Efficiency Based Flight Analysis for a Novel Quadcopter System

by

Harsh Lal

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Panagiotis Artemiadis, Chair
Hyunglae Lee
Wenlong Zhang

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

For a conventional quadcopter system with 4 planar rotors, flight times vary between 10 to 20 minutes depending on the weight of the quadcopter and the size of the battery used. In order to increase the flight time, either the weight of the quadcopter should be reduced or the battery size should be increased. Another way is to increase the efficiency of the propellers. Previous research shows that ducting a propeller can cause an increase of up to 94% in the thrust produced by the rotor-duct system. This research focused on developing and testing a quadcopter having a centrally ducted rotor which produces 60% of the total system thrust and 3 other peripheral rotors. This quadcopter will provide longer flight times while having the same maneuvering flexibility in planar movements.

ACKNOWLEDGEMENTS

I would like to thank Dr. Panos Artemiadis for being a great advisor and always having faith in me during my thesis. I thank him for giving me confidence whenever I faced difficulties with my project. I would like to thank my friends and peers at the HORC Lab who were there to share their ideas on my project. I would also like to mention all those people at the 3D print labs at ASU who helped me print the parts for my project. I really appreciate their help.

Lastly, I would dearly like to thank my parents for their support during these past 2 years. I thank my dad for being a great mentor to me. I'm grateful for all the time and effort he spent discussing my project which helped me increase my understanding of it.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

In a world of robots, Unmanned Aerial Vehicles (UAVs) have gained the highest popularity amongst people in almost every field of work. They can sense their surroundings and maneuver autonomously while collecting data using onboard sensors and cameras. Their low cost and availability are also a major factor for their popularity. Among all the UAVs, quadcopters provide the maximum ability to operate in cluttered environments and provide the highest efficiency during hover. Commercial quadcopters like the DJI Mavic are used for photography and video surveillance due to their camera quality and, more importantly, their stability in flight. These drones can also be used for inspection missions or weather and crop monitoring, and security surveillance. Unlike ground robots, quadcopters have the ability to move in 6 degrees of freedom which makes them extremely useful during search and rescue operations as they can easily move through buildings and rubble during a natural disaster. This technology is also being developed for package delivery by companies like Amazon, with their Prime Air delivery drones being proposed to deliver packages to people's homes in 30 minutes or less.

The simplicity of quadcopter design makes it easy for manufactures to develop

Figure 1.1: Conventional Quadcopter

drones tailored to specific applications. Conventional quadcopters, having a rigid frame with 4 extended arms to mount the rotors, come in various sizes and shapes. Small sized drones show the maximum amount of agility and can maneuver easily in cluttered environments. It makes them popular among drone enthusiasts who do not have a lot of experience flying drones and are just looking for something to satisfy their interests. These include mini drones like the Crazyflie Nano Quadcopter to larger ones like the Parrot Bebop. Big sized drones like the DJI Mavic and DJI Phantom are most common among professional photographers who are experienced flyers. But the main issue with these drones is their battery life which lasts for only 20 - 30 minutes of flight time. This makes them ill-suited for applications involving long range travel like aerial surveillance and recon missions. Increasing the size of the quadcopters just leads to additional weight due to bigger rotors, frames and batteries and would not lead to any significant change in the flight time of the drone. So, there arises a need to develop a system which can increase the flight time of a quadcopter without compromising on its weight. One such area where we can improve the performance of a quadcopter is by increasing the efficiency of its propellers.

1.1 Motivation

Theoretically, there's a large amount of energy loss when air flows through a propeller. As the propeller rotates, air flow occurs from the central part of the propeller to its tip due to the centrifugal force caused during rotation. As the air reaches the tip it forms a vortex causing energy loss in the form of heat and noise, as shown in figure (1.2). This vortex occurs due to the collision of high pressure region at the bottom of the propeller and the low pressure region formed at the top, which is the main cause of lift on the propeller. Creating a barrier around the rotating propeller

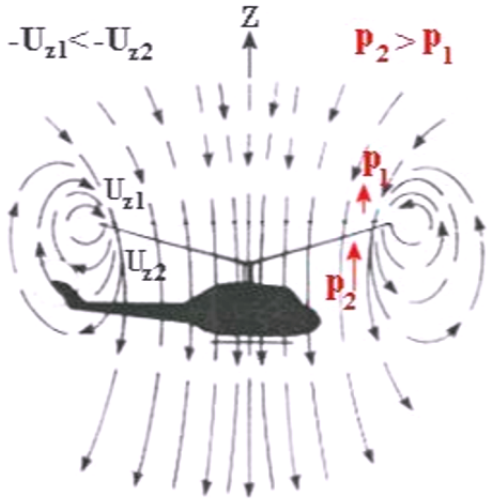

Figure 1.2: Air Flow Geometry for Propeller

can prevent the formation of this vortex and prevent this loss in energy.

Pereira (2008) investigates the potential of shrouded rotors in the design of UAV's. Momentum theory has been used to provide a first-order prediction of the performance and characteristics of shrouded rotors. Several ducts of varying diffuser length (L_d), angle (θ_d)and the inlet lip radius (r_{lip}) have been used on a propeller of size 6.3 inches in diameter (D_t). Refer to figure (1.3). The blade tip clearance (δ_{tip}) is kept as small as possible. The shrouded rotors showed an increase in thrust by up to 94% at the same power consumption or a decrease in power by up to 62% for the same amount of thrust produced. The optimal values for the diffuser angle and length were found to be 10° and 50% of the shroud throat diameter respectively. Increasing the lip radius to get a favorable pressure distribution over the inlet was also an important factor that was considered. Without the lip, the rotor can theoretically produce 26% more thrust that the open rotor of the same size, or it can consume 29% less power for the same amount of thrust produced. It is also stated that a larger diameter rotor is likely to give a better performance compared to a smaller one. The increase in

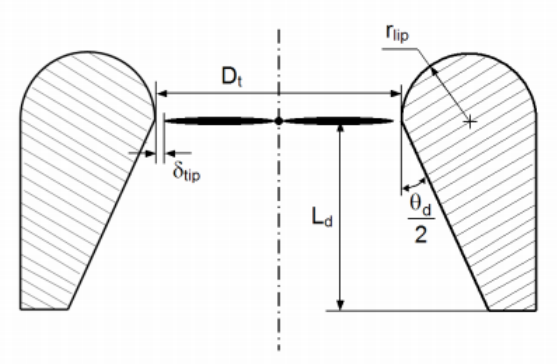

Figure 1.3: Duct Configuration

performance is primarily due to the ability of the diffuser to channel the air after it passes the rotor.

Besides its aerodynamic benefits, the duct can also attenuate noise generated by the propeller up to a degree which acts as advantage to drones in covert operations. The duct also acts as a safety feature to protect both the propeller from damage in case of a collision as well as people from getting injured. This makes it safer for drones to operate in crowded environments.

Over the past century, extensive research has been carried out on ducted rotor configurations. Hrishikeshavan *et al.* (2000) have tested this idea on Micro Air Vehicles. Results have shown improved performance in terms of increased thrust and power reduction. However, the weight and size of the duct leads to only a minimal increase in efficiency or in some cases it may even lead to a reduction in the performance of the quadcopter. This makes it impractical to use ducted rotors on conventional quadcopter models. Another disadvantage to having a duct, is the effect of drag induced on it during planar movements, as shown in figure (1.4). The size of the added duct on the quadcopter makes it difficult for the drone to maneuver, thus making it less agile in cluttered environments. During forward flight, when the quadcopter is pitched forward, the duct provides the required lift as the air flow over

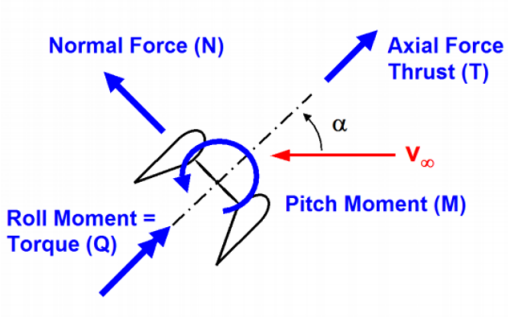

Figure 1.4: Moment Induced Due to Drag

the inlet is much better compared to when the quadcopter is hovering. This makes it easier for it to operate at higher angles of attack when compared to the open rotor configuration. But the duct also acts as a natural stabilizer for the drone during these roll and pitch movements. The pitching moment induced on the duct causes a reverse effect on the quadcopter thereby reducing its speed during forward motion.

It is therefore essential that the design of the quadcopter be optimized for a duct of fixed weight and size, to provide the maximum improvement in performance over the open rotor configuration of the quadcopter. For the ducted rotor to be a viable option, the increase in thrust due to the duct should be more than the weight of the duct itself.

This thesis presents a novel quadcopter model based on a ducted propeller system. It provides a way to test the efficiency of a ducted quadcopter while reducing the adverse effects of the duct on the system. The quadcopter, called DragonFly is designed to have a centrally ducted rotor which would provide about 60% thrust to the system while also having 3 peripheral rotors which provide the desired directional movement. The thesis will discuss the design of the main quadcopter frame based on the duct. It will discuss the challenges associated with the dynamics of such a quadcopter system along with the modelling of a closed-loop system. Simulations have been presented based on both the controller model presented and high-level controller

of the Flight Controller (FC) used by the quadcopter to determine its dynamic stability. Finally, flight tests are conducted on actual hardware to assess whether the ducted quadcopter system provides an increase in the efficiency as compared to its open rotor counterpart.

Chapter 2

QUADCOPTER DESIGN

DragonFly is designed to provide an improvement in flight efficiency in terms of flight time and stability in flight. The quadcopter has a centrally ducted rotor which needs to produce as much thrust as possible for the entire system. This requires a large ducted-rotor which leads to a large sized quadcopter. The 3 peripheral rotors are designed to provide only directional movement to the drone and are kept as small as possible. The size of the central rotor has, therefore been fixed based on its thrust requirements on the overall system. Keeping the thrust from the ducted rotor to be at 60% of the overall thrust generated by the system, first the duct and then the quadcopter frame have been designed in Solidworks.

2.1 Duct Design

Careful design of the duct is important to provide maximum possible performance improvement while minimizing the increase in size and weight due to the duct. The duct has been designed for a dual blade propeller of size 9 inches in diameter. The design of the duct is based on a venturi tube, which allows for a high velocity release of the fluid at the outlet. Based on the specifications from Pereira (2008), the nozzle and diffuser height are kept at 4.5 inches i.e. 50% of the throat diameter, the nozzle angle is kept at 10° while the diffuser angle is taken to be 30°. The radius of the inlet has been taken as 0.98 inches, which provides a favorable air distribution over the inlet. It is also important to keep the blade tip clearance between the duct and the propeller to be as minimal as possible to provide maximum performance. A small blade tip clearance will only lead to a formation of a small vortex leading to

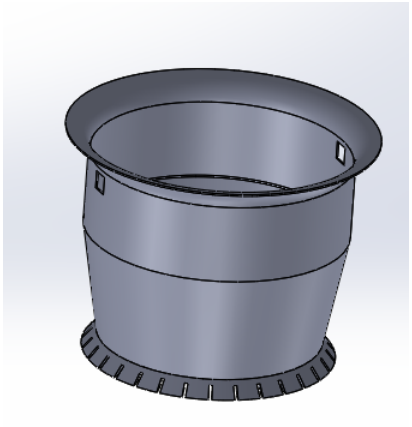

Figure 2.1: Duct Design

an increased efficiency of the rotor. Slits have been cut at the bottom of the duct to provide a passage for the air once it exits the duct. This is helpful as it prevents any vibrations during takeoff.

Due to its design, the duct has been 3D printed using ABS material to provide a smooth air flow around the duct. The thickness of the duct has been kept at 1.5 mm to keep the weight as minimum as possible. The duct is covered with a Styrofoam layer to increase it thickness. It prevents the duct from cracking due to any collisions or hard landings.

2.2 Quadcopter Frame

The frame of the quadcopter consists of 3 peripheral rotors connected using 2 shafts placed perpendicular to each other. The shafts are made of CFRP (Carbon Fiber reinforced plastic) which provides strength to the structure as the entire quadcopter frame and the duct rests on them. The wing shaft is connected at the middle to the front end of the tail shaft using a 3D printed joint. The rotors of the quadcopter are mounted onto protection rings which are then connected to the frame. The protection rings are there just to protect the rotors during collisions. The thickness of these rings is kept at 1.5 mm to keep the weight of the quadcopter as low as possible.

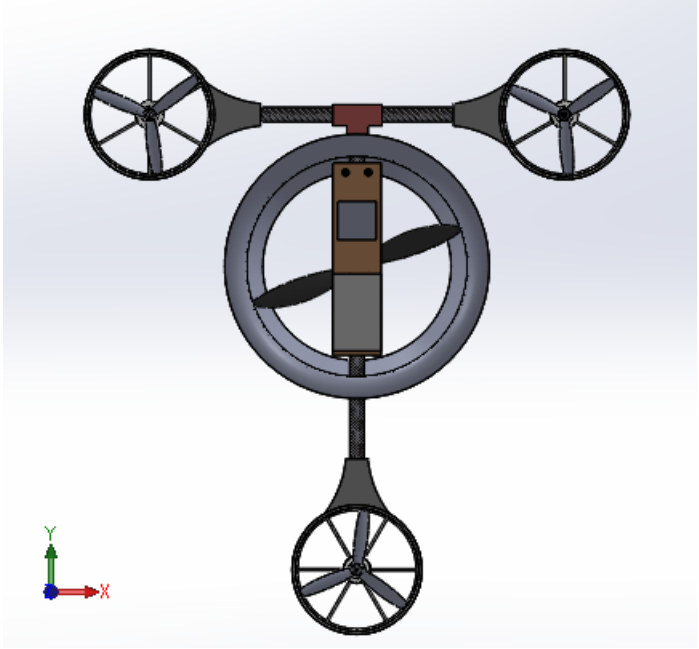

Figure 2.2: Quadcopter Frame Design

These are covered with a Styrofoam collar to increase their stiffness and prevent them from cracking during collision.

