

A Framework for Optimal Path Planning and Nonlinear Guidance  
for Autonomous Mobile Robots

by  
PAUL A. QUILLEN

Presented to the Faculty of the Graduate School of  
The University of Texas at Arlington in Partial Fulfillment  
of the Requirements  
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2018

Copyright © by Paul A. Quillen 2018

All Rights Reserved

I would like to dedicate this to my Mom and Dad for always being there for me and  
for encouraging me to pursue my dreams.

## ACKNOWLEDGEMENTS

I would like to first thank my supervising professor, Dr. Kamesh Subbarao for his mentorship and making the learning experience fun both inside of the classroom and out. I am very thankful for his insight and patience over the course of my studies.

Similarly, I would like to acknowledge the members of my committee from UTA, Dr. Panayiotis Shiakolas, Dr. Alan Bowling, Dr. Bo P. Wang and Dr. Gaik Ambartsoumian for their instruction and guidance both in forming my dissertation and over the course of my graduate and undergraduate studies. Moreover, I would like to thank the administrative staff in the Mechanical and Aerospace Engineering department. I am convinced this process would be impossible without them.

Another important group I would like to thank is the Space Scholar program at the Air Force Research Lab. I have met many wonderful people through my internship work at Kirtland AFB; most notably my internship mentor, Dr. Josué Muñoz. I am truly thankful for the ways he made me approach each problem, and for always having the perfect questions to further my understanding and for helping me through the research process at AFRL and as a member of my dissertation committee. Further, I would like to give special thanks to the Air Force Research Lab Space Vehicle directorate for supporting my research through award # FA945316-1-0058 and through the Space Scholar program.

To my friends, schoolmates and lab mates, thank you all for listening to my rants and for all the memories that I will take with me. I know that being around each of you is a great pleasure and it has helped me stay grounded over the duration of this work.



Most importantly, I would like to thank my family. I could not have gotten here without the encouragement of my brothers and my parents. I cannot put into words the significance of having parents who have inspired me to be curious from the very beginning and to be fearless in chasing my dreams. And to my brothers, Brian and Steven, your excellent examples and our friendly sibling rivalries have truly been a blessing.

July 26, 2018

## ABSTRACT

A Framework for Optimal Path Planning and Nonlinear Guidance  
for Autonomous Mobile Robots

Paul A. Quillen, Ph.D.

The University of Texas at Arlington, 2018

Supervising Professor: Kamesh Subbarao

The purpose of this research is to investigate methods and technology for enhancing autonomous capabilities for mobile robots. The measures of autonomy which are specifically covered in this dissertation pertain to a mobile robot's ability to make decisions and act, in other words guidance and control. This dissertation puts forth a framework using optimal path planning and nonlinear guidance techniques to address these matters. The path plans are synthesized using a numerical navigation function algorithm that will form its potential contour levels based on the minimum control effort of the system. Additionally, extensions of the path planning algorithm in the presence of uncertainty using modified versions of the RRT\* and D\* algorithms are studied. Then, an improved nonlinear model predictive control (NMPC) approach is employed to generate high-level guidance commands for the mobile robot to track a trajectory fitted along the path plan leading to the goal. A backstepping-like nonlinear guidance law is also implemented for comparison with the NMPC formulation. Furthermore, a cooperative control policy, making use of a combination of artificial potential functions (APF) and the numerical navigation function, is devised to guide

multiple mobile robots in cooperative aggregation and social foraging tasks. The results of this research are verified in simulation and validated experimentally using the mobile robot testing platforms in the Aerospace Systems Laboratory at The University of Texas at Arlington.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iv
ABSTRACT . . . . .	vi
LIST OF ILLUSTRATIONS . . . . .	xii
LIST OF TABLES . . . . .	xvi
Chapter	Page
Executive Summary . . . . .	xvii
1. Introduction and Motivation . . . . .	1
1.1 Background . . . . .	3
1.1.1 Path Planning with Control Effort and Navigation Functions . . . . .	3
1.1.2 Nonlinear Guidance Law . . . . .	8
1.1.3 Cooperative Control with Artificial Potential Functions . . . . .	13
1.2 Objectives and Contributions . . . . .	16
1.2.1 List of Contributions . . . . .	16
1.2.2 List of Published Works . . . . .	17
1.3 Dissertation Outline . . . . .	18
2. Framework and Problem Description . . . . .	20
2.1 Framework Setup . . . . .	20
2.2 Mobile Robot Kinematic Model . . . . .	24
2.2.1 Nonlinear Kinematic Equations . . . . .	25
2.2.2 Kinematics in Linear State-Space Form . . . . .	26
3. Path Planning to a Reachable State using Numerical Navigation Functions . . . . .	28
3.1 Navigation Function Generation . . . . .	30

3.1.1	Initialization Block . . . . .	30
3.1.2	Main Block . . . . .	31
3.2	Minimum Control Energy Approach . . . . .	35
3.3	Inverse Dynamics Approach . . . . .	39
3.4	Simulation Results . . . . .	41
3.4.1	Case 1: No obstacles present . . . . .	41
3.4.2	Case 2: Obstacles Present . . . . .	43
3.4.3	Effects of Changing $u_{max}$ . . . . .	45
4.	Path Planning Extensions . . . . .	48
4.1	Modified-RRT* Algorithm . . . . .	49
4.2	Modified D* Algorithm . . . . .	52
4.3	Simulation Results . . . . .	56
4.3.1	Modified RRT* Path Plan Results . . . . .	57
4.3.2	Modified D* Path Plan Results . . . . .	61
4.3.3	Comparing the Path Plan Algorithms . . . . .	62
5.	State Dependent Coefficient Based Nonlinear Model Predictive Control . . . . .	66
5.1	State Dependent Coefficient Representation of the Vehicle Kinematics . . . . .	66
5.2	Nonlinear Model Predictive Control Design . . . . .	70
5.3	Input and State Constraints . . . . .	71
5.4	Guidance Command Synthesis Using the Linear Matrix Inequality (LMI) Form . . . . .	73
5.5	Stability of Constrained, Sampled-Data, SDC-based NMPC . . . . .	75
5.6	Simulation Results . . . . .	76
6.	Nonlinear Guidance Law Design . . . . .	81
6.1	Guidance Law Design for Path Plan Algorithm . . . . .	82
6.2	Framework Guidance Law Design . . . . .	85

6.3	Guidance Design with Disturbance in the Acceleration . . . . .	90
6.4	Simulation Results . . . . .	94
6.4.1	Trajectory Tracking . . . . .	95
6.4.2	Trajectory Tracking with Disturbance Present . . . . .	97
7.	Cooperative Control with Artificial Potential Functions . . . . .	103
7.1	Artificial Potential Function for ‘Conflict-Free’ Trajectory Synthesis .	103
7.1.1	Swarm Aggregation APF Design . . . . .	105
7.1.2	Minimum Control Effort Navigation Function for Social Foraging	106
7.1.3	APF for Collision Avoidance . . . . .	108
7.2	Control Design . . . . .	109
7.3	Inter-Vehicle Communication . . . . .	110
7.4	Simulation Results . . . . .	111
8.	Real-Time Experiment Setup and Results . . . . .	117
8.1	Mobile Robot Platform . . . . .	118
8.1.1	Hardware Components . . . . .	118
8.1.2	Cooperative Platform Setup . . . . .	121
8.1.3	Software . . . . .	122
8.2	Real-Time Experiment Results . . . . .	123
8.2.1	Individual Mobile Robot Results . . . . .	125
8.2.2	Multiple Robot Cooperative Control Results . . . . .	135
9.	Summary, Conclusions and Future Work . . . . .	148
9.1	Summary and Conclusions . . . . .	148
9.2	Future Work . . . . .	150
Appendix		
A.	Preliminary Material . . . . .	152
REFERENCES . . . . .		169

BIOGRAPHICAL STATEMENT . . . . . 179

## LIST OF ILLUSTRATIONS

Figure	Page
1.1 ASL Gremlin Mobile Robot. . . . .	2
1.2 Local minimum example with traditional APF. . . . .	5
1.3 Illustration of mpc control horizon. . . . .	9
2.1 Block diagram depicting general GNC framework. . . . .	21
2.2 Flow of information for cooperative control framework. . . . .	23
2.3 Kinematics of the wheeled mobile robot. . . . .	24
3.1 Control effort based navigation function algorithm. . . . .	29
3.2 Bitmap representation of working environment. . . . .	30
3.3 Wavefront expansion cost propagation. . . . .	32
3.4 Grid set up in two dimensional space for a mobile robot model. . . . .	38
3.5 Minimum control energy path plan and potential field. . . . .	42
3.6 Inverse dynamics approach, path plan and potential field. . . . .	43
3.7 Minimum control energy path plan and potential field. . . . .	44
3.8 Inverse dynamics approach, path plan and potential field. . . . .	44
3.9 Effects of tuning $u_{max}$ with MCE approach, first example. . . . .	46
3.10 Effects of tuning $u_{max}$ with MCE approach, second example. . . . .	46
3.11 Effects of tuning $u_{max}$ with ID approach, first example. . . . .	47
3.12 Effects of tuning $u_{max}$ with ID approach, second example. . . . .	47
4.1 Flowchart of modified RRT* algorithm. . . . .	49
4.2 Steer procedure within RRT* algorithm. . . . .	51
4.3 Main algorithm flowchart for the modified/original D* algorithm. . . . .	54



4.4	Process-State procedure of D* algorithm. . . . .	56
4.5	Exploring Tree using modified-RRT* with 500 samples . . . . .	57
4.6	Exploring Tree using modified-RRT* with 1000 samples . . . . .	58
4.7	Exploring tree with generated path plan overlaid using 500 samples. .	59
4.8	Generated path plan using modified-RRT* with 500 samples. . . . .	59
4.9	Exploring tree with generated path plan overlaid using 1000 samples.	60
4.10	Generated path plan using modified-RRT* with 1000 samples. . . . .	60
4.11	Modified-D* path plan with no change in terrain detected. . . . .	61
4.12	Modified-D* path plans with a change in terrain traversability detected.	62
4.13	Comparative example with the different planning algorithms. . . . .	63
5.1	Example 1 constrained trajectory tracking with SDC based NMPC guid- ance commands and an MCE path plan. . . . .	78
5.2	Example 2 constrained trajectory tracking with SDC based NMPC guid- ance commands and an MCE path plan with obstacles. . . . .	78
5.3	Example 3 constrained trajectory tracking with SDC based NMPC guid- ance commands and an ID path plan. . . . .	79
5.4	Example 4 constrained trajectory tracking with SDC based NMPC guid- ance commands and an ID path plan with obstacles. . . . .	79
6.1	Example 1, position tracking with backstepping guidance commands and MCE path plan. . . . .	96
6.2	Example 2, position tracking with backstepping guidance commands and MCE path plan. . . . .	96
6.3	Example 3, position tracking with backstepping guidance commands and ID path plan. . . . .	97
6.4	Example 4, position tracking with backstepping guidance commands and ID path plan. . . . .	97

6.5	Scenario setup and trajectory tracking with disturbance present. . . .	99
6.6	Position error plots with $d = -2m/s^2$ . . . . .	99
6.7	Velocity error plot with $d = -2m/s^2$ . . . . .	100
6.8	Plot of rover velocity with $d = -2m/s^2$ . . . . .	101
6.9	Heading angle error plot with $d = -2m/s^2$ . . . . .	102
7.1	Communication protocol. . . . .	111
7.2	Example 1 with swarm aggregation policy, setting $\delta = 1$ in eq. (7.3). .	113
7.3	Example 1 with social foraging policy, setting $\delta = 0$ in eq. (7.3). . . .	113
7.4	Example 2 environment and navigation function potential field. . . . .	115
7.5	Example 2 with social foraging policy, setting $\delta = 0$ in eq. (7.3). . . .	115
7.6	Example 3 environment and navigation function potential field. . . . .	116
7.7	Example 3 with social foraging policy, setting $\delta = 0$ in eq. (7.3). . . .	116
8.1	ASL-Gremlin Mobile Robot Testing Platform. . . . .	117
8.2	On-board hardware components for the ‘ASL-Gremlin’ mobile robot platform. . . . .	119
8.3	Flow of information between hardware components for mobile robot platform. . . . .	120
8.4	Flow of information within the GNC framework for the mobile robot platform. . . . .	121
8.5	Flow of information for cooperative control experiments. . . . .	122
8.6	Waypoint setup for individual experiment 1. . . . .	126
8.7	Robot’s waypoint traversal for individual experiment 1. . . . .	127
8.8	Robot’s heading angle for individual experiment 1. . . . .	128
8.9	Robot’s velocity profile for individual experiment 1. . . . .	129
8.10	Robot’s wheel speed commands for individual experiment 1. . . . .	130
8.11	Waypoint setup for individual experiment 2. . . . .	131

8.12	Robot's waypoint traversal for individual experiment 2. . . . .	132
8.13	Robot's heading angle for individual experiment 2. . . . .	133
8.14	Robot's velocity profile for individual experiment 2. . . . .	134
8.15	Robot's wheel speed commands for individual experiment 2. . . . .	135
8.16	Setup for cooperative experiment 1. . . . .	136
8.17	Robot positions for cooperative experiment 1. . . . .	137
8.18	Wheel speed commands for cooperative experiment 1. . . . .	138
8.19	Robot's relative distance for cooperative experiment 1. . . . .	139
8.20	Setup for cooperative experiment 2. . . . .	140
8.21	Robot positions for cooperative experiment 2. . . . .	141
8.22	Wheel speed commands for cooperative experiment 2. . . . .	142
8.23	Robot's relative distance for cooperative experiment 2. . . . .	143
8.24	Setup for cooperative experiment 3. . . . .	144
8.25	Robot positions for cooperative experiment 3. . . . .	145
8.26	Wheel speed commands for cooperative experiment 3. . . . .	146
8.27	Robot's relative distance for cooperative experiment 3. . . . .	147
A.1	Example result using the traditional wavefront expansion algorithm. .	162
A.2	Traditional wavefront expansion algorithm. . . . .	163

## LIST OF TABLES

Table	Page
4.1 Main parameters used with respective algorithms. . . . .	63
4.2 Computation time to generate path plan. . . . .	64

## Executive Summary

The design of robust guidance and control algorithms for increasing autonomy with wheeled mobile robots and other vehicles have received a great deal of attention across different research communities. There have been many different approaches presented in the literature discussing either guidance or control individually or both aspects concurrently. The main motivation for developing guidance methods, in the form of path planning, is that this area has been identified as a key component for autonomous operations given the kinds of uncertain environments the different types of vehicles may encounter. And the addition of stable control designs to safely follow the resulting path plans intuitively follows. The goal of this research is to present a framework consisting of control effort based path planning algorithms along with two nonlinear guidance methods for a wheeled mobile robot.

The first problem discussed in this dissertation is the development of a reliable path planning algorithm to safely guide a mobile robot through a constrained environment. The environment will be considered constrained by the presence of static obstacles obstructing the paths to an objective destination. The path planning algorithm will be designed based on a special class of artificial potential functions called navigation functions. Specifically, a grid-based numerical navigation function will be implemented as one of the main contributions of this research. This dissertation will detail a methodology for numerically constructing a navigation function using a metric defined by the control effort of the system instead of a traditional distance based metric. To accomplish this, the formation of the navigation function relies on setting a desired reachable state for the robot to achieve. This enables the algorithm

to not just consider where the goal is located but also how the robot can best form its approach. Additionally, two widely used optimal path planning algorithms are modified to include a control effort-based metric to guide the vehicle through uncertain environments.

The next problem is to derive stable nonlinear guidance laws which will enable the vehicle to follow the path plan and arrive at its objective. Two different stable nonlinear guidance designs will be introduced in this work. The first guidance design will make use of a nonlinear model predictive control (NMPC) methodology. In this research, the NMPC formulation will result from a state dependent coefficient (SDC) factorization of the nonlinear kinematic model for a wheeled mobile robot. The SDC form has been researched in connection with finding linear optimal control solutions involving nonlinear models. The NMPC methodology in this dissertation will make use of the SDC form of the robot's kinematics to design its guidance inputs. The use of the NMPC method here will allow constraints to be enforced on the robot's inputs and state as it tracks a trajectory to the goal.

The second guidance design will be based on a backstepping-like approach. The stable backstepping guidance law in this dissertation will allow the robot to track a trajectory along the path plan and ensure the task will be performed with bounded tracking errors.

Another important consideration in this dissertation will cover cooperative control involving multiple wheeled mobile robots. It is reasonable to assume that multiple vehicles may be present in a given operation and a procedure to accomplish tasks cooperatively is necessary. In this dissertation, the cooperative control tasks are handled with artificial potential functions. The main tasks involved with this technique becomes aggregation and social foraging. In other words, the robots must be able to coalesce and travel to designated areas of interest while avoiding obstructions and col-

lisions among the vehicles within the group. This technique will utilize the numerical navigation function algorithm to set the areas of interest and obstacle avoidance with additional cooperative potentials introduced to ensure there is no collisions within the group.

In this dissertation, all the above concepts are applied to mobile robot vehicles in simulation. Additionally, some of the concepts, such as the nonlinear guidance laws and cooperative control framework are validated experimentally using mobile robot testing platforms in the Aerospace Systems Laboratory at The University of Texas at Arlington.

## CHAPTER 1

### Introduction and Motivation

The focus of this research is to present a framework for path planning and guidance for autonomous mobile robots. The framework is designed as a step towards increased autonomy for wheeled mobile robots. The quantifiable measures of autonomy recognized in this dissertation are the vehicle's ability to observe, orient, make decisions, and act [1]. Of those autonomy measures, this research will focus primarily on expanding a mobile robot's ability to make decisions and to control its actions within a given environment.

There is an increasing need for reliable path planning and guidance algorithms for wheeled mobile robots such as planetary exploration rovers [2–4] as well as autonomous cars and other car-like vehicles [5, 6]. The need for guidance, in the form of path planning, for mobile robots arises from the kinds of environments they may encounter. The environment may possess obstacles of varying sizes as well as different kinds of terrain, such loose rocks, sand, or embedded pointy rocks, etc. The vehicle must be able to account for these details and suggest safe paths for it to follow. To help the vehicles make decisions and find safe paths in the environment, a numerical navigation function algorithm is presented in this research.

Another important issue with these mobile robots is how to follow the safe paths which have been generated. The vehicle must have the capability to arrive at its destination safely in the presence of uncertainty in the environment or its own physical limitations. The ability to act within the vehicle's confines is of particular interest in this research and is addressed in the form of an improved nonlinear model



predictive control derivation. Additionally, the ability to have guidance in how to act when multiple vehicles are acting in the same environment is of interest and is addressed through a cooperative control policy.

The components of the framework are verified in simulation and extended to real-time testing platforms, such as the vehicle in figure 1.1, to provide experimental validation.



Figure 1.1: ASL Gremlin Mobile Robot.

## 1.1 Background

### 1.1.1 Path Planning with Control Effort and Navigation Functions

The first component of the framework presented in this dissertation is a path planning algorithm to help a mobile robot make decisions in finding safe reference paths through the environment. In general, path planning is the process of finding a safe path between two points for a mobile vehicle to travel. There are a variety of different methods for path planning found in textbooks and papers alike. Many of the path planning methods currently being researched consider collision and obstacle avoidance as a primary objective. And while obstacle avoidance is considered with this research, its novel contribution is an investigation into how to include control information into the path plan through a grid-based numerical potential field construction.

Other path planning techniques that make use of the control effort and the kinematic model of the system are discussed in terms of rapid exploring randomized trees (rrt and rrt\*) in [7, 8], kinodynamic rrt in [9], and probabilistic roadmap approach in [10]. These methods use a randomized approach with state information of the system to determine the path plans, and the paths are designed with information of the environment and the initial location. While these methods can be computationally intensive, depending on the model, they also generate plans without needing to know where the objective is located.

The path planning methods discussed in references [7–10] can incorporate knowledge of a system’s control effort to generate a path plan in a randomized sampling based manner. In contrast, the methodology discussed with this work makes use of a special class of artificial potential functions called navigation functions. This con-

struction method will make use of the control effort of the system and enable a path plan to be formed from almost any point in the environment.

In general, potential fields for path planning purposes are generated such that their structure, potential levels and shapes, can intuitively reflect the virtual makeup of the workspace. And a vehicle within a potential field is treated as a particle under the influence of gravity. The paths with these methods are then generated through a method that is similar to a steepest descent optimization problem [11–14].

Traditional potential field methods were first designed as an online collision avoidance scheme in which the attractive potential is represented by a parabolic well at the goal and the repulsive potentials are defined in the constrained space which will tend towards infinity for the points in their vicinity [11]. Then, a path, as well as the control input, can be found by following the negative gradient of this function (similar to steepest descent).

Although traditional APF methods can effectively create collision-free paths, there is one possible drawback. Since the method by which the paths are generated is similar to a steepest descent problem, the paths derived for the system could reach equilibrium at a configuration that is not its goal. This is known as the local minimum problem [12]. An example of the local minimum problem arises when considering the dynamical system at a configuration where the attractive potential from the goal is equal to the repulsive potential of the constrained space.

An illustrative example of a potential field with a local minimum and a U-shaped obstacle using the traditional APF method is shown in figure 1.2. It is evident from figure 1.2 that not every path plan following the negative gradient of the potential field will arrive at the goal. The local minima problem with potential field methods is what gave rise to the development of navigation functions, which possess only a global minimum in its potential field that is located at the goal [12–14].

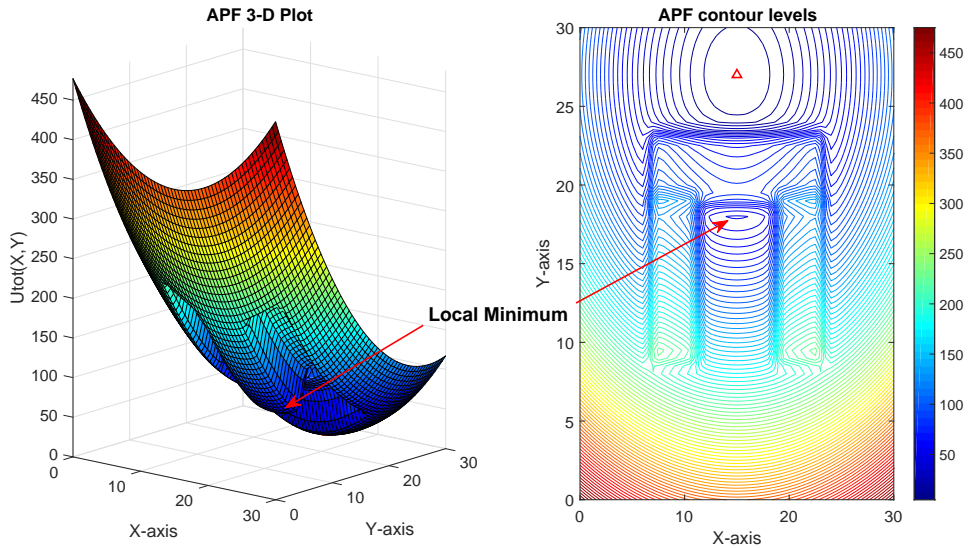


Figure 1.2: Local minimum example with traditional APF.

There are two main approaches to create a navigation function. The first method is to define an analytic function which possesses an attractive component associated with the objective and repulsive components attached to the obstacles. These approaches are motivated by the research introduced in [11] for online collision avoidance using artificial potential functions. Examples of the analytical approach with a NF are presented in [15–17] whose results originate from the NF definition introduced in [14]. With this approach, the navigation function is defined as a composition of several functions each designed to satisfy specific properties established in [14]. The specific properties established to define a navigation function analytically are summarized as:

1. The function is continuous and differentiable (smooth function) on the path connected set to the goal
2. The function is uniformly maximal on the boundaries
3. The function must have a unique and global minimum at the goal

4. The function must be a Morse function

And while effective, this method requires proper tuning of several parameters within the function before the local minima can be removed and for the NF to be properly defined.

The second method for constructing a viable navigation function is to construct it numerically in a discrete grid. This can be done either by using numerical solutions to partial differential equations (PDE) as seen in references [18–20] or by assigning potentials to a discrete workspace based on their distance from the objective [12, 13, 21–24]. The numerical navigation functions described in references [19] and [20] use harmonic functions to represent the workspace with the boundary conditions enforced to ensure that a viable path is found. The navigation function in reference [18] is constructed similarly in that it uses the finite difference method to solve a PDE representation of the Hamilton-Jacobi-Bellman (HJB) equation over the workspace. Additionally, the method described in reference [18] can be made to rely upon the system’s dynamics.

Numerical potential functions, such as those described in references [12, 13, 21–24], have an advantage of being constructed in such a way that the goal location is given the minimum value and the rest of the potential values are propagated throughout the rest of the free operating space. These approaches are done through wavefront expansion with counting and logic involved.

Navigation function path planners are effective in generating safe paths to the goal, however the methods that have been introduced in the past are primarily formed only with knowledge of the distance to the goal and the connectivity of the free regions in the workspace. Conversely, the novel algorithm introduced in this dissertation will generate the navigation function based on the control effort of a given model to go from one grid point to another in a given environment. The novel algorithm will leverage

the construction method for the navigation functions described in references [12] and [13] but will use the system's control effort to determine the contour levels.

The path plan algorithm introduced in this dissertation can generate reference paths from anywhere in the free environment. However, the algorithm will require knowledge of both the objective location as well as a final desired state and would need to be regenerated if new information is acquired. The result, however, is a path plan that will guide the robot safely to its objective while also considering the control effort to get there as well as how to form its approach to a goal state. Additionally, the path plan algorithm enables the inclusion of the kinematics of the mobile robot in its formation without increasing the dimensions of the configuration space of the system. The planner in this research considers the model of the mobile robot to consist of four state variables, but it generates the path plan within a two-dimensional environment making it computationally cheap.

While the navigation function path planner is able to generate path plan from anywhere in the free space, the fact that it will need to be regenerated whenever new knowledge is obtained can make it inefficient in the presence of uncertainty. To overcome this issue, extensions of the path planning algorithm are presented using modified versions of RRT\* and D\* with a minimum control effort-based metric to determine the cost to move between configurations.

The RRT\* algorithm, as presented in reference [8] is designed to form its tree from the starting configuration of the robot and can use the tree to find a minimum cost traversal to anywhere in the free space. This ability gives the planner the ability to determine a path with an uncertain goal and eliminates the dependence on grid resolution. The D\* algorithm is another grid-based path planner and is discussed in references [25, 26]. This algorithm is designed as a dynamic A\* algorithm, discussed in references [27] and [28], where it has the ability to dynamically re-plan a path

in the presence of an uncertain environment. These algorithms are chosen due to their ability to account for uncertain objectives, terrain types and can be used to find optimal traverses.

### 1.1.2 Nonlinear Guidance Law

The next component of the framework applied in this research is a nonlinear guidance technique. This part of the research is intended to provide a mobile robot with an increased ability to act in the given scenarios. The guidance laws are designed to provide commands to ensure that the vehicle can track the reference paths it is given.

#### 1.1.2.1 Nonlinear Model Predictive Control-Based Guidance

The main nonlinear guidance technique that is applied within this framework is based on nonlinear model predictive control (NMPC). The NMPC algorithm is chosen due to its ability to incorporate constraints on the inputs and outputs of the system being studied. Additionally, the NMPC algorithm is solved by making use of the State Dependent Riccati Equation (SDRE) and its associated state-dependent coefficient formulation.

The use of model predictive control, or receding horizon control (RHC), is a widely researched topic in controls engineering for a variety of applications. Model predictive control is a process control method where the current inputs to the system are determined by forecasting the behavior of the system model over a finite horizon. The control is designed to minimize a cost function over the finite horizon and can be used for regulation or tracking of a reference trajectory [29]. For MPC to be implemented, a continuous system is discretized given a sampling time based on the process being studied. Then, a prediction horizon is taken as the number of time



steps into the future being considered in the forecast. This allows for a control input to be determined at each time step over the horizon. Although the model behavior is considered over the time horizon, only the first input is applied to the mobile robot using the technique in this research, as illustrated in figure 1.3.

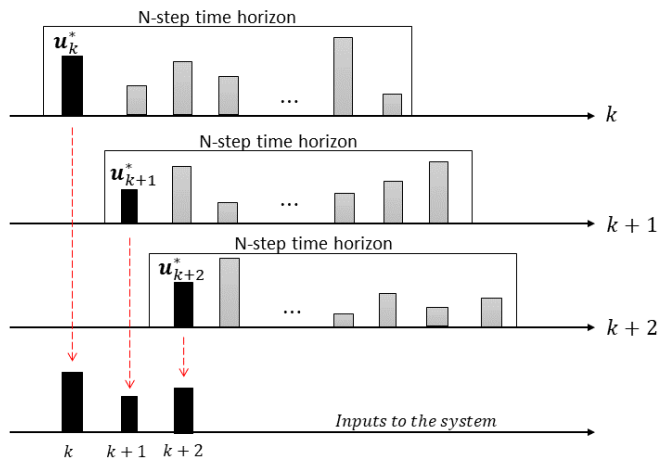


Figure 1.3: Illustration of mpc control horizon.

This approach is used to generate high-level guidance commands for the mobile robot to track a given trajectory. The derived guidance commands will be unique since it will make use of the state-dependent coefficient (SDC) form of the nonlinear kinematic equations and the system's inputs will be found by quadratic programming. The SDC formulation is used because it can preserve the nonlinear nature of the system being studied.

The SDC form for nonlinear systems as it pertains to controller, observer or filter design is discussed in references [30–33]. The focus of using the SDC form is to transform a nonlinear system into a pseudo-linear form and then implement optimal control or estimation techniques through solving the SDRE. As it relates to control design, the SDC form allows for synthesis of nonlinear feedback controllers that are



similar to the LQR structure. In references [30] and [31], the problems are posed to use the SDRE to solve problems such as the infinite horizon quadratic regulator for some example nonlinear dynamics. The authors use direct parameterization to place the dynamics into the SDC form, which they note is not unique for a given system.

The SDC form and control designs using the SDRE discussed in references [30–33] center on solutions over an infinite time horizon. In contrast, the results described in references [34] and [35] look at solutions for control problems with a finite horizon. Both papers include a change of variables to solve for the control over a finite horizon as well as fixed terminal constraints, so that they can achieve their respective objectives [34, 35].

The results presented in references [30–35] highlight the use of the SDC form and the nonlinear feedback control laws that can be found through solving the SDRE. These results, however, do not consider constraints on the system’s inputs or outputs (other than some terminal constraints, shown in references [34] and [35]). The contribution of the work presented in this dissertation will make use of an SDC formulation and nonlinear model predictive control, which will enforce input and output constraints while performing reference trajectory tracking.

Traditionally, nonlinear model predictive control involves linearizing the system’s dynamics about a nominal trajectory [29, 36, 37]. References [29] and [36] give an overview of the traditional formulation for NMPC. And reference [38] discusses important results for MPC in general with considerations towards both stability and optimality of the results.

An alternate approach for NMPC employs a Control Lyapunov Function (CLF) to help with achieving stability and applying an approximation to the terminal cost to the tail of the infinite horizon problem [39–44]. The CLF approach for receding horizon control along with an SDC factorized system is shown in references [41] and [42],

without considering constraints. Reference [39] presents the CLF technique in detail with receding horizon control and focuses on time varying and input constrained systems. Also, the author in [39] propose that finding an appropriate CLF is equivalent to finding a continuous stabilizing control law for the system.

Then, the author in reference [40] further the discussion from [39] covering the CLF approach with RHC. The developments found in [40] pertain to the stabilization of unconstrained nonlinear systems and use the CLF as a terminal cost function. Furthermore, the authors conclude that there is no need for constraints on the system or the CLF to achieve stability, with the proposed methodology.

The contribution of the work covered with this research is motivated by the developments in reference [45]. However, the research presented in this dissertation distinguishes itself in that the NMPC guidance design is applied to a wheeled mobile robot and is verified both in simulations and through experiments by applying the design to a real-time mobile robot testing platform.

#### 1.1.2.2 Nonlinear Control of Rover Vehicles

Another nonlinear guidance method that will be derived and used in the framework is a backstepping-like guidance law. This design is provably stable and guarantees bounded trajectory tracking errors. This will ensure that the mobile robot can safely and accurately follow a path plan generated by the navigation function. Also, this method is widely-researched and can be used for comparison with the NMPC design.

There have been implementations of different nonlinear control designs in a wide variety of mobile robot applications [46–55]. Some of the applications of nonlinear control with wheeled robots have only been verified through simulation, as in references [46–52], while others have been validated by hardware experiments such as

references [53–55]. Reference [46] provides a control design based on a virtual structure approach to follow a simple user defined trajectory. And in reference [47], an adaptive control design is implemented in simulation to control a mobile robot where it is also proven to be robust to input saturation and disturbances.

The experimental results presented in reference [53] validates a learning based nonlinear model predictive control design that is constructed so that it must learn a path through repetition to improve its model parameters. The experimental results discussed in reference [54] are for an adaptive dynamic control design that applies its control inputs to the dynamics in order to govern the kinematics of a wheeled robot. The applications of this work are for an autonomous load carrying wheeled mobile robot in an industrial setting [54]. Also, in reference [55] the results show a velocity scheduling based controller that utilizes dynamic feedback linearization of a mobile robot with only two out of four wheels being actuated.

Backstepping control designs for wheeled mobile robots are discussed in references [48–52]. The control laws in references [48] and [49] apply backstepping to control the kinematic model directly from the dynamics of the system and torques on the vehicle are applied as the control inputs. In contrast, the backstepping control design in references [50, 51] account for commanding the robot’s heading angle turn rate and its forward acceleration. Also, a backstepping-like control design is proposed in reference [52], which is derived so that the control inputs are given in terms of the wheel speeds of a robot in order to facilitate its implementation on an experimental testbed.

The backstepping-like guidance law discussed in this framework will be motivated by the developments presented in references [50, 51]. The results from [50, 51] are expanded upon in this dissertation with updated stability considerations and presentation of real-time implementation results using the derived guidance laws.

### 1.1.3 Cooperative Control with Artificial Potential Functions

It is possible that more than one mobile robot may be present in a given scenario and they will need to collaborate with one another. Therefore, it is practical to consider a guidance methodology for scenarios involving multiple vehicles needing to cooperate. The chosen approach to investigate the interactions between the vehicles will be evaluated based on an artificial potential function (APF) approach.

With this approach, the vehicles will be modeled based on a wheeled robot's kinematic equations and the commanded velocity and heading angle guidance commands will be determined from a composite potential function. The composite potential function will consist of a numerical navigation function, an analytical potential function governing the interaction forces between the vehicles and an additional repulsive potential function. The numerical navigation function will represent the given environment, given obstacle locations and an objective gathering location. The analytical APF component will have attractive and repulsive characteristics to dictate the behavior of the individual agents within the group. Finally, the additional repulsive term is used as an extra layer to ensure the vehicles do not collide.

In general, the goal of this cooperative control policy is to derive guidance commands that govern how individuals within a group move so that they stay together and/or avoid collisions. The set of rules called Reynold's Rules were defined in order to capture the collective motion of large groups based on observations in nature. Formally, Reynolds' rules for collective motion are [56]:

1. Collision avoidance
2. Velocity matching - matching speed and motion direction
3. Flock centering

Traditional approaches for cooperative control frameworks involve portraying the communication topology of the group as a graph of nodes and edges, called

the graph theoretic framework [56]. In this arrangement, the flow of information is structured within a communication graph. The resulting graph helps to illustrate which agents communicate with each other and which information is available to the group. In the graph theoretic framework, the feedback control laws are derived based on graph theory, involving the formation of an adjacency matrix, based on the information flow, and an in-degree matrix, based on the number of agents in communication with a particular node. Then, the feedback control law is found by evaluating the graph Laplacian, which is the difference between the in-degree matrix and the adjacency matrix, multiplied by the state of the group considering integrator dynamics [56].

An alternate approach to the graph theoretic framework for cooperative control involves defining potential energy functions, also called artificial potential functions (APF). The APF method was initially introduced as an online collision avoidance algorithm, as detailed in reference [11]. The design is such that it possesses an attractive component as well as a repulsive component. The attractive component is designed to draw the system to a desired state and the repulsive component is designed to avoid potential hazards or undesired states for the system [11]. For cooperative control scenarios, however, the APF is used to influence the behavior of the group in a decentralized approach. Thus, the individual vehicles act in accordance with their respective locations relative to the other vehicles present.

There are some defined objectives in the literature for cooperative control using APFs, which are similar to Reynold's Rules. The main behavioral objectives using the potential function approach for a group of vehicles are aggregation, social foraging and formation control [57, 58]. For aggregation, the objective is to bring the group together while avoiding inter-vehicle collisions. And in social foraging, the behavior resembles the search of an environment for areas of interest while avoiding areas

of potential danger. And formation control is to have the vehicles achieve a final geometrical structure while moving together [58].

One approach using APF cooperative control is demonstrated in reference [59]. In [59], double integrator dynamics are assumed and the potential function defines the interaction between neighboring vehicles with a virtual leader providing a moving reference trajectory to track. The virtual leader is introduced to provide direction and possibly manipulate the group's geometry. Another example using a point mass dynamics is shown in reference [60] where guidelines for constructing a potential energy function is discussed. In [60], the potential function is defined as a composition of different functions each designed to attain a certain performance. Both references [59] and [60] discuss flocking behavior of multiple agents moving together to different objectives in a given environment.

While references [59] and [60] consider vehicles with point mass dynamics, the author in reference [61] provides a cooperative APF framework for unicycle mobile robots. In [61], each robot is assumed to have a safety area and communication area which are designed to dictate the communication protocol of the group. The APF introduced in [61] is a smooth  $p$ -differentiable bump function designed such that the vehicles in the group can track a reference trajectory while avoiding collisions with all other robots.

The research presented in reference [57] gives an alternate APF formulation along with guidance for defining such a potential function for cooperative control. The authors in [57] define a general class of odd functions that have attractive and repulsive components that can be solved to find the equilibrium distances between the agents with tuning of several design parameters. Several objectives can be reached due to the properties of the APF defined by the authors in [57] such as stable aggregation, and formation control for a group. The work in reference [58] extends the results in [57] to

include social foraging considerations and applies the results to non-holonomic agents where the APF provides reference values to be tracked with a sliding mode controller.

Other extensions inspired by the research in reference [57] can be found in references [62] and [63]. Reference [62] presents guidelines for overcoming some potential pitfalls with APF frameworks. The problem areas addressed in [62] consider a non-reachable goal (local minimum), obstacle collision (when the attractive potential overwhelms the repulsive component), obstacle collisions in swarms and inter-agent collisions. The guidelines in [62] combines additive and multiplicative configurations of the APF to address the problems with a point mass system. Then, reference [63] makes use of the APF guidelines found in [57] for use on a system of quadcopters with multi-loop control and some considerations towards obstacle avoidance.

The research that will be discussed in this dissertation is influenced by the APF design found in reference [57]. However, this work will consider wheeled mobile robot's kinematics. An additional contribution from this aspect of the dissertation is the insertion of the numerical navigation function for social foraging tasks.

## 1.2 Objectives and Contributions

### 1.2.1 List of Contributions

- I. Developed a path planning algorithm that considers a system's kinematics and control effort.
- II. Designed path planning algorithm that plans to a reachable state.
- III. Derived a stable backstepping guidance law that ensures bounded tracking errors.
- IV. Derived a stable nonlinear model predictive control-based guidance law that can enforce constraints on the inputs and outputs of the system.

- V. Implemented a combination of the path planning algorithm and either of the nonlinear controllers to construct a guidance and control framework.
- VI. Combined the path planning algorithm's potential field with cooperative potential functions for a group of rover vehicles for aggregation and social foraging tasks.
- VII. Applied guidance techniques, individual and cooperative, to real-time mobile robot platforms.

### 1.2.2 List of Published Works

#### Journal Publications:

(a) Objectives I.,II.,III.,V.:

P. Quillen, K. Subbarao and J. Muñoz, "Path Planning to a Reachable State Using Minimum Control Effort Based Navigation Functions," in *Journal of Astronautical Sciences*. (*In review*)

(b) Objectives I.,II.,IV.,V.:

P. Quillen, K. Subbarao and J. Muñoz, "Minimum Control Effort Based Path Planning and Nonlinear Guidance for Autonomous Mobile Robots," in *The International Journal of Advanced Robotic Systems*. (*In review*)

(c) Objectives I.,II.,VI.,VII.:

P. Quillen, K. Subbarao and J. Muñoz, "Cooperative Control with Minimum Control Effort Based Navigation Functions," in *Journal of Intelligent Robotics*. (*Pending*)

(d) Objectives III.,IV.,VII.:

P. Quillen, K. Subbarao, "Real-Time Nonlinear Model Predictive Control for Wheeled Mobile Robots," in *The Journal of Advanced Robotics Systems*. (*Pending*)



Conference Publications:

(a) Objectives III.:

P. Quillen, K. Subbarao and J. Muñoz, “Guidance and Control of a Mobile Robot via Numerical Navigation Functions and Backstepping for Planetary Exploration Missions,” in AIAA Space 2016, AIAA Space Forum, (AIAA 2016-5237). (Reference [50])

(b) Objectives I.,II.,,III.,IV.:

P. Quillen, J. Muñoz and K. Subbarao, “Path Planning to a Reachable State Using Inverse Dynamics and Minimum Control Effort Based Navigation Functions,” in AAS/AIAA Space Flight Mechanics Meeting, no. AAS 17-849, 2017. (Reference [51])

(c) Objective III.:

A. Godbole, V. Murali, P. Quillen and K. Subbarao, “Optimal Trajectory Design and Control of a Planetary Exploration Rover,” in Advances in the Astronautical Sciences Spaceflight Mechanics, vol. 160, 2017. (Reference [52])

### 1.3 Dissertation Outline

This dissertation is organized as follows: A description of the framework is given in chapter 2 along with a summary of its components and the mobile robot kinematic model used for the results. Chapter 3 details the numerical navigation function algorithm using a modified wavefront expansion for path planning with control effort. Chapter 4 presents extensions of the path planning algorithm using modified versions of RRT\* and D\* to find safe paths. The nonlinear model predictive control-based guidance law is presented in chapter 5. A different nonlinear guidance law based on a backstepping-like approach is covered in chapter 6. The cooperative guidance policy for multiple mobile robots is presented in chapter 7. The simulation results using the

components of the framework are given respectively in chapters 3 through 7. The experiment results using the mobile robot testing platforms with the guidance techniques covered in this dissertation are compiled in chapter 8. Finally, the conclusions and future work with this research is given in chapter 9.

## CHAPTER 2

### Framework and Problem Description

This dissertation involves approaches which either involve a single mobile robot or a group of mobile robots in a given scenario. The individual tasks will be used to verify the nonlinear guidance techniques in tandem with the path planning algorithm. Then, the group tasks will be used to verify the cooperative control policy using a combination of potential functions with the potential field constructed by the navigation function to perform aggregation and social foraging tasks.

#### 2.1 Framework Setup

The proposed framework in this dissertation for individual mobile robots will consist of two main capabilities meant to enhance the vehicle's autonomy. Figure 2.1 illustrates the main flow of information in a typical guidance, navigation, and control framework. The main contributions of this research will be in the areas of a navigation function path planner and two different nonlinear guidance designs, as highlighted in figure 2.1. For the results given, it is assumed that information concerning the environment such as obstacle positions and objectives are known to the vehicle and that the maneuvers occur within a flat environment.

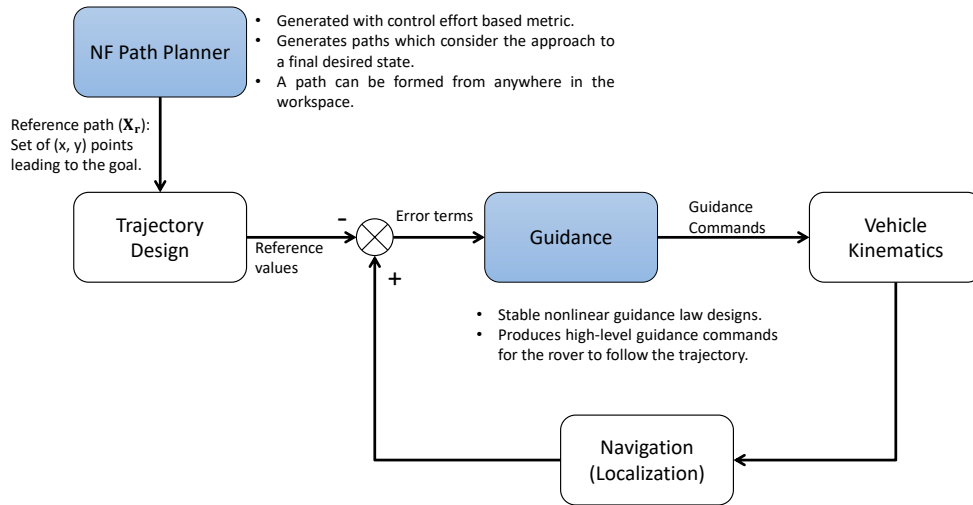


Figure 2.1: Block diagram depicting general GNC framework.

The first capability of the framework, evident with the path planning algorithm, has the ability to guide the vehicle safely through a hazardous, obstacle laden environment to a reachable state. This aspect of the work will be addressed through a special class of artificial potential functions called navigation functions. The approach shown in this dissertation will be discussed with the algorithm described in chapter 3. The objective with this component of the framework is to introduce the technique with the vehicle's kinematic model and to generate a path plan that directs the approach of the vehicle to a final desired reachable state.

Extensions of the path planning algorithm in chapter 3 are also provided in chapter 4. These extensions make use of the same minimum control effort approach but utilize different methods for planning the path. The two extensions studied in this dissertation use modified version of the RRT\* and D\* path planning algorithms.

The second capability is a guidance law which generates high-level commands to have the vehicle track a desired path through the environment. This aspect within the framework is handled through two different nonlinear guidance techniques. The first technique is based on a nonlinear model predictive control derivation and the second is a backstepping-based design. The NMPC technique applied in this framework is given in chapter 5 and is based on the developments found in [45]. For this methodology, the SDC form is used to preserve the nonlinear nature of the vehicle’s kinematic equations and the weight on the terminal state of the system is found by solving the discrete time state-dependent Riccati equation. The guidance commands are then found by quadratic programming with input and state constraints enforced on the system.

Then, the backstepping guidance law in this framework is presented in chapter 6 of this dissertation. This approach is chosen for comparison with the NMPC design as it is an extensively studied method used for controlling differential drive mobile robots. This guidance law is designed to provide provably stable, high-level guidance commands.

Simulation results are presented in chapters 5 and 6 and involve the combination of the NF path planner with the nonlinear guidance designs to close the loop of the framework. Additionally, experimental validation of the nonlinear guidance techniques is presented in chapter 8 of this work.

