



Design and Control of Autonomous Drone

Achieved by:

Mais Jamal **Youshaa Murhij** **Zain Ali**

Supervised by:

Dr. Iyad Hatem

July 2, 2017

A project report submitted in final fulfillment of the requirements for the award of the degree of Bachelor of Mechatronics Engineering for Distinguished
Program of Mechatronics for Distinguished
University of Tishreen

Declaration: We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Tishreen University or other institutions.
Signature:

Name :	Mais Jamal	Youshaa Murhij	Zain Ali
ID No. :	1010	1013	1002

Date : July 2, 2017

APPROVAL FOR SUBMISSION: We certify that this project report entitled **Design and Control of Autonomous Drone** was prepared by Mais Jamal, Youshaa Murhij and Zain Ali has met the required standard for submission in final fulfillment of the requirements for the award of Bachelor of Mechatronics Engineering for Distinguished at University of Tishreen.

Approved by,
Signature :

Supervisor: Dr. Iyad Hatem

Date : July 2, 2017

The copyright of this report belongs to the author as qualified by Intellectual Property Policy of University of Tishreen.
Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

©2017, Mais Jamal, Youshaa Murhij, Zain Ali . All right reserved.

Abstract With the huge advancement of sensors, motors and controllers, researchers now are more interested in developing new advanced flying vehicles in order to be used in research, commercial and military application. Quadcopter is a very common flying vehicle.
In this project, mathematical linear and nonlinear dynamic models of the quadcopter were formulated using both Newton-Euler and Lagrange methods. Depending on the two models to obtain a stability in attitude and a trajectory tracking, several control methods were used, The first method is a linear control method called "*PID*", the second is a nonlinear method called "*Backstepping*" control method, and the last one was "*Fuzzy logic*" control method..
A full created design of a quadcopter was presented and applied in order to test the used control methods on. Then, a full simulation on *MATLAB* was done. Also, the genetic algorithm was used to find the best control constants for each control method. and a manual tuning procedure was used to find the *PID* constants.
As a first step of the project, due to the hardware that was available, attitude stability on the three quadcopter angles (Roll, Pitch, yaw) using the mentioned methods was obtained. In order to continue for trajectory tracking, the next step was to build a Real-Time Indoor Positioning System based on image processing and neural networks to control the position of the quadcopter and to enforce the quadcopter to hover in a specific range of space.
After application, a comparison of stability between various types of control methods was made. It has been found that Backstepping method is more reliable.
Fuzzy logic control method was also used to control Altitude. Finally, a discussion of results was made.
Keywords: UAVs, Quadrotor, trajectory tracking, nonlinear control, Backstepping, PID, Neural Networks, Indoor Positioning , Fuzzy Altitude Control.

Contents

1 Introduction **11**
- 1.1 A Brief History of UAVs . . . 11
- 1.2 Quadcopter Overview . . . 13
- 1.3 Literature review . . . 14
- 1.4 Aims and Objectives . . . 15

2 Hardware Design **16**
- 2.1 Selecting Motors . . . 16
- 2.2 Selecting Propellers . . . 18
- 2.3 ESC connection . . . 19
- 2.4 ESC Calibration . . . 19
- 2.5 Frame Design . . . 20
  - 2.5.1 Frame layouts: . . . 20
  - 2.5.2 Frame Configurations: . . . 26
- 2.6 Material Strength and Stress Analysis . . . 28
- 2.7 Results and Notes . . . 34

3 Sensors **36**
- 3.1 Gyroscope and accelerometer sensors: . . . 36
- 3.2 Barometric Pressure Sensor . . . 38
- 3.3 Magnetometer Sensor (Compass) . . . 39

4 Quadcopter Dynamics **40**
- 4.1 Modeling . . . 40
  - 4.1.1 Quadcopter Modeling Using Euler-Newton method 41
  - 4.1.2 Quadcopter Modeling Using Euler-Lagrange method 45

5 Control Methods **51**
- 5.1 Implementing PD controller for position tracking . . . 51
- 5.2 Applying Lyapunov Stability Criteria on the Dynamic model 55
- 5.3 Implementing a Backstepping Controller . . . 56

5.4 Applying Backstepping Nonlinear Control Method 60
5.5 Results 62

6 Project Progress **68**

7 Experimental Test of Control Methods **69**
7.1 PID Control: 69
7.2 Implementing Genetic Algorithm 71
7.3 Comparison between PID and Backstepping Responses . . 73
7.4 Yaw and Altitude Responses to Backstepping Control . . . 75

8 Obstacle Detection and Collision Avoidance **77**

9 Building a Low-Cost Indoor positioning system to control the trajectory of the quadcopter **79**
9.1 Detecting the quadcopter using image processing techniques 80
9.2 Building a 3D positioning system using two IP Cameras and Neural Networks 82
9.2.1 Neural network structure: 82
9.2.2 Training Algorithm- Levenberg-Marquardt: 84
9.2.3 Validation and testing: 85
9.2.4 Real-time results: 87

10 Fuzzy Altitude Control for Quadrotor with Dynamic Feed-forward Compensation **91**

Appendices **96**

A Stress Analysis Figures **97**

B List of Symbols **103**

List of Figures

2.1 Brushless Motor 16
2.2 Types of Propellers 17
2.3 Selecting Propellers 18
2.4 ESC connection schematic 19
2.5 Tricopter 21
2.6 A Quadcopter 22
2.7 A Hexacopter 22
2.8 An Octacopter 23
2.9 Frame Materials 25
2.10 X-frame 27
2.11 H-frame 29
2.12 Final Drone Design using CATIA Software: 30
2.13 Force Distribution 31
2.14 Our Final Quadcopter 34

3.1 Orientation of Axes of Sensitivity and Polarity of rotation . 38

4.1 Mode"+" of the quadcopter 40
4.2 Quadcopter movements a) Thrust, b)Roll, c)Pitch, and d)Yaw 41
4.3 Earth frame and Body frame 42

5.1 PID Control Simulink Model 52
5.2 PID Results (X,Y,Z and angles) 53
5.3 PID Position Control Feedback Loop 54
5.4 Lyapunov Parameters 60
5.5 Lyapunov Simulink Model 61
5.6 Response to fixed input values $x = 1000, y = 1000, z = 10$: . . 63
5.7 Response to spiral trajectory - 3D View 64
5.8 Response to spiral trajectory - Coordinates 65
5.9 Response to rectangular trajectory - 3D View 66
5.10 Response to rectangular trajectory - Coordinates 67

7.1 PID Tuning Implementation 70
7.2 Simulink model for quadcopter 71
7.3 GA Attitude Step Response due to Settling Time criteria . . 72
7.4 GA Attitude Step Response due to Settling Time criteria . . 73
7.5 GA Attitude Step Response due to Settling Time criteria . . 73
7.6 Backstepping Response Error over set of data 74
7.7 Backstepping control response of Yaw 75
7.8 Backstepping control response of Altitude - Good Constants 76
7.9 Backstepping control response of Altitude - Bad Constants 76

8.1 Collision Avoidance - Distance Zones 78

9.1 3D representation of the Positioning system 80
9.2 (a) Original Camera Frame - (b) Frame after converting to HSV- (c) Thresholded image - (d) Final image after Eroding and Dilating 81
9.3 NN Positioning System structure 83
9.4 Main structure of a neural network 83
9.5 A neural network with 4 hidden neurons 84
9.6 Pure Linear Function - Hyperbolic Tangent Sigmoid Function 84
9.7 Neural network structure (MATLAB) 85
9.8 Regression related to X and Y axes NNs 86
9.9 Regression related to Z axis NN 86
9.10 Convergence and Training Performance on X-data 87
9.11 Convergence and Training Performance on Y-data 87
9.12 Convergence and Training Performance on Z-data 87
9.13 Current Position Vs Set Position along X-axis(cm) 88
9.14 Current Position Vs Set Position along y-axis(cm) 88
9.15 Current Position Vs Set Position along Z-axis(cm) 89
9.16 a) Simple flight trajectory - b) X-Y plan projection[X=const] - c) X-Y plan projection[Y=const] - d) Y-Z plan projection . 89

10.1 Fuzzy logic Altitude controller System Diagram 92
10.2 The membership functions of the Fuzzy System Z error input 93
10.3 The membership functions of the Fuzzy System Z error rate input 93
10.4 The membership functions of the output U 93
10.5 Rules Surface 94
10.6 Fuzzy Altitude Response Test 95

A.1 Force Distribution and Fixing Methods 98
A.2 Static Case Solution.1 - Deformed mesh.2 99

A.3 Static Case Solution.1 - Von Mises stress (nodal values).2 . . 99
A.4 Static Case Solution.1 - Deformed mesh.1 100
A.5 Static Case Solution.1 - Von Mises stress (nodal values).1 . . 100
A.6 Static Case Solution.1 - Translational displacement vector.1 101
A.7 Static Case Solution.1 - Estimated local error.1 101
A.8 Static Case Solution.1 - Stress principal tensor symbol.1 . . 102

List of Tables

2.1 Properties of materials used in the aerospace industry . . . 24
2.2 MESH . . . 29
2.3 ELEMENT TYPE . . . 29
2.4 ELEMENT QUALITY . . . 29
2.5 Materials . . . 29
2.6 STRUCTURE Computation . . . 30
2.7 FACTORIZED Computation . . . 32
2.8 Minimum pivot . . . 32
2.9 Translational pivot distribution . . . 32
2.10 Equilibrium . . . 33
2.11 Global Sensors . . . 33

7.1 PID Tuning Results through Test Methods . . . 70
7.2 Lyapunov Constants.1 . . . 72
7.3 Lyapunov Constants.2 . . . 72
7.4 Lyapunov Constants.3 . . . 72

9.1 Training Parameters . . . 85

10.1 Rule Set . . . 94

Chapter 1

Introduction

1.1 A Brief History of UAVs

One of the earliest explorations of the concept of the combat drone was by Dr. Lee De Forest, an early inventor of radio devices, and U. A. Sanabria, a TV engineer. They presented their idea in an article in a 1940 publication of Popular Mechanics.[1] The modern military drone as known today was the brainchild of John Stuart Foster Jr., a nuclear physicist and former head of the Lawrence Livermore National Laboratory (then called the Lawrence Radiation Laboratory). In 1971, Foster was a model airplane hobbyist and had the idea this hobby could be applied to building weapons. He drew up plans and by 1973 DARPA (Defense Advanced Research Projects Agency) built two prototypes called "Praeire" and "Calere", they were powered by a modified lawn-mower engine and could stay aloft for two hours while carrying 28-pounds of load.[2]

UAVs were not used at wars until 1898, this technology was used in the Spanish-American War, resulting in the first military aerial reconnaissance photos. However, after wars researchers started thinking about UAVs as a useful tool not just in military but also in other fields. They developed a new type of UAVs called multi-copter or multi-rotor helicopters. For instance, a quadcopter is an aerial vehicle propelled by four rotors. The propellers have a fixed pitch, which makes the quadcopter mechanically simpler than an ordinary helicopter. However, the quadcopter is inherently unstable, and therefore, its control is rather difficult. The progress on the field of control engineering allowed to deal with inherent

instability of the Quadrotors, and therefore, they have started to appear in military, security and surveillance systems.

Nowadays, the quadcopters can perform quick and complex maneuvers, navigate autonomously in structured and unstructured environments and cooperate in manipulation and transportation tasks. However, the commercial Quadrotor helicopters are too expensive to be used by students or small research teams. Although there exist several quadcopter toys, these are too small to carry necessary sensor equipment. In recent years, several community projects aimed to develop an affordable Quadrotor helicopter have appeared. However, these projects are still in progress and have not filled the gap between expensive commercial platforms and sensorless toys.

In the autumn of 2010 an affordable quadcopter, equipped with the necessary sensors and with a suitable software interface has appeared on the market. Originally intended as a high-tech toy for augmented reality games, the drone quickly caught attention of universities and research institutions, and nowadays is being used in several research projects. For instance, machine learning approaches were applied to predict the position errors of the UAV following a desired flight path. Other research groups used the drone as an experimental platform for autonomous surveillance tasks, human-machine interaction, and even as a sport assistant, which aids the athletes by providing them external imagery of their actions.[3]

Applications of Drones (UAVs) unmanned aircraft systems (UAS), have practical applications. Here are a few to consider: [4]
1. They can save lives. In natural and manmade disasters, UAS can be positioned to survey damage, locate stranded and injured victims, and assess ongoing threats without risking the safety of rescue teams and first-responders.
2. They can support law enforcement. UAS can be used to search for lost children, provide tactical surveillance and suspect tracking, assist in accident investigations, and monitor large crowds.
3. They can contribute to safe infrastructure maintenance and management, considering the difficulty of inspecting the underside of a bridge or the top of a skyscraper, not to mention the costs and risks. With UAS, scaffolding, cranes, or harnesses are not required. Just deploy the system to assess the structure's condition remotely.

4. They can streamline agriculture management. Using a crop management system to observe, measure, and respond to variability in individual plants, farmers can target areas requiring attention. By pinpointing these areas, farmers can provide care only where needed—improving yield, conserving resources, and avoiding waste.
5. They can give media access to hard-to-reach places. Aerial photography for a news broadcast or a blockbuster film can be efficiently, economically, and safely captured by a UAS.
6. Hurricane Hunting. Drones can charge into the heart of a storm without risking human life and limb. That is one reason NASA, the National Oceanic and Atmospheric Administration (NOAA), and Northrop Grumman teamed up on a three-year, $30-million experiment to use long-range Unmanned Aerial Vehicles (UAV) to spy on storms as they evolve.[5]
7. 3-D Mapping. Small, lightweight drones may look like simple model airplanes, but they can survey landscapes with thousands of digital images that can be stitched together into 3-D maps. Military and other government satellites produce similar maps, but emerging UAV technology can put that capability in the hands of small companies and individuals, to be customized and used for a seemingly endless variety of applications.

1.2 Quadcopter Overview

A **Quadcopter,** also called a **quadrotor helicopter** or **quadrotor**, is a multirotor helicopter that is lifted and propelled by four rotors. Quadcopters are classified as rotorcraft vehicle, opposed to fixed-wing aircraft, because their lift is generated by a set of rotors (vertically oriented propellers).

Quadcopters generally use two pairs of identical fixed pitched propellers; two clockwise (CW) and two counter-clockwise (CCW). These use independent variation of the speed of each rotor to achieve control. By changing the speed of each rotor it is possible to specifically generate a desired total thrust; to locate for the center of thrust both laterally and longitudinally; and to create a desired total torque, or turning force.

Quadcopters differ from conventional helicopters which use rotors which are able to vary the pitch of their blades dynamically as they move around the rotor hub. In the early days of flight, quadcopters were seen as possible solutions to some of the persistent problems in vertical flight; torque-induced control issues (as well as efficiency issues originating from the tail

rotor, which generates no useful lift) can be eliminated by counter-rotation and the relatively short blades are much easier to construct.

Advances in electronics allowed the production of lightweight flight controllers, accelerometers, global positioning system (GPS) and cameras. This resulted in a rapid proliferation of small, consumer quadcopters along with other multi rotor designs. Quadcopter designs also became popular in unmanned aerial vehicle (UAV or drone) research. With their small size and maneuverability, these quadcopters can be flown indoors as well as outdoors. [7]

At a small size, Quadcopters are cheaper and more durable than conventional helicopters due to their mechanical simplicity. Their smaller blades are also advantageous because they possess less kinetic energy, reducing their ability to cause damage. For small-scale Quadcopters, this makes the vehicles safer for close interaction. It is also possible to fit Quadcopters with guards that enclose the rotors, further reducing the potential for damage. However, as size increases, fixed propeller Quadcopters develop disadvantages over conventional helicopters. Increasing blade size increases their momentum. This means that changes in blade speed take longer, which negatively impacts control.

At the same time, increasing blade size improves efficiency as it takes less energy to generate thrust by moving a large mass of air at a slow speed than by moving a small mass of air at high speed. Therefore, increasing efficiency comes at the cost of control. Helicopters do not experience this problem as increasing the size of the rotor disk does not significantly impact the ability to control blade pitch.

Due to their ease of construction and control, Quadcopter aircraft are frequently used as amateur model aircraft projects.

1.3 Literature review

The spread of drone technology in the 21st century has sparked a revealing, if still rather limited, literature that has sought to examine the modalities, stabilization and techniques of the deployed aircraft. There are at least two main aspects to highlight from this body of work.
A first point relates to stabilization, In a research article by Lucas M.A,

Willian C.R, Paulo E.S and Renato A.A(2012)[8], LQR controllers was used. This controller is robust and produces a very low steady state error, but with a big transition delay and using six feedback gains, that makes them a bad choice when the system needs fast parameters update and has no direct access to all states of the plant.
In another research article by Atheer L.S, M. Moghavvemi, Haider A. F. Mohamed and Khalaf S.G (2010)[9], the stability in flying the Quad-rotor flying object was obtained after developments of a PID (proportional-integral-derivative) control method and the model was a linear model. As a result, the model of the vehicle was very realistic. Meanwhile in another research paper by Samir B and Roland S (2005)[10] the stabilization was obtained using another method that depends on backstepping control of "OS4" VTOL system and lyapunov functions and the results were impressive. The system was more stable and the main reason is that their model was a nonlinear model which gived more precision.
The second point relates to the way of control. On one hand, in an article by Ankyda Ji and Kamran Turkoglu (2015) [11], they used a Receiver and RC controller in order to let the user control the vehicle. Data logging was originally done using the Xbee Pro 900 RPSMA on the 900MHz frequency. The problem was that the Xbee has a low reliability in transmission from packet loss. On the other hand, in many other articles researchers used an Autonomous Quadrotor flight, and the control was only by the controller that the quadcopter has. In this case, the study is harder but more realistic and precise.
After a review of the previous articles, it was been decided to build an Autonomous flight quadcopter, which will receive a position feedback information from a central computer using a WiFi-indoor connection.

1.4 Aims and Objectives

The main goal was to design an Autonomous Quadrotor Drone and analyze its dynamical system in order to control it using PID control method and Backstepping control based on Lyapunov Stability Criteria, in addition to fuzzy logic control. In this project, we aimed to build an indoor positioning system to improve the drone's autonomous navigation using image processing techniques and Neural networks features. The final goal of the project is to make the drone hold in space autonomously.

Chapter 2

Hardware Design

2.1 Selecting Motors

Motor specification such as power, thrust, rpm... , are provided by either the seller or manufacturer. Here is an example of the MT2213-935KV Brushless motor on Hobbyking.com.

Figure 2.1: Brushless Motor



Spec:KV(RPM/V): **935** Lipo cells: **2-4s** Max Surge Watts(W): **200** Max.current(10s): **15 amps** No Load Current: **0.4 amps** Intemal Resistance: **0.180 ohm** Number of Poles: **14** Dimensions(Dia.xL): **28 x 26 mm (not counting the propeller shaft)** Motor Shaft: **3mm (not the propeller shaft)** propeller shaft: **Slotted (DJI) style 8mm round (7mm flat to flat) stepped down to 6mm and a M6 propeller nut** Prop: **10x4.5(3S) 8 x 4.5 (4S)** Weight: **55g** bolt hole spacing: **16mm * 19mm** [12]

To choose a motor, we first need to know the weight of the quadcopter and then, to work out the thrust required to lift the quadcopter. A general rule is that the drone should provide a thrust equal to twice the weight of itself.
A rule of thumb is **Required Thrust per motor = (Weight x 2) / 4**
In the case of our quadcopter, the estimated weight was around *1 kg*. using the previous equation, the total thrust would be *2 kg* and *500 g* per motor. The estimated weight includes weights of the quadcopter frame, motors, propellers, connectors, controllers and battery.
The lighter the multicopter, the easier to control it. So, we had to choose a battery with a good capacity and a light weight. When it comes to efficiency, battery capacity and light are not the only effecting factors. Motors efficiency and power are also important factors.. When choosing motors, apart from motor KV and thrust, we also need to look at Watt's and efficiency.
The same applies to brushless motor, the higher efficiency the better. A 70% efficient motor produces 70% power and 30% heat. A 90% efficient motor produces 90% power and 10% heat. With less efficient motors not only we are wasting a lot of power (thus flight time) but, also we are getting a smaller thrust on full throttle. In the case of inefficient motors, the response time will suffer from delay. It will take the motors more time to change the rpm speed and this will influence the stability of the quadcopter, and it will be less stable.[13]

Figure 2.2: Types of Propellers



2.2 Selecting Propellers

A quadcopter uses two clockwise (CW) and two counter-clockwise (CCW) propellers. Propellers are classified by length and pitch. For example, 9×4.7 propellers are 9 inch long and has a pitch of 4.7.

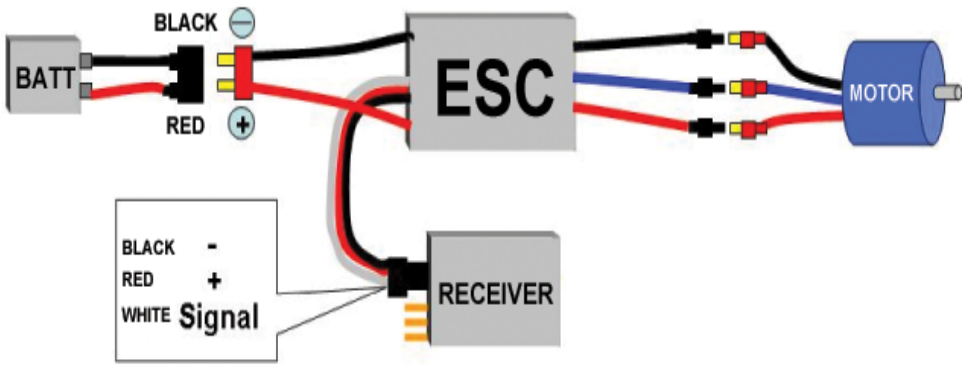
Figure 2.3: Selecting Propellers



Generally, increasing propeller pitch and length will draw more current. Also the pitch can be defined as the travel distance of one single propeller rotation. In a nutshell, the higher the pitch, the slower the rotation. Although this will increase the vehicle speed, it will use more power.
On the propeller length and pitch, there should be a good balance. Generally, a propeller with a low pitch number, can generate more torque and the motors will pull less current. Lower pitch propellers will also improve stability.
The higher the pitch propeller move, the greater the amount of air, which could create turbulence and cause the aircraft to wobble during hovering. In case of that, choosing a lower pitched propeller is better. When it comes to the length, propeller efficiency is closely related to the contact area of a propeller with air, so a small increase in propeller length, will increase the propeller efficiency. (pretty much like swimmers with larger hands and feet can swim faster, but also more tiring for them) .
A smaller propeller is easier to stop or speed up while a larger propeller takes longer to change speeds (inertia of movement). Smaller propeller also means it draws less current, that is why Hexacoptors and Octocopters tend to use smaller propellers than quadcopter of similar size. For larger quadcopters that carry payloads, large propellers and low-kv motors tend to work better. These have more rotational momentum, and will more easily maintain your aircraft's stability.

2.3 ESC connection

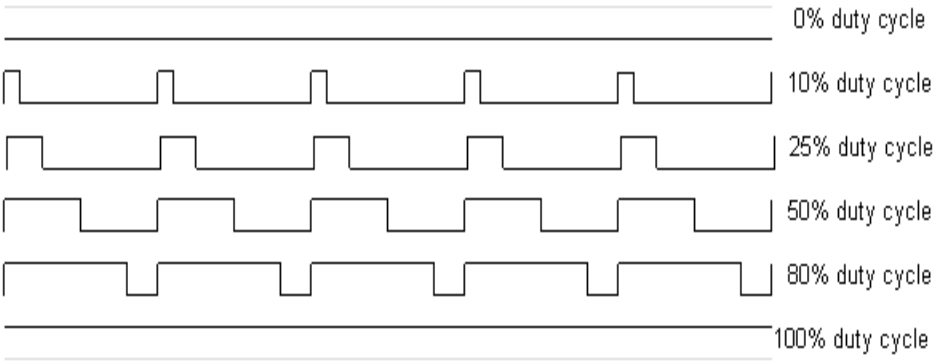
Figure 2.4: ESC connection schematic



Usually ESCs need a voltage higher than the one provided by the Arduino from his 5V pin: typically they need 3 LiPo cells (around 12V). To achieve that all the circuit must be powered from an external power supply connected directly to the ESC and not via the Arduino, which will be powered by the BEC circuit of the ESC. To make that happen it is enough to connect the red and black of the control connector to the 5V and GND of the Arduino board. As shown in figure 2.4.