The central rotor is mounted on a rectangular plate at the bottom of the tail shaft. This bottom plate along with a top plate mount the Flight Controller, Companion Computer, battery and the Electronic Speed Controller (ESC). These plates are 3D printed as well and can be easily assembled on the entire structure. The battery is mounted on the top plate and as close to the central motor as possible to align the center of mass with the central rotor.

Referring to figure (2.2), the quadcopter is designed such that altitude gain (positive z-axis motion) occurs by the increase in thrust of all 4 rotors. Forward (positive y-axis) motion occurs by the increase in thrust of the rear rotor with respect to the front 2 rotors and vice versa. Motion in the positive x direction occurs by the increase in thrust of left rotor with respect to the right one. Yaw (torque about positive z-axis) in the counter-clockwise direction occurs by an increase in thrust of the central

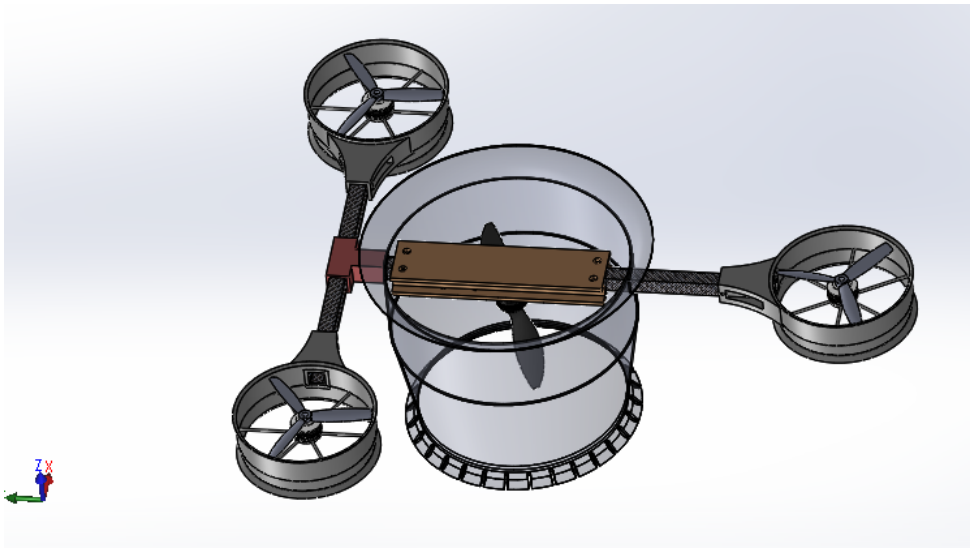

Figure 2.3: Quadcopter Perspective View

rotor relative to the peripheral rotors. This is further discussed during the dynamic modelling of the quadcopter in Chapter 3.

2.3 Hardware Components

The following components are used to develop the quadcopter prototype shown in figures (2.4) and (2.5).

- Carbon fiber reinforced plastic (CFRP) shafts having a 15 * 15 mm^2 cross-sectional area.
- There are 2 different motor sizes used for the central and the peripheral rotors.
  - KDE2315XF brushless motor used for the central rotor. This has a motor velocity constant (Kv) of 885 RPM/V and provides a maximum thrust of 9.51 N for a propeller of size 9 * 4.5 inches at 11.1 V.
  - KDE2304XF brushless motor used for the peripheral rotors. It has a Kv value of 2350 RPM/V and gives a maximum thrust of 3.92 N for a propeller of size 5 * 4 inches at 11.1 V.

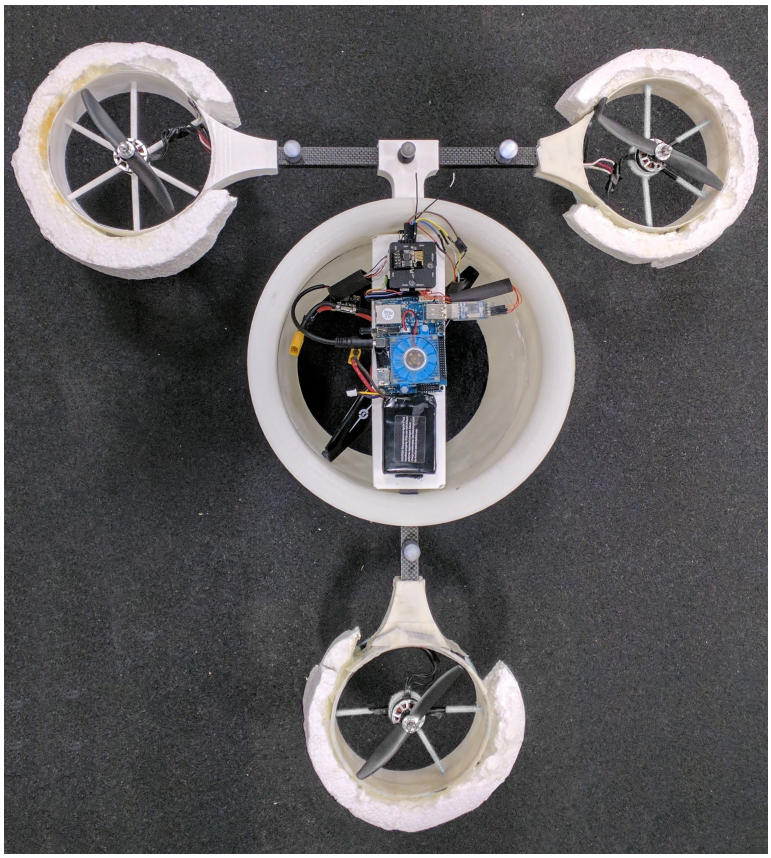

Figure 2.4: DragonFly Prototype Top View

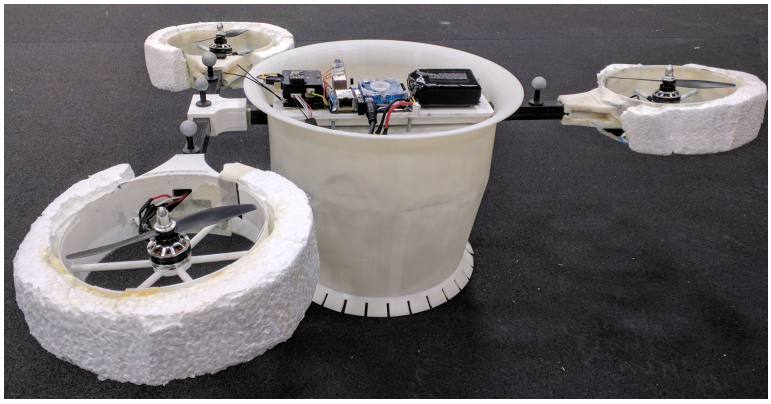

Figure 2.5: DragonFly Prototype Side View

- The propellers used are as follows:
  - 9 * 4.5 HQ carbon composite dual - blade clockwise propeller for the central rotor.
  - T5040 dual-blade counter-clockwise propeller made of ABS plastic used for peripheral rotors.
- Pixracer Flight Controller (FC) is used which sends PWM signals to the motors through the ESC. It has a total of 6 PWM outputs and is specifically designed for small quadcopters. It has two 6-axis accelerometers to provide the linear acceleration, gyroscopes to provide the angular velocity measurements and magnetometers to provide the heading. It also has a barometer for altitude measurements and operates at 5V DC supply using a Power module.
- A 4 in 1 Electronic Speed Controller (ESC) that splits power from the battery into the 4 motors. It operates at 11.1 V and can provide a continuous current output of 30 A.
- A powerdayAPM Power module is used for splitting power between the ESC and the Flight controller. It has a maximum operating voltage of 30V.
- An ODROID-XU4 companion computer used for autonomous control of the quadcopter. It connects to the Flight controller using a USB to TTL serial (UART) converter. It has an octa-core Heterogeneous Multi-Processing ARM CPU and 2GB RAM. It is compatible with various versions of Linux which can be used to setup the required ROS environment for communicating with the flight controller.
- 3 cell 11.1 V Lipo batteries having a capacity of 2000 mAh are used.

Chapter 3

SYSTEM DYNAMICS AND CONTROLLER DESIGN

3.1 Rigid Body Dynamics

Quadcopters have 6 degrees of freedom and exhibit a 3-axial motion in the x, y and z directions along with a yaw torque about the z axis. The 6 degrees of freedom are represented by the forces along the x, y, z axes together with the torques about these axes. These equations are together known as the system dynamics for a quadcopter.

For a typical quadcopter, the dynamics are quite well-defined and relatively simple compared to a quadcopter like DragonFly. Mellinger (2012), Hehn and D'Andrea (2011) and Larsson (2016) have worked on the control and simulation for a conventional quadcopter model. A free-body diagram for this quadcopter is shown in figure (3.1). Here, the rotors are of equal size and are placed equidistant from the center of mass of the drone which allows it to be symmetrical. To achieve dynamic equilibrium, the two opposite rotors T_1 and T_3 rotate in the counter-clockwise direction while rotors T_2 and T_4 rotate in the clockwise direction. The thrust produced by the rotors is directly linked to their rotational velocity. Increasing the rotational velocity of all 4 motors equally, causes a total system thrust of T_1+T_2+T_3+T_4, produced in the z_b direction causing an upward motion. Increasing the rotational velocity of rotors T_1 and T_3 relative to the other 2 rotors causes a yaw torque in the counter-clockwise direction. Increasing rotational velocity of rotor T_3 relative to T_1 leads to a pitch torque and motion in the y_b direction, while an increase in the rotational velocity of rotor T_4 relative to T_2 leads to a roll torque and motion in x_b direction. The axes x_b,y_b,z_b are with respect to the quadcopter's body and define the body-fixed frame of

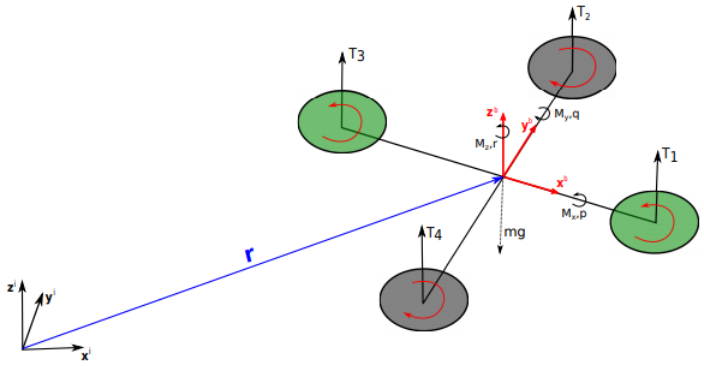

Figure 3.1: Free Body Diagram for a Conventional Quadcopter

reference. The other axes x_i, y_i, z_i are with respect to the ground plane and define the inertial frame of reference. It is easy to convert between the inertial frame and the body-fixed frame by using a rotation matrix which is constructed from the roll, pitch and yaw Euler angles.

$$\mathbf{R}_b^i = \begin{bmatrix} c_\psi c_\theta - s_\phi s_\psi s_\theta & -c_\phi s_\psi & c_\psi s_\theta + c_\theta s_\phi s_\psi \\ c_\theta s_\psi + c_\psi s_\phi s_\theta & c_\phi c_\psi & s_\psi s_\theta - c_\psi c_\theta s_\phi \\ -c_\phi s_\theta & s_\phi & c_\phi c_\theta \end{bmatrix} \tag{3.1}$$

In the above equation, roll is denoted by ϕ, pitch denoted by θ and yaw denoted by ψ. c_ϕ, c_θ and c_ψ are the cosine components of these angles, while s_ϕ, s_θ and s_ψ are the sine components.

We extend the same idea for generating the dynamic equations for the DragonFly. The complexity of the system arises by the fact that the rotors are of different sizes and the system is non-symmetrical. The central rotor which provides the majority of the thrust is bigger in size and would therefore have a much higher rotational moment compared to the smaller rotors. It is fixed based on the thrust requirements of the system. The moment produced by this central rotor needs to balance the moments produced by the combination of all the 3 peripheral rotors. This allows us to choose

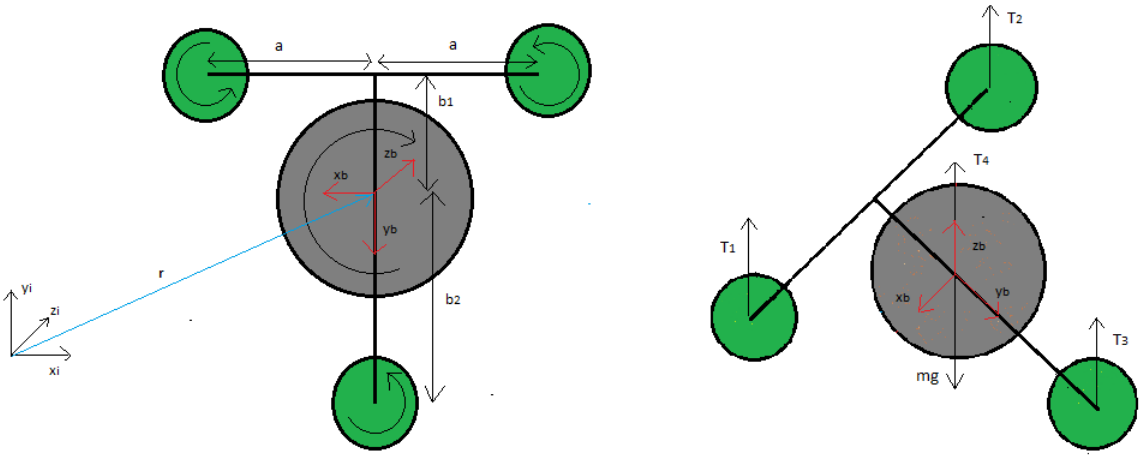

Figure 3.2: Free Body Diagram for DragonFly

the size of the smaller rotors.