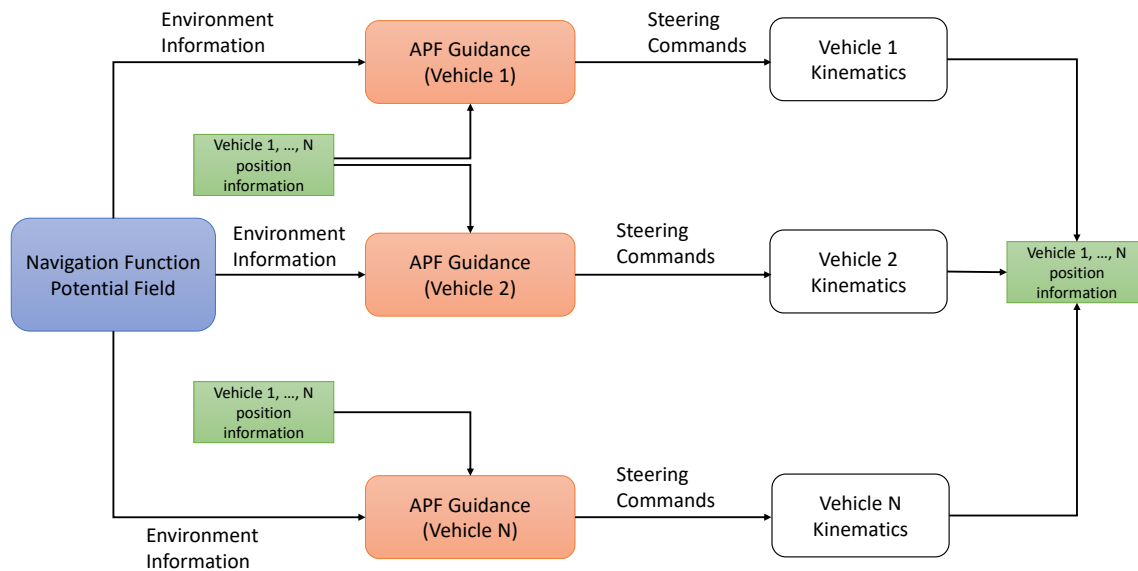


Figure 2.2: Flow of information for cooperative control framework.

The next aspect of this research involves cooperative control with multiple mobile robots. An illustration of the flow of information within the cooperative control framework is given in figure 2.2.

The potential field generated by the navigation function algorithm allows for a path to be formed from anywhere in the given environment. It also provides information such as areas to avoid (obstacles) and the objective location for the group to arrive at. The APF guidance block within the framework uses the environment information provided by the navigation function potential field and combines it with information about the relative positions of the other robots to generate steering commands to safely guide the vehicles to their objective. The APF guidance uses the relative position information along with different potential functions intended to generate conflict free trajectories through the environment for the group of robots. The

cooperative control policy and simulation results are presented in chapter 7 with experimental results given in chapter 8.

## 2.2 Mobile Robot Kinematic Model

The kinematic model of the wheeled mobile robot considered in this research is illustrated in figure 2.3. For the results discussed, an east-north-up (ENU) convention is used. The heading angle,  $\psi$ , is defined positively going in a counter clockwise direction about the z-axis (up) going from the inertial x-axis ( $x_I$ ) to the body x-axis ( $x_b$ ). The state variables of the robot is taken as the inertial position in  $x$  and  $y$  coordinates, its heading angle and its forward velocity, and it can be represented by the vector  $\mathbf{x} = [x, y, \psi, v]^T$ . Without loss of generality, the subscript I for the inertial  $x$  and  $y$  positions of the robot is dropped.

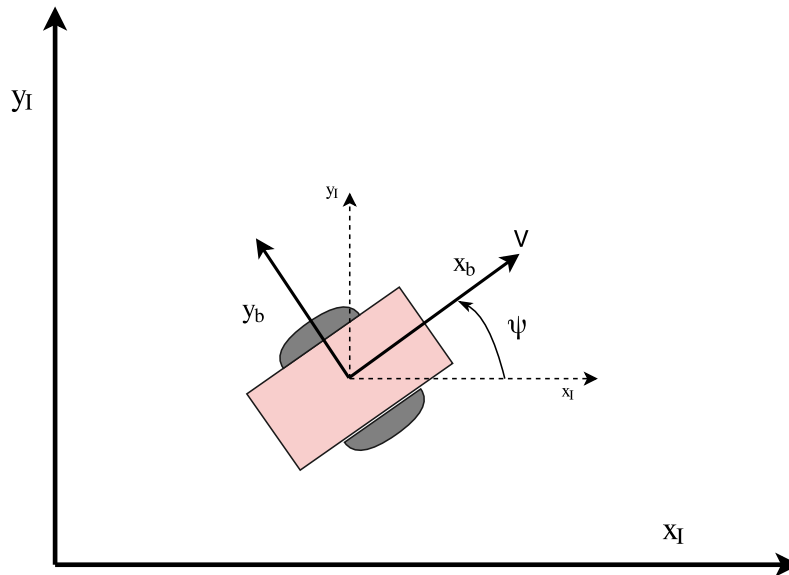


Figure 2.3: Kinematics of the wheeled mobile robot.

### 2.2.1 Nonlinear Kinematic Equations

There are two sets of nonlinear kinematics used here to represent a wheeled mobile robot. The first set of equations are given by

$$\begin{aligned}\dot{x} &= v \cos(\psi) \\ \dot{y} &= v \sin(\psi) \\ \dot{\psi} &= u_1 \\ \dot{v} &= u_2\end{aligned}\tag{2.1}$$

where  $u_1$  and  $u_2$  are the guidance inputs. These kinematics can be used as a representative model with inputs influencing the heading angle turn rate and the vehicle's forward acceleration. The set of equations defined in eq. (2.1) are used for the path planning algorithm and the nonlinear model predictive control derivation. Stability with the set of equations in eq. (2.1) with a backstepping approach is shown in chapter 6 to prove their feasibility in the navigation function algorithm.

The second representation of the mobile robot's kinematics are considered with the backstepping guidance derivation and are given by

$$\begin{aligned}\dot{x} &= v \cos(\psi) \\ \dot{y} &= v \sin(\psi) \\ \dot{\psi} &= \alpha_1(\psi^c - \psi) \\ \dot{v} &= \alpha_2(v^c - v)\end{aligned}\tag{2.2}$$

In eq. (2.2), the high level guidance commands are  $\psi^c$  and  $v^c$ , which denote the commanded heading angle and commanded velocity, and are used to track a reference trajectory. The equations for  $\dot{\psi}$  and  $\dot{v}$  are changed from eq. (2.1) to add an extra layer of control for implementing the nonlinear backstepping guidance law.



### 2.2.2 Kinematics in Linear State-Space Form

A linear version of the kinematics are needed for the path planning algorithm described in chapter 3. The kinematics given in eq. (2.1) need to be placed in a linear state-space form, i.e.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (2.3)$$

where  $\mathbf{x} \in \mathfrak{R}^n$ ,  $\mathbf{u} \in \mathfrak{R}^m$ ,  $\mathbf{A} \in \mathfrak{R}^{n \times n}$  and  $\mathbf{B} \in \mathfrak{R}^{n \times m}$  are the state and input vectors and state and input system matrices respectively. In order to achieve this, the deviation of the state from a nominal trajectory is considered. And in order to place the system of equations in a linear state-space form, the jacobian of eq. (2.1) needs to be evaluated about a nominal trajectory,  $\mathbf{x}_n \in \mathfrak{R}^n$ . Given a nonlinear autonomous system of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

the  $\mathbf{A}$  matrix is equal to

$$\mathbf{A} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}_n}$$

and the  $\mathbf{B}$  matrix is equal to

$$\mathbf{B} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{\mathbf{x}_n}$$

Hence, the linear system  $\mathbf{A}$  and  $\mathbf{B}$  matrices for the wheeled mobile robot are defined as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -v_n \sin(\psi_n) & \cos(\psi_n) \\ 0 & 0 & v_n \cos(\psi_n) & \sin(\psi_n) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.4)$$

and

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.5)$$

Thus, the linear kinematics for the mobile robot are of the form

$$\delta\dot{\mathbf{x}} = \mathbf{A}\delta\mathbf{x} + \mathbf{B}\delta\mathbf{u} \quad (2.6)$$

where the  $\mathbf{A}$  and  $\mathbf{B}$  matrices are evaluated through eqs. (2.4) and (2.5) respectively and  $\delta\mathbf{x} \in \mathfrak{R}^n$ , and  $\delta\mathbf{u} \in \mathfrak{R}^m$  are perturbations of the state and input vectors from the nominal trajectory, i.e.  $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_n$  and  $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}_n$ .

For the path planning algorithm, the nominal trajectory for the robot is considered as a straight-line, un-accelerating trajectory between two points in a grid over a constant sample time,  $\Delta t$ . The nominal values for  $\psi_n$  and  $v_n$  are evaluated as

$$\begin{aligned} \psi_n &= \tan^{-1} \left( \frac{\Delta y}{\Delta x} \right) \\ v_n &= \sqrt{\left( \frac{\Delta x}{\Delta t} \right)^2 + \left( \frac{\Delta y}{\Delta t} \right)^2} \end{aligned} \quad (2.7)$$

Within the path planning algorithm described in chapter 3, the values for  $\Delta x$  and  $\Delta y$  are evaluated as the cell difference of two neighboring points in a grid.

## CHAPTER 3

### Path Planning to a Reachable State using Numerical Navigation Functions

The main contribution of this research is the construction of the navigation function, which is motivated by the wavefront expansion described in [12] and [13]. However, the novel design discussed here distinguishes itself from the traditional methods since it will use a metric based on the control effort of the system to form the contour levels in the potential field as opposed to a distance based metric. The work presented in this dissertation makes use of the kinematic equations for a mobile robot and observing how it influences the potential contour levels with the algorithm. The end result is a path plan algorithm that is model-dependent.

The full algorithm is outlined in figure 3.1 with the contributions to the algorithm highlighted. These modifications were added to the original wavefront expansion algorithm to enable the inclusion of a metric defined by the control effort of the system and allow for some constraints to be considered.

Two methods are introduced for constructing the navigation function. The first method will make use of the solution to the minimum control effort problem given a fixed initial and final state for a linear system. This method is constructed to plan a path to an objective reachable state, expressed as  $\mathbf{x}_{goal}$ , and will form a minimum control effort path plan. Then, the second method will be to take an inverse dynamics approach to a reachable state. This approach is considered for the nonlinear mobile robot kinematic model. Both methods are constructed to reach an objective reachable state, expressed as  $\mathbf{x}_{goal}$ , and will form a minimum control effort path plan.

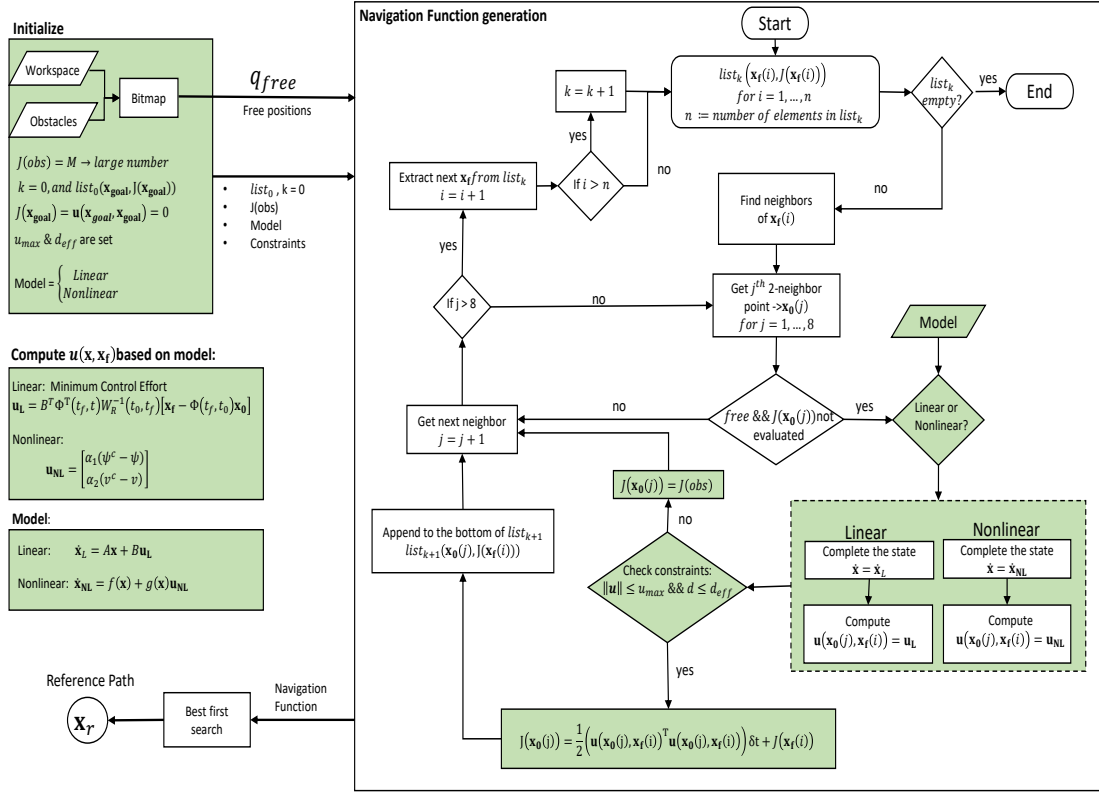


Figure 3.1: Control effort based navigation function algorithm.

In this chapter, the algorithm will first be described in general without specifying which method is used. Then, the specific details involved with each method will be discussed followed by some example simulation results for each case.

### 3.1 Navigation Function Generation

#### 3.1.1 Initialization Block

The first step of the algorithm takes place with the initialization block, as shown in figure 3.1. The initialization block of this numerically constructed potential field begins by discretizing the workspace into an evenly spaced grid. This grid can be scaled to fit over any two-dimensional working environment. The discrete grid and the obstacle positions are then used to form a bitmap representation of the environment, as illustrated in figure 3.2. This allows the free points to be identified and extracted. Note that the free space is denoted as  $\mathbf{q}_{\text{free}}$  in the algorithm flowchart in figure 3.1, where  $\mathbf{q}$  in general represents a configuration in the workspace. In the two-dimensional workspace considered in this research,  $\mathbf{q} \in \mathfrak{R}^2$ , and is denoted by the vector  $\mathbf{q} = [x, y]^T$ .

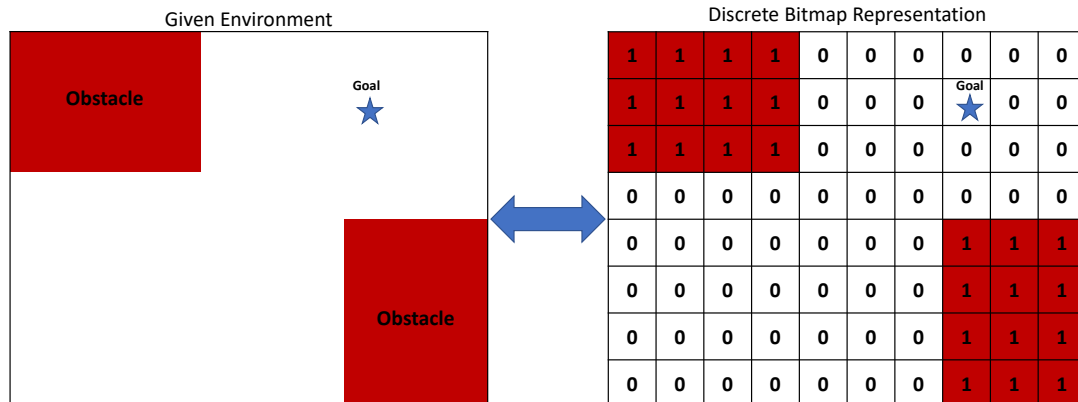


Figure 3.2: Bitmap representation of working environment.

Also, during initialization, the obstacle potential levels are set uniformly to a large number and these configurations are separated for the rest of the algorithm. The potential level of the desired final state is then set to zero, to ensure that it is the global minimum, and these values are inserted into a list,  $list_k$ , where the index  $k = 0$  initially, i.e.  $list_0(\mathbf{x}_{goal}, 0)$ .

Another attribute that is set during initialization is the model to base the navigation function generation. The model will motivate how to set the final objective state,  $\mathbf{x}_{goal}$ , how the state is approximated in the neighboring points and how the cost associated with the control effort is computed. For the results in this dissertation, the model is defined based on the linear and nonlinear kinematic equations for a two-wheel differential drive mobile robot. The models and their nuances are described in sections 3.2 and 3.3.

### 3.1.2 Main Block

Once set, the information from the initialization block is passed into the algorithm's main block to generate the navigation function. Within the algorithm, the states and potentials within  $list_k$  are used to evaluate the potential values of their neighbors,  $\mathbf{x}_0(j)$ , where the index  $j$  denotes the  $j^{th}$  neighbor point. For two-dimensional workspace examples, the algorithm considers each of the 2-neighbors of the  $i^{th}$  state,  $\mathbf{x}_f(i)$ , in  $list_k$ . The 2-neighbors are defined as the configurations in the grid that have at most two coordinates differing from the central point. There are eight 2-neighbors for a configuration when considering a two-dimensional grid [12]. An illustration of this procedure, as well as the grid representation for the mobile robot model is shown in figure 3.3.

Next, it must be determined if the neighbor point is in the free space and if the potential has not yet been computed. If the neighbor point is both free and has

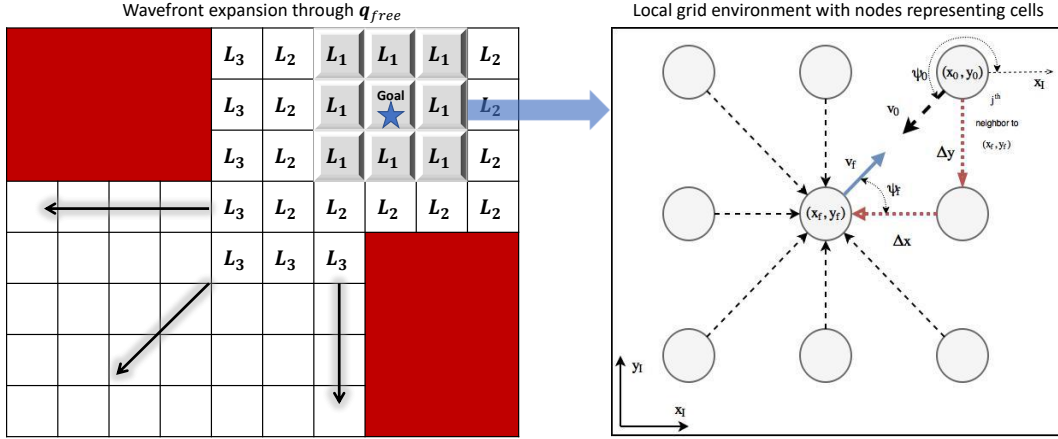


Figure 3.3: Wavefront expansion cost propagation.

not been visited by the wavefront expansion, then the algorithm continues; otherwise that point is disregarded and the next neighbor point,  $\mathbf{x}_0(j + 1)$ , is considered. This step ensures that there are no overlapping values and that the navigation function is only evaluated in the free space.

Also, for the free neighbor points, the algorithm completes the state information of  $\mathbf{x}_0(j)$ , since the only information known at this point is the position in the discrete grid. Additionally, the control effort to go from  $\mathbf{x}_0(j)$  to  $\mathbf{x}_f(i)$ , is computed based on the method chosen (Minimum Control Energy or Inverse Dynamic). And the value for this control effort,  $\mathbf{u}(\mathbf{x}_0(j), \mathbf{x}_f(i))$ , is calculated using either eq. (3.5) or eq. (3.9).

The resulting control vector and position information are compared with the constraints

$$\|\mathbf{u}_0\| \leq u_{max} \quad (3.1)$$

$$d \leq d_{eff} \quad (3.2)$$

where  $\mathbf{u}_0 = \mathbf{u}(\mathbf{x}_0(j), \mathbf{x}_f(i))$ , and  $u_{max} \in \mathfrak{R}$  and  $d_{eff} \in \mathfrak{R}$  are scalar parameters denoting the maximum control limit and effective distance respectively which are set during initialization. The value of  $d$ , in eq. (3.2), represents the Euclidean distance of the corresponding neighbor point,  $\mathbf{x}_0(j)$ , to the final desired position in the grid,  $\mathbf{x}_{goal}$ .

Effectively, the  $d_{eff}$  term in eq. (3.2) does make this method partially dependent on a measure of distance. However, the  $d_{eff}$  term is associated solely with the maximum control effort constraint,  $u_{max}$ . The combination of the constraints in eqs. (3.1) and (3.2) are used to enforce the notion that it would be cost prohibitive to approach the final desired state from certain configurations, such as facing the opposite direction at close proximity. The  $d_{eff}$  parameter in this sense restricts the  $u_{max}$  constraint to only affect grid points within a certain neighborhood of the goal and has no bearing on the potential levels through the rest of the free space.

Now, if the constraints in eqs. (3.1) and (3.2) are violated at a given neighbor point,  $\mathbf{x}_0(j)$ , then its associated cost is set equal to that of the obstacle configurations, i.e.  $J(\mathbf{x}_0(j)) = J(obs)$ . But, if the cost for the neighboring states do not violate the constraints, then it is computed by

$$J(\mathbf{x}_0(j)) = \frac{1}{2} (\mathbf{u}_0^T \mathbf{u}_0) \delta t + J(\mathbf{x}_f(i)) \quad (3.3)$$

where  $\delta t$  is a time step set during initialization for the maneuver to occur. The cost computed in equation (3.3) is used to create the contour levels making up the navigation function.

There are a couple of considerations with the cost function in eq. (3.3). The first is that the cost is always increasing from one level to the next with the wavefront expansion, this feature is what enables us to claim that there is a unique and global minimum at the goal. Second, the choice of grid resolution, or the spacing between



the grid points, and the time-step parameter  $\delta t$ , does affect the magnitude of the cost through the grid; however, they do not affect the overall shape of the potential field and its resulting path plan.

Finally, when the cost for  $\mathbf{x}_0(j)$  is computed, it is appended to the bottom of a new list, indexed at  $k+1$ , written as

$$\text{list}_{k+1}(\mathbf{x}_0(j), J(\mathbf{x}_0(j)))$$

in figure 3.1. This process is continued for each neighboring configuration of  $\mathbf{x}_f(i)$ .

If all the neighbors have been visited, then the next final state and cost in  $\text{list}_k$  are used, i.e.  $\mathbf{x}_f(i+1)$  and  $J(\mathbf{x}_f(i+1))$ . However, if all the elements in  $\text{list}_k$  have been used, then the values from the next list,  $\text{list}_{k+1}$  are considered. The algorithm continues until  $\text{list}_k$  is empty. When  $\text{list}_k$  is empty, then all the points in  $\mathbf{q}_{\text{free}}$  that are connected to the objective position have been visited by the wavefront expansion and the navigation function generation is finished.

Once the navigation function algorithm finishes, the reference path,  $\mathbf{x}_r$ , is obtained through a best-first graph search following the negative gradient of the resulting potential field from a given initial location to the objective location. Through this method, the paths are attained by observing the potential values of the neighboring points and then choosing the neighbor with the lowest value. This is done in an iterative manner until the objective is found. In other words, the path is found by starting at some initial location in the environment and continuing along a path of minimal control effort until it reaches the goal.

### 3.2 Minimum Control Energy Approach

The minimum control effort based path plan is designed considering the solution to the optimal control problem for finding the minimum energy controller. The cost function associated with this problem is given by

$$\min_{\mathbf{u} \in \mathbb{R}^m} J = \frac{1}{2} \int_{t_0}^{t_f} \mathbf{u}^T \mathbf{u} dt \quad (3.4)$$

subject to

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}(t_f) &= \mathbf{x}_f \end{aligned}$$

Within the context of the grid-based path planner, the above represents the cost going from one grid point at  $t_0$  to a neighboring grid point at  $t_f$  (denoted as  $\mathbf{x}_0$  and  $\mathbf{x}_f$  respectively). The solution to this problem can be found in most optimal control textbooks such as reference [64]. The analytic solution for the open-loop, minimum energy controller for a linear model is given by

$$\mathbf{u}_L = \mathbf{B}^T \Phi^T(t_f, t) \mathbf{W}_R^{-1}(t_0, t_f) [\delta \mathbf{x}_f - \Phi(t_f, t_0) \delta \mathbf{x}_0] \quad (3.5)$$

where  $\delta \mathbf{x}_f$  and  $\delta \mathbf{x}_0$  are perturbations of the state about the nominal trajectory at  $t_f$  and  $t_0$  respectively. Also,  $\mathbf{W}_R(t_0, t_f)$  is the reachability Gramian and  $\Phi(t_f, t)$  is the state transition matrix of the system [64]. The state transition matrix is computed as

$$\Phi(t_f, t) = e^{\mathbf{A}(t_f - t)}$$

and the reachability Gramian is computed by

$$\mathbf{W}_R(t_0, t_f) = \int_{t_0}^{t_f} \Phi(t_f, \tau) \mathbf{B} \mathbf{B}^T \Phi^T(t_f, \tau) d\tau \quad (3.6)$$

which needs to be non-singular in order for the state at  $t_f$  to be reachable.

Solving for the reachability Gramian using eq. (3.6) is not straightforward; however, there are simplifications to find a solution which exist in the linear systems literature, such as reference [65]. One method to find a solution to eq. (3.6) is to look at the analytic solution of the differential Lyapunov equation. The solution to the differential Lyapunov equation of the form

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^T + \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T \quad (3.7)$$

is given by

$$\mathbf{P}(t) = \Phi(t, t_0)\mathbf{P}(t_0)\Phi^T(t, t_0) + \int_{t_0}^t \Phi(t, \tau)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\Phi^T(t, \tau) d\tau \quad (3.8)$$

Therefore, by taking  $\mathbf{P}(t_0) = 0$  and setting  $\mathbf{R}$  equal to the identity matrix, the right hand side of eq. (3.8) is equal to the right hand side of eq. (3.6). Hence, the reachability Gramian can be found by integrating the differential Lyapunov equation from  $t_0$  to  $t_f$ , in eq. (3.7), assuming  $\mathbf{W}_R(t_0, t_0) = \mathbf{0}$ .

Additionally, the aspect of reachability for the system being studied is formally addressed with the following remark. The motivation for this assertion can be found in the discussions of chapter 6 from reference [65] and chapter 3 in reference [66] involving the concept of a system's controllability and its implications with regards to reachability. The aim is to show that if the linear system is controllable, then it can be directed to successive local reachable states, with each one leading the system to the final objective reachable state at the end of the path plan.

**Remark 1.** *Consider the nonlinear system in eq. (2.1) and its linearization in eq. (2.6) about the nominal state,  $\mathbf{x}_n$  and  $\mathbf{u}_n$ . If the linear system is controllable, then the nonlinear system is locally controllable in the neighborhood of  $(\mathbf{x}_n, \mathbf{u}_n)$ . The linear and the nonlinear system are also locally reachable. Thus, successive state transitions of the linear system from controllable to controllable configurations, ensures*

that the nonlinear system is transferred through locally reachable states to the goal location [65, 66].

In eq. (3.5) the input vector,  $\mathbf{u}_L \in \mathfrak{R}^2$ , is defined as  $\mathbf{u}_L = [\delta u_1, \delta u_2]^T$ , where  $\delta u_1$  and  $\delta u_2$  are the commanded heading angle turn rate and forward acceleration respectively.

$\mathbf{A}$  and  $\mathbf{B}$  for the linear kinematic model are obtained using the nominal trajectory. The values for the nominal trajectory  $\mathbf{x}_n$  are computed by specifying a sampling time,  $\Delta t$ , and assume it is a straight line maneuver between the node position from  $\mathbf{x}_0$  (i.e. the values  $x_0$  and  $y_0$ ) and the node position from  $\mathbf{x}_f$  (i.e. the values  $x_f$  and  $y_f$ ) within a grid as illustrated in figure 3.4. Then, the nominal heading angle is taken as the direction pointing from the node position of  $\mathbf{x}_0$  to the node position of  $\mathbf{x}_f(i)$  within the grid, given by

$$\psi_n = \tan^{-1} \left( \frac{y_f - y_0}{x_f - x_0} \right)$$

And the nominal velocity is computed as

$$v_n = \sqrt{\left( \frac{x_f - x_0}{\Delta t} \right)^2 + \left( \frac{y_f - y_0}{\Delta t} \right)^2}$$

To compute the control effort from equation (3.5), the state information at  $t_0$  and at  $t_f$  needs to be determined. Within the path planning algorithm, the final state at each level is taken as the  $i^{th}$  state extracted from the  $k^{th}$  list, i.e.  $\mathbf{x}_f(i)$ , which is initially set as the desired final reachable state. The final objective reachable state is denoted as the vector  $\mathbf{x}_{goal}^{MCE} = [x_g, y_g, \psi_g, v_g]^T$ . And figure 3.4 illustrates how the state is interpreted for the kinematic model within a discrete grid environment as considered in this algorithm.

At  $t_0$ , the state information is denoted as  $\mathbf{x}_0$ . And the state vector,  $\mathbf{x}_0$ , is found by considering the 2-neighbor points of  $\mathbf{x}_f(i)$  and is completed assuming that the vehicle at the neighbor location is approaching the state at  $t_f$ . In other words, the



determined by taking the difference between the state at  $t_f$ , or  $t_0$ , and the nominal values,  $\mathbf{x}_n$ , as

$$\delta\mathbf{x}_f = \mathbf{x}_f - \mathbf{x}_n$$

$$\delta\mathbf{x}_0 = \mathbf{x}_0 - \mathbf{x}_n$$

This procedure is done for each of the free neighbors of  $\mathbf{x}_f(i)$ . This approach will ultimately influence the shaping of the navigation function's contours based on the final desired reachable state that considers the linear kinematic model of a mobile robot with the minimum energy controller solution.

### 3.3 Inverse Dynamics Approach

If the model is chosen as the nonlinear kinematic model during initialization, then an inverse dynamics based method for finding the control effort is used. The robot's state vector is given by  $\mathbf{x} = [x, y, \psi, v]^T$  which represents the position, heading angle and velocity taken at a given instant. The nonlinear kinematic model is considered as

$$\dot{x} = v \cos(\psi)$$

$$\dot{y} = v \sin(\psi)$$

$$\dot{\psi} = u_1$$

$$\dot{v} = u_2$$

for this approach with the path plan algorithm. The control terms are found in the heading angle turn rate and the acceleration of the system ( $u_1$  and  $u_2$ ). The control values can be rewritten as

$$u_1 = \alpha_1(\psi^c - \psi)$$

$$u_2 = \alpha_2(v^c - v)$$

where  $\alpha_1 > 0$ ,  $\alpha_2 > 0$  are scalar constants and  $\psi^c$  and  $v^c$  are the guidance commands for the heading angle and velocity of the system. Then, the two-dimensional control effort vector is

$$\mathbf{u}_{\text{NL}} = \begin{bmatrix} \alpha_1 (\psi^c - \psi) \\ \alpha_2 (v^c - v) \end{bmatrix} \quad (3.9)$$

The control effort computed using equation (3.9) is motivated by the stable nonlinear trajectory tracking guidance design detailed in chapter 6 along with a proof of its stability.

The actual commanded heading angle and velocity values are computed based on the stability analysis of the system. However, for the navigation function algorithm, it is assumed that the commanded heading angle and velocity values are set as constants in the state at the end of a maneuver, at time  $t_f$ . Therefore, it is included as part of the final state, denoted as  $\mathbf{x}_f$ . The final state,  $\mathbf{x}_f$ , is initially set as the desired reachable state given by  $\mathbf{x}_{goal}^{ID} = [x_g, y_g, \psi_g, v_g]^T$ .

In order to compute the control effort using equation (3.9), the state for the neighboring points must be completed based on the model. The position information is extracted from the discrete workspace, but the heading angle and velocity values still need to be calculated in order to complete the neighbor's state  $\mathbf{x}_0$ . Figure 3.4 illustrates how the heading and velocity values are found for the mobile robot model.

The values for  $\psi_0$  and  $v_0$  at the neighbor points to  $\mathbf{x}_f$  are found assuming an approaching maneuver from  $\mathbf{x}_0$  to  $\mathbf{x}_f$  over a fixed time step,  $\delta t = t_f - t_0$ . From figure 3.4, the heading angle at the neighboring point is taken as an approach angle directed towards the desired final position, and it is measured positively from the inertial  $x$ -axis in a counter-clockwise direction. So, the heading angle can be found by

$$\psi_0 = \tan^{-1} \left( \frac{\dot{y}_0}{\dot{x}_0} \right)$$

where

$$\begin{aligned}\dot{y}_0 &= v_f \sin(\psi_f) + \frac{y_f - y_0}{\delta t} \\ \dot{x}_0 &= v_f \cos(\psi_f) + \frac{x_f - x_0}{\delta t}\end{aligned}$$

And the velocity at the neighboring point is found using

$$v_0 = \sqrt{\dot{x}_0^2 + \dot{y}_0^2}$$

These values are computed for each of the free neighboring points of  $\mathbf{x}_f$ . This approach will influence the shaping of the navigation function's contours based on the final reachable state that is given based on the nonlinear kinematic model and the inverse dynamics approach.

### 3.4 Simulation Results

The examples presented in this chapter are based in a two-dimensional, flat, Euclidean workspace. It is assumed that knowledge of the workspace such as the obstacle regions, model properties, and the objective reachable state are fully available to the vehicle.

There will be two example environments shown. The first will have no obstacles present, this example will illustrate the features of the potential field's formation with this algorithm. Then, the second example will have some obstacles present, which will be used to demonstrate the obstacle avoidance capability of the algorithm. The two methods with this algorithm, the minimum control energy and inverse dynamics approaches, will be applied to the given scenarios.

#### 3.4.1 Case 1: No obstacles present

For the first example, the final desired reachable state is  $\mathbf{x}_{goal}^{MCE} = \mathbf{x}_{goal}^{ID} = [30, 30, \frac{\pi}{4}, 0.5]^T$ . This results in a path plan that would approach the objective



location with a heading angle pointing towards the upper right hand corner of the workspace.

The navigation function algorithm is used with both of the methods described in this chapter. The first set of results are shown in figures 3.5 and 3.6. The minimum control energy approach is shown in figure 3.5 and the inverse dynamics approach in figure 3.6. The results both direct the path to the final desired reachable state.

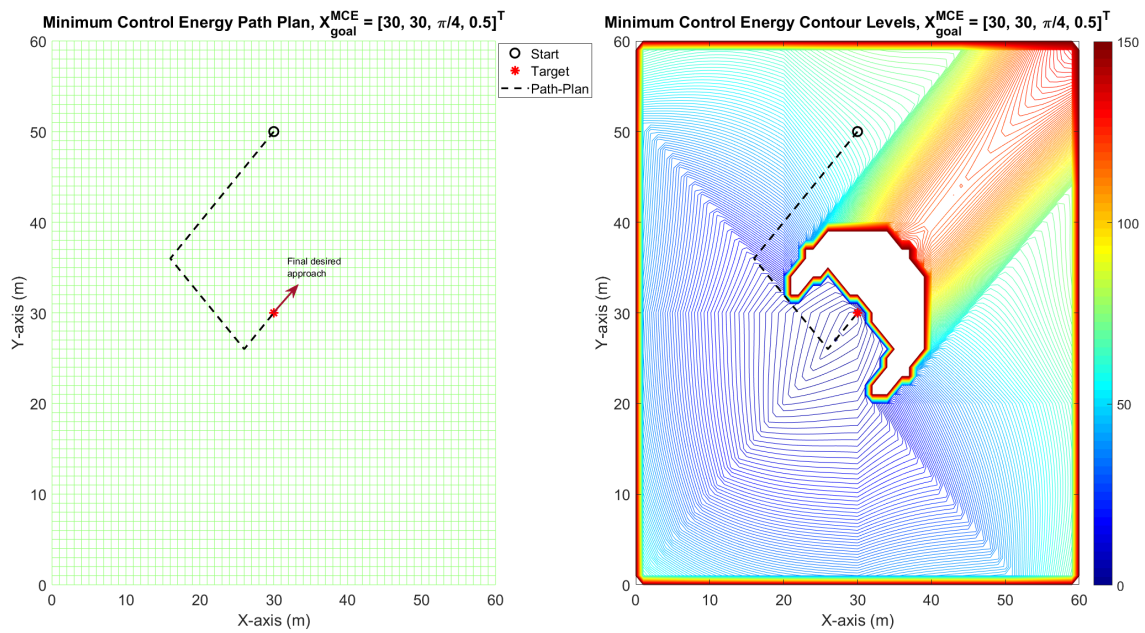


Figure 3.5: Minimum control energy path plan and potential field.

The value of  $u_{max}$  is set as  $u_{max} = 7$  for both approaches. The effects of the constraint of  $u_{max}$  is seen in both figures as the crescent shape peaked in the potential contours. This demonstrates that it will block paths from being formed that approach the objective state from unfavorable directions.

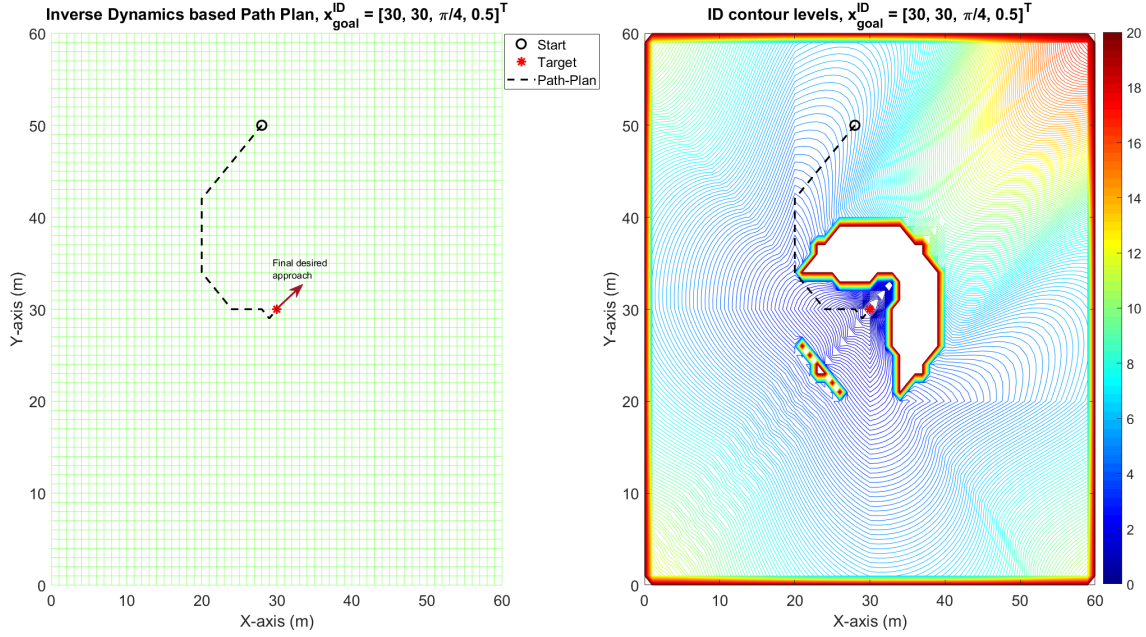


Figure 3.6: Inverse dynamics approach, path plan and potential field.

### 3.4.2 Case 2: Obstacles Present

The following examples demonstrate the obstacle avoidance capability of the path planner with multiple obstacles present. The final desired reachable state for these examples is taken as  $\mathbf{x}_{goal}^{MCE} = \mathbf{x}_{goal}^{ID} = [30, 35, \frac{\pi}{2}, 0.5]^T$ . This objective reachable state would direct the path to a position in between two obstacles with a resulting heading angle pointed in the north direction, i.e.  $\psi_g = \pi/2$ . The resulting path plans and potential contour levels are shown in figures 3.7 and 3.8.

The results with the path planner show that the paths generated avoid collisions and direct the vehicle to a state in between two obstacles. The resulting path plans are different because of the different models being observed.

For the results shown in figures 3.7 and 3.8, the value for  $u_{max}$  is set differently. For the minimum control energy approach, the value is set at  $u_{max} = 7$  and for the inverse dynamics approach, the value is  $u_{max} = 10$ .

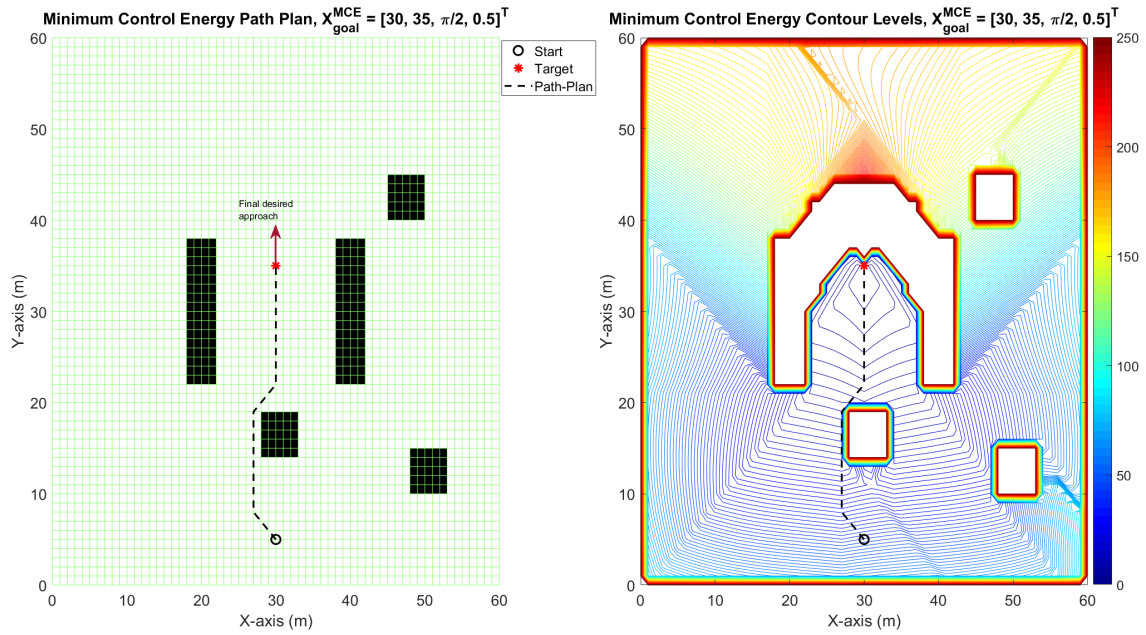


Figure 3.7: Minimum control energy path plan and potential field.

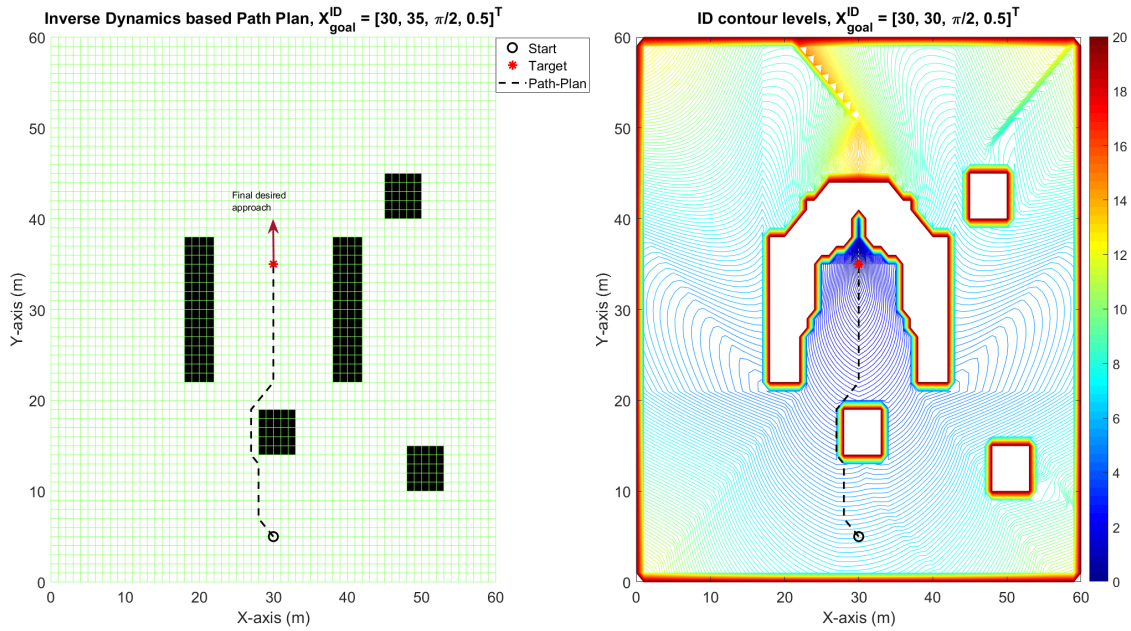


Figure 3.8: Inverse dynamics approach, path plan and potential field.

Also, from the results shown here and in the previous subsection, it can be seen that the potential fields only possess one global minimum. This is a result that comes

from the wavefront expansion construction method used. This also implies that a path plan can be formed to the final reachable state from any free configuration in the given environment. This also implies that the navigation function only needs to be generated once for the given environment, or if new information is obtained, and the path can be found as needed.

### 3.4.3 Effects of Changing $u_{max}$

The value of  $u_{max}$  in the path planning algorithm is set as a user defined parameter to design the path plans that best approach the final desired state. Finding a proper value of  $u_{max}$  can help shape both the path plan and the potential field, which is illustrated in figures 3.9-3.12.

Note that the figures which contain white space depict the effects of setting the value of  $u_{max}$  too small. When this is done, the potential field will not be fully formed as the spaces around the final desired state are deemed too costly and violate the constraints set.

Overall, the results in figures 3.9-3.12 depict the effects of changing  $u_{max}$  for the given scenarios. The plots reveal the potential level contours in the given environment. It can be seen from the contour levels that the final approach of the path plan to the reachable state is affected by the changing the value of  $u_{max}$ .

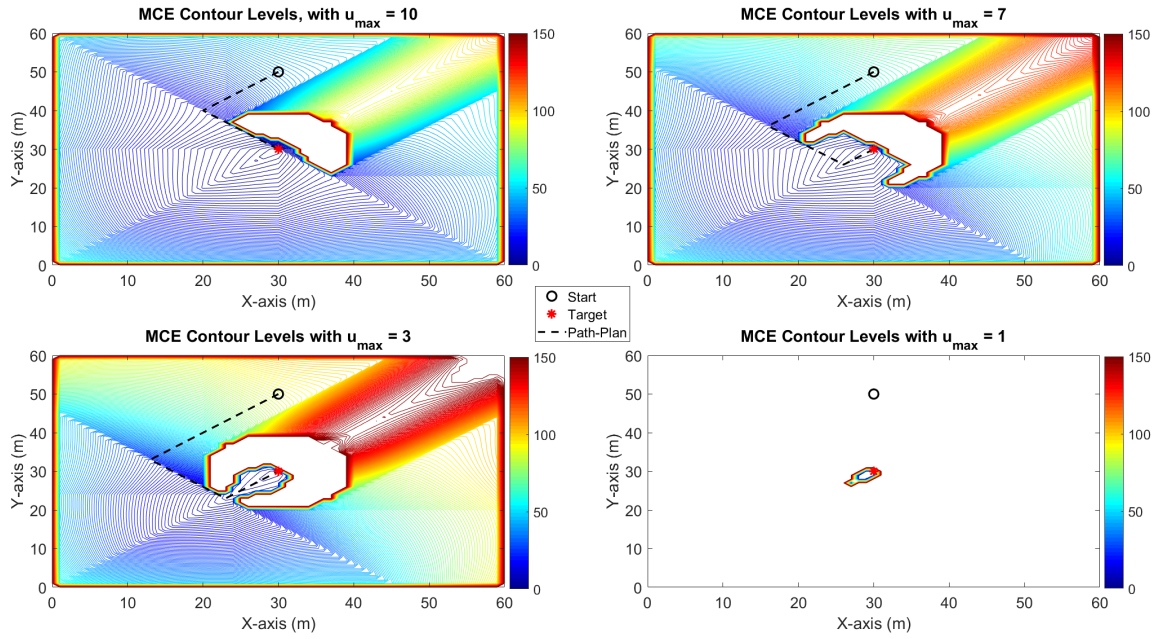


Figure 3.9: Effects of tuning  $u_{max}$  with MCE approach, first example.

