2019

System Identification and Model-Based Control of Quadcopter UAVs

Andrew P. Szabo
Wright State University

System Identification and Model-Based Control of Quadcopter UAVs

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

by

Andrew P. Szabo
B.S.E.E., Wright State University, 2012

2019
Wright State University

Wright State University
Graduate School

May 18, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Andrew P. Szabo ENTITLED System Identification and Model-Based Control of Quadcopter UAVs BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Masters of Science in Engineering.

Xiadong Zhang, Ph.D.
Thesis Advisor

Fred D. Garber, Ph.D.
Interim Chair, Department of Electrical Engineering

Committee on
Final Examination

Xiadong Zhang, Ph.D

Jonathan Muse, Ph.D

Kuldip S. Rattan, Ph.D

Barry Milligan, Ph.D.
Interim Dean, Graduate School

Abstract

Szabo, Andrew P. M.S.Egr. Department of Electrical Engineering, Wright State University, 2019. System Identification and Model-Based Control of Quadcopter UAVs.

As control systems become more sophisticated, more accurate system models are needed for control law design and simulation. In this research, a nonlinear dynamic model of a quadcopter UAV is presented and model parameters are estimated off-line using in-flight experimental data. In addition, a model-based classical control law for the quadcopter UAV is designed, simulated, and then deployed in UAV flight tests. The intent of this research is to identify a model which may be simple enough to easily use for control law design, and accurate enough for simulation. In addition, a model-based classical control law is designed to for flight control.

The parameters of the nonlinear dynamic model are estimated with the Linear Least Squares Error method. In-flight disturbances are introduced in flight tests to ensure frequency rich data. The performances of different models are compared using validation flight test data to select an accurate model. This model is used as the simulation model and the design model. Model-based control law design techniques are used to create a flight control law which provides good performance both in the simulator, as well as when deployed to the quadcopter.

To perform these tests, the Real-Time - Marseille Grenoble Project software is used for the creation of ground station programs and flight control algorithms in Simulink. This test environment integrates a VICON camera systems, QuaRC Real Time system, a 3DR APM 2.6 micro-controller unit, and a Gumstix Overo AirSTORM micro-controller unit to create a low-cost quadcopter research platform.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to thank my advisers Dr. Xiaodong Zhang and Dr. Jonathan Muse for the guidance they have provided me, and lessons they taught me during my time at Wright State University. I would also like to thank Dr. Marian K. Kazimierczuk for his instruction. Without these men's help I would not be the engineer I am today.

Through out this research I was assisted by Dr. Remus Arvam, Mohsen Khalili, and Neel Dalwadi. I would like to thank them for this assistance.

Dedicated to

My Mother and Father, Cathryn and Francis Szabo, for the immeasurable amount of guidance they have given me in my education and life.

Chapter 1: Introduction

Wright State University unmanned aerial vehicles (UAV) research group seeks to use UAV platforms for research of model-based control law design, fault diagnosis and compensation, artificial intelligence, and swarm algorithms as well as other areas of research. Quadcopter vehicles were selected as the UAV platform due to their low cost, ease of customization, small size, and their ability to fly at low speed. The latter two reasons allow the quadcopter to be flown indoors. Flying indoors allows the removal of weather conditions as well as the ability to use a camera system to accurately measure the position and orientation of the quadcopter.

Most of the above areas of research require a dynamic model of the quadcopter for algorithm development. In addition, the ability to create an accurate simulation model of the vehicle provides more insight into an algorithm's performance before flight tests. This increases confidence in performance and reduces the risk of crashing and damaging the vehicle.

This research presents a method of creating a non-linear model for quadcopter vehicles, which is suitable for control law design and simulation studies. A quadcopter platform is developed for flight tests. Experimental flight data is used to develop a non-linear dynamic model using the Linear Least Square Error method (LLSE) of parameter estimation. The quadcopter model obtained is used to conduct a model-based classical control law design.

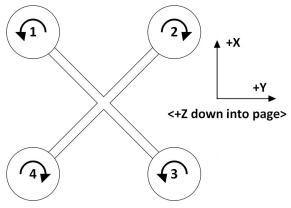

Figure 1.1: 'x' quadcopter layout

1.1 Overview of Quadcopter, Motion, & Control

A quadcopter, also known as a quadrotor helicopter, is a rotorcraft platform defined by being operated by four separate propeller motors. Quadcopter can be implemented as low cost flight platforms. In addition, quadcopters are suitable to flight in confined spaces due to their ability to hover and operate effectively at low velocities. Different variations of quadcopters exist such as those with tilting rotors or non-symmetrically sized motors. However, only fixed and symmetric motor quadcopters are used in this research due to their simpler dynamics and because the vehicle may be controlled by only controlling the motor velocities. A quadrotor vehicle moves about space by changing its roll and pitch angles to vector the thrust in the desired direction.

Quadcopter's four motors control the vehicle's attitude and thrust. Two of the motors spin clockwise, and the other two spin counter clockwise. The contra-rotating motors allow the yaw of the quadcopters to be controlled, since each rotor produces a yawing torque. Quadcopters are usually configured in a '+' or 'x' configuration. In the '+' configuration, the roll and pitch are each controlled by two motors. In the 'x' configuration roll and pitch are controlled via the velocities of all four motors [2]. For this research the quadcopter is used in the 'x' configuration. The decision was determined by the geometry of the vehicle such that the axis of the body frame is aligned with the approximate axis of symmetry of the vehicle. As shown in Figure 1.1 (seen from the top), motors 1 and 3 rotate counter-clockwise, and motors 2 and 4 rotate clockwise. The coordinate system of the quadcopter vehicle's body is defined as the X-axis pointing to the front of the quadcopter, the Y-axis pointing to the right, and the Z-axis pointing to the bottom.

The thrust and torque generated by each motor is approximately proportional to the square of the rotational velocity of the respective motor. At hover all motors are generating torques that cancel out the others. To generate a positive change in roll (rotation about X-axis), the velocity of motors 1 and 4 are increased, and the velocity of motors 2 and 3 are decreased. To generate a positive change in pitch (rotation about Y-axis), the velocity of motors 1 and 2 are increased, and the velocity of motors 3 and 4 are decreased. To generate a positive change in yaw (rotation about Z-axis), the velocity of motors 1 and 3 are increased, and the velocity of motors 2 and 4 are decreased [2].

Autonomous flights will be controlled by classical controllers with successive loop closures. An inner loop controls the attitude and altitude, and an outer loop controls the X and Y position in the flight space. The inner loop is always active since it is required for stabilizing the vehicle. The outer loop for controlling X and Y position determines the desired roll and pitch attitude command. In the inner loop, the altitude loop determines the desired thrust (T_z), and the attitude loop determines the desired torques (τ_ϕ for the roll torque,

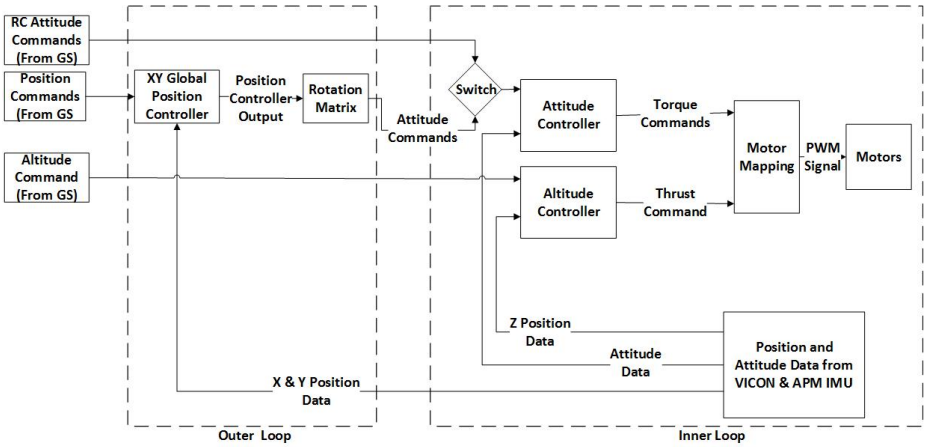

Figure 1.2: Control law architecture

τ_θ for the pitch torque, τ_ψ for the yaw torque) in each direction. A mapping matrix is used to convert the desired thrust and torques to the desired motor velocities. The desired velocities are converted to obtain the corresponding Pulse Width Modulation (PWM) signals (δ_i is the PWM signal of motor i, where $i = 1, 2, 3, 4$) which are sent to the motors [2, 3, 4, 5, 6, 7, 8, 9]. The control laws use measurements from a VICON vision system and on-board IMU. An overview of the architecture is shown in Figure 1.2.

1.2 Literature Review & Motivation

The first stable quadcopter vehicle was built as manned vehicle by Etienne Oehmichen in 1922. He continued to developed the aircraft until he was dissatisfied with altitude performance. Then, he abandoned the configuration in favor of a single main rotor vehicle [10]. The manned 1956 Convertawings Model A Quadrotor was the next major development in which thrust to each motor is varied for control [10]. However, lack of commercial and military interest led to the design being discontinued [11].

Interest in quadcopters was revived in the 2000's with renewed interest in micro air vehicles (MAVs). In 2004, Bouabdallah, Siegwart, and Murrieri [2] published a paper in the IEEE Conference on Robotics Automation, detailing a basic quadrotor dynamic model based on the Newton-Euler equations of motion. The model uses a simple thrust and torque model that is proportional to the square of the rotor velocity, which was used in this research. The model also includes first order motor dynamics. In addition, it proposes a controller design based on Lyapunov Functions. However, this model does not include drag or damping terms, which introduces error in simulation or control design. Physical parameters of the model may be found in bench tests. This simple model is very useful and is often sufficient for research [5, 6, 7]. Parameters which are difficult to find in bench tests are approximated.

Hoffmann, Huang et al. [3] presented a more advanced dynamic model in their paper at the 2007 AIAA Guidance, Navigation and Control Conference and Exhibit. This model presents a more accurate thrust model, including the airflow through the rotors, and takes into account the translational motion of the rotors in addition to the rotor speed. The model also considers the effects of propeller flapping. In addition, the model includes simple linear drag terms in the transnational dynamics. The dynamics presented are accurate. However, parameters are difficult to be identified numerically or using bench tests. In addition, it is difficult to model and measure the airflow and flapping when the aircraft is in flight and maneuvering.

Other research activities focused on finding simplified linear models that often have damping and drag information embedded. Flight data is used in either time domain analysis[8, 9] or frequency domain analysis [4] to find a linear state space model. Those models use the thrust and torques as the system input and uses mapping between the thrust and torque and the motor velocity as in [2]. However, as they are linear models, they are only accurate in a narrow performance envelope.

Motivated by the above discussions, the objectives of this research include: (1) use flight data and system identification techniques to construct an accurate non-linear quadcoptor model with parameters that is suitable for both design and simulation; (2) design and experimentally demonstrate a model-based classical control laws based off of the model. These objectives require a model that is more accurate across a greater range of conditions than the linear models given in [4, 8, 9] as well as the basic model given in [2], but simpler than the complex model constructed in [3]. In addition to control law design, this model may also be used for fault diagnostics, intelligent control, and other applications.

Chapter 2: Quadcopter Dynamic Model

A dynamic model of quadcopter systems is necessary for model-based control law designs. The model of the quadcopter is described by the Newton Euler Equations of Motion [2]. It should be noted that all motion, except for position, which is in the inertial frame, is modeled in the vehicle body frame, where the 'X'-axis, 'Y'-axis, and 'Z'-axis are aligned with the 'front', the 'right', and the bottom of the quadcopter, respectively. This choice is made due to all forces, aside from gravity, acting directly on the body of the vehicle regardless of its orientation with respect to the inertial frame. Specifically we have the following:

$$\begin{aligned} \dot{X}_E &= R_E^B(\phi,\theta,\psi)V_B \\ \dot{\eta} &= \Pi(\phi,\theta)\omega_B \\ \sum f_B &= m(\dot{V}_B + \tilde{\omega}_B V_B) \\ \sum M_B &= \tilde{\omega}_B I_B \omega_B + I_B \dot{\omega}_B \end{aligned} \tag{2.1}$$

where $X_E \triangleq [x_e, y_e, z_e]^T$ is the intertial position, $V_B \triangleq [u, v, w]^T$ is the body velocities, $\eta \triangleq [\phi, \theta, \psi]^T$ is the Euler angles, $\omega_B \triangleq [p, q, r]^T$ is the body axis rates, $\sum f_B$ is sum of all forces in the body frame, and $\sum M_B$ is sum of all moments in the body frame. $R_E^B(\phi,\theta,\psi)$ is a 3-2-1 rotation matrix from the body frame to the inertial frame as shown below [12]:

$$R_E^B(\phi,\theta,\psi) = \begin{bmatrix} C(\psi)C(\theta) & C(\theta)S(\psi) & -S(\theta) \\ C(\psi)S(\phi)S(\theta) - C(\phi)S(\psi) & C(\phi)C(\psi) + S(\phi)S(\psi)S(\theta) & C(\theta)S(\phi) \\ S(\phi)S(\psi) + C(\phi)C(\psi)S(\theta) & C(\phi)S(\psi)S(\theta) - C(\psi)S(\phi) & C(\phi)C(\theta) \end{bmatrix} \tag{2.2}$$

where $C(\cdot)$, $S(\cdot)$, $T_a(\cdot)$ denotes $cos(\cdot)$, $sin(\cdot)$, and $tan(\cdot)$, respectively. $\Pi(\phi, \theta)$ is a conversion matrix from the body axis rates to the Euler angle rates define as[13]:

$$\Pi(\phi,\theta) = \begin{bmatrix} 1 & S(\phi)T_a(\theta) & C(\phi)T_a(\theta) \\ 0 & C(\phi) & -S(\phi) \\ 0 & \frac{S(\phi)}{C(\theta)} & \frac{C(\phi)}{C(\theta)} \end{bmatrix} \tag{2.3}$$

