ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
ELECTRICAL AND COMPUTER ENGINEERING
DEPARTMENT

Model Predictive Control of Unmanned Aerial Vehicle for Locust Detection and Bio-pesticide Spraying

A thesis submitted to School of Graduate Studies, Addis Ababa Institute of Technology, Addis Ababa University in partial fulfillment of the requirement for the Degree of Master of Science in Electrical Engineering (Control Engineering)

By
Eden Getiye

Advisor
Dr. Lebsework Negash and Dr. Dereje Shiferaw

November 18, 2021
Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Model Predictive Control of Unmanned Aerial Vehicle for Locust Detection and Bio-pesticide Spraying

By: Eden Getiye

APPROVED BY BOARD OF EXAMINERS

Name	Signature	Date
_____	_____	_____
(Dean, School of Graduate Committee)		
_____	_____	_____
(Advisor)		
_____	_____	_____
(Internal Examiner)		
_____	_____	_____
(External Examiner)		

Declaration

I, the undersigned, declared that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other University and all sources and materials used for the thesis have been duly acknowledged..

Name	**Signature**
_____	_____

Place:
Addis Ababa Institute of Technology, AAiT
Addis Ababa University, AAU
Addis Ababa,
Ethiopia.

Submitted in: November 18, 2021

This thesis has been submitted for examination with my approval as a university advisor.

Advisors	**Signature**
_____	_____
_____	_____

Acknowledgment

I am very grateful to have the chance of thanking my almighty God who let me see the final fruit of my work. The assistance provided by my advisers Dr.Lebsework and Dr,Derege was greatly appreciated and deserves a big gratitude. I am very thank full for their initiative and motive to convincingly comment, suggest and guide me right from the beginning.

I would also like to express my special regards for my family who supported me throughout my life.

Abstract

Swarm of Locust are very harmful for food security, quality and quantity of agriculture products. Ethiopia is one of the countries which is extensively affected by locust invasion. The locust swarms have destroyed large swaths of food and pasture in Ethiopia which lead to famine and displacing thousands of people from their home. Ethiopia battled the swarms by spraying pesticides from air using helicopters leased from FAO. With this consideration, precise locust detection and bio-pesticide spraying is significant for preventing locust plagues.

This thesis is going to focus on the design of Model Predictive Control of UAV for locust detection and bio-pesticide Spraying. To accomplish this design: First the dynamics of the system was understood then the mathematical model of the system was done and it was based on an agriculture spray drone (JMR-X1400). The Newton-Euler formalism was used to model the dynamic system and verified in Simulink. The flight controller is designed and MPC is implemented for this thesis. For this non-linear dynamic system of a quad-copter NMPC (non-linear MPC) is chosen. Multiple shooting method is selected to transform the optimal control problem to nonlinear program (NLP). To solve the NLP, CasADi in MATLAB is used and the solver is Ipopt (Interior Point Optimizer). The NMPC was able to control the quad-copter, which means the quad-copter was able to follow the given reference trajectory with minimum control effort. Since the quad-copter is used to spray pesticide, there will be a change in mass when it sprays. For this reason the Recursive Least Square Estimation (RLSE) is used to estimate the mass change and the model can be updated using the estimation. The proposed method works adequately. The RLSE was able to estimate the mass change and the quad-copter was still able to track the reference.

Manual monitoring is a labor-intensive job and expensive for large farms. To tackle this problem, image recognition have provided a promising solution for detecting pests. So for this thesis Image recognition system is developed to detect and recognize the Locust swarm. Since it is an Image classification, CNN is chosen and the programming language is in python. After passing through different procedures the final training accuracy of the machine is 95.19%.

Keywords:Locust Swarm, UAV, MPC, Multiple Shooting, Mass Variation, RLSE, Image Recognition, CNN

Contents

List of Figures

List of Tables

List of Acronyms

- UAV : Unmanned Aerial Vehicle
- FAO : Food and Agriculture Organization
- MIMO : Multiple Input Multiple Output
- SISO : Single Input Single Output
- MPC : Model Predictive Controller
- NMPC : Non-linear Model Predictive Controller
- SQP : Sequential Quadratic Programming
- RLSE : Recursive Least Square Estimation
- Ipopt : Interior Point Optimizer
- OCP : Optimal Control Problem
- NLP : Non-linear Programming Problem
- CNN : Convolutional Neural Network
- CCW : Counter Clock wise
- CW : Clock wise
- DoF : Degree of Freedom
- LQR : Linear Quadratic Regulator
- PID : Proportional, Integrator and Derivative
- PSO : Particle Swarm Optimization

Chapter 1

Introduction

This chapter provides a general overview on the Background, Statement of the Problem, Objective, Methodology, Scope and Thesis Outline.

1.1 Background

Locust swarms are typically in motion and can cover vast distance. They may travel more than 130 kilometer or more a day. One of the most concerning effect of locust swarm is their consumption of green vegetation. In particular, crops within agricultural regions and in rural areas. In a single day, a swarm of locusts that covers one square kilometer can consume more food than 35,000 people would in the same time frame[1]. In a region already affected by food insecurity, the locust outbreak only worsens the problem and could potentially lead to five million people in Africa facing starvation.

Finding versatile, easy-to-use new tools to tackle these destructive locust swarms becomes more crucial because the local farmers and response teams have been struggling to contain them. In times of this crisis, UAV's provide an innovative complementary solution to the more expensive manned aircraft or the less effective manual spraying method. They can be used to conduct chemical spraying to kill the locusts, especially in the impacted areas and places where it is inaccessible for ground vehicles and airplanes.

Commonly used UAV's are classified as: Fixed-wing UAVs and Multi-copter UAV's. Fixed-wing UAVs use a wing like a normal aeroplane to provide the lift rather than vertical lift rotors. Because of this, they only need to use energy to move forward, not hover in the air, so are much more efficient. The main downside of a Fixed-wing aircraft is their inability to hold themselves

in one spot, lack maneuverability which is vital for UAVs and are very expensive. On the other hand, Multi-copter UAVs which can be considered as a type of helicopter, which has three or more propellers. It also has the ability of manuverability and hovering over targets. The most popular Multi-copter is the Quad-copter. A Quad-copter has four rotors powered by independent motors all mounted on a fixed frame.

The UAVs choice for this thesis is a Quad-copter. It is a UAV with four rotors which can differ in speed from each other to achieve a movement in any direction and a lift. A Quad-copter has 6 degree of freedom (DOF), it can translate in all three directions $[X, Y, Z]$ and it can also rotate $[Roll(\phi), Pitch(\theta), Yaw(\psi)]$ around the three axes.

The goal of this thesis is to design a controller for UAV which is capable of detecting a locust swarm and can also spray a bio-pesticide. Using UAVs we can detect and eliminate locust swarms in early stages so we can decrease the destruction. Model Predictive Control (MPC) is used as a controller for the UAV. MPC is a feedback control algorithm that uses a plant model to make predictions about the future outputs of a process. The main purpose of MPC is to find an optimal vector control function that minimize or maximize a performance index subject to a given process model (usually a nonlinear differential equation system) as equality constraints, and boundary conditions as inequality constraints on the states and controls. It can handle multiple input and multiple output (MIMO) systems and input-output constraints. For this thesis NMPC (non-linear Model Predictive Control) is used. NMPC is the most powerful one as it uses the most realistic and highly nonlinear representation of the plant, namely a nonlinear plant model. Therefore, the predictions made by NMPC are more accurate which also lead to better control actions. However it is also the most challenging one to solve in real-time, Since having a nonlinear system and a nonlinear constraints makes the optimization problem non-convex. The cost function may have multiple local optima and finding the global optimum may be hard. The efficiency of solving the non-convex optimization problem that requires a large number of computations depends on the nonlinear solver used. The main tools used are:

- For modeling the actual quad-copter (with disturbance and mass variation consideration) Simulink is used
- For the non linear optimization (Non-linear prediction control) CasADi in MATLAB is used
- For image processing and image recognition python programming in Visual Studio is used.

1.2 Statement of the Problem

According to the United Nations Food and Agriculture Organization (FAO), the locust invasion is Ethiopia's worst in 25 years. An estimation of 200,000 hectares of land have been damaged since January 2020, threatening food supplies. These swarms put high pressure on the access to food in different regions of Ethiopia. In April the FAO calculated that a million of people have been affected and require emergency food assistance.

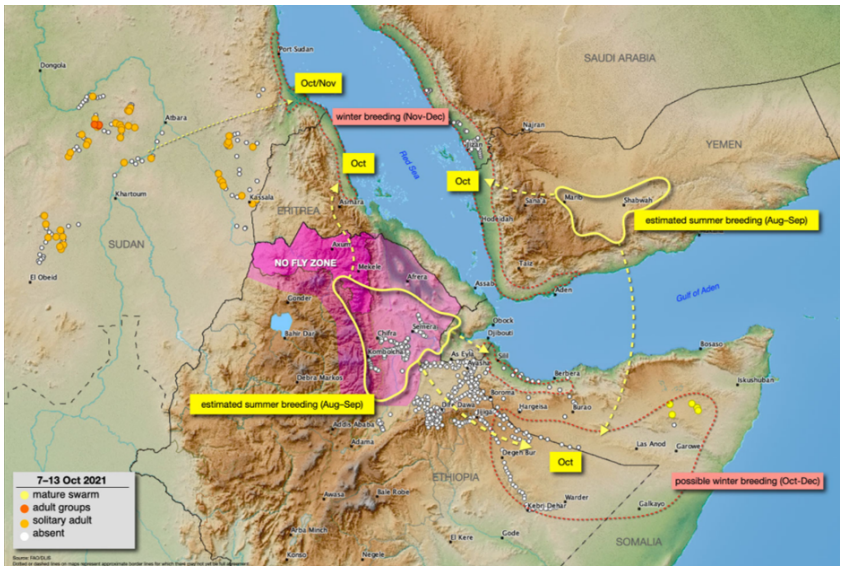


Figure 1.1: Latest situation and forecast of Ethiopia's Locust Swarm Invasion

In some regions of Ethiopia the community tried to scare them off by banging on cans and pans, blowing whistles and honking motorcycle horns while experts sprayed pesticides from vehicles, but the total ground effort wasn't enough to stop the pests. A company called N jet dynamics working in drone manufacturing and assembling for different purpose like agriculture, transport, inspection, mapping etc. And now the company uses the drones to spray pesticides on locust swarm. The locusts have already destroyed large swaths of food and pasture in different regions.

In the last decades the use of UAVs has become more and more popular. The autonomous capabilities of UAVs have improved over the last years due to improvements in control technology and improvements of computational power.

In order to overcome the difficulties stated above, Unmanned Aerial Vehicle (UAV) controller will be designed, which will be used to spray bio-pesticides. It will have image recognition capability to detect the locust swarm before spraying for efficient and enviroment concisions use of pesticide.

Figure 1.2: The Community trying to scare off the Locust swarm

1.3 Objective

1.3.1 General Objective

To design a Model predictive control of Unmanned Aerial Vehicle for locust detection and bio-pesticide spraying.

1.3.2 Specific Objective

- To define the nonlinear dynamics of a quad-copter using a comprehensible modeling technique
- Design a flight controller using Model Predictive Control for point stabilization and trajectory tracking using CasADi in MATLAB
- Parameter estimation for mass using Recursive Least Square Estimator(RLSE)
- Build image recognition to detect the locust
- Simulate the whole system considering image recognition and mass variation

1.4 Methodology

Methodology	**Task in Detail**
Literature review	Reading published research papers, books, articles related to this topic
Modeling of a Quad-copter	Formulating the dynamics of the Quad-copter
Model verification	Verify the modeled dynamics using Simulink
MPC design for point stabilization	Based on the obtained model, design MPC for a constant reference frame using CasADi in MATLAB
MPC design for trajectory tracking	Based on the obtained model, design MPC for time varying reference frame using CasADi in MATLAB
Simulation	To examine the controller simulate the whole system
Inertia tensor calculation	To consider the mass variation, moment of inertia of the main components is calculated
Parameter estimation	Using the Recursive Least Square Estimator (RLSE), estimate the mass of the Quad-copter
Simulation after mass variation	Simulate to check if the controller can handle the mass variation
Image recognition	Collecting the datasets of locust swarm and train the system, then test the model using python programming
Final simulation	Simulate considering mass variation and image recognition

Table 1.1: Methodology

1.5 Scope

In this thesis the dynamics of quad-copter is studied and a mathematical model will be formulated using the Newton-Euler formulation. Based on the model, a controller is designed. The controller governs the quad-copter to track the reference and use less amount of control effort. NMPC is used as a controller for this thesis. The parameters or the specifications of the quad-copter for simulation are adopted from JMR-X1400, which is an agriculture spray drone.

For considering the mass variation, Recursive Least Square Estimation is used and for image recognition, Convolutional Neural Network (CNN) in Python programming language is implemented. the simulation of the proposed techniques uses Simulink, CasADi in MATLAB and Visual Studio for python programming.

1.6 Thesis Outline

Chapter 2 provides key literature reviews.
Chapter 3 provides an explanation how the dynamics of the model is obtained using Newton-Euler formalism and verified in Simulink .
Chapter 4 presents the design of the controller, so that the quad-copter tracks the reference with minimum energy consumption.
Chapter 5 discuses the method used for considering the mass variation, which is Recursive Least Square Estimation Method and show the Image Recognition procedure using CNN in python.
Chapter 6 provides the simulation result for the controller and for image recognition test result.
Chapter 7 provides the conclusion drawn and recommendations for future work.
References used in this thesis are presented in numerical order.
Additional concepts and proofs that are relevant to this work are included in appendix.

Chapter 2

Literature Review

This chapter provides key examples of literature reviews.

A lot of work has been done in the areas of advanced control of UAVs, different categories of Model Predictive controller and image recognition. These works are too numerous for comprehensive listing. So key examples will be provided.

2.1 Literature reviews on Quad-copter

Quad-copter has received substantial attention from researchers as the complex phenomena of the quad-copter has created numerous areas of interest. The starting point for all of the studies is the basic dynamical model of the quad-copter.

In the paper by R.Tesfaye [2] the modeling and control of a quad-copter UAV at hovering position is done, and in her work the dynamics of quad-copter and the mathematical model was formulated using the Newton-Euler formulation. Both LQR and PID controllers were implemented and compared. But the paper does not provide the model and control of a non-hovering operation.

In [3], the quad-copters dynamics is presented using differential equations. The dynamics are derived from both the Euler-Lagrange equation and Newton-Euler equations. Performance of the model was verified by simulating the flight of a quad-copter model using MATLAB and PD controller was used for stabilizing the attitude of the quad-copter. It concludes that the simulation presented proves the mathematical model of the quad-copter to be realistic. However, position x and y were not considered during the formulation of PD controller.

2.2 Literature reviews on MPC

Optimal control is a popular control technique which is hugely discussed in different literatures. This control approach selects the optimal control input so that the plant can follow the reference trajectory with minimum control effort. It uses an objective function which consists of the error and the control effort. MPC is one type of optimal control. It is a closed-loop constrained optimal control[4].

In [5], it implements both linear and nonlinear model predictive control law for tracking reference trajectories and constrained control of a quad-copter. To design the nonlinear model predictive control law, it first discretizes the continuous system and then use Sequential Quadratic Programming (SQP) solver while accounting for input, output and state constraints. In this research paper, it shows the difference and similarity of the closed loop stability analysis of the linear and nonlinear control schemes. And it concludes that NMPC outperformed the LMPC with superior transient performance in both nominal conditions and in the presence of disturbance. But it uses a state-dependent coefficient representation of the nonlinear dynamics system.

There are some research papers which compare and contrast different controllers including MPC. Author in [6] compares the performance of 3 different controllers: the Proportional-Derivative (PD) controller, the Sliding Mode Controller (SMC), and the Model Predictive Controller (MPC), when subjected to uncertain load mass on the quad-copter. The uncertain load was treated as bounded external disturbance on the quad-copter. The authors used the small angle approximation to obtain a linear discrete-time space equation for MPC design. The paper concludes that the main impact of the load mass uncertainty is on the quad-copter's stability rather than the trajectory tracking error and if there is a strict requirement on the load transportation stability or fast response, a robust or fast controller design like MPC is suggested.

The authors in [7], formulated a new approach to Nonlinear Model Predictive Control (NMPC). They propose to use multiple shooting method for discretizing the dynamics system, through which the optimal control problem is transformed to a nonlinear program (NLP). In this paper to solve the NLP problem where the state variable and their gradient need to be computed at the end of each shooting, it proposes to use method of collocation on finite elements for the computation. It concludes as: Due to its high numerical accuracy, the computation efficiency for the integration of model equations can be enhanced, in comparison to the existing multiple shooting method where an ODE solver is applied for the integration and the chain-rule for the gradient computation. C++ is used to implement the numerical solution framework. But now the proposed algorithm has been

realized in the framework of the numerical algorithm group (NAG) and Ipopt in the C/C++ environment.

The author in [8], presents an optimization based solution for control and estimation in dynamic systems, specifically implemented on Mobile Robots. It uses MPC to control the mobile robot. It computes both single shooting and multiple shooting methods to change optimal control problem to nonlinear programming problem then evaluates their performance. It states the drawback of single shooting as non linearity propagation, which means integrator function tends to become highly nonlinear for large prediction horizon. It uses CasADi in MATLAB to design the MPC. In this research the mobile robot only has 2 inputs (linear and angular velocities of the robot) and 3 output (translational: X and Y, Rotational: θ).

In [9], it presents a detailed comparison between a classical Linear Model Predictive Controller (LMPC) and the more advanced Nonlinear Model Predictive Controller (NMPC), that considers the full system model. The comparison has been performed during different scenarios: hovering, step response and aggressive trajectory reference tracking under external disturbances. For LMPC it linearized the vehicle model around the hovering condition assuming small attitude angles then it repeatedly solves the optimal control problem. For NMPC it uses multiple shooting to change the optimal control problem (OCP) to nonlinear programming problem (NLP). And to solve the NLP it uses the Sequential Quadratic Programming (SQP). The research concludes that both NMPC and LMPC controllers performed well in reference trajectory tracking while the NMPC was slightly better at disturbance rejection capability.

In [10], the author proposes implementing a Nonlinear Model Predictive Controller (NMPC) which is computationally fast and used as a high-level controller to solve the trajectory tracking problem for Unmanned Aerial Vehicle, Quad-copter. To design the NMPC, the model used is an accurate representation of the plant, So it requires heavy computation. To solve the NMPC, the Newton generalized minimal residual (Newton/GMRES) method is applied. This method solves optimization problem by applying the Hamiltonian method. it concludes that the flaw result of the simulated NMPC showed that the performance of NMPC depends on the accuracy of feedback.

2.3 Quad-copter Mass Variation

The ability of a quad-copter to carry a variable payload is an increasing demand. A payload is the additional weight a quad-copter can carry. If there is a payload variation then there is a mass change, which in turn affects the inertia tensor of the UAV. Inertia tensor describes how the mass is distributed in a rigid body. In [11], it designs an online estimator to determine the dynamic system parameters of a 3 degree of freedom quad-copter hover platform, as the payload is varied (fig 2.1). In particular, the inertia tensor is estimated using the Recursive Least Square Estimation (RLSE) and covariance resetting is incorporated to improve the estimators convergence rate. The paper concludes that the designed estimator was able to estimate accurately the 3x3 inertia tensor matrix with the presence of a sudden payload variation. However in this paper the quad-copter is in hovering position therefore there is no translational movement.

Figure 2.1: 3DOF Quad-copter hover platform

In [12], it presents a simple and effective method on how to estimate the rigid-body inertia based on a two-wire pendulum and using on-board integrated sensor system. It uses quad-copter for the inertia estimation and the quad-copter is suspended by two thin parallel wires to form a bifilar torsional pendulum about the vertical axis. The authors used an on-board sensors and acquire data from the flight controller unit of the quad-copter. Then the pendulum oscillations are documented and processed in order to get a noise-free and trend free signals used in the final inertia estimation phase. It also calculated the moment of inertia of different main components of the UAV and it compared it with the experimental output. It concludes that the developed method of inertia estimation has been experimentally verified.

2.4 Image recognition

In recent years there have been a great advancement in image detection and recognition on several datasets using numerous machine learning algorithms. In particular, deep learning has shown a great enhancement in accuracy on various datasets. In [13], CNN models are built using the

MINIST and CIFAR-10 datasets. MINIST dataset is a dataset that contains handwritten digits and is used for checking performance of a classification algorithm. And CIFAR-10 is a dataset used for object detection and it includes 10 classes to be classified. Therefore the CNN models evaluate its performance on detection datasets and image recognition. The accuracy of the model is 96.6%.

The author in [14], presents a low cost UAV based optical tracking Australian plague locusts employing a technique based on optics. It combines the benefits of transponder tagging and aerial photography that allow for the surveying of large areas, as well as tracking individuals.

In [15], it was attempted to make a model for identification of locusts. And the identification of locust is performed mainly based on the extracted shape and texture of the training image. The performance of the image identification system was assessed by a prototype developed using MATLAB. In this paper the overall success for the identification of locusts is 95.2%.

Chapter 3

Modeling and Model Verification

This chapter provides the main types of configuration of a Quad-copter, the working principle of Quad-copter, Quad-copter Kinematics and Dynamics.

A **Model** is a the development of equations, constraints, and logic rules to represent a system of interest. A model is alike to but simpler than the plant it signifies, while resembling most of the features of the real system as close as possible. A good model is a judicious balance between simplicity and realism. The main feature of a plant model is manipulability. **Modeling** is the act of building a model.

Simulation is the process of using a plant model to study the behavior and performance of a real or theoretical system. The purpose of a simulation is to study the characteristics of a real-life or fictional system by manipulating variables that cannot be controlled in a real system. Simulations allow evaluating a model to optimize system performance or to make predictions about a real system. Simulations are useful to study properties of a model of a real-life system that would otherwise be too complex. While a model aims to be true to the system it represents, a simulation can use a model to explore states that would not be possible in the original system. **Simulating** is the act of using a model for a simulation.

3.1 Modeling

3.1.1 Types of Configuration

A Quad-copter has this two main configurations which are common:

1. Plus (+) configuration, where a single rotor leads the air craft (fig 3.1).

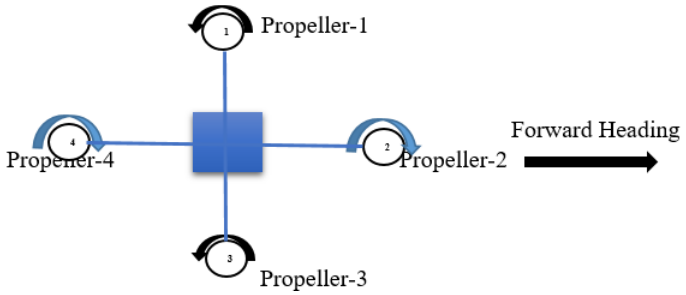


Figure 3.1: Plus Configuration

2. Cross (X) configuration, where two rotors lead the air craft (fig 3.2).

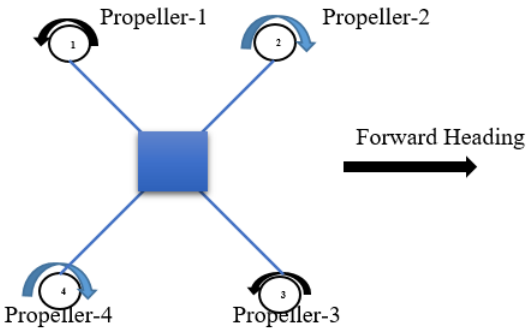


Figure 3.2: Cross Configuration

- For this thesis the cross configuration is used and the main advantage of using cross configuration is:
  - It provides more stability than plus configuration because the cross configuration uses two rotors to lead the quad-copter
  - The plus configuration can cover the camera with its frame.

3.1.2 Common Conventions

Reference frame alignment

There are two reference frames essential to describe the behavior of a quad-copter, the Body-fixed reference frame and Earth inertial reference frame[16]. The reason why two frames are needed is that quad-copter has many sensors, like gyroscope and accelerometer, that give readings with respect to body frame whereas it has other sensors, like GPS and magnetometer, that give readings with respect to inertial frame. Therefore, a means of transformation is required to derive system equations according to a single frame. The **Body-fixed reference frame** is associated with the quad-copter and its origin corresponds to the center of gravity (COG) of the vehicle. Since quad-copter has a symmetrical shape with a central core and four identical rotors attached at its arms, the quad-copter COG is located at the core center. The **Earth inertial reference frame**, on the other hand, is generally represented by North-East-Up coordinates system. The origin of the inertial frame is chosen to be a fixed point on Earth, this will consider all the force and torque that act on the quad-copter.
Quad-copter has 6 degree of freedom, this means that 6 variables are needed to express its position and orientation in space. And to describe the motion of the quad-copter the two reference frames are defined:

1. Body-fixed reference frame (O_B, X_B, Y_B, Z_B)
   - O_B coincides with center of the quad-copter
   - Linear Velocity $(V^B = [u, v, w]^T)$
   - Angular Velocity $(\omega^B = [p, q, r]^T)$
   - Force (F^B)
   - Torque (τ^B)
2. Earth inertial reference frame (O_E, X_E, Y_E, Z_E)
   - This frame is used to define the linear position $(L^E = [X, Y, Z]^T)$ and the angular position $(A^E = [\phi, \theta, \psi]^T)$
   - L^E is determined by coordinates of the vector between O_B and O_E with respect to the earth inertial frame and A^E is orientation of the body-fixed frame with respect to the earth inertial frame

Right hand rule is used to the direction of the coordinate system

$*$ Thumb: Positive direction of X-axis

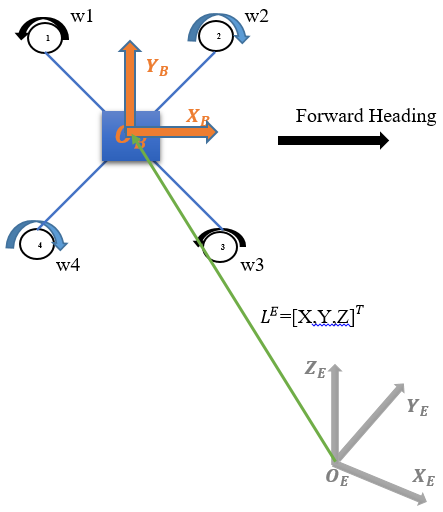


Figure 3.3: Reference Frame

- 1^{st} Finger: Positive direction of Y-axis
- Middle Finger: Positive direction of Z-axis

In the analysis to be followed, for the representation of rotation of the rigid body Euler angle parametrization will be developed. The Euler angles are three angles (ϕ, θ, ψ) used to describe the orientation of a rigid body with respect to a fixed coordinate system.
Z-Y-X convention is used.

Assumptions

- The Quad-copter is a rigid structure
- Thrust is proportional to the square of rotor speed $(F_i = k_f {\omega_i}^2)$
- Torque is proportional to the square of rotor speed$(\tau_i = k_m {\omega_i}^2)$
- The center of mass of the quad-copter and the origin of the body-fixed frame coincide.
- The specification of the quad-copter design is obtained from JMR-X1400[17], which is an Agriculture spray drone (fig 3.4).The Specifications are:

  1. Model number: JMR-X1400
  2. Net weight: 10.5Kg
  3. Tank Capacity: 10.2L
  4. Total weight with full tank: 20.7Kg

5. Maximum take off weight: 26Kg
6. Dimension (lxwxh): $109 \times 109 \times 45$cm

Figure 3.4: JMR-X1400: agricultural spray Quad-copter

3.1.3 Working principle of Quad-copter

There are 4 basic movements which allow the quad-copter to reach a certain height and attitude (orientation).

1. **Throttle** It is performed by increasing the speed of all four rotors simultaneously (fig 3.5). This action will increase the thrust of the quad-copter without producing any net moments.

