COMPLEX AERIAL FLIGHT UTILIZING MODEL PREDICTIVE CONTROL

A Thesis in
Aerospace Engineering
by
Pranjali Padghan

© 2018 Pranjali Padghan

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2018
The thesis of Pranjali Padghan was reviewed and approved* by the following:

Alan Wagner
Associate Professor of Aerospace Engineering
Thesis Advisor

Eric Johnson
Professor of Aerospace Engineer

Amy Pritchett
Professor of Aerospace Engineering
Head of the Department of Department or Aerospace Engineering

*Signatures are on file in the Graduate School
ABSTRACT

In this dissertation a quadcopter model is developed, and Model Predictive Control techniques are implemented for trajectory tracking and obstacle avoidance. The major application that has been presented in this thesis is choreographed moves for a swarm of quadcopters. First a nonlinear model of the quadcopter is derived from the Newton-Euler’s equations and linearized into state space form. This model was then further used for implementing Model Predictive Control algorithm for the purpose of trajectory tracking.

The thesis concludes with an investigation of multi-vehicle obstacle avoidance. Model Predictive Control takes account of the obstacles as constraints as a part of an optimization process. To deal with the obstacles MOANTOOL has been used which serves as an environment for formulating Model Predictive Control problems while also avoiding obstacles. Matlab simulations demonstrates the system.
TABLE OF CONTENTS

List of Figures........................................................................................................vi
List of Tables..........................................................................................................xii
Acknowledgements............................................................................................xiii

Chapter 1 Introduction .........................................................................................1
  1.1 Motivation........................................................................................................1
  1.2 System Overview ...........................................................................................3
  1.3 Problem Description .....................................................................................5
    1.3.1 Physical Constraints ...............................................................................5
  1.4 Contributions ..................................................................................................6
  1.5 Structure of Thesis .........................................................................................7

Chapter 2 Trajectory Tracking and Obstacle Avoidance .....................................8
  2.1 Problem Statement .........................................................................................8
  2.2 Background ...................................................................................................10
    2.2.1 Linear Control ........................................................................................11
  2.3 Latest application of Model Predictive Control ...........................................12
  2.4 Summary .......................................................................................................13

Chapter 3 Model Predictive Control ..................................................................14
  3.1 Chapter Overview ..........................................................................................14
  3.2 MPC Formulation ..........................................................................................15
  3.3 Prediction .......................................................................................................16
  3.4 Objective function .........................................................................................18
  3.5 Unconstrained Quadratic Programming ......................................................21
  3.6 Introduction to MOANTOOL ........................................................................21
    3.6.1 Vehicle Path Planning and Obstacle Avoidance Problem Setup ..........23
  3.7 Summary .......................................................................................................25

Chapter 4 Results of Simulation .........................................................................27
  4.1 Chapter Overview ..........................................................................................27
  4.2 Trajectory tracking using Model Predictive Control ....................................27
  4.3 Obstacle Avoidance ......................................................................................82
  4.4 Summary .......................................................................................................85

Chapter 5 Conclusion ..........................................................................................86
  5.1 Summary of Contributions ...........................................................................86
  5.2 Limitations and Assumptions .......................................................................88
  5.3 Future Scope ..................................................................................................89
REFERENCES .................................................................................................................. 91
Appendix A ..................................................................................................................... 94
  A.1 Parameters of MPC Controller ................................................................................. 94
Appendix B ..................................................................................................................... 100
  B.1 Velocity Plots .......................................................................................................... 100
Appendix C ..................................................................................................................... 110
  C.1 MOANTOOL Commands ....................................................................................... 110
LIST OF FIGURES

Figure 1-1: Swarm of quadcopters performing in Orlando [2]. .................................................. 2

Figure 1-2: System overview of an unmanned aerial system showing various discretized layers [1]. ................................................................................................................................. 3

Figure 2-1: Process of converting a physical system into state-space form. ......................... 10

Figure 3-1: Basic concept of Model Predictive Control [27]. ......................................................... 15

Figure 3-2: Work flow of prediction using initial value of state space vector ...................... 18

Figure 3-3: The idea of obstacle avoidance. The reference trajectory is the dashed line. At every time step of the prediction horizon, the red line indicates the convex output constraints $Yk$ are updated so that there is no collision with the obstacle O [26]. .......... 23

Figure 3-4: Convex body defining a convex polytope ................................................................. 24

Figure 4-1(a): Case 1: A single quadcopter following infinity loop trajectory, the blue dot denotes the quadcopter and the reference trajectory is shown in red. ......................... 30

Figure 4-2(a): Case 1: Variation of Control input $u1$ with respect to time ......................... 32

Figure 4-2(b): Case 2: Variation of Control input $u1$ with respect to time ......................... 32

Figure 4-2(c): Variation of Control input $u1$ with respect to time for case 3. ................. 33

Figure 4-3(a): The variation of control input $u2$ (torque) with respect to time for case 1 ...... 34

Figure 4-3(b): The variation of control input $u2$ (torque) with respect to time for case 1 ...... 34

Figure 4-3(c): Variation of Control input $u2$ with respect to time for case 3. ................. 35

Figure 4-4(a): Root mean squared error (in meters) of the position with respect to time for case 1 ................................................................................................................................. 36

Figure 4-4(b): Root mean squared error (in meters) of the position with respect to time for case 1 ................................................................................................................................. 36

Figure 4-4(c): Root mean squared error (in meters) of the position with respect to time for case 3 ................................................................................................................................. 37

Figure 4-5(a): Two quadcopters following a taco-shell trajectory for case 1. The reference trajectory is shown in red and the resultant trajectory is shown in black ....................... 38
Figure 4-5(b): Two quadcopters following a taco-shell trajectory for case 2. The reference trajectory is shown in red and the resultant trajectory is shown in black. .......39

Figure 4-5(c): Two quadcopters following a taco-shell trajectory for case 3. The reference trajectory is shown in red and the resultant trajectory is shown in black. ..............39

Figure 4-6(a): The control input u1 variation with respect to time for case 1. .................40
Figure 4-6(b): The control input u1 variation with respect to time for case 2. .................41
Figure 4-6(c): Variation of Control input u1 with respect to time for case 3. .................41

Figure 4-7(a): The variation of control input u2 (torque) with respect to time for case 1. ......42
Figure 4-7(b): The variation of control input u2 (torque) with respect to time for case 2. ......42
Figure 4-7(c): Variation of Control input u2 with respect to time for case 3. .................43

Figure 4-8(a): Root mean squared error (RMSE) for position in x, y and z for case 1. .......44
Figure 4-8(c): Root mean squared error of the position with respect to time of case 3 .......45

Figure 4-9(a): Ten quadcopters in a stacked taco-shell trajectory for case 1. The reference trajectory is shown in red and the resultant trajectory is shown in black. .................46
Figure 4-9(b): Ten quadcopters in a stacked taco-shell trajectory for case 2. The reference trajectory is shown in red and the resultant trajectory is shown in black. .................46
Figure 4-9(c): Ten quadcopters in a stacked taco-shell trajectory for case 3. The reference trajectory is shown in red and the resultant trajectory is shown in black. .................47

Figure 4-10(a): The variation of control input u1 (Thrust) with respect to time for case 1 .....48
Figure 4-10(b): The variation of control input u1 (Thrust) with respect to time for case 2 .....48
Figure 4-10(c): Variation of Control input u1 with respect to time for case 3. .................49

Figure 4-11(a): The variation of control input u3 with respect to time for case 1. .................50
Figure 4-11(b): The variation of control input u3 with respect to time for case 1. .................50
Figure 4-11(c): Variation of Control input u3 with respect to time for case 3. .................51

Figure 4-12(a): Root mean squared error of the position with respect to time of case 1 .......52
Figure 4-12(b): Root mean squared error of the position with respect to time of case 2 .......52
Figure 4-12(c): Root mean squared error of the position with respect to time of case 3 .......53
Figure 4-13(a): Three quadcopters spelling a word in 3-D space for case 1 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.................................................................54

Figure 4-13(b): Three quadcopters spelling a word in 3-D space for case 2 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.................................................................54

Figure 4-13(c): Three quadcopters spelling a word in 3-D space for case 3 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.................................................................55

Figure 4-14(a): The change in control input $u_1$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the values of case 3 parameter values........56

Figure 4-14(b): Variation of control input $u_1$ with respect to time for case 2.......................56

Figure 4-14(c): Variation of control input $u_1$ with respect to time for case 3.......................57

Figure 4-15(a): Variation of control input $u_3$ with respect to time for case 1.......................58

Figure 4-15(b): Variation of control input $u_3$ with respect to time for case 2.......................58

Figure 4-15(c): Variation of control input $u_3$ with respect to time for case 3.......................59

Figure 4-16(a): Root mean squared error of the position for the first quadcopter following the trajectory of the alphabet L with respect to time for case 1.................................60

Figure 4-16(b): Root mean squared error of the position for the first quadcopter following the trajectory of the alphabet A with respect to time for case 1.................................60

Figure 4-16(c): Root mean squared error of the position for the first quadcopter following the trajectory of the alphabet N with respect to time for case 1.................................61

Figure 4-17(a): RMSE of the position values of the quadcopter trying to follow the trajectory of L case 2..............................................................................................................62

Figure 4-17(b): RMSE of the position values of the quadcopter trying to follow the trajectory of A case 2..............................................................................................................62

Figure 4-17(c): RMSE of the position values of the quadcopter trying to follow the trajectory of N case 2..............................................................................................................63

Figure 4-18(a): RMSE of the position values of the quadcopter trying to follow the trajectory of L case 3..............................................................................................................63

Figure 4-18(b): RMSE of the position values of the quadcopter trying to follow the trajectory of A case 3..............................................................................................................64

Figure 4-18(c): RMSE of the position values of the quadcopter trying to follow the trajectory of N case 3..............................................................................................................64
Figure 4-19(a): Quadcopter trying to form a swarm of a heart shaped for case 1 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.

Figure 4-19(b): Quadcopter trying to form a swarm of a heart shaped for case 2 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.

Figure 4-19(c): Quadcopter trying to form a swarm of a heart shaped for case 3 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.

Figure 4-20(a): Variation of control input $u_1$ with respect to time for case 1 parameters.

Figure 4-20(b): Variation of control input $u_1$ with respect to time for case 2 parameters.

Figure 4-20(c): Variation of control input $u_1$ with respect to time for case 3 parameters.

Figure 4-21(a): Variation of control input $u_2$ with respect to time for case 1 parameters.

Figure 4-21(b): Variation of control input $u_2$ with respect to time for case 1 parameters.

Figure 4-21(c): Variation of control input $u_2$ with respect to time for case 3 parameters.

Figure 4-22(a): Variation of control input $u_3$ with respect to time for case 1 parameter.

Figure 4-22(b): Variation of control input $u_3$ with respect to time for case 2 parameter.

Figure 4-22(c): Variation of control input $u_3$ with respect to time for case 3 parameters.

Figure 4-23(a): RMSE for the position of quadcopter number 5 for case 1 plotted against time.

Figure 4-23(b): RMSE for the position of quadcopter number 12 for case 1 plotted against time.

Figure 4-23(c): RMSE for the position of quadcopter number 20 for case 1 plotted against time.

Figure 4-24(a): RMSE for the position of quadcopter number 5 for case 2 plotted against time.

Figure 4-24(b): RMSE for the position of quadcopter number 12 for case 2 plotted against time.

Figure 4-24(c): RMSE for the position of quadcopter number 20 for case 2 plotted against time.

Figure 4-25(a): RMSE for the position of quadcopter number 5 for case 3 plotted against time.
Figure 4-25(b): RMSE for the position of quadcopter number 12 for case 3 plotted against time. ..........................77

Figure 4-25(c): RMSE for the position of quadcopter number 20 for case 3 plotted against time.................................................................78

Figure 4-26: Variation of control input u1 with respect to parameter $P \ (1,1)$ ........................................79

Figure 4-27: The variation of Root Mean Squared Error of trajectory in x direction with respect to $Q \ (1,1)$.................................................................80

Figure 4-28: The value of Thrust ($u1$) and RMSE in z plotted against $Q \ (5,5)$. The red line indicates the thrust values and the blue line shows the variation of RMSE in z direction .........................................................81

Figure 4-29: Obstacle avoidance. The figure shows a quadcopter avoiding four obstacles represented by the red blocks. The green dotted line shows the reference circular trajectory, and the pink line is the actual trajectory of the quadcopter. .................. 84

Figure B-1: Linear Velocity of the quadcopter with respect to time for case 1 of experiment 1. ..........................................................100

Figure B-2: Velocity of the quadcopter in x, y and z with respect to time for case 2 of experiment 1..............................................................101

Figure B-3: Velocity of the quadcopter in the x, y and z direction with respect to time for test case 3 of experiment 1..................................................101

Figure B-4: Variation in velocity with respect to time case 1 of experiment 2. ..................102

Figure B-5: Variation of velocity in the x, y and z direction case 1 of experiment 3. ...........102

Figure B-6: Variation of velocity of the quadcopter following the L trajectory with respect to time case 1 of experiment 4. .................................103

Figure B-7: Velocity of the quadcopter following the L trajectory with respect to time case 2 of experiment 4................................................................104

Figure B-8: Velocity of the quadcopter following the trajectory 'L' shown here with respect to time case 3 of experiment 4. ..............................104

Figure B-9: Velocity of the quadcopter no. 5 plotted with respect to time case 1 of experiment 5. ....................................................................105

Figure B-10: Velocity of the quadcopter no. 12 plotted with respect to time case 1 of experiment 5.................................................................105

Figure B-11: Velocity of the quadcopter no. 20 plotted with respect to time case 1 of experiment 5.................................................................106

Figure B-12: Velocity of quadcopter no. 5 plotted against time case 2 of experiment 5......106
Figure B-13: Velocity of quadcopter no. 12 plotted against time case 2 of experiment 5......107

Figure B-14: Velocity of quadcopter no. 20 plotted against time case 2 of experiment 5......107

Figure B-15: Velocity of the quadcopter no. 5 plotted against time case 3 of experiment 5. .................................................................................................................................................................108

Figure B-16: Velocity of the quadcopter no. 12 plotted against time case 3 of experiment 5. ..................................................................................................................................................................................................................108

Figure B-17: Velocity of the quadcopter no. 20 plotted against time case 3 of experiment 5. ..................................................................................................................................................................................................................109
LIST OF TABLES

Table 1: Values of penalizing terms.................................................................31
Table 2: Values of penalizing terms.................................................................40
Table 3: Values of penalizing terms.................................................................47
Table 4: Values of penalizing terms.................................................................55
Table 5: Values of penalizing terms.................................................................67
Table 6: Predicted range of Value for best performance of the Quadcopters........82
Table 7: Test cases to evaluate the performance of MPC controller .................87
ACKNOWLEDGEMENTS

I would like to pay utmost gratitude to my advisor Dr. Alan Wagner for guiding me throughout. I would also like to thank the rest of my committee members Dr. Eric Johnson and our department head Dr. Amy Pritchett for reviewing my work.

