avoid this, additional requirement for matching two piecewise continuous into composite functions tend not to be used.




(a) (b)

**Figure 3-3: (a) An attractive potential field created using Equation (3.4); and (b) Contour plot of the attractive potential field.**

The difference between the attractive potential created using Equation(3.4) and the one created using Equation(3.2) can be observed from their respective contour plots- In Figure 3-3(b), the contours are more uniformly spread than the one shown in Figure 3-2(b).

Traditionally, potential field are only function of position only. However, attractive potential field that include the velocity information of the robot is also possible. This type of attractive potential was first used by Krogh (only velocity of the robot) and later by Ge and Cui [6, 29]. The one used by Ge and Cui has the following form:

$$U_{Att}(p) = \alpha_P \left\| \mathbf{q}_{Tar} - \mathbf{q}_{Rob} \right\|^m + \alpha_V \left\| \mathbf{v}_{Tar} - \mathbf{v}_{Rob} \right\|^n \tag{3.5}$$

where $\mathbf{v}_{Tar}$ and $\mathbf{v}_{Rob}$ denote the velocities of the target and robot, respectively; $\mathbf{v} = [\dot{x}, \dot{y}]^T$ in 2-dimensional space and $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ in 3-dimensional space; $\left\| \mathbf{v}_{Tar} - \mathbf{v}_{Rob} \right\|$ is the magnitude of the relative velocity between the target and the robot; $\alpha_P$ and $\alpha_V$ are positive scalar parameters; and $m$ and $n$ are positive constants.

LATER !

Such a formulation is advantageous if we have a moving target: while the first term in Equation (3.5) drives the robot to the target and shortens the distance between them, the second term drives the robot to move at the same velocity of the target and then vanished when the robot reaches its target velocity.

### 3.1.2 Repulsive Potential Field

In general, repulsive potential fields are intended to generate a high potential around the obstacle, such that the gradient flow points away from the obstacle, as shown in Figure 3-4.

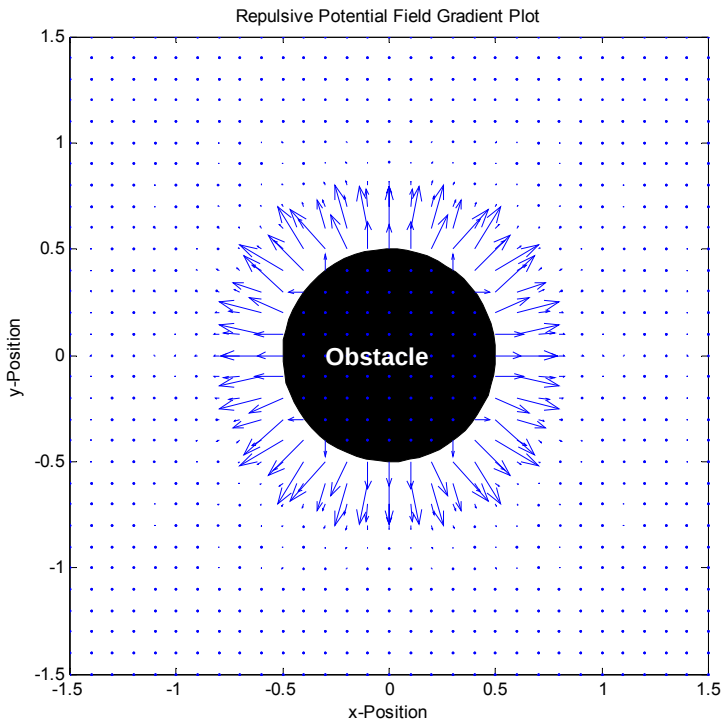



**Figure 3-4: Gradient plot of a general repulsive potential field, where all the gradients pointed away from the obstacle.**

A repulsive potential function that is useful for modeling obstacles in the workspace should posses following attributes [26, 30, 35]:

*1) The potential should have spherical symmetry for large distance to avoid the creation of local minima when this potential is added to an attractive well;*

*2) The potential contours near the surface should follow the surface contour so that large portions of the workspace are not effectively eliminated;*

*3) The potential of an obstacle should have a limited range of influence; and*

*4) The potential and the gradient of the potential must be continuous.*

Different types of repulsive potential were proposed in the literatures. Here, we would like to discuss some of the most widely use repulsive potential functions.

#### 3.1.2.1 FIRAS function

A repulsive potential that have spherical symmetry (i.e. modeled the obstacle as a circular disc) met all of the requirements mention above. However, a spherically symmetric repulsive potential does not follow the contour of polyhedral objects. Hence, an object that has a long shape (i.e. a rectangle) when modeled as circular disc eliminates much more volume from the workspace than it is necessary or desirable. Repulsive potentials that follow the object shape were proposed to address the insufficiency of radically symmetric potentials. The *Force Involving and Artificial Repulsion from the Surface Function* (or FIRAS Function, in French), as proposed by Khatib [30], is one of the example of such function. The potential of the FIRAS function is described by:

$$U_{Rep}(\mathbf{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_{RO}} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho_{RO} \le \rho_0 \\ 0, & \text{if } \rho_{RO} > \rho_0 \end{cases} \tag{3.6}$$

where $\eta$ is a positive scaling factor, $\rho_{RO} = \|\mathbf{q}_{Obs} - \mathbf{q}_{Rob}\|$ is the shortest Euclidean distance between the robot from the obstacle surface, and $\rho_0$ is the limit distance of the repulsive potential field influence. To visualize such potential, a potential created using Equation(3.6) are shown in Figure 3-5. In this plot, the workspace is within $-2 < x < 2$ and $-2 < y < 2$, and the size of the rectangle obstacle is $-0.5 < x < 0.5$ and $-0.5 < y < 0.5$; with $\eta = 2$ and $\rho_0 = 1.2$. The flat surface on top of the potential field is the result of truncated value, which is set to 10. To better understanding this potential, the mesh plot and contour plot of the same potential field are also given in Figure 3-6.

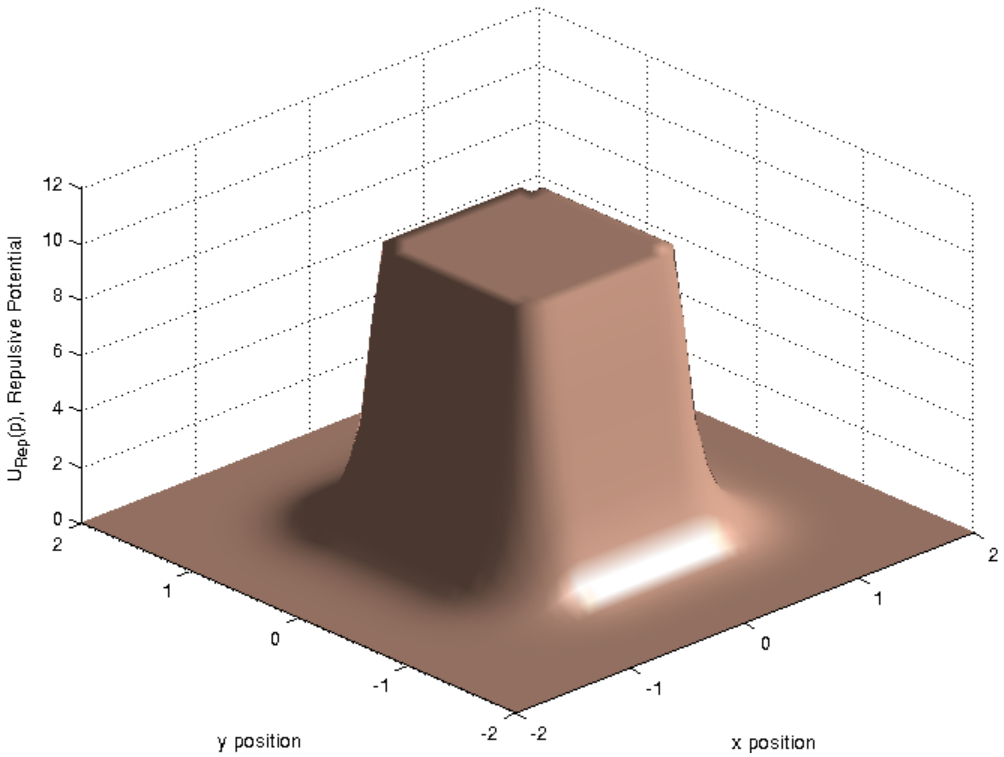
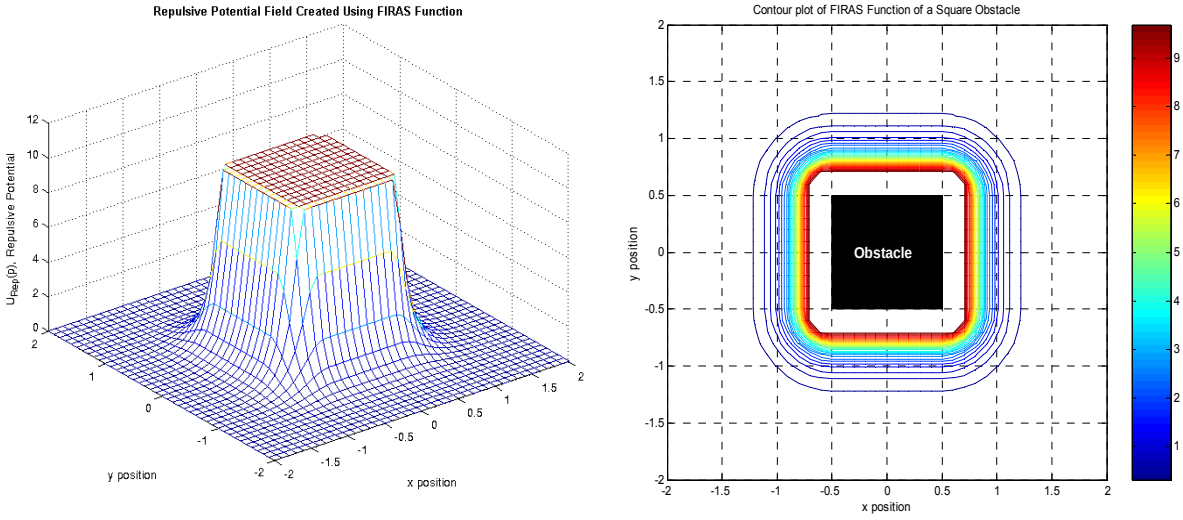



**Figure 3-5: 3D surface plot of a FIRAS repulsive potential result from a square obstacle. Large values of the potential are truncated and result a flat surface on top of the potential.**

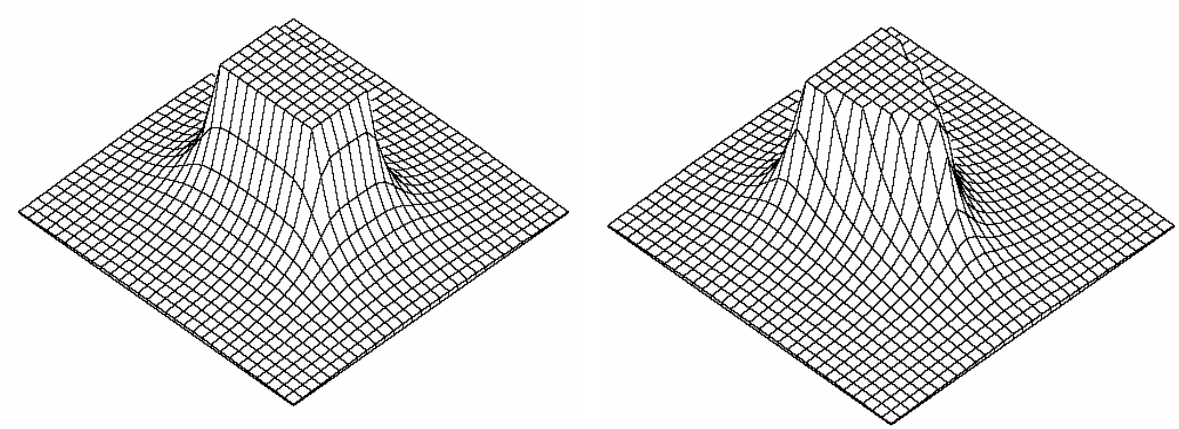



(a) (b)

**Figure 3-6: (a) A mesh plot of repulsive potential field of a square-shaped obstacle created using FIRAS function, and (b) Contour plot of such potential. In both plots, large values are truncated.**

#### 3.1.2.2 Superquadric Potential

The superquadric repulsive potential includes two potential fields, which named avoidance potential and approach potential [44, 45]. The avoidance potential has potential energy at the surface larger than the kinetic energy of the robot, thus prevent it from colliding obstacle. On the other hand, the approach potential reduced the kinetic energy of the robot as the robot approach the obstacle, thus slow down the robot as it approach obstacle. Superquadric potential avoid local minima result from potential function like FIRAS function, where a sharp and flat contour exist for shape such as rectangle. We will show this situation in more detailed in section 3.2.1.1. In general, superquadric potential solve the problem of local minima that result from single obstacle by create symmetric contour around the obstacles. However, it can not solve the problem of local minima resulted from multiple obstacles, such as when obstacles arranged in a "U" shape. Figure 3-7 shows two examples of avoidance potentials created using superquadric potential function: one with triangular shape and one with rectangular shape.



(a) (b)

**Figure 3-7: Example of avoidance potential field created using superquadric potential function. (a) Avoidance potential for obstacle with rectangular shape; and (b) Avoidance potential for obstacle with triangular shape. Reproduced from [35].**

Clearly, the superquadric potential function can only guaranteed global minima with single obstacle. For more than one obstacle, the potential field generated using superquadric potential function cannot guarantee global minima and thus is not a suitable candidate for our situation.

#### 3.1.2.3 Harmonic Potential NO !

Proposed by Kim and Khosla [32, 39], a harmonic potential is the solution to the Laplace Equation of the following form:

$$\nabla^2 \phi = 0 \tag{3.7}$$

where $\nabla^2 \equiv \mathrm{V} \cdot \mathrm{V}$ is the *Laplacian* operator; V denote the velocity field; and $\phi$ is a scalar velocity potential. The general *n*-dimensional expression of Equation (3.7) can be written in the polar coordinates as:

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial r} + \frac{n-1}{r} + angular\ terms. \tag{3.8}$$

We then assumed that the harmonic function $\phi$ is a function of $r$ only, i.e. $\phi = \phi(t)$. After re-arranging and integrating with respect to $r$, Equation (3.8) becomes:

$$\frac{\partial \phi}{\partial r} = \frac{c}{r^{n-1}} \tag{3.9}$$

where $c$ is a scalar. For $n = 2$, i.e. the 2-dimensional space, the solution of Equation (3.9) is:

$$\phi = c \log r + c_1 \tag{3.10}$$

for $n > 2$, the solution is:

$$\phi = \frac{c \big/ (2-n)}{r^{n-2}} + c_1 \tag{3.11}$$

where $r$ is the Euclidean distance from the origin. From Equation (3.10) and Equation (3.11) we observed that harmonic function which symmetry has its singularity at the origin, i.e. where $r = 0$. However, since the origin can be placed anywhere (the Laplace equation is invariant under translation), we can always choose the origin outside the free space for a manipulator and a mobile robot. That is, by locating the origins of the harmonic functions on the surface of obstacles or inside obstacles, we can build an artificial potential field of a 2-dimensional workspace based on Equation (3.10).

In hydrodynamic, the above harmonic function with spherical symmetry is called *source* or *sink*, depending on the sign of $c$ in Equation (3.10) and (3.11). A source can be use to model point obstacle, and sink is use to model target point. In 2-dimensional workspace, a source/sink is conveniently modeled using the modified form of Equation (3.10):

$$\phi = \frac{\lambda}{2\pi} \log r \tag{3.12}$$

where $\lambda$ is called the strength of the source $(\lambda < 0)$ or the sink $(\lambda > 0)$. Since both are singular at the origin, they are also called the point of singularities. Beside Equation (3.12), another harmonic potential function called the uniform flow, whose potential varies linearly along the direction of flow, is also useful for creating the total potential of the workspace. In 2-dimensions, when the fluid flows in a direction that makes an angle $\alpha$ with the x-axis, the potential function for this uniform flow is:

$$\phi = -\delta\left(x\cos\alpha + y\sin\alpha\right) \tag{3.13}$$

where $\delta$ denoted the strength of uniform flow. Equation (3.13) is use to derive a more effective potential flow from a starting position to a goal position for the unbounded

environment. Example of a sink, i.e. the attractive potential to model the target using Equation(3.12) with $\lambda = 20$, is shown in Figure 3-8. In this figure, the target point is located at $(0,0)$.

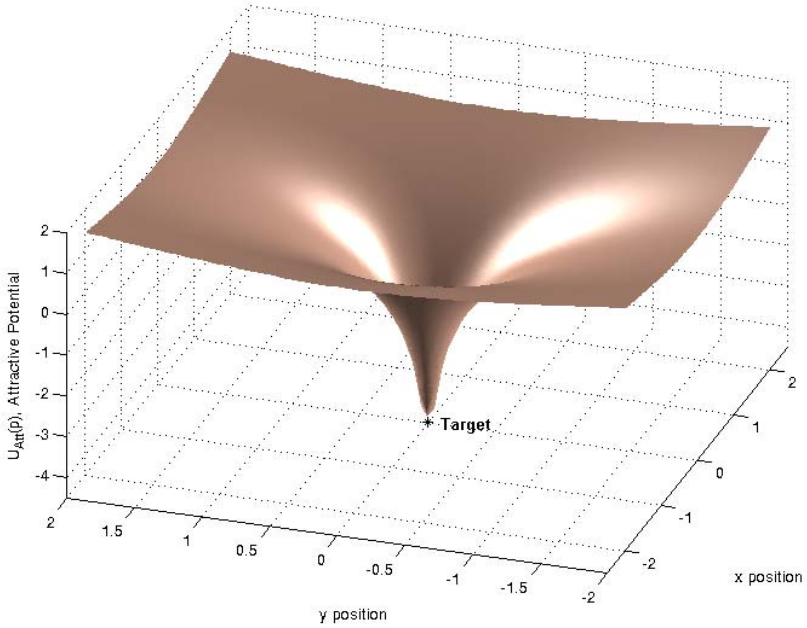



**Figure 3-8: Attractive potential field located at (0,0), created using harmonic potential function. Point at singularity is assigned a value close to its surrounding area.**

In Figure 3-9, a repulsive potential field to model a point obstacle located at $(0,0)$ using the same equation with $\lambda = -20$ is shown. In both potential shown, the singular value at the origin is replaced by a value similar to its surrounding points.

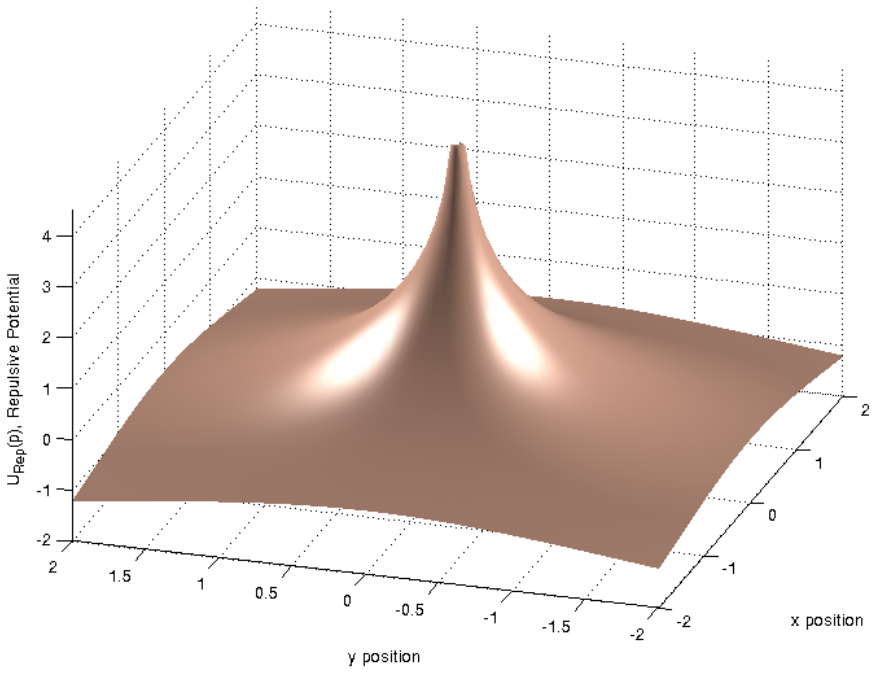



**Figure 3-9: A Repulsive potential field located at (0,0), created using harmonic potential function. Point at singularity is assigned a value close to its surrounding area.**

One important properties of a harmonic function is the principle of superposition, which follows from the linearity of the Laplace equation. That is, if $\phi_1$ and $\phi_2$ are harmonic functions, then any linear combination of $\phi_1$ and $\phi_2$ is also a harmonic function and a solution of the Laplace equation. Other properties include the *mean-value property, the minimum principle, and the maximum principle,* are to unsure the potential field are free of local minima. Harmonic potential function allowed us to model line obstacle (such as wall) using *single panel method*, and model obstacle with various shapes using *multiple panel method*. However, in this case, the process of modeling the configuration space become very complex and losses its simplicity form of Equation (3.12). For detailed description, see [32, 39].

Harmonic potential function is gaining its popularity among researchers worked on potential field approach for robot motion planning [37, 38, 48-51, 69-73]. However, a

harmonic function will create a situation called the "structural local minimum" [32, 39]. A structural local minimum is defined as: *a structural minimum occurs when there exists at least one possible trajectory and the robot is in static equilibrium with repulsive forces from obstacles and attractive force from the goal sink* [39]. This is the one type of local minimum that results from multiple obstacles that we discussed in section 3.1.2.3. Further, the simplest form if this potential can only model point obstacles, which is not a desired feature for our purpose. A potential field of a workspace generated using harmonic potential function is given in Figure 3-14.

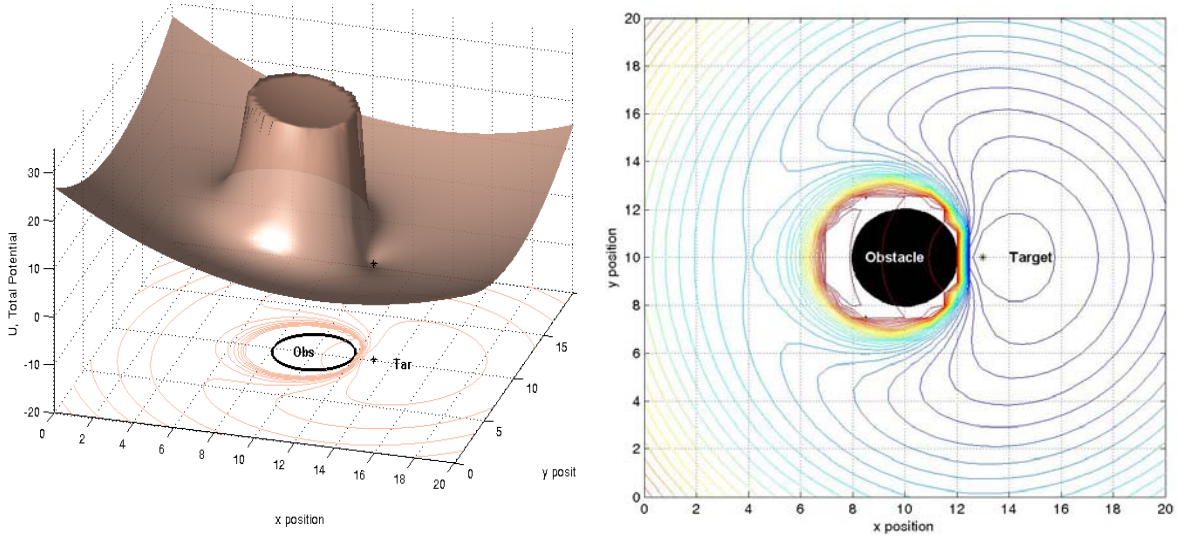
#### 3.1.2.4 Ge New Potential

Modified from the FIRAS function, Ge and Cui proposed a new potential function [33, 36] intended to solve the GNRON (*Goal Non-Reachable with Obstacles Nearby*) problem found in FIRAS function. This function takes the following form:

$$U_{Rep}(\mathbf{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_{RO}} - \frac{1}{\rho_0}\right)^2 \rho_{RT}^n, & \text{if } \rho_{RO} \le \rho_0 \\ 0, & \text{if } \rho_{RO} > \rho_0 \end{cases} \tag{3.14}$$

where $\eta$ is a positive scaling factor, $\rho_{RO}$ denotes the minimal Euclidean distance from the robot to the obstacle, $\rho_{RT}$ denotes the minimal Euclidean distance from robot to the target, and $\rho_0$ is a positive constant that defined the influence range of the obstacle. Compare Equation (3.14) with the FIRAS function given in Equation (3.6), the introduction of the term $\rho_{RT}^n$ ensures that the total potential will reach its global minimum, if and only if the robot reaches the target where $\rho_{RT} = 0$. Figure 3-10 shows

the potential field generated using Equation (3.14) and Equation (3.2) with $\eta = 2$, $\xi = 0.2$, and $m = n = 2$.




**(a)** **(b)**

**Figure 3-10: Potential field generated using Ge new potential of a workspace where the target is placed close to the obstacle. (a) 3D view and (b) the respective contour plot of the potential field. Large value of the potential was truncated.**

In Figure 3-10, the target is located at $(13, 13)$, and the obstacle is located at $(10, 10)$ with a radius of $2$. The influence range of the repulsive is $5$ times the radius of the obstacle. We can see that even the target is located inside the influence range of the repulsive potential, the global minima is still located at the target position.

### 3.1.2.5 Repulsive Potential Functions with Velocities Information

Ge *et al*, extend the potential function to solve the motion planning of a mobile robot where the robot's target and obstacles are moving [6]. The idea is to generate a potential field that takes both velocities and position information of the robot, obstacles, and target. Similar type of potential field that include velocities information is the GPF (*Generalized Potential Field*) proposed by Krogh [29] in the early 80's. While the GPF

function only takes the velocity information of the robot into account, the potential function proposed by Ge includes the velocities information of the robot, obstacles, and the target. As a result, this potential field is able to provide motion planning in a *dynamic* workspace where the obstacles and the target are both moving. Detailed formulation of this potential function can be found in [6]. Other potential functions that include the velocities information were also well summarized in [6]. In this thesis, we did not implement this type of potential function for collision avoidance. However, this type of potential function is possible to implement with the motion planning methods for group of robots used in this thesis.

### 3.1.3 Total Potential Field

The total potential field is obtained by adding the repulsive potential resulting from all obstacles in the workspace and the attractive potential from the target. As given in Equation (3.1):

$$U_{Total}(\mathbf{q}) = U_{Att}(\mathbf{q}) + U_{Rep}(\mathbf{q})$$

This idea is illustrated in Figure 3-11. In Figure 3-11, the obstacle is located at $(-1.0,\ 1.0)$, with a radius of 0.5. The target point is located at $(1.0,\ -1.0)$. The arrows pointed toward the negative of the gradient of the total potential field. From the figure, we can see that almost all the points in the workspace pointed towards the target, except at the surrounding area of the obstacle. The steepest gradient (indicated by larger arrows) occurred at the surface of the obstacle, where the potential goes to infinity.

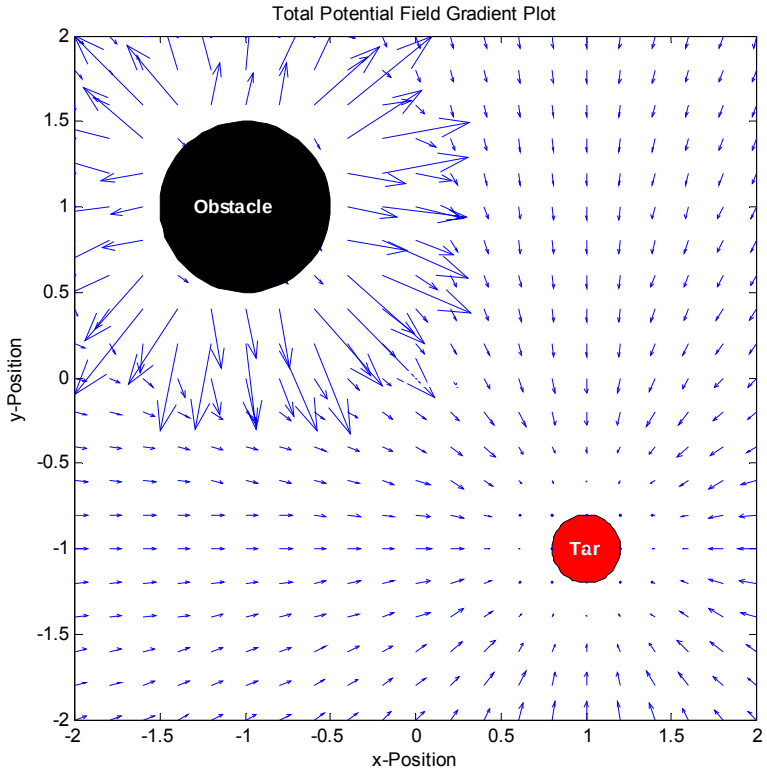
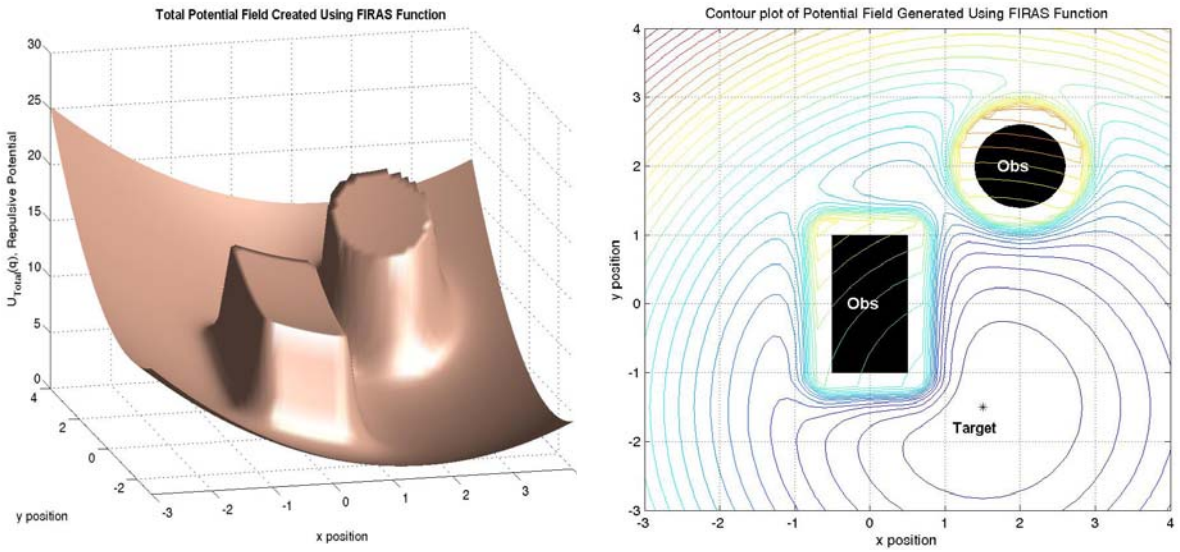



**Figure 3-11: Negative gradient of the total potential field pointed towards the target almost every point in the workspace, except at the surrounding of the obstacle, where the potential approaches infinity at the obstacle wall.**

In the following, three total potential fields are given as examples of total potential field generated by adding repulsive potential and attractive potential. Figure 3-12 shows the total potential field generated using FIRAS function; Figure 3-13 shows the total potential generated using Ge's new potential function; and Figure 3-14 shows the total potential generated using Harmonic potential function.

In Figure 3-12, the obstacles are: rectangular obstacle with 2 units in height and 1 unit in width located at $(0,\ 0)$; and a circular obstacle located at $(2,\ 2)$ with a radius of $0.5$. The potential field is generated using Equation (3.6), with $\eta = 2$ and $\rho_0 = 3\rho$. The desired location is $(1.5,\ -1.5)$.




**(a)** **(b)**

**Figure 3-12: Total potential field generated using FIRAS Function, (a) 3D plot of the potential field; and (b) Contour plot of the potential field. Large value of the potential is truncated.**

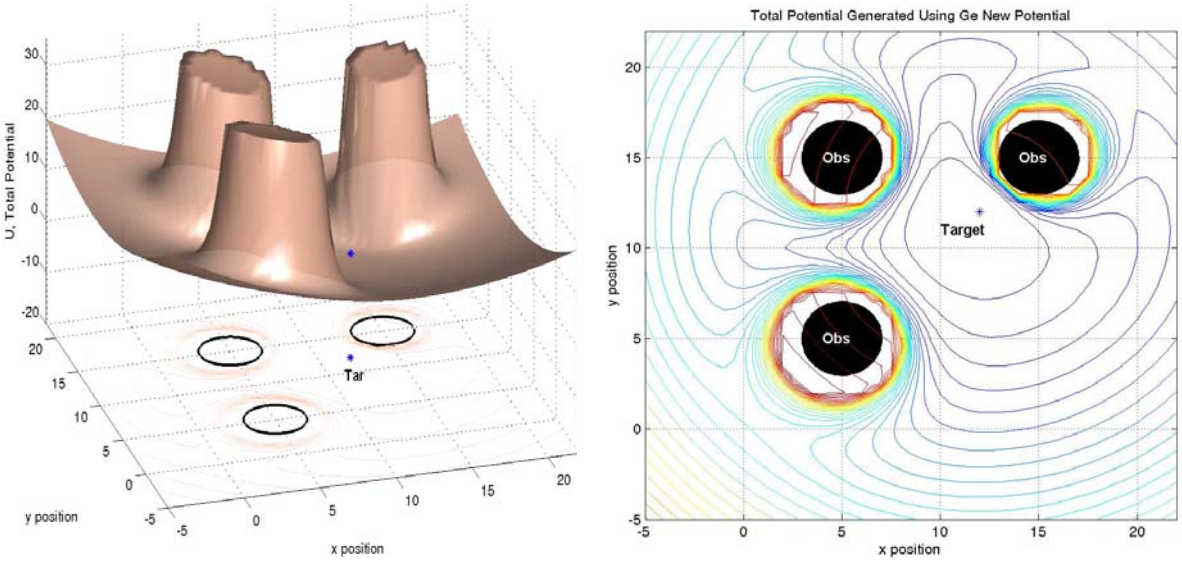
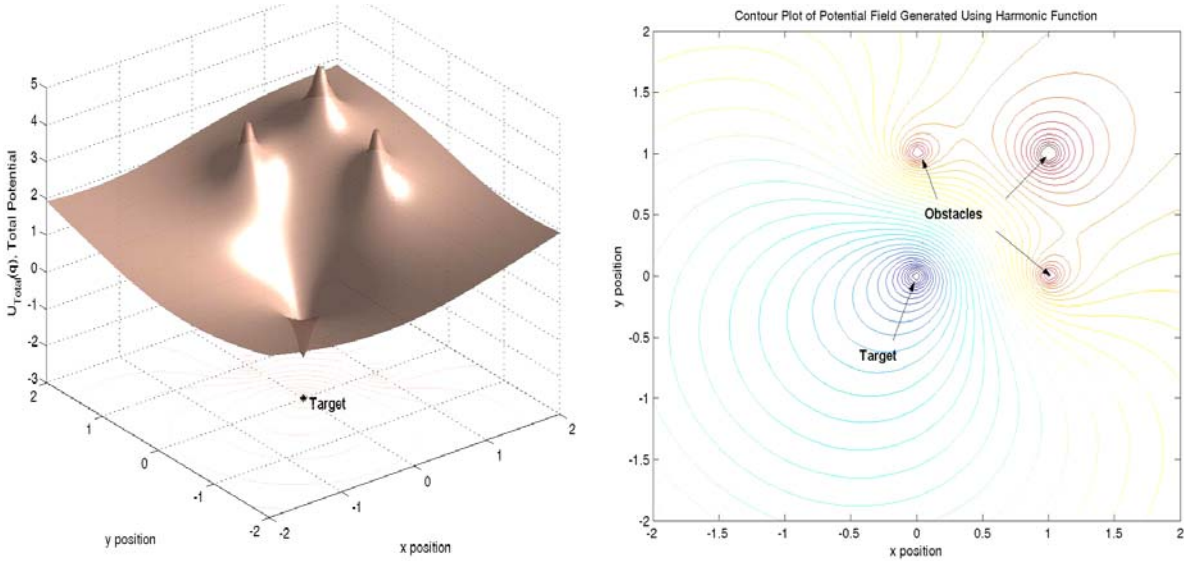



**Figure 3-13: Total potential field generated using Ge's New Potential Function, (a) 3D plot of the potential field; and (b) Contour plot of the potential field.**

In Figure 3-13, three circular obstacles with radius $2$ are each located at $(15,\ 15)$, $(5,\ 15)$, and $(5,\ 5)$. The target, is located at $(12,\ 12)$. This potential field is generated

using Equation (3.14) and Equation (3.2) with $\eta = 1$, $\xi = 0.1$, $m = n = 2$, and $\rho_0$ is five times the obstacle's radius.




**(a)** **(b)**

**Figure 3-14: Total potential field generated using Harmonic Potential Function, (a) 3D plot of the potential field; and (b) Contour plot of the potential field.**

In Figure 3-14, the location of the point obstacles are $(1, 0)$, $(0, 1)$, and $(1, 1)$, respectively. The target is located at $(0, 0)$. This potential field is generated by adding repulsive and attractive potential using Equation (3.12) with $\lambda = 10$ and the uniform flow potential using Equation (3.13) with $\delta = 1$.

Ideally, adding these two types of potentials (attractive and repulsive) will yield a resulting total potential such that, in any point of the workspace, the negative gradient flow of the potential field points toward the target. It is then intuitive to let the robot in the potential field to follow the negative gradient of the potential field, and eventually converge to the target. This idea is simple and elegant. However, several limitations do exist with this idea. These limitations were addressed in the next section.

## 3.2 Limitations with Potential Field Method

Potential field method is attractive because of its mathematical elegance and simplicity. However, several limitations inherent in potential field method had been addressed systematically based on mathematical analysis done by Koran and Borenstein in [74]. Some of these limitations were also well documented in [6, 32-37, 43]. Here, we discussed the known limitations, and some possible solutions to these limitations.

### 3.2.1 Trap Situation Due to Local Minima.

Potential field created from some potential functions can have local minima and thus create a 'trapped situation' for the mobile robot. In fact, this is the best-known and most cited problem with potential field method. Local minima can cause by either one obstacle or combination of obstacles. We examined these two different situations in this section.

#### 3.2.1.1 Local Minima Result From Single Obstacle

Recall some of the properties that are essential for a repulsive potential function that we had discussed in Section 3.1.1, the second attribute required that the potential contours near the surface should follow the surface contour so that large portions of the workspace are not effectively eliminated. If, let say the first attribute is not satisfied at the same time, a local minima will occurs even with one obstacle. This situation is best explained with the help of Figure 3-15.

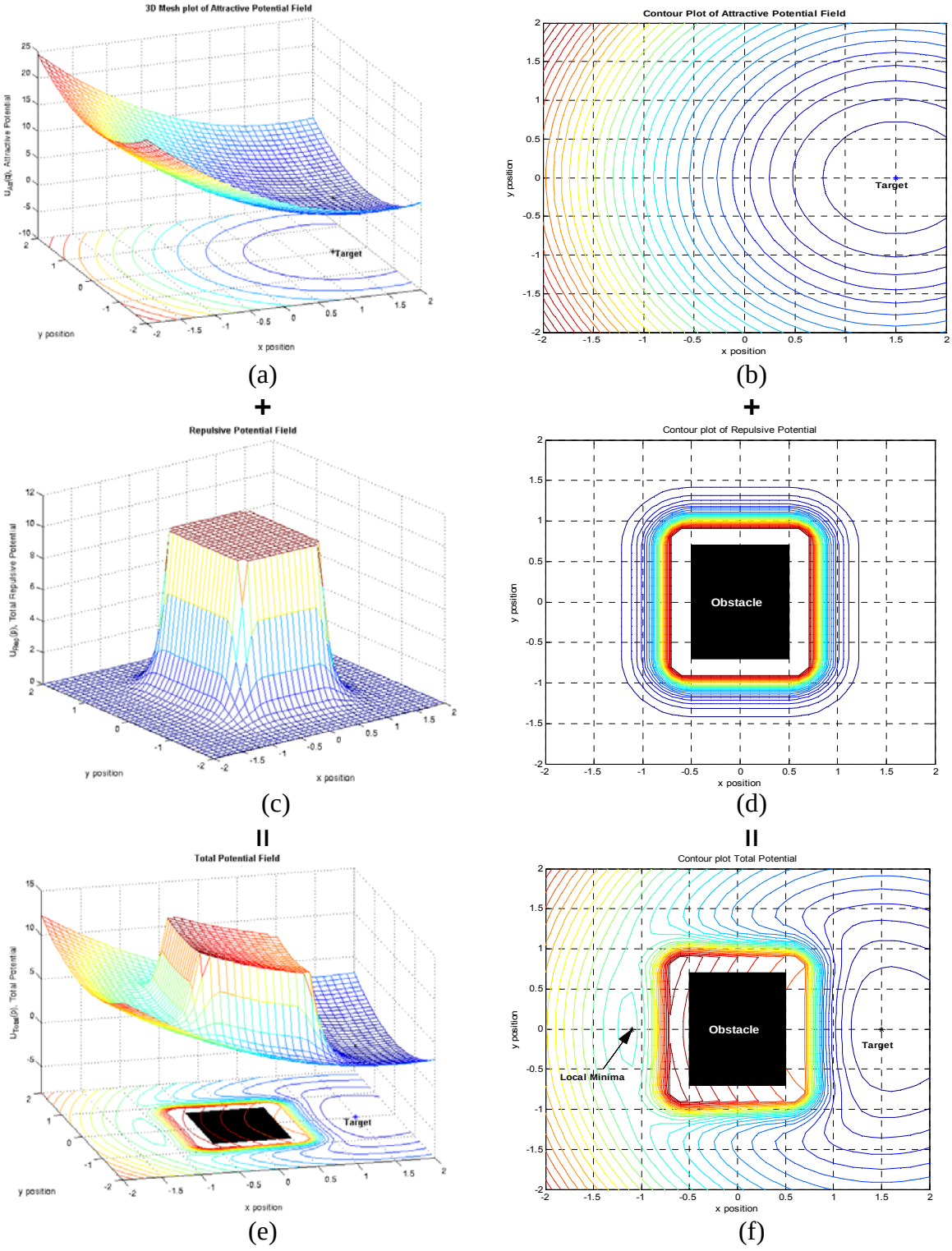



**Figure 3-15: (a) 3D View of attractive potential field with one local minima at the target position (1.5, 0.0); (b) Contour plot of the attractive potential; (c) 3D View of repulsive potential field and (d) its contour plot; (e) The total potential in the workspace after adding both attractive and repulsive potential; and (f) The contour plot showing a local minimum occurred at around (-1.1, 0.0).**

In Figure 3-15 (a), a attractive potential field created using Equation(3.2) with target position located at $(1.5, 0)$. This attractive potential is then added with repulsive potential field shown in Figure 3-15 (b), created to model a rectangle obstacle with size $-0.5 < x < 0.5$ and $-0.7 < y < 0.7$ using Equation(3.6). Their resulting total potential field has the shape of Figure 3-15 (e). In Figure 3-15(f), we can see that a local minimum created at the left side of the obstacle at position $(-1.1,\ 0.0)$. In this case, if any robot initial position located at the left side of the obstacle, it may trap in the local minima and never reach the desired target position.