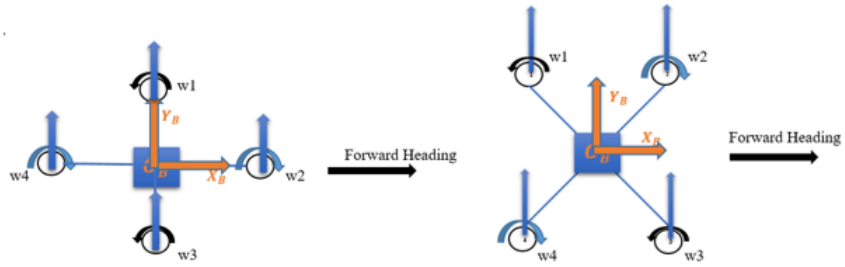


Figure 3.5: Throttle

2. **Roll** It is performed by increasing or decreasing the speed of the rotor on the left and by decreasing or increasing the speed of the rotor on the right (fig 3.6). The plus configuration uses only two rotors to roll. While the cross configuration engages all of its rotors to perform rolling.

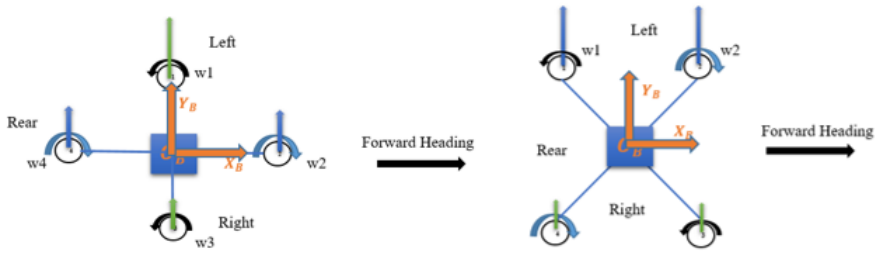


Figure 3.6: Roll

3. **Pitch** It is performed by increasing or decreasing the speed of the rotor on the rear and by decreasing or increasing the speed of the rotor on the front (fig 3.7). The plus configuration uses only two of its rotors in contrary to the cross configuration which uses all of its four rotors.

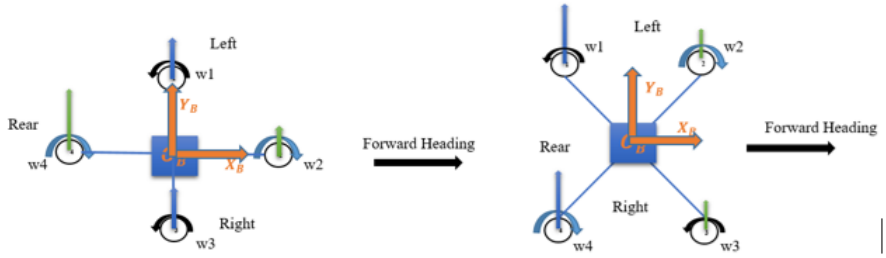


Figure 3.7: Pitch

4. **Yaw** For plus configuration: It is performed by increasing or decreasing the speed of the rotor on the first pair (rear and front) and by decreasing or increasing the speed of the rotor on the second pair (left and right). For cross configuration: It is performed by increasing or decreasing the speed of the rotor of the first pair (top right(w2) and bottom left(w4)) and by increasing or decreasing the speed of the rotor of the second pair (top left(w1) and bottom right(w3)) (fig 3.8).

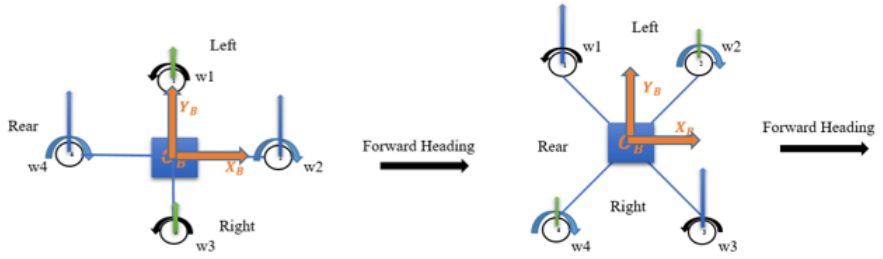


Figure 3.8: Yaw

Remark:

– For the plus configuration, considering the pitch movement, where a single rear rotor speeds up and a single front rotor slows down it generates a nose down pitching. Since torque does not vary linearly with speed of a rotor ($\tau_i = k_f \omega_i^2$) the increase in torque of the CW spinning rear rotor does not identically cancel with the torque reduction of the CW spinning front rotor, resulting net yaw moment. So, the plus configuration requires a compensation with a yaw control input. Similarly, considering the roll movement, where a single left rotor speeds up and a single right rotor slows down it generates a roll-right moment.

– For the cross configuration, considering the pitch movement, where the two rear rotors speed up and the two front rotors slow down it generates a nose-down pitching. Of the two rear rotors speeding up one rotates in CW direction and the other in CCW direction, and the torque generated cancels out. The same is true for the front rotors slowing down. Thus, pitch control does not introduce a net yaw moment. Similarly, considering the roll movement, where the two right rotors slow down and the two left rotors speed up it generates a roll-right movement. Of the two right rotors slowing down one rotates in CW direction and the other in CCW direction, and the torque generated cancels out. The same is true for the left rotors speeding up. Thus, roll control does not introduce a net yaw moment in cross configuration.

3.1.4 Quad-copter Kinematics

Kinematics is a branch of mechanics which studies the motion of a plant or a system without considering the Force and Torque acting on it. To describe the motion of 6 DOF rigid body it is useful to define two reference frames(fig 3.3).

Rotation Matrix

Rotation matrices are 3×3 matrices that have the properties of orthogonality and determinant which is equal to 1. Euler angles are often used to describe rotations but, There are also other methods such as Quaternions. Quaternions offer a singularity-free attitude representation. In contrary to the approach used in Euler angle's representation, Quaternionic attitude representation uses a single axis and an angle of rotation to describe attitude. Quaternions lack clear physical intuition which Euler angles have. For this thesis Euler angle representation is used.

Euler angles can be defined by a composition of rotations. The composition of elemental rotations about the axes of a certain coordinate system is sufficient to reach any target orientation. Using this principle, rotation in a rigid body in 3-dimensional space will be expressed using the

composition of three consecutive rotations in the analysis to be followed.

Since Z-Y-X convention is used, 1^{st} the rotation about Z_E axis of the angle ψ(Yaw), 2^{nd} the rotation about the Y axis of the angle θ(Pitch) and at last the rotation about the X axis of the angle ϕ(Roll)

$$ {}^{E}_{B}R = \begin{bmatrix} cos(\theta)cos(\psi) & sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi) \\ cos(\theta)sin(\psi) & cos(\theta)cos(\psi) + sin(\phi)sin(\theta)sin(\psi) & -sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\phi)cos(\theta) \end{bmatrix} \tag{3.1} $$

The rotation matrix ${}^{E}_{B}R$(Matrix 3.1) is used to transform linear quantities from body-fixed frame (rotating frame) to earth inertial frame.
Transformation of linear velocity 'V^B':

$$ \begin{bmatrix} \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \end{bmatrix} = {}^{E}_{B}R \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.2} $$

Details of the successive rotations from the rotating frame to earth inertial frame are shown in Appendix A.

Transfer Matrix

$A^E = [\phi, \theta, \psi]^T$ is in inertial frame and Angular Velocity ($\omega^B = [p, q, r]^T$) is in body frame to relate the two we need a transfer matrix(T): $\dot{A}^E = T\omega^B$

$$ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.3} $$

$$ T = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix}, T^{-1} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{3.4} $$

The details about the derivation of the transfer matrix is in Appendix B

3.1.5 Quad-copter Dynamics

Dynamics is a branch of mechanics which studies the effect of Forces and Torques on the motion of a plant or a system. There are different techniques which can be used to drive the equations of a rigid body with 6 DOF. For this thesis Newton-Euler method is used.

Newton-Euler Method

The Newton-Euler method will be used to model the flight dynamics of the Quad-copter. The equations of motion are developed as a combination of the translational and rotational parameters of the six-degree-of-freedom system(3 position and 3 angular orientation), using Euler's two laws of motion (Newton second law and Newton first law (law of inertia)). Common parameters used for the formulation are:

- Linear Velocity ($V^B = [u, v, w]^T$)
- Angular Velocity ($\omega^B = [p, q, r]^T$)
- Position Vector ($A^E = [\phi, \theta, \psi]^T$)
- Orientation Vector ($L^E = [X, Y, Z]^T$)

$$\begin{bmatrix} \text{Roll rate} \\ \text{Pitch rate} \\ \text{Yaw rate} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}_B \tag{3.5}$$

External Forces and Moments

1. **Force**: affects the translational motion of a quad-copter. From Newton second law, the forces acting on a quad-copter can be expressed as: $\sum F = F_{trust} + F_{grav} = M\vec{a}$. The translational dynamics of the quad-copter can be found using both frames:

   I In **Body-fixed reference frame**:
   Trust Force: is the total sum of all trust generated by the propellers. Trust has quadratic relation with angular velocity ($F_i = k_f {\omega_i}^2$). where: k_f is constant and depends on many factors such as: Density of the surrounding air, area swept by propeller, Back EMF, Torque proportionality constant etc. It is found empirically (experimental and observation).

   $$F^B_{trust} = F_1 + F_2 + F_3 + F_4 = k_f {\omega_1}^2 + k_f {\omega_2}^2 + k_f {\omega_3}^2 + k_f {\omega_4}^2 \tag{3.6}$$

   Gravitational Force: is the force of gravity acting on the quad-copter. It occurs only in the Z-direction and it is more intuitively expressed in the earth inertial frame. It can

be expressed with respect to the body-fixed frame using Euler angles.

$$F^B_{grav} = ({}^E_B R)^{-1} F^E_{grav}, \text{ where: } ({}^E_B R)^{-1} = ({}^E_B R)^T, \text{ and } F^E_{grav} = \begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix} \tag{3.7}$$

$$F^B_{grav} = \begin{bmatrix} Mgsin(\theta) \\ -Mgcos(\theta)sin(\phi) \\ -Mgcos(\theta)cos(\phi) \end{bmatrix} \tag{3.8}$$

Using **Newton Euler method**, Euler's 1^{st} axioms of Newton 2^{nd} law: $F = M\vec{a}$, in body-fixed frame $\sum F^B = M\frac{d\vec{v}}{dt} \rightarrow \vec{v} = ui + vj + wk$, To find the acceleration, $\vec{v}$ is derivated:

$\frac{d\vec{v}}{dt} = \frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k + \frac{di}{dt}u + \frac{dj}{dt}v + \frac{dk}{dt}w$ If body orientation was constant, the derivative would simply be the derivative of the velocity magnitude in each i, j, k components of the reference frame because the unit vectors (i, j, k) would be constant, therefore their derivative is zero. With rotating body the chain rule derivation should be considered (take account). The derivation is in Appendix C.
Application of the chain rule results the following:

$$\frac{d\vec{v}}{dt} = \frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k + (\omega \times i)u + (\omega \times j)v + (\omega \times k)w \tag{3.9}$$

$$\sum F^B = M\frac{d\vec{v}}{dt} = \underbrace{M\left(\frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k\right)}_{\mathrm{M}\dot{\vec{v}}} + \underbrace{M((\omega \times i)u + (\omega \times j)v + (\omega \times k)w)}_{\text{Coriols force}} \tag{3.10}$$

$$coriolsforce = M(\omega \times (ui + vj + wk), \text{ where: } \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.11}$$

$$coriolsforce = M\left(\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}\right) \tag{3.12}$$

After the cross multiplication:

$$\sum F^B = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} M(\dot{u} + (qw - vr)) \\ M(\dot{v} + (ru - pw)) \\ M(\dot{w} + (pv - uq)) \end{bmatrix} \tag{3.13}$$

$$\sum F^B = F^B_{trust} + F^B_{grav} = M\vec{a} \tag{3.14}$$

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ F_1+F_2+F_3+F_4 \end{bmatrix}}_{F^B_{trust}} + \underbrace{\begin{bmatrix} Mgsin(\theta) \\ -Mgcos(\theta)sin(\phi) \\ -Mgcos(\theta)cos(\phi) \end{bmatrix}}_{F^B_{grav}} = \underbrace{\begin{bmatrix} M(\dot{u}+(qw-vr)) \\ M(\dot{v}+(ru-pw)) \\ M(\dot{w}+(pv-uq)) \end{bmatrix}}_{M\vec{a}} \tag{3.15}$$

$\therefore$ The dynamic equation of the translational motion defined in the body fixed frame is written as:

$$\begin{aligned} \boldsymbol{\dot{u} = gsin(\theta) - (qw - vr)} \\ \boldsymbol{\dot{v} = -gsin(\phi)cos(\theta) - (ru - pw)} \\ \boldsymbol{\dot{w} = -gcos(\phi)cos(\theta) - (pv - uq) + \frac{F^B_{trust}}{M}} \end{aligned} \tag{3.16}$$

II The **Earth inertial reference frame** description of the translational forces:

Trust Force: is expressed in body-fixed frame and it's earth inertia frame description can be acquired using the rotation matrix (${}^E_BR$).

$F^E_{trust} = {}^E_BR * F^B_{trust}$

$$F^E_{trust} = \begin{bmatrix} (cos(\phi)sin(\theta)cos(\psi)+sin(\phi)sin(\psi))F^B_{trust} \\ (cos(\phi)sin(\theta)sin(\psi)-sin(\phi)cos(\psi))F^B_{trust} \\ (cos(\phi)cos(\theta))F^B_{trust} \end{bmatrix} \tag{3.17}$$

Gravitational Force: is expressed in the earth inertial frame.

$$F^E_{grav} = \begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix} \tag{3.18}$$

Using **Newton Euler method**, Euler's 1^{st} axioms of Newton 2^{nd} law: $F = M\vec{a}$, in earth inertial reference frame

$$\sum F^E = F^E_{trust} + F^E_{grav} = M\vec{a} \tag{3.19}$$

$$\underbrace{\begin{bmatrix} (cos(\phi)sin(\theta)cos(\psi)+sin(\phi)sin(\psi))F^B_{trust} \\ (cos(\phi)sin(\theta)sin(\psi)-sin(\phi)cos(\psi))F^B_{trust} \\ (cos(\phi)cos(\theta))F^B_{trust} \end{bmatrix}}_{F^E_{trust}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix}}_{F^E_{grav}} = \underbrace{M\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix}}_{M\vec{a}} \tag{3.20}$$

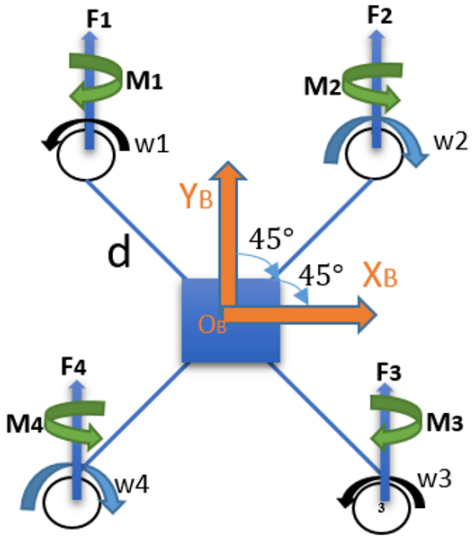
Let: $\boldsymbol{U_1 = F^B_{trust} = k_f(\omega_1{}^2 + \omega_2{}^2 + \omega_3{}^2 + \omega_4{}^2)}$

$\therefore$ The dynamics equation of the translational motion defined in the earth inertial reference frame is written as:

$$\begin{aligned}\ddot{X} &= (cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi))\frac{U_1}{M}\\ \ddot{Y} &= (cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi))\frac{U_1}{M}\\ \ddot{Z} &= (cos(\phi)cos(\theta))\frac{U_1}{M} - g\end{aligned} \tag{3.21}$$

2. **Torque**: affects the rotational motion of a quad-copter. Analogous to the developed translational dynamics, the rotational dynamics can be formulated as follows by taking the developed net torque and reaction forces in to consideration: $\sum \tau = M_{trust} + F_{gyro}$. The rotational dynamics of the quad-copter can be found using both frames:

   I In **Body-fixed reference frame**:
   Moments of trust forces: $\sum (r_i \times F_i)$, Reaction moments has quadratic relation with angular velocity $M_i = k_m {\omega_i}^2$, constant k_m. k_m depends on propellers: number of blade, diameter, pitch, material, air viscosity...
   Total moment in each coordinate:



$$\begin{aligned}\tau_x^B &= -F_1 dcos(45°) + F_2 dcos(45°) + F_3 dcos(45°) - F_4 cos(45°)\\ \tau_y^B &= F_1 dsin(45°) + F_2 dsin(45°) - F_3 dsin(45°) - F_4 dsin(45°)\\ \tau_z^B &= -M_1 + M_2 - M_3 + M_4\end{aligned} \tag{3.22}$$

Where d is the distance from center of Quad-copter to center of propeller

$$\begin{aligned}\tau_x^B &= -F_1 d\frac{\sqrt{2}}{2} + F_2 d\frac{\sqrt{2}}{2} + F_3 d\frac{\sqrt{2}}{2}) - F_4 d\frac{\sqrt{2}}{2}\\ \tau_y^B &= F_1 d\frac{\sqrt{2}}{2} + F_2 d\frac{\sqrt{2}}{2} - F_3 d\frac{\sqrt{2}}{2} - F_4 d\frac{\sqrt{2}}{2}\\ \tau_z^B &= -M_1 + M_2 - M_3 + M_4\end{aligned} \tag{3.23}$$

$F_i = k_f {\omega_i}^2$ and $M_i = k_m {\omega_i}^2$ (Reaction moment) ,For this thesis CCW direction is Positive and CW direction is Negative.

$$\begin{aligned}\tau_x^B &= \frac{\sqrt{2}}{2} d k_f({\omega_1}^2 + {\omega_2}^2 - {\omega_3}^2 - {\omega_4}^2)\\ \tau_y^B &= \frac{\sqrt{2}}{2} d k_f({\omega_1}^2 - {\omega_2}^2 - {\omega_3}^2 + {\omega_4}^2)\\ \tau_z^B &= -M_1 + M_2 - M_3 + M_4 = -k_m {\omega_1}^2 + k_m {\omega_2}^2 - k_m {\omega_3}^2 + k_m {\omega_4}^2\end{aligned} \tag{3.24}$$

In Equation(3.24)Let: $\boldsymbol{\tau_x^B = U_2}$ = Roll, $\boldsymbol{\tau_y^B = U_3}$ = Pitch and $\boldsymbol{\tau_z^B = U_4}$ = Yaw (3.25)

Gyroscopic effects

Since two of the propellers are rotating CW (propeller 2 and 4) and the other two rotating CCW (propeller 1 and 3), an overall imbalance will happen when the algebraic sum of the rotor speeds is not equal to zero. This imbalance will cause a gyroscopic effect. Furthermore, this effect is proportional to the roll and pitch rates. The following equation defines the overall propellers speed.

$${F^B}_{gyro} = -\sum_{k=1}^{4} Jtp\left(\begin{bmatrix}p\\q\\r\end{bmatrix} \times \begin{bmatrix}0\\0\\1\end{bmatrix}\right)(-1)^k \omega_k \tag{3.26}$$

$$\begin{bmatrix}p\\q\\r\end{bmatrix} \times \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}q\\-p\\0\end{bmatrix} \tag{3.27}$$

$${F^B}_{gyro} = -\sum_{k=1}^{4} Jtp\left(\begin{bmatrix}q\\-p\\0\end{bmatrix}\right)(-1)^k \omega_k \tag{3.28}$$

$${F^B}_{gyro} = -Jtp\left(\begin{bmatrix}-q\\p\\0\end{bmatrix}\omega_1 + \begin{bmatrix}q\\-p\\0\end{bmatrix}\omega_2 + \begin{bmatrix}-q\\p\\0\end{bmatrix}\omega_3 + \begin{bmatrix}q\\-p\\0\end{bmatrix}\omega_4\right) \tag{3.29}$$

$$F^B{}_{gyro} = -Jtp\left(\begin{bmatrix} -q & q & -q & q \\ p & -p & p & -p \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \right) \tag{3.30}$$

$$F^B{}_{gyro} = Jtp\left(\begin{bmatrix} q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \right) \tag{3.31}$$

Let $\boldsymbol{U_5 = \omega_1 - \omega_2 + \omega_3 - \omega_4}$

$$F^B{}_{gyro} = Jtp\left(\begin{bmatrix} q\omega_1 - q\omega_2 + q\omega_3 - q\omega_4 \\ -p\omega_1 + p\omega_2 - p\omega_3 + p\omega_4 \\ 0 + 0 + 0 + 0 \end{bmatrix} \right) \tag{3.32}$$

$$F^B{}_{gyro} = Jtp\left(\begin{bmatrix} q(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ p(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \\ 0 \end{bmatrix} \right) \tag{3.33}$$

$$F^B{}_{gyro} = Jtp\left(\begin{bmatrix} q(U_5) \\ p(-U_5) \\ 0 \end{bmatrix} \right) \tag{3.34}$$

Euler equation: Euler's second axiom of Newton 2^{nd} law (law of balance of torques) states that in an inertial frame the time rate of change of angular momentum L of an arbitrary portion of a continuous body is equal to the total applied torque M acting on that portion[18]. $\sum \tau^E = M^E = \left(\frac{dL}{dt}\right)_E$

Angular momentum(L) $= I\omega$ where:$I = mR^2$ and R is radius from rotation axis

$Torque = I\alpha \approx F = ma$, I is the inertia tensor.

$\therefore \left(\frac{dL}{dt}\right)_E = \frac{I_E\omega}{dt}$

- **Moment of Inertia (Rotational Inertia)**: It represents how much a body resists being angularly accelerated, So if moment of inertia is big it is hard to rotate the mass(m). It is good to know the moment of inertia because it will be easier to determine how difficult it might be to angularly accelerate an object. $I = mR^2$ and $Torque = I\alpha$

  Inertia Tensor: is a convenient way to summarize all moments of inertia of an object with one quantity. A rigid body's Inertia tensor is expressed as:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{3.35}$$

  I_{xx}: denotes the moment of inertia around the x-axis when objects are rotated around the x-axis.

I_{xy}: denotes the moment of inertia around the y-axis when objects are rotated around the x-axis.
In Matrix equation (3.35) $I_{xy} = I_{yx}, I_{zx} = I_{xz}$ and $I_{yz} = I_{zy}$ but for objects with geometric symmetry like a standard multi-copter one has $I_{xy} = I_{xz} = I_{yz} = 0$
In this thesis there is a mass variation because the quad-copter sprays a bio-pesticide, hence the inertia tensor should be calculated. Moment of inertia is affected by mass ($I = mR^2$:this is for point mass when the total mass rotates with an axis distance R)
The inertia tensor calculation is shown in Appendix D

Angular momentum$(L_E) = I_E\omega$

$$\text{Angular momentum}(L_E) = I\omega = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} I_{xx}\omega_1 \\ I_{yy}\omega_2 \\ I_{zz}\omega_3 \end{bmatrix}_E \tag{3.36}$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.37}$$

$$L_E = \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix}_E \tag{3.38}$$

$L_E = I_E\omega$: works well if in body frame because in the inertial frame the orientation of the body will be continuously changing which means it is not possible to use a constant inertia tensor, It should be calculated all the time. therefore, it is for body frame we can write a constant inertia tensor times the angular velocity.

$$\left(\frac{dL}{dt}\right)_{B-frame} = \left(\frac{dL}{dt}\right)_{E-frame} + \vec{\omega} \times L \tag{3.39}$$

$\tau^{\mathbf{B}} = \dot{\tilde{\mathbf{L}}} + \tilde{\omega} \times \mathbf{L}$: This is known as Euler Equation

$$\tau^B = \begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix} \tag{3.40}$$

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} rqI_{zz} - rqI_{yy} \\ rpI_{xx} - rpI_{zz} \\ pqI_{yy} - pqI_{xx} \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} + rq(I_{zz} - I_{yy}) \\ I_{yy}\dot{q} + rp(I_{xx} - I_{zz}) \\ I_{zz}\dot{r} + pq(I_{yy} - I_{xx}) \end{bmatrix} \tag{3.41}$$

The equation 3.41 determines the motion relative to the body frame and the total torque

applied is now given in the body frame.

$$\sum M^B = M^B_{trust} + F^B_{gyro} = \dot{\vec{L}} + \vec{\omega} \times L \tag{3.42}$$

$$\underbrace{\begin{bmatrix} I_{xx}\dot{p} + rq(I_{zz} - I_{yy}) \\ I_{yy}\dot{q} + rp(I_{xx} - I_{zz}) \\ I_{zz}\dot{r} + pq(I_{yy} - I_{xx}) \end{bmatrix}}_{\dot{\vec{L}} + \vec{\omega} \times L} = \underbrace{\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix}}_{M^B_{trust}} + \underbrace{Jtp\left(\begin{bmatrix} q(U_5) \\ p(-U_5) \\ 0 \end{bmatrix} \right)}_{F^B_{gyro}} \tag{3.43}$$

$\therefore$ The dynamics equation of the rotational motion defined in the body-fixed reference frame is written as:

$$\begin{aligned} \dot{\boldsymbol{p}} &= \frac{\boldsymbol{U_2}}{\boldsymbol{I_{xx}}} - \frac{\boldsymbol{qr(I_{zz} - I_{yy})}}{\boldsymbol{I_{xx}}} + \frac{\boldsymbol{JtpU_5 q}}{\boldsymbol{I_{xx}}} \\ \dot{\boldsymbol{q}} &= \frac{\boldsymbol{U_3}}{\boldsymbol{I_{yy}}} - \frac{\boldsymbol{rp(I_{xx} - I_{zz})}}{\boldsymbol{I_{yy}}} - \frac{\boldsymbol{JtpU_5 p}}{\boldsymbol{I_{yy}}} \\ \dot{\boldsymbol{r}} &= \frac{\boldsymbol{U_4}}{\boldsymbol{I_{zz}}} - \frac{\boldsymbol{pq(I_{yy} - I_{xx})}}{\boldsymbol{I_{zz}}} \end{aligned} \tag{3.44}$$

II The **Earth inertial reference frame** dynamics description is obtained by using the transfer matrix:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = T \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.45}$$

$$\begin{aligned} \ddot{\boldsymbol{\phi}} &= \dot{\boldsymbol{p}} + \dot{\boldsymbol{q}}(\boldsymbol{sin(\phi)tan(\theta)}) + \dot{\boldsymbol{r}}(\boldsymbol{cos(\phi)tan(\theta)}) \\ \ddot{\boldsymbol{\theta}} &= \dot{\boldsymbol{q}}(\boldsymbol{cos(\phi)}) - \dot{\boldsymbol{r}}(\boldsymbol{sin(\phi)}) \\ \ddot{\boldsymbol{\psi}} &= \dot{\boldsymbol{q}}(\boldsymbol{sin(\phi)sec(\theta)}) + \dot{\boldsymbol{r}}(\boldsymbol{cos(\phi)sec(\theta)}) \end{aligned} \tag{3.46}$$

3. The Dynamics of a Quad-copter:
   $\therefore$ there are 4 control inputs for the quad-copter: $U = [U_1 U_2 U_3 U_4]^T$, where U_1 is the resultant force of lift force (Altitude control input), U_2 is the resultant force that affects the roll angle of the quad-copter, U_3 is the resultant force that affects the pitch angle of the quad-copter

and U_4 is the resultant force that affects the yaw angle of the Quad-copter.

$$\begin{aligned}
\ddot{X} &= (cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi))\frac{U_1}{M} \\
\ddot{Y} &= (cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi))\frac{U_1}{M} \\
\ddot{Z} &= (cos(\phi)cos(\theta))\frac{U_1}{M} - g \\
\ddot{\phi} &= \frac{U_2}{I_{xx}} - \frac{(I_{zz} - I_{yy})}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{J_{tp}\dot{\theta}U_5}{I_{xx}} \\
\ddot{\theta} &= \frac{U_3}{I_{yy}} - \frac{(I_{xx} - I_{zz})}{I_{yy}}\dot{\phi}\dot{\psi} - \frac{J_{tp}\dot{\phi}U_5}{I_{yy}} \\
\ddot{\psi} &= \frac{U_4}{I_{zz}} - \frac{(I_{yy} - I_{xx})}{I_{zz}}\dot{\phi}\dot{\theta}
\end{aligned} \tag{3.47}$$