$\tilde{\omega}_B$ is a 'cross-product' matrix of ω_B defined as [12]:

$$\tilde{\omega}_B = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.4}$$

I_B is the Inertia matrix of the quadcopter vehicle. With the vehicle assumed to be symmetrical about its axis, the matrix I_B is given by.

$$I_B = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix} \tag{2.5}$$

As discussed previously, the control of the vehicle is done by changing the speed of the four motors. Each motor applies a thrust (T_{zi}) in the negative body Z direction and generates a torque (τ_ψ) in the opposite direction of motor rotation along the body Z axis. In addition, gravity applies a force along the positive intertial Z direction, and a linear drag is used to model aerodynamic drag acting on the vehicle. The effect of these forces is modeled as follows:[14].

$$\sum f_B = \begin{bmatrix} 0 \\ 0 \\ -\sum_{i=1}^{4} C_T \rho A_R R_R^2 \omega_i^2 \end{bmatrix} + R_E^{BT} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \begin{bmatrix} C_{du} & 0 & 0 \\ 0 & C_{dv} & 0 \\ 0 & 0 & C_{dw} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.6}$$

where the thrust generated by the motors is $T_{zi} \triangleq C_T \rho A_R R_R^2 \omega_i^2$, C_T is the thrust coefficient of the motor, ρ is the air density, A_R is the area of the rotor blades, R_R is the length of the rotor blade, and ω_i is the rotational velocity of rotor i, for $i = 1, 2, 3, 4$. It also includes the effects of the gravity rotated into the vehicle body

frame as well as linear drag, with C_{du}, C_{dv}, and C_{dw} being the drag coefficient in the 'X', 'Y', and 'Z' directions, respectively. The effects of the torques are given by:

$$\begin{aligned}\sum M_B &= \rho A_R R_R^2 \begin{bmatrix} \sqrt{2}lC_1^p & -\sqrt{2}lC_2^p & -\sqrt{2}lC_3^p & \sqrt{2}lC_4^p \\ \sqrt{2}lC_1^q & \sqrt{2}lC_2^q & -\sqrt{2}lC_3^q & -\sqrt{2}lC_4^q \\ R_R C_1^r & R_R C_2^r & R_R C_3^r & R_R C_4^r \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} - \begin{bmatrix} C_{dp} & 0 & 0 \\ 0 & C_{dq} & 0 \\ 0 & 0 & C_{dr} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ &\quad + \tilde{\omega}_B I_r \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 \omega_i \end{bmatrix} \end{aligned} \tag{2.7}$$

where the torques ($\tau_{\phi i}$, $\tau_{\theta i}$, $\tau_{\psi i}$) generated by each motor are modeled as the product of a mapping matrix and the square of the motor velocities. C_i^p, C_i^q, and C_i^r denote the torque coefficients about the 'X', 'Y' and 'Z' directions, respectively, corresponding to by motor i, for $i = 1, 2, 3, 4$. The constant l is the length from the center of gravity of the quadcopter to the center of the motor projected on the body 'XY' axis of the quad. The effect of linear drag is also included, with C_{dp}, C_{dq}, C_{dr} being the drag coefficients about 'X', 'Y' and 'Z' directions, respectively. In addition, the third term of (2.7) models the gyroscopic effects of the spinning motors, with I_r being the moment of inertia of each motor.

The sum of forces and torques given by (2.6), and (2.7), along with the Newton-Euler equations of motion given by (2.1), provide a simple model of the dynamics of a quadcopter. For the convenience of control law design, these equations are rearranged into the following state space model (note: for brevity $\beta = \rho A_R R_R^2$):

$$
\begin{aligned}
\dot{x}_e &= uC(\psi)C(\theta) - wS(\theta) + vC(\theta)S(\psi) \\
\dot{y}_e &= v(C(\phi)C(\psi) + S(\phi)S(\psi)S(\theta)) - u(C(\phi)S(\psi) - C(\psi)S(\phi)S(\theta)) + wC(\theta)S(\phi) \\
\dot{z}_e &= u\,S(\phi)S(\psi) + C(\phi)C(\psi)S(\theta) - v(C(\psi)S(\phi) - C(\phi)S(\psi)S(\theta)) + wC(\phi)C(\theta) \\
\dot{\phi} &= p + rC(\phi)T(\theta) + qS(\phi)T(\theta) \\
\dot{\theta} &= qC(\phi) - rS(\phi) \\
\dot{\psi} &= (rC(\phi))/C(\theta) + (qS(\phi))/C(\theta) \\
\dot{u} &= rv - qw - gS(\theta) - \frac{C_{du}}{m}u \\
\dot{v} &= pw - ru + gC(\theta)S(\phi) - \frac{C_{dv}}{m}v \\
\dot{w} &= qu - pv + gC(\phi)C(\theta) - \frac{C_{dw}}{m}w - \frac{C_T\beta}{m}\sum_{i=1}^{4}\omega_i^2 \\
\dot{p} &= \frac{1}{I_{XX}}\left((I_{YY} - I_{ZZ})\,qr + I_r q\sum_{i=1}^{4}\omega_i + \sqrt{2}l\beta \begin{bmatrix} C_1^p & -C_2^p & -C_3^p & C_4^p \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} - C_{dp}p \right) \\
\dot{q} &= \frac{1}{I_{YY}}\left((I_{ZZ} - I_{XX})\,pr + I_r p\sum_{i=1}^{4}\omega_i + \sqrt{2}l\beta \begin{bmatrix} C_1^q & C_2^q & -C_3^q & -C_4^q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} - C_{dq}q \right) \\
\dot{r} &= \frac{1}{I_{ZZ}}\left((I_{XX} - I_{YY})\,pq + \beta R_R \begin{bmatrix} C_1^r & C_2^r & C_3^r & C_4^r \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} - C_{dr}r \right)
\end{aligned}
\tag{2.8}
$$

where the constants C_T, C_{du}, C_{dv}, C_{dw}, I_{XX}, I_{YY}, I_{ZZ}, I_r, C_i^p, C_i^q, C_i^r (for $i = 1, 2, 3, 4$), C_{dp}, C_{dq}, and C_{dr} are not known and must be determined by a parameter estimation procedure from flight test data.

The above basic model takes into account the motor thrust and torque, linear drag terms, gravity, and gyroscopic effects. However, other dynamics may not be captured in this model. To improve the accuracy of the quadcopter model, additional dynamics may be added to the equations representing the acceleration dynamics ($\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$) through a system identification process, as will be described in Section 4.3.

Chapter 3: Hardware and Software Setup

This research uses a quadcopter test environment that integrates a Gumstix Overo AirSTORM Computer on Module (COM) for flight control, a ground station using Simulink Real-Time QuaRC target to transmit data to the quadcopter and to recieve and record flight data, a VICON vision system for attitude and position tracking, and a WiFi network for communication between the COM and the ground station. The Real-Time - Marseille Grenoble Project (RT-MaG) Toolbox [1] is used to program the ground station and COM. In addition, experiments were ran on a motor test stand to map PWM to motor speed. An overview of the setup of the test environment and test stand is given in this chapter.

The overall system architecture is shown in Figure 3.1. The ground station receives the VICON information as well as user commands. The ground station transmits the position and attitude commands, VICON position and attitude data, the controller gains, controller mode (inner loop control only or outer loop control), additional PWM injections for system ID, the motor cutoff switch, and a ramp function to act as a timer to the COM via a WiFi UDP link. The COM receives this information from the ground station as well as inertial measuring unit (IMU) and battery information via an RS232 connection from a 3DR AMP 2.6 micro-controller unit. The COM runs the control algorithms and sends PWM signals to the electronic speed controllers (ESC) to control the speed of the motors. The COM also sends the attitude control commands, the VICON and IMU attitude, the VICON and IMU body rates, the VICON position, the IMU translational acceleration, the PWM signal, the ground station timer ramp, a timer ramp generated by the COM, and the battery voltage and current back to the ground station for recording.

The quadcopter is flown indoors. Therefore, flights can be performed in a consistent environment and the VICON system may be easier and more consistent in implementation. The quad is flown within the safety net cage shown in Figure 3.2. The cage protects people and other equipment in the room if the quadcopter loses control and crashes. The flight cage area is $7.6m$x$5.2m$x$2.8m$. For safety reasons, the flight control system prohibits position commands outside of a $4m$x$2m$x$2m$ area centered on the ground in the middle of the flight cage.

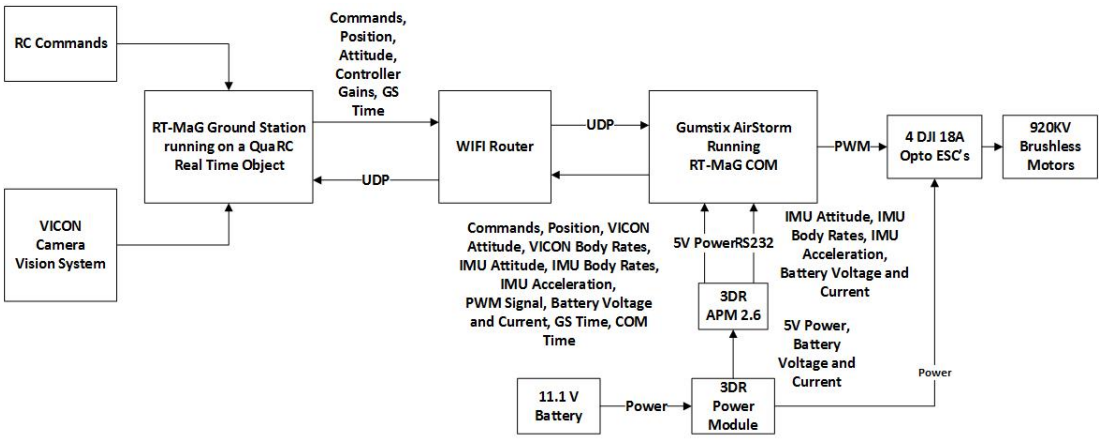

Figure 3.1: System architecture

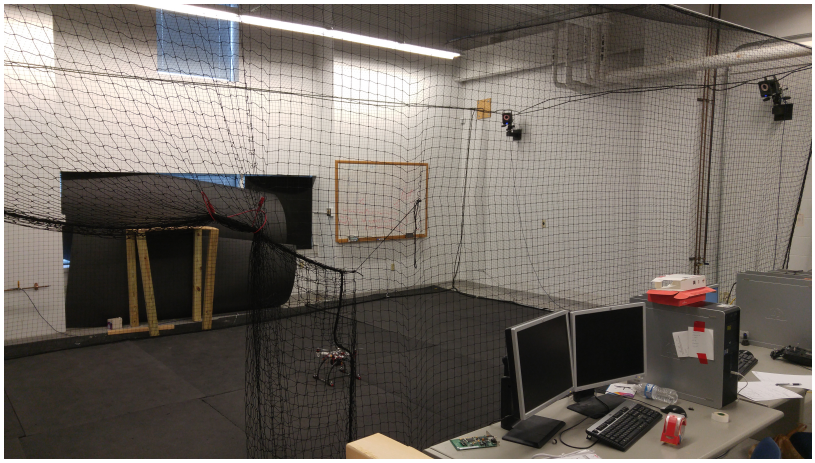

Figure 3.2: Quadcopter flying area

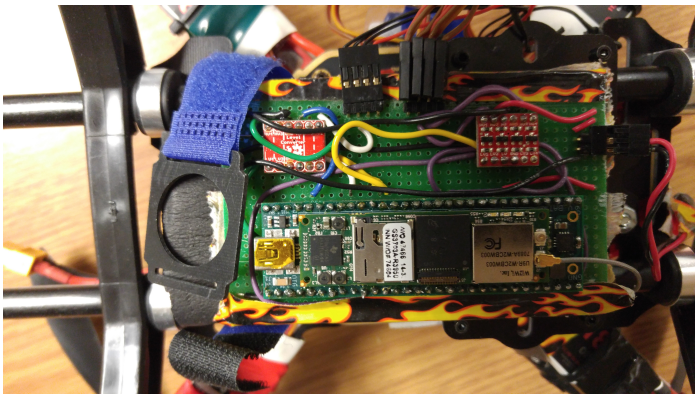

Figure 3.3: Gumstix proto-board

3.1 Quadcopter Hardware

The quadcopter is built on a DJI Flamewheel F330 air frame with a ZJchao landing skid. It's powered by a 11.1v 2200mAh 30c Lithium Polymer battery which powers four DJI 920kV Brushless Motors with 8in propellers as well as other electronics on board. These electronics include, a Gumstix Overo AirSTORM mounted on a Gumstix Pinto-TH breakout board which acts as the central COM, a 3DR APM 2.6 which provides IMU measurements, a 3DR APM power module which supplies 5V power to the Gumstix and APM, and four DJI 18A OPTO electronic speed controllers (ESCs) which control the power supplied to the motors. The AirSTORM only communicates in 1.8V signals. The APM serial port communicates at 3.3V, and the ESCs read 5V PWM signals. Two SparkFun bi-directional logic level converters are used to shift the voltage levels between the Gumstix and the APM, and between the Gumstix and the ESCs. The Pinto-TH and logic level converters are soldered to a proto-board (shown in Figure 3.3) which has header connections for power, the RS232, and the ESC output. The assembled quadcopter is shown in Figure 3.4. The mass of the quad (m), the radius of the propellers (R_R), the distance from center of the motors to the center of mass (l), and the air density are measured. These measurements are given in Table 3.1. Figure 3.4 shows how the lengths were measured.