This trap situation happened due to the nature of the shape of the repulsive potential function. When this kind of potential added to the attractive potential with the shape of a quadratic well, local minima will be created. This limitation follow by the nature of FIRAS function can be overcome by potential function such as Harmonic Potential Function, Superquadric Potential Function, or Navigation Function.

#### 3.2.1.2 Local Minima Result From Multiple Obstacles

A trap situation can also be created by various obstacles configurations, and different type of traps can be distinguished. An example of this is a situation where two obstacles are closely spaced, as seem in Figure 3-16, can create a local minima in between the two obstacles. In this figure, the attractive potential is created using Equation(3.2); and the repulsive potential is created using Equation(3.6). As seen in Figure 3-16, if the initial position of the robot is close to the local minima, it is possible that the robot be trapped inside the local minima.

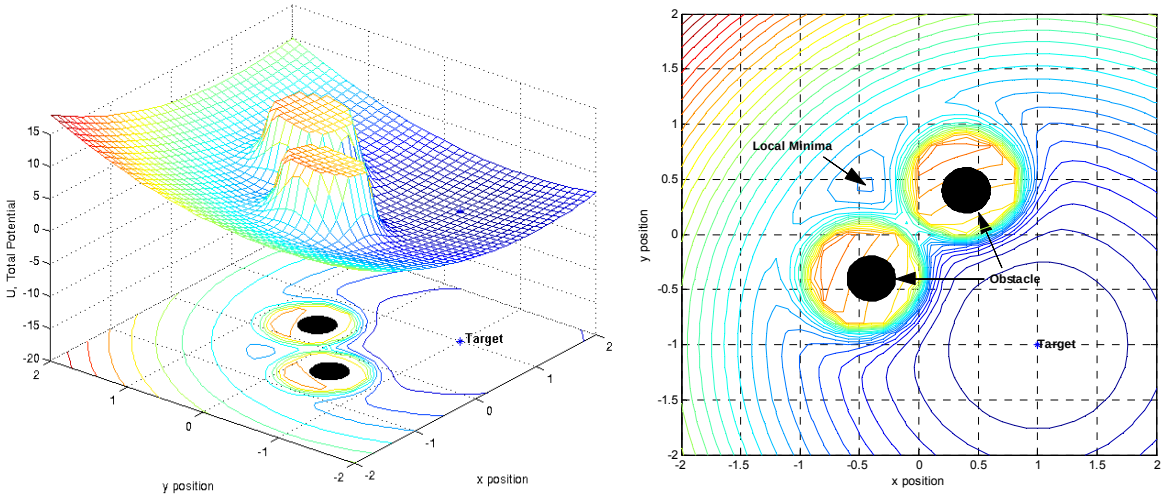



**Figure 3-16: (a) Potential Field showing two closely space obstacles create an local minima and (b) Contour plot of the Potential Field showing a local minimum occurred at (-0.45, 0.45). The flat surface shown on top of the two repulsive potential fields are result of truncated potential value.**

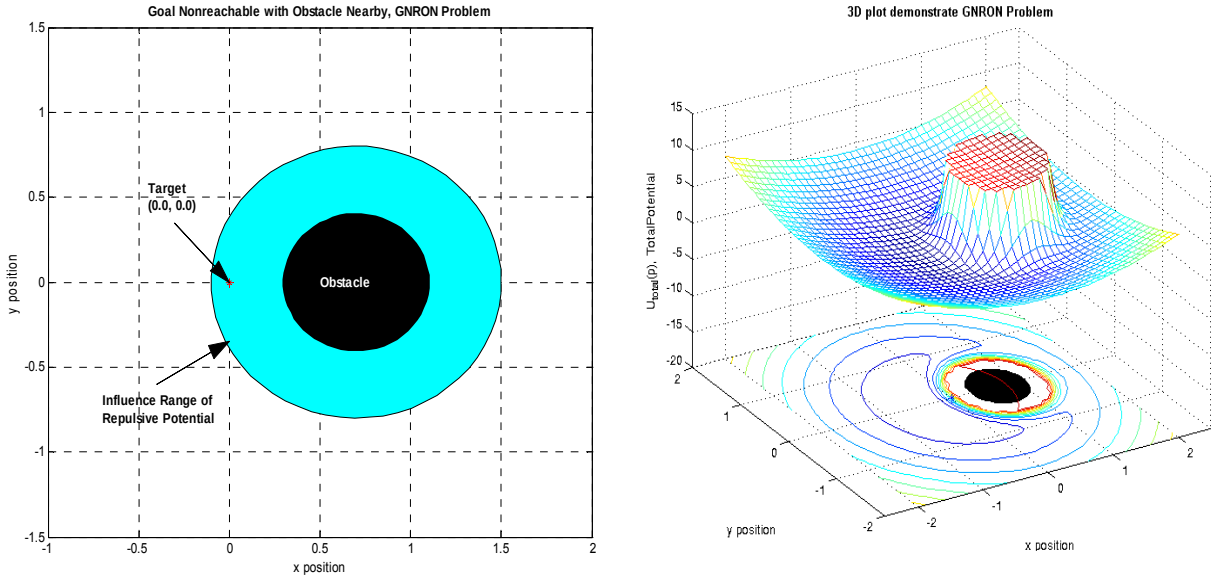
This trapped situation can be resolved by heuristic or global recovery methods [74]. Although superquadric potential solved the problem of local minima, it still could not solve the local minimum problem that result from several obstacles [35]. This problem is common for all local potential functions. Global approaches such as navigation function, on the other hand, do not posses this limitation.

### 3.2.2 Target close to obstacle

When a target is placed very close to the obstacle, that is, within the influence range of the obstacle, the global minimum point may not at the target position. This scenario, different from local minima problem, is that the global minimum is being 'push' away from where it supposed to be (i.e. the target), to another position. As a result, the robot cannot converge to the correct point, which is the target. This situation, as

mentioned before as the *Goals Non-Reachable with Obstacle Nearby*, or the GNRON problem, is well documented in [6, 33, 36, 43].

An example demonstrating this limitation is given in Figure 3-17. In Figure 3-17(a), the target is placed close to the obstacle and within the influence range of the repulsive potential field from the obstacle (Consider the scenario where the target of the mobile robot is the water cooler that is usually attached to the wall). A 3-D view of the total potential field is shown in Figure 3-17(b).




(a) (b)

**Figure 3-17: (a) Target is placed close to the obstacle and within the influence range of the repulsive potential field; and (b) A 3D view of the total potential field created.**

In Figure 3-17(b), the obstacle located in $(0.7,\ 0.0)$ with a radius of $R = 0.4$. The repulsive potential is created using the FIRAS function given by Equation(3.6) with $\eta = 2.5$ and $\rho_0 = 2 \times R$. On the other hand, the attractive potential is created using Equation(3.2) with $m = 2$ and within a workspace of $-2 < x < 2$ and $-2 < y < 2$. From Figure 3-17, it is difficult to demonstrate if the global minimum point had been 'pushed'

away by the repulsive potential field to another point. We shall see this situation more clearly in Figure 3-18, the contour plot of the total potential shown in Figure 3-17(b).

As a result of the strong repulsive potential from the obstacle, the global minimum point was shifted from $(0,0)$ to $(-0.4,\ 0.0)$.

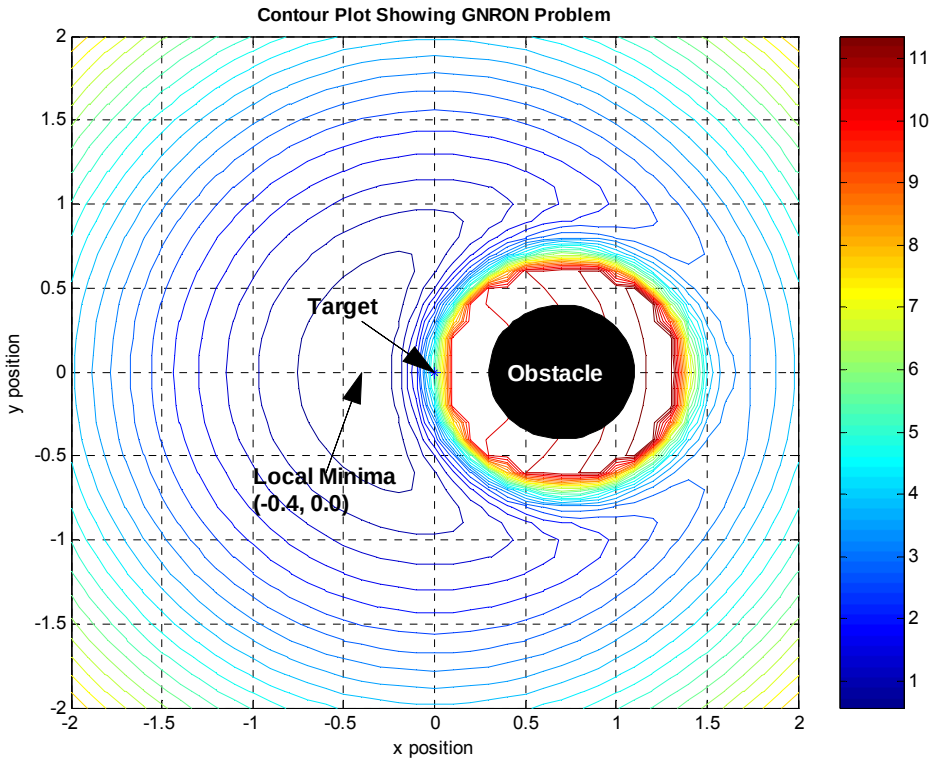



**Figure 3-18: Contour plot demonstrating the GNRON problem with FIRAS Function. The global minimum point has shifted to (-0.4, 0.0) from (0.0, 0.0) resulted from the strong influence of nearby repulsive potential field.**

Potential functions that posses this limitations are the FIRAS Functions and superquadric potential functions. Ge and Cui's new potential function are designed to solve this type of problem [33, 36]. On the other hand, navigation functions and harmonic potential functions do not have this limitation.

### 3.2.3 No Passage between Closely Spaced Obstacles.

Koren and Borenstein [74] demonstrated that there is a possibility that the robot cannot pass through two closely spaced obstacles as a result from the repulsive forces create by the two obstacles if the attractive force is not large enough to overcome this repulsive force. This situation can happen when the mobile robot needs to pass through a door, and the target's location does not allow it to generate sufficient attractive force.

### 3.2.4 Non-Optima Paths

This limitation is primarily due to the nature of the potential field method. Since we does not explicitly specify the path to be travel by the robot, the path that traveled by the robot is often not an optima path [7]. An immediate result of this drawback is it shortens the battery life of a mobile robot.

### 3.2.5 Implementation Related Limitations

Two implementation-related issues are presence of oscillations and limitation on actuator's torque. Oscillations are observed during the implementation of potential based approach with actual robot while torque limitation is contradiction to the requirement of 'infinite' torque on obstacle surface.

#### 3.2.5.1 Oscillations in the Presence of Obstacles

The limitation of oscillation motion happened when a disturbance occurs in the workspace. This become apparent only when the potential field method is implemented in a high-speed real-time system [74]. This limitations, is not obvious until actual implementation of the potential field method with actual mobile robot. However, at low speed, this limitation disappeared.

#### 3.2.5.2 Oscillations in Narrow Passages

Similar to previous limitation, oscillation motion is observed when the robot traveled in narrow passage with high speed. This is because the robot experiences the repulsive forces from both side of the wall. A detailed mathematical prove of this limitation is given in [74].

#### 3.2.5.3 Infinite torque is not possible

Potential function approaches has the following seemingly contradictory requirement: in order to prevent collision with obstacles, it seems that the obstacles should exert an infinitely large repulsive potential as the robot approaches their boundary. But at the same time, there is a practical bound on the robot's actuators impose a limit on the allowable repulsive potential. A desirable repulsive potential requires that the potential function be uniformly maximal over the boundary of configuration space, that is, maximal at the workspace and obstacle boundary [52, 54, 75].

# 4 Global Potential Field Approach - The Navigation Function

In Chapter 3, we discussed various forms of local potential functions, their characteristics, and the general limitations for motion planning. Many of these limitations can be overcome using the potential function introduced in this chapter: the *navigation function*. The navigation function is considered as a 'global' strategy: to construct such potential, we need to know the complete information of the configuration space. Thus, we can construct a configuration space that is free of local minima at the cost of losing the simplicity computational advantage of the original local potential field approach. The potential field created using navigation function provides us an ideal platform to develop and test our motion planning algorithms for robots collectives in chapter 6. Hence, although we realize that this global approach may not be suitable for real-time application, we will use this navigation function to generate the potential field of a given workspace that served as a test-bed for our motion planning algorithm.

Rimon and Koditschek introduced the navigation function and discussed its various characteristics in their series of papers [34, 53, 55-58, 76]. The navigation function, once constructed, can solve the problem of local minima. Further, through proper diffeomorphisms, obstacle with various kinds of shapes can be mapped into circular shape.

In the remainder of this chapter, we first discussed the properties that constituted a navigation function, followed by introduced a number of simple functions that will be used to construct navigation functions. Finally, shows the steps to construct a potential field of a sphere world using navigation function.

## 4.1 Properties of Navigation Function

A Navigation Function is defined as follow:

*Definition 1: Let* $\Psi$ *be a robot free configuration space, and let* $\mathbf{q}_{Tar}$ *be a goal point in the interior of* $\Psi$ *, A map* $\varphi : \Psi \to [0,1]$ *is a Navigation Function if it is:*

1) *Smooth on* $\Psi$ *, that is, at least a* $\mathbb{C}^2$ *function.*
2) *Polar at* $\mathbf{q}_{Tar}$ *,i.e., has a unique minimum at* $\mathbf{q}_{Tar}$ *on the path-connected component of* $\Psi$ *containing* $\mathbf{q}_{Tar}$ *.*
3) *Admissible on* $\Psi$ *, i.e., uniformly maximal on the boundary of* $\Psi$ *.*
4) *A Morse Function.*

In general, the first term ensure that $\nabla\varphi$ be continuously differentiable and consequently that the classical existence and uniqueness results of solutions for the closed-loop robot system apply. It also ensured that $\varphi$ to be effectively computable.

The second term shows the beauty of the navigation function, i.e., a unique minimum in the workspace. This can be achieved if the fourth term is satisfied. The equilibrium states of a close-loop system coincide with the critical points of the potential function and that they are non-degenerate if and only if the Hessian (The second derivative of a matrix) of the potential function evaluated at this critical points is

nonsingular. In standard mathematical convention, such function is referred as a *Morse Function*.

On the other hand, although a navigation function provide a workspace with a global minimum, it does not guarantee an "essential" global convergence. A global convergence means convergence from almost all initial configurations. In fact, it was shown that there exist at least as many saddles points as there are internal obstacles [34]. However, in practical, these spurious unstable equilibrium point need not cause any practical difficulties since only "few" initial configuration will get stuck on them. Further, for a group of robots with formation constraints, the chances of getting stuck on these saddle points is further decreased: if one robot stuck on the saddle point, other robots in the same formation will drives that robot out of the saddle point because of formation constraints.

The third term in the definition ensured that $\varphi$ is *admissible*. By definition, a real-valued function on the free configuration space is said to be admissible if it is uniformly maximal on the boundary of $\Psi$, i.e., where the robot touches an obstacle. That is:

$$V(\mathbf{q}) = \begin{cases} = c, & \textit{for all } \mathbf{q} \in \text{ boundary } (\Psi) \\ < c, & \textit{for all } \mathbf{q} \in \text{ interior } (\Psi) \end{cases} \tag{4.1}$$

This guarantee that trajectories starting in the interior of $\Psi$ stay inside an explicitly specified subset of $\Psi$, thus providing for a safety clearance from the obstacles. It is shown that a navigation function always exists in the robot free configuration space: For every smooth connected and compact manifold with boundary, $M$, and any interior point, $\mathbf{q}_0 \in M$, there exists a smooth navigation function with a unique minimum at $q_0$. Ideally, one would like to obtain a closed-form formula in terms of the geometric data

and $\mathbf{q}_d$ for a navigation function on a completely general free configuration space $\Psi$. However, proving that a navigation function exist is not constructive, and there remains the task of actually construct them.

Specifically, *Definition 1* implies that two navigation problems are actually the same if there exist a coordinate transformation that mapped one to the other. In other words, this implies that the navigation properties are invariant under transformation of both the domain and the range spaces. This property is important since it implies that the navigation function not only able to model generalized sphere world but also enlarge the scope to more complicated obstacle shapes, if a *diffeomorphism* exist between the two spaces. Formally, a map between two spaces that is *smooth*, *one-to-one* and *onto*, and has a smooth inverse is called a diffeomorphism.

## 4.2 Distance-to-the-Target Function

At each position of the workspace, we define the distance between the robot and the target/goal as Distance-to-the-Target Function. In Euclidean space, this function is given by:

$$\gamma_\kappa(\mathbf{q}) = \left\|\mathbf{q} - \mathbf{q}_{Tar}\right\|^{2\kappa} \tag{4.2}$$

where $\mathbf{q}$ denoted the position of the robot, $\mathbf{q}_{Tar}$ denoted the position of the target, and $\left\|\mathbf{q} - \mathbf{q}_{Tar}\right\|$ is the Euclidean norm and $\kappa > 0$ is a control parameter.

## 4.3 Obstacles Modeling

In constructing the obstacle model, a disc-shape obstacle modeling is the most widely used one. But in a real world, obstacle varies in shapes. To accurately model the

obstacles in a workspace, it is possible to model obstacle in variety of shapes, and through proper diffeomorphism, mapped the shapes in to a circular disc-shape. Here, instead of defining obstacle function with one specific modeling function, we generalized the Obstacle Function as a real-valued map $\beta_i$ representing it in the form:

$$Obstacle_i = \{\mathbf{q} : \beta_i \leq 0\} \tag{4.3}$$

where $\beta_i$ is the implicit form of obstacle $i$ model's geometric equation. The subscript $i = 0$ represents the implicit representation of the bounded workspace, and $i = 1,\ 2, \cdots, M$ for $M$ number of obstacle.

The product of all the obstacle functions, which will be useful in the construction of navigation function, is given by:

$$\beta = \prod_{i=0}^{M} \beta_i \tag{4.4}$$

In Equation (4.4), if we assumed all the obstacles in the workspace does not intersecting each other, $\beta < 0$ if a point is inside an obstacle, or outside the workspace; and $\beta \geq 0$ if a point is outside the obstacle or on the obstacle's surface, and within the workspace.

## 4.4 Conditioning Functions

In this section, we define two real-valued functions of a single real argument will be used constantly in the construction of navigation function. These two function serve for the purpose of "conditioning" their argument without changing certain essential properties. We shall use the notation $f_1 \circ f_2 = f_1\left(f_2\left(x\right)\right)$ to denote function decomposition.

### 4.4.1 Analytic Switch Function

A diffeomorphism function that mapped the extended reals $[0, \infty]$ onto the unit interval $[0, 1]$ is given by:

$$\sigma_{\lambda}(x) = \frac{x}{\lambda + x} \tag{4.5}$$

Where $\lambda > 0$ is a same control parameter as seen in Equation (4.2). The properties of this function are such that it takes zero to zero, map the "point at infinity" to unity, and vary smoothly in between. The main purpose of this function is to bound functions that achieve their maximal value uniformly on some set of interest by blowing up to $+\infty$; as seen in the previously described potential functions [6, 32, 33, 35, 36, 38].

### 4.4.2 Sharpening Function

The sharpening function is a $\kappa^{th}$ root function such that:

$$\rho_{\kappa}(x) = x^{1/\kappa} \tag{4.6}$$

Equation (4.6) is used to "*sharpen*" its argument. For example, $\gamma_2(\mathbf{q}) = \|\mathbf{q} - \mathbf{q}_{Tar}\|^4$ is not a Morse Function since its Hessian at the origin is the zero matrix. However, Using Equation (4.6): $\rho_2 \circ \gamma_2 = \|\mathbf{q} - \mathbf{q}_{Tar}\|^2$, $\gamma_2(\mathbf{q})$ is a Morse Function.

## 4.5 Navigation Function in a Sphere World

In this section, we demonstrated the construction of a navigation function in a *sphere world*. A sphere world is a compact connected subset of $E^n$ whose boundary is

formed from the disjoint union of a finite number, say $M+1$, of $(n-1)$ spheres [52]. It follows that there is one large sphere which bounds the workspace, $\mathcal{W}$:

$$\mathcal{W} = \left\{ \mathbf{q} \in E^n : \|\mathbf{q}\|^2 - r_0^2 \leq 0 \right\} \tag{4.7}$$

and $M$ number of smaller spheres which bound the obstacles,

$$\mathcal{O} = \left\{ \mathbf{q} \in E^n : \|\mathbf{q} - \mathbf{q}_j\|^2 - r_j^2 \right\}, \quad j = 1 \cdots M \tag{4.8}$$

note that the spheres are represented by listing their $M+1$ positive radii, $\{r_j\}_{j=0}^{M}$, and $M$ center points, $\{\mathbf{q}_i\}_{i=1}^{M}$. For ease of construction, the bounded workspace $E^n - \mathcal{W}$ is referred as the zero[th] obstacle, and is centered at the origin of our coordinate system.

The free configuration space, is the space remains after removing all the obstacle spaces from the workspace:

$$\mathcal{F} = \mathcal{W} - \bigcup_{j=1}^{M} \mathcal{O}_j \tag{4.9}$$

Two constraints must be imposed for $\mathcal{F}$ to be a valid sphere world. The first one is that all the obstacles must contain in the interior of the workspace:

$$r_0 > 0 \quad \text{and} \quad \|\mathbf{q}_i\| + r_i < r_0, \quad \textit{for} \quad 1 \leq i \leq M \tag{4.10}$$

and the second constraint is that none of them intersect each other:

$$\|\mathbf{q}_i - \mathbf{q}_j\| > r_i + r_j, \quad \textit{for} \quad 1 \leq i, j \leq M \tag{4.11}$$

This sphere world constitute as the 'model world' and more complicated obstacles shapes can be mapped into this model world through diffeomorphism since the navigation function is invariant under diffeomorphism.

Considering the Euclidean sphere world (i.e., a sphere world in Euclidean Space), as seen in Figure 4-1.

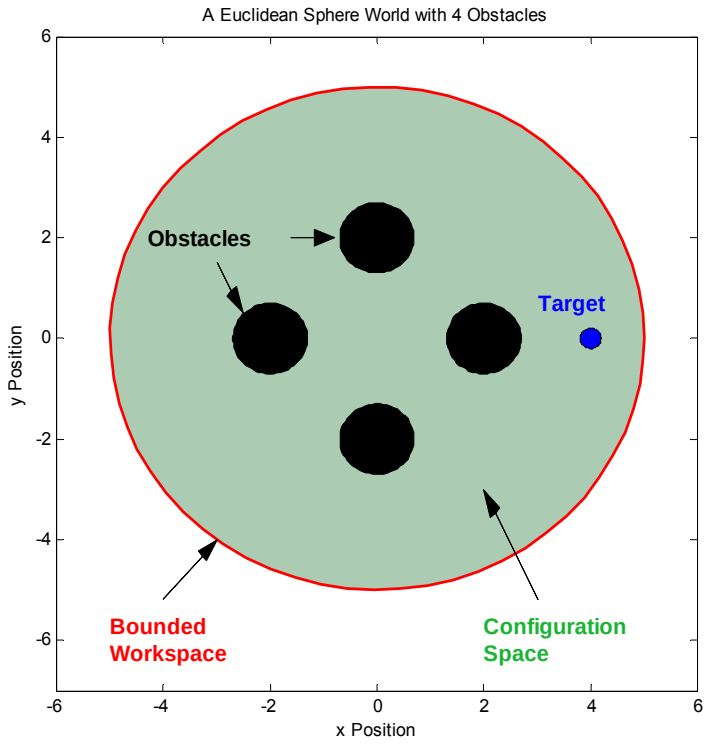



**Figure 4-1. A Euclidean Sphere world with 4 obstacles, showing disc-shaped obstacles, disc-shaped workspace, and free configuration space describe by Equation(4.12).**

Let $A(\mathbf{q}, \rho)$ denote a Euclidean *n*-dimensional disc with center at $\mathbf{q} \in E^n$, and radius $\rho$. A *Euclidean Sphere World* is formed by removing from a large *n*-dimensional disc, $A_0(0, \rho_0)$ (i.e. centered at (0, 0) with radius $\rho_0$), *M*-numbered of disc-like "Hill", $A_j(\mathbf{q}_j, \rho_j)$ for $j = 1 \cdots M$, called the obstacles. Let us define the complement of $A_0$ in $E^n$ as the *zeroth* obstacle. The *free configuration space (or simply, configuration space)*, $\Psi$ that remains after removing all the internal obstacles from $A_0$ is:

$$\Psi = A_0 - \bigcup_{j=1}^{M} Obstacle_j \tag{4.12}$$

For $\Psi$ to be a valid sphere world, the obstacles' closure must be disjoint and be contained in the interior of $A_0$. Thus, for this sphere world, there are $M+1$ centers, $q_i$ and radii $\rho_i$, for $i = 1 \cdots M+1$.

Also, for this example, the Bounding Function is:

$$\beta_0(q) = -\left\|\mathbf{q} - \mathbf{q}_0\right\|^2 + \rho_0^2 \tag{4.13}$$

and spherical obstacle function given by:

$$\beta_j(q) = -\left\|\mathbf{q} - \mathbf{q}_j\right\|^2 + \rho_j^2 \quad , \quad for\ j = 1 \cdots M. \tag{4.14}$$

These formulas are expressed in terms of the implicit representation of the constituent shape, which assumed to be known. We are now ready to construct the navigation function for the sphere world. The Navigation of the sphere world is thus defined as:

$$\varphi_\kappa(\mathbf{q}) = \begin{cases} \left(\rho_\kappa \circ \sigma_1 \circ \dfrac{\gamma_\kappa}{\beta}\right)(\mathbf{q}) & \text{for } \beta > 0 \\ 0 & \text{for } \beta \le 0 \end{cases} \tag{4.15}$$

Where $\gamma$ and $\beta$ are implicit representation of the geometric data as describe in section (4.2) and (4.13). $\sigma_1$ is the analytic switch function defined in section (4.5) with $\lambda = 1$, and $\rho_\kappa$ is the sharpening function defined in section (4.6).

From Equation (4.15), analytic switch function $s(q, \lambda)$ performed the following operation:

$$s(\mathbf{q}, \lambda) = \left(\sigma_\lambda \circ \frac{\gamma}{\beta}\right)(\mathbf{q}) = \frac{\gamma(\mathbf{q})}{\lambda\beta(\mathbf{q}) + \gamma(\mathbf{q})} \tag{4.16}$$

Equation (4.16) has the following properties: it vanishes exactly at the zeros of $\gamma$, achieves its upper bound of unity exactly at the zeroes of $\beta$, and varies smoothly between the two elsewhere.

Using Equation (4.16), the sharpening function given in Equation (4.6), combined with Equation (4.16), we obtained the following *Navigation Function* in the sphere world:

$$\varphi_\kappa(\mathbf{q}) = U_{Total}(\mathbf{q}) = \begin{cases} \left(\rho_\kappa \circ \sigma_1 \circ \frac{\gamma_\kappa}{\beta}\right) = \frac{\|\mathbf{q}-\mathbf{q}_{Tar}\|^2}{\left[\|\mathbf{q}-\mathbf{q}_{Tar}\|^{2\kappa} + \beta(\mathbf{q})\right]^{1/\kappa}} & \text{for } \beta > 0 \\ 1 & \text{for } \beta \le 0 \end{cases} \tag{4.17}$$

The value of $\varphi_\kappa(\mathbf{q})$ varies between $[0,\ 1]$. $\beta$ is the product of obstacle functions as defined in Equation (4.4), $\beta \le 0$ constitute the points "inside" the obstacle, and thus gives a maximum value of 1. By varying the $\kappa$ value, we can obtained different shape of navigation function. This is shown in Figure 4-4 (a)-(f), Figure 4-3(g)-(l), and Figure 4-4(m)-(r), where the 3D potential field of a workspace with four obstacles and their contour are plotted. In these figure, obstacles are located at $(2,\ 0)$, $(0,\ 2)$, $(-2,\ 0)$, and $(0,\ -2)$, with a radius of $0.5$. On the other hand, target is located at $(4,\ 0)$. By observing these figures, we can see that the potential field generated from navigation function has a smooth contour only with a proper selected $\kappa$ value. However, there is no equation to determine the value of $\kappa$ that guaranteed a smooth navigation function. To overcome this issue, we created a MATLAB™ GUI to assist the selection of $\kappa$ value. We will discuss this with more detail in section 4.6.

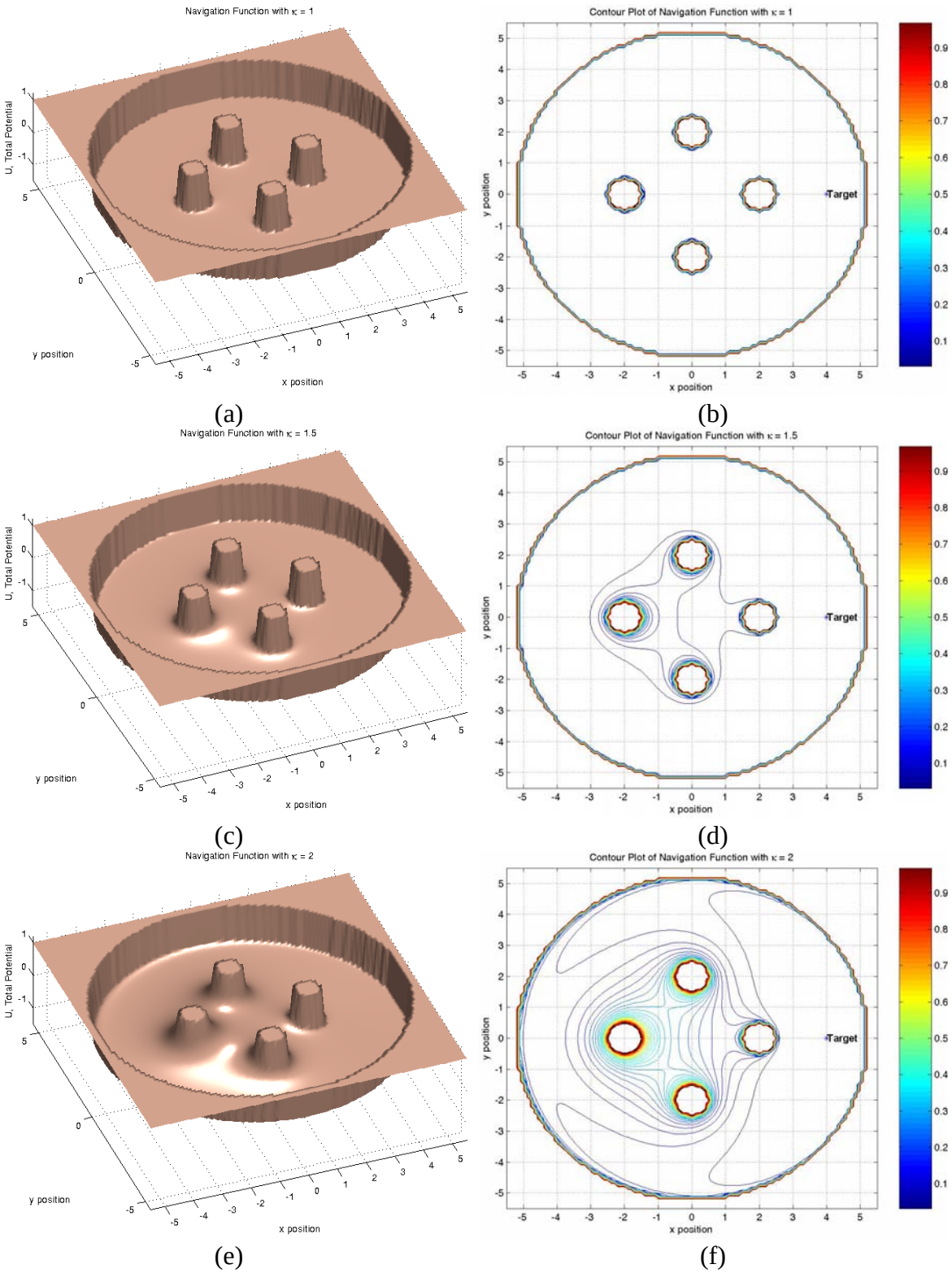



**Figure 4-2(a)-(f): The 3D potential field and its respective contour plot generated using navigation function for a workspace with 4 obstacles: observe the shape changes as the $\kappa$ value increase from 1 to 2.**

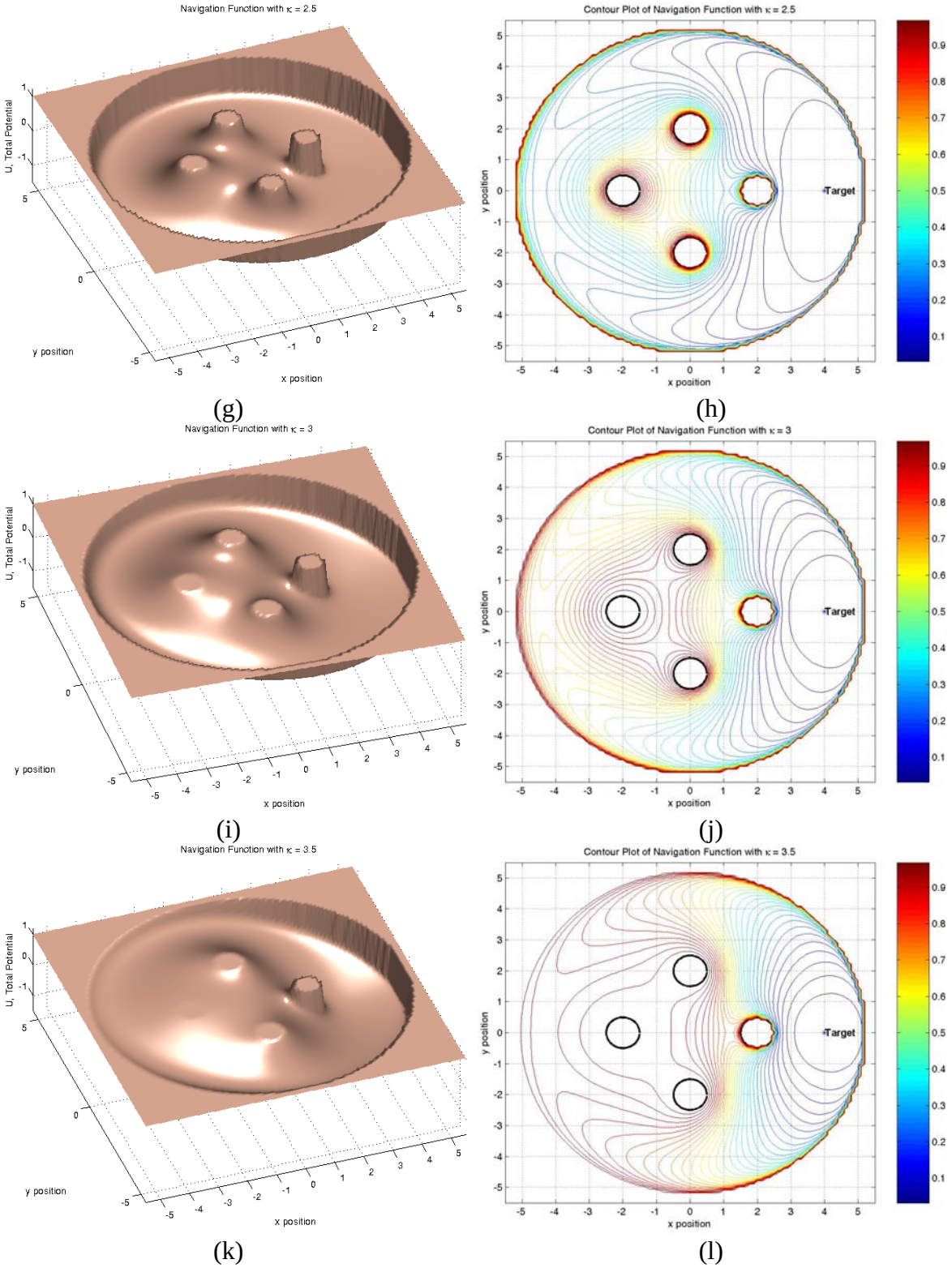
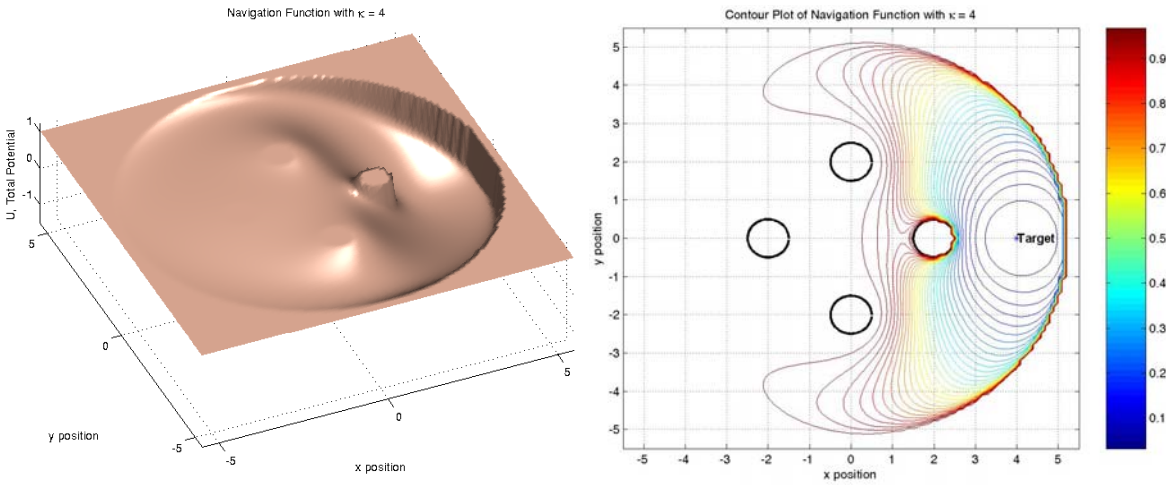


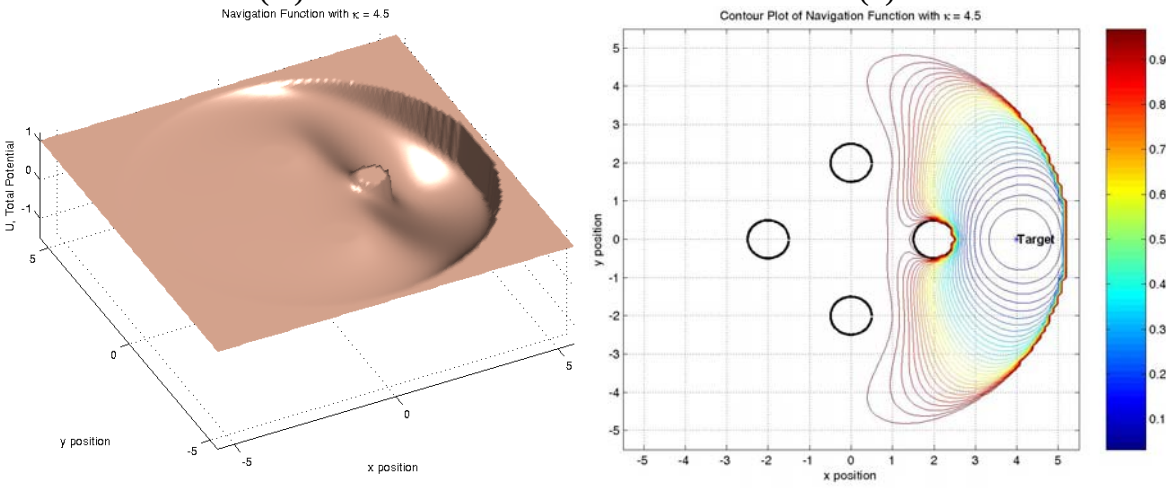

**Figure 4-3(g)-(l): The 3D potential field and its respective contour plot generated using navigation function for a workspace with 4 obstacles: observe the shape changes as the $\kappa$ value increase from 2.5 to 3.5.**




(m)



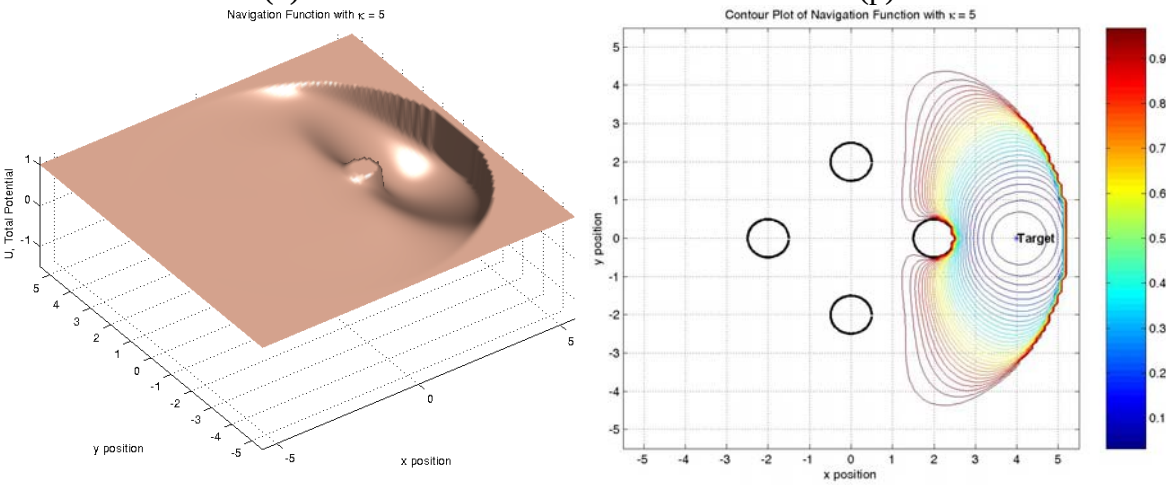

(n)




(o)




(p)




(q)




(r)

**Figure 4-4(m)-(r): The 3D potential field and its respective contour plot generated using navigation function for a workspace with 4 obstacles: observe the shape changes as the $\kappa$ value increase from 4 to 5.**

In some arrangement of the obstacles, undesired local minima may exists at low value of $\kappa$. However, the local minima disappeared as the $\kappa$ value is increased. One example is shown in Figure 4-5, the contour plot of a workspace with 4 obstacles having the same radius in a specific arrangement created using Equation (4.17). With and a $\kappa$ value of 2.6, a local minima occurs at position $(-3.5, 2)$. However, as $\kappa$ value increased to 3.6, the local minima disappeared.