4. To find the desired Roll (ϕ) and Pitch (θ) angles:

$$\phi_d = sin^{-1}\left(\frac{m}{U_1}(\ddot{X}sin(\psi) - \ddot{Y}cos(\psi))\right) \tag{3.48}$$

$$\theta_d = sin^{-1}\left(\frac{m\ddot{X}}{U_1} - (sin(\phi)sin(\psi))\right)\frac{1}{\cos(\phi)cos(\psi)} \tag{3.49}$$

   The details about the calculation is in Appendix E

5. The rotors angular velocity is calculated from the control inputs, which an inverse relationship is defined as:

$$\begin{aligned}
\omega_1 &= \sqrt{\frac{U_1}{4k_f} + \sqrt{2}\frac{U_2}{4k_f d} + \sqrt{2}\frac{U_3}{4k_f d} - \frac{U_4}{4k_m}} \\
\omega_2 &= \sqrt{\frac{U_1}{4k_f} + \sqrt{2}\frac{U_2}{4k_f d} - \sqrt{2}\frac{U_3}{4k_f d} + \frac{U_4}{4k_m}} \\
\omega_3 &= \sqrt{\frac{U_1}{4k_f} - \sqrt{2}\frac{U_2}{4k_f d} - \sqrt{2}\frac{U_3}{4k_f d} - \frac{U_4}{4k_m}} \\
\omega_4 &= \sqrt{\frac{U_1}{4k_f} - \sqrt{2}\frac{U_2}{4k_f d} + \sqrt{2}\frac{U_3}{4k_f d} + \frac{U_4}{4k_m}}
\end{aligned} \tag{3.50}$$

3.2 Model Verification

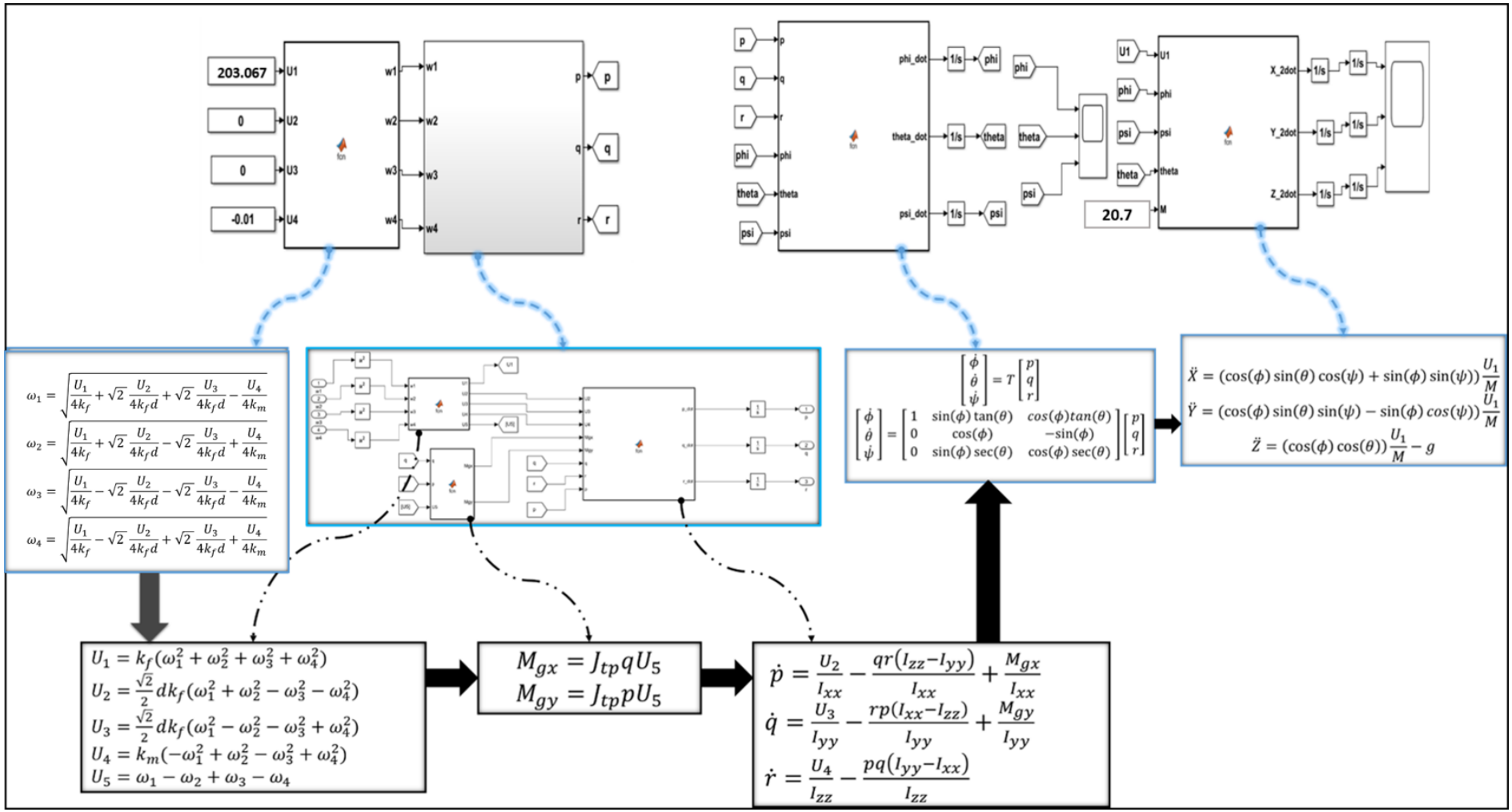


Figure 3.9: Model Dynamics

Parameter (Unit)	**Symbol**	**Value**
Mass of quad-copter (kg)	M	20.7
Drag coefficient (Nms^2)	k_m	$1.1 * 10^{-6}$
Trust coefficient (Ns^2)	k_f	$54.2 * 10^{-6}$
Inertia around X-axis (Nms^2)	I_{xx}	1.9579
Inertia around Y-axis (Nms^2)	I_{yy}	1.957
Inertia around Z-axis (Nms^2)	I_{zz}	0.9318
Gravitational acceleration (ms^{-2})	g	9.81
Distance from center of quad-copter to center of propeller distance (m)	d	0.5205
Inertia around propeller(Nms^2)	Jtp	0.000104

Table 3.1: Quad-copter Parameters for Model Verification

There are Four control inputs in Quad-copter dynamics ($U_1 - Trust, U_2 - Roll, U_3 - Pitch, U_4 - Yaw$) and to verify the model the inputs are given different values to see the change they make in the states.

1. **Thrust**: all rotor speed up or slow down at the same time ($\omega_1 = \omega_2 = \omega_3 = \omega_4$). $U_1 = k_f(\omega_1 + \omega_2 + \omega_3 + \omega_4)$

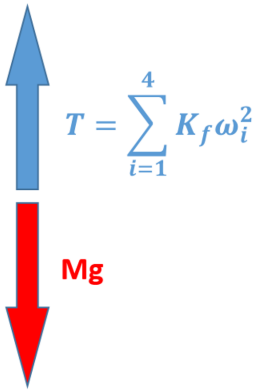
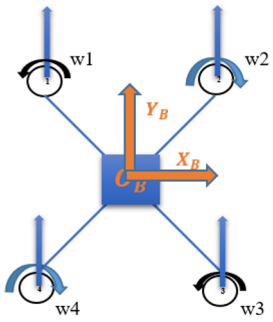
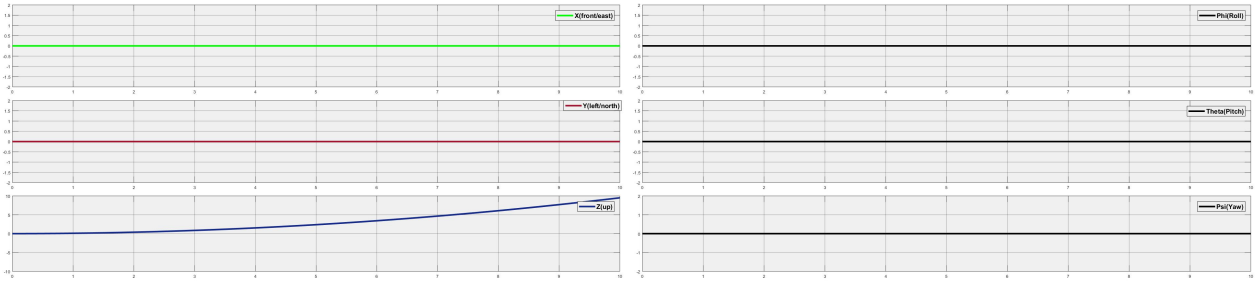


Figure 3.10: Force decomposition in Z-direction

- $\boldsymbol{U_1 = 204}$: Increase speed of all rotors ($Mg = 20.7kg * 9.81ms^{-2} = 203.067$, $U_1 > Mg$). The quad-copter goes upward.

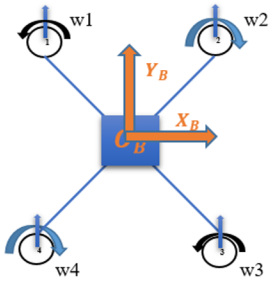


$T > Mg$....Upward Movement

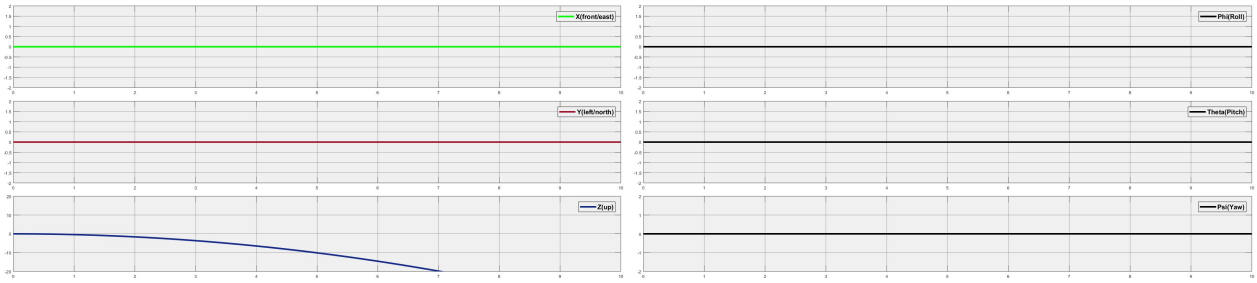


XYZ and Euler Angles plot for U_1 = 204

- $\boldsymbol{U_1 = 202}$: decrease speed of all rotors ($U_1 < Mg$). The quad-copter goes downward.



$T < Mg$....Downward Movement



XYZ and Euler Angles plot for U_1 = 202

Note: From fig(3.10) if $T = Mg$ then the quad-copter Hovers

2. **Roll**: $U_2 = \frac{\sqrt{2}}{2} dk_f({\omega_1}^2 + {\omega_2}^2 - {\omega_3}^2 - {\omega_4}^2)$

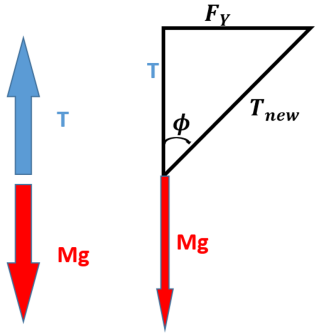
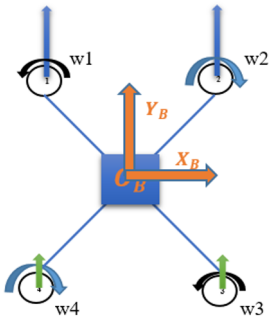
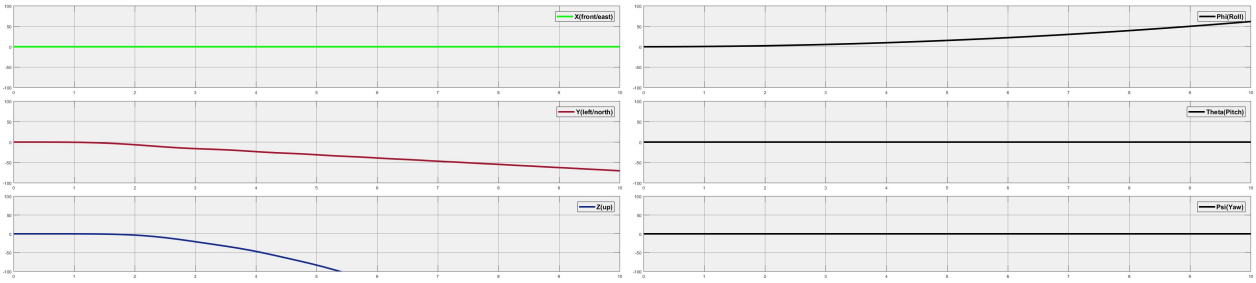


Figure 3.13: Force decomposition during Roll movement

- $\boldsymbol{U_2 = 0.01}$: $\omega_1 \& \omega_2 > \omega_3 \& \omega_4$, quad-copter moves to the right.

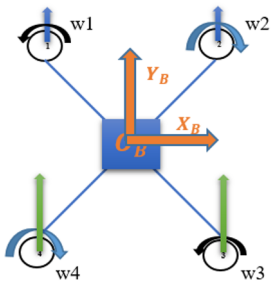


$U_2 > 0$....Moves to negative Y-axis

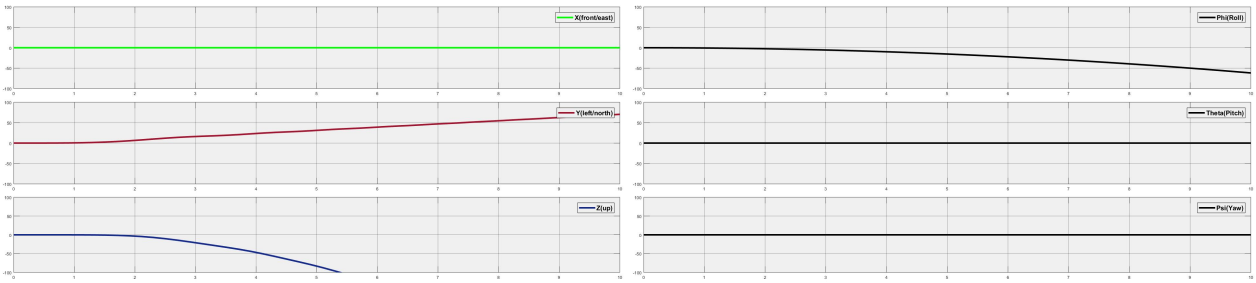


Euler Angles and XYZ Plot for U_2 = 0.01

- $\boldsymbol{U_2 = -0.01}$: $\omega_1 \& \omega_2 < \omega_3 \& \omega_4$, quad-copter moves to the left.



$U_2 < 0$....Moves to positive Y-axis



Euler Angles and XYZ Plot for U_2 = -0.01

Note: In both cases the quad-copter goes downward in Z-direction because: from fig(3.13) $F_y = T_{new} sin(\phi)$ and $T = T_{new} cos(\phi)$, where: $T = Mg(Hovering)$

3. **Pitch**: $U_3 = \frac{\sqrt{2}}{2} dk_f({\omega_1}^2 - {\omega_2}^2 - {\omega_3}^2 + {\omega_4}^2)$

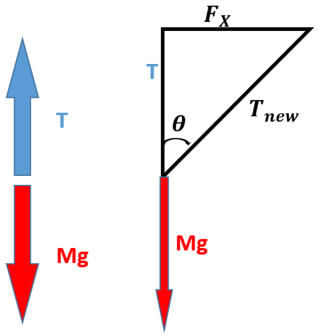
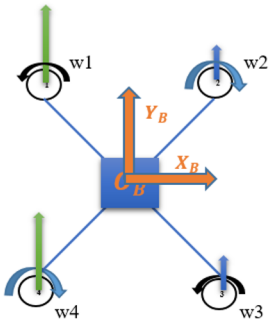
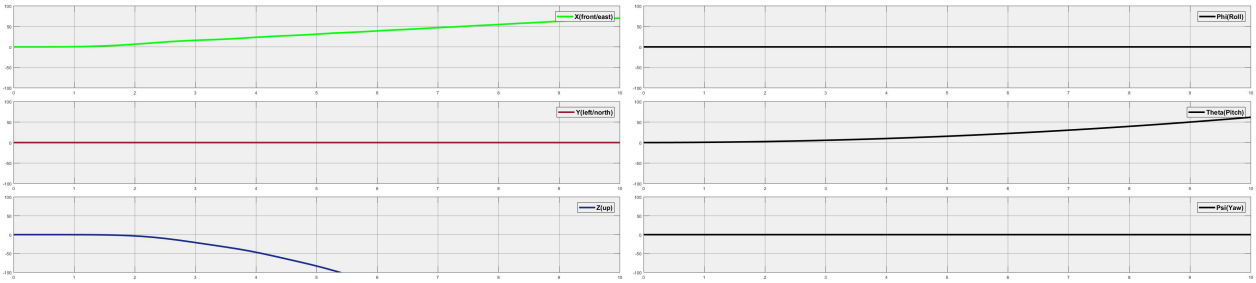


Figure 3.16: Force decomposition during Pitch movement

- $\boldsymbol{U_3 = 0.01}$: $\omega_1 \& \omega_4 > \omega_2 \& \omega_3$, quad-copter moves forward.

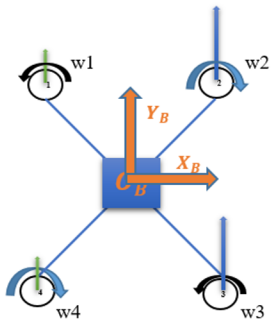


$U_3 > 0$....Moves to positive X-axis

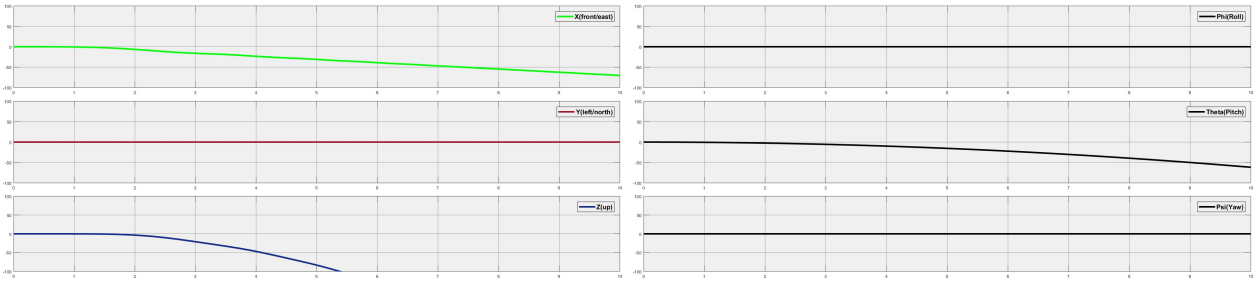


Euler Angles and XYZ Plot for $U_3 = 0.01$

- $\boldsymbol{U_3 = -0.01}$: $\omega_1 \& \omega_4 < \omega_2 \& \omega_3$, quad-copter moves backward.



$U_3 < 0$....Moves to negative X-axis

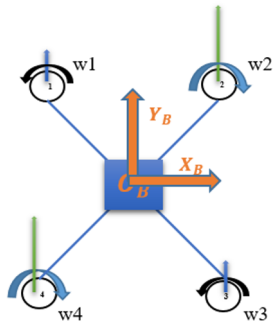


Euler Angles and XYZ Plot for U_3 = -0.01

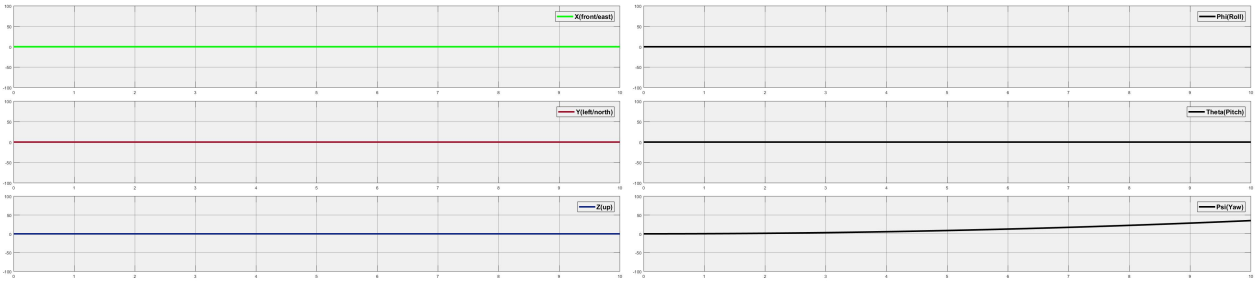
Note: In both cases the quad-copter goes downward in Z-direction because: from fig(3.16) $F_x = T_{new}sin(\theta)$ and $T = T_{new}cos(\theta)$, where: $T = Mg(Hovering)$

4. **Yaw**: $U_4 = -k_m{\omega_1}^2 + k_m{\omega_2}^2 - k_m{\omega_3}^2 + k_m{\omega_4}^2$

   - $\boldsymbol{U_4 = 0.01}$: $\omega_1 \& \omega_3 < \omega_2 \& \omega_4$, quad-copter rotates CCW

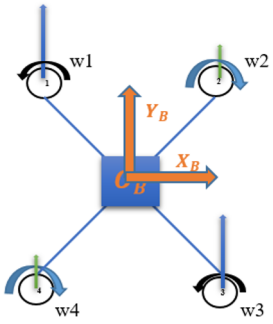


$U_4 > 0$....Rotates CCW

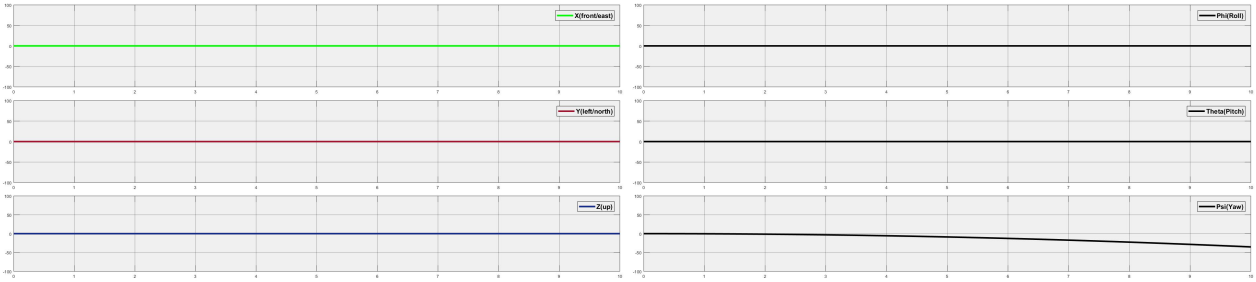


Euler Angles and XYZ Plot for U_4 = 0.01

- $\boldsymbol{U_4 = -0.01}$: $\omega_1 \& \omega_3 > \omega_2 \& \omega_4$, quad-copter rotates CW



$U_4 < 0$....Rotates CW



Euler Angles and XYZ Plot for U_4 = -0.01

Note: In both cases the quad-copter's position vectors are not affected

Chapter 4

MPC Design of a Quad-copter

This chapter provides basic introduction to Model Predictive Controller and its Working Principle, Mathematical Formulation and Overall Design Procedure.

4.1 Introduction

In a control problem, the objective of the controller is to compute the input to the plant such that the plant output follows a desired reference. A Model Predictive Control(MPC) approach to calculate this input is to predict the future of the plant output. MPC uses a model of the plant to make prediction about the future plant behavior. It also uses an optimizer, which certifies that the predicted future plant output tracks the desired reference.

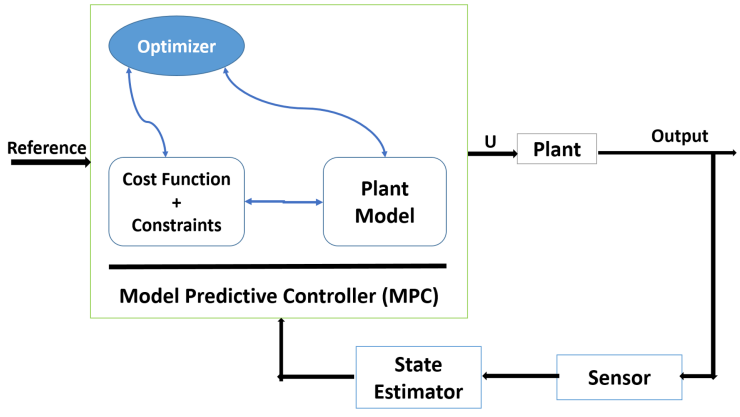


Figure 4.1: MPC Controller

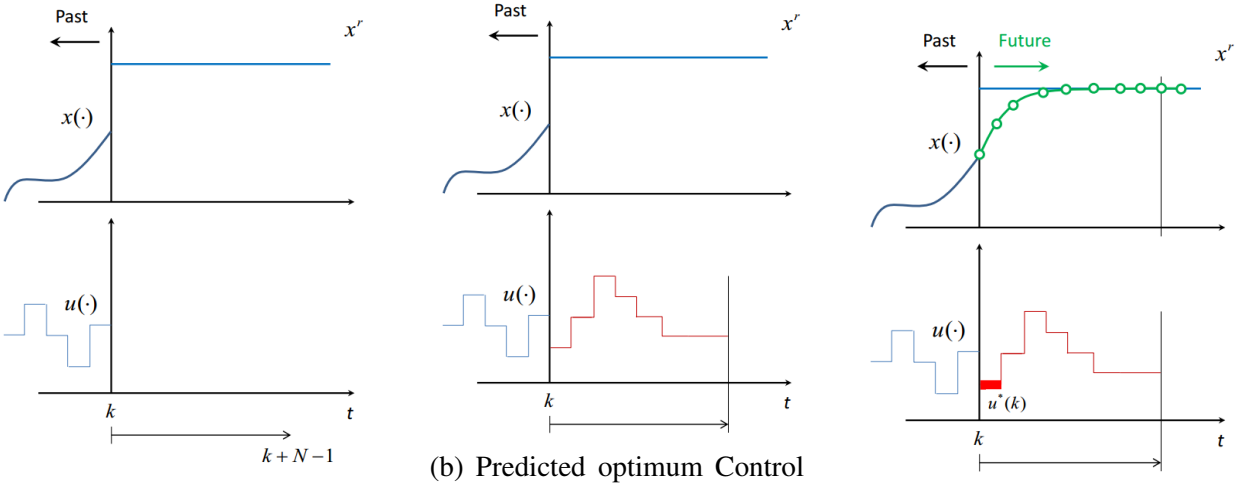
MPC can handle multi-input multi-output(MIMO) systems that might have interactions between their inputs and outputs[19]. Hence the advantage of MPC is: it is multivariable controller

that controls the output simultaneously by taking into account all the interactions between system variables. Another advantage of the controller is that it can handle both states and control constraints where constraints are important in control system because violating them can lead to undesired consequences. In addition, MPC has a preview capability which means it can incorporate future reference information into the control problem to improve controller performance.

MPC is approximately optimal control and requires online optimization at each time steps. Therefore, it requires a powerful, fast processor with a large memory.

4.1.1 MPC Working Principle

Considering a single input single output (SISO) system $(X(k+1) = f(X(k), U(k)))$, MPC working principle is explained using 2 graphs: The Control action (U) and the State (X). In fig (4.2a), U was the control action and X is the evolved state in the past (before k). Now the system is in k and it is desired to take a new decision or new action so that the state X eventually reaches some reference (x^r), therefore the MPC looks in to the future N time steps and tries to find the best or optimal sequence of control actions (fig 4.2b), that can be applied to the system for the states to approach or actually reach the state reference (fig 4.2c). In conclusion, the decision variables (after k) optimization problem will be solved to get an optimal sequence of control actions to have the prediction of states going to the reference.