The dynamic equations for the quadcopter are given by equations (3.2) and (3.3) below. Equation (3.2) defines the total system thrust while equation (3.3) defines the system torque, in the x, y and z axes.

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}^i = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}^i + \mathbf{R}_b^i \begin{bmatrix} 0 \\ 0 \\ T_1 + T_2 + T_3 + T_4 \end{bmatrix}^b \tag{3.2}$$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^b = \begin{bmatrix} (T_1 + T_2) * b_1 - T_3 * b_2 \\ (T_1 - T_2) * a \\ M_4 - M_1 - M_2 - M_3 \end{bmatrix}^b - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}^b \tag{3.3}$$

Here, $\mathbf{R}_b^i$ represents the rotation matrix for converting from the body-fixed frame to the inertial frame of reference as defined in equation (3.1) above. a, b_1, b_2 are the shaft lengths as shown in figure (3.2). M_1, M_2, M_3 and M_4 are the respective moments

for the rotors. While, $\begin{bmatrix} p \\ q \\ r \end{bmatrix}$ defines the angular velocity vector of the system in the body-fixed frame. We define the moment of inertia tensor (I), mass (m) and the gravitational constant (g).

$$M = \begin{bmatrix} (T_1 + T_2) * b_1 - T_3 * b_2 \\ (T_1 - T_2) * a \\ M_4 - M_1 - M_2 - M_3 \end{bmatrix} \tag{3.4}$$

Equation (3.4) represents the moments acting on the system in the body-fixed frame of reference. We must also consider the rate of change of angular momentum defined by equation (3.5),

$$\mathbf{H} = \mathbf{I}^b \omega^b \tag{3.5}$$

Here, H is the angular momentum vector and ω is the rotational rate of the quadcopter in the body-fixed frame. Considering this angular momentum component along with equation (3.4), we get the rotational dynamics of the system as defined in equation (3.3).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & -c_\phi s_\theta \\ 0 & 1 & s_\phi \\ s_\theta & 0 & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.6}$$

Equation (3.6) gives us the conversion between the angular velocity in the inertial frame of reference to its counterpart in the body-fixed frame. We can get the angular velocities in the inertial frames using equation (3.7).

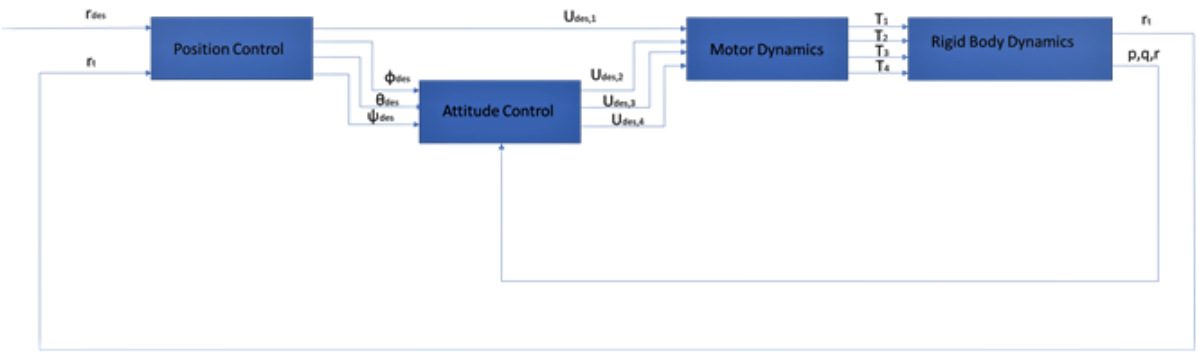

Figure 3.3: Controller Diagram

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta/c_\phi & 0 & c_\theta/c_\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.7}$$

3.2 Controller Design

Based on the dynamics of the quadcopter (Rigid body dynamics), the following controller design is proposed. It consists of an inner loop attitude controller for controlling the roll (ϕ), pitch (θ), yaw (ψ) angles and their angular rates, and an outer loop position controller for controlling the position in x, y, z axes along with their linear velocities.

An Inertial Measurement Unit (IMU) on the quadcopter is used to provide its linear acceleration, angular velocity and heading (ψ). These measurements are coupled with other sensors like the GPS, Vicon/Optitrack, Lidar or any other Vision sensors to get a position estimate of the drone (r_t) which is then fed back to the position controller.

$$r_t(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \psi(t) \end{bmatrix} \qquad r_{des}(t) = \begin{bmatrix} x_{des}(t) \\ y_{des}(t) \\ z_{des}(t) \\ \psi_{des}(t) \end{bmatrix} \tag{3.8}$$

3.2.1 Position Control

Based on the current position (r_t) and the desired position (r_{des}), the position controller first computes the desired linear velocities $\dot{x_{des}}$, $\dot{y_{des}}$ and $\dot{z_{des}}$.

$$\dot{x_{des}} = k_{p,x}(r_{des,1} - r_1) \tag{3.9}$$

$$\dot{y_{des}} = k_{p,y}(r_{des,2} - r_2) \tag{3.10}$$

$$\dot{z_{des}} = k_{p,z}(r_{des,3} - r_3) \tag{3.11}$$

Here, $k_{p,x}$, $k_{p,y}$, $k_{p,z}$ are the proportional gains of the system. $r_{des,1}$, $r_{des,2}$, $r_{des,3}$ are the desired x,y,z positions respectively while r_1, r_2, r_3 are the corresponding current positions.

These velocities are then used to compute the desired accelerations $\ddot{x_{des}}$, $\ddot{y_{des}}$ and the thrust $u_{des,1}$ in the z axis.

$$\ddot{x_{des}} = k_{p,\dot{x}}(\dot{x_{des}} - \dot{x}) + k_{d,\dot{x}}(\ddot{x_{des}} - \ddot{x}) + k_{i,\dot{x}}(\int(\dot{x_{des}} - \dot{x})) \tag{3.12}$$

$$\ddot{y_{des}} = k_{p,\dot{y}}(\dot{y_{des}} - \dot{y}) + k_{d,\dot{y}}(\ddot{y_{des}} - \ddot{y}) + k_{i,\dot{y}}(\int(\dot{y_{des}} - \dot{y})) \tag{3.13}$$

$$u_{des,1} = k_{p,\dot{z}}(\dot{z_{des}} - \dot{z}) + k_{d,\dot{z}}(\ddot{z_{des}} - \ddot{z}) + k_{i,\dot{z}}(\int(\dot{z_{des}} - \dot{z})) \tag{3.14}$$

A PID controller is used for controlling the velocity of the system. Linearizing the non-linear system of equations (3.2) about the nominal hover condition, we get the desired roll and pitch angles,

$$\phi_{des} = (\frac{1}{g})(\ddot{x_{des}}sin(\psi_{des}) - \ddot{y_{des}}cos(\psi_{des})) \tag{3.15}$$

$$\theta_{des} = (\frac{1}{g})(\ddot{x_{des}}cos(\psi_{des}) + \ddot{y_{des}}sin(\psi_{des})) \tag{3.16}$$

3.2.2 Attitude Control

The attitude controller uses a similar P-PID model to output the desired x, y, z system torques $u_{des,2}$, $u_{des,3}$, $u_{des,4}$ based on the desired RPY angles ϕ_{des}, θ_{des}, ψ_{des}.

$$\dot{\phi_{des}} = k_{p,\phi}(\phi_{des} - \phi) \tag{3.17}$$

$$\dot{\theta_{des}} = k_{p,\theta}(\theta_{des} - \theta) \tag{3.18}$$

$$\dot{\psi_{des}} = k_{p,\psi}(\psi_{des} - \psi) \tag{3.19}$$

Here, $\dot{\phi_{des}}$, $\dot{\theta_{des}}$, $\dot{\psi_{des}}$ are the desired angular rates for the system.

The angular velocities p,q,r obtained from the IMU are with respect to the body-fixed frame and are converted in the inertial frame using equation (3.7). These are then used to obtain the RPY angular estimates ϕ, θ, ψ.

A PID angular rate controller is then used to obtain the following desired system torques in the inertial-frame u_2, u_3, u_4.

$$u_2 = k_{p,\dot{\phi}}(\dot{\phi_{des}} - \dot{\phi}) + k_{d,\dot{\phi}}(\ddot{\phi_{des}} - \ddot{\phi}) + k_{i,\dot{\phi}}(\int(\dot{\phi_{des}} - \dot{\phi})) \tag{3.20}$$

$$u_3 = k_{p,\dot{\theta}}(\dot{\theta_{des}} - \dot{\theta}) + k_{d,\dot{\theta}}(\ddot{\theta_{des}} - \ddot{\theta}) + k_{i,\dot{\theta}}(\int(\dot{\theta_{des}} - \dot{\theta})) \tag{3.21}$$

$$u_4 = k_{p,\dot{\psi}}(\dot{\psi_{des}} - \dot{\psi}) + k_{d,\dot{\psi}}(\ddot{\psi_{des}} - \ddot{\psi}) + k_{i,\dot{\psi}}(\int(\dot{\psi_{des}} - \dot{\psi})) \tag{3.22}$$

u_2, u_3 and u_4 are converted into the body-fixed frame to get $u_{des,2}$, $u_{des,3}$ and $u_{des,4}$.

3.2.3 Motor Dynamics

Based on the thrust in the z-axis $u_{des,1}$ and the RPY torques $u_{des,2}$, $u_{des,3}$, $u_{des,4}$ we can compute the required motor thrusts T_1, T_2, T_3 and T_4.

$$u_{des} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ b & b & -2b & 0 \\ a & -a & 0 & 0 \\ -r_s & -r_s & -r_s & r_b \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{3.23}$$

Here,

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ b & b & -2b & 0 \\ a & -a & 0 & 0 \\ -r_s & -r_s & -r_s & r_b \end{bmatrix} \tag{3.24}$$

Therefore,

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = G^{-1} u_{des} \tag{3.25}$$

where, r_s and r_b are the radius of the smaller and bigger propellers respectively.

These motor thrusts are fed into the rigid body dynamics which forms the entire closed loop control for the quadcopter. In order to test this controller model, dynamic simulations are performed in Simulink. A tuning methodology is presented along with the simulation in the next chapter.

Chapter 4

QUADCOPTER SIMULATIONS

4.1 Simulink

4.1.1 Setting up the Simulation Environment

The quadcopter model designed in Solidworks has been imported into Simulink using the Simscape Multibody Link plugin. Modelling of rigid and flexible bodies in Simulink has been well documented by Sekkai (2014), Miller *et al.* (2017) and MathWorks (2019). This plugin converts the model into an XML format which can then be imported into Simulink using the *smimport* function in MATLAB. This creates a model of the rigid body in Simulink, as shown in figure (4.1) where each link of the rigid body is imported as a .STEP file from Solidworks. It also contains the required transformations between each link when imported from Solidworks.

The dynamic model is further modified to include revolute joints between the motors and propellers. The radius of the rotors has been given in the form of gains to the revolute joints and force/torque blocks are included to send thrust inputs to each rotor. A 6-DOF joint is also added between the inertial frame of reference and the body-fixed frame of the quadcopter which allows the quadcopter to exhibit a 6 degree of freedom motion during the simulation. Parameters such as mass of the model and its moment of inertia are imported into Simulink from Solidworks while the gravitational constant is defined with respect to the z-axis in the model.

As shown in figure (4.2), an IMU sensor connected to the base link (defined by the center of mass of the quadcopter) is used to get linear acceleration and angular velocity of the system. This data is used as feedback for the position and attitude

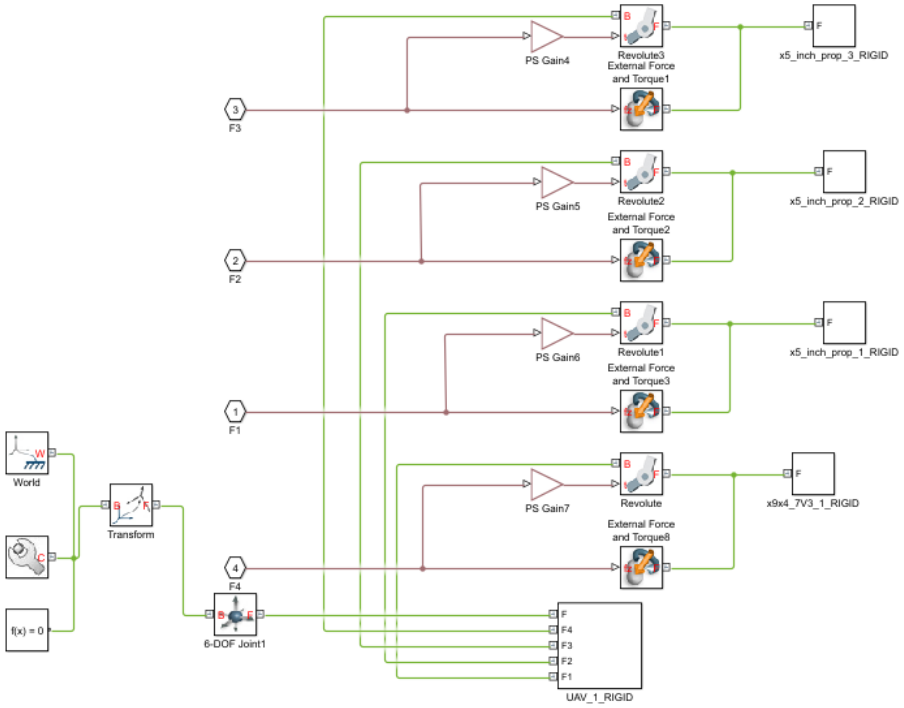

Figure 4.1: Rigid Body Dynamics

controllers.