Table 3.1: Basic quadcopter measurements

Mass m	1.214	kg
Rotor radius R_R	0.1016	m
Arm Length l	.167	m
Air Density ρ	1.225	$\frac{kg}{m^3}$

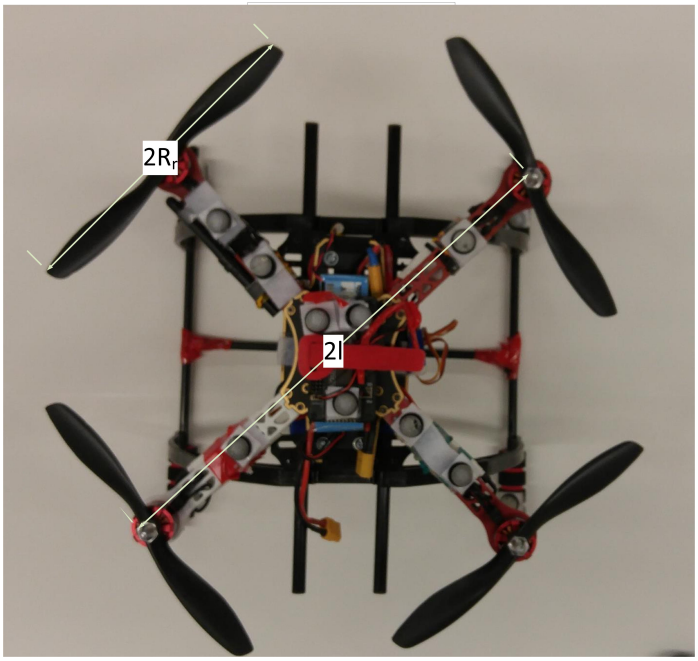

Figure 3.4: Quadcopter used during research

3.2 VICON

The flight area uses a VICON MX GiganetBox T-Series system using four T-160 cameras. VICON is a visual tracking system that tracks infrared light's reflection from the retro-reflective pearl markers attached to the quad. Each of the four cameras emits infrared light from LED's. The VICON system uses the 2D location of the pearl markers on each camera to synthesize the marker's most probable location in a 3D space using the known locations of each of the cameras. A collection of pearl markers points may be associated to form an object. Then, as long as the relative positions of the markers does not change, VICON will be able to recognize the object and send the information of the position relative to the origin, and the attitude relative to object's orientation when the object was initially created. An example of object tracking of VICON is shown in Figure 3.5. The position and attitude information is streamed to the ground station using RT-MaG's VICON QuaRC block.

The VICON vision system requires the pearl markers to be attached to the quadcopter so that the VICON vision system may create an object to track. VICON determines the center of the object based on the center of geometry of the pearl markers. However, flight dynamics are described relative to the center of mass. The quadcopter's center of mass is approximately $5.4cm$ above the center of geometry.

VICON sends its attitude Euler angles in a X-Y-Z configuration, where the object defined based on a X-Y-Z rotation sequence to find the orientation (where ϕ_{VC}, θ_{VC}, ψ_{VC} are the angles about the 'X', 'Y' and

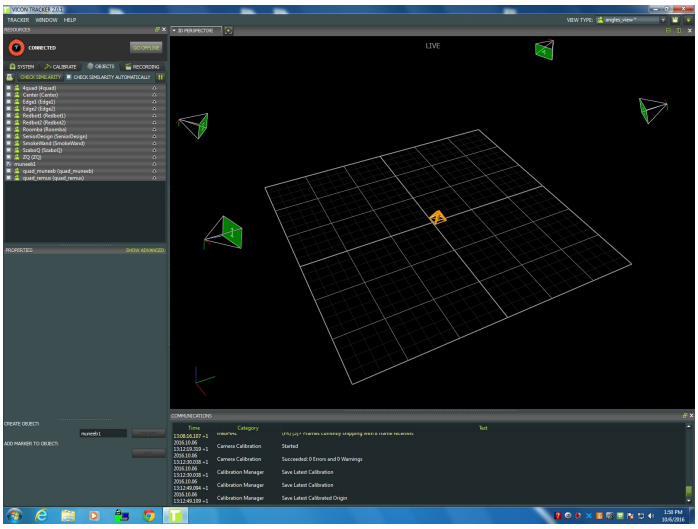

Figure 3.5: VICON tracker system

'Z' axis, respectively). This research uses the Standard Aerospace Z-Y-X rotation sequence for orientation (where ϕ, θ, ψ are the angles about the 'X', 'Y' and 'Z' axis, respectively). Therefore, the Euler angle must be converted to the new configuration. This is done by equating the two rotation matrices and solving for the Aerospace Euler angle based on the numerical values obtained from the VICON Euler angle rotation matrix. Specifically, this process is given below:

$$
\begin{aligned}
R_{11} &= cos(\theta_{VC})cos(\psi_{VC}) \\
R_{12} &= cos(\phi_{VC})sin(\psi_{VC}) + sin(\phi_{VC})sin(\theta_{VC})cos(\psi_{VC}) \\
R_{13} &= sin(\phi_{VC})sin(\psi_{VC}) - cos(\phi_{VC})sin(\theta_{VC})cos(\psi_{VC}) \\
R_{23} &= sin(\phi_{VC})cos(\psi_{VC}) + cos(\phi_{VC})sin(\theta_{VC})sin(\psi_{VC}) \\
R_{33} &= cos(\phi_{VC})cos(\theta_{VC}) \\
\theta &= -asin(R_{13}) \\
\phi &= atan2(\frac{R_{23}}{cos(\theta)}, \frac{R_{33}}{cos(\theta)}) \\
\psi &= atan2(\frac{R_{12}}{cos(\theta)}, \frac{R_{11}}{cos(\theta)})
\end{aligned}
\tag{3.1}
$$

The VICON camera system is a valuable tool to provide information of the attitude and position of the quadcopter to the control system, enabling the parameter estimation and control of the quadcopter.

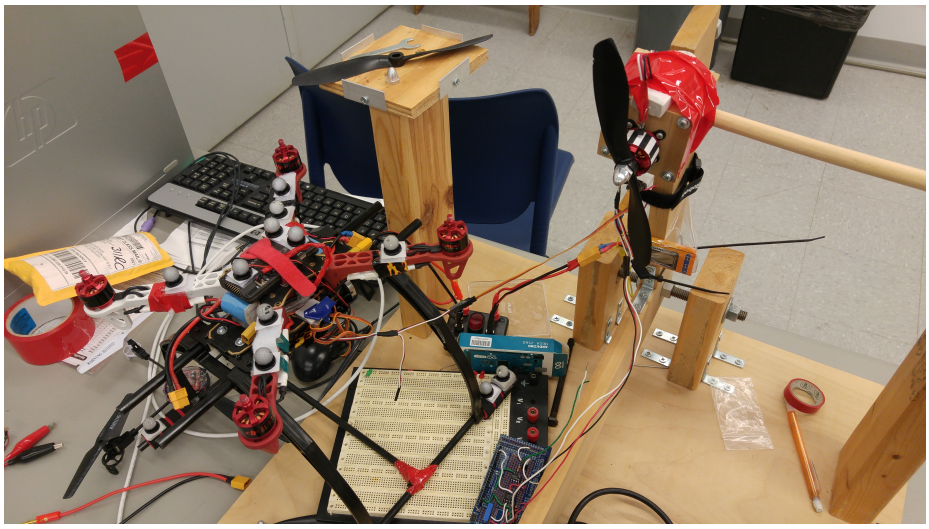

Figure 3.6: Motor test stand

3.3 Motor Test Stand

Ideally, the motor's velocity would be directly proportional to the PWM signal sent to the ESC. However, during testing, it was found that the motor velocity decreased as the battery voltage decreased. To characterize the relationship between battery voltage, PWM, and motor velocity, a bench test stand (shown in Figure 3.6) was used.

The test stand measures the velocity of one motor directly using a light sensor measuring the light reflected from a strip with black and white bands taped around a motor. When the light sensor detects the light reflected by one of the white bands, its emits a voltage pulse. An Ardiuno micro-controller is used to count these pulses and find the velocity of the motor using the knowledge of the density of black and white bands[15].

To make sure the test is as close to flight conditions as possible, the motor on the test stand was supplied with PWM signals from the quadcopter. A sum of sine waves were sent to all four motors (for safety reasons, the propellers were taken off from all motors except the motor on the test stand), such that it would be under a similar current load to real flight conditions. In addition, the test stand motor was powered by the quadcopter's battery, so that the voltage will decay as the battery's energy is consumed in a similar way to real flight conditions. The PWM signal and battery voltage are measured by the Gumstix, but the motor velocity is stored on a separate system. These signals need to be synchronized manually for accurate measurements. Further details of this test are given in Section 4.2.5.

3.4 RT-MaG

The Real-Time - Marseille Grenoble Project (RT-MaG) is a project developed by Gipsa-Lab and the Institute of Mouvement Sciences. It is a MATLAB toolbox that can rapidly prototype robot control systems for research and academic purposes [1]. This toolbox uses MATLAB and Simulink to auto-code programs for both a ground station (using a QuaRC real-time target) and COM (creating COM executable programs). The basic layout of RT-MaG's process is shown in Figure 3.7.

Separate Simulink models are created for both the ground station (host) and the COM. MATLAB's Embedded Coder generates C-code from these models. The ground station's model is compiled via Microsoft Visual Studio such that it can be used in a QuaRC real-time target. The COM model is compiled into C-code build to run on the PREEMPT-RT Linux Kernel running on the Gumstix. The Gumstix uses WiFi to load a COM control program build from an FTP server. Then, after it is loaded, it can be executed in real time. During this research, both the ground station and the COM were ran at a 100Hz sample rate.

RT-MaG also provides drivers for various I/O on the COM and ground station. The COM has drivers for RS232, I2C, SPI communication, four PWM outputs, and UDP communication over WiFi. The ground station can communicate with the COM via UDP communication over WiFi and read the data from VICON.

The value of RT-MaG is that it allows for quick build and reconfiguration of prototype quadcopter flight control programs using Simulink. Simulink uses intuitive block diagrams for coding making it easier to read and learn versus C-code. In addition, as an open source project, it allows advanced users to further develop the toolbox's capability and modify it to suit their needs.

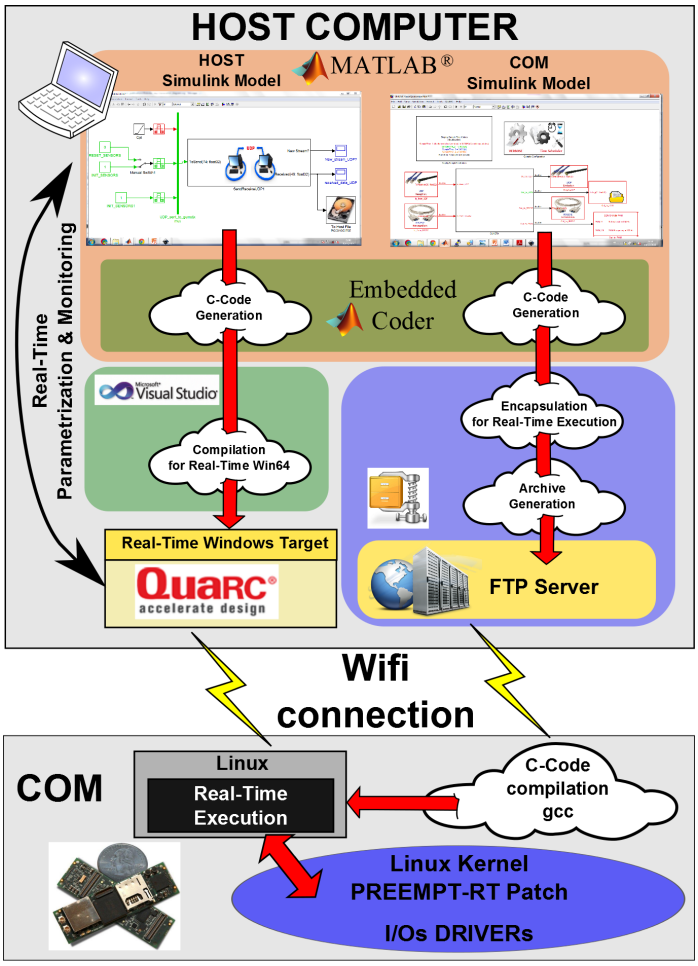

Figure 3.7: RT-MaG generation process [1]

Chapter 4: Quadcopter Model Identification

4.1 Linear Least Squares Error Method

The dynamic model proposed in Chapter 2 has unknown parameters, such as the thrust and torque coefficients. Parameter estimation is used to determine the values of these parameters, based on flight test data. Many methods of parameter estimation require computationally expensive Jacobians of models, which must be derived again if there are any changes to the model. An exception of this is in the case that the model is linear in the parameters. A linear in the parameters model is written as $\mathbf{Y} = \mathbf{XC}$ where $\mathbf{X} \in \mathbb{R}^{N \times m}$ is the observation matrix, $\mathbf{C} \in \mathbb{R}^{m}$ is the parameter vector, and $\mathbf{Y} \in \mathbb{R}^{N}$ is the target data vector. In this case, the Linear Least Squares Error (LLSE) Method of parameter estimation can be used.