(a) U and X in the past

(b) Predicted optimum Control action

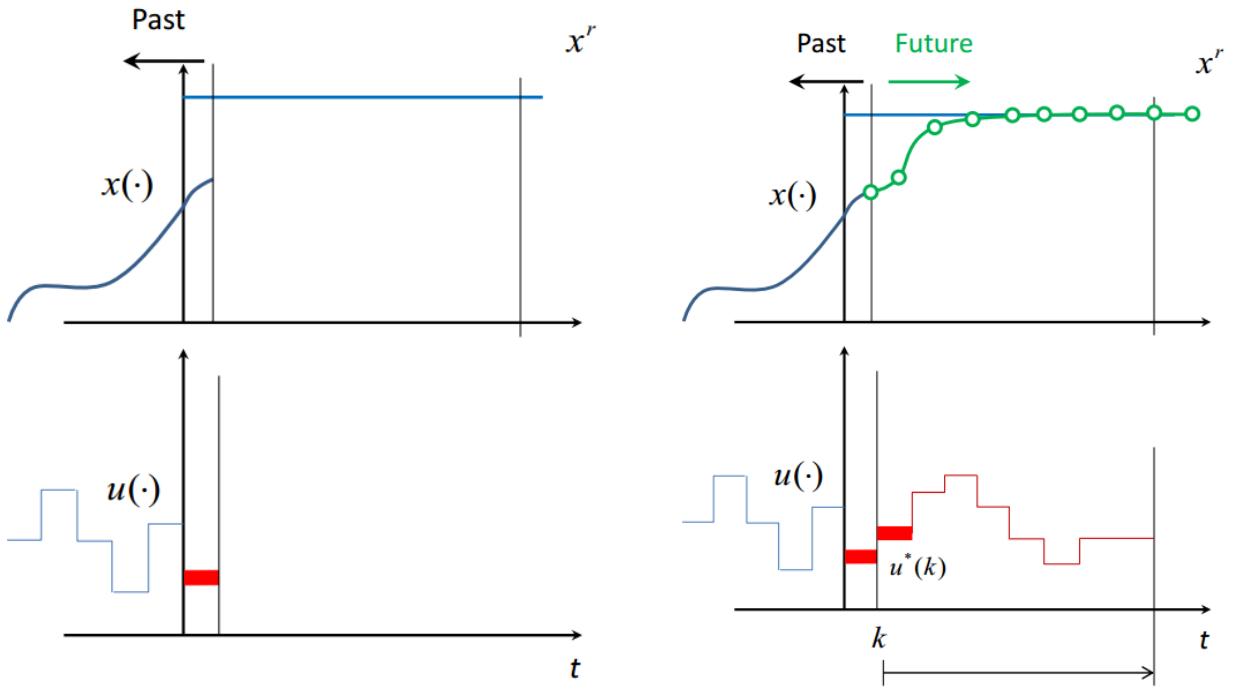
(c) Predicted State

Figure 4.2: MPC Prediction Procedure

After getting the optimum control sequence and the predicted states, MPC applies only the first portion or the 1^{st} step of the sequence of control to the system and disregards the rest (fig 4.3a). Based on the input control applied to the plant there will be some changes in the plant output. The plant actual output might be different than what the MPC predicted before, It could be because of

many reasons like unmeasured disturbance, and that is why after applying the first control action there is another step of MPC (fig 4.3b) and a new control sequence will be calculated and the first step control action will be applied again. The MPC does the same thing over and over so it gets closer to the reference but it is not done randomly but in a systematic way and this is where the optimizer comes into the picture. The purpose of taking new measurements at each time step is to compensate for unmeasured disturbances and model inaccuracy, in both cases the system output is caused to be different from the one predicted by the model.
In MPC there are 3 main things done (MPC Strategy Summary):

- Prediction: to know how the state acts or evolves
- online optimization by minimizing the difference between the prediction and the reference state via manipulating the control actions
- Receding horizon implementation: it is done every sampling time or every time step.



(a) Actual State (b) Next Step Prediction

Figure 4.3: MPC next step prediction

4.1.2 MPC Design Parameters

The MPC parameters not only affect the controller performance but also the computational complexity of the MPC algorithm that solves an online optimization problem at each time step, therefore choosing proper values for this parameters is important.
MPC can be mathematically formulated as follows:

$$\begin{aligned}
\min_{U,X} \quad & J(X,U;N,M) = \sum_{k=0}^{N}(X_{pred}-X_{ref})^T * Q * (X_{pred}-X_{ref}) + (U(k)^T * R * U(k))), \text{ where N and M} \\
& \text{are parameteres.} \\
\text{s.t.} \quad & X_U(k+1) = f(X_U(k), U(k)) \\
& X_U(k) \in X, \forall \in [0,N] \\
& U(k) \in U, \forall \in [0,M]
\end{aligned} \tag{4.1}$$

Sample time(T_s)

T_s determines the rate time t at which the controller execute the control algorithm. If it is to big the controller won't be able to react to the disturbance fast enough. And if it is small the controller can react much faster to disturbance but causes an excessive computational load. The aim is to find the right balance between performance and computational effort

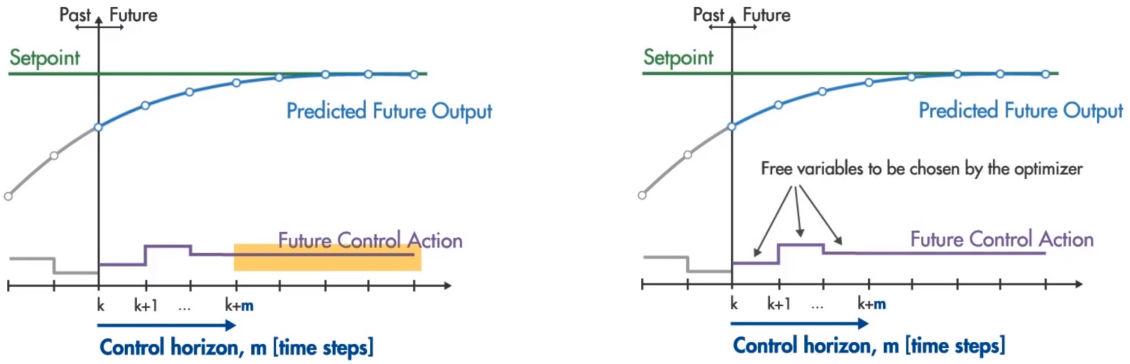
Prediction Horizon(N)

Prediction horizon is a measure of how far ahead MPC looks into the future. It is often represented by the length of time into the future or the number of future time steps. Due to the forward moving nature of the prediction horizon (from fig 4.2 to fig 4.3), MPC is also referred to as **Receding or Moving horizon control**. Increasing N results in less aggressive control action. If prediction horizon is too small: necessary futures might not be seen, if prediction horizon is too large: if disturbance occurs then it might not be able to detect it. Prediction horizon should be chosen so that it will cover the significant dynamics of the system.

Control Horizon(M)

The number of control moves to time steps M are called Control horizon, The rest of the input is constant (fig 4.4). Each control moves in the control horizon can be thought of as a free variable that needs to be computed by the optimizer (fig 4.4b). The smaller the control horizon, the fewer the computations. Although Control horizon of 1 can be chosen, it might not give the best possible

maneuver. Increasing M makes the controller more aggressive and increase computational effort but better predictions can also be found. Good choice for M is to make it 10 to 20% of the prediction horizon (N),$0.1 * N \leq M \leq 0.2 * N$.



(a) Control Horizon with constant value after M steps (b) Control moves to be computed by the optimizer

Figure 4.4: Control Horizon

Weight(Q and R)

It is desired for the output to track as close as possible to their set points but at the same time, it is also preferred to have smooth control moves (equation 4.1). **Q**: this term implicitly measures convergence rate (rise time and settling time). **R**: this term penalizes aggressive use of the input. If $Q >> R$, it is more a concern to decrease the difference between the desired and the predicted value than penalizing the input and if $R >> Q$, means it is more preferred to decrease the control effort as much as possible. To get Q and R PSO (Particle Swarm Optimization) is used.

- **PSO (Particle Swarm Optimization)** is an intelligent optimization algorithm. It is a simple yet powerful optimization algorithm which is based on the paradigm of swarm intelligence and it is inspired by social behavior of animals like fish and birds[20].
  The main components of PSO are maximum number of iteration, particle velocity, particle position, swarm population size. The structure of the MPC with PSO algorithm is shown in Fig.4.5. The PSO model contains a swarm of particles, which are first initialized with a population of random candidate solutions. To search for new solutions, the swarm of particles move iteratively through the specified range. Each particle has a position vector X_k^i, and a velocity vector V_k^i. The particle remembers its own best position P_{best}^i and the global best among the swarm position vectors is stored in P_{global}^i. In all this vectors i represents the index of the particle and k is the number of iteration.
  During the iteration time k, to get the new velocity from the previous velocity and position, it uses:

$$
\begin{aligned}
Vi_{k+1} \quad &= wV_k^i + c_1R_1(P_{best}^i - X_k^i) + c_2R_2(P_{global}^i - X_k^i) \\
Where: &c_1 \text{ is Personal acceleration coefficient} \\
&c_2 \text{ is Social acceleration coefficient} \\
&w \text{ is Inertia coefficient} \\
&R_1 \text{ and } R_2 \text{ are random numbers}
\end{aligned} \tag{4.2}
$$

The sum of the new velocity and the previous position gives the current position:

$$X_{k+1}^i = X_k^i + V_{k+1}^i \tag{4.3}$$

At the first iteration the position of a particle was random then A particle decides where to move next by considering its own experience, which is its best past position memory, and the experience of the most successful particle in the swarm.

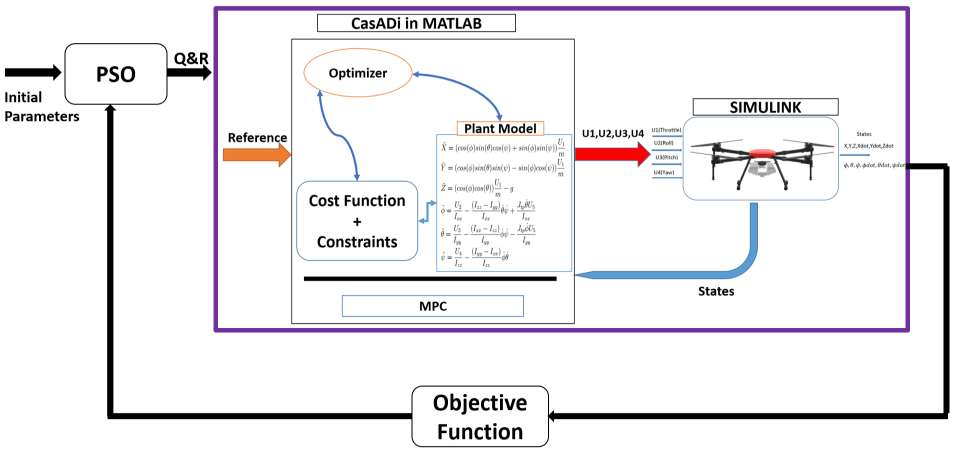


Figure 4.5: Structure of the MPC with PSO algorithm

4.2 Introduction to NMPC Mathematical Formulation

Mathematical formulation of MPC (equation 4.1) is very similar to LQR (Linear Quadratic Regulator)

1. The dynamic model should be discretized, since prediction is used in MPC. The model should be used to do predictions for the state. The method used for discretization is Rung-kuta 4^{th} order method.
   Details about Rung-kuta 4^{th} order method is shown in Appendix E

2. The **Running(Stage) Cost** will be defined, which characterizes the control objective. $l(X,U) = (X_{pred} - X_{ref})^T * Q * (X_{pred} - X_{ref}) + (U(k)^T * R * U(k))$. It penalizes the difference between the predicted state and the reference state and similarly do so the control action.

3. The **Cost function**: which is the summation of the running cost over the prediction horizon.

$$J_N(X,U;N,M) = \sum_{k=0}^{N} l(X_U(k),U(k)), \text{ where N and M are parameteres.} \tag{4.4}$$

4. The MPC problem formulation can be formulated as an optimal control problem , where it is preferred to minimize the cost function by manipulating the optimization variables.
   The cost function minimization is subjected to: The relation between the prediction of X and the control action U which is our system model $(X_U(k+1) = f(X_U(k),U(k)))$, the initial value of the state for the prediction $(X_U(t_0) = X_0)$, control action constraint which it should be respected over all the prediction steps $(U(k) \in U, \forall \in [0,M])$ and state constraint which it should be respected over all the prediction steps $X_U(k) \in X, \forall \in [0,N]$.

$$\begin{aligned} \min_{U,X;N,M} \quad & J_N(X,U) = \sum_{k=0}^{N} l(X_U(k),U(k)) \\ \text{s.t.} \quad & X_U(k+1) = f(X_U(k),U(k)) \\ & X_U(t_0) = X_0 \\ & U(k) \in U, \forall \in [0,M] \\ & X_U(k) \in X, \forall \in [0,N] \end{aligned} \tag{4.5}$$

5. In order to solve the optimal control problem (OCP) numerically, it needs to be transformed to **Nonlinear Programming Problem(NLP)**. NLP is a standard problem formulation in

numerical optimization having the general form:

$$\begin{aligned} \min_{w} \quad & \Phi(w), \text{ Objective function} \\ s.t: \quad & g_1(w) \leq 0, \text{ Inequality constraint} \\ & g_2(w) = 0, \text{ Equality constraint} \end{aligned} \tag{4.6}$$

$\therefore$: NLP consists of the following three ingredients:

- An objective function, $\Phi(w)$, that shall be minimized or maximized
- decision variable, w ,that can be chosen and
- constraints that shall be respected, e.g. of the form $g_1(w) = 0$ (equality constraints) or $g_2(w) \geq 0$ (inequality constraints).

Special cases of NLP include:

- **Linear programming(LP)**: when $\Phi(w)$, $g_1(w)$ and $g_2(w)$ are affine, which means these functions can be expressed as linear combination of the elements of w.
- **Quadratic programming(QP)**: when $g_1(w)$ and $g_2(w)$ are affine, but the objective function $\Phi(w)$ is a linear quadratic function.

6. Methods to express an OCP as an NLP. There exists a variety of methods to numerically solve continuous time OCP. What all approaches have in common is that at one point, the infinite-dimensional problem needs to be discretized.

   - The **Indirect Methods** is based on first derive optimality conditions in continuous time by algebraic manipulations and only discretize the resulting continuous time at the very end of the procedure. One characterizes the indirect methods often as "first optimize, then discretize". It can solve simple problems.
   - The **Direct Methods**: first discretizes the continuous time OCP, to convert it into a finite-dimensional optimization problem. The finite-dimensional optimization problem can then be solved by tailored algorithms from the field of numerical optimization. The direct methods are often characterized as "first discretize, then optimize". These methods are most widely used in MPC applications. Next some of the most widely used direct methods are presented:

     (a) **Direct Single Shooting**

     The problem decision variable is just the input: $w = [U_0, U_1 ... U_{N-1}]$. The single-shooting method is a fully sequential approach that treats all intermediate state values computed in the numerical integration routine as hidden variables, and solves

the optimization problem in the space of control parameters and initial values only. The main drawback of Direct Single Shooting is the nonlinearity propagation: integrator function tends to become highly nonlinear for large prediction horizon(N). It is not suitable method for nonlinear and/or unstable systems when optimizing over a long prediction horizon.

(b) **Direct Multiple Shooting**
Direct Multiple Shooting is another way of expressing the optimal control problem as a NLP. In multiple shooting the states (X) are considered as a decision variable in the optimization problem as well as the input control (U). The key idea is to break down the system integration into short term intervals, i.e. use the system model as a state constraint at each optimization step. Applying the equality constraint (difference between predicted state and the model output) at every prediction step or every shooting step, that is why it is called multiple shooting.

7. Solve the NLP:
To solve the NLP, CasADi is used as a facilitator.
CasADi: is a free and open source tool which has a general scope of Numerical optimization. It offers a flexible approach to optimal control, providing building blocks that simplifies the process of implementing efficient algorithms for optimal control using methods such as direct multiple shooting and direct single shooting. CasADi is available for MATLAB, Python and C++ with little or no difference in performance[21].
4 standard problems can be handled by CasADi

- QP's (Quadratic programs)
- NLP's (Nonlinear programs)
- Root finding problems
- Initial-value problems in ODE (Ordinary differential equations)

There are several NLP solvers interfaced with CasADi. The most popular one is IPOPT (Interior Point Optimizer), an open-source primal-dual interior point method which is included in CasADi installations.

4.3 NMPC Mathematical Formulation for Quad-copter

1. Since MPC is a model based controller, the system model is in MATLAB where the MPC exists and the output of MPC which is the control effort is applied in SIMULINK where the quad-copter model resides (fig:4.6).

2. Dynamics of a Quad-copter

$$\begin{aligned}
\ddot{X} &= (cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi))\frac{U_1}{M} \\
\ddot{Y} &= (cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi))\frac{U_1}{M} \\
\ddot{Z} &= (cos(\phi)cos(\theta))\frac{U_1}{M} - g \\
\ddot{\phi} &= \frac{U_2}{I_{xx}} - \frac{(I_{zz} - I_{yy})}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{J_{tp}\dot{\theta}U_5}{I_{xx}} \\
\ddot{\theta} &= \frac{U_3}{I_{yy}} - \frac{(I_{xx} - I_{zz})}{I_{yy}}\dot{\phi}\dot{\psi} - \frac{J_{tp}\dot{\phi}U_5}{I_{yy}} \\
\ddot{\psi} &= \frac{U_4}{I_{zz}} - \frac{(I_{yy} - I_{xx})}{I_{zz}}\dot{\phi}\dot{\theta}
\end{aligned} \tag{4.7}$$

   The system has 12 states $[X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ and 4 inputs $[U_1, U_2, U_3, U_4]$

3. Control objective:

   - Point Stabilization: the reference values of the state vector are constant over the control period.

$$\begin{bmatrix} X_r(t) \\ Y_r(t) \\ Z_r(t) \end{bmatrix} = \begin{bmatrix} X_d \\ Y_d \\ Z_d \end{bmatrix} \tag{4.8}$$

   - Trajectory tracking: the reference values of the state vector are time varying over the control period..

$$\begin{bmatrix} X_r(t) \\ Y_r(t) \\ Z_r(t) \end{bmatrix} = \begin{bmatrix} X_d(t) \\ Y_d(t) \\ Z_d(t) \end{bmatrix} \tag{4.9}$$

   Both control objectives are implemented later.

4. Discretize the dynamic system using 4^{th} order Runge-Kuttas method ,which is shown in Appendix E

5. Running Cost:

   - For the error penalization: the predicted states are stored in $States_{pred}$, the reference values are stored in parameter P_{euler} and Q is the weighting matrix.
   - For control effort penalization: the predicted input stored in $input$ and R is the weighting matrix.
   - For control effort change penalization: the predicted input change stored in $input_{change}$ and R_{change} is the weighting matrix.

$$\begin{aligned} obj = \quad & ((States_{pred} - P_{euler})' * Q * (States_{pred} - P_{euler}))+ \\ & (input' * R * input) + (input'_{change} * R_{change} * input_{change})) \end{aligned} \tag{4.10}$$

6. The **Cost function**:
   The summation of the running cost over the prediction horizon (N)

$$\begin{aligned} obj \quad = obj + & ((States_{pred} - P_{euler})' * Q * (States_{pred} - P_{euler}))+ \\ & (input' * R * input) + (input'_{change} * R_{change} * input_{change})) \end{aligned} \tag{4.11}$$

7. The MPC problem formulation formulated as an optimal control problem:

$$\begin{aligned} \min_{U,X} \quad & obj = obj + ((States_{pred} - P_{euler})' * Q * (States_{pred} - P_{euler}))+ \\ & (input' * R * input) + (input'_{change} * R_{change} * input_{change})) \\ \text{s.t.} \quad & X_U(k+1) = f(X_U(k), U(k)) \\ & X_U(t_0) = X_0 \\ & U(k) \in U, \forall \in [0, M] \\ & X_U(k) \in X, \forall \in [0, N] \end{aligned} \tag{4.12}$$

8. **Multiple Shooting**: used to transform the OCP to NLP.

$$\begin{aligned} \min_{w} \quad & \Phi(w), \text{ Objective function} \\ s.t: \quad & g_1(w) \leq 0, \text{ Equality constraint} \\ & g_2(w) = 0, \text{ Equality constraint} \end{aligned} \tag{4.13}$$

   - The decision variables are: the 12 States $[X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ and 4 inputs $[U_1, U_2, U_3, U_4]$
   $w = [U_0, ..., U_M, X_0....., X_N]$

- The Multiple shooting applies additional path constraint at each optimization step (shooting step). $X(k+1) - f(X(k), U(k)) = 0$, this difference is basically the model.
- Inequality Constraint (Control Inputs, map margins):
  The use of Euler angles has a disadvantage of singularity which occurs by changing the coordinates when rotating the body-fixed frame of reference. In Z-Y-X convention, singularity occurs in pitching motion of the quad-copter with angle θ equal to $\frac{\pi}{2}$. To over come this problem an Inequality constraint will include: $\left[-\frac{\pi}{2} < \theta < \frac{\pi}{2} \right]$. Given the nature of the quad-copter, there is also an inequality constraint for yaw and roll movement: $\left[-\frac{\pi}{2} < \phi < \frac{\pi}{2} \right]$ and $\left[-\pi < \psi < \pi \right]$.
  For Z there is an inequality constraint of: $[0 < Z < 100$, since the minimum position of the quad-copter in Z-direction is 0 and the maximum where the quad-copter is able to go is 100m.
  This inequality constraints have to be respected over all the prediction steps.
- Equality Constraint (System dynamics)

$$g_2(w) = \begin{bmatrix} \overline{X_0} - X_0 \\ f(X_0, U_0) - X_1 \\ \vdots \\ f(X_{N-1}, U_{N-1}) - X_N \end{bmatrix}$$

  Where $\overline{X_0}$ is the optimization variable which comes from the model and X_0 is the predicted state.

9. Solve the NLP using IPOPT solver in CasADi: CasAdi facilitates the solution of NLP's which means it does not actually solve the NLP. The solver used is IPOPT, which is an open source software package for large scale non-linear optimization. It can be used to solve general NLP's. NLP solvers are created using CasADi's "nlpsol" function. Different solvers and interfaces are implemented as plugins.

   - To solve the NLP using Ipopt, first the decision variables (the states and the input) have to be in a column matrix. and the states are $12 * N$ because there are N prediction horizons and for input $4 * M$ because there are M control horizons.
     $OPT\_variables\_euler = [reshape(X\_pre\_euler, 12 * (N), 1); reshape(U\_euler, 4 * M, 1)]$
   - Then a structure is created with Objective function, Decision variable, Constraints and Parameters.

$$NLP\_prob\_euler = struct('f', obj\_euler, 'x', OPT\_variables\_euler, 'g', g\_euler, 'p', P\_euler)$$

- Afterwards some options (opts) are set such as the maximum iteration. acceptable tolerance and so on.
- Now the solver can be called:
  $solver\_euler = nlpsol('solver', 'ipopt', NLP\_prob\_euler, opts)$

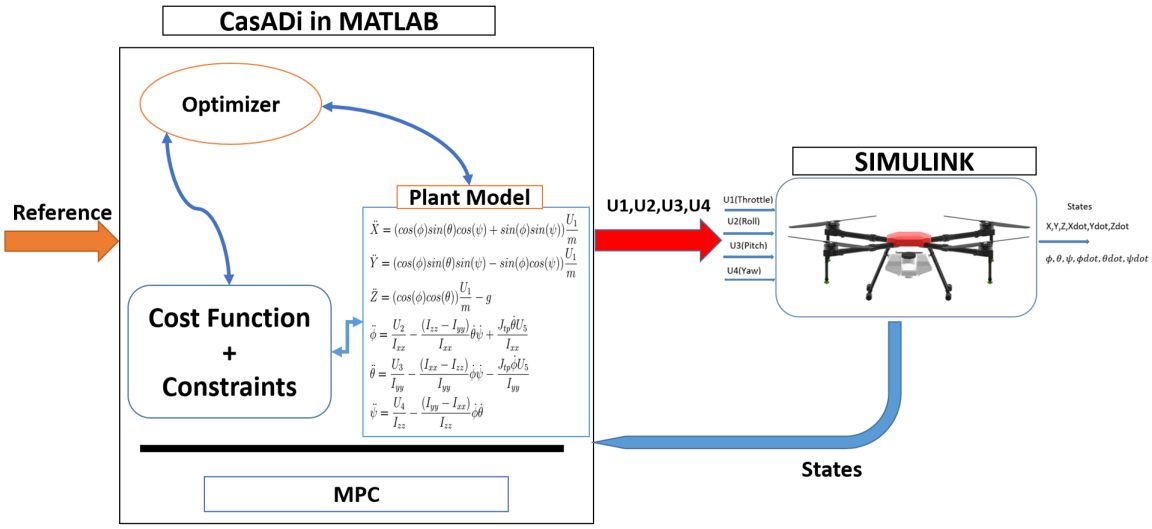


Figure 4.6: Implementation of MPC in Quad-copter

Chapter 5

Mass Variation and Image Recognition of Locust Swarm

This chapter provides the method used to consider the Mass Variation, which is the Recursive Least Square Estimation and the procedure taken for Image Recognition of Locust swarm.

5.1 Mass Variation

The Mass Variation of the Quad-copter is caused by the spraying system. When the quad-copter starts to spray the pesticide, mass of the quad-copter decreases.

- Full tank Mass of the Quad-copter = 20.7kg
- Empty tank Mass of the Quad-copter = 10.5kg

5.1.1 Parameter Estimation

Parameter Estimation provides a means of estimating the parameters directly from data. It is necessary because some of the parameters of the model can have unknown or uncertain values. Before Parameter Estimation, some details should be known about the model being used:

- Model selection: Linear or Non-linear model
- Model order
- Data measurements: Digital or Analogue
- Methodologies:

i Simple least squares

ii Generalized least squares

iii Least squares

iv Recursive Least Squares

5.1.2 Recursive Least Square Estimation(RLSE)

For estimating the mass of the quad-copter the RLSE is used. The RLSE estimates the parameters of a system using a model that is linear in those parameters ($Y = \Phi\beta$). It considers the effect of additional observation data, makes it useful for online parameter estimation(mass).
Note:
$Y = \Phi\beta$,
Where: β is Parameter, Φ is Input and Y is Output

$$\begin{bmatrix} Y_N \\ \vdots \\ Y_{N+1} \end{bmatrix} = \begin{bmatrix} \Phi_N \\ \vdots \\ \Phi_{N+1} \end{bmatrix} \beta \tag{5.1}$$

Where: Y_{N+1} is the new data found and the over all new data will be:

$$X_{N+1}^T = \begin{bmatrix} -Y(N) & \dots & -Y(N+1-n) & U(N) & \dots & U(N+1-n) \end{bmatrix} \tag{5.2}$$

Simple least square parameter estimation has a form of:

$$\beta = [\Phi^T\Phi]^{-1}\Phi^T Y \tag{5.3}$$

With the effect of additional new data:

$$\beta_{N+1} = \left(\begin{bmatrix} \Phi_N^T & X_{N+1} \end{bmatrix} \begin{bmatrix} \Phi_N \\ \\ X_{N+1}^T \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} \Phi_N^T & X_{N+1} \end{bmatrix} \begin{bmatrix} Y_N \\ \\ Y_{N+1} \end{bmatrix} \right) \tag{5.4}$$

In Equation(5.4), X_{N+1} is the additional data. The Equation can be expanded as:
$\beta_{N+1} = (\Phi_N^T\Phi_N + X_{N+1}X_{N+1}^T)^{-1}(\Phi_N^T Y_N + X_{N+1}Y_{N+1})$

It can be simplified by introducing new variables:

$$P_N^{-1} = \Phi_N^T\Phi_N \tag{5.5}$$

$$K_{N+1} = \frac{P_N X_{N+1}}{\lambda + X_{N+1}^T P_N X_{N+1}} \tag{5.6}$$

$$P_{N+1} = \frac{P_N}{\lambda}\left(I - \frac{X_{N+1}X_{N+1}^T P_N}{\lambda + X_{N+1}^T P_N X_{N+1}}\right) \tag{5.7}$$

$$\boldsymbol{\therefore \beta_{N+1} = \beta_N + K_{N+1}(Y_{N+1} - X_{N+1}^T \beta_N)} \tag{5.8}$$

Where: β_{N+1} is the new estimation, β_N is the old estimation, K_{N+1} is gain, $(Y_{N+1} - X_{N+1}^T \beta_N)$ is the prediction error (Y_{N+1} is the real data and $X_{N+1}^T \beta_N$ is the predicted out put). The gain and the prediction are called correction term together.