I would like to dedicate my work to my parents Siddharth Padghan and Jyoti Padghan and my elder brother Prashant Padghan, who made it possible for me to be a part of such a prestigious university, without them I am nothing. Finally, a sincere thanks to my dear friend Kevin Carvalho for his motivation and support.
Chapter 1

Introduction

This thesis presents a method for trajectory tracking and obstacle avoidance using a control technique called Model Predictive Control. This work is motivated by the goal of implementing a control algorithm that will enable autonomous UAVs to navigate and follow a specified trajectory. The term UAV in this work is used to refer to a Quadcopter or Drone. Autonomy is a major area of current interest in UAVs. The term complete autonomy refers to the capability of the UAV to act without any supervision in an environment filled with uncertainty [1]. The field of Robotics and the field of dynamics and controls explores solutions for path planning, obstacle avoidance, decision-making etc.

For a quadcopter, designing a control system can be challenging because these systems are under-actuated, non-linear, and highly unstable system. Moreover, quadcopter’s robustness to environmental changes, reliable stabilization and sensor noise filtering ability. A great deal of effort is currently focused on quadcopters in order to achieve on-board stabilization and the ability to plan and execute movements through a path [1].

1.1 Motivation

Efforts are being made by researchers to develop technology for quadcopters that would enable onboard stabilization, path planning, and obstacle avoidance. For example, Intel Inc. collaborated with Disney World to create a choreographed routine using 300 quadcopters in Orlando [2]. This aerial performance demonstrates the implementation of path planning, swarm intelligence, and distributive control. One of the experiments performed in this dissertation including a swarm of
quadcopters is inspired by this aerial performance. This is the beginning of a whole new sector of entertainment.

![A swarm of quadcopters performing in Orlando](image)

Figure 1-1: A swarm of quadcopters performing in Orlando [2].

Given the variety of UAV applications, it is important to develop systems capable of functioning in a complex environment with numerous uncertainties. These uncertainties include external uncertainties such as the presence of obstacles, other flying objects, no-fly-zones, wind, the loss of a GPS signal and internal uncertainties such as modeling error. To summarize, Hanna Kurniawati states that there are three major sources of uncertainty: Control error, sensing error and having an incorrect map of the environment [3]. As a result of these uncertainties, a vehicle’s predefined trajectory may be useless at mission time. It is, therefore, necessary to create a UAV trajectory using an online generator rather than relying on an offline preplanning. It is important to develop motion planning algorithms which take into account the different sources of uncertainty using the system’s onboard computation capability. This thesis aims to develop an approach that could be applied to UAVs in 3D space. We utilized a control technique called Model Predictive Control which is a type of high-level controller. It allows multiple UAVs to perform choreographed
maneuvers and track complicated trajectories as a part of a swarm. Part of this thesis demonstrates the obstacle avoidance capability of Model Predictive Control.

1.2 System Overview

The figure 1-2 above shows the general system architecture for a notional UAV. The system design, development, and deployment are divided into various sub-sections. The mechanics and electronics
layer deals with the hardware specifications. It includes onboard microcontrollers which play an important role in autonomy. The vehicle has to follow certain regulations set by the Federal Aviation Administration (FAA) before deployment. The regulation and permit layer provides these guidelines. The mission planning layer entails the objective of the flight and its specifications. The ground station layer includes a GPS reference station and laptops/computers for monitoring the vehicle during flight. The communication and Telemetry layer takes care of the wireless communication between vehicle and ground station. Telemetry is the process of measuring the parameters of an object whose measurement data is transmitted wirelessly. The vehicle autonomy layer divides tasks based on path planning and vehicle control. It is important to manage all of the layers of the control architecture. A common approach is to decouple the entire control architecture into a hierarchy based on the complexity of the tasks. Task allocation is an important layer for a mission scenario with multiple agents. Task allocation refers to finding a solution to the specified task and allocating it among the vehicles. For instance, the task could be as simple as finding a particular target within a search area. This task would require an agent with an infrared onboard camera to search for the target and another agent would take high-resolution images of the search area. The higher levels of architecture often include mission specific task allocation, perception related obstacle detection and characterization, and path planning. The path planning layer typically provides the waypoints to be followed while avoiding obstacles. It may also include a trajectory generation layer that connects the waypoints with the feasible paths. The lower level control architecture typically includes the vehicle control layer, trajectory generation and collision avoidance layers. In this dissertation, we assume a known environment and address the collision avoidance, trajectory generation and vehicle control layers of Figure 1-2.

In the past, linear control techniques have been used on UAVs for the purpose of stabilization [1]. Due to the limitations of linear control approaches in terms of robust trajectory tracking and obstacle avoidance capability, non-linear control or model-based control approaches have become popular [1]. These control algorithms have a history of assuring better flight
performance in terms of robust and accurate trajectory tracking [1]. For this dissertation, collision avoidance is achieved by using MOANTOOL - a MATLAB based toolbox. Trajectory generation and vehicle control are accomplished by developing a Model Predictive Controller in MATLAB. We assume that a series of waypoints are provided to the system in advance. Given a series of waypoints, the Model Predictive Controller finds a way through the environment based on the value of a cost function. The cost function is formulated so that its resulting value increases if it encounters an obstacle in its path. A MATLAB based optimizer - Quadprog is used to constantly monitor and keep the value of cost function minimal.

1.3 Problem Description

The main goal of this dissertation is to present a control strategy for a UAV to guide the vehicle along a given trajectory and avoid obstacles on the way. This is a hard problem because control of a quadcopter demands management, implementation and testing of an unstable and non-linear system.

1.3.1 Physical Constraints

The UAV is an under-actuated, unstable, non-linear model. The term under-actuated means that a system has fewer actuators than the degree of freedom. The UAV model has six degrees of freedom and has four rotors (or actuators) which makes it an under-actuated system with nonholonomic constraints and is inherently non-linear in nature. It is not possible to control all the states simultaneously. To understand the behavior of a non-holonomic system we can look at a simple example of a car. A car which has 3 degrees of freedom- \( x \), \( y \) position and turn angle \( \theta \) and only two actuators throttle and steering. If this system was holonomic, a car could move directly to any position or rotation angle. Since it is a nonholonomic system it has constraints moving in the lateral
direction, the act of parallel parking demonstrates these constraints. Similarly, a quadcopter in particular cannot take sharp turns because of the nonholonomic constraints. In order to tackle the problem of under-actuation, it is extremely important to use a control law and a system model which would be able to capture the behavior of the quadcopter. Model Predictive Control even has the ability to produce stabilizing feedback for these kinds of vehicles [4]. In this dissertation, a linearized model is used to demonstrate a discrete time-invariant MPC algorithm on an unstable and under-actuated vehicle.

1.4 Contributions

This thesis makes the following contributions:

- **Develop the framework for trajectory tracking and obstacle avoidance based on Model Predictive Control:** Once a mathematical model of the vehicle is obtained, the Model Predictive Control algorithm is derived for the implementation of the trajectory tracking problem. A MATLAB based tool MOANTOOL allows us to formulate this problem in terms of MPC.

- **Present simulation results for a swarm of UAVs tracking complicated trajectories:** Simulation results show that the Model Predictive Control is capable of handling the navigation and obstacle avoidance problem. The reference trajectory can be smoothly followed using the Model Predictive Controller. Various experiments are performed using a swarm of UAVs to track complicated trajectories with different parameters. A range of MPC parameters was also predicted which would be useful while designing a control system for UAV using MPC. The system’s obstacle avoidance capability is tested using MOANTOOL.
1.5 Structure of Thesis

The dissertation is divided into five chapters which are as follows:

- Chapter 2 presents the prior work in the area of motion planning techniques.
- Chapter 3 introduces Model Predictive Control (MPC) with its core concepts. The MPC formulation and implementation is thoroughly explained. Optimization is an integral part of MPC. Also, a MATLAB based tool called as MOANTOOL is introduced. MOANTOOL allows the users to formulate the MPC based problems with the flexibility to use any type of optimization solver suitable for the problem. In this thesis, QUADPROG – a MATLAB based function is used to solve the optimization.
- Chapter 4 presents the results from all the experiments that were performed. The major emphasis on the trajectory tracking for a swarm of quadcopters.
- Chapter 5 summarizes the conclusions based on the experiments and provides an insight related to the future work that can be done using Model Predictive Control.
Chapter 2

Trajectory Tracking and Obstacle Avoidance

This chapter presents the problem definition and related work. The problem statement focuses on the development and implementation of a Model Predictive Control framework for trajectory tracking and obstacle avoidance for a swarm of UAVs. Addressing this problem statement demands work on the following related steps: 1) Mathematically modeling the UAV, 2) Implementing MPC for trajectory tracking, and 3) Obstacle avoidance using the MOANTOOL.

In addition to describing the problem statement in detail, this chapter provides background information in the form of a literature review. Specifically, control techniques that have been used in the past are reviewed. This chapter also sheds light on some of the references that were studied in order to understand and compare the working of MPC with respect to other popular techniques. Also, a section is dedicated to outlining recent applications of MPC.

2.1 Problem Statement

This dissertation addresses the problem of trajectory tracking and obstacle avoidance using Model Predictive Control. The simulation is developed in MATLAB where the UAV is represented as a point object. There are two separate experiments presented in this dissertation.

The first experiment which is the major focus of this dissertation deals with the trajectory tracking and application of Model Predictive Control. There are five different trajectories provided to see how MPC can deal with multiple quadcopters and trajectory tracking based on the values of parameters. The goal behind performing a series of these experiments with different trajectories was to see the effect of parameters of MPC on the resultant trajectory and to find its trade-off with the control input required. For the trajectory tracking, the Model Predictive Control algorithm is used in this thesis, which takes a series of waypoints as the input. Once it is known that a quadcopter
is capable of reaching a destination location, a reference trajectory can be fed to the quadcopter. The resultant trajectory is the output in this case. While tracking a trajectory the quadcopter also needs to be equipped to avoid obstacles. This is referred to as trajectory planning.

This brings us to the second experiment in this thesis. Trajectory planning is addressed in this thesis by using an external toolbox called MOANTOOL. The input to MOANTOOL is a state space model of the vehicle, obstacle position, and reference trajectory. The output of MOANTOOL is an obstacle-free trajectory. This is discussed in more details in chapter 5. The vehicle’s state is describe as \( X = [x \ y \ z \ \phi \ \theta \ \psi] \) where \( x, \ y, \ z \) are the coordinates of the vehicle’s position in 3D space and where the roll, pitch and yaw angles are given by \( \phi, \ \theta, \ \psi \) respectively. The dynamics of the vehicle are captured by the linearized state space model. The state space model is a mathematical representation of a physical system using differential equations or equations of motion. Figure 2-1 briefly explains the steps needed to convert a physical system into state space form. Once the linearized equations of motion are obtained, Jacobian linearization results in \( A, B, C, D \) matrices which are then used to write the state space form. Generally, MPC uses a discrete-time system representation which is explained in detail in chapter 3.
2.2 Background

Once the vehicle dynamics model has been derived we can now plan paths for the vehicle while also attempting to avoid obstacles. There are several methods that have been developed to generate a path for a UAV traveling from one point to another while also avoiding the obstacles. Since the major focus of this dissertation is trajectory tracking using MPC, this section presents the latest application of MPC and its comparison with a few popular methods.
2.2.1 Linear Control

Linear control techniques use linear system models. Linear system models are governed by linear differential equations. Linear systems are the ones that follow the principle of superposition. Linear Quadratic Regulator (LQR) are the most well-known linear control methods. The following section explains LQR briefly.

2.2.1.1 Linear Quadratic Regulator (LQR)

Linear Quadratic Regulator (LQR) is a special case of optimal control which uses a linear state space model and a quadratic form of cost function. A state feedback controller is used to minimize the value of a linear quadratic cost function. It uses a continuous state space form with the equation,

$$\dot{x} = Ax + Bu.$$  

(1)

The major part of LQR is designing a state feedback controller $K$ matrix in a way that the cost function $J$ is minimized. The cost function $J$ is given by [6],

$$J = \int_{0}^{\infty} (x^T Q x + u^T R u) \, dt$$  

(2)

Where $Q$ and $R$ are weighting matrices for the purpose of penalizing. Now the feedback control law to minimize the value of cost function is given by:

$$u = -Kx$$

$$K = R^{-1}B^TP$$  

(3)

where $P$ can be found through the solution of continuous time Riccati Equation,

$$A^TP + PA - PB R^{-1} B^TP + Q = 0$$  

(4)

LQR is used in attitude control and stabilization [5] and for trajectory tracking [7], but is limited in its ability to accurately track a trajectory when encountering obstacles [8]. More analysis is needed to determine the performance of LQR in the presence of multiple obstacles. The major difference between LQR and MPC is the formulation. MPC solves the optimization problem using a moving
time horizon whereas LQR solves a similar problem within a fixed window. MPC is formulated as the repeated solution of a finite horizon open-loop optimal control problem subject to the input and predicted behavior and state constraints [9]. The biggest advantage of using a moving time horizon window is the ability to perform real-time optimization with hard constraints on system variables. The research paper titled "A comparison of LQR and MPC control algorithms of an inverted pendulum" [9] addresses the difference between the two control techniques, by implementing them for the same purpose of stabilization of inverted pendulum. The research concluded that MPC results in better trajectory tracking and also outlined the key differences which are as follows:

1. Both the controllers use different variables for determining the control matrix. For instance, in the case of LQR the weight matrices corresponding to the control input $u(k)$ and the states $x(k)$. In the case of an MPC controller, the optimum control signal is determined by using output error $e(k)$ and increments of control signal $u(k)$ [9].