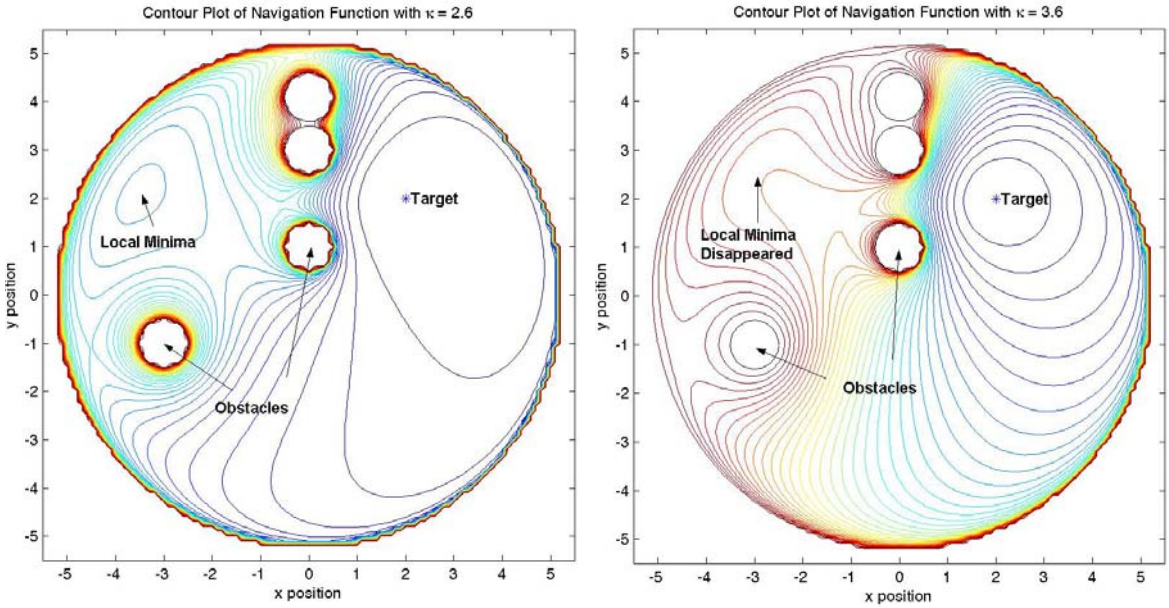


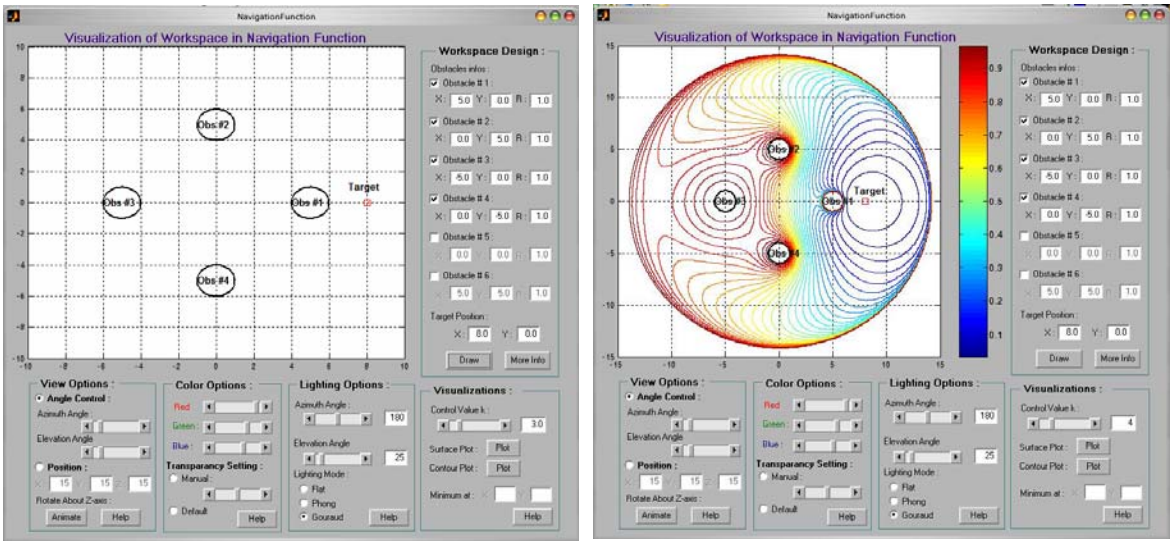

**Figure 4-5: Contour plot of a potential field for a workspace with 5 obstacles created using navigation function. In (a), local minima exist with a $\kappa$ value of 2.6; and (b) Local minima disappeared as $\kappa$ value increased to 3.6.**

## 4.6 Visualization of Navigation Function – MATLAB™ GUI

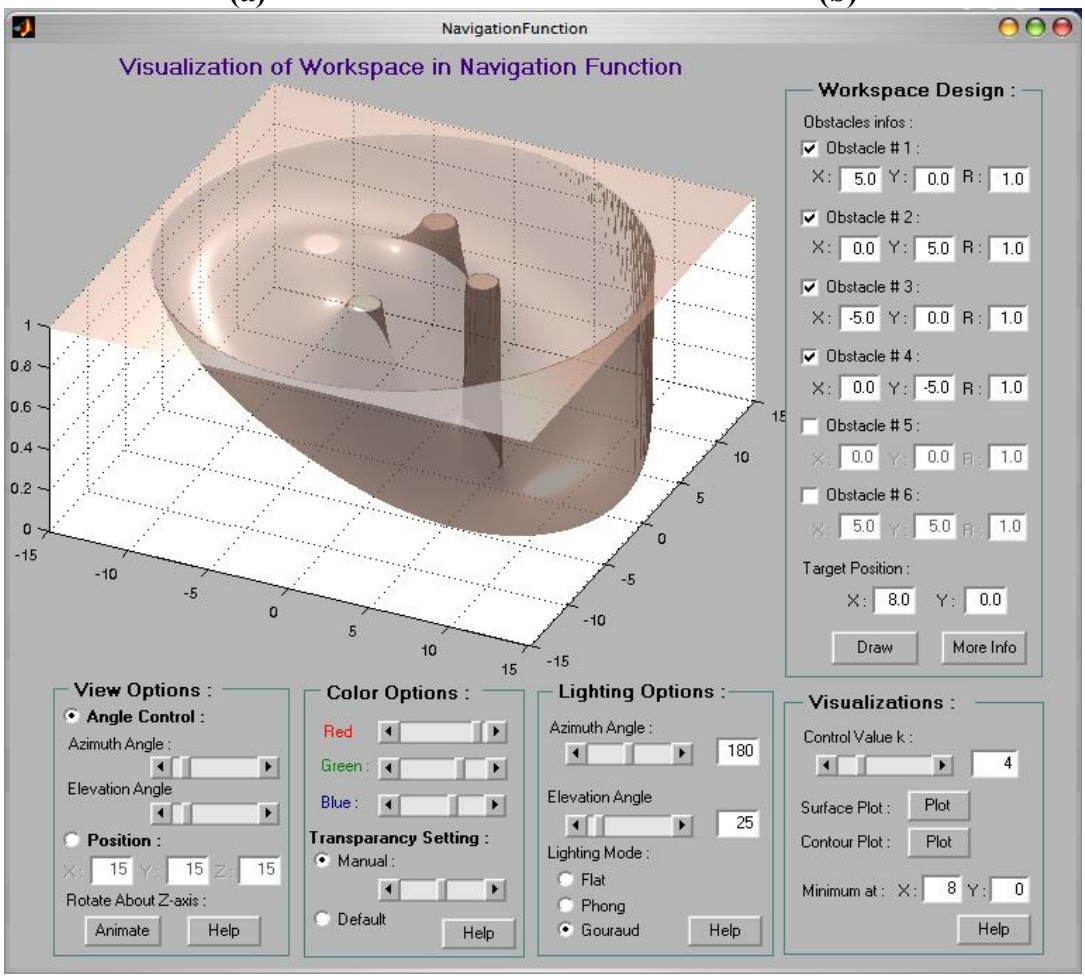
As we mentioned in the beginning of this chapter, we would like to use navigation function to create the potential field of the workspace as the test-bed for multi-robots motion planning describe in subsequence chapters. Navigation function of a workspace

provides us a potential field with no local minima when a proper value of $\kappa$ is selected. However, as mention earlier, there is no equation that can provide us the value of $\kappa$ that gives us a smooth navigation function. One observation can be made is: the higher number of obstacles in the workspace, the higher value of $\kappa$ needed to assign for the navigation function in Equation (4.17) for a smooth potential field. Therefore, we need a tool that allow us to design the workspace to use for our motion planning test-bed, and visualize the potential field of that workspace created using navigation function, both 3-dimensional and contour plot. Specifically, this tool will allow us to: 1) design the workspace consist of different number, position and location of obstacles and target; 2) visualize the potential field in its 3D plot and contour plot; and 3)allow us to change the value of $\kappa$ in Equation (4.17), visualized the respective changes in the potential field, and finally determined the best value of $\kappa$. Further, we also realized that the potential field created using navigation function is a global approach and is not suitable for real-time implementation. Hence, the process of determining the best value of $\kappa$ is done off-line, before the motion taking place.

As a result, we developed a GUI using MATLAB™ that provide us a convenient user interface to design the potential field of a workspace using navigation function. This GUI is shown in Figure 4-6(a) and (b). Detailed functionalities provide by this GUI can be found in Appendix A.4.




(a) (b)




(c)

**Figure 4-6: A design and visualization GUI created using MATLAB™ that allow user to (a) Design the workspace in 2D environment; and (b) Visualize the contour plot of the potential field; and (c) Visualize the 3D potential field of the workspace, created using navigation function.**

## 4.7 Summary

While navigation function provides a solution to the local minima and torque limit problem found in local potential field approaches, it remains a global method. This means that it losses the advantages of local potential fields approach where only local information is needed. Further, while the repulsive potential function in local potential approaches have a limited range of influence (which is a desired feature), all obstacles in the navigation function contributes to the final shape of the potential field, no matter how far they located.

In this thesis, we would like to implement potential field approaches for motion planning of robot collectives. Hence, it is crucial to select a potential field that is free of local minima to use as a 'test-bed' or a 'platform' to allow us to implement our formation control algorithm developed in subsequent chapters. The navigation function that we discuss in this chapter is, in our opinion, a perfect candidate for our implementation. By using navigation function, we are guaranteed to get a potential field of the configuration that is free of local minima that allow us to test the performance of our algorithms for formation control on a common platform.

# 5 Robot Kinematics and Dynamics Formulation

In this chapter, we first formulate the kinematics and dynamics equations for single robot module, followed by the simulations of the motion of single robot with these formulations in the potential fields described in the previous chapters. We also formulated the kinematics and dynamics equations of group of robots without formation constraints, and study their behaviors in potential field. The kinematics and dynamics of group of robots with formation constraints will be treated in Chapter 5.

As we mentioned in Chapter 2, the dynamic of a robot in a potential field is like a marble rolling down to the lowest point of a surface due to the effects of gravitational force. An example of this analogy is illustrated in Figure 5-1. In mathematical words, we want the marble to flow in a direction along the *negative of the gradient* (toward the lowest point) and stop when it reached the lowest point of a potential field.

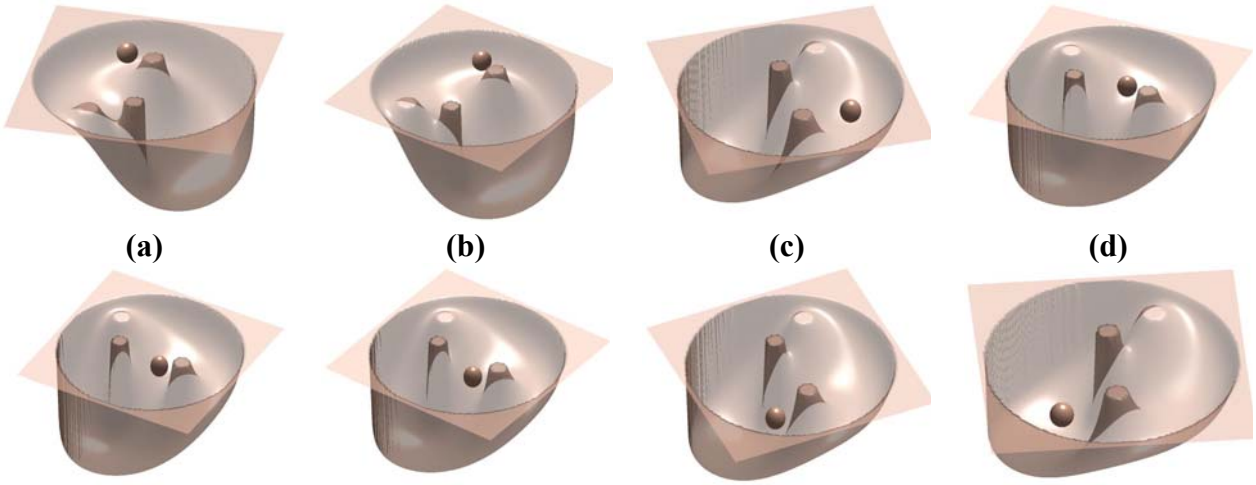



**Figure 5-1: Graphs showing a marble rolling down a potential field from its initial position to the lowest point in the potential, from different view angles. Each position is obtained through simulation using the dynamic equation formulation for single point-mass robot given in this chapter.**

Thus, the gradient information is used as an input to the dynamic system of the robot. As a result, we transformed the motion planning problem to a simulation of forward dynamics problem. For all the points beside the target point in the configuration space, gradient information always exists and serves as the input to the system. At the lowest point, the gradient becomes zero, and there will be no more input force to the system, the robot will stop there. This is simply letting $\mathbf{u} = -\mathbf{K}_f \left( \nabla_{\mathbf{q}} U_{Total} \right)$ in the kinematics or dynamic equation of a point-mass robot, where $\mathbf{u}$ is the input to the system; $\mathbf{q}$ is the position vector of the robot, $\mathbf{K}_f$ is a positive diagonal scaling matrix, and $U_{Total}$ is the total potential field of the workspace defined in previous chapter. For simplicity, we will use $U = U_{Total}$ in the subsequent chapters.

In the following sections, we formulate the dynamic equation for single point-mass robot, and the dynamic equation for multiple robots without formation constraints. Also, several case studies were performed to provide us a better understanding the potential approach in robot motion planning.

## 5.1 Single Point-Mass Model

A point mass robot is the simplest model of real robot where each robot is assumed to exhibit the dynamics of single point mass particle. By assuming we have full control of the actuators on the robot, we derive the following kinematic and dynamic equations.

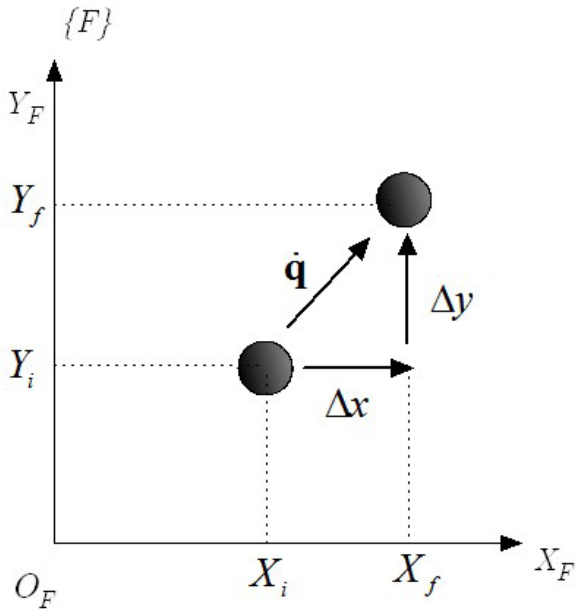



**Figure 5-2: A single point mass robot on x-y plane has two degree of freedom.**

### 5.1.1 Kinematics Model

The kinematics equation of a point-mass robot which position vector is given by $\mathbf{q} = \left[x, y\right]^T$ is governed by following equation:

$$\dot{\mathbf{q}} = \mathbf{u} = -\mathbf{K}_f \left(\nabla_{\mathbf{q}} U\right) \tag{5.1}$$

where $\mathbf{u}$ denote the input vector to the system, $\nabla_{\mathbf{q}} U$ is the gradient of the potential field, and $\mathbf{K}_f = k_f \mathbf{I}_{2\times 2}$ is a positive diagonal scaling matrix.

### 5.1.2 Dynamic Model

The dynamics equation of a point-mass robot can be described by the following equation:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{u} - \mathbf{K}\dot{\mathbf{q}} = -\mathbf{K}_f \left(\nabla_{\mathbf{q}} U\right) - \mathbf{K}\dot{\mathbf{q}} \tag{5.2}$$

where $\mathbf{M}$ denote the mass of the point-mass robot, has the form of $\mathbf{M} = diag\{m_n\}_{2n\times 2n}$; $\mathbf{u} = -\mathbf{K}_f\left(\nabla_{\mathbf{q}} U\right)$ is the input force to the system, and $\mathbf{K}\dot{\mathbf{q}}$ is dissipative term added to stabilize the system, where $\mathbf{K} = diag\{k\}_{2n\times 2n}$.

## 5.2 Nonholonomic Wheeled Mobile Robot Model

The configuration of a differentially driven wheled mobile robot can be locally parametrized by a configuration vector as $\mathbf{q} = [x, y, \phi]^T$. This mobile base is subjected to a nonholonomic constraint that can be written as:

$$\dot{x}\cos\phi - \dot{y}\sin\phi = 0 \tag{5.3}$$

or in Pffafian form as:

$$A(\mathbf{q})\dot{\mathbf{q}} = 0 \tag{5.4}$$

where $A(\mathbf{q}) = [\sin\phi \quad -\cos\phi \quad 0]$ and $\dot{\mathbf{q}} = [\dot{x} \quad \dot{y} \quad \dot{\phi}]^T$. The set of feasible configuration velocities $\dot{\mathbf{q}}$ that are consistent with this nonholonomic constraint can thus be determined to lie in the null space of this constraint matrix. Let $S(\mathbf{q})$ be a matrix whose column span the null space of $A(\mathbf{q})$, i.e., $A(\mathbf{q})S(\mathbf{q}) = 0$. While the choice of $S(\mathbf{q})$ is non-unique, one common parametrization of the feasible configuration velocities is given by:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5.5}$$

where $v$ and $\omega$ are the forward translational velocity and angular velocity of the WMR.

### 5.2.1 Kinematics Model

From Equation (5.5), we can derived the kinematic model for the WMR as:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2 \tag{5.6}$$

where $u_1$ be the input from the gradient of the workspace, and $u_2$ be the angular velocities of the WMR. This is illustrated in Figure 5-3.

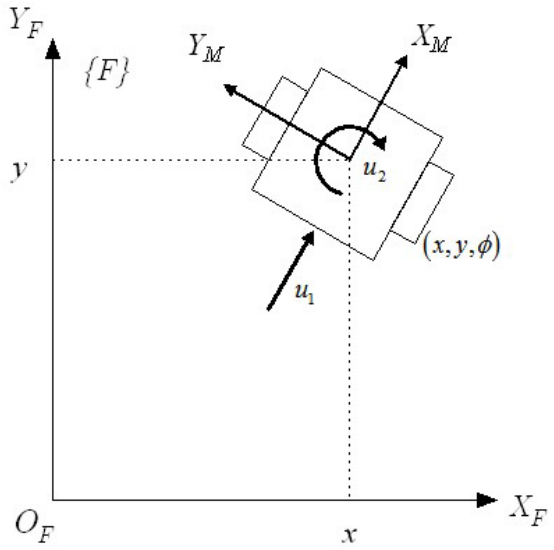



**Figure 5-3: A Nonholonomic Wheeled Mobile Robot (NH-WMR) kinematic model. Two inputs to the system are $u_1$, the forward velocity; and $u_2$, the angular velocity.**

Here, we dopted method proposed in [77] to determine the value of $u_1$ and $u_2$. Re-write Equation (5.5) in the form of:

$$\dot{\mathbf{q}} = S(\mathbf{q})\mathbf{u} \tag{5.7}$$

Given any desired trajectory $\mathbf{q}_d$, a straightforward approach is to design the input command $\mathbf{u}$ using pseudoinversion:

$$\mathbf{u} = S(\mathbf{q})^{\#} \dot{\mathbf{q}}_d = \left[ S^T(\mathbf{q}) S(\mathbf{q}) \right]^{-1} S^T(\mathbf{q}) \dot{\mathbf{q}}_d \tag{5.8}$$

This solution minimizes the error $\left(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}\right)$ in a least square sense. $\dot{\mathbf{q}}_d$ in Equation (5.8) can be chosen as the output of an incremental holonomic planner or in our case, a potentials used to drive the robot:

$$\dot{\mathbf{q}}_d = -\nabla_{\mathbf{q}} U \tag{5.9}$$

Using Equation (5.8), the control input can be chosen as:

$$\mathbf{u} = S^{\#}\left(\mathbf{q}\right)\dot{\mathbf{q}}_d = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\phi}_d \end{bmatrix} \tag{5.10}$$

where $\dot{x}_d = -\dfrac{\partial U}{\partial x}$, $\dot{y}_d = -\dfrac{\partial U}{\partial y}$, and $\dot{\phi}_d$ is obtain from:

$$\dot{\phi}_d = atan2\left(\dot{x}_d, \dot{y}_d\right) - \phi \tag{5.11}$$

This formulation takes the magnitude of the gradient of the potential field as the input velocity to the system and aligns the NH-WMR angle $\phi$ to the direction of the gradient flow. As a result, the kinematic model for the NH-WMR in a potential field is given by:

$$\begin{aligned} \dot{\mathbf{q}} &= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2 \\ u_1 &= k_p\left(\dot{x}_d \cos\phi + \dot{y}_d \sin\phi\right) \\ u_2 &= k_\phi\left(atan2\left(\dot{x}_d, \dot{y}_d\right) - \phi\right) \end{aligned} \tag{5.12}$$

where $k_p$ and $k_\phi$ are gains introduced to allow for additional freedom in weighting the two input commands. A simulation study of this model in a potential field is given in section 5.4.4.

## 5.3 Group of Point-mass Robots without Formation Constraint

Assuming all robots only moves in $SE(2)$, i.e. robots only move in $x$ and $y$ directions. Given $n$-number of robots, their generalized position in $SE(2)$ can be written as:

$$\mathbf{q} = \left[x_1, y_1, x_2, y_2 \ldots, x_n, y_n\right]^T \tag{5.13}$$

### 5.3.1 Kinematics Model

The kinematics equation of a group of $n$-number of point-mass robot without formation is described by following equation:

$$\dot{\mathbf{q}} = \mathbf{u} \tag{5.14}$$

where $\mathbf{q} = \left[x_1, y_1, \cdots x_n, y_n\right]^T$, and $\mathbf{u} = -\mathbf{K}_f \nabla_{\mathbf{q}} U$ where $\mathbf{K}_f = diag\left\{k_{f_n}\right\}_{2n\times 2n}$.

### 5.3.2 Dynamics Model

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{u} - \mathbf{K}\dot{\mathbf{q}} \tag{5.15}$$

where $\mathbf{M} = diag\left\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\right\}$, a $2n \times 2n$ diagonal mass matrix; $\mathbf{u} = -\mathbf{K}_f \nabla_{\mathbf{q}} U$, and $\nabla_{\mathbf{q}} U = \dfrac{\partial U}{\partial \mathbf{q}} = \left[\dfrac{\partial U}{\partial x_1}, \dfrac{\partial U}{\partial y_1}, \dfrac{\partial U}{\partial x_2}, \dfrac{\partial U}{\partial y_2}, \cdots, \dfrac{\partial U}{\partial x_n}, \dfrac{\partial U}{\partial y_n}\right]^T$; and $\mathbf{K} = diag\left\{k_{x_1}, k_{y_1}, \cdots, k_{x_n}, k_{y_n}\right\}$, also a $2n \times 2n$ diagonal control matrix. To write in state-space form:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{M}^{-1}\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{u} \tag{5.16}$$

where $\mathbf{u} = -\mathbf{K}_f \nabla_{\mathbf{q}} U$. Equation (5.15) and (5.16) are the dynamics equation for group of point-mass robot with potential field approach. These formulations do not require the

robots to maintain a certain formation throughout their motions. Thus, one would expect the point-mass robot converge to the target while avoiding obstacles, but collision between each robot are not guaranteed.

## 5.4 Simulation Results

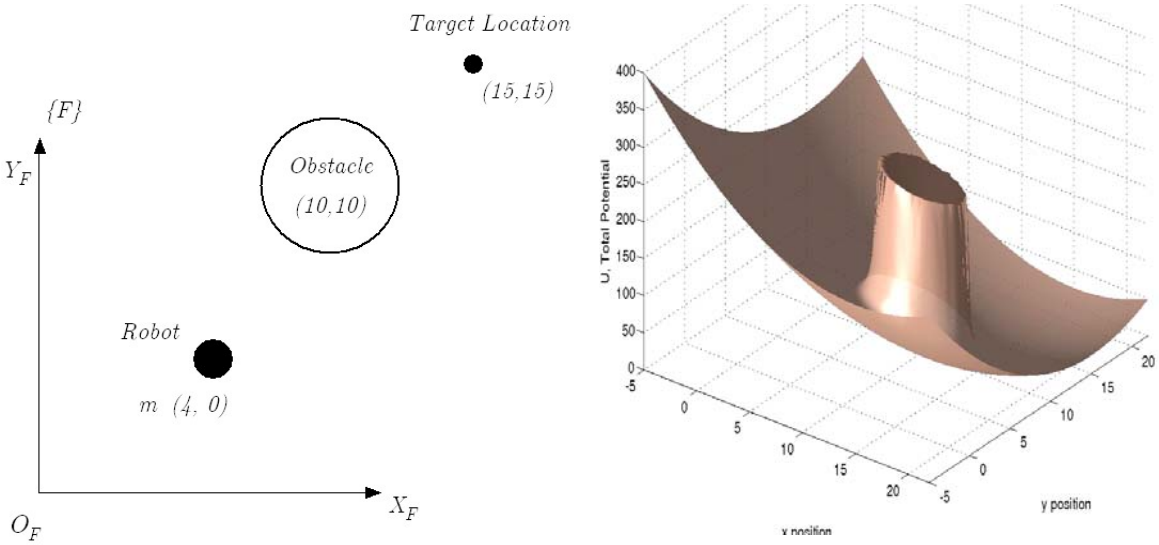
In this section, we provide several case studies using the dynamic equation that we shown in previous sections. In particular, motion planning for a single point-mass robot is studied using various potential functions discussed in Chapter 3. In all of the simulation, the solver being used is the fixed-time step solver-ODE5, provided by MATLAB™, unless otherwise stated. For more detailed information on solvers provided by MATLAB™, please see Appendix A.5.

### 5.4.1 Case Study 1: Motion Planning of Single Point Mass Robot in Workspace with One Obstacle

In the first case study, a single point-mass robot needs to move to the target while an obstacle lies between them. The target is locate at $(15, 15)$, and the obstacle locate at $(10, 10)$ with a radius of $3$. The robot start from initial position $(4, 0)$. The arrangement of the workspace and the robot is shown in Figure 5-4(a). The corresponding potential field shown in Figure 5-4(b) is generated using FIRAS function given by Equation (3.6), where $\rho_0 = 4 \times Obs_{radius}$, and $\eta = 2$.

Figure 5-5 shows the contour plot of the potential field and also the simulation result. From the contour plot, we can see that FIRAS function generates high potential at the surface of the obstacle, and the potential drops sharply beside the surface. This

simulation is carried out in 10 seconds, with a fixed time-step of $1\times10^{-3}$ seconds. The gradient information is obtained analytically, by differentiate Equation (3.6).




(a) (b)

**Figure 5-4: (a) 2D workspace arrangement for case study 1; and (b) the potential field created using FIRAS Function, large value had been truncated.**

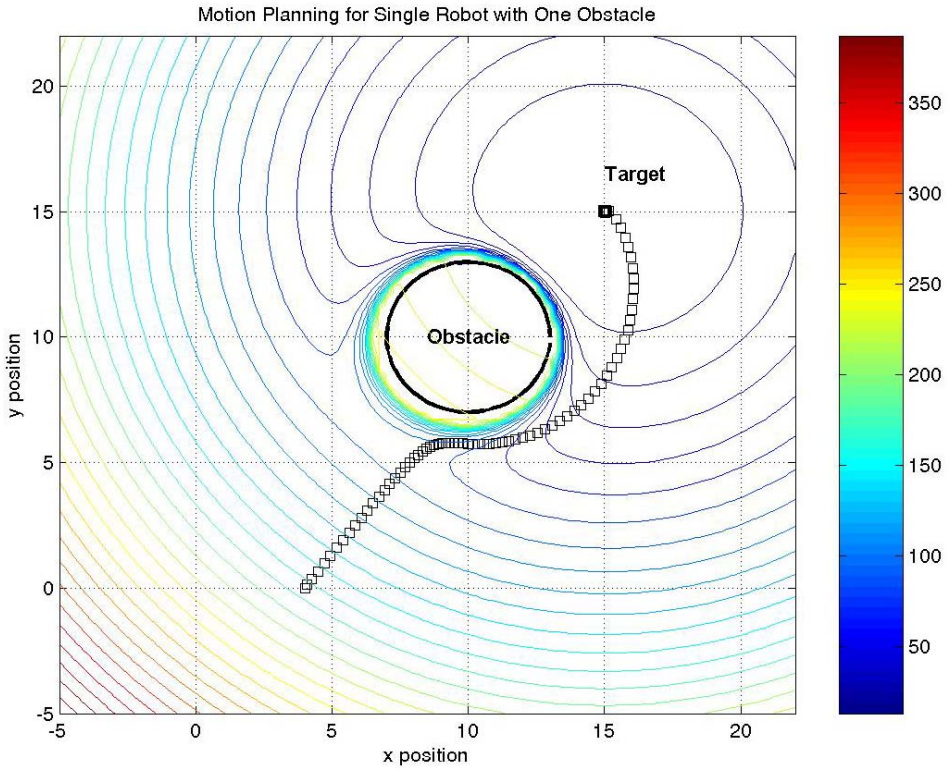
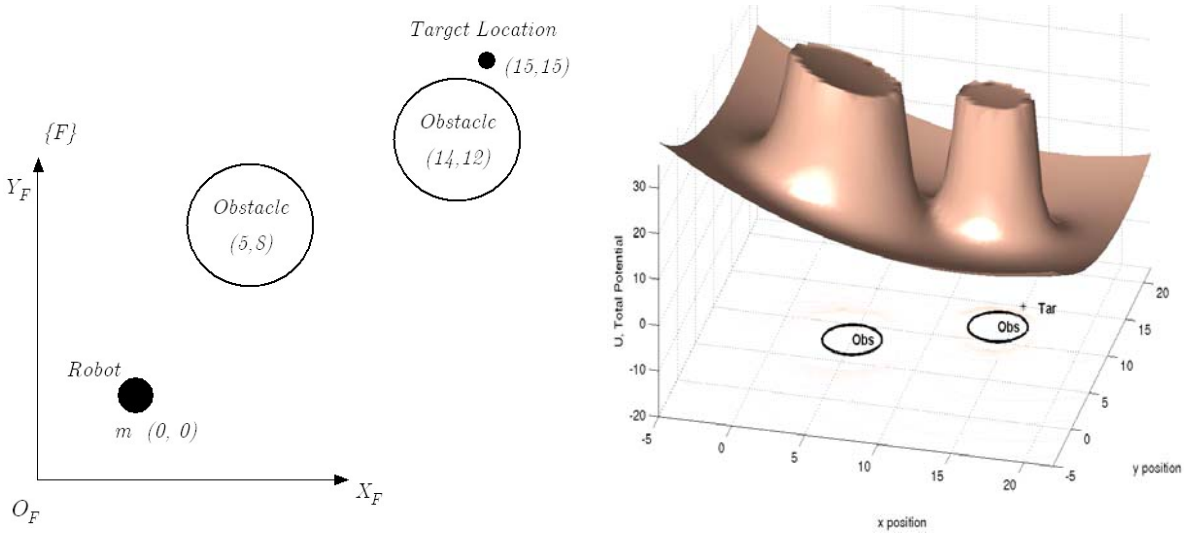



**Figure 5-5: Simulation result of single point-mass robot in workspace with one obstacle created using FIRAS function with fixed time-step solver. Position of robot is plotted with interval of 0.1 seconds.**

The dynamic equation used in this simulation is given by Equation (5.2), with $k_f = 10$ and $k = 5$. From this simple example, we demonstrated the use of potential field in obstacle avoidance for a single robot. The robot avoided collision with the obstacle by following the gradient of the potential field and reach the desired location. As shown in [30], we can also control the velocity of the robot such that it will not exceed the speed limit. The solver used in this simulation is ODE5, provided by MATLAB™. More information on this fixed time step solver is provided in Appendix A.5.

### 5.4.2 Case Study 2: Motion Planning of Single Point Mass Robot in Workspace with Two Obstacles

In the second case study, we placed a target close to the obstacle (i.e., the obstacle is inside the influence range of the obstacle's repulsive potential). There are two obstacles in the workspace, such that the robot needs to pass through the two obstacles to reach its target location. The 2D workspace of this case study is provided in Figure 5-6(a). Here, the robot initial position located at $(0,\ 0)$, and the target is located at $(15,\ 15)$. Two obstacles are located at $(5,\ 8)$ and $(14,\ 12)$ respectively, each with a radius of $2$. The potential field is generated by superimposed Ge's new potential, give in Equation (3.14) and the attractive potential field given in Equation (3.2). Where the influence range of the obstacle, is given by $\rho_0 = 5 \times Obs_{Radius}$, i.e., five times the obstacle radius. The simulation time is 20 seconds, with a fixed time-step of $4 \times 10^{-3}$. The simulation result is given in Figure 5-7.




**(a)** **(b)**

**Figure 5-6: (a) 2D workspace arrangement for case study 2; and (b) the respective potential field created using Ge's new potential, large value had been truncated.**

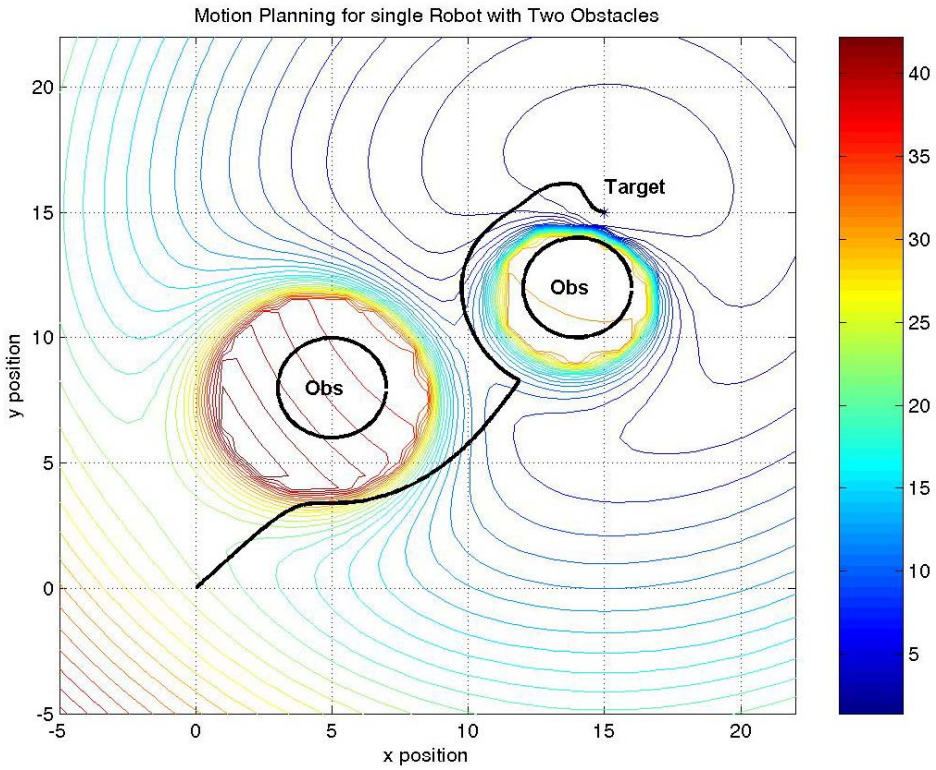
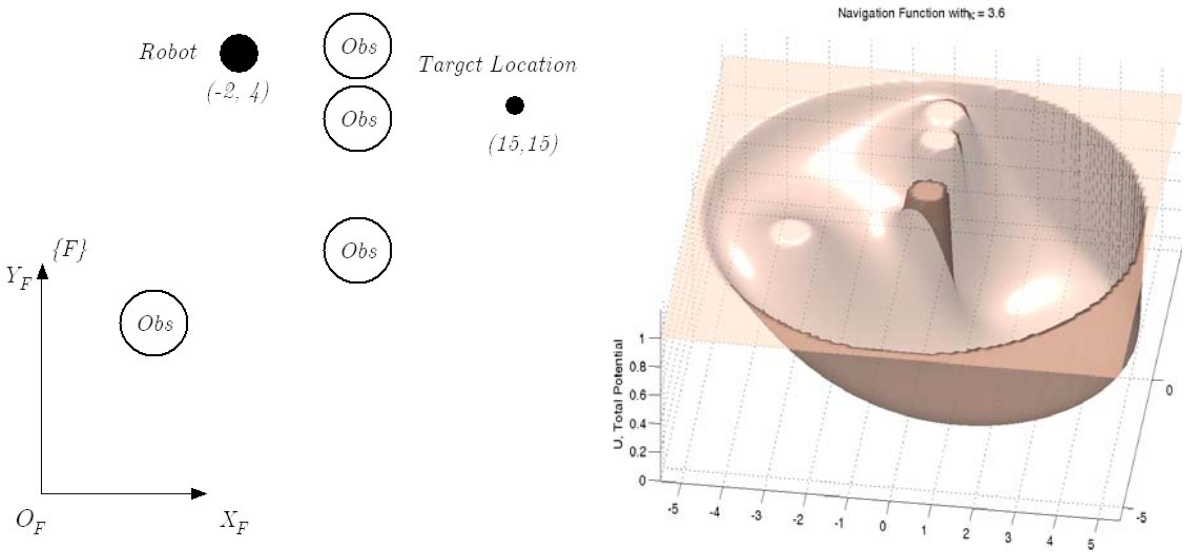



**Figure 5-7: Simulation result for single point-mass robot in a workspace with two obstacles, where the target point is placed close to the obstacle.**

From the simulation result, we can see that the potential field created using Ge's new potential function solved the problem of GNRON as mention in section 3.1.2.4.

### 5.4.3 Case Study 3: Motion Planning of Single Point-mass Robot in Navigation Function

In this case study, four obstacles are arranged such that they blocked the robot from the target, as given in Figure 5-6(a). If potential functions that we discussed in chapter 3 were used to generate the potential field for this workspace, local minima will occur. Thus, navigation function is used to generate the potential field for this case study. As shown in section 4.5, at lower lever of $\kappa$, the potential field created using navigation function for this workspace had a local minima. Here, we use a $\kappa = 3.6$ value such that the local minima disappeared. The 3D view of the potential field is given in Figure 5-8(b). Here the potential field is set to transparent for easy viewing. Initially, the robot is located at $(-2,\ 4)$ and the target is located at $(15,\ 15)$, in between them is four obstacles with radius of $0.5$ and each located at $(-3,-1)$, $(0,\ 1)$, $(0,\ 3)$, and $(0,\ 4.1)$.