The entire quadcopter control model is shown in figure (4.3). The UAV_Controller block contains the position and attitude controllers, and the motor dynamics matrix for getting the individual motor thrust data. Integrator blocks are used for converting angular velocities and linear accelerations into RPY angles and current positions respectively. The required transformations between the inertial frame and the body-fixed frame are also applied using the equations discussed in Chapter 3.

4.1.2 Tuning PID Parameters

Tuning the altitude gain/loss in the z axis is the easiest as it involves tuning the respective P - PID gains while no tuning of the lower level attitude controller is required. While, x and y motions involve tuning the attitude controller before the position controller. It is performed such that the settling time of the system is as low

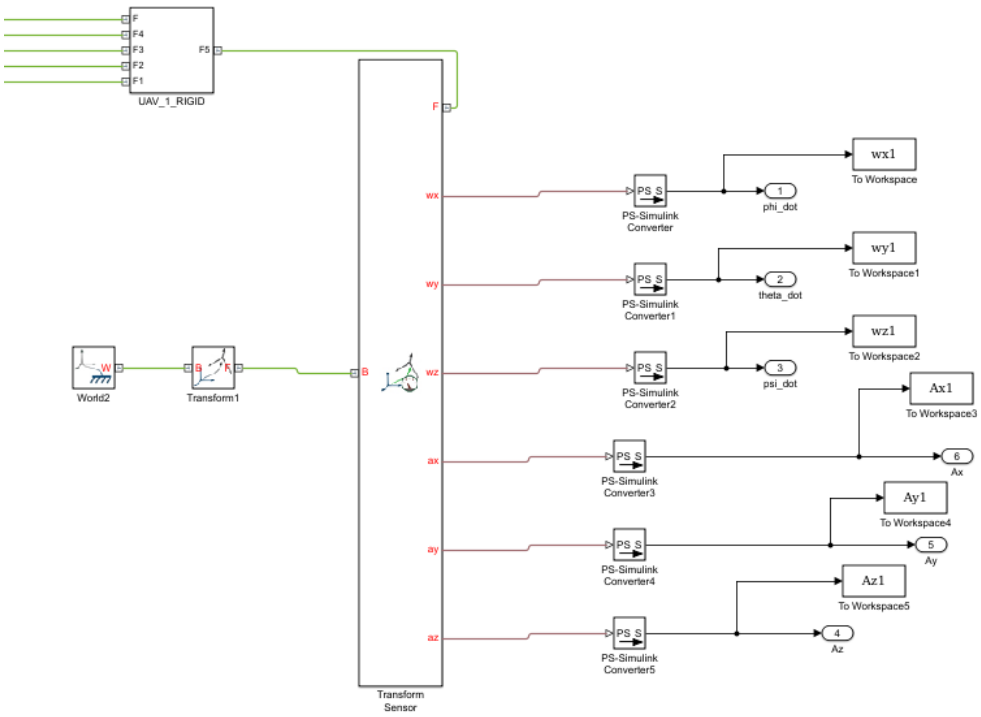

Figure 4.2: IMU Sensor Feedback

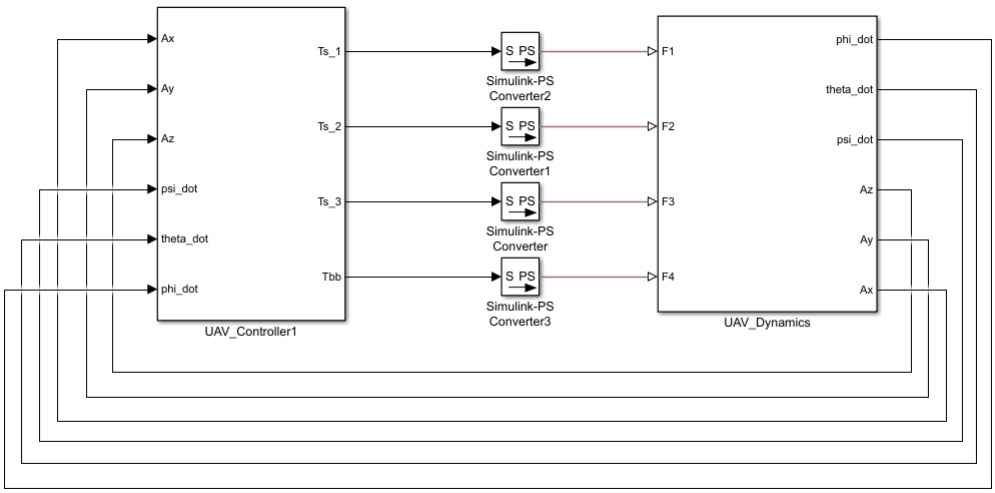

Figure 4.3: Controller Model

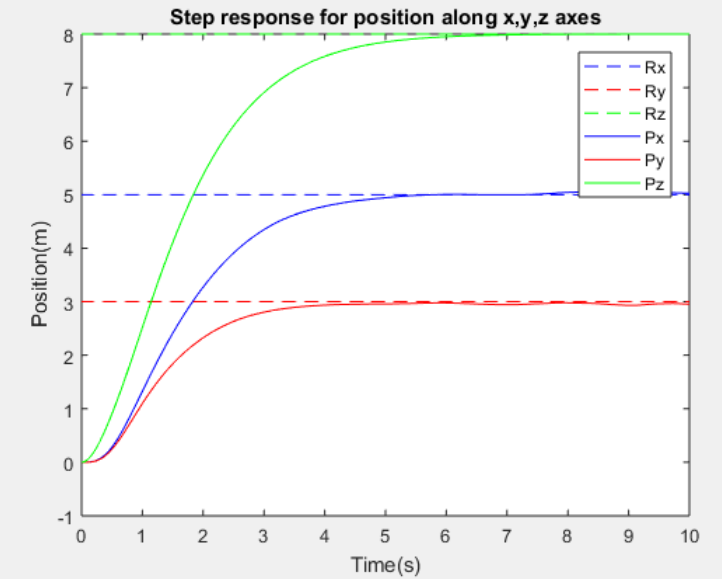

Figure 4.4: Position Response Curve for a Desired X,Y,Z Position

as possible while avoiding any overshoots.

The tuning is also performed during the hover condition of the quadcopter. This is done by feedforwarding the required motor thrusts produced by the quadcopter for hovering. The individual motor thrusts are calculated based on the weight of the quadcopter along with the thrust ratio between the bigger and the smaller motors.

The step-response curves in figures (4.4), (4.5), (4.6) are obtained taking the reference position r_{des} as $[5\ 3\ 8\ 0]^T$. Figure (4.4) shows a settling time of about 6 seconds for the given x,y,z step inputs. R_x, R_y, R_z denotes the desired x, y, z position setpoint while P_x, P_y, P_z denotes the current x, y, z position of the quadcopter. Figure (4.5) represents the angular change based on the step response obtained. Angular change, ϕ corresponds to the motion in the y-axis while θ corresponds to the motion in the x-axis. As the quadcopter accelerates, θ increases due to a counter-clockwise rotation about the y-axis while ϕ decreases due to a clockwise rotation about the x-axis. Figure (4.6) shows the motor thrust outputs generated based on the output of the controller. Thrust output of the central rotor denoted by T_4 decreases from

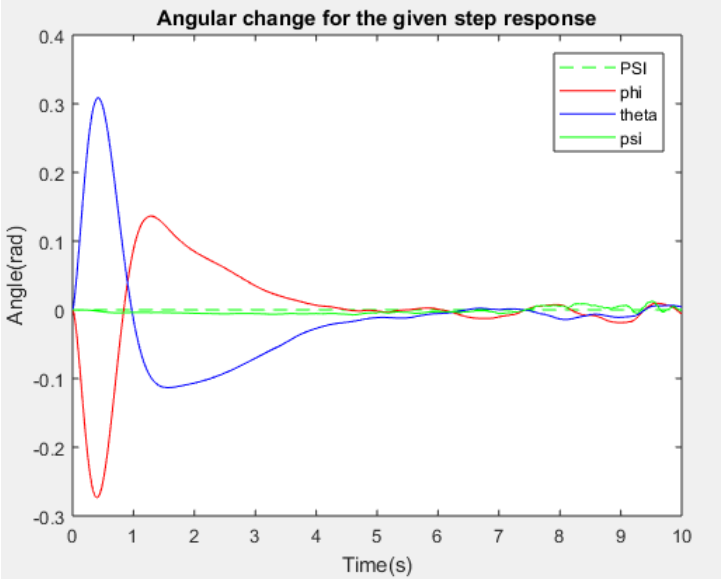

Figure 4.5: Angular Response Curve for the X,Y,Z Step Response

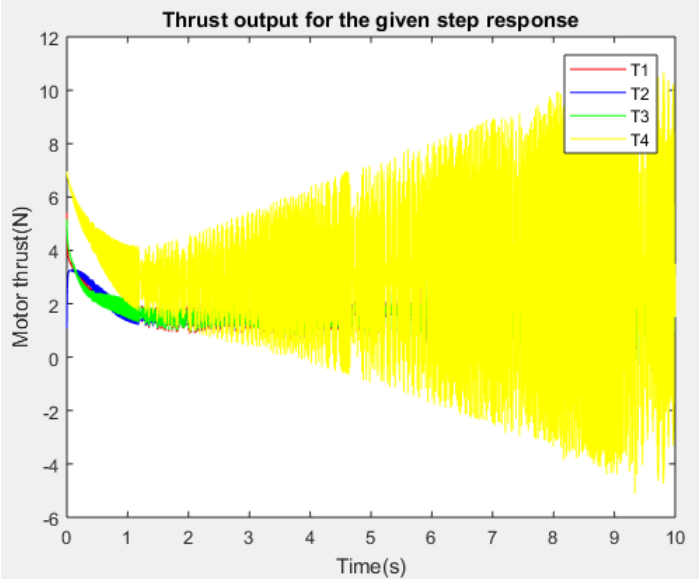

Figure 4.6: Motor Thrust Output for the X,Y,Z Step Response

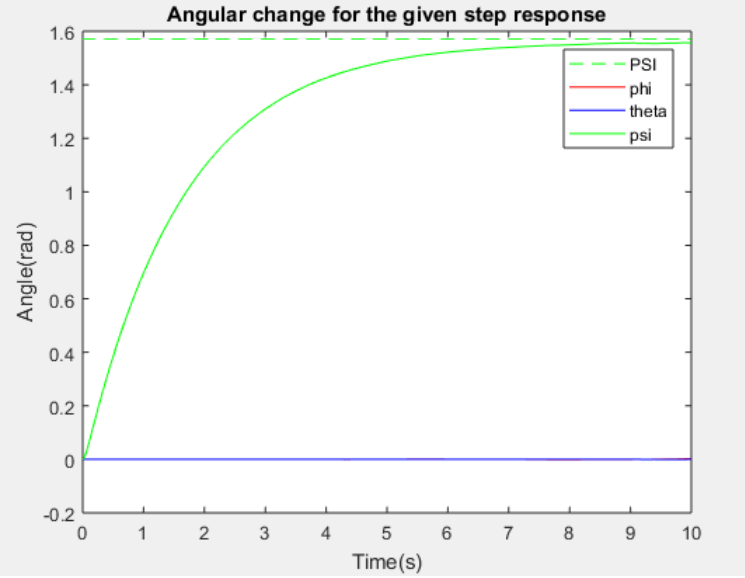

Figure 4.7: Angular Response Curve for a Desired Yaw

7N to about 3.5N during this motion, the thrust of rear rotor T_3 and left wing rotor T_1 decreases from 6N down to 2N, while the right wing rotor exhibits an increase in thrust from 1N to 2N. This shows that the individual rotor thrusts are well within range of the corresponding motor specifications. The motor thrust output can be further filtered which will reduce the oscillations produced during steady state.

The roll and pitch angles in the low-level attitude controller are tuned first such that the system exhibits a fast angular response while keeping within the desired angular range of 25° and avoiding any angular oscillations.

Similarly, the yaw angle (ψ) is tuned based on a desired ψ of 1.5708 radian. Figure (4.7) shows a settling time of 10 seconds based on the given step input. As shown in figure (4.9), the motor thrust outputs vary from 8N to 5N for rotor T_4, while T_1, T_2 and T_3 vary from 0.5N to 1.5N which is within range of the motor specifications. Figure (4.8) shows the x,y,z positions during the yaw motion.