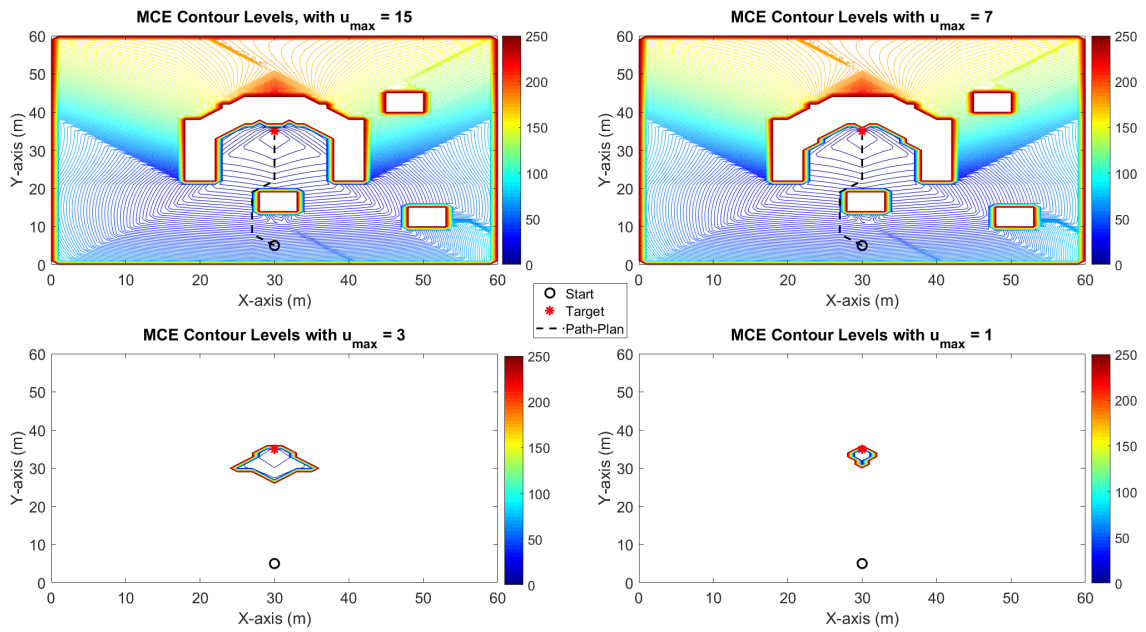


Figure 3.10: Effects of tuning  $u_{max}$  with MCE approach, second example.



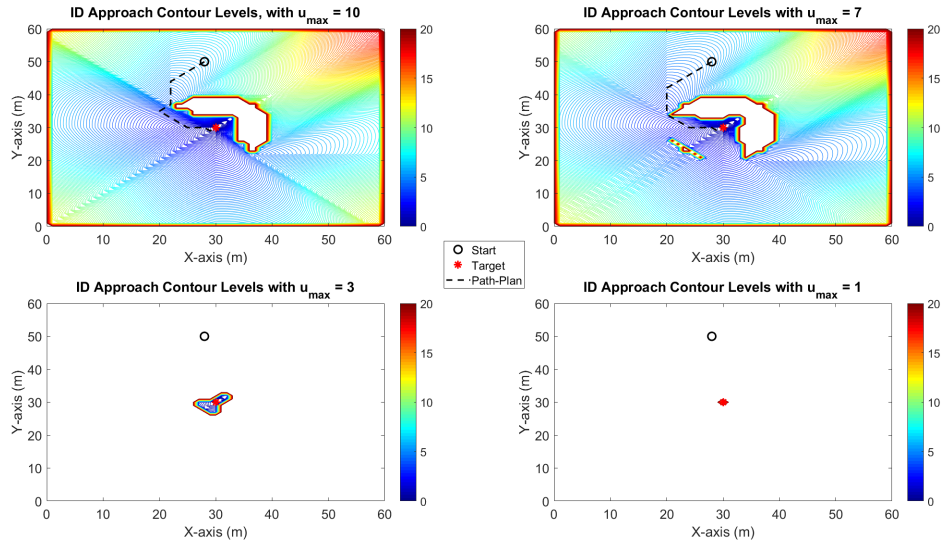


Figure 3.11: Effects of tuning  $u_{max}$  with ID approach, first example.

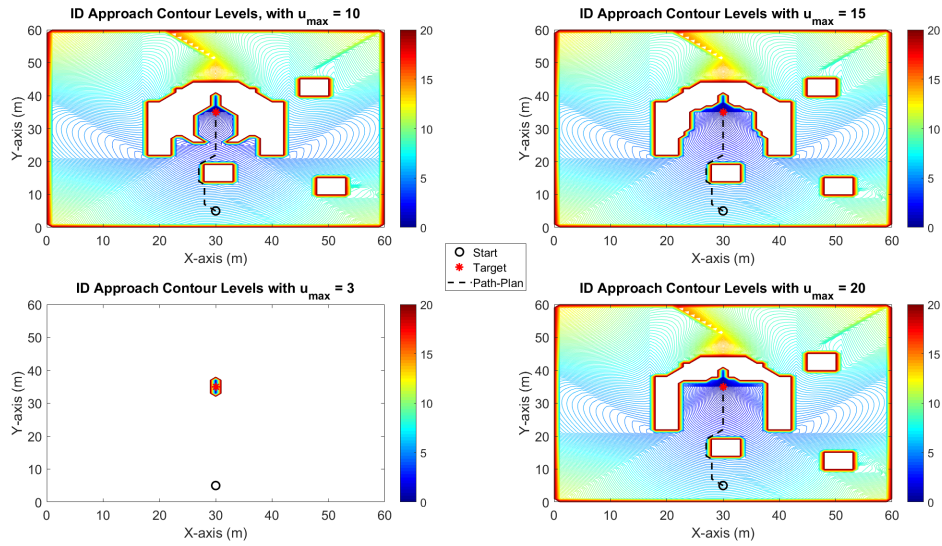


Figure 3.12: Effects of tuning  $u_{max}$  with ID approach, second example.

## CHAPTER 4

### Path Planning Extensions

This chapter presents extensions of the work discussed in chapter 3 with the minimum control effort-based approach. The results in this chapter make use of modified versions of RRT\* and D\* to generate path plans with a control effort-based metric. References [8] and [25] provide a full description of the RRT\* and D\* algorithms, respectively, and insight into their ability to find optimal path plans in the presence of uncertainty since they are special cases of dynamic programming [27]. This chapter will present a summary of the algorithms with added changes to make use of the minimum control effort metric discussed in section 3.2 of this dissertation.

The objective with this aspect of the dissertation is to explore different construction methods with the minimum control effort metric. The RRT\* construction method was chosen due to its ability to rapidly explore a given environment without knowledge of the objective. Also, RRT\* eliminates the issues with resolution and curse of dimensionality that arise with grid-based planners such as the method outlined in chapter 3. The D\* algorithm is chosen because it can account for uncertainty in the terrain and can be used to quickly regenerate path plans when new information is acquired. This chapter will summarize both algorithms and give simulated path planning results demonstrating the desired capabilities.

Similar to the numerical navigation function planner in chapter 3, these approaches will consider the kinematics of a wheeled mobile robot with a state vector  $\mathbf{x} = [x, y, \psi, v]^T$  denoting the vehicle's  $x$  and  $y$  position coordinates, its heading angle,  $\psi$  and its forward velocity  $v$ .





$N$  leads to a larger portion of the environment being explored. Also, as  $N \rightarrow \infty$ , the minimum cost path approaches the optimal value with a probability of 1 [8].

Other quantities set during initialization are the maximum step size ( $\Delta\mathbf{q}$ ) and the maximum time step ( $\delta t$ ) required to go that distance from a node. The time step is used to complete the state of the node and compute its cost. The maximum search range ( $\epsilon_{near}$ ) is used to find the nodes in the neighborhood that can be rewired to find an optimal traversal.

The starting point and state ( $\mathbf{q}_{init}$ ) is also set during initialization, with a cost of 0, and placed within the node list with an empty edge pointer ( $E(\mathbf{q}_{init}) = Empty$ ). The node list is used in the algorithm to keep track of the nodes in the tree and their associated cost and edge pointers.

#### Modified RRT\* - Main Block

The first step is to extract the  $i^{th}$  sample of the environment, denoted as  $\mathbf{q}_{rand}$ . The random sample is sent to the  $Nearest(\text{node-list}, \mathbf{q}_{rand})$  procedure which extracts the nearest node  $\mathbf{q}_{nearest}$  from the node list. The nearest node and the random node are used to find the new node, denoted as  $\mathbf{q}_{new}$  which lies within  $\Delta\mathbf{q}$  of  $\mathbf{q}_{nearest}$  using the  $steer(\mathbf{q}_{nearest}, \mathbf{q}_{rand})$  procedure.

An illustration of how  $\mathbf{q}_{new}$  is obtained using the  $steer$  procedure is given in figure 4.2. Essentially, if the distance between  $\mathbf{q}_{nearest}$  and  $\mathbf{q}_{rand}$  is greater than  $\Delta\mathbf{q}$ , then the position of  $\mathbf{q}_{new}$  is found by interpolating along the line connecting  $\mathbf{q}_{nearest}$  and  $\mathbf{q}_{rand}$  and the time step ( $\delta t$ ) is associated to travel from  $\mathbf{q}_{nearest}$  to  $\mathbf{q}_{new}$ . Conversely, if the distance from  $\mathbf{q}_{nearest}$  to  $\mathbf{q}_{rand}$  is smaller than  $\Delta\mathbf{q}$ , then  $\mathbf{q}_{new} = \mathbf{q}_{rand}$  and  $\delta t$  is found through interpolation.

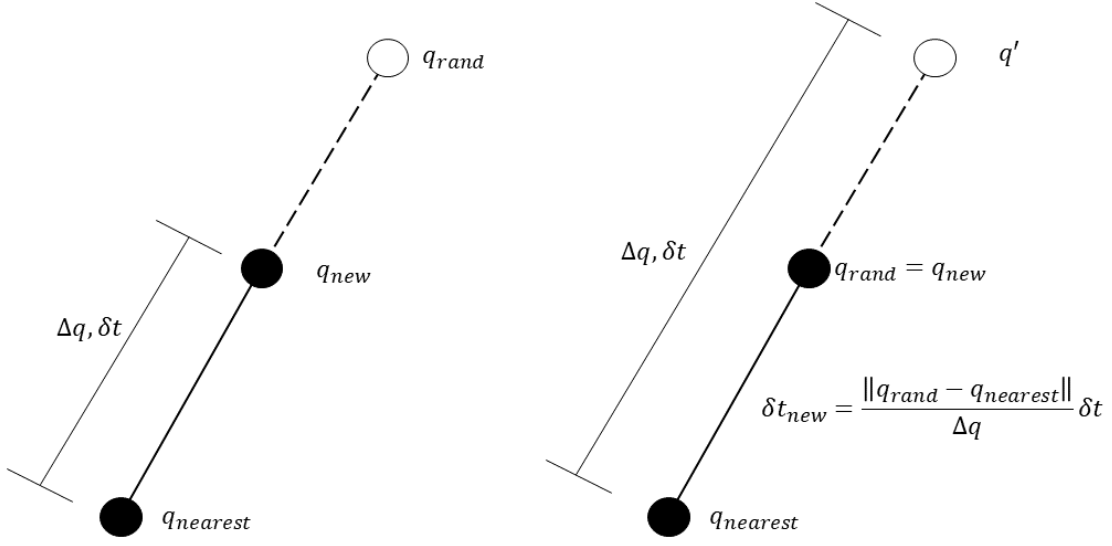


Figure 4.2: Steer procedure within RRT\* algorithm.

Once  $\mathbf{q}_{new}$  is obtained, it is checked by the  $\text{Free}(\mathbf{q}_1, \mathbf{q}_2)$  procedure, which is used to determine if the edge connecting two nodes, say  $\mathbf{q}_1$  and  $\mathbf{q}_2$ , intersects with any obstacles present. If the line segment is free, then the set of neighboring nodes (within a range  $\epsilon_{near}$  of  $\mathbf{q}_{new}$ ) from the node list is obtained, denoted by  $\mathbf{X}_{near}$ . Each of the nodes in  $\mathbf{X}_{near}$  is checked to find if a less costly connection can be made to  $\mathbf{q}_{new}$ . The optimal connection to  $\mathbf{q}_{new}$  is denoted as  $\mathbf{q}_{min}$  in the algorithm flowchart. This procedure is done a second time to ‘rewire’ the connections in the tree if it is more optimal for them to connect through the new node, i.e. if  $E(\mathbf{X}_{near}(j)) = \mathbf{q}_{new}$  is cheaper than the previous connection.

After the rewiring phase is complete, the node  $\mathbf{q}_{new}$  is inserted into the node list with an edge pointer to  $\mathbf{q}_{min}$ , i.e.  $E(\mathbf{q}_{new}) = \mathbf{q}_{min}$  and its associated cost is obtained as

$$J(\mathbf{q}_{new}) = \frac{1}{2}(\mathbf{u}^T \mathbf{u})\delta t + J(\mathbf{q}_{min})$$

which is the same form as equation 3.3 in chapter 3. The control effort vector ( $\mathbf{u}$ ) is obtained using the minimum control energy approach as

$$\mathbf{u} = \mathbf{B}^T \Phi^T (t_f, t) \mathbf{W}_R^{-1} (t_0, t_f) [\delta \mathbf{x}_f - \Phi (t_f, t_0) \delta \mathbf{x}_0] \quad (4.1)$$

which is the same control effort in equation 3.5, with  $\delta t = t_f - t_0$ . The main difference in this algorithm, to what was covered in chapter 3, is that the state is completed by assuming a maneuver going from the state at node  $\mathbf{q}_{min}$  to the state at  $\mathbf{q}_{new}$ , denoted as the initial state  $\mathbf{x}_0$  and the final state  $\mathbf{x}_f$ , respectively, in eq. (4.1).

The main block is executed until the number of random samples has been exhausted. At the end of the algorithm, a goal location is chosen and is connected to the closest node within the generated tree. Then, the edge pointers are used to find the minimum cost path within the generated tree. Simulation results using this procedure are shown at the end of this chapter.

## 4.2 Modified D\* Algorithm

The modified D\* algorithm is the next path planning procedure covered in this dissertation. The modified-D\* algorithm is very similar to the wavefront expansion method from chapter 3. The key difference lies in how each grid point is sorted through the process and through the use of back-pointers to rewire and connect optimal traverses over the grid. Otherwise, the method to assign a cost between two nodes and complete the state information is the same as the approach covered in section 3.2. The modified-D\* algorithm is presented in flowchart form through figures 4.3 and 4.4 and is summarized in this section in terms of the original algorithm from reference [25].

Within the D\* algorithm, each node  $\mathbf{q}$  in the grid has the following properties associated with it: A tag ( $t(\mathbf{q})$ ) which can be set to New, Open or Closed; a cost

$(h(\mathbf{q}))$ , a key value  $(k(\mathbf{q}))$  for sorting the nodes, and a back-pointer  $(b(\mathbf{q}))$  which connects  $\mathbf{q}$  to the node with the smallest cost as well as an associated heading angle and velocity to complete its state. Note, that the tag of a node represents whether it has been assigned a cost value, i.e. whether it is New or Open, and if that cost value is considered optimal then it is set to closed.

To add the minimum control energy approach with the D\* algorithm, the path costs for traversing between two nodes say  $\mathbf{q}_1$  and  $\mathbf{q}_2$ , denoted by  $c(\mathbf{q}_1, \mathbf{q}_2)$  in the flowchart depicted in figure 4.4, is evaluated as

$$c(\mathbf{q}_1, \mathbf{q}_2) = \frac{1}{2}(\mathbf{u}^T \mathbf{u})\delta t$$

where

$$\mathbf{u} = \mathbf{B}^T \Phi^T (t_f, t) \mathbf{W}_R^{-1} (t_0, t_f) [\delta \mathbf{x}_f - \Phi (t_f, t_0) \delta \mathbf{x}_0] \quad (4.2)$$

and  $\mathbf{x}_0$  denotes the state at node  $\mathbf{q}_1$  and  $\mathbf{x}_f$  is the state of node  $\mathbf{q}_2$ . And the state of a node is completed based on the same approach outlined in section 3.2.

#### Main D\* Procedure

An illustration of the main D\* algorithm in flowchart form is given in figure 4.3, also called Move-Robot in reference [25].

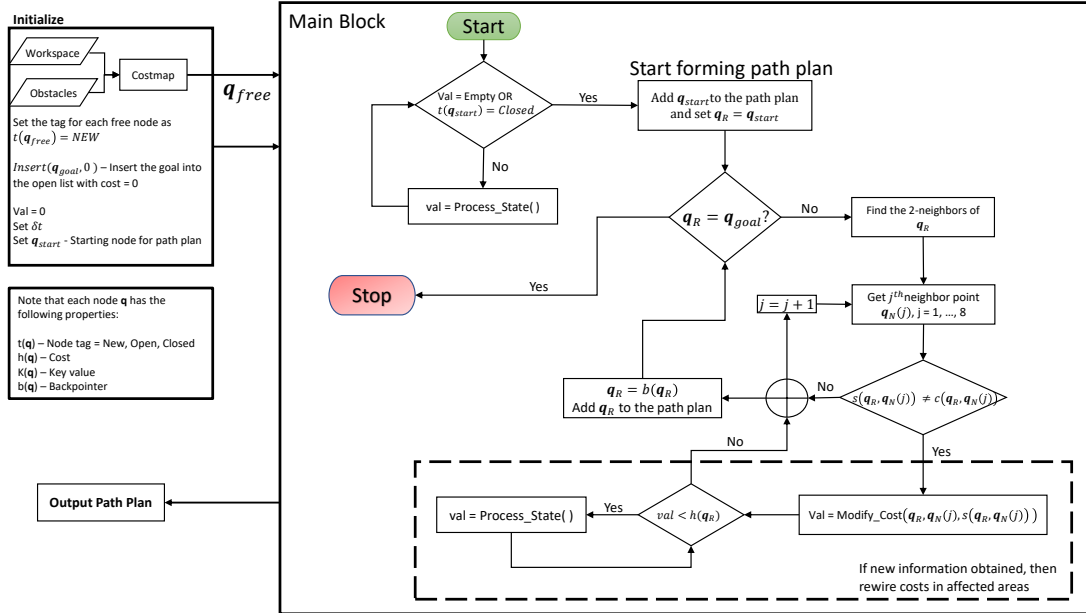


Figure 4.3: Main algorithm flowchart for the modified/original D\* algorithm.

From the flowchart in figure 4.3, the initialization block is nearly identical to the initialization of the navigation function from chapter 3. The goal state is set, with a cost of 0 and the environment is discretized and the free space, denoted as  $\mathbf{q}_{free}$  is separated from the obstacle areas. Also, for this algorithm, the tag for each of the nodes in the free space is set to New, i.e.  $t(\mathbf{q}_{free}) = \text{New}$ . The operation  $\text{Insert}(\mathbf{q}, h(\mathbf{q}))$  in the algorithm sets the tag of node  $\mathbf{q}$  to open, i.e.  $t(\mathbf{q}) = \text{Open}$ , and assigns the node a cost and key value of  $h(\mathbf{q})$ . Within the initialization, the goal node, denoted as  $\mathbf{q}_{goal}$ , is set to open and given a cost and key of zero.

Also, during initialization, the time step to traverse between two nodes in the grid ( $\delta t$ ) is set and the starting node is identified and the parameter  $val$  is set to not be empty. Setting the start node and the parameter  $val$  are conditions to determine when the main process stops. The D\* algorithm is designed to stop when the starting

node has been visited and assigned a tag of  $t(\mathbf{q}_{start})=Closed$  or if the parameter  $val$  is empty.

The propagation of the cost to each of the nodes in the grid is handled by the Process-State procedure, illustrated in figure 4.4. The process-state procedure is iterated upon, assigning costs to each node, and stops when either it has processed the starting node  $\mathbf{q}_{start}$  or if it returns an empty value. Once the process-state loop has been evaluated, the main block in figure 4.3 starts to find a path plan. The path plan is generated from the starting node,  $\mathbf{q}_{start}$ , following back-pointers until it reaches the goal.

The main block of the algorithm has a built in mechanism such that whenever a change in the sensed cost between two nodes, say node  $\mathbf{q}_1$  to node  $\mathbf{q}_2$ , is different from the cost associated with the planned cost. Then, the grid points in the affected areas are ‘rewired’ to find the optimal path. Note that the sensed cost in the flowchart is denoted by  $s(\mathbf{q}_1, \mathbf{q}_2)$  and the current path cost between the two nodes is  $c(\mathbf{q}_1, \mathbf{q}_2)$ .

When this discrepancy occurs, the Modify-Cost procedure is employed to change the cost in the affected areas and then run the process-state procedure until the path is optimally ‘rewired.’ This mechanism is what makes the D\* algorithm attractive in that it only makes changes to the cost in the affected areas and doesn’t need to completely regenerate the cost map over the grid to find the path plan.

### Process-State Procedure

The process-state procedure is at the center of the D\* algorithm and is given in flowchart form in figure 4.4 and is based on reference [25]. It is integral to how the nodes are processed, sorted and assigned costs. At the start of the process-state procedure, the node with the minimum key value and that has its tag set to open is

extracted, denoted by  $\mathbf{X}$  in figure 4.4. The neighboring nodes of  $\mathbf{X}$  are gathered and denoted by  $\mathbf{Y}(j)$  where the index  $j$  denotes the  $j^{th}$  neighbor node.

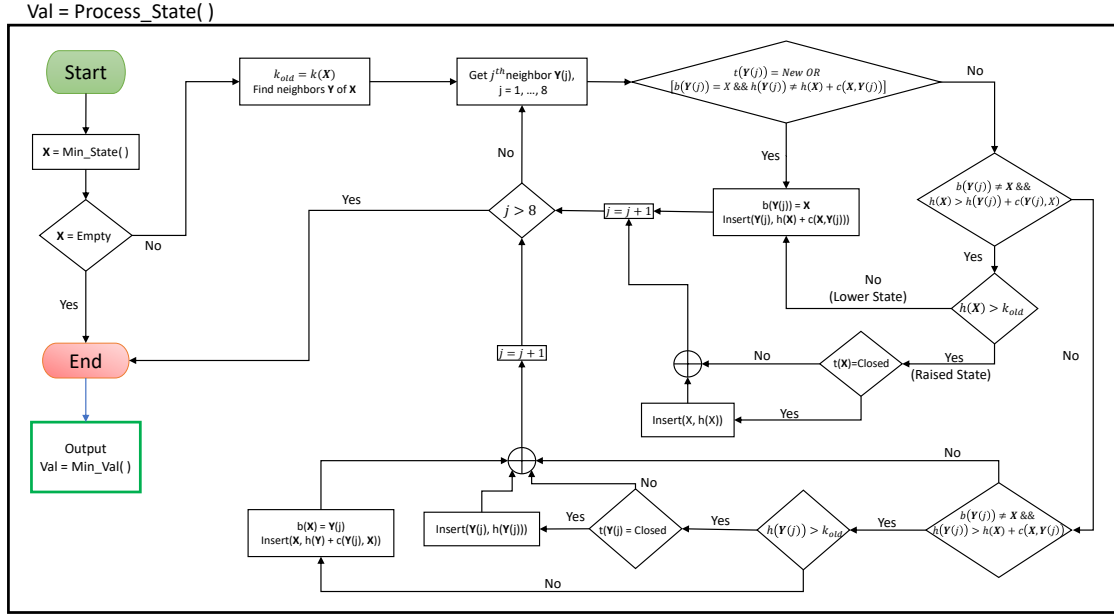


Figure 4.4: Process-State procedure of D\* algorithm.

Through process-state, the path costs of  $\mathbf{Y}(j)$  are evaluated and checked if its path cost  $h(\mathbf{Y}(j))$  can be reduced. The process directs the back-pointers of  $\mathbf{Y}(j)$  such that they direct the plan along an optimal path cost route. This is done for each neighbor of  $\mathbf{X}$ . The output of the process-state procedure is the value of the key of  $\mathbf{X}$ , denoted as  $k(\mathbf{X})$ . This procedure is invoked to reduce or change the costs in the environment.

### 4.3 Simulation Results

The simulation results in this chapter are presented for each algorithm with a similar environment setup. In both scenarios, there is a single obstacle present in

the middle of the environment and a path plan is generated to an objective in the far-upper corner of the two-dimensional environment. The only difference is that the modified-D\* algorithm considers an environment discretized into a 60x60 grid and the modified-RRT\* algorithm considers a 30mx30m environment.

### 4.3.1 Modified RRT\* Path Plan Results

The results using the modified RRT\* algorithm are presented in this subsection. In the simulation examples given, the same environment is used but with a different number of samples taken of the 2D environment. For each case the following parameters are set as  $\Delta \mathbf{q} = 0.5 \text{ m}$ ,  $\delta t = 1 \text{ sec.}$ ,  $\epsilon_{near} = 2 \text{ m}$ , and the initial state is set as  $\mathbf{q}_{init} = [0, 0, 45 \text{ deg.}, 0 \text{ m/s}]^T$ . The first case has 500 samples taken and the second has 1000. The change in the number of samples is chosen to illustrate how it affects the exploration of the given environment.

The resulting tree expanded through the given environment is shown in figures 4.5 and 4.6 for the 500 sample and 1000 sample cases respectively.

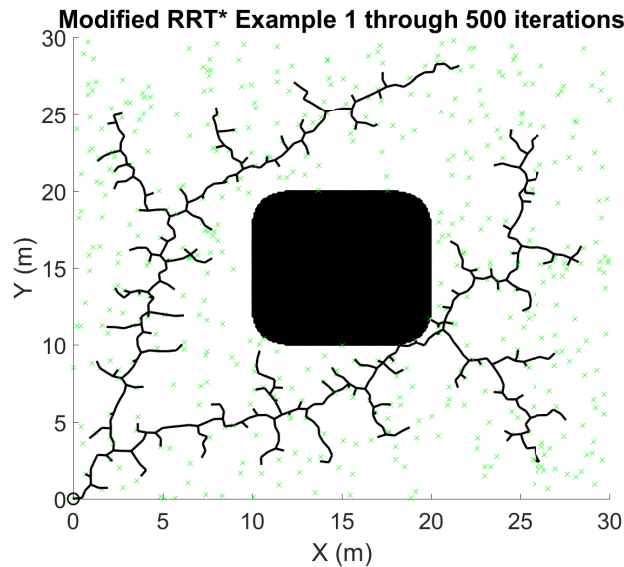


Figure 4.5: Exploring Tree using modified-RRT\* with 500 samples



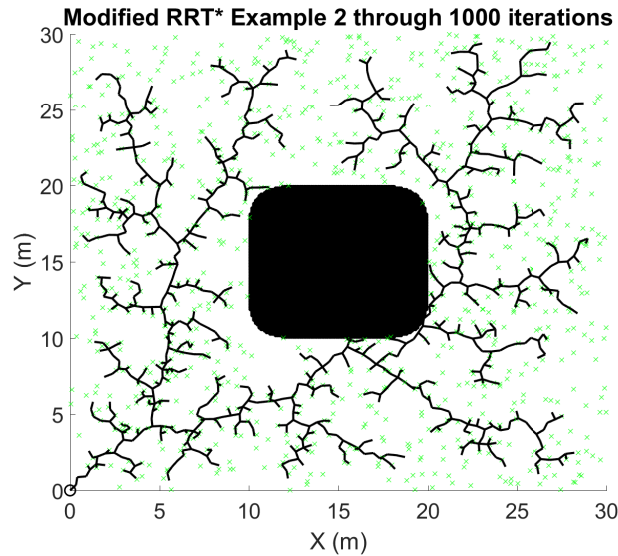


Figure 4.6: Exploring Tree using modified-RRT\* with 1000 samples

From the results, it is evident that when the number of samples increases, so does the area of exploration. Also, from the figures, the tree is expanded without any knowledge of where the goal is located making this approach suitable for cases when the destination is not entirely obvious or given. One potential drawback of this approach is that as the number of samples increase, so does the computational burden of the algorithm.

The resulting path plans are given in figures 4.7, 4.8 for the case with 500 samples and in figures 4.9 and 4.10 for the case with 1000 samples. It is important to note that since this is a random sampling based method, the path plans generated will rarely be the same. The green  $x$  marks in the figures represent the uniformly distributed random samples of the 2D configuration space.

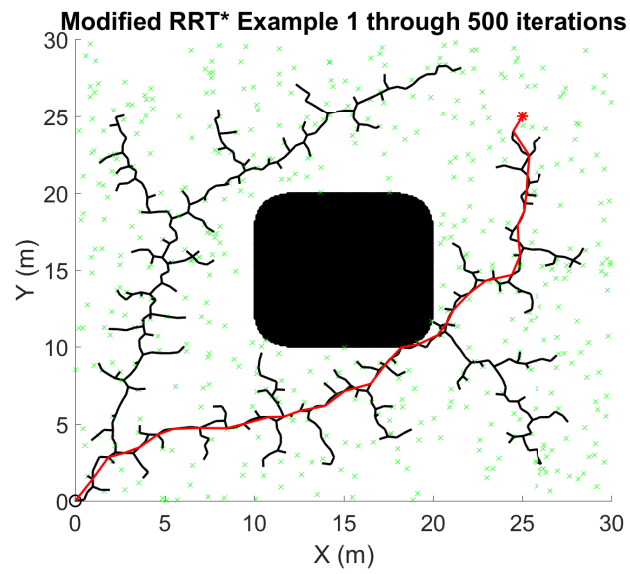


Figure 4.7: Exploring tree with generated path plan overlayed using 500 samples.

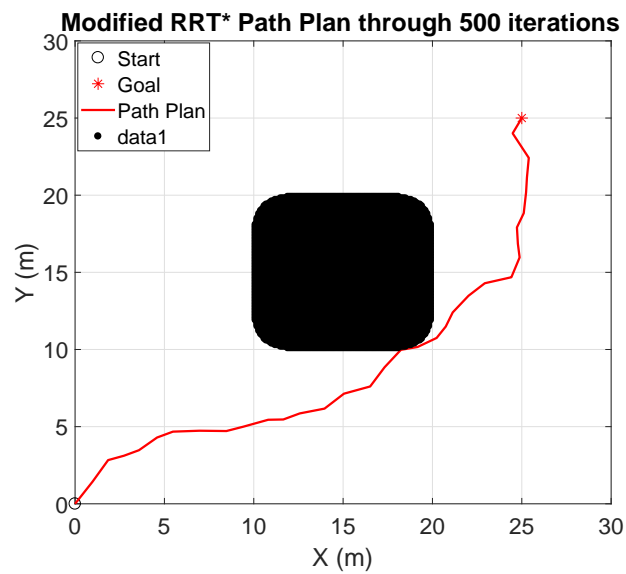


Figure 4.8: Generated path plan using modified-RRT\* with 500 samples.

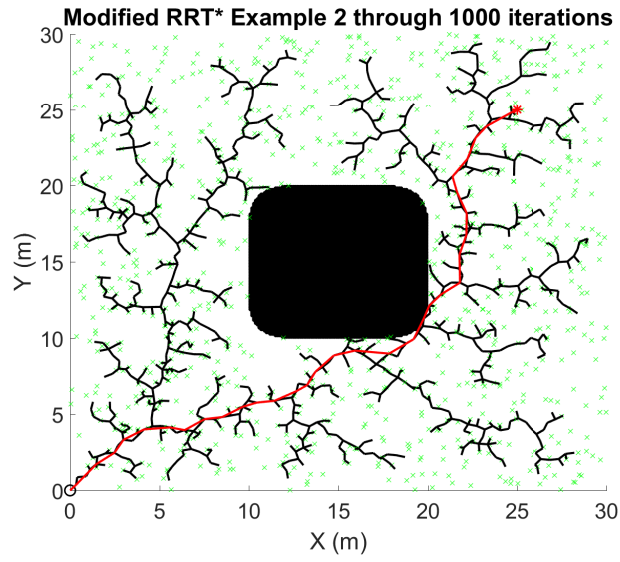


Figure 4.9: Exploring tree with generated path plan overlaid using 1000 samples.

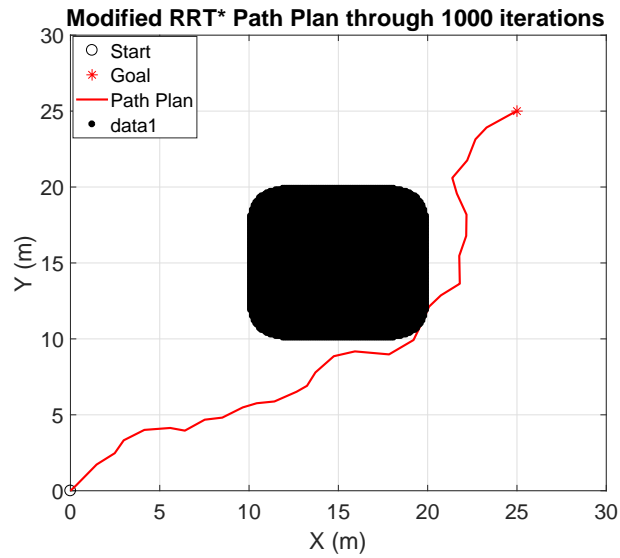


Figure 4.10: Generated path plan using modified-RRT\* with 1000 samples.

### 4.3.2 Modified D\* Path Plan Results

This section provides the simulated path plans generated using the modified D\* algorithm. For these results, the objective goal state is set as  $\mathbf{q}_{goal} = [50, 50, 0, 0.5]^T$ , the time step is  $\delta t = 10$  seconds. There are two sets of results presented. The first result has just the single obstacle in the middle of the environment. The result for the first case is shown in figure 4.11 and shows that the path plan is formed to the objective state. An important feature of D\*, evident in figure 4.11, is that the cost contours in the right-hand figure show that a cost value is not assigned to all the points in the environment. This result is indicative of the fact that the modified-D\* algorithm stops once it processed the starting node  $\mathbf{q}_{start}$  and then generated its path plan.

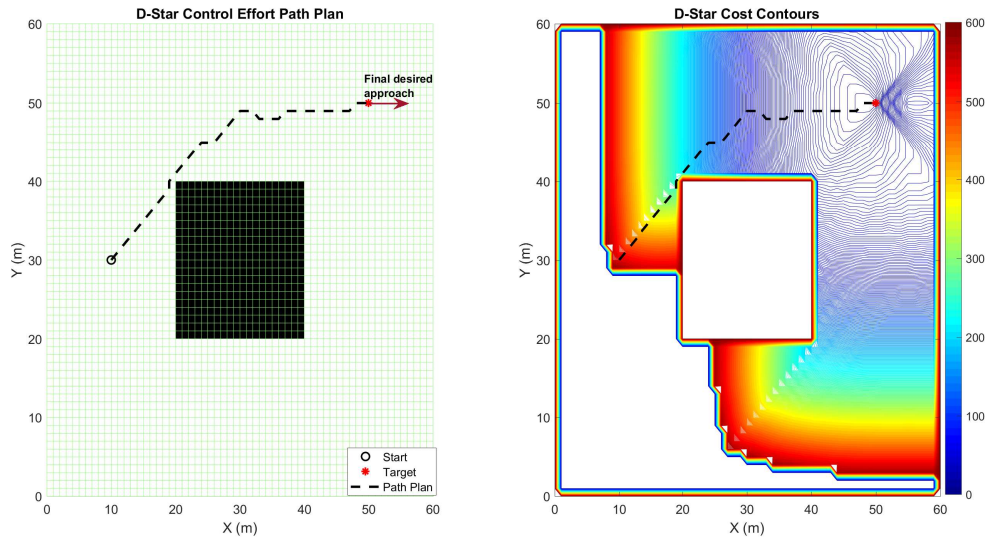


Figure 4.11: Modified-D\* path plan with no change in terrain detected.

The case presented in figure 4.11 is also representative of a situation before any change in the terrain is detected. However, say that a change in the terrain was

detected in the area where the original path plan was formed, portrayed in figure 4.12. And that the terrain was considered rough to traverse. Hence, this area is assigned an increased cost penalty. Then, the modify-cost mechanism of the D\* algorithm is implemented. This allows for the cost map in the affected area to be rewired and the path plan is regenerated.

This scenario is illustrated in figure 4.12 with a modified cost map and path plan. It is evident that area most affected by the new information on the cost map lie in the vicinity of the rough terrain, and the rest of the costs remain the same. This is demonstrative of how the D\* algorithm can quickly adapt to uncertain terrain types and apply newly sensed information in an optimal fashion.

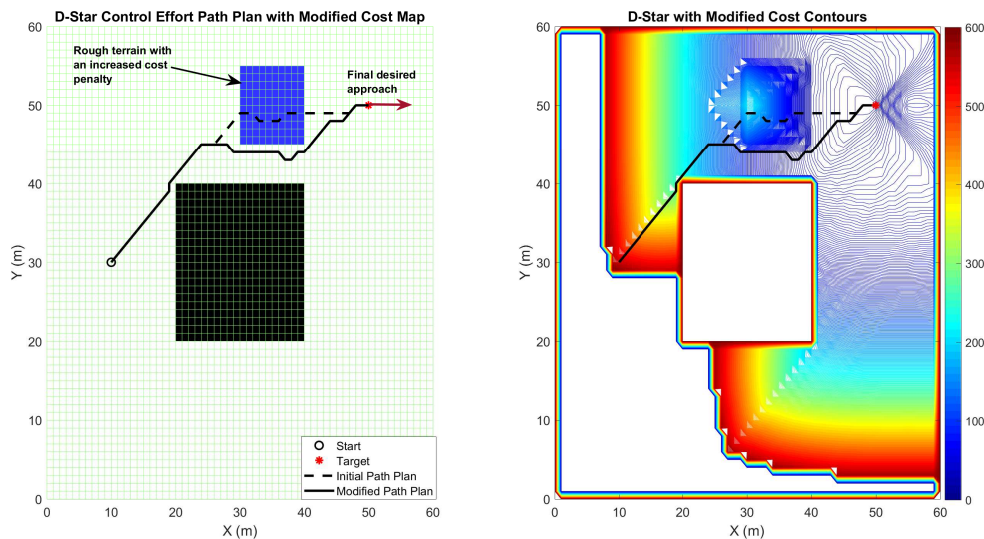


Figure 4.12: Modified-D\* path plans with a change in terrain traversability detected.

### 4.3.3 Comparing the Path Plan Algorithms

An example scenario is used to compare the different path planning algorithms presented in this dissertation. This scenario has multiple obstacles present in the

environment. The resulting path plans are illustrated in figure 4.13. The results were obtained using the modified wavefront expansion based navigation function (WFE) from chapter 3, and the modified RRT\* and D\* algorithms in this chapter. The main parameters set for each of the algorithms are summarized in table 4.1.

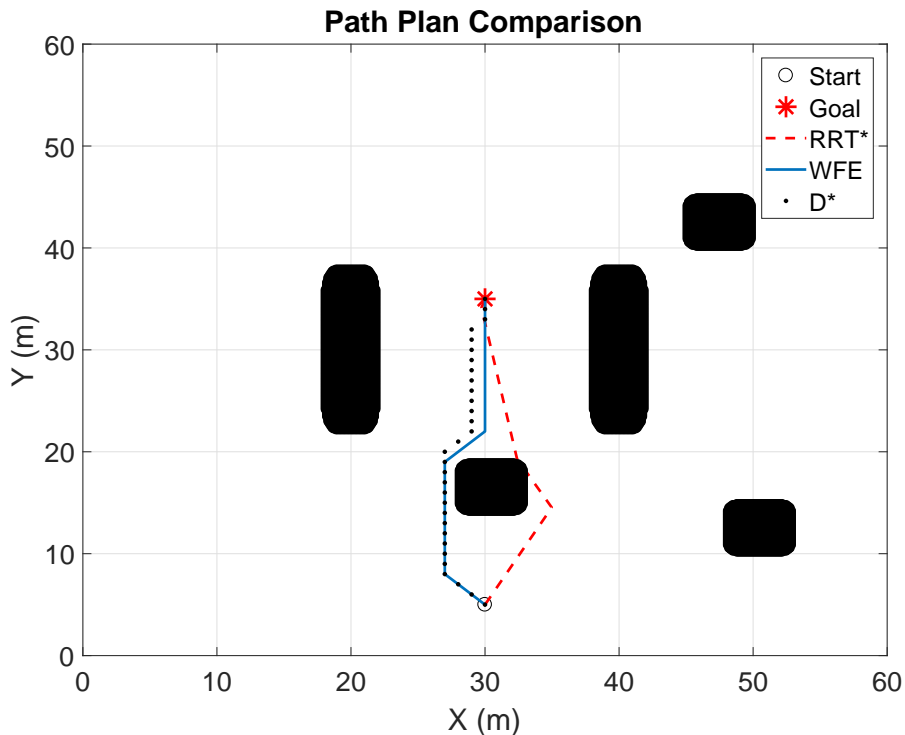


Figure 4.13: Comparative example with the different planning algorithms.

Table 4.1: Main parameters used with respective algorithms.

RRT*	2000 samples, $\Delta q = 1\text{m}$ , $\epsilon_{prox} = 20\text{m}$
WFE and D*	60x60 grid, $\delta t = 10 \text{ sec.}$ , $\mathbf{x}_{goal} = [30, 35, \pi/2, 0.5]^T$

From the results illustrated in figure 4.13, it is evident that each of the algorithms can form a direct path plan to the objective. The modified D\* and the

modified wavefront expansion based navigation function, denoted as WFE in the figure, produce similar results. The difference between the D\* and WFE path plans can be attributed to the methodology involved with their respective algorithms.

While the path plans are each able to reach the objective, it is important to note the difference in computation time required to generate these results. A summary of the computation time with each algorithm is given in table 4.2

Table 4.2: Computation time to generate path plan.

Algorithm	Time
Modified-WFE	6.76 sec.
Modified-D*	35.7 sec.
Modified-RRT*	30 min.

From table 4.2, it is shown that the navigation function path planning algorithm is the quickest. This is due to the fact that it has no ‘re-wiring’ steps to check for optimal traverses, since it is using the wavefront expansion (WFE) construction method. Also, note that the modified D\* and RRT\* algorithms take more time due to the approach used to determine the path costs. By using the minimum control energy approach with these algorithms to find the path costs, the computational burden associated with checking for optimal paths is increased. Moreover, the parameters used with the modified RRT\*, specifically setting  $\epsilon_{prox} = 20m$ , meant that a larger portion of the nodes in the exploring tree needed to be searched for ‘re-wiring’ which increases the computation time.

A subject for future research into these algorithms can be posed as an inquiry into the conditions where the resulting path plans are identical. However, the results put forth in this dissertation serve as an introduction to the algorithms with insight into their capabilities. Ultimately, the main differences demonstrated in this disserta-

tion can be attributed to the different nuances involved with generating the resulting path plans.



## CHAPTER 5

### State Dependent Coefficient Based Nonlinear Model Predictive Control

The primary guidance technique discussed in this dissertation uses a nonlinear model predictive control (NMPC) approach and is presented in this chapter. The procedure is motivated by the work discussed in [45] where the performance of the traditional approach of NMPC is compared with that of a state dependent coefficient (SDC) approach. The objective in this dissertation is to present this method for an autonomous mobile robot and to verify its trajectory tracking capability.

#### 5.1 State Dependent Coefficient Representation of the Vehicle Kinematics

Given a general nonlinear system of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

a state-space representation of the system is obtained wherein the system matrices are given as functions of the current state of the system as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})\mathbf{u} \\ \mathbf{y} &= \mathbf{C}(\mathbf{x})\mathbf{x} + \mathbf{D}(\mathbf{x})\mathbf{u}\end{aligned}\tag{5.1}$$

where  $\mathbf{x} \in \mathfrak{R}^n$ ,  $\mathbf{u} \in \mathfrak{R}^m$ ,  $\mathbf{y} \in \mathfrak{R}^{n_o}$  are respectively the state, input and output vectors and  $\mathbf{A}(\mathbf{x}) \in \mathfrak{R}^{n \times n}$ ,  $\mathbf{B}(\mathbf{x}) \in \mathfrak{R}^{n \times m}$ ,  $\mathbf{C}(\mathbf{x}) \in \mathfrak{R}^{n_o \times n}$ ,  $\mathbf{D}(\mathbf{x}) \in \mathfrak{R}^{n_o \times m}$  are the continuous state dependent system matrices in the SDC factored form. The pairs  $(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x}))$  and  $(\mathbf{A}(\mathbf{x}), \mathbf{C}(\mathbf{x}))$  are considered controllable and observable  $\forall \mathbf{x} \in \mathfrak{R}^n$  respectively.

In general, the formation of SDC matrices for the system are not unique unless observing a scalar system [30–33]. Therefore, different SDC forms may be obtained for a given system and solutions to the optimization problems posed may vary.

Two particular SDC factorizations of the kinematics are derived for the results in this chapter. The nonlinear kinematic equations for a mobile robot, also given in eq. (2.1), are

$$\begin{aligned} \dot{x} &= v \cos(\psi) \\ \dot{y} &= v \sin(\psi) \\ \dot{\psi} &= u_1 \\ \dot{v} &= u_2 \end{aligned} \tag{5.2}$$

For the results, it is assumed that the full state is available at each instant, i.e. the system output matrix  $\mathbf{C}(\mathbf{x}) = \mathbf{C}$  is considered as the constant identity matrix, i.e.  $\mathbf{I} \in \mathfrak{R}^{4 \times 4}$ . Also, from the kinematic equations, it is evident that  $\mathbf{D}(\mathbf{x}) = \mathbf{0}$  and the input matrix  $\mathbf{B}(\mathbf{x}) = \mathbf{B}$  is a constant matrix given by

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.3}$$

And one possible solution for the SDC factored  $\mathbf{A}(\mathbf{x})$  matrix can be given by

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{2 \times 2} & \mathbf{A}_{12}(\mathbf{x}) \\ \mathbf{0}_{2 \times 2} & \mathbf{0}_{2 \times 2} \end{bmatrix}$$

where

$$\mathbf{A}_{12}(\mathbf{x}) = \begin{bmatrix} v \left( \frac{\psi^3}{4!} - \frac{\psi}{2!} \right) & 1 \\ v \cos\left(\frac{\psi}{2}\right) \left( \frac{1}{2} - \frac{\psi^2}{2^3 3!} + \frac{\psi^4}{2^5 5!} \right) & \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\psi}{2}\right) \end{bmatrix} \tag{5.4}$$

This result is derived using the following trigonometric identity and expansions

$$\begin{aligned}\sin(2\psi) &= 2\sin(\psi)\cos(\psi) \\ \sin(\psi) &= \psi - \frac{\psi^3}{3!} + \frac{\psi^5}{5!} - \dots \\ \cos(\psi) &= 1 - \frac{\psi^2}{2!} + \frac{\psi^4}{4!} - \dots\end{aligned}$$

The above derivation of the  $\mathbf{A}_{12}$  quadrant in eq. (5.4) has some issues in implementation where the heading angle is limited from  $-\frac{\pi}{2} \leq \psi \leq \frac{\pi}{2}$ . Therefore, another formulation of the  $\mathbf{A}_{12}$  quadrant is derived as

$$\mathbf{A}_{12}(\mathbf{x}) = \begin{bmatrix} v \left( \frac{\psi^3}{4!} - \frac{\psi}{2!} \right) & 1 \\ v \left( -\frac{\psi^2}{3!} + \frac{\psi^4}{5!} \right) & \psi \end{bmatrix} \quad (5.5)$$

Note that with both SDC matrix results from eqs. (5.4) and (5.5), the psuedo-linear system  $\mathbf{A}(\mathbf{x})$  matrix becomes rank deficient when  $v = 0$ . This complication can be addressed by adding a state constraint on the velocity so that  $v > 0$  in the solution. Also, note that if  $\psi = 0$ , the resulting SDC matrix with  $\mathbf{A}_{12}$  coming from eq. (5.5) becomes uncontrollable. Therefore, in implementation, the  $\mathbf{A}(\mathbf{x})$  matrix is switched from using the  $\mathbf{A}_{12}$  quadrant from eq. (5.5) to using the formulation in eq. (5.4) whenever  $\|\psi\| \leq \psi_{tol}$ . Where  $\psi_{tol}$  is a tolerance value defined in the interval  $[-\pi/2, \pi/2]$ .