2. Another critical difference between the LQR and MPC is the form of feedback that is used. LQR finds its solution in the form of a control matrix. MPC sets the control over a prediction horizon, which indicates that if the model is incorrect the control signal should be calculated at every time step $k$ [9].

Another author compares LQR and MPC for the purpose of flood control [10]. They show that LQR can be used to control irrigation canals using set-point control but cannot be used to prevent flooding in the presence of large disturbances because of the difficulty associated with tuning the $W$ and $R$ weight matrices. MPC can be used in this application for both set-point control and flood control through the use of cost constraints [10].

### 2.3 Latest application of Model Predictive Control

MPC has been used for the purpose of 2D and 3D path planning owing to its ease of application of constraints. The soft constraints can be included as a part of weighting matrices where the input
and deviations are penalized. The hard constraints can be included in terms of linear inequalities which are used while solving the optimization problem. A recent application of Model Predictive Control algorithm is being used for autonomous landing of an UAV on a moving platform [11]. A linear MPC controller was implemented and compared to LQR, PID and Backstepping control techniques. Compared to LQR and PID, Backstepping and MPC show better performance [11].

Another crucial aspect of implementing MPC on multiple agents is the use of a centralized or decentralized approach. The centralized approach refers to using one central controller based on MPC which will be responsible for motion control all the quadcopters. For the decentralized MPC approach each vehicle will plan its own trajectory and will be responsible for the individual motion. Sina Mansouri’s work compares a centralized and decentralized MPC framework in terms of computation time and flight time. It was found that the computation time required for a centralized framework is much higher than that for decentralized approach but the flight time for centralized framework is shorter than decentralized MPC [12]. This approach used in this dissertation is simulation based on centralized framework.

2.4 Summary

This chapter discussed some recent applications of the MPC and presented its comparison with popular control methods. In the next chapter, we discuss the basic formulation of MPC. We also derive analytical solution to trajectory tracking problem and provide an introduction to MOANTOOL which addresses the obstacle avoidance capability of MPC.
Chapter 3

Model Predictive Control

3.1 Chapter Overview

In this chapter Model Predictive Control (MPC) is introduced and the important parameters for the derivation of the MPC framework are presented. MPC operates by computing the iterative finite horizon optimization of a dynamic system model. MPC has become quite popular in its application as an integral controller in self-driving cars and even autonomous underwater vehicles (AUV). MPC provides the optimal solution to the cost function by predicting the behavior of the system and can handle constraints on the system hence making it ideal for online computation and problems dealing with uncertainties. The major application of MPC is for obstacle avoidance. While controlling the MIMO variables of the system and satisfying various constraints on both input and output variables, MPC uses a dynamic model of the system to predict future values of output using the current measurement of the state. MPC requires the state-space model of the system, the initial conditions, and the desired future states sequence. The objective of MPC is to determine how to change the system’s input variables in order for the predicted values to move closer to the set goal for those variables and to minimize the error. Figure 3-1 depicts the working principle graphically and the explanation of the figure is given below.
In the figure 3-1, the actual output is \( y \), and the predicted output is \( \hat{y} \). The number of predictions is denoted as \( P \) i.e. the Prediction Horizon. At the current time step \( k \), the MPC gives \( P \) number of predicted input values which is \( u(k + i - 1) \) where \( i = 1, 2, 3, \ldots, P \). The control horizon, \( M \) that is used in this dissertation is taken as 10. The number of control input produced during one iteration of the prediction is control horizon. The set of values will contain the current input denoted by \( u(k) \) and \( M-1 \) number of future inputs. The input is calculated such that the predicted output reaches the actual output in an optimal manner. To find the optimal value of input an Objective (Cost) function is formulated which can later be solved using quadratic programming techniques in order to minimize the cost. The \( D \) matrix is associated with the output vector in the state space form which is explained in the section 3.2.

### 3.2 MPC Formulation

In order to formulate an MPC problem, one must represent the dynamics of the system in the state space form. State Space form is a representation of the physical system in a matrix form whereby
the first order differential equations give the relation between input, output and state variables in a vector form [13]. In this dissertation a Discrete Time-Invariant system is used which is of form:

\[ \begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \\
    y(k) &= Cx(k) + Du(k)
\end{align*} \] (5)

where \( x(k) \) is the state space vector and \( y(k) \) is the output vector of the system, \( u(k) \) is the input vector and \( k \) is the time instant. \( A, B, C, D \) are the matrices associated with the respective vectors. Since the quadcopter dynamics are non-linear, the differential equations have to be linearized first in order to write the system in the form above. The linearization process was described in detail in section 2. Following the linearization, the state space form of the equations of motion are obtained and the state space vector, which is a vector containing the linear, angular positions and velocities, is denoted as follows for time step \( k \),

\[
X(k) = [x(k), \dot{x}(k), y(k), \dot{y}(k), z(k), \phi(k), \dot{\phi}(k), \theta(k), \dot{\theta}(k), \psi(k), \dot{\psi}(k)]^T
\] (7)

### 3.3 Prediction

The state space vector is introduced in the previous section. This state space vector is used to perform the prediction in MPC. Predicting the future states based on the initial condition and the system input is the core principle of the MPC algorithm. In order to perform a prediction, only the first stage of the control strategy is implemented which is \( u(k) \), then the plant state is sampled again at \( k+1 \) and the calculations are repeated starting from the current state, giving us a new control and predicted state path. The horizon keeps shifting forward hence MPC is also called receding horizon control. Ultimately, MPC generates a control action by sampling at each instant of time. Since it is a prediction based algorithm, for \( k = 1,2, \ldots \), Eq. (5) can be written as:

\[
x(1) = Ax(0) + Bu(0)
\]
\[ x(2) = Ax(1) + Bu(1) \]
\[ = A(Ax(0) + Bu(0)) + Bu(1) \]
\[ = A^2 x(0) + ABu(0) + Bu(1) \]  

The general form of Eq. 8 can be given as follows,

\[ x(k) = A^k x(0) + \sum_{i=1}^{k-1} A^i Bu(i - 1) + Bu(0). \]  

The predicted states and input can be written in column vector form as:

\[ \mathbf{x} = (x_1^T, x_2^T, ..., x_p^T) \]  

\[ \mathbf{u} = (u_1^T, u_2^T, ..., u_p^T). \]

Using this notation, Eq. 10 can be rewritten in matrix form as,

\[
\begin{bmatrix}
X[0] \\
X[1] \\
\vdots \\
X[p]
\end{bmatrix} = 
\begin{bmatrix}
A & 0 & 0 & 0 \\
A^2 & AB & B & 0 \\
\vdots & \vdots & \vdots & \vdots \\
A^{p-1} & A^{p-2}B & \cdots & B
\end{bmatrix}
\begin{bmatrix}
X[0] \\
X[1] \\
\vdots \\
X[p]
\end{bmatrix} 
+ 
\begin{bmatrix}
B & 0 & 0 & 0 \\
AB & B & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots \\
A^{(p-1)}B & A^{p-2}B & \cdots & B
\end{bmatrix}
\begin{bmatrix}
U[0] \\
U[1] \\
\vdots \\
U[p-1]
\end{bmatrix} \]

Simplified this becomes,

\[ \mathbf{x} = \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u} \]

Figure 4-2, summarizes and explains the process of MPC prediction.

1. The initial condition is used for the state space vector (Eq. 10).
2. Using the initial values, the state values are calculated over the prediction horizon using Eq. 43. The prediction process is very well described by Eqs. 10, 11, and 12.
3. The next step is using the optimization tool to calculate the optimal control input, \( \mathbf{u}^* \), to minimize the error between the predicted output and actual output.
4. This control input is then implemented as the first step to calculate the output and the sampling is done again for next time step.

This section focused on using the initial value of the state space vector and deriving the predictions. The upcoming sections focus on deriving the cost function and the use of the optimization tool.
We will next implement MPC on the actual problem. This chapter defines and derives the parameters necessary for implementing the MPC algorithm for the purpose of controlling a UAV. The MPC formulation for the problem involves a quadratic optimization. This problem can then be defined with constraints or without constraints. The only thing that differentiates trajectory tracking and obstacle avoidance is the application of the constraints on the system.

### 3.4 Objective function

The objective function, also known as the cost function, is a critical component of using quadratic optimization for MPC. One must create a cost function and later minimize it based on the constraints provided by the problem. The cost function directly influences the accuracy of the
solution. With respect to this dissertation, the solution is symbolized as $u$. The optimal solution of the cost function is denoted by $u^*$. As mentioned in the previous chapter, the trajectory tracking part of the problem does not require any constraints. Thus, in order to develop a tracking application, we only need a cost function. Hence the focus of this chapter is on creating the cost function and applying it.

When used on a trajectory tracking problem, MPC simply follows the given trajectory as closely as possible. Hence, the cost function can be represented as the difference between the desired position and predicted position, the value of which should be minimized. Therefore, we define error as,

$$
e(x) = x(k) - xd(k)$$
$$
e(y) = y(k) - yd(k)$$
$$
e(z) = z(k) - zd(k)$$

(13)

where $x(k), y(k),$ and $z(k)$ are the predicted values of the trajectory positions and $xd(k), yd(k),$ and $zd(k)$ are the desired values of the position at time instant $k$. This is the linear form of the error. The major drawback of using a linear model is that it can penalize the deviation only in one direction. In other words, if the value of current position is less than the value of desired position; the error value will be negative. This might trick the optimizer in believing that the solution is optimal since the value of error will be less than zero. To overcome this, the square of the error is taken. But, the summing of $e_x^2(i), e_y^2(i),$ and $e_z^2(i)$ can produce high input action values and can make the system uncontrollable. Hence the value of the input actions need to be penalized as well. Taking these factors into consideration, the cost function that we use is $J$,

$$J = \left(\frac{1}{2}\right) \sum_{i=1}^{P} \left( q(e_{x(i)}^2 + e_{y(i)}^2 + e_{z(i)}^2) + s(u_{1(i-1)}^2 + u_{2(i)}^2 + u_{3(i)}^2 + u_{4(i)}^2) \right)$$

(14)
where \( e = (e_1, e_2, \ldots, e_p)^T \), \( q \) and \( s \) are parameters of the cost function which penalize input values versus output errors impacting how the system behaves. The error value at the first time instant is determined by the initial condition and is not dependent on the input \( u \). The above equation can be written in matrix form,

\[
J = \frac{1}{2} \sum_{i=1}^{p} (e_{x(i)}^T Q e_i + u_{i-1}^T P u_{i-1})
\]  

(15)

where \( Q \) and \( P \) are penalizing matrices. \( Q \) penalizes deviations from the given trajectory in terms of position and velocity whereas \( P \) matrix penalizes the input. The penalizing matrices should be positive definite to ensure that the cost function remains strictly convex. We can now introduce matrices \( \hat{Q} \) and \( \hat{P} \),

\[
\hat{Q} = \begin{bmatrix}
Q & \ldots & 0 \\
0 & Q & \vdots \\
0 & \vdots & Q
\end{bmatrix}
\quad \text{and} \quad
\hat{P} = \begin{bmatrix}
P & \ldots & 0 \\
0 & P & \vdots \\
0 & \vdots & P
\end{bmatrix}
\]  

(16)

This allows us to write the cost function equation as follows,

\[
J(u) = \left( \frac{1}{2} \right) (e^T \hat{Q} e + u^T \hat{P} u)
\]

\[
J(u) = \left( \frac{1}{2} \right) \left( (\hat{A} x_{[0]} + \hat{B} u - x_d)^T \hat{Q} (\hat{A} x_{[0]} + \hat{B} u - x_d) + u^T \hat{P} u \right)
\]

\[
J(u) = \left( \frac{1}{2} \right) \left( 2 (\hat{Q} \hat{B})^T \hat{A} x_{[0]} u + u^T \hat{B}^T \hat{Q} u - 2 (\hat{Q} \hat{B})^T x_d u + u^T \hat{P} u \right)
\]

\[
= \left( \frac{1}{2} \right) (2(\hat{Q} \hat{B})^T \hat{A} x_{[0]} u + u^T \hat{B}^T \hat{Q} u - 2 (\hat{Q} \hat{B})^T x_d u + u^T \hat{P} u)
\]

\[
= \left( \frac{1}{2} \right) (2(\hat{Q} \hat{B})^T \hat{A} x_{[0]} u + u^T \hat{B}^T \hat{Q} u - 2 (\hat{Q} \hat{B})^T x_d u + u^T \hat{P} u)
\]

(17)

The aim is to minimize the cost function \( J(u) \). The part of equations that do not depend on \( u \) can be ignored and we can rewrite the new cost function as,

\[
J(u) = \left( \frac{1}{2} \right) u^T (\hat{B}^T \hat{Q} \hat{B} + \hat{P}) u + (\hat{Q} \hat{B})^T (\hat{A} x_{[0]} - x_d) u
\]

(18)
In a compressed form the above equation can be rewritten as,

\[
J(u) = \left( \frac{1}{2} \right) u^T \bar{H} u + \bar{c} u
\]  

(19)

While the trajectory tracking problem does not require constraints, equation 19 can be used to find the optimal solution, represented as \( u^* \).