(a) (b)

**Figure 5-8: (a) 2D workspace arrangement of robot, target and four obstacles; and (b) the respective potential field of the workspace created using navigation function.**

Using a fixed time step solver ODE5, with a total simulation time of 10 seconds and a time step of $1\times10^{-3}$ seconds, the simulation result is given in Figure 5-9. In this simulation, the gradient information is obtained numerically. When using navigation function to create the potential field, it is difficult to obtain the gradient information analytically by differentiate the potential function. Hence, the gradient information is obtained numerically whenever a navigation function is used to create a potential field. For more detailed information on obtaining the gradient numerically, please refer to Appendix A.2.

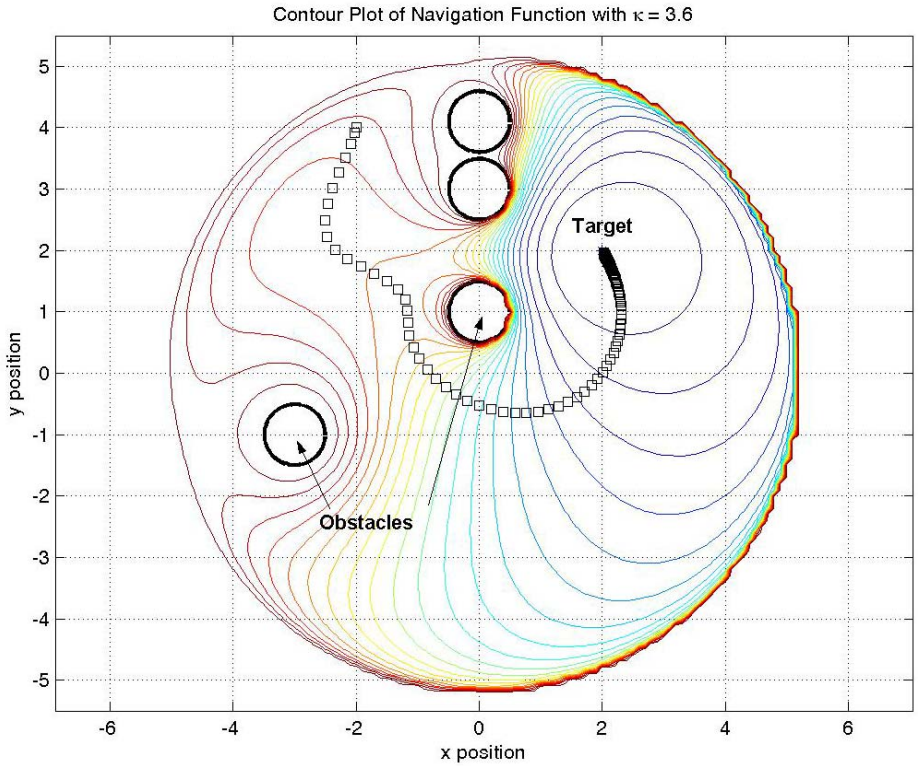



**Figure 5-9: Simulation result for case study 3 using fixed time step solver. Note that only selected step of the robot position are plotted.**

In Figure 5-9, we can see that the robot successfully reached the target without collides with the obstacles. In the result shown, we only plotted the position of the robot every $5\times10^{-3}$ seconds, for better viewing.

### 5.4.4 Case Study 4: Motion Planning of Single Nonholonomic Wheel Mobile Robot in Navigation Function

Using the same workspace setting given in case study 3, motion planning for a single nonholonomic wheeled mobile robot (NH-WMR) is performed. The initial position and orientation of the NH-WMR is given by $(-2,\ 4)$ and $0^\circ$. The configuration space is shown in Figure 5-10. The NH-WMR is represented as a triangular to better illustrate its angle in the simulation. The resulting potential field is created using navigation function with a $\kappa$ value of $3.6$.

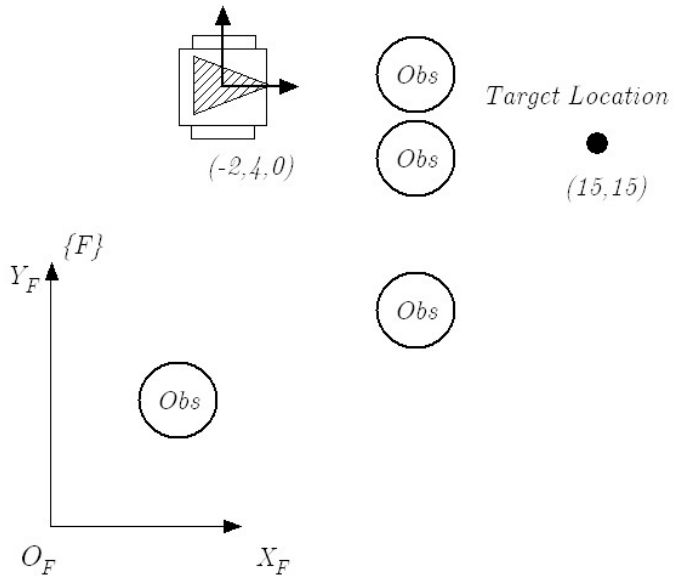



**Figure 5-10: Arrangement of robot, obstacles, and target position used in case study 4.**

Using a fixed time time step solver ODE5 with a simulation time of $7$ seconds, and a time step of $1\times10^{-3}$ second, the simulation result is shown in Figure 5-11. The kinematic equation used in this simulation is Equation (5.12), with $k_p = 1$ and $k_\phi = 5$.

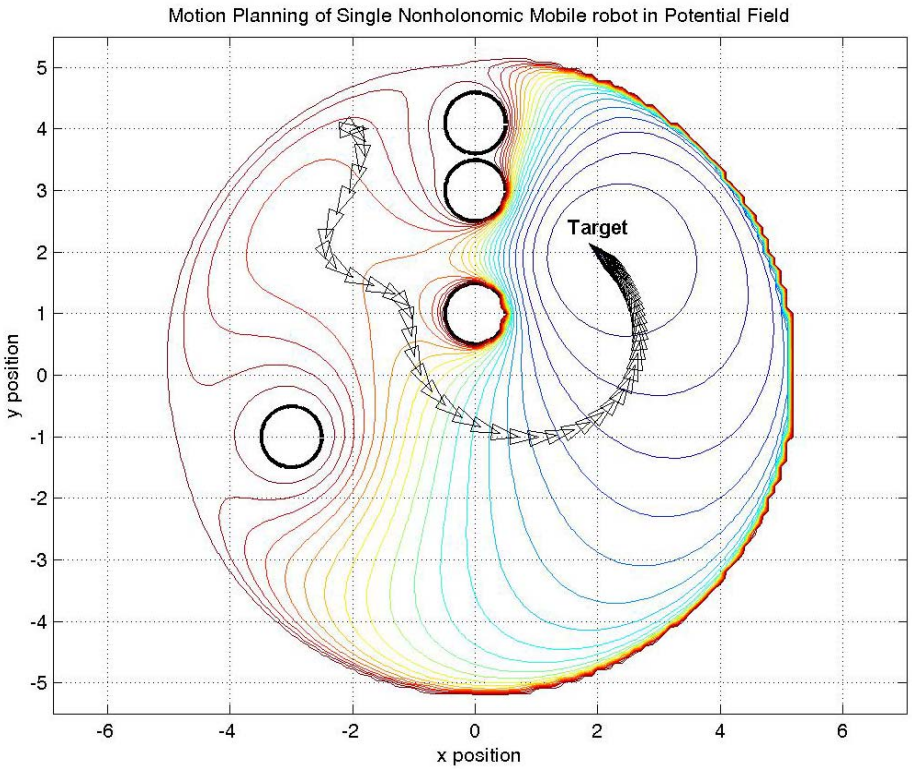



**Figure 5-11: Motion planning of a nonholonomic wheeled mobile (NH-WMR) robot in a potential field with 4 obstacles created using navigation function.**

From the simulation, note that the WMR is pointed towards the negative gradient of the potential field. In this simulation plot, only steps in an interval of $1\times10^{-2}$ seconds were plotted.

### 5.4.5 Case study 5: Motion Planning of Three Point-mass Robot without Formation Constraint in Workspace Free of Obstacles

In this case study, we would like to show the dynamic behavior of group of three robots in a potential field without formation constraint between them. Three point mass robots that initially form an equal lateral triangular moving in workspace free of obstacles. Three of the robots were drive to the target in a attractive potential field described by Equation (3.2). The dynamic equation that govern this system is given by Equation (5.16).

Since there is no formation constraint between each robot, one would expect the robot cannot maintain the formation and may collides into each other. Each of the robot are initially located at $(10,\ 9)$, $(10,\ 12)$, and $(7.41,\ 10.5)$, respectively. The total simulation time is $10$ seconds with a fixed time-step of $1\times10^{-3}$ seconds.

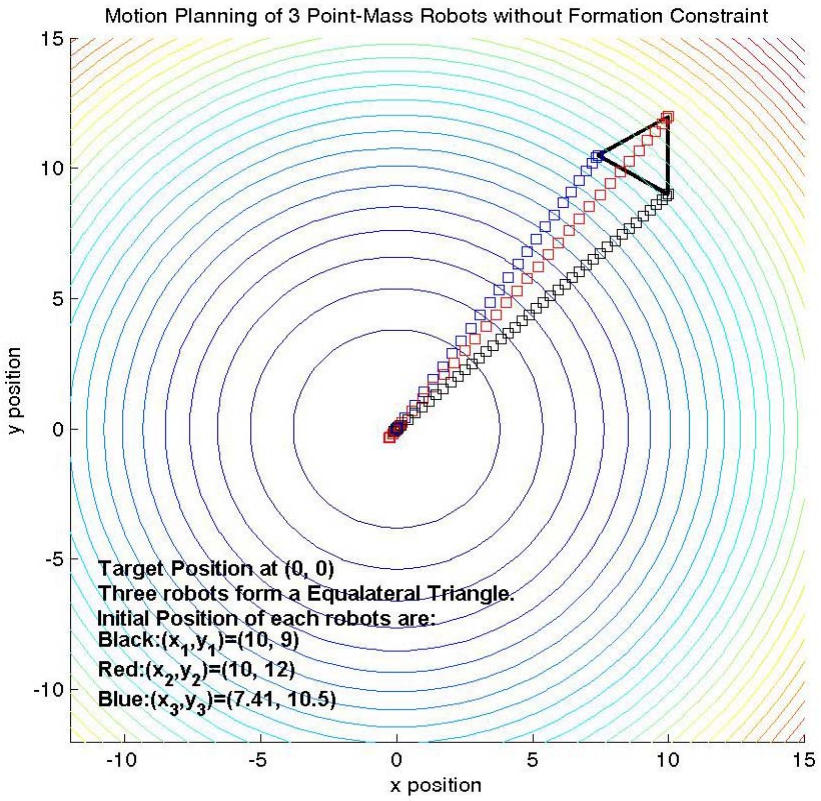
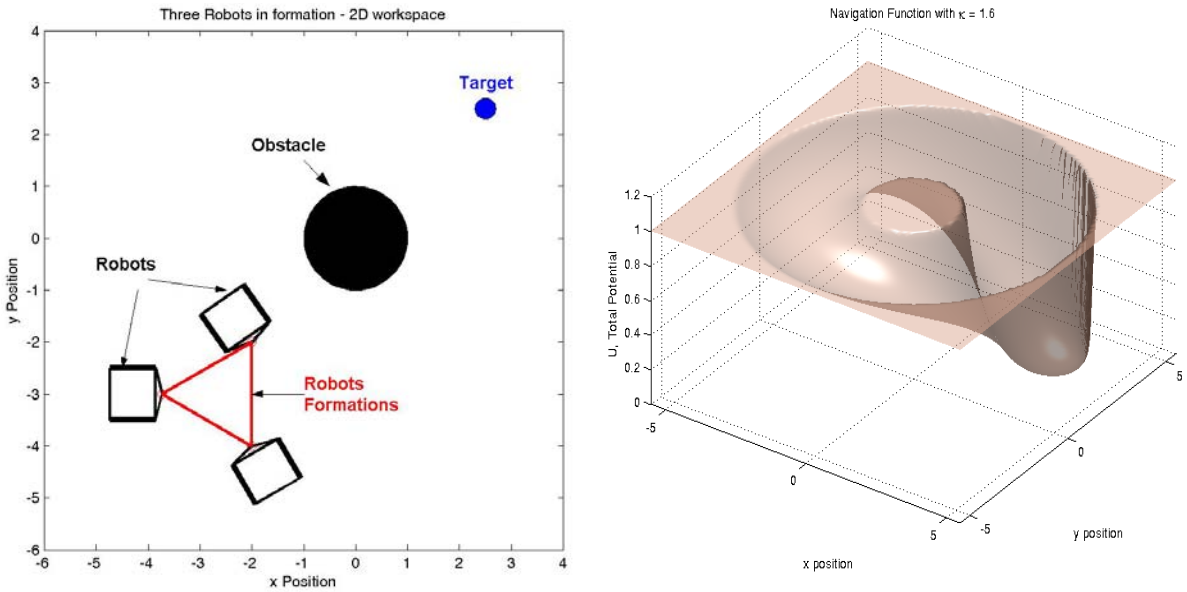



**Figure 5-12: Simulation result of three point-mass robots without formation constraint in a quadratic attractive potential field.**

From Figure 5-12, we can see that each of the robots traces the steepest decent direction of the potential field and converge to the target. However, with this formulation, they were moving independently without considering the motion of other robots. The dynamic equation for group of robots in a potential field will be given in next chapter.

### 5.4.6 Case study 6: Motion Planning of Three Point-mass Robots without Formation Constraint in Workspace with One Obstacle

In this case study, we show the motion planning of three point-mass robots without formation constraint in a workspace with one obstacle. In this case study, the potential field is created using navigation function. The three robots are each located at $(-2,\ -3)$, $(-2,\ -4)$, and $(-2.866,\ -3.5)$ respectively. The obstacle is located at the origin and has a radius of 1.5 while the target is located at $(2.5,\ 2.5)$. The dynamic equation used in this simulation is given by Equation (5.16). The gradient information at each simulation step is obtained numerically. The total simulation time is 8 seconds with a fixed time step of $1\times10^{-3}$ seconds. The result is plotted in Figure 5-14.




**(a)** **(b)**

**Figure 5-13: (a) 2D arrangement of the workspace, containing three robots and obstacle; and (b) The resulting potential field of the workspace created using navigation function.**

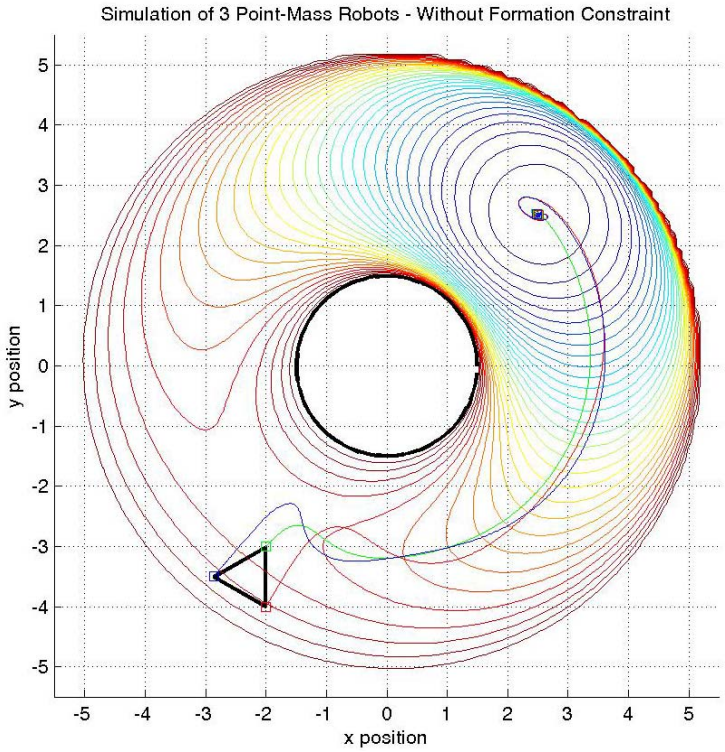



**Figure 5-14: Simulation of motion planning for three point-mass robots without formation constraint in a workspace with one obstacle.**

From Figure 5-14, we can see that the each of the robot moves in the steepest decent direction of the potential field. There is no formation maintenance requirement in this system of three robots. We calculated the total formation error for this system by using the following equation:

$$\Delta_{Error} = \sqrt{\sum_{i=1}^{M} \left(c - \bar{c}\right)_i^2} \tag{5.17}$$

where $\Delta_{Error}$ denote the total formation error, $M$ denote number of holonomic constraints in the equation, $c$ is the Euclidean distance of each holonomic constraint at each instant, and $\bar{c}$ is the desired Euclidean distance for each holonomic constraint. In this case study, where there is no formation constraint, when the robots converged to the

target, $c = 0$ and $\bar{c} = 1$. Eventually, the total formation error will converge to $\sqrt{3} = 1.732$, as shown in Figure 5-15.

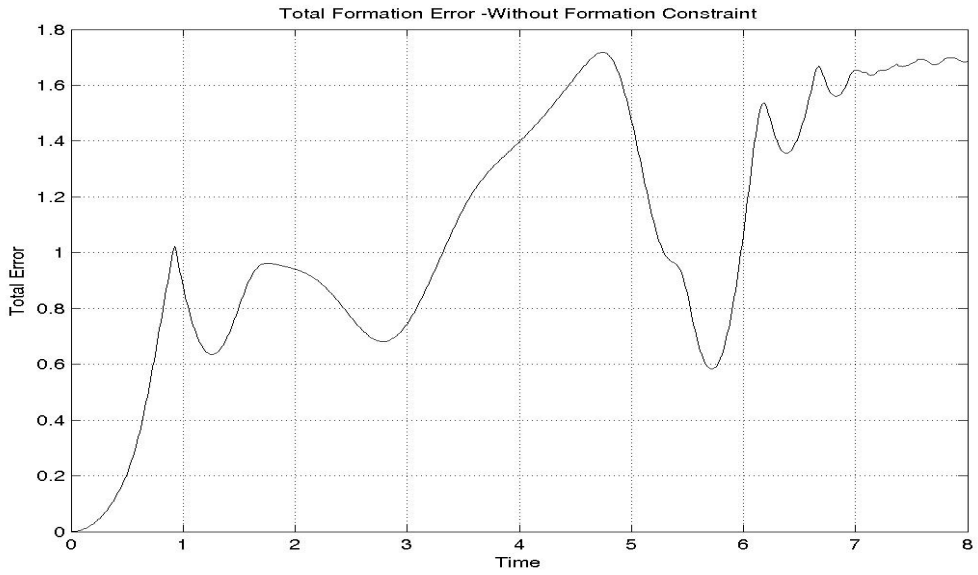



**Figure 5-15: Total formation error for group of three point-mass robots without formation constraint.**

This case study shows that we are able to use navigation function for our group robots and Figure 5-15 shows how we will evaluate the performance of the group formation maintenance ability in later chapters. In this case study, each individual robot finds its gradient information numerically and follows the steepest decent direction to reach the desired location.

## 5.5 Discussions

In this chapter, various simulations were performed using the potential functions discussed in Chapter 3 and 4. From these simulations, we see how the potential field approaches is used for motion planning of single robot. In the following chapters, we will formulate the dynamic equation for group of robots within the potential framework.

# 6 Dynamic Formulation for a Group of Robots

In formulating the dynamics equations for a group of robots, we need to ensure that each robot not only changes to a desired target position but also maintains a desired formation (encoded as distances/angles with respect to its neighbours) for the entire duration. While the artificial potential field (APF) approach discussed in the previous chapters can help avoid collision with obstacles, in its current form it cannot help maintain the desired formation or prevent collision within members of the group. Hence in this chapter, we will examine different ways in which the APF approaches can be adapted for use with such groups of robots moving in formation.

While considerable literatures exists for motion planning of individual mobile robots, the renewed challenge lies in creating motion plans for the entire team moving in formation. By 'formation', we refer to the relative geometric arrangements of individual robots in a team that needs to be maintained throughout their journey to the destination. The approaches for formation maintenance ranged from leader-following [59, 60], virtual structure [61, 62], and virtual leaders [3, 4, 31]. In these approaches, motion plans are generated for a preferred real or virtual team-leader and motion plans for all others are derived from the motion plans of the team-leader. The principal benefit is that the group control problem now reduces to a well-known single agent control from which the other agents derive their control laws with the requirement of communication of some coordination information. In early variants of such approaches, motion-plan of the

followers was determined relative to the real or virtual team-leader but without the ability to affect the dynamics of the leader. More recent work has focused on introducing some forms of 'formation-feedback' from the members to the overall group, using natural or artificially introduced dynamics. In particular, the introduction of the additional dynamics had taken the form of adding 'behavioral' constraints-such as following distance and follower angle. These constraints were formulated and introduced within the overall framework initially as virtual spring and dampers (and later in more general form as limited range, local artificial potential) that were intended to maintain the desired constraints. The principal benefit is that it allows us to adopt a dynamic system theoretic perspective to examine, analyze and derive provable performance results for decentralized multi-robot behavioral control in the context of formation [3, 4, 31].

As discussed in Chapter 1, many of these ideas derive their inspiration from observations of similar types of group behavior in nature. Groups in nature appear to make use of distributed control architecture, in which individual not only respond to their sensed environment (with a limited range), but are also constrained by the behavior of their neighbors. Recent research has suggested that local behavioral elements such as: (1) attraction to neighbors up to a maximum distance, (2) repulsion from neighbors if too close, and (3) alignment or velocity matching with neighbors, are often adequate to engender and sustain a group structure/formation in such natural aggregation. Further, several group-level behaviors such as flocking as well as directed group motion (such as finding a food source or moving to higher temperatures) while avoiding obstacles have been demonstrated to arise from such purely local interactions.

Potential field based formulations provide an intuitive way to model the behavior of group of robot with above characteristics. At group level, the workspace is modeled as a potential space with the target exerting an attractive potential and all obstacles exerting repulsive potentials and where the global minimum is at the target. At the individual level, the potential field is modeled such that a preferred distance is maintained. i.e., the potential field is designed such that an attractive force is exerted if the robots are too far apart and a repulsive force is exerted if they are too close. This potential field is also designed such that each individual robot is only affected by its neighboring robots up to a maximum distance. Such a potential, we notice, can be modeled as a *nonlinear virtual spring* that connects the robots and virtual leaders. Thus the group level motion planning problem may be treated as a gradient climbing problem. The resulting individual control laws can be computed explicitly as gradients of potential that can be implemented in a decentralized manner while facilitating a Lyapunov based analysis.

Thus such potential-based approaches are gaining their popularity for motion planning for multiple mobile robots moving in formation due to their seeming ease of formulation, decentralization, and scalability. However, we note that while stability guarantees may be obtained, such potential-based formation motion planning and control algorithms are unable to to guarantee strict formation maintenance. In situations such as cooperative payload transport by collectives or in distributed sensor deployment application, strict maintainance of geometric formation is critical throughout the motion. Hence we examine methods for maintenance of formations for groups of robots within the APF framework.

In our case, we note that the group of independent mobile robots moving together in geometric formation can alternatively be viewed as a constrained mechanical system. The requirements for formation maintenance may be formulated as the addition of the holonomic constraints to an unconstrained dynamic system. By doing so, we can link and leverage the extensive literature on formulation and implementation of computational simulation of multi-body systems to help address our group motion planning problem.

Thus, the overall online motion planning problem may now be treated in the context of forward dynamics simulation of a constrained mechanical system, i.e, for our case, we are interested in solving the forward-dynamics in which the input force to the system is provided using the potential field approaches, and we would like to see how the system behaves under these forces. Various methods have been proposed in the literature for formulation and solution of the forward dynamics equations of such constrained dynamics system. Three methods that will be evaluated in this thesis are: 1) *direct Lagrange multiplier elimination method*; 2) *penalty-formulation approach*; and 3) *constraint manifold projection approach*. We compare and contrast these methods on the basis of *modular formulation*, *distributed computation* and *relative computational efficiency and accuracy*. These aspects will be studied in the context of the motion-planning of a group of 3 point-mass robots which are constrained together by holonomic constraints.

The remainder of this chapter will be organized as follows: In section 6.1, we provide relevant background on *constrained dynamics* and derive the equations of motion of the constrained mechanical system in the general form. In the subsequent sections, we discuss the three methods that allow us to solve the dynamic equation numerically,

focusing specifically from the viewpoint of distributed computation. We will then compare, evaluate, and discuss these various methods for their effectiveness in maintaining group formation from different aspects using three point-mass robots forming a triangle as the case study. In chapter 6, we provide simulation result of group of robot in potential field based on the dynamics formulation in this chapter.

## 6.1 Background on Constrained Dynamics

When robots form a group with specific geometric formation, the geometric formation requirement imposes holonomic constraint on the dynamics system, creating the so called *holonomic constrained dynamics system*, studied in classical mechanic. See, for example, chapter 4 of [12] and chapter 1 of [78]. Many examples of dynamic systems coupled by holonomic constraints may be found among robotic systems. Hence, methods to solve (both forward dynamic and inverse dynamics) of such dynamic systems are studied extensively in the area of robotics [15, 19, 79, 80]. Solving such dynamics system effectively and accurately is essential for computer simulation of such system and implementation in real-time application. *Physically Based Modeling* in the area of computer animations is another place where constrained dynamics systems are also being studied extensively, examples are [13, 14, 81].

Yun *et. al* developed a unified formulations for robotic systems subjected to both holonomic and nonholonomic constraints and examined its application for single wheeled mobile manipulators [16]. We will extend and implement a similar framework to help ensure formation maintenance in a group of multiple point-mass robots. A more detailed background on constrained dynamic system can be found in Appendix A.1.

The dynamics of group or robots can be formulated as Lagrangian equations of the first kind [12] as:

$$\dot{\mathbf{q}} = \mathbf{v} \tag{6.1}$$

$$\mathbf{M}\left(\mathbf{q}\right)\dot{\mathbf{v}} = \mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) - \mathbf{J}\left(\mathbf{q}\right)^{T}\boldsymbol{\lambda} \tag{6.2}$$

$$\mathbf{C}\left(\mathbf{q},t\right) = \mathbf{0} \tag{6.3}$$

where $\mathbf{q}$ is the $n$-dimensional vector of generalized coordinates; $\mathbf{v}$ is the $n$-dimensional vector of generalized velocities; $\mathbf{M}\left(\mathbf{q}\right)$ is the $2n \times 2n$ dimensional inertia matrix; $\mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right)$ is the $n$-dimensional vector of external forces; $\mathbf{u}$ is the vector of input forces, which is $-\mathbf{k}_{f}\nabla_{\mathbf{q}}U$ ; $\mathbf{C}\left(\mathbf{q},t\right)$ is a $m$ -dimensional vector of holonomic constraints which may or may not depends on time, $t$ ; and $\mathbf{J}\left(\mathbf{q}\right) = \dfrac{\partial \mathbf{C}\left(\mathbf{q}\right)}{\partial \mathbf{q}}$ is the *Jacobian* matrix.

These equations can be solved using the converted ODE approach, where all the algebraic position and velocity level constraints are differentiated and represented at the acceleration level to obtain an augmented index-1 DAE, in terms of the unknown accelerations and the unknown multipliers. To do so, differentiating the position constraints in Equation(6.3), yield the velocity level constraints:

$$\dot{\mathbf{C}} = \mathbf{J}\left(\mathbf{q}\right)\dot{\mathbf{q}} \tag{6.4}$$

further differentiating again with respect to time, yields the acceleration level constraints:

$$\ddot{\mathbf{C}} = \mathbf{J}\left(\mathbf{q}\right)\dot{\mathbf{v}} + \dot{\mathbf{J}}\left(\mathbf{q}\right)\mathbf{v} \tag{6.5}$$

Hence, Equation(6.5) can be written together with Equation(6.2) as an index-1 DAE as:

$$\begin{bmatrix} \mathbf{M}\left(\mathbf{q}\right) & \mathbf{J}^{T}\left(\mathbf{q}\right) \\ \mathbf{J}\left(\mathbf{q}\right) & \mathbf{0} \end{bmatrix}_{(2n+m)\times(2n+m)} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix}_{(2n+m)\times 1} = \begin{bmatrix} \mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) \\ -\dot{\mathbf{J}}\left(\mathbf{q}\right)\mathbf{v} \end{bmatrix}_{(2n+m)\times 1} \tag{6.6}$$

Equation(6.6) can be solved by various methods [19]: 1) *Direct Lagrange multiplier elimination:* In classical mechanics, it is usually explicitly computing the Lagrange multiplier by a projection into the constrained force space [12, 78]; 2) *A penalty-formulation approach:* Approximating the Lagrange multiplier using artificial compliance elements such as virtual springs and dampers [15]; or 3) *A constraint manifold projection based approach:* By projecting the equations of motion onto the tangent space of the constraint manifold in a variety of ways to obtain constraint-reaction free equations of motions [16, 82]. In the following sections, we will formulate using these three methods.

## 6.2 Method I: Direct Lagrange Multiplier Elimination Approach

The direct Lagrange multiplier elimination is a centralized approach which eliminates the Lagrange multiplier directly by projecting the forces on each individual robot into a feasible region, ensures that the formation constraint is not violated throughout its motion [13, 14]. In the following, we describe the steps in constructing the constrained dynamics for general constraint system. Later, we will demonstrate how we can apply this result in constructing the constrained dynamics for group of robots

A vector of constraint functions which depends on the state $\mathbf{q}$ and possibly directly dependent on time, $t$, can be written in general form as:

$$\mathbf{C}(\mathbf{q},t)=\mathbf{0} \tag{6.7}$$

For constraint $\mathbf{C}(\mathbf{q},t)$ to remain at 0 from some initial time $t_0$, it suffices that $\mathbf{C}(t_0)=\mathbf{0}$, $\dot{\mathbf{C}}(t_0)=\mathbf{0}$, and $\ddot{\mathbf{C}}(t_0)=\mathbf{0}$. The *velocities level constraint*s are obtained by differentiating Equation (6.7), with the use of chain rule:

$$\dot{\mathbf{C}} = \frac{\partial}{\partial t}\mathbf{C}(\mathbf{q},t) = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{C}}{\partial t} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \frac{\partial \mathbf{C}}{\partial t} \tag{6.8}$$

where $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}$, which is the *Constraint Jacobian* matrix, and $\dot{\mathbf{q}} = \frac{\partial \mathbf{q}}{\partial t}$.

The acceleration level constraint, can obtained by differentiating Equation (6.8):

$$\ddot{\mathbf{C}}(\mathbf{q},t) = \frac{\partial}{\partial t}\left[\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \frac{\partial \mathbf{C}}{\partial t}\right] = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \frac{\partial^2 \mathbf{C}}{\partial t^2} \tag{6.9}$$

Next, from Equation (6.2), we obtained $\ddot{\mathbf{q}} = \mathbf{M}^{-1}\left[\mathbf{f}(\mathbf{q},\mathbf{v},t,\mathbf{u}) - \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}\right]$, which upon substitution into Equation (6.9), gives the following condition:

$$\mathbf{J}(\mathbf{q})\left[\mathbf{M}^{-1}\left(\mathbf{f}(\mathbf{q},\mathbf{v},t,\mathbf{u}) - \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}\right)\right] + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \frac{\partial^2 \mathbf{C}}{\partial t^2} = \mathbf{0} \tag{6.10}$$

Equation (6.10) is a system of linear equations with only the constraint force vector $\boldsymbol{\lambda}$ unknown. Solving for $\boldsymbol{\lambda}$. With some arrangement, Equation (6.10) becomes:

$$\left[\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{q})\right]\boldsymbol{\lambda} = \mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\mathbf{f}(\mathbf{q},\mathbf{v},t,\mathbf{u}) + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \frac{\partial^2 \mathbf{C}}{\partial t^2} \tag{6.11}$$

The entire right hand side of Equation (6.11) is known, and the matrix on the left hand side, the product of the constraint Jacobian, the inverse mass matrix, and the Jacobian transpose, is a square matrix with the dimensions of the constraints. Once Equation (6.11) is solved, the constraint dynamic equation of the system can then be computed by Equation (6.2).

Having derived the constrained dynamics for general case in the previous section, we would like to now write the constraint dynamics in state-space form. Since most of the control system theories are developed with state-space formulation, it would be an advantage to write the constraint dynamics in state-space form. This allows us to apply the state space analysis and design techniques to the control of the constrained systems.

The state-space equation obtained using Equation (6.11) and Equation (6.2) can be written as:

$$\begin{bmatrix} \dot{\mathbf{q}}_{2n\times 1} \\ \ddot{\mathbf{q}}_{2n\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\left[\mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) - \mathbf{J}^{T}\boldsymbol{\lambda}\right] \end{bmatrix} \tag{6.12}$$

where: $\boldsymbol{\lambda} = \left[\mathbf{J}\left(\mathbf{q}\right)\mathbf{M}^{-1}\mathbf{J}^{\mathbf{T}}\left(\mathbf{q}\right)\right]^{-1}\left[\mathbf{J}\left(\mathbf{q}\right)\mathbf{M}^{-1}\mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) + \dot{\mathbf{J}}\left(\mathbf{q}\right)\dot{\mathbf{q}} + \frac{\partial^{2}\mathbf{C}}{\partial t^{2}}\right]$

## 6.3 Method II: Penalty-Formulation Approach

In the penalty-formulation approach, the holonomic constraints are relaxed and replaced by linear/non-linear spring with dampers thereby incorporating the constraint equations as a dynamical system penalized by a large factor. Here, the Lagrange multipliers are explicitly approximated as the force of a virtual spring or damper based on the extent of the constraint violation and assumed spring stiffness and damping constant. By doing this, the Lagrange multiplier is eliminated from the list of $n+m$ unknowns, leaving a system of $2n$ first order ODEs. On one hand, we note that this may creates a stiff dynamic equation with poor numerical conditioning, when a large penalty factor is selected. On the other hand, this spring force only approximates the true value of the constraint forces, may not be able to maintain strict formation requirement, if a relative small penalty factor is selected.

The restoring force, which is proportional to the extent of the constraint violation, is expressed as:

$$\boldsymbol{\lambda} = \mathbf{K}_{S}\mathbf{C}\left(\mathbf{q}\right) + \mathbf{K}_{D}\dot{\mathbf{C}}\left(\mathbf{q}\right) \tag{6.13}$$

where $\mathbf{K}_S$ is the spring constant, $\mathbf{K}_S = diag\{K_{S1}, K_{S2}, \cdots, K_{Sn}\}$, an $n \times n$ diagonal matrix; $\mathbf{K}_D$ is the damping constant, where $\mathbf{K}_D = diag\{K_{D1}, K_{D2}, \cdots, K_{Dn}\}$, also an $n \times n$ diagonal matrix; and $\mathbf{C}(\mathbf{q})$ is the vector of constraint violation in the direction of the respective $\boldsymbol{\lambda}$.

As we mentioned in the beginning of this chapter, potential field based approach in maintaining formation is essentially placing a virtual spring between robots and virtual leaders. An example of three point-mass robot formulated using penalty based approach can be view replacing the constraints between each robot with a virtual spring and virtual damper, as shown in Figure 6-1. In [3, 4, 31, 66], a nonlinear spring without the damper model is used to maintain the formation.

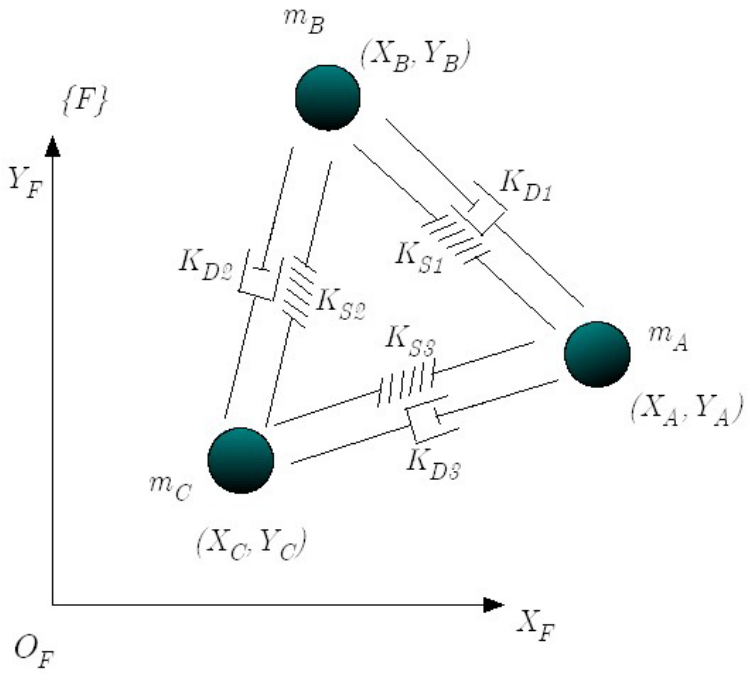



**Figure 6-1: Penalty based approach is equivalent to placing virtual spring and damper between each robot to maintain formation, showed here is three point mass robots interconnected by spring and dampers.**

By substiting Equation (6.13) into Equation (6.2), the dynamic equation using penalty-formulation can be expressed as:

$$\begin{bmatrix} \dot{\mathbf{q}}_{2n\times 1} \\ \ddot{\mathbf{q}}_{2n\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\left[\mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) - \mathbf{J}^{T}\left(\mathbf{K}_{S}\mathbf{C}\left(\mathbf{q}\right) + \mathbf{K}_{D}\dot{\mathbf{C}}\left(\mathbf{q}\right)\right)\right] \end{bmatrix} \tag{6.14}$$

Note that with this formulation, we can also have the shape change through letting:

$$\boldsymbol{\lambda} = \mathbf{K}_{S}\mathbf{C}\left(\mathbf{q},t\right) + \mathbf{K}_{D}\dot{\mathbf{C}}\left(\mathbf{q},t\right) \tag{6.15}$$

where the constraint matrix $\mathbf{C}$ is now $\mathbf{C}\left(\mathbf{q},t\right)$, a function of both position and time.

## 6.4 Method III: Constraint Manifold Projection Approach

NO !

In this approach, the dynamic equation with constraint-reactions is separated into the tangent and orthogonal-complement cotangent subspace. The holonomic constraint $\mathbf{C}\left(\mathbf{q},t\right) = 0$, can be written in differential form as:

$$\left[\frac{\partial \mathbf{C}\left(\mathbf{q},t\right)}{\partial \mathbf{q}}\right]\dot{\mathbf{q}} + \left[\frac{\partial C\left(\mathbf{q},t\right)}{\partial t}\right] = \mathbf{0} \tag{6.16}$$

or simply:

$$\mathbf{J}\left(\mathbf{q}\right)\dot{\mathbf{q}} = \mathbf{a}\left(\mathbf{q}\right) \tag{6.17}$$

where $\mathbf{J}\left(\mathbf{q}\right)$ is the Jacobian of $\mathbf{C}\left(\mathbf{q}\right)$, and $\mathbf{a}\left(\mathbf{q}\right)$ is the time differentiation of the constraint $\mathbf{C}\left(\mathbf{q}\right)$ (However, after adding Baumgarte stabilization technique, $\mathbf{a}\left(\mathbf{q}\right)$ will include the stabilization term as well. See section 6.5). Let $\mathbf{S}\left(\mathbf{q}\right)$ be a $n\times\left(n-m\right)$ dimensional full rank matrix whose column space is in the null space of $\mathbf{A}\left(\mathbf{q}\right)$, i.e. $\mathbf{A}\left(\mathbf{q}\right)\mathbf{S}\left(\mathbf{q}\right) = \mathbf{0}$. The orthogonal subspace is spanned by the so-called constraint vectors

that form the rows of matrix $\mathbf{A}(\mathbf{q})$ while the tangent subspace complements this orthogonal subspace in the overall generalized velocity vector space. All feasible velocities, $\dot{\mathbf{q}}$, of a constrained multibody system that belong to this tangent space, appropriately called the space of feasible motions. This space is parameterized by an $n-m$ dimensional vector of independent velocities as:

$$\dot{\mathbf{q}} = \mathbf{v} = \mathbf{S}(\mathbf{q})\boldsymbol{\upsilon}(t) + \boldsymbol{\eta}(\mathbf{q}) \tag{6.18}$$

where $\boldsymbol{\eta}(\mathbf{q})$ is the particular solution of Equation (6.17). Differentiating Equation (6.18) further:

$$\begin{aligned} \dot{\mathbf{v}} &= \mathbf{S}(\mathbf{q})\dot{\boldsymbol{\upsilon}}(t) + \dot{\mathbf{S}}(\mathbf{q})\boldsymbol{\upsilon}(t) + \dot{\boldsymbol{\eta}}(\mathbf{q}) \\ &= \mathbf{S}(\mathbf{q})\dot{\boldsymbol{\upsilon}}(t) + \boldsymbol{\gamma}(\mathbf{q},\boldsymbol{\upsilon}) \end{aligned} \tag{6.19}$$

where $\boldsymbol{\gamma}(\mathbf{q},\boldsymbol{\upsilon}) = \dot{\mathbf{S}}(\mathbf{q})\boldsymbol{\upsilon}(t) + \dot{\boldsymbol{\eta}}(\mathbf{q})$.