Next, a discrete-time equivalent of the system shown in eq. (5.1) can be obtained by introducing a zero-order-hold (ZOH) with a specified sample time ( $\Delta t$ ). With the ZOH, a sampled point is held constant over a sampled time interval. The smaller the sample time, the more accurate the approximation of the continuous signal. The discrete-time equivalent system is given in the following form

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{\Phi}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k\end{aligned} \quad (5.6)$$



## 5.2 Nonlinear Model Predictive Control Design

The guidance commands are obtained as the solution to the minimization of the finite-horizon linear quadratic tracking cost function with free-final state and is subject to both state and input constraints. The cost function is given as

$$J(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) = \sum_{j=0}^{N-1} [(\mathbf{x}_{k+j} - \mathbf{x}_{k+j}^r)^T \mathbf{Q}(\mathbf{x}_{k+j} - \mathbf{x}_{k+j}^r) + \mathbf{u}_{k+j}^T \mathbf{R} \mathbf{u}_{k+j}] + (\mathbf{x}_{k+N} - \mathbf{x}_{k+N}^r)^T \mathbf{Q}_f (\mathbf{x}_{k+N} - \mathbf{x}_{k+N}^r) \quad (5.9)$$

where  $\mathbf{x}_k^r \in \mathfrak{R}^n$  in general denotes the reference trajectory at the  $k^{\text{th}}$  time step. The above cost function can be placed into batch form for the SDC system as

$$J(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) = (\mathbf{X}_k - \mathbf{X}_k^r)^T \bar{\mathbf{Q}}_N (\mathbf{X}_k - \mathbf{X}_k^r) + \mathbf{U}_k^T \bar{\mathbf{R}}_N \mathbf{U}_k + (\mathbf{x}_{k+N} - \mathbf{x}_{k+N}^r)^T \mathbf{Q}_f (\mathbf{x}_{k+N} - \mathbf{x}_{k+N}^r)$$

where

$$\mathbf{X}_k^r = \begin{bmatrix} \mathbf{x}_k^r \\ \mathbf{x}_{k+1}^r \\ \vdots \\ \mathbf{x}_{k+N-1}^r \end{bmatrix}$$

is the batch form of the reference trajectory over the N-step horizon and  $\bar{\mathbf{Q}}_N = \text{diag}\{\mathbf{Q}, \dots, \mathbf{Q}\}$  and  $\bar{\mathbf{R}}_N = \text{diag}\{\mathbf{R}, \dots, \mathbf{R}\}$  are block diagonal matrices consisting of the state and input weighting matrices respectively.

After proper substitution, the objective function can be rewritten in a quadratic-like form as

$$\begin{aligned} J(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) &= \mathbf{U}_k^T \left( \mathbf{H}(\mathbf{x}_k)^T \bar{\mathbf{Q}}_N \mathbf{H}(\mathbf{x}_k) + \bar{\mathbf{R}}_N + \bar{\mathbf{\Gamma}}(\mathbf{x}_k)^T \mathbf{Q}_f \bar{\mathbf{\Gamma}}(\mathbf{x}_k) \right) \mathbf{U}_k \\ &+ 2 \left[ (\mathbf{F}(\mathbf{x}_k) \mathbf{x}_k - \mathbf{X}_k^r)^T \bar{\mathbf{Q}}_N \mathbf{H}(\mathbf{x}_k) + (\mathbf{\Phi}(\mathbf{x}_k)^N \mathbf{x}_k - \mathbf{x}_{k+N}^r)^T \mathbf{Q}_f \bar{\mathbf{\Gamma}}(\mathbf{x}_k) \right] \mathbf{U}_k \\ &+ (\mathbf{F}(\mathbf{x}_k) \mathbf{x}_k - \mathbf{X}_k^r)^T \bar{\mathbf{Q}}_N (\mathbf{F}(\mathbf{x}_k) \mathbf{x}_k - \mathbf{X}_k^r) \\ &+ (\mathbf{\Phi}(\mathbf{x}_k)^N \mathbf{x}_k - \mathbf{x}_{k+N}^r)^T \mathbf{Q}_f (\mathbf{\Phi}(\mathbf{x}_k)^N \mathbf{x}_k - \mathbf{x}_{k+N}^r) \end{aligned} \quad (5.10)$$

The objective function depends on the current state of the system and is calculated at the beginning of every sample interval. Also, the weight matrix for the final state,  $\mathbf{Q}_f$ , is obtained by solving the state-dependent discrete algebraic Riccati equation (SDDARE) at time instant  $k$ . The SDDARE is of the form

$$\mathbf{P}(\mathbf{x}_k) = \Phi(\mathbf{x}_k)^T \mathbf{P}(\mathbf{x}_k) \Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k)^T \mathbf{P}(\mathbf{x}_k) \Gamma (\mathbf{R} + \Gamma^T \mathbf{P}(\mathbf{x}_k) \Gamma)^{-1} \Gamma^T \mathbf{P}(\mathbf{x}_k) \Phi(\mathbf{x}_k) + \mathbf{Q} \quad (5.11)$$

where  $\mathbf{Q}_f$  is set equal to the solution  $\mathbf{P}(\mathbf{x}_k)$ . The solution to eq. (5.11) is found at each sample instant  $k$ .

### 5.3 Input and State Constraints

The constraints are used in this formulation to ensure that the tracking capabilities of the system are feasible. The nonlinear kinematic equations for the vehicle represented in eq. (5.2) enable constraints to be placed on the heading angle turn rate and acceleration of the vehicle without added complication. Overall, the form of the kinematics chosen enables constraints to be placed on the system's velocity, heading angle, its turn rate and its forward acceleration. In general, the input and state constraints are represented by the following inequalities

$$\begin{aligned} \mathbf{u}_{lb} &\leq \mathbf{u}_{k+j} \leq \mathbf{u}_{ub}, \quad j = 0, 1, \dots, N-1 \\ \mathbf{g}_{lb} &\leq \mathbf{G}\mathbf{x}_{k+j} \leq \mathbf{g}_{ub}, \quad j = 0, 1, \dots, N \end{aligned} \quad (5.12)$$

where the subscripts  $lb$  and  $ub$  denote the lower and upper bounds respectively of the constraints and the matrix  $\mathbf{G}$  is considered as an output matrix for the states that are constrained. The constraints can be placed in batch form as

$$\begin{bmatrix} \mathbf{u}_{lb} \\ \mathbf{u}_{lb} \\ \vdots \\ \mathbf{u}_{lb} \end{bmatrix} \leq \mathbf{U}_k \leq \begin{bmatrix} \mathbf{u}_{ub} \\ \mathbf{u}_{ub} \\ \vdots \\ \mathbf{u}_{ub} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{g}_{lb} \\ \mathbf{g}_{lb} \\ \vdots \\ \mathbf{g}_{lb} \end{bmatrix} \leq \bar{\mathbf{G}}_N (\mathbf{F}\mathbf{x}_k + \mathbf{H}\mathbf{U}_k) \leq \begin{bmatrix} \mathbf{g}_{ub} \\ \mathbf{g}_{ub} \\ \vdots \\ \mathbf{g}_{ub} \end{bmatrix} \quad (5.13)$$

where  $\bar{\mathbf{G}}_N = \text{diag}\{G, \dots, G\}$  is a block diagonal matrix formed by the output matrix  $\mathbf{G}$ . Then, the constraints of the system need to be incorporated into a single matrix equation of the form

$$\mathbf{M}(\mathbf{x}_k)\mathbf{U}_k \leq \Upsilon(\mathbf{x}_k) \quad (5.14)$$

where

$$\mathbf{M}(\mathbf{x}_k) = \begin{bmatrix} \mathbf{M}_U \\ \bar{\mathbf{G}}_N \mathbf{H}(\mathbf{x}_k) \end{bmatrix}, \quad \Upsilon(\mathbf{x}_k) = \begin{bmatrix} \mathbf{U}_b \\ \mathbf{g} - \bar{\mathbf{G}}_N \mathbf{F}(\mathbf{x}_k) \end{bmatrix}$$

and

$$\mathbf{M}_U = \begin{bmatrix} \begin{bmatrix} \mathbf{I}_{m \times m} \\ -\mathbf{I}_{m \times m} \end{bmatrix} \\ \begin{bmatrix} \mathbf{I}_{m \times m} \\ -\mathbf{I}_{m \times m} \end{bmatrix} \\ \dots \\ \begin{bmatrix} \mathbf{I}_{m \times m} \\ -\mathbf{I}_{m \times m} \end{bmatrix} \end{bmatrix}$$

where  $\mathbf{I}_{m \times m}$  is the  $m \times m$  identity matrix. Also,

$$\mathbf{U}_b = \begin{bmatrix} \begin{bmatrix} \mathbf{u}_{ub} \\ -\mathbf{u}_{lb} \end{bmatrix} \\ \begin{bmatrix} \mathbf{u}_{ub} \\ -\mathbf{u}_{lb} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{u}_{ub} \\ -\mathbf{u}_{lb} \end{bmatrix} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \begin{bmatrix} \mathbf{g}_{ub} \\ -\mathbf{g}_{lb} \end{bmatrix} \\ \begin{bmatrix} \mathbf{g}_{ub} \\ -\mathbf{g}_{lb} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{g}_{ub} \\ -\mathbf{g}_{lb} \end{bmatrix} \end{bmatrix}$$

The guidance commands are synthesized by minimizing the quadratic cost function in eq. (5.10), subject to the input and state constraints in eq. (5.14) using quadratic programming.

#### 5.4 Guidance Command Synthesis Using the Linear Matrix Inequality (LMI) Form

For readability with the following LMI formulation, the notation for state dependence is changed from using brackets to a subscript, in other words  $(\cdot)(\mathbf{x}_k)$  will be cast as  $(\cdot)_k$ . First, note that the quadratic-like cost function in eq. (5.10) can be decomposed into the following form

$$J(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) = J_1(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) + J_2(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k)$$

where

$$\begin{aligned} J_1(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) &= \mathbf{U}_k^T \mathbf{W}_k \mathbf{U}_k + \omega_k^T \mathbf{U}_k + [\mathbf{F}_k \mathbf{x}_k - \mathbf{X}_k^r]^T \bar{\mathbf{Q}}_{Nk} [\mathbf{F}_k \mathbf{x}_k - \mathbf{X}_k^r] \\ J_2(\mathbf{x}_k, \mathbf{x}_k^r, \mathbf{u}_k) &= (\Phi_k^N \mathbf{x}_k + \bar{\Gamma}_k \mathbf{U}_k - \mathbf{x}_{k+N}^r)^T \mathbf{Q}_{fk} (\Phi_k^N \mathbf{x}_k + \bar{\Gamma}_k \mathbf{U}_k - \mathbf{x}_{k+N}^r) \end{aligned}$$



and  $\mathbf{W}_k$  and  $\omega_k$  are defined as

$$\begin{aligned}\mathbf{W}_k &= \mathbf{H}_k^T \bar{\mathbf{Q}}_{Nk} \mathbf{H}_k + \bar{\mathbf{R}}_N \\ \omega_k &= 2\mathbf{H}_k^T \bar{\mathbf{Q}}_{Nk}^T [\mathbf{F}_k \mathbf{x}_k - \mathbf{X}_k^T]\end{aligned}$$

Then, for each  $\mathbf{x}_k$ , there must exist a set of  $(\mathbf{Q}_{fk}, \mathbf{Q}_k, \mathbf{R}_k, \mathbf{U}_k)$  that satisfy the following conditions

$$\begin{aligned}J_1(\mathbf{x}_k, \mathbf{x}_k^T, \mathbf{u}_k) &\leq \gamma_1 \\ J_2(\mathbf{x}_k, \mathbf{x}_k^T, \mathbf{u}_k) &\leq \gamma_2\end{aligned}\tag{5.15}$$

$$\begin{bmatrix} \gamma_1 - 2\mathbf{x}_k^T \mathbf{F}_k^T \bar{\mathbf{Q}}_{Nk} \mathbf{H}_k \mathbf{U}_k - \mathbf{x}_k^T \mathbf{F}_k^T \bar{\mathbf{Q}}_{Nk} \mathbf{F}_k \mathbf{x}_k & \mathbf{U}_k^T \\ \mathbf{U}_k & (\mathbf{H}_k^T \bar{\mathbf{Q}}_{Nk} \mathbf{H}_k + \bar{\mathbf{R}}_{Nk})^{-1} \end{bmatrix} \geq 0\tag{5.16}$$

$$\begin{bmatrix} \gamma_2 & [\Phi_k^N \mathbf{x}_k + \bar{\Gamma} \mathbf{U}_k]^T \\ \Phi_k^N \mathbf{x}_k + \bar{\Gamma} \mathbf{U}_k & \mathbf{Q}_{fk}^{-1} \end{bmatrix} \geq 0\tag{5.17}$$

$$\begin{bmatrix} \mathbf{Q}_{fk}^{-1} & (\Phi_k \mathbf{Q}_{fk} - \Gamma \mathbf{Y}_k)^T & (\sqrt{\mathbf{Q}_k} \mathbf{Q}_{fk}^{-1})^T & (\sqrt{\mathbf{R}_k} \mathbf{Y}_k)^T \\ \Phi_k \mathbf{Q}_{fk} - \Gamma \mathbf{Y}_k & \mathbf{Q}_{fk}^{-1} & \mathbf{0}_{n \times n} & \mathbf{0}_{n \times n} \\ \sqrt{\mathbf{Q}_k} \mathbf{Q}_{fk} & \mathbf{0}_{n \times n} & \mathbf{I}_{n \times n} & \mathbf{0}_{n \times n} \\ \sqrt{\mathbf{R}_k} \mathbf{Y}_k & \mathbf{0}_{n \times n} & \mathbf{0}_{n \times n} & \mathbf{I}_{n \times n} \end{bmatrix} \geq 0\tag{5.18}$$

$$\mathbf{M}(\mathbf{x}_k) \mathbf{U}_k \leq \Upsilon(\mathbf{x}_k)\tag{5.19}$$

where  $\mathbf{Y}_k = \mathbf{K}_k \mathbf{Q}_{fk}^{-1}$ , and  $\mathbf{I}_{n \times n}$ ,  $\mathbf{0}_{n \times n}$  denote the  $n \times n$  identity matrix and  $n \times n$  null matrix respectively. The set of LMIs given above are used to establish the feasibility of the control design and are important for proving stability. When a suitable set of  $(\mathbf{Q}_{fk}, \mathbf{Q}_k, \mathbf{R}_k)$  satisfying the above conditions are obtained, the input over the N-step horizon,  $\mathbf{U}_k^*$ , can then be found through the following quadratic programming problem:

$$\mathbf{U}_k^* = \min_{\mathbf{U}_k} \gamma_1 + \gamma_2$$

subject to the constraints in eqs. (5.16), (5.17), (5.18) and (5.19). And upon solving the quadratic programming problem,  $\mathbf{u}_k^*$  can be obtained by

$$\mathbf{u}_k^* = [\mathbf{I}_{m \times m} \ \mathbf{0}_{m \times m} \ \cdots \ \mathbf{0}_{m \times m}] \mathbf{U}_k^* \quad (5.20)$$

which will extract the guidance commands corresponding to the  $k^{th}$  time step to the system in eq. (5.2). The stability of the closed loop system using the SDC-based NMPC guidance law is given in the next section.

### 5.5 Stability of Constrained, Sampled-Data, SDC-based NMPC

Without loss of generality, the stability for a general regulator system is presented. In other words, the cost function in eq. (5.9) is rewritten in SDC form as

$$J(\mathbf{x}_k, \mathbf{u}_k) = \sum_{j=0}^{N-1} [\mathbf{x}_{k+j}^T \mathbf{Q}(\mathbf{x}_k) \mathbf{x}_{k+j} + \mathbf{u}_{k+j}^T \mathbf{R}(\mathbf{x}_k) \mathbf{u}_{k+j}] + \mathbf{x}_{k+N}^T \mathbf{Q}_f(\mathbf{x}_k) \mathbf{x}_{k+N} \quad (5.21)$$

or in batch form as

$$J(\mathbf{x}_k, \mathbf{U}_k) = \mathbf{X}_k^T \bar{\mathbf{Q}}_N(\mathbf{x}_k) \mathbf{X}_k + \mathbf{U}_k^T \bar{\mathbf{R}}_N(\mathbf{x}_k) \mathbf{U}_k + \mathbf{x}_{k+N}^T \mathbf{Q}_f(\mathbf{x}_k) \mathbf{x}_{k+N}$$

Then, in order to show stability for the constrained sampled-data NMPC formulation, the cost monotonicity relation must be satisfied and the system must be feasible at the start of each sample interval [29, 45]. The left hand side of the cost function can be written to include information of the current time step,  $k$ , and prediction horizon,  $N$ , as  $J(\mathbf{x}_k, \mathbf{U}_k, k, N)$ . Then, the optimal cost can be represented as  $J^*(\mathbf{x}_k, \mathbf{U}_k^*, k, N)$ . Then, the cost monotonicity relation for an optimal cost  $J^*(\mathbf{x}_k, \mathbf{U}_k^*, k, N)$  is given by

$$J^*(\mathbf{x}_\tau, \mathbf{U}_\tau^*, \tau, N + 1) \leq J^*(\mathbf{x}_\tau, \mathbf{U}_\tau^*, \tau, N) \quad (5.22)$$

for any  $\tau \leq N$  and where  $\mathbf{U}_\tau^*$  is the optimal control derived from the cost function. The theorem used to prove that a system satisfies the cost monotonicity relation in eq. (5.22) is given below and can be found in reference [29].

**Theorem 1.** *If the system is feasible, and if the matrix  $\mathbf{Q}_f(\mathbf{x}_k) \in \mathfrak{R}^{n \times n}$  in eq. (5.21) satisfies the following inequality*

$$\mathbf{Q}_f(\mathbf{x}_k) \geq \mathbf{Q}(\mathbf{x}_k) + \mathbf{K}(\mathbf{x}_k)^T \mathbf{R}(\mathbf{x}_k) \mathbf{K}(\mathbf{x}_k) + (\Phi(\mathbf{x}_k) - \Gamma \mathbf{K}(\mathbf{x}_k))^T \mathbf{Q}_f(\mathbf{x}_k) (\Phi(\mathbf{x}_k) - \Gamma \mathbf{K}(\mathbf{x}_k)) \quad (5.23)$$

for some matrix  $\mathbf{K}(\mathbf{x}_k) \in \mathfrak{R}^{m \times n}$ , then the optimal cost satisfies the cost monotonicity relation in eq. (5.22) [29].

Notice that by having  $\mathbf{Q}_f(\mathbf{x}_k)$  as the solution to the SDDARE in eq. (5.11), it can be shown that the inequality in equation (5.23) holds by letting

$$\mathbf{K}(\mathbf{x}_k) = (\mathbf{R}(\mathbf{x}_k) + \Gamma^T \mathbf{Q}_f(\mathbf{x}_k) \Gamma)^{-1} \Gamma^T \mathbf{Q}_f(\mathbf{x}_k) \Phi(\mathbf{x}_k)$$

Additionally, the linear matrix inequalities (LMI) given in eqs. (5.16)-(5.19) are formed to establish feasibility of the system. Note that eq. (5.18) is the LMI form for the cost monotonicity condition [29].

## 5.6 Simulation Results

For the results presented, the navigation function path plan results from chapter 3 are used. And the reference trajectory is generated along the path plan as described in section A.5.1 with a desired time to reach the goal set at  $t_{goal} = 180$  seconds (or 3 minutes). The sampling time used to discretize the SDC system is chosen as  $\Delta t = 0.1$  seconds and the prediction horizon length is chosen as  $N = 20$  time steps. Also, the input and state weighting matrices are chosen as

$$\mathbf{R} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \quad (5.24)$$

The input constraints chosen for the simulation are chosen as

$$\begin{aligned} -0.5 \text{ (rad/s)} &\leq \dot{\psi} \leq 0.5 \text{ (rad/s)} \\ -0.15 \text{ (m/s}^2\text{)} &\leq \dot{v} \leq 0.15 \text{ (m/s}^2\text{)} \end{aligned} \tag{5.25}$$

for the heading angle turn rate and forward acceleration respectively. The state, or output, constraints are placed on the heading angle and velocity respectively as

$$\begin{aligned} -\pi \text{ (rad)} &\leq \psi \leq \pi \text{ (rad)} \\ 0.001 \text{ (m/s)} &\leq v \leq 1 \text{ (m/s)} \end{aligned} \tag{5.26}$$

These constraints are chosen based on experiments conducted with the experimental platform in the Aerospace Systems Laboratory at the University of Texas at Arlington. However, the lower bounds of 0.001 m/s for the robot's velocity is chosen due to the lack of controllability with the SDC form of the kinematic equations (from eq. (5.1)) when the velocity is equal to zero.

The simulation results with the above mentioned information illustrate the constrained trajectory tracking control's ability to follow the designed trajectories leading to the objective in each case. Furthermore, the results demonstrate the performance of the guidance law design to act within the limitations set by the input and state constraints of the system.

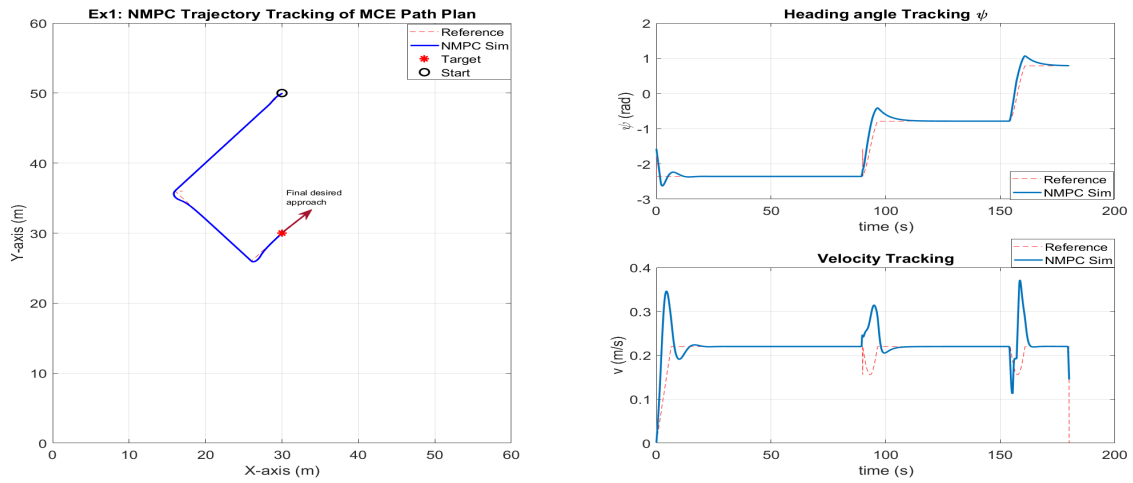


Figure 5.1: Example 1 constrained trajectory tracking with SDC based NMPC guidance commands and an MCE path plan.

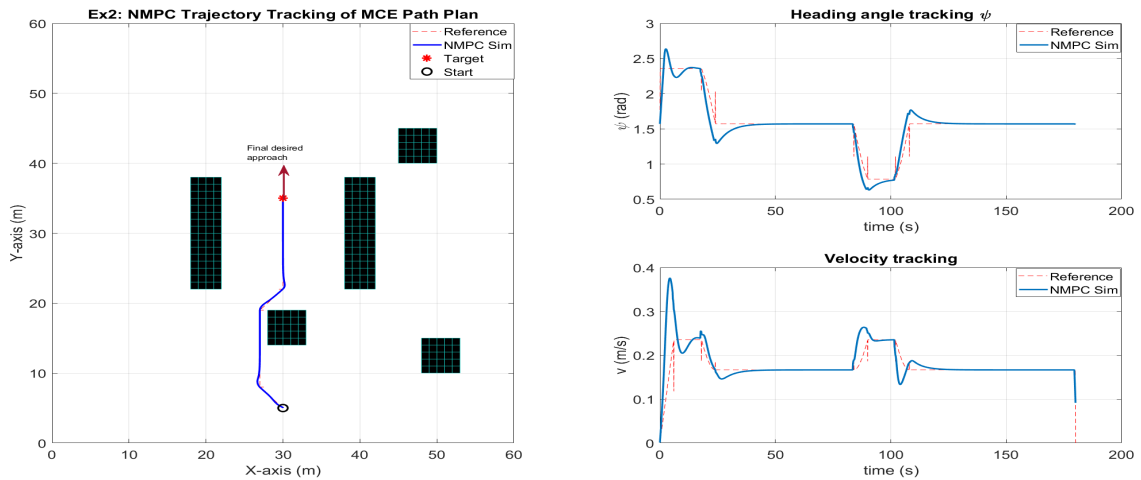


Figure 5.2: Example 2 constrained trajectory tracking with SDC based NMPC guidance commands and an MCE path plan with obstacles.

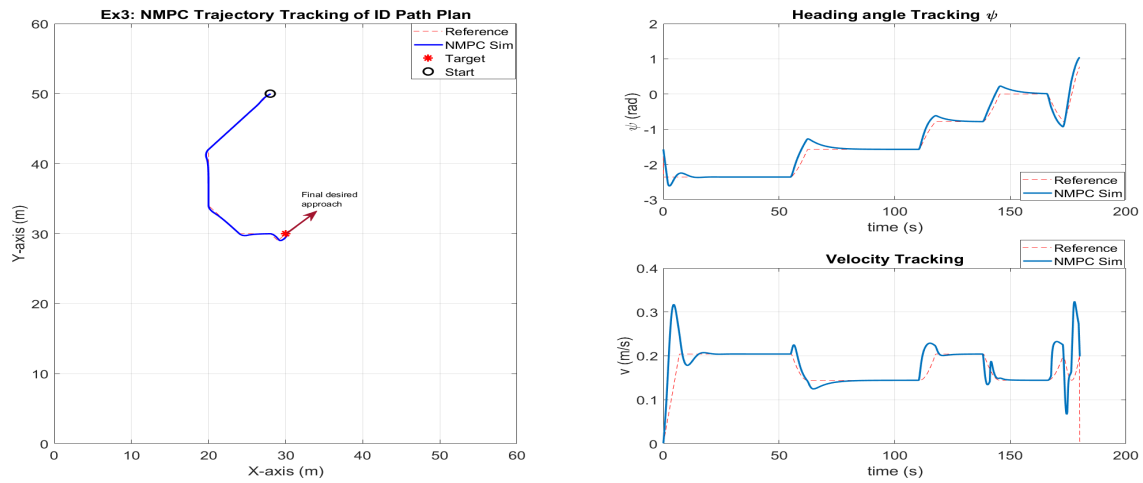


Figure 5.3: Example 3 constrained trajectory tracking with SDC based NMPC guidance commands and an ID path plan.

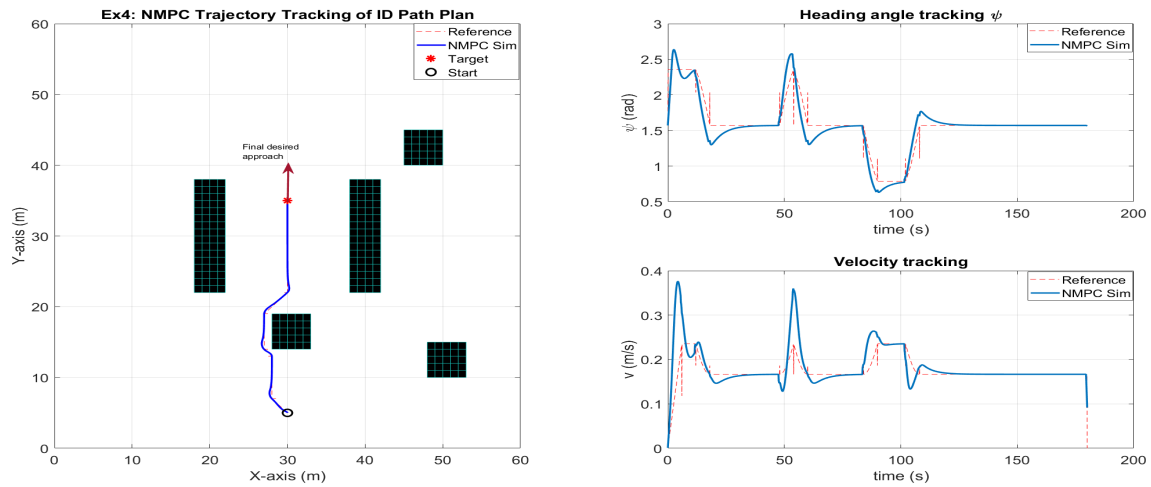


Figure 5.4: Example 4 constrained trajectory tracking with SDC based NMPC guidance commands and an ID path plan with obstacles.

The results show tracking of the path plan in the  $x$  and  $y$  position plots and also show the tracking of the velocity and heading angle reference signals during the simulation. It is evident that the spikes present in the tracking of the velocity

correspond to the sudden change of the heading angle as the vehicle moves along the trajectory. Also, in figure 5.3 it can be seen that the tracking performance is less accurate, this is due to the small sudden changes in the path plan as it shapes its approach to the objective and the controller is tracking subject to the input and state constraints.

Also, note that the spikes in the reference signals, evident in figures 5.2 and 5.4, result from the linear interpolation method used to generate the trajectory to track. Even though these spikes in the reference signals occur, they do not appear to affect the tracking performance of the SDC based nonlinear model predictive control design implemented on the mobile robot model.

## CHAPTER 6

### Nonlinear Guidance Law Design

Integrator backstepping is one of the nonlinear guidance design techniques chosen for a mobile robot to close the loop in the framework in combination with the navigation function path planner. The nonlinear kinematic model is described in section 2.2.1, and it is evident that the position elements of the state are directly influenced by the velocity and heading angle. Thus, the guidance commands derived in this chapter are designed to directly influence the velocity and heading angle with the overall objective of achieving stable trajectory tracking along a reference path generated by the navigation function algorithm.

This chapter will discuss the guidance law design and prove its stability using Lyapunov stability theory. The first section will detail the initial guidance law design, which influences the inverse dynamic approach used by the navigation function path planner, outlined in section 3.3. Then, the guidance law implemented within the framework, is derived. After the guidance law derivations, the stability of the resulting system is shown with extensions toward considering the presence of external disturbances. Finally, this chapter will conclude by presenting and discussing the simulation results. Experimental results on a mobile robot platform using this guidance design are summarized in chapter 8.



## 6.1 Guidance Law Design for Path Plan Algorithm

The kinematic equations for the vehicle considered in the path plan algorithm are given in section 2.2.1 in eq. (2.1) as

$$\dot{x} = v \cos(\psi)$$

$$\dot{y} = v \sin(\psi)$$

$$\dot{\psi} = u_1$$

$$\dot{v} = u_2$$

The guidance commands  $u_1$  and  $u_2$  are for the turn rate and forward acceleration of the robot. The guidance commands that result from eq. (2.1) are considered only for the path planning algorithm and their stability is shown to prove their feasibility.

**Proposition 1.** *Given a 2-D,  $C^2$  trajectory with references values,  $[x_r, y_r, \dot{x}_r, \dot{y}_r, \ddot{x}_r, \ddot{y}_r]$ . Then, the guidance commands in equations (6.1), and (6.2) along with the kinematics defined in equation (2.1) will guarantee that  $\|x - x_r\| \rightarrow 0$ , and  $\|y - y_r\| \rightarrow 0$  as  $t \rightarrow \infty$*

$$u_1 = \dot{\psi}_d - \lambda_\psi e_\psi \tag{6.1}$$

$$u_2 = \dot{v}_d - \lambda_v e_v \tag{6.2}$$

where  $\lambda_\psi > 0$ ,  $\lambda_v > 0$ , and where  $\psi_d$ ,  $v_d$ ,  $\dot{\psi}_d$  and  $\dot{v}_d$  are nonlinear functions of the reference trajectory, the state and the state tracking errors; also,  $e_\psi = \psi - \psi_d$  and  $e_v = v - v_d$ .

The guidance commands are formulated to guarantee stability, in the sense of Lyapunov, with a backstepping-like approach. Note, the position tracking errors are given by  $e_x = x - x_r$ ,  $e_y = y - y_r$  and their time derivatives are  $\dot{e}_x = \dot{x} - \dot{x}_r$  and  $\dot{e}_y = \dot{y} - \dot{y}_r$ . The desired heading and velocity signals,  $\psi_d$  and  $v_d$  respectively, are

prescribed such that the position tracking error dynamics decay exponentially, i.e.  $\dot{e}_x = -\lambda_x e_x$  and  $\dot{e}_y = -\lambda_y e_y$  for  $\lambda_x > 0$ , and  $\lambda_y > 0$ . It follows that

$$\begin{aligned} v_d \cos(\psi_d) &= \dot{x}_r - \lambda_x e_x \\ v_d \sin(\psi_d) &= \dot{y}_r - \lambda_y e_y \end{aligned} \quad (6.3)$$

Then, using equation (6.3), the desired heading and velocity signals are obtained as

$$\psi_d = \tan^{-1} \left( \frac{y_r - \lambda_y e_y}{x_r - \lambda_x e_x} \right) \quad (6.4)$$

$$v_d = \sqrt{(\dot{x}_r - \lambda_x e_x)^2 + (\dot{y}_r - \lambda_y e_y)^2} \quad (6.5)$$

The guidance commands are derived such that the heading and velocity tracking error dynamics,  $e_\psi = \psi - \psi_d$  and  $e_v = v - v_d$ , also decay exponentially. Thus, the error dynamics are prescribed as  $\dot{e}_\psi = -\lambda_\psi e_\psi$  and  $\dot{e}_v = -\lambda_v e_v$ . The time derivatives of these state tracking errors are given by  $\dot{e}_\psi = \dot{\psi} - \dot{\psi}_d$  and  $\dot{e}_v = \dot{v} - \dot{v}_d$ . Combining equation (2.1) with the state tracking error time derivatives leads to

$$\begin{aligned} u_1 &= \dot{\psi}_d - \lambda_\psi e_\psi \\ u_2 &= \dot{v}_d - \lambda_v e_v \end{aligned}$$

which are the control terms introduced in equations (6.1) and (6.2).

The time derivatives for the desired heading angle and velocity values are obtained by differentiating eqs. (6.4) and (6.5) with respect to time, leading to

$$\dot{\psi}_d = \frac{1}{v_d} [\cos(\psi_d) (\ddot{y}_r - \lambda_y \dot{e}_y) - \sin(\psi_d) (\ddot{x}_r - \lambda_x \dot{e}_x)] \quad (6.6)$$

$$\dot{v}_d = \cos(\psi_d) (\ddot{x}_r - \lambda_x \dot{e}_x) + \sin(\psi_d) (\ddot{y}_r - \lambda_y \dot{e}_y) \quad (6.7)$$

where  $\dot{e}_x = v \cos(\psi) - \dot{x}_r$  and  $\dot{e}_y = v \sin(\psi) - \dot{y}_r$ .

Note that when  $v_d = 0$ ,  $\dot{\psi}_d$  is not defined. Therefore, a singularity avoidance scheme must be in place to handle this situation in practice. However, since the

kinematics in eq. (2.1) are only implemented in the path planner, the singularity avoidance is disregarded for now.

The closed loop stability of the system is verified through a Lyapunov stability analysis. For a function to be considered a valid Lyapunov function it must be positive definite, its derivative must be negative definite and both the function and its derivative must be equal to zero at the equilibrium points (i.e. the origin) [67].

With that in mind, a candidate Lyapunov function is chosen as

$$V_1 = \frac{1}{2} (e_x^2 + e_y^2 + e_\psi^2 + e_v^2) \quad (6.8)$$

which is positive definite and is only equal to zero at the equilibrium. The time derivative of  $V_1$  is

$$\dot{V}_1 = e_x \dot{e}_x + e_y \dot{e}_y + e_\psi \dot{e}_\psi + e_v \dot{e}_v \quad (6.9)$$

The stability of the system will be shown without considering a singularity avoidance algorithm for the kinematics given in equation (2.1).

*Proof. (No singularity,  $v_d \neq 0$ )*

Using the guidance commands given in equations (6.1) and (6.2), it can be shown that equation (6.9) simplifies to

$$\dot{V}_1 = -\lambda_x e_x^2 - \lambda_y e_y^2 - \lambda_\psi e_\psi^2 - \lambda_v e_v^2 \quad (6.10)$$

which is negative definite and only equal to zero at the origin. Therefore, equation (6.8) is a valid Lyapunov function and the system given in equation 2.1 with the guidance commands in equations (6.1) and (6.2) is asymptotically stable [67].  $\square$

## 6.2 Framework Guidance Law Design

To implement the guidance law on the robot, the input terms ( $u_1$  and  $u_2$ ) in equations (6.1) and (6.2) may also be rewritten as

$$u_1 = \alpha_1 (\psi^c - \psi) \quad (6.11)$$

$$u_2 = \alpha_2 (v^c - v) \quad (6.12)$$

with  $\alpha_1 > 0$ ,  $\alpha_2 > 0$  and where  $\psi^c$  and  $v^c$  are guidance command values. Thus,  $\psi^c$  and  $v^c$  are obtained as

$$\psi^c = \psi + \frac{u_1}{\alpha_1} \quad (6.13)$$

$$v^c = v + \frac{u_2}{\alpha_2} \quad (6.14)$$

with the values for  $u_1$  and  $u_2$  being computed from equations (6.1) and (6.2). The kinematic equations of the robot are thus recast as,

$$\dot{x} = v \cos(\psi)$$

$$\dot{y} = v \sin(\psi)$$

$$\dot{\psi} = \alpha_1 (\psi^c - \psi)$$

$$\dot{v} = \alpha_2 (v^c - v)$$

as mentioned in section 2.2.1, eq. (2.2). The above along with the guidance commands  $\psi^c$ , and  $v^c$  are used by the robot to track the reference trajectory.

As previously mentioned, there is a singularity present in equation (6.6), for  $\dot{\psi}_d$ , when  $v_d = 0$ , which needs to be addressed to implement the guidance commands given in equations (6.13) and (6.14). In order to avoid this situation, the following singularity avoidance algorithm is implemented. First, take  $\dot{\psi}_d = 0$  when  $v_d \leq \epsilon$ , for some  $\epsilon \ll 1$ . Also, assume that  $\psi_d$  is held constant as a reference value computed based on the reference position values the vehicle is traveling in-between (call it  $\psi_r$

which can be obtained from the path plan or waypoints). Formally, the following algorithm is used to avoid the singularities and keep the vehicle on the path.

**if**  $v_d \leq \epsilon$  **then**

$$\dot{\psi}_d = 0;$$

$$\psi_r = \tan^{-1} \left( \frac{\Delta y}{\Delta x} \right);$$

$$\psi_d = \psi_r$$

**end if**

The values of  $\Delta y$  and  $\Delta x$  are computed as a cell difference from the reference position values as

$$\Delta y = y_r(i+1) - y_r(i)$$

$$\Delta x = x_r(i+1) - x_r(i), \forall i = 1, \dots, n-1$$

where  $n$  is the number of position points in the reference trajectory and the ordered pair  $(x_r(i), y_r(i))$  denotes the  $(x, y)$  location of the  $i^{\text{th}}$  point of the reference path.

**Proposition 2.** *Given a 2-D,  $\mathcal{C}^2$  trajectory with reference values,  $[x_r, y_r, \dot{x}_r, \dot{y}_r, \ddot{x}_r, \ddot{y}_r]$ .*

*Then, the guidance commands in equations (6.13) and (6.14) along with the kinematics defined in equation (2.2) will guarantee that  $\|x - x_r\| \rightarrow 0$ , and  $\|y - y_r\| \rightarrow 0$  and are otherwise bounded during the singularity avoidance phase.*

To prove the closed loop stability of the system with the guidance commands in equations (6.13) and (6.14) and the kinematics given in equation (2.2), we begin with the same candidate Lyapunov function as before,

$$V_2 = \frac{1}{2} (e_x^2 + e_y^2 + e_\psi^2 + e_v^2) \quad (6.15)$$

which is positive definite and is only equal to zero at the origin. And its time derivative is

$$\dot{V}_2 = e_x \dot{e}_x + e_y \dot{e}_y + e_\psi \dot{e}_\psi + e_v \dot{e}_v \quad (6.16)$$

Two proofs will be given, one will show asymptotic stability of the system without the singularity avoidance, and the second will demonstrate the boundedness of the tracking errors during the singularity avoidance phase.

*Proof.* (No singularity,  $v_d \neq 0$ )

For the case when  $v_d \neq 0$ , the time derivative of  $V_2$  can be shown to be

$$\dot{V}_2 = -\lambda_x e_x^2 - \lambda_y e_y^2 - \lambda_\psi e_\psi^2 - \lambda_v e_v^2, \quad (6.17)$$

which is negative definite and only equal to zero at the origin. Therefore, equation (6.15) is a valid Lyapunov function. Furthermore, the system given in equation (2.2) with the guidance commands in equations (6.13) and (6.14) is asymptotically stable and the errors will decay to the origin [67].  $\square$

As soon as  $v_d \leq \epsilon$  for some  $\epsilon \ll 1$ , the singularity avoidance algorithm is implemented. With this algorithm, the state of the system is considered with  $\psi_d = \psi_r$  and the desired velocity  $v_d \leq \epsilon$ .

*Proof.* (Singularity avoidance,  $v_d \leq \epsilon$  for some  $\epsilon \ll 1$ )

With the candidate Lyapunov function defined in equation (6.15), its time derivative can be evaluated along with the guidance commands from equations (6.13) and (6.14). First, consider the  $e_\psi \dot{e}_\psi$  term in eq. (6.16) which simplifies as

$$\begin{aligned} e_\psi \dot{e}_\psi &= e_\psi \left( \dot{\psi} - \dot{\psi}_d \right) = e_\psi \left( \alpha_1 (\psi^c - \psi) - 0 \right) \\ &= e_\psi \left[ \alpha_1 \left( \psi + \frac{1}{\alpha_1} (0 - \lambda_\psi e_\psi) - \psi \right) \right] \\ &= -\lambda_\psi e_\psi^2 \end{aligned} \quad (6.18)$$

since  $\dot{\psi}_d = 0$  and  $e_\psi = \psi - \psi_d$ , with  $\psi_d$  being equal to the reference value  $\psi_r$ . Similarly,

$$\begin{aligned}
e_v \dot{e}_v &= e_v (\dot{v} - \dot{v}_d) = e_v [\alpha_2 (v^c - v) - \dot{v}_d] \\
&= e_v \left[ \alpha_2 \left( v + \frac{1}{\alpha_2} (\dot{v}_d - \lambda_v e_v) - v \right) - \dot{v}_d \right] \\
&= -\lambda_v e_v^2
\end{aligned} \tag{6.19}$$

And

$$\dot{v}_d = \cos(\psi_r) (\ddot{x}_r - \lambda_x \dot{e}_x) + \sin(\psi_r) (\ddot{y}_r - \lambda_y \dot{e}_y)$$

by taking the derivative of equation (6.5) and considering  $\psi \rightarrow \psi_r$ .

To show that the position errors are bounded during the singularity avoidance phase, the following relationships are established. Recall,

$$v_d = \sqrt{(\dot{x}_r - \lambda_x e_x)^2 + (\dot{y}_r - \lambda_y e_y)^2}$$

with  $v_d \leq \epsilon$ . Also, define  $\epsilon_x$  and  $\epsilon_y$  as

$$\epsilon_x = \dot{x}_r - \lambda_x e_x$$

$$\epsilon_y = \dot{y}_r - \lambda_y e_y$$

Then, from the definition of  $v_d$ , we note  $\|\epsilon_x\| \leq v_d$  and  $\|\epsilon_y\| \leq v_d$ .

Consider the following,

$$V_x = \frac{1}{2} e_x^2$$

whose time derivative is given by

$$\dot{V}_x = e_x \dot{e}_x = e_x (v \cos(\psi) - \dot{x}_r) \tag{6.20}$$

Equation (6.20) can be rewritten as

$$\begin{aligned}
\dot{V}_x &= e_x (v \cos(\psi) - \lambda_x e_x + \lambda_x e_x - \dot{x}_r) \\
&= e_x (v_d \cos(\psi_r) - \lambda_x e_x) + e_x (\lambda_x e_x - \dot{x}_r)
\end{aligned} \tag{6.21}$$

for  $v \rightarrow v_d$  and  $\psi \rightarrow \psi_r$ . With  $v_d = \epsilon$ , equation (6.21) becomes

$$\begin{aligned}\dot{V}_x &= -\lambda_x e_x^2 + e_x (\epsilon \cos(\psi_r) - (\dot{x}_r - \lambda_x e_x)) \\ &= -\lambda_x e_x^2 + e_x (\epsilon \cos(\psi_r) - \epsilon_x)\end{aligned}\quad (6.22)$$

Then, by taking the norm of the right hand side of equation (6.22), we get

$$\begin{aligned}\dot{V}_x &\leq -\lambda_x \|e_x\|^2 + \|e_x\| (\epsilon + \|\epsilon_x\|) \\ &\leq -\lambda_x \|e_x\|^2 + \|e_x\| (\epsilon + v_d) \\ &\leq -\lambda_x \|e_x\|^2 + \|e_x\| (2\epsilon)\end{aligned}\quad (6.23)$$

since  $\|\epsilon_x\| \leq v_d \leq \epsilon$ , and  $\|\epsilon \cos(\psi_r)\| \leq \epsilon$ . Finally, equation (6.23) can be further simplified as

$$\dot{V}_x \leq -(\lambda_x - 1) \|e_x\|^2 + \epsilon^2 \quad (6.24)$$

since  $2\|e_x\|\epsilon \leq \|e_x\|^2 + \epsilon^2$ .