Linear Least Squares Error Method of parameter estimation finds the values of the parameters which result in the smallest squared error is the vector between the target data ($\mathbf{Y}$) and the output estimate generated by the proposed estimate model ($\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{C}}$), where $\hat{\mathbf{C}}$ is the parameter estimate. This error is $\mathbf{E} \in \mathbb{R}^{N}$ defined as $\mathbf{E} \triangleq \mathbf{Y} - \hat{\mathbf{Y}}$. The error is squared to make positive and negative errors equal in weight. The Linear Least Square Error Method is briefly described below, further details can be found in [16].

$$\begin{aligned} \mathbf{E}^T\mathbf{E} &= (\mathbf{Y} - \hat{\mathbf{Y}})^T(\mathbf{Y} - \hat{\mathbf{Y}}) \\ \mathbf{E}^T\mathbf{E} &= (\mathbf{Y} - \mathbf{X}\hat{\mathbf{C}})^T(\mathbf{Y} - \mathbf{X}\hat{\mathbf{C}}) \end{aligned} \tag{4.1}$$

To perform the LLSE method, the error is minimized with respect to the parameter vector $\hat{\mathbf{C}}$. The process of this minimization is described as follows.

$$\begin{aligned}\frac{\partial \mathbf{E}^T\mathbf{E}}{\partial \hat{\mathbf{C}}} = \frac{\partial(\mathbf{Y}-\mathbf{X}\hat{\mathbf{C}})^T(\mathbf{Y}-\mathbf{X}\hat{\mathbf{C}})}{\partial \hat{\mathbf{C}}} &= 0 \\ -2\mathbf{X}^T\mathbf{Y} + 2(\mathbf{X}^T\mathbf{X})\hat{\mathbf{C}} &= 0 \\ (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} &= \hat{\mathbf{C}}\end{aligned} \tag{4.2}$$

This method requires being able to invert large ($N \times N$) matrices. However, MATLAB is capable of such tasks. This method will estimate parameter values, which minimize the squared error between the model estimate and the target data. Increasing the number of parameters will always decrease this squared error. However, due to the presence of noise and modeling error, there is a risk that purely minimizing the squared error will cause the parameters to fit that noise, but not be representative of the dynamic system (this case is referred to as an over-parameterized model). To verify that the proposed model is representative of the dynamic system, different data sets are used for identification and verification. If a term is representative of the model, the parameter values found from the identification model will reduce the error when testing the verification data. If the term is not representative model, the parameter values found from the identification model will not change or even increase the error when testing the verification data. Therefore, to select the most representative model, different terms are tested against the verification data to minimize the error.

4.1.1 Quadcopter Model in LLSE arrangement

To use the LLSE method to estimate the unknown parameters in (2.8), the equation must be rearranged to fit into the form of $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{C}}$ vector. In several instances, this can only be done by combining unknown parameters into a composite parameter. This is acceptable, due to the general dynamic of the quadcopter being of more interest than any particular parameter. Only the $\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r}$ terms have unknown parameters, so they are the only ones that need to be rearranged. This rearranged model is shown below:

$$\begin{aligned} m\left(\dot{u} - rv + qw + gS(\theta)\right) &= uC_{du} \\ m\left(\dot{v} - pw + ru - gC(\theta)S(\phi)\right) &= vC_{dv} \\ m\left(\dot{w} - qu + pv - gC(\phi)C(\theta)\right) &= \begin{bmatrix} -\beta\sum_{i=1}^{4}\omega_i^2 & w \end{bmatrix}\begin{bmatrix} C_T & C_{dw} \end{bmatrix}^T \end{aligned}$$

$$
\begin{aligned}
\dot{p} &= \begin{bmatrix} \sqrt{2}l\beta\omega_1^2 & -\sqrt{2}l\beta\omega_2^2 & -\sqrt{2}l\beta\omega_3^2 & \sqrt{2}l\beta\omega_4^2 & qr & q\sum_{i=1}^{4}\omega_i & p \end{bmatrix} \\
&\quad \times \begin{bmatrix} \frac{C_1^p}{I_{XX}} & \frac{C_2^p}{I_{XX}} & \frac{C_3^p}{I_{XX}} & \frac{C_4^p}{I_{XX}} & \frac{I_{YY}-I_{ZZ}}{I_{XX}} & \frac{I_r}{I_{XX}} & \frac{C_{dp}}{I_{XX}} \end{bmatrix}^T \\
\dot{q} &= \begin{bmatrix} \sqrt{2}l\beta\omega_1^2 & \sqrt{2}l\beta\omega_2^2 & -\sqrt{2}l\beta\omega_3^2 & -\sqrt{2}l\beta\omega_4^2 & pr & p\sum_{i=1}^{4}\omega_i & q \end{bmatrix} \\
&\quad \times \begin{bmatrix} \frac{C_1^q}{I_{YY}} & \frac{C_2^q}{I_{YY}} & \frac{C_3^q}{I_{YY}} & \frac{C_4^q}{I_{YY}} & \frac{I_{ZZ}-I_{XX}}{I_{YY}} & \frac{I_r}{I_{YY}} & \frac{C_{dq}}{I_{YY}} \end{bmatrix}^T \\
\dot{r} &= \begin{bmatrix} \beta R_R\omega_1^2 & \beta R_R\omega_2^2 & \beta R_R\omega_3^2 & \beta R_R\omega_4^2 & pq & r \end{bmatrix} \\
&\quad \times \begin{bmatrix} \frac{C_1^r}{I_{ZZ}} & \frac{C_2^r}{I_{ZZ}} & \frac{C_3^r}{I_{ZZ}} & \frac{C_4^r}{I_{ZZ}} & \frac{I_{XX}-I_{YY}}{I_{ZZ}} & \frac{C_{dr}}{I_{ZZ}} \end{bmatrix}^T
\end{aligned}
\tag{4.3}
$$

For each model the thrust coefficient (C_T) of each motor is assumed to be the same. While C_T does, in fact, vary between motors in practice, having different C_T was seen to over-parameterize the model. When so arranged, the unknown parameters are in the necessary form for LLSE parameter estimation and the parameters can be estimated, if given suitable flight data.

As mentioned in Chapter 2, additional dynamics may be added, if necessary, to improve the accuracy of the model. Fortunately, new terms may be easily added to the observation matrix. For example, if a r term was to be added to model the 'Z' force dynamic, then the dynamic in (4.3) would be changed into the following

$$
m\left(\dot{w} - qu + pv - gC(\phi)C(\theta)\right) = \begin{bmatrix} -\beta\sum_{i=1}^{4}\omega_i^2 & w & r \end{bmatrix} \begin{bmatrix} C_T & C_{dw} & C_{dr}^w \end{bmatrix}^T \tag{4.4}
$$

4.2 Gathering & Processing Flight Data

4.2.1 Flight Excitation

As discussed in Section 1.1, the quadcopter is directly controlled by varying the speed of the four motors. In order to estimate parameters from flight data, the data must be frequency rich. Since there are four seperate controlled inputs, each input must not be correlated [17]. Inputs must be uncorrelated so that effects of each input is distinct. To achieve this, a sine wave is applied to each motor PWM signal while it is in flight to ensure a frequency rich test. The frequency and amplitude of this signal is determined randomly by SimuLink's Band-Limited White Noise Blocks. The frequency has a noise power (height of the power spectral density) of .1, and the result of the absolute value of the Band-Limited White Noise Block is multiplied by $8Hz$ to

convert to hertz and expand the variance. The amplitude has a noise power of .25, and the results of the Band-Limited White Noise Block is multiplied by 15 μs and then has 15μs added to the result. The absolute value of this result is used as the amplitude. These adjustments are made to apply a large enough amplitude signal sent to the PWM ports of the RT-MaG COM to excite the flight control to obtain frequency rich data, but not so large to overwhelm the control law and crash the quadcopter. The amplitude and frequency of the sine wave is kept constant for several seconds and then changed at different times (2, 2.1, 1.8, and 2.31 seconds for each respective motor). This staggered time change, as well as the random frequencies and amplitude, decreases the likelihood of correlated input signals. In addition to the random signals injected, the control law's output attempting to stabilize the aircraft also makes the signals more rich. In addition, different flight paths are executed for each test. Also the control law for these tests includes an inner loop Proportional-Derivative (PD) controller for roll, pitch, and yaw control, and an outer-loop Proportional-Integral-Derivative (PID) controller for position control. The gains of these loops are hand tuned for stable and controlled flight.

A total of five flight tests are obtained, with each test flying a different position pattern. And due to the random nature of the excitation, each test was excited differently, so that the test will be representative of how the quadcopter behaves. Although each flight had a different duration of time, $20s$ of data are used for Parameter Estimation.

4.2.2 Test Flight Control Law

Quadcopter UAVs are inherently unstable platforms, so a stabilizing control law is necessary to obtain flight data. Since there is no model, the control law is created via tuning the gains. For the inner loop (see Figure 1.2), a PD control was used for roll, pitch, and yaw, and a PID control was used for altitude. For the outer loop, PID control was used for the X and Y positions. For all control laws, estimated velocity feedback was used.

Since the model parameters have not been found, exact motor mapping is not feasible and only the signs the parameters are mapped. The thrust and torque values are converted to PWM values via the mapping matrix given below:

$$\begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{4.5}$$

Where δ_i is the value of the PWM signal sent to the ith motor, for $i = 1, 2, 3, 4$, in μs.

4.2.3 Flight Data Preprocessing

Flight data is supplied by the VICON system, the APM's IMU and battery voltage measurement, as well as the PWM output of the control signal generated by the COM. Although the data is recorded on the ground station, all measurements are fed through the Quadcopter COM. So that all measurements are synchronized. This keeps the data in sync, relative to the control law decisions. The VICON directly provides global position and Euler angles. The APM provides body angle rates and battery voltage. All this data must be processed to generate the target data (**Y**) and Observation Matrix (**X**).

For the Euler Angles, the VICON measurements are used instead of the APM IMU measurements, because VICON transmission is believed to have less lag and less noise. In addition, the APM yaw measurements drift, even if the Quadcopter is stationary. For the body rates, the APM measurements have a bias, but are less noisy at high frequencies. To use VICON, the Euler rates are estimated from VICON Euler Angles on board via a linear 2^{nd} order band-pass filter $\frac{2500s}{s^2+100s+2500}$. The Euler rates are rotated to generate body rates by multiplying the inverse of Equation (2.3). The VICON body rates do not have a bias and are accurate at low frequency. But they are noisier at high frequencies, so a complementary filter was added to combine the signals as shown below

$$PQR_{combined} = \frac{5}{s+5} * PQR_{VICON} + \frac{s}{s+5} * PQR_{APM} \tag{4.6}$$

where $PQR \triangleq [p, q, r]^T$ is the body rates, 5 $\frac{rad}{s}$ was found to work well via trial and error. There is no direct measurement signal for body rate acceleration, so the derivative of the combined body rates with respect to time (via the polynomial regression filter described in Section 4.2.4) was used.

The body frame velocities were found by taking the derivative of the global position (also via polynomial regression filter) and then multiplying rotation matrices (inverse of (2.2)). As noted in Section 3, the VICON provides a Center of Geometry (CoG), not the Center of Mass (CoM). The Center of Mass is $5.4cm$ above the Center of Geometry. To obtain the body frame velocity, the velocity of that difference between the CoG and CoM must be taken into account using the following equation:

$$V_{BCoM} = V_{BCoG} - \tilde{\omega}_B \begin{bmatrix} 0 \\ 0 \\ 0.054 \end{bmatrix} \tag{4.7}$$

Where $\tilde{\omega}_B$ is the cross-product matrix given in (2.4). The body acceleration provided by the IMU is very

noisy and not reliable, therefore the derivative of the body velocity obtained via polynomial regression filter is used. In addition, for all flight measurements, a polynomial regression filter is used for data smoothing.

4.2.4 Polynomial Regression Filter

Measurement noise and the need for derivatives necessitate the use of filters to smooth the data and obtain derivatives. Linear filters are often used, but they introduce a phase lag into the data. For parameter estimation, this lag is not desirable, because lag would cause data to become out of sync. To prevent this lag and allow a smooth derivative to be computed, a non-causal FIR filter is used, taking advantage of the fact that this data is processed after the flight and may use future values during processing.