2.4 ESC Calibration

Calibration, in terms of ESCs, means to set the max and min speeds of the motor in relation to the max and min width of the PWM signal sent by the Arduino. A PWM signal is simple a square wave signal consisting of high and low (5v and 0v) signals for certain durations. Some sample PWM waves are shown below.



The PWM signal read by the ESC is the same type as a servo signal, meaning the Adruino can be used to calibrate and control the ESCs. The ESC sets the speed of the motor depending on the ratio of high to low signals. Calibration involves programming the ESC to understand the PWM waves corresponding to the stop and maximum speeds of the motor.
The default signal range for most servo motors and ESCs is a high signal width between 1000 and 2000 microseconds over a repetition period of 20 milliseconds (assuming a 50hz PWM signal). However, for the quadcopter we want the range to be as wide as possible to allow a greater incremental control of the motor. So, we calibrated the ESCs to read a signal width from 700 to 2000 microseconds with 700 being the stop speed and 2000 being the max speed. We found that the ESC could not read a signal lower than 700 microseconds.
Calibrating the ESCs was quite simple. To enter programming mode, the maximum servo signal (2000 microseconds) is sent to the ESC, the ESC is powered on and waits for two seconds, then the minimum servo signal is sent (700 microseconds). Once the ESC emits a series of confirmation beeps (special wave signals sent to the motor to emit beeping sounds), the ESC is calibrated.

2.5 Frame Design

Multicopter frames are available as bare frame kits, almost ready to fly copters but we decided to build a quadcopter from scratch.

2.5.1 Frame layouts:

There are several popular frame layouts for multi-copters:

Tricopter: It has three motor/propeller units (see Fig. 2.5) and it is not very popular.- Pros of a Tricopter:

1. it is simple and analogue stabilized with the gyros (no firmware/ programming), collapsible/easy transport.

Cons of a Tricopter:

1. More strain on each motor and propeller (the bearings takes a lot more beating).
2. Not that acrobatic as a quadcopter can be.
3. It has an additional mechanical part on the yaw.

Figure 2.5: Tricopter



Quadcopter: It has four motor/propeller units (see Fig. 2.6) and it is very popular.
Pros of a quadcopter:

1. Quadcopters are symmetrical and embody the simplest principle of operation for controlling roll, pitch, yaw and motion.
2. Simply varying the speed of each of the motors provides full 3D motion and rotation and hover capability.
3. Relatively cheap to manufacture.

Figure 2.6: A Quadcopter



4. Great maneuverability.
5. Powerful enough to add accessories.
6. Greater thrust and power versus Tricopters.

Cons of a quadcopter:

1. Not as powerful as a Hexacopter or Octocopter.

Hexacopter: It has six motor/propeller units (see Fig. 2.7) and it is popular.

Figure 2.7: A Hexacopter



- Pros of a Hexacopter:

1. Greater overall power, speed and elevation.
2. Safety provided through additional motors.
3. Higher overall payload.
4. Great control and flight speed.

- Cons of a Hexacopter:

1. Priced higher than a quadcopter.
2. Larger in size, making the copter harder to fly in tight spaces.
3. Motor parts are more expensive if they need to be replaced.

Octocopter: It has eight motor/propeller units (see Fig. 2.8) and it is popular.

Figure 2.8: An Octacopter



Pros of an Octocopter:

1. Very fast and agile.
2. Reach exceptionally high elevations.
3. Extremely powerful.
4. Can hold heavy camera equipment.

5. Very safe and stable.

Cons of an Octocopter:

1. Big in size.
2. Expensive compared to the hexacopter and quadcopter.
3. Battery life is often far less.

Frame material: The frame is what ties everything else in an AeroQuad together. Its purpose is to attach the motors to each other, and to support the electronics. Although that sounds simple,the frame has to be as light as possible to be able to fly. The frame also needs to be as stiff as possible to provide good handling characteristics. These two goals are contradictory to each other, as the stiffer you make a frame, the heavier it will be and this is where material selection comes into play.
Most professional grade Multicopters are made from fiberglass and carbon fiber cut tubing and plate. However, many other grade Multicopters are made from other materials such as plates of Plexiglas (acrylic) or Balsa wood or even made with injection molded plastic parts and aluminum tubing. (see Fig. 2.9)

Table 2.1 shows the properties of materials that are commonly used both in the aerospace industry and in the RC Hobby industry.

Table 2.1: Properties of materials used in the aerospace industry

Material	Density (lb/cuin)	Young's Modulus (Msi)	Tensile Strength (Ksi)
Carbon Fiber	0.05	9.3	120
Titanium	0.16	16.82	130
Aluminum	0.10	10.3	15-75
Fiberglass	0.07	2.7	15-50
Steel	0.29	29	30-290
Polycarbonate	0.05	0.75	8-16
Acrylic	0.04	0.38	8-11
Balsa	0.0027-0.0081	0.16-0.9	1-4.6

Figure 2.9: Frame Materials



Specific material characteristics:
Carbon fiber:
Carbon Fiber is very light, very stiff, and very strong, but is also brittle and can shatter on impact, it is also hard to machine and to glue to. unfortunately, it is also very expensive and can be dangerous to machine as the fibers are toxic to breathe.
Fiberglas:

Fiberglass is produced in pretty much the same way as carbon fiber. However, fiberglass is somewhat heavier, softer, and weaker than carbon fiber. It is however easier to machine, as less precautions need to be taken when cutting it, and it is much cheaper. Fiberglass has gotten a bad rap over the years, although a lot of that has to do with its difficulty in being repaired.
Aluminum:
Aluminum has been used in aircraft since WW2, and is still being used today. It is also a great material for Aero-Quads. It is readily available and fairly inexpensive. It is also easy to machine, as carbide and steel tools can machine it fairly easily. Because it is a homogeneous material three dimensional shapes can be machined from it as well, something that can not really be done with composites like carbon fiber or fiberglass. There are a fair range of different aluminum alloys available, and although they generally have similar densities and stiffness's, the strength can vary greatly.
Balsa Wood:
Balsa wood has been used in RC Aircraft forever, and for good reason. It is inexpensive, extremely light, and fairly stiff and strong. It is also very easily machined and readily available at most hobby shops. This means that replacement parts can be easily obtained without great expensive. However due to its low density it also requires a fairly large cross section to produce arms that are reasonably stiff and this will block some of the rotors downwash and reduce thrust.
Acrylic:
Acrylic, also known as Plexiglas, is readily available and fairly inexpensive. it is a Shatterproof, homogeneous material because it is very pure. It is also lighter than aluminum and easier to manufacture using a 2D laser cutting machine, when it is designed as plates.
According to the previous properties, acrylic was chosen to be the material of our quadcopter and it will be manufactured as plates.

2.5.2 Frame Configurations:

There are several popular configurations for quadcopters:

1. **X-frame or Square frame:**

   It has frame arms connecting between motors (and usually intermediate frame arms supporting a central frame), (see Fig. 2.10). It is a good all-around choice because:

   (a) It is a symmetrical design, motor propeller units are spaced the same distance from each other and from the center of the copter.

(b) The weight (or mass) is also concentrated in the center of the copter.

(c) This symmetry means all motor propeller units can be treated as equals and control forces distributed equally and this will simplify the quadcopter mathematical model and control.

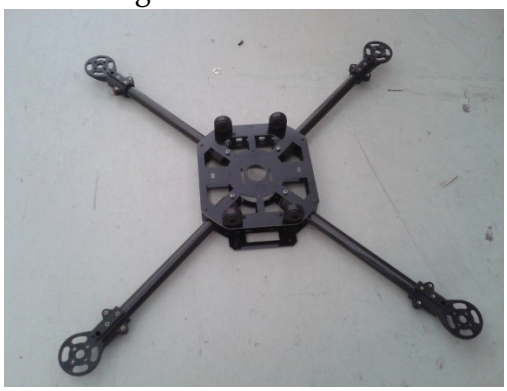
(d) For photo, video or FPV use, the camera(s) must be placed under the copter in order to have visual clearance of the propellers.

(e) This lowers the center of mass increasing forces required for pitching and rolling the copter reducing performance somewhat.

(f) It also requires an extended landing gear (which must also not visually interfere with the camera) to clear the camera / gimbal.

(g) With a camera the X frame also presents a taller profile which is affected more by wind and gusts.

Figure 2.10: X-frame



2. **"Dead Cat" or H frame:**

It has a variation on X which in which the frame arms are further apart in the front and the back than they are side to side (see Fig. **??**). It is specifically designed for photo / video applications and permits a front mounted camera.

1. H design implies a central frame member interconnecting front and back members that mount the propeller/motor units on each end.

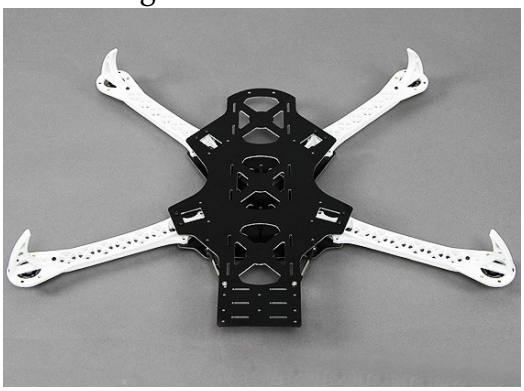
2. Originally H might have had a single central tube and could be subject to twist, but for our examples we assume a rigid frame.
3. Dead Cat is named for a German enthusiast who turned his stuffed pet cat into a quadcopter.
4. Dead Cat infers that clearance between front and back propellers will be larger than side to side clearance.
5. Since the camera is in line with the frame, it presents a minimal side profile and is less affected by wind and gusts.
6. On an H or Dead Cat a short landing gear is normally used and mechanical complexity and weight is somewhat reduced.
7. The mass is still centered and is narrow side to side but long front to back with the battery counter balancing the forward camera.
8. This long, thin mass means that more force is required for pitch control than for roll control.
9. Prop/motor units are close together on the sides but wider apart in the front and back to clear the frame and the cameras view.
10. This also means more force is necessary to change pitch than is required for roll.
11. The combination of these two effects definitely requires that pitch be given more "authority" than roll.

In this project the x-frame was chosen regarding to its advantages in simplifying model and control of the quadcopter.
As a final result we chose to build a quadcopter made as hollow plates of Acrylic(Plexiglas) with the shape of X-frame. the final design of the quadcopter is shown in figure 2.12.

2.6 Material Strength and Stress Analysis

A Computer Aided Design (CAD) of the quadcopter was made on CATIA software (see Fig. 2.12). All forces, torques and constraints that the quadcopter will be subjected to, were added. A full strength and stress analysis was done by the software using the finite element theory. Analyses showed high-stress and high-displacement locations.

Figure 2.11: H-frame



??

Analysis of Frame Material Strength (Finite Elements)

Table 2.2: MESH

Entity	**Size**
Nodes	1283
Elements	3076

Table 2.3: ELEMENT TYPE

Connectivity	**Statistics**
TE4	3076 (100.00%)

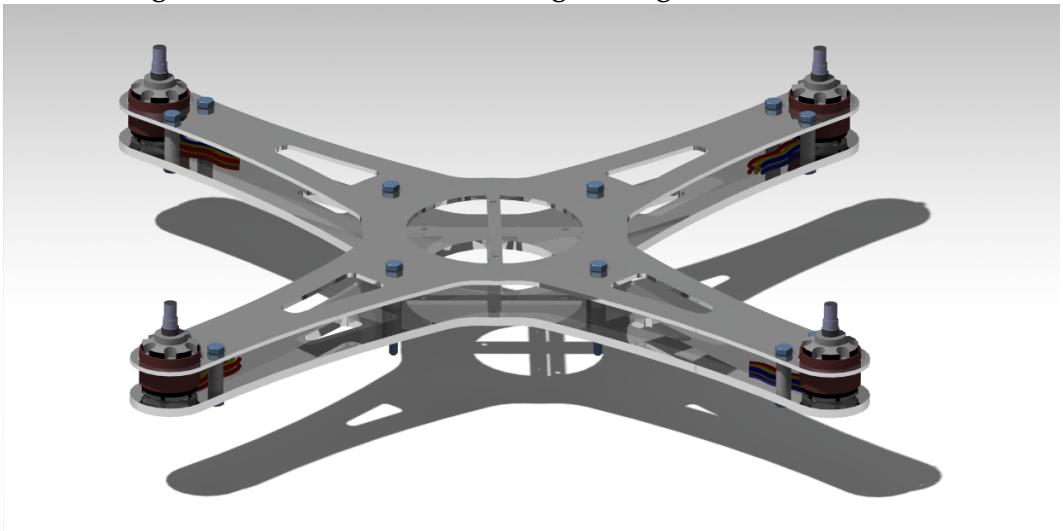
Table 2.4: ELEMENT QUALITY

Criterion	**Good**	**Poor**	**Bad**	**Worst**	**Average**
Stretch	3066 (99.67%)	10 (0.3%)	0 (0.0%)	0.276	0.490
Aspect Ratio	1226 (39.86%)	1816 (59.0%)	34 (1.1%)	6.262	2.783

Table 2.5: Materials

Material	Plexiglass
Young's modulus	$2.9e+009N_m2$
Poisson's ratio	0.4
Density	$1200kg_m3$
Coefficient of thermal expansion	$7e-005_Kdeg$

Figure 2.12: Final Drone Design using CATIA Software:



Yield strength	$5.5e+007N_m2$

Static Case:
Boundary Conditions

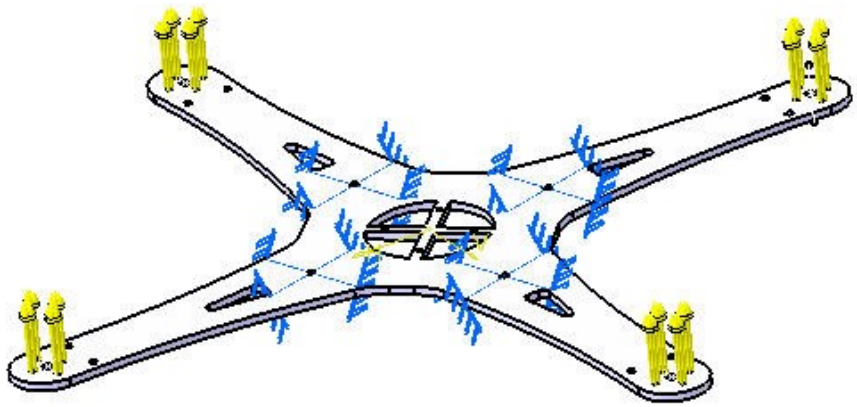
Table 2.6: STRUCTURE Computation

Number of nodes	:	1283
Number of elements	:	3076
Number of D.O.F.	:	3849
Number of Contact relations	:	0
Number of Kinematic relations	:	0
Linear tetrahedron	:	3076

RESTRAINT Computation
Name: Restraints.1: Number of S.P.C: 78

LOAD Computation
Name: Loads: Applied load resultant :
Fx = 1 . 023e-010 N
Fy = -1 . 354e-010 N
Fz = 1 . 000e+001 N
Mx = -1 . 199e-003 Nxm

Figure 2.13: Force Distribution





My = 1 . 199e-003 Nxm
Mz = -6 . 121e-011 Nxm

STIFFNESS Computation

Number of lines : 3849
Number of coefficients : 59484
Number of blocks : 1
Maximum number of coefficients per bloc : 59484
Total matrix size :0. 70 Mb

SINGULARITY Computation

Restraint: Restraints.1
Number of local singularities : 0
Number of singularities in translation : 0
Number of singularities in rotation : 0
Generated constraint type : MPC

CONSTRAINT Computation

Restraint: Restraints.1 Number of constraints : 78
Number of coefficients : 0

Number of factorized constraints : 78
Number of coefficients : 0
Number of deferred constraints : 0
FACTORIZED Computation
Method : SPARSE
Number of factorized degrees : 3771
Number of supernodes : 527
Number of overhead indices : 15717
Number of coefficients : 147159
Maximum front width : 93
Maximum front size : 4371
Size of the factorized matrix (Mb) : 1 . 12273
Number of blocks : 1
Number of Mflops for factorization : 7 . 314e+000
Number of Mflops for solve : 6 . 075e-001
Minimum relative pivot : 7 . 354e-004
Minimum and maximum pivot

Table 2.7: FACTORIZED Computation

Value	Dof	Node	x (mm)	y (mm)	z (mm)
4.5746e+006	Tz	774	-7.5973e+000	-1.3164e+002	4
7.0910e+009	Tz	119	-1.8986e+001	2.3228e+001	4

Table 2.8: Minimum pivot

Value	Dof	Node	x (mm)	y (mm)	z (mm)
5.9382e+006	Tz	177	-2.0897e+002	-1.7971e+001	0
1.0497e+007	Tz	1277	2.1000e+002	8.0000e+000	2
1.1620e+007	Tz	453	1.3794e+001	-9.5512e+000	4
1.4153e+007	Tz	1075	-1.4657e+001	-3.0142e+001	0
1.7868e+007	Tz	1278	-2.2201e+002	-2.9551e+000	4
1.8026e+007	Tz	1024	-7.0267e+001	7.8909e+000	0
2.2059e+007	Ty	1265	7.2501e+000	-7.2501e+000	0
2.2437e+007	Tz	1283	2.2124e+002	4.7452e+000	1.8

Table 2.9: Translational pivot distribution

Value	Percentage

10.E6 –> 10.E7	5.3036e-002
10.E7 –> 10.E8	3.0761e+000
10.E8 –> 10.E9	8.1543e+001
10.E9 –> 10.E10	1.5327e+001

DIRECT METHOD Computation Name: Static Case Solution.1: Restraint: Restraints.1 Load: Loads.1: Strain Energy : 8.667e-005 J

Table 2.10: Equilibrium

Components	**Applied Forces**	**Reactions**	**Residual**	**Relative Magnitude Error**
Fx (N)	1.0228e-010	-1.0865e-010	-6.3705e-012	1.4074e-013
Fy (N)	-1.3540e-010	1.7173e-010	3.6331e-011	8.0261e-013
Fz (N)	1.0000e+001	-1.0000e+001	7.1879e-011	1.5879e-012
Mx (Nxm)	-1.1986e-003	1.1986e-003	1.2213e-011	1.1834e-012
My (Nxm)	1.1986e-003	-1.1986e-003	2.9454e-011	2.8539e-012
Mz (Nxm)	-6.1205e-011	5.7561e-011	-3.6444e-012	3.5312e-013

Table 2.11: Global Sensors

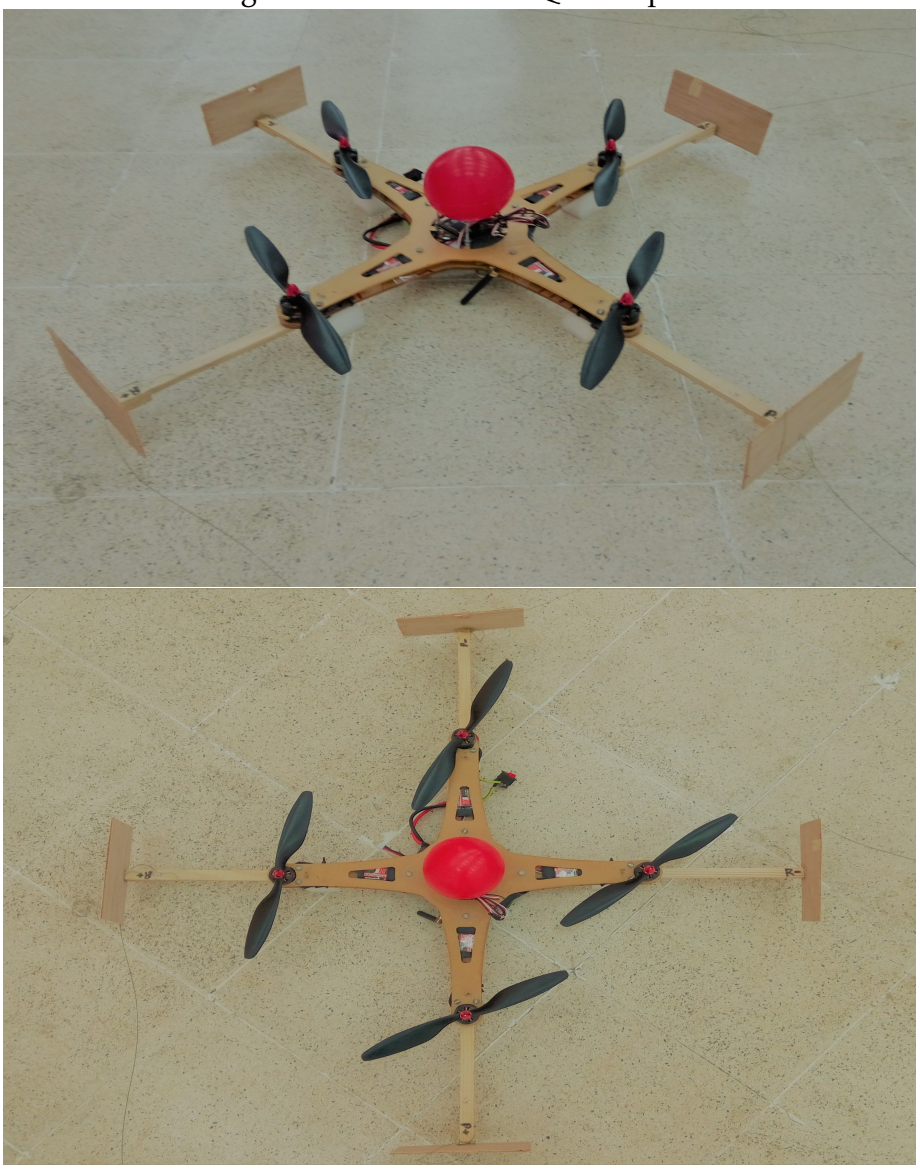
Sensor Name	**Sensor Value**	**Sensor Name**
Energy	8.667e-005J	Energy
Global Error Rate (%)	66.389862061	Global Error Rate (%)

Detailed figures are listed in Appendix A.

2.7 Results and Notes

The previous analysis of stress and strength showed that a 2mm Plexiglas plate will have a displacement on its arms. So, a new analysis on a thicker double-plate quadcopter frame was done, which showed that the plates will not have a displacement anymore. The plates were linked with appropriate 3M and 4M screws and special links.

Figure 2.14: Our Final Quadcopter



Chapter 3

Sensors

3.1 Gyroscope and accelerometer sensors:

MPU-6050 Gyro/Acc
The MPU-60X0 is the world's first integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package [MPU-6050 Product Specification Sheet].
Features:
1.Gyroscope Features:
The triple-axis MEMS gyroscope in the MPU-6050 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000 deg/*sec*
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA

- Standby current: 5μA
- Factory calibrated sensitivity scale factor
- User self-test

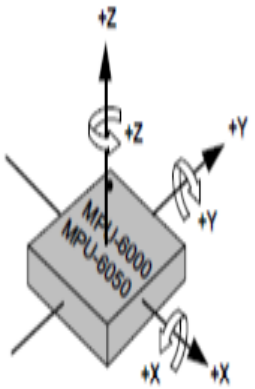
2.Accelerometer Features:
The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500μA
- Low power accelerometer mode current: 10μA at 1.25Hz, 20μA at 5Hz, 60μA at 20Hz, 110μA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

Orientation of Axes:
Figure 3.1 shows the orientation of the axes of sensitivity and the polarity of rotation. Where, the pin 1 identifier is (•).

Self-Test: Self-test allows for the testing of the mechanical and electrical portions of the sensors. The self-test for each measurement axis can be activated by means of the gyroscope and accelerometer self-test registers (registers 13 to 16).
When self-test is activated, the electronics cause the sensors to be actuated and produce an output signal. The output signal is used to observe the self-test response.
The self-test response is defined as follows:

Figure 3.1: Orientation of Axes of Sensitivity and Polarity of rotation



Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled
When the value of the self-test response is within the min/max limits of the product specification, the part has passed self-test. When the self-test response exceeds the min/max values, the part is deemed to have failed self-test.