If the prediction error is zero, it means there is no mass change: $\beta_N = \beta_{N+1}$. But if there is a change in a system then there will be a prediction error and is multiplied by some gain (K_{N+1}), then added to the previous estimation, So now the new estimation is different from the previous estimation.

The main task of RLSE is to detect a change in the system so that the model can be improved. And this can be done if data is collected from the system at some sampling time.

5.1.3 RLSE computation in Matlab

I The non-linear system has to be linearly related with the parameter, which is mass.
Considering $\ddot{Z}$:
$\ddot{Z} = (cos(\phi)cos(\theta))\frac{U_1}{M} - g$
$\ddot{Z} + g = (cos(\phi)cos(\theta))U_1\beta$
$\therefore Y = \ddot{Z} + g$, $\Phi = (cos(\phi)cos(\theta))U_1$ and $\beta = \frac{1}{M}$

II To start the algorithm the initial value of the parameter($\beta(0)$) and $P(0)$ have to be declared. $P(0) = \alpha I$, α it indicates how confident the system is in the initial value of β. So if $\beta = 0$, and $\alpha = (10)^6$ then it shows that there is very less assurance in the initial value of β.

III Forgetting Factor(λ): It tells how much the new parameter estimation actually forgets the previous parameter estimation.

- if $\lambda = 0$, then the new parameter estimation is completely forgetting the previous parameter estimation, means it is going to be a new data on the estimation processes.
- if $\lambda = 1$, then the new parameter estimation remembers the previous parameter estimation value, therefore there is no that much difference in the new parameter estimation.
- The Forgetting Factor (λ) can't be exactly 1 or 0.
- The exact value of λ is found using trial and error between 0 and 1.

IV RLSE pseudocode:

Algorithm 1 RLSE Pseudocode

Numebr of parameter $\leftarrow 1$
Forgetting factor$(\lambda) \leftarrow 5 * 10^{-5}$
$Max \leftarrow$ maximum MPC iteration
if MPC iteration ≤ 1 **then**
    $\beta_n \leftarrow zeros(Numebrofparameter, Max)$
    $Y_{pred} \leftarrow [\,]$
    $X_N \leftarrow zeros(Numebrofparameter, 3)$
    $P_N \leftarrow 10^5 * I$ ▷ I is 3x3 identity matrix 3 for [X,Y,Z]
end if
$X_N \leftarrow U_{data}$ ▷ U_{data} is the input data

$$K_{N+1} \leftarrow \frac{P_N X_N}{\lambda + X_N^T P_N X_N}$$

$$P_{N+1} \leftarrow \frac{P_N}{\lambda}(I - X_{N+1}^T K_{N+1})$$

$P_N \leftarrow P_{N+1}$
$Y_{pred} \leftarrow X_N * \beta_N$
$error \leftarrow Y_{data} - Y_{pred}$ ▷ Y_{data} is the output data collected
$\beta_{N+1} \leftarrow \beta_N + K_{N+1} * error$
$\beta_N \leftarrow \beta_{N+1}$

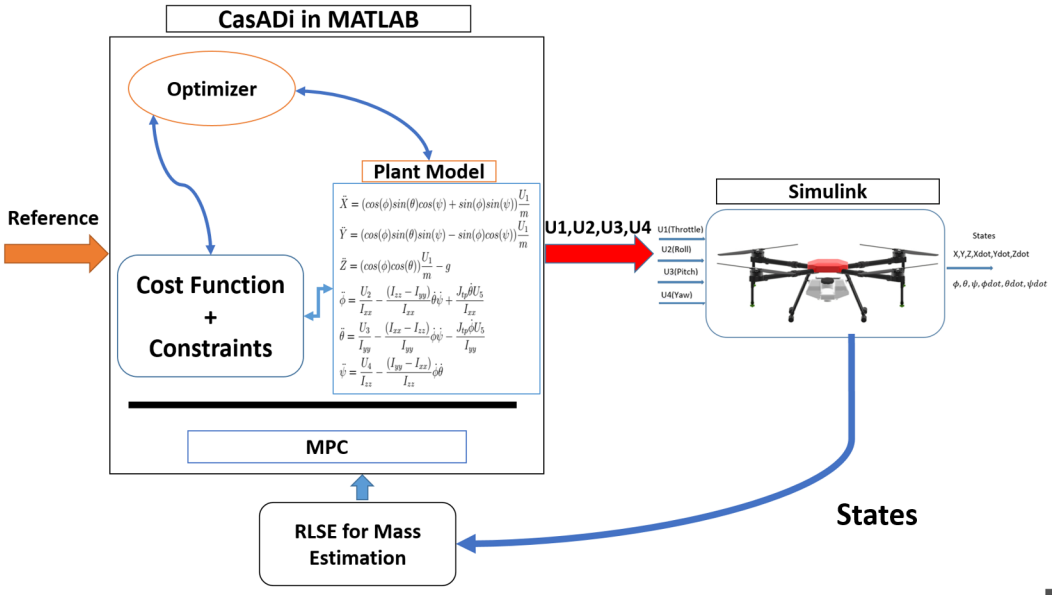


Figure 5.1: The System Block Diagram with RLSE

5.2 Image Recognition of Locust Swarm

For image recognition of Locust, 5 categories where created. The categories are: Bees, Butterfly, Crop, Ladybug and Locust. This categories are chosen, because this are some of the main insects which are beneficial for farms and it is necessary to protect them from pesticide. The pesticide should be sprayed when there is an existence of just locust. To do the image category the main steps taken are listed down below:

5.2.1 Data Collection

The first step for image recognition procedure is to collect data, which was the most difficult aspect of this thesis. The images where collected using different image dataset sources like:

- **Bing image downloader**(fig:5.2): it is a Python library, which is used to download bulk of images from Bing.com. To use this library, first it has to be installed using pip install bing-image-downloader. The drawback of Bing image downloader is there will be a duplicated images.

```
from bing_image_downloader import downloader
downloader.download("locust swarm", limit = 200, output_dir='Locust_images')
```

Figure 5.2: Bing image downloader

- **Bulk image downloader(BID)**: is a windows only app that must be installed separately with Chrome. It is not a freeware application but it is possible to use the trial version although it has some limitations like, it can only download 100 images at a time. It can be added as extension in Chrome, which allows to quickly open the current web page with BID[22].

- **kaggle**: is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets[23].

The main drawback of the stated sources is, there is duplication of images. And to remove the duplicated images an app called **Photo Forensics free** was used. It doesn't automatically delete the duplicated images but points out the images.

After the image dataset was collected it has to be divide into 3 data sets:

- **Training dataset**: contains $80\% \rightarrow 90\%$ of the total collected data. This data is used to train the machine about the different categories of images stated.

- **Validation dataset**: contains $5\% \rightarrow 20\%$ of the total collected data. This tests how well the machine performs against known labeled data.
- **Test dataset**: contains the rest of the data in unlabeled format. It is used to test how well the machine can classify data, it has never seen.

The number of datasets in each category should be balanced or equal not to make the decision a biased one.

The data structure is as follows:

- Collected data: 2005
  - Training dataset: 1605
    - Bees: 321
    - Butterfly: 321
    - Crop: 321
    - Ladybug: 321
    - Locust: 321
  - Validation dataset: 400
    - Bees: 80
    - Butterfly: 80
    - Crop: 80
    - Ladybug: 80
    - Locust: 80

5.2.2 Image Pre-processing

The second procedure is to prepare the collected dataset for training. Images come in different sizes and shapes. They also come through different sources. Taking all these variations into consideration, Pre-processing of any image data is necessary. It includes:

- Assigning paths or directories of the training and validation dataset
- Creating labels or categories:$[' Bee' : 0,' Butterfly' : 1,' Crop' : 2,' Ladybird' : 3,' Locust' : 4]$
- Open-cv reads image data as array(img_array = cv2.imread(img_path))
- Making the images data the same size: 200×200
- Normalization: is the most crucial step in the pre-processing part. This refers to rescaling the pixel values so that they lie within a confined range. Images are stored in the form of a matrix of numbers in a computer where these numbers are known as pixel values. These pixel values represent the intensity of each pixel. 0 represents black and 255 represents white. Therefore image dataset ranges between $0 \rightarrow 255$ so using normalization, it can be changed to range between $0 \rightarrow 1$. $X = \frac{X}{255}$, where X is the pixel value. Main purpose of normalization is to make computation efficient by reducing values between 0 to 1.
- Shuffle the data (random.shuffle(data)): During training the images order should be random to get a good accuracy because if not random after training a single category and the machine learns to predict that category only and then if there is a change to a new directory there will be mix up if it goes back and forth so it should be random.
- Creating two variables to store the image feature, which is the array of the image data, and the label of that image data.
- Not to do the whole pre-processing, the data can be saved using Pickle, So the feature data and the label is saved.

5.2.3 Convolutional Neural Network(CNN)

Convolutional Neural Network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. Image classification involves the extraction of main features from the image data to detect some pattern in the dataset. Convolution basically means a point wise multiplication of two functions to produce a third function. The first function is the image pixel matrix and then perform convolution operation

with the filter, the filter scans through the image input and produce the feature map or activation map. There are multiple convolutional layers extracting features from the image input and produce an output layer which puts it in the artificial neural network[24]. The types of layers in which the CNN is composed of are:

i **Convolution layer:** takes one feature map and it scans each pixel and put it in a separate matrix. Basically, It filters for main features (Eg:legs of locust, butterfly wings...) in an image data and at the same time resizes the image.

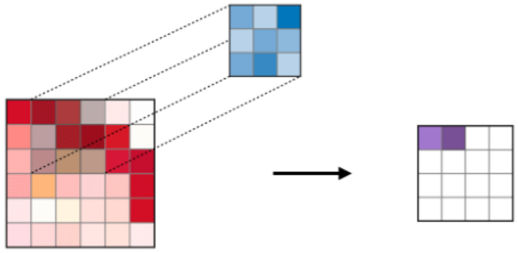


Figure 5.3: Convolution layer

ii **Pooling layer:** reduces or filters the image to the essential information. For Maximum pooling, each pooling operation selects the maximum value of the current view. And Average pooling, each pooling operation averages the values of the current view.

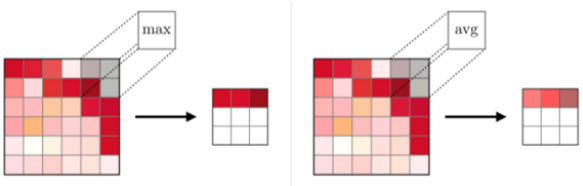


Figure 5.4: Pooling layer

iii **Flatten:** is to convert the matrix (MxM) into a single column matrix (Mx1), usually found towards the end of the CNN architecture.

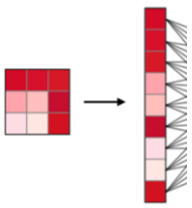


Figure 5.5: Flattening layer

After flattening, the matrix gained acts as neuron input to the Artificial Neural Network (ANN). ANN has three layers: Input layer, hidden layer and output. The flattened layer acts as an input layer and goes to the hidden layer where it performs some computation. Hidden layer using some activation function, it will train the dataset.

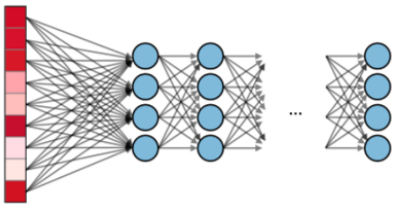


Figure 5.6: Artificial Neural Network(ANN)

5.2.4 Model Test

After the training is done and found the best model, it is good to save the model so that the machine doesn't have to be trained every time, during the testing procedure. To test the machine, the image used should not be included both in the training set and validation set. The test image should pass through the same image pre-processing steps.

Chapter 6

Simulation Result

6.1 Controller based Result

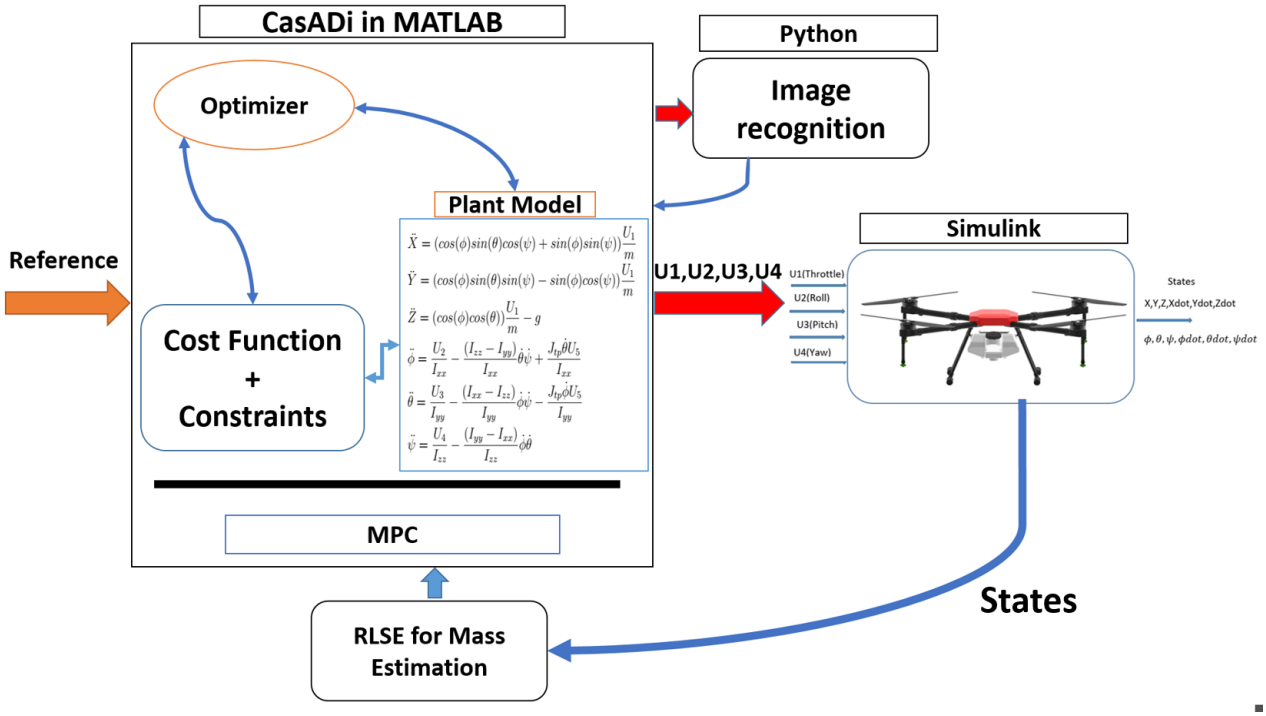


Figure 6.1: The Whole System Block Diagram

Parameter (Unit)	**Symbol**	**Value**
Mass of quad-copter with full tank (kg)	M_{full}	20.7
Mass of the-copter with empty tank (kg)	M_{empty}	10.5
Drag coefficient(Nms^2)	k_m	$1.1 * 10^{-6}$
Trust coefficient(Ns^2)	k_f	$54.2 * 10^{-6}$
Inertia around X-axis (Nms^2)	I_{xx}	1.9579
Inertia around Y-axis (Nms^2)	I_{yy}	1.9570
Inertia around Z-axis (Nms^2)	I_{zz}	0.9318
Gravitational acceleration (ms^{-2})	g	9.81
Distance from center of quad-copter to center of propeller distance (m)	d	0.5205
Inertia around propeller(Nms^2)	Jtp	0.000104
Prediction horizon	N	10
Control horizon	M	3

Table 6.1: Quad-copter and MPC Parameters

Parameter	**Symbol**	**Value**
Maximum Iteration	Max_{Iter}	10
Number of unknown population	nPop	13
Inertia coefficient	w	0.7298
Damping ratio of inertia coefficient	wdamp	1
Personal acceleration coefficient	c1	1.4962
Social acceleration coefficient	c2	1.4962

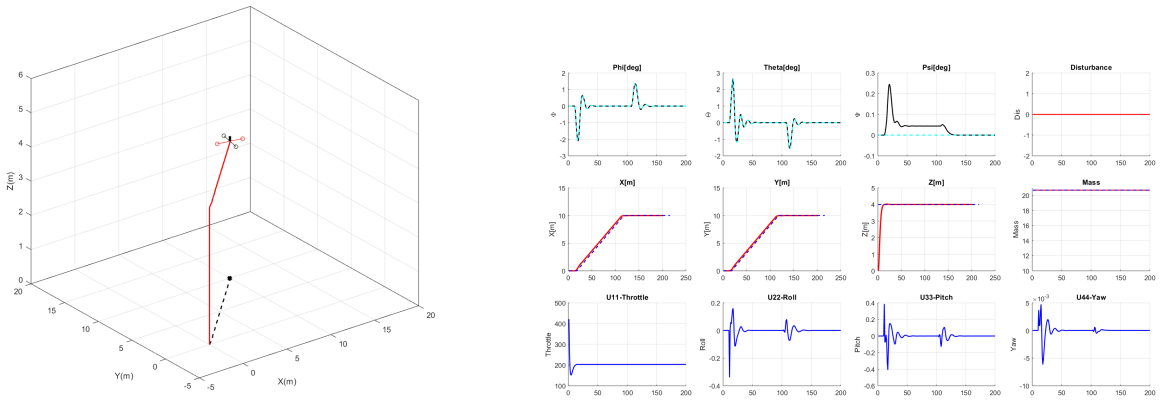
Table 6.2: PSO Parameters

6.1.1 Point Stabilization

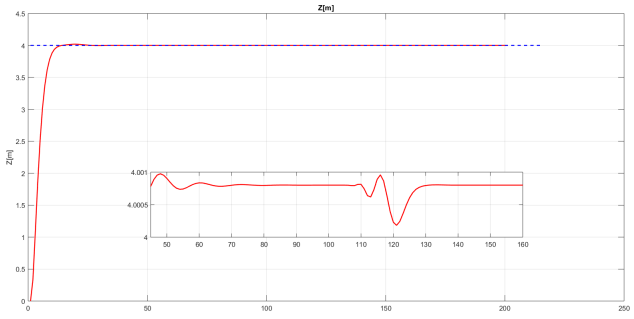
∗ For X-Y-Z control going to point (10,10,4)

Q_X	Q_Y	Q_Z	Q_ϕ	Q_θ	Q_ψ	$Q_{\dot{X}}$	$Q_{\dot{Y}}$	$Q_{\dot{Z}}$	$R_{Throttle}$	R_{Roll}	R_{Pitch}	R_{Yaw}
306.5739	356.6697	240	0.1681	0.1394	300	100	100	100	0.5	8.4233	0.8	90

A) For X-Y-Z control without Disturbance and No Mass variation

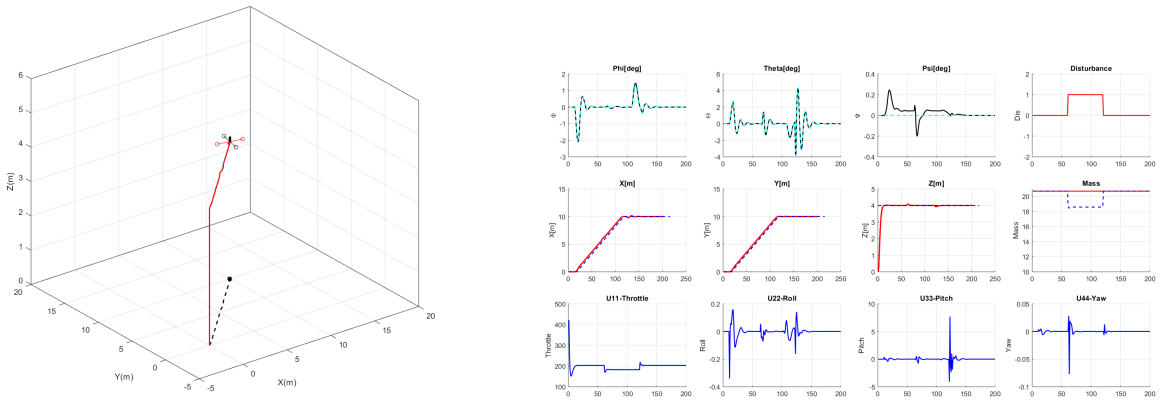


3D plot and Data figure without Disturbance and No Mass variation

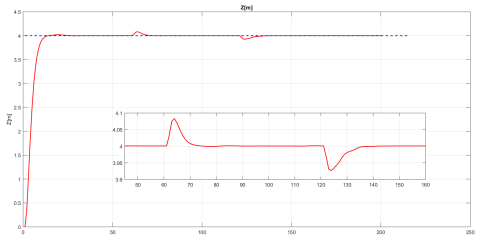


Z plot without Disturbance and No Mass variation

B) For X-Y-Z control with Disturbance and No Mass variation

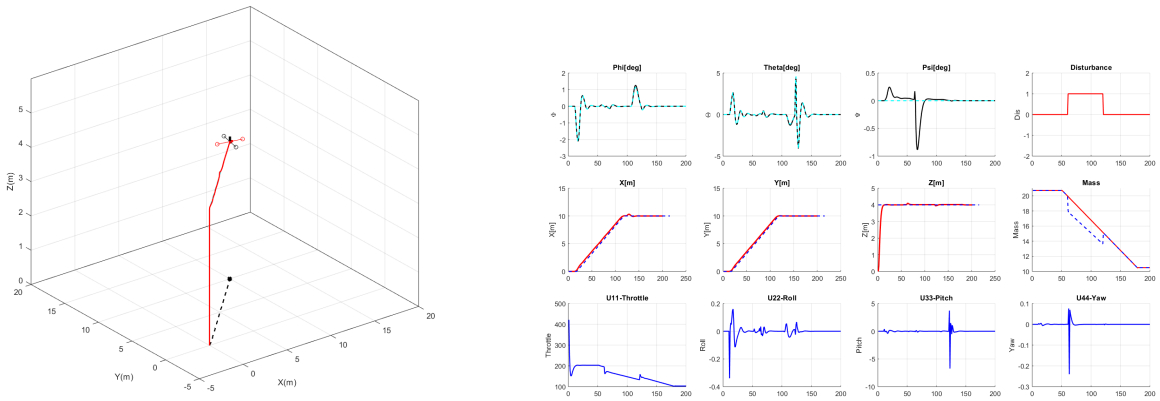


3D plot and Data figure with Disturbance and No Mass variation

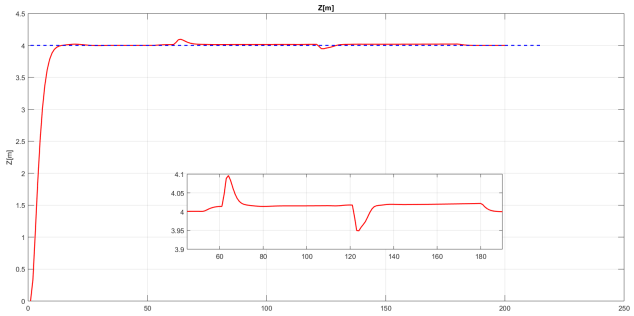


Z plot with Disturbance and No Mass variation

C) For X-Y-Z control with Disturbance and Mass variation



3D plot and Data figure with Disturbance and Mass variation



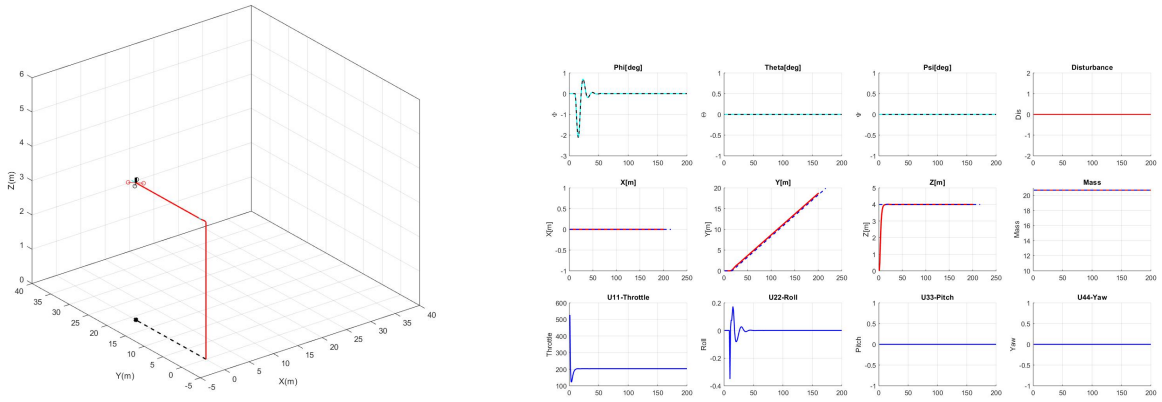
Z plot with Disturbance and Mass variation

6.1.2 Trajectory Tracking

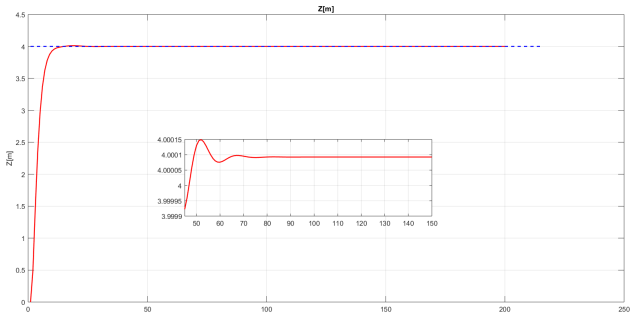
1. For Z($0 \rightarrow 4$), then after the Quad-copter reaches at least 3.8 meter then starts to move in Y($0 \rightarrow t$) direction.