### 3.5 Unconstrained Quadratic Programming

The problem formulated above can be solved without constraints as long as the cost function \( J \) is convex. Because \( Q \) is positive semi-definite and \( P \) is positive definite, \( \bar{H} \) is a positive semi-definite matrix. We approach this problem by finding the gradient of the cost function, \( \nabla J \) [14]. According to [15], the gradient of the cost function is derived as,

\[
\nabla J( u^* ) = \bar{H} u^* + \bar{c} = 0
\]

\[
\bar{H} u^* = -\bar{c}
\]

\[
u^* = -\bar{H}^{-1} \bar{c}
\]  

(20)

Equation 20 is the analytical solution for \( u^* \). The presence of an analytic solution is one reason why Model Predictive Control is fast for on-board calculations. Because it is not a function of the system state and is completely dependent on the values such as \( q, p \) and system model, \( \bar{H}^{-1} \) can be easily calculated. The solution of MPC is however extremely sensitive to the values of gains \( q, p \) and the state space matrices of the system dynamics \( A, B, C, D \).

### 3.6 Introduction to MOANTOOL

To manage a scenario, involving obstacle avoidance constraints must be added. The constraints are imposed on the vehicle’s input and state to ensure safe maneuvers. The Slovak University of
Technology has recently developed a toolbox, called MOANTOOL, for applying these constraints to the system [16]. We use MOANTOOL to apply constraints within a Model Predictive Control framework for the purpose of obstacle avoidance. MOANTOOL also allows one to select their choice of optimizer for constraint satisfaction. MOANTOOL offers various optimizers that can be used for optimization. For this dissertation, we used a quadratic programming optimizer for different 2D problems. MOANTOOL can also be used to address obstacle avoidance problems optimally or sub-optimally. The subsections below briefly describe each approach, the details for each can be found in [16].

**Optimal Obstacle Avoidance** uses binary variables which making the problem a Mixed Integer Optimization problem [16]. Mixed-Integer formulation avoids obstacles but provides an optimal solution. These problems are non-convex but can be tackled efficiently for a modest amount of obstacles. For a large and moderate number of obstacles sub-optimal obstacle avoidance must be used.

**Sub-Optimal Obstacle Avoidance** uses time varying output constraints. The term sub-optimal is used in context with the trajectories that might result from heuristically determining which direction the vehicle should deflect. In other words, the system might not be able to track the reference trajectory as closely as possible. Depending on the current position of the obstacle and current position of the vehicle, a get-around position is calculated heuristically. Once the direction is fixed, a constraint set is generated which includes $y_k$ such that vehicle does not collide with the obstacle. The advantage of using this method is that the constraint remains convex and the runtime is shorter. The figure 3-3 below shows the optimal trajectory generated by solid circles, whereas the blue circle denote the possible sub-optimal trajectory achieved by time varying constraints.
Figure 3-3: The idea of obstacle avoidance. The reference trajectory is the dashed line. At every time step of the prediction horizon, the red line indicates the convex output constraints $Y_k$ are updated so that there is no collision with the obstacle $O$ [16].

### 3.6.1 Vehicle Path Planning and Obstacle Avoidance Problem Setup

The vehicle dynamics are governed by the discrete time state update equation and output as mentioned in the previously in equation (8) and (9) from Chapter 3. Before getting into the application, it is crucial to understand how the vehicle avoids obstacles inside the MOANTOOL environment. The major assumption is that the obstacles are all polytopes in the half-space representation [16]. A convex space can be defined as the space in which a line segment connecting any two points lies inside the space. A plane dividing a space results in two half-spaces. Half-spaces are convex in nature. The intersection of such half-spaces results in polytopes. In half-space representation, obstacles are assumed polytopes or bounded polyhedra in order to keep the optimization convex in nature.
MOANTOOL can also be described as a high level language which allows the user to design MPC problem, perform the optimization and visualize the results in simulation. To perform and automate these tasks, MOANTOOL uses an architecture of classes. MOANTOOL utilizes four classes: the agent, obstacle, planner, and simulator class. These classes allow the user to interact with the tool.

- **The Agent Class** allows the user to define a vehicle’s dynamics, the constraints on the system such as the minimum and maximum values of the states, input and output and as well as the physical dimensions. MOANTOOL is capable of handling vehicles governed by both linear and nonlinear dynamics. All the parameters relating to the dynamics of the vehicle can be defined in the agent class. The dimensions of the states is denoted by $nx$, and the dimension of input and output is denoted by $nu$ and $ny$. For quadcopter, $nx$ is 6 which is the input dimension and $nu$ is 2 torques/rotor rpm. The dimension of the output is
corresponding to input $n_y$ is 6. The number for the prediction horizon $N$ is specified in this class. MOANTOOL allows the user to create constraints. The agent class lets the user set constraints in terms of the minimum and maximum values of the states, the input and the output. The dimensions of the quadcopter can also be given by in terms of height and width.

- **The Obstacle Class** is used to initialize and manage obstacles in MOANTOOL. The number of obstacles and their size and position are specified.

- **The Planner Class** solves the optimization problem. Users can specify the use of specific solvers/optimizers such as Gurobi, Cvx, and Quadprog depending on the type of problem. This dissertation used Quadprog.

- **The Simulator Class** allows the user to visualize the results. The simulator class takes the planner as input. In this dissertation, the quadcopter starts from $(0, 0)$. MOANTOOL allows the users to initialize the problem at a definite point or the definite value of the state space vector and then can run the simulation from that point.

### 3.7 Summary

This chapter has presented a detailed derivation of Model Predictive Control and explained how MPC is formulated for our problem of creating an autonomous quadcopter that avoids obstacles. We have shown how cost functions play a role with the MPC framework and how MPC can be formulated as a quadratic programming problem which is solved for in terms of the cost function. We have also detailed the creation of a cost function for the UAV trajectory planning and obstacle avoidance problem. Because an MPC based approach to obstacle avoidance is difficult to compute directly, we have introduced the constrained optimization tool, MOANTOOL, to assist with trajectory optimization in situations where obstacles need to be avoided.

This chapter contributes an MPC framework and cost function that can be applied to a swarm of quadcopters avoiding obstacles. Before implementing a control algorithm on hardware,
it is crucial to test it in simulation in order to have a better understanding of how the system will respond. The next chapter presents simulation results using the techniques developed in this chapter using MPC and MOANTOOL.
Chapter 4

Results of Simulation

4.1 Chapter Overview

The following chapter presents the results of simulations of various experiments performed to test MPC algorithm on the dynamic model of UAV. The experiments were performed using multiple UAVs to show its application in swarm robotics. The first set of results demonstrates the capability of the Linear MPC to track a complicated trajectory. A number of experiments are performed considering different scenarios and increasing order of complexity. Before using MPC, a stabilized quadcopter model was developed and a PD control technique was used to achieve the stability. This model was then transformed into state-space form and used for MPC trajectory tracking. The results include the simulation obtained in MOANTOOL which showcase the obstacle avoidance capability of Model Predictive Control. MATLAB simulation tool was used in this dissertation to perform all the experiments. For the stabilization experiment no external tool was required and all the software was written in MATLAB. A laptop with 2.5-GHz Intel Core i5-2520M and a 4GB RAM was used for performing simulations.

4.2 Trajectory tracking using Model Predictive Control

The primary objective of this dissertation was to implement Model Predictive Control to control quadcopters in order to track complex trajectories and to demonstrate the system’s obstacle avoidance capability. This section presents the results of a trajectory tracking simulation performed on the quadcopters. For these experiments, the state space matrices $A, B, C, D$ were obtained using
equations 27 and 30 as explained in figure 2.1. Once these matrices are obtained, the state space vector was initialized with the linear positions and the desired trajectory is specified. The MPC problem formulation with prediction is implemented using equations 10, 11, 12, 20 as explained in figure 3.2. For performing the optimization, the first value of control input $u^*$ is calculated using equation 20 and is then substituted with equation 19 and is later updated at every iteration. The quadcopter model is difficult to control and the input action results in chaotic motion if the input is not penalized. The deviations in the position and the input actions were thus penalized. During experimentation it was found that the value of the elements in the $Q$ matrix which penalizes the position deviation had to be relatively high, while the value of elements of $P$ matrix which penalizes the input were less than 1. The detailed values of parameters used for each experiment are included in Appendix A.

Another important parameter which had a significant effect on the performance was the prediction horizon. The larger the prediction horizon, the better the trajectory was tracked. For the experiments, the value of the prediction horizon was 100. As mentioned in Chapter 1, one recent application of autonomous quadcopters is swarm coordination to perform choreographed moves. Model Predictive Control can be used to make a swarm of quadcopters perform various moves and form aesthetically appealing shapes which could be used for aerial performances. To examine this hypothesis, various tests were performed using 3D trajectories of irregular shapes in order to see if a quadcopter could accurately track these trajectories.

The experiments we performed use irregular reference trajectories while measuring the performance of the MPC based controller in terms of trajectory accuracy. For these experiments, it is critical to also verify that the control inputs are actually feasible by the quadcopter. Penalizing constants have a major impact on the performance of the MPC controller. We, therefore, vary the penalizing the constants $q$ and $p$ which, as discussed in section 3.4, $p$ penalizes deviations and $q$ penalizes inputs. Because the MPC based controller uses unconstrained optimization, the control inputs that are obtained for very high penalizing values are also relatively high and might be
infeasible for the actual quadcopter. Hence, it is important to find the range of parameters that result in reasonable control inputs to track the given reference trajectory. The experiments lead us to a range of values that result in the quadcopter following the trajectory as closely as possible.

To examine different penalizing values \(q\) and \(p\) for the MPC controller we used three test cases. The first test case uses a value of the penalizing constants that give accurate trajectory tracking, which was found via trial-and-error. The second test case considers a variation in the constants penalizing trajectory deviations while keeping the constants that penalize the inputs the same. The third case accounts for variation in the constants penalizing the inputs.

**Experiment 1:**

Figure 4-1 below shows a quadcopter following an infinity loop trajectory and the trajectory that results from all three test cases. This was an early experiment that was conducted in order to understand and test whether Model Predictive Control could handle these irregular trajectories. The desired trajectory \(x, y, z\) is given as the function of sine and cosine with respect to time. The experiment parameters are provided in Appendix A.
Figure 4-1(a): Case 1: A single quadcopter following infinity loop trajectory, the blue dot denotes the quadcopter and the reference trajectory is shown in red.

Figure 4-1(b): Case 2: A single quadcopter following infinity loop trajectory, the blue dot denotes the quadcopter and the reference trajectory is shown in red.
Figure 4-1(c): Case 3: A single quadcopter following infinity loop trajectory, the blue dot denotes the quadcopter and the reference trajectory is shown in red.

Table 1: Values of penalizing terms

<table>
<thead>
<tr>
<th>$Q$ matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 1 Values</th>
<th>Case 1 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(1,1)$</td>
<td>500</td>
<td>500</td>
<td>150</td>
</tr>
<tr>
<td>$Q(3,3)$</td>
<td>500</td>
<td>500</td>
<td>150</td>
</tr>
<tr>
<td>$Q(5,5)$</td>
<td>1000</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$ matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 1 Values</th>
<th>Case 1 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(1,1)$</td>
<td>0.4</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>$P(2,2)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(3,3)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(4,4)$</td>
<td>1.0</td>
<td>1.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 1 shows the values used for each case. For the first test case, deviations are penalized with higher values than the values used for penalizing the input. For higher values of $q$, it was seen that the quadcopter tracks the reference trajectory very accurately as seen from figure 4-1 (a). For test case 2, the value of the penalizing constants for deviation $q$ was further reduced to see the variation in trajectory. The resultant trajectory obtained can be seen in figure 4-1 (b). Test case 3
changes the penalizing constants applied to the inputs and the deviations. As depicted in the reference trajectory in figure 4-1 (c) the quadcopter becomes unstable and moves farther away from the reference trajectory.

Figure 4-2(a): The change in control input $u_1$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 1 parameter values.

Figure 4-2(b): The change in control input $u_1$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 2 parameter values.
Figure 4-2(c): The change in control input $u_1$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 3 parameter values.

In figures 4-2(a), 4-2(b), 4-2(c), we can see the variation of control input $u_1$ which is responsible for the upward motion of the quadcopter. For this particular quadcopter model, the term $u_1$ is thrust ($T$). The value of the control input varies between -3 and 3 for test case 1 and the final value of the control input approaches 0.7 by the end of the simulation as seen from figure 4-2 (a). For test case 2, the variation of the control input $u_1$ is depicted in the top right graph in figure 4-2 (b). Notice that the value is slightly decreased from the previous case as the deviation is less penalized and penalizing constants for the inputs are kept the same. The figure 4-2(c) shows variation of control input $u_1$ for the test case 3. Here the quadcopter deviations are lightly penalized the quadcopter produces higher control inputs and goes further away from the reference trajectory into a chaotic motion.
Figure 4-3 (a): The change in control input $u_2$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 1 parameter values.

Figure 4-3 (b): The change in control input $u_2$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 1 parameter values.
Figure 4-3(c): The change in control input $u_2$ with respect to time can be seen for a single quadcopter following an infinity loop trajectory using the case 1 parameter values.

The figure 4-3(a) depicts the control input $u_2$ denoted torque (given in Newton-meter Nm) for test case 1 which is responsible for the roll action of the quadcopter oscillates between -100 and 100 Nm. The value of control input can be seen fluctuating near zero at the end of the simulation. Since the deviations are heavily penalized and control inputs are lightly penalized, the quadcopter follows the trajectory but there are extreme fluctuations in control inputs. Providing such high variation in the control input in a short amount of time is not be feasible for the hardware. For test case 2, the roll torque $u_2$ is in the range of -60 and 60 Nm (figure 4-3(b)). For test case 3, the control input $u_2$ is oscillating with high magnitude for larger amount of time during the simulation. The smaller the constants, the lighter the constraints. Hence as seen from figure 4-3(c), the control input $u_2$ oscillated from a shorter amounts of time with lesser amplitude initially but as the simulation progresses the value of the control input $u_2$ fluctuates between -100 and 100.