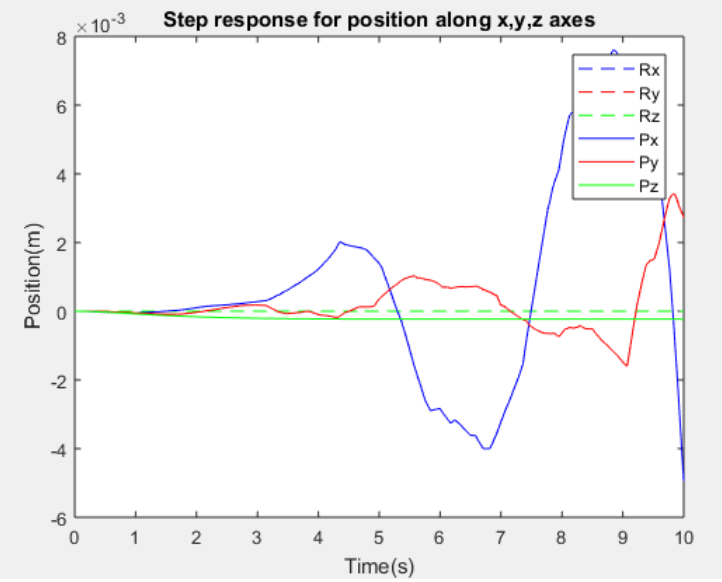

Figure 4.8: Positional Change Based on the Yaw Response

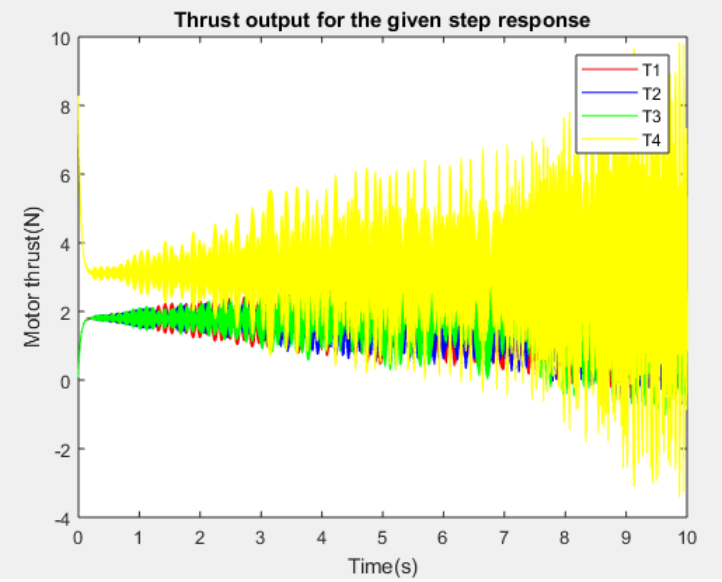

Figure 4.9: Motor Thrust Output for the Yaw Response

4.1.3 Limitations of the Simulink Environment

Quadcopter simulations in Simulink are useful for validating the system dynamics. The tuning of the position and attitude controllers is also quite simple. But this simulation environment has several limitations for simulating quadcopter motions. The first drawback is that the quadcopter becomes unstable when both yaw and x,y,z positions are commanded together which is not the case when implemented on actual hardware. This occurs as the controller uses a PID model for both position and attitude control while the Flight controller (FC) on the actual quadcopter uses a Model Predictive Controller (MPC) for position control. Also, this simulation is slow when it comes to testing a predefined trajectory, which makes it ill-suited for testing codes before implementing them on actual hardware. But, the major drawback is that these simulations are not accurate enough to show how the actual system with the Pixracer Flight Controller would behave. Pixracer uses Extended Kalman Filters (EKF) to estimate the position of the quadcopter using several sensor measurements. This data is sent to a high-level MPC controller which then computes the desired roll and pitch angles for the attitude controller.

Gazebo, which is better suited for robotic simulations is used to overcome these limitations. This can be shown in Meyer *et al.* (2012) who have used gazebo to comprehensively simulate quadcopters in a complex environment. This allows testing of the system in a simulation before implementing it in real environments, which limits hardware damage. The quadcopter model is also integrated with the Flight controller firmware to give more accurate results.

4.2 Gazebo Simulations

4.2.1 Setting up the Simulation Environment

Gazebo provides the flexibility to vary the simulation environment based on user preference which makes it ideal for robotic simulations. The user can modify the environment by adding predefined objects such as buildings and fields and can also change the surface of the simulation grid by adding surfaces such as gravel and sand. This allows the user to add in multiple obstacles in the path of the robot allowing it to sense and maneuver around these objects. The user can also create custom objects according to his needs. Qian *et al.* (2014) have simulated a virtual manipulator to grasp and place an object using ROS and gazebo. Yao *et al.* (2015) have created a soccer simulation in gazebo to perform multi-robot collaboration using robotic soccer players.

For performing robotic simulations in gazebo, we can either create a custom robot using links and joints or import a model from Solidworks into this environment. Importing the Solidworks model into gazebo requires a Solidworks to URDF converter which converts the Solidworks assembly file into an XML based URDF file. This URDF file defines individual parts used in the quadcopter assembly as links and defines joints between them. This is exported as a ROS (Robot Operating System) package in Solidworks which includes launch files for running gazebo simulations and visualize the transformations between the robotic joints. Due to the complexity of the quadcopter design, this method was chosen for importing the model into gazebo.

The quadcopter model generated is integrated with the PX4 firmware used by the Flight controller, which provides plugins for different sensors and motors and even provides a plugin for communicating with the drone using ROS. It also provides access to a GUI application called QGroundControl, which has a functionality for

Figure 4.10: Gazebo Simulation Environment

setting waypoints for the quadcopter and changing the flight parameters based on the quadcopter model used. The Extended Kalman Filter (EKF) estimator used in PX4, estimates the current position of the quadcopter based on measurements obtained from the IMU and either the global position (GPS) or the local position (Vicon/Optitrack). PX4 also uses a Model Predictive Controller (MPC) for the high-level position control while a P-PID controller is used for attitude control. The following steps are used to integrate the model with the PX4 firmware.

- Import the URDF model into gazebo and open the model editor.
- Add the required plugins to the quadcopter model.
  - IMU plugin defined on the base link.
  - Motor plugins which define the motor constant, moment constant, motor drag coefficient and other parameters.

  - Mavlink plugin for the quadcopter to communicate with ROS and QGroundControl.

- Save this model as an SDF file in the Models folder in the PX4 directory.
- Create a world file for the quadcopter which includes the model file and the customized increments to the gazebo environment, which include defining the orientation of the quadcopter when it is initialized in gazebo.
- Add a mixer geometry which defines the rotor axes of rotation, positions and the thrust and moment ratios between the bigger and the smaller rotors.
- Add an airframe file which defines the PWM motor output channels and the default parameters used by the drone.
- Add a mixer file which specifies the rotor limits for each of the thrust, roll, pitch and yaw commands.

After integrating the quadcopter model with PX4, the simulation is started, using the command *make px4_sitl_default gazebo_hawk* which opens a gazebo window with all the parameters for the quadcopter defined in the firmware. This allows the execution of basic commands like arming and disarming of the quadcopter, takeoff and land through the terminal window. The Mavlink plugin which acts as a bridge between ROS and gazebo is started by the *px4.launch* file in the MAVROS package.

Running this launch file in a separate terminal window establishes the communication link between the quadcopter with QGroundControl and ROS. This allows for either manually controlling the quadcopter using PlayStation/Xbox controllers or autonomously controlling it using predefined position setpoints. These setpoints can be defined via either QGroundControl or they can be sent by the user using C++/Python codes in ROS.

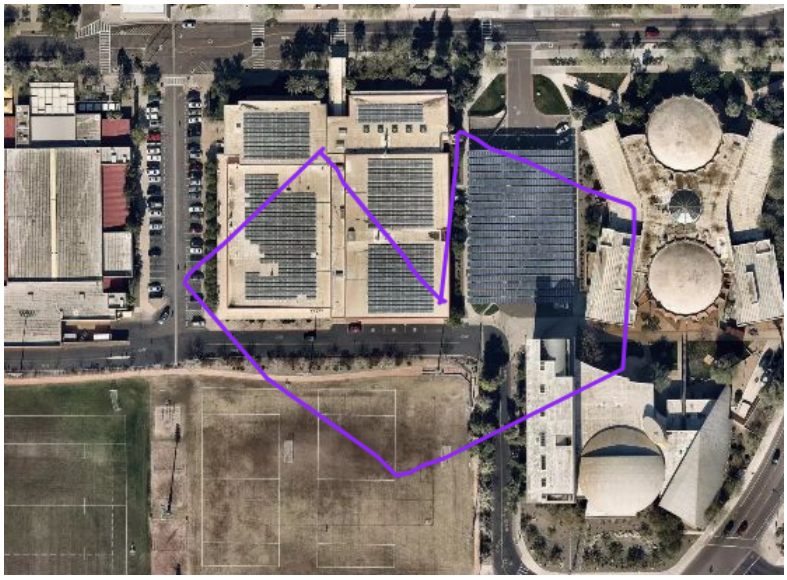

Figure 4.11: Desired Trajectory Using GPS Coordinates

4.2.2 Testing and Results

The quadcopter is initially tested by giving it GPS setpoints using QGroundControl which allows the drone to go into the Mission mode. Here, it executes the desired trajectory by going to each of the defined setpoints in turn.

Figure (4.11) shows the desired trajectory of the quadcopter designed in QGroundControl using GPS setpoints. Figure (4.12) represents the actual estimated trajectory of the quadcopter along with its desired setpoint trajectory. This simulation shows that the quadcopter follows the defined trajectory with only slight deviations from its actual path. These occur when the quadcopter changes direction after reaching a setpoint.

Based on this simulation, tests are conducted by giving position setpoints through ROS. This allows the quadcopter to go into the offboard mode which is done during the actual flight tests as well. Simulations are performed both with (4.14) and without the duct (4.13) to observe changes in the system response.

In these simulations, the quadcopter initially takes off from the ground to an

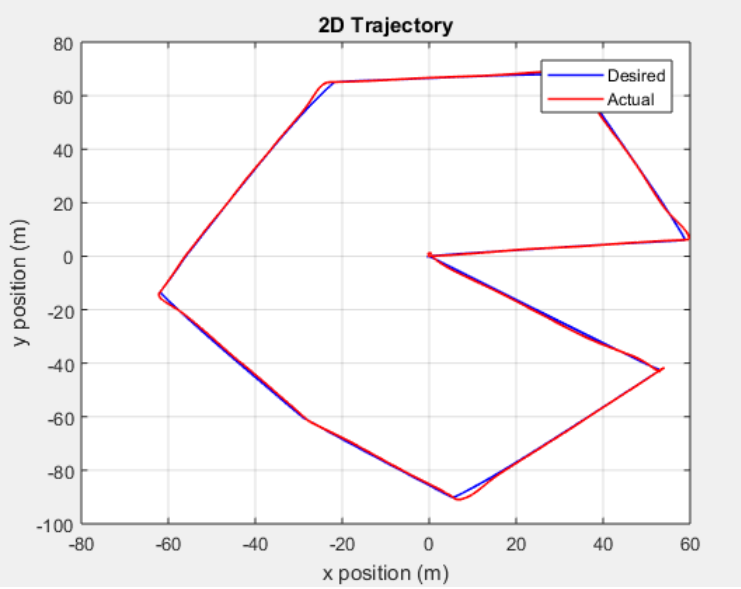

Figure 4.12: Desired Vs. Actual 2D Trajectory

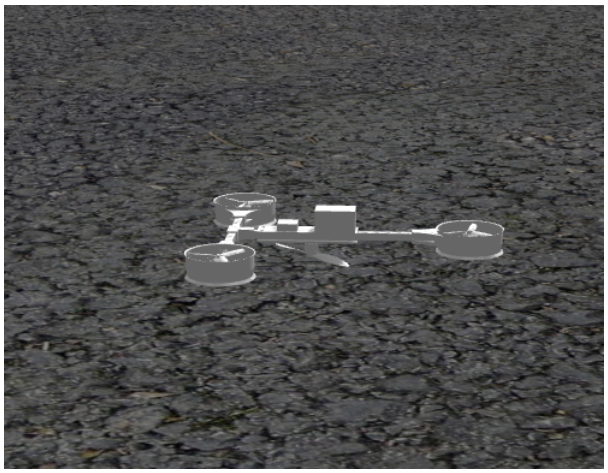

Figure 4.13: Ductless DragonFly Simulation

Figure 4.14: Ducted DragonFly Simulation

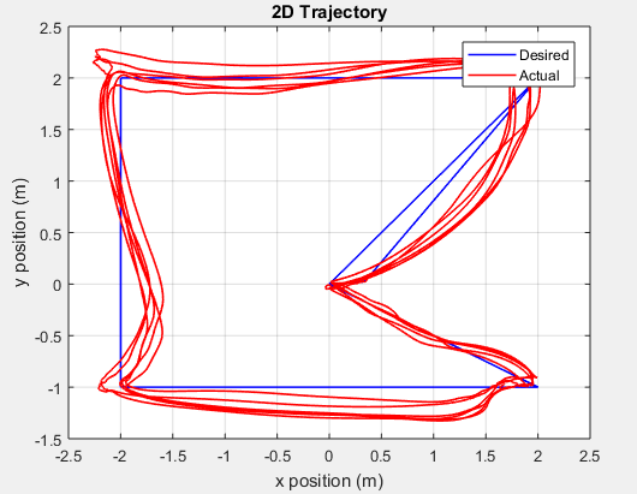

Figure 4.15: 2D Trajectory for Ductless System

altitude of 2m and then goes to the position setpoints defined in the trajectory. As shown in figures (4.15) and (4.19), the quadcopter goes to each of the following setpoints, $[0\ 0\ -3]$, $[2\ 2\ -3]$, $[-2\ 2\ -3]$, $[-2\ -1\ -3]$ and $[2\ -1\ -3]$. The whole simulation runs for 300 seconds such that the quadcopter executes the trajectory multiple times. The position setpoints are published at a frequency of 10 Hz that causes the simulation to run for a total of 3000 time samples, i.e. for 300s.