Such a projection process works well in a Riemanian setting where the notion of orthogonal complement subspace exists. However, special care needs to be exercised when treating configuration spaces such as $SE(2)$ or $SE(3)$. A family of projection exists depending on the selection of dependent/independent velocities. Also, once the projection is selected, the dynamic equations of motion can now be projected on to the instantaneous feasible motion directions, to obtain the so-called constraint-reaction-free equation of motion. Pre-multiply both sides of Equation (6.2) by $\mathbf{S}^{\mathbf{T}}$ and noting that $\mathbf{S}^{\mathbf{T}}\mathbf{J}^{\mathbf{T}} = \mathbf{0}$, we get:

$$\mathbf{S}^{\mathbf{T}}\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{S}^{\mathbf{T}}\mathbf{f}(\mathbf{q},\mathbf{v},t,\mathbf{u}) \tag{6.20}$$

By substituting Equation (6.19) into Equation (6.20) and solve for $\dot{\boldsymbol{\upsilon}}$:

$$\dot{\boldsymbol{\upsilon}} = \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left(\mathbf{S}^{\mathbf{T}}\mathbf{f}(\mathbf{q},\mathbf{v},t,\mathbf{u}) - \mathbf{S}^{\mathbf{T}}\mathbf{M}\boldsymbol{\gamma}\right) \tag{6.21}$$

And the overall system of ODEs can be expressed in state-space form as:

$$\begin{bmatrix} \dot{\mathbf{q}}_{2n\times 1} \\ \dot{\boldsymbol{\upsilon}}_{(2n-m)\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{S}\boldsymbol{\upsilon} + \boldsymbol{\eta} \\ \left(\mathbf{S}^{\mathrm{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left(\mathbf{S}^{\mathrm{T}}\mathbf{f}\left(\mathbf{q},\mathbf{v},t,\mathbf{u}\right) - \mathbf{S}^{\mathrm{T}}\mathbf{M}\boldsymbol{\gamma}\right) \end{bmatrix} \tag{6.22}$$

To numerically calculate all the terms in Equation (6.22), we adopted the method used in [16].

## 6.5 Baumgarte Stabilization NO !

As mentioned in [16], the drawbacks of the converted ODE approach include: (1) the need to provide additional consistent initial conditions. The initial condition must satisfy $\mathbf{C}(\mathbf{q}) = 0$, such that the constraints can be maintained throughout the simulation; and (2) the mild stability of the differentiated constraints resulting in state-drift from the position-level constraint manifold. While the growth rate can be reduced by lowering the error tolerance and by using smaller step-sizes or greater numerical precision, this comes at the cost of higher and more expensive computations. To circumvent these problems, we adopted the Baumgarte stabilization method [83].

Baumgarte stabilization method involves the creation of an artificial first or second order dynamical system which has the algebraic position-level constraint as its attractive equilibrium configuration. Using a first order Baumgarte stabilization technique, the holonomic constraint of Equation (6.3) is replaced with:

$$\dot{\mathbf{C}}(\mathbf{q}) + \sigma\mathbf{C}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \sigma\mathbf{C}(\mathbf{q}) = \mathbf{0}, \qquad \sigma > 0 \tag{6.23}$$

where $\sigma$ is the rate of convergence. The equilibrium condition for this first order system is the constraint manifold $\mathbf{C}(\mathbf{q}) = \mathbf{0}$, and also for any initial condition $\mathbf{q}(t)$, which may or may not satisfy the constraint Equation (6.3). Equation (6.23) also guarantees exponential

convergence that will be determined by $\sigma$, which can be chosen arbitrarily based on applications. The solution of Equation (6.23) is given by:

$$\mathbf{C}(t) = \mathbf{C}_0 e^{-\sigma t} \tag{6.24}$$

For any positive $\sigma$, $\mathbf{C}(t)$ goes to zero as the time $t$ goes to infinity. Therefore, even starting from an initial condition that does not satisfy holonomic constraints, the use of Equation (6.23) ensures that holonomic constraints will eventually be satisfied. This also holds true if the holonomic constraints are initially satisfied but are violated some time later. With this in mind, the holonomic constraint mechanical system that we described in Equation (6.2) and Equation (6.3) is now modified by changing Equation (6.3) with:

$$\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \sigma\mathbf{C}(\mathbf{q}) = \mathbf{0} \tag{6.25}$$

This method may be explained in terms of manifolds. For a given initial condition $\mathbf{q}(\mathbf{0}), \dot{\mathbf{q}}(\mathbf{0})$ such that $\mathbf{J}(\mathbf{q}(\mathbf{0}))\dot{\mathbf{q}}(\mathbf{0}) = \mathbf{0}$, the trajectory belongs to the manifold defined by $\mathbf{C}(\mathbf{q}) = \mathbf{C}(\mathbf{q}(\mathbf{0}))$. Replacing the holonomic constraints with Equation (6.25), a particular manifold, defined by $\mathbf{C}(\mathbf{q}) = \mathbf{0}$, is made invariant and attractive. For any initial condition, this method generates a trajectory that does not remain on one manifold but crosses different manifolds, and exponentially converges to this particular manifold. The Baumgarte stabilization method is very popular in the engineering application community, principally due to the resulting augmented ODE formulation, the practical selection of the parameters of the stabilization system depends both on the discretization methods and step-size and is widely regarded as an open research problem [8].

The state space representation with this modified constraint can be derived using the procedures outlined in previous sections. Instead of presenting the modified procedure here, it is derived in the next section, where a 3 point-mass robot model is studied.

## 6.6 Case Study: A Triangular Formation of 3 Point Mass Robots

To show the steps in formulating the equation of motion using the three methods that we discussed in the previous sections, we formulate the state-space equation using all three methods with this case study. In this example, we consider a robot collective formed by 3 robots $(n=3)$, each with point-mass $m_i$ operating in a horizontal plane with a configuration of each robot given by $\mathbf{q}_i = [x_i, y_i]^T \in \mathrm{R}^2,\ \forall i = A, B, C$ w.r.t. the inertial frame $\{F\}$, as shown in Figure 6-2:

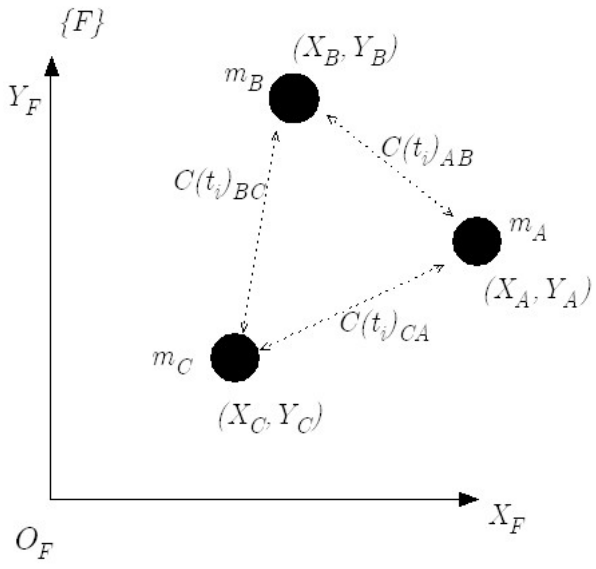



**Figure 6-2: Schematic diagram of the 3 point-mass robots used in the case study.**

These three robots are required to maintain the formation throughout their journey, thus the constraint matrix is only a function of their positions, $\mathbf{C} = \mathbf{C}(\mathbf{q})$. The governing

equation of motion for each of the point-mass robot take the form of $\mathbf{M}_i(\mathbf{q})\ddot{\mathbf{q}}_i + \mathbf{k}\dot{\mathbf{q}}_i = \mathbf{u}_i$, where $\mathbf{M}_i = diag\{m_i\},\ \forall i = A,B,C$. Here, the three robots are each denoted as robot $A$, robot $B$, and robot $C$. The governing equations of motion for this collective of robots moving in formation can be written in an index-3 DAE form as:

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{v} \\ \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) &= \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{J}^{\mathbf{T}}\boldsymbol{\lambda} \\ \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \sigma\mathbf{C}(\mathbf{q}) &= \mathbf{0} \end{aligned} \tag{6.26}$$

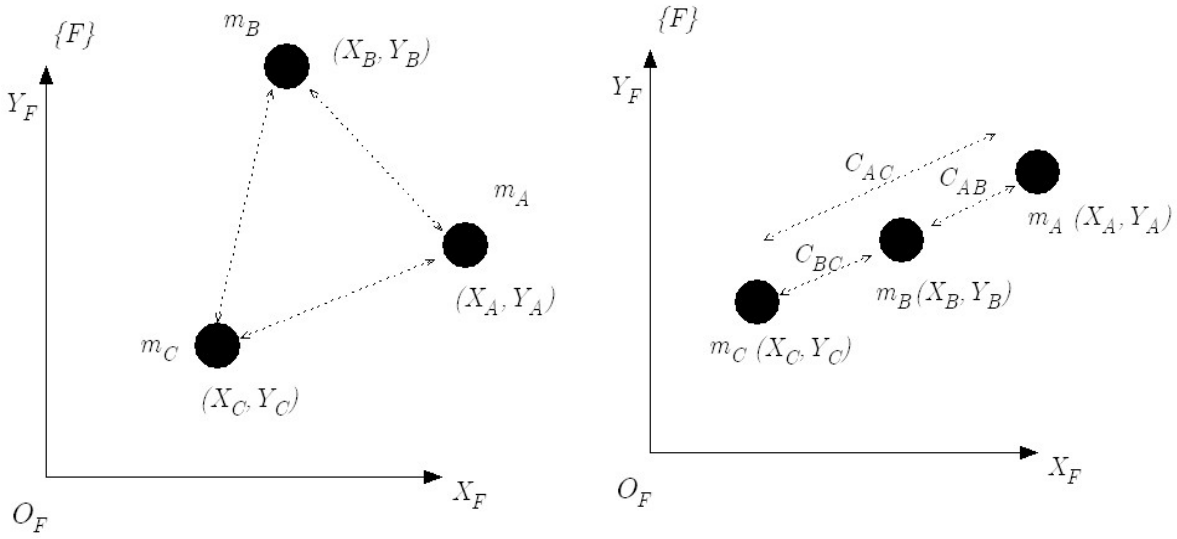
where:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} [\mathbf{M_A}]_{2\times2} & 0 & 0 \\ 0 & [\mathbf{M_B}]_{2\times2} & 0 \\ 0 & 0 & [\mathbf{M_C}]_{2\times2} \end{bmatrix} \qquad \mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{K}\dot{\mathbf{q}} \qquad \mathbf{G}(\mathbf{q}) = \mathbf{0}$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_B \\ \mathbf{q}_B \end{bmatrix} \qquad \mathbf{u} = -\mathbf{K}_f\left(\nabla_{\mathbf{q}}U\right)^T \qquad \mathbf{E}(\mathbf{q}) = \mathbf{I}_{6\times6}$$

where $\mathbf{q} = [x_A, y_A, x_B, y_B, x_C, y_C]^T$, the position of each robot; $\mathbf{M}(\mathbf{q})$ is the mass matrix of the system; $\mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{K}\dot{\mathbf{q}}$ is the dissipative term added to stabilize the system; $\mathbf{u} = -\mathbf{K}_f\left(\nabla_{\mathbf{q}}U\right)^T$ is the input to the system, where the positive scaling diagonal matrix $\mathbf{K}_f$ and in the direction pointed to the negative gradient of the potential field in the workspace.

One interesting question arises when we consider the different possible formations that are available for a number of 3 robots, see Figure 6-3.




(a) (b)

**Figure 6-3: Two possible formations are available for a group of 3 robots: (a) three robots arranged in a straight line; and (b) three robots arranged in triangular shape.**

The different arrangement of the robot resulted in different shapes of the formation but will not change the numbers of constraints available to the system if a *rigid formation* is concerned. This scenario will be discussed in section 6.7. Here, we only consider the case where *rigid formation* formed by the 3 robots. With this requirements in mind, the number of constraint equations is given by $2n-3$, where $n$ is the number of robots [84]. As shown in Figure 6-2, the constraint equations that maintain the rigidity from the requirement the each robot will maintain a desired distance with other robots is given by:

$$C(q)=\begin{cases}(x_A-x_B)^2+(y_A-y_B)^2-c_{AB}^2=0\\(x_B-x_C)^2+(y_B-y_C)^2-c_{BC}^2=0\\(x_C-x_A)^2+(y_C-y_A)^2-c_{CA}^2=0\end{cases} \tag{6.27}$$

where $c_{ij}$ denote the Euclidean distance between $i$ and $j$, write in matrix form:

$$\mathbf{C}(\mathbf{q})=\begin{bmatrix}(x_A-x_B)^2+(y_A-y_B)^2-c_{AB}^2\\(x_B-x_C)^2+(y_B-y_C)^2-c_{BC}^2\\(x_C-x_A)^2+(y_C-y_A)^2-c_{CA}^2\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix} \tag{6.28}$$

Differentiated, with the use of Chain rule:

$$\dot{\mathbf{C}}(\mathbf{q}) = \frac{d}{dt}\left(\mathbf{C}(\mathbf{q})\right) = \frac{\partial \mathbf{C}(\mathbf{q})}{\partial \mathbf{q}}\left(\frac{\partial \mathbf{q}}{\partial t}\right) = \mathbf{0} \tag{6.29}$$

we have:

$$\frac{\partial \mathbf{C}(\mathbf{q})}{\partial \mathbf{q}} = \mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial C_1(\mathbf{q})}{\partial x_A} & \frac{\partial C_1(\mathbf{q})}{\partial y_A} & \frac{\partial C_1(\mathbf{q})}{\partial x_B} & \frac{\partial C_1(\mathbf{q})}{\partial y_B} & \frac{\partial C_1(\mathbf{q})}{\partial x_C} & \frac{\partial C_1(\mathbf{q})}{\partial y_C} \\ \frac{\partial C_2(\mathbf{q})}{\partial x_A} & \frac{\partial C_2(\mathbf{q})}{\partial y_A} & \frac{\partial C_2(\mathbf{q})}{\partial x_B} & \frac{\partial C_2(\mathbf{q})}{\partial y_B} & \frac{\partial C_2(\mathbf{q})}{\partial x_C} & \frac{\partial C_2(\mathbf{q})}{\partial y_C} \\ \frac{\partial C_3(\mathbf{q})}{\partial x_A} & \frac{\partial C_3(\mathbf{q})}{\partial y_A} & \frac{\partial C_3(\mathbf{q})}{\partial x_B} & \frac{\partial C_3(\mathbf{q})}{\partial y_B} & \frac{\partial C_3(\mathbf{q})}{\partial x_C} & \frac{\partial C_3(\mathbf{q})}{\partial y_C} \end{bmatrix} \tag{6.30}$$

This yield:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} (x_A - x_B) & (y_A - y_B) & -(x_A - x_B) & -(y_A - y_B) & 0 & 0 \\ 0 & 0 & (x_B - x_C) & (y_B - y_C) & -(x_B - x_C) & -(y_B - y_C) \\ -(x_C - x_A) & -(y_C - y_A) & 0 & 0 & (x_C - x_a) & (y_C - y_A) \end{bmatrix} \tag{6.31}$$

Or, we can write partitioned it into:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_A(\mathbf{q}) & \mathbf{J}_B(\mathbf{q}) & \mathbf{J}_C(\mathbf{q}) \end{bmatrix} \tag{6.32}$$

where:

$$\mathbf{J_A}(\mathbf{q}) = \begin{bmatrix} (x_A - x_B) & (y_A - y_B) \\ 0 & 0 \\ -(x_C - x_A) & -(y_C - y_A) \end{bmatrix} \quad \mathbf{J_B}(\mathbf{q}) = \begin{bmatrix} -(x_A - x_B) & -(y_A - y_B) \\ (x_B - x_C) & (y_B - y_C) \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{J_C}(\mathbf{q}) = \begin{bmatrix} 0 & 0 \\ -(x_B - x_C) & -(y_B - y_C) \\ (x_C - x_a) & (y_C - y_A) \end{bmatrix}$$

In the following sections, we derived the equation of motion using the three different methods that we described earlier. Whenever possible, we formulated the equation of motion from the decentralization point of view.

### 6.6.1 Method I: Direct Lagrange Multiplier Elimination Approach

As shown in section 6.2, we need the second derivative of the holonomic constraint $\mathbf{C}(\mathbf{q})$:

$$\ddot{\mathbf{C}}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} = 0 \tag{6.33}$$

However, in Equation (6.33), $\dot{\mathbf{J}}(\mathbf{q}) = \dfrac{\partial \mathbf{J}}{\partial \mathbf{q}}\dot{\mathbf{q}}$ and taking the derivative of a matrix with respect to a vector yield rank 3 tensors. We can avoid this by writing equivalently:

$$\dot{\mathbf{J}} = \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{q}}$$

where:

$$\dot{\mathbf{C}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} (x_A - x_B)\dot{x}_A + (y_A - y_B)\dot{y}_A - (x_A - x_B)\dot{x}_B - (y_A - y_B)\dot{y}_B \\ (x_B - x_C)\dot{x}_B + (y_B - y_C)\dot{y}_B - (x_B - x_C)\dot{x}_C - (y_B - y_C)\dot{y}_C \\ (x_C - x_a)\dot{x}_C + (y_C - y_A)\dot{y}_C - (x_C - x_A)\dot{x}_A - (y_C - y_A)\dot{y}_A \end{bmatrix} \tag{6.34}$$

and $\dot{\mathbf{J}}$ become:

$$\dot{\mathbf{J}} = \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{q}} = \begin{bmatrix} (\dot{x}_A - \dot{x}_B) & (\dot{y}_A - \dot{y}_B) & (\dot{x}_B - \dot{x}_A) & (\dot{y}_B - \dot{y}_A) & 0 & 0 \\ 0 & 0 & (\dot{x}_B - \dot{x}_C) & (\dot{y}_B - \dot{y}_C) & (\dot{x}_C - \dot{x}_B) & (\dot{y}_C - \dot{y}_B) \\ (\dot{x}_A - \dot{x}_C) & (\dot{y}_A - \dot{y}_C) & 0 & 0 & (\dot{x}_C - \dot{x}_A) & (\dot{y}_C - \dot{y}_A) \end{bmatrix} \tag{6.35}$$

Also, instead of letting $\ddot{\mathbf{C}}(\mathbf{q}) = \mathbf{0}$, we apply the Baumgarte stabilization method (second order) to prevent numerical drift, such that:

$$\ddot{\mathbf{C}}(\mathbf{q}) = -\sigma \mathbf{C}(\mathbf{q}) - \beta \dot{\mathbf{C}}(\mathbf{q}) \tag{6.36}$$

As a result, the acceleration level constraints, modified from Equation (6.9), now become:

$$\ddot{\mathbf{C}}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} = -\sigma \mathbf{C}(\mathbf{q}) - \beta \dot{\mathbf{C}}(\mathbf{q}) \tag{6.37}$$

From Equation (6.26), we have $\ddot{\mathbf{q}} = \mathbf{M}^{-1}\left[\mathbf{u} - \mathbf{K}\dot{\mathbf{q}} - \mathbf{J}^{\mathbf{T}}(\mathbf{q})\boldsymbol{\lambda}\right]$, substitute in Equation (6.37), we obtained:

$$\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\left[\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}-\mathbf{J}^{\mathrm{T}}(\mathbf{q})\boldsymbol{\lambda}\right]+\dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}=-\sigma\mathbf{C}(\mathbf{q})-\beta\dot{\mathbf{C}}(\mathbf{q}) \tag{6.38}$$

Solve for $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}=\left[\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{q})\right]^{-1}\left[\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\left(\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}\right)+\dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}+\beta\dot{\mathbf{C}}(\mathbf{q})+\sigma\mathbf{C}(\mathbf{q})\right] \tag{6.39}$$

Finally, substitute Equation (6.39) back into Equation (6.26), we obtained the dynamic equation in the state-space form using direct Lagrange multiplier elimination method as:

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \ddot{\mathbf{q}}\end{bmatrix}=\begin{bmatrix}\mathbf{v}\\ \mathbf{M}^{-1}\left[\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}-\mathbf{J}^{\mathrm{T}}(\mathbf{q})\boldsymbol{\lambda}\right]\end{bmatrix} \tag{6.40}$$

where:

$\boldsymbol{\lambda}=\left[\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{q})\right]^{-1}\left[\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}\left(\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}\right)+\dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}+\beta\dot{\mathbf{C}}(\mathbf{q})+\sigma\mathbf{C}(\mathbf{q})\right]$ and

$\mathbf{u}=-\mathbf{K}_f\nabla_{\mathbf{q}}U$

Note that this method does not offer any significant potential for distributed computation and we do not make any such attempts.

### 6.6.2 Method II: Penalty Formulation Approach

The governing equation for the robot collective maintaining formation using artificial potential based on the relative distance [31] are the same as those resulting from the penalty formulation that we derived in this section for the forward dynamics simulation. The formulations that we derived in this section are based on the view point of distributing the motion-planning computations between the multiple agents. In this method, the $\boldsymbol{\lambda}$ in Equation (6.26) is replaced by the constraints expressed in Equation (6.13). Also note that by doing this, we also incorporated the Baumgarte stabilization

method in the equation. The centralized formulation of the penalty formulation approach is modified from Equation (6.14):

$$\begin{bmatrix} \dot{\mathbf{q}}_{2n\times 1} \\ \ddot{\mathbf{q}}_{2n\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\left[\mathbf{u} - \mathbf{K}\dot{\mathbf{q}} - \mathbf{J}^T(\mathbf{q})\left(\mathbf{K}_S\mathbf{C}(\mathbf{q}) + \mathbf{K}_D\dot{\mathbf{C}}(\mathbf{q})\right)\right] \end{bmatrix} \tag{6.41}$$

Note that the state vector $\mathbf{q} = \left[\mathbf{q}_A^T, \mathbf{q}_B^T, \mathbf{q}_C^T\right]^T$ has the state variables that belong to robot A, B, and C. The distributed model of each of respective robot may be obtained in the state-space form as:

$$\begin{aligned} \begin{bmatrix} \dot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_A \end{bmatrix}_{4\times 1} &= \begin{bmatrix} \mathbf{v}_A \\ \mathbf{M}_A^{-1}\left[\mathbf{E}_A\mathbf{u}_A - \mathbf{K}_A\dot{\mathbf{q}}_A - \mathbf{J}_A^T\left(\mathbf{K}_{S,A}\mathbf{C} + \mathbf{K}_{D,A}\dot{\mathbf{C}}_A\right)\right] \end{bmatrix} \\ \begin{bmatrix} \dot{\mathbf{q}}_B \\ \ddot{\mathbf{q}}_B \end{bmatrix}_{4\times 1} &= \begin{bmatrix} \mathbf{v}_B \\ \mathbf{M}_B^{-1}\left[\mathbf{E}_B\mathbf{u}_B - \mathbf{K}_B\dot{\mathbf{q}}_B - \mathbf{J}_B^T\left(\mathbf{K}_{S,B}\mathbf{C} + \mathbf{K}_{D,B}\dot{\mathbf{C}}_B\right)\right] \end{bmatrix} \\ \begin{bmatrix} \dot{\mathbf{q}}_C \\ \ddot{\mathbf{q}}_C \end{bmatrix}_{4\times 1} &= \begin{bmatrix} \mathbf{v}_C \\ \mathbf{M}_C^{-1}\left[\mathbf{E}_C\mathbf{u}_C - \mathbf{K}_C\dot{\mathbf{q}}_C - \mathbf{J}_C^T\left(\mathbf{K}_{S,C}\mathbf{C} + \mathbf{K}_{D,C}\dot{\mathbf{C}}_C\right)\right] \end{bmatrix} \end{aligned} \tag{6.42}$$

where $\mathbf{u}_i = \nabla_{q_i} U$ , $\dot{\mathbf{C}}_i = \left[\mathbf{J}_i\right]\left[\mathbf{q}_i\right]$ , and $K_{S,i}$, $K_{D,i}$ with $i = A, B, C$ are the compliance matrices for springs and dampers respectively.

The three dynamic sub-systems shown in Equation (6.42), can be simulated in a distributed manner if at every time step: (1) either the information pertaining to $\mathbf{C}(\mathbf{q})$, the extend of the constraint violation, is made available explicitly or (2) computed by exchanging information between the robots. The sole coupling between the three sub-parts is due to the Lagrange multipliers, which are now explicitly calculated using the virtual spring.

### 6.6.3 Method III: Constraint Manifold Projection Approach NO !

In constraint manifold projection approach, the first step we need to do is to determine the $\mathbf{S}(\mathbf{q})$ matrix. The size of the $\mathbf{S}(\mathbf{q})$ matrix is $2n \times (2n-m)$, where $n$ is the number of robots, and $m$ is the number of constraints equations. In this case, we have 3 robots and 3 constraint equations, thus the number of independent velocities is $(2n-m)$, which is 3. Let $\dot{\mathbf{q}}_{ind} = [\dot{x}_1, \dot{y}_1, \dot{x}_2]^T$ be the independent velocities, and $\dot{\mathbf{q}}_d = [\dot{y}_2, \dot{x}_3, \dot{y}_3]^T$ be the dependent velocities. The constraint equation of (6.26) can be written in the following form:

$$\begin{bmatrix} \mathbf{J}_{ind}(\mathbf{q}) & \mathbf{J}_d(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{ind} \\ \dot{\mathbf{q}}_d \end{bmatrix} + \sigma \mathbf{C}(\mathbf{q}) = \mathbf{0} \tag{6.43}$$

where $\dot{\mathbf{q}}_{ind}$ denoted the independent velocities; $\dot{\mathbf{q}}_d$ denoted the dependent velocities, $\mathbf{J}_{ind}$ and $\mathbf{J}_d$ denoted the Jacobian that belongs to the independent and dependent velocities, respectively. Write the dependent velocities in terms of the independent velocities, we have:

$$\dot{\mathbf{q}}_d = \mathbf{J}_d^{-1} \left[ -\mathbf{J}_{ind}(\mathbf{q}) \dot{\mathbf{q}}_{ind}(\mathbf{q}) - \sigma \mathbf{C}(\mathbf{q}) \right] \tag{6.44}$$

and expressed in terms of $\dot{\mathbf{q}}$ as the standard form of Equation (6.18), we obtained:

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q}) \boldsymbol{\upsilon}(t) + \boldsymbol{\eta}(\mathbf{q}) \tag{6.45}$$

where:

$$\boldsymbol{\upsilon}(t) = \mathbf{q}_{ind}(t) \qquad \mathbf{S}(\mathbf{q}) = \begin{bmatrix} \mathbf{I}_{(2n-m)\times(2n-m)} \\ -\mathbf{J}_d^{-1}(\mathbf{q}) \mathbf{J}_{ind}(\mathbf{q}) \end{bmatrix} \qquad \boldsymbol{\eta}(t) = \begin{bmatrix} \mathbf{0}_{(2n-m)\times 1} \\ -\sigma \mathbf{J}_d^{-1}(\mathbf{q}) \mathbf{C}(\mathbf{q}) \end{bmatrix}$$

Proceed with steps that shown in section 6.3, we get the centralized dynamics equation as:

$$\begin{bmatrix} \dot{\mathbf{q}}_{2n\times 1} \\ \dot{\boldsymbol{\upsilon}}_{(2n-m)\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{S}\boldsymbol{\upsilon} + \boldsymbol{\eta} \\ \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left(\mathbf{S}^{\mathbf{T}}\mathbf{u} - \mathbf{S}^{\mathbf{T}}\mathbf{M}\boldsymbol{\gamma} - \mathbf{K}\mathbf{S}^{\mathbf{T}}\dot{\mathbf{q}}\right) \end{bmatrix} \tag{6.46}$$

To formulate the decentralized dynamics equations, note that the projected dynamics equations may be partitioned in the following manner:

$$\begin{aligned} &\begin{bmatrix} \mathbf{S}_A^T & \mathbf{S}_B^T & \mathbf{S}_C^T \end{bmatrix} \begin{bmatrix} \mathbf{M}_A & 0 & 0 \\ 0 & \mathbf{M}_B & 0 \\ 0 & 0 & \mathbf{M}_C \end{bmatrix} \begin{bmatrix} \mathbf{S}_A^T \\ \mathbf{S}_B^T \\ \mathbf{S}_C^T \end{bmatrix} + \begin{bmatrix} \mathbf{S}_A^T & \mathbf{S}_B^T & \mathbf{S}_C^T \end{bmatrix} \begin{bmatrix} \mathbf{M}_A & 0 & 0 \\ 0 & \mathbf{M}_B & 0 \\ 0 & 0 & \mathbf{M}_C \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma}_A^T \\ \boldsymbol{\gamma}_B^T \\ \boldsymbol{\gamma}_C^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}_A^T & \mathbf{S}_B^T & \mathbf{S}_C^T \end{bmatrix} \begin{bmatrix} \mathbf{I}_{2n\times 2n} \end{bmatrix} \begin{bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \\ \mathbf{u}_C \end{bmatrix} - \begin{bmatrix} \mathbf{S}_A^T & \mathbf{S}_B^T & \mathbf{S}_C^T \end{bmatrix} \begin{bmatrix} k_A\mathbf{I}_{2\times 2} & 0 & 0 \\ 0 & k_B\mathbf{I}_{2\times 2} & 0 \\ 0 & 0 & k_C\mathbf{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_B \\ \mathbf{q}_C \end{bmatrix} \end{aligned} \tag{6.47}$$

Rearranging the terms, we get:

$$\begin{aligned} &\left[\mathbf{S}_A^T\mathbf{M}_A\mathbf{S}_A + \mathbf{S}_B^T\mathbf{M}_B\mathbf{S}_B + \mathbf{S}_C^T\mathbf{M}_C\mathbf{S}_C\right]\dot{\boldsymbol{\upsilon}} = \\ &\quad -\left[\mathbf{S}_A^T\mathbf{M}_A\boldsymbol{\gamma}_A + \mathbf{S}_B^T\mathbf{M}_B\boldsymbol{\gamma}_B + \mathbf{S}_C^T\mathbf{M}_C\boldsymbol{\gamma}_C\right] \\ &\quad +\left[\mathbf{S}_A^T\mathbf{u}_A + \mathbf{S}_B^T\mathbf{u}_B + \mathbf{S}_C^T\mathbf{u}_C\right] \\ &\quad -\left[\mathbf{S}_A^T k_A\mathbf{I}_{2\times 2}\mathbf{q}_A + \mathbf{S}_B^T k_B\mathbf{I}_{2\times 2}\mathbf{q}_B + \mathbf{S}_C^T k_C\mathbf{I}_{2\times 2}\mathbf{q}_C\right] \end{aligned} \tag{6.48}$$

Note that the coefficient of $\dot{\boldsymbol{\upsilon}}$ in Equation (6.48) is $\left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)$ and collecting terms, we get:

$$\begin{aligned} \dot{\boldsymbol{\upsilon}} &= \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_A^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_A - \mathbf{M}_A\boldsymbol{\gamma}_A - k_A\mathbf{I}_{2\times 2}\mathbf{q}_A\right)\right] \\ &\quad +\left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_B^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_B - \mathbf{M}_B\boldsymbol{\gamma}_B - k_B\mathbf{I}_{2\times 2}\mathbf{q}_B\right)\right] \\ &\quad +\left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_C^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_C - \mathbf{M}_C\boldsymbol{\gamma}_C - k_C\mathbf{I}_{2\times 2}\mathbf{q}_C\right)\right] \end{aligned} \tag{6.49}$$

Thus, it is now possible to distribute the computation among the robots with following dynamic equations distributed to each robot:

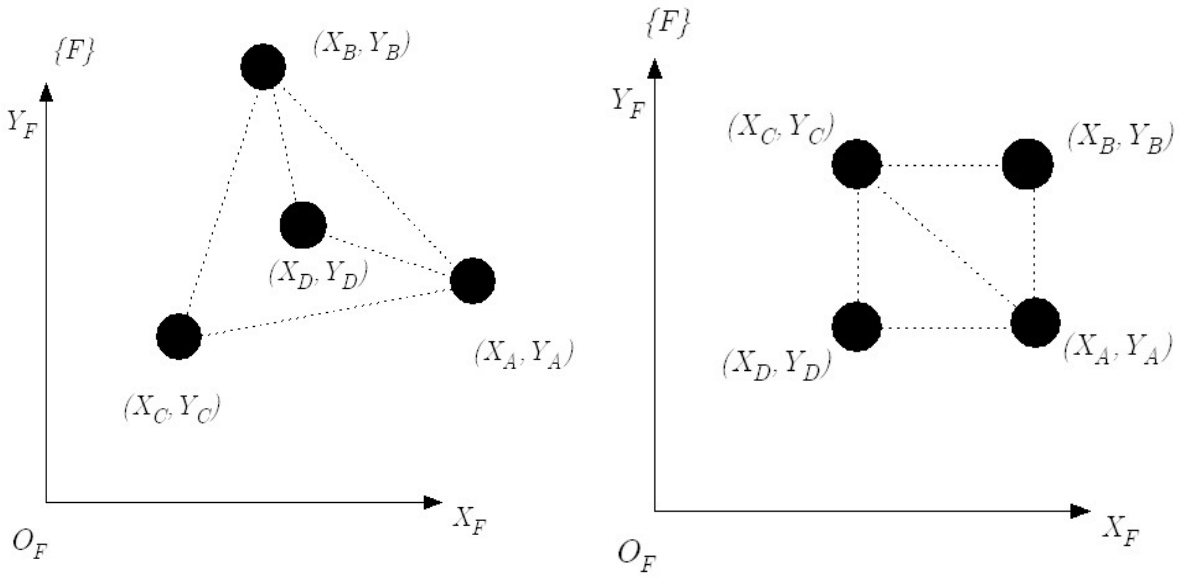
$$\begin{bmatrix}\dot{\mathbf{q}}_A \\ \dot{\boldsymbol{\upsilon}}_A\end{bmatrix} = \begin{bmatrix}\mathbf{S}_A\boldsymbol{\upsilon} + \boldsymbol{\eta}_A \\ \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_A^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_A - \mathbf{M}_A\boldsymbol{\gamma}_A - k_A\mathbf{I}_{2\times 2}\mathbf{q}_A\right)\right]\end{bmatrix}$$
$$\begin{bmatrix}\dot{\mathbf{q}}_B \\ \dot{\boldsymbol{\upsilon}}_B\end{bmatrix} = \begin{bmatrix}\mathbf{S}_B\boldsymbol{\upsilon} + \boldsymbol{\eta}_B \\ \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_B^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_B - \mathbf{M}_B\boldsymbol{\gamma}_B - k_B\mathbf{I}_{2\times 2}\mathbf{q}_B\right)\right]\end{bmatrix} \qquad (6.50)$$
$$\begin{bmatrix}\dot{\mathbf{q}}_C \\ \dot{\boldsymbol{\upsilon}}_C\end{bmatrix} = \begin{bmatrix}\mathbf{S}_B\boldsymbol{\upsilon} + \boldsymbol{\eta}_B \\ \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left[\mathbf{S}_C^T\left(\mathbf{I}_{2\times 2}\mathbf{u}_C - \mathbf{M}_C\boldsymbol{\gamma}_C - k_C\mathbf{I}_{2\times 2}\mathbf{q}_C\right)\right]\end{bmatrix}$$

By examining Equation (6.50) above, we note that the overall system can be evaluated in a distributed manner if states $\mathbf{q}$ and $\boldsymbol{\upsilon}$ are made available. Each independent subpart can now be numerically integrated on a mobile robot thereby permitting independent operation. At each time instant, the complete state of the system needs to be exchanged between the subparts. The coupling between the various sub-parts is due to the existence of the $\left(\mathbf{S}^T\mathbf{M}\mathbf{S}\right)^{-1}$. This matrix inverse needs to be computed on each and every processors, although we note that the explicit calculation of the inverse is typically avoided by using an optimal equation solver. Alternatively, state information from the slace processors could be collected by a central processor at each time instant, where $\left(\mathbf{S}^T\mathbf{M}\mathbf{S}\right)^{-1}$ is computed and the result is broadcasted to all robots.

## 6.7 Choosing Formation Constraints

When formulating the equation of motion with formation constraints, it is important to know the numbers of constraint equations are needed for a particular configuration. As shown in the case study of a three robots, it is not difficult to see that the total number of constraints needed to maintain the triangular formation is $3$. For a

system of four robots, there exist two types of arrangements for the robots, as seen in Figure 6-4.




(a) (b)

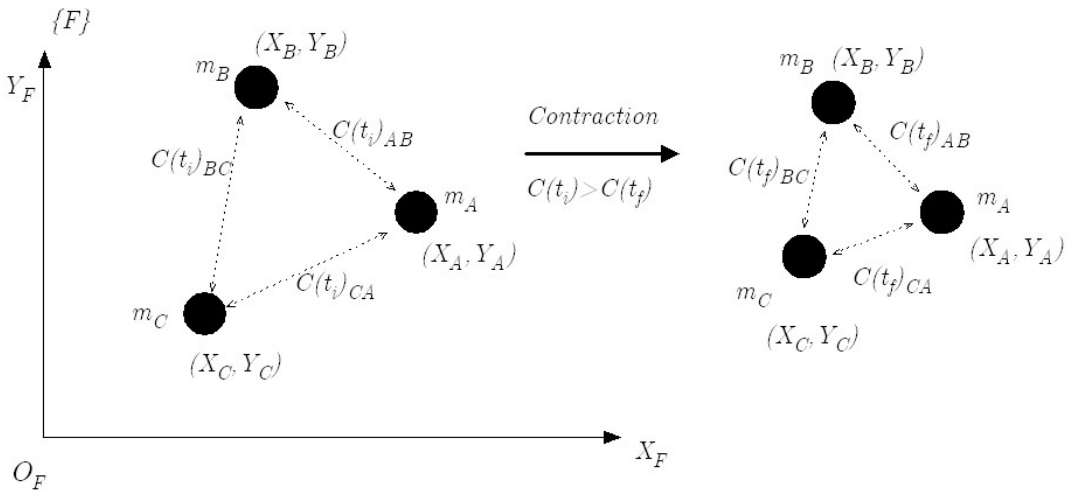
**Figure 6-4: Two different arrangements are possible for *N*=4 number of robots: (a) triangular arrangement where one robot is surrounded by other three robots; and (b) rectangular arrangement where four robot occupied the coners of a rectangle. In both cases, the dotted line is the holonomic constraints needed for maintaining the formation.**

While the two formation arrangements for $N = 4$ robots are different, the number of holonomic constraints needed is the same (the holonomic constraints are shown using dotted line between robots in Figure 6-4). However, as number of robots increases, it become more and more difficult to determine the number of constraints needed in for a group of robots and how the constraints should be distribute among each robot. Fortunately, it is possible to solve this kind of problem using *Graph Theory*. It has been shown that by using graph theory, we can systematically determine the minimum number of constraints needed to maintain a *rigid graph*. The reader is referring to [84-87] for greater detail on determining the constraints needed in a network of robots. In this thesis,

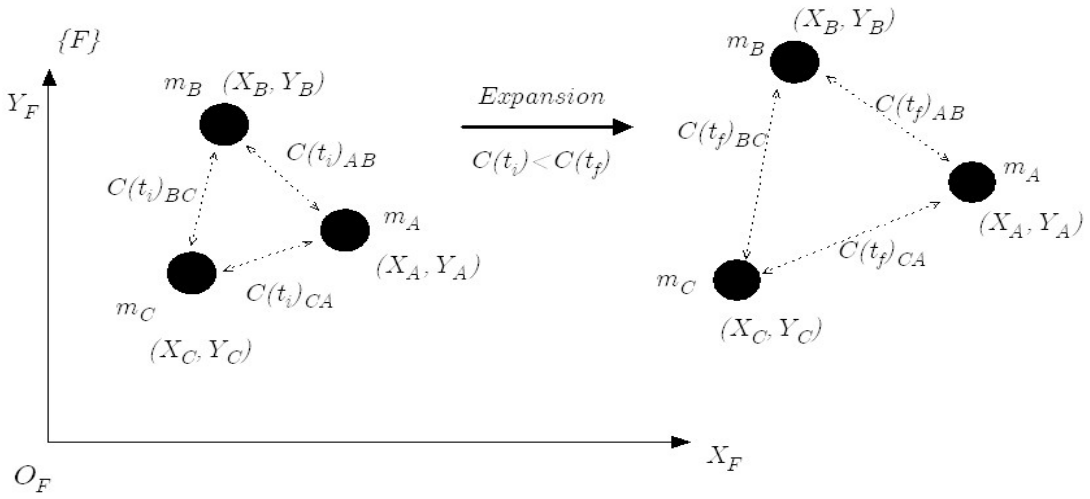
however, we will assume that the number of constraints and their location are known a priori.

## 6.8 Change of Formation- Expand, Contract, of Shape Change

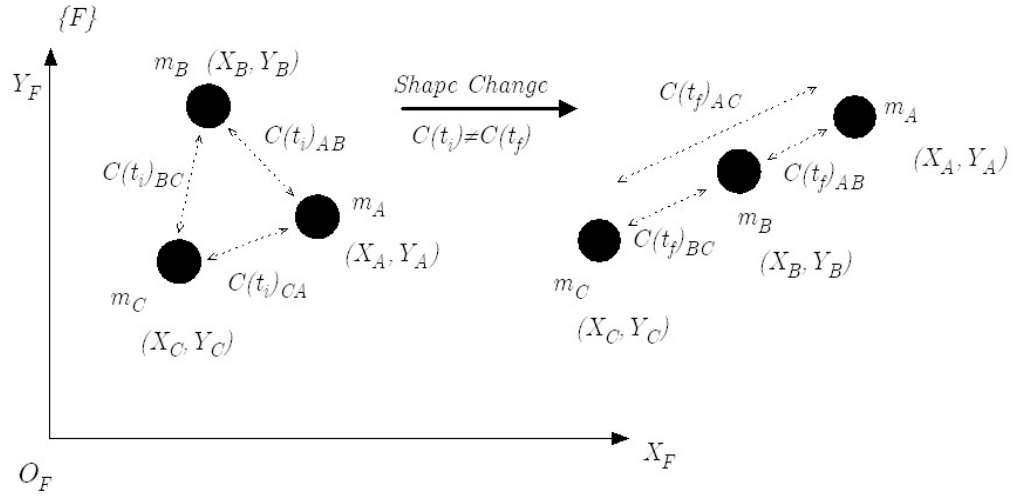
For cooperative payload transport by collective or in distributed sensor deployment applications where the robots are to form some geometric pattern, change of formation may not be a desired feature. Such changes may result in failure to perform the specific task. However, in some situation, such as a group of robots is required to pass through a narrow doorway, change of formation may be a desire feature. In our formulation, change of formation can be achieved through the constraint matrix $\mathbf{C}(\mathbf{q}) = \mathbf{0}$. As we mentioned in earlier this chapter, the constraint matrix can also be a function of time $t$, i.e. $\mathbf{C}(\mathbf{q}, t)$. Through this parameter, we can expand, contract a particular shape or change the formation shape, whenever necessary.