The variation of $u_3$ shows the exact same trend as $u_2$ for all three test cases. The control input $u_4$, which is responsible for the yaw motion, is zero for all these scenarios as there is no lateral
motion for the respective reference trajectories. Hence, control input $u_4$ is not shown.

Figure 4-4(a): Root mean squared error (in meters) of the position in $x$, $y$ and $z$ with respect to time for case 1 parameter values for a given trajectory.

Figure 4-4(b): Root mean squared error (in meters) of the position in $x$, $y$ and $z$ with respect to time for case 2 parameter values for a given trajectory.
To validate the accuracy of the experiment Root Mean Squared Error (RMSE) is plotted in figures 4-4(a), 4-4(b) and 4-4(c) between the reference trajectory provided and the resultant trajectory obtained. For test case 1, RSME value falls between 0 and 0.35 as seen in figure 4-4(a). The RMSE in x and y are exactly the same and graph depicts them as overlapping. The lower the value of RSME the better the performance. The linear velocity in x, y and z are included in appendix B as figure 1, 2 and 3 for case 1,2 and 3 respectively. For test case 2, the maximum RMSE for z was around 0.7 meters which can be seen in figure 4-4(b). The difference between the resultant trajectory and the reference trajectory is increased as seen from figure 4-4(b). The error seems to be increasing with decrease in the value of penalizing constants $q$. As the error increases, RMSE increases. The error is the largest for test case 3, reaching a value of 1.7 meters as seen in figure 4-4 (c). Hence, we can conclude that the parameter settings for this third test case are undesirable. Because the parameter settings for the first test case result in unrealistic torques, we conclude that, overall, of the three test cases, the parameter settings for test case 2 would result in reasonable performance with achievable torques.
**Experiment 2:**

Next we tried irregular trajectories with multiple quadcopters. Two quadcopters were used and were made to follow a taco shaped trajectory (figure 4-13).

Figure 4-5(a): Two quadcopters following a taco-shell trajectory for case 1. The reference trajectory is shown in red and the resultant trajectory is shown in black.
Figure 4-5(b): Two quadcopters following a taco-shell trajectory for case 2. The reference trajectory is shown in red and the resultant trajectory is shown in black.

Figure 4-5(c): Two quadcopters following a taco-shell trajectory for case 3. The reference trajectory is shown in red and the resultant trajectory is shown in black.
Table 2 : Values of penalizing terms

<table>
<thead>
<tr>
<th>$Q$ matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Case 3 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(1,1)$</td>
<td>500</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>$Q(3,3)$</td>
<td>500</td>
<td>250</td>
<td>150</td>
</tr>
<tr>
<td>$Q(5,5)$</td>
<td>1000</td>
<td>150</td>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$ matrix diagonal elements</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(1,1)$</td>
<td>0.4</td>
<td>0.4</td>
<td>0.87</td>
</tr>
<tr>
<td>$P(2,2)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(3,3)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(4,4)$</td>
<td>1.0</td>
<td>1.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 2 shows the values used for each case. For the first test case the deviations are penalized with higher values than the values used for penalizing the input. For higher values of $q$, it was seen that the quadcopter tracks the reference trajectory very accurately as seen in figure 4-5(a). For test case 2, the value of the penalizing constant for deviation, $q$, is reduced and the values of $p$ are kept constant. As a result difference between the resultant trajectory and the reference trajectory increases as seen from figure 4-5(b). Test case 3 changes the penalizing constants applied to both the inputs and the deviations. As depicted in the reference trajectory in the figure 4-5(c), the quadcopter becomes unstable and moves farther away from the reference trajectory.

Figure 4-6(a): The change in control input $u_t$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 1 parameter values.
Figure 4-6(b): The change in control input $u_1$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 2 parameter values.

Figure 4-6(c): The change in control input $u_1$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 3 parameter values.

In figure 4-6, we can see the variation of control input $u_1$ which is thrust ($T$). The value of the control input varies between -2 and 5 for test case 1 as seen from figure 4-6(a). The thrust values range from -2 to 2 with a sudden peak at 4 in the beginning. For test case 2, the variation of the control input $u_1$ is depicted in the figure 4-6(b). The value fluctuates between -2.5 to 2.5. The figure
4-6(c) shows variation of control input $u_1$ for the test case 3. Here the quadcopter deviations are lightly penalized the quadcopter produces higher control inputs and goes further away from the reference trajectory into a chaotic motion.

Figure 4-7(a): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 1 parameter values.

Figure 4-7(b): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 2 parameter values.
Figure 4-7(c): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory using the case 3 parameter values.

For case 1, the values of roll torques are extremely high i.e. within $-80 \text{ Nm}$ to $80 \text{ Nm}$ as seen from figure 4-7(a). The quadcopter is able to follow the reference trajectory but the quadcopter seems unstable with such high fluctuations in the torque. The values of constants penalizing the deviation is too high which puts a lot of pressure on the dynamics of the quadcopter. This results in the high fluctuating values for the torques as the quadcopter struggles to follow the reference trajectory. For case 2, the variation in torque can be observed in figure 4-7(b) which shows smaller values of torques within the range $-40$ to $40 \text{ Nm}$. For case 3, figure 4-7(c) shows the torque varying from $-100$ to $100 \text{ Nm}$. The control input $u_4$, which is responsible for the yaw motion, is zero for all these scenarios as there is no lateral motion for the respective reference trajectories. Hence, control input $u_4$ is not shown.
Figure 4-8(a): Root mean squared error (in meters) of the position in $x$, $y$ and $z$ with respect to time for case 1 parameter values for the given trajectory.

Figure 4-8(b): Root mean squared error (in meters) of the position in $x$, $y$ and $z$ with respect to time for case 2 parameter values for a given trajectory.
Figure 4-8(c): Root mean squared error (in meters) of the position in $x$, $y$ and $z$ with respect to time for case 3 parameter values for the given trajectory.

The values of pitch torque, $u_3$, is the same as $u_2$ and hence are not shown. The linear velocity in $x$, $y$ and $z$ are included in appendix B as figure 4. The figure shows absolute linear velocity of the quadcopter for the case scenario the velocity in $x$ approaches $9 \text{ m/s}$.

For case 1, the value of RMSE is acceptable as is very low between 0-0.35 meters as seen from figure 4-8(a). For case 2, it can be seen from the figure 4-8(b) that the quadcopters slightly deflect from the given trajectories but the RMSE is still less than 0.5 meters. For case 3, the quadcopters become completely unstable with these values and RMSE reaches approximately 5.0 meters as observed from the figure 4-8(c). The quadcopter error dramatically increases after increasing the value of penalizing constants for the same input.

**Experiment 3:**

To test whether MPC is capable of handling multiple quadcopters another experiment using taco shaped trajectory was performed but this time with ten quadcopters (Figure 4-9). For these experiments, the desired trajectory was given as a function of sine, cosine, and tangent with respect
to time. Although ten quadcopters were used in this experiment, only the results for the first quadcopter are depicted in this section.

Figure 4-9(a): Ten quadcopters in a stacked taco-shell trajectory for case 1. The reference trajectory is shown in red and the resultant trajectory is shown in black.

Figure 4-9(b): Ten quadcopters in a stacked taco-shell trajectory for case 2. The reference trajectory is shown in red and the resultant trajectory is shown in black.
Figure 4-9(c): Ten quadcopters in a stacked taco-shell trajectory for case 3. The reference trajectory is shown in red and the resultant trajectory is shown in black.

Table 3 : Values of penalizing terms

<table>
<thead>
<tr>
<th>Q matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Case 3 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(1,1)$</td>
<td>500</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>$Q(3,3)$</td>
<td>500</td>
<td>300</td>
<td>150</td>
</tr>
<tr>
<td>$Q(5,5)$</td>
<td>1000</td>
<td>350</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Case 3 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(1,1)$</td>
<td>0.4</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>$P(2,2)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(3,3)$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(4,4)$</td>
<td>1.0</td>
<td>1.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 3 shows the values used for each case. For the first test case, deviations are penalized with higher values than the values used for penalizing the input. For higher values of $q$, it was seen that the quadcopter tracks the reference trajectory very accurately as seen from figure 4-9(a). For test case 2, the value of the penalizing constants for deviation $q$ was further reduced to see the variation in trajectory. The resultant trajectory obtained can be seen in figure 4-9(b). Test case 3 changes the penalizing constants applied to the inputs and the deviations. As depicted in the reference trajectory from figure 4-9(c) the quadcopter becomes unstable and moves farther away from the reference trajectory.
Figure 4-10(a): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using case 1 parameter values.

Figure 4-10(b): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using the case 2 parameter values.
Figure 4-10(c): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using the case 2 parameter values.

In figure 4-10, the variation of control input $u_1$ which is thrust ($T$) is shown. The figure 4-10(a) shows $u_1$ varying with respect to time for the values of penalizing constants for test case 1. The value of the control input varies between -1.5 and 3 for test case 1. Although the control input is constantly fluctuating, it reduces in amplitude by the end of the simulation as the thrust approaches a value between 0 and 1. For test case 2, the variation of the control input $u_1$ is depicted in figure 4-10(b). The value fluctuates between -1.5 to 2.5. Figure 4-10(c) shows the variation of control input $u_1$ for the test case 3. Here the quadcopter deviations are lightly penalized the quadcopter produces higher control inputs and goes further away from the reference trajectory into a chaotic motion and that results in the thrust reaching a maximum value of 4 Newtons.
Figure 4-11(a): The change in control input $u_3$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using the case 1 parameter values.

Figure 4-11(b): The change in control input $u_3$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using the case 2 parameter values.
Figure 4-11(c): The change in control input $u_2$ with respect to time can be seen for a quadcopter following a taco-shell loop trajectory when 10 quadcopters together are tracking the trajectory using the case 3 parameter values.

Figure 4-11 shows the variation of control input $u_3$ with respect to time. For test case 1, the variation in control input $u_3$ which corresponds to the pitch torque is between -50 to 50 Nm as seen in figure 4-11(a). For case 2, the variation in control input $u_3$ which corresponds to the pitch torque is between -60 to 60 Nm as seen in the figure 4-11(b). For case 3, the variation in control input $u_3$ which corresponds to the pitch torque is between -50 to 50 Nm as seen in the figure 4-11(c). The graph shows oscillations after every 2.5 seconds as supposed to the previous case where oscillations are observed after every 3.5 seconds. The variation of control input $u_2$ follows a similar trend with similar peak values of the torque. Control input $u_4$ for this case was zero. The linear velocity in $x$, $y$ and $z$ are included in appendix B as figure 5.
Figure 4-12(a): Root mean squared error (in meters) of the position in x, y and z with respect to time for case 1 parameter values for the given trajectory.

Figure 4-12(b): Root mean squared error (in meters) of the position in x, y and z with respect to time for case 2 parameter values for the given trajectory.
For case 1, the RMSE in x and y is less than 0.1 meters whereas the RMSE in z is reaches 0.4 meters as seen from the figure 4-12(a). For case 2, the trajectory from figure 4-9 can be seen fluctuating a little bit from the reference trajectory as the value of constant penalizing the deviation is reduced. For the chosen value of constants as seen from table 5, the quadcopter’s RMSE value stayed below 0.7 meters as observed from figure 4-12(b). For test case 3, the error RMSE approached a value of 2.5 meters resulting in chaotic motion as seen in figure 4-12(c).

**Experiment 4:**

The previous experiments considered a trajectory that was a function of sine, cosine and tangent with respect to time. This experiment applies MPC to multiple quadcopters traveling through a series of waypoints. This task is valuable for military and search and rescue applications. For instance, the quadcopters could be tasked with writing out an SOS message during an emergency. Figure 4-13 displays the result of a simulation where three quadcopters were used to spell a word.
Figure 4-13(a): Three quadcopters spelling a word in 3-D space for case 1 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.

Figure 4-13(b): Three quadcopters spelling a word in 3-D space for case 2 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.
Figure 4-13(c): Three quadcopters spelling a word in 3-D space for case 3 parameters value, the blue dot denotes the quadcopter and the reference trajectory is shown in red.

Table 4: Values of penalizing terms

<table>
<thead>
<tr>
<th>Q matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Case 3 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(1,1)</td>
<td>500</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Q(3,3)</td>
<td>500</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Q(5,5)</td>
<td>1000</td>
<td>250</td>
<td>50</td>
</tr>
<tr>
<td>P matrix diagonal elements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P(1,1)</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>P(2,2)</td>
<td>0.001</td>
<td>0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>P(3,3)</td>
<td>0.001</td>
<td>0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>P(4,4)</td>
<td>1.0</td>
<td>10.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>

Table 4 shows the values used for each case. The first test case again examines higher values of $q$ and shows that the quadcopter accurately tracks the reference trajectory in the figure 4-13(a). For test case 2, the value of the penalizing constants for deviation $q$ are reduced but the values of coefficients penalizing the inputs are kept same as the first case. For this case, the distance between the resultant trajectory and the reference trajectory increases as seen from figure 4-13 (b). Test case 3 changes the penalizing constants applied to the inputs and the deviations. As depicted in the reference trajectory in the figure 4-13(c) the quadcopter becomes unstable and moves farther away from the reference trajectory.
Figure 4-14(a): The change in control input $u_1$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 1 parameter values.

Figure 4-14(b): The change in control input $u_1$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 2 parameter values.
Figure 4-14(c): The change in control input $u_1$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 3 parameter values.