The polynomial regression filter is designed with the following objectives: 1) low lag, 2) reduce measurement noise, 3) obtain a derivative of the signal with a minimal amount of noise. The polynomial regression filter accomplishes this by taken a polynomial regression of a section of the signal, with an equal amount of data point before and after the current value, and then repeating this process for every point. Polynomial regression is used because polynomials are easily differentiable analytically. The parameters of the polynomial regression filter are the order of the polynomial regression (M), the time window (W) that will be used for the polynomial estimate, and (T_s) the sampling period of the data. The polynomial regression is taken with respect to time, and only the time relative to the current point ($y(n_j)$, where n_j is the current sample) is necessary, using an equal number of samples $\left(\frac{W}{2T_s}\right)$ before and after current sample. The form of this polynomial regression is given below:

$$y(n) \quad = \quad \sum_{q=0}^{M} G_q(n-n_j)^q + \epsilon(n-n_j), \qquad n \in \left[-\frac{W}{2T_s}+n_j, \frac{W}{2T_s}+n_j\right] \tag{4.8}$$

Where G_q is the the coefficient of the q-th order polynomial and $\epsilon(n-n_j)$ is the error between the polynomial estimate and the actual data. Polynomials coefficients are linear in their parameters, thus the LLSE method, described in Section 4.1, may be used to solve for the coefficients. The observation matrix ($\mathbf{X}$), being the $(n-n_j)^q$'s, the target data ($\mathbf{Y}$) is $y(n)$, and solves for the coefficients ($\mathbf{X}$). Since $(n-n_j)$ is does not depend on the current sample, the observation matrix ($\mathbf{X}$) of the polynomial regression will be constant. And the input signal $y(n)$ and $\frac{W}{2T_s}$ points before and after the current value. The polynomial regression will take a form of the an FIR filter designed so that the first row of The weight matrix ($\mathbf{G}$) is used to find the filtered data estimate ($\widehat{y}(n)$), and the second row is used to find the derivative estimate $\left(\dot{\widehat{y}}(n)\right)$, specifically the filter takes the form below

$$
\begin{aligned}
\mathbf{Y} &= \Big[y(n_j - \tfrac{W}{2T_s}) \quad y(n_j - \tfrac{W}{2T_s} + 1) \quad \dots \quad y(n_j - 1) \quad y(n_j) \quad y(n_j + 1) \\
&\qquad \dots \quad y(n_j + \tfrac{W}{2T_s} - 1) \quad y(n_j + \tfrac{W}{2T_s})\Big]^T \\
\mathbf{X} &= \begin{bmatrix}
1 & (-\frac{W}{2T_s})^1 & \dots & (-\frac{W}{2T_s})^{M-1} & (-\frac{W}{2T_s})^M \\
1 & (-\frac{W}{2T_s}+1)^1 & \dots & (-\frac{W}{2T_s}+1)^{M-1} & (-\frac{W}{2T_s}+1)^M \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
1 & (-1)^1 & \dots & (-1)^{M-1} & (-1)^M \\
1 & 0 & \dots & 0 & 0 \\
1 & (1)^1 & \dots & (1)^{M-1} & (1)^M \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
1 & (\frac{W}{2T_s}-1)^1 & \dots & (\frac{W}{2T_s}-1)^{M-1} & (\frac{W}{2T_s}-1)^M \\
1 & (\frac{W}{2T_s})^1 & \dots & (\frac{W}{2T_s})^{M-1} & (\frac{W}{2T_s})^M
\end{bmatrix} \\
\mathbf{G} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \qquad (4.9) \\
\widehat{y}(n_j) &= \mathbf{G}_{1,:}\mathbf{Y} \\
\widehat{\dot{y}}(n_j) &= \mathbf{G}_{2,:}\mathbf{Y}
\end{aligned}
$$

Where $\mathbf{G}_{1,:}$ is the first row of $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ and $\mathbf{G}_{2,:}$ is the second row of $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. This process is repeated for every point estimated, this estimate excludes the first and last $\frac{W}{2T_s}$ data points, because of the need to use $\frac{W}{2T_s}$ data points before and after a point to find an estimate.

For this research, the order of the polynomial used was $M = 4$ and the window was $W = .3s$. This was found to to produce reliable estimate for a derivative and smooth the data. The result generated by the polynomial regression filter for the roll measurement of a flight test is shown in Figure 4.1. The results for the estimating the global X velocity of a flight test is shown in Figure 4.2.

4.2.5 Motor Velocity Mapping & Dynamic Characteristics

The quadcopter's COM only provides the PWM value sent to each of the motors. For flight control, the actual rotational velocity of the rotors is of interest. To find the relationship between the PWM signal and the actual motor velocity, the test station in Section 3.3 was used. With one PWM output connected to testing apparatus and the rest connected normally (note: the rotor blades are removed from those three

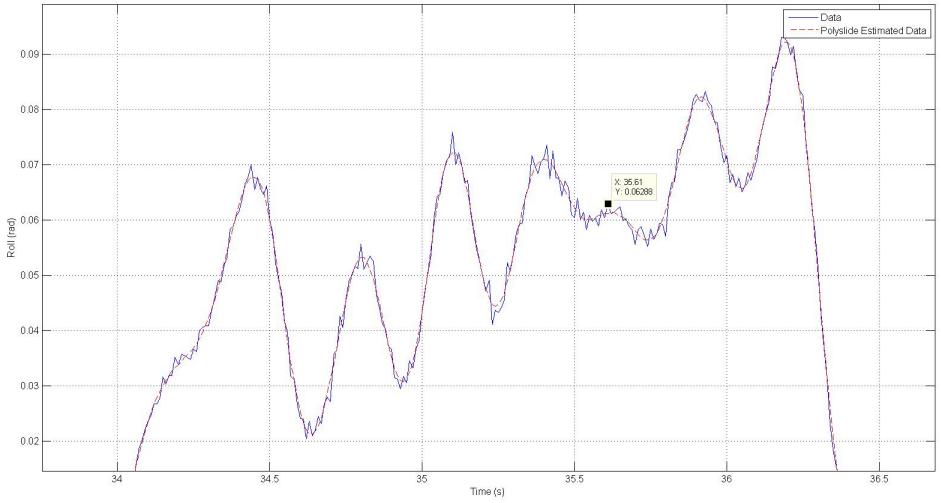

Figure 4.1: PolySlide estimate compared to raw signal

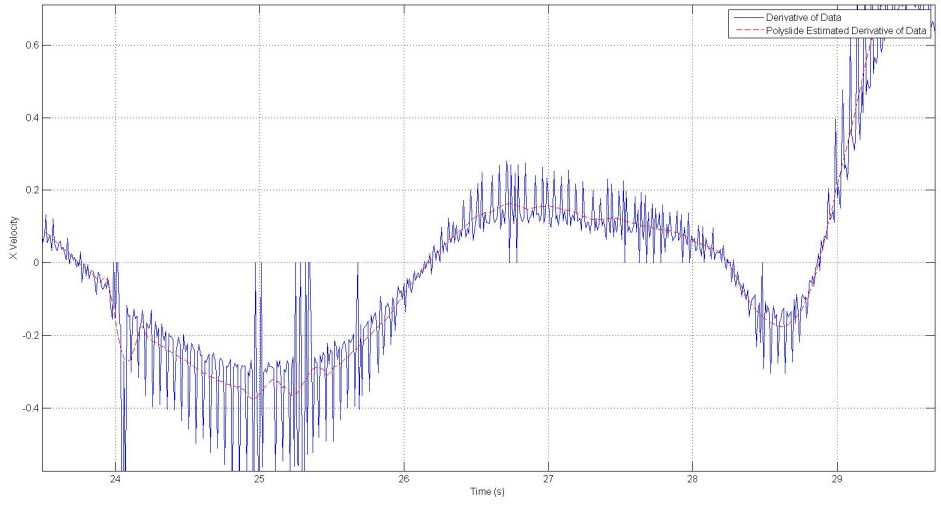

Figure 4.2: PolySlide derivative estimate compared to raw signal derivative

motors), the motors are given a predetermined PWM signals of a saw-tooth and a Sine wave, while the single rotor's velocity is measured. The COM records the PWM signals and the battery voltage (the battery voltage is filtered to reduce noise by a linear filter $\frac{.02}{z-.98}$). The computer which controls the test stand records the rotational velocity. Since the values are from two separate unsynchronized sources, they must be synchronized later. To map the PWM value to the rotational velocity of the motor, LLSE is used to minimize error which is configured as follows:

$$\begin{aligned} \mathbf{Y} &= \omega_{motor} \\ \mathbf{X} &= \begin{bmatrix} \delta & V_{battery} & 1 \end{bmatrix} \end{aligned} \tag{4.10}$$

$$\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 0.7850 & 44.6505 & -1205.1 \end{bmatrix}^T \tag{4.11}$$

where ω_{motor} is the rotational velocity of the motor, δ is the PWM signal, and $V_{battery}$ is the battery voltage. In addition, the DC motors used have their own time constant, which affects the rotor velocity. This time constant is modeled as a first-order transfer function $\left(\frac{21.5}{s+21.5}\right)$ [15]. This filter is applied after the rotational velocity is obtained using 4.10. The DC motor time constant and estimate of the rotational velocity of the motors are used in the parameter estimation.

4.3 Parameter Estimation Process

Five flight data sets were obtained, labeled Flight Data Sets A, B, C, D, and E. Flight Data Sets A, B, and C were used to estimate the parameters, and Flight Data Sets D and E were used as validation data. The parameter estimation process is shown in Figure 4.3. The raw data is first pre-processed, as described in Section 4.2.3, to obtain the desired signal for parameter estimation. The signals are then "stacked" on top of one another, such that all three estimation data sets are combined into one super-set. If the individual observation matrices are $\mathbf{X}_A$, $\mathbf{X}_B$, $\mathbf{X}_C$, then the combined observation matrix is $\mathbf{X}_{ABC} = \begin{bmatrix} \mathbf{X}_A^T & \mathbf{X}_B^T & \mathbf{X}_C^T \end{bmatrix}^T$). The parameter estimates $\left(\hat{\mathbf{C}}\right)$ are generated from this super-set of data. To verify the data, Flight Datasets D, and E are loaded and pre-processed to generate the desired signals. The estimated parameters used with this data to find the sum of squared error of the validation data. For example, the equation below shows the sum of squared error of Flight Data D:

$$e_D^2 = (\mathbf{Y}_D - \mathbf{X}_D\hat{\mathbf{C}})^T(\mathbf{Y}_D - \mathbf{X}_D\hat{\mathbf{C}}) \tag{4.12}$$

The sum of squared error from Flight Data D and E are then summed to obtain the total error.

4.4 Parameter Estimation Results

Different models were run through the estimation process to find the most accurate type of model. These models were compared via the sum of squared errors. Tables 4.1, 4.3, 4.5, 4.7, 4.9, and 4.11 show the error resulting from the different models. The first column are additional dynamics added as described in Section 4.1.1, the second column is the error the model produced, the third column is the percent change in error compared to the basic model, the first row is the basic model, with no additional terms. It should be noted, that the error value itself is only useful in comparison with errors of other models for the same system.

This comparison of error shows that, in nearly every case, the basic model can be significantly improved by the addition of the other "damping" terms. The verification error of each proposed model as well as the parameter values are itemized below. In addition to the reduction of error, the selected model also shows improved qualitative performance over the basic model as shown in plots of the target data ($\mathbf{Y}_D$) and estimated data ($\hat{\mathbf{Y}}_D$) given in Figures 4.4 - 4.13.

- For the body 'x' velocity dynamic $\dot{u}$, the addition of only the pitch rate q term has the smallest sum of error (8.71% decrease in error versus the basic model, see Table 4.1). For body 'y' velocity dynamic $\dot{v}$, the roll rate p model has the smallest sum of error (12.73% decrease in error versus the basic model, see Table 4.3). The estimated values for the parameters of the model are given in Tables 4.2 and 4.4, respectively.

  Figure 4.4, 4.5, 4.6, and 4.7 show the model estimate for the low frequency data is reasonably good, while the estimation for the high frequency data is a bit worse. This is considered acceptable, because the high frequency movement is most likely a result of the difference between the VICON center of geometry and the true center of mass. Thus, torques cause a displacement in the linear directions. To model this, terms related ω_i^2 would be added. However, in addition to the location being not exactly known, the center of gravity is not consistent between flights due to shifting of the battery, therefore, this effect is left out of the model to prevent over-parameterization. The q and p terms in the $\dot{u}$ and $\dot{v}$ improve performance because they partially reflect the center of mass/center of geometry difference as well as lateral forces caused by blade-flapping.