In a similar fashion, the  $y$  position error can be shown to be bounded during the singularity avoidance phase by considering the term,

$$V_y = \frac{1}{2} e_y^2$$

And by following a similar approach as outlined for the  $x$  position error, it can be shown that

$$\dot{V}_y \leq -(\lambda_y - 1) \|e_y\|^2 + \epsilon^2 \quad (6.25)$$

Therefore, the time derivative of the candidate Lyapunov function in eq. (6.15) can be reformed as

$$\dot{V}_2 \leq -(\lambda_x - 1) \|e_x\|^2 - (\lambda_y - 1) \|e_y\|^2 - \lambda_\psi e_\psi^2 - \lambda_v e_v^2 + 2\epsilon^2 \quad (6.26)$$

Note, the above can be re-written in the following form

$$\dot{V}_2 \leq -\gamma V_2 + 2\epsilon^2 \quad (6.27)$$



where,  $\gamma = 2\mu(\mathbf{Q})$ ,  $\mu(\cdot)$  is the spectral radius (maximum eigenvalue) operator, and  $\mathbf{Q} = \text{diag}[(\lambda_x - 1), (\lambda_y - 1), \lambda_\psi, \lambda_v]$ . Clearly, we have  $\dot{V}_2 \leq 0$  whenever  $V_2 \geq V_{20} \triangleq \frac{1}{(\gamma/2)}\epsilon^2 \triangleq \frac{1}{\mu(\mathbf{Q})}\epsilon^2$ ; thus implying boundedness of all the errors and the terms in the guidance laws during the singularity avoidance phase.  $\square$

### 6.3 Guidance Design with Disturbance in the Acceleration

During a typical maneuver, it is safe to assume that a mobile robot will encounter diverse terrain types and varying levels of landscape topology. Therefore, it is practical to consider how this may have an impact on the vehicle's ability to track the reference trajectories with the given guidance commands.

The main consideration in this section pertains to the stability of the guidance law in the presence of external disturbances applied to the forward acceleration of the robot. The disturbance in the acceleration may be considered in scenarios when the robot is traveling up-hill or down-hill during a maneuver. Or it may be considered when the vehicle encounters terrain types that are less than ideal, which may cause added friction or slippage.

With this in mind, the kinematic equations for a mobile robot with external disturbance in the acceleration is given by

$$\begin{aligned}
 \dot{x} &= v \cos(\psi) \\
 \dot{y} &= v \sin(\psi) \\
 \dot{\psi} &= \alpha_1(\psi^c - \psi) \\
 \dot{v} &= \alpha_2(\Delta u - v) + d(t)
 \end{aligned} \tag{6.28}$$

where  $\alpha_1 > 0$ ,  $\alpha_2 > 0$ , and  $\psi^c$  is the guidance command given in eq. (6.13). The quantity  $\Delta u$  is a guidance input composed of the commanded velocity,  $v^c$  from eq. (6.14),

and an additional guidance input  $u_d$ ; hence,  $\Delta u = v^c + u_d$ . Also, the disturbance in eq. (6.28), denoted as  $d(t)$ , is assumed to be bounded by  $d_{max}$ , i.e.  $\|d(t)\| \leq \|d_{max}\|$ .

**Proposition 3.** *Given a 2-D,  $C^2$  trajectory with references values,  $[x_r, y_r, \dot{x}_r, \dot{y}_r, \ddot{x}_r, \ddot{y}_r]$ . Then, the guidance commands in equations (6.13), (6.14) and (6.29) along with the vehicle kinematics given in equation (6.28) will guarantee that the tracking errors are bounded.*

Recall, the guidance commands  $\psi^c$  and  $v^c$  are given by

$$\begin{aligned}\psi^c &= \psi + \frac{u_1}{\alpha_1} \\ v^c &= v + \frac{u_2}{\alpha_2}\end{aligned}$$

And the additional guidance input,  $u_d$ , is defined as

$$u_d = -\kappa_v e_v \tag{6.29}$$

where  $\kappa_v > 0$ . There will be two proofs given. One where there is no singularity in the equation for  $\dot{\psi}_d$ , i.e. when  $v_d \neq 0$ . The second will be considered during a singularity avoidance phase in the maneuver, i.e. when  $v_d \leq \epsilon$  for some  $\epsilon \ll 1$ , similar to the proof given in section 6.2.

To prove the closed loop stability of the system with external disturbance in the acceleration, a candidate Lyapunov function is chosen as

$$V_3 = \frac{1}{2} (e_x^2 + e_y^2 + e_\psi^2 + e_v^2) \tag{6.30}$$

whose time derivative is

$$\dot{V}_3 = e_x \dot{e}_x + e_y \dot{e}_y + e_\psi \dot{e}_\psi + e_v \dot{e}_v \tag{6.31}$$

*Proof.* ( No Singularity,  $v_d \neq 0$  )

The components of the time derivative of the candidate Lyapunov function in eq. (6.31) are evaluated with the guidance commands in equations (6.13), (6.14) and (6.29). First, the  $e_\psi \dot{e}_\psi$  term simplifies as

$$\begin{aligned}
e_\psi \dot{e}_\psi &= e_\psi (\dot{\psi} - \dot{\psi}_d) = e_\psi \left[ \alpha_1 (\psi^c - \psi) - \dot{\psi}_d \right] \\
&= e_\psi \left[ \alpha_1 \left( \psi + \frac{1}{\alpha_1} (\dot{\psi}_d - \lambda_\psi e_\psi) - \psi \right) - \dot{\psi}_d \right] \\
&= -\lambda_\psi e_\psi^2
\end{aligned} \tag{6.32}$$

Then, to consider the boundedness of the velocity tracking error, consider the following

$$V_v = \frac{1}{2} e_v^2$$

whose time derivative is

$$\dot{V}_v = e_v \dot{e}_v = e_v (\dot{v} - \dot{v}_d) \tag{6.33}$$

Equation (6.33) can be simplified as

$$\begin{aligned}
\dot{V}_v &= e_v [\alpha_2 (v^c - v) + d(t) + \alpha_2 u_d - \dot{v}_d] \\
&= e_v \left[ \alpha_2 \left( v + \frac{1}{\alpha_2} (\dot{v}_d - \lambda_v e_v) - v \right) + d(t) - \alpha_2 \kappa_v e_v - \dot{v}_d \right] \\
&= e_v (-\lambda_v e_v + d(t) - \alpha_2 \kappa_e e_v)
\end{aligned} \tag{6.34}$$

By defining  $\kappa = \lambda_v + \alpha_2 \kappa_v$ , eq. (6.34) simplifies to

$$\dot{V}_v = -\kappa e_v^2 + d(t) e_v \tag{6.35}$$

Taking the norm of the right hand side of eq. (6.35), the following is established

$$\begin{aligned}
\dot{V}_v &\leq -\kappa \|e_v\|^2 + \|d(t)\| \|e_v\| \\
&\leq -\kappa \|e_v\|^2 + \|d_{max}\| \|e_v\| \\
&\leq -\left( \kappa - \frac{1}{2} \right) \|e_v\|^2 + \frac{1}{2} \|d_{max}\|^2
\end{aligned} \tag{6.36}$$

since  $\|d(t)\| \leq \|d_{max}\|$ .

Next, considering when  $\psi \rightarrow \psi_d$  and  $v \rightarrow v_d$ , the position error terms in eq. (6.30) can be shown to be

$$\begin{aligned} e_x \dot{e}_x &= -\lambda_x e_x^2 \\ e_y \dot{e}_y &= -\lambda_y e_y^2 \end{aligned}$$

Thus, the time derivative of the candidate Lyapunov function in eq. (6.30) is found to be

$$\dot{V}_3 \leq -\lambda_x e_x^2 - \lambda_y e_y^2 - \lambda_\psi e_\psi^2 - \left(\kappa - \frac{1}{2}\right) \|e_v\|^2 + \frac{1}{2} \|d_{max}\|^2 \quad (6.37)$$

The above can be rewritten as

$$\dot{V}_3 \leq -\gamma V_3 + \frac{1}{2} \|d_{max}\|^2 \quad (6.38)$$

where,  $\gamma = 2\mu(\mathbf{Q})$ ,  $\mu(\cdot)$  is the spectral radius (maximum eigenvalue) operator, and  $\mathbf{Q} = \text{diag}[\lambda_x, \lambda_y, \lambda_\psi, (\kappa - \frac{1}{2})]$ . Clearly, we have  $\dot{V}_3 \leq 0$  whenever  $V_3 \geq V_{30} \triangleq \frac{1}{2\gamma} \|d_{max}\|^2 \triangleq \frac{1}{4\mu(\mathbf{Q})} \|d_{max}\|^2$ , thus implying boundedness of all the errors and the terms in the guidance laws. Furthermore, the effects of the disturbance on the tracking errors are limited by  $d_{max}$  and can be lessened by choosing sufficiently large control gains.  $\square$

The singularity avoidance implications in section 6.2 are also considered when the external disturbance is present. As mentioned in the previous section, when  $v_d \leq \epsilon$  for some  $\epsilon \ll 1$ , the singularity avoidance algorithm is implemented. The state of the system is considered with  $\psi_d = \psi_r$  and the desired speed  $v_d \leq \epsilon$ .

*Proof.* (Singularity avoidance,  $v_d \leq \epsilon$  for some  $\epsilon \ll 1$ )

With the candidate Lyapunov function defined in eq. (6.30), its time derivative is evaluated by combining the procedure from the singularity avoidance proof in

section 6.2 with the bounded stability proof in this section. Through the analysis conducted with the previous proofs, it can be shown that the time derivative of the candidate Lyapunov function in eq. (6.31) will lead to

$$\dot{V}_3 \leq -(\lambda_x - 1) \|e_x\|^2 - (\lambda_y - 1) \|e_y\|^2 - \lambda_\psi \|e_\psi\|^2 - \left(\kappa - \frac{1}{2}\right) \|e_v\|^2 + 2\epsilon^2 + \frac{1}{2} \|d_{max}\|^2 \quad (6.39)$$

Furthermore, equation (6.39) can be given in the following form

$$\dot{V}_3 \leq -\gamma V_3 + \left(2\epsilon^2 + \frac{1}{2} \|d_{max}\|^2\right) \quad (6.40)$$

where,  $\gamma = 2\mu(\mathbf{Q})$ ,  $\mu(\cdot)$  is the spectral radius (maximum eigenvalue) operator, and  $\mathbf{Q} = \text{diag} [(\lambda_x - 1), (\lambda_y - 1), \lambda_\psi, (\kappa - \frac{1}{2})]$ . Hence, we have  $\dot{V}_3 \leq 0$  whenever  $V_3 \geq V_{30} \triangleq \frac{1}{(\gamma/2)} (\epsilon^2 + \frac{1}{4} \|d_{max}\|^2) \triangleq \frac{1}{\mu(\mathbf{Q})} (\epsilon^2 + \frac{1}{4} \|d_{max}\|^2)$ , thus implying boundedness of all the errors and the terms in the guidance laws. Additionally, the effects of the disturbance and the singularity avoidance are bounded by  $d_{max}$  and the tolerance limit,  $\epsilon$ , respectively. Furthermore, the effects can be further limited with a larger value coming from  $\mu(\mathbf{Q})$ , which results from choosing sufficiently large control gains.  $\square$

## 6.4 Simulation Results

The results presented in this section are found by simulating the kinematics, given in eq. (2.2), with the guidance commands designed in sec 6.2, eqs. (6.13) and (6.14). The guidance commands are designed to track a smooth trajectory that is defined along path plans generated by the navigation function algorithm results presented in chapter 3. Additionally, the trajectory along the path plans is designed according to the methodology described in section A.5.1.

### 6.4.1 Trajectory Tracking

For the results presented, the control gains are chosen as constants where  $\lambda_x = \lambda_y = 2$ , and  $\lambda_\psi = \lambda_v = 5$  and  $\alpha_1 = \alpha_2 = 5$ . Also, for the trajectory generation, the goal time to reach the objective destination in the path plan is 180 seconds (3 minutes).

The results for each of the examples considered demonstrate the effectiveness of the nonlinear backstepping guidance design in following the desired trajectory.

Notice that within the position error plots, there are spikes in the errors at certain times. These spikes correlate to the times when the vehicle reaches a point where it must turn to continue on the trajectory. Even as the spikes occur, the errors quickly decay to zero as expected based on the stability analysis of the guidance law. A cause of this phenomena could be due to the fact that the trajectory design does not consider a desired heading profile for the robot to track at the turn points.

Additionally, there is a small amount of terminal state error present in the position tracking towards the end of the simulations. The source of this error can be attributed to the singularity avoidance algorithm. In these instances, the heading angle is locked to the reference value due to the desired velocity being small,  $v_d \leq \epsilon$ , as the robot approaches its objective and is therefore not able to adjust accordingly. This behavior is predicted with the stability proof given in section 6.2.

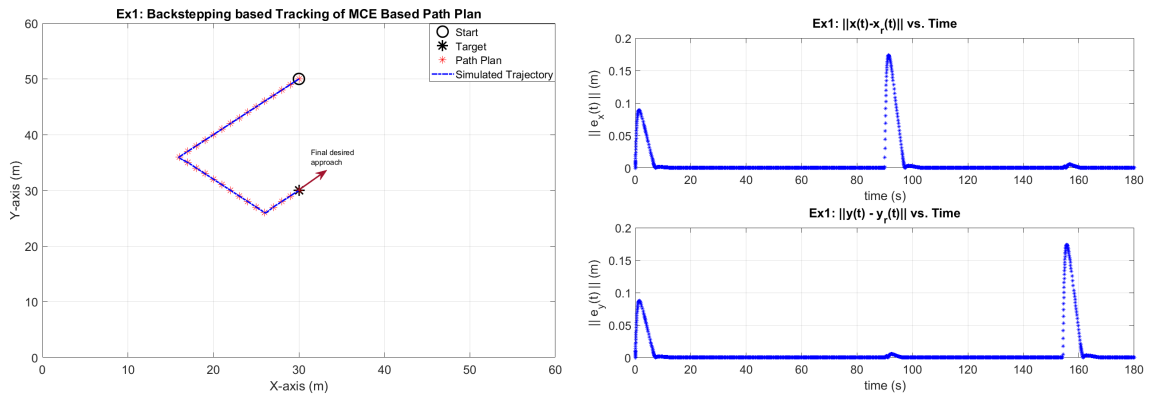


Figure 6.1: Example 1, position tracking with backstepping guidance commands and MCE path plan.

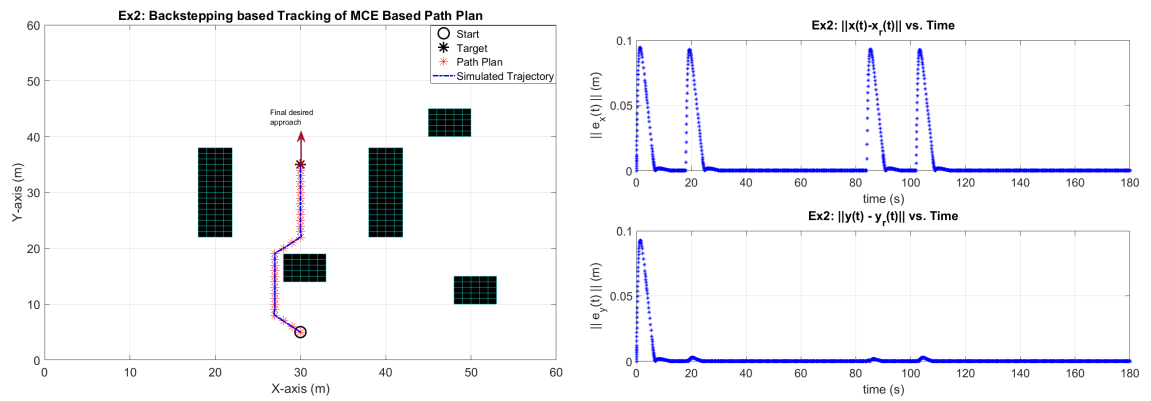


Figure 6.2: Example 2, position tracking with backstepping guidance commands and MCE path plan.

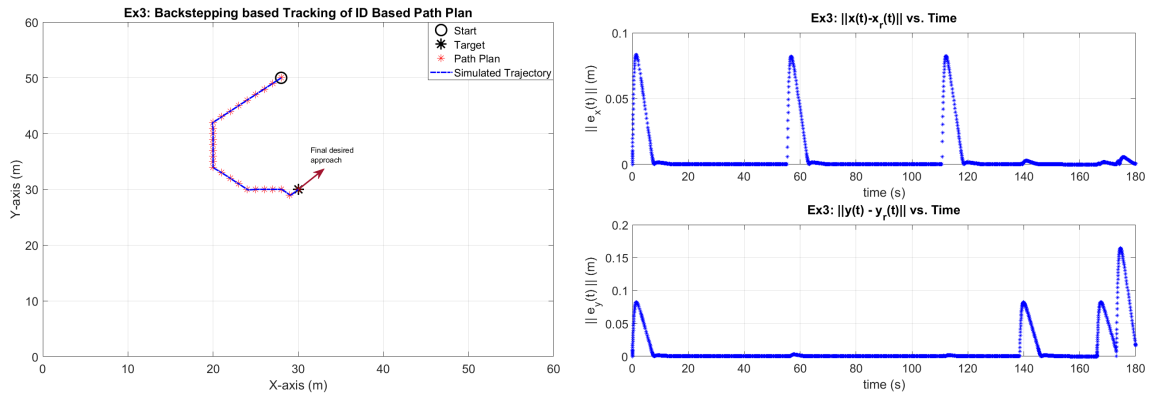


Figure 6.3: Example 3, position tracking with backstepping guidance commands and ID path plan.

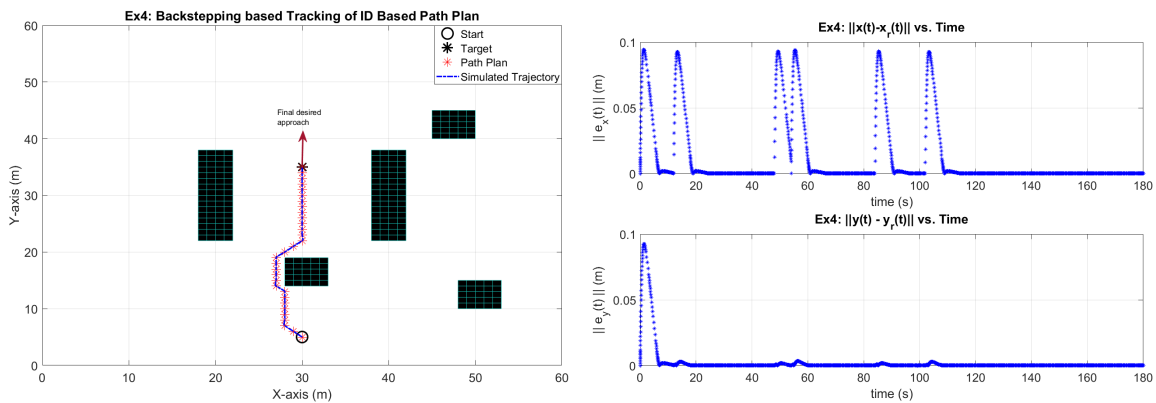


Figure 6.4: Example 4, position tracking with backstepping guidance commands and ID path plan.

#### 6.4.2 Trajectory Tracking with Disturbance Present

The results in this subsection detail an example with the presence of an external disturbance in the forward acceleration of the robot. The results will be presented with an example using the framework design, that is a path plan generated by the navigation function, which is then tracked by the backstepping guidance commands as outlined in section 6.3.



For the following results, it is assumed there is a constant disturbance in the vehicle's acceleration. This illustrates an example of the robot executing a maneuver going up-hill in a given environment. It is assumed that the disturbance is  $d(t) = -2m/s^2$ . Also, the extra guidance command for the velocity, is set as  $u_d = -\kappa_v e_v$ , with  $\kappa_v = 10$ . The other control gains are chosen as,  $\lambda_x = \lambda_y = 2$ , and  $\lambda_\psi = \lambda_v = 5$  and  $\alpha_1 = \alpha_2 = 5$ . Also, for the trajectory generation, the goal time to reach the objective destination in the path plan is 180 seconds (3 minutes).

The scenario for the robot is illustrated in figure 6.5 along with its tracking of the trajectory reaching the goal. Plots of the tracking errors are given in figures 6.6, 6.7 and 6.9, since the effect of the disturbance cannot be fully visualized in figure 6.5. The tracking error plots give the results for the vehicle's performance both with and without the disturbance effects.

The effects of the acceleration disturbance have little effect on the vehicle's heading angle tracking, as demonstrated by the heading angle error plot in figure 6.9. However, the position and velocity tracking error plots show there is a slight increase in the errors with the disturbance present. And as predicted, the tracking errors are ultimately bounded. The presence of some terminal state errors are also shown in figures 6.6 and 6.7, which result from a combination of the external disturbance and the singularity avoidance algorithm, described in section 6.2. Additionally, the velocity error, in figure 6.7, exists as the simulation concludes; however, it is important to note that the robot has stopped, as evident by the velocity plot going to zero in figure 6.8.

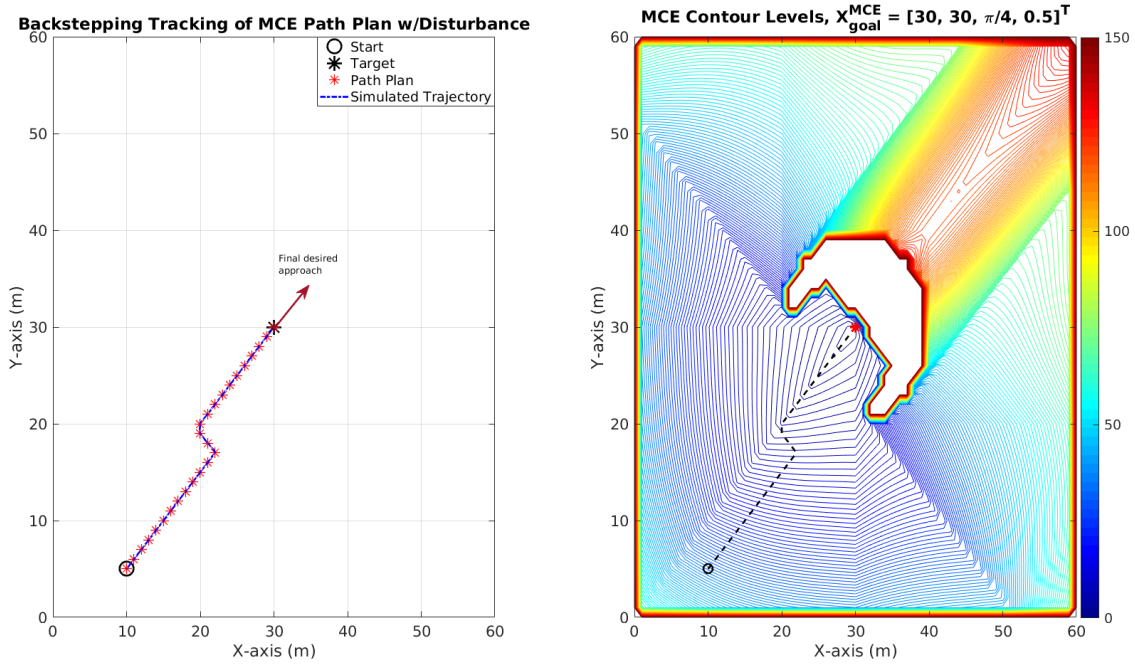


Figure 6.5: Scenario setup and trajectory tracking with disturbance present.

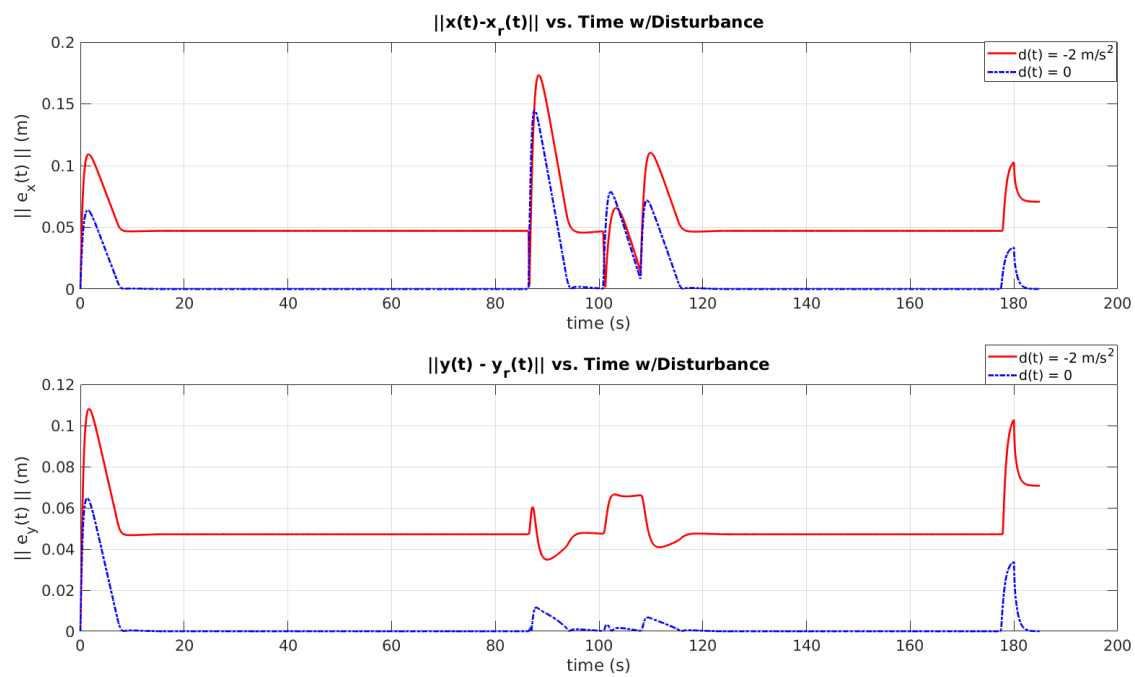


Figure 6.6: Position error plots with  $d = -2m/s^2$ .

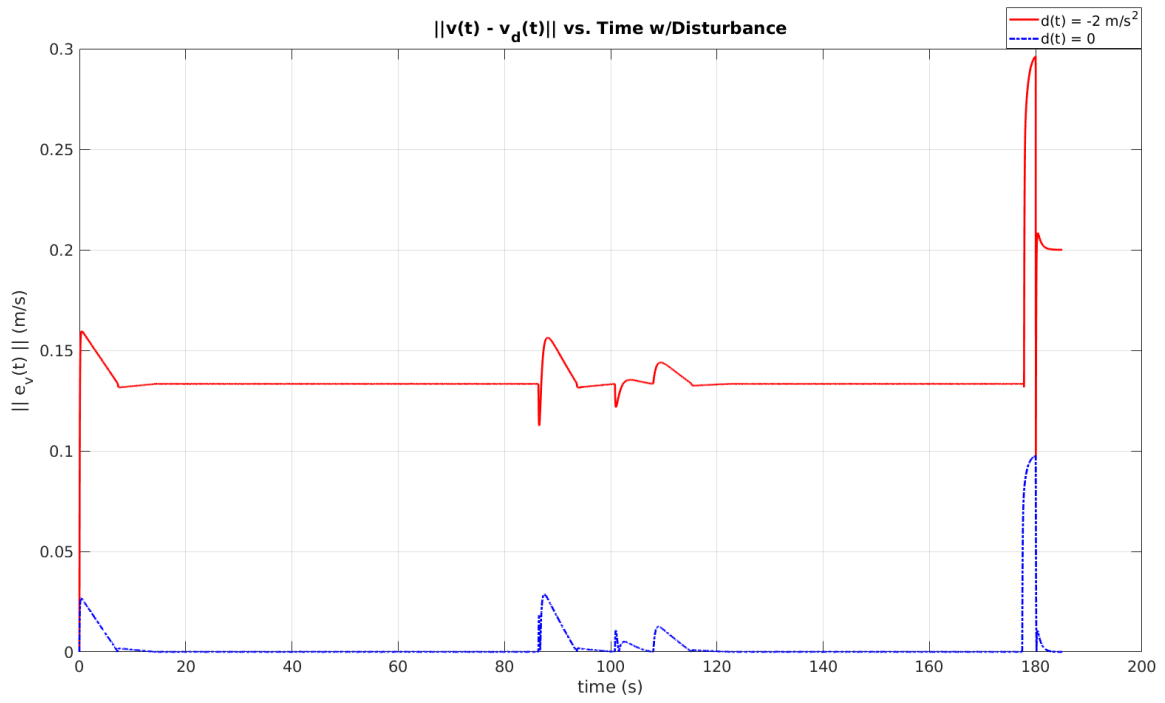


Figure 6.7: Velocity error plot with  $d = -2m/s^2$ .

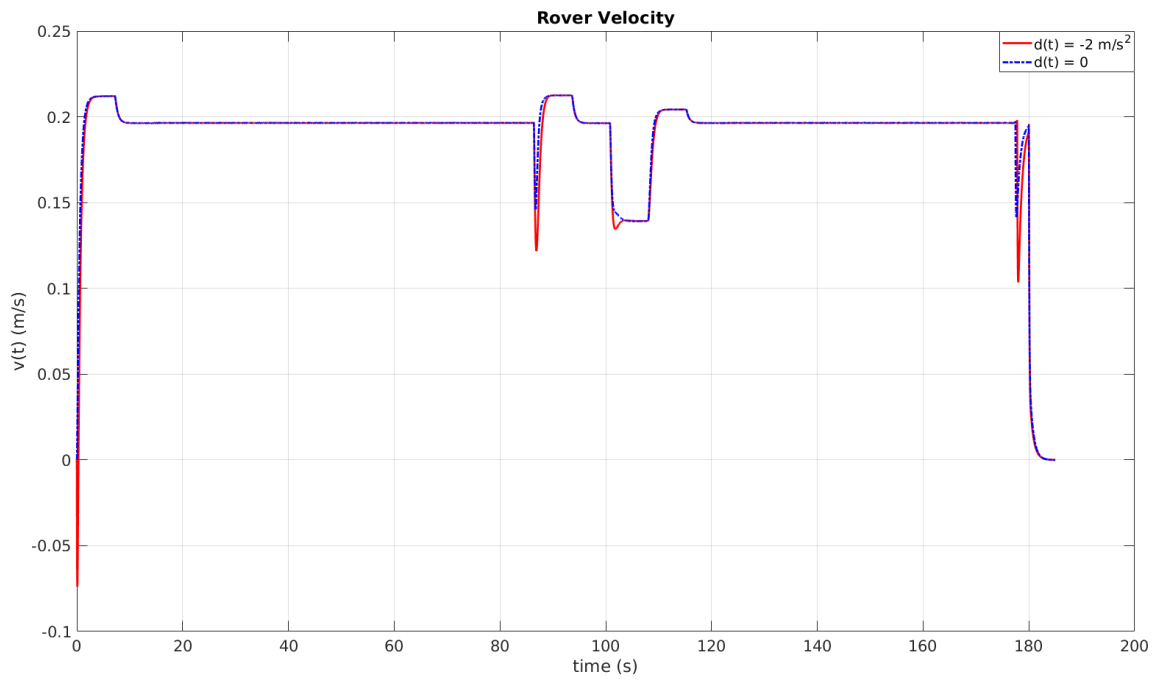


Figure 6.8: Plot of rover velocity with  $d = -2m/s^2$ .

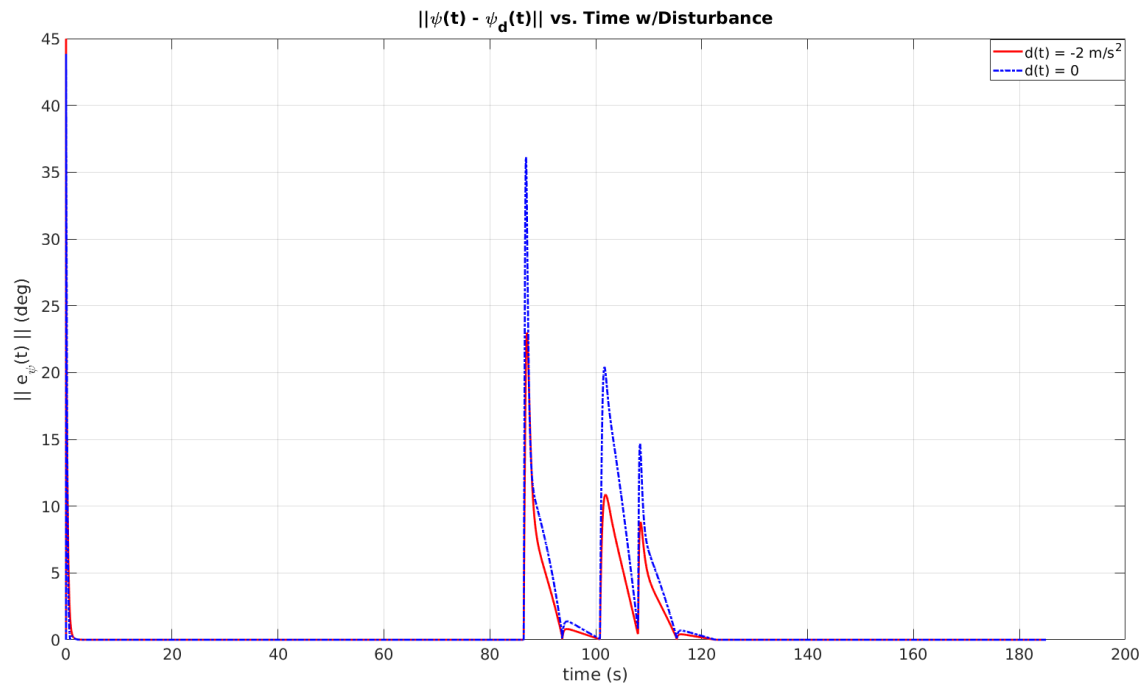


Figure 6.9: Heading angle error plot with  $d = -2m/s^2$ .

## CHAPTER 7

### Cooperative Control with Artificial Potential Functions

The strategy for employing multiple mobile robots cooperatively within the framework of this dissertation is discussed in this chapter. The objective is to demonstrate the ability of the numerical navigation function (described in chapter 3) to be utilized by multiple vehicles in the same environment. The navigation function algorithm can represent the operational environment of the vehicles and can form path plans leading to the goal from any point in the obstacle free space. Hence, only a single potential field needs to be generated for the vehicles to use and it can incorporate each of their sensed information. The fact that a single potential field can be used for each vehicle can conceivably decrease the computational burden of the process.

The cooperative control policy described in this chapter will handle tasks concerning vehicle aggregation and social foraging. For cooperative aggregation tasks, the objective is to bring the vehicles together at a set location while avoiding collisions among the group. And for social foraging, the objective becomes having the robots find ‘conflict-free’ trajectories to areas of interest while avoiding collisions with hazards present in the environment [57, 58]. The cooperative control policy will include different components designed to accomplish each task.

#### 7.1 Artificial Potential Function for ‘Conflict-Free’ Trajectory Synthesis

The cooperative trajectory synthesis for multiple vehicles using artificial potential functions (APF) is considered in this section. Assume that there are  $N$  mobile robots present in a given environment and  $\mathbf{r}^i \in \mathfrak{R}^2$  is the position vector of each

vehicle and  $\mathbf{r}^T = [\mathbf{r}^{1,T}, \dots, \mathbf{r}^{N,T}]$  is the vector containing the position vectors for the group. Then, a steering command for the  $i^{th}$  robot, using the APF design can be given by

$$\dot{\mathbf{r}}_{pot}^i = -\nabla_{\mathbf{r}^i} \mathcal{J}(\mathbf{r}) \quad (7.1)$$

where  $\mathcal{J}(\mathbf{r})$  is a composite potential function consisting of the cooperative potentials, constraints, the navigation function potential as well as other suitable potentials employed for the tasks specified.

The composite potential function has the following form

$$\mathcal{J}(\mathbf{r}) = \delta (\mathcal{J}_{coop}(\mathbf{r}) + \mathcal{J}_{goal}(\mathbf{r})) + (1 - \delta) \mathcal{J}_{NF}(\mathbf{r}) + \mathcal{J}_{rep}(\mathbf{r}) \quad (7.2)$$

where  $\mathcal{J}_{coop}(\mathbf{r})$ ,  $\mathcal{J}_{goal}(\mathbf{r})$ ,  $\mathcal{J}_{NF}(\mathbf{r})$  and  $\mathcal{J}_{rep}(\mathbf{r})$  denote the potential functions for swarm aggregation, goal position, navigation function, and collision avoidance respectively. The scalar quantity  $\delta$  is user defined based on the cooperative task assigned to the group, and is set to either 1 for swarm aggregation at a goal position or 0 for social foraging using the navigation function. Notice that regardless of the choice of  $\delta$ , the collision avoidance potential is always present to ensure the robots do not collide.

Then, the steering command for the  $i^{th}$  robot using the composite potential function in eq. (7.2) is given by

$$\begin{aligned} \dot{\mathbf{r}}_{pot}^i &= \delta (-\nabla_{\mathbf{r}^i} \mathcal{J}_{coop}(\mathbf{r}) - \nabla_{\mathbf{r}^i} \mathcal{J}_{goal}(\mathbf{r})) + (1 - \delta) (-\nabla_{\mathbf{r}^i} \mathcal{J}_{NF}(\mathbf{r})) - \nabla_{\mathbf{r}^i} \mathcal{J}_{rep}(\mathbf{r}) \\ &= \delta (\dot{\mathbf{r}}_{coop}^i + \dot{\mathbf{r}}_{goal}^i) + (1 - \delta) \dot{\mathbf{r}}_{NF}^i + \dot{\mathbf{r}}_{rep}^i \end{aligned} \quad (7.3)$$

The design of each component in eqs. (7.2) and (7.3) are detailed in the following sections.

### 7.1.1 Swarm Aggregation APF Design

This section will detail a suitable potential function used for swarm aggregations and is based on the work presented in references [57, 58, 63]. The APF design has the following form

$$\mathcal{J}_{coop}(\mathbf{r}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \mathcal{J}_{ij}(\|\mathbf{r}^i - \mathbf{r}^j\|) \quad (7.4)$$

making it a function of the norm of the relative position of the respective robots. The potential function given in references [57, 58] possess' both an attractive and a repulsive component. The potential function is designed to simultaneously draw the vehicles in the group together while keeping them apart at a safe distance. The cooperative potential function is defined as

$$\mathcal{J}_{coop}(\mathbf{r}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left[ \frac{a}{2} \|\mathbf{r}^i - \mathbf{r}^j\|^2 + \frac{bc}{2} \exp\left(-\frac{\|\mathbf{r}^i - \mathbf{r}^j\|^2}{c}\right) \right] \quad (7.5)$$

where  $a, b, c \in \Re$  are scalar shaping parameters for the cooperative potential. The  $a$  and  $b$  parameters are used to set the strength of their respective components while the  $c$  parameter sets the region of influence of the repulsive component. The steering command for the  $i^{th}$  robot generated with eq. (7.5) is

$$\dot{\mathbf{r}}_{coop}^i = - \sum_{j=1, j \neq i}^N \left[ \|\mathbf{r}^i - \mathbf{r}^j\| \left( a - b \exp\left(-\frac{\|\mathbf{r}^i - \mathbf{r}^j\|^2}{c}\right) \right) \right] \quad (7.6)$$

The aggregation component of the composite potential function also contains an attractive potential defined at an objective gathering point. This enables a mission planner to designate where in the environment the group should converge. The attractive potential function has the following form

$$\mathcal{J}_{goal}(\mathbf{r}) = \sum_{i=1}^N \mathcal{J}_{i,goal}(\|\mathbf{r}^i - \mathbf{r}^{goal}\|) \quad (7.7)$$



where  $\mathbf{r}^{goal} \in \mathfrak{R}^2$  is the position vector of the gathering point. The attractive potential for the gathering point is given as a parabolic well defined as

$$\mathcal{J}_{goal}(\mathbf{r}) = \sum_{i=1}^N [k_a \|\mathbf{r}^i - \mathbf{r}^{goal}\|^2] \quad (7.8)$$

where  $k_a \in \mathfrak{R}$  is a design parameter to adjust the strength of the potential function.

The steering command for the  $i^{th}$  robot generated by the attractive potential is

$$\dot{\mathbf{r}}_{goal}^i = -k_a \|\mathbf{r}^i - \mathbf{r}^{goal}\| \quad (7.9)$$

The combination of the steering commands in eqs. (7.6) and (7.9) allows for the collective to travel together and ultimately gather at a desired location. However, one aspect that was not addressed with this aggregation design is obstacle avoidance. Thus, the aggregation component of the steering commands in eq. (7.3) assumes the robots are working in an obstacle free space. The issue of obstacle avoidance leading to an objective gathering point is addressed by the navigation function in the following section.

### 7.1.2 Minimum Control Effort Navigation Function for Social Foraging

In short, the task of social foraging refers to having locations in a given environment that are either considered favorable or unfavorable for a group of vehicles to consider. In this dissertation, these regions are defined through the navigation function framework discussed in chapter 3. In the navigation function algorithm, the favorable region is set as the goal location and the unfavorable regions are considered as obstacles.

It is assumed that the vehicles share information such as obstacle positions or other constraints with each other. This information is then used to form the navigation function's potential field. Hence, each robot knows the potential levels generated from the navigation function algorithm.

The navigation function algorithm in chapter 3 details how to form a numerical potential field in an evenly spaced grid given knowledge of the operational environment. The algorithm, as outlined in chapter 3, generates a potential field which is then used to find a path plan using an iterative graph search method (best-first search). So, the reference path is found before the robot has the ability to act. This approach does not allow for a reactive element, i.e. generating steering commands, directly based on the environment. The task then becomes how to use the resulting potential field from the algorithm in chapter 3 to generate steering commands allowing a robot to react to its environment. This result will allow the group of mobile robots to navigate a constrained environment towards a desired gathering point.

#### Central Difference Approximation to NF Gradient

This section will detail how to find the numerical gradient of the navigation function and how it is used within the cooperative control policy. The gradient of the navigation function needs to be generated numerically since it is defined over a discrete representation of the environment. The method chosen to compute the gradient of the navigation function is the Central-Difference method [68].

In order to use the grid-based potential function and the central-difference approximation of the gradient, the position of a robot needs to be defined in terms of the evenly spaced grid. With this in mind, let  $\tilde{\mathbf{r}}^i \in \mathbb{R}^2$  denote the approximate position of the  $i^{th}$  robot in the evenly spaced grid. And let  $\tilde{x}^i$  and  $\tilde{y}^i$  be the approximate position components of  $\tilde{\mathbf{r}}^i$ , i.e.  $\tilde{\mathbf{r}}^i = [\tilde{x}^i, \tilde{y}^i]^T$ . Also, let  $\Delta x$  and  $\Delta y$  denote the spacing between grid points in the  $x$  and  $y$  directions respectively. And since the grid is evenly spaced,  $\Delta x = \Delta y$ . Then, the components of the numerical gradient, computed with the

central-difference method, is determined in terms of the robot's approximate position in the grid as

$$\begin{aligned}\frac{\partial \mathcal{J}_{NF}(\tilde{\mathbf{r}}^i)}{\partial x} &= \frac{\partial \mathcal{J}_{NF}([\tilde{x}^i, \tilde{y}^i]^T)}{\partial x} \approx \frac{\mathcal{J}_{NF}([\tilde{x}^i + \Delta x, \tilde{y}^i]^T) - \mathcal{J}_{NF}([\tilde{x}^i - \Delta x, \tilde{y}^i]^T)}{2\Delta x} \\ \frac{\partial \mathcal{J}_{NF}(\tilde{\mathbf{r}}^i)}{\partial y} &= \frac{\partial \mathcal{J}_{NF}([\tilde{x}^i, \tilde{y}^i]^T)}{\partial y} \approx \frac{\mathcal{J}_{NF}([\tilde{x}^i, \tilde{y}^i + \Delta y]^T) - \mathcal{J}_{NF}([\tilde{x}^i, \tilde{y}^i - \Delta y]^T)}{2\Delta y}\end{aligned}\quad (7.10)$$

And the steering command based on the numerical gradient of the navigation function can be set as

$$\dot{\mathbf{r}}_{NF}^i = \begin{bmatrix} -\frac{\partial \mathcal{J}_{NF}(\tilde{\mathbf{r}}^i)}{\partial x} \\ -\frac{\partial \mathcal{J}_{NF}(\tilde{\mathbf{r}}^i)}{\partial y} \end{bmatrix}\quad (7.11)$$

### 7.1.3 APF for Collision Avoidance

Within the cooperative framework, there is an extra collision avoidance potential denoted as  $\mathbf{r}_{rep}^i$ . The collision avoidance term consists of only a repulsive term designed to prevent the robots from colliding with one another. Note that this repulsive potential function may also be used to avoid collisions with obstacles in the environment. The collision avoidance potential is of the form

$$\mathcal{J}_{rep}(\mathbf{r}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \mathcal{J}_{ij}(\|\mathbf{r}^i - \mathbf{r}^j\|)\quad (7.12)$$

which is similar to the cooperative potential form in eq. (7.4) and is a function of the position information of the robots. The repulsive potential chosen for the results covered is consistent with the repulsive potential component in eq. (7.5) given by

$$\mathcal{J}_{rep}(\mathbf{r}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left[ \frac{b_r c_r}{2} \exp\left(-\frac{\|\mathbf{r}^i - \mathbf{r}^j\|^2}{c_r}\right) \right]\quad (7.13)$$

where the parameters  $b_r, c_r \in \mathfrak{R}$  are used to respectively set the strength and region of influence of the repulsive potential. The subscript  $r$  is used to distinguish the

parameters in eq. (7.13) from the parameters in eq. (7.5). The resulting steering command for the  $i^{th}$  vehicle is given by

$$\dot{\mathbf{r}}_{rep}^i = \sum_{j=1, j \neq i}^N \left[ b_r \|\mathbf{r}^i - \mathbf{r}^j\| \exp \left( -\frac{\|\mathbf{r}^i - \mathbf{r}^j\|^2}{c_r} \right) \right] \quad (7.14)$$

## 7.2 Control Design

Typical control designs with artificial potential functions, including cooperative control designs, are applied to integrator dynamic systems (or sometimes double integrator). The goal of this work is to apply the control design to a system with the nonlinear mobile robot kinematics, as described in section 2.2.1. Thus, the kinematics for the  $i^{th}$  vehicle are

$$\begin{aligned} \dot{x}^i &= v^i \cos(\psi^i) \\ \dot{y}^i &= v^i \sin(\psi^i) \\ \dot{\psi}^i &= \alpha_1 (\psi_c^i - \psi^i) \\ \dot{v}^i &= \alpha_2 (v_c^i - v^i) \end{aligned} \quad (7.15)$$

where  $\alpha_1, \alpha_2 \in \mathfrak{R}$  are positive proportional control gains and  $\psi_c^i, v_c^i$  are the guidance commands for the  $i^{th}$  vehicle's heading angle and velocity. The control design affects the robot's heading angle turn rate and forward acceleration, which are proportional control laws designed to drive the heading angle and velocity to the commanded values which are obtained through the steering commands from the APF described in the previous section.