(a)




(b)




(c)

**Figure 6-5: Possible type of reconfigurations using $\mathbf{C}(\mathbf{q},t)$ for group of three point-mass robots: (a) Contraction; (b) Expansion; and (c) Simple shape change.**

However, it is not possible to spilt and re-joint a formation, by changing $\mathbf{C}(\mathbf{q},t)$, as shown in Figure 6-6. These issues are topics beyond the scope of this thesis, and will not be discussed here.

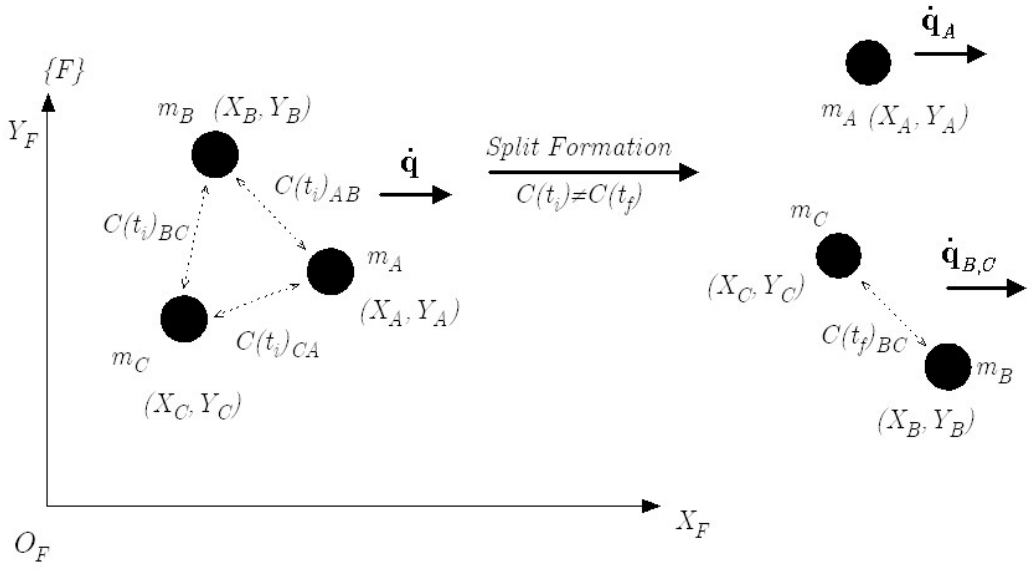



**Figure 6-6: Split of formation from group of three robots to a single robot and a group of two robots is not possible by changing $\mathbf{C}(\mathbf{q},t)$, and is beyond the scope of this thesis.**

## 6.9 Summary

In this chapter, we formulated the dynamic equation for robot collectives by treating them as a constrained dynamic system. The motion planning problem thus becomes simulation of the forward dynamic of the constrained dynamic system. The resulting constrained dynamic system, however, can be solved by using three different methods. Namely the direct Lagrange multiplier elimination method, the penalty formulation method, and the constraint manifold projection method. We provided the general steps used to solve the dynamics equation using these three methods, and shown these step using a case study that has three robot collectives. We also shown that with these formulations, simple formation shape change such as expansion and contraction can be obtained easily. In Chapter 7, we will study the effectiveness of these methods in maintaining formation for robot collectives, with three case studies.

# 7 Simulations and Results

In chapter 6, we formulated the dynamics equation for group of robots and solve the constrained dynamic using three different approaches. In this chapter, we set up standard test bed to evaluate the performance for each of the three methods for their ability to maintain strict formation for task such as cooperative payload transport. We will use Method I, the direct Lagrange multiplier elimination method as the performance benchmark. Using this benchmark, we would like to specifically compare Method II, penalty formulation approach, and Method III, constraint manifold projection approach.

Four case studies were performed:

- Case Study 1: Motion planning for three mobile robots moving in a potential field without obstacles;
- Case Study 2: Motion planning for three mobile robots moving in a potential field with obstacles;
- Case Study 3: Motion planning for three mobile robots moving in a potential field without obstacle while permitting formation expansion;
- Case Study 4: Motion planning for three mobile robots moving in a potential field without obstacle while permitting change of formation shapes.

In each of these case studies, we will use the point-mass robot model for the individual robots. The forward dynamics simulation where performed using MATLAB™. MATLAB™ provides adaptive time-steps solvers such as ODE23, ODE45, ODE15, and

ODE15s, along with fixed-time step solvers such as ODE1, ODE2, ODE3, ODE4, and ODE5. A more detailed description of capabilities of each solver is given in Appendix A.5. We performed the simulation using fixed time-step solver, in consideration of actual implementation, where the information from sensors is evaluated in a specific time interval. In particular, ODE5 is chosen since it has the highest accuracy among other fixed time-step solvers. Before we proceed to any of the case study, several terms that used extensively in this chapter should clarify beforehand. *Simulation Time* is the total time span we set to perform the ODE calculation; *Computational Time* referred to the actual time taken by the specific processor used in the simulation to perform such ODE calculation; and *Time Step* refers to the amount of time at each interval, which can be obtained by dividing the simulation time by number of steps.

## 7.1 Case Study 1: Motion Planning of Three Mobile Robots in Quadratic Potential Field

In the first case study, a group of three robots are required to maintain a triangular formation to move from their initial position to the target. The workspace is free of obstacles. The target point is located at $(0,\ 0)$, the three robots are each located at: robot A, $(10,\ 10)$; robot B, $(10,\ 12)$; and robot C, $(11.732,\ 11)$. The workspace and the arrangement of the robots are shown in Figure 7-1(a). The potential field is a quadratic potential field of the form given in Equation (3.2). The dynamic equations used in this simulation are:

*For method I-Direct Lagrange Multiplier Elimination Approach:*

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \ddot{\mathbf{q}}\end{bmatrix}=\begin{bmatrix}\mathbf{v}\\ \mathbf{M}^{-1}\left[\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}-\mathbf{J}^{\mathbf{T}}\boldsymbol{\lambda}\right]\end{bmatrix} \tag{7.1}$$

$$\boldsymbol{\lambda}=\left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathbf{T}}\right]^{-1}\left[\mathbf{J}\mathbf{M}^{-1}\left(\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}\right)+\dot{\mathbf{J}}\left(\mathbf{q}\right)\dot{\mathbf{q}}+\beta\dot{\mathbf{C}}\left(\mathbf{q}\right)\right]$$

*For method II: Penalty-Formulation Approach:*

$$\begin{bmatrix}\dot{\mathbf{q}}_{2n\times 1}\\ \ddot{\mathbf{q}}_{2n\times 1}\end{bmatrix}=\begin{bmatrix}\mathbf{v}\\ \mathbf{M}^{-1}\left[\mathbf{u}-\mathbf{K}\dot{\mathbf{q}}-\mathbf{J}^{T}\left(\mathbf{K}_{S}\mathbf{C}\left(\mathbf{q}\right)\right)\right]\end{bmatrix} \tag{7.2}$$

*For method III: Constraint Manifold Projection Approach:*

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\upsilon}\end{bmatrix}=\begin{bmatrix}\mathbf{S}\upsilon+\boldsymbol{\eta}\\ \left(\mathbf{S}^{\mathbf{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left(\mathbf{S}^{\mathbf{T}}\mathbf{u}-\mathbf{S}^{\mathbf{T}}\mathbf{M}\boldsymbol{\gamma}-\mathbf{S}^{\mathbf{T}}\mathbf{K}\dot{\mathbf{q}}\right)\end{bmatrix} \tag{7.3}$$

where $\mathbf{u}=-\mathbf{K}_{f}\nabla_{\mathbf{q}}U$ , other terms are defined in chapter 5.

Simulation time is 10 seconds, with fixed time step of 0.001 seconds, and run on a Pentium III™ Intel™ 1.0GHz processor with 512 RAM. The parameters used in this simulation are listed in Table 7-1.

| | Robot A | Robot B | Robot C |
|---|---|---|---|
| Mass of robot | $m_A=1$ | $m_B=1$ | $m_C=1$ |
| Initial x position | 10 | 10 | 11.732 |
| Initial y position | 10 | 12 | 11 |
| Solver used | ODE5 | ODE5 | ODE5 |
| Time span | 10s | 10s | 10s |
| Time step | 0.001s | 0.001s | 0.001s |

**Table 7-1: Parameters and setting used for each of the three robots in case study 1.**

The simulation results are given in Figure 7-1(b), (c), and (d), using method I, method II, and method III respectively.

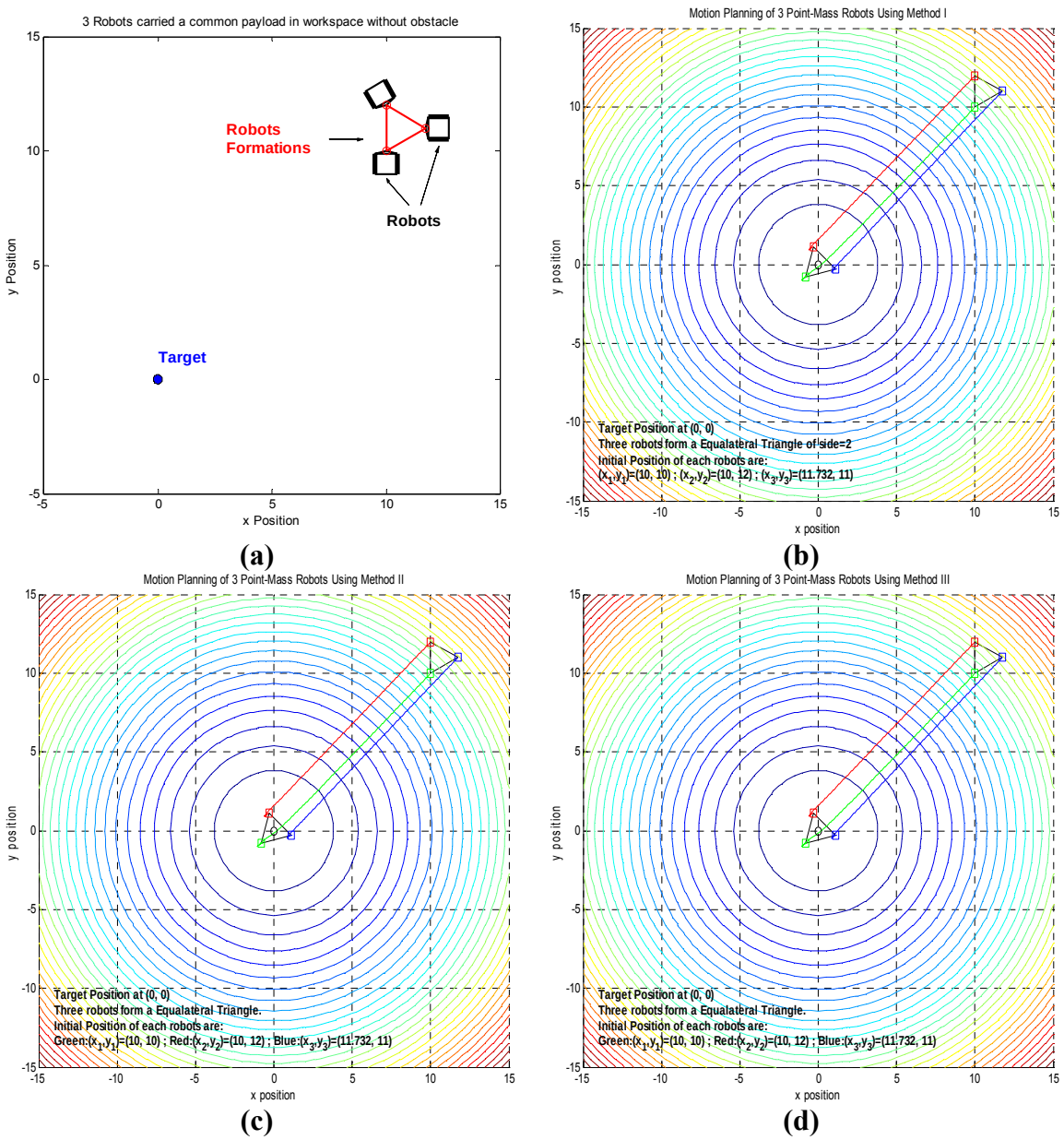
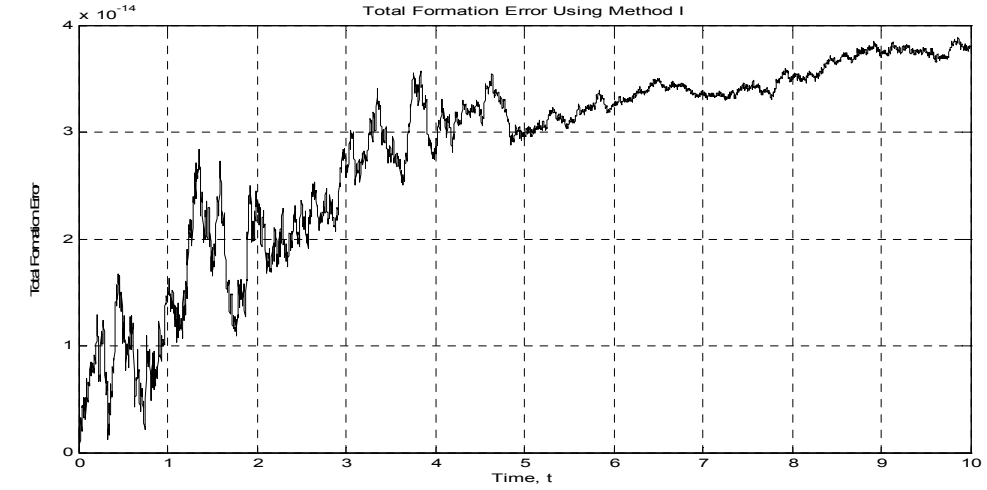



**Figure 7-1: (a) The workspace and position of each robot; (b) Simulation result using Method I; (c) Simulation result using Method II; and (d) Simulation result using Method III.**

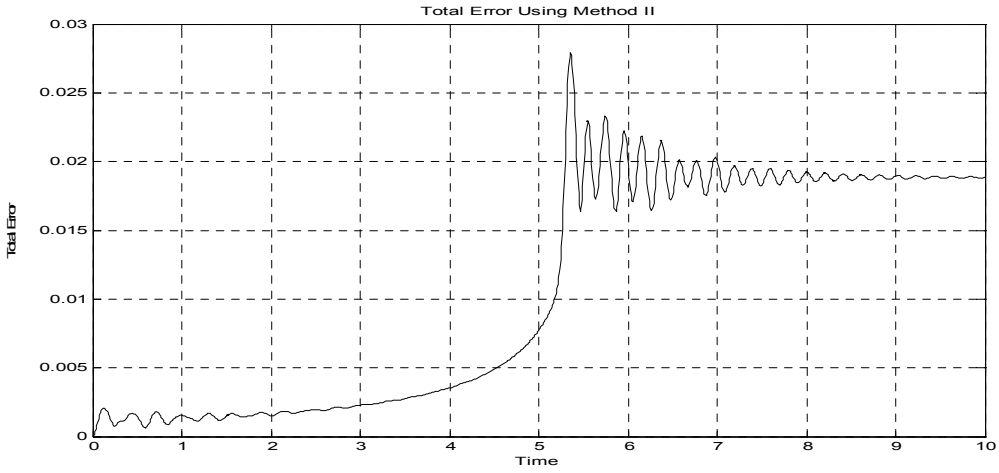
From the simulation result shown in Figure 7-1(b), (c), and (d), it is difficult to distinguish the performance of these three methods. Thus, we plotted the total error at each time step for each of these three methods, and use these plots to compare their performance in maintaining formation. The total error used to measure the performance of each method is given by:

$$\Delta_{Error} = \sqrt{\left(c_{AB} - \bar{c}_{AB}\right)^2 + \left(c_{BC} - \bar{c}_{BC}\right)^2 + \left(c_{CA} - \bar{c}_{CA}\right)^2} \tag{7.4}$$

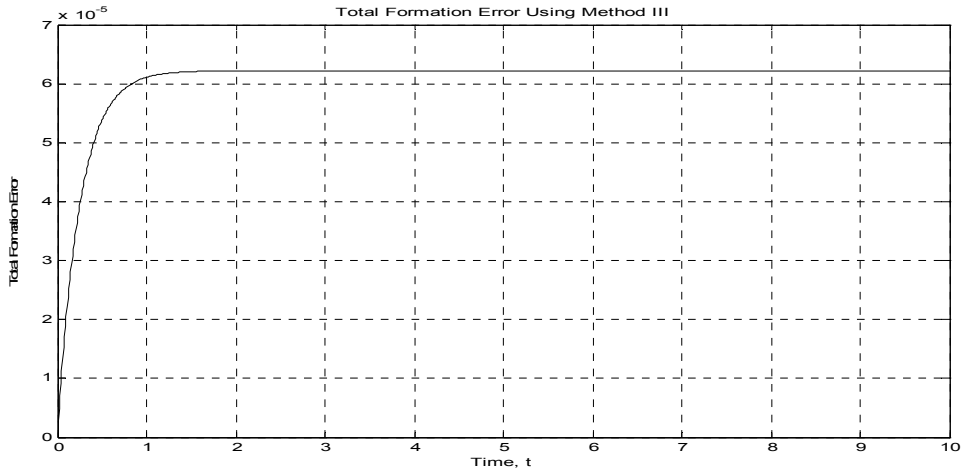
where $\Delta_{Error}$ is the total formation error; $c_{ij}$ is the *actual* Euclidean distance between robot $i$ and robot $j$; and $\bar{c}_{ij}$ is the *desired* Euclidean distance between robot $i$ and robot $j$. The result of the total error at each time step is given in Figure 7-2.




**(a)**




**(b)**




**(c)**

**Figure 7-2: (a) Total formation error using method I; (b) Total formation error using method II; and (c) Total formation error using method III.**

From the simulation, the maximum formation error occurred in the simulation for each of the three methods are: $3.3882\times10^{-14}$ using method I; $3.87\times10^{-2}$ using method II with $K=10$ is; $9.269\times10^{-9}$ using method III with $\sigma=10$. Setting method I as the benchmark, the performance of method II and method III is varied in the order of $10^{7}$. However, we observed that the performance of method II and method III are both affect by the parameter used in their respective formulation. Namely the spring stiffness $\mathbf{K}_S$ used in method II and $\sigma$ used in method III. Varying the value of these parameters may alter the performance of their respective methods, at the same time may also affect the computation time. Thus, it is worth to study the relationship between formation error and parameter's value used in each method by: (1) Plot the formation error at each time step with different parameter's value for each method; and (2) plot the total accumulated formation error in one simulation against parameter value used. In this way, we can better quantify their performance.

The semi-log plot of formation error versus $K_S$ (spring stiffness), with values ranged from 10 to 10,000 for method II is given in Figure 7-3.

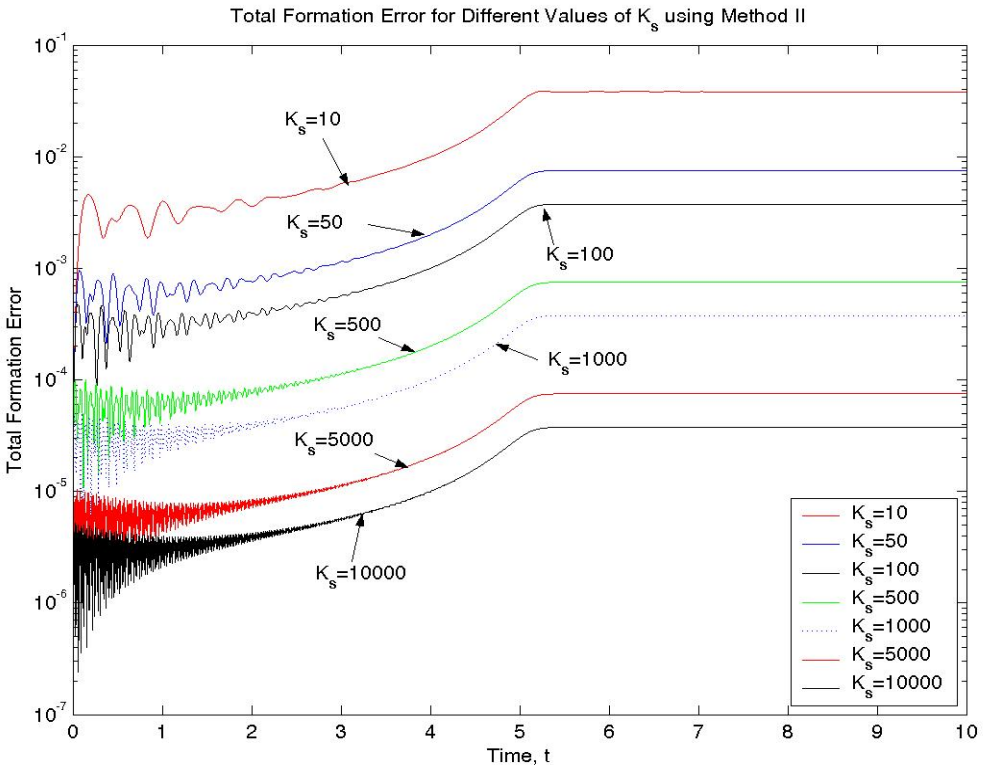



**Figure 7-3: Compare of total formation error with $K_S$ value increased from 10 to 10000, with a simulation time step of 0.001 sec, using method II.**

On the other hand, the semi-log plot of total formation error versus value of $\sigma$ ranging from 10 to 1000 is plotted in Figure 7-4.

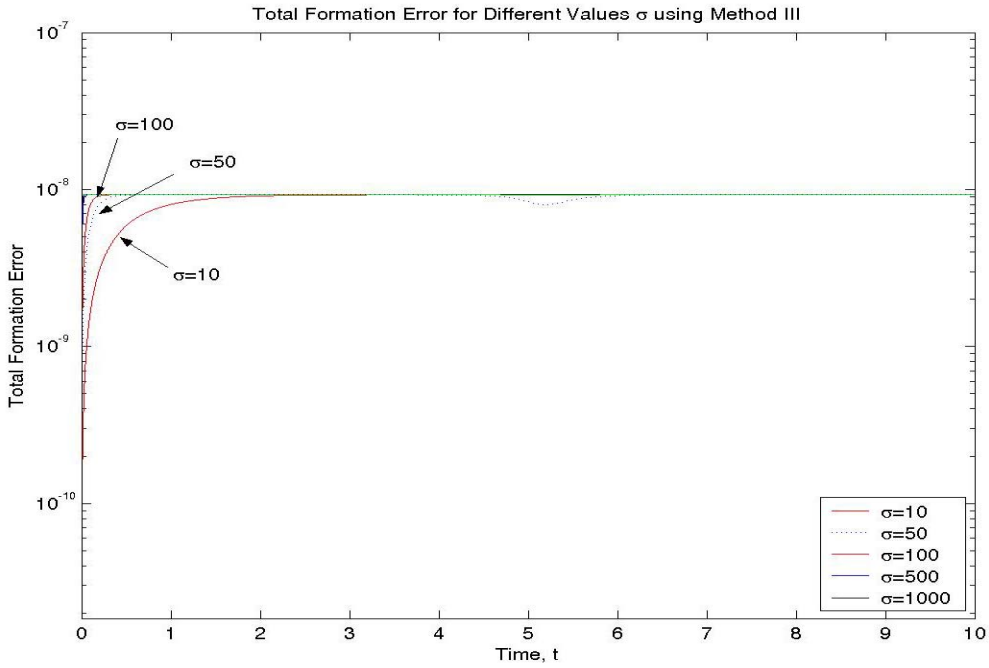



**Figure 7-4: Plot of total formation error versus $\sigma$ value that ranged from 10-10000, with time step of 0.001 sec, using method III.**

From Figure 7-3, we observed that the formation decreases as the value of spring stiffness increases. For this simulation, when the spring stiffness increases from 10 to 10,000, the formation error drops from maximum formation error of $3.87\times10^{-2}$ to $3.7571\times10^{-5}$. For spring stiffness above 10,000, the total formation error starts to increase again, due to the numerical ill stiff equation that we discussed in the beginning of chapter 5. On the other hand, by observing Figure 7-4, we found that the total formation error have a stable value at $9.2699\times10^{-9}$, where the $\sigma$ value only affect how fast the system converged to this value of total formation error: the larger the value of $\sigma$, the faster the convergence rate, which is consistent to our formulation.

The best formation error that can be obtained by that of method II (in the order of $10^{-5}$) never reach the best formation error value that can be obtained by using method III

(in the order of $10^{-9}$). On the other hand, the time taken to perform the ODE calculation recorded in the following table:

| Method II | |
|---|---|
| Parameter Value | Time Taken (sec) |
| $K_S = 10$ | 56.481 |
| $K_S = 50$ | 56.011 |
| $K_S = 100$ | 56.391 |
| $K_S = 500$ | 56.952 |
| $K_S = 1000$ | 57.093 |
| $K_S = 5000$ | 56.822 |
| $K_S = 10000$ | 56.902 |

| Method III | |
|---|---|
| Parameter Value | Time Taken (sec) |
| $\sigma = 1$ | 101.586 |
| $\sigma = 5$ | 101.206 |
| $\sigma = 10$ | 100.875 |
| $\sigma = 50$ | 100.775 |
| $\sigma = 100$ | 100.895 |
| $\sigma = 500$ | 100.464 |
| $\sigma = 1000$ | 101.696 |

**Table 7-2: Average time taken to perform ODE calculation for different K value for method II and different $\sigma$ value for method III.**

Compare to average time of 38.215 seconds used for method I to perform the ODE calculation, the average time taken to perform the ODE calculation is 56.664 seconds using method II, and is 101.071 seconds with method III. From these observations, we see that changing the values of the parameter do not affect the time taken to perform ODE calculation. The reason is that the solver used to perform the ODE calculation is a fixed time-step solver, which performed the ODE calculation at each given time steps, and the time differences for each method is the result of the function evaluation at each steps. As a result, method III took about 44.407 seconds more in the average calculation time compare to method II. This can be explained as at each time step, the computation of the inverse of Jacobian matrix, which is a time consuming calculation, is needed for method III.

## 7.2 Case Study 2: Motion Planning of Three Robots in Navigation Function with One Obstacle

In the second case study, an obstacle is place between the target and a group of three robots. These three robots are required to maintain a triangular formation and move toward the target. The initial positions of the three robots are: robot A, $(-2,\ -3)$; robot B, $(-2,\ -4)$; and robot C, $(-2.866,\ -3.5)$. The target is located at $(2.5,\ 2.5)$. The obstacle is located at the origin and has a radius of 1.5. This arrangement is shown in Figure 7-5(a). The potential field of the workspace is created using navigation function, and a $\kappa = 1.6$ value is selected using the GUI described in chapter 3. The 3D potential field of the workspace is shown in Figure 7-5(b), and the contour plot of this potential field is shown in Figure 7-5(c).

The total simulation time is 8 seconds, with a fixed time-step of $1\times10^{-3}$ second. The solver used is ODE5, and the dynamic equations of the this group of robot is given by Equation (6.26). Once again, this dynamic equation is solved using the three methods that we described in Chapter 5. With method I, it is solved using Equation (7.1), with method II, the equation used is Equation (7.2), and with method III, Equation (7.3) is used. This simulation is run on a Pentium III Intel™ 1.0GHz processor with 512 RAM. The parameters used in this simulation are listed in Table 7-1.

| | Robot A | Robot A | Robot C |
|---|---|---|---|
| Mass of robot | $m_A = 1$ | $m_B = 1$ | $m_C = 1$ |
| Initial x position | -2 | -2 | -2.866 |
| Initial y position | -3 | -4 | -3.5 |
| Time span | 8s | 8s | 8s |
| Time step | 0.001s | 0.001s | 0.001s |

**Table 7-3: Parameters and settings for each of the three robots used in case study 2.**

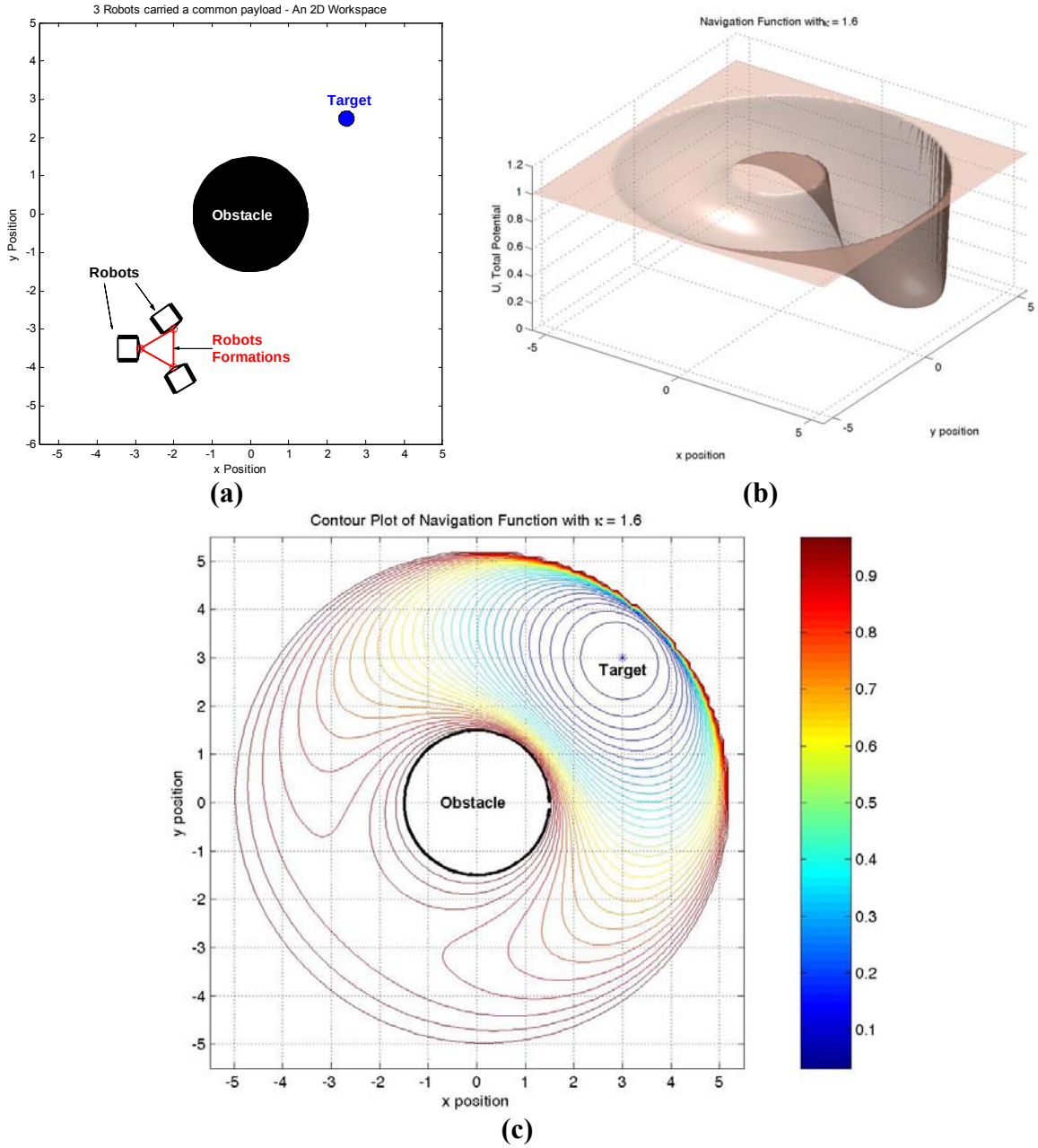



**Figure 7-5: (a) The actual workspace showing the position of the three robots, target, and obstacle; (b)The 3D potential field of the workspace created using navigation function with a value of $\kappa = 1.2$ ; and (c) The contour plot of the workspace potential field showing the gradient information.**

The result of the simulation is given in Figure 7-6. The gradient information is obtained numerically.

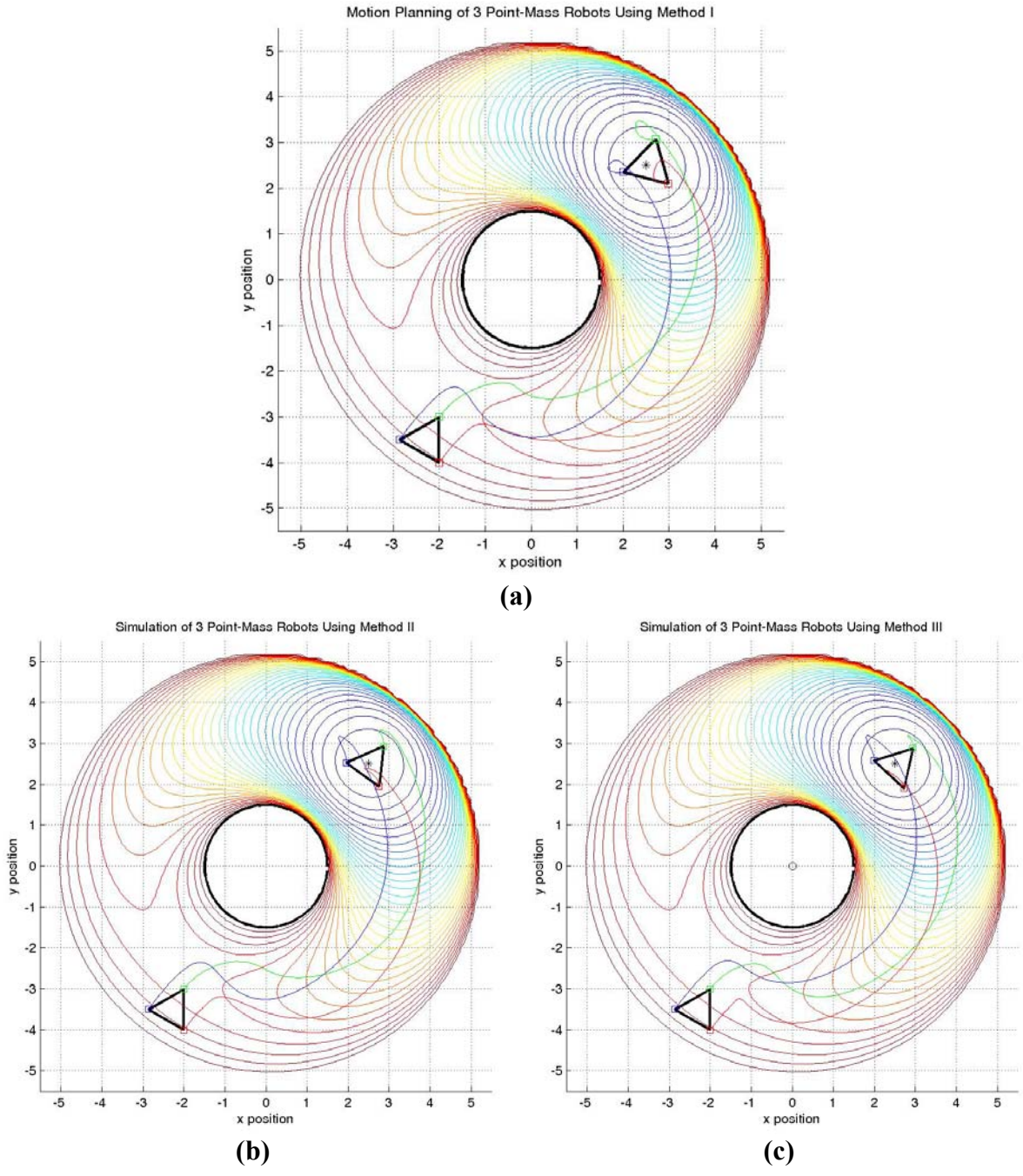
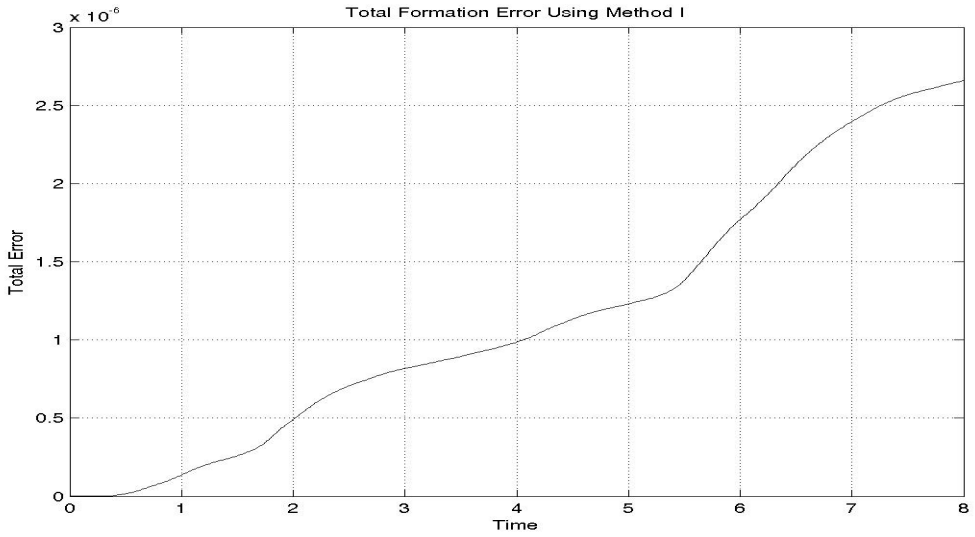


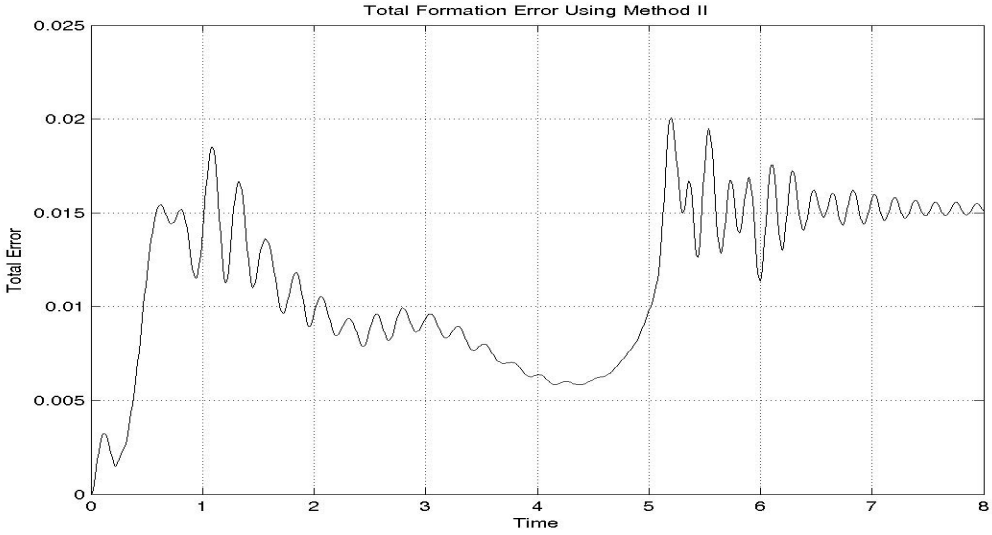

**Figure 7-6: Simulation result of three point mass robots (robot A in Green color; robot B in Red color; and robot C in Blue color) moving in formation towards the target and avoiding obstacle, using (a) Method I, (b) Method II, and (c) Method III. Observe the different path taken in different method used.**

From Figure 7-6, we can see that the there are some differences in the path taken to reach the final position for different methods. In the above simulation, spring stiffness

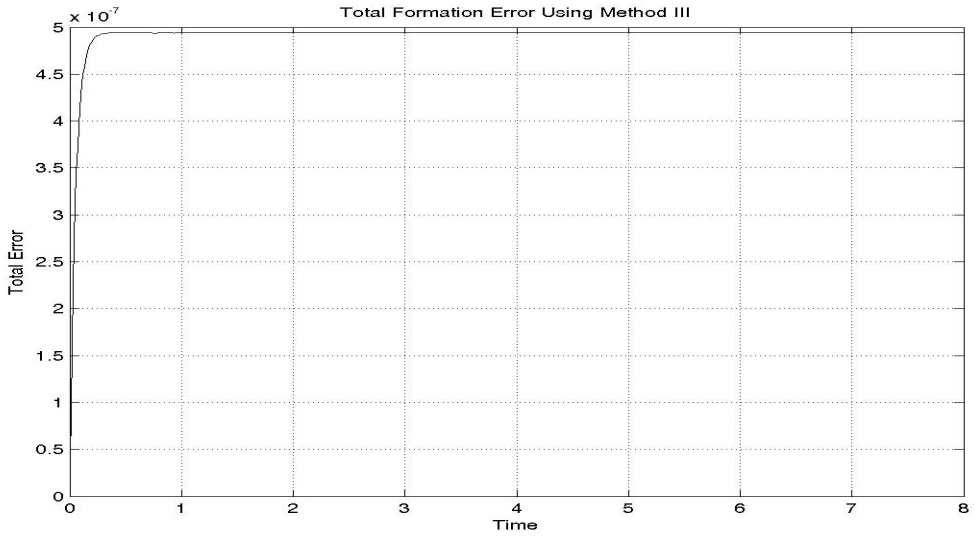
$K_S = 100$ is used for method II, and $\sigma = 10$ is used for method III. To evaluate the performance of each method in maintaining formation, formation error in each step calculated using Equation (7.4) is plotted in Figure 7-7.