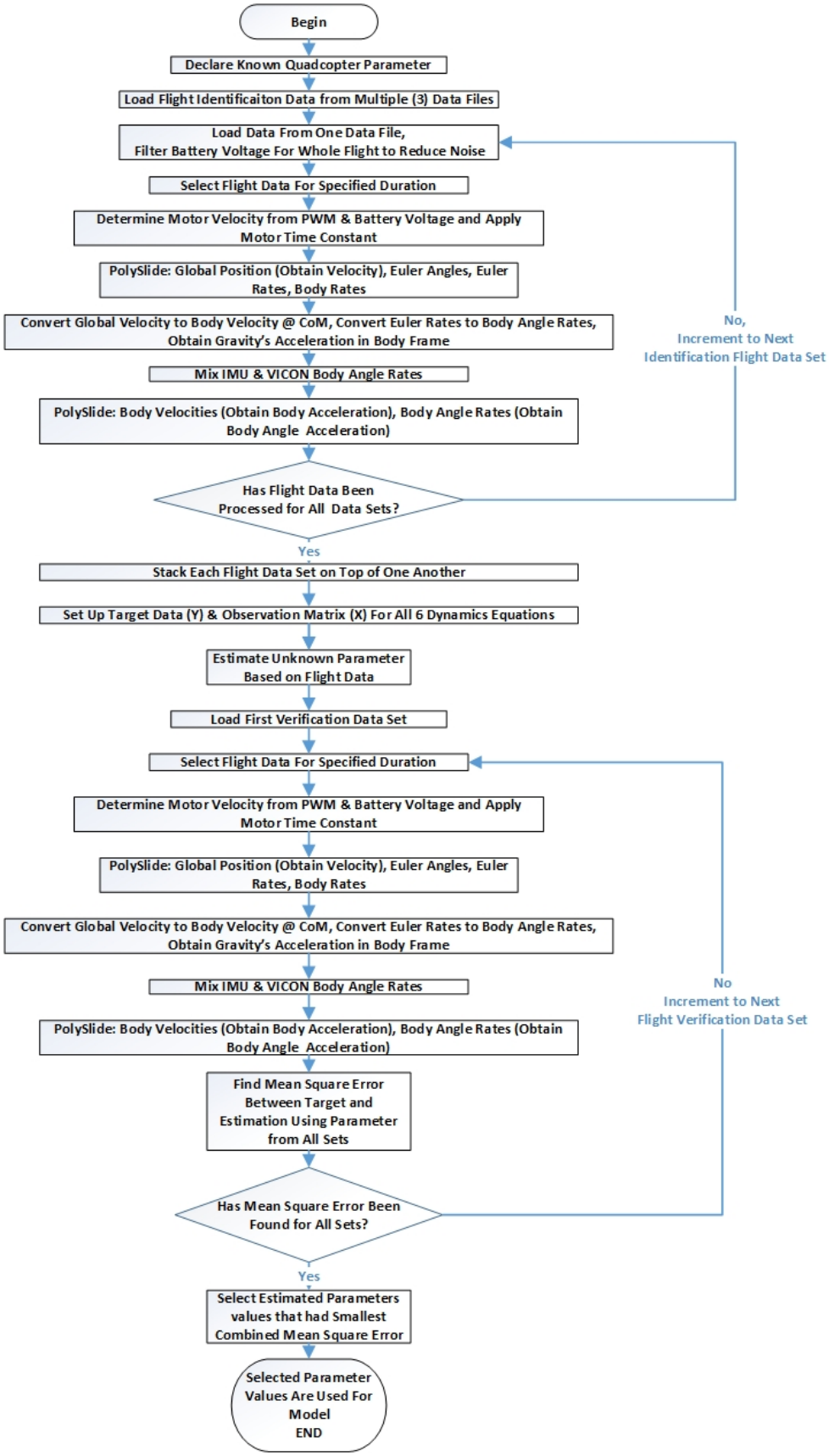

Figure 4.3: Flowchart of the parameter estimation process

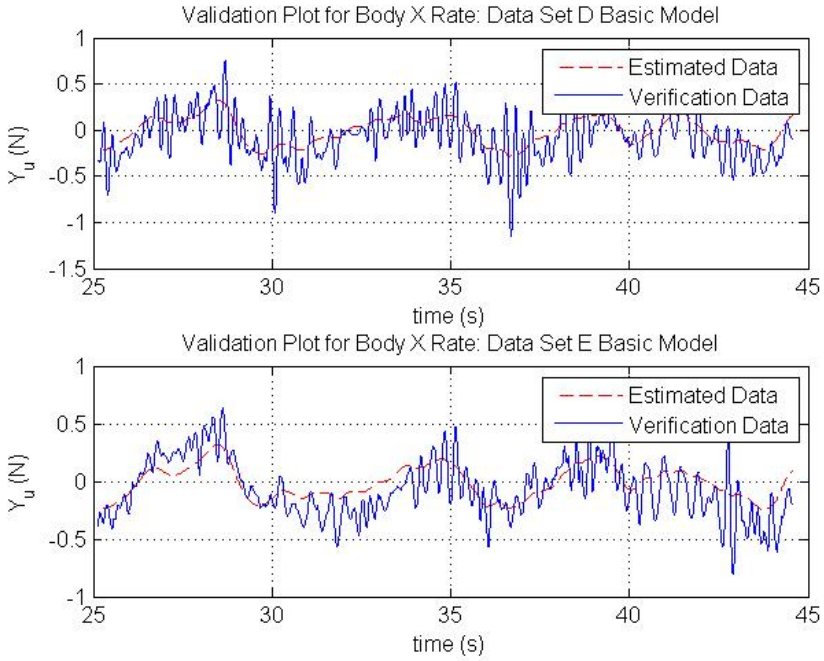

Figure 4.4: Validation plot of basic model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{u}$

Table 4.1: Sum of squared error $\dot{u}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	140.4984	
p	140.6412	-0.102%
q	128.2659	8.707%
$\{q, r\}$	129.396	7.902%
$\{q, v\}$	139.5803	0.653%
$\{q, w\}$	128.7230	8.381%

Table 4.2: Parameter values for $\dot{u}$ for selected model

C_{du}	-0.2520
C_{dq}^{u}	-0.1717

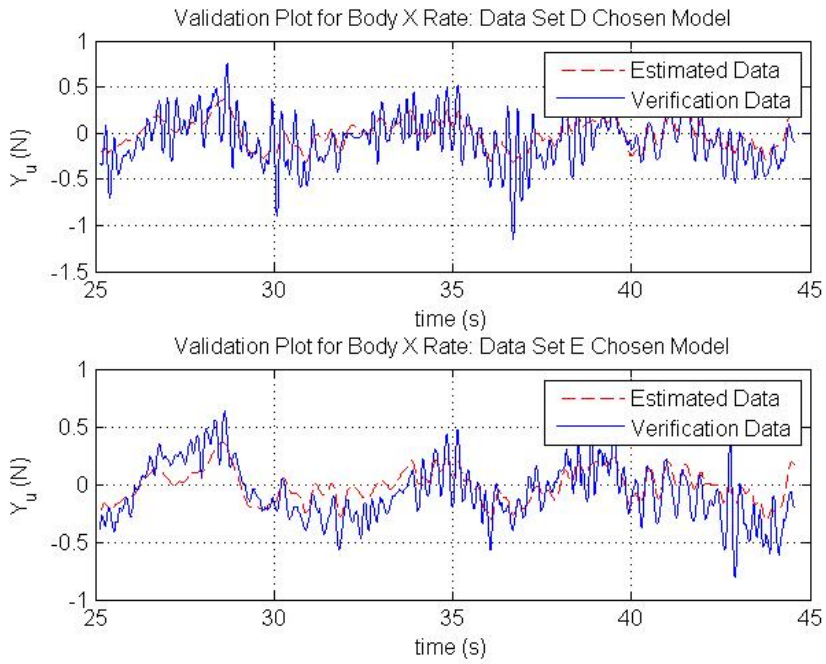

Figure 4.5: Validation plot of chosen model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{u}$

Table 4.3: Sum of squared error of $\dot{v}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	100.6242	
p	87.8119	12.733%
$\{p, q\}$	88.7965	11.754%
$\{p, r\}$	88.069	12.477%
$\{p, v\}$	98.2854	2.324%
$\{p, w\}$	89.8687	10.689%

Table 4.4: Parameter values for $\dot{v}$ for selected model

C_{dv}	-0.2684
C^v_{dp}	0.1627

- For 'z' body velocity $\dot{w}$, the basic model is used even though it does not have the smallest error. This is because pitch rate q alone did not cause a significant decrease in error (1.7% decrease in error, see Table 4.5), and it would cause an asymmetric model. In addition, the 'x' and 'y' body velocity $\{u, v\}$

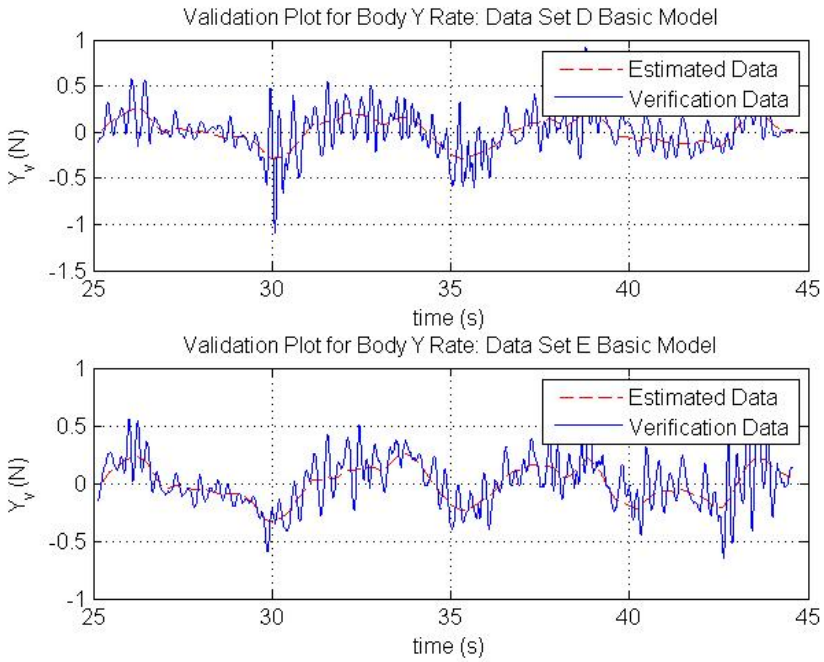

Figure 4.6: Validation plot of basic model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{v}$

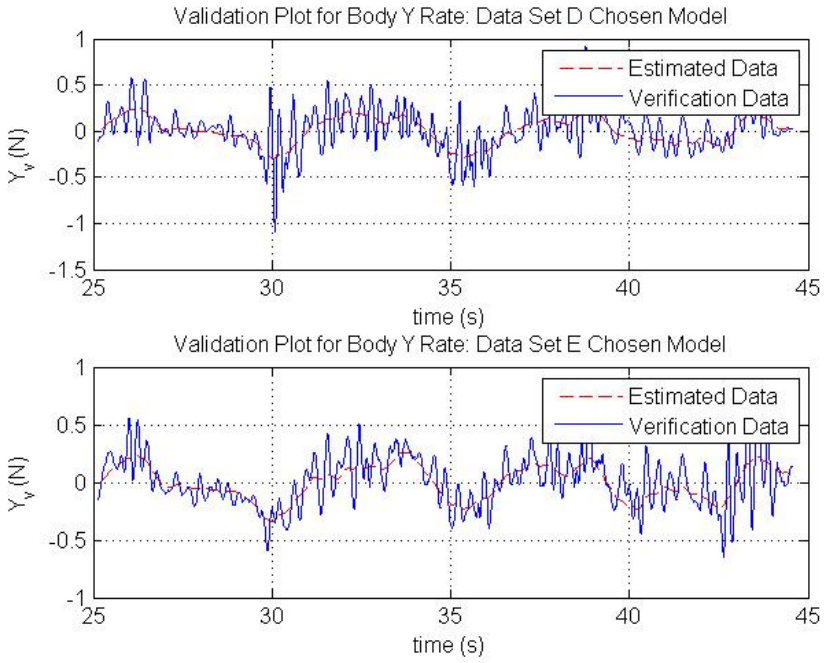

Figure 4.7: Validation plot of chosen model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{v}$

model was not used, although it is symmetrical, because the small increase in performance (0.22% decrease in error) did not seem to justify the use of the more complex model. The estimated values for the model parameters are given in Tables 4.6. Figure 4.8 shows good tracking for the $\dot{w}$ direction.

Table 4.5: Sum of squared error of $\dot{w}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	309.2878	
p	311.0144	-0.558%
q	307.9265	0.440%
$\{q, r\}$	309.2394	0.016%
$\{q, u\}$	303.9992	1.710%
$\{q, u, v\}$	304.0073	1.707%
$\{u, v\}$	308.617	0.217%
$\{p, q\}$	309.6122	-0.105%
r	310.7106	-0.460%

Table 4.6: Parameter values for $\dot{w}$ for selected model

C_T	0.0180
C_{dw}	-0.2609

- The body 'x' velocity v and the body 'y' velocity u models are used for pitch rate $\dot{p}$ and roll rate $\dot{q}$ dynamics, respectively, even though other models have a slightly smaller error. This was done to preserve symmetry between roll and pitch. Beyond these analogous models, adding extra terms may improve one model, but degrade the other. Also, the difference in error from adding terms does not appear significant. Therefore, the simpler model is used. Overall there models provide a 46.34% (see Table 4.7) and 12.67% (see Table 4.9) decrease in error for $\dot{p}$ and $\dot{q}$, respectively, compared to the basic model. The estimated values for the model parameters are given in Tables 4.8 and 4.10, respectively.

  In Figures 4.9, 4.10, 4.11 and 4.12, there is a drift at lower amplitudes of accelerations in the basic model of $\dot{p}$ and $\dot{q}$. The v and u terms, respectively, largely account for this drift, providing good tracking. This further reinforces the coupling of $\dot{u}$ and $\dot{q}$, and $\dot{v}$ and $\dot{p}$. This coupling is likely the result of an airflow, moving laterally across the rotors, as well as the displacement of the VICON center of geometry and center of mass.