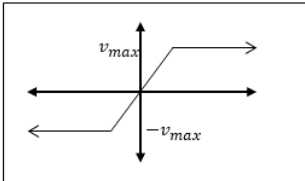
Let  $\dot{\mathbf{r}}_{pot,x}^i$  and  $\dot{\mathbf{r}}_{pot,y}^i$  denote the  $x$  and  $y$  directional components of the gradient terms that make up the steering command from eq. (7.2). Then, the heading angle guidance command,  $\psi_c^i$ , for the  $i^{th}$  robot is defined as

$$\psi_c^i = \tan^{-1} \left( \frac{\dot{\mathbf{r}}_{pot,y}^i}{\dot{\mathbf{r}}_{pot,x}^i} \right) \quad (7.16)$$

And the velocity guidance command for the  $i^{th}$  robot is

$$v_c^i = \|\mathbf{r}_{pot}^i\| \quad (7.17)$$

which is the 2-norm of the steering command vector. It is possible that the velocity command may give an unfeasible value. Therefore, the velocity command term is saturated as

$$v_c^i = sat(v_c^i) =$$


This implies that  $|v_c^i| \leq v_{max}$  where the saturation limit,  $v_{max}$ , is a user defined quantity for the forward velocity of the vehicle. This signifies that the steering commands will mainly influence the heading angle of the robot. Thus, the vehicles can continue to travel forward and adjust their heading angles as needed while traversing the environment with the guidance commands generated through the cooperative control policy.

### 7.3 Inter-Vehicle Communication

For the results in this dissertation, it is assumed that there is full communication among the vehicles. A graph representation is used to illustrate the communication protocol involved in the simulations. In graph theory, a graph is described as a pair  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ , with  $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_N\}$  a set of  $N$  nodes and  $\mathcal{E}$  as a set of edges. The elements of  $\mathcal{E}$  are denoted as  $(\mathcal{N}_i, \mathcal{N}_j)$  which is termed as an edge from node  $\mathcal{N}_i$  to

node  $\mathcal{N}_j$  [56]. In the setting of cooperative control, the nodes represent the vehicles and the edges represent the communication of information.

An illustration of the communication graph for the cooperative control examples with three mobile robots is given in figure 7.1. In the examples given, it is assumed that the navigation function information as well as the position vectors are shared among the robots through the communication protocol.

Also, the communication graph is assumed to be undirected. Making the graph undirected implies that the information shared goes in both directions and has equal importance, i.e. node  $\mathcal{N}_i$  shares its information with node  $\mathcal{N}_j$  and vice-versa [56]. Therefore, in figure 7.1, the edges are denoted as  $\mathcal{E}_{ij} = \mathcal{E}_{ji}$ , which represents an undirected communication link between nodes  $\mathcal{N}_i$  and  $\mathcal{N}_j$ ,

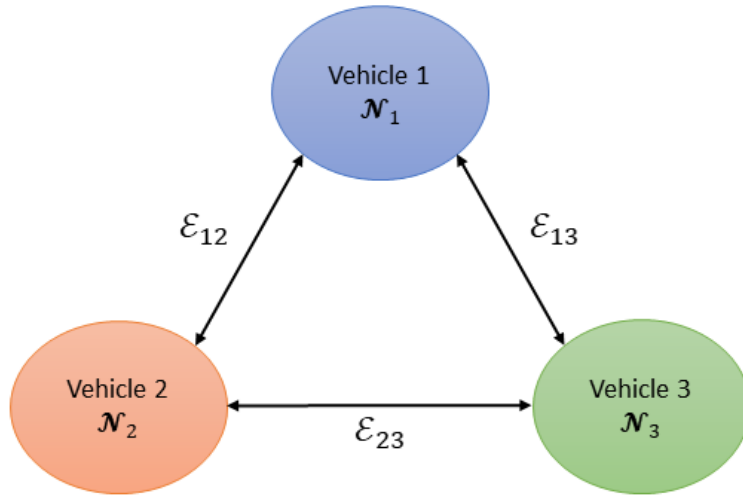


Figure 7.1: Communication protocol.

#### 7.4 Simulation Results

The simulation results in this chapter consider scenarios involving three vehicles in a given environment. The cooperative APF parameters for the steering command

in eq. (7.6) for the simulation are chosen as  $a = 0.3$ ,  $b = 1$ ,  $c = 10$  and the parameter for the steering command to the goal in eq. (7.9) as  $k_a = 0.5$ . The collision avoidance parameters for the steering command in eq. (7.14) are set as  $b_r = 1$ ,  $c_r = 2$ . The control gains for the guidance command inputs in eq. (7.15) are set as  $\alpha_1 = 0.5$  and  $\alpha_2 = 0.5$ . The maximum velocity for each robot is set at  $v_{max} = 0.75 \text{ m/s}$ . The navigation function algorithm method used for the results in this chapter is the minimum control effort approach, detailed in section 3.2.

The first scenario, shown in figures 7.2 and 7.3, is for an environment with no obstacles present. This example represents a case where the swarm aggregation policy can be tested and compared with the social foraging policy to the same goal. The swarm aggregation policy is enacted by setting  $\delta = 1$  in eq. (7.3) and the social foraging policy is enacted by setting  $\delta = 0$ .

The results in figures 7.2 and 7.3 show the positions of the robots, starting in close proximity in the bottom corner and traveling towards the opposite corner of the environment while avoiding collisions within the group. The robots position components look similar, but their heading angles fluctuate largely in figure 7.3 with the social foraging policy. This behavior begins as the robots are approaching the safety zone around the goal position. When the vehicles enter the safety zone, their velocities are driven to zero so they stop, but the heading angles are directing the robots away from each other. In contrast, the heading angle plot in figure 7.2 shows the robots smoothly achieving consensus as they enter the safety zone around the goal.

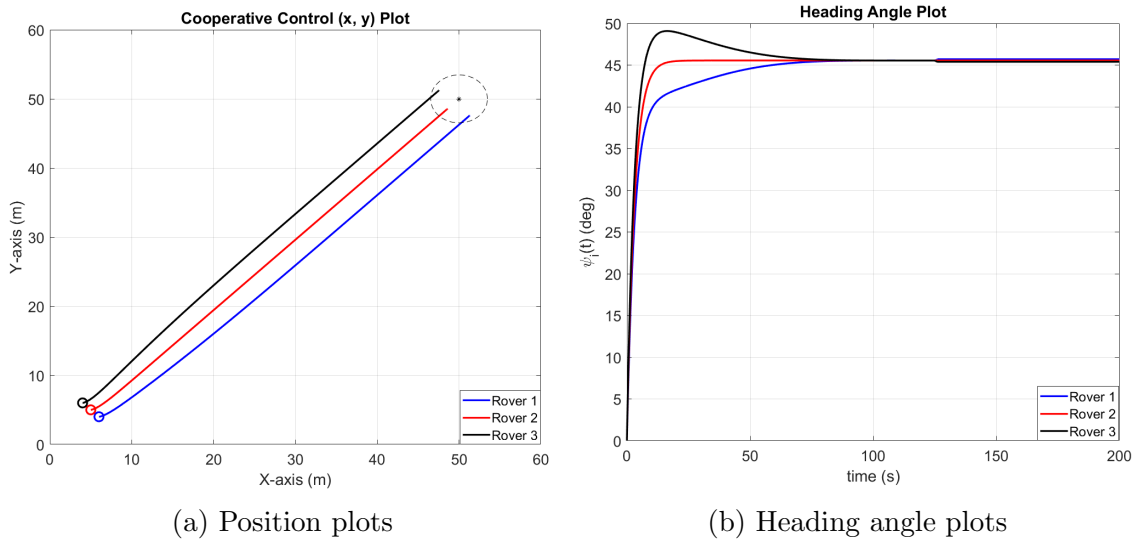


Figure 7.2: Example 1 with swarm aggregation policy, setting  $\delta = 1$  in eq. (7.3).

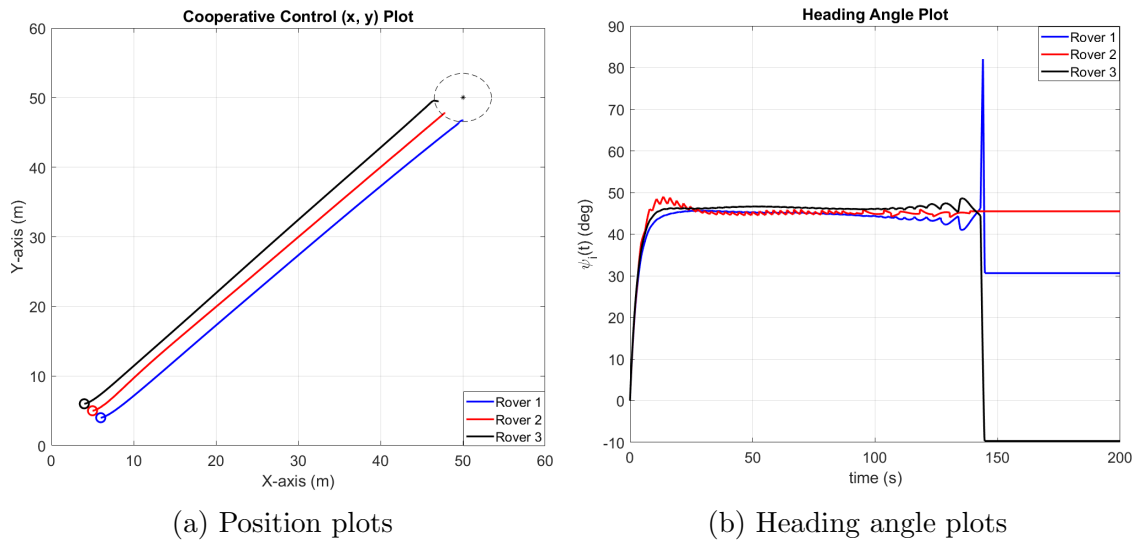


Figure 7.3: Example 1 with social foraging policy, setting  $\delta = 0$  in eq. (7.3).

The next two examples are considered only with social foraging behavior, by setting  $\delta = 0$  in eq. (7.3). The steering commands are generated by the navigation function's potential field combined with the collision avoidance potential from equa-



tions (7.11) and (7.14) respectively. The first set of plots show the given obstacle laden environment and its associated navigation function potential field. The second set of plots show the simulation results with the position plots and the heading angle plots.

The results illustrated in figures 7.4 and 7.5 are for an environment where multiple scattered hazards lie between the robots and their objective. The simulation results in figure 7.5 show that the robot's steering commands allow the vehicles to avoid colliding with the obstacles and each other while reaching the safety zone near the objective.

The next set of results illustrated in figures 7.6 and 7.7 are for an environment where the vehicles must come together through a tight opening before spreading apart and reaching the goal. The simulation results in figure 7.7 demonstrate that the robots achieve their task without conflict as they come together to go through the opening then spread apart in the free space on the other side and ultimately arrive at the safety zone near the objective.

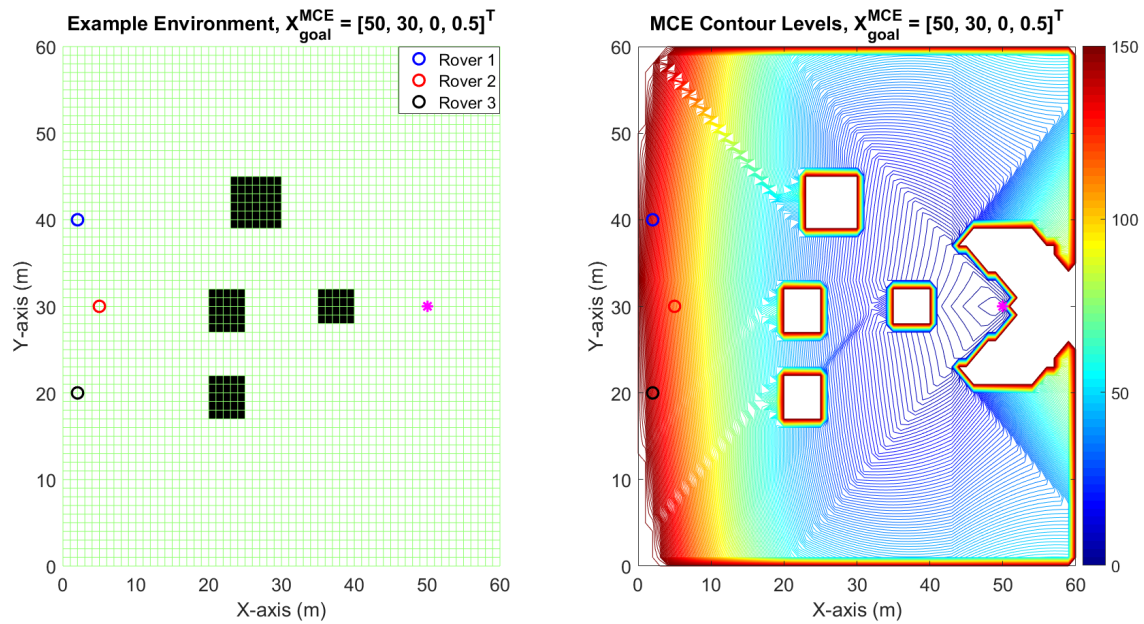
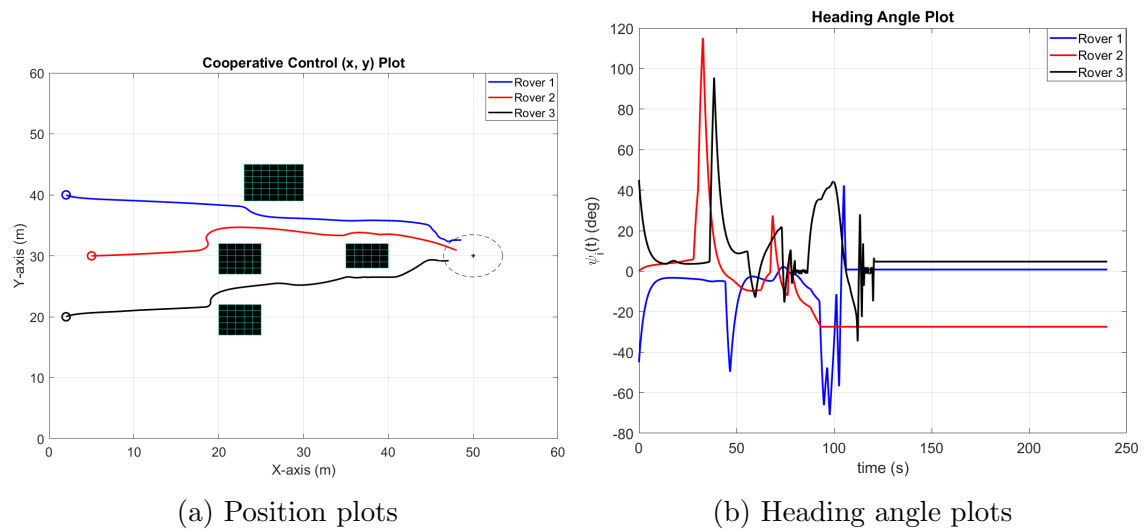


Figure 7.4: Example 2 environment and navigation function potential field.



(a) Position plots

(b) Heading angle plots

Figure 7.5: Example 2 with social foraging policy, setting  $\delta = 0$  in eq. (7.3).

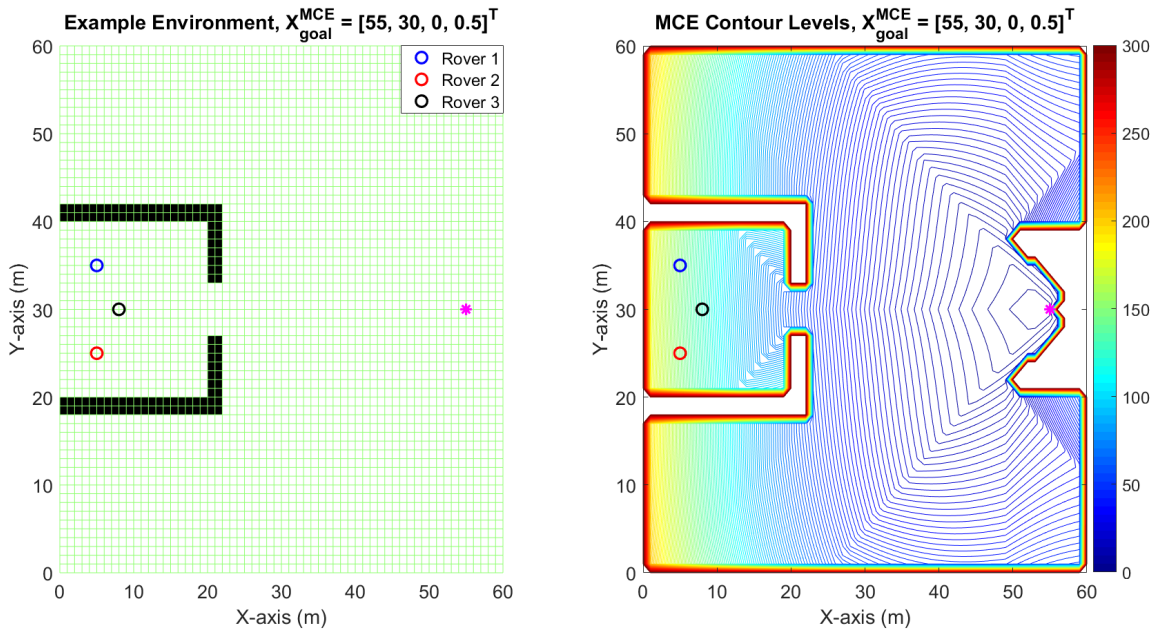
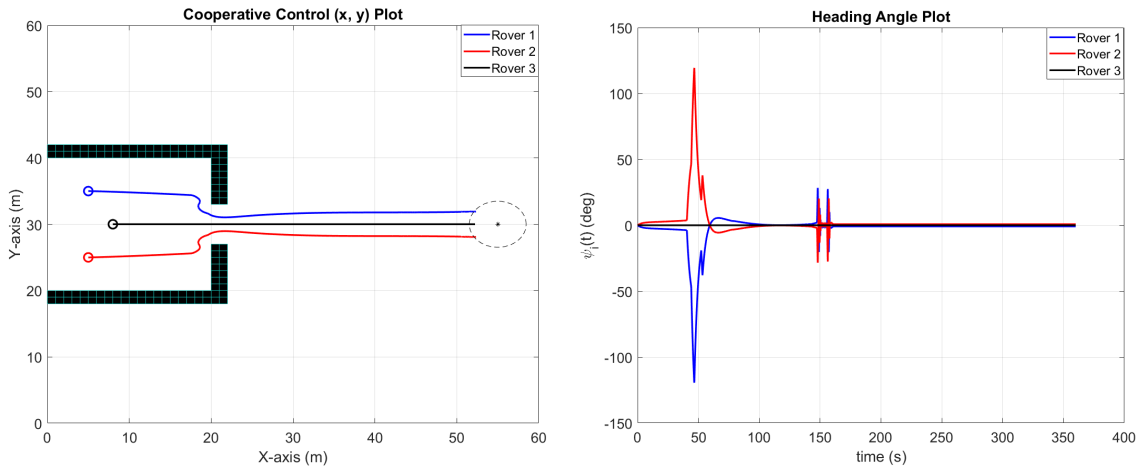


Figure 7.6: Example 3 environment and navigation function potential field.



(a) Position plots

(b) Heading angle plots

Figure 7.7: Example 3 with social foraging policy, setting  $\delta = 0$  in eq. (7.3).

## CHAPTER 8

### Real-Time Experiment Setup and Results

To validate the results for the nonlinear guidance laws in chapters 5 and 6 as well as the cooperative control policy outlined in chapter 7, the differential drive mobile robot testing platforms from the Aerospace Systems Laboratory were used. An image of the mobile robot platform is given in figure 8.1. A full description, covering the assembly and component details of the testing platforms, nicknamed the ‘ASL-Gremlin,’ is given in reference [69]. A summary of the important components and flow of information in the experimental setup is given in this chapter along with a summary of the experimental results gathered to validate the methods discussed in this dissertation.



Figure 8.1: ASL-Gremlin Mobile Robot Testing Platform.

## 8.1 Mobile Robot Platform

This section gives a summary description of the hardware and software components that makeup the testing platforms. Additionally, this section will detail the flow of information within the framework for the given tasks.

### 8.1.1 Hardware Components

The mobile robot platforms consist of a combination of multiple hardware components designed to test different guidance, navigation and control techniques. The main component directing the communication between the different parts of the robot is the single-board computer (SBC). The SBC is an on-board computer that is used to interface with the hardware and process information. The SBC used in the experiments presented in this dissertation is an Odroid-XU4 computer. Its main task is to collect sensor data, relay information and command the input signals to a micro-controller that handles the actuators (motors attached to wheels) of the vehicle.

Another component used in the mobile robot platform is the Pixhawk autopilot. The autopilot module for these platforms is a small, light-weight component that is connected to a suite of sensors used by the mobile robot to localize itself within a given environment. The main sensors used in this research are a global positioning system (GPS), which is used to find the local position and velocity of the vehicles and a compass, used to determine the heading angle of the vehicles. Although the GPS and compass are the main sensors used in this research, the Pixhawk autopilot system does come with a 3-axis gyroscope, 3-axis accelerometer and a magnetometer which can be used in future research to enhance the vehicles autonomous capabilities.

The next important component on the platform is the Arduino micro-controller. For the types of experiments conducted in this research, the micro-controller is responsible for digital signal processing and voltage regulation to control the vehicle's

wheels. Specifically, for the differential drive robots, the micro-controller is used to convert commanded pulse-width-modulation (PWM) signals to a motor input voltage to control the speed which the motor rotates its attached wheel. The mobile robot platforms consists of four 6V brushed DC motors which come complete with gear-boxes and quadrature encoders on the motor’s shaft. Note that the encoders may also be used to gather position and orientation information for the vehicle through a technique called “Dead-Reckoning.” For the results in this dissertation, however, the encoders are not used due to the potential for the drift-error in the information that is caused by wheel slippage.

An illustration of the top-view of a single mobile robot platform is given in figure 8.2 which shows the hardware components on-board the vehicle. Then, a chart depicting the the flow of information between the different components as used for the results in this dissertation is given in figure 8.3.

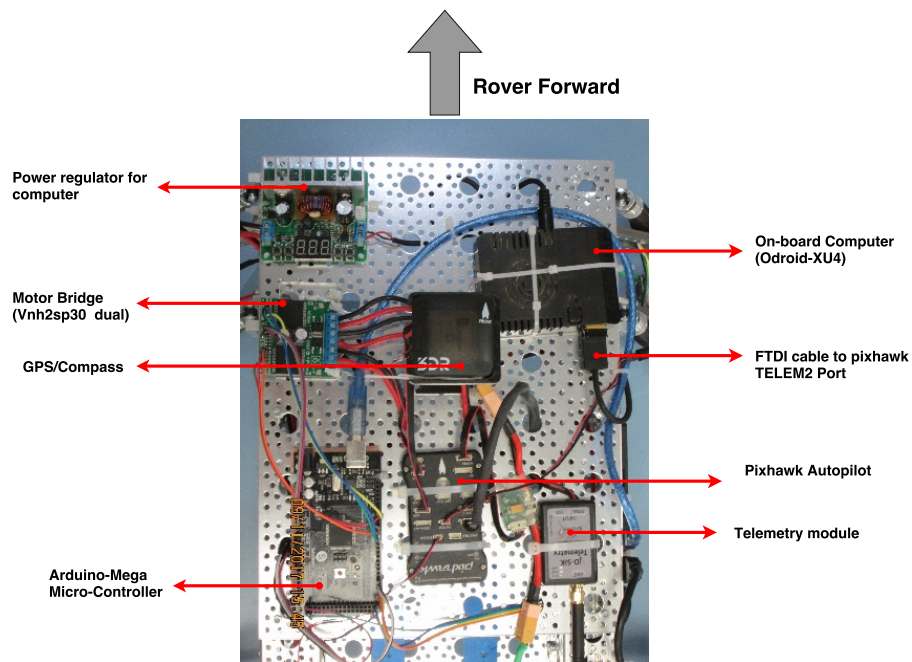


Figure 8.2: On-board hardware components for the ‘ASL-Gremlin’ mobile robot platform.

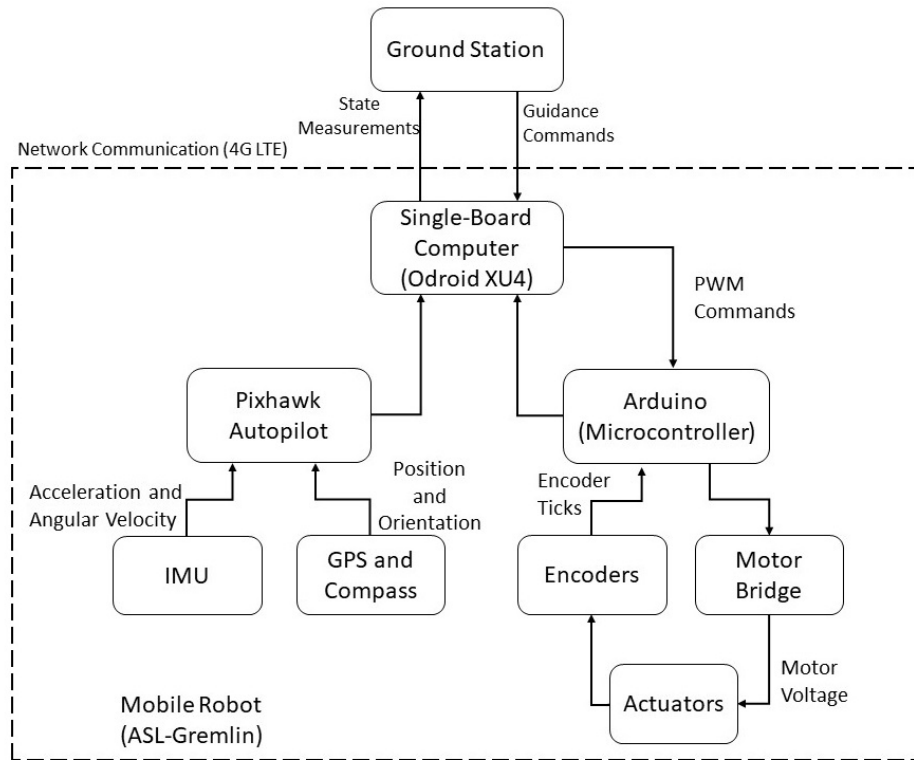


Figure 8.3: Flow of information between hardware components for mobile robot platform.

The flow of information, for a single vehicle, illustrating how this architecture is used within the GNC framework, is given in figure 8.4. The results in this dissertation make use of a cyber-physical system architecture wherein the main data processing and guidance commands are generated on a separate ground station computer which dictates the desired behavior for the physical mobile robot system, as illustrated in figure 8.3. The information is transmitted between a ground station CPU and the Odroid on-board CPU through a 4G LTE mobile network hotspot.

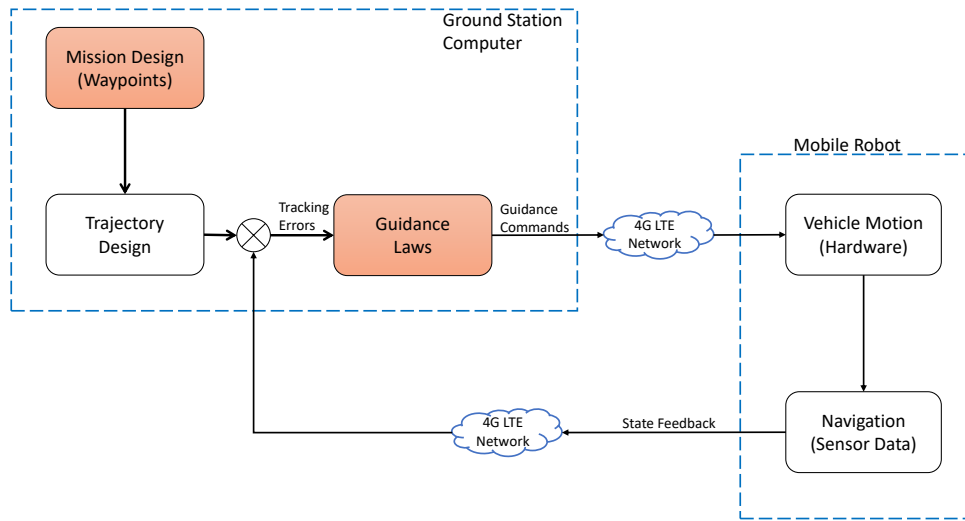


Figure 8.4: Flow of information within the GNC framework for the mobile robot platform.

### 8.1.2 Cooperative Platform Setup

For the cooperative control experiments presented in this chapter, two mobile robot platforms are used in combination with a ground station computer. Similar to the individual mobile robot tasks, the communication of information will take place over a 4G LTE mobile network and is considered as a cyber-physical system. The flow of information for the cooperative tasks presented, is illustrated in figure 8.5.



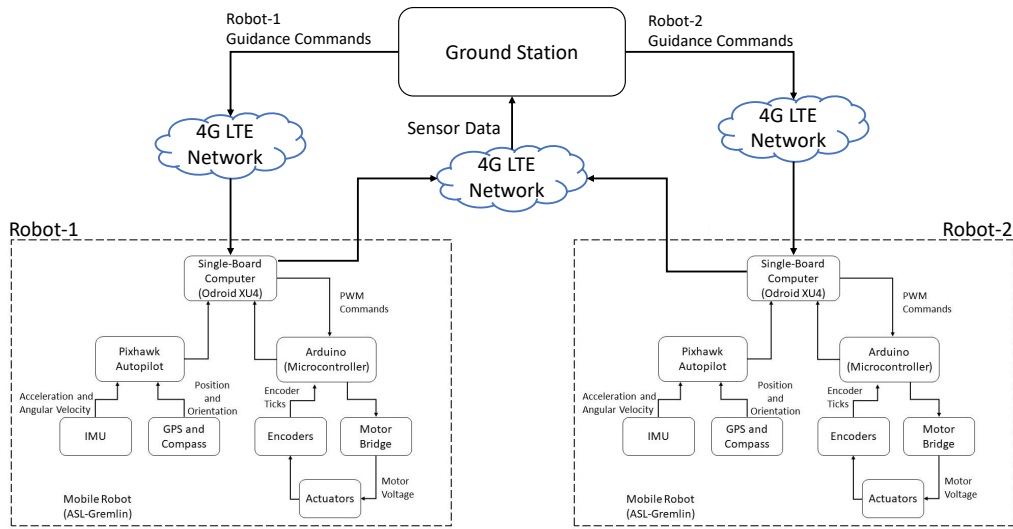


Figure 8.5: Flow of information for cooperative control experiments.

Note that the cooperative guidance commands, as derived in chapter 7, are designed within a decentralized control framework. However, for the experiments in this dissertation, the guidance commands are synthesized on a centralized ground station computer and are then transmitted to the individual vehicles to complete their tasks.

### 8.1.3 Software

The transmission of data between the different components used in the experiments is handled through the Robot Operating System (ROS). ROS is a flexible software framework that is widely used by a variety of research communities and can be applied to different robotic platforms <sup>1</sup>. This software communication architecture

<sup>1</sup>[www.ros.org](http://www.ros.org)

is chosen because it simplifies the programming tasks and enables rapid implementation of theoretical robotics concepts to actual hardware.

Within the mobile robot platforms used in this dissertation, ROS is installed on both the ground station CPU and the on-board Odroid CPU, both running flavors of the Ubuntu Linux operating system. ROS is used to gather the sensor information from the Pixhawk autopilot and make it available for use by the methods discussed in this dissertation. Additionally, ROS is used to relay the wheel speed commands to the micro-controller used to move the vehicle.

The methods derived in this dissertation were applied to the mobile robot platforms using MATLAB, Simulink and the Robotics System Toolbox. The Robotic System Toolbox allows for the integration of ROS software within the MATLAB/Simulink environment.

## 8.2 Real-Time Experiment Results

This section covers real-time experiment results using the guidance command techniques from chapters 5 and 6 and the cooperative control steering commands from chapter 7. First, to use the mobile robot platforms, a relationship between the guidance commands and the wheel speeds is established. The relationships between the wheel speeds of the differential drive vehicles and the forward velocity and heading angle turn rate are given by

$$v = \frac{r}{2}(\omega_R + \omega_L) \quad (8.1)$$

$$\dot{\psi} = \frac{r}{b}(\omega_R - \omega_L) \quad (8.2)$$

where  $\omega_L$  and  $\omega_R$  respectively denote the left and right wheel speeds,  $b$  is the base length of the vehicle and  $r$  is the radius of the wheel. The physical parameter dimensions of the vehicles used in these experiments are  $r = 0.0686 \text{ m}$  for the wheel radius

and  $b = 0.3358 \text{ m}$  for the base-length. Equations (8.1) and (8.2) can be solved simultaneously to give the wheel speeds in terms of the velocity and turn rate as

$$\omega_L = \frac{1}{2} \left( \frac{2}{r} v - \frac{b}{r} \dot{\psi} \right) \quad (8.3)$$

$$\omega_R = \frac{1}{2} \left( \frac{2}{r} v + \frac{b}{r} \dot{\psi} \right) \quad (8.4)$$

If the guidance commands, from chapters 6 and 7, are generated in terms of the commanded velocity ( $v^c$ ) and heading angle ( $\psi^c$ ), then the velocity and turn rate values in eqs. (8.3) and (8.4) are determined by

$$v = v^c$$

$$\dot{\psi} = \alpha_1 (\psi^c - \psi)$$

where  $\alpha_1$  is the scalar control gain used for the guidance command and  $\psi$  is the vehicle's current heading angle. But, if the guidance commands are generated in terms of the vehicle's acceleration ( $u_2$ ) and heading angle turn rate ( $u_1$ ), as in chapter 5, then the velocity and turn rate values in eqs. (8.3) and (8.4) are determined by

$$v = v_0 + u_2 \Delta t$$

$$\dot{\psi} = u_1$$

where  $v_0$  is the vehicle's current velocity and  $\Delta t$  is the given sample time used with the NMPC formulation.

The experiment results are given in two sets. The first set will consist of the results coming from the nonlinear guidance methods from chapters 5 and 6. These results will consider a single mobile robot traversing multiple given waypoints using the guidance commands to follow a trajectory. The second set of results will be obtained using the APF cooperative control policy detailed in chapter 7. These results will consider two mobile robots performing cooperative aggregation and social foraging tasks.

### 8.2.1 Individual Mobile Robot Results

For the individual mobile robot tasks, a series of waypoints are given for the vehicle to reach while using the two guidance techniques from chapters 5 and 6. A polynomial trajectory is designed between each of the waypoints for the vehicle to track, according to the approach given in appendix A.5.2. There are two different experiment setups used to test the guidance laws. The setups provide the waypoints in patterns with different angles and distances between them. The maximum acceleration used to define the trajectory between the waypoints is  $a_{max} = 0.08 \text{ m/s}^2$ .

The input and state constraints used for the NMPC guidance law are  $\|u_1\| \leq 150 \text{ deg/s}$ ,  $\|u_2\| \leq 0.08 \text{ m/s}^2$ ,  $0.0001 \text{ m/s} \leq v \leq 1 \text{ m/s}$ , and  $\|\psi\| \leq \pi \text{ rad}$ , respectively. The input and state weight matrices are defined as diagonal matrices with  $R = \text{diag}(0.2, 0.2)$  and  $Q = \text{diag}(5, 5, 5, 5)$ , respectively. Also, a waypoint proximity is defined so that once the vehicle is within this proximity to a waypoint, it can switch to approaching the subsequent waypoint in the given set. The waypoint proximity used in these experiments is  $\epsilon_{prox} = 0.8 \text{ m}$ .

#### Individual Experiment 1

The first experiment, using the nonlinear guidance laws, has three waypoints defined to form a C-shape in the environment. This example is comparable to a vehicle needing to go around an obstacle to get to a destination on the other side. The setup for this experiment is illustrated in figure 8.6.

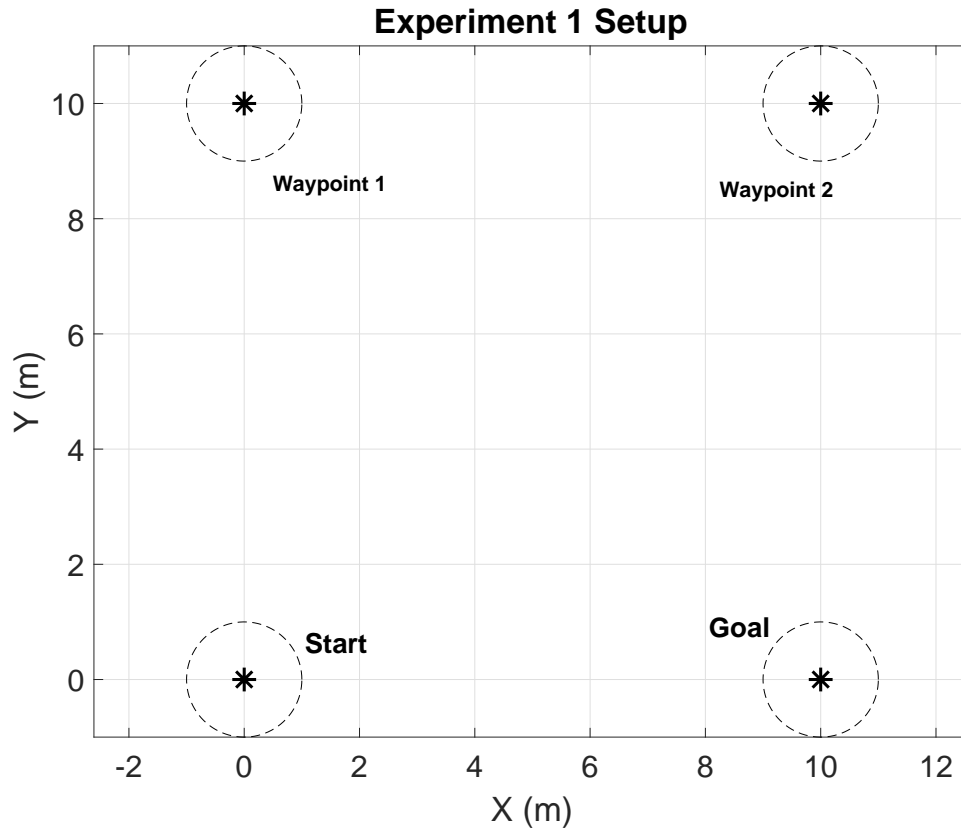


Figure 8.6: Waypoint setup for individual experiment 1.

And the resulting trajectories of the mobile robot using both guidance laws is shown in figure 8.7. The robot's resulting positions in figure 8.7 is given based on the GPS position measurements. The spacing between data points in the position data represents a loss of information either due to communication delay or some other miscue. In both instances, the robot is able to reach each waypoint and arrives at the goal point at the end of the experiment.

While both guidance laws reach each of the desired waypoints, the trajectory of the robot with the nonlinear backstepping guidance law seems more direct. In comparison, the NMPC guidance law result shows the robot gradually aligning itself with the desired waypoints. This difference in behavior can be attributed to the

constraints being explicitly enforced in the NMPC guidance law synthesis whereas these same constraints are not considered for the backstepping design.

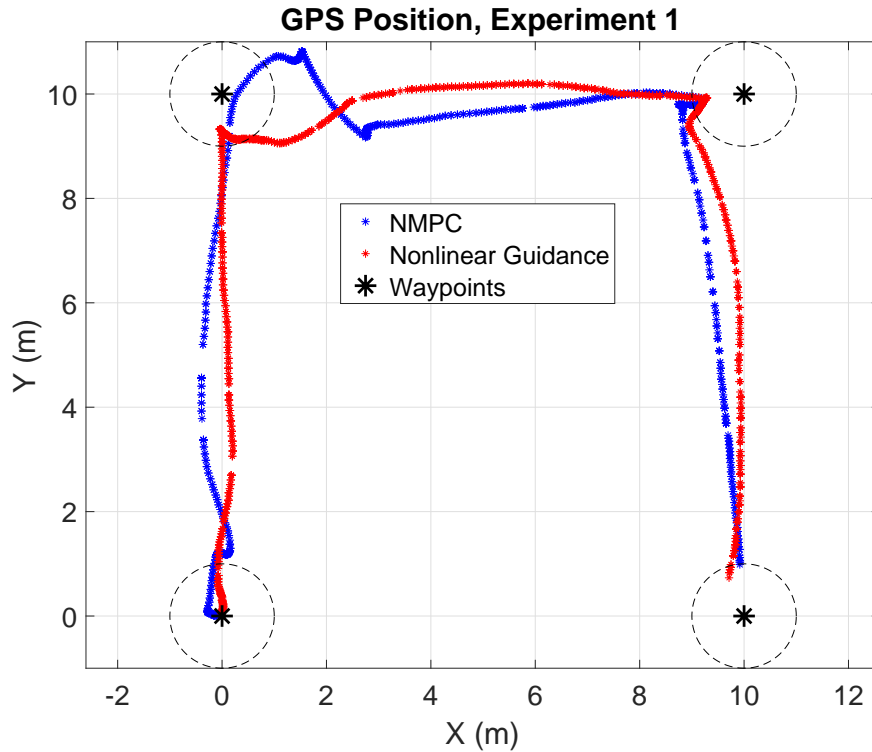


Figure 8.7: Robot's waypoint traversal for individual experiment 1.

The result in figure 8.8 illustrates the heading angle of the vehicle as it moves between the waypoints. The behavior displayed in figure 8.8 shows that with the NMPC guidance design, there is a gradual change in the heading angle attributed to the constraints on the system. Additionally, the changes in the heading angle from the backstepping guidance law result are sudden and direct.

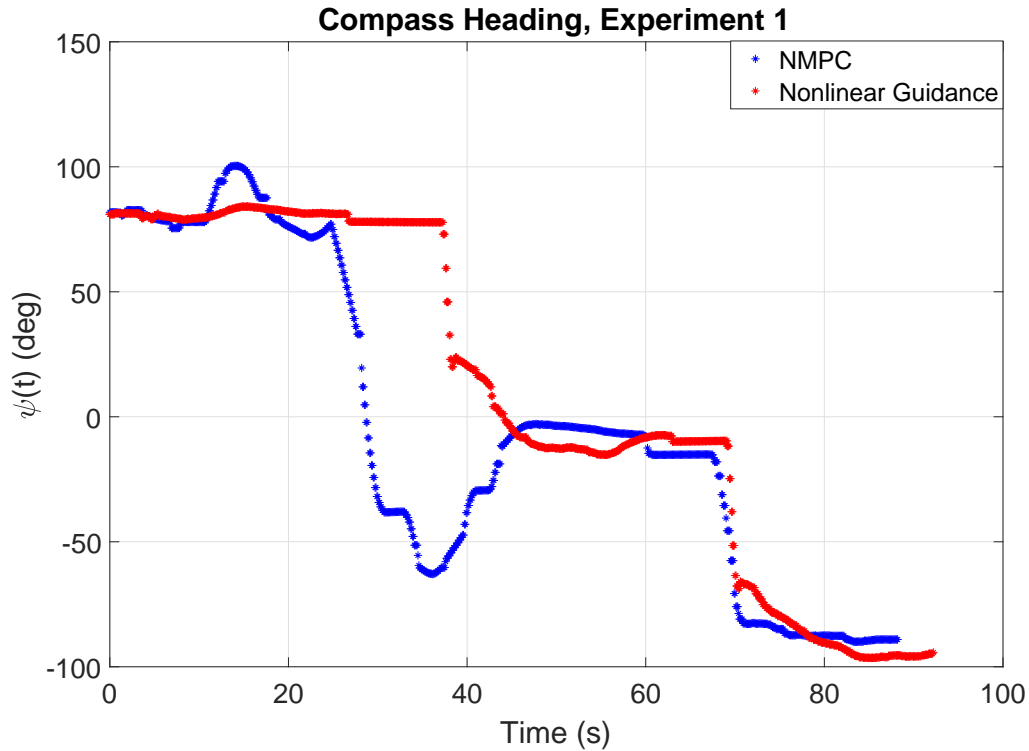


Figure 8.8: Robot’s heading angle for individual experiment 1.

Figure 8.9 shows the vehicle’s velocity for experiment 1. The velocity profiles using both guidance laws show interesting behavior. Both velocity profiles show gradual changes in the velocity of the vehicle, but the NMPC result has a few more obvious peaks compared to the backstepping design. The jumps in the vehicle’s velocity, with the NMPC result, occur as the vehicle is coming to a stop-and-go near the waypoint proximity limit. This behavior, with the NMPC guidance law, could be attributed to having the wheel speed commands not being sufficiently large to overcome the friction force on the vehicle and then jumping as the tracking errors accumulate to get back near the desired trajectory.

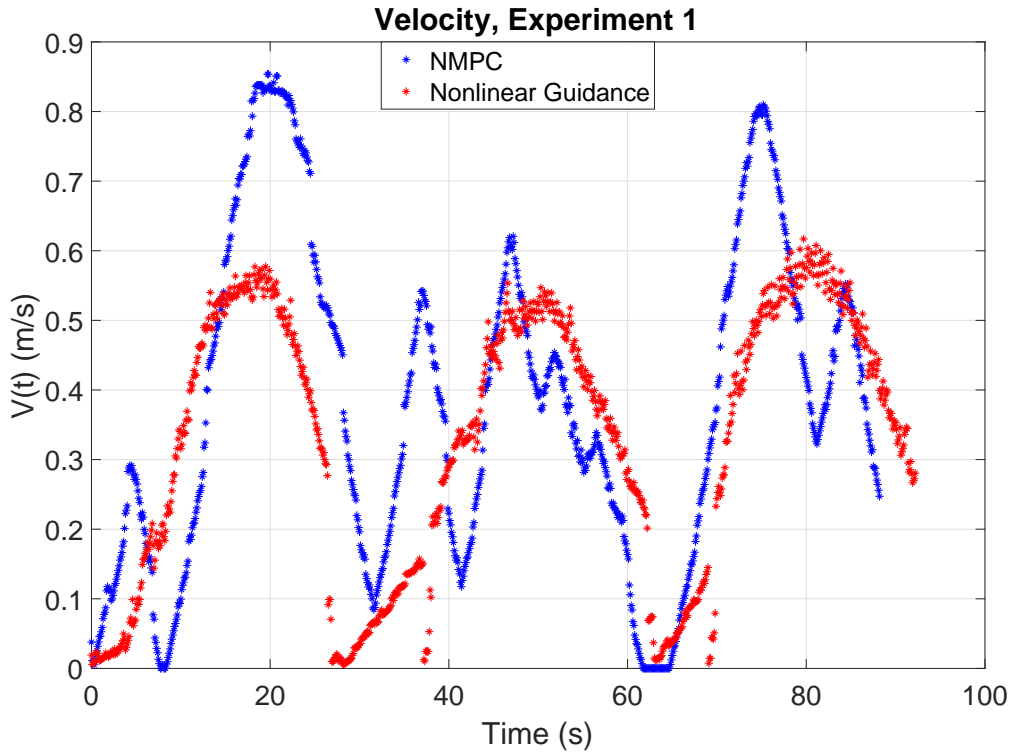


Figure 8.9: Robot’s velocity profile for individual experiment 1.