Figures (4.16), (4.17), (4.18) show the system response for the ductless quadcopter

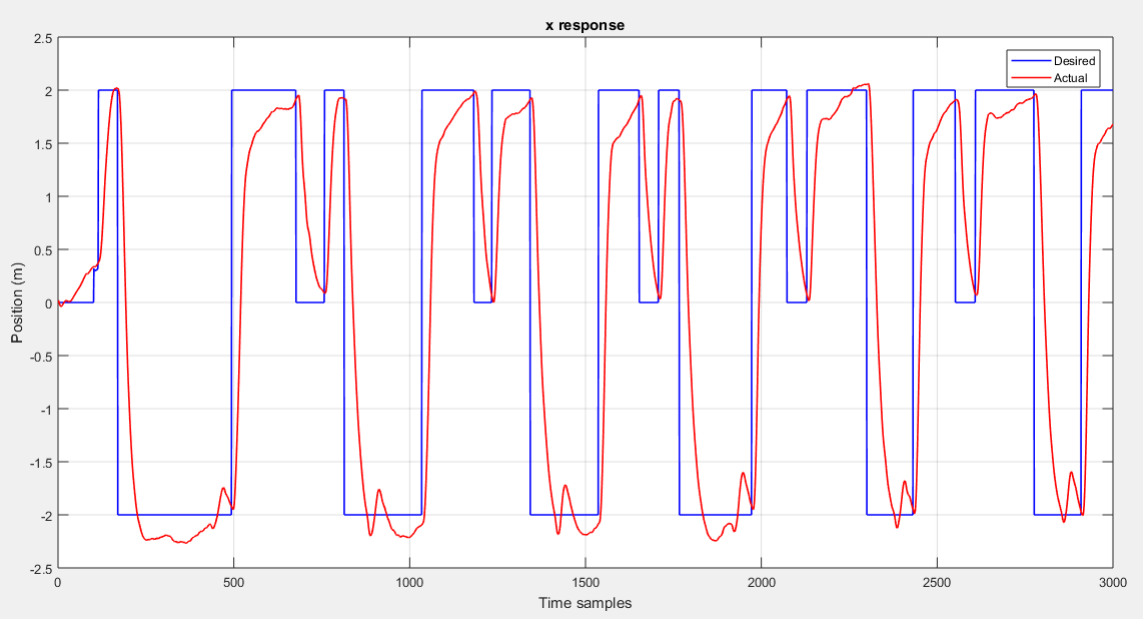

Figure 4.16: X Position Response for Ductless System

in the x,y,z axes respectively. Figure (4.15) shows the trajectory generated for this system in the x,y plane.

The system response in the x and y axes is consistent with only a little overshoot in position. The maximum settling time observed for the system is about 15 seconds with minor oscillations.

Figures (4.20), (4.21), (4.22) show the system response for the ducted quadcopter system in the x,y,z axes respectively. Figure (4.19) shows the trajectory generated for this system in the x,y plane.

The PID gains for the ducted and ductless quadcopter configurations are kept the same. This is done to observe changes in the system response during both cases.

For the ductless system, the maximum settling time for the x and y positions, as shown in figures (4.16) and (4.17) is about 15 seconds. While, for a ducted system, figures (4.20) and (4.21) the maximum settling time is about 12 seconds. This occurs as the quadcopter becomes more stable due to the additional weight of the duct about the center of mass. The overshoot in the system response observed for the ducted

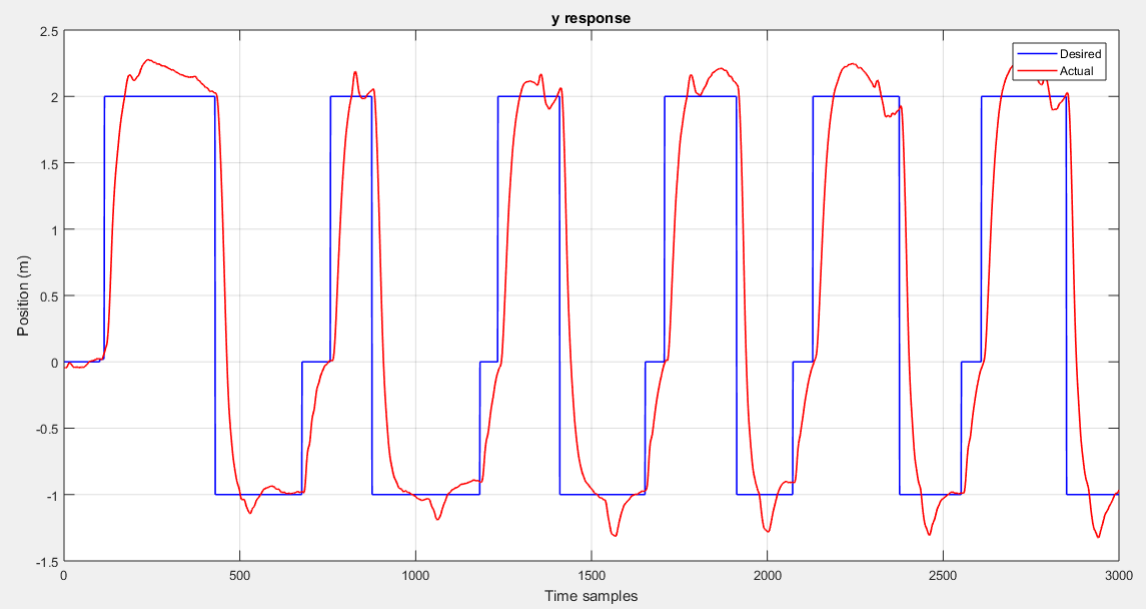

Figure 4.17: Y Position Response for Ductless System

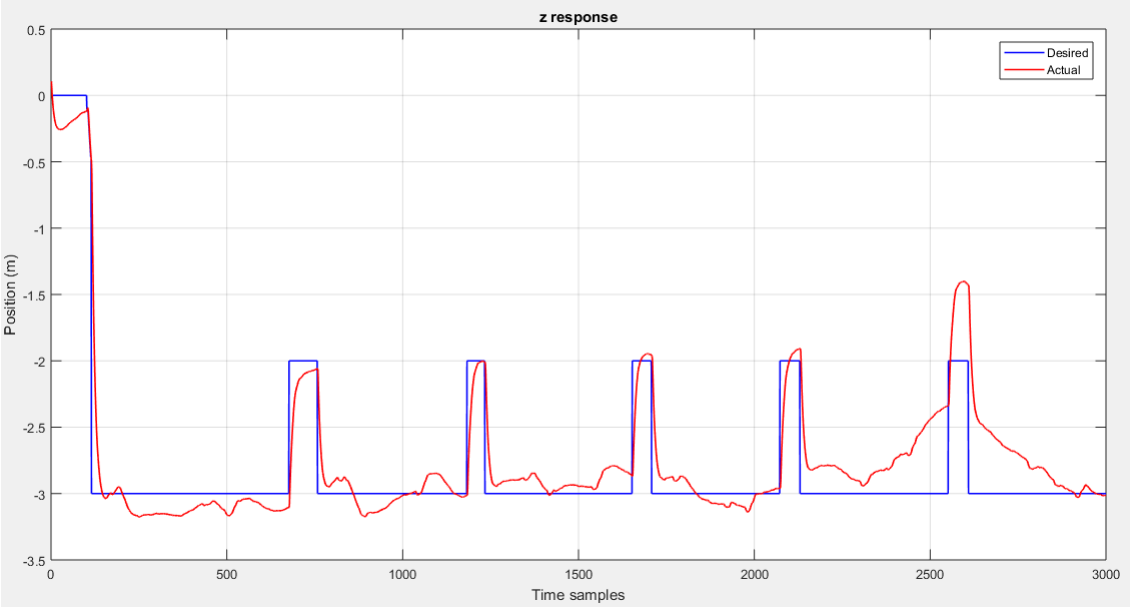

Figure 4.18: Z Position Response for Ductless System

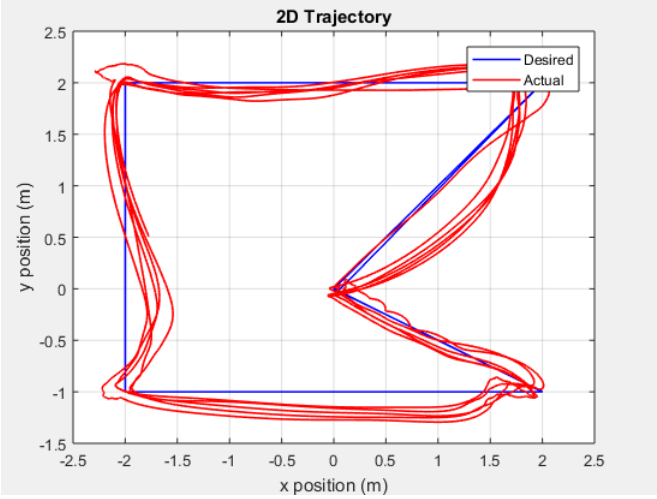

Figure 4.19: 2D Trajectory for Ducted System

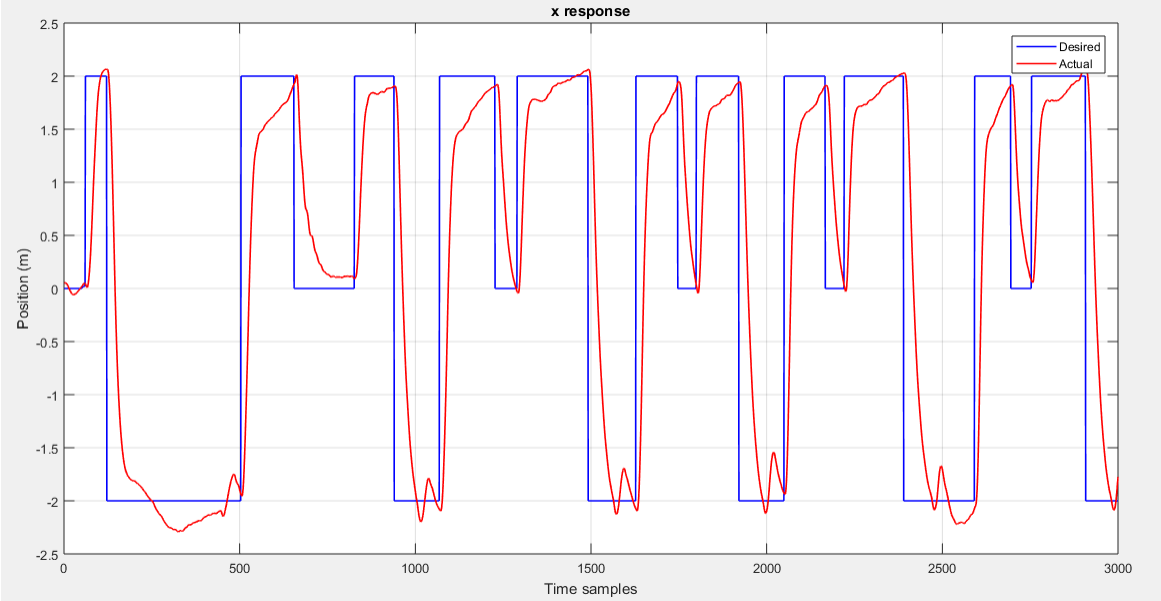

Figure 4.20: X Position Response for Ducted System

system is about 10% as compared to the 15% overshoot observed for the ductless system. This is apparent from the y position response curves.

These response curves show how the quadcopter will behave during the actual hardware tests. It also provides a way to validate the C++ coded trajectory to prevent any crashes during the hardware testing.