The graph of the thrust (figure 4-14) shows constants and sharp peaks. For test case 1, the value of control input $u_1$ in the figure 4-14 (a) is oscillating between -0.6 to 0.6 and eventually approaches zero. This indicates that the quadcopter is stabilizing in the $z$ direction at the end of the simulation also can be seen in figure 4-14(b). This behavior is also depicted in the velocity plot (Appendix B as figure 7). This plot shows the value of velocity in the $z$ direction approaching zero. For case 3, the control input $u_1$ shows an increase in the value of thrust -3 to 3 Newtons from figure 4-14(c) because the inputs are lightly penalized. The control input $u_2$ i.e. the roll torque is zero. There is no movement in $y$-axis.
Figure 4-15(a): The change in control input $u_3$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 1 parameter values.

Figure 4-15(b): The change in control input $u_3$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 2 parameter values.
Figure 4-15(c): The change in control input $u_3$ with respect to time can be seen for a quadcopter following an ‘L’ shaped trajectory using the case 3 parameter values. The control input, $u_3$, responsible for the pitch torque for test case 1, starts oscillating after 3 seconds and keeps on increasing in the amplitude in figure 4-15 (a). The linear velocities in $x$, $y$ and $z$ are included in Appendix B as figure 6. The value of the control input $u_3$ oscillates between -100 to 100 for case 2 in the figure 4-15(b). A similar trend as seen in the previous test cases with the value varying between -100 to 100 Nm is depicted in figure 4-15(c). The linear velocity for this test case are included in Appendix B as figure 8.
Figure 4-16(a): Root mean squared error of the position in $x$, $y$ and $z$ for the first quadcopter following the trajectory of the alphabet L with respect to time for case 1.

Figure 4-16(b): Root mean squared error of the position in $x$, $y$ and $z$ for the first quadcopter following the trajectory of the alphabet A with respect to time for case 1.
Figure 4-16(c): Root mean squared error of the position in x, y and z for the first quadcopter following the trajectory of the alphabet N with respect to time for case 1.

Because the trajectory for this experiment is a function of waypoints we must interpolate the reference waypoints to generate a trajectory. For this experiment the controller’s performance is very accurate. As depicted in figure 5-24, the RMSE is within the range 0- 0.035 meters for the first quadcopter following the ‘L’ trajectory in the figure 4-16(a). Similarly, for other two quadcopters following the trajectories ‘A’ and ‘N’ the RMSE is below 0.09 meters (figure 4-16 (b) and figure 4-16 (c) respectively).
Figure 4-17(a): RMSE of the position values in $x$, $y$ and $z$ of the quadcopter trying to follow the trajectory of L case 2.

Figure 4-17(b): RMSE values of the position in $x$, $y$ and $z$ of the quadcopter trying to follow the trajectory of A case 2.
Figure 4-17(c): RMSE of the position in x, y and z values of the quadcopter trying to follow the trajectory of N case 2.

For test case 2 as seen in figure 4-17(b), the RMSE value is increases when compared to the first test case as seen in figure 4-17(b) approaching 0.9 meters.

Figure 4-18(a): RMSE of the position in x, y and z values of the quadcopter trying to follow the trajectory of L case 3.
Figure 4-18(b): RMSE of the position in $x$, $y$ and $z$ values of the quadcopter trying to follow the trajectory of A case 3.

Figure 4-18(c): RMSE of the position in $x$, $y$ and $z$ values of the quadcopter trying to follow the trajectory of N case 3.
For test case, the value of constants penalizing the matrices are comparatively smaller and 
the value of constants for penalizing the input were increased. This drastically increased the RMSE 
value which was 3.0 meters as seen in figure 4-18(c).

Experiment 5:

As a final experiment we used waypoints to move the entire swarm of quadcopter along a particular 
axis. For this experiment the quadcopters create a heart shape. Once the heart shape is formed the 
quadcopters move along the x-axis. There were 23 quadcopters used for this experiment. The 
behavior of three randomly chosen quadcopters is presented in detail here. Quadcopters 5, 12 and 
20 were chosen. The behavior of these quadcopters was again examined by varying the penalizing 
terms.

![Figure 4-19(a): Quadcopter trying to form swarm of a heart shaped for case 1 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.](image)
Figure 4-19(b): Quadcopter trying to form swarm of a heart shaped for case 2 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.

Figure 4-19(c): Quadcopter trying to form swarm of a heart shaped for case 3 parameters. The blue dots show the actual trajectory of the chosen quadcopters. Red dots are the reference trajectories.
Table 5: Values of penalizing terms

<table>
<thead>
<tr>
<th>Q matrix diagonal elements</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Case 3 Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(1,1)</td>
<td>500</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>Q(3,3)</td>
<td>500</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>Q(5,5)</td>
<td>1000</td>
<td>250</td>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P matrix diagonal elements</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P(1,1)</td>
<td>0.4</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td>P(2,2)</td>
<td>0.001</td>
<td>0.001</td>
<td>0.003</td>
</tr>
<tr>
<td>P(3,3)</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>P(4,4)</td>
<td>1.0</td>
<td>5.0</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Table 5 shows the values used for each case. The first test case the deviations are heavily penalized while the inputs were lightly penalized. The same first case that is used previously for experiment 4 is used here and these penalizing constants results in an accurate trajectory as observed from figure 4-19(a). For test case 2, the value of the penalizing constants for deviation $q$ are reduced in value. As a result, the distance between the resultant trajectory and the reference trajectory is increased as seen from figure 4-19(b). Test case 3 changes the penalizing constants applied to the inputs and the deviations. As depicted in the reference trajectory from the figure 4-19(c) the quadcopter becomes unstable and moves farther away from the reference trajectory.

Figure 4-20(a): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 1 parameter values.
Figure 4-20(b): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 2 parameter values.

Figure 4-20(c): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 1 parameter values.

For test case 1, it can be seen in figure 4-20(a) after 3 seconds the quadcopter’s thrust stabilizes as it is approaching zero. This can again be verified using the velocity plot included in Appendix B figures 9, 10, and 11. This behavior is also observed in quadcopter 12 and quadcopter
20. The figure 4-20(b) shows the variation of control input $u_1$, the quadcopter’s thrust for test case 2. The thrust is also approaching zero at the end of the simulation. The figure 4-20(c) shows the value of the variation of control input $u_1$ for test case 3. The value of the control input $u_1$ for this simulation ranges between -1.5 to 1 Newtons for this case.

Figure 4-21(a): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 1 parameter values.
Figure 4-21(b): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 2 parameter values.

Figure 4-21(c): The change in control input $u_1$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 1 parameter values.
The figure 4-21 (a) depicts the control input $u_2$ the roll torque for test case 1 which oscillates between -100 and 100 Nm. The value of this control input fluctuates near zero at the end of the simulation. Because the deviations are heavily penalized and control inputs are lightly penalized, the quadcopter follows the trajectory but there are extreme fluctuations in control inputs. Providing such high variation in the control input in a short amount of time is not be feasible for a real quadcopter. For test case 2, the roll torque, $u_2$, is in the range of -60 and 60 Nm (figure 4-21 (b)). For test case 3, the control input $u_2$ oscillates strongly for most of the simulation. The smaller the constants, the lighter the constraints. Hence, as seen in the figure 4-21(c), the control input $u_2$ initially oscillated for shorter amounts of time with less amplitude initially but as the simulation progresses the value of the control input $u_2$ fluctuates between -2000 and 2000.

![Figure 4-21(a): The change in control input $u_2$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 1 parameter values.](image)
Figure 4-22(b): The change in control input $u_3$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 2 parameter values.

Figure 4-22(c): The change in control input $u_3$ with respect to time can be seen for a quadcopter following the specified trajectory using the case 3 parameter values.

The variation of the control input $u_3$ can be seen in the figure 4-22(a), (b), (c) for all three cases. The values of roll torque oscillates between -100 and 100 Nm and pitch torque oscillates
between the -200 and 200 Nm for case 1 as seen from figure 4-22(a). Although, the control inputs \( u_2 \) and \( u_3 \) show similar trends the range of variation is different for cases 1 and 2. The control input \( u_2 \) varies from -60 to 60 Nm in figure 4-22(b) whereas the control input \( u_3 \) varies -300 to 300 Nm in figure 4-22(c).

![Graph showing RMSE for position in x, y, and z of quadcopter number 5 for case 1 plotted against time.](image)

Figure 4-23(a): RMSE for the position in x, y and z of quadcopter number 5 for case 1 plotted against time.
Figure 4-23(b): RMSE for the position in $x$, $y$ and $z$ of quadcopter number 12 for case 1 plotted against time.

Figure 4-23(c): RMSE for the position in $x$, $y$ and $z$ of quadcopter number 20 for case 1 plotted against time.
The RMSE of for test case 1 for quadcopter 5, 12 and 20 are shown in figure 4-23(a), figure 4-23(b) and figure 4-23(c) respectively. The RMSE value is below 0.7 meters for all three quadcopters.

Figure 4-24(a) : RMSE for the position of quadcopter number 5 for case 2 plotted against time.

Figure 4-24(b) : RMSE for the position in x, y and z of quadcopter number 12 for case 2 plotted against time.
Figure 4-24(c): RMSE for the position in $x$, $y$ and $z$ of quadcopter number 20 for case 2 plotted against time.

For test case 2 the maximum RMSE for quadcopter 5 is 0.5 meters, for quadcopter 12 the max error 1.2 meters and for quadcopter 20 it is approximately 0.8 meters (figure 4-24 (a), 4-24 (b), and 4-24 (c) respectively). The maximum RMSE is initially in the $z$ direction. But as the simulation proceeds the RMSE in $x$ and $z$ goes to zero. Hence, the thrust also goes to zero as the end of the simulation approaches.
Figure 4-25(a): RMSE for the position in \( x \), \( y \) and \( z \) of quadcopter number 5 for case 3 plotted against time.

Figure 4-25(b): RMSE for the position in \( x \), \( y \) and \( z \) of quadcopter number 12 for case 3 plotted against time.
Figure 4-25(c): RMSE for the position in $x$, $y$ and $z$ of quadcopter number 20 for case 3 plotted against time.

The RMSE of quadcopter 5, 12 and 20 are respectively shown in figure 4-25 (a), figure 4-25(b) and figure 4-25(c) respectively for test case 3. The values of linear velocities are extremely high for the quadcopters. The velocity in $x$ direction and $z$ direction for all the quadcopters are attainable velocities. But the maximum velocity in $y$ direction for quadcopter 5 is 160 m/s, which is an unattainable speed for the quadcopter. The linear velocity in $x$, $y$ and $z$ direction are included in appendix B as figure 15, 16 and 17.

**Trajectory Tracking Conclusions:**

A total of 5 experiments were performed to understand how a MPC controller reacts to a change in the value of penalizing constants. Now, to further understand and have a clear picture of the correlation of control input and error in the trajectory with the penalizing constants, the following analysis was performed where approximately 20 data points were collected to establish the trend between the variations.
To summarize the series of experiments that were conducted, we can now plot the variation of control inputs and trajectory errors with respect to the change in the penalizing constants. The variation of control input \( u_1 \) which is thrust has been previously plotted in the figure 4-10, 4-11, 4-12 for experiment no. 1 according to the time history. The figure 4-26 shows the variation of thrust with respect to the parameter \( P(1,1) \). For the purpose of obtaining the correlation of the control input (\( u_1 \)) and parameter \( P(1,1) \), the root mean squared (RMS) value of the thrust vector for each iteration of the experiment 1 was used. The trend in the plot can also be visualized to conclude that the value of control input increases as we increase the value of parameter \( P(1,1) \). The higher the value of penalizing constant, the higher will be the control input. Assuming the thrust by weight ratio of the quadcopter is 2.5, a quadcopter weighing 0.498 kg will produce the maximum thrust of 1.24 kg. As seen from the plot, to attain a trust of 1.24 or less than 1.24 kg, the value of \( P(1,1) \) should be between 0.1 to 0.5. The figure 4-26 can be used by control system designers to set the value of parameter \( P(1,1) \) based on the required amount of thrust. For a lot of experiments, the ideal value that was taken was 0.4. This can be seen in table 11.
Figure 4-27: The variation of Root Mean Squared Error of trajectory in x direction with respect to $Q (1,1)$

Figure 4-27 shows the variation of root mean squared error (RMSE) of trajectory in $x$ direction and how it reduces by increasing the value parameter $Q (1,1)$ which penalizes deviations in $x$-direction. It was found that the RMSE is extremely sensitive to the value of penalizing constant $Q (1,1)$ between the range 20–40. Hence to conclude, penalizing the deviations more results in lower values of error.
Figure 4-28: The value of Thrust ($u_1$) and RMSE in z plotted against $Q(5,5)$. The red line indicates the thrust values and the blue line shows the variation of RMSE in z direction.

Figure 4-28 depicts the combined effect of control input and find a perfect tradeoff between the root mean square error of position with respect to the parameter $Q(5,5)$ which penalizes deviations in z-direction, the above plot can be observed. It can be noted that, to attain the average value of thrust which is around 1.3 and the value of root mean squared error (RMSE) in z-direction to be less than 0.2 the value of $Q(5,5)$ should be in the range of 150-500. The above plot can serve as an integral tool for MPC control system designers that can allow them to choose the value of penalizing constant $Q(5,5)$ depending on what range of RMSE is acceptable.

From the series of experiments that were performed, we can determine a range of values for the constants of penalizing matrices which give an accurate solution to the trajectory tracking problem with feasible responses from the actuators. This range of values are summarized Table 6.
Table 6 : Predicted range of Value for best performance of the Quadcopters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$ matrix diagonal elements penalizing the deviations</td>
<td></td>
</tr>
<tr>
<td>$Q(1,1)$</td>
<td>150 – 500</td>
</tr>
<tr>
<td>$Q(3,3)$</td>
<td>250 – 500</td>
</tr>
<tr>
<td>$Q(5,5)$</td>
<td>250 – 500</td>
</tr>
<tr>
<td>$P$ matrix diagonal elements penalizing the inputs</td>
<td></td>
</tr>
<tr>
<td>$P(1,1)$</td>
<td>0.3 - 0.4</td>
</tr>
<tr>
<td>$P(2,2)$</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(3,3)$</td>
<td>0.001</td>
</tr>
<tr>
<td>$P(4,4)$</td>
<td>1 - 5</td>
</tr>
</tbody>
</table>

These experiments show the ability of Model Predictive Control to generate various complex trajectories. These experiments have assumed that the environment is obstacle-free. But, that is hardly the case in real life. To explore and ensure that MPC is capable of avoiding obstacles in an environment, a simulation was performed using a Matlab based tool - MOANTOOL. The next section shows the experiment that was performed.