**(a)**




**(b)**




**(c)**

**Figure 7-7: Formation error for each method in case study 2, where (a) Total Formation Error using Method I; (b) Total Formation Error using Method II; and (c) Total Formation Error using Method III. In particular,** $K_S = 100$ **is used in method II and** $\sigma = 10$ **is used for method III.**

From Figure 7-7, the formation error is in the order of $10^{-6}$ using method I, $10^{-2}$ using method II, and the formation error of $10^{-7}$ can be achieved using method III. To examine how precise each method can achieve, we again increase the value of $K_S$ in method II and value of $\sigma$ in method III. Figure 7-8 shown the relationship between total formation error and value of $K_S$: as value of $K_S$ increases, the formation error decreases. For $K_S$ value greater than 1000, the formation error starts increase again.

Figure 7-9, on the other hand, shown the relationship between total formation error and $\sigma$ value: there are no changes to the maximum total formation error as value of $\sigma$ increases, it only affect how fast the error converge to the final maximum total formation error.

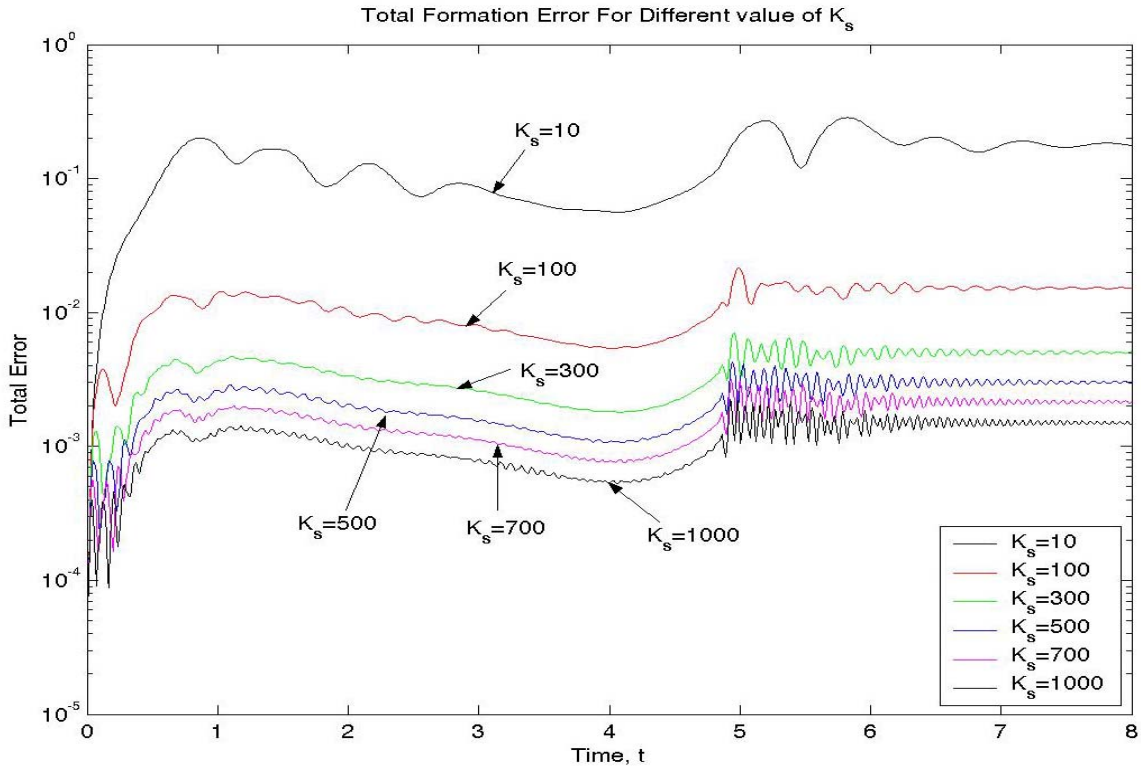



**Figure 7-8: Total formation error using method II, for $\kappa$ value between 10 and 1000.**

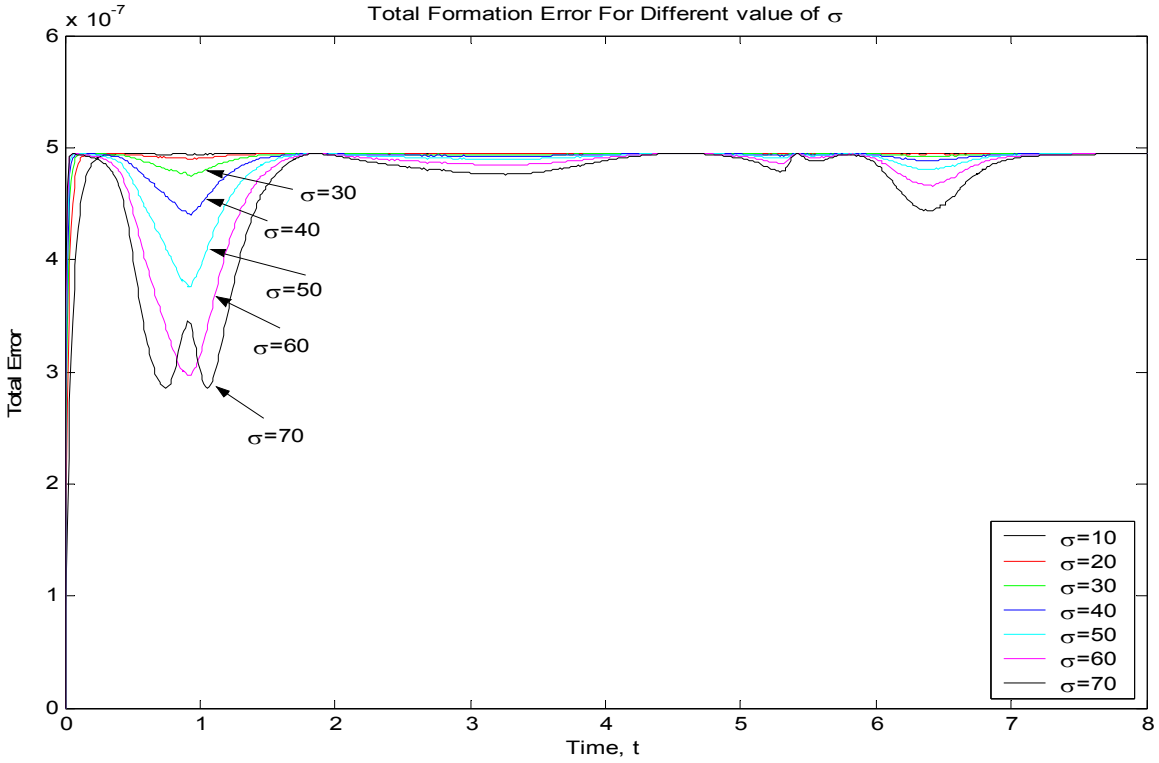



**Figure 7-9: Total Formation Error for method III, for $\sigma$ value between 10 and 70.**

For $\sigma > 70$, the total formation error started to increase. From the view point of total formation error each method can achieved, the result of this case study is consistent with case study 1.

In the first case study, we see that the computation time does not varied significantly with different parameters used in the formulation. Thus, instead of calculating the computation time for each method, the average computational time is given here. The average computation time to perform the ODE calculation is 147.38 seconds for Method I, 145.27 seconds for Method II, and 148.52 second for Method III. The reason for the similar computation time used for each method is due to the numerical gradient finding method (see Appendix A.2 for detail) that we implemented within the ODE function is a time consuming process.

## 7.3 Case Study 3: Motion Planning of Three Mobile Robots Moving in Potential Field While Permitting Formation Expansion

In the third case study, we study a situation where three point mass robot travel to a designated point, while expanding the formation. We want the three robots, which initially forming an equilateral triangle with side of 2 unit to expand the formation shape to form a equilateral triangular of side 4 units. This can be done, as we mention in section 6.2, by modifying the constraint matrix $\mathbf{C}(\mathbf{q})$ to $\mathbf{C}(\mathbf{q},t)$. We will use the same workspace arrangement in case study 1, except now we want the holonomic distance

constraint between each robot change from 2 to 4 in 4 seconds, and maintained at 4 throughout:

$$c_{ij} = \begin{cases} 2 + 0.5t, & 0 \le t \le 4 \\ 4, & t > 4 \end{cases} \tag{7.5}$$

where $c_{ij}$ denote the Euclidean distance between robot $i$ and robot $j$.

The total simulation time is 10 seconds, with a fixed time-step of $1\times10^{-3}$ seconds. The simulation setting for each of the three robots is given in Table 7-4.

| | Robot A | Robot A | Robot C |
|---|---|---|---|
| Mass of robot | $m_A = 1$ | $m_B = 1$ | $m_C = 1$ |
| Initial x position | -2 | -2 | -2.866 |
| Initial y position | -3 | -4 | -3.5 |
| Time span | 8s | 8s | 8s |
| Time step | 0.001s | 0.001s | 0.001s |

**Table 7-4: Parameters and setting used for each of the three robots in case study 3.**

The dynamic equations used for each of the three methods in the simulation are given as follow, with slightly modification from the one we use in case study I and II:

*For Method I:*

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\left[\mathbf{u} - \mathbf{K}\dot{\mathbf{q}} - \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}\right] \end{bmatrix} \tag{7.6}$$

where:

$$\boldsymbol{\lambda} = \left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\right]^{-1}\left[\mathbf{J}\mathbf{M}^{-1}\left(\mathbf{u} - \mathbf{K}\dot{\mathbf{q}}\right) + \dot{\mathbf{J}}\dot{\mathbf{q}} + \beta\dot{\mathbf{C}} + \sigma\mathbf{C} + \frac{\partial \mathbf{C}^2}{\partial t^2}\right]$$

$\mathbf{C} = \mathbf{C}(\mathbf{q}, t)$ and $\mathbf{u} = -\mathbf{K}_f \nabla_{\mathbf{q}} U$

*For Method II:*

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \ddot{\mathbf{q}}\end{bmatrix}=\begin{bmatrix}\mathbf{v}\\ \mathbf{M}^{-1}\left[\mathbf{u}-\mathbf{J}^{T}\left(\mathbf{K}_{S}\mathbf{C}(\mathbf{q},t)\right)\right]\end{bmatrix} \tag{7.7}$$

*For Method III:*

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\mathbf{v}}\end{bmatrix}=\begin{bmatrix}\mathbf{S}\mathbf{v}+\boldsymbol{\eta}\\ \left(\mathbf{S}^{\mathrm{T}}\mathbf{M}\mathbf{S}\right)^{-1}\left(\mathbf{S}^{\mathrm{T}}\mathbf{u}-\mathbf{S}^{\mathrm{T}}\mathbf{M}\boldsymbol{\gamma}-\mathbf{S}^{\mathrm{T}}\mathbf{K}\dot{\mathbf{q}}\right)\end{bmatrix} \tag{7.8}$$

where now:

$$\boldsymbol{\eta}(\mathbf{q})=\begin{bmatrix}\mathbf{0}_{(2n-m)\times 1}\\ -\sigma\mathbf{J}_{d}^{-1}(\mathbf{q})\mathbf{C}(\mathbf{q},t)-\dfrac{\partial\mathbf{C}(\mathbf{q},t)}{\partial t}\end{bmatrix}$$ and other terms are defined in chapter 5.

We found that for Method I, the $\frac{\partial^2\mathbf{C}}{\partial t^2}$ term does not affect the simulation result in a significant way, the same observation also given in [14]. Thus, we drop out this term in our simulation. Thus, in Method I, the only term that makes sure the formation changes as required is the Baumgarte stabilization term. In Method II, incorporate of formation expansion is straight forward: by changing the distance of the virtual spring and damper. With Method III, the only modification to the equation is the $\boldsymbol{\eta}(\mathbf{q},t)$ term. The simulation result using the respective three methods is given Figure 7-10. In these three simulations, the average computation time required for Method I is 104.3 seconds; for Method II is 45.12 seconds; and for Method III is 159.62 seconds. To see the performance for each of the method during formation expansion (or contraction), the total formation error plotted against the final desired formation shape is given in Figure 7-11. For each of these simulation, three of the robots successfully reach the target $(0,\ 0)$, at

about 4.5 second. Thus we can see that after the robots reach the target, the formation error starts decrease sharply.

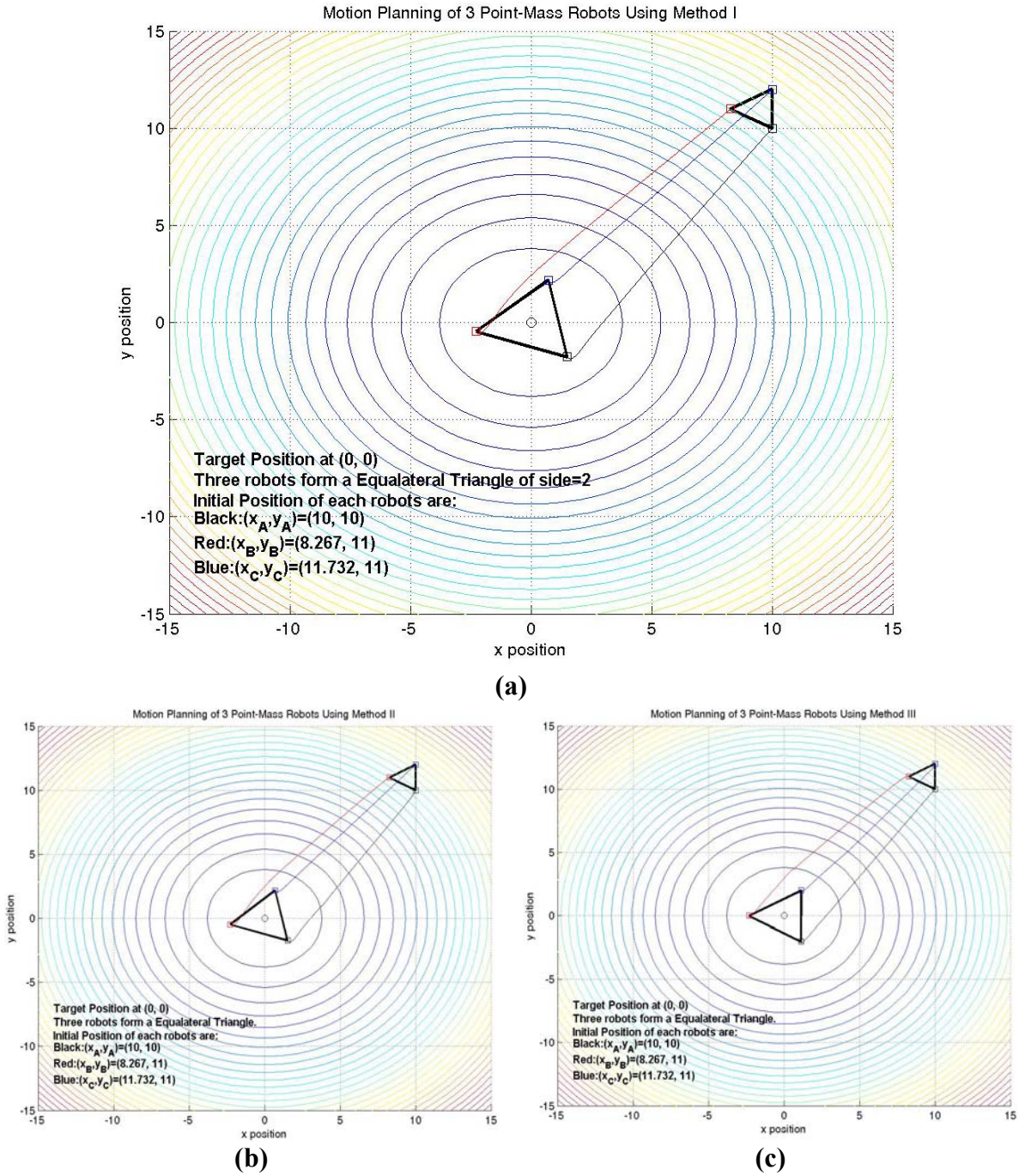
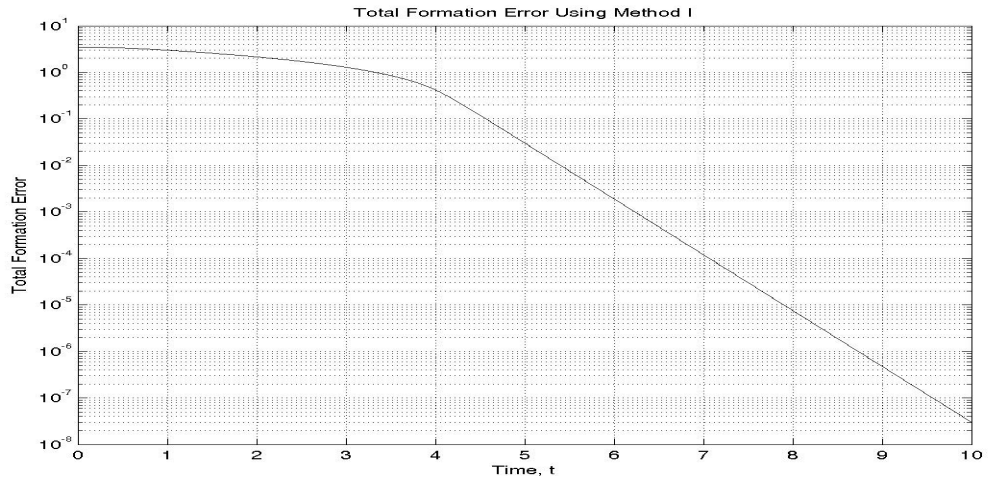
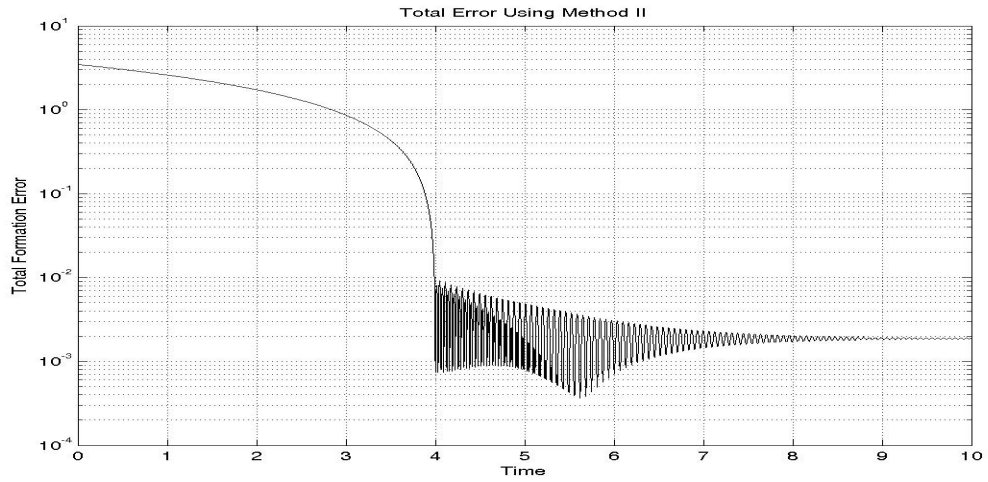


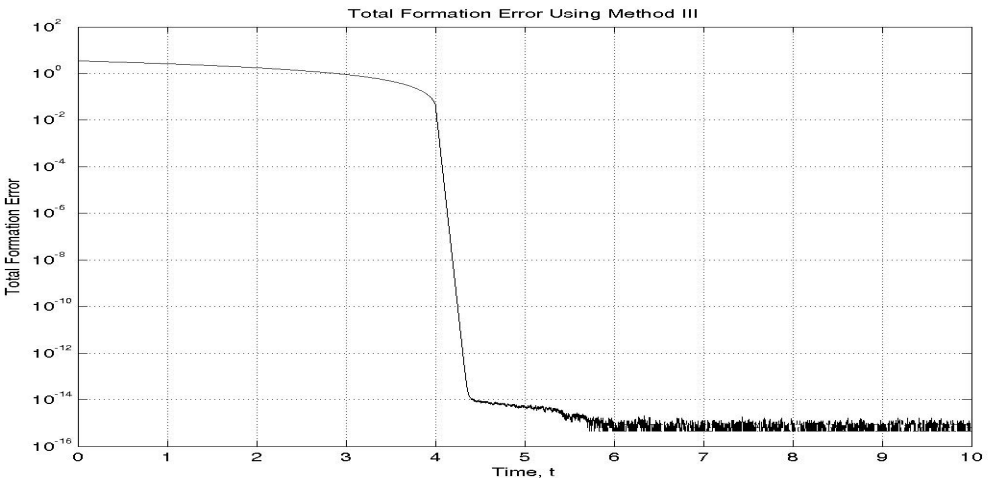

**Figure 7-10: Simulation result of three point-mass robots moving in potential field with formation expansion; using (a) Method I; (b) Method II; and (c) Method III.**




**(a)**




**(b)**

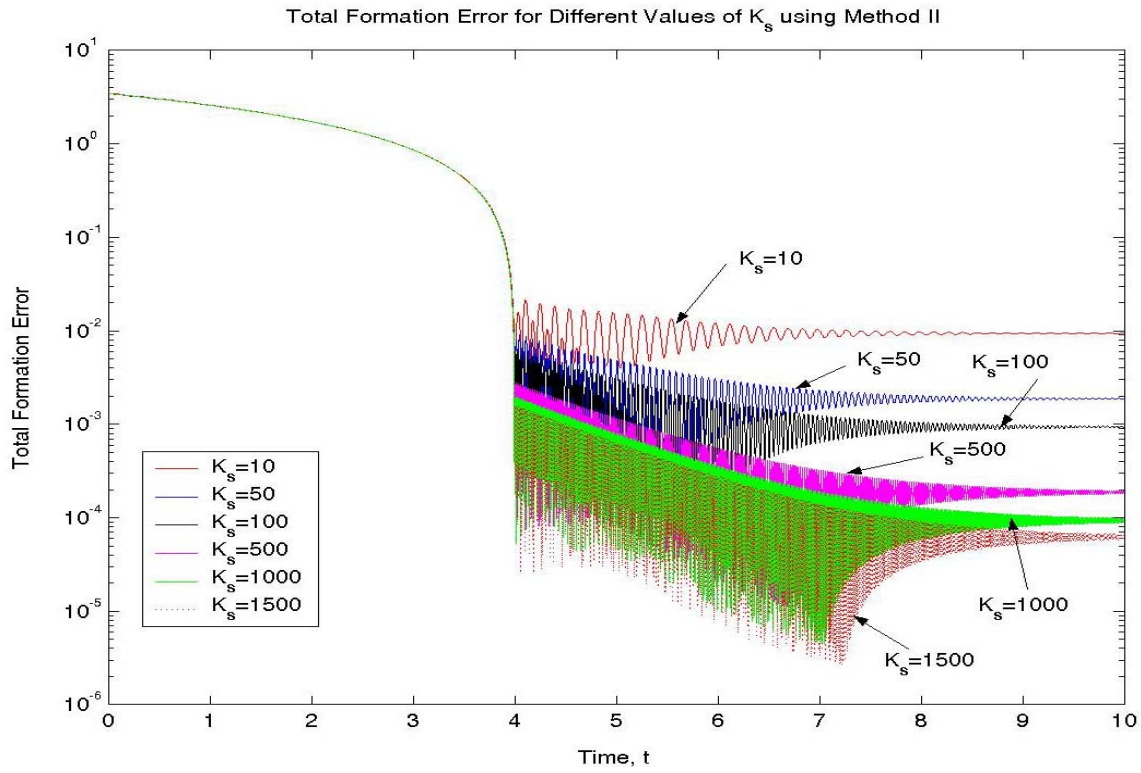



**(c)**

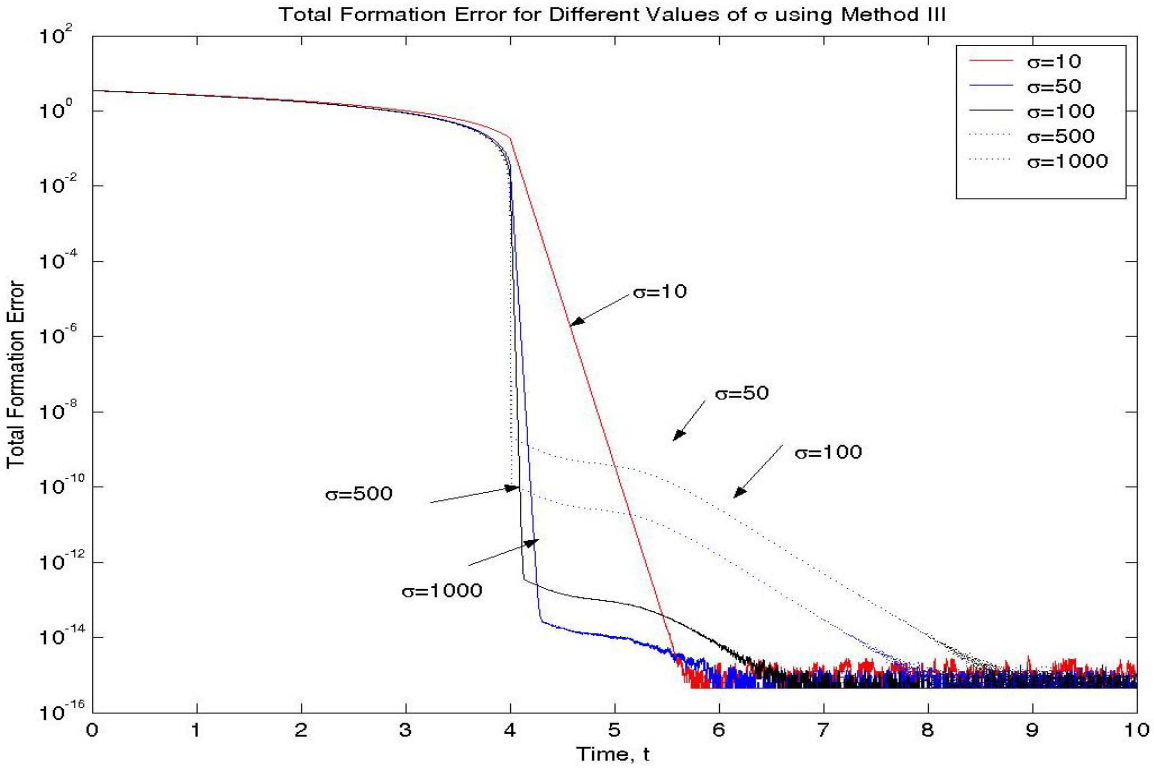
**Figure 7-11: Plot of total formation error for each method in case study 3: (a) Method I; (b) Method II with $K_S = 100$; and (c) Method III with $\sigma = 100$.**

The performance of each of these methods can be compared by looking at the total formation error plot. As we can see from Figure 7-11(a), the total formation error using Method I decrease about linearly after 4.5 second (where the robots reached the target) and reached accuracy of $1\times10^{-7}$. In Method II, we can see that the total formation error decrease more rapidly compare to one obtained using Method I and Method III. However, the smallest formation error can obtained with this result is in the order of $1\times10^{-3}$. From Figure 7-11(c), we can observed that the formation error decrease sharply after 4 second, and reaches a relatively stable value at the order of $1\times10^{-15}$.

To evaluate the performance of Method II and Method III, we plotted the formation error obtained using this two methods for different parameters used. As shown in Figure 7-12. From Figure 7-12(a), we can see that the best performance can achieved using Method II is in the order of $1\times10^{-4}$. Also, increase $K_S$ value from 10 to 1500 decreases the total formation error from $1\times10^{-2}$ to $1\times10^{-4}$. On the other hand, Method III gives a better performance. In Figure 7-12(b), we can see that for different value of $\sigma$, the total error is able to maintain at order of $1\times10^{-15}$. Increase value of $\sigma$ increase the converging rate of the total formation error to a order at $10^{-15}$.




**(a)**




**(b)**

**Figure 7-12: Total formation error plotted for (a) Different value of $K_S$ for Method II; and (b) Different value of $\sigma$ for Method III.**

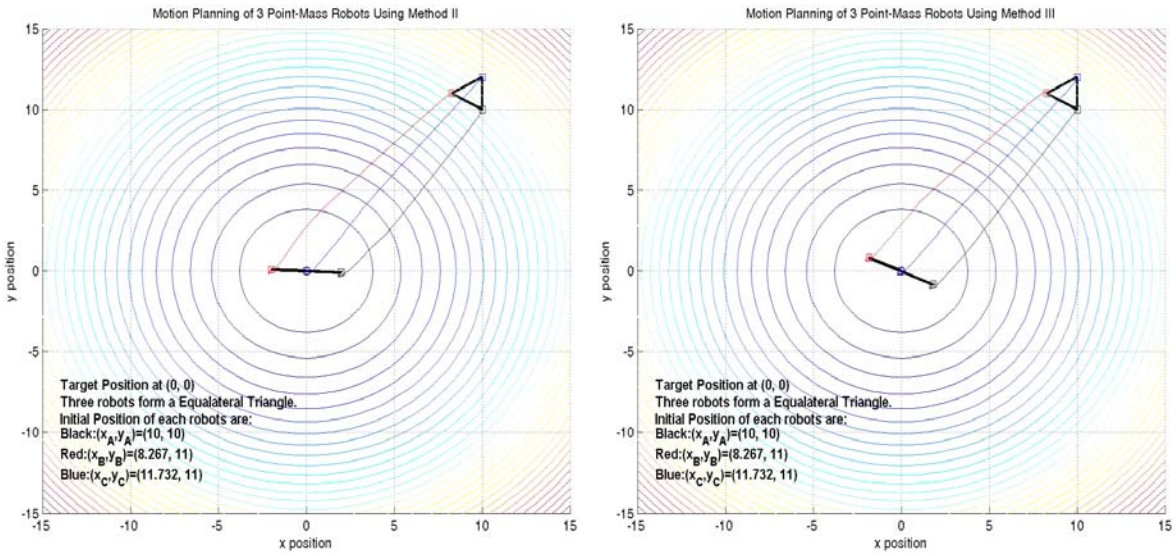
## 7.4 Case Study 4: Motion Planning of Three Point Mass Robots Moving in Potential Field while Permitting Formation Shape Change

In this case study, we would like to evaluate the performance for each of these three methods in performing formation shape change as described in Figure 6-5(c). That is, we want the three robots, which initially forming a triangular shape, change their shape such that they are on a straight line and move to the target. This can be done by extending one of the three sides forming the triangular shape. In particular, we want the side between robot A and robot B change from 2 unit to 4 unit in the first 4 second of the simulation. This can be expressed as:

$$c_{AB} = \begin{cases} 2 + 0.5t, & 0 \le t \le 4 \\ 4, & t > 4 \end{cases} \tag{7.9}$$

where $c_{AB}$ is the Euclidean distance between robot A and robot B.

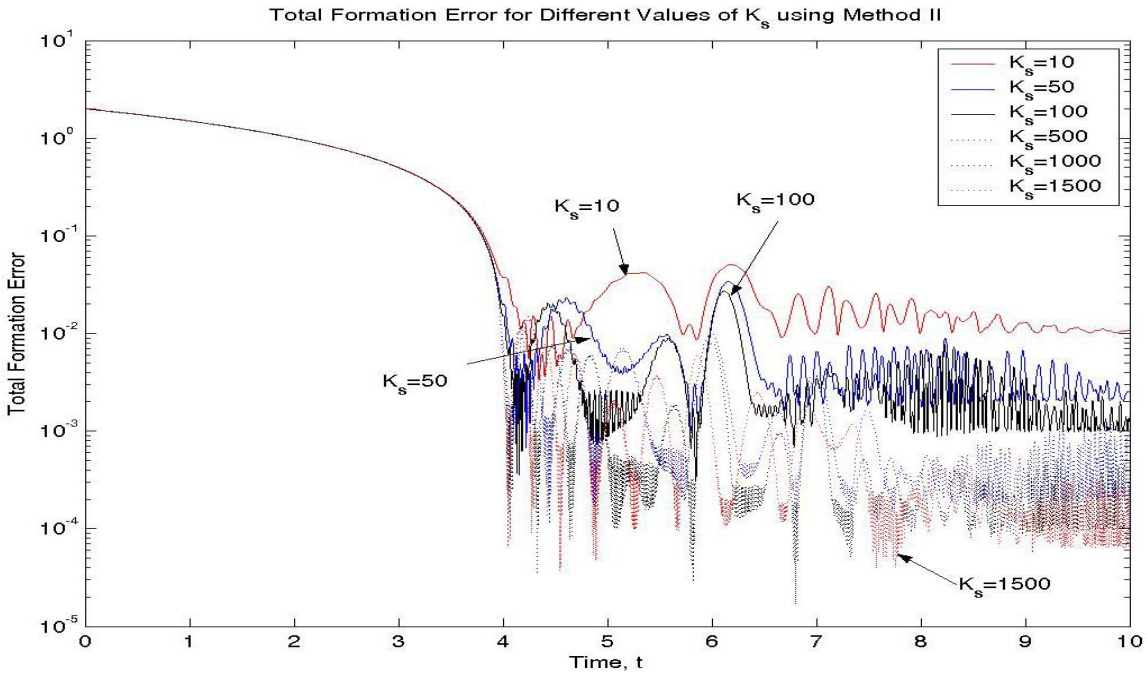
The dynamic equations for each of the methods used in this simulation are Equation (7.6), (7.7), and (7.8), respectively. However, this change of formation shape to a straight line cannot be performed using Method I. The reason is when the three robots are in a straight line, the component of the Jacobian of the constraint matrix is dependent on each other. Since Method I required the inverse of the Jacobian, which in this case, results in a singular matrix, will cause the numerical computation failed. As a result, we only able to show the simulation result for Method II with $K_S = 50$ and Method III with $\sigma = 50$, and compare their performance. The simulation result is shown in Figure 7-13.



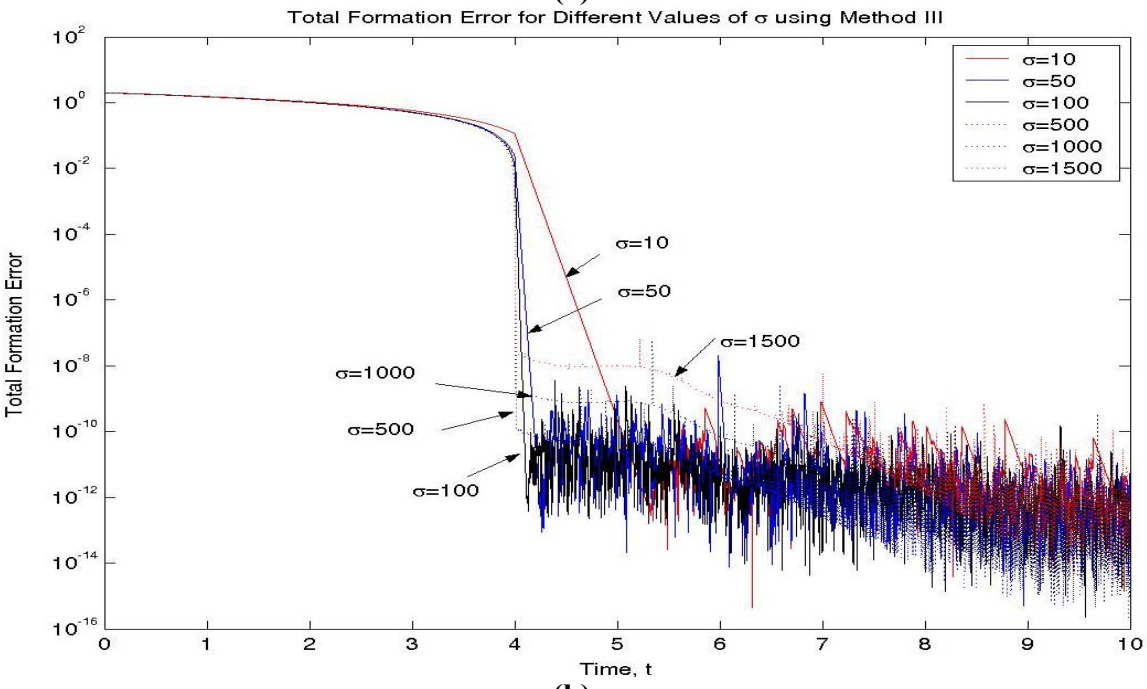

(a) (b)

**Figure 7-13: Simulation result of three point mass robot moving in potential field while permitting change of formation from triangular shape to a straight line, using (a) Method II; and (b) Method III.**

The average total computational time used for Method II is around 43.56 seconds, and about 85.67 seconds for Method III. The total formation error plot for different value of $K_S$ and $\sigma$ is given in Figure 7-14. From Figure 7-14, we can see that as the value of $K_S$ increases from 10 to 1500, the total error decrease from an order of $10^{-2}$ to about $10^{-4}$. On the other hand, as value of $\sigma$ increases from 10 to 100, the total error remain in an order of $10^{-12}$, with different converging rate; and continuing increase $\sigma$ the total error start increase to an order of $10^{-8}$. In this case study, we can see that Method III has a better performance in maintaining formation constraints compare to Method II. However, this comes at a higher cost in terms of computation time: on average, Method III spent about 2 times the time required by Method II. We can also see that these two methods are good for motion planning compare to method I as they permit the change of formation shape, which may be a beneficial feature for motion planning.




**(a)**




**(b)**

**Figure 7-14: Total formation error in case study 4 plotted for different value of (a) $K_S$ for Method II; and (b) $\sigma$ for Method III.**

## 7.5 Discussion of General Characteristics of Each Methods

We developed the dynamics equation for each method to use with potential field approach in motion planning for group of robots in chapter 5. In previous sections, various case studies were performed to study the performance of each of the three methods. In particular, the first two case studies evaluate the performance of each of the three methods for their ability in maintaining a particular formation shape throughout the motion planning; while the other two case studies evaluate their performance in situation where formation shape changes is needed. We would like to evaluate their performance relative to each other based on various aspects. Specifically, we would like to discuss the accuracy, computational speed, decentralization ability, and formation related concern for each of the three methods.

### 7.5.1 Accuracy

From the first three case studies that we have performed, we observed that Method III, in general provide better accuracy compare to Method III, and in case study 2, the accuracy it gave even exceed those generated using Method I. Although we note that the spring constant $K_S$ played an important role in the accuracy of Method II, there is a limitation to this: for a particular simulation, the accuracy improve with increase value of $K_S$ for a certain range of $K_S$; and increase again if exceed that range. To further reduce the error with increasing $K_S$ value, we would need to further reduce the simulation step size. The order of average total formation error each method gave in the four case studies performed is listed in Table 7-5. From Table 7-5, we can see that except in case study 1, Method III provide better accuracy in maintaining the constraints – thus give a better

performance for formation maintenance. Compare to Method II, which is the commonly use potential field approach in maintaining formation, these case studies show that Method III provide a better accuracy in formation maintenance at an order of at least $10^4$.

| | Method I | Method II | Method III |
|---|---|---|---|
| Case Study 1 | $10^{-14}$ | $10^{-5}$ | $10^{-9}$ |
| Case Study 2 | $10^{-6}$ | $10^{-3}$ | $10^{-7}$ |
| Case Study 3 | $10^{-7}$ | $10^{-4}$ | $10^{-15}$ |
| Case Study 4 | - | $10^{-4}$ | $10^{-12}$ |

**Table 7-5: Order of average total formation error obtained in all four case studies using Method I, Method II, and Method III.**

### 7.5.2 Computation Time

Another important aspect to evaluate each of these methods is computational speed. Computation speed is important since an algorithm that required less computation time is more likely for real-time application. We also realize that with different hardware available, we can improve the computation time that we have been recorded in our case studies. Further, since our code is not optimized, it is also possible to improve the computation time by optimized the code. Despite the factors mentioned that can affect the computation time used in each method, we would like to provide a comparison on the computation time taken for each method, relatively.

As we mentioned in case study 1, that since we are using ODE5, a fixed time-step solver, changing the parameters in the dynamic equation does not affect the computation time significantly. Thus, we will use the *average* computation time taken instead of focus on particular simulation. The average computation time taken for each method in the four case studies is given in Table 7-6.

| | Method I | Method II | Method III |
|---|---|---|---|
| Case Study 1 | 38.21 | 56.66 | 101.07 |
| Case Study 2 | 147.38 | 145.27 | 148.52 |
| Case Study 3 | 24.27 | 45.12 | 159.62 |
| Case Study 4 | - | 43.56 | 85.67 |

**Table 7-6: Average computational time in seconds used for each method in all four case studies.**

Except in case study 2, where the computation time for all three methods is almost the same, in other case studies, we can see their difference. The reason the computation time in case study is almost the same for all three methods is because the numerical gradient finding method were used in this case study. The numerical gradient finding account for most of the computation time in this case study. In other case study, we generally say that Method I used least computation time to perform the simulation and Method III used the most computation time. In case study 1, 3, & 4, the gradient are determine analytically, thus the part that consume most time is the calculation of the inverse of the Jacobian matrix. By looking at the dynamic equation formulated using each method, Method III required to compute an inverse matrix 3 times where Method I only compute once. The reason account why Method II take more computation time than Method I is because while approximating the Lagrange Multiplier using spring and damper, we make the equation numerically stiff to compute. From the case studies we performed, we observed that Method III, in general take more computation time than Method II, at least twice the time used for Method II.

### 7.5.3 Decentralized/ Parallel Processing Ability

A centralized controller allows high-level control necessary for tasks such as taking seismographic readings or configuring a network of satellite dishes. These tasks demand a precision and directed intentionality which cannot merely emerge from low

level interaction. Another advantage of centralized control is that it facilitates human interface. A commander must be able to tell a squadron of planes to abort their mission. If there is centralized control, the human will need only to issue one command. Without centralized control, it will be more difficult, though not impossible, for the command to reach each agent.