The data obtained from these simulations is used to calculate the efficiency of the

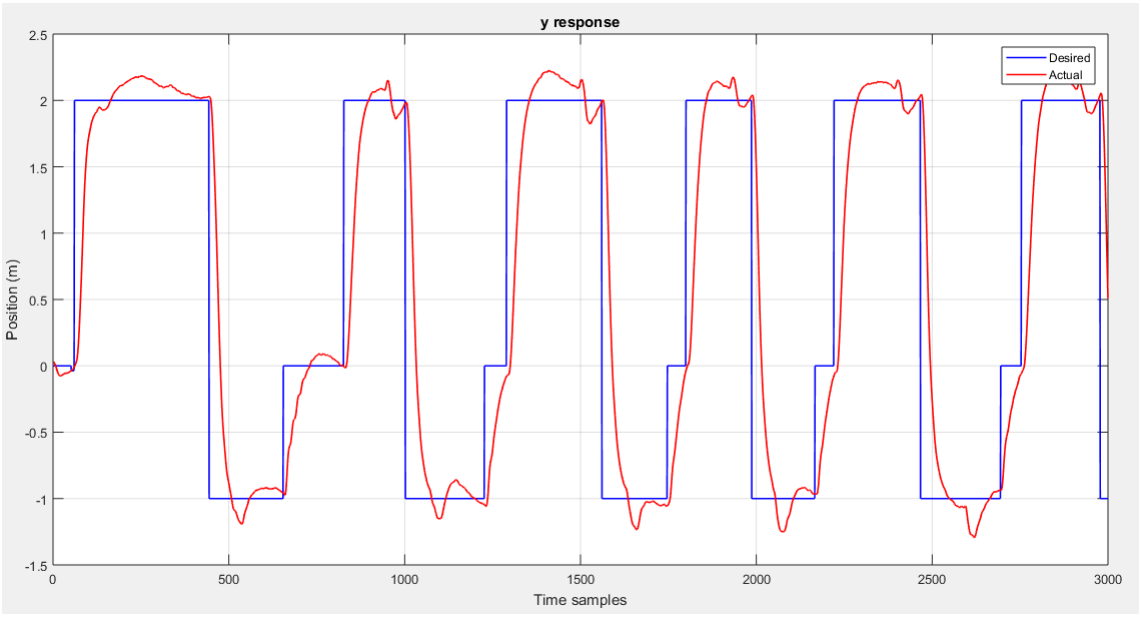

Figure 4.21: Y Position Response for Ducted System

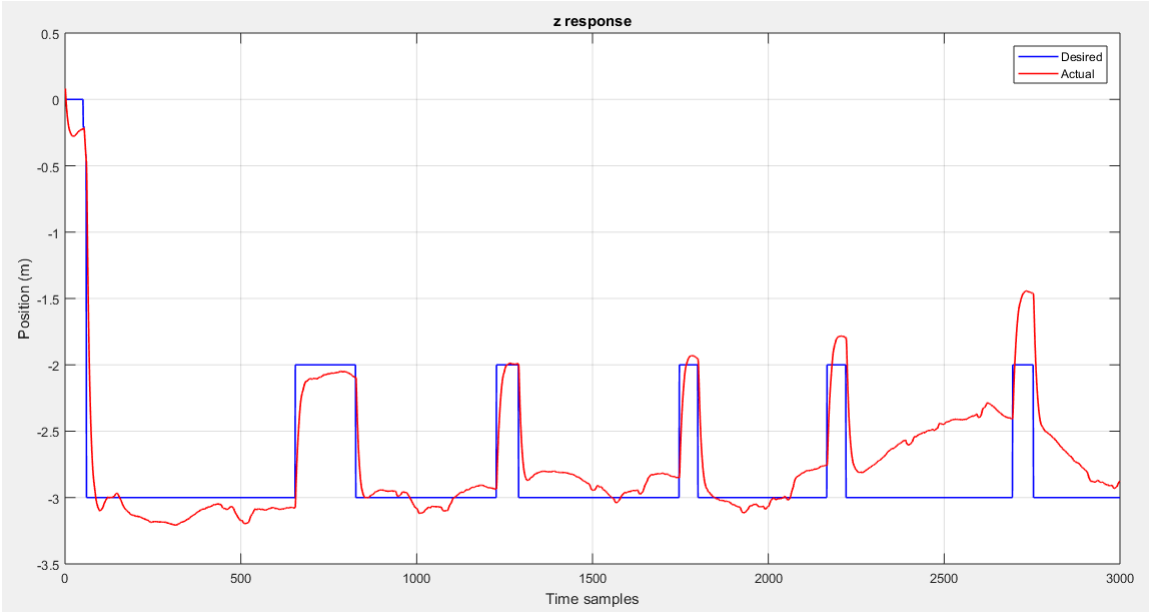

Figure 4.22: Z Position Response for Ducted System

ducted quadcopter system over the ductless system.

4.2.3 Flight Efficiency Test

The linear velocity data generated during the simulations is used to compute the flight efficiency of the system. Linear velocities in the x, y and z directions for both the ducted and ductless system are compared over 1500 time samples, where the quadcopter executes the trajectory thrice. This gives an accurate estimation of the efficiency of the quadcopter as it moves in all 3 axes during the simulation. Linear velocities also take into account the roll and pitch of the quadcopter during its motion, while yaw is kept constant throughout the simulation. This provides a quantitative measure of the maneuverability of the system as well.

The total kinetic energy for the ductless system is computed using equation (4.1),

$$E_{x,dl} = \frac{1}{2}m(v_{x,dl}^2 + v_{y,dl}^2 + v_{z,dl}^2) = 479.22J \tag{4.1}$$

Here, $v_{x,dl}$, $v_{y,dl}$, $v_{z,dl}$ are the linear velocities in the x,y,z directions respectively for the ductless quadcopter configuration.

Similarly, computing the total kinetic energy for the ducted system,

$$E_{x,d} = 419.28J \tag{4.2}$$

The comparison of the kinetic energy in those two conditions (ducted and ductless) results in a flight efficiency of 12.51% calculated using the following equation,

$$\eta = \frac{E_{x,dl} - E_{x,d}}{E_{x,d}} * 100\% \tag{4.3}$$

Chapter 5

EXPERIMENTAL RESULTS

5.1 Experimental Setup

The flight tests are performed in an indoor environment without the use of GPS for getting the position of the drone. Instead, the Optitrack motion capture system is used to get the positional data of the drone which is used by the internal estimators of the Flight controller onboard the quadcopter.

Figure (5.1) demonstrates how the position data from the Motion Capture (Mo-Cap) PC which runs the Motive software used by the Optitrack, is first sent to the ODROID companion computer onboard the quadcopter. The companion computer then sends this data to the Pixracer flight controller serially through a UART adapter.

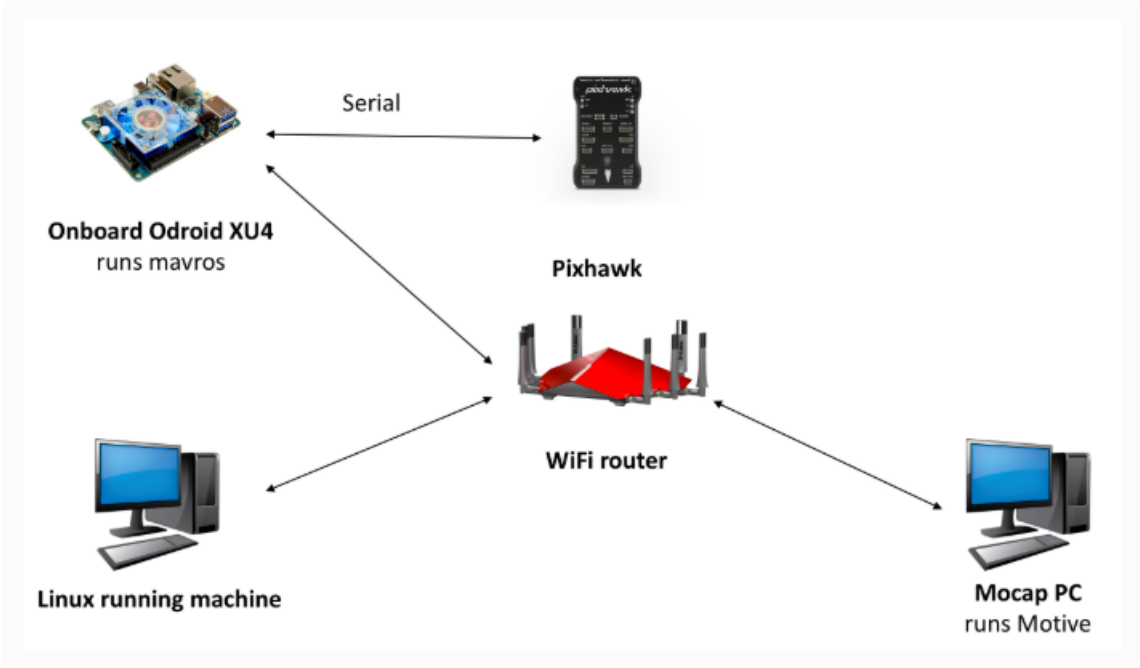

Figure 5.1: Hardware Setup

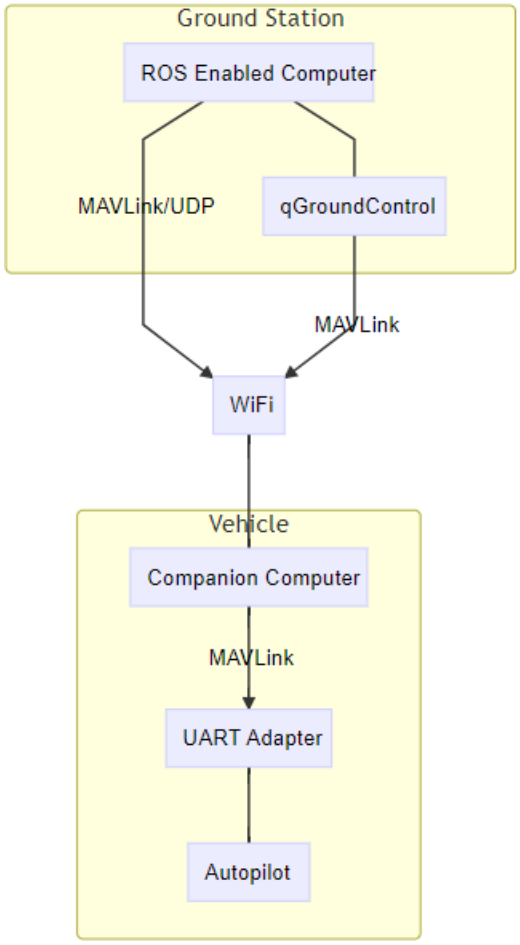

Figure 5.2: Communication Flow Chart

The ground station running the Linux OS interfaces with the companion computer through a WIFI connection to send it the desired position setpoints through ROS. All this communication takes place on a common network router placed in close proximity to the quadcopter, the ground station and the Mocap PC. The communication of the pose data is done using the MAVLink communication protocol. The communication diagram is shown in the figure (5.2).

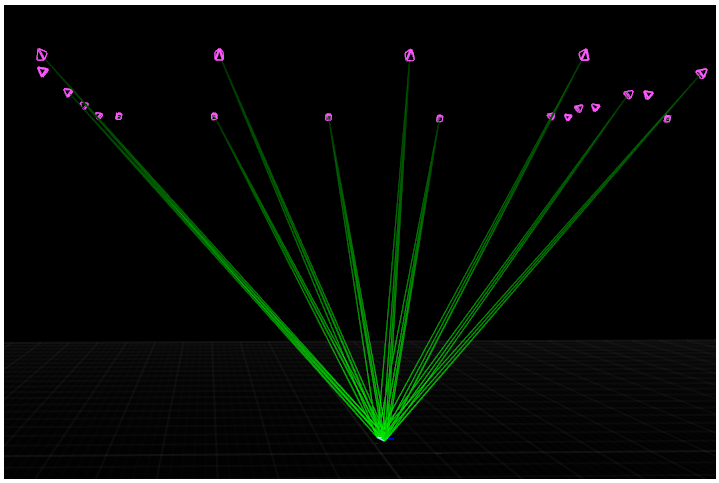

Figure 5.3: Rigid Body in Motive

5.1.1 Motion Capture System

As shown in figure (5.3), the optitrack motion capture system uses 17 tracking cameras to get the position and orientation (pose) of the quadcopter. It has a minimum accuracy of 0.5 mm and can transmit data at a frame rate of up to 360 Hz.

The pose data of the quadcopter is streamed over the common network using a predefined IP address. This data is then received by the Ground station PC by running the ROS package vrpn_client_ros. The IP address of the connected datastream needs to be configured in the launch file used by this package to receive the mocap data in the form of a ROS topic.

5.1.2 Companion Computer

The ODROID companion computer has a ROS base installed on its Linux OS. It also contains the MAVROS package which starts the MAVLink communication channel between the Ground Station and ODROID and between ODROID and Pixracer. Running the *px4.launch* file on MAVROS also gives a ROS topic list of position, velocity and thrust setpoints that can be commanded by the user. It also allows the

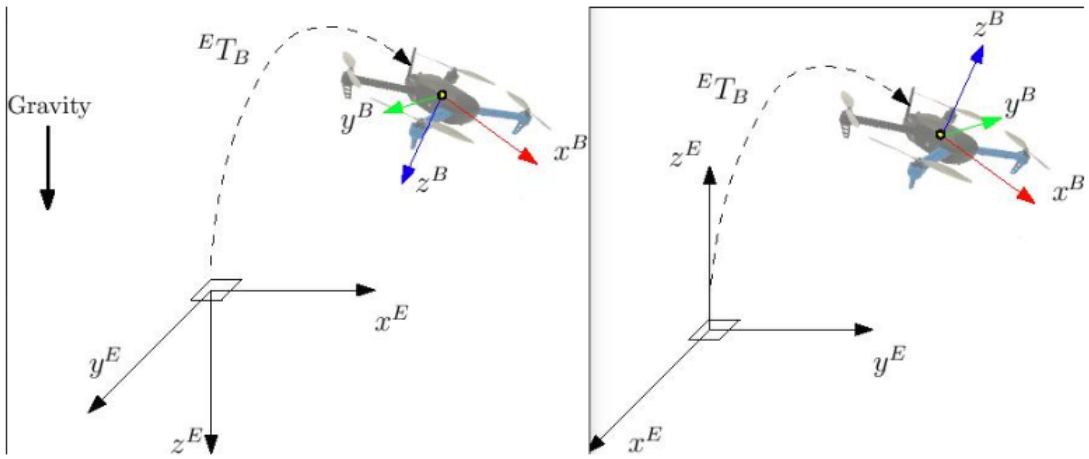

Figure 5.4: Frames of Reference

user to send in the current pose of the quadcopter obtained through Optitrack by mapping it on either the /mocap/pose topic or the /vision_pose/pose topic. This allows the estimator on the flight controller to read this pose data and output the position estimate of the quadcopter on the topic /local_position/local. This is then used as feedback by the position and attitude controllers of Pixracer. It is important to align the quadcopter with the positive x-axis while creating the object in Motive. This allows the correct mapping of the pose data obtained from Motive to the pose used by the estimator. MAVROS also transforms the pose data from the ENU (East, North, Up) reference frame used by ROS to NED (North, East, Down) which is the frame of reference used by the flight controller. Figure (5.4) shows the NED frame of reference on the left and ENU on the right.