Q_X	Q_Y	Q_Z	Q_ϕ	Q_θ	Q_ψ	$Q_{\dot{X}}$	$Q_{\dot{Y}}$	$Q_{\dot{Z}}$	$R_{Throttle}$	R_{Roll}	R_{Pitch}	R_{Yaw}
306.5739	356.6697	240	0.1681	0.1394	300	100	100	100	0.5	8.4233	0.8	90

A) Without Disturbance and No Mass Variation:

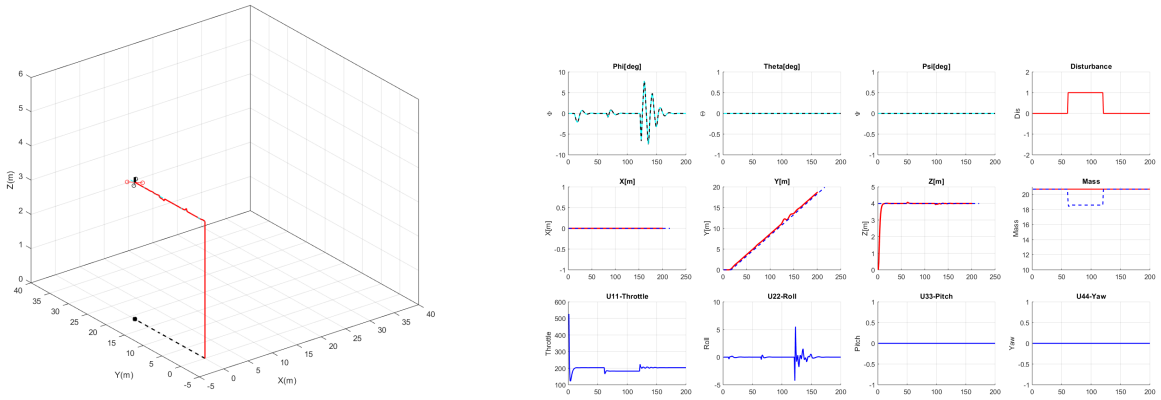


3D plot and Data figure without Disturbance and No Mass variation

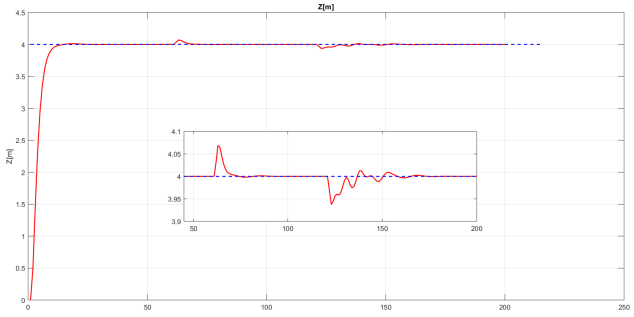


Z plot without Disturbance and No Mass variation

B) With Disturbance(1: assuming the Quad-copter being pulled upward) and No Mass Variation:

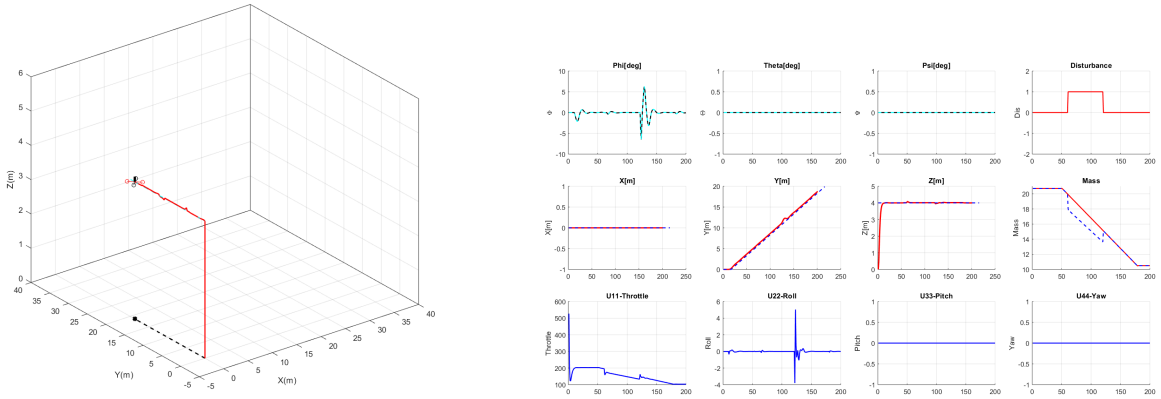


3D plot and Data figure with Disturbance and No Mass variation

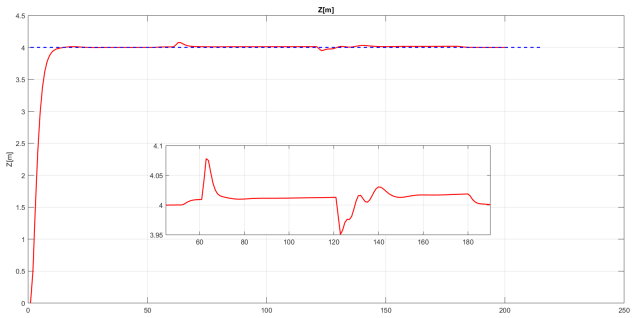


Z plot with Disturbance and No Mass variation

C) With Disturbance and Mass Variation:



3D plot and Data figure with Disturbance and Mass variation

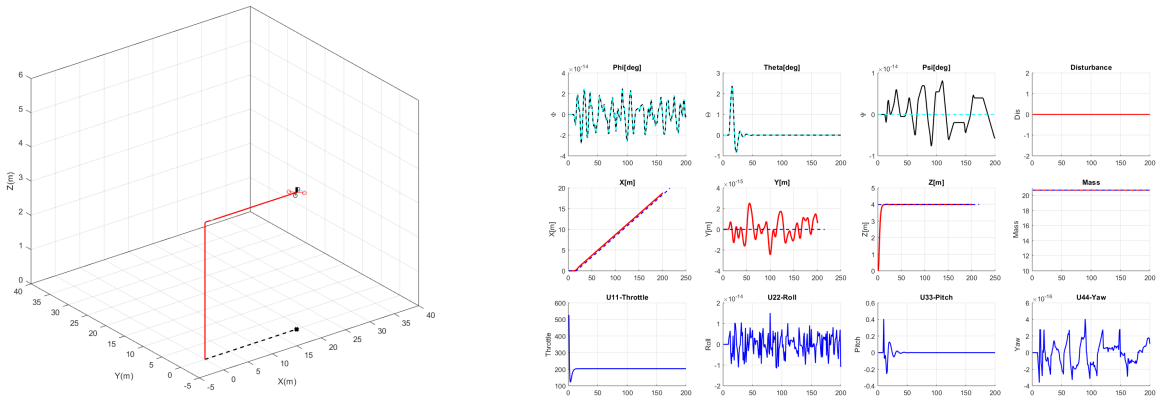


Z plot with Disturbance and Mass variation

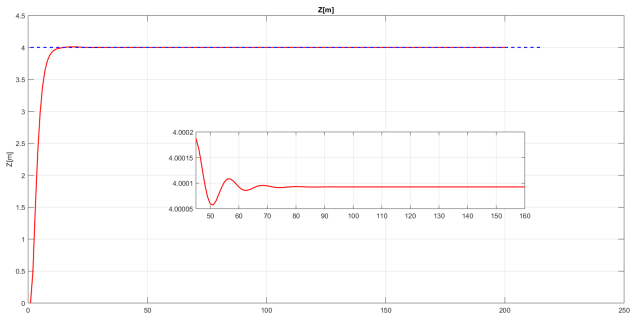
2. For Z($0 \rightarrow 4$), then after the Quad-copter reaches at least 3.8 meter then starts to move in X($0 \rightarrow t$) direction.

Q_X	Q_Y	Q_Z	Q_ϕ	Q_θ	Q_ψ	$Q_{\dot{X}}$	$Q_{\dot{Y}}$	$Q_{\dot{Z}}$	$R_{Throttle}$	R_{Roll}	R_{Pitch}	R_{Yaw}
306.5739	356.6697	240	0.1681	0.1394	300	100	100	100	0.5	8.4233	0.8	90

A) Without Disturbance and No Mass Variation:

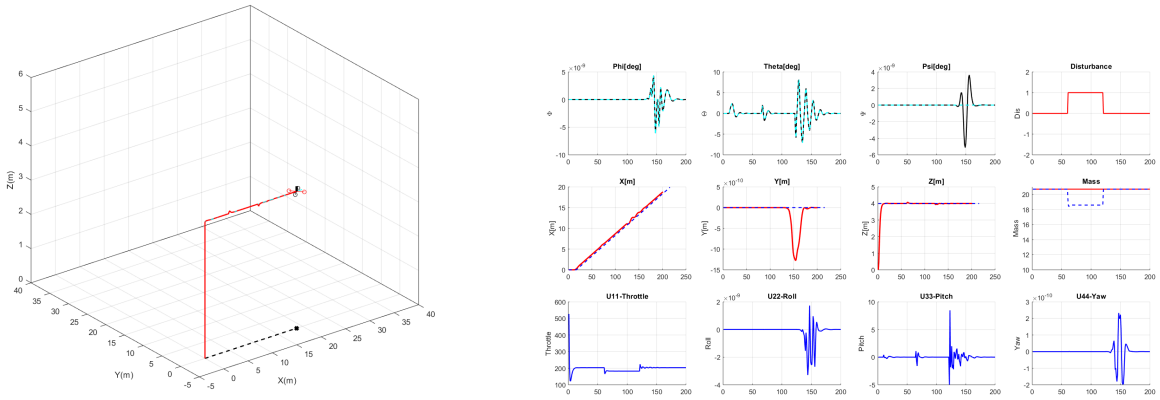


3D plot and Data figure without Disturbance and No Mass variation

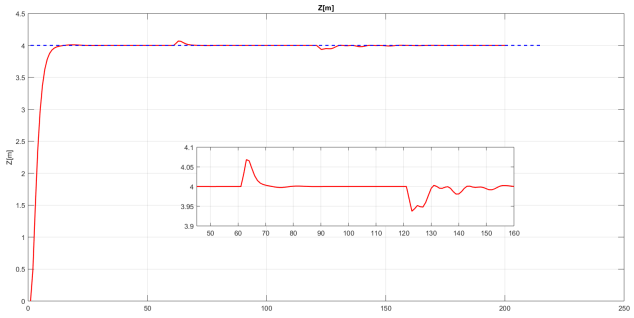


Z plot without Disturbance and No Mass variation

B) With Disturbance(1: assuming the Quad-copter being pulled upward) and No Mass Variation:

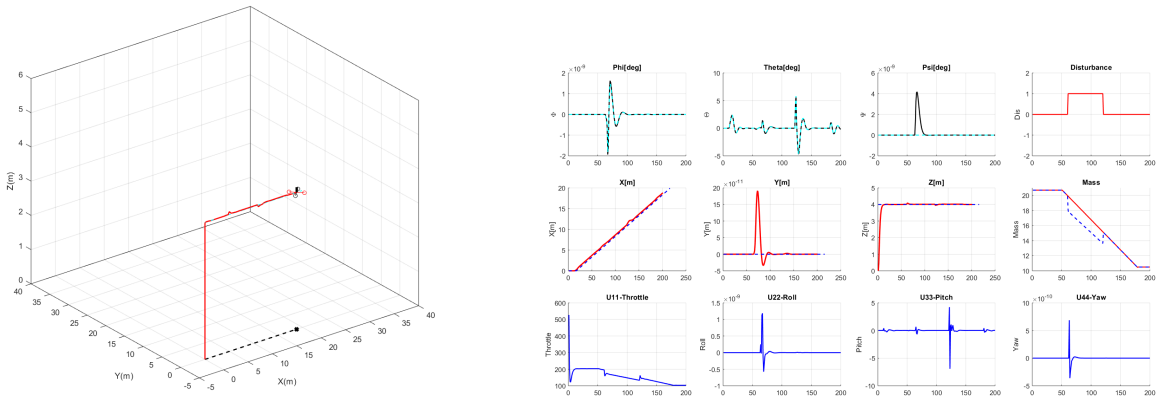


3D plot and Data figure with Disturbance and No Mass variation

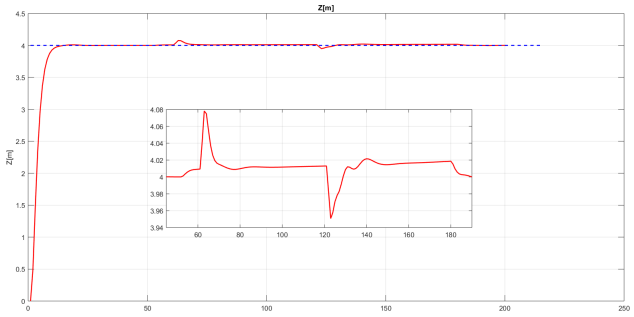


Z plot with Disturbance and No Mass variation

C) With Disturbance and Mass Variation:



3D plot and Data figure with Disturbance and Mass variation



Z plot with Disturbance and Mass variation

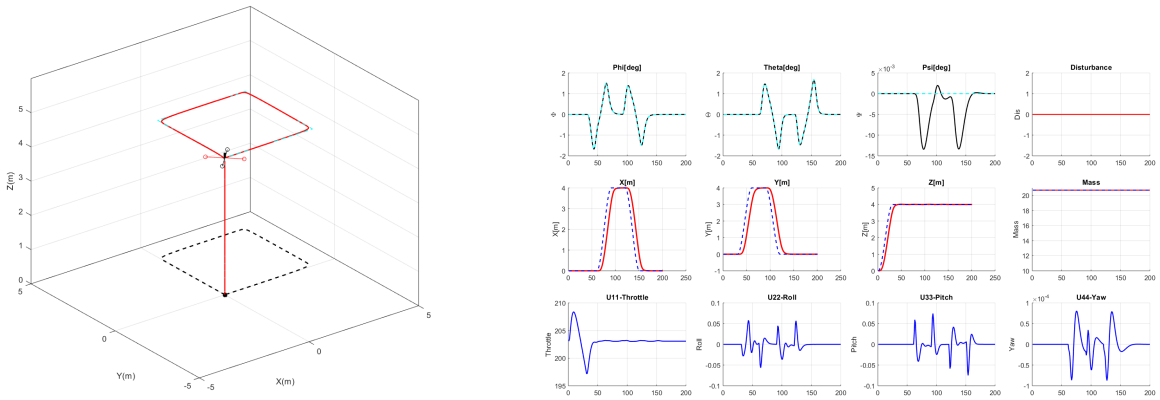
3. For Z($0 \rightarrow 4$), then after the quad-copter reaches at least 3.8 meter then moves in X-Y direction in Square shaped trajectory covering $16m^2$ area (4×4). For computing the reference of the trajectory tracking, polynomials are used. Polynomials are natural choices for providing smooth, continuous motion, with some level of continuous derivatives. For this thesis 3^{rd} order polynomial is used: $X(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

Algorithm 2 Polynomiyal Pseudocode For Rectangular trajectory

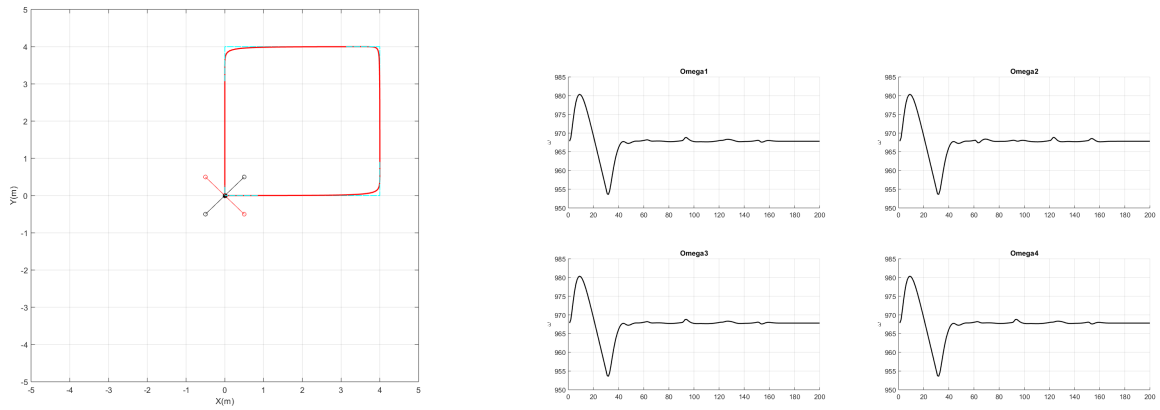
if $t \leq 30$ **then**
    $X_{ref} \leftarrow 0$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 0.01333 * t^2 - 0.000296296 * t^3$
else if $t > 30$ and $t \leq 60$ **then**
    $X_{ref} \leftarrow 0$
    $Y_{ref} \leftarrow 19.999984 - 1.5999984 * t + 0.03999996 * t^2 - 0.000296296 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 60$ and $t \leq 90$ **then**
    $X_{ref} \leftarrow 111.999888 - 4.79999952 * t + 0.0666666 * t^2 - 0.000296296 * t^3$
    $Y_{ref} \leftarrow 4$
    $Z_{ref} \leftarrow 4$
else if $t > 90$ and $t \leq 120$ **then**
    $X_{ref} \leftarrow 4$
    $Y_{ref} \leftarrow -319.99968 + 9.5999904 * t - 0.09333324 * t^2 + 0.000296296 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 120$ and $t \leq 150$ **then**
    $X_{ref} \leftarrow -699.9993 + 15.999984 * t - 0.11999988 * t^2 + 0.000296296 * t^3$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 4$
end if

Q_X	Q_Y	Q_Z	Q_ϕ	Q_θ	Q_ψ	$Q_{\dot{X}}$	$Q_{\dot{Y}}$	$Q_{\dot{Z}}$	$R_{Throttle}$	R_{Roll}	R_{Pitch}	R_{Yaw}
800	800	1000	500	500	300	89.4353	400	267.0252	0.1	1	1	50

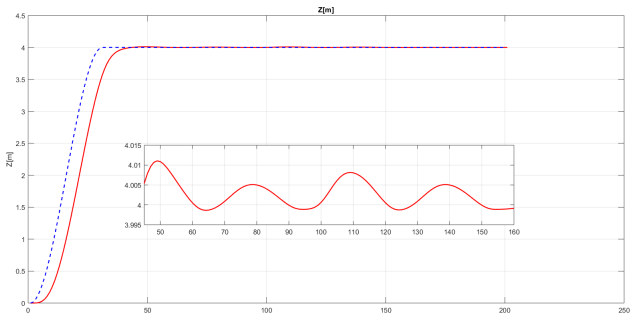
A) Without Disturbance and No Mass Variation:



3D plot and Data figure without Disturbance and No Mass variation

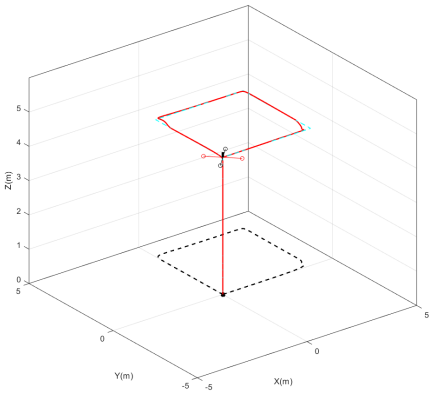


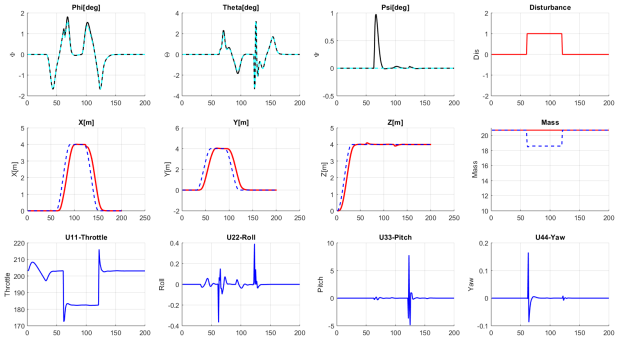
XY and ω plot without Disturbance and No Mass variation



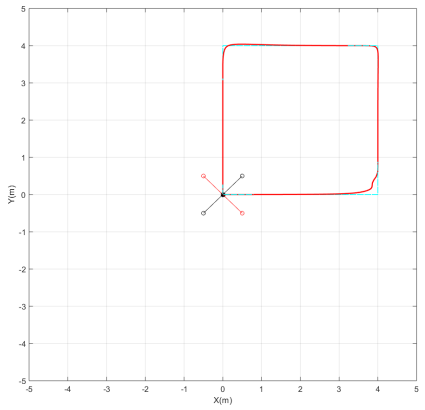
Z plot without Disturbance and No Mass variation

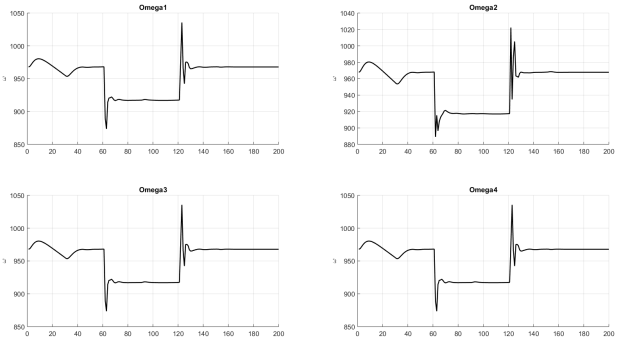
B) With Disturbance and No Mass Variation:



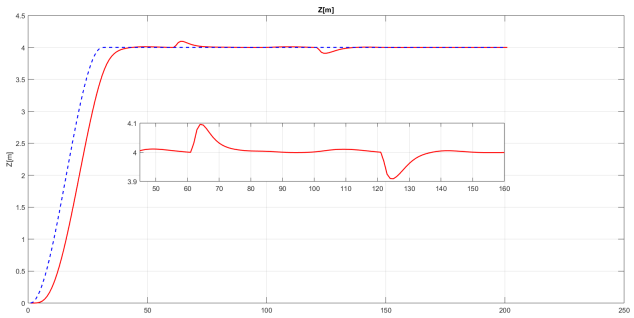


3D plot and Data figure with Disturbance and No Mass variation



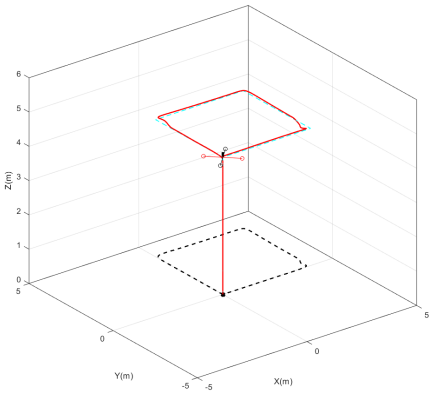


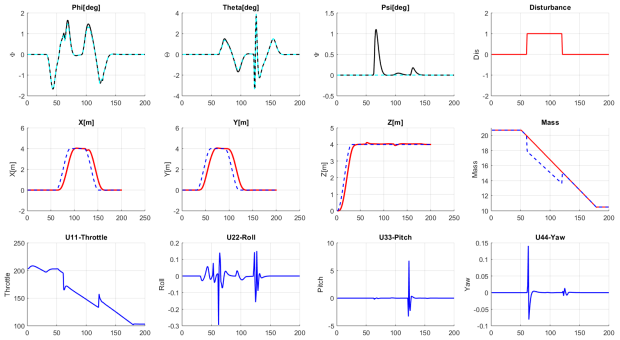
XY and ω plot with Disturbance and No Mass variation



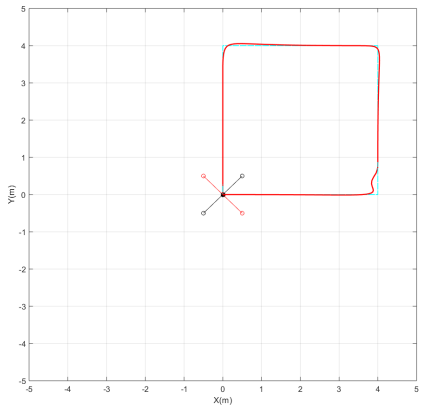
Z plot with Disturbance and No Mass variation

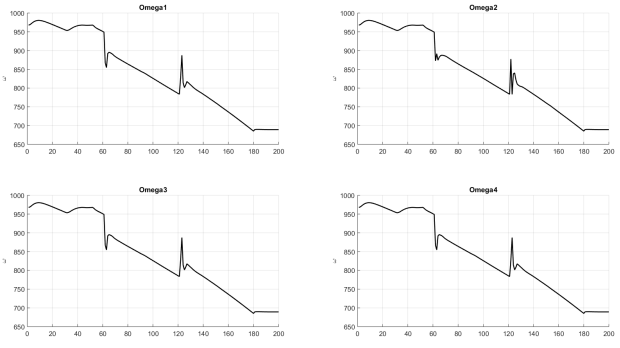
C) With Disturbance and Mass Variation:



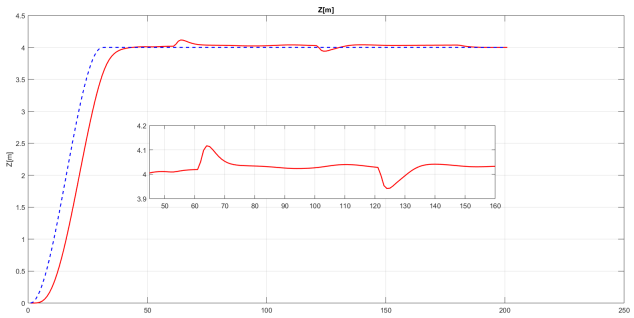


3D plot and Data figure with Disturbance and Mass variation



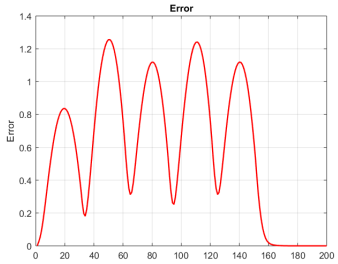


XY and ω plot with Disturbance and with Mass variation

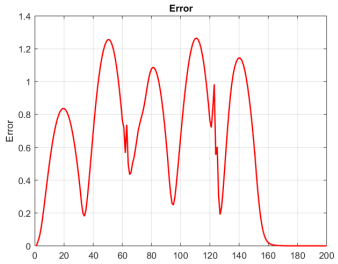


Z plot with Disturbance and Mass variation

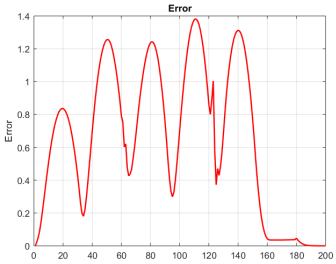
Note: The Error is calculated as:
$Error = \sqrt{(Y_{actual} - Y_{ref})^2}$, where Y_{actual} is a vector containing the actual value of the 12 states and Y_{ref} is a vector containing the reference value of the states.



Error without Disturbance and NO Mass variation



Error with Disturbance and NO Mass variation

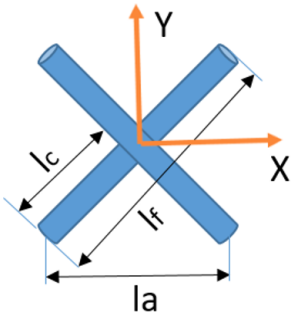


Error with Disturbance and with Mass variation

The Application based Trajectory Tracking

In order to verify the effectiveness of the designed system, $30m \times 30$ ($900m^2$ area) target area is chosen.

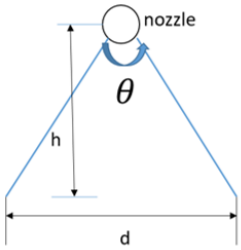
The spray amplitude of the Quad-copter is related to the installation distance of the nozzles. According to the basic requirements of the spray nozzle, the installation space of the adjacent sprinkler must be more than 50cm, thus a better spraying effect can be obtained when spraying[25].



- $l_c = \frac{l_f}{2} = \frac{109cm}{2} = 54.5cm$
- ${l_a}^2 = {l_c}^2 + {l_c}^2 = 54.5^2 + 54.5^2 = 5940.5$
- $l_a = 77.0746$, which is greater than 50cm.

For standard nozzle of type (110-015 types, LECHLER company, Germany), the spray angle of the nozzle is 110°.

To calculate the spray width of a single nozzle (d):

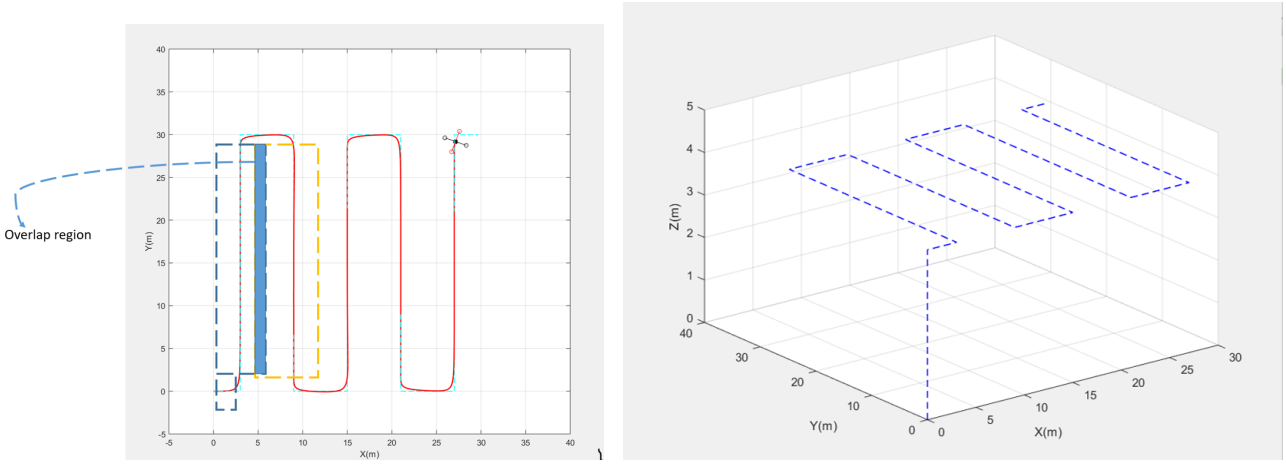


- h is the height from the nozzle to the tip of the crop in m,
- θ is the spray angle
- d is the nozzle spray width of the corresponding height in m

$d = 2 * h * tan(\frac{\theta}{2})$, where $h = 1m$ and $\theta = 110°$

$d = 2 * 1 * tan(\frac{110°}{2}) = 2.85629m$

In order to decrease the width of the overlap area of the adjacent nozzles, the nozzle can be inclined to the other direction.