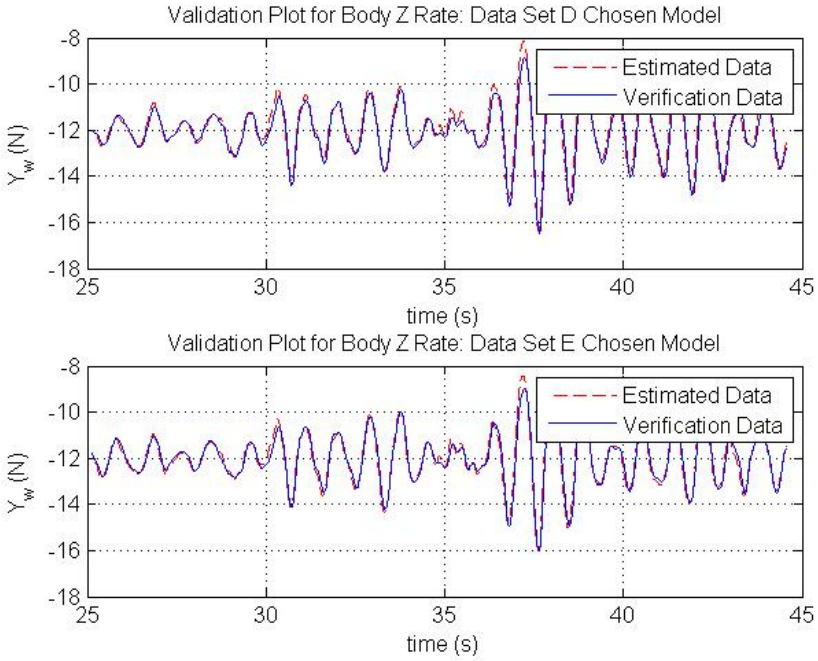

Figure 4.8: Validation plot of chosen model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{w}$

Table 4.7: Sum of squared error of $\dot{p}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	10455	
q	10453	0.019%
r	10679	-2.143%
u	11260	-7.700%
v	5610.3	46.339%
$\{u, v\}$	5826.5	44.271%
$\{v, w\}$	5698.7	45.493%
$\{v, q\}$	5535.9	47.050%

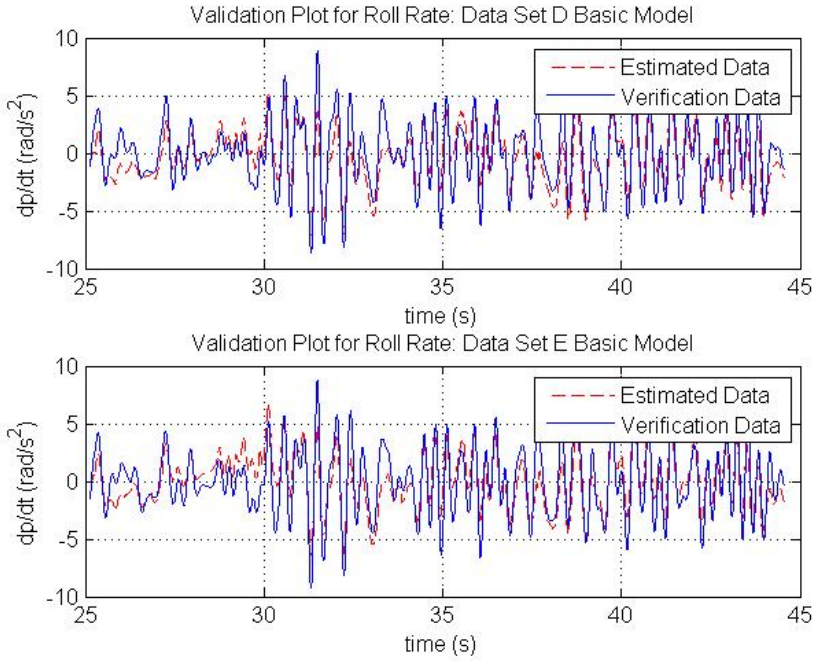

Figure 4.9: Validation plot of basic model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{p}$

Table 4.8: Parameter values for $\dot{p}$ for selected model

$\frac{C_1^p}{I_{XX}}$	0.6170
$\frac{C_2^p}{I_{XX}}$	0.6229
$\frac{C_3^p}{I_{XX}}$	0.5510
$\frac{C_4^p}{I_{XX}}$	0.5574
$\frac{I_{YY}-I_{ZZ}}{I_{XX}}$	-0.1410
$\frac{I_r}{I_{XX}}$	-0.0007
$\frac{C_{dp}}{I_{XX}}$	1.2151
C_{dv}^p	-3.7800

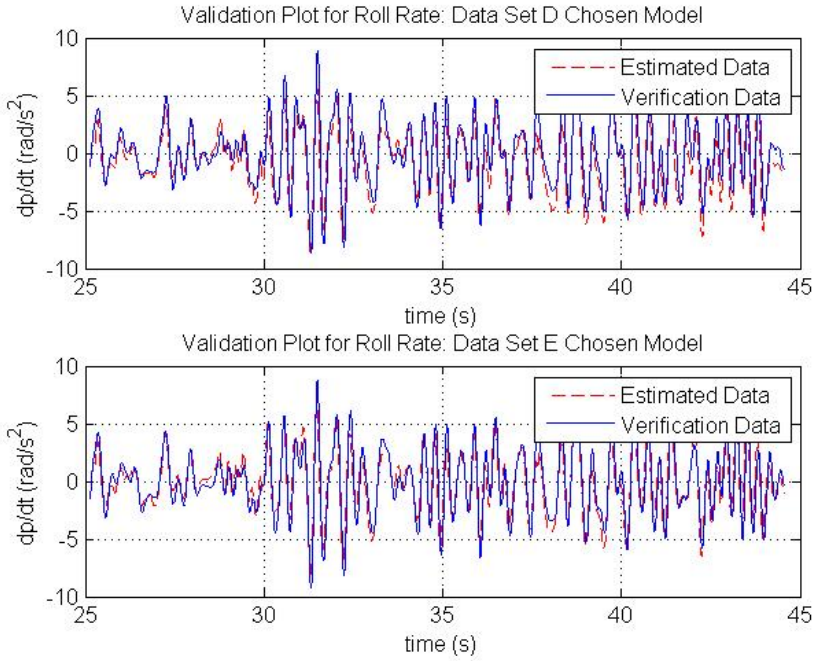

Figure 4.10: Validation plots of chosen model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{p}$

Table 4.9: Sum of squared error of $\dot{q}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	11244	
p	11278	-0.302%
r	11595	-3.122%
u	9819.7	12.667%
$\{u, v\}$	9810.2	12.752%
$\{u, v, w\}$	9810.1	12.753%
$\{u, w\}$	9819.6	12.668%
$\{v, q\}$	9853.4	12.367%

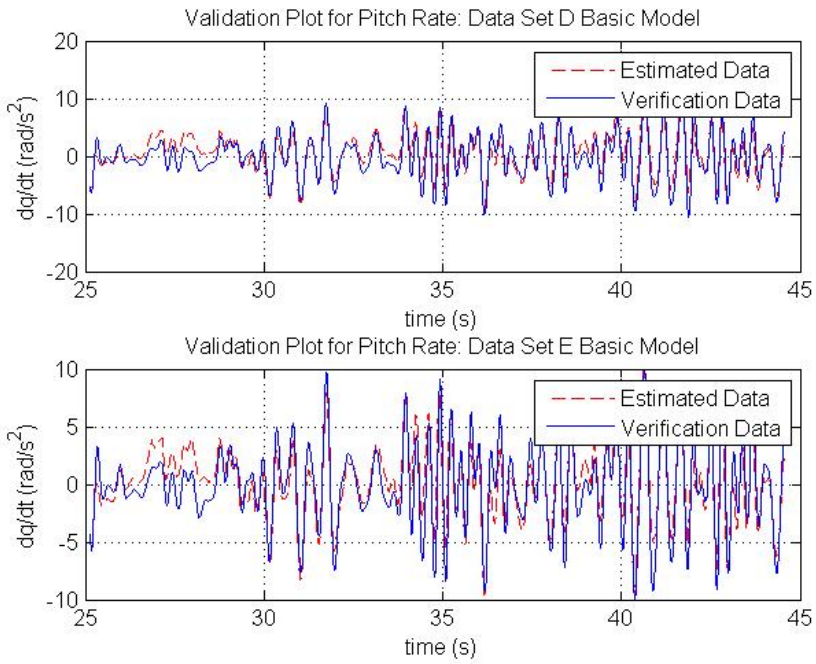

Figure 4.11: Validation plot of basic model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{q}$

Table 4.10: Parameter values for $\dot{q}$ for selected model

$\frac{C_1^q}{I_{YY}}$	0.6008
$\frac{C_2^q}{I_{YY}}$	0.7203
$\frac{C_3^q}{I_{YY}}$	0.6058
$\frac{C_4^q}{I_{YY}}$	0.6267
$\frac{I_{ZZ}-I_{XX}}{I_{YY}}$	0.7610
$\frac{I_r}{I_{YY}}$	-0.0046
$\frac{C_{dq}}{I_{YY}}$	1.9710
C_{du}^q	2.7540

- For the yaw rate $\dot{r}$, the basic model is used because no other model tested resulted in improved performance (aside from roll rate p, which is asymmetrical and does not result in a significant enough (2.43%) improvement of estimate, see Table 4.11). The estimated values for the parameters of the model are given in Tables 4.12. The $\dot{r}$ model has a low frequency drift in Figure 4.13, but tracks the higher frequency changes well. This is an issue, but an integral controller would be able to reduce the drift.

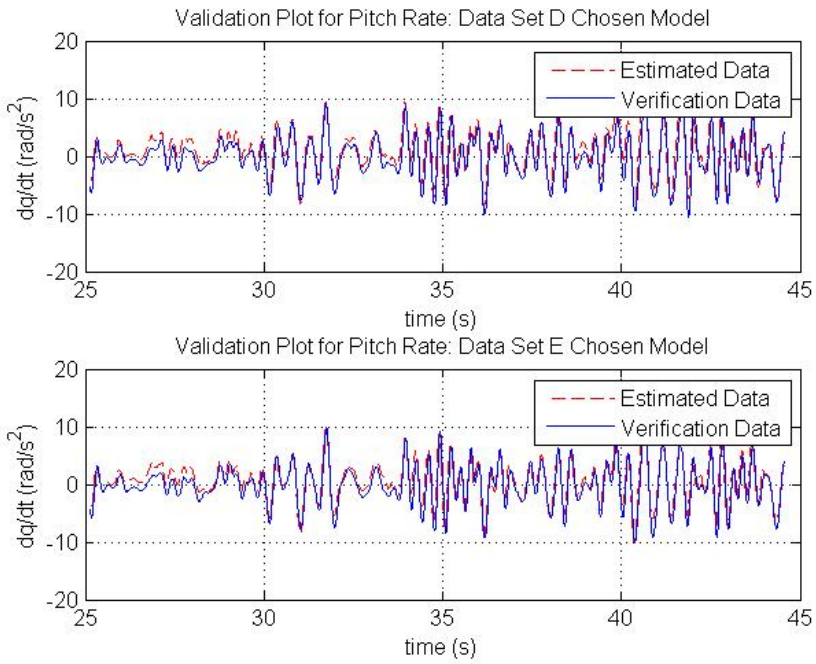

Figure 4.12: Validation plots of chosen model for Flight Dataset D (top), and Flight Dataset E (bottom) for $\dot{q}$

Table 4.11: Sum of squared error of $\dot{r}$ for different models

Model	$\sum e^2$	$\%\Delta \sum e^2$
Basic	1267.4	
u	1283.70	-1.286%
v	1289.10	-1.712%
w	1270.10	-0.213%
p	1236.60	2.430%
q	1292.80	-2.004%

Table 4.12: Parameter values for $\dot{r}$ for selected model

$\frac{C_1^r}{I_{ZZ}}$	0.3554
$\frac{C_2^r}{I_{ZZ}}$	-0.3326
$\frac{C_3^r}{I_{ZZ}}$	0.2981
$\frac{C_4^r}{I_{ZZ}}$	-0.2743
$\frac{I_{XX}-I_{YY}}{I_{ZZ}}$	0.0834
$\frac{C_{dr}}{I_{ZZ}}$	-0.4833

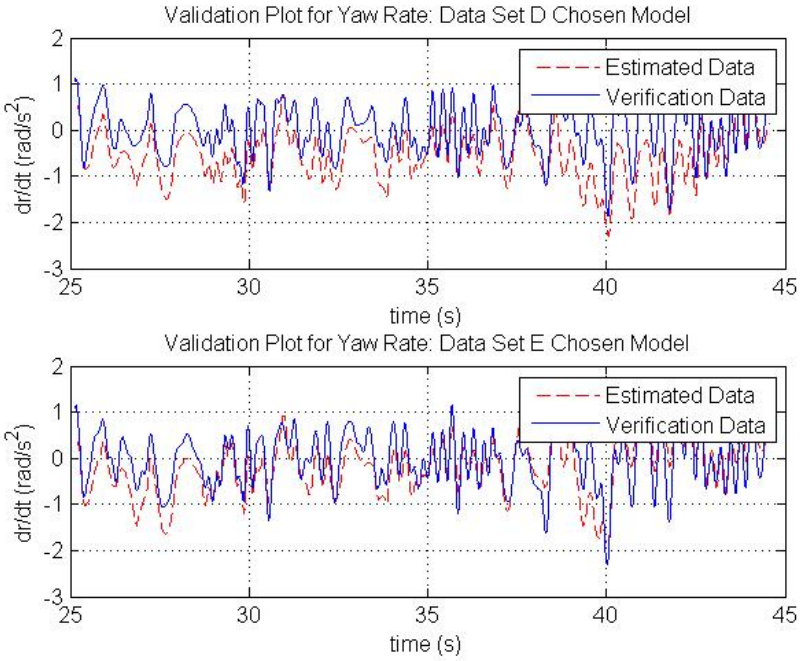

Figure 4.13: Verification plots of chosen model for Flight Dataset D (top), and Flight Dataset E (Bottom) for $\dot{r}$

With the parameters of the model identified, model-based design and simulations are possible. In addition, these experiments show a close coupling between the body 'x' velocity u and pitch rate $\dot{q}$ as well as the body 'y' velocity $\dot{v}$ and roll rate $\dot{p}$, respectively. The addition of the u parameter in $\dot{q}$, and q parameter in $\dot{u}$ each cause a significant improvement in the performance of the $\dot{q}$, and $\dot{u}$ models (with a similar behavior between $\dot{v}$ and $\dot{p}$).