3.2 Barometric Pressure Sensor

MS5611-01BA03 Barometric Pressure Sensor, with stainless steel cap has these main features: [MS5611-01BA03 Barometric Pressure Sensor datasheet]

1. High resolution module, 10 cm
2. Fast conversion down to 1 ms
3. Low power, 1 μA (standby < 0.15 μA)
4. QFN package 5.0 x 3.0 x 1.0 mm3
5. Supply voltage 1.8 to 3.6 V
6. Integrated digital pressure sensor (24 bit ΔΣ ADC)

7. Operating range: 10 to 1200 mbar, -40 to +85 °C
8. I2C and SPI interface up to 20 MHz
9. No external components (Internal oscillator)
10. Excellent long term stability

3.3 Magnetometer Sensor (Compass)

3-Axis Digital Compass IC HMC5883L The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry. The HMC5883L includes our state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I2C serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9 mm surface mount 16-pin leadless chip carrier (LCC). Applications for the HMC5883L include Mobile Phones, Netbooks, Consumer Electronics, Auto Navigation Systems, and Personal Navigation Devices. [HMC5883L 3-Axis Digital Compass datasheet]

Features:

1. 3-Axis Magnetoresistive Sensors and ASIC in a {3.0x3.0x0.9mm} LCC Surface Mount Package
2. 12-Bit ADC Coupled with Low Noise AMR Sensors Achieves 2 milli-gauss Field Resolution in ±8 Gauss Fields
3. Built-In Self Test
4. Low Voltage Operations (2.16 to 3.6V) and Low Power Consumption (100 μA)
5. Built-In Strap Drive Circuits
6. I2C Digital Interface
7. Lead Free Package Construction
8. Wide Magnetic Field Range (+/-8 Oe)
9. Software and Algorithm Support Available
10. Fast 160 Hz Maximum Output Rate

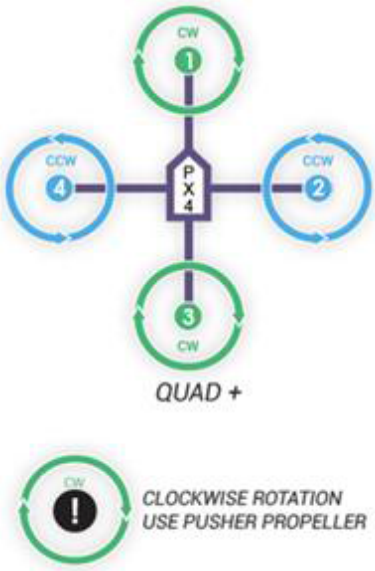
Chapter 4

Quadcopter Dynamics

4.1 Modeling

Quadcopters have two kinds of modes: “+” and “x”. In the present project, the “+” mode is used. As shown in figure 4.1, rotors refer to 1 Fontal, 2 Right, 3 Rear, and 4 Left.

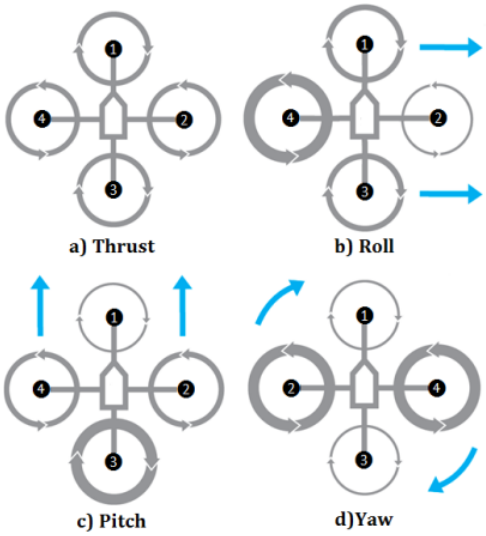
Figure 4.1: Mode"+" of the quadcopter



A quadcopter has four rotors, each one of which has independent speed ω_i, allowing generating the thrust and accelerations in the desired directions. Quadcopters are six-degrees-of-freedom (DoF) systems, what means that they can move along the three space axes X, Y and Z, and

turn on the aircraft's body axes describing roll (φ), pitch (θ) and yaw (ψ) angles. However, in the aircraft, there is only direct control over four DoF, corresponding to the altitude and the three angles: roll, pitch and yaw. Roll and pitch movements also generate motion along the Y and X axes of space, respectively. So the two DoF on axes X, Y will be ignored. A Pitch movement can achieve a movement along X, when 3 increases its speed and 1 decreases its speed. A Roll movement can achieve a movement along Y, when 4 increases its speed and 2 decreases its speed. A Yaw movement can be achieved by increasing both speeds of 2 and 4 and decreasing both 1 and 3. A vertical thrust can be achieved by increasing all speeds of the four rotors. As shown in figure 4.2.

Figure 4.2: Quadcopter movements a) Thrust, b)Roll, c)Pitch, and d)Yaw

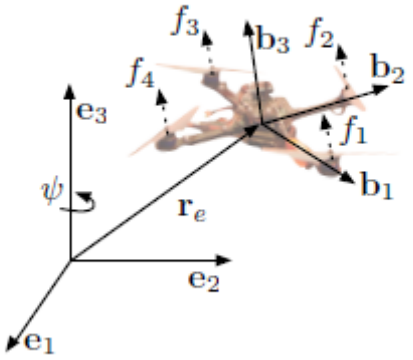


4.1.1 Quadcopter Modeling Using Euler-Newton method

The quadcopter is assumed to be a rigid body and thus Newton-Euler equations can be used to describe its dynamics. We will start deriving quadcopter Euler-Newton's equations by introducing the two frames in which will operate; The EF system (Earth-Fixed Frame) is a system that is fixed in attitude and position, placed on earth with its X axis pointing North, Y axis pointing West, and Z axis perpendicular to both, pointing up. The BF system (Body-Fixed Frame) is a fixed system on the aircraft, its X axis pointing forward along the body, its Y axis in the left direction

of the aircraft, and its Z axis perpendicular to both (upwards). As shown in figure 4.3. Because of the motion of BF with respect to EF, the variables represented in EF are identified with an E subscript, and the variables in BF with a B subscript.

Figure 4.3: Earth frame and Body frame



The force contribution in the system generates linear movements, and so they are defined in EF. The general equation of forces according to Newton's laws is:

$$F_E = ma_E \tag{4.1}$$

where, F_E is the force vector on EF, a_E is the vector of linear acceleration of the quadcopter on EF, and m is the mass of the quadcopter. To determine the force model, every force acting on the aircraft needs to be considered. The gravitational force on EF is a constant vector, because its direction its always pointing downward along the Z axis of EF. The gravity value is represented with g.

$$G_E = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{4.2}$$

The thrust generated by the propellers is a direction constant vector on the Z axis of BF. To obtain the thrust on EF, the rotation matrix R must be used:

$$R = R_z(\psi).R_y(\theta).R_x(\emptyset) \tag{4.3}$$

$$R_x(\emptyset)=\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\emptyset & -s\emptyset \\ 0 & s\emptyset & c\emptyset \end{bmatrix} R_y(\theta)=\begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \quad R_z(\varphi)=\begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$R=\begin{bmatrix} c\theta c\psi & s\emptyset s\theta c\psi - c\emptyset s\psi & c\emptyset s\theta c\psi + s\emptyset s\psi \\ c\theta s\psi & s\emptyset s\theta s\psi + c\emptyset c\psi & c\emptyset s\theta s\psi - s\emptyset c\psi \\ -s\theta & s\emptyset c\theta & c\emptyset c\theta \end{bmatrix} \tag{4.5}$$

$$T_E = RT_B = R\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = \begin{bmatrix} (c\emptyset s\theta c\psi + s\emptyset s\psi)T \\ (c\emptyset s\theta s\psi - s\emptyset c\psi)T \\ (c\emptyset c\theta)T \end{bmatrix} \tag{4.6}$$

where, T_E is the thrust on EF, T_B is the thrust on BF, and T is the scalar value of the thrust generated by the quadcopter. The value of T is given by:

$$T = k(\Omega_1{}^2 + \Omega_2{}^2 + \Omega_3{}^2 + \Omega_4{}^2) \tag{4.7}$$

where k is the aerodynamic lift coefficient, and Ω the speed of each rotor. So Euler-Newton equations will be:

$$RT_B - mg = ma \tag{4.8}$$

And the force contribution on the quadcopter results in the system of the following equations, which describes the linear dynamics of the aircraft on EF:

$$\ddot{x} = (cos\emptyset sin\theta cos\psi + sin\emptyset sin\psi)\frac{T}{m} \tag{4.9}$$

$$\ddot{y} = (cos\emptyset sin\theta sin\psi - sin\emptyset cos\psi)\frac{T}{m} \tag{4.10}$$

$$\ddot{z} = -g + (cos\theta cos\emptyset)\frac{T}{m} \tag{4.11}$$

The angular dynamics use BF and analyze the torque that generates angular acceleration on the system. The general equation for the torque on a body is given by:

$$\tau_B = I\dot{\omega}_B + \omega_B \times (I\omega_B) \tag{4.12}$$

where, τ_B is the vector of torques acting on a body on BF, I is the body inertia, better known as the matrix of inertia, and $\dot{\omega}_B$ is the vector of angular acceleration on BF. Because of the assumptions made that the origin of BF

matches the aircraft's center of gravity, and the axes of the quadcopter also match the axes of BF, is defined as:

$$I=\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{4.13}$$

So we can obtain these equations:

$$\tau_{\emptyset}=I_{xx}\dot{p}+q\ r\ \left(I_{zz}-I_{yy}\right) \tag{4.14}$$

$$\tau_{\theta}=I_{yy}\dot{q}+p\ r\ \left(I_{xx}-I_{zz}\right) \tag{4.15}$$

$$\tau_{\psi}=I_{zz}\dot{r}+q\ p\ \left(I_{yy}-I_{xx}\right) \tag{4.16}$$

where, p, q, r are the angular velocities on BF.
The spinning rotors also produce a gyroscopic effect whose contribution to the torque model is given by:

$$O_B\Omega=J_{TP}\begin{bmatrix} -q \\ p \\ 0 \end{bmatrix}(-\Omega_1+\Omega_2-\Omega_3+\Omega_4) \tag{4.17}$$

where, O_B is the momentum vector produced by the gyroscopic effect on BF, J_{TP} is the total momentum of inertia on the rotor shaft, ω_B is the vector of angular velocities, and Ω is the total algebraic sum of the rotors' speeds. Torque generated by the propellers is summarized in the system of the following equations, obtained by the operation principle of the quadcopter.

$$\tau_{\emptyset}=kl\ \left({\Omega_4}^2-{\Omega_2}^2\right) \tag{4.18}$$

$$\tau_{\theta}=kl\ \left({\Omega_3}^2-{\Omega_1}^2\right) \tag{4.19}$$

$$\tau_{\psi}=d\ \left({\Omega_4}^2+{\Omega_2}^2-{\Omega_1}^2-{\Omega_3}^2\right) \tag{4.20}$$

The torques produced by the propellers are directly associated with the system inputs. The torque contribution gives the angular dynamics of the system:

$$\dot{p}=\frac{\left(I_{yy}-I_{zz}\right)}{I_{xx}}q\ r+\frac{J_{TP}}{I_{xx}}q\Omega+\frac{\tau_{\emptyset}}{I_{xx}} \tag{4.21}$$

$$\dot{q}=\frac{(I_{zz}-I_{xx})}{I_{yy}}p\ r+\frac{J_{TP}}{I_{yy}}p\Omega+\frac{\tau_{\theta}}{I_{yy}} \tag{4.22}$$

$$\dot{r}=\frac{\left(I_{xx}-I_{yy}\right)}{I_{zz}}p\ q+\frac{\tau_{\psi}}{I_{zz}} \tag{4.23}$$

The total quadcopter dynamics model results in the system of equations:

$$\ddot{x}=(cos\emptyset sin\theta cos\psi+sin\emptyset sin\psi)\frac{T}{m} \tag{4.24}$$

$$\ddot{y}=(cos\emptyset sin\theta sin\psi-sin\emptyset cos\psi)\frac{T}{m} \tag{4.25}$$

$$\ddot{z}=-g+(cos\theta cos\emptyset)\frac{T}{m} \tag{4.26}$$

$$\dot{p}=\frac{\left(I_{yy}-I_{zz}\right)}{I_{xx}}q\ r+\frac{J_{TP}}{I_{xx}}q\Omega+\frac{\tau_{\emptyset}}{I_{xx}} \tag{4.27}$$

$$\dot{q}=\frac{(I_{zz}-I_{xx})}{I_{yy}}p\ r+\frac{J_{TP}}{I_{yy}}p\Omega+\frac{\tau_{\theta}}{I_{yy}} \tag{4.28}$$

$$\dot{r}=\frac{\left(I_{xx}-I_{yy}\right)}{I_{zz}}p\ q+\frac{\tau_{\psi}}{I_{zz}} \tag{4.29}$$

In the previous model, the dynamic model of the DC motors has been disregarded. PID controllers are only suitable for linear systems, so the model must be linearized around an operating point, which is hover flight. Using small-angle approximations and disregarding the equations of the X and Y movements that are not used in the control, the system of equations is:

$$\ddot{z}=-g+\frac{T}{m} \tag{4.30}$$

$$\ddot{\emptyset}=\frac{\tau_{\emptyset}}{I_{xx}} \tag{4.31}$$

$$\ddot{\theta}=\frac{\tau_{\theta}}{I_{yy}} \tag{4.32}$$

$$\ddot{\psi}=\frac{\tau_{\psi}}{I_{zz}} \tag{4.33}$$

4.1.2 Quadcopter Modeling Using Euler-Lagrange method

The model stands under the assumptions that the motor dynamics are relatively fast and may therefore be neglected. Also, the rotor blades are assumed to be perfectly rigid and no blade flapping occurs. research. External wind forces are also not considered at this stage. The equations of

motion are developed in terms of the translational and rotational parameters of the six-degree-of-freedom system, using generalized co-ordinates in a vector q:

$$q = [\ x\ \ y\ \ z\ \ \emptyset\ \ \theta\ \ \Psi\ ]^T \tag{4.34}$$

These co-ordinates can be separated into translational, ξ, and rotational, η, co-ordinates where: $\xi = [\ x\ \ y\ \ z\ ]^T$ and $\eta = [\ \ \emptyset\ \ \theta\ \ \Psi\ ]^T$ Therefore,

$$q = [\ \xi\ \ \eta] \tag{4.35}$$

A Lagrangian is obtained by modelling the energy of the system, where the difference between kinetic and potential energy is taken. Kinetic energy of the system is modelled for both translational and rotational motion.
The potential energy of the system is related only to the altitude of the rotorcraft. The Lagrangian, L, can be expressed as,

$$L = T - U \tag{4.36}$$

Where, T is the kinetic energy and U is the potential energy of the system. The total kinetic energy of the system is the sum of the energy due to the translational
motion and rotational motion. Therefore, the final expression obtained for the Lagrangian is,

$$L(q, q^{\cdot}) = \frac{1}{2}\ \xi^{\cdot T} m\ \xi^{\cdot} + \frac{1}{2}\ \eta^{\cdot T}\ I\eta^{\cdot} - mgz \tag{4.37}$$

This can be reduced to,

$$L(q, q^{\cdot}) = \frac{1}{2}\ m(x^{\cdot 2} + y^{\cdot 2} + z^{\cdot 2}) + \frac{1}{2}\ (I_{xx}\emptyset^{\cdot 2} + I_{yy}\ \theta^{\cdot 2} + I_{zz}\ \Psi^{\cdot 2}) - mgz \tag{4.38}$$

For this analysis, the Euler-Lagrange equation with an external generalized force is used.

$$\frac{d}{dt}\frac{\partial L}{\partial q^{\cdot}} - \frac{\partial L}{\partial q} = F \tag{4.39}$$

Where, the force F represents all the external forces acting on the body of the rotorcraft. Again, this can be split into translational, F_ξ and rotational, τ components. Where,

$$F_\xi = \left[F_x\ F_y\ F_z\right]^T$$

And

$$\tau = \left[\tau_\phi\ \tau_\theta\ \tau_\psi\right]^T$$

Therefore

$$F = \begin{bmatrix} F_\xi & \tau \end{bmatrix}^T$$

The PDE's $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}$ *and* $\frac{\partial L}{\partial q}$ can be expressed as,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \\ I_{xx}\ddot{\phi} + I_{\dot{x}x}\,\dot{\phi} \\ I_{yy}\,\ddot{\theta} + I_{\dot{y}y}\dot{\theta} \\ I_{zz}\ddot{\psi} + I_{\dot{z}z}\dot{\psi} \end{bmatrix} \tag{4.40}$$

And,

$$\frac{\partial L}{\partial q} = [0 \quad 0 \quad -mg \quad 0 \quad 0 \quad 0\ ]^T \tag{4.41}$$

The only forces acting on the body are those from the rotor. The translational forces are the thrust forces resulting from each rotor, and the rotational forces are due to the drag moment as well as the moment caused by thrust forces from opposite rotors about the center of gravity. Therefore, the translational force is,

$$F_{rotor} = [\ 0 \quad 0 \quad f\ ]^T \tag{4.42}$$

Where, f is the total thrust force. However, this force is always in the z-direction of the body reference frame. In order to obtain a force that will correlate with the global co-ordinates and sensor readings, the rotation matrix $R = R_x\,.\,R_y\,.\,R_z$ is used. This allows to compare sensor readings with calculated values to find a correlation between theoretical and measured parameters. Therefore,

$$F_\xi = R\,.\,F_{rotor} \tag{4.43}$$

It can be seen that,

$$F_\xi = \begin{bmatrix} -f\sin\theta \\ f\sin\phi\,\cos\theta \\ f\cos\phi\,\cos\theta \end{bmatrix} \tag{4.44}$$

For the rotational component τ , moments are taken from about the center of gravity in the roll and pitch directions.
Therefore,

$$\tau_\phi = (F_{M3} - F_{M1})\,l \tag{4.45}$$

$$\tau_\theta = (F_{M2} - F_{m4})\,l \tag{4.46}$$

Where, l is the linear distance from the centre of the rotor to the center of gravity. In the direction of yaw, the sum of all the drag moments produced by the rotors are taken into account,

$$\tau_{\psi} = \sum_{i-1}^{4} \tau_{Mi} \tag{4.47}$$

Where, τ_{Mi} is the torque produced by rotor i. Therefore, the total force function can be represented as the following vector,

$$F = \begin{bmatrix} -f\sin\theta \\ f\sin\phi\ \cos\theta \\ f\cos\phi\ \cos\theta \\ (F_{M3} - F_{M1})\,l \\ (F_{M2} - F_{m4})\,l \\ \sum_{i-1}^{4} \tau_{Mi} \end{bmatrix} \tag{4.48}$$

Therfore,

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \\ I_{xx}\ddot{\phi} + I_{\dot{x}x}\,\phi^{\cdot} \\ I_{yy}\,\ddot{\theta} + I_{\dot{y}y}\theta^{\cdot} \\ I_{zz}\ddot{\psi} + I_{\dot{z}z}\psi^{\cdot} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ mg \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -f\sin\theta \\ f\sin\phi\ \cos\theta \\ f\cos\phi\ \cos\theta \\ (F_{M3} - F_{M1})\,l \\ (F_{M2} - F_{m4})\,l \\ \sum_{i-1}^{4} \tau_{Mi} \end{bmatrix} \tag{4.49}$$

Resulting in,

$$\ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \phi^{\cdot\cdot} \\ \theta^{\cdot\cdot} \\ \psi^{\cdot\cdot} \end{bmatrix} = \begin{bmatrix} -\frac{f}{m}\sin\theta \\ \frac{f}{m}\sin\phi\ \cos\theta \\ \frac{f}{m}\cos\phi\ \cos\theta\ -g \\ \frac{1}{I_{\dot{x}x}}((F_{M3} - F_{M1})\,l - I_{\dot{x}x}\,\phi^{\cdot}) \\ \frac{1}{I_{\dot{y}y}}((F_{M2} - F_{m4})\,l - I_{\dot{y}y}\theta^{\cdot}) \\ \frac{1}{I_{\dot{z}z}}(\ \sum_{i-1}^{4} \tau_{Mi} - \ I_{\dot{z}z}\psi^{\cdot}) \end{bmatrix} \tag{4.50}$$

Near hover, the angular rates can be taken to be negligible.
Attitude ($\emptyset$, θ and ψ) and altitude (z) had to be taken into account in order to stabilize the rotorcraft. Position (x *and* y) is dependent on the roll and pitch angle
orientation. Thus, the position can be controlled via attitude control. Therefore, the system is an under actuated one, having six degrees of freedom and only

four control inputs. Essentially, all maneuvers that are executed by the rotorcraft are resultant from the manipulation of thrust and drag moment created by the
four rotors. These parameters are related using the relationship described in equation:

$$F_i = b \, . \, w_i^2 \tag{4.51}$$

and equation:

$$\tau_i = d. \, w_i^2 \tag{4.52}$$

It is evident from this that by controlling the rotational speed of the motors, one can effectively control the rotorcraft. Therefore, the following are chosen for control inputs:

$$u_1 = b.(w_1^2 + w_2^2 + w_3^2 + w_4^2)$$

$$u_2 = \ b.\left(w_3^2 - \ w_1^2\right).l$$

$$u_3 = \ b.\left(w_2^2 - \ w_4^2\right).l$$

$$u_4 = d.(w_1^2 - w_2^2 + w_3^2 - w_4^2)$$

Where, u_1, u_2, u_3 and u_4 are control inputs for altitude, roll,pitch and yaw respectively. From this, the simplified model derived becomes,

$$\ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \phi^{\cdot\cdot} \\ \theta^{\cdot\cdot} \\ \psi^{\cdot\cdot} \end{bmatrix} = \begin{bmatrix} -\frac{u_1}{m}\sin\theta \\ \frac{u_1}{m}\sin\phi\,\cos\theta \\ \frac{u_1}{m}\cos\phi\,\cos\theta - g \\ \frac{1}{I_{xx}}u_2 \\ \frac{1}{I_{yy}}u_3 \\ \frac{1}{Izz}u_4 \end{bmatrix} \tag{4.53}$$

It is evident from the last equation that the simplified system is linear and not coupled. Thus, a linear PD controller design could be chosen to be used.
The control inputs $u_i$described above may be represented in vector form,

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ -bL & 0 & bL & 0 \\ 0 & bL & 0 & -bL \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{4.54}$$

The last Equation can be rearranged to be found in terms of the rotational speed vector,

$$\begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & -\frac{1}{2bL} & 0 & \frac{1}{4d} \\ \frac{1}{4b} & 0 & -\frac{1}{2bL} & -\frac{1}{4d} \\ \frac{1}{4b} & \frac{1}{2bL} & 0 & \frac{1}{4d} \\ \frac{1}{4b} & 0 & -\frac{1}{2bL} & -\frac{1}{4d} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{4.55}$$