On the other hand, a totally centralized approach places immense computational demands on the centralized controller and often prohibits real time action. For cooperative payload transport, a balance between these centralized and decentralized control might be needed. In this thesis, we did not evaluate which type of control is more suitable for cooperative payload transport but we provide the formulation such that our algorithm can be use in not only centralized manner, but also in a decentralized manner.

As we formulated in Chapter 6, Method I is a centralized method, where the $\left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\right]^{-1}$ term needs the full state information of the system. On the other hand, Method III can be implemented in a decentralized manner. Decentralized computation capability is important, because it allows the algorithm to run on each individual processor that located onboard on each robot. Method III, as we formulated in Chapter 6, can be partially decentralized. Figure 7-15 summarized the decentralize capabilities for these three methods.

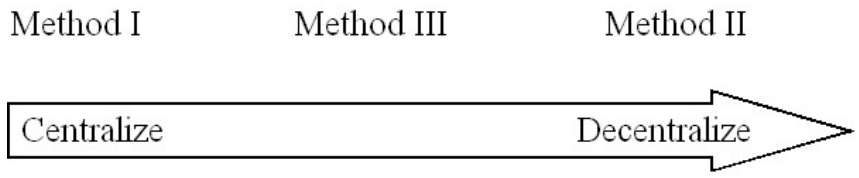



**Figure 7-15: Method I is a centralized method, method II is a decentralized method, while method III, lies between these two.**

### 7.5.4 Formation Related Concerns

Case study 2 and 3 shows the ability of each method in maintaining the formation constraint when the constraint is changing throughout the motion. From case study 4, we can see that Method I poses a limitation where the method failed when three robots aligned in a straight line. When taking formation constraint, which is a holonomic constraint, to obtain the Jacobian of the constraint, precaution must be given to avoid the inverse of the Jacobian become singular. This singularities happened when the holonomic constraints is arrange in certain shape, or direction. Method III, on the other hand, has the same limitation in very limited cases. While formulate the dynamic equation using Method II, we need to take the inverse of the Jacobian of the dependent velocities that we choose. Here, depending on the selection of the dependent velocities, it posses limitation on the formation shape such that the inverse of the Jacobian is not singular. Nonetheless, Method III only limits on certain orientation on the formation shape but have more flexibility on formation shape compared to Method I.

Method II, since no inverse of Jacobian needed in its formulations, does not posses such limitation. From these case studies, we observed that Method II in general does not causes computation problem such as inversion of a matrix is singular, it does not provide good accuracy in maintaining formation shape. Method III, as comparison, does posses limitation on formation shape, but provides much better accuracy than Method II in maintaining formation.

### 7.5.5 Summary

In this chapter, four case studies were performed to study three methods that used to formulate the dynamic equation for motion planning of robot collectives in formation using potential approach. Their accuracy, computational time, decentralization abilities, and formation limitation were compared and discussed. We would like to provide our observation based on these case studies, to evaluate how each method performed, relatively to each other, and give our comments on using these methods for motion planning for robot collectives in formation.

While most of the reseach on motion planning for robot collective focus on derive various methods to maintain formation to avoid collision, they do not required a strict formation. i.e., only a loose formation is required. In constrast, we want a method that can strictly maintain formation requirement for application such as cooperative payload transport. We have shown, using various case studies, the commonly used potential based formulation for formation constraint gives relatively poor performance compared to constraint manifold projection method.

Method II, the penalty formulation approach, which is widely used as potential-based method in maintaining formation, while provide decentralized computation capabilities, moderate computation time and no formation limitations, gives relatively poor ability in maintaining strict formation requirement. Further, to achieve better performance, the penalty factor $\kappa$ may have to set at a large value, which potentially creates a stiff equation. Method III, the constraint manifold projection method, which generally gives a good performance in maintaining formation compare to Method III and

have moderate decentralization capabilities, at a cost of higher computational time and limitation on formation shape.

However, we note that some of the limitations on Method II can be overcome. For example, computation time can improve with a better processor, and the limitation on formation shape can be solved by optimized the Jacobian of the dependent velocities such that it move in a moves in a direction that will not cause the Jacobian matrix to be singular. In summary, we tabulated the result that we observed in Table 7-7.

| | Method I | Method II | Method III |
|---|---|---|---|
| Accuracy | Moderate | Poor | Good |
| Computation Speed | Moderate | Good | Poor |
| Decentralization | Poor | Good | Moderate |
| Ability to change formation | Poor | Good | Moderate |

**Table 7-7: Comparison table for three different approaches used in solving constrained dynamics.**

# 8 Conclusion

In this thesis, we examined various approaches for motion planning of group of robots within a potential framework. In the first part of this thesis, we studied potential functions to determine which potential functions is more suitable to use in creating the potential field of a given workspace. Various kinds of local potential functions commonly used in the literature were studied in Chapter 3. We studied the characteristic and limitations of local potential functions, and found that navigation function (Chapter 4) is more suitable to use in our study. Various simulations were performed in Chapter 5 to gain valuable insights piori to formulate the dynamic equation for group of robots in Chapter 6. Chapter 6 starts the second part of the thesis, in which we formulate the dynamic equation for group of robot using constrained dynamics. In Chapter 7, various simulations were performed to study the performance of each method in formation maintenance. We also compare the effectiveness of each method from different aspects.

## 8.1 Research Questions Revisited

In this section, we will revisit the research questions posed in Chapter 1, and provide the answer based on our studies.

*Which type of potential function is more suitable for use in motion planning for robot groups?*

To answer this question, various commonly used potential functions were studied. In particular, we studied FIRAS function, GPF function, Ge's new potential function, superquadric potential function, and harmonic potential function. These were local potentials, in which each of them had some form of limitations. Navigation function, on the other hand, is a perfect candidate since it provides a unique minimum at the target location. Although we need global information to construct such navigation function, we believe this is the best candidate to construct the potential field of a workspace to use for motion planning of group of robots.

*How can we extend the potential field framework to help maintain formations? And how may this need to be further extended to realize the tight formation contact required of cooperative payload transport?*

In Chapter 6, we treat the system of group of robots as a constrained mechanical system and thus we can formulate the dynamic equation for group of robots using constrained dynamic. The motion planning for the system of group of robots become solving the forward dynamics of a constrained mechanical system. This forward dynamic is solved using three different methods, namely the direct Lagrange elimination method, the penalty method, and the constraint manifold projection method. Simulations were performed using these methods and their performance in maintaining tight formation is studied along with other considerations such as decentralize capabilities, computational time, and change of formation ability.

## 8.2 List of Contributions

In this section, the principal contributions of this thesis will be listed.

- ***Evaluation of various potential functions:*** Various local potential functions were studied in the thesis in Chapter 2. These included FIRAS Function, GPF Function, Superquadric potential function, Ge New potential, and harmonic potential field. Beside this various motion planning for single robot using these local potential functions were performed in Chapter 5. This enabled to examine the general characteristics and limitations of these local potential functions.

- ***Development of a GUI tool for generating navigation functions:*** Generating a potential space with a unique minimum can be challenging. As shown in Chapter 4, the navigation function approach helps to overcome this. Although it is a global method, it guaranteed a unique minimum at the target location. In particular, we developed a designer tool in the form of a GUI that allowed us to visualize the potential field of a workspace generated using navigation function and modified the $\kappa$ value to obtain a smooth potential field, to use in motion planning for group of robots.

- ***Develop the group motion planning problem as a constrained forward dynamic simulation:*** By treating the formation constraints as holonomic constraints we treat the system of group of robots as a constrained mechanical system. The motion planning can now be treated as forward dynamic simulation of a constrained mechanical system. This enable use to systematically develop various approaches for group motion planning while maintaining formations.

- ***Evaluation of three different methods in solving motion planning problem of group of robots*** – Leveraging the insight from constrained mechanical system simulation literature, three approaches were developed. Namely, Method I, direct Lagrange multiplier elimination approach; Method II, penalty formulation approach; and Method III, constraint manifold projection based approach. By setting Method I as the benchmark, we specifically compare Method II and Method III, for their performance in maintaining tight formation for application such as formation payload transport.

- ***Critical evaluation of the performance (formation maintenance) by the three approaches*** – from various case studies, we confirmed that Method III, the constraint manifold projection approach, provide a better formation maintenance ability than the commonly use penalty formulation approach (Method II). This provides us an alternative to use the penalty formulation approach for formation maintenance when tight formation constraint is required. Although computation time for Method III is higher than Method II, the decentralization capability is lower than Method II, we noticed that Method III provides us a much better accuracy in formation maintenance.

## 8.3 Future Work

In this section, list of future work can be done base on our current result are summarized as follow:

- ***Provide a way to avoid Jocobian become singular*** – In Method II, there is a possibility that the Jacobian matrix of the dependent velocities $\mathbf{J}_\mathbf{d}(\mathbf{q})$ become singular.

The configuration at which this singularity happens depends on the selection of the dependent velocities. We can find the condition at which the singularity happened by looking at determinant of $\mathbf{J}_\mathbf{d}(\mathbf{q})$, $\det(\mathbf{J}_\mathbf{d}(\mathbf{q}))$. One possible way of eliminating this singularity is to maximize the manipulability of $\mathbf{J}_\mathbf{d}(\mathbf{q})$.

- ***Incorporate nonholonomic constraints in the formulation*** – nonholonomic constraints can be included in the formulation using Method III, the constraint manifold projection method. Our goal of this research is to finally incorporate a WMR and WMM in the formulation using this framework.

- ***Implement a more efficient gradient finding method by utilizing information from each robot*** – in our algorithm; each robot does not share their gradient information. Each robot need to search the steepest decent gradient by searching its surrounding. This is a time consuming process. One possible way of doing this is approximate the gradient of a workspace by using the potential information on each robot. Since the potential of each robot is known, a direction vector can be determined by pointing to the robot that has lowest potential value, and move to that direction.

- ***Implement the alrigthim in a decentralized computation manner*** – the first step towards this goal is to write the algorithm using Simulink™ and verified the results. Eventually, we would like to be able to test our algorithm using real robots.

# Bibliography


[1] Y. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 1-23, 1997.

[2] L. Moreau, R. Bachmayer, and N. E. Leonard, "Coordinated gradient descent: A case study of lagrangian dynamics with projected gradient information," in *Proc. IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control*, Seville, Spain, 2003.

[3] P. Ogren and N. E. Leonard, "Obstacle avoidance in formation," in *Proc. IEEE International Conference on Robotics and Automation*, 2003.

[4] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transactions on Automatic Control*, 2004.

[5] M. Vendittelli, J. P. Laumond, and C. Nissoux, "Obstacle distance for car-like robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 678-691, 1999.

[6] S. S. GE and Y. J. CUI, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, pp. 207-222, 2002.

[7] V. Kumar, M. Zefran, and J. Ostrowski, "Motion planning and control of robots," in *Handbook of industrial robotics*, J. Wiley and Sons, 1997.

[8] U. Ascher, H. Chin, L. Petzold, and S. Reich, "Stabilization of constrained mechanical system with does and invariant manifolds," *Mechanical Structures & Machines*, vol. 23, pp. 135-158, 1995.

[9] J. G. d. Jalon and E. Bayo, *Kinematic and dynamic simulation of multibody systems: The real time challenge*. New York: Springer-Verlag, 1994.

[10] E. J. Haug, *Computer aided kinematics and dynamics of mechanical systems: Basic methods*: Prentice Hall, 1989.

[11] A. A. Shabana, *Dynamics of multibody systems*. New York: Wiley, 1989.

[12] V. I. Arnold, *Mathematical methods of classical mechanics*, vol. 60. New York: Springer-Verlag, 1978.

[13] A. Witkin and W. Welch, "Fast animation and control of non-rigid structures," in *Proc. 1990 Siggraph*, vol. 24, pp. 243-252, 1990.

[14] A. Witkin, M. Gleicher, and W. Welch, "Interactive dynamics," in *Proc. 1990 Symposium on 3-D Interactive Graphics*, vol. 24, 1990.

[15] J. Wang, C. M. Gosselin, and L. Cheng, "Dynamic modeling and simulation of parallel mechanisms using the virtual spring approach," in *Proc. ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Coference*, Baltimore, Maryland, 2000.

[16] X. Yun and N. Sarkar, "Unified formulation of robotic systems with holonomic and nonholonomic constraints," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 640-650, 1998.

[17] N. Sarkar, X. Yun, and V. Kumar, "Control of mechanical systems with rolling constraints: Application to dynamic control of mobile robots," *International Journal of Robotic Research*, vol. 13, no. 1, pp. 55-69, 1994.

[18] W. A. Khan, "Distributed dynamics of systems with closed kinematic chains," M. Eng. Thesis, Department of Mechanical Engineering, McGill University, Montreal QC, 2002

[19] W. A. Khan and V. Krovi, "Comparison of two alternate methods for distributed forward dynamic simulation of a four-bar linkage," in *Proc. WORKSHOP on Fundamental Issues and Future Research Direction for Parallel Mechanisms and Manipulators*, Quebec City, Quebec, Canada, 2002.

[20] V. Kumar, M. Zefran, and J. Ostrowski, "Motion planning and control of robots," in *Handbook of industrial robotics*, J. Wiley and Sons, October 1997.

[21] R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. no. 3, pp. 190-197, 1983.

[22] J. F. Canny, *The complexity of robot motion planning*. Cambridge, MA: MIT Press, 1988.

[23] J. C. Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.

[24] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specific paths," *International Journal of Robotic Research*, vol. 4, no. No. 3, pp. 3-17, 1985.

[25] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688-1703, 1985.

[26] J. R. Andrews and N. Hogan, "Impedance control as a framework for implementing obstacle avoidance in a manipulator," in *Proc. Control of Manufacturing Processes and Robotic Systems*, pp. 243-251, 1983.

[27] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 785-797, 1991.

[28] Y. Uno, M. Kawato, and R. Suzuki, "Formation and control of optimal trajectory in human multi-joint arm movement," *Biological Cybernetics*, vol. 61, pp. 89-101, 1989.

[29] B. Krogh, "A generalized potential field approach to obstacle avoidance control," in *Proc. ASME Conf. of Robotic Research: The Next Five Years and Beyond*, Bethlehem, Pennsylvania, 1984.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotic Research*, vol. 5, no. 1, pp. 90, 1986.

[31] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proc. IEEE Conference on Decision and Control*, Orlando, FL, 2001.

[32] P. K. K. Jin-Oh Kim, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, 1992.

[33] S. S. GE and Y. J. CUI, "Path planning for mobile robots using new potential functions," in *Proc. Asian Control Conference*, Shanghai, 2000.

[34] E. Rimon and D. E. Koditchek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, 1992.

[35] R. Volpe and P. Khosla, "Manipulator control with superquadric artificial potential functions: Theory and experiments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 6, 1990.

[36] S. S. GE and Y. J. CUI, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615, 2000.

[37] J. O. Kim and P.Khosla, "Real-time obstacle using harmonic potential functions," in *Proc. IEEE Conference on Robotics and Automation*, Sacramento, CA, pp. 790-796, 1991.

[38] C. I. Connolly, "Applications of harmonic functions to robotics," in *Proc. IEEE International Symposium on Intelligent Control*, pp. 498-502, 1992.

[39] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," in *Proc. IEEE Conference on Robotics and Automation*, Sacramento, CA, pp. 790-796, 1991.

[40] R. W. Beard and T. W. McLain, "Motion planning using potential field tutorial," Provo, UTAH, 2003.

[41] A. G. Feldman, "Change in the length of the muscle as a consequence of a shift in equilibrium in the muscle-load system.," *Biofizika*, vol. 19, pp. 534-538, 1974.

[42] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Robotics research*, Eds. M. Brady and R. Paul, MIT Press, Cambridge MA, 1984, pp. 735-747.

[43] P.Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Proc. 1998 IEEE Internaitional Conference on Robotics and Automation.*, Philadelphia, PA, April 26-281988.

[44] A. Jaklic, A. Leonardis, and F. Solina, "Segmentation and recovery of superquadrics," in *Computational imaging and vision*, vol. 20, Kluwer, Dordrecth, 2000, pp. Chapter 2.

[45] M. Gardiner, " The superellipse: A curve that lies between the ellipse and the rectangle," *Scientific American*, 1965.

[46] S.Akishita, S. Kawamura, and K. Hayashi, "Laplace potential for moving obstacle avoidance and approach of a mobile robot," in *Proc. Japan-USA Symposium on Flexible Automation, A PacificRim Conference*, Kyoto, Japan, pp. 139-142, 1990.

[47] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in *Proc. 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[48] C. I. Connolly and R. A. Grupen, "Harmonic control [robot applications]," in *Proc. Intelligent Control, 1992., Proceedings of the 1992 IEEE International Symposium on*, pp. 503-506, 1992.

[49] J. Guldner and V. I. Utkin, "Sliding mode control for an obstacle avoidance strategy based on an harmonic potential field," in *Proc. Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, pp. 424-429 vol.1, 1993.

[50] C. I. Connolly, "Applications of harmonic functions to robotics," in *Proc. Intelligent Control, 1992., Proceedings of the 1992 IEEE International Symposium on*, pp. 498-502, 1992.

[51] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 338-349, 1992.

[52] D. E. Koditchek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, pp. 412-442, 1990.

[53] E. Rimon and D. E. Koditchek, "The construction of analytic diffeomorphisms for exact robot navigation on star world," *IEEE Transactions on Robotics and Automation*, 1989.

[54] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 5, pp. 501-518, 1992.

[55] E. Rimon and D. E. Koditschek, "Exact robot navigation in geometrically complicated but topologically simple spaces," in *Proc. IEEE International Conference of Robotic and Automation*, Cincinnati, OH, pp. 1937-1943, May 1990.

[56] D. E. Koditschek, "Autonomous mobile robots controlled by navigation functions," in *Proc. IEEE/RSJ International Workshop on Intelligent Robots and System*, Tsukuba, Japan, pp. 639-645, 1989.

[57] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Transactions of the American Mathematical Society*, 1989.

[58] D. E. Koditchek and E. Rimon, "Exact robot navigation using cost functions: The case of distinct spherical boundaries on $e^n$," Yale University 8803, Jan 1988 1988.

[59] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905-908, 2001.

[60] P. K. C. Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *Journal of Robotic Systems*, vol. 8, no. 2, pp. 177-195, 1991.

[61] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A feedback architeture for formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777-790, 2001.

[62] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous Robots*, vol. 4, no. 4, pp. 387-403, October 1997.

[63] B. J. Young, R. W. Beard, and J. M. Kelsey, "A control scheme for improving multi-vehicle formation maneuvers," ACC01-IEEE1575, in *Proc. of the American Control Conference*, Arlington, VA, 2001.

[64] P. Ogren, E. Fiorelli, and N. E. Leonard, "Formations with a mission: Stable coordination of vehicle group maneuvers," in *Proc. 15th International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[65] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 6, pp. 933-941, 2003.

[66] J. R. T. Lawton, B. J. Young, and R. W. Beard, "A decentralized approach to elementary formation maneuvers," in *Proc. 2000 IEEE International Conference of Robotics and Automation*, San Francisco, CA, 2000.

[67] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947-951, 2001.

[68] P. Ogren, M. Egerstedt, and X. Hu, "A control lyapunov function approach to multi-agent coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 847-851, 2002.

[69] A. A. Masoud and S. A. Masoud, "Motion planning in the presence of directional and obstacle avoidance contraints using nonlinear, anisotropic, harmonic potential fields," in *Proc. 2000 IEEE International Conference on Robotics & Automation*, San Francisco, CA, pp. 2944-2951, 2000.

[70] C. I. Connolly, "Harmonic functions and collision probabilities," in *Proc. Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3015-3019 vol.4, 1994.

[71] J. Guldner, V. I. Utkin, H. Hashimoto, and F. Harashima, "Obstacle avoidance in r<sup>n</sup> based on artificial harmonic potential fields," in *Proc. Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3, pp. 3051-3056 vol.3, 1995.

[72] J. Guldner and V. I. Utkin, "Sliding mode control for an obstacle avoidance strategy based on an harmonic potential field," in *Proc. IEEE 32nd Conference on Decision and Control*, San Antonio, Texas, pp. 424-429, 1993.

[73] A. A. Masoud, "Evasion of multiple, intelligent puruers in a stationary, cluttered environment: A harmonic potential field approach.," in *Proc. 2002 IEEE International Symposium on Intelligent Control*, Vancouver, Canada, pp. 654-659, 2002.

[74] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. International Conference on Robotics and Automation*, Sacramento, California, 1991.

[75] E. Rimon, "Exact robot navigation using artificial potential functions," Ph.D. Dissertation Thesis, Electrical Engineering, Yale University, New Haven, CT, 1990

[76] D. E. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *Proc. IEEE International Conference on Robotics and Automation*, Releigh, NC, pp. 1-6, March 1987.

[77] D. Luca, Alessandro, and G. Oriolo, "Local incremental planning for nonholonomic mobile robots," in *Proc. 1994 IEEE Conference on Robotic and Automation*, San Diego, pp. 104-110, 1994.

[78] H. Goldstein, *Classical mechanics*, 6 ed. Reading, Mass.: Addison-Wesley, 1959.

[79] U. M. Ascher, D. K. Pai, and B. P. Cloutier, "Forward dynamics, elimination methods, and formulation stiffness in robot simulation," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 749-758, 1996.

[80] M. R., Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. New York: CRC Press, 1993.

[81] R. Barzel and A. H. Barr, "A modeling system based on dynamic constraints," *Computer Graphics*, vol. 22, pp. 179-188, 1988.

[82] G. d. Jalon and Bayo, *Kinematic and dynamic simulation of multibody system: The real-time challenge*. New York: Springer-Verlag, 1994.

[83] J. Baumgarte, "A new method of stabilization for holonomic constraints," *ASME Journal of Applied Mechanics*, vol. 50, pp. 869-870, 1983.

[84] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle system," in *Proc. 41th IEEE Conference on Decision and Control*, vol. 3, pp. 2965-2971, 2002.

[85] T. Eren, "Rigid formations of autonomous agents," Department of Electrical Engineering and Computer Science, Yale University, New Haven, 2003

[86] T. Eren, P. N. Belhumeur, B. D. O. Anderson, and A. S. Morse, "A framework for maintaining formations based on rigidity," in *Proc. 2002 IFAC Congress*, Barcelona, Spain, 2002.

[87] T. Eren, P. N. Belhumeur, and A. S. Morse, "Closing ranks in vehicle formation based on rigidity," in *Proc. 41th IEEE Conference on Decision and Control*, Las Vegas, pp. 2959-2964, 2002.

[88] E. Hairer and G. Wanner, *Solving ordinary differential equations ii, stiff and differential-algebraic problems*. Berlin: Springer-Verlag, 1991.

[89] J. D. Lambert, *Numerical methods for ordinary differential equations. The initial value problem*. Chichester: Wiley, 1991.

[90] L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka, "Solving index-1 daes in matlab and simulink," *SIAM Review*, vol. 41, no. 3, pp. 538-552, 1999.

[91] J. A. Weideman and S. C. Reddy, "A matlab differentiation matrix suite," *ACM Transaction on Mathematical Software (TOMS)*, vol. 26, no. 4, pp. 465-519, 2000.

[92] L. F. Shampine and M. W. Reichelt, "The matlab ode suite," *Journal on Scientific Computing*, vol. 18, no. 1, pp. 1-22, 1997.

[93] C. Moler. "Numerical computing with matlab," SIAM. 2004. Available: http://www.mathworks.nl/moler/chapters.html

[94] R. Ashino, M. Nagase, and R. Vaillancourt, "Behind and beyond the matlab ode suite," *Computers & Mathematics Application*, vol. 40, no. 4-5, pp. 491-512, 2000.

[95] MathWorks. "1510-differential equations in matlab," MathWorks. 2004. Available: http://www.mathworks.nl/support/technotes/1500/1510.html

# Appendix

## A.1 Background on Constrained Dynamics

In classical mechanics, constraints play a role as a means of describing physical system. Taking the example of sphere sliding freely on a rigid wire, one important observation about its dynamics is that "the sphere stays on the wire, no matter what". This observation, or this constraint, has physical consequences.

Treating the sphere as a particle, the motion equation is governed by:

$$\vec{F} = m\vec{a} \tag{G.10}$$

Equation (G.10) gives the relation between its motion and the total force that act on the particle. The force and acceleration lies in the same direction, with their magnitudes scaled by the particle' mass, *m*. However, with the sphere-on-wire constraint implies that the sphere will never accelerate in a way that moves it off the wire, despite the magnitude of the force applied. If we think of a straight wire, this constraint just means that the sphere's acceleration, and therefore the total force, must lie tangent to the wire, even if the force applied to the sphere points in some other direction. This implies that the force $\vec{F}$ in Equation (G.10) cannot be the *total force*. Instead, there must also exist a constraint force, $\vec{F}_C$ such that the total force:

$$\vec{F} = \vec{F}_a + \vec{F}_C = k\vec{t} \tag{7.11}$$

Where $\vec{t}$ is the wire's tangential direction and $k$ is some scalar.

Equation (7.11) simply means that the constraint force $\vec{F}_C$ is whatever force needs to apply to the sphere to make the sphere accelerate inconsistent with the constraint.

*Constrained Dynamics* is thus concerned with making the system behavior consistent with the force of constraint. The mathematics of constrained dynamic is hardly new, and was being used extensively in the area of computer graphics. In our work, we use the idea of constraint force to formulate the dynamics equation of group of robots, such that each robot in a group maintained a formation throughout their motion.

### A.1.1 Restoring Force

Ideally, one would like to treat the constraint force as a restoring force that only take effect when the constraint is violated. As in the example of the sphere on wire, a more accurate physical description would show the sphere and wire deforming a little bit as we tried to pull them apart, inducing a restoring force that cancels the applied force, such that constraint will not be violated. One could think of this restoring force as a rubber band connecting the sphere to the wire, with a force:

$$\vec{F}_{Spring} = -kc \tag{7.12}$$

Where $c$ is the displacement of the sphere apart the wire and $k$ is the stiffness of the wire. The same scenario happen in the case of group of robots, each robot is to maintain a certain distance $c$ apart; one can model this using a spring with a rest length $c$. Apparently, this formulation gives a more accurate physical description of the model in both sphere on wire or the group of robots case.

In order for the constraint to hold approximately at equilibrium, we need:

$$c \leq \varepsilon \tag{7.13}$$

That is, we want the displacement $c$ is controlled to be equal or less than the *allowable displacement* $\varepsilon$. To ensure Equation (7.12) hold, the stiffness $k$ must be sufficiently large that the restoring force $-kc$ can cancels any applied force, i.e. the value of $k$ is to determine from:

$$k = \frac{\vec{F}_{\max}}{\varepsilon} \tag{7.14}$$

Where $\vec{F}_{\max}$ is the largest force applied to the system. From Equation (7.14), we can see that in order to make $\varepsilon$ small, we must make $k$ large accordingly. However, the problem with making $k$ large is that it produces differential equations that are numerically intractable, or so called the *stiff* equations.

To understand the problem with stiff equation, we can use the sphere on wire example again. When we try to pull the sphere off the wire from rest with an applied force $\vec{F}_{\max}$, the applied force begin to displace the sphere, and the rubber band starts to exert a restoring force proportional to the displacement. The restoring force balances the applied one when $c = \varepsilon$. When we solve these differential equations numerically, the distance travel by the sphere accelerating from rest under force $\vec{F}_{\max}$ in a single time step $\Delta t$ must be in the order of $\varepsilon$ to avoid substantial overshoot and instability. As a consequence, the step size must be so small that the largest permitted applied force $\vec{F}_{\max}$ makes objects move only a negligible distance $\varepsilon$ in a single time step, which means you never get anywhere. Thus, although using stiff spring force can be a better description of the system, they are not a good way to enforce constraint numerically.

### A.1.2 Constraint Force

The problem of stiffness describe in the previous section can be avoided by letting $\varepsilon$ go to zero, and thus stiffness $k$ goes to infinity. In this limiting case, the rule $c = 0$ governs the system exactly. Since there is no displacement, and hence nothing to restore, the restoring force is renamed as *constraint force*. These constraint forces, in addition to depending on the state of the system and on time, as most forces do, it depends on other forces. In this section, we systematically develop a system of linear equations that yield constraint forces, in which, when added to the ordinary applied force, ensure the system to accurately satisfy the constraints.

In order to make this result general, we consider the general system that describes by the equation of motion of the following form:

$$\mathbf{M}\ddot{\mathbf{q}} = \vec{F}_a + \vec{F}_c \tag{7.15}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{q}$ is the vector of the system's independent variables, $\vec{F}_a$ is the vector of known applied forces, and $\vec{F}_c$ is the vector of unknown constraint forces. Equation (7.15) is the generalized form of Equation (G.10). Ultimately, we would like to solve for $\ddot{\mathbf{q}}$, given $\dot{\mathbf{q}}$ and $\mathbf{q}$, and write in state-space form, allowing us to integrate the differential equation forward through time.

### A.1.3 Virtual Work

Equation (6.10) states that the system's acceleration must not move the constraint functions from zero, but in an underconstrained system, a whole subspace of such "legal" acceleration exist. Given a constraint force that satisfies Equation (6.10), nothing said so

far prohibiting us from adding any additional force to it, as long as the acceleration it induces lies in that legal subspace. To remove this ambiguity, it is reasonable to add a restriction: that the constraints never add or remove energy from the system. In other words, is to say the constraints may do no work. To guarantee this we require that the work done by the constraint force vanish, under any small displacement of the system that are consistent with the constraints. Hence, for every legal displacement $d\mathbf{q}$, $\vec{F}_c$ must satisfy $\vec{F}_c d\mathbf{q} = 0$, which simply requires the constraint force to point in a direction in which the system is restricted to move. This requirement, known as the principle of virtual work, is not derived from anything else. It is a restriction on the constraints to be considered. In the case of a single scalar constraint $c$, the "legal" displacements are those lying in the tangent plane to the surface $c = 0$. Because the gradient $\partial c / \partial \mathbf{q}$ is normal to the tangent plane, this means that every legal displacement must satisfy $\left(\partial c / \partial \mathbf{q}\right) d\mathbf{q} = 0$. The forbidden displacements are those that satisfy $d\mathbf{q} = \boldsymbol{\lambda}\left(\partial c / \partial \mathbf{q}\right)$ for any scalar $\boldsymbol{\lambda}$.

The multidimensional generalizations of the tangent plane and the gradient direction are the *null space* and *null space complement* of the constraint Jacobian matrix. The null space contains the displacements satisfying:

$$\frac{\partial \mathbf{C}}{\partial \mathbf{q}} d\mathbf{q} = \mathbf{0} \tag{7.16}$$

While the null space complement contains those that satisfy:

$$d\mathbf{q} = \boldsymbol{\lambda} \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \tag{7.17}$$

For some vector $\boldsymbol{\lambda}$. Viewing the constraint vector as a collection of scalar constraints, the null space is the set of vectors which lie in every constraint's tangent plane, while the null

space compliment is the set of linear combinations of the constraint gradients. To lie in the null space, the constraint force must therefore satisfy:

$$\vec{F}_c = \left( \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \right)^T \boldsymbol{\lambda} = \mathbf{J}^T(\mathbf{q})\boldsymbol{\lambda} \tag{7.18}$$

To understand what Equation (7.18) means, it helps to regard the matrix $\mathbf{J}(\mathbf{q})$ as a collection of vectors, each of which is the gradient of one of the scalar constraint function comprising $\mathbf{C}$. Since our fundamental requirement is that $\mathbf{C} = \mathbf{0}$, these gradients are normal to the constraint hypersurfaces, representing the state-space directions in which the system is not permitted to move. The vectors that have the form $\mathbf{J}^T\boldsymbol{\lambda}$ are the linear combinations of these gradient vectors, and hence span exactly the set of prohibited directions. Restricting the constraint force to this set ensures that its dot product with any legal displacement if the system will be zero, which is exactly what the principle of virtual work demands.

In matrix parlance, the set of vectors $\mathbf{J}^T\boldsymbol{\lambda}$ is known as the *null space complement* of $\mathbf{J}$. The *null space* of $\mathbf{J}$ is the vectors (let say, $\mathbf{v}$) that satisfy $\mathbf{J}\mathbf{v} = 0$. The null space vectors are the *legal displacements*, while the null space complement vectors are the *prohibited* ones. The component of $\boldsymbol{\lambda}$ are known as the Lagrange multiplier. These quantities, which determine how much of each constraint gradient is mixed into the constraint force, are the unknowns for which we must solve.

By using this idea, the dynamics of group or robots can be formulated as Lagrangian equations of the first kind [12] as:

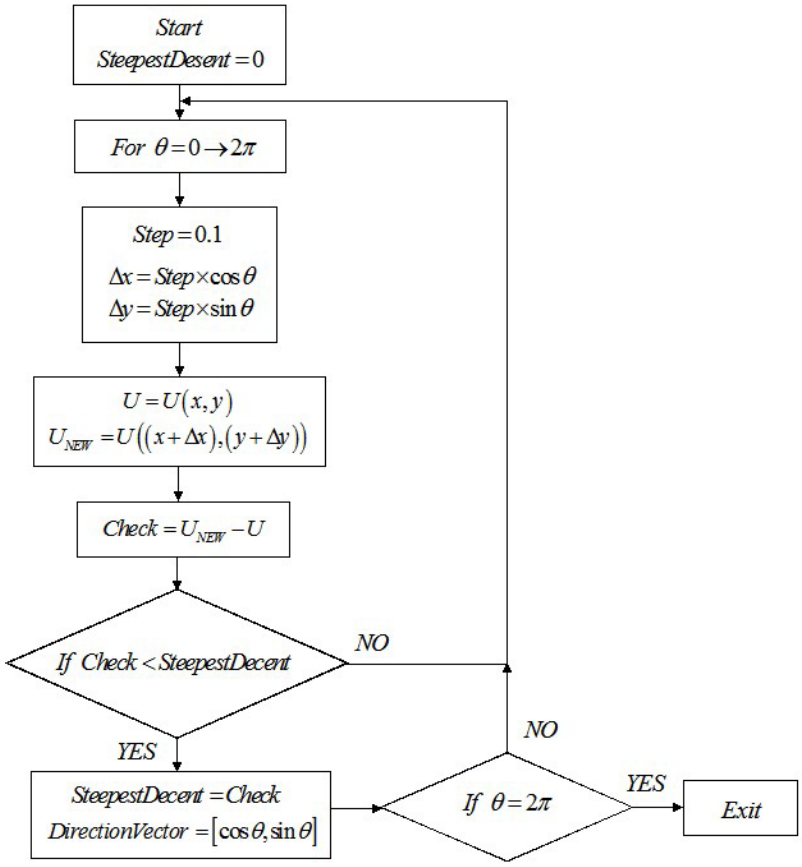
$$\dot{\mathbf{q}} = \mathbf{v} \tag{7.19}$$

$$\mathbf{M}(\underset{\sim}{\mathbf{q}})\dot{\mathbf{v}} = \mathbf{f}(\underset{\sim}{\mathbf{q}}, \mathbf{v}, t, \mathbf{u}) - \mathbf{J}(\underset{\sim}{\mathbf{q}})^T \boldsymbol{\lambda} \tag{7.20}$$

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0} \tag{7.21}$$

where $\mathbf{q}$ is the *n*-dimensional vector of generalized coordinates; $\mathbf{v}$ is the *n*-dimensional vector of generalized velocities; $\mathbf{M}\left(\underset{\sim}{\mathbf{q}}\right)$ is the $n \times n$ dimensional inertia matrix; $\mathbf{f}\left(\underset{\sim}{\mathbf{q}}, \mathbf{v}, t, \mathbf{u}\right)$ is the *n*-dimensional vector of external forces; $\mathbf{u}$ is the vector of input forces, which is $-k_f \nabla_{\mathbf{q}} \mathbf{V}$; $\mathbf{C}(\mathbf{q}, t)$ is a *m*-dimensional vector of holonomic constraints which may or may not depends on time, $t$; and $\mathbf{J}(\mathbf{q}) = \dfrac{\partial \mathbf{C}(\mathbf{q})}{\partial \mathbf{q}}$ is the *Jacobian* matrix.

## A.2 Numerical Gradient Finding Methods




The above flow chart shows the algorithm for a robot to find the steepest decent direction in a potential field created using navigation function. The goal of this algorithm is to find the direction vector in which the steepest decent occurred. A variable called *SteepestDecent* is first declared. Then, using a for-loop to go through an angle from zero to $2\pi$. For each angle, we find the potential value $U_{NEW}$ along a radius of *Step*. *Check* is the difference between the potential at each step, $U_{NEW}$; and the potential where the robot located, $U$. If value of *Check* is less than *SteepestDecent*, the direction vector is recorded and value of *SteepestDecent* is replaced by *Check*. After looping through each angle, we will get the direction of steepest decent in the potential for a particular robot.

This algorithm is computational expensive. In stead, we could utilize the potential information on each robot to approximate the steepest decent direction. Using group of three robot (A, B, C) as an example, where $U_A > U_B > U_C$. Direction vector from A to B ($\hat{n}_{AB}$) and B to C ($\hat{n}_{BC}$) can be found, an estimate of the direction vector for all of the three robots is found by $\hat{n}_{AB} + \hat{n}_{BC}$.

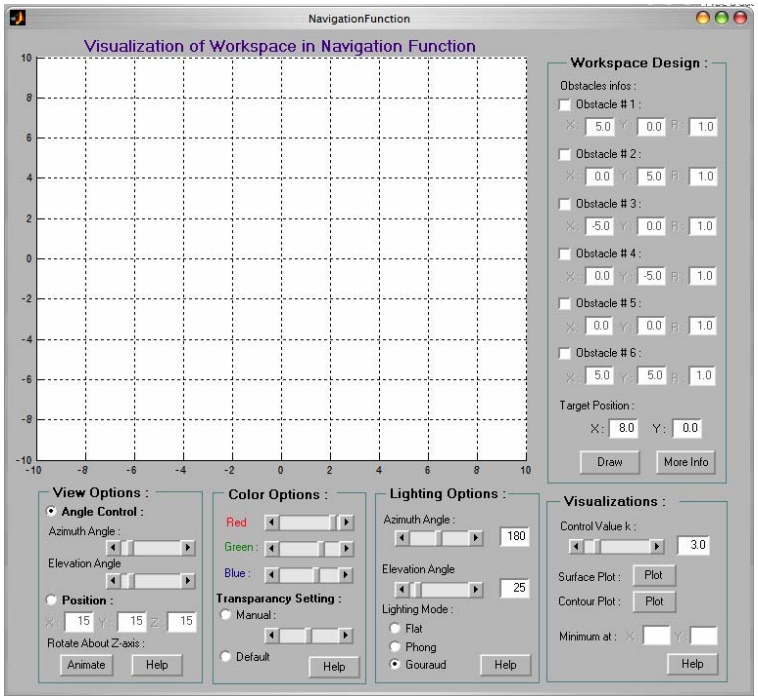
## A.3 Stiff Equation

There is much controversy about the correct mathematical definition of stiffness. The most pragmatic opinion and also the historically the first one is [88]: stiff equations are equations where certain implicit methods, in particular backward differentiation methods, perform much better than explicit ones.

The phenomenon of the stiffness appears under various aspects [89]: (1) A linear constant coefficient system is stiff is all of its eigenvalues have negative real parts and the stiffness ratio is large; (2) Stiffness occurs when stability requirements, rather than those of accuracy, constrain the step size; (3) Stiffness occurs when some components of the solution decay much more rapidly than others; and (4) A system is said to be stiff in a given interval $I$ containing $t$ if in $I$ the neighbour solution curves approach the solution curve at a rate which is very large in comparison with the rate at which the solution varies in that interval.

For example, an analytical solution of a stiff equation is given by: $u(t) = 2e^{-t} - e^{-100t}$. The solution $u(t)$ have a second term that has a negligible effect on the solution for $t$ greater than zero but can restrict the step size of a numerical solver. To limit the numerical instability, we will need to reduce the step size. If a particular equation is ‘too stiff’, and causes the step size needed to guarantee numerical stability become ‘infinitely’ small, it will then required ‘infinite’ amount of computational time. This is a situation where one would try to avoid.

## A.4 Visualization of Navigation Function Potential-MATLAB GUI




This Graphical User Interface (GUI) for visualization of potential field of a workspace generated using navigation function is created using MATLAB™ GUIDE. With this GUI, the user allowed to:

1. Design the size and location of up to six obstacles in a workspace;
2. Visualize the 3-Dimensional potential field of the workspace and provide the location of the minimum point in the workspace;
3. Visualize the contour plot of the created workspace;
4. Most importantly, allow the user to change the value of $\kappa$ to create a smooth potential field; and
5. Better visualize the potential field by: setting transparency and colors to the 3D potential field; select a better view angle or rotate the 3D potential field; and change the lighting options and light position.

Detailed information and user manual of this GUI can be downloaded from following web: http://mechatronics.eng.buffalo.edu/

## A.5 MATLAB™ ODE Functions

For detailed information on ODE functions, please see reference [90-94]. In general, MATLAB provide two types of ODE solver: Fixed time-step solvers and variable time step solver. Example of variable ODE solvers are: ode45, ode23, ode113, ode15s, ode23s, ode23b, and ode23tb. Detail information and the use of these ode solvers and can be easily found using MATLAB help, or from www.mathwork.com website. The fixed time-step solvers are: ode1, ode2, ode3, ode4, and ode5. These fixed time-step solver are included in Simulink™. On the other other hand, if not using Simulink™, they need to downloaded from MathWorks ftp site(see [95]).

The fixed time-step function and solvers are listed in table below:

| ODE Function | Solver |
|---|---|
| ode1 | First order Euler Method |
| ode2 | A second-order Euler method |
| ode3 | A third-order Runge-Kutta method |
| ode4 | A fourth-order Runge-Kutta method |
| ode5 | A fifth-order Runge-Kutta method |

In this thesis, ode5 is used in all of the simulations studies. This provides us a standard solver to compare the performance of Method I, Method II, and Method III.