(a) Covers $900m^2$ in 330sec (b) Desired trajectory

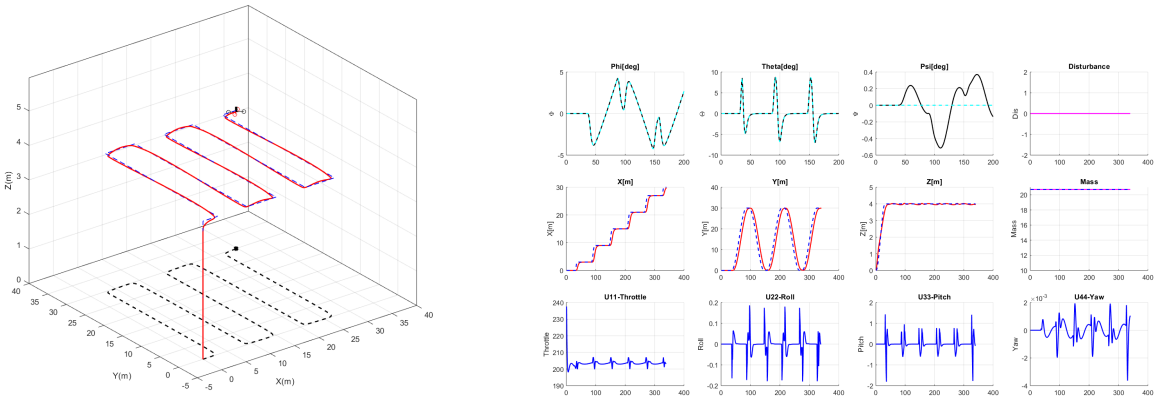
Figure 6.18: Application based Trajectory

∴ First the Quad-copter moves from ground to 3.8 meter in Z-direction. then moves to XY direction, which is the given trajectory. 4m is the desired height because height of a corn, which is a plant mostly affected by locust swarm, is about 2.6m and the quad-copter will be 1.2m above the tip of the corn.

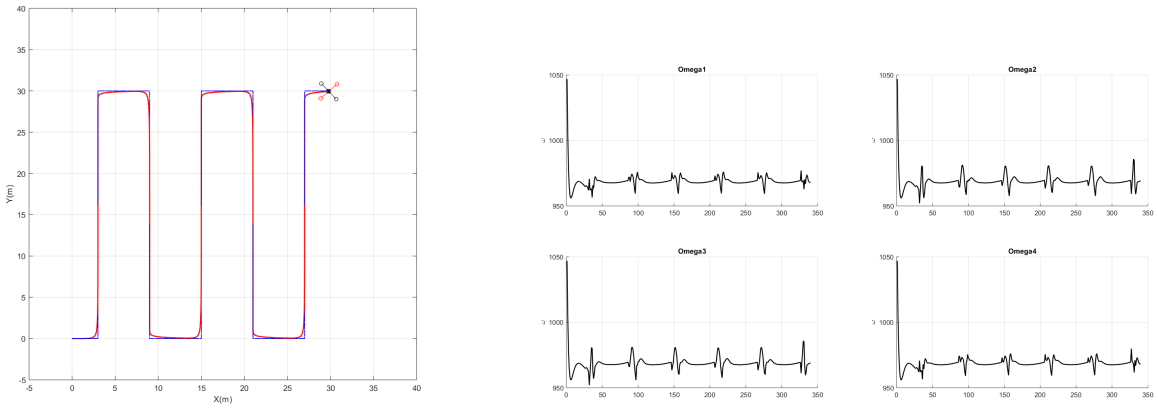
Q_X	Q_Y	Q_Z	Q_ϕ	Q_θ	Q_ψ	$Q_{\dot{X}}$	$Q_{\dot{Y}}$	$Q_{\dot{Z}}$	$R_{Throttle}$	R_{Roll}	R_{Pitch}	R_{Yaw}
300	300	350	500	500	300	100	100	100	0.0001	0	0	300

To check the effectiveness of the design, different scenarios are created:

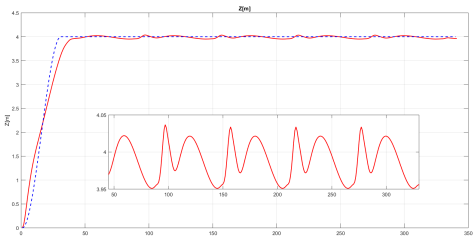
A) Without Disturbance and No Mass Variation:



3D plot and Data figure without Disturbance and No Mass variation

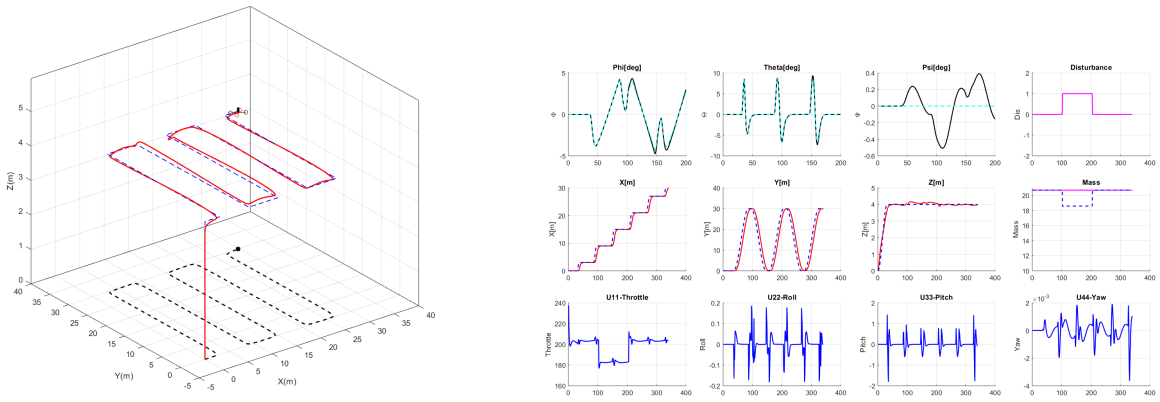


XY and ω plot without Disturbance and No Mass variation

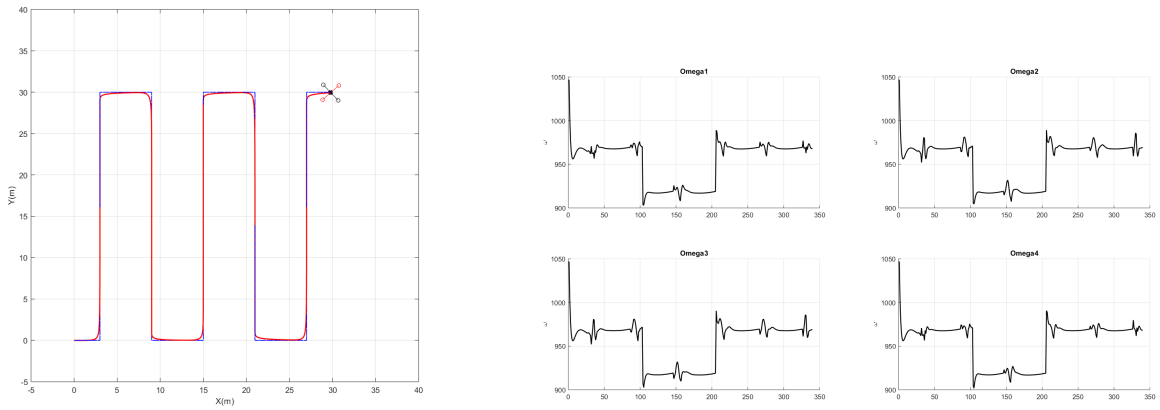


Z plot without Disturbance and No Mass variation

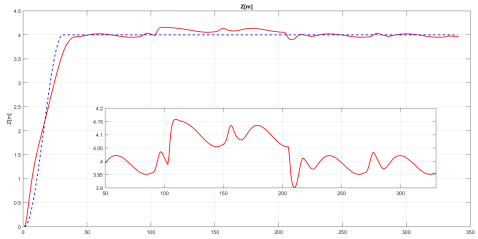
B) With Disturbance and No Mass Variation:



3D plot and Data figure with Disturbance and No Mass variation

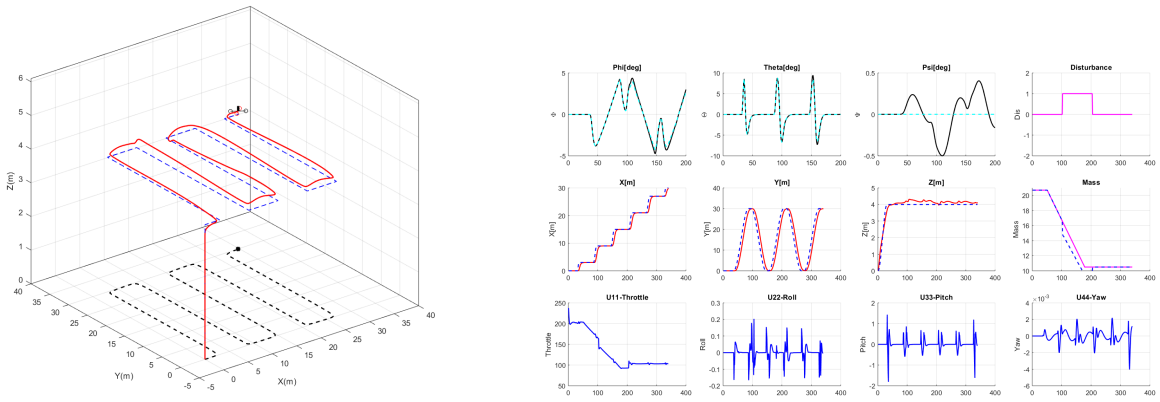


XY and ω plot with Disturbance and No Mass variation

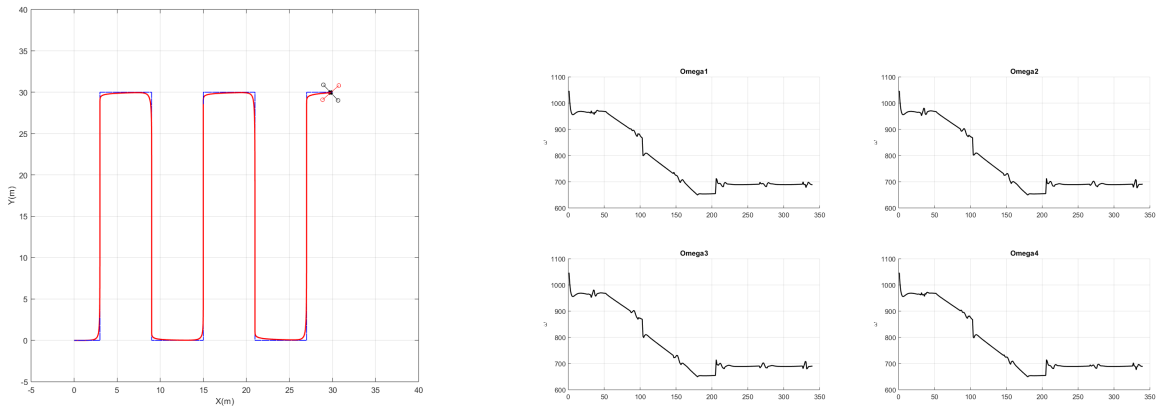


Z plot with Disturbance and No Mass variation

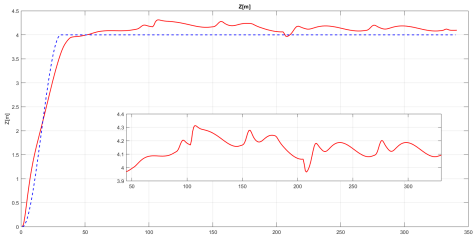
C) With Disturbance and Mass Variation:



3D plot and Data figure with Disturbance and Mass variation

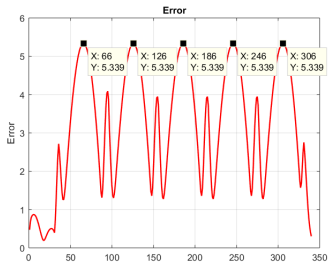


XY and ω plot with Disturbance and Mass variation

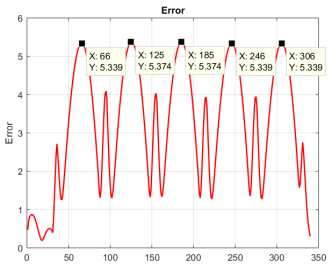


Z plot with Disturbance and Mass variation

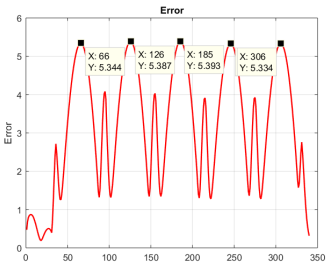
Note: The Error is calculated as:
$Error = \sqrt{(Y_{actual} - Y_{ref})^2}$, where Y_{actual} is a vector containing the actual value of the 12 states and Y_{ref} is a vector containing the reference value of the states.



Error without Disturbance and NO Mass variation



Error with Disturbance and NO Mass variation



Error with Disturbance and with Mass variation

$\therefore$ In the above figure it can be seen, the change in the error due to the disturbance added is from $5.339 \rightarrow 5.374$, It is accurate to say that the controller incorporated with the RLSE handled the disturbance well. In case of the mass variation and disturbance applied together it can also be concluded that the controller with RLSE handled it sufficiently.

6.2 Image Recognition Result

6.2.1 Training Result

- Used 1605 datasets for training and 400 for validation (each divided equally to the categories)
- Image resized to 200×200

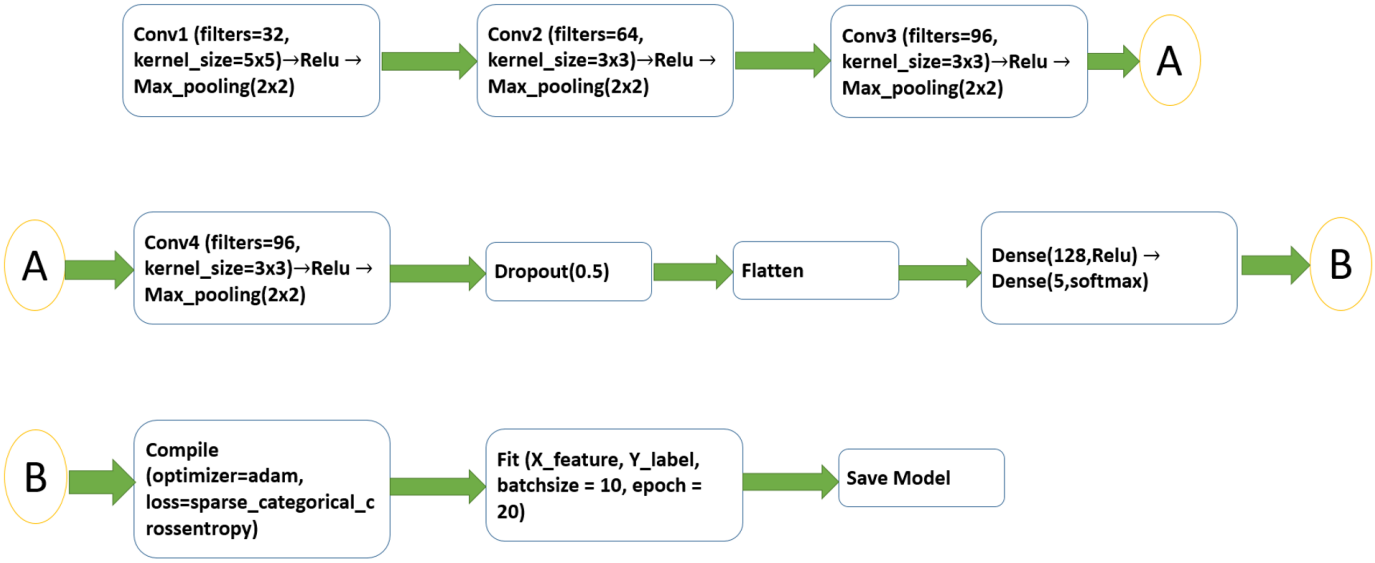
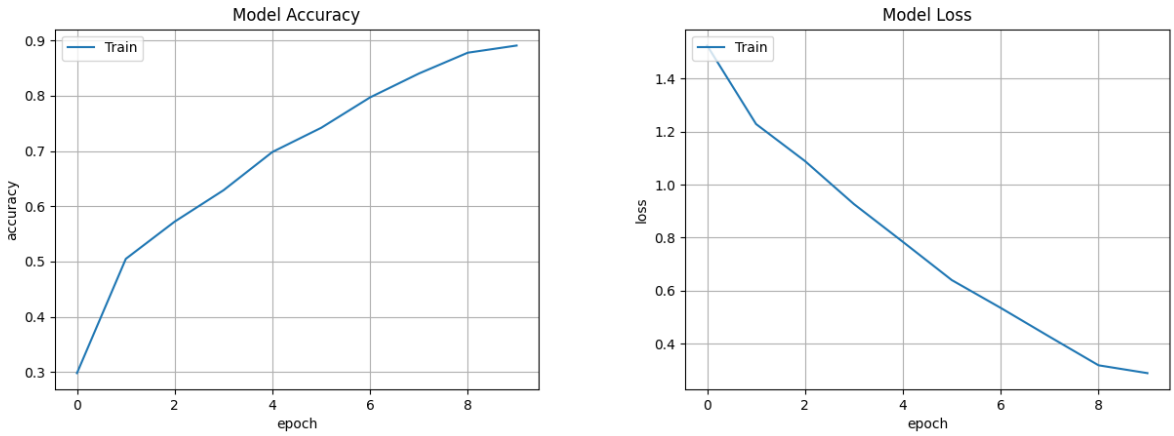


Figure 6.26: CNN Model

The Accuracy and the loss plot are for epochs:10 and epochs:20:-

i Epochs:10

- Training Accuracy: 89.07%
- Training loss: 0.2897
- Test Accuracy: 80.95%



(a) Accuracy Plot for Epochs:10 (b) Loss Plot for Epochs:10

Figure 6.27: Accuracy and Loss Plot for Epochs:10

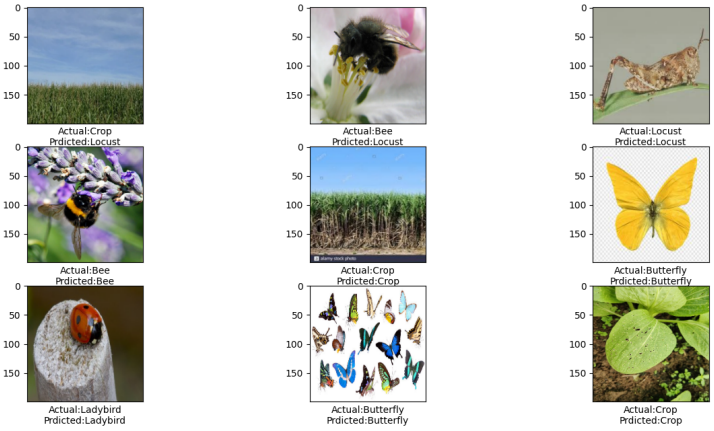
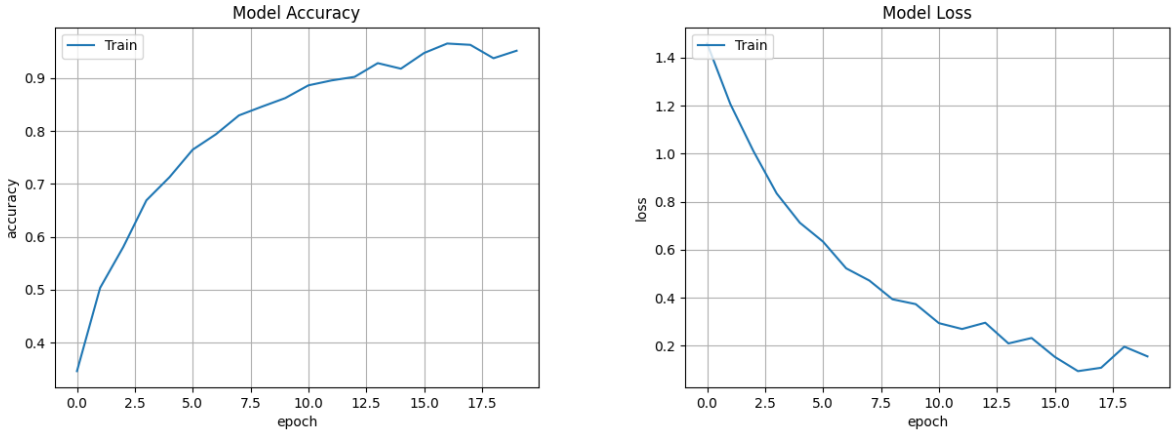


Figure 6.28: Image test result for Epochs 10

ii Epochs:20

- Training Accuracy: 95.19%
- Training loss: 0.1560
- Validation Accuracy: 93.24%



(a) Accuracy Plot for Epochs:20

(b) epoch 20 loss:20

Figure 6.29: Accuracy and Loss Plot for Epochs:20

6.2.2 Test Result

The categories where five: $['Bee' : 0, 'Butterfly' : 1, 'Crop' : 2, 'Ladybird' : 3, 'Locust' : 4]$ so the test was done randomly on each categories.

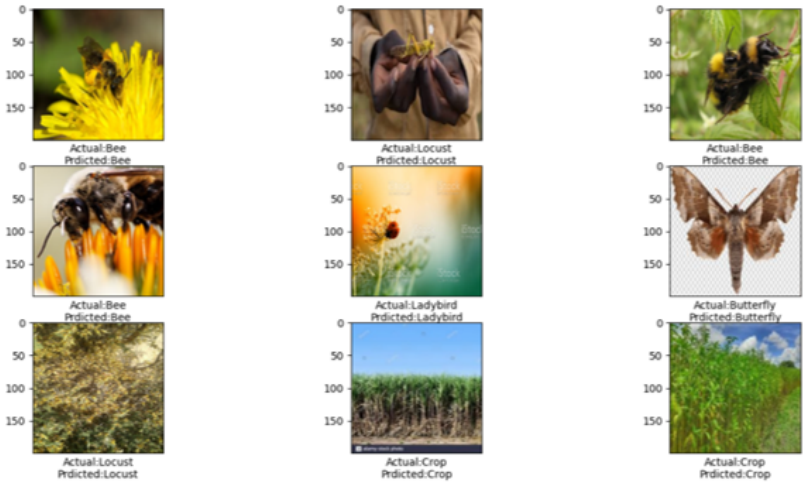


Figure 6.30: Image test result 1

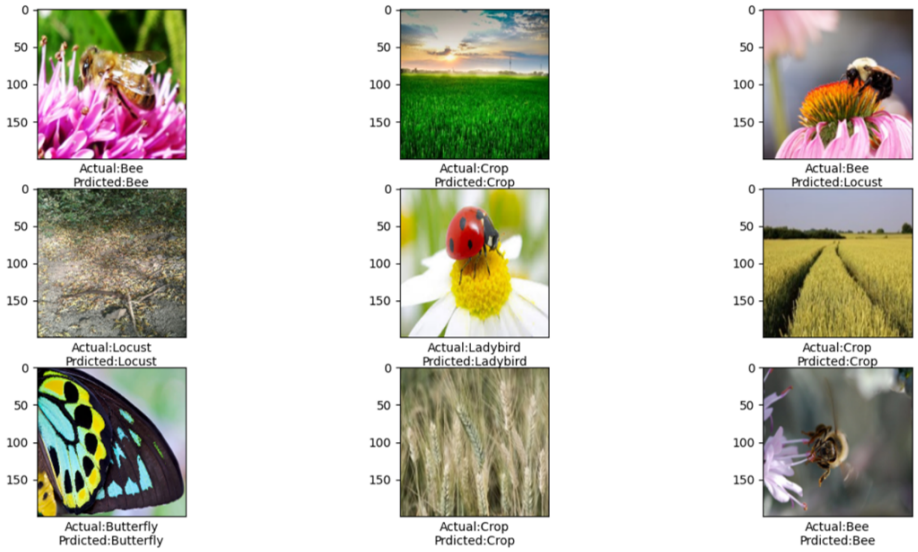


Figure 6.31: Image test result 2

Algorithm 3 Polynomiyal Pseudocode the application based Trajectory

if $t \leq 30$ **then**
    $X_{ref} \leftarrow 0$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 0.01333 * t^2 - 0.000296296 * t^3$
else if $t > 30$ and $t \leq 35$ **then**
    $X_{ref} \leftarrow 1620 - 151.2 * +4.68 * t^2 - 0.048 * t^3$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 4$
else if $t > 35$ and $t \leq 85$ **then**
    $X_{ref} \leftarrow 3$
    $Y_{ref} \leftarrow 64.68 - 4.284 * t + 0.0864 * t^2 - 0.00048 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 85$ and $t \leq 95$ **then**
    $X_{ref} \leftarrow 8673 - 290.7 * t + 3.24 * t^2 - 0.012 * t^3$
    $Y_{ref} \leftarrow 30$
    $Z_{ref} \leftarrow 4$
else if $t > 95$ and $t \leq 145$ **then**
    $X_{ref} \leftarrow 9$
    $Y_{ref} \leftarrow -706.44 + 19.836 * t - 0.1728 * t^2 + 0.00048*^3$
    $Z_{ref} \leftarrow 4$
else if $t > 145$ and $t \leq 155$ **then**
    $X_{ref} \leftarrow 40377 - 809.1 * t + 5.4 * t^2 - 0.012 * t^3$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 4$
else if $t > 155$ and $t \leq 205$ **then**
    $X_{ref} \leftarrow 15$
    $Y_{ref} \leftarrow 2652.36 - 45.756 * t + 0.2592 * t^2 - 0.00048 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 205$ and $t \leq 215$ **then**
    $X_{ref} \leftarrow 110961 - 1586.7 * t + 7.56 * t^2 - 0.012 * t^3$
    $Y_{ref} \leftarrow 30$
    $Z_{ref} \leftarrow 4$
else if $t > 215$ and $t \leq 265$ **then**
    $X_{ref} \leftarrow 21$
    $Y_{ref} \leftarrow -6404.52 + 82.044 * t - 0.3456 * t^2 + 0.00048 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 265$ and $t \leq 275$ **then**
    $X_{ref} \leftarrow 235977 - 2623.5 * t + 9.72 * t^2 - 0.012 * t^3$
    $Y_{ref} \leftarrow 0$
    $Z_{ref} \leftarrow 4$
else if $t > 275$ and $t \leq 325$ **then**
    $X_{ref} \leftarrow 27$
    $Y_{ref} \leftarrow 12705 - 128.7 * t + 0.432 * t^2 - 0.00048 * t^3$
    $Z_{ref} \leftarrow 4$
else if $t > 325$ and $t \leq 330$ **then**
    $X_{ref} \leftarrow 1685802 - 15444 * t + 47.16 * t^2 - 0.048 * t^3$
    $Y_{ref} \leftarrow 30$ and $Z_{ref} \leftarrow 4$
end if

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In this thesis a model predictive controller is designed to control a quad-copter, which is used for locust detection and bio-pesticide spraying. First the mathematical formulation and modeling was done using Newton-Euler formulation.