Chapter 5

Control Methods

5.1 Implementing PD controller for position tracking

The quadrotor system is multi-ODEs equations. The linearized model is usually used especially for position tracking, therefore one can determine the desired x-y positions based on the roll and pitch angles. As a result, one nested loop which has inner and outer controllers is needed for controlling the quadrotor.
The dynamics equations of the nonlinear model of the quadrotor are:

$$\begin{cases} \ddot{\emptyset} = \theta \cdot \psi \cdot \left(\frac{I_y - I_z}{I_x}\right) - \frac{J_r}{I_x}\,\theta \cdot \Omega + \frac{l}{I_x}\,U_2 \\ \ddot{\theta} = \phi \cdot \psi \cdot \left(\frac{I_z - I_x}{I_y}\right) - \frac{J_r}{I_y}\,\phi \cdot \Omega + \frac{l}{I_y}\,U_3 \\ \ddot{\psi} = \theta \cdot \phi \cdot \left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}\,U_4 \\ \ddot{z} = z - g + (cos\phi\ .\ cos\theta)\,\frac{1}{m}\,U_1 \\ \ddot{x} = (cos\phi.\sin\theta.\ cos\psi + sin\phi.\ sin\psi\ )\,\frac{1}{m}U_1 \\ \ddot{y} = (cos\phi.\sin\theta.\ sin\psi - sin\phi.cos\psi\ )\,\frac{1}{m}U_1 \end{cases} \tag{5.1}$$

if we assume the small angles, the model is reduced to the following equations:

$$\begin{cases} \ddot{\emptyset} = \frac{l}{I_x}\,U_2 \\ \ddot{\theta} = \frac{l}{I_y}\,U_3 \\ \ddot{\psi} = \frac{1}{I_z}\,U_4 \\ \ddot{z} = -g + \frac{1}{m}\,U_1 \\ \ddot{x} = (\theta cos\psi + \phi.\ sin\psi\ )\,\frac{1}{m}U_1 \\ \ddot{y} = (\theta.\ sin\psi - \phi.cos\psi)\,\frac{1}{m}U_1 \end{cases} \tag{5.2}$$

The aforementioned equations are linear. For position tracking, we need to control $x, y,$ and z, therefore we choose the desired roll and pitch (i.e. ϕ^d and θ^d)

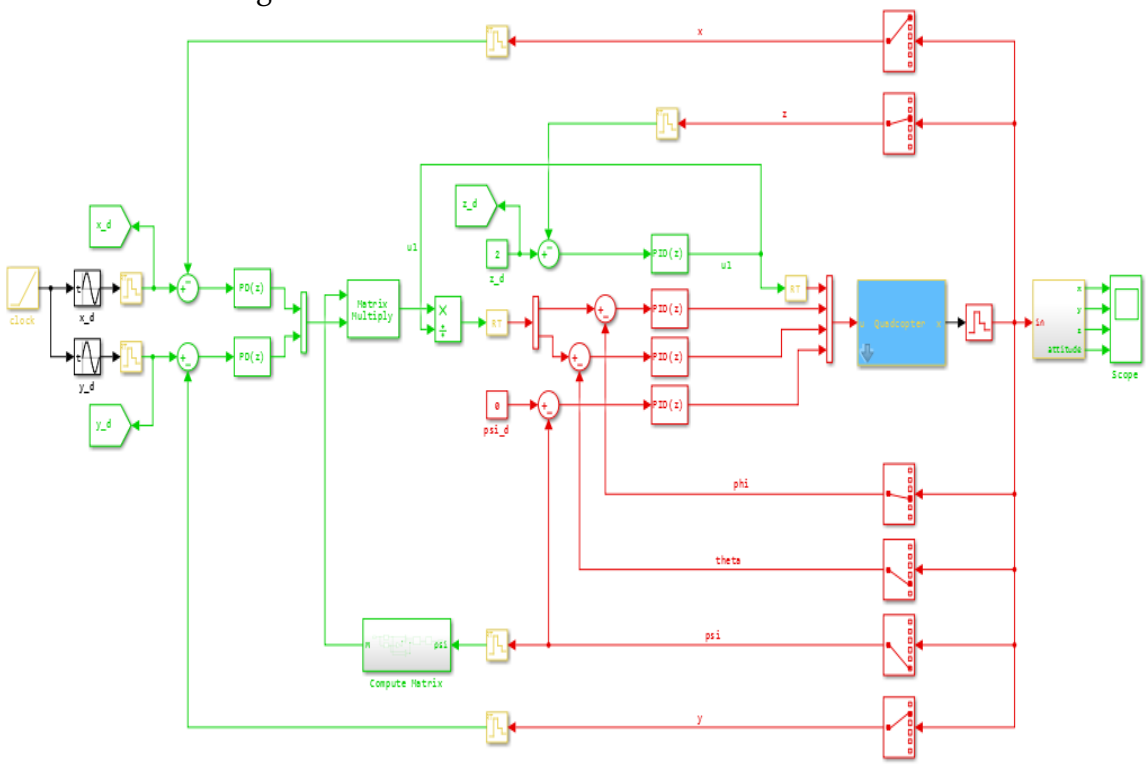
$$\ddot{x}^d = \left(\theta^d cos\psi + \phi^d . sin\psi\ \right)\frac{1}{m}U_1 \tag{5.3}$$

$$\ddot{y}^d = \left(\theta^d . sin\psi - \phi^d . cos\psi\right)\frac{1}{m}U_1 \tag{5.4}$$

Therefore, the closed form for the desired angles can be obtained as follows:

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \begin{bmatrix} sin\psi & cos\psi \\ -cos\psi & sin\psi \end{bmatrix}^{-1} \left(\frac{m}{U1}\right) \begin{bmatrix} \ddot{x}^d \\ \ddot{y}^d \end{bmatrix} \tag{5.5}$$

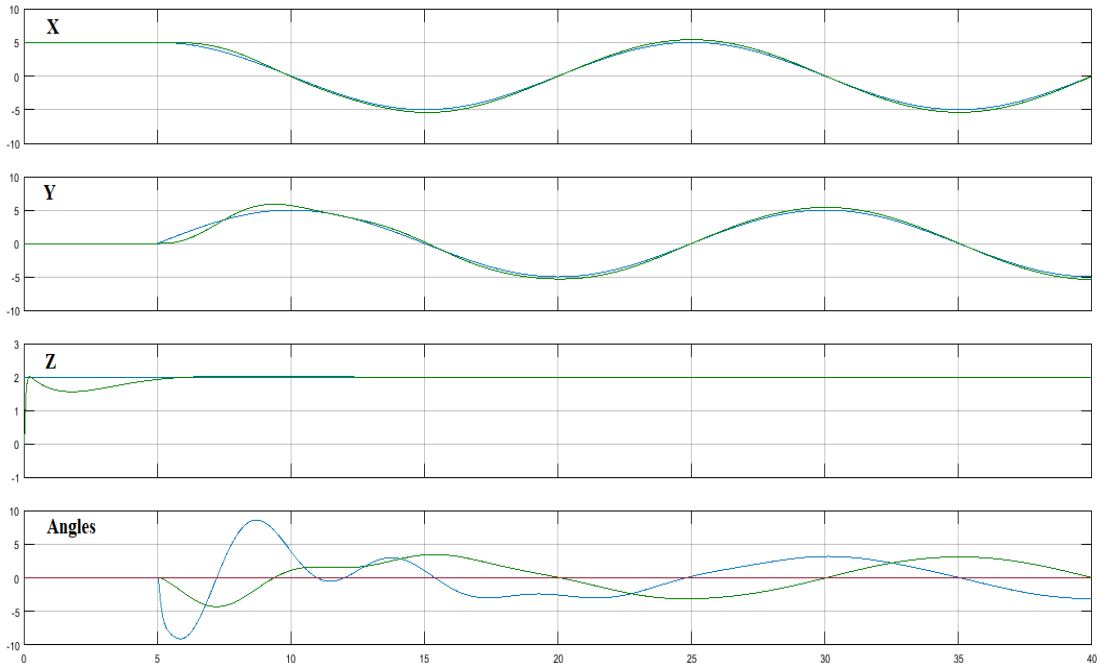
Figure 5.1: PID Control Simulink Model



The stream in red presents the attitude control which runs at a 1kHz speed, while the green stream represents the linear displacement tracking at a speed of 100Hz.

Since we have to ensure that the "*small angles*" assumption remains valid throughout the simulation, the **take-off** phase (i.e. reaching the desired height z_d) needs to be almost completed before starting the **displacement tracking** of the $x_d(t)$ *and* $y_d(t)$ time-varying linear coordinates (a circle of radius $5m$ in the model). Further, for the same reason, the desired circular trajectory must start from the quadcopter's initial position $(x(0), y(0))$. Depending on the PID gains, we can get quite nice tracking results:

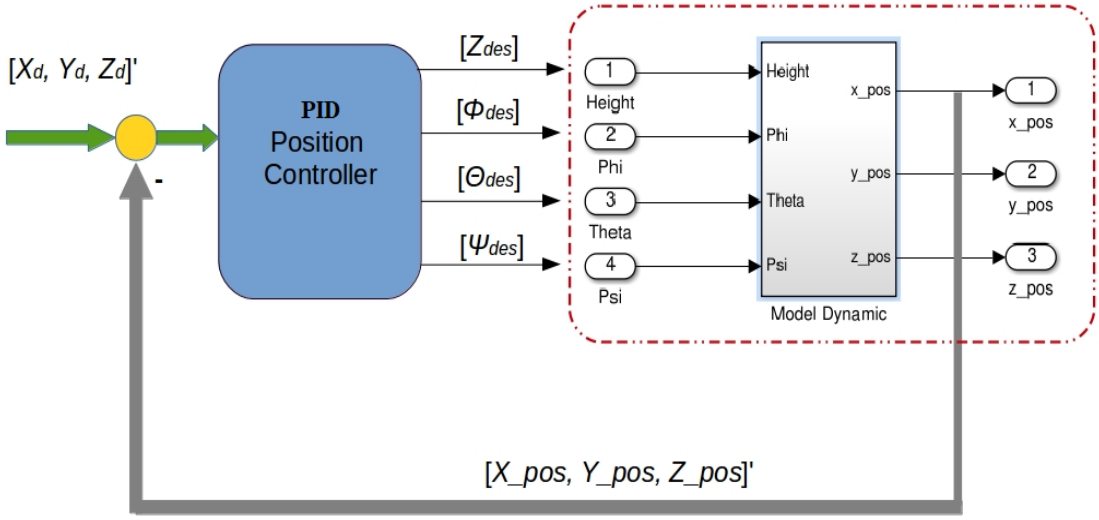
Figure 5.2: PID Results (X,Y,Z and angles)



The first top three diagrams shows the x, y *and* z components (in green), respectively, while following the targets (in blue). The last plot reports on the roll (blue), pitch (green) and yaw (red). Time is expressed in seconds, displacements are in meters and angles are in degrees
In my MATLAB model the quadrotor takes 4 inputs: the desired altitude ($\boldsymbol{Z_{des}}$) and the desired attitude angles($\boldsymbol{\Phi_{des}}$, $\boldsymbol{\Theta_{des}}$, $\boldsymbol{\Psi_{des}}$) which reflects the motion described by the differential equations of the model.

Here is an insight into the implemented MATLAB dynamic model. As can be seen (Fig.5.3) it has a structure like an inner loop controller.

Figure 5.3: PID Position Control Feedback Loop



The quadcopter hovers perfectly on the starting point. So the next step is to do position control to make the quadcopter reach the desired position, which is defined as 3 coordinates values $[X_d, Y_d, Z_d]$.
To control the position, another PID controller was used.The output of the controller will be the desired angles of row and pitch. While yaw can be chosen as:

$$\psi = atan2((y_d - y_{actual}), (x_d - x_{actual})) \tag{5.6}$$

The total diagram of the system, consists of two nested loops.
The outer loop controller assumes that the angles are tracked perfectly and treats angles as inputs, rather than separate states. So, it takes the position error and converts it to a desired roll/pitch (via PID, just like any other system where we use a controller to calculate desired system inputs), and those desired angles are passed into the inner loop controller.
The outer loop output is the desired angles (yaw is assumed to be zero):

$$\phi^d = k_{py}(y - y_d) + k_{dy}(v_y - v_{dy}) + k_{iy}\int t_0(y - y_d)dt \tag{5.7}$$

$$\theta^d = -k_{px}(x - x_d) = k_{dx}(v_x - v_{dx}) - k_{ix}\int t_0(x - x_d)dt \tag{5.8}$$

where $\widehat{\phi}^d$ is the desired roll angle and $\widehat{\theta}^d$ is the desired pitch angle. When $yaw \neq 0$,the following equations should be used:

$$\phi_d = cos(\psi)\widehat{\phi}^d - sin(\psi)\widehat{\theta}^d \tag{5.9}$$

$$\theta_d = sin(\psi)\widehat{\phi}^d + cos(\psi)\widehat{\theta}^d \tag{5.10}$$

where ψ is the yaw angle.

5.2 Applying Lyapunov Stability Criteria on the Dynamic model

From the Dynamic model of Quadrotor (Newton's Method) we have:

$$\ddot{\emptyset} = \dot{\theta}.\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J_r}{I_x}\dot{\theta}.\Omega + \frac{l}{I_x}U_2 \tag{5.11}$$

$$\ddot{\theta} = \dot{\emptyset}.\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) - \frac{J_r}{I_y}\dot{\varphi}.\Omega + \frac{l}{I_y}U_3 \tag{5.12}$$

$$\ddot{\psi} = \dot{\theta}.\dot{\emptyset}\left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}U_4 \tag{5.13}$$

$$\ddot{z} = -g + (cos\theta.cos\emptyset)\frac{U}{m} \tag{5.14}$$

$$\ddot{y} = (cos\emptyset.sin\theta.sin\psi - sin\emptyset.cos\psi)\frac{U}{m} \tag{5.15}$$

$$\ddot{x} = (cos\emptyset.sin\theta.cos\psi + sin\emptyset.sin\psi)\frac{U}{m} \tag{5.16}$$

The previous model can be rewritten in a state-space form $\dot{X} = f(X, U)$ by introducing $X = (x_1 ... x_{12})^T \in R^{12}$ as state vector of the system.

$x_1 = \emptyset$	$x_7 = z$
$x_2 = \dot{x}_1 = \dot{\emptyset}$	$x_8 = \dot{x}_7 = \dot{z}$
$x_3 = \theta$	$x_9 = x$
$x_4 = \dot{x}_3 = \dot{\theta}$	$x_{10} = \dot{x}_9 = \dot{x}$
$x_5 = \psi$	$x_{11} = y$
$x_6 = \dot{x}_5 = \dot{\psi}$	$x_{12} = \dot{x}_{11} = \dot{y}$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} \dot{\emptyset} \\ \ddot{\emptyset} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{z} \\ \ddot{z} \\ \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 x_6 a_1 + x_4 a_2 \Omega + \mathrm{b}_1 \mathrm{U}_2 \\ x_4 \\ x_2 x_6 a_3 + x_2 a_4 \Omega + b_2 U_3 \\ x_6 \\ x_4 x_2 a_5 + b_3 U_4 \\ x_8 \\ -g + (\cos x_1 \cos x_3) \frac{U_1}{m} \\ x_{10} \\ u_x \frac{U_1}{m} \\ x_{12} \\ u_y \frac{U_1}{m} \end{bmatrix} \tag{5.17}$$

5.3 Implementing a Backstepping Controller

In this section, a nonlinear backstepping controller[13] is designed. The aim of the controller is to track a desired trajectory asymptotically.
To get more accurate results, the effects of aerodynamic drag for translational motion and the second order derivatives of desired trajectory are added into the control law. In addition, a simpler flight approach is obtained by controlling the yaw angle such that the heading of the quadrotor and the direction of motion are along the same line.
The backstepping control method can be explained in three steps. First step is the position control of the quadrotor. In this step, control output $U1$ and functions u_x *and* u_y which relates to the orientation of $U1$, are formulated. Second step is to find desired "Euler angles" (desired orientation) which are used to find control outputs $U2$, $U3$ and $U4$ are found using the functions u_x, u_y formulated in the position control. The third step is the attitude control of the quadrotor. In this step, control outputs $U2$, $U3$, $U4$ which are directly related to the attitude of the quadrotor, are formulated.
It is also important to note that, one of the most difficult and important part of backstepping controller design is the selection of control parameters. The backstepping controller obtained in this project has 12 control parameters(α_1, $\alpha_2 \ldots..$ α_{12}) to be determined. Therefore, "MATLAB Optimization Toolbox" is used to obtain control parameters that give satisfactory results.
The simplified nonlinear dynamic model of the quadrotor can be rewritten in a complete nonlinear dynamic model of the quadrotor expressed as

mentioned before like this:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \\ \dot{x_7} \\ \dot{x_8} \\ \dot{x_9} \\ \dot{x_{10}} \\ \dot{x_{11}} \\ \dot{x_{12}} \end{pmatrix} = \begin{pmatrix} \dot{\emptyset} \\ \ddot{\emptyset} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} x_2 \\ x_4 x_6 a_1 + b_1 U_2 \\ x_4 \\ x_2 x_6 a_3 + b_2 U_3 \\ x_6 \\ x_4 x_2 a_5 + b_3 U_4 \\ x_8 \\ u_x \frac{U_1}{m} - K_x \frac{x_8}{m} \\ x_{10} \\ u_y \frac{U_1}{m} - K_y \frac{x_{10}}{m} \\ x_{12} \\ -g + (\cos x_1 \ \cos x_3\ ) * \frac{U_1}{m} - K_z \frac{x_{12}}{m} \end{pmatrix} \tag{5.18}$$

Where:

$$a_1 = \frac{I_y - I_z}{I_x}\ ; a_3 = \frac{I_z - I_x}{I_y}; a_5 = \frac{I_x - I_y}{I_z}; b_1 = \frac{d}{I_x}\ \ ; b_2 = \frac{d}{I_y}; b_3 = \frac{1}{I_z};$$

$$u_x \ = \ (cos(x_1)sin(x_3)cos(x_5) \ + \ sin(x_1)sin(x_5)) \tag{5.19}$$

$$u_y \ = \ (cos\,(x_1)sin\,(x_3)sin\,(x_5) - sin(x_1)cos(x_5)) \tag{5.20}$$

The terms $a_1;\ a_3;\ a_5$ *and* $b_1;\ b_2;\ b_3$ are defined according to the final form of the simplified nonlinear dynamic model for rotational motion. Similarly, the terms u_x *and* u_y are defined according to the final form of the nonlinear dynamic model for translational motion. In addition, $U1,\ U2,\ U3, U4$ *and* $K_x,\ K_y,\ K_z$ are control inputs and aerodynamic drag coefficients for translational motion, respectively.

Step 1: Attitude Control:

In this part, the desired orientation or "Euler angles" of the quadcopter are obtained. Then, according to Lyapunov Theorem, and using the backstepping approach, one can synthesize the control law forcing the system to follow the desired trajectory. For the first step we consider the tracking-error:

$$z_1 = x_{1d} - x_1 \tag{5.21}$$

And we use the Lyapunov theorem by considering the Lyapunov function $z_1$positive definite and its time derivative negative semi-definite:

$$V\,(z_1) = \frac{1}{2} z_1^2 \tag{5.22}$$

$$\dot{V}(z_1) = z_1(x_{1d} - x_2) \tag{5.23}$$

The stabilization of z1 can be obtained by introducing a virtual control input x_2:

$$x_2 = \dot{x}_{1d} + \alpha_1 z_1 \tag{5.24}$$

with $\alpha_1 > 0$. Then $\dot{V}(z_1)$ will be:

$$\dot{V}(z_1) = -\alpha_1 z_1^2 \tag{5.25}$$

let us proceed to a variable change by making:

$$z_2 = x_2 - \dot{x}_{1d} - \alpha_1 z_1 \tag{5.26}$$

For the second step we consider the augmented Lyapunov function:

$$V(z_1, z_2) = \frac{1}{2}(z_1^2 + z_2^2) \tag{5.27}$$

And its time derivative is then:

$$\dot{V}(z_1 z_2) = z_2(a_1 x_4 x_6 + a_2 x_4 \Omega + b_1 U_2) - z_2(\ddot{x}_{1d} - \alpha_1(z_2 + \alpha_1 z_1)) - z_1 z_2 - \alpha_1 z_1^2 \tag{5.28}$$

The control input U_2 is then extracted ($\ddot{x}_{1,2,3d} = 0$), satisfying $\dot{V}(z_1 z_2) < 0$:

$$U_2 = \frac{1}{b_1}(z_1 - \alpha_1 x_4 x_6 - \alpha_2 x_4 \Omega - \alpha_1(z_2 + \alpha_1 z_1) - \alpha_2 z_2) \tag{5.29}$$

The term $z_1 z_2$ with $\alpha_2 > 0$ is added to stabilize z_1.
The same steps are followed to extract U_3, U_4.

$$U_3 = \frac{1}{b_2}(z_3 - \alpha_3 x_2 x_6 - \alpha_4 x_2 \Omega - \alpha_3(z_4 + \alpha_3 z_3) - \alpha_4 z_4) \tag{5.30}$$

$$U_4 = \frac{1}{b_3}(z_5 - \alpha_5 x_2 x_4 - \alpha_5(z_6 + \alpha_5 z_5) - \alpha_6 z_6) \tag{5.31}$$

Where:

$$z_3 = x_{3d} - x_3$$

$$z_4 = x_4 - \dot{x}_{3d} - \alpha_3 z_3$$

$$z_5 = x_{5d} - x_5$$

$$z_6 = x_6 - \dot{x}_{5d} - \alpha_5 z_5$$

$$z_7 = x_{7d} - x_7$$

$$z_8 = x_8 - \dot{x}_{7d} - \alpha_7 z_7$$

Step 2: Position Control:
In this part, first, control output U1 will be obtained using the same method. Control output U1 represents the total force generated by four propellers of quadrotor. Therefore, it is directly related to the motion in translational directions (x, y, z), which can be clearly seen in the dynamic model of the quadrotor. Moreover, by considering the dynamic model that is defined previously, one can see that motion in x and y directions are also effected by u_x *and* u_y, respectively. Therefore, in the first part of position control, altitude control will be obtained by choosing U_1 properly and in the second part, motion in x *and* y directions will be obtained by choosing u_x *and* u_y properly. As mentioned before, in the first part of position control, U_1 will be chosen to make the error between desired and actual altitude converge to zero according to the Lyapunov theory.
Then, proper choose of U_1 is defined as:

$$U_1 = \frac{m}{\cos x_1 \cos x_3}(z_7 + g - \alpha_7 (z_8 + \alpha_7 z_7) - \alpha_8 z_8) \tag{5.32}$$

As can be seen in the previous Equation, the effect of second order derivative of desired altitude ($\ddot{x_{1d}}$) is added to the controller for better tracking performance. This effect was neglected while driving attitude controller.
Using the same method, functions u_x *and* u_y which are directly related to the motion in x and y directions are found as following:

$$u_x = (m/U_1)(z_7 + \ddot{x_{7d}} + \alpha_7 z_8 - \alpha_7^2 z_7 + \frac{K_x x_8}{m} + \alpha_8 z_8) \tag{5.33}$$

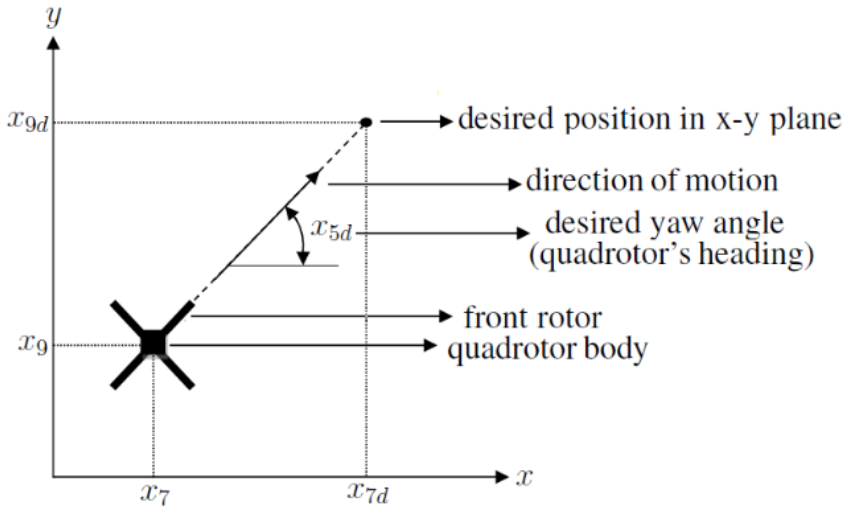
$$u_y = (m/U_1)(z_9 + \ddot{x_{9d}} + \alpha_9 z_{10} - \alpha_9^2 z_9 + \frac{K_y x_{10}}{m} + \alpha_{10} z_{10}) \tag{5.34}$$