4.3 Obstacle Avoidance

In the previous section, we have seen that the MPC controller is capable of generating complex trajectories. Another important application of Model Predictive Control is obstacle avoidance. As mentioned in section 3.3 that the difference between path planning and obstacle avoidance is the application of constraints. The greatest advantage of using Model Predictive Control is that it can handle a large number of constraints. For the purpose of this dissertation, linear inequalities were used as constraints. These linear inequalities are defined in MOANTOOL as constraints on the input variables as well as the state variables. In order to validate the obstacle avoidance capability of MPC the following experiment was performed in MOANTOOL.
The experiment started by setting up each class as mentioned in the section 3.6. For the agent class, the 2D quadcopter model was defined using the state space matrices $A, B, C, D$ which were derived earlier as per figure 2.1. Along with the matrices, the dimensions of input, output and prediction horizon were specified as given in section 3.6. The physical dimensions of the quadcopter were given in terms of height and width which was 1 and 1 respectively. The penalizing matrices $Q$ and $P$ as discussed in section 4.4 were also specified here along with the sampling time which was 0.1 seconds. The constraints on the input, output and the states were set in the agent class. The reference trajectory was given as a circular path with radius 10 with a center at origin. The quadcopter's initial position was the origin $x = 0$ and $y = 0$. After defining the parameters of the agent class, the obstacle parameters were given. The obstacles were chosen to be rectangular with height and width as 3 and 2 respectively. For this experiment, four obstacles were used. They were positioned to pass through the reference trajectory provided. The position of the obstacles is given in Appendix B. The planner was then instantiated. The quadcopter, obstacles and the optimization solver are in this class. The 'quadprog' optimization solver was used. The simulator class was then used for visualizing the results. The simulator class generated the results below.
Figure 4-29: Obstacle avoidance. The figure shows a quadcopter avoiding four obstacles represented by the red blocks. The green dotted line shows the reference circular trajectory, and the pink line is the actual trajectory of the quadcopter.

Figure 4-29 depicts the quadcopter is following the trajectory avoiding the obstacles. Although the quadcopter successfully avoided all the four obstacles, the resulted trajectory could be followed more precisely. The ideal case would have been for the quadcopter to follow a circular trajectory with radius 10 and while avoiding the obstacles. This result shows that the quadcopter attempted to follow a circular trajectory with radius 10 but could only reach an almost circular trajectory of approximately radius 5. The time taken by the laptop for generating this result was approximately 40 seconds. The overview of the commands that were used to do the problem setup
using each class of MOANTOOL is given in Appendix C. The result depicted in figure 4-29 is the best performance that could be obtained using MOANTOOL.

4.4 Summary

This chapter has presented the simulation results for several experiments. A number of experiments were performed using MPC for complex trajectory tracking which can be seen in the Section 4.2. Model Predictive Control works extremely well for aerial choreography using a swarm of UAVs. The range for the values of penalizing constants was derived based on the various experiments. The ability to avoid obstacles is an important characteristic of Model Predictive control which was shown in Section 4.3, although the performance can be improved by using better optimizer. The next chapter describes the conclusions, assumptions, and future work that lies ahead.
Chapter 5

Conclusion

5.1 Summary of Contributions

Quadcopter applications require more than simply going from a point to another in space. This dissertation attempted to present Model Predictive Control as a solution to the problem of complex path planning and obstacle avoidance. In order to do that, a controller employing Model Predictive Control was developed and deployed for simulated quadcopters. This controller could be the basis for allowing teams of vehicles to perform coordinated aerial maneuvers. Once the MPC framework was derived, various experiments were performed using different test cases. The test cases that were used are summarized in Table 7. The MPC algorithm has been extensively tested in Matlab using the test case scenarios as given below. The simulation results show that in order to track the desired trajectory as closely as possible, the system might require a very high control input for instance thrust. The required thrust might not be of feasible magnitude to be achieved by the UAV. In other words, to achieve accurate trajectory tracking, the simulation results showed the required thrust to be of infeasible magnitude that cannot be achieved by hardware in real world. To avoid this, one must find a tradeoff between acceptable error while tracking the trajectory and the control input needed for the task. After testing MPC in different scenarios, a range of penalizing constants was derived which assured a perfect tradeoff between accurate trajectory tracking in other words least error with the amount of control effort required. Although an extensive array of simulated experiments have been successfully conducted, there is still scope for future efforts. Another important factor, that can be looked into is the verification of the desired trajectory. A given trajectory could be considered unachievable if after performing the simulations, the error in
the position is not acceptable for a perfectly achievable magnitude of control input. This also highlights the importance of simulation while designing a control system of aerial vehicles.

Table 7: Test cases to evaluate the performance of MPC controller

<table>
<thead>
<tr>
<th>CASE 1</th>
<th>One quadcopter following an irregular trajectory. The trajectory is defined as a function of the cosine and sine with respect to time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE 2</td>
<td>Two quadcopters following an irregular trajectory. The reference trajectory is provided as a function of the cosine, sine and tangent with respect to time.</td>
</tr>
<tr>
<td>CASE 3</td>
<td>Ten quadcopters following an irregular trajectory. The reference trajectory is provided as a function of the cosine, sine and tangent with respect to time.</td>
</tr>
<tr>
<td>CASE 4</td>
<td>Three quadcopters spelling a word in 3D space. The reference trajectory is provided as waypoints.</td>
</tr>
<tr>
<td>CASE 5</td>
<td>Twenty three quadcopters forming a heart shape in 3D space. The reference trajectory is provided as a function of the waypoints and the entire swarm is moved along a particular axis.</td>
</tr>
</tbody>
</table>

We demonstrated that quadcopters could avoid obstacles using the MOANTOOL program which dynamically applies constraints to the control problem. Unfortunately, the performance is not adequate, because while the controller successfully avoids obstacles, it is evident that the trajectory could be more closely followed. Hence, there is a need for better optimizers which would capture and follow the constraints on the system more accurately. The MPC algorithm demonstrated in the thesis can be implemented on embedded hardware, making real-time experimentation possible. The above algorithms can also be applied to UAVs in a more complex 3D environment with dynamic obstacles.
5.2 Limitations and Assumptions

Although Model Predictive Control handles constraints well, it is sensitive to noise. The success of a closed loop controller based on MPC is heavily reliant on the accuracy of the controller predictions. In the presence of noise, the predictions can differ from the actual behavior of the system. Hence, efforts are being made to modify Robust Model Predictive Control and to use Kalman filters along with MPC to improved performance. The linear MPC method that has been presented in this dissertation does not account for any disturbances in terms of noise. Given the number of application of aerial vehicles like quadcopters it is important that the quadcopter perform the assigned task irrespective of the external circumstances and internal disturbances such as noise. Hence, research is focused on disturbance modeling. Robust Model predictive Control (RMPC) is a control technique wherein an unmeasured disturbance acting on the system is taken into consideration while implementing it on a vehicle. For the experiments performed here, the state space model that was used did not include disturbances.

The main motivation of this dissertation was to implement Model Predictive Control in order to test and validate autonomous path planning for groups of UAVs. With this objective in mind, there are some assumptions that were made:

1. A linear model of the UAV was used in order to implement Linear Model Predictive Control (LMPC);
2. For the reference trajectory tracking problem as described in section 2.1, there were no disturbances involved. This implies that the controller did not have to deal with any kind of noise within the system.
3. For the obstacle avoidance case as described in section 2.1, the obstacles were static and immobile.
In real life scenarios, the quadcopter has to encounter static as well as moving obstacles. One of the major drawbacks of MOANTOOL is that it cannot handle the problem if the obstacles are moving.

5.3 Future Scope

This dissertation serves as a preliminary work towards testing the accuracy of a Model Predictive Controller used for vehicle autonomy. There are multiple ways that the present research can be expanded. The simulation results presented in the previous chapter show that the MPC can accurately track a reference trajectory. The MPC controller is validated on the hardware. It would be interesting to see how well functions in real world on aerial vehicles where the computation is performed on-line. Another crucial issue that was faced during performing the simulations was the high magnitude of control input required for tracking a given reference trajectory. In order to avoid the issue of obtaining extremely high control inputs, this architecture can further be improved by using constraints on the control input and then MPC would be a constrained optimization problem.

The obstacle avoidance using MPC can be performed in a complex 3D environment. More analysis should be performed using Nonlinear Model Predictive Control to evaluate the performance on the quadcopter. A comparative study could be performed between similar optimization based control techniques such as LQR control. MPC has come a long way starting from chemical industry. The literature on MPC suggests and shows that it has now been applied to autonomous functioning in the automotive industry. The approach also is applicable to the marine industry where autonomous underwater vehicles are used for collecting maritime data. Overall, we believe that Model Predictive Control could be employed to manage a swarm of UAVs tasked with a variety of challenging path planning and obstacle avoidance tasks. Thus, catering to the growing applications of quadcopters. The applications including Surveillance missions, rescue operations
and aerial performances can benefit by using MPC. The fact that MPC will not override the constraints makes it an excellent choice for onboard calculation in presence of uncertainty.
REFERENCES


J. Maciejowski (2000), Predictive Control with Constraints, Prentice Hall.


A. Pavlov, H. Nordahl and M. Breivik (2009), "MPC-based optimal path following for underactuated vessels," Proceedings of the 8th IFAC International Conference on Manoeuvring and Control of Marine Craft Guarujá (SP), pp. 16-18, Brazil.


Appendix A

A.1 Parameters of MPC Controller

Parameters for the MPC controller for the experiment with a quadcopter following an infinity loop trajectory as shown in figure 5.9

<table>
<thead>
<tr>
<th>Sampling time</th>
<th>$0.001$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired $x$</td>
<td>$\sin (time)$</td>
</tr>
<tr>
<td>Desired $y$</td>
<td>$\sin (time)$</td>
</tr>
<tr>
<td>Desired $z$</td>
<td>$\sin (2 \ast time)$</td>
</tr>
<tr>
<td>Prediction horizon $N$</td>
<td>$100$</td>
</tr>
</tbody>
</table>

Parameters for the MPC controller for the experiment with two quadcopters following a taco shell shaped trajectory as shown in figure 4.1

<table>
<thead>
<tr>
<th>Sampling time</th>
<th>$0.001$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired $x_1$</td>
<td>$\sin (time)$</td>
</tr>
<tr>
<td>Desired $y_1$</td>
<td>$\cos (time)$</td>
</tr>
<tr>
<td>Desired $z_1$</td>
<td>$\sin (2 \ast time)$</td>
</tr>
<tr>
<td>Desired $x_2$</td>
<td>$0.75 \ast \sin (time)$</td>
</tr>
<tr>
<td>Desired $y_2$</td>
<td>$0.75 \ast \cos (time)$</td>
</tr>
<tr>
<td>Desired $z_2$</td>
<td>$1 + \sin (2 \ast time)$</td>
</tr>
</tbody>
</table>
Parameters for the MPC controller for the experiment with 10 quadcopters following a taco shell shaped trajectory as shown in figure 4.5

| Desired $x_i$ | Desired $y_i$ | Desired $z_i$ | Desired $x_{i+1}$ | Desired $y_{i+1}$ | Desired $z_{i+1}$ | Desired $x_{i+2}$ | Desired $y_{i+2}$ | Desired $z_{i+2}$ | Desired $x_{i+3}$ | Desired $y_{i+3}$ | Desired $z_{i+3}$ | Desired $x_{i+4}$ | Desired $y_{i+4}$ | Desired $z_{i+4}$ | Desired $x_{i+5}$ | Desired $y_{i+5}$ | Desired $z_{i+5}$ | Desired $x_{i+6}$ | Desired $y_{i+6}$ | Desired $z_{i+6}$ | Desired $x_{i+7}$ | Desired $y_{i+7}$ | Desired $z_{i+7}$ | Desired $x_{i+8}$ | Desired $y_{i+8}$ | Desired $z_{i+8}$ | Desired $x_{i+9}$ | Desired $y_{i+9}$ | Desired $z_{i+9}$ | Desired $x_{i+10}$ | Desired $y_{i+10}$ | Desired $z_{i+10}$ |
|---------------|---------------|---------------|------------------|------------------|------------------|-------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 2 + sin $(2 * time)$ |
| 4 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 8 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 10 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 12 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 14 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * cos $(time)$ |
| 16 + sin $(2 * time)$ |
| 0.75 * sin $(time)$ |
| 0.75 * sin $(time)$ |
| \(Desired\ y_{10}\) | \(0.75 \times \cos (\text{time})\) |
| \(Desired\ z_{10}\) | \(18 + \sin (2 \times \text{time})\) |

Parameters for the MPC controller for the experiment with a quadcopter following an infinity loop trajectory as shown in figure 4.9