Chapter 5: Control Law Design, Simulation, & Deployment

One of the goals of this research, is to create a model that can be used for both control law design and for simulation. The control laws previously utilized in flight have been tuned by hand. Now that a dynamic model has been found, model-based design is possible. In this chapter, the quadcopter model will be linearized for a classical control law design. Design requirements will be specified. A model-based design will be created to meet those requirements. The control law design will be tested using a simulation model. Then, the control law will be deployed to the actual quadcopter for real-time flight testing. All designs are implemented digitally. Filters are converted from S-Domain to Z-domain using the "matched Z-transform" in MATLAB. Also the $100Hz$ sample frequency is assumed to be fast enough that the controllers gains do not need to be redesigned for the discrete time conversion.

5.1 Control Algorithms

Only classical control techniques will be used for this research. In this section, the control techniques used will be described, including the Proportional-Integral-Derivative (PID) controller, pole-zero cancellation, and pre-filtering. All classical control laws are based on the principle of negative feedback. The control decisions made are based on the tracking error (e) between the desired and measured value of the controlled signal (x). The control law output (Z) is designed as a function of the tracking error, specifically:

$$\begin{aligned} e(t) &= x_{Desired}(t) - x_{Actual}(t) \\ Z(t) &= f(e(t)) \end{aligned} \tag{5.1}$$

(5.2)

PID Controller

In PID controllers, the control law output is based solely on a weighted sum of the error, the integral of the error, and the derivative of the error. Each of these weights is labeled the gain (denoted by K_p for proportional gain,K_i for integral gain,K_d for derivative gain). The mathematical representations of the PID controller in the time domain and S-domain, respectively, are as follows:

$$f\left(e\left(t\right)\right) = K_p e(t) + K_i \int e(t)dt + K_d \dot{e}(t) \tag{5.3}$$

$$f\left(E\left(s\right)\right) = K_p E(s) + K_i \frac{E(s)}{s} + K_d s E(s) \tag{5.4}$$

A large value of the gain (K_p) can help increase the bandwidth (speed up the system response) and reduce steady state error, but it also degrades phase margin and increases overshoot. If the proportional gain is too large, the system may become unstable. The integral component ($K_i \int e(t)dt$) represents the accumulation of error. Its primary purpose is the elimination of steady state error, since any error will cause changes in the integral component, until the error is eliminated. However, this component increases settling time and may introduce a low frequency time constant causing drift. In addition, an integral component suppresses effects of input disturbance and eliminates the steady state error to a step input, further details may be found in Appendix B. The derivative component $K_d \dot{e}(t)$ adds damping to the system, hence improving phase margin, decreasing overshoot, and overall making the system more stable [18]. Implementing derivatives in real systems has many issues. First, derivatives will amplify noise of the measured signal, which can cause vibrations in the control output and may drive the system to become unstable. This effect can be suppressed by using a velocity estimator. This takes the form of a 2^{nd}-order bandpass filter, acting as a derivative at low frequencies, but attenuating higher frequencies. For example, the following S-domain band-pass filter, with a natural frequency (ω_n), can be used as a velocity estimator when the frequency of the signal is less than ω_n:

$$sE(s) \approx \frac{\omega_n^2 s}{s^2 + 2\omega_n s + \omega_n^2} E(s) \tag{5.5}$$

$$\tag{5.6}$$

A second issue with the derivative component is that a rapid change in the command signal may cause a controller output that is too large. In addition, it introduces a finite zero in closed-loop transfer function. In situations where this is undesirable, velocity feedback is used. Instead of using rate of change of the error,

velocity feedback only uses the rate of change of the measured signal. Therefore, it will produce output to oppose high rates of change [18]. If the rate of change of the measured signal is unavailable from sensors, or if the measured signal is undesirable due to noise or bias, a velocity estimator is used to approximate the rate of change of the signal. Velocity feedback, via a velocity estimator, is the form of derivative control used in this research. Specifically, the form of the PID controller takes the following form.

$$f\left(E\left(s\right), X_{Actual}(s)\right) = K_p E(s) + K_i \frac{E(s)}{s} - K_d \frac{\omega_n^2 s}{s^2 + 2\omega_n s + \omega_n^2} X_{Actual}(s) \tag{5.7}$$

$$\tag{5.8}$$

Lead/Lag Controller

When designing control laws, sometimes the undesirable location of a particular pole or zero may hamper the performance of a control system. Particularly, when the pole or zero is at low frequencies. A low frequency pole will create a slow time constant, which will degrade settling time. A low frequency zero will cause a very high steady state error. In these situations, a lead or lag controller is used to reduce the undesirable effect. A lead or a lag controller, in the following form, is used to perform the pole-zero mitigation[18]:

$$G_c(s) = \frac{\frac{s}{z_1} + 1}{\frac{s}{p_1} + 1} \tag{5.9}$$

$$\tag{5.10}$$

If a zero is desired to be attenuated, a pole p_1 is set to the same location as the zero, and a new zero z_1 is set to a higher frequency where it will be less problematic and vice-versa. A practical consideration is that there may be some error between the pole/zero locations in the model and the actual dynamic system. This error degrades the effect of the pole-zero cancellation, since the pole/zero will not be exactly canceled. However, so long as this error is not too large, it still will improve performance. Another consideration, particularly with zero cancellation, is the effect of this controller on the stability of the closed-loop system, as lag controllers tent to degrade phase margin and stability. Thus, the difference between the old and new locations of the pole or zero must be significant enough to have the desired effect, which may be limited by practical constraints.

Prefilters

While designing a control system, it may be determined that there are frequencies at which it may be advantageous to suppress the command signal. This could be due to an a undesirable resonant peak in the closed-loop response, or a desire to limit the input bandwidth, etc. To accomplish this, a low-pass filter, notch

filter, or rate limiter is often applied to the command signal before the error is calculated.

5.2 Linearization

Classical control law designs require a linear model. The model found previously is a non-linear model. It must be linearized before use in classical control design. To linearize the model, an equilibrium point will be found to linearize about. The equilibrium point chosen for this research is stationary hover. In addition, the inputs to the linearized system will be the angular acceleration commands (τ_ϕ^*, τ_θ^*, and τ_ψ^*), and thrust command (T_z). Since the model has the velocities of each motor as inputs to the model, the command thrust and angular accelerations must be mapped to motor velocities using the estimated mapping matrix K which converts each motor's velocity into the resultant thrust and angular accelerations. Specifically, the inverse of K is used to map commanded thrust and angular acceleration to motor velocities, as shown below:

$$
\begin{aligned}
K &= \begin{bmatrix} \beta C_T & \beta C_T & \beta C_T & \beta C_T \\ \sqrt{2}l\beta\frac{C_1^p}{I_{XX}} & -\sqrt{2}l\beta\frac{C_2^p}{I_{XX}} & -\sqrt{2}l\beta\frac{C_3^p}{I_{XX}} & \sqrt{2}l\beta\frac{C_4^p}{I_{XX}} \\ \sqrt{2}l\beta\frac{C_1^q}{I_{YY}} & \sqrt{2}l\beta\frac{C_2^q}{I_{YY}} & -\sqrt{2}l\beta\frac{C_3^q}{I_{YY}} & -\sqrt{2}l\beta\frac{C_4^q}{I_{YY}} \\ \beta R_R\frac{C_1^r}{I_{ZZ}} & \beta R_R\frac{C_2^r}{I_{ZZ}} & \beta R_R\frac{C_3^r}{I_{ZZ}} & \beta R_R\frac{C_4^r}{I_{ZZ}} \end{bmatrix} \\
\begin{bmatrix} T_Z \\ \tau_\phi^* \\ \tau_\theta^* \\ \tau_\psi^* \end{bmatrix} &= K \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^4 \\ \omega_4^2 \end{bmatrix} \\
\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^4 \\ \omega_4^2 \end{bmatrix} &= K^{-1} \begin{bmatrix} T_Z \\ \tau_\phi^* \\ \tau_\theta^* \\ \tau_\psi^* \end{bmatrix} \\
K^{-1} &= \begin{bmatrix} 31389 & 4243 & 4067 & 18755 \\ 33789 & -3788 & 4022 & -17395 \\ 33715 & -5054 & -4065 & 19532 \\ 36340 & 4599 & -4025 & -20893 \end{bmatrix}
\end{aligned}
\tag{5.11}
$$

The motor velocities (ω_i) are converted to the PWM value (δ_i). This series of conversions is shown

below:

$$\begin{aligned} \omega_i &= 0.7850\delta_i + 44.6505V_{battery} - 1205.1 \\ \delta_i &= 1.2739\omega_i - 56.8796V_{battery} + 1535.2 \end{aligned} \tag{5.12}$$

where $V_{battery}$ is the current battery voltage, and δ_i is the PWM signal sent to each motor for $i = 1, 2, 3, 4$. With this mapping completed, MATLAB's FMINCON function is used to find the hover equilibrium point, using the dynamic model found in Chapter 4.4. The linear model uses the angular acceleration and thrusts as inputs (u). The equilibrium point is obtained by minimizing a cost function J, given below:

$$\begin{aligned} \dot{Z} &= f(Z, u) \\ \dot{Z} &= \begin{bmatrix} x_e & y_e & z_e & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \\ u &= \begin{bmatrix} T_Z & \tau_\phi^* & \tau_\theta^* & \tau_\psi^* \end{bmatrix}^T \\ J &= \dot{Z}^T \dot{Z} \end{aligned} \tag{5.13}$$

The equilibrium values of this system are shown below:

$$\begin{aligned} Z_N &= \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ u_N &= \begin{bmatrix} 11.9093 & 0 & 0 & 0 \end{bmatrix}^T \end{aligned} \tag{5.14}$$

where Z_N is the equilibrium system states, and u_N is the equilibrium inputs.

These nominal values are then used to linearize the non-linear model, via small perturbations. Specifically the linearized model is given by:

$$\begin{aligned} \Delta\dot{Z} &= \mathbf{A}\Delta Z + \mathbf{B}\Delta u \\ \Delta Y_{sys} &= \mathbf{C}\Delta Z + \mathbf{D}\Delta u \end{aligned} \tag{5.15}$$

where ΔZ is a vector of the deviation of Z from the equilibrium values, Δu is a vector of the deviation of u from the equilibrium values, Y_{sys} is the system output, and $\mathbf{A} \in \mathbb{R}^{12\times12}$, $\mathbf{B} \in \mathbb{R}^{12\times4}$, $\mathbf{C} \in \mathbb{R}^{12\times12}$, $\mathbf{D} \in \mathbb{R}^{12\times4}$ are matrices of the linear state space model. Note that the $\mathbf{C}$ is assumed to be an identity matrix, and $\mathbf{D}$ is assumed to be a zero matrix. The $\mathbf{A}$ and $\mathbf{B}$ matrices obtained are shown below:

$$
\mathbf{A} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.2076 & 0 & 0 & 0 & -9.8080 & 0 & 0 & -0.1414 & 0 \\
0 & 0 & 0 & 0 & -0.2211 & 0 & 9.8080 & 0 & 0 & 0.1340 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.2149 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & -3.7800 & 0 & 0 & 0 & 0 & 1.2151 & 0.0327 & 0 \\
0 & 0 & 0 & 2.7540 & 0 & 0 & 0 & 0 & 0 & 0.2170 & 1.9710 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.4833
\end{bmatrix}
$$

$$
\mathbf{B} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-0.8237 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \tag{5.16}
$$

To perform classical control design, the state space model is converted to transfer functions for each channel to be controlled by the respective input. The controlled dynamics are roll $\left(\frac{\Phi(s)}{\tau_{\phi}^{*}(s)}\right)$, pitch $\left(\frac{\Theta(s)}{\tau_{\theta}^{*}(s)}\right)$, yaw $\left(\frac{\Psi(s)}{\tau_{\psi}^{*}(s)}\right)$,

altitude $\left(\frac{z_e(s)}{T_Z(s)}\right)$, X position $\left(\frac{x_e(s)}{\tau_\theta^*(s)}\right)$, and Y position $\left(\frac{y_e(s)}{\tau_\phi^*(s)}\right)$. The corresponding transfer functions are:

$$\frac{y_e(s)}{\tau_\phi^*(s)} = \frac{0.134s + 9.808}{s^4 - 0.9888s^3 + 0.2274s^2 + 37.03s} \tag{5.17}$$

$$\frac{x_e(s)}{\tau_\theta^*(s)} = \frac{-0.1414s - 9.808}{s^4 - 1.769s^3 - 0.01226s^2 + 27.05s} \tag{5.18}$$

$$\frac{z_e(s)}{T_Z(s)} = \frac{0.8237}{s^2 + 0.2149s} \tag{5.19}$$

$$\frac{\Phi(s)}{\tau_\phi^*(s)} = \frac{s + 0.2149}{s^3 - 0.9888s^2 + 0.2274s + 37.03} \tag{5.20}$$

$$\frac{\Theta(s)}{\tau_\theta^*(s)} = \frac{s + 0.2076}{s^3 - 1.769s^2 - 0.01226s + 27.05} \tag{5.21}$$

$$\frac{\Psi(s)}{\tau_\psi^*(s)} = \frac{1}{s^2 + 0.4833s} \tag{5.22}$$

These transfer functions will be used to design control laws for the quadcopter.

5.3 Inner Loop Control Laws Design

The inner loop control laws must be designed before the outer loop, as the dynamics of the outer loop depends on