The functions u_x *and* u_y which are obtained in previous Equations respectively, will be used to find desired Euler angles x_{1d} *and* x_{3d} which will be the inputs of attitude controller.
Step 3: Obtaining Desired Angles:
In this part, desired Euler angles x_{1d}, x_{3d} *and* x_{5d} which are the inputs of attitude controller, will be obtained. First, desired yaw angle(x_{5d}) will be obtained such that quadrotor's heading and the direction of motion in X-Y plane, are on the same line. Using Fig. 5.4 and basic trigonometry, desired yaw angle(x_{5d}) can be obtained as follows:

$$x_{5d} = arctan[(x_{9d} - x_9), (x_{7d} - x_7)] \tag{5.35}$$

x_7 *and* x_9 represent the actual position in $x-y$ plane whereas, x_{7d} *and* x_{9d} represent the desired position in $x-y$ plane. Therefore, x_{7d} *and* x_{9d} are

Figure 5.4: Lyapunov Parameters



given to the system as inputs that define desired trajectory to be followed. Once the desired yaw angle (x_{5d}) is obtained, desired roll (x_{1d}) and desired pitch (x_{3d}) angles can be obtained using both 5.19 and 5.20 equations. Desired roll angle (x_{1d}) is obtained as:

$$x_{1d} = arcsin[u_x sin(x_{5d}) - u_y cos(x_{5d})] \tag{5.36}$$

Using a similar approach desired pitch angle(x_{3d}) is obtained as:

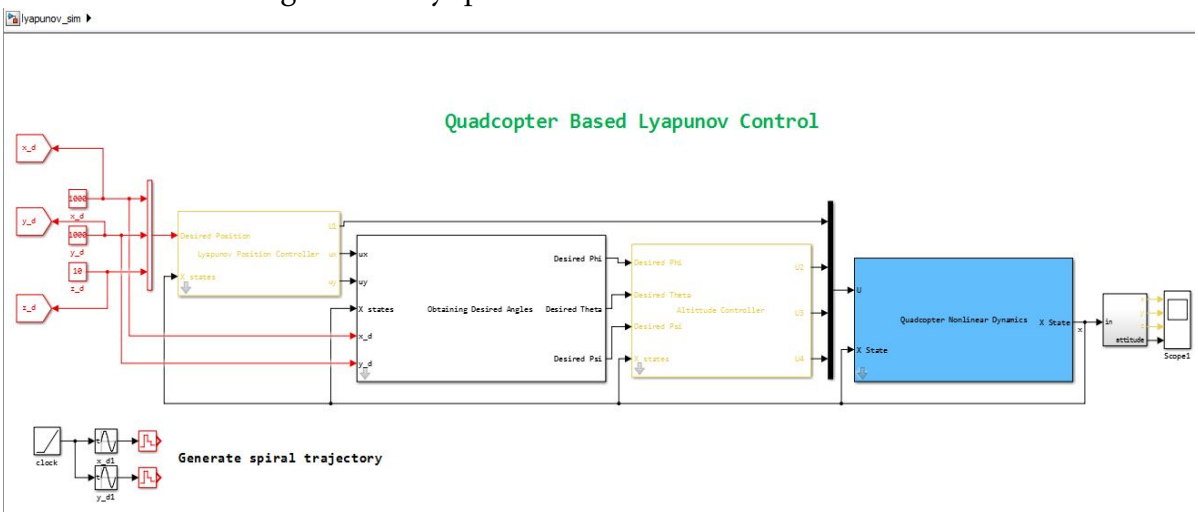
$$x_{3d} = arcsin[(u_x cos(x_{5d}) + u_y sin(x_{5d})), cos(x_{1d})] \tag{5.37}$$

5.4 Applying Backstepping Nonlinear Control Method

This system is divided into four main sub-systems in addition to the input and output units and figure 5.5 illustrates these system components:

- **Red input blocks** and wires represent the inputs which are the goal coordinates $x_{final}, y_{final}, z_{final}$. As shown in the figure they could be fixed values or dynamic values to track a specific trajectory. In other words, these values are the reference (fixed) values which the controller will compare with the actual values received from the gyroscope and other sensors.

Figure 5.5: Lyapunov Simulink Model



- Using those input blocks and/or another appropriate blocks we can generate any trajectory to make the quadcopter follow it (should be real trajectory).

- The first subsystem in our project is the **Lyapunov Position Controller** which has two inputs are the desired position from the input system and a state vector (twelve variables represent the system state at any time).

- This block calculates the first input force $U1$ based on Lyapunov theorem using the actual and reference values of the coordinates. In addition, the sub-forces u_x, u_y are determined to calculate the desired $Phi, Theta$ angles. The output of this subsystem is the three mentioned forces $U1, u_x, u_y$.

- The next subsystem is the **Desired Angles Calculator** which takes the output of the position controller as an input in addition to the state vector and the reference x, y values.

- In this subsystem the three required angles are calculated to compensate the error while flying and the three values are updated using Lyapunov variables and sent as outputs $(desiredPhi, desiredTheta, desiredPsi)$ to the next subsystem.

- The next subsystem is the **Attitude Controller** which takes the output of the **Angles calculator** as input in addition to the state vector

and the output would be the three left main forces $U2, U3, U4$.

- These forces can be calculated using Lyapunov variables depending on the state vector variables and the reference ones. At this stage, we end up with all the required forces which will control the dynamics of the flying drone.

- The next subsystem represents the **Quadcopter Nonlinear Dynamics** which are in fact the Newton's representation of the quadcopter as mention the chapter 5 and this system needs the four main forces $U1, U2, U3, U4$ to be triggered in addition to the state vector to update the necessary values and the output of this subsystem is the new and complete state vector in this closed loop iteration.

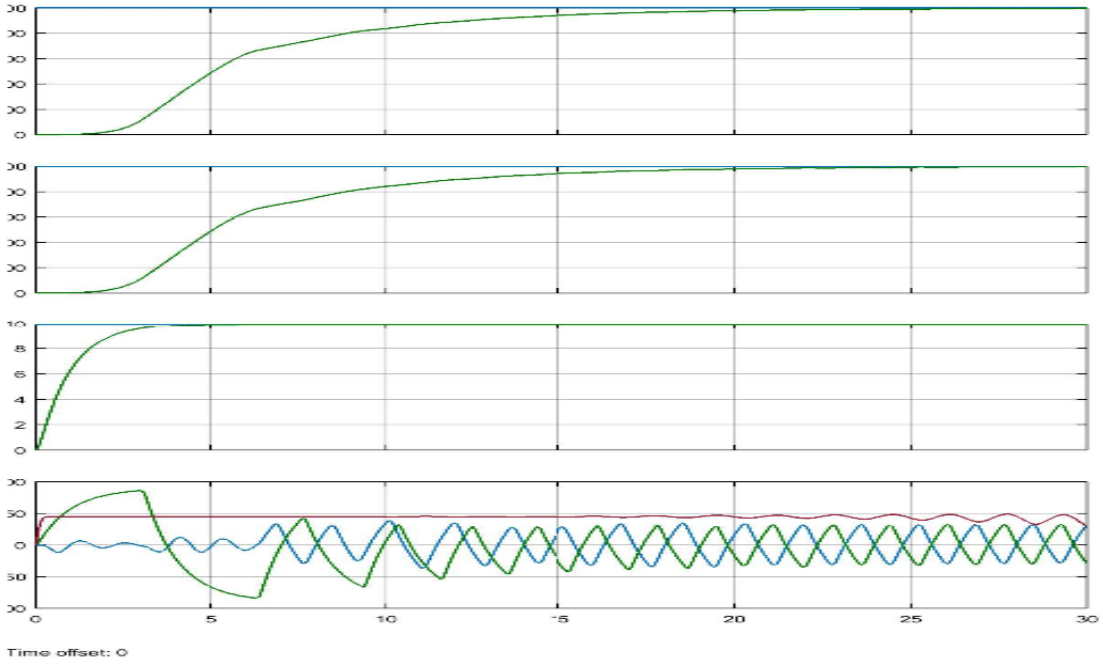
- The result of the system is the derivative of the state vector so we need an integrator to obtain the required state vector values.

- The last subsystem is the **output** selective block in which we select and out value to display it on a scope block

5.5 Results

The backstepping control based system has been tested on both fixed desired position and a dynamic trajectory.
Figure 5.6 shows the response to a fixed input values $x = 1000, y = 1000, z = 10$. As can be seen, it is a fast robust response.

Figure 5.6: Response to fixed input values $x = 1000, y = 1000, z = 10$:



Figures 5.7 and 5.8 shows the system response to a spiral trajectory. As a result, the backstepping control based system was able to track the desired trajectory robustly with a precision of few centimeters.

Figure 5.7: Response to spiral trajectory - 3D View

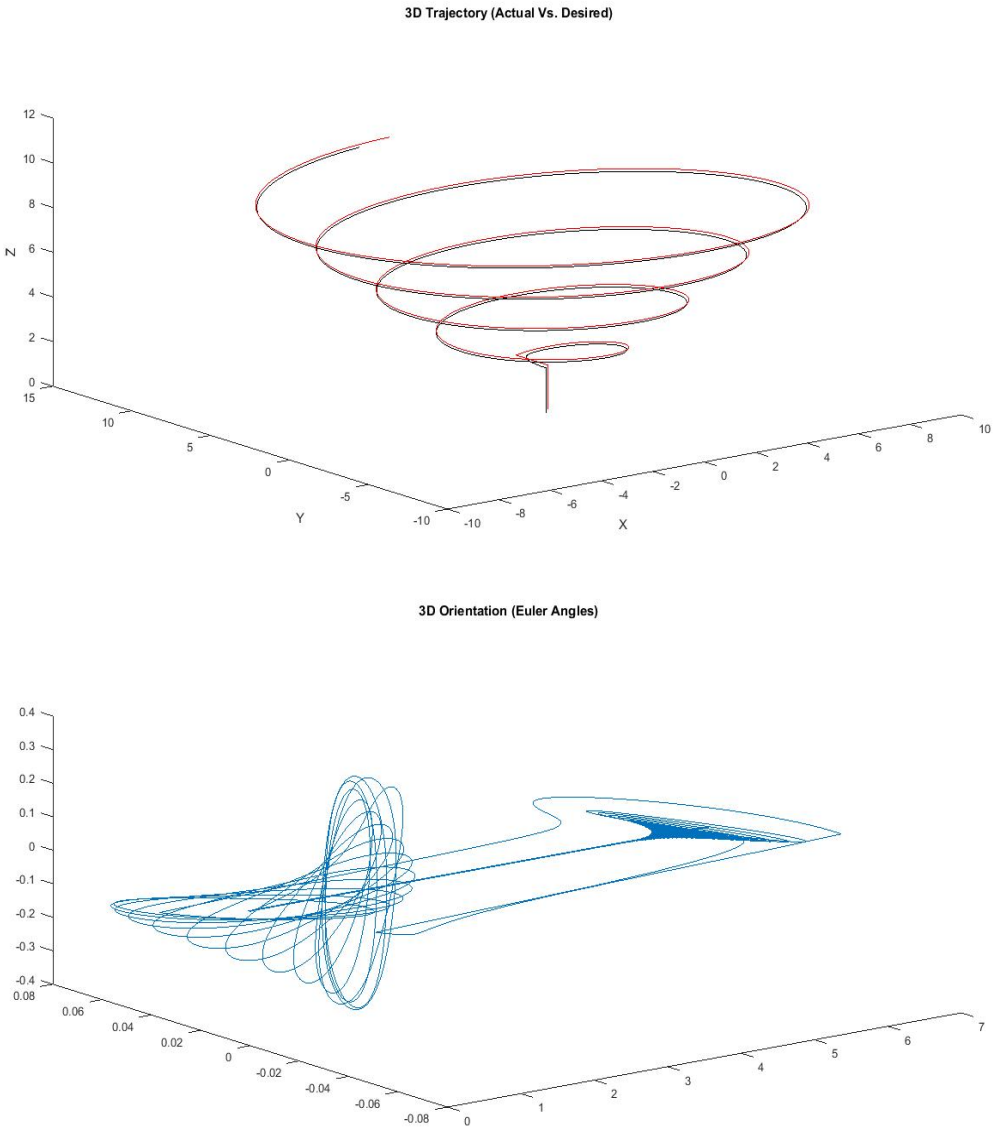


Figure 5.8: Response to spiral trajectory - Coordinates

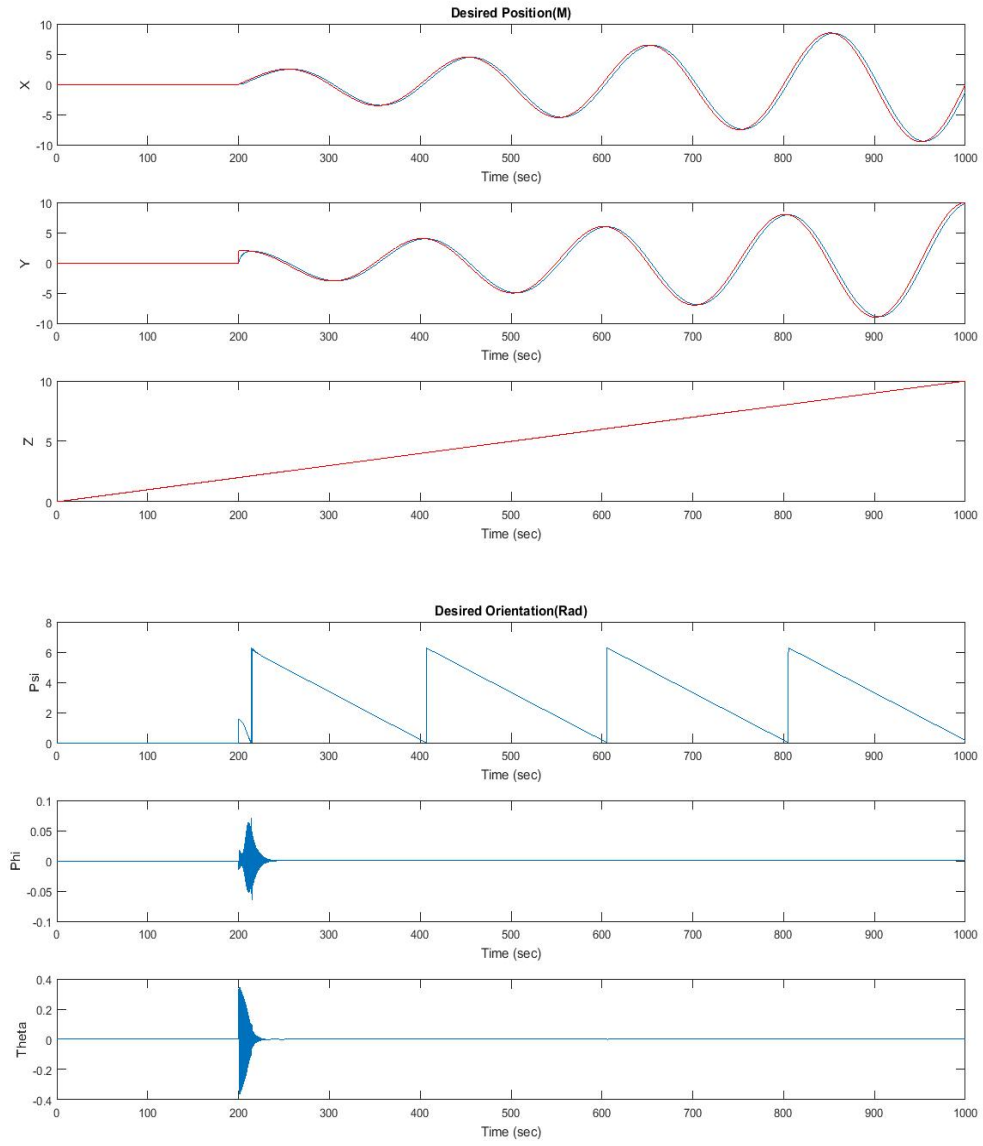


Figure 5.9: Response to rectangular trajectory - 3D View

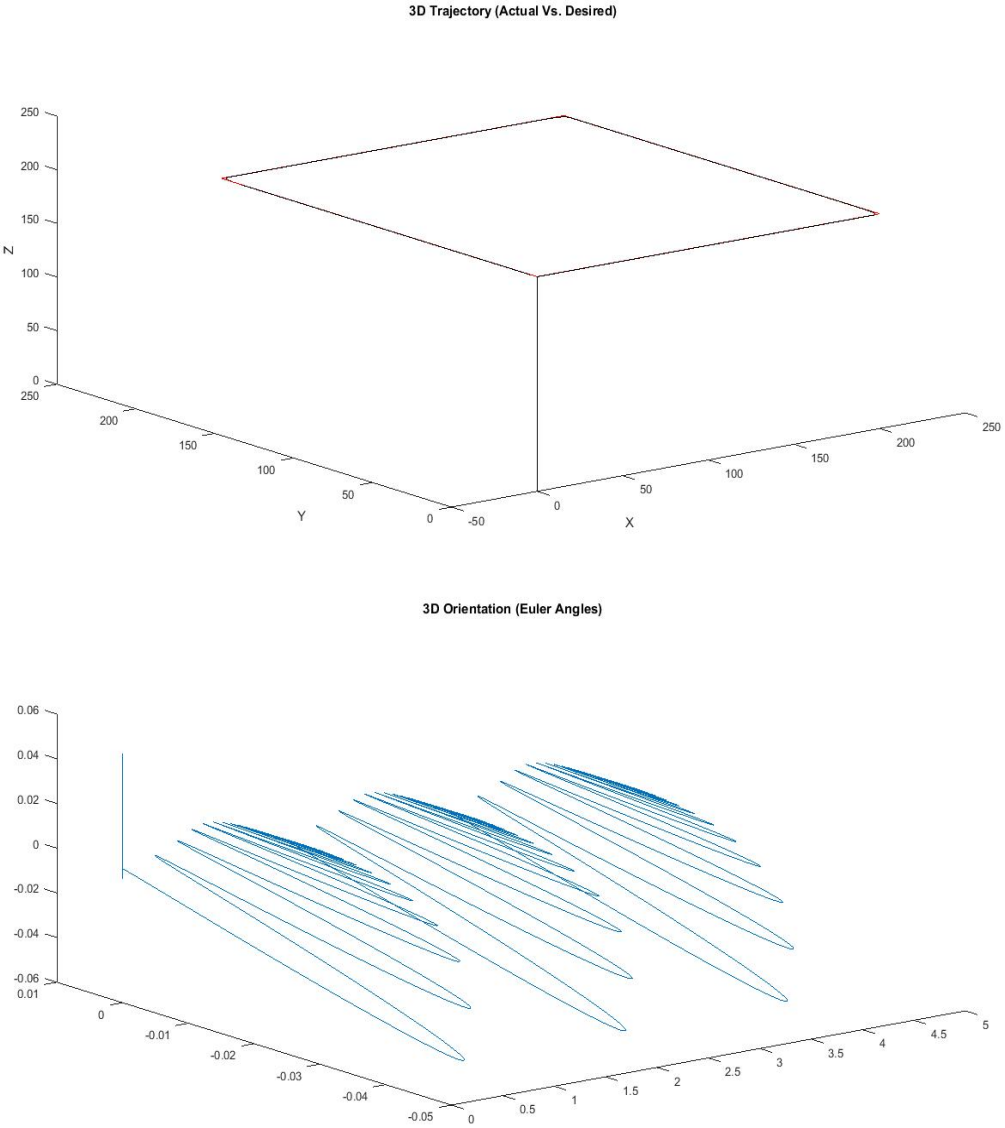
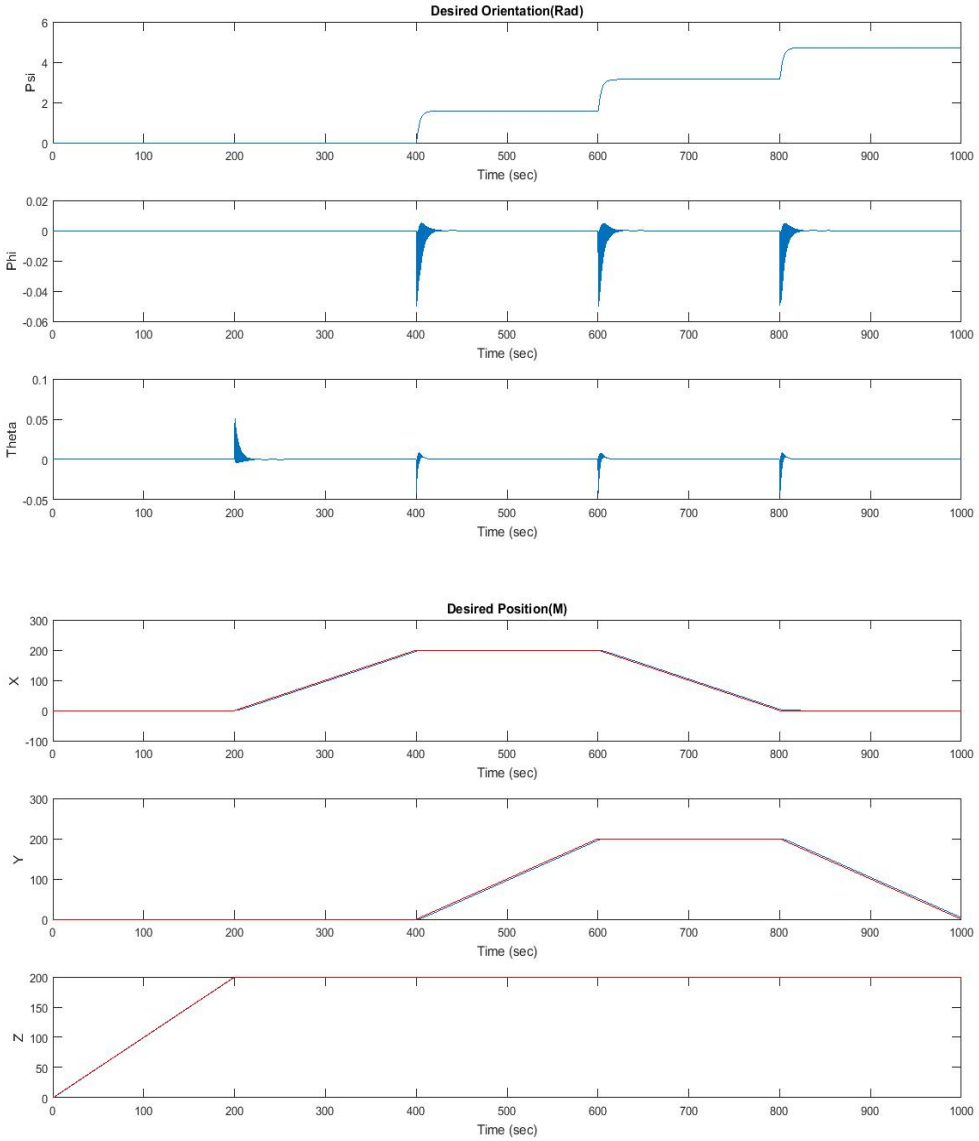


Figure 5.10: Response to rectangular trajectory - Coordinates



Chapter 6

Project Progress

We started by the theoretical and analytical work. First, we looked for the best material and the best frame design to build; then, we chose basic ones, in order to do some modifications later; second, we chose the needed electrical parts: motors, battery, Multiwii, sensors and camera. We ordered parts from many different companies and the shipment took some time. While waiting for the parts, we made some assumptions to assume an approximated weight for our quadcopter, in order to build our Simulink model on MATLAB. We build it using two different methods of control, Lyapunov and PID. Then we did an analytical comparison using MATLAB, between the two methods, and we found that Lyapunov model is more reliable.
We physically build a light test frame made of aluminum, to test the quadcopter before flying it with the original frame. We put together main electronics that contained the sensors and motors that the quadcopter would need to fly.
After testing the motors and ESCs and building the final frame, we needed to find the best control constants for both PID and Backstepping. For PID we used the manual tuning and for Backstepping we used Genetic Algorithm.
The first achieved goal was to control the quadcopter attitude (Roll, Pitch and yaw angles). The quadcopter is able to hover stably in different conditions like windy weather. But it could not maintain in a stable position. The next step was to control quadcopter position. to control the position, we needed a positioning system to feedback the control loop. So, we build a new positioning system combined of image processing techniques and neural networks.