The commanded wheel speeds used for each guidance law are given in figure 8.10. One of the main differences with the two guidance commands can be seen in the results from figure 8.10. In this set of results, it can be seen that the back-stepping guidance law generates wheel speed commands that change suddenly and even exhibits some “chattering” behavior at some intervals. In contrast, the wheel speed commands coming from the NMPC design show smooth, gradual changes being applied to the vehicle to track the trajectory between the waypoints.



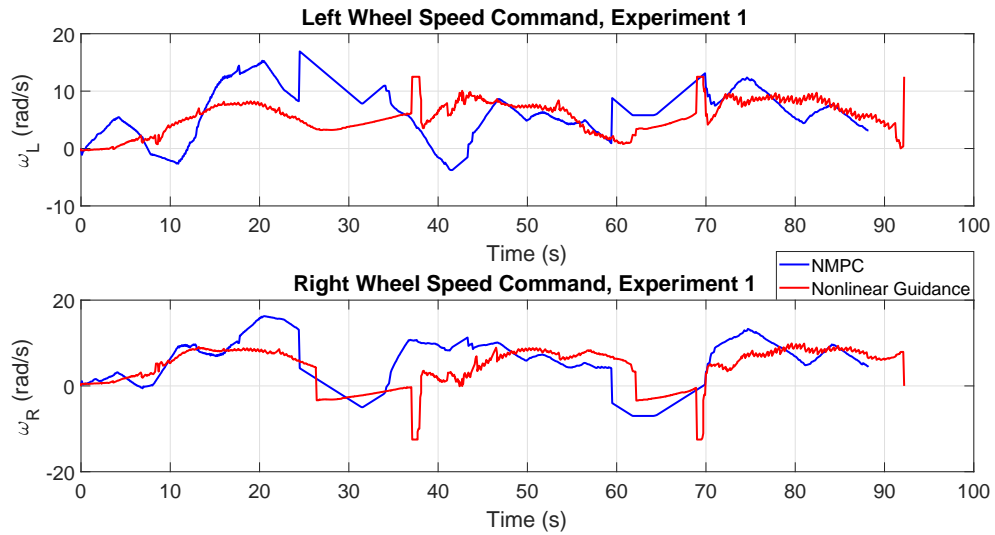


Figure 8.10: Robot's wheel speed commands for individual experiment 1.

### Individual Experiment 2

The waypoint setup for the second experiment using the guidance laws for an individual mobile robot is given in figure 8.11. For this experiment, the waypoints were chosen to give the vehicle a varying set of heading angles and distances to reach.

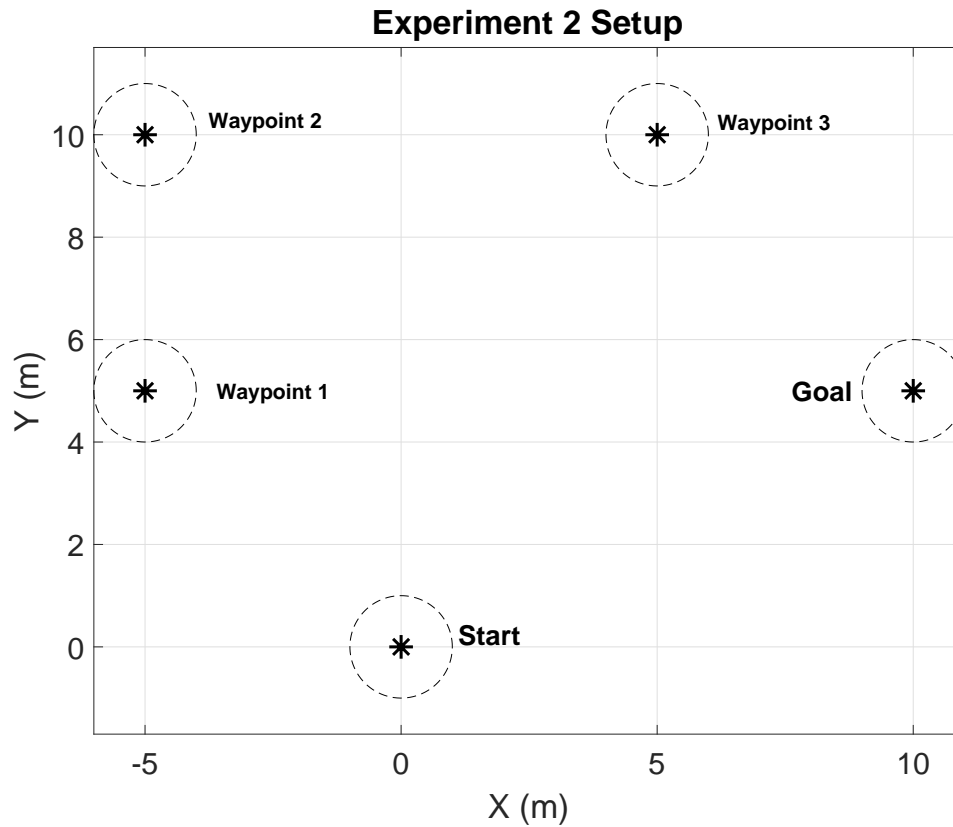


Figure 8.11: Waypoint setup for individual experiment 2.

The resulting trajectories of the mobile robot using both guidance laws for the second experiment is shown in figure 8.12. Same as the first experiment, the robot's resulting positions in figure 8.12 are given based on the GPS position measurements. The spacing between data points in the position data represents loss of information either due to communication delay or some other issue. With both guidance laws, the robot is able to reach each waypoint and arrives at the goal point at the end of the experiment.

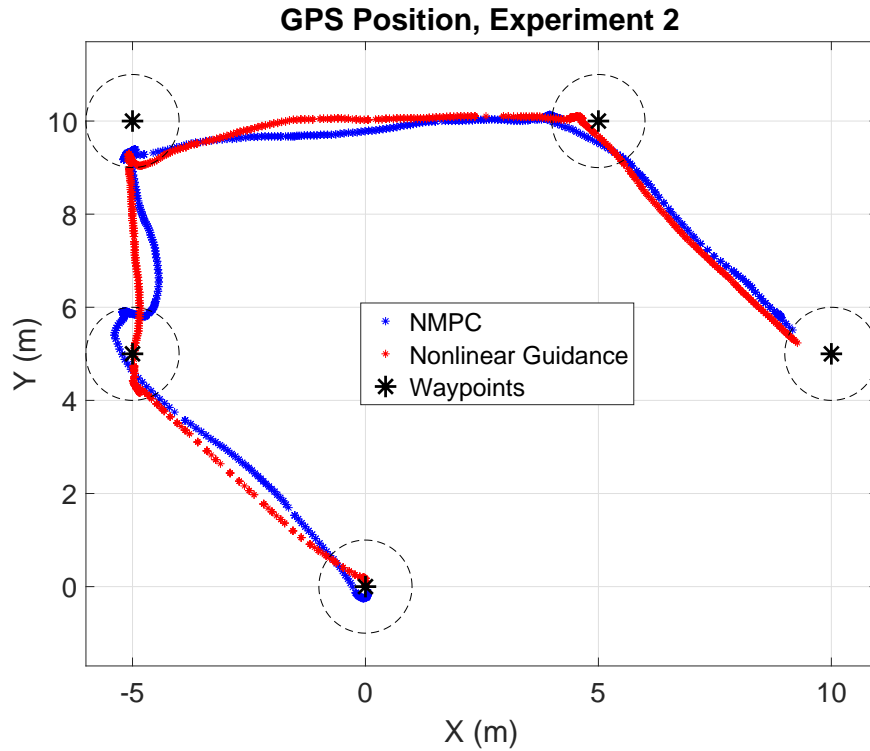


Figure 8.12: Robot’s waypoint traversal for individual experiment 2.

The result in figure 8.13 illustrates the heading angle of the vehicle as it moves between the waypoints. The behavior shown in the heading angle profiles in figure 8.13 is similar to what is seen in the results from experiment 1 in figure 8.8. There are sudden changes in the heading angle using the backstepping guidance law whereas the NMPC guidance law has gradual changes in the heading angle over the course of the experiment.

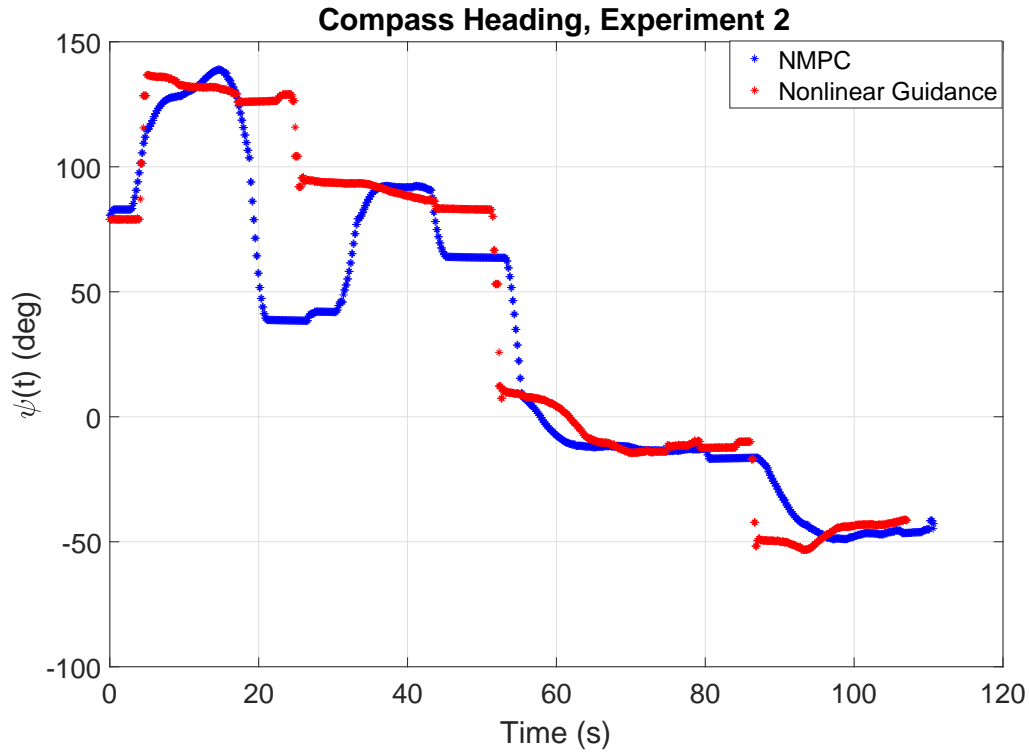


Figure 8.13: Robot’s heading angle for individual experiment 2.

Figure 8.14 shows the vehicle’s velocity for experiment 2. The behavior in figure 8.14 is similar to the vehicle’s velocities from experiment 1, in figure 8.9. In experiment 2, there are more waypoints, therefore, there are more peaks in the NMPC design resulting from the jumps in velocity as it exhibits the stop-and-go behavior near each waypoint.

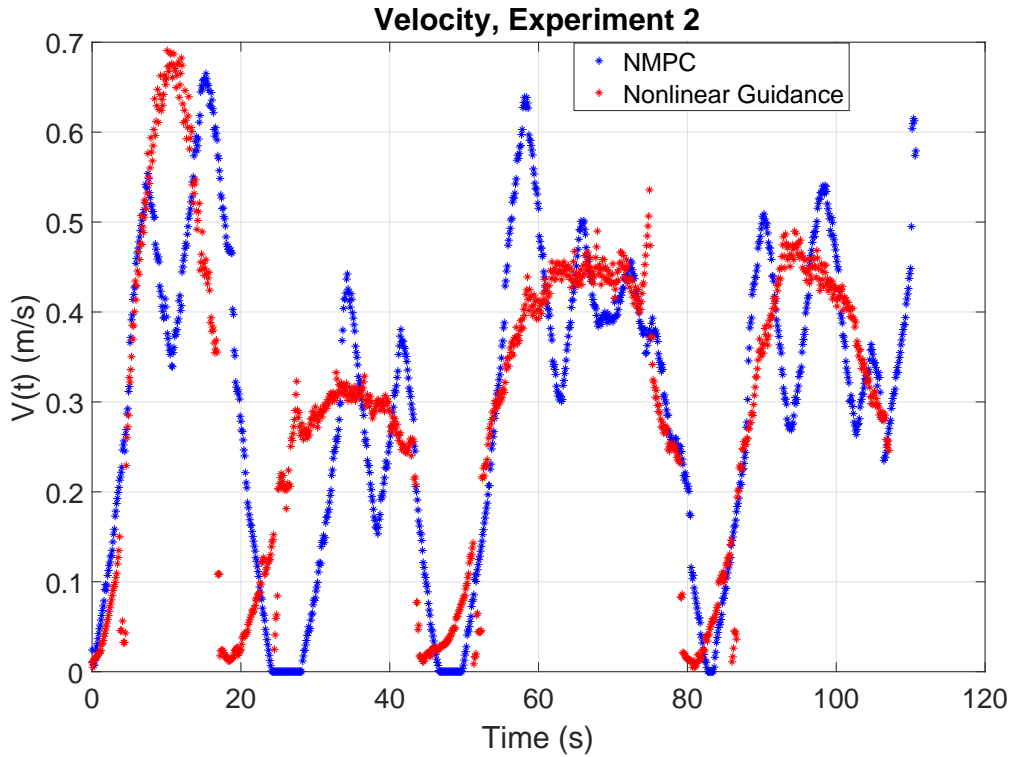


Figure 8.14: Robot’s velocity profile for individual experiment 2.

The commanded wheel speeds used for each guidance law are given in figure 8.15. The results in figure 8.15 also demonstrate the key difference using both guidance laws in that the wheel speed commands over the course of the experiment change smoothly using the NMPC design compared to sudden, large or chattering changes in the commands from the backstepping guidance law.

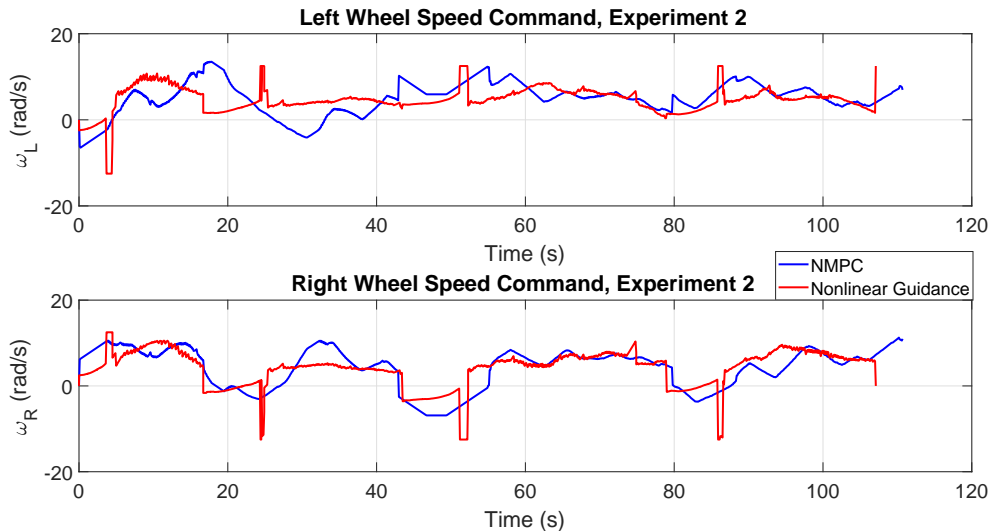


Figure 8.15: Robot’s wheel speed commands for individual experiment 2.

### 8.2.2 Multiple Robot Cooperative Control Results

For the cooperative control experiments, three different setups are used along with two mobile robot platforms to demonstrate the effectiveness of the cooperative control policy outlined in chapter 7. The first two results validate the aggregation tasks with the cooperative artificial potential functions. And the final result establishes the social foraging capability of the potential field generated by the navigation function algorithm from chapter 3.

For the three sets of results, there is an objective location given along with a proximity safety zone used to stop the experiment when one, or both, of the robots enters a desired “closeness” to the goal. For the first two experiments, the parameters for the cooperative potential function are set as  $a = 0.3$ ,  $b = 0.8$ , and  $c = 10$  for the gains in the attractive and repulsive components. And for all three experiments, the parameters for the repulsive potential function (from equation 7.14) are  $b_r = 1$  and  $c_r = 2$ .

## Cooperative Experiment 1

The setup for the first cooperative experiment is illustrated in figure 8.16. For this experiment, the goal position is located at (20m, 5m) in the local ENU frame. The maximum velocity constraint on each robot is  $v_{max} = 0.65 \text{ m/s}$  and the safety zone proximity is set at  $\epsilon_{prox} = 1.5 \text{ m}$ .

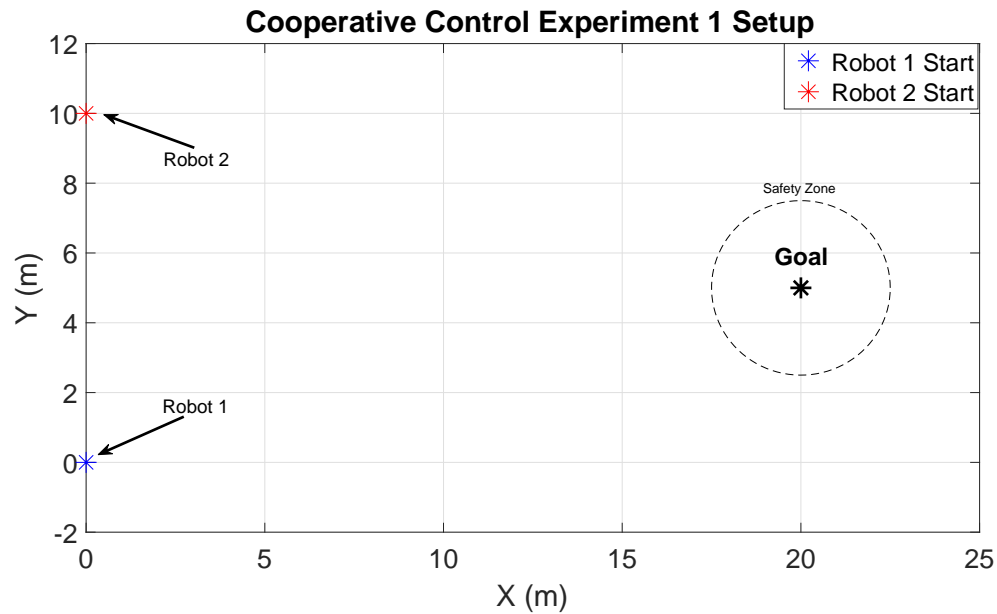


Figure 8.16: Setup for cooperative experiment 1.

In this experiment, the goal is known to both vehicles and the objective is to have them come together and aggregate near the final position. The resulting trajectories taken by the two robots is given in figure 8.17. The result has the robots coming together as each of them approach the objective until robot 2 enters the safety zone at the end of the experiment. The robots also avoid colliding with one another with the effect of the repulsive potential evident in the result.

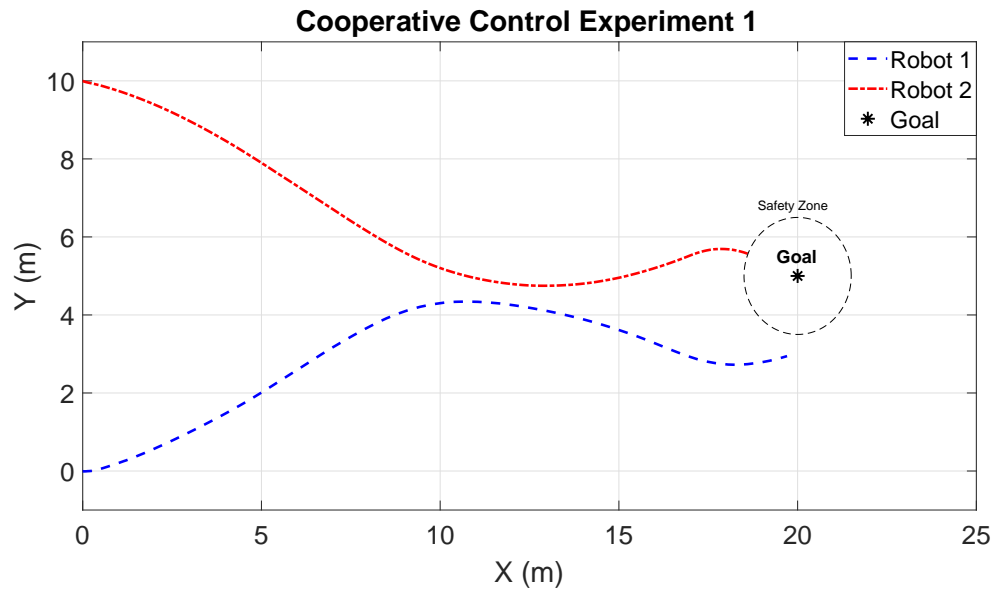


Figure 8.17: Robot positions for cooperative experiment 1.

The resulting wheel speed commands generated by the cooperative APF guidance method is depicted in figure 8.18. The wheel speed commands show the vehicles are maneuvering and are also constrained by the saturation involved with setting  $v_{max}$  within the APF method. While the velocities were constrained, the robots still have the capability to adjust their headings, as evident by their resulting trajectories in figure 8.17.



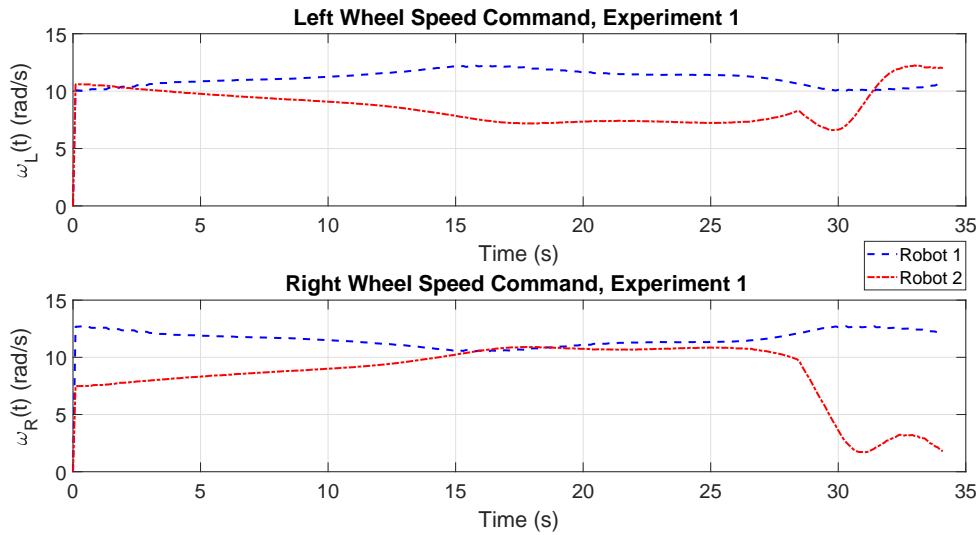


Figure 8.18: Wheel speed commands for cooperative experiment 1.

The relative distance between each of the vehicles is shown in figure 8.19. From this result, it can be seen that the two vehicles are approaching each other and come within close proximity toward the end of the experiment. While the vehicles do come within 1 meter of one another, they eventually repel each other using the repulsive potential function. Furthermore, this result demonstrates that the two vehicles do not collide over the course of the experiment.

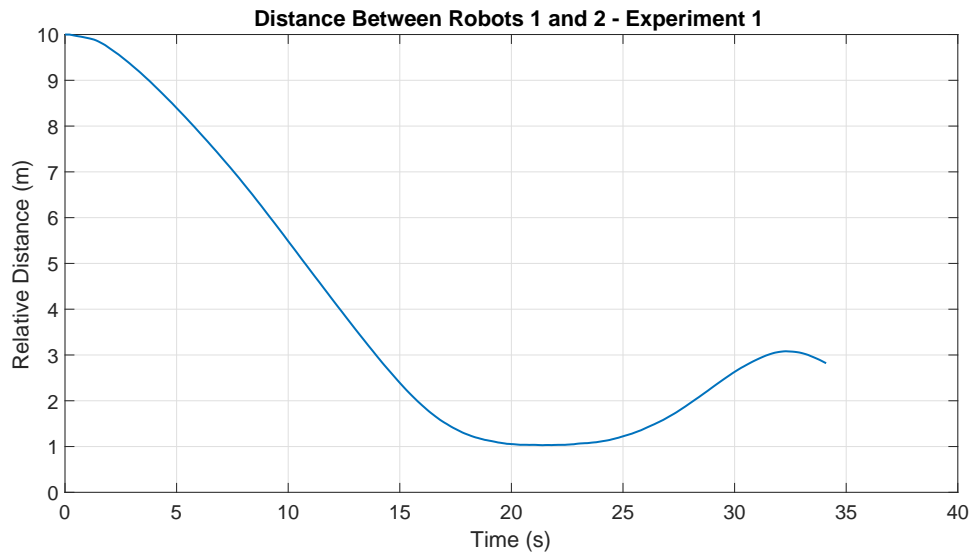


Figure 8.19: Robot's relative distance for cooperative experiment 1.

### Cooperative Experiment 2

The setup for the second cooperative experiment is illustrated in figure 8.20. For this experiment, the goal position is also located at (20m, 5m) in the local ENU frame. The maximum velocity constraint on each robot is  $v_{max} = 0.65 \text{ m/s}$  and the safety zone proximity is set at  $\epsilon_{prox} = 1.5 \text{ m}$ .

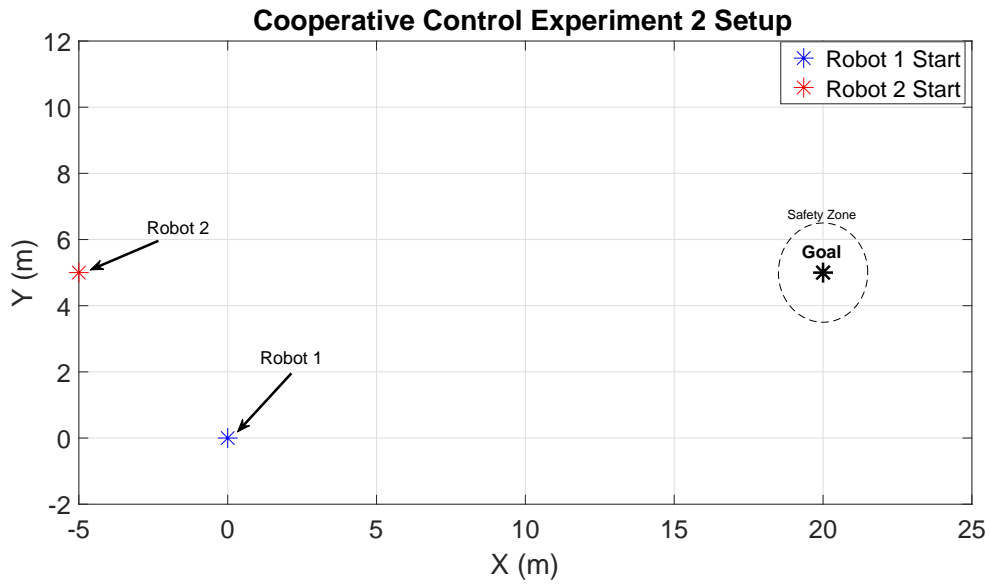


Figure 8.20: Setup for cooperative experiment 2.

In this experiment, the goal is known only to robot 1, while robot 2 only knows to follow, or cooperate with, robot 1. The objective is to have robot 1 go to the goal and have robot 2 follow; thus demonstrating that the two vehicles are in fact cooperating.

The resulting trajectories taken by the two robots is given in figure 8.21. This result depicts robot 1 moving directly to the goal and robot 2 is maneuvering to follow, as desired.

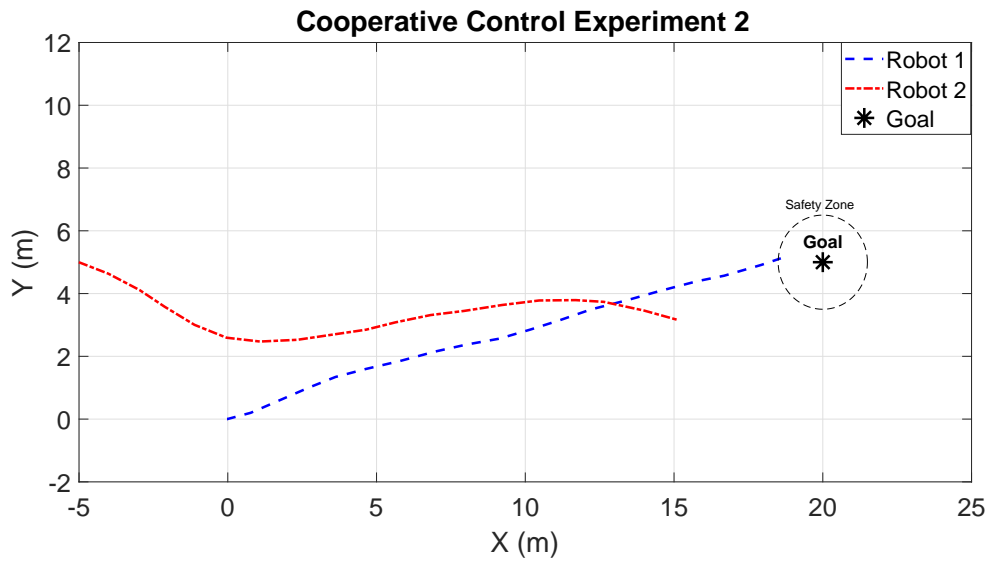


Figure 8.21: Robot positions for cooperative experiment 2.

The resulting wheel speed commands generated by the cooperative APF guidance method is depicted in figure 8.22. The results from figure 8.22 show that robot 1's wheel speed commands are smoothly directing it to the goal whereas the commands for robot 2 has the vehicle maneuvering to follow.

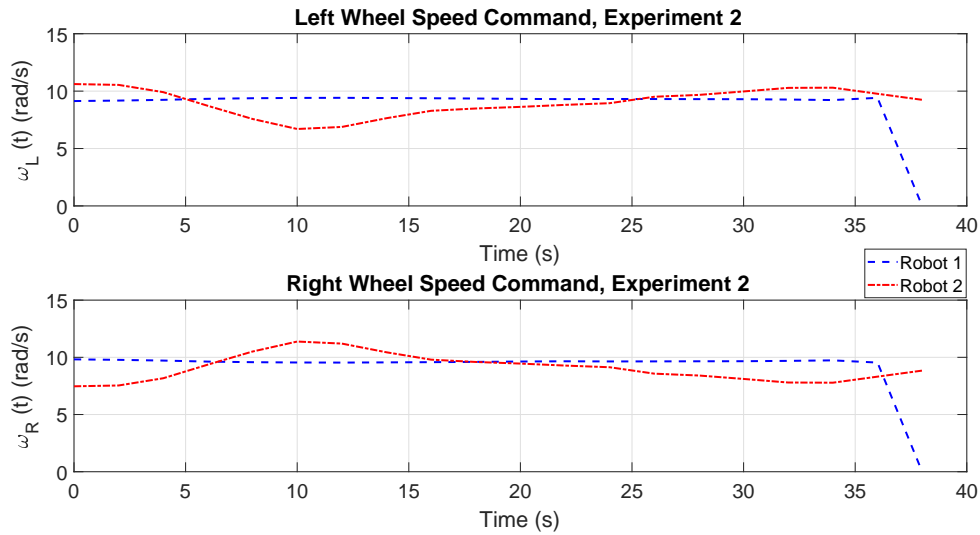


Figure 8.22: Wheel speed commands for cooperative experiment 2.

The relative distance between each of the vehicles is shown in figure 8.23. From this figure, it can be seen that the two vehicles do not collide. Also, the distance between the robots is shrinking suggesting that robot 2 is following and trying to close the gap.

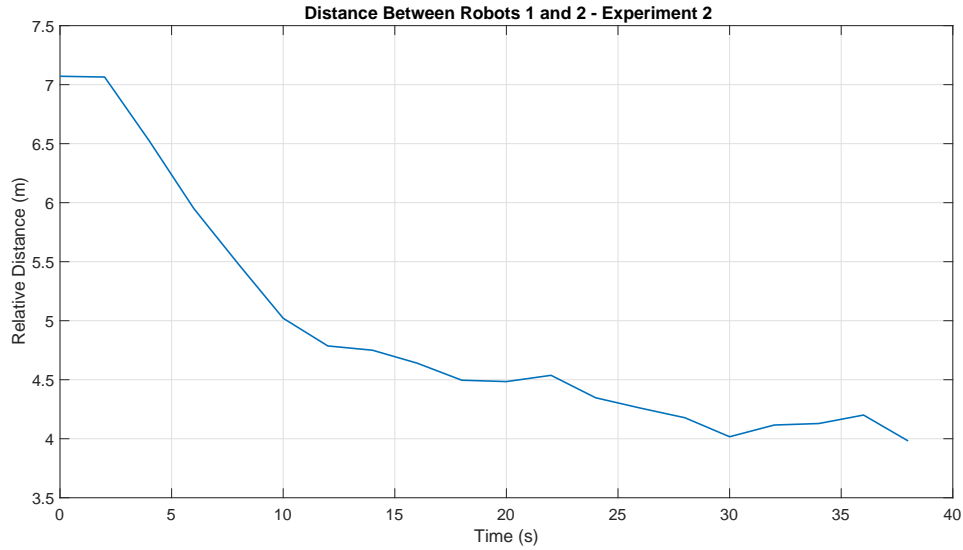


Figure 8.23: Robot's relative distance for cooperative experiment 2.

### Cooperative Experiment 3

The setup for the third cooperative experiment is illustrated in figure 8.24. For this experiment, the goal position is located at (20m, 10m) in the local ENU frame. The maximum velocity constraint on each robot is  $v_{max} = 0.60 \text{ m/s}$  and the safety zone proximity is set at  $\epsilon_{prox} = 2 \text{ m}$ .

In this experiment, there is an obstacle placed between the vehicles and the goal. The obstacle is placed at a distance of 9 m from the robots' starting positions and their objective is to avoid collisions with each other and the obstacle and arrive at the objective location. Also shown in figure 8.24 is the potential field generated by the navigation function algorithm to direct the robots to the goal. Notice that the constraints for  $u_{max}$  and  $d_{eff}$  for the navigation function were removed, this was done to enable the vehicles to find their way to the goal from anywhere in the free space.

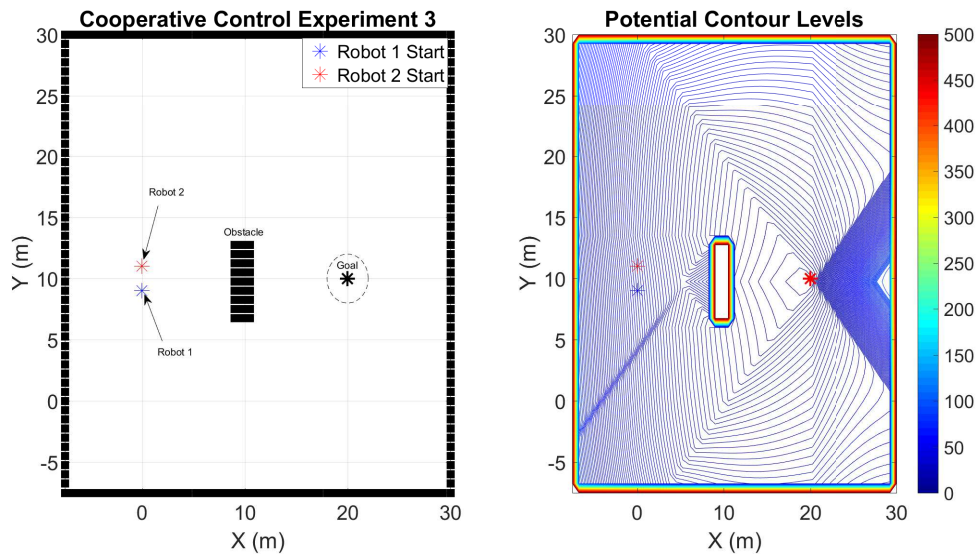


Figure 8.24: Setup for cooperative experiment 3.

The resulting trajectories taken by the two robots is given in figure 8.25. From this result, it is evident that both vehicles are repelled by each other at the start and gradually navigate around the obstacle. Robot 2 comes within close proximity to the corner obstacle. One drawback of this method is due to the cyber-physical system architecture employed, as there are inherent delays in the transmission of information over a 4G LTE mobile network. This leads to the delays in gathering positioning information and applying the guidance commands to the system. This issue also leads to close calls such as what is shown in figure 8.25.

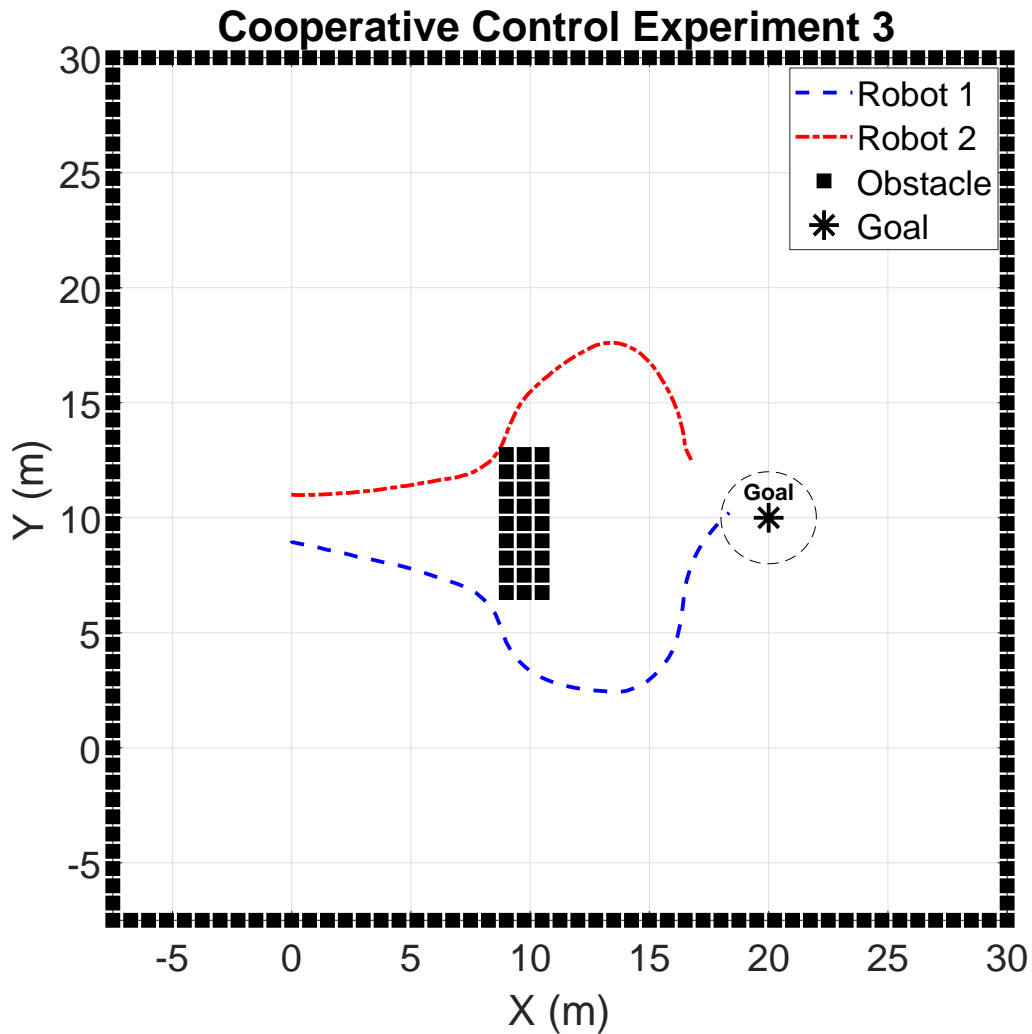


Figure 8.25: Robot positions for cooperative experiment 3.

The resulting wheel speed commands generated by the cooperative APF guidance method is depicted in figure 8.26. The behavior of the wheel speed command profiles for each robot suggest they are both maneuvering to both avoid colliding with each other as well as the obstacle present.



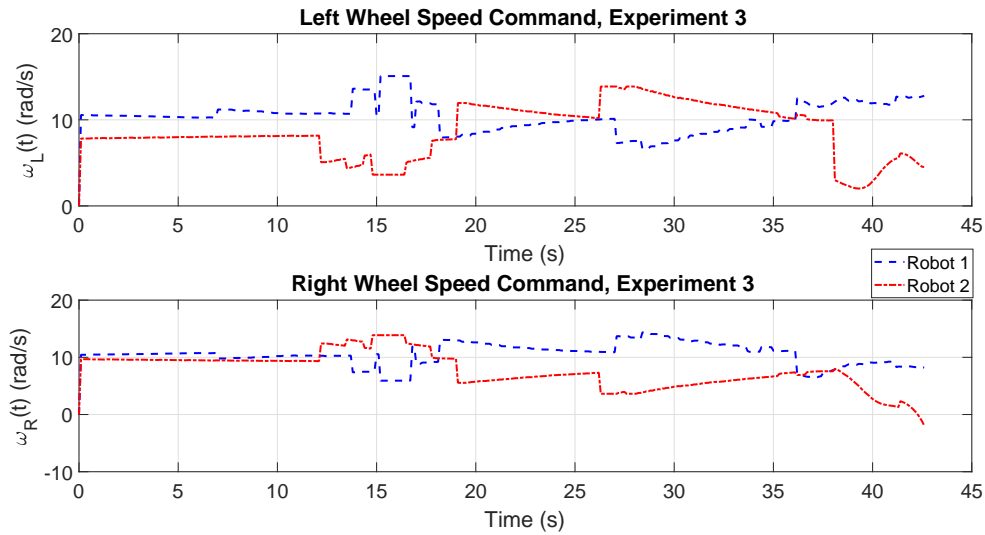


Figure 8.26: Wheel speed commands for cooperative experiment 3.

The relative distance between each of the vehicles is shown in figure 8.27. This result illustrates how the two robots start within a close proximity to one another and then gradually drift apart to go around the obstacle and eventually come back together and the end of the experiment.

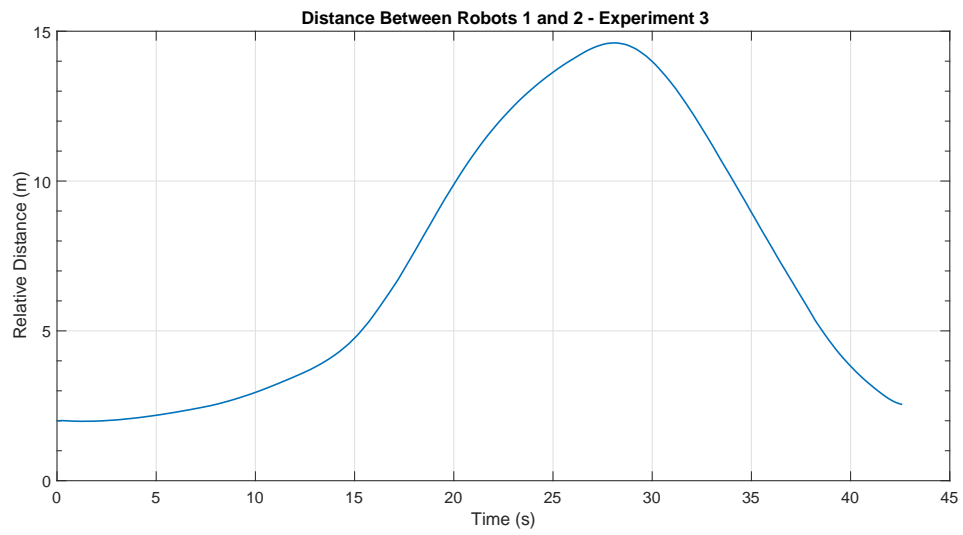


Figure 8.27: Robot's relative distance for cooperative experiment 3.

## CHAPTER 9

### Summary, Conclusions and Future Work

#### 9.1 Summary and Conclusions

The purpose of this research is to put forth methods to increase the level of autonomy for wheeled mobile robots. This dissertation presented procedures for developing a reliable path planning and guidance framework for mobile robots to accomplish this task. The framework discussed in this dissertation presents a new model based path planning algorithm in addition to two nonlinear guidance laws that form a reliable path planning and guidance framework for individual mobile robot vehicles which was verified in simulation and parts of which have also been validated experimentally using the mobile robot platforms in the Aerospace Systems Laboratory at The University of Texas at Arlington.

The path planning algorithm is introduced in this work as a design that is derived from a numerical navigation function. The novel contribution from the path planning algorithm is that it is a special form of artificial potential functions that is constructed based on the minimum control effort to a reachable state. This method makes use of the wavefront expansion construction procedure to form its potential contours with a control effort based metric. The end result of this path planning method is that it can find an obstacle free path that not only guides the robot to its objective but also takes into account the final approach of the vehicle to a reachable state. Additionally, this algorithm allows for obstacle free paths to be formed from anywhere in the environment, making it useful for cooperative tasks.

Additionally, extensions of the path planning algorithm's control effort metric were applied using modified versions of RRT\* and D\*. These extensions address the path planning problem with different kinds of uncertainty.

This dissertation also discusses two nonlinear guidance designs meant to track trajectories in a given environment. These guidance laws are derived to provide high-level commands governing the robot's velocity and heading angle as it tracks a trajectory. The first guidance design is a state dependent coefficient-based nonlinear model predictive control formulation. With this method, the robot's physical constraints are taken into account in finding its guidance commands to track the desired trajectory. Hence, this method provides a constraint-based design ensuring that the robot will operate within its capabilities. The contribution of this work lies in the method's use with a wheeled mobile robot vehicle that is verified in simulation and also applied to a real-time mobile robot platform for experiment results.

The next guidance law is based on a nonlinear backstepping-like approach. With this design, the contribution comes from proving that the robot is guaranteed to have bounded tracking errors as it traverses the trajectory leading to its goal. The ability of this guidance technique to closely follow the trajectory makes it a suitable choice when the robot needs extra assurance in tracking on its way to the goal. Also, this guidance law is verified in simulation and applied to a mobile robot platform for real-time experiment results.

The final contribution covered in this dissertation is the cooperative control formulation with artificial potential functions. This contribution arose when studying the capabilities of the novel navigation function algorithm. The significant aspect realized with regards to the navigation function path planner and performing cooperative tasks is that it has the capacity to form a path plan from any point within a known environment. This ability allows the potential field generated by the algorithm

to be used by any and all of the robots performing a cooperative task. Hence, the mobile robots can share their knowledge of the environment among themselves and form a single potential field for the collective which could potentially decrease the computational burden required. The contribution from this aspect of the research is the cooperative control policy outlined for aggregation and social foraging tasks with the minimum control effort navigation function algorithm. In this dissertation, the cooperative control policy is verified in simulation and also applied to two mobile robot platforms to provide real-time experiment results.

The combination of the different aspects described in this dissertation can form a framework that can be used for path planning and guidance involving a variety of mobile robot platforms. However, the main contributions covered in this dissertation lies in the application of the methods described to differential-drive wheeled mobile robots both in simulation and in real-time experiments.

## 9.2 Future Work

The research presented in this dissertation emphasizes analytical and experimental results for the path planning and guidance methods employed within the framework. The results demonstrate the effectiveness of each component within the framework individually and in combination through simulation and real-time experiments. However, there are still areas within the framework that can be expanded upon. One area that can be explored further is proving the optimality of the SDC-based NMPC guidance technique. The optimality of SDC-based control designs is at this time an open area of research with progress limited to specific applications.