Once the model has been formulated and verified using Simullink, the controller was designed. The type of MPC used is Non-linear MPC, therefore the dynamics system was not linearized. To design the NMPC, first it was descritized using 4^{th} order Rung-Kutta (RK4) method. To change the optimal control problem to non-linear program problem, multiple shooting was implemented. After the NLP formulation, the solver Interior point optimizer(Ipopt) was used which is CasADi in MATLAB.

Next to check if the designed controller was working well, both point stabilization and trajectory tracking, which are control objectives were applied. The controller was a able to track the reference trajectory or reference point well. Afterwards the controller was tested if it can handle disturbance, which was add in $\ddot{Z}$ assuming it is being pulled upward and the controller was able to handle the disturbance and stabilized the quad-copter. To consider the mass variation, Recursive Least Square Estimation(RLSE) method was executed and was capable at estimating the mass during the mass variation. The RLSE was also performing very well at estimating when there is disturbance so even if there is no mass variation, if there is disturbance, it can estimate the change in the system and the model can be improved.

For the image recognition of Locust, python programming is used. Five categories were used:$[' Bee' : 0,' Butterfly' : 1,' Crop' : 2,' Ladybird' : 3,' Locust' : 4]$ to train the machine. After training the machine with the given dataset an accuracy of 95.19% was obtained. The model is then tested using different images, other than the training and validation image data, and was able to predict the images with 93.24% accuracy.

To incorporate the image recognition and the NMPC controller, MATLAB calls python at some sampling time and python predicts the test image. If the prediction is locust then there will be a mass variation in the plant due to spraying (Simulink) so RLSE estimates the change and the model (MATLAB) will be improved. If the prediction is not a locust then no mass variation because there will not be any spraying.

7.2 Recommendation

- Examine if the designed controller can handle other complicated trajectories, such as: Elliptical and Helical path.
- Adding an obstacle as a path constraint and check the proposed controller design. It is desired to maintain lower bound for the Euclidean distance between the prediction of the robot position and the obstacle position. Therefore, it is needed to impose the following path constraints: $(P_q - P_o)^2 > d^2$, where P_q and P_o are the Quad-copter position and obstacle position and d is the desired distance. It needs to ensure the above inequality constraint for all prediction steps, to keep the Quad-copter away from pre-defined obstacles.
- Diagnosing the controller output fault as by adding uncertainty terms in the control oriented model especially in X and Y direction.
- NMPC requires a power full, fast processor with large memory because it does online optimization, So when considering to use this controller this has to be in mind.
- To consider other categories for image recognition to decrease the false positive.

References

[1] 2021. [Online]. Available: https://www.fao.org/resilience/resources/resources-detail/en/c/278608/

[2] R. Tesfaye, "Modeling and control of a quad-rotor unmanned aerial vehicle at hovering position," Master's thesis, Addis Ababa University, 2012.

[3] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.

[4] A. Raemaekers, "Design of a model predictive controller to control uavs," *Technische Universiteit Eindhoven*, vol. 2007, 2007.

[5] P. Ru and K. Subbarao, "Nonlinear model predictive control for unmanned aerial vehicles," *Aerospace*, vol. 4, no. 2, p. 31, 2017.

[6] X. Zhou, X. Zhang, J. Zhang, and R. Liu, "Stabilization and trajectory control of a quadrotor with uncertain suspended load," *arXiv preprint arXiv:1612.04324*, 2016.

[7] J. Tamimi and P. Li, "Nonlinear model predictive control using multiple shooting combined with collocation on finite elements," *IFAC Proceedings Volumes*, vol. 42, no. 11, pp. 703–708, 2009.

[8] M. W.Mehrez, "Optimization based solutions for control and state estimation in dynamical systems (implementation to mobile robots)," Jan. 2019.

[9] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[10] H. Mohammadi Daniali, "Fast nonlinear model predictive control of quadrotors: Design and experiments," Master's thesis, University of Waterloo, 2019.

[11] M. Dhaybi and N. Daher, "Real-time estimation of the inertia tensor elements of a quadcopter hover platform," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2019, pp. 1347–1352.

[12] M. Krznar, D. Kotarski, P. Piljek, and D. Pavković, "On-line inertia measurement of unmanned aerial vehicles using on board sensors and bifilar pendulum," *Interdisciplinary Description of Complex Systems: INDECS*, vol. 16, no. 1, pp. 149–161, 2018.

[13] R. Chauhan, K. K. Ghanshala, and R. Joshi, "Convolutional neural network (cnn) for image detection and recognition," in *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. IEEE, 2018, pp. 278–282.

[14] N. Tahir and G. Brooker, "Feasibility of uav based optical tracker for tracking australian plague locust," in *Austlalasian Conference on Robotics and Automation (ACRA 2009). Sydney, Australia*, 2009.

[15] M. C. Gebretensiae, "Locust identification system using deep neural network," Mar. 2021.

[16] Q. Quan, *Introduction to Multicopter Design and Control*. Beijing, China: Springer Nature, Dec. 2016.

[17] [Online]. Available: https://www.alibaba.com/product-detail/JMR-X1400-Quad-10L-agriculture-drone_62187405969.html

[18] [Online]. Available: https://en.m.wikipedia.or/wiki/Euler%27s_laws_of_motion

[19] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*, 2nd ed. Nob Hill Publishing Madison, WI, 2019, vol. 2.

[20] A. Myrtellari, P. Marango, and M. Gjonaj, "Analysis and performance of linear quadratic regulator and pso algorithm in optimal control of dc motor," *International Journal of Latest Research in Engineering and Technology*, vol. 2, no. 4, 2016.

[21] [Online]. Available: https://web.casadi.org/docs/

[22] [Online]. Available: https://bulkimagedownloader.com

[23] [Online]. Available: https://www.kaggle.com

[24] P. Chudzik, A. Mitchell, M. Alkaseem, Y. Wu, S. Fang, T. Hudaib, S. Pearson, and B. Al-Diri, "Mobile real-time grasshopper detection and data aggregation framework," *Scientific reports*, vol. 10, no. 1, pp. 1–10, 2020.

[25] S. Wen, Q. Zhang, J. Deng, Y. Lan, X. Yin, and J. Shan, “Design and experiment of a variable spray system for unmanned aerial vehicles based on pid and pwm control,” *Applied Sciences*, vol. 8, no. 12, p. 2482, 2018.

Appendix A

Rotation Matrix

Linear velocity (u, v, w) in body-fixed frame can be used to obtain linear velocity in earth inertial frame $\left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}\right)$. 1^{st} the rotation about Z_E axis (fig A.1) of the angle ψ(Yaw), 2^{nd} the rotation about the Y_1 axis (fig A.2) of the angle θ(Pitch) and at last the rotation about the X_2 axis (fig A.3) of the angle ϕ(Roll).

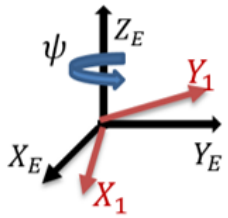


Figure A.1: Rotation about Z_E axis

$$cos(\psi) = \frac{X_E}{X_1}, sin(\psi) = \frac{Y_E}{X_1}, cos(\psi) = \frac{Y_E}{Y_1}, sin(\psi) = -\frac{X_E}{Y_1} \tag{A.1}$$

$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \tag{A.2}$$

$$cos(\theta) = \frac{X_1}{X_2}, sin(\theta) = -\frac{Z_1}{X_2}, cos(\theta) = \frac{Z_1}{Z_2}, sin(\theta) = \frac{X_1}{Z_2} \tag{A.3}$$

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{A.4}$$

$$cos(\phi) = \frac{Z_1}{Z_2}, sin(\phi) = -\frac{Y_1}{Z_2} cos(\phi) = \frac{Y_1}{Y_2}, sin(\phi) = \frac{Z_1}{Y_2} \tag{A.5}$$

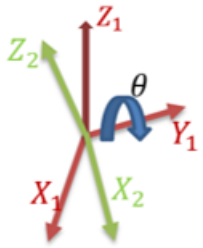


Figure A.2: Rotation about Y_1 axis

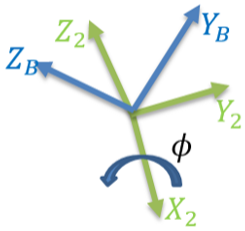


Figure A.3: Rotation about X_2 axis

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} \tag{A.6}$$

The 3 rotation matrices will multiply (A.7): ${}^{E}_{B}R = R_{\psi}R_{\theta}R_{\phi}$

$${}^{E}_{B}R = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{A.7}$$

$${}^{E}_{B}R = \begin{bmatrix} cos(\theta)cos(\psi) & sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi) \\ cos(\theta)sin(\psi) & cos(\theta)cos(\psi) + sin(\phi)sin(\theta)sin(psi) & -sin(\phi)cos(\psi) + cos(phi)sin(\theta)sin(\psi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\phi)cos(\theta) \end{bmatrix} \tag{A.8}$$

Appendix B

Transfer Matrix

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{B.1}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = {R_\phi}^{-1} {R_\theta}^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + {R_\phi}^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{B.2}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{B.3}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ sin(\phi)sin(\theta) & cos(\phi) & sin(\phi)cos(\theta) \\ cos(\phi)sin(\theta) & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{B.4}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} -sin(\theta)\dot{\psi} \\ sin(\phi)cos(\theta)\dot{\psi} \\ cos(\phi)cos(\theta)\dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ cos(\phi)\dot{\theta} \\ -sin(\phi)\dot{\theta} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{B.5}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{B.6}$$

$$T^{-1} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{B.7}$$

Appendix C

Total Differentiation of a vector in a Rotating Reference Frame

Before Newton's second law of motion for a reference frame rotating with the earth is written, the relationship between the total derivative of a vector in an inertial reference frame should be developed and the corresponding derivative in a rotating system. Let:

1. $\vec{v} = ui + vj + wk$, is in an inertial frame of reference and

2. $\vec{v} = u'i' + v'j' + w'k'$, is in rotating frame of reference

For 1 the derivative will be:

$$\frac{d\vec{v}}{dt} = \frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k + \frac{di}{dt}u + \frac{dj}{dt}v + \frac{dk}{dt}v \tag{C.1}$$

Since the coordinate axes are in an inertial frame of reference,

$$\frac{di}{dt} = \frac{dj}{dt} = \frac{dk}{dt} = 0 \tag{C.2}$$

$$\therefore \frac{d\vec{v}}{dt} = \frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k \tag{C.3}$$

For 2 the derivative will be:

$$\frac{d\vec{v}}{dt} = \frac{du'}{dt}i' + \frac{dv'}{dt}j' + \frac{dw'}{dt}k' + \frac{di'}{dt}u' + \frac{dj'}{dt}v' + \frac{dk'}{dt}v' \tag{C.4}$$

Since Equation C.3 and Equation C.4 have equal left side:

$$\frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k = \frac{du'}{dt}i' + \frac{dv'}{dt}j' + \frac{dw'}{dt}k' + \frac{di'}{dt}u' + \frac{dj'}{dt}v' + \frac{dk'}{dt}v' \tag{C.5}$$

$$\left(\frac{du}{dt}i + \frac{dv}{dt}j + \frac{dw}{dt}k\right) = \left(\frac{du'}{dt}i' + \frac{dv'}{dt}j' + \frac{dw'}{dt}k'\right) + \left(\frac{di'}{dt}u' + \frac{dj'}{dt}v' + \frac{dk'}{dt}v'\right) \tag{C.6}$$

$$\left(\frac{d\vec{v}}{dt}\right)_{inertia} = \left(\frac{d\vec{v}}{dt}\right)_{rotating} + \text{Effect of Rotation} \tag{C.7}$$

To interpret $\frac{di'}{dt}, \frac{dj'}{dt}, \frac{dk'}{dt}$, the unit vector (i, j, k) is considered as a position vector.
linear velocity = angular velocity $\times$ position vector
$\vec{V} = \omega \times \vec{r}$
Since $\vec{V} = \frac{d\vec{r}}{dt}, \frac{d\vec{r}}{dt} = \omega \times \vec{r}$

Thus:
$\frac{di'}{dt} = \omega \times i', \frac{dj'}{dt} = \omega \times j', \frac{dk'}{dt} = \omega \times k'$

$$\left(\frac{d\vec{v}}{dt}\right)_{inertia} = \left(\frac{d\vec{v}}{dt}\right)_{rotating} + \omega \times \vec{v} \tag{C.8}$$

Appendix D

Inertia Tensor

The total UAV inertia tensor is obtained by combining the inertia contributions of each body:Frame, Box to cover the frame center, Flight computer, Tank to hold the pesticide, Electronics, bar to support the Quad-copter, Brushless motors, Propellers and the sprayer.

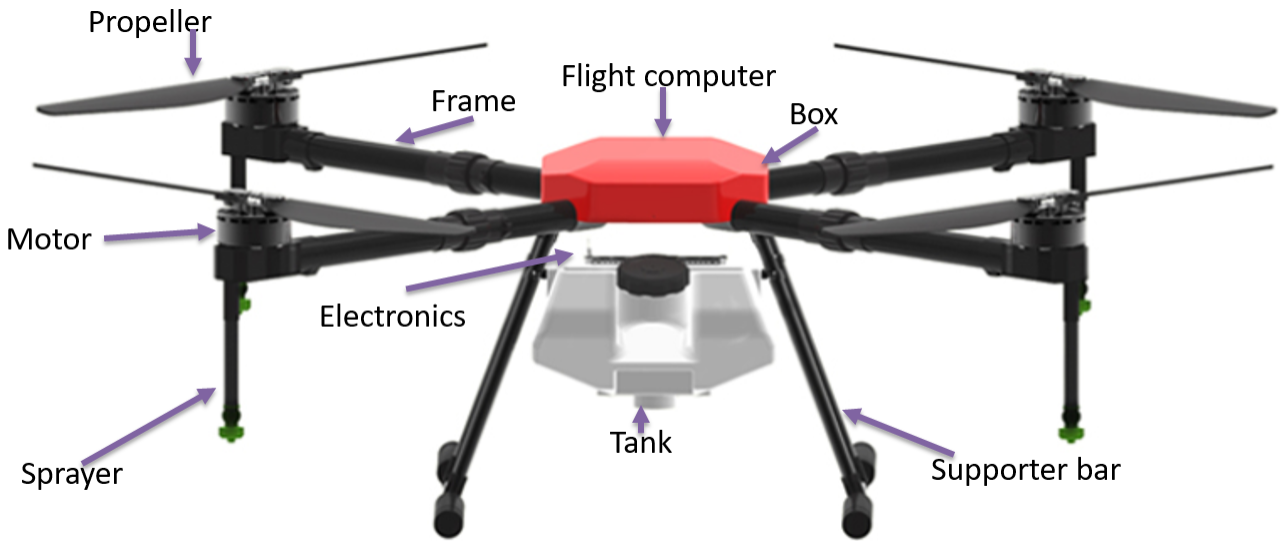
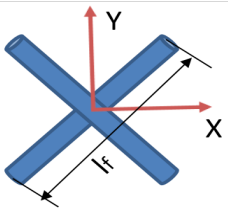
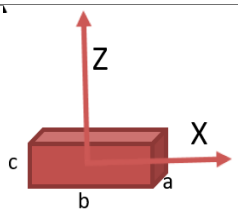
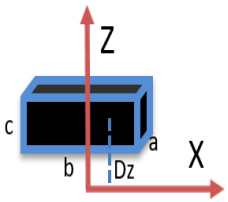
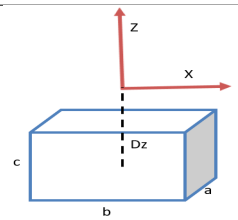
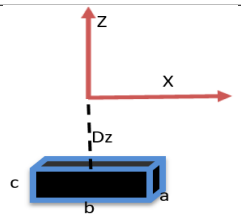
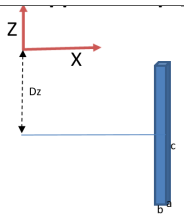
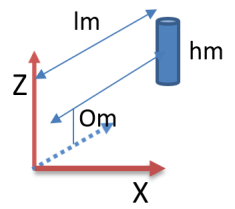
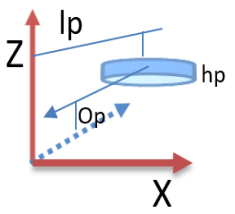
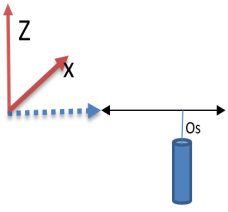


Figure D.1: Main Components of a Quad-copter

Table D.1: Inertia tensor

Main Component	**Formula**	**Geometry**	**Dimenssions**	**Calculated inertia,** $kg \cdot m^2$
Frame	• $I_{xx} = M\left(\frac{{l_f}^2}{12}\right) + M\left(\frac{{r_f}^2}{4}\right) + M\left(\frac{{r_f}^2}{2}\right)$ • $I_{yy} = M\left(\frac{{l_f}^2}{12}\right) + M\left(\frac{{r_f}^2}{4}\right) + M\left(\frac{{r_f}^2}{2}\right)$ • $I_{zz} = M\left(\frac{{l_f}^2}{12} + \frac{{r_f}^2}{4}\right) + M\left(\frac{{r_f}^2}{4} + \frac{{l_f}^2}{12}\right)$		• $r_f = 0.02935m$ • $M = 3.22967kg$ • $l_f = 1.09m$	• $I_{xx} = 0.64092$ • $I_{yy} = 0.64092$ • $I_{zz} = 0.2276$
Box	• $I_{xx} = M\left(\frac{a^2}{12} + \frac{c^2}{12}\right)$ • $I_{yy} = M\left(\frac{c^2}{12} + \frac{b^2}{12}\right)$ • $I_{zz} = M\left(\frac{b^2}{12} + \frac{a^2}{12}\right)$		• $a = b = 0.105m$ • $M = 2.6748kg$ • $c = 0.10481m$	• $I_{xx} = 0.004897$ • $I_{yy} = 0.004897$ • $I_{zz} = 0.004897$
Flight computer	• $I_{xx} = M\left(\frac{a^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{yy} = M\left(\frac{b^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{zz} = M\left(\frac{b^2}{12} + \frac{a^2}{12}\right)$		• $a = 0.055m$ • $b = 0.08m$ • $M = 0.038kg$ • $c = 0.018m$ • $D_z = 0.0614m$	• $I_{xx} = 0.000152$ • $I_{yy} = 0.0001627$ • $I_{zz} = 0.000029845$
Tank	• $I_{xx} = M\left(\frac{a^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{yy} = M\left(\frac{b^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{zz} = M\left(\frac{b^2}{12} + \frac{a^2}{12}\right)$		• $a = 0.21544m$ • $b = 0.21544m$ • $M = 10.2kg$ • $c = 0.21544m$ • $D_z = 0.16012m$	• $I_{xx} = 0.3404$ • $I_{yy} = 0.3404$ • $I_{zz} = 0.0789$

Electronics	• $I_{xx} = M\left(\frac{a^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{yy} = M\left(\frac{b^2}{12} + \frac{c^2}{12} + {D_z}^2\right)$ • $I_{zz} = M\left(\frac{b^2}{12} + \frac{a^2}{12}\right)$		• $a = 0.2m$ • $b = 0.12m$ • $M = 0.4kg$ • $c = 0.05m$ • $D_z = 0.2928m$	• $I_{xx} = 0.0357$ • $I_{yy} = 0.03487$ • $I_{zz} = 0.0005633$
Supporter bar	• $I_{xx} = M\left(\frac{a^2}{12} + \frac{c^2}{12} + {D_z}^2 + {D_y}^2\right)$ • $I_{yy} = M\left(\frac{b^2}{12} + \frac{c^2}{12} + {D_x}^2 + {D_z}^2\right)$ • $I_{zz} = M\left(\frac{b^2}{12} + \frac{a^2}{12} + {D_x}^2 + {D_y}^2\right)$		• $a = 0.058692m$ • $b = 0.058692m$ • $M = 0.4kg$ • $c = 0.45m$ • $D_z = 0.2928m$	• $I_{xx} = 0.17364$ • $I_{yy} = 0.17364$ • $I_{zz} = 0.08628$
Motor	• $I_{xx} = M\left(\frac{{r_m}^2}{4} + \frac{{h_m}^2}{12} + {O_m}^2\right)$ • $I_{yy} = M\left(\frac{{r_m}^2}{4} + \frac{{h_m}^2}{12} + {l_m}^2 + {O_m}^2\right)$ • $I_{zz} = M\left(\frac{{r_m}^2}{2} + {l_m}^2\right)$		• $r_m = 0.0245m$ • $h_m = 0.0695m$ • $M = 0.410kg$ • $O_m = 0.064m$ • $l_m = 0.545m$	• $I_{xx} = 0.4876$ • $I_{yy} = 0.4876$ • $I_{zz} = 0.34432$
Propellers	• $I_{xx} = M\left(\frac{{r_p}^2}{6} + \frac{{h_p}^2}{12} + {O_p}^2\right)$ • $I_{yy} = M\left(\frac{{r_p}^2}{6} + \frac{{h_p}^2}{12} + {l_p}^2 + {O_p}^2\right)$ • $I_{zz} = M\left(\frac{{r_p}^2}{3} + {l_p}^2\right)$		• $r_p = 0.2159m$ • $h_p = 0.018m$ • $M = 0.042kg$ • $O_p = 0.1081m$ • $l_p = 0.545m$	• $I_{xx} = 0.0524$ • $I_{yy} = 0.0524$ • $I_{zz} = 0.03492$
Sprayer	• $I_{xx} = M\left(\frac{{r_s}^2}{4} + \frac{{h_s}^2}{12} + {O_s}^2\right)$ • $I_{yy} = M\left(\frac{{r_s}^2}{4} + \frac{{h_s}^2}{12} + {l_s}^2 + {O_s}^2\right)$ • $I_{zz} = M\left(\frac{{r_s}^2}{2} + {l_s}^2\right)$		• $r_s = 0.005m$ • $h_s = 0.225m$ • $M = 0.187kg$ • $O_s = 0.1418m$ • $l_s = 0.545m$	• $I_{xx} = 0.2222$ • $I_{yy} = 0.2222$ • $I_{zz} = 0.15428$

* The total moment of inertia with full tank is:

$$\begin{bmatrix} I_{xx} = 1.9579 \\ I_{yy} = 1.9570 \\ I_{zz} = 0.9318 \end{bmatrix} \tag{D.1}$$

* The total moment of inertia with mass variation
  - $I_{xx} = 1.6175 + 0.03337 M_{tank}$
  - $I_{yy} = 1.6166 + 0.03337 M_{tank}$
  - $I_{zz} = 0.848 + 0.007736 M_{tank}$

Appendix E

Desired Roll and Pitch angles

$$\begin{aligned} \ddot{X} &= (cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi))\frac{U_1}{M} \\ \ddot{Y} &= (cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi))\frac{U_1}{M} \end{aligned} \tag{E.1}$$

I) θ_d, From $\ddot{X}$, $\frac{\ddot{X}M}{U_1} - (sin(\phi)sin(\psi)) = cos(\phi)sin(\theta)cos(\psi)$

$$\left(\frac{\ddot{X}M}{U_1} - (sin(\phi)sin(\psi))\right)\frac{1}{cos(\phi)cos(\psi)} = sin(\theta)$$

$$\boldsymbol{\theta_d = sin^{-1}\left(\frac{\ddot{X}M}{U_1} - (sin(\phi)sin(\psi))\right)\frac{1}{cos(\phi)cos(\psi)}}$$

II) ϕ_d, From $\ddot{X}$ and $\ddot{Y}$

$$\left(\frac{\ddot{X}M}{U_1} - (sin(\phi)sin(\psi))\right)\frac{1}{cos(\psi)} = cos(\phi)sin(\theta)$$

$$\left(\frac{\ddot{Y}M}{U_1} + (sin(\phi)cos(\psi))\right)\frac{1}{sin(\psi)} = cos(\phi)sin(\theta)$$

By equating the two equations above: $\left(\frac{\ddot{X}M}{U_1} - (sin(\phi)sin(\psi))\right)\frac{1}{cos(\psi)} = \left(\frac{\ddot{Y}M}{U_1} + (sin(\phi)cos(\psi))\right)\frac{1}{sin(\psi)}$

Multiply by $cos(\psi)sin(\psi)$

$$\frac{\ddot{X}Msin(\psi)}{U_1} - (sin(\phi)sin^2(\psi)) = \frac{\ddot{Y}Mcos(\psi)}{U_1} + (sin(\phi)cos^2(\psi))$$

$$\frac{\ddot{X}Msin(\psi)}{U_1} - \frac{\ddot{Y}Mcos(\psi)}{U_1} = (sin(\phi)sin^2(\psi)) + (sin(\phi)cos^2(\psi))$$

$$\frac{M}{U_1}(\ddot{X}sin(\psi) - \ddot{Y}cos(\psi)) = sin(\phi)$$

$$\boldsymbol{\phi_d = sin^{-1}\left(\frac{M}{U_1}(\ddot{X}sin(\psi) - \ddot{Y}cos(\psi))\right)}$$

Appendix F

Rung-Kutta(RK4) Method

The idea behind 4^{th} order Rung-Kutta (RK4) method is to solve a differential equations, in this system it is a 2^{nd} order ODE: $\ddot{Y} = f(\dot{Y}, Y, t)$ RK4 method is a weight average slope method for numerically solving a differential equation, it is different from Euler method of integration where the slope of a function seen here is simply multiplied by the time step in order to get the change in position and the change is taken and add to the current position. RK4 on the other hand uses a weight average of 4 slopes in order to obtain the change in position.

$Y_n = Y_{n-1} + h * k_1, where : -k_1 = f(Y_{n-1}, X)$ Euler Discretization

F.1 Working Procedure of RK4 for 2^{nd} order ODE

1. Given a second order ODE:$\ddot{Y} = f(\dot{Y}, Y, X)$
   The 2^{nd} order ODE is reduced to two 1^{st} order ODE. $\dot{Y} = Z$ so $\ddot{Y} = \dot{Z}$ where:$\dot{Y} = \dfrac{dY}{dX}$ and $\ddot{Y} = \dfrac{d^2Y}{dX^2}$

2. From the above step, there are two 1^{st} order ODE. $\dot{Y} = Z$ and $\ddot{Y} = \dot{Z}$ hence $\dot{Y} = g_1(X, Y, Z)$ and $\dot{Z} = g_2(X, Y, Z)$ are taken

3. So using the slops, we define an expression for the next state (Y_n) as the current state (Y_{n-1}) plus the weight average of these slopes multiplied by the sampling time h.

Slopes:

$$
\begin{aligned}
k_1 &= h * g_1(X_0, Y_0, Z_0) \\
l_1 &= h * g_2(X_0, Y_0, Z_0) \\
k_2 &= h * g_1(X_0 + \frac{h}{2}, Y_0 + \frac{k_1}{2}, Z_0 + \frac{l_1}{2}) \\
l_2 &= h * g_2(X_0 + \frac{h}{2}, Y_0 + \frac{k_1}{2}, Z_0 + \frac{l_1}{2}) \\
k_3 &= h * g_1(X_0 + \frac{h}{2}, Y_0 + \frac{k_2}{2}, Z_0 + \frac{l_2}{2}) \\
l_3 &= h * g_2(X_0 + \frac{h}{2}, Y_0 + \frac{k_2}{2}, Z_0 + \frac{l_2}{2}) \\
k_4 &= h * g_1(X_0 + h, Y_0 + k_3, Z_0 + l_3) \\
l_4 &= h * g_2(X_0 + h, Y_0 + k_3, Z_0 + l_3)
\end{aligned}
\tag{F.1}
$$

4. The slopes can be substituted and the ODE is discretized now:

$$
\begin{aligned}
Y_{n+1} &= Y_n + \frac{1}{6}(k_1 + 2 * k_2 + 2 * k_3 + k_4) \\
Z_{n+1} &= Z_n + \frac{1}{6}(l_1 + 2 * l_2 + 2 * l_3 + l_4)
\end{aligned}
\tag{F.2}
$$