Chapter 7

Experimental Test of Control Methods

7.1 PID Control:

The PID control equation may be expressed as:

$$Drive = k_p * Error + K_I * \sum Error + K_D * dP/dT \tag{7.1}$$

where Error is the difference between the current value of the process variable and the desired set point, usually written as; $Error = Value - Setpoint$, $\sum Error$ is the summation of previous Error values; and dP/dT is the time rate of change of the process variable being controlled, or of the error itself. The proportional coefficientk_p, the integral coefficient k_I , and the derivative coefficient k_D are gain coefficients which tune the PID equation to the particular process being controlled.
To implement a PID controller in the quadcopter control system, the first step is to tune the PID parameters. The parameters was tuned using the following method:

Manual Tuning of PID: For small, low torque motors with little or no gearing, one procedure we can use to get a good baseline tune is to probe its response to a disturbance. What disturbance we use depends on the mechanism the controller is attached to. Normally moving the mechanism by hand away from the setpoint and letting go is enough. (Figure 7.1) If the oscillations grow bigger and bigger then we need to reduce the P gain.
Tuning Procedure:

1. Set all gains to 0.

2. Increase K_D until the system oscillates.
3. Reduce K_D by a factor of 2-4.
4. Set K_P to about 1% of K_D.
5. Increase K_P until oscillations start.
6. Decrease K_P by a factor of 2-4.
7. Set K_I to about 1% of K_P.
8. Increase K_I until oscillations start.
9. Decrease K_I by a factor of 2-4.

If we set the D gain too high the system will begin to chatter (vibrate at a higher frequency than the P gain oscillations). If this happens, then we should reduce the D gain until it stops.

Figure 7.1: PID Tuning Implementation

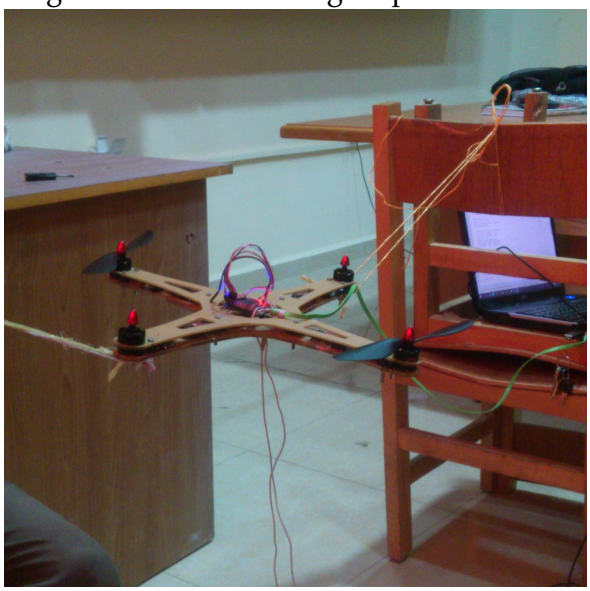


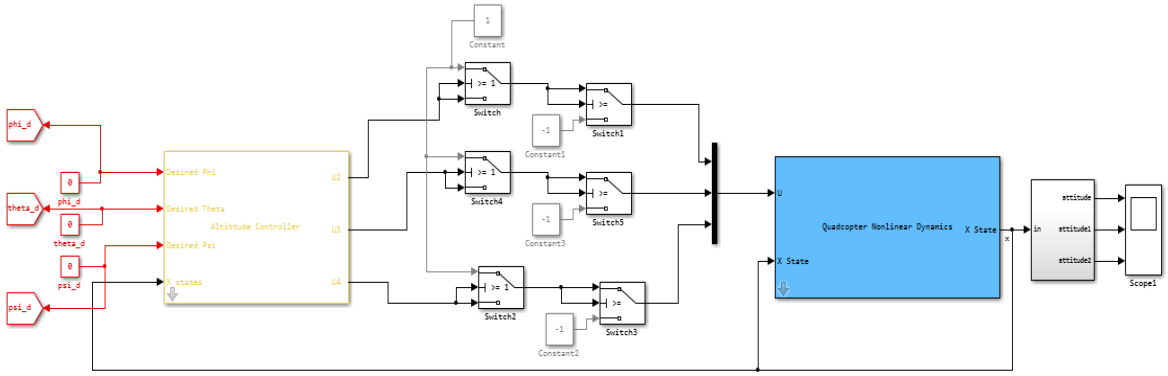
Table 7.1: PID Tuning Results through Test Methods

PID Param	practical method	MATLAB Model
K_P	0.00006	-6.6e-05
K_I	0.002	-5.2e-07
K_D	1.25	-0.000823

7.2 Implementing Genetic Algorithm

GA is very suitable for our nonlinear model of the Backstepping method. So, the parameters and gains of Backstepping controller was tuned using Genetic Algorithm (GA). Computer based simulations were implemented on MATLAB/Simulink and were used to assess the performance of the developed controller.

Figure 7.2: Simulink model for quadcopter



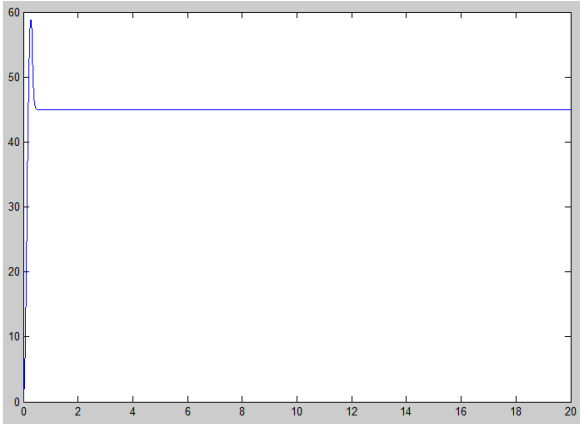
At the end of each iteration the GA will call a performance function to evaluate the current generation. This function returns only one value due to the needed criterion.
At first, the Settling time was chosen to be the performance criterion. (Fig. 7.3) After applying the GA, the best Lyapunov constants were:

Table 7.2: Lyapunov Constants.1

α_1	α_2	α_3	α_4	α_5	α_6
42.3459	35.6359	40.5037	40.5037	42.0623	13.4128

And the step response was:

Figure 7.3: GA Attitude Step Response due to Settling Time criteria



As shown in the previous step response the settling time is very small but the response has a big overshoot.This is not the best response. So, the Overshoot was chosen to be the performance criterion (Fig. 7.4). After applying the GA, the best Lyapunov constants were:

Table 7.3: Lyapunov Constants.2

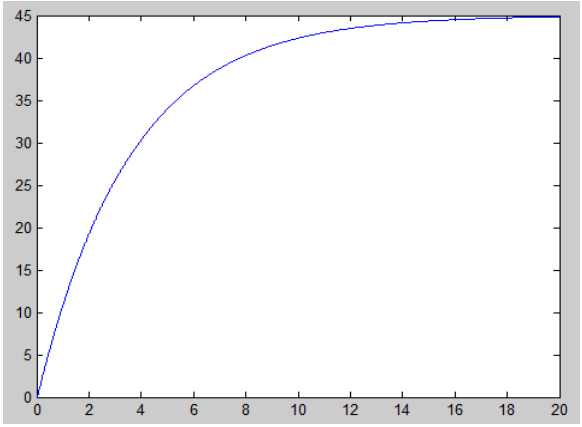
α_1	α_2	α_3	α_4	α_5	α_6
0.2558	37.8605	0.5532	42.9992	0.5336	1.5816

And the step response was:

As shown in the previous step response the response has no overshoot but the settling time gets very big and unsuitable for our system. So, a new criterion was chosen. This criterion considers two parameters: Settling Time and Overshoot.(Fig. 7.5) After applying the GA to the previous Simulink model with the two performance criterions we get the following results as the best Lyapunov needed constants:

Table 7.4: Lyapunov Constants.3

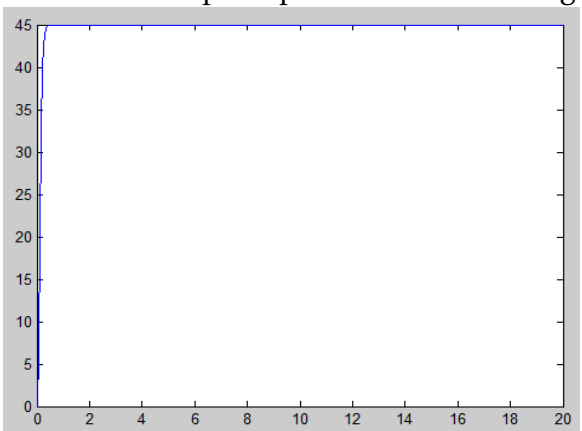
Figure 7.4: GA Attitude Step Response due to Settling Time criteria



α_1	α_2	α_3	α_4	α_5	α_6
23.1974	20.1228	22.5501	23.5565	9.4526	12.9192

And the step response was:

Figure 7.5: GA Attitude Step Response due to Settling Time criteria

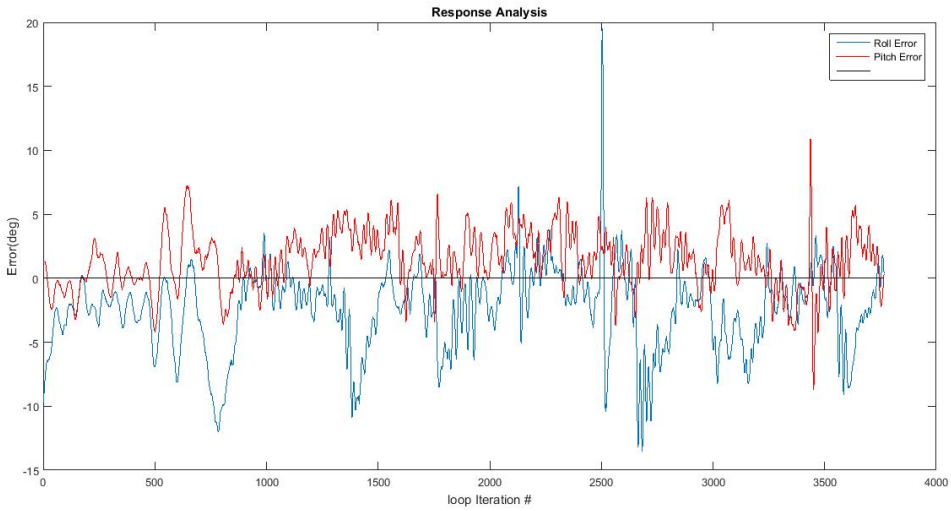


7.3 Comparison between PID and Backstepping Responses

After implementing the previous mentioned control strategies we have analyzed their response in case of free unconstrained hovering and we

have collected some data representing the error in Roll, Pitch and Yaw correction depending on PID and Backstepping Control Algorithms. After importing the error data and analyzing the Bias and Variance in these data set, we obtained these two responses, the first on Backstepping Control method and the second is based on PID Control.

Figure 7.6: Backstepping Response Error over set of data



In figure 7.6, the first 1000 loop iterations contains the response of the first hovering moments, They are less noisy but have large error values. The next range represents the period of flying with a noise ranging from -5 to 5 degrees. The first blue peak at iteration 2000 represents a forced step error on the roll axis and a fast correction response. while the first red peak represents a forced step error on the pitch axis and its correction response.
As mentioned before, this response is based on Backstepping control and Loop time is around 4 milliseconds the Roll error rate is -0.1218 degree per loop iteration and the Pitch error rate is 0.7874 degree per loop iteration.
In figure **??**, the first 500 loop iterations contains the response of the first hovering moments, As we can see, they are less noisy. The next range represents the period of flying with a noise ranging from -10 to 10 degrees. The blue peaks at iteration 1700 to 2000 represents a forced step error on the roll axis and a fast correction response. While the first red peak represents a forced step error on the pitch axis and its correction response.
As mentioned before, this response is based on PID control and Loop time

is around 8 milliseconds, the Roll error rate is -2.4518 degree per loop iteration and the Pitch error rate is 1.4327 degree per loop iteration.
After comparison between both methods, we find that controlling based on Backstepping method is better than PID control and has a robust and fast response to aggressive external noises.

7.4 Yaw and Altitude Responses to Backstepping Control

Figure 7.7 shows Backstepping control response of yaw when the desired angle was set to zero and the initial angle was set to 40 degrees. The response shows a fast correction of yaw in less than a second.

Figure 7.7: Backstepping control response of Yaw

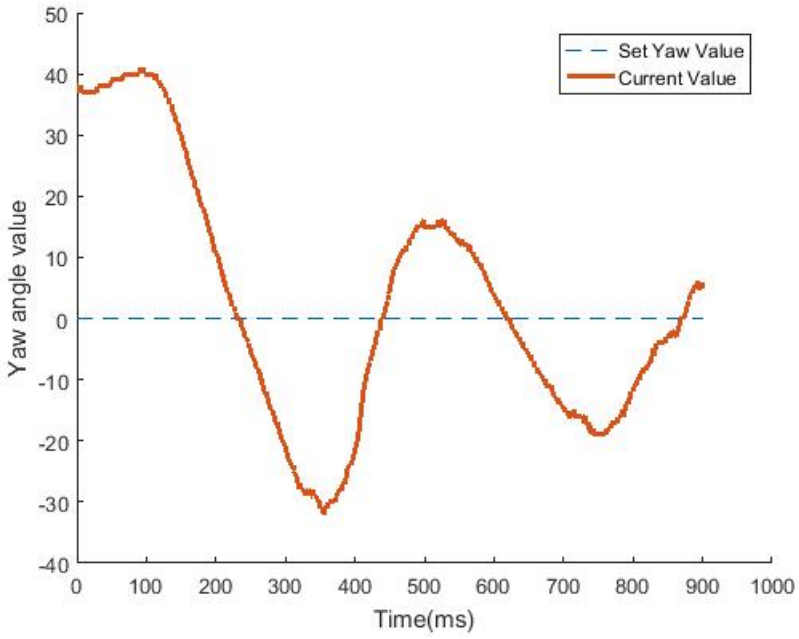


Figure 7.8 shows Backstepping Altitude control response when the desired altitude was set to 150 cm and the quadcopter started hovering from ground. The response shows a steady state error of +-50 cm around the set point, while in Figure 7.9, the response shows shows a steady state error of 75 cm over the set point. Responses differ due to the constants of backstepping control method. The final constants were chosen to give the response of the figure 7.8 because it has a less error.

Figure 7.8: Backstepping control response of Altitude - Good Constants

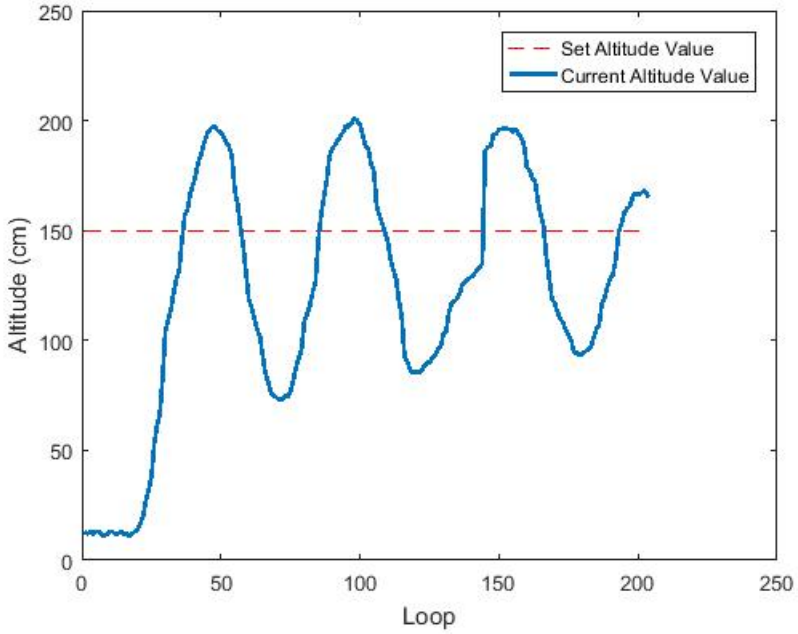
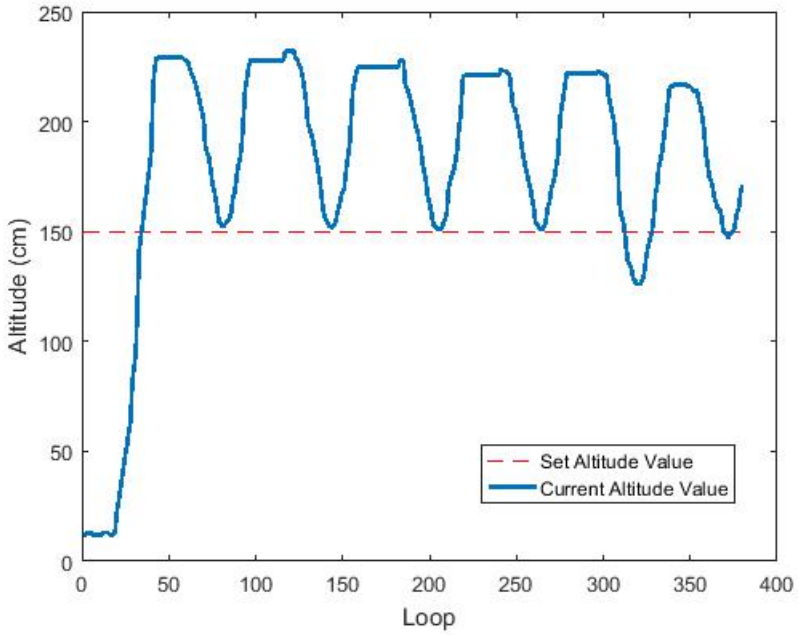


Figure 7.9: Backstepping control response of Altitude - Bad Constants



Chapter 8

Obstacle Detection and Collision Avoidance

Obstacle Detection In the obstacle detection module, redundant ultrasonic sensors are used to increase detection resolution and sensor data reliability. Since ultrasonic sensors have a width dihedral detection angle, the resolution of detected obstacles is very low. The implemented approach uses always two ultrasonic sensors for obstacle detection on X and Y. Hence, due to the double amount of sensors is needed, the redundancy and resolution is also doubled.

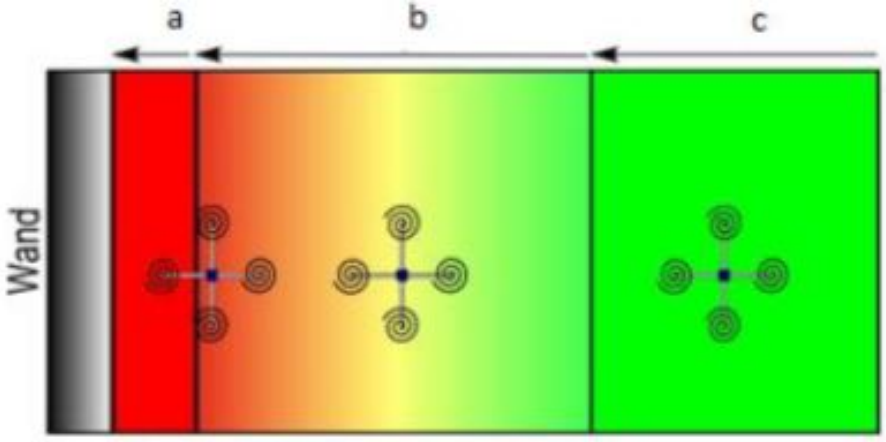
Collision Avoidance The collision avoidance module divides the area around the quadcopter, depending on the measured distance, into three zones for each direction as in figure 8.1:

1. Far or safe zone (green)
2. Close zone (yellow)
3. Dangerous zone (red)

The behavior of the collision avoidance module can be described best with a state machine (Figure 9.2). Initially, if the autonomous collision avoidance is off, the quadcopter is in state 0. If the autonomous mode is activated, the quadcopter can switch between states 1, 2 or 3 depending on the measured distance to a nearby object. State 1 (safe zone) is active, if there is no obstacle nearby (object distance > a+b).If an obstacle is detected in the close zone (a > object distance > a+b), state 2 (close area) is activated and the corresponding pitch or roll angle towards the obstacle

is limited depending on the measured distance reducing the speed of approach. In the dangerous zone (a < object distance), state 3 is activated and the distance to the obstacle is controlled using a PID controller or a back-stepping controller, preventing a further approach to the obstacle. Hence, such a state machine is necessary for every direction.[14]

Figure 8.1: Collision Avoidance - Distance Zones



Chapter 9

Building a Low-Cost Indoor positioning system to control the trajectory of the quadcopter

Drone positioning has been subjected to extensive research regarding real-time measures to establish the best control. The most common positioning technique used in drones is the global navigation systems. However, global navigation satellite systems (GPS or GNSS) are generally not suitable to establish indoor locations, since microwaves are attenuated and scattered by roofs, walls and other objects. In order to make positioning signals ubiquitous, indoor positioning system can be used. Most common indoor positioning systems are based on either wireless technologies or non-radio technologies. The wireless technologies based systems can provide increased accuracy at the expense of costly equipment and installations. However, the kind of indoor positioning system depends on the type of application; for an autonomous quadcopter project, quadcopters need a positioning system to hold a position or to follow a specific path. A real-time cost-effective indoor positioning system with a precision of a few centimeters for the X, Y and Z axes is needed. As a result, a real-time 3D positioning system based on known visual features and three trained neural networks were developed; using two low-cost IP cameras. The system has accomplished a precision of ±5cm and a cycle speed of 60 ms.

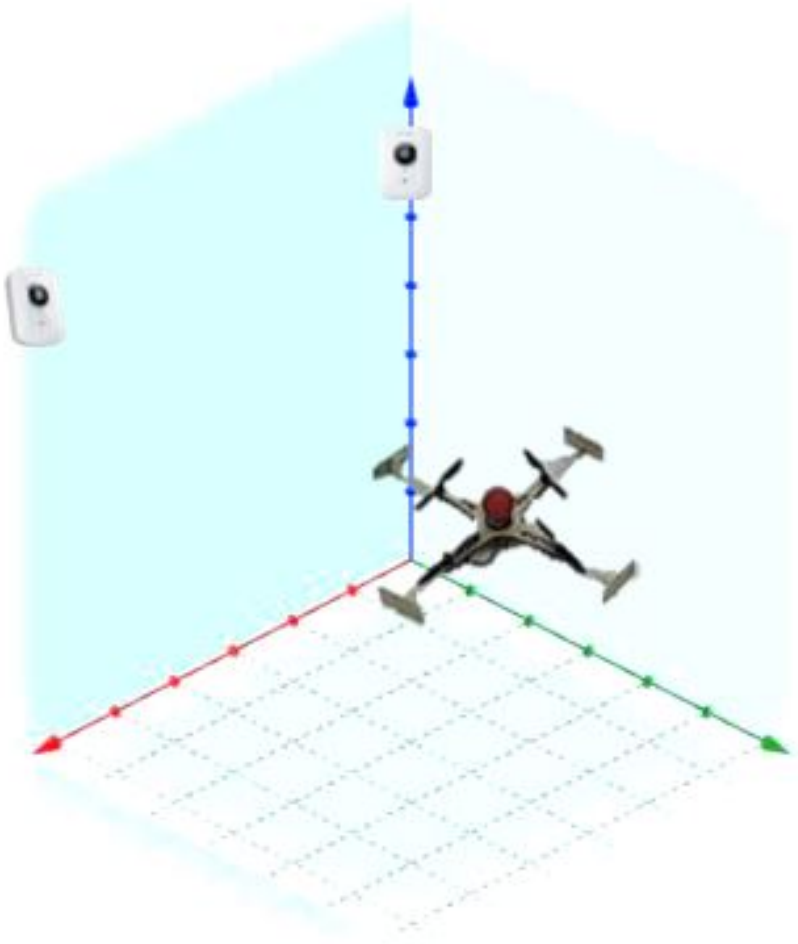


Figure 9.1: 3D representation of the Positioning system

9.1 Detecting the quadcopter using image processing techniques

We added an object of interest to our quadcopter to make its detection and tracking easier.The object is chosen to be a sphere with a suitable volume because it is shape-invariant due to different perspectives. The sphere has a specific color to simplify image processing operations (red color in our case). This will simplify detection process and reduce the overall time required to do this procedure. As known, Controlling a quadcopter requires a real time system which is capable of providing position parameters with no lower rate than 16Hz. So, we need to reduce the time cost of computing

these parameters as much as we can. Detecting an object requires a procedure based on several image processing techniques which were done using *OpenCV* libraries. Basically, we used two TP-Link IP cameras which were fixed rigidly to cover the required workspace and the process was done off-board using *Pentium4 core-i3* processor. As shown in the previous figure. First, video frames were grabbed from each camera and converted from RGB into HSV in a rate of *30fps* (which will limit the maximum rate to 30Hz). See Figures 9.2. The main idea of converting to HSV is to isolate the object of interest by its Hue value.