Also, with regards to the path planning algorithms. The results were presented in a simulated environment and weren't rigorously tested on the real-time platform. Hence, one possible extension of the path-planning research would involve applying

the planners to a testing platform with virtual obstacles constraining the environment. Another extension would involve real-time planning of an environment with the path planning algorithms applied in tandem with other sensors, such as lidar or sonar, gathering information about obstacles present. These applications can also include comparisons of the different planning methods presented in this dissertation to determine which one works best under certain circumstances.

Possible extensions with the cooperative control policy outlined in this dissertation can include adding more vehicles into the given setups. The experiments covered in this dissertation were limited to two mobile robots, while the simulated results investigated three robots. Another possible extension of this work would be to remove the cyber-physical system architecture which constrains the robot's communication by way of the ground station computer. This would allow the cooperative control policy to be validated in a decentralized manner as it was presented in chapter 7.

APPENDIX A  
Preliminary Material

This appendix details some preliminary information and material which influenced the research presented in this dissertation. The first part will cover some notable definitions used. This is followed by considerations toward stability of dynamical systems and optimization techniques. Then, a brief overview of the wavefront expansion is given followed by a section that covers the design of trajectories considered in this research.

### A.1 Definitions

Define the 2-norm of a vector  $\mathbf{x} \in \mathfrak{R}^n$  as

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$$

A general configuration in a two dimensional workspace is denoted as the vector  $\mathbf{q} \in \mathfrak{R}^2$ . For the problems considered,  $\mathbf{q} = [x, y]^T$ , representing a two dimensional position vector composed of the  $x$  and  $y$  position coordinates in a grid based workspace [12].

Given a configuration  $\mathbf{q}$  in a two dimensional workspace, its 1-neighbors are defined as the configurations in the grid that have at most one coordinate differing from  $\mathbf{q}$ . The amount of difference is equal to one increment in either direction, or the cell difference in the grid, represented as  $\Delta x$  in the x-direction and  $\Delta y$  in the y-direction. For example, a configuration  $\mathbf{q}' = [x \pm \Delta x, y]^T$ , or  $\mathbf{q}' = [x, y \pm \Delta y]^T$ , is a 1-neighbor of  $\mathbf{q} = [x, y]^T$ . In total, there are four 1-neighbors in a two dimensional workspace [12].

Similarly, given a configuration  $\mathbf{q}$  in a two dimensional workspace, its 2-neighbors are defined as the configurations in the grid that have at most two coordinates differing from  $\mathbf{q}$ . There are eight 2-neighbors for a configuration when considering a two dimensional workspace [12].



A square  $n \times n$  matrix  $\mathbf{Q}$  is considered to be positive definite (p.d.) if given a vector  $\mathbf{x} \in \Re^n$ , where  $\mathbf{x} \neq 0$ , then the resulting inequality holds with a quadratic function

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0$$

Furthermore, the following holds true for a p.d. matrix  $\mathbf{Q}$

$$\mu_{min}(\mathbf{Q}) \|\mathbf{x}\|^2 \leq \mathbf{x}^T \mathbf{Q} \mathbf{x} \leq \mu_{max}(\mathbf{Q}) \|\mathbf{x}\|^2$$

where  $\mu_{min}(\cdot)$  is the minimum eigenvalue operator and  $\mu_{max}(\cdot)$  is the spectral radius (maximum eigenvalue) operator.

A continuous scalar function  $V : \Re^n \rightarrow \Re$  is said to be positive definite if  $V(\mathbf{0}) = 0$  and  $V(\mathbf{x}) > 0$  over the whole state space. Also, a continuous scalar function  $V(\mathbf{x})$  is considered negative definite if  $-V(\mathbf{x})$  is positive definite.

The fundamental or state transition matrix of a continuous system, denoted as  $\Phi(t, t_0) \in \Re^{n \times n}$ , is used to map the state at the initial time,  $\mathbf{x}(t_0) \in \Re^n$  to the current state,  $\mathbf{x}(t) \in \Re^n$ , as [70]:

$$\mathbf{x}(t) = \Phi(t, t_0) \mathbf{x}(t_0)$$

## A.2 Stability of Dynamical Systems

The concepts of stability in this dissertation are presented using Lyapunov stability theory. For a more complete analysis of Lyapunov stability theory, refer to references [67, 71]. In short, a point in the state space is considered stable if it starts in a nearby region, and stays nearby. Furthermore, the point is considered asymptotically stable if the state converges to an equilibrium point as time goes to infinity.

The following notation is adopted, as expressed in reference [67]. A spherical region denoted by  $\mathbf{B}_R$ , also called a ball, is defined by  $\|\mathbf{x}\| < \mathbf{B}_R$  in the state-space of the system.

Now, consider an autonomous system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \quad (\text{A.1})$$

where the function  $f : D \rightarrow \mathfrak{R}^n$  is a locally Lipschitz mapping from a subset  $D \subset \mathfrak{R}^n$  into  $\mathfrak{R}^n$ . Then, a state  $\mathbf{x} = \mathbf{0}$  is considered an equilibrium point if  $f(\mathbf{0}) = \mathbf{0}$ . And the equilibrium point is considered stable in the sense of Lyapunov if given some  $r$ , where  $0 < r < R$ ,

$$\|\mathbf{x}(0)\| < \mathbf{B}_r \implies \|\mathbf{x}(t)\| < \mathbf{B}_R, \forall t \geq 0$$

Additionally, an equilibrium point  $\mathbf{x} = \mathbf{0}$  is considered asymptotically stable if given some  $r > 0$ , then

$$\|\mathbf{x}(0)\| < \mathbf{B}_r \implies \mathbf{x}(t) \rightarrow \mathbf{0} \text{ as } t \rightarrow \infty$$

The objective with the stability proofs is to ultimately show that the system's energy is continuously dissipated and that it will eventually settle at an equilibrium point. Then, for a system, of the form in eq. (A.1), the concept of stability is considered with Lyapunov's direct method [67]. With this method, a scalar energy function,  $V : \mathfrak{R}^n \rightarrow \mathfrak{R}$ , is defined and is considered as a representation of the total energy of the system. The goal becomes to show that the function  $V(\mathbf{x})$  is a valid Lyapunov function, which will establish the asymptotic stability of the system.

Let  $V(\mathbf{x})$  be a non-negative function with its time derivative given by  $\dot{V}(\mathbf{x})$  along the state trajectories of system (A.1). Then, in order for the function to be considered a valid Lyapunov function, the following properties must be satisfied

- $V(\mathbf{x})$  is positive definite.
- $\dot{V}(\mathbf{x})$  is negative definite.

- $V(\mathbf{x}) = 0$ , if  $\mathbf{x}$  is an equilibrium point.

When these properties hold, the function  $V(\mathbf{x})$  is a Lyapunov function and the system is considered asymptotically stable [67].

### A.3 Optimization

Another important aspect of this research involves optimization. This is particularly important within the nonlinear model predictive control design outlined in chapter 5. The presentation and notation used in this section are based on references [29, 70]. Given an initial state vector  $\mathbf{x}_0$  and assuming a discrete system model of the form

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k \quad (\text{A.2})$$

where  $\mathbf{\Phi} \in \mathfrak{R}^{n \times n}$ ,  $\mathbf{\Gamma} \in \mathfrak{R}^{n \times m}$ ,  $\mathbf{x}_k \in \mathfrak{R}^n$  and  $\mathbf{u}_k \in \mathfrak{R}^m$  are the state transition matrix, input matrix, state vector and input vector respectively. Then, the optimization problems concerned in this dissertation are for finite horizon quadratic cost functions with a free terminal state, constrained optimization and semi-definite (specifically quadratic) programming.

#### A.3.1 System and Constraints in Batch Form

The discrete system model in equation (A.2) is placed in the predictive form (described in reference [29]):

$$\mathbf{x}_{k+j+1} = \mathbf{\Phi}\mathbf{x}_{k+j} + \mathbf{\Gamma}\mathbf{u}_{k+j} \quad (\text{A.3})$$

where the index  $k$  is the current time and the index  $j$  represents the time-distance from the current time. The input and state constraints are given by

$$\begin{aligned} \mathbf{u}_{lb} &\leq \mathbf{u}_{k+j} \leq \mathbf{u}_{ub}, & j = 0, 1, \dots, N-1 \\ \mathbf{g}_{lb} &\leq \mathbf{G}\mathbf{x}_{k+j} \leq \mathbf{g}_{ub}, & j = 0, 1, \dots, N \end{aligned} \quad (\text{A.4})$$

where the matrix  $\mathbf{G}$  is considered as an output matrix for the constrained states.

Also, the states in the closed interval  $[k, k + N]$  can be defined in batch form as

$$\mathbf{X}_k = \mathbf{F}\mathbf{x}_k + \mathbf{H}\mathbf{U}_k \quad (\text{A.5})$$

where  $\mathbf{X}_k$ ,  $\mathbf{F}$ ,  $\mathbf{H}$ ,  $\mathbf{U}_k$  are defined as:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \\ \vdots \\ \mathbf{x}_{k+N-1} \end{bmatrix}, \quad \mathbf{U}_k = \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_{k+1} \\ \vdots \\ \mathbf{u}_{k+N-1} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \mathbf{I} \\ \Phi \\ \vdots \\ \Phi^{N-1} \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{0} & & & & & \\ \Gamma & \mathbf{0} & & & & \\ \Phi\Gamma & \Gamma & \mathbf{0} & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ \Phi^{N-2}\Gamma & \Phi^{N-3}\Gamma & \dots & \Gamma & \mathbf{0} & \end{bmatrix}$$

And the terminal state is defined as

$$\mathbf{x}_{k+N} = \Phi^N \mathbf{x}_k + \bar{\Gamma} \mathbf{U}_k \quad (\text{A.6})$$

where

$$\bar{\Gamma} \triangleq \begin{bmatrix} \Phi^{N-1}\Gamma & \Phi^{N-2}\Gamma & \dots & \Phi\Gamma & \Gamma \end{bmatrix}$$

The constraints given in eq. (A.4) can also be placed in batch form as

$$\begin{bmatrix} \mathbf{u}_{lb} \\ \mathbf{u}_{ub} \\ \vdots \\ \mathbf{u}_{lb} \end{bmatrix} \leq \mathbf{U}_k \leq \begin{bmatrix} \mathbf{u}_{ub} \\ \mathbf{u}_{ub} \\ \vdots \\ \mathbf{u}_{ub} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{g}_{lb} \\ \mathbf{g}_{lb} \\ \vdots \\ \mathbf{g}_{lb} \end{bmatrix} \leq \bar{\mathbf{G}}_N (\mathbf{F}\mathbf{x}_k + \mathbf{H}\mathbf{U}_k) \leq \begin{bmatrix} \mathbf{g}_{ub} \\ \mathbf{g}_{ub} \\ \vdots \\ \mathbf{g}_{ub} \end{bmatrix} \quad (\text{A.7})$$

where  $\bar{\mathbf{G}}_N = \text{diag}\{G, \dots, G\}$  is a block diagonal matrix formed by the output matrix  $\mathbf{G}$ .

### A.3.2 Constrained Finite Horizon, Quadratic Cost Regulator with Free Terminal State

The quadratic cost function for the constrained optimization problem is chosen as

$$J(\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{2} \sum_{j=0}^{N-1} (\mathbf{x}_{k+j}^T \mathbf{Q} \mathbf{x}_{k+j} + \mathbf{u}_{k+j}^T \mathbf{R} \mathbf{u}_{k+j}) + \frac{1}{2} \mathbf{x}_{k+N}^T \mathbf{Q}_f \mathbf{x}_{k+N} \quad (\text{A.8})$$

with the constraints given in batch form in equation (A.7).

The cost function in equation (A.8) can be separated into two parts as

$$J(\mathbf{x}_k, \mathbf{u}_k) = J_1(\mathbf{x}_k, k) + J_2(\mathbf{x}_{k+N}, k) \quad (\text{A.9})$$

where

$$J_1(\mathbf{x}_k, k) \triangleq \frac{1}{2} \sum_{j=0}^{N-1} (\mathbf{x}_{k+j}^T \mathbf{Q} \mathbf{x}_{k+j} + \mathbf{u}_{k+j}^T \mathbf{R} \mathbf{u}_{k+j}) \quad (\text{A.10})$$

$$J_2(\mathbf{x}_{k+N}, k) \triangleq \frac{1}{2} \mathbf{x}_{k+N}^T \mathbf{Q}_f \mathbf{x}_{k+N} \quad (\text{A.11})$$

The constrained control is obtained by solving the above optimization problem at time  $k$  and repeating it at the next time  $k + 1$ .

This problem can be reduced to the following semi-definite programming problem [29]:

$$\mathbf{U}_k^* = \min \gamma_1 + \gamma_2 \quad (\text{A.12})$$

where  $J_1(\mathbf{x}_k, k) \leq \gamma_1$  and  $J_2(\mathbf{x}_{k+N}, k) \leq \gamma_2$ , and is subject to

$$\begin{bmatrix} \gamma_1 - 2\mathbf{x}_k^T \mathbf{F}^T \bar{\mathbf{Q}}_N \mathbf{H} \mathbf{U}_k - \mathbf{x}_k^T \mathbf{F}^T \bar{\mathbf{Q}}_N \mathbf{F} \mathbf{x}_k & \mathbf{U}_k^T \\ \mathbf{U}_k & (\mathbf{H}^T \bar{\mathbf{Q}}_N \mathbf{H} + \bar{\mathbf{R}}_N)^{-1} \end{bmatrix} \geq 0 \quad (\text{A.13})$$

$$\begin{bmatrix} \gamma_2 & [\Phi^N \mathbf{x}_k + \bar{\Gamma} \mathbf{U}_k]^T \\ \Phi^N \mathbf{x}_k + \bar{\Gamma} \mathbf{U}_k & \mathbf{Q}_f^{-1} \end{bmatrix} \geq 0 \quad (\text{A.14})$$

where  $\bar{\mathbf{Q}}_N = \text{diag}\{\mathbf{Q}, \dots, \mathbf{Q}\}$  and  $\bar{\mathbf{R}}_N = \text{diag}\{\mathbf{R}, \dots, \mathbf{R}\}$  are block diagonal matrices consisting of the state and input weighting matrices respectively. The op-

timal control  $\mathbf{U}_k^*$  can be obtained by solving eq. (A.12) subject to the linear matrix inequalities in eqs. (A.13) and (A.14).

### A.3.3 Linear Matrix Inequalities

The guidance input synthesis involved in this research makes use of linear matrix inequalities to show stability and feasibility. A matrix  $\mathbf{F}(\mathbf{x})$  is considered a linear matrix inequality (LMI) if it has the following form

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^m x_i \mathbf{F}_i > \mathbf{0} \quad (\text{A.15})$$

where the variable vector  $\mathbf{x} = [x_1, \dots, x_m]^T \in \Re^m$  and the symmetric matrices  $\mathbf{F}_i \in \Re^{n \times n}$ , for  $i = 0, \dots, m$  are given.

A LMI of the form in Eq. (A.15) is considered convex on  $\mathbf{x}$ , in other words, the set defined by  $\{\mathbf{x} | \mathbf{F}(\mathbf{x}) > \mathbf{0}\}$  is convex. The LMI form can be used to represent nonlinear matrix inequalities such as quadratic inequalities.

In fact, nonlinear matrix inequalities can be represented as LMI's using the Schur complement. For example, consider the LMI given by

$$\begin{bmatrix} \mathbf{Q}(\mathbf{x}) & \mathbf{S}(\mathbf{x}) \\ \mathbf{S}(\mathbf{x})^T & \mathbf{R}(\mathbf{x}) \end{bmatrix} > \mathbf{0} \quad (\text{A.16})$$

where  $\mathbf{Q}(\mathbf{x})$ ,  $\mathbf{R}(\mathbf{x})$  are symmetric and  $\mathbf{Q}(\mathbf{x})$ ,  $\mathbf{R}(\mathbf{x})$ , and  $\mathbf{S}(\mathbf{x})$  each depend affinely on  $\mathbf{x}$ . Then, the LMI is equivalent to the following matrix inequalities using the Shur compliment of  $\mathbf{R}(\mathbf{x})$  and  $\mathbf{Q}(\mathbf{x})$  respectively

$$\mathbf{R}(\mathbf{x}) > \mathbf{0}, \quad \mathbf{Q}(\mathbf{x}) - \mathbf{S}(\mathbf{x})\mathbf{R}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^T > \mathbf{0}. \quad (\text{A.17})$$

and

$$\mathbf{Q}(\mathbf{x}) > \mathbf{0}, \quad \mathbf{R}(\mathbf{x}) - \mathbf{S}(\mathbf{x})^T\mathbf{Q}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x}) > \mathbf{0}. \quad (\text{A.18})$$

### A.3.4 Semidefinite Programming

A short introduction to semidefinite programming (SDP) is given here since it is used to obtain results in this dissertation, shown in chapter 5. The SDP discussion here is based on information available in reference [29].

SDP is an optimization routine used to minimize a linear function of a variable  $\mathbf{x} \in \Re^m$  subject to a matrix inequality. Consider the following optimization problem:

$$\text{minimize } \mathbf{c}^T \mathbf{x}$$

$$\text{subject to: } \mathbf{F}(\mathbf{x}) > 0$$

where  $\mathbf{c} \in \Re^m$  and  $\mathbf{F}(\mathbf{x})$  is a LMI.

The SDP problem can also be used to represent other important optimization problems, such as a quadratic programming (QP) problem, which will be used in this proposal. Consider a QP problem given by

$$\text{minimize } \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{q}^T \mathbf{x} + r \tag{A.19}$$

$$\text{subject to: } \mathbf{G} \mathbf{x} < \mathbf{h}$$

where  $\mathbf{G} \in \Re^{m \times m}$ ,  $\mathbf{P} \in \Re^{m \times m}$ ,  $\mathbf{P} > 0$ ,  $\mathbf{q} \in \Re^m$ ,  $\mathbf{x} \in \Re^m$ , and  $r \in \Re$ . The QP problem in eq. (A.19) can be equivalently represented in SDP form as

$$\text{minimize } t \tag{A.20}$$

$$\text{subject to: } \begin{bmatrix} t - r - \mathbf{q}^T \mathbf{x} & \mathbf{x}^T \\ \mathbf{x} & 2\mathbf{P}^{-1} \end{bmatrix} > 0$$

$$\mathbf{G} \mathbf{x} < \mathbf{h}$$

where  $t \in \Re$ . This problem involves the QP subject to linear constraints. However, in order to solve the kinds of optimization problems required, the QP problem needs

to be subject to a quadratic constraint. For example, consider the following convex quadratic constraint

$$(\mathbf{Ax} + \mathbf{b})^T(\mathbf{Ax} + \mathbf{b}) - \mathbf{c}^T\mathbf{x} - \mathbf{d} < 0$$

which can be rewritten in the following form

$$\begin{bmatrix} \mathbf{I} & \mathbf{Ax} + \mathbf{b} \\ (\mathbf{Ax} + \mathbf{b})^T & \mathbf{c}^T\mathbf{x} + \mathbf{d} \end{bmatrix} < \mathbf{0} \quad (\text{A.21})$$

Note that the left hand side in eq. (A.21) depends afinely on  $\mathbf{x}$ . Furthermore, the inequality in eq. (A.21) can be placed in the LMI form of eq. (A.15), where values for  $\mathbf{F}_0$  and  $\mathbf{F}_i$  can be defined as

$$F_0 = \begin{bmatrix} \mathbf{I} & \mathbf{b} \\ \mathbf{b}^T & \mathbf{d} \end{bmatrix}, \quad \mathbf{F}_i = \begin{bmatrix} \mathbf{0} & \mathbf{a}_i \\ \mathbf{a}_i^T & \mathbf{c}_i \end{bmatrix}, \quad i = 1, \dots, m \quad (\text{A.22})$$

where  $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_m]$  [29].

#### A.4 Numerical Navigation Functions

The numerical navigation functions which influence this research are described fully in references [12,13]. The two main numerical navigation functions are the simple wavefront expansion algorithm and the improved wavefront expansion algorithm [12, 13]. Both methods utilize counting and logic to construct the navigation function's potential contours. The simple wavefront expansion algorithm is the main influence of the work presented in this dissertation and is briefly described in this appendix.

The first step for the wavefront expansion algorithm is to discretize the workspace into an evenly spaced grid. After the grid representation of the workspace is formed, there are three main steps to be performed. The steps for the wavefront expansion algorithm are:



1. Identify the free configurations.
2. Set the potential at the goal to zero.
3. Expand the potential within the free space.

The free space is identified by forming a bitmap over the grid where the free configurations are represented by 0 and the restricted configurations are represented by 1. Then, the potential at the goal configuration, denoted as  $\mathbf{q}_{goal}$  is set to 0. This is done to ensure that it will be the global minimum in the potential field.

After the potential at the goal is set, the potential for all the 1-neighbors are set to 1; and to 2 at every 1-neighbor of these configurations; etc. This process is repeated until it has visited every free configuration reachable from  $\mathbf{q}_{goal}$ . An example result of the wavefront expansion algorithm is illustrated in figure A.1 and the full algorithm is shown in the flowchart in figure A.2.



Figure A.1: Example result using the traditional wavefront expansion algorithm.

The result in figure A.1 shows that the potential contour levels are formed by a metric defined by the distance from the goal. Also, the algorithm shown in figure

A.2 shows that after the navigation function’s potential field is formed, it is sent to a best-first search algorithm to find the reference path from a given initial position.

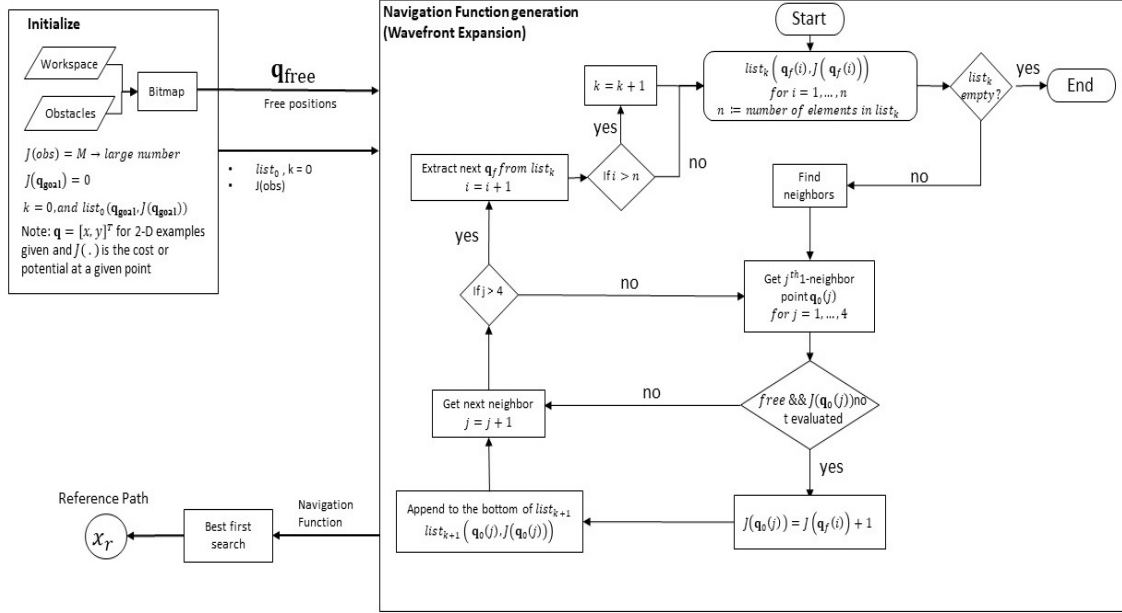


Figure A.2: Traditional wavefront expansion algorithm.

The reference path,  $\mathbf{x}_r$ , is obtained by following the negative gradient of the resulting potential field from a given initial location to the objective location. Through the best-first search method, the paths are attained by observing the potential values of the neighboring points and then choosing the neighbor with the lowest value. This is done in an iterative manner until the objective is found. Hence, the path is found by starting at some initial location in the environment and continuing along a path of minimal potential until it reaches the goal. The best-first graph search algorithm is fully described in reference [12]. In short, the search algorithm creates a tree of nodes from the grid that when followed leads to the objective position.

## A.5 Trajectory Generation

This section describes two different approaches for designing a trajectory for a mobile robot to track. The first approach is to design a trajectory that is fitted along a reference path, or set of  $x$  and  $y$  coordinates, leading to a goal. The second approach is to define a trajectory between two waypoints.

### A.5.1 Trajectory Design Along a Path Plan

For the guidance designs introduced, there needs to be a two dimensional,  $\mathcal{C}^2$ , trajectory to track. The reference signals needed for the trajectory tracking controller are  $[x_r, y_r, \dot{x}_r, \dot{y}_r, \ddot{x}_r, \ddot{y}_r]$ , which represent the desired position, velocity and acceleration in the  $x$  and  $y$  directions. The trajectory is designed to fit over the reference path,  $\mathbf{x}_r$ , that is generated through the navigation function algorithm described in chapter 3.

In order to generate the trajectory, a desired time to reach the goal needs to be set. With the goal time set, a time stamp to reach each point can be assigned by creating a time vector that can be evenly spaced across all the points in the reference path. In other words the time step,  $\delta t$ , it takes to go from one point to the next along the path is held constant.

Then, the reference velocities along the path can be computed by

$$\begin{aligned}\dot{x}_r(i+1) &= \frac{x_r(i+1) - x_r(i)}{\delta t}, \\ \dot{y}_r(i+1) &= \frac{y_r(i+1) - y_r(i)}{\delta t},\end{aligned}$$

where  $i$  is the index number of the point in the path. And it is assumed that the initial velocity is zero, i.e.  $\dot{x}_r(0) = 0$  and  $\dot{y}_r(0) = 0$ . The reference acceleration signals are computed by

$$\begin{aligned}\ddot{x}_r(i+1) &= \frac{\dot{x}_r(i+1) - \dot{x}_r(i)}{\delta t}, \\ \ddot{y}_r(i+1) &= \frac{\dot{y}_r(i+1) - \dot{y}_r(i)}{\delta t},\end{aligned}$$

where  $i$  is the index number and it is assumed that the initial acceleration is zero, i.e.  $\ddot{x}_r(0) = 0$  and  $\ddot{y}_r(0) = 0$ . Finally, when the vehicle is traveling between points in  $\mathbf{x}_r$ , the references are found by linear interpolation.

For the NMPC design, the reference heading angle and velocity can be found by

$$\begin{aligned}\psi_r(i+1) &= \arctan\left(\frac{\dot{y}_r(i+1)}{\dot{x}_r(i+1)}\right) \\ v_r(i+1) &= \sqrt{\dot{x}_r^2(i+1) + \dot{y}_r^2(i+1)}\end{aligned}$$

#### A.5.2 Trajectory Design Between Two Waypoints

For the results in this dissertation, a constrained acceleration design is used to define the trajectory between two waypoints. Hence, a 5<sup>th</sup> order polynomial is considered, which will provide sufficient boundary conditions to solve for reference position, velocity and acceleration profiles in the  $x$  and  $y$  directions.

The boundary conditions on the position, velocity and acceleration of the robot can be used to define the coefficients of the polynomial in terms of the final time,  $t_f$ . Also, a constraint on the maximum magnitude of the acceleration is used to find the final time to reach the next waypoint as well as the coefficients of the trajectory polynomial.

The cost function associated with the minimum jerk trajectory problem is given by

$$J(\bar{\mathbf{X}}(t)) = \frac{1}{2} \int_{t_0}^{t_f} \left[ \left( \frac{d^3x(t)}{dt^3} \right)^2 + \left( \frac{d^3y(t)}{dt^3} \right)^2 \right] dt, \quad (\text{A.23})$$

where  $\bar{\mathbf{X}}(t) = [x(t) \ y(t)]^T$  is a two dimensional position vector. Through minimizing the above cost function, the trajectory profiles for the position, velocity and acceleration can be obtained. Furthermore, the coefficients of the polynomials generated can be found by considering the constraints on the magnitude of the maximum accelerations of the system, i.e.  $|\ddot{x}(t)| \leq |\ddot{x}_{max}|$  and  $|\ddot{y}(t)| \leq |\ddot{y}_{max}|$ .

To define the resulting trajectory, the position, velocity and acceleration profiles are given in terms of a generalized coordinate  $q(t)$  as

$$q(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5, \quad (\text{A.24})$$

$$\dot{q}(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4, \quad (\text{A.25})$$

$$\ddot{q}(t) = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3, \quad (\text{A.26})$$

where  $q(t)$ , in general, can represent  $x(t)$  or  $y(t)$ . Then, the boundary conditions on the position, velocity and acceleration of the robot are given by

$$q(0) = q_i, \quad (\text{A.27})$$

$$q(t_f) = q_f, \quad (\text{A.28})$$

$$\dot{q}(0) = \dot{q}(t_f) = 0, \quad (\text{A.29})$$

$$\ddot{q}(0) = \ddot{q}(t_f) = 0, \quad (\text{A.30})$$

where  $t_f$  denotes the final time. Substituting the boundary conditions into equations A.24, A.25 and A.26, the following relations for the polynomial coefficients are obtained

$$a_0 = q(0), \quad (\text{A.31})$$

$$a_1 = 0, \quad (\text{A.32})$$

$$a_2 = 0, \quad (\text{A.33})$$

$$q_f - q_i = a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5, \quad (\text{A.34})$$

$$0 = 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4, \quad (\text{A.35})$$

$$0 = 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3. \quad (\text{A.36})$$

And through solving equations A.34, A.35 and A.36 simultaneously, the coefficients,  $a_3, a_4, a_5$ , are obtained in terms of the boundary conditions and the final time,  $t_f$ , as

$$a_3 = 10 \left( \frac{\Delta q}{t_f^3} \right), \quad (\text{A.37})$$

$$a_4 = -15 \left( \frac{\Delta q}{t_f^4} \right), \quad (\text{A.38})$$

$$a_5 = 6 \left( \frac{\Delta q}{t_f^5} \right), \quad (\text{A.39})$$

where  $\Delta q = q_f - q_i$ . Then, the final time for the maneuver can be determined by considering the following inequality with the maximum acceleration constraint

$$|\ddot{q}(t)| \leq |a_{max}|, \quad (\text{A.40})$$

This leads to the following necessary condition on the jerk of the system,

$$\left. \frac{d(\ddot{q}(t))}{d(t)} \right|_{\ddot{q}(t)=a_{max}} = 0. \quad (\text{A.41})$$

By defining  $\tau = \frac{t}{t_f}$  and using equations A.26, and A.37-A.40, the following inequality in terms of the boundary conditions, current time, final time and maximum acceleration is obtained

$$\left| 60 \left( \frac{\Delta q}{t_f^2} \right) [\tau(1 - 3\tau + 2\tau^2)] \right| \leq |a_{max}|. \quad (\text{A.42})$$

The inequality in equation A.42 is then solved using the necessary condition to find the value for  $\tau$  as,

$$\tau = \frac{1}{2} \pm \frac{1}{\sqrt{12}}. \quad (\text{A.43})$$

By using this result for  $\tau$  along with the inequality in equation A.42, the final time  $t_f$  is obtained in terms of the maximum acceleration of the robot and the boundary conditions as

$$t_f \geq \sqrt{\frac{10}{\sqrt{3}} \frac{\Delta q}{a_{max}}}. \quad (\text{A.44})$$

Using the result for  $t_f$  in equation A.44, the values for  $a_3$ ,  $a_4$  and  $a_5$  can be obtained. Overall, the minimum jerk trajectory design will provide not just reference profiles for the position, velocity and acceleration, but also a solution for the expected time it would take for a vehicle to reach a point based on its maximum acceleration.

## REFERENCES

- [1] B. T. Clough, “Metrics, schmetrics! how the heck do you determine a uav’s autonomy anyway,” Air Force Research Lab, Wright Patterson AFB, Tech. Rep., 2002, <http://www.dtic.mil/dtic/tr/fulltext/u2/a515926.pdf>.
- [2] T. Huntsberger, H. Aghazarian, Y. Cheng, E. T. Baumgartner, E. Tunstel, C. Leger, A. Trebi-Ollennu, and P. S. Schenker, “Rover autonomy for long range navigation and science data acquisition on planetary surfaces,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 3. IEEE, 2002, pp. 3161–3168, doi: 10.1109/ROBOT.2002.1013713.
- [3] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone, and J. Yen, “Risk-aware planetary rover operation: Autonomous terrain classification and path planning,” in *2015 IEEE Aerospace Conference*. IEEE, 2015, pp. 1–10, doi: 10.1109/AERO.2015.7119022.
- [4] M. B. Quadrelli, L. J. Wood, J. E. Riedel, M. C. McHenry, M. Aung, L. A. Cangauala, R. A. Volpe, P. M. Beauchamp, and J. A. Cutts, “Guidance, navigation, and control technology assessment for future planetary science missions,” *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 7, pp. 1165–1186, 2015, doi: <https://doi.org/10.2514/1.G000525>.
- [5] T. M. Howard and A. Kelly, “Optimal rough terrain trajectory generation for wheeled mobile robots,” *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007, doi: <https://doi.org/10.1177/0278364906075328>.



- [6] T. Howard, C. Green, and A. Kelly, “Receding horizon model-predictive control for mobile robot navigation of intricate paths,” in *Field and Service Robotics: Results of the 7th International Conference*. Springer, Berlin, Heidelberg, 2010, pp. 69–78, doi: [https://doi.org/10.1007/978-3-642-13408-1\\_7](https://doi.org/10.1007/978-3-642-13408-1_7).
- [7] S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” Iowa State University, Tech. Rep., 1998, computer science dept., tr: 98-11.
- [8] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010, doi: 10.15607/RSS.2010.VI.034.
- [9] S. M. LaValle and J. J. Kuffner Jr, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001, doi: <https://doi.org/10.1177/02783640122067453>.
- [10] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002, doi: <https://doi.org/10.1177/027836402320556421>.
- [11] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Autonomous Robot Vehicles*. Springer, New York, NY, 1990, pp. 396–404, doi: [https://doi.org/10.1007/978-1-4613-8997-2\\_29](https://doi.org/10.1007/978-1-4613-8997-2_29).
- [12] J.-C. Latombe, *Robot Motion Planning*. Springer Science & Business Media, 2012, vol. 124.
- [13] J. Barraquand, B. Langlois, and J.-C. Latombe, “Numerical potential field techniques for robot path planning,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992, doi: 10.1109/21.148426.

- [14] E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992, doi: 10.1109/70.163777.
- [15] Y. Wang, D. Wang, and S. Zhu, “A new navigation function based decentralized control of multi-vehicle systems in unknown environments,” *Journal of Intelligent & Robotic Systems*, vol. 87, no. 2, pp. 363–377, 2017, doi: <https://doi.org/10.1007/s1084>.
- [16] W. Kowalczyk, M. Przybyla, and K. Kozłowski, “Set-point control of mobile robot with obstacle detection and avoidance using navigation function-experimental verification,” *Journal of Intelligent & Robotic Systems*, vol. 85, no. 3-4, pp. 539–552, 2017, doi: <https://doi.org/10.1007/s1084>.
- [17] W. Kowalczyk, “Rapidly converging navigation function control for differentially driven mobile robots,” in *11th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2017, pp. 244–250, doi: 10.1109/RoMoCo.2017.8003920.
- [18] M. B. Horowitz and J. W. Burdick, “Optimal navigation functions for nonlinear stochastic systems,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. IEEE, 2014, pp. 224–231, doi: 10.1109/IROS.2014.6942565.
- [19] C. I. Connolly, J. B. Burns, and R. Weiss, “Path planning using laplace’s equation,” in *IEEE International Conference on Robotics and Automation.*, vol. 3. IEEE, 1990, pp. 2102–2106, doi: 10.1109/ROBOT.1990.126315.
- [20] A. A. Masoud and M. M. Bayoumi, “Robot navigation using the vector potential approach,” in *IEEE International Conference on Robotics and Automation, 1993.*, vol. 1. IEEE, 1993, pp. 805–811, doi: 10.1109/ROBOT.1993.292076.

- [21] S. Garrido, L. Moreno, D. Blanco, and F. Martin, “Smooth path planning for non-holonomic robots using fast marching,” in *IEEE International Conference on Mechatronics, ICM 2009*. IEEE, 2009, pp. 1–6, doi: 10.1109/ICMECH.2009.4957121.
- [22] E. Ralli and G. Hirzinger, “Fast path planning for robot manipulators using numerical potential fields in the configuration space,” in *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems’ 94. ‘Advanced Robotic Systems and the Real World’, IROS’94.*, vol. 3. IEEE, 1994, pp. 1922–1929, doi: 10.1109/IROS.1994.407663.
- [23] Y. Wang and W. Cao, “A global path planning method for mobile robot based on a three-dimensional-like map,” *Robotica*, vol. 32, no. 4, pp. 611–624, 2014, doi: 10.1017/S0263574713000738.
- [24] O. Brock, “Generating robot motion: The integration of planning and execution,” Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2000, aAI9961867.
- [25] A. Stentz, “The d\* algorithm for real-time planning of optimal traverses.” CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1994, cMU-RI-TR-94-37.
- [26] —, “The focussed d\* algorithm for real-time replanning,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1652–1659.
- [27] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
- [28] N. J. Nilsson, *Principles of artificial intelligence*. Morgan Kaufmann, 2014.
- [29] W. H. Kwon and S. H. Han, *Receding Horizon Control: Model Predictive Control for State Models*. Springer Science & Business Media, 2006.

- [30] J. R. Cloutier, “State-dependent riccati equation techniques: An overview,” in *Proceedings of the 1997 American Control Conference.*, vol. 2. IEEE, 1997, pp. 932–936, doi: 10.1109/ACC.1997.609663.
- [31] J. R. Cloutier and D. T. Stansbery, “The capabilities and art of state-dependent riccati equation-based design,” in *Proceedings of the 2002 American Control Conference.*, vol. 1. IEEE, 2002, pp. 86–91, doi: 10.1109/ACC.2002.1024785.
- [32] T. Cimen, “State-dependent riccati equation (sdre) control: a survey,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3761–3775, 2008, doi: <https://doi.org/10.3182/20080706-5-KR-1001.00635>.
- [33] —, “Survey of state-dependent riccati equation in nonlinear optimal feedback control synthesis,” *Journal of Guidance, Control and Dynamics*, vol. 35, no. 4, pp. 1025–1047, 2012, doi: <https://doi.org/10.2514/1.55821>.
- [34] A. Heydari and S. Balakrishnan, “Path planning using a novel finite horizon suboptimal controller,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 4, pp. 1210–1214, 2013, doi: <https://doi.org/10.2514/1.59127>.
- [35] A. Khamis and D. Naidu, “Nonlinear optimal tracking using finite horizon state dependent riccati equation (sdre),” in *Proceedings of the 4th International Conference on Circuits, Systems, Control, Signals (WSEAS)*, 2013, pp. 37–42.
- [36] J. B. Rawlings, “Tutorial overview of model predictive control,” *IEEE Control Systems*, vol. 20, no. 3, pp. 38–52, 2000, doi: 10.1109/37.845037.
- [37] G. Klančar and I. Škrjanc, “Tracking-error model-based predictive control for mobile robots in real time,” *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460–469, 2007, doi: <https://doi.org/10.1016/j.robot.2007.01.002>.
- [38] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, doi: [https://doi.org/10.1016/S0005-1098\(99\)00214-9](https://doi.org/10.1016/S0005-1098(99)00214-9).

- [39] J. A. Primbs, “Nonlinear optimal control: a receding horizon approach,” Ph.D. dissertation, California Institute of Technology, 1999.
- [40] A. Jadbabaie, “Receding horizon control of nonlinear systems: a control lyapunov function approach,” Ph.D. dissertation, California Institute of Technology, 2001.
- [41] M. Sznaier and R. Suarez, “Suboptimal control of constrained nonlinear systems via receding horizon state dependent riccati equations,” in *Proceedings of the 40th IEEE Conference on Decision and Control.*, vol. 4. IEEE, 2001, pp. 3832–3837, doi: 10.1109/CDC.2001.980461.
- [42] M. Sznaier, R. Suárez, and J. Cloutier, “Suboptimal control of constrained nonlinear systems via receding horizon constrained control lyapunov functions,” *International Journal of Robust and Nonlinear Control*, vol. 13, no. 3-4, pp. 247–259, 2003, doi: 10.1002/rnc.816.
- [43] D. Gu and H. Hu, “A stabilizing receding horizon regulator for nonholonomic mobile robots,” *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1022–1028, 2005, doi: 10.1109/TRO.2005.851357.
- [44] ———, “Receding horizon tracking control of wheeled mobile robots,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 743–749, 2006, doi: 10.1109/TCST.2006.872512.
- [45] P. Ru and K. Subbarao, “Nonlinear model predictive control for unmanned aerial vehicles,” *Aerospace*, vol. 4, no. 2, p. 31, 2017, doi: 10.3390/aerospace4020031.
- [46] W. Dixon, D. Dawson, E. Zergeroglum, and F. Zhang, “Robust tracking and regulation control for mobile robots,” in *Proceedings of the 1999 IEEE International Conference on Control Applications.*, vol. 2. IEEE, 1999, pp. 1015–1020, doi: 10.1109/CCA.1999.801026.
- [47] J. Huang, C. Wen, W. Wang, and Z.-P. Jiang, “Adaptive stabilization and tracking control of a nonholonomic mobile robot with input saturation and distur-

- bance,” *Systems and Control Letters*, vol. 62, no. 3, pp. 234–241, 2013, doi: <https://doi.org/10.1016/j.sysconle.2012.11.020>.
- [48] D. G. Wilson and I. Robinett, “Robust adaptive backstepping control for a nonholonomic mobile robot,” in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5. IEEE, 2001, pp. 3241–3245, doi: 10.1109/ICSMC.2001.972018.
- [49] R. Fierro and F. L. Lewis, “Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics,” in *Proceedings of the 34th IEEE Conference on Decision and Control.*, vol. 4. IEEE, 1995, pp. 3805–3810, doi: 10.1109/CDC.1995.479190.
- [50] P. Quillen, K. Subbarao, and J. Muñoz, “Guidance and control of a mobile robot via numerical navigation functions and backstepping for planetary exploration missions,” in *AIAA SPACE 2016, AIAA Space Forum*, 2016, no. 2016-5237, doi: <https://doi.org/10.2514/6.2016-5237>.
- [51] P. Quillen, J. Muñoz, and K. Subbarao, “Path planning to a reachable state using inverse dynamics and minimum control effort based navigation functions,” in *2017 AAS/AIAA Astrodynamics Specialist Conference in Columbia River Gorge, Stevenson, WA*, August 2017, paper no. AAS 17-849.
- [52] A. Godbole, V. Murali, P. Quillen, and K. Subbarao, “Optimal trajectory design and control of a planetary exploration rover,” in *Advances in the Astronautical Sciences Spaceflight Mechanics 2017*, Feb. 2017, vol. 160, 27th AAS/AIAA Space Flight Mechanics Meeting, AAS 17-481.
- [53] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, “Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments,” in *IEEE International Confer-*

- ence on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4029–4036, doi: 10.1109/ICRA.2014.6907444.
- [54] F. N. Martins, W. C. Celeste, R. Carelli, M. Sarcinelli-Filho, and T. F. Bastos-Filho, “An adaptive dynamic controller for autonomous mobile robot trajectory tracking,” *Control Engineering Practice*, vol. 16, no. 11, pp. 1354–1363, 2008, doi: <https://doi.org/10.1016/j.conengprac.2008.03.004>.
- [55] D. Bucciari, D. Perritaz, P. Mullhaupt, Z.-P. Jiang, and D. Bonvin, “Velocity-scheduling control for a unicycle mobile robot: Theory and experiments,” *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 451–458, 2009, doi: 10.1109/TRO.2009.2014494.
- [56] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. Springer Science & Business Media, 2013.
- [57] V. Gazi and K. M. Passino, “A class of attractions/repulsion functions for stable swarm aggregations,” *International Journal of Control*, vol. 77, no. 18, pp. 1567–1579, 2004, doi: <https://doi.org/10.1080/00207170412331330021>.
- [58] V. Gazi, B. Fidan, Y. S. Hanay, and İ. Köksal, “Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques,” *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 15, no. 2, pp. 149–168, 2007.
- [59] N. E. Leonard and E. Fiorelli, “Virtual leaders, artificial potentials and coordinated control of groups,” in *Proceedings of the 40th IEEE Conference on Decision and Control, 2001.*, vol. 3. IEEE, 2001, pp. 2968–2973, doi: 10.1109/CDC.2001.980728.
- [60] R. O. Saber and R. M. Murray, “Flocking with obstacle avoidance: Cooperation with limited communication in mobile networks,” in *42nd IEEE Conference on*

- Decision and Control, 2003. Proceedings.*, vol. 2. IEEE, 2003, pp. 2022–2028, doi: 10.1109/CDC.2003.1272912.
- [61] K. D. Do, “Formation tracking control of unicycle-type mobile robots with limited sensing ranges,” *IEEE Transactions on Control Systems Technology*, vol. 16, no. 3, pp. 527–538, 2008, doi: 10.1109/TCST.2007.908214.
- [62] D. H. Kim, H. Wang, and S. Shin, “Decentralized control of autonomous swarm systems using artificial potential functions: Analytical design guidelines,” *Journal of Intelligent and Robotic Systems*, vol. 45, no. 4, pp. 369–394, 2006, doi: 10.1007/s10846-006-9050-8.
- [63] E. De Vries and K. Subbarao, “Cooperative control of swarms of unmanned aerial vehicles,” in *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Orlando, USA, 4-7 January 2011; AIAA 2011-78*. American Institute of Aeronautics and Astronautics (AIAA), 2011, doi: <https://doi.org/10.2514/6.2011-78>.
- [64] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.
- [65] C.-T. Chen, *Linear system theory and design*. Oxford University Press, Inc., 1995.
- [66] R. L. Williams, D. A. Lawrence, *et al.*, *Linear state-space control systems*. John Wiley & Sons, 2007.
- [67] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199.
- [68] S. J. Wright and J. Nocedal, *Numerical optimization*. Springer Science, 1999.
- [69] N. V. M. Veerapaneni, “Real-time minimum jerk optimal trajectory synthesis and tracking for ground vehicle applications,” Master’s thesis, The University of Texas at Arlington, 2017, isbn: 9780355886504.



- [70] J. L. Crassidis and J. L. Junkins, *Optimal estimation of dynamic systems*. CRC press, 2011.
- [71] H. K. Khalil, *Nonlinear Systems*. Prentice-Hall, New Jersey, 1996.

## BIOGRAPHICAL STATEMENT

Paul Quillen pursued his first undergraduate degree in Applied Mathematics at Baylor University. Upon completion of his first undergraduate degree, Paul pursued a second baccalaureate in Aerospace Engineering from the University of Texas at Arlington. After graduating Cum Laude with honors, he enrolled in the bachelor's to doctoral program at the same institution. Paul has been a member of the Air Force Research Laboratory's Space Scholar program over the course of his doctoral studies and was awarded the summer 2018 Dissertation Fellowship from the University of Texas at Arlington. After earning his Ph.D., Paul will be joining AeroVironment Inc. as a Robotics GNC Engineer working on unmanned aerial vehicles.