<table>
<thead>
<tr>
<th>Sampling time</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired (x_1)</td>
<td>[1, 2, 3, 2.5, 1.5]</td>
</tr>
<tr>
<td>Desired (y_1)</td>
<td>[1, 1, 1, 1, 1]</td>
</tr>
<tr>
<td>Desired (z_1)</td>
<td>[1, 5, 1, 3, 3]</td>
</tr>
<tr>
<td>Desired (x_2)</td>
<td>[4, 4, 4, 4, 4]</td>
</tr>
<tr>
<td>Desired (y_2)</td>
<td>[1, 1, 1, 1, 1]</td>
</tr>
<tr>
<td>Desired (z_2)</td>
<td>[5, 4, 3, 2, 1]</td>
</tr>
<tr>
<td>Desired (x_3)</td>
<td>[6, 6, 7, 8, 8]</td>
</tr>
<tr>
<td>Desired (y_3)</td>
<td>[1, 1, 1, 1, 1]</td>
</tr>
<tr>
<td>Desired (z_3)</td>
<td>[1, 5, 3, 5, 1]</td>
</tr>
</tbody>
</table>

Parameters for the MPC controller for the experiment with 23 quadcopters forming a heart shape in the 3D and moving along \(x\)-axis trajectory as shown in figure 4.19

<table>
<thead>
<tr>
<th>Sampling time</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired (x_1)</td>
<td>[0, 6.3, 6.3, 6.3]</td>
</tr>
<tr>
<td>Desired (y_1)</td>
<td>[0, 2.1, 2.1, 2.1]</td>
</tr>
<tr>
<td>Desired (z_1)</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired (x_2)</td>
<td>[0, 5.8, 5.8, 5.8]</td>
</tr>
<tr>
<td>Desired (y_2)</td>
<td>[0, 3, 3, 3]</td>
</tr>
<tr>
<td>Desired (z_2)</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired (x_3)</td>
<td>[0, 5.3, 5.3, 5.3]</td>
</tr>
<tr>
<td>Desired (y_3)</td>
<td>[0, 3.6, 3.6, 3.6]</td>
</tr>
<tr>
<td>Desired (z_3)</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired (x_4)</td>
<td>[0, 4.9, 4.9, 4.9]</td>
</tr>
<tr>
<td>Desired (y_4)</td>
<td>[0, 4.2, 4.2, 4.2]</td>
</tr>
<tr>
<td>Desired $z_4$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_5$</td>
<td>[0.4, 4, 4, 4, 4, 4]</td>
</tr>
<tr>
<td>Desired $y_5$</td>
<td>[0.5, 5, 5, 5]</td>
</tr>
<tr>
<td>Desired $z_5$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_6$</td>
<td>[0.3, 6, 3, 6, 3, 6]</td>
</tr>
<tr>
<td>Desired $y_6$</td>
<td>[0.6, 4, 6, 4, 6, 4]</td>
</tr>
<tr>
<td>Desired $z_6$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_7$</td>
<td>[0, 3, 3, 3]</td>
</tr>
<tr>
<td>Desired $y_7$</td>
<td>[0, 8, 8, 8]</td>
</tr>
<tr>
<td>Desired $z_7$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_8$</td>
<td>[0.2, 8, 2, 8, 2, 8]</td>
</tr>
<tr>
<td>Desired $y_8$</td>
<td>[0, 9, 9, 9]</td>
</tr>
<tr>
<td>Desired $z_8$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_9$</td>
<td>[0.2, 9, 2, 9, 2, 9]</td>
</tr>
<tr>
<td>Desired $y_9$</td>
<td>[0, 10, 10, 10]</td>
</tr>
<tr>
<td>Desired $z_9$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_{10}$</td>
<td>[0.3, 6, 3, 6, 3, 6]</td>
</tr>
<tr>
<td>Desired $y_{10}$</td>
<td>[0, 10.6, 10.6, 10.6]</td>
</tr>
<tr>
<td>Desired $z_{10}$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_{11}$</td>
<td>[0.4, 5, 4, 5, 4, 5]</td>
</tr>
<tr>
<td>Desired $y_{11}$</td>
<td>[0, 10.7, 10.7, 10.7]</td>
</tr>
<tr>
<td>Desired $z_{11}$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired $x_{12}$</td>
<td>[0.5, 6, 5, 6, 5, 6]</td>
</tr>
<tr>
<td>Desired $y_{12}$</td>
<td>[0, 10, 10, 10]</td>
</tr>
<tr>
<td>Desired $z_{12}$</td>
<td>[0, 1, 3, 1]</td>
</tr>
<tr>
<td>Desired (x_{13})</td>
<td>([0,6.3,6.3,6.3])</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Desired (y_{13})</td>
<td>([0,8.5,8.5,8.5])</td>
</tr>
<tr>
<td>Desired (z_{13})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{14})</td>
<td>([0,7,7,7])</td>
</tr>
<tr>
<td>Desired (y_{14})</td>
<td>([0,10,10,10])</td>
</tr>
<tr>
<td>Desired (z_{14})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{15})</td>
<td>([0,8,8,8])</td>
</tr>
<tr>
<td>Desired (y_{15})</td>
<td>([0,10.6,10.6,10.6])</td>
</tr>
<tr>
<td>Desired (z_{15})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{16})</td>
<td>([0,8.5,8.5,8.5])</td>
</tr>
<tr>
<td>Desired (y_{16})</td>
<td>([0,10.7,10.7,10.7])</td>
</tr>
<tr>
<td>Desired (z_{16})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{17})</td>
<td>([0,9.3,9.3,9.3])</td>
</tr>
<tr>
<td>Desired (y_{17})</td>
<td>([0,10.6,10.6,10.6])</td>
</tr>
<tr>
<td>Desired (z_{17})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{18})</td>
<td>([0,9.9,9.9,9.9])</td>
</tr>
<tr>
<td>Desired (y_{18})</td>
<td>([0,9.3,9.3,9.3])</td>
</tr>
<tr>
<td>Desired (z_{18})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{19})</td>
<td>([0,9.7,9.7,9.7])</td>
</tr>
<tr>
<td>Desired (y_{19})</td>
<td>([0,8,8,8])</td>
</tr>
<tr>
<td>Desired (z_{19})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{20})</td>
<td>([0,9.3,9.3,9.3])</td>
</tr>
<tr>
<td>Desired (y_{20})</td>
<td>([0,7.1,7.1,7.1])</td>
</tr>
<tr>
<td>Desired (z_{20})</td>
<td>([0,1,3,1])</td>
</tr>
<tr>
<td>Desired (x_{21})</td>
<td>([0,8.9,8.9,8.9])</td>
</tr>
<tr>
<td>Desired $y_{21}$</td>
<td>[0,6.4,6.4,6.4]</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Desired $z_{21}$</td>
<td>[0,1,3,1]</td>
</tr>
<tr>
<td>Desired $x_{22}$</td>
<td>[0,8,8,8]</td>
</tr>
<tr>
<td>Desired $y_{22}$</td>
<td>[0,5,5,5]</td>
</tr>
<tr>
<td>Desired $z_{22}$</td>
<td>[0,1,3,1]</td>
</tr>
<tr>
<td>Desired $x_{23}$</td>
<td>[0,6.7,6.7,6.7]</td>
</tr>
<tr>
<td>Desired $y_{23}$</td>
<td>[0,3,3,3]</td>
</tr>
<tr>
<td>Desired $z_{23}$</td>
<td>[0,1,3,1]</td>
</tr>
</tbody>
</table>
Appendix B

B.1 Velocity Plots

Figure B-1: Linear Velocity of the quadcopter with respect to time for case 1 of experiment 1.

The figure above shows the absolute linear velocity with respect to time. The linear velocity in the x and y directions are the same in magnitude and hence are overlapping. The velocity in the z direction fluctuates between 0 m/s to 2 m/s. The value in the x and y directions fluctuates between 0 and 7 m/s.
Figure B-2: Velocity of the quadcopter in x, y and z with respect to time for case 2 of experiment 1.

The absolute velocity can be seen in figure 5-19, the velocity does not seem to be stabilizing over the course of the simulation. The quadcopter is climbing within the range of velocity of 0 - 3 m/s; whereas the horizontal (lateral) velocity approaches 7m/s.

Figure B-3: Velocity of the quadcopter in the x, y and z direction with respect to time for test case 3 of experiment 1.
The velocity of the quadcopter approaches 10 m/s. As the value of penalizing constants decreases, the velocity increases.

Figure B-4: Variation in velocity with respect to time case 1 of experiment 2.

Figure B-5: Variation of velocity in the x, y and z direction case 1 of experiment 3.
The figure above shows how the linear velocity in the x, y and z direction changes as the simulation proceeds. The climbing velocity stays between $0 - 2\ m/s$ whereas the velocity in the x and y direction peaks between $0 - 7\ m/s$.

![Graph](image)

Figure B-6: Variation of velocity of the quadcopter following the L trajectory with respect to time case 1 of experiment 4.

As discussed earlier there is no motion in y-axis, hence the velocity along y-axis is zero which can be seen from above figure.
Figure B-7: Velocity of the quadcopter following the L trajectory with respect to time case 2 of experiment 4.

Figure B-8: Velocity of the quadcopter following the trajectory 'L' shown here with respect to time case 3 of experiment 4.
Figure B-9: Velocity of the quadcopter no. 5 plotted with respect to time case 1 of experiment 5.

The figure 5-73 shows the value of the velocity variation of the quadcopter 5. As seen earlier the thrust stabilizes and eventually approaches zero.

Figure B-10: Velocity of the quadcopter no. 12 plotted with respect to time case 1 of experiment 5.
Figure B-11: Velocity of the quadcopter no. 20 plotted with respect to time case 1 of experiment 5.

Figure B-12: Velocity of quadcopter no. 5 plotted against time case 2 of experiment 5.
Figure B-13: Velocity of quadcopter no. 12 plotted against time case 2 of experiment 5.

Figure B-14: Velocity of quadcopter no. 20 plotted against time case 2 of experiment 5.
Figure B-15: Velocity of the quadcopter no. 5 plotted against time case 3 of experiment 5.

Figure B-16: Velocity of the quadcopter no. 12 plotted against time case 3 of experiment 5.
Figure B-17: Velocity of the quadcopter no. 20 plotted against time case 3 of experiment 5.
Appendix C

C.1 MOANTOOL Commands

The MOANTOOL uses a specific state for commands for setting up the MPC obstacle avoidance are shown here. The linearized dynamics can be defined by:

1. Agent Class:

   \[
   \begin{align*}
   &\text{agent} = \text{LinearAgent} \left( \text{'nx', \textit{nx}, 'nu', \textit{nu}, ... 'ny', \textit{ny}, 'PredictionHorizon', \textit{N}} \right); \\
   &\text{agent} = \text{LinearizedAgent} \left( \text{'nx', \textit{nx}, 'nu', \textit{nu}, ... 'ny', \textit{ny}, 'PredictionHorizon', \textit{N}} \right);
   \end{align*}
   \]

   The defining matrices A, B, C, D can be specified as:

   \[
   \begin{align*}
   &\text{agent.} \text{A.\ Value} = \text{A} ; \\
   &\text{agent.} \text{B.\ Value} = \text{B} ; \\
   &\text{agent.} \text{C.\ Value} = \text{C} ; \\
   &\text{agent.} \text{D.\ Value} = \text{D} ;
   \end{align*}
   \]

   Constrains:

   \[
   \begin{align*}
   &\text{agent.} \text{X.\ Min} = \text{x_min} ; \\
   &\text{agent.} \text{X.\ Max} = \text{x_max} ; \\
   &\text{agent.} \text{U.\ Min} = \text{u_min} ; \\
   &\text{agent.} \text{U.\ Max} = \text{u_max} ;
   \end{align*}
   \]

   Dimension:

   \[
   \begin{align*}
   &\text{agent.} \text{Size.\ Value} = [ \text{width} ; \text{height} ];
   \end{align*}
   \]

   Penalizing matrices \( Qu \) and \( Qy \):

   \[
   \begin{align*}
   &\text{agent.} \text{Y.\ Penalty} = Q_y ;
   \end{align*}
   \]
agent.\ U.\ Penalty = Q_u;

2. Obstacle class:

\texttt{obstacles = Obstacle (agent, n\_obs);}

The position of the obstacle can also be specified:

\texttt{obstacles(i).Size.Value = [width_i; height_i];}

3. Planner Class:

The planner can be initialized in the following way:

\texttt{planner = Planner (agent, obstacles,... 'solver', 'gurobi', 'MixedInteger', flag );}

The reference trajectory can also be defined in this class.

\texttt{planner.Parameters.Agent.Y.Reference = yref;}

4. Simulator Class:

The simulation class is initialized by providing the planner as input.

\texttt{psim = Simulator (planner);}

\texttt{psim.run (x0, Nsim );}

Where Nsim is the number of simulation steps to be run.
<table>
<thead>
<tr>
<th>Parameters used in MOANTOOL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling step</td>
<td>0.1</td>
</tr>
<tr>
<td>Prediction Horizon, $N$</td>
<td>30</td>
</tr>
<tr>
<td>Dimensions of inputs $nu$</td>
<td>2</td>
</tr>
<tr>
<td>Dimensions of output $ny$</td>
<td>6</td>
</tr>
<tr>
<td>Dimension of states $nx$</td>
<td>6</td>
</tr>
<tr>
<td>Simulation steps $Ns_{sim}$</td>
<td>150</td>
</tr>
<tr>
<td>Position of Obstacle 1</td>
<td>[10, 0]</td>
</tr>
<tr>
<td>Position of Obstacle 2</td>
<td>[-10, 0]</td>
</tr>
<tr>
<td>Position of Obstacle 3</td>
<td>[0, 10]</td>
</tr>
<tr>
<td>Position of Obstacle 4</td>
<td>[0, -10]</td>
</tr>
<tr>
<td>Quadcopter Size</td>
<td>$1 \times 1$</td>
</tr>
<tr>
<td>Obstacle Size</td>
<td>$3 \times 2$</td>
</tr>
<tr>
<td>$Q(1,1)$</td>
<td>10000</td>
</tr>
<tr>
<td>$Q(2,2)$</td>
<td>10000</td>
</tr>
<tr>
<td>$P(1,1)$</td>
<td>800</td>
</tr>
<tr>
<td>$P(1,1)$</td>
<td>101</td>
</tr>
</tbody>
</table>