Figure 9.2: (a) Original Camera Frame - (b) Frame after converting to HSV- (c) Thresholded image - (d) Final image after Eroding and Dilating

Then, Gaussian filter with a suitable mask was used to blur the resulting image and enhance the edges of the target object to reduce the variation in edge pixels due to light density changes after thresholding. The next step is to threshold the resulting image to keep only the object of interest (a red sphere in this case). To eliminate unwanted small areas or noise in the

resulting image, Eroding, Dilating, and connected components morphology operations were done [15]. X and Y pixel coordinates of the center of the object are used in the proposed positioning system. Moments of the image and object area were calculated as equation (1) and used to find the center of mass which represents the location of the object of interest in each frame.

$$M_{ij} = \Sigma_x.\Sigma_y((I(x,y).x^i.y^j)) \tag{9.1}$$

where $I(x,y)$ is pixel intensity. [15]

9.2 Building a 3D positioning system using two IP Cameras and Neural Networks

To build the positioning system, three neural networks were developed, for X, Y and Z. Each network has four inputs: x_1, y_1, x_2, y_2 where x_i, y_i are the pixel coordinates of object center in the image frame grabbed from each camera. Every neural network is a feed-forward network and has one hidden layer. The hidden layer has four neurons. (Fig. 9.3) Each network has one output: the real position along one of the three cartesian axis X,Y and Z in our virtual 3D system.

9.2.1 Neural network structure:

Figure 9.4 shows the main structure of a neural network with one hidden layer and one output layer: Where: P is the input vector
W_i is the weights matrix of the layer i.
b_i is the bias matrix of the layer i.
S_i is the number of layer i neurons.
The hidden layer had been chosen to have four neurons (Fig. 9.5) which were enough to build the relationship between the four inputs and the one output X, Y or Z. The output of the hidden layer:

$$a^1 = f^1(n^1) \tag{9.2}$$

where $n^1 = W^1P + b^1$ and f^1 is Hyperbolic Tangent Sigmoid

$$a^1 = f^1(W^1P + b^1) \tag{9.3}$$

$$a^1 = tansig(W^1P + b^1) \tag{9.4}$$

The output of the output layer:

$$a^2 = f^2(W^2a^1 + b^2) \tag{9.5}$$

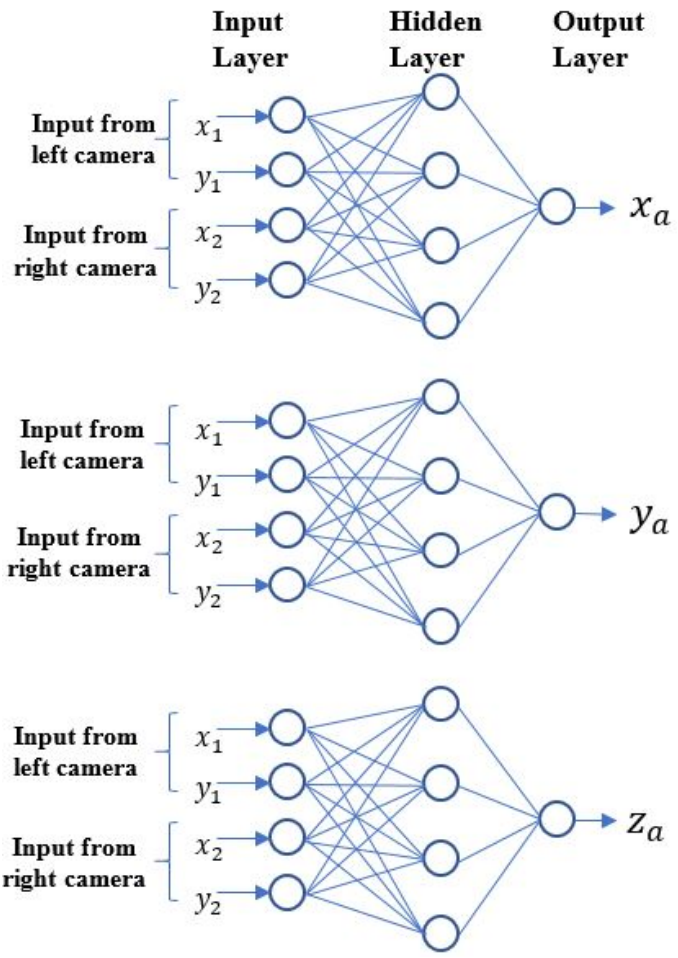


Figure 9.3: NN Positioning System structure

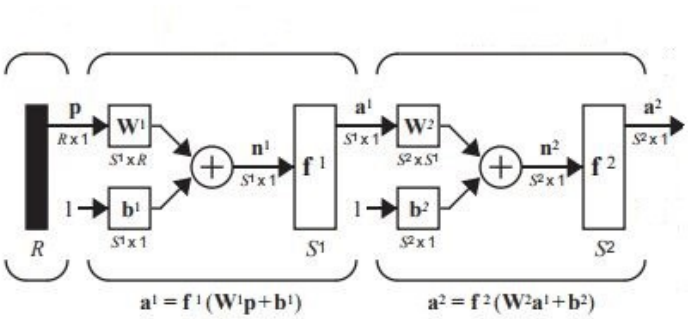


Figure 9.4: Main structure of a neural network

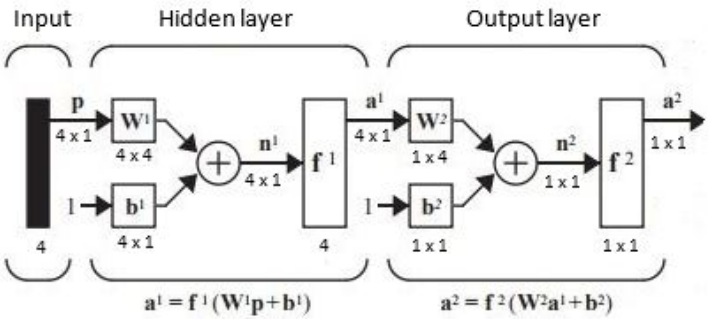


Figure 9.5: A neural network with 4 hidden neurons

Where f^2 is the linear function $f(n) = n$, the final output will be:

$$a^2 = W^2 a^1 + b^2 \tag{9.6}$$

The activation function *Tansig* uses the following output equation:

$$a = 2/(1 + e^{-2n}) - 1; \tag{9.7}$$

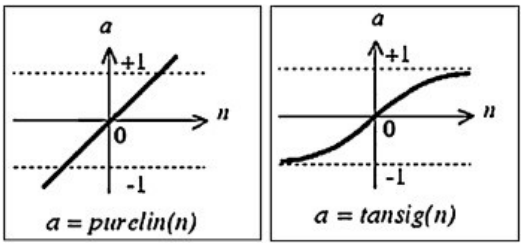


Figure 9.6: Pure Linear Function - Hyperbolic Tangent Sigmoid Function

9.2.2 Training Algorithm- Levenberg-Marquardt:

After building the three NN for X,Y and Z, networks had been trained using Levenberg-Marquardt Algorithm [16], [17]. The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed-forward networks), then the Hessian matrix can be approximated as $H = JTJ$ and the gradient can be computed as: $g = JTe$ where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the

weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [JTJ + \mu I] - JTe \tag{9.8}$$

This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights). It also has an efficient implementation in MATLAB software, so this training algorithm has been used to train our neural network.
Training function: TRAINLM
Adaption learning function: LEARNGDM
Performance function: MSE

Table 9.1: Training Parameters

Epochs:	**1000**
Goal:	0
Min_grad:	1e-07
Max_fail:	6
μ:	0.001
μ_dec:	0.1
μ_inc:	10

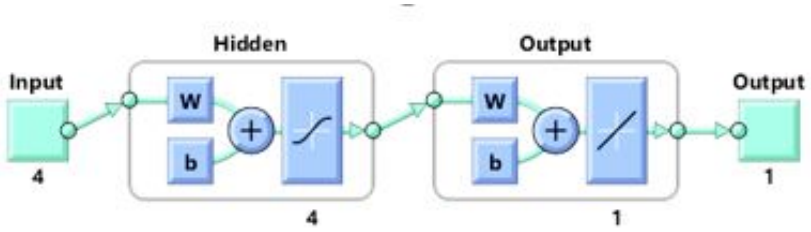


Figure 9.7: Neural network structure (MATLAB)

9.2.3 Validation and testing:

Over than one hundred data samples were taken manually by measuring the real distance of the object along a virtual 3D space constructed in our application workspace. NNs were trained on 80% of samples and

validated on 10% samples and tested on the remaining 10% of samples. MATLAB software and OpenCV libraries were used to construct and train our NN based system to get the appropriate weights and offsets for each neuron.
The results of training, validation and testing are shown in figures 9.8 and 9.9.

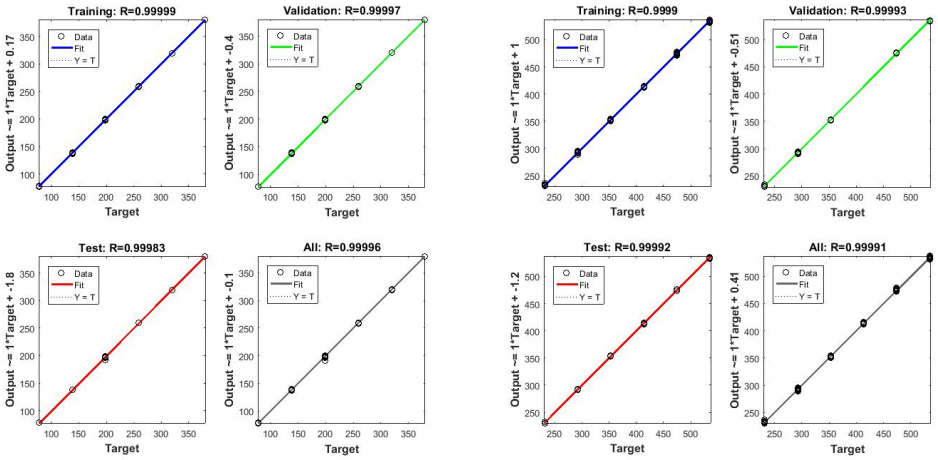


Figure 9.8: Regression related to X and Y axes NNs

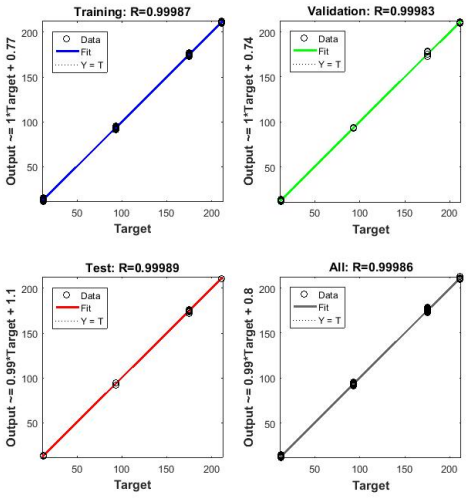


Figure 9.9: Regression related to Z axis NN

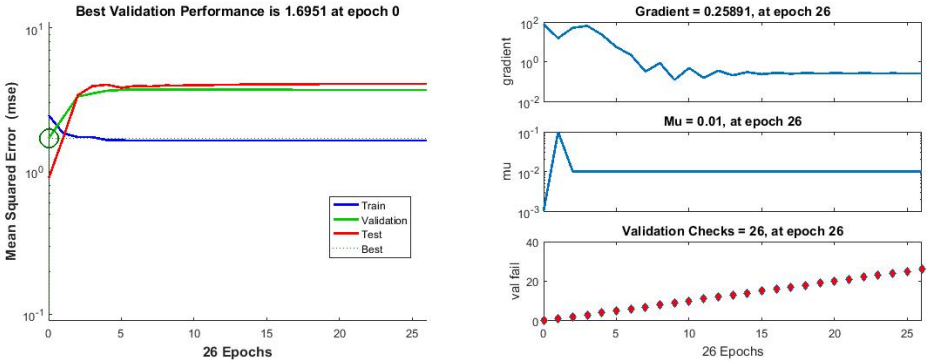


Figure 9.10: Convergence and Training Performance on X-data

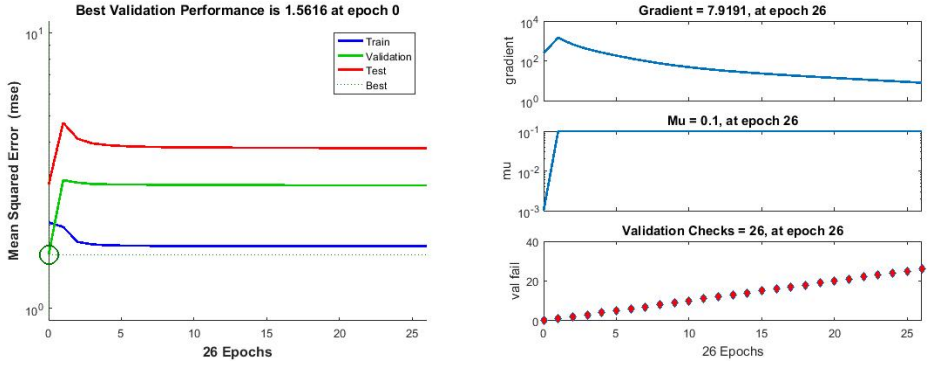


Figure 9.11: Convergence and Training Performance on Y-data

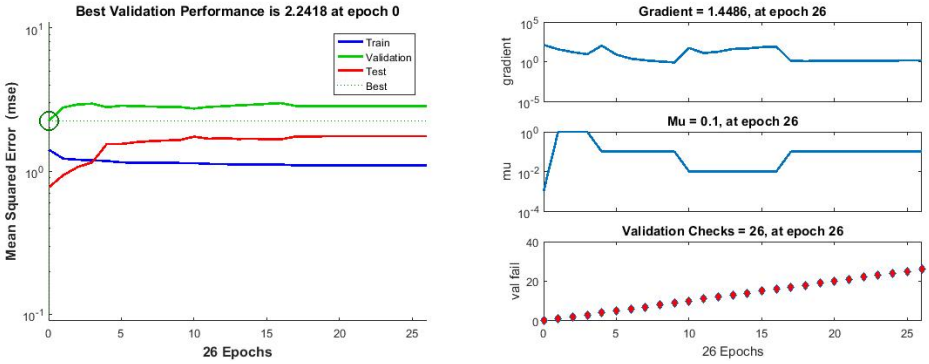


Figure 9.12: Convergence and Training Performance on Z-data

9.2.4 Real-time results:

The positioning system performances on X, Y and Z axes for the flying quadcopter have been tested and the results are shown in following fig-

ures. Showing that the developed positioning system has good-precision of few centimeters which was a satisfying precision to control the quadcopter in real time.

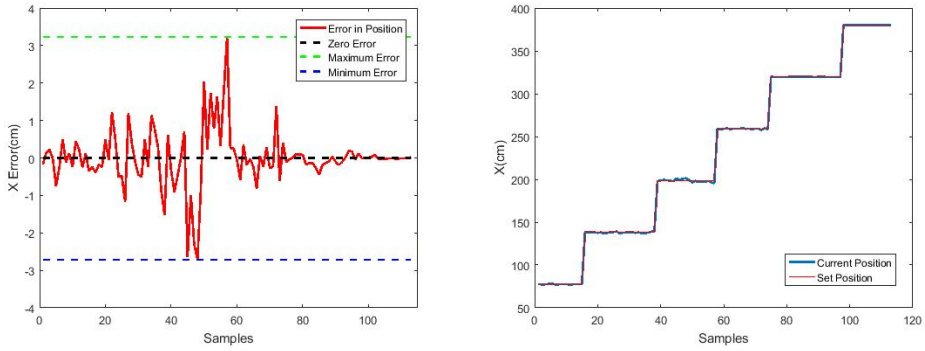


Figure 9.13: Current Position Vs Set Position along X-axis(cm)

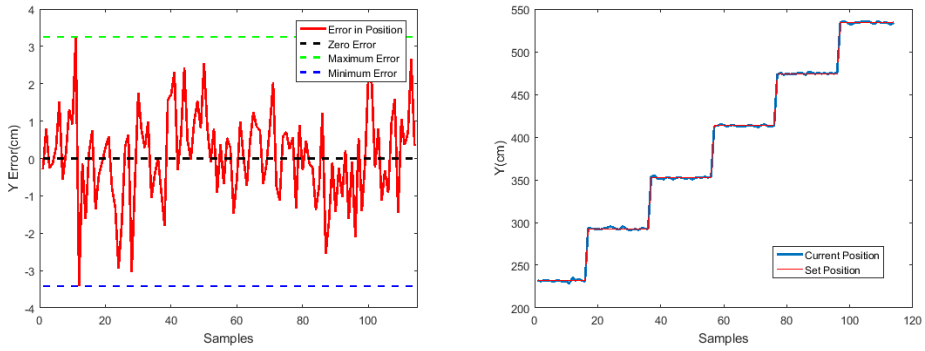


Figure 9.14: Current Position Vs Set Position along y-axis(cm)

Testing this positioning system in a real-time application of quadcopter on new data sets and inputs has shown a similar precision to the training results and proved the validity of using this method indoor positioning. Figure 9.16 .a shows the real drone trajectory VS the output of the positioning system. Figures 9.16 .b-d represent a projections on X-Y plan[X=const], X-Y plan[Y=const] and Y-Z plan, respectively. results shows that the system has a precision of ±1 cm on X NN positioning, a precision of ±4 cm on Y NN positioning and a precision of ±3 cm on Z NN positioning.
Tests have demonstrated the validity of the proposed neural-network and image-processing based technique to estimate the real-time 3D position of a flying quadcopter.

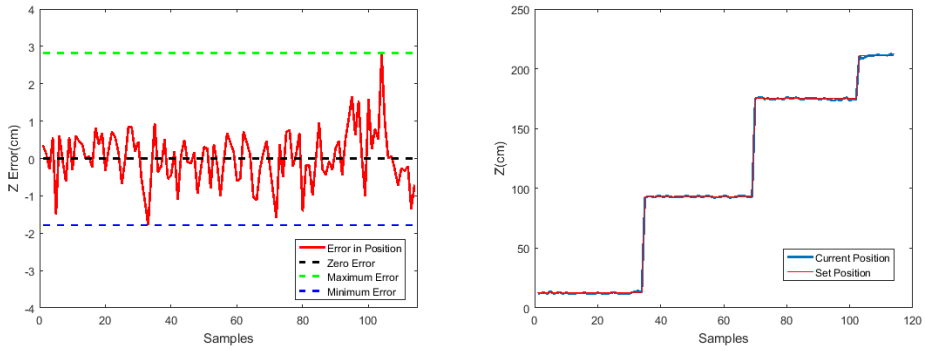


Figure 9.15: Current Position Vs Set Position along Z-axis(cm)

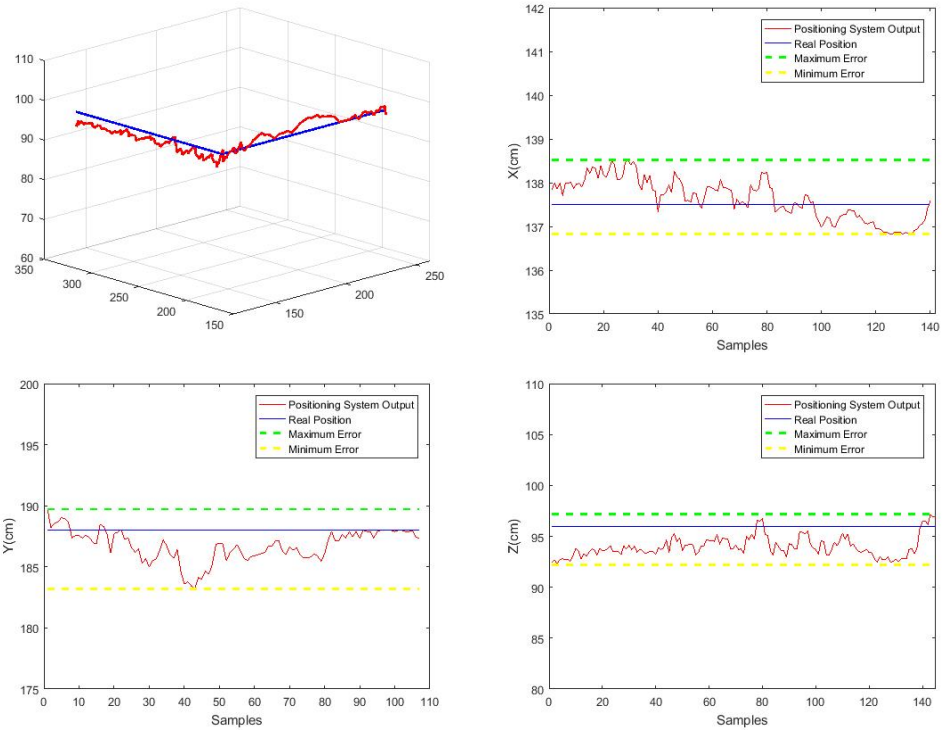


Figure 9.16: a) Simple flight trajectory - b) X-Y plan projection[X=const] - c) X-Y plan projection[Y=const] - d) Y-Z plan projection

First, the used image processing techniques to detect the object were introduced. Then, the network structure and learning algorithm of the proposed NN based positioning system were introduced in detail. Finally, the positioning system performances on X, Y and Z axes for the flying quadcopter were tested. We have demonstrated that the proposed positioning

system is a low-cost good-precision positioning system with a precision of few centimeters capable of controlling the position of the quadcopter in real time.

Chapter 10

Fuzzy Altitude Control for Quadrotor with Dynamic Feed-forward Compensation

Fuzzy Altitude Control Principle As concluded in the quadcopter modeling section, one can obtain acceleration along Z axis or the altitude dynamics as

$$\ddot{z} = g - (cos\phi.cos\theta)U/m \tag{10.1}$$

Where U is the total thrust and torques produced by the four rotors. Normally, quadcopter control systems use static forward gravity compensators. The compensation term $U_{FF} = m.g$ can be obtained from the previous equation by assuming that ψ and θ are zeros. (which means the quadcopter is in hovering mode) and setting $\ddot{z} = 0$. However, calculating feed-forward term this way leads to insufficient when angles ψ and θ are different from zero. (which will occur in all flight modes except hovering). The insufficiency of the gravity compensation may lead to a decrease in UAV altitude. Known as (altitude loss). In addition, the loss of altitude will increase for bigger angles ψ and θ , although their maximal values are usually restricted in most practical quadcopter UAVs. Therefore, we propose the use of a dynamic feed-forward compensator term that will change with the roll and pitch angles according to

$$U_{FF} = mg/((cos\psi.cos\theta)) \tag{10.2}$$

If the roll and pitch angles are zeros, the proposed compensator behaves identically to the conventional static gravity compensator. However, if the roll and pitch angles start to change, the value of the feedforward term increases, which reduces the drop in the UAV altitude. In this manner,

it is possible to decrease the loss in altitude when the quadcopter flight mode transitions from hovering to normal flight.