5.1.3 Ground Station

The ground station PC is the ROS master and is used to command the quadcopter. It allows the transmission of pose data obtained from Motive to ODROID and also allows the user to send in the desired setpoints. It can also run QGroundControl while connected over the MAVLink interface which allows the user to track

Figure 5.5: Flight Test for the Ductless Quadcopter Configuration

the quadcopter during autonomous control and also change the vehicle parameters as required. QGroundControl can also be used to change from the offboard mode to the manual mode during emergencies as it allows the user to control the drone using the RC.

5.2 Flight Tests

Flight tests are performed for the ductless quadcopter configuration to determine its flight time based on a desired trajectory. In figure (5.5), the styrofoam ring around the central rotor is there to protect the propeller and does not provide any performance improvement to the system. A few initial setpoints are streamed which allows the quadcopter to go into Offboard mode and arm itself after which the trajectory setpoints are sent over. These setpoints need to be streamed at a rate greater than 2 Hz or else the quadcopter goes out of offboard mode into a failsafe mode defined by the user using QGroundControl.

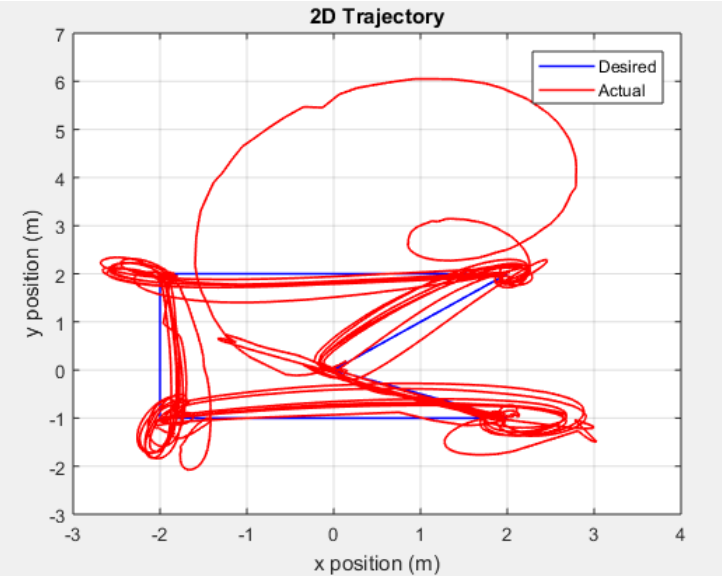

Figure 5.6: 2D Trajectory for Ductless Flight Test

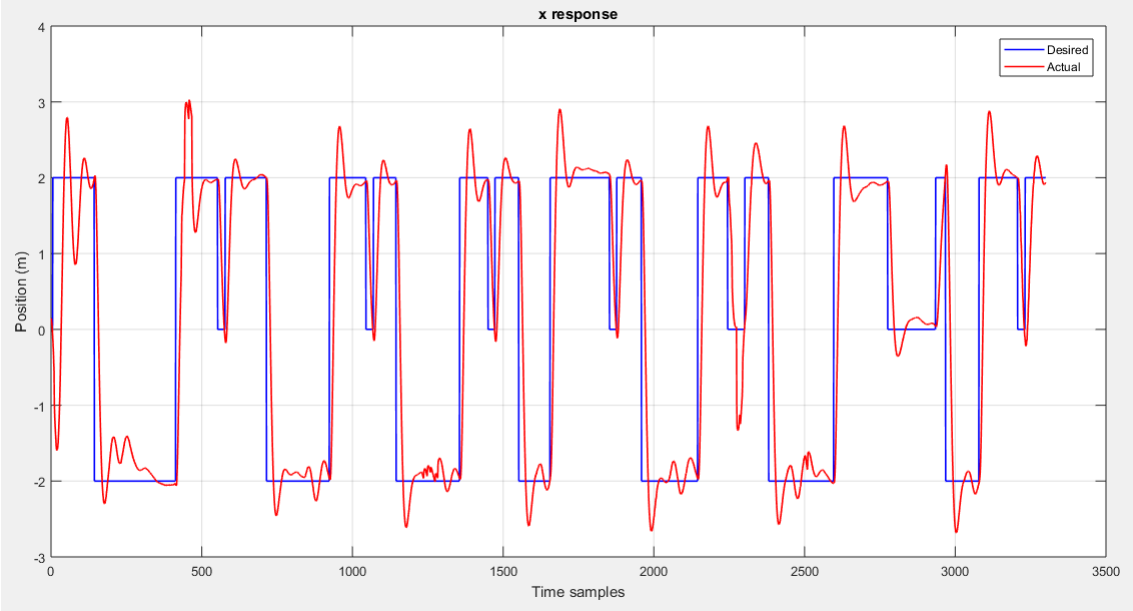

Figure 5.7: X Position Response for Ductless Flight Test

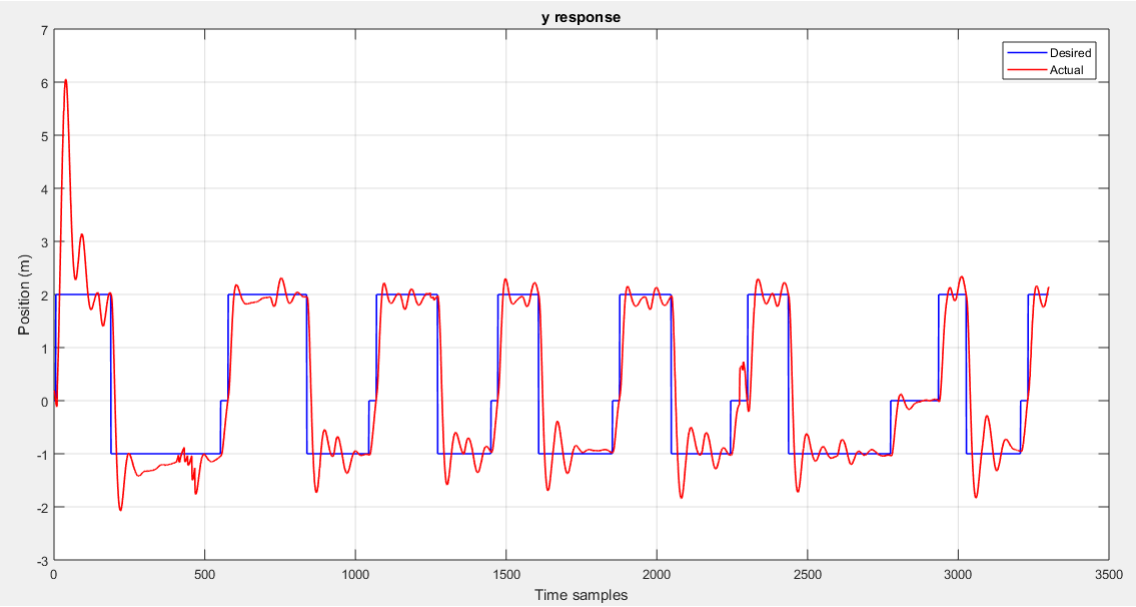

Figure 5.8: Y Position Response for Ductless Flight Test

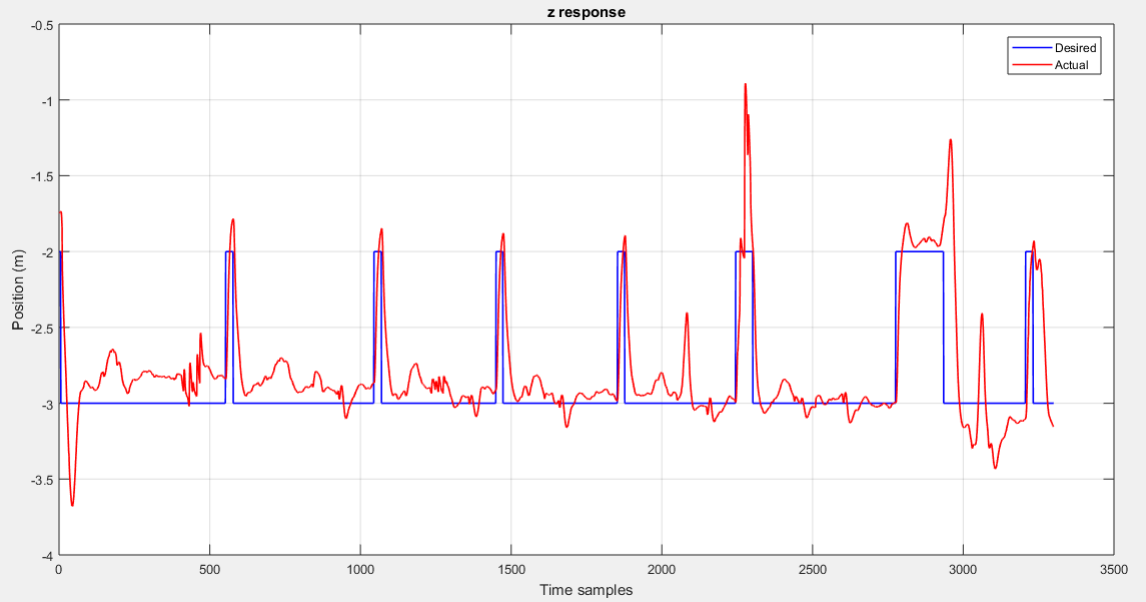

Figure 5.9: Z Position Response for Ductless Flight Test

The 2D trajectory generated for the ductless quadcopter configuration is shown in figure (5.6). The individual system response curves obtained for x, y and z positions are shown in figures (5.7), (5.8) and (5.9) respectively. The time samples are generated at a rate of 10 Hz.

As evident from the response curves, the quadcopter overshoots its position during takeoff as it moves from the origin to the first setpoint in the trajectory. This happens as the quadcopter exhibits both x,y,z and yaw motions together. After reaching the first setpoint, the quadcopter holds a constant heading of 90° throughout its flight.

This flight test was performed on a charged 11.1 V, 2000 mAh battery. The total flight time obtained is 6.033 minutes which also includes the time consumed by the quadcopter to land when the battery discharges.

The experimental flight tests are conducted to determine how close these results are compared to those obtained from the simulations. Based on the efficiency tested from simulations, the expected flight time for the ducted system should be about 6.8 minutes.

Chapter 6

CONCLUSIONS AND FUTURE WORK

The thesis presented an energy based method to improve the flight efficiency of quadcopters. A novel quadcopter prototype was developed with the goal of having increased flight efficiency based on a ducted rotor configuration. The dynamic complexity of such a system was discussed in relation with the conventional quadcopter systems and a control model was designed. This was then validated using Simulink and the PID parameters were tuned to obtain the desired system response. Simulations were also performed in Gazebo which were used to overcome the limitations of Simulink. These gave more accuarate results which were based on a specific desired trajectory. This trajectory was tested for the ducted and the ductless quadcopter configurations to determine the energy consumption during flight. An efficiency of 12.5% for the ducted quadcopter system over the ductless system was observed.

The desired trajectory tested on the simulation was also used during the flight tests conducted on the quadcopter prototype. A flight time of 6 minutes 2 seconds was observed for the ductless system on a battery of 2000 mAh. This can be further tested on the ducted system as well. By comparing the flight times obtained from both these tests, we can find the exact efficiency that the ducted quadcopter provides over the ductless system, thus proving the hypothesis.

The efficiency tests were conducted based on a specific setpoint trajectory. Further flight tests need to be conducted which also take into account the yaw of the system along with variable trajectories to get a range of flight efficiency data. Changes in trajectory of the quadcopter will decide the extent to which the flight time of the ducted system is affected with respect to the ductless system.

As mentioned in Pereira (2008), a larger sized ducted propeller provides a much higher thrust output. This should be the next step in the design of ducted quadcopters, where the central propeller provides upto 95% of the total system thrust. This will complicate the system dynamics further as the wing rotors should now be placed at an angle to compensate for the yaw produced due to the central rotor.

The idea of developing quadcopters around ducted rotors will open new avenues towards applications involving long range travel of drones especially in the military industry.

REFERENCES

Hehn, M. and R. D'Andrea, "Quadrocopter trajectory generation and control", (2011).

Hrishikeshavan, V., J. Black and I. Chopra, "American helicopter society international specialists meeting on unmanned rotorcraft", (2000).

Larsson, D., "Dynamics, modelling, simulation and control of mid-flight coupling", (2016).

MathWorks, "Simscape multibody documentation", URL `https://www.mathworks.com/help/physmod/sm/index.html` (2019).

Mellinger, D. W., *Trajectory Generation and Control for Quadrotors*, Ph.D. thesis, University of Pennsylvania (2012).

Meyer, J., A. Sendobry, S. Kohlbrecher, U. Klingauf and O. von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo", in "SIMPAR", (2012).

Miller, S., T. Soares, Y. V. Weddingen and J. Wendlandt, "Modeling flexible bodies with simscape multibody software", (2017).

Pereira, J. L., *Hover and Wind-Tunnel Testing of Shrouded Rotors for Improved Micro Air Vehicle Design*, Ph.D. thesis, University of Maryland, College Park (2008).

Qian, W., Z. Xia, J. Xiong, Y. Gan, Y. Guo and S. W. et al., "Manipulation task simulation using ros and gazebo", in "IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)", (2014).

Sekkai, S., "Lecture on modelling and simulation for a quadcopter", (2014).

Yao, W., W. Dai, J. Xiao, H. Lu and Z. Zheng, "A simulation system based on ros and gazebo for robocup middle size league", in "IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)", (2015).