Design of Fuzzy Altitude Controller To improve the performance of altitude control, the PD and the backstepping altitude controllers was replaced by a fuzzy logic controller. The proposed fuzzy logic controller provides a faster response with less overshoot by allowing different system parameters to exhibit their most advantageous features at different times. In the proposed fuzzy logic controller, the desired height value z^* is compared with the actual height z of the UAV. The difference between these two values is called the position error (e_z). The slope angle θ_{e_z} of the error derivative, or the error change rate, can be then calculated as:

$$\theta_{e_z} = atan(\delta e_z/\delta t) \tag{10.3}$$

The inputs of the fuzzy controller are the position error e_z (proportional action) and the theta error θ_{e_z} (derivative action). The output of the proposed fuzzy controller U_{Fuzzy} is multiplied by a gain

$$K_F = 0.25 * U_{FF} \tag{10.4}$$

Both inputs and the output of the controller are designed to have five membership functions, namely negative large (NL), negative small (NS), zero (Z), positive small (PS), and positive large (PL). The membership functions of the inputs and output are designed using MATLAB Software and shown below respectively:



Figure 10.1: Fuzzy logic Altitude controller System Diagram

The fuzzy controller output indicates how much we need to add to the initial feedforward speed. The output thrust due to the height controller is calculated as the equation below:

$$U_{=}mg/(cos\psi.cos\theta) + K_F.U_{Fuzzy} \tag{10.5}$$

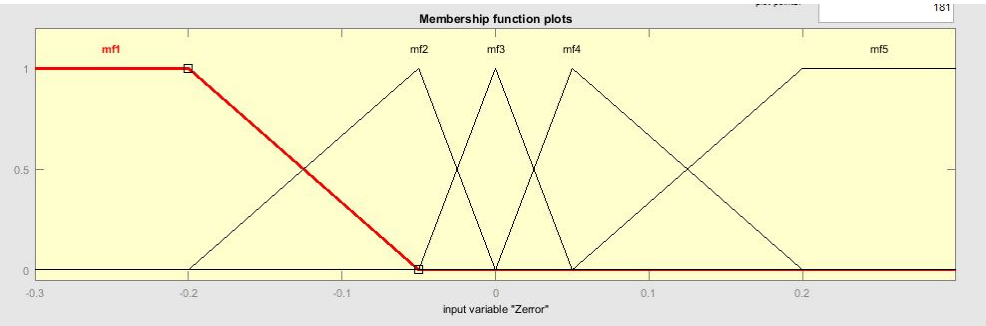


Figure 10.2: The membership functions of the Fuzzy System Z error input

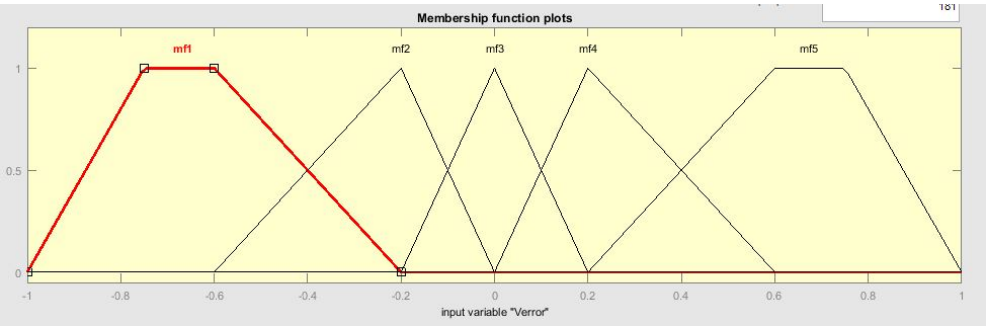


Figure 10.3: The membership functions of the Fuzzy System Z error rate input

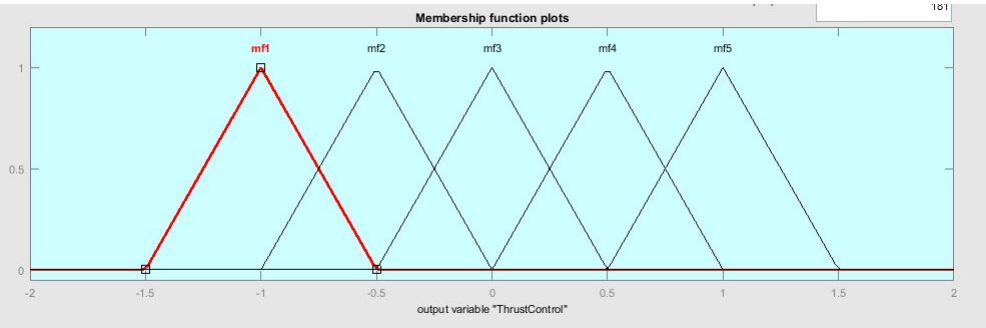


Figure 10.4: The membership functions of the output U

It should be noted that, in the proposed rules, "negative error" means that the value of the actual position is greater than the one of the desired position. Therefore, in this case, we need to decrease the speed of all four motors to reduce the thrust force. On the contrary, "positive error" means that the value of the actual position is lower than the one of the desired position; thus, the speed of the motors should be increased to create a larger

thrust force. Using these restrictions, the rules of the fuzzy controller are shown in table 10.1. Fuzzy inference generated using the Mamdani method is used. The used method of defuzzification is centroid method.

Table 10.1: Rule Set

θ_{e_z}/e_z	**NL**	**NS**	**Z**	**PS**	**PL**
NL	PL	PL	PL	PS	Z
NS	PL	PL	PS	Z	NL
Z	PL	PS	Z	NS	NL
PS	PS	Z	NS	NL	NL
PL	Z	NS	NL	NL	NL

The output fuzzy surface according to the fuzzy inputs is shown in figure 10.5.

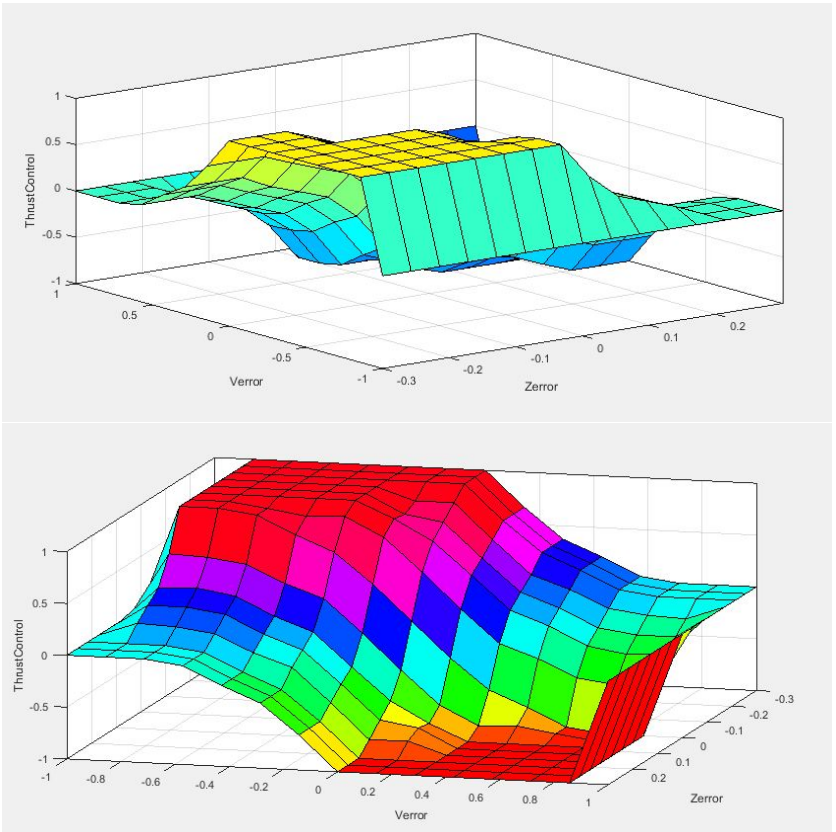


Figure 10.5: Rules Surface

Experimental Results Figure 10.6 shows the quadcopter altitude response to the fuzzy controller output as the desired height is set to 100 cm. Although there is a big overshoot and a big settling time, too; it is converging better to the set point compared to normal feed-forward compensator.

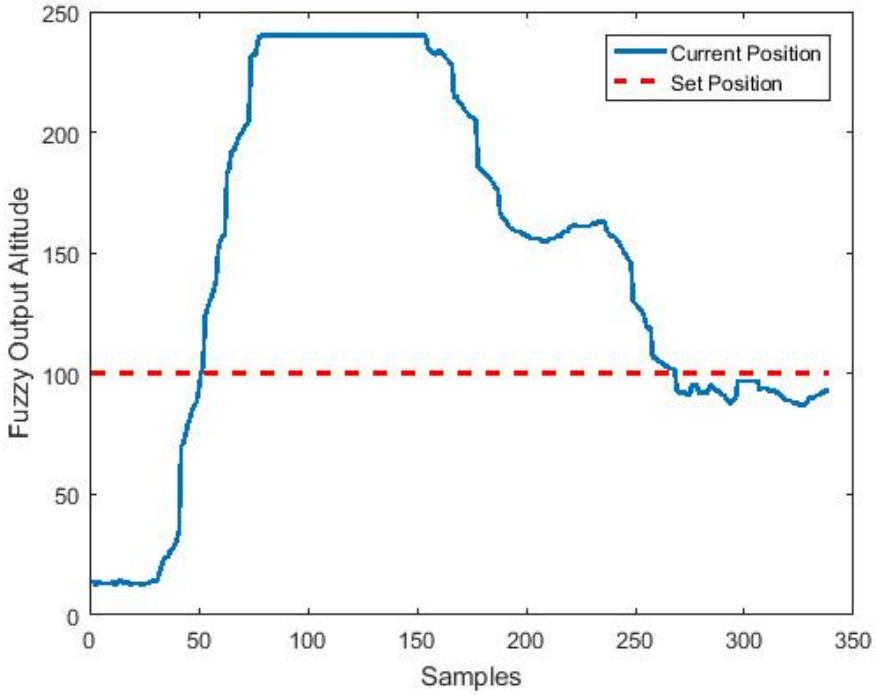


Figure 10.6: Fuzzy Altitude Response Test

Appendices

Appendix A

Stress Analysis Figures

Figure A.1: Force Distribution and Fixing Methods

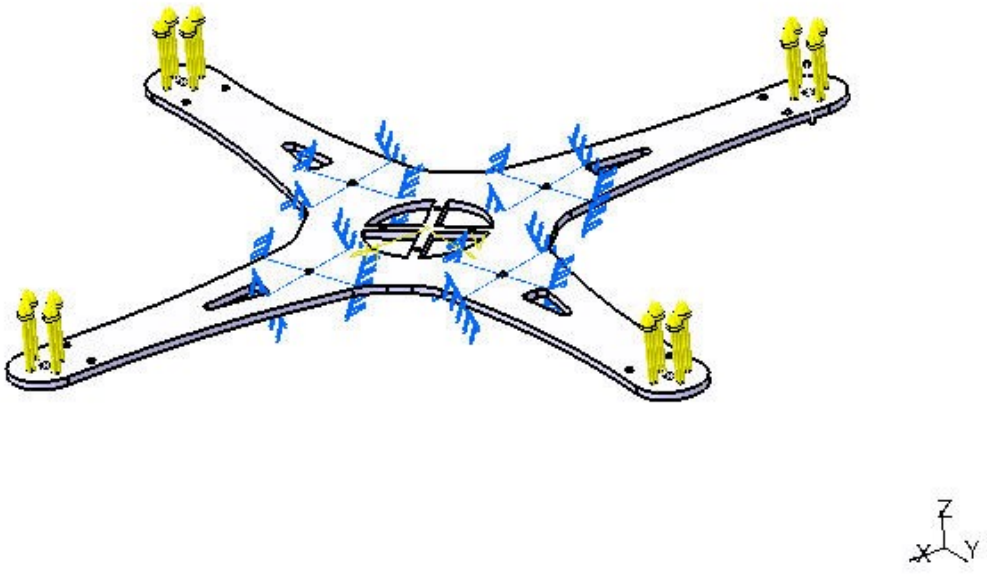


Figure A.2: Static Case Solution.1 - Deformed mesh.2



Figure A.3: Static Case Solution.1 - Von Mises stress (nodal values).2

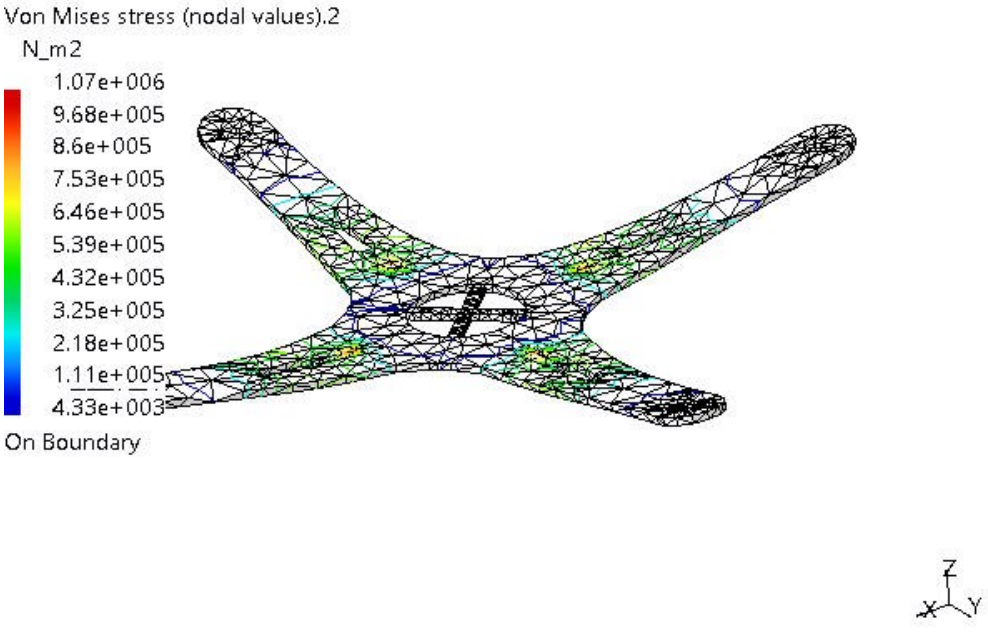


Figure A.4: Static Case Solution.1 - Deformed mesh.1



Figure A.5: Static Case Solution.1 - Von Mises stress (nodal values).1

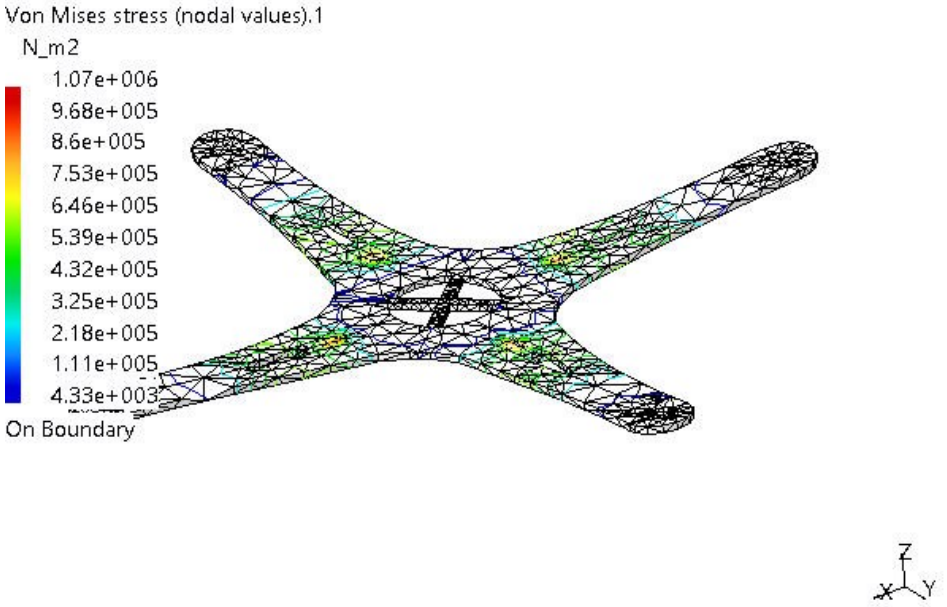


Figure A.6: Static Case Solution.1 - Translational displacement vector.1

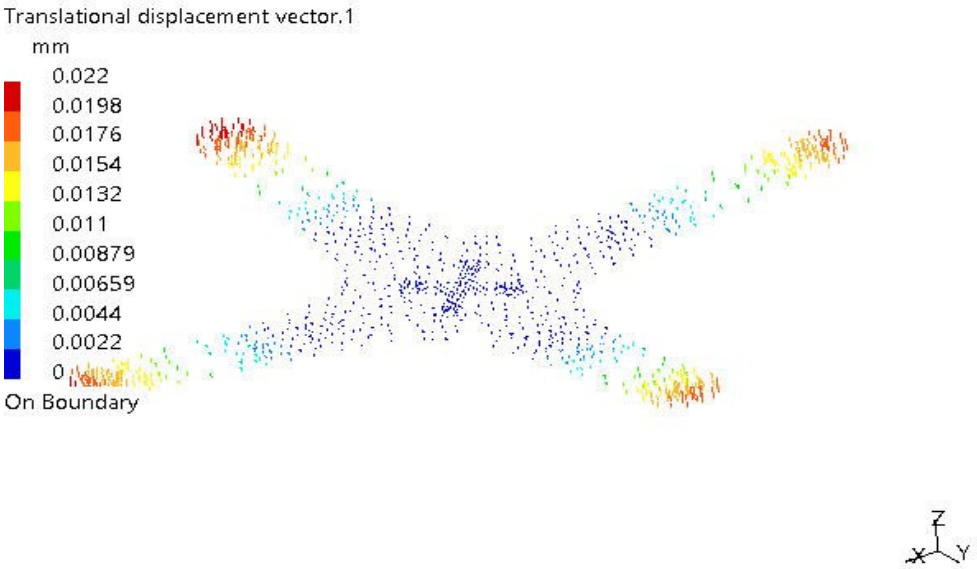


Figure A.7: Static Case Solution.1 - Estimated local error.1

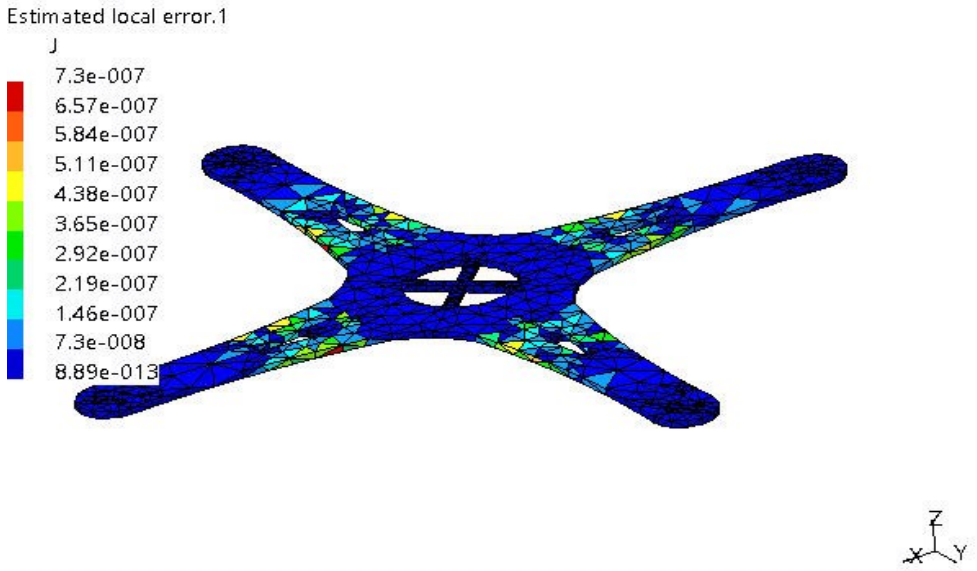
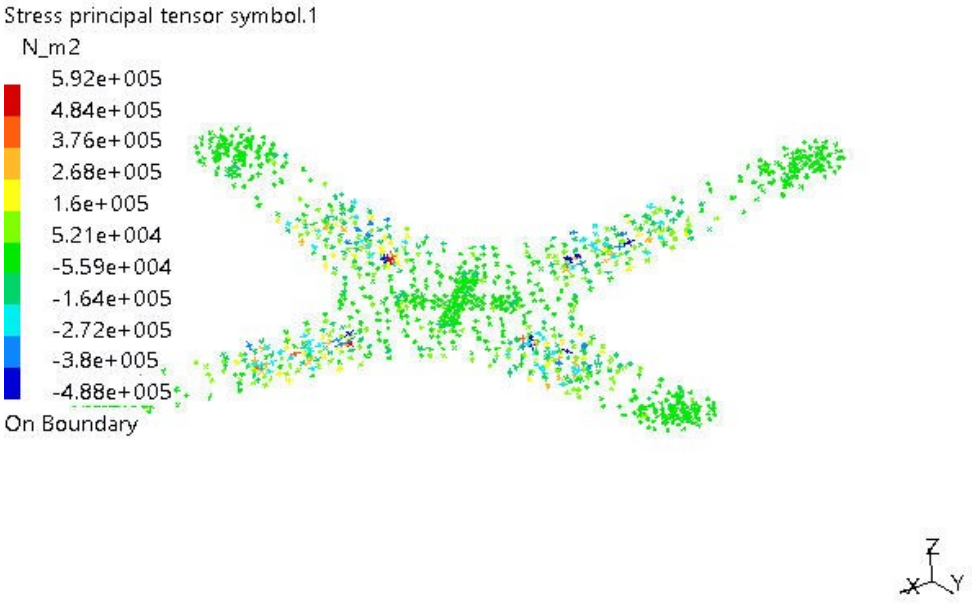


Figure A.8: Static Case Solution.1 - Stress principal tensor symbol.1



Appendix B

List of Symbols

b thrust factor
C_Q drag coefficient
C_T ethrust coefficient
d drag factor
g acceleration due to gravity
H hub force
i motor current
Ixx, yy, zz inertia moments
J_m motor inertia
J_r rotor inertia
J_t total rotor inertia seen by the motor
t_a torque in body coordinate frame
t_d motor load
t_m motor torque
v induced velocity
ψ roll angle
ϕ yaw angle
w body angular rate

Bibliography

[1] *"Robot Television Bomber"*,Popular Mechanics,June 1940.

[2] Fred Kaplan,*"The World as Free-Fire Zone"* . MIT Technology Review. Retrieved June 17, 2013.

[3] *"Research and Education in Robotics'* - EUROBOT 2011 International Conference, Prague, Czech Republic, June 15-17, 2011.

[4] By Michael Melle, for SRI International, Aug 7, 2013

[5] By Brian Handwerk , for National Geographic , Dec, 2012

[6] by Phil for Humanity , Feb,2013

[7] Pounds, P.; Mahony, R.; Corke, P.*"Modelling and Control of a Quad-Rotor Robot"* (December 2006).

[8] by Lucas M.A, Willian C.R, Paulo E.S and Renato A.A.*" PID, LQR and LQR-PID on a Quadcopter Platform"*

[9] Atheer L. Salih, M. Moghavvemi, Haider A. F. Mohamed and Khalaf Sallom Gaeid. Centre for Research in Applied Electronics (CRAE), Electrical Engineering Department, University Malaya Kuala Lumpur, Malaysia.

[10] Samir Bouabdallah and Roland Siegwart.*Autonomous Systems* Lab Ecole Polytechnique FÂ´edÂ´erale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

[11] Ankydbbb.a Ji and Kamran Turkoglu.*" Development of a Low-Cost Experimental Quadcopter Testbed Using an Arduino Controller and Software"*

[12] `http://www.hobbyking.com/hobbyking/store/__43884__MT2213_935KV_MultiStar_Motor_and_Propeller_Combo_10x4_5_CW_CCW.html`

[13] S. Bouabdallah and R. Siegwart,"*Backstepping and sliding-mode techniques applied to an indoor micro quadrotor*", Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005, pp.2247-2252.

[14] Autonomous Flying Robots, Spring 2010, Kenzo Nonami

[15] Gonzalez R, Woods R. Digital Image Processing. Prentice Hall, 2008.

[16] Argonne National Lab, United States. Department of Energy. Levenberg-Marquardt Algorithm: Implementation and Theory. Argonne National Lab, 1977.

[17] Martin T. Hagan and Howard B. Demuth, Mark Hudson Beale, Orlando De Jesus,? Neural Network Design?, 2nd Edition, Verlag Martin T. Hagan, 1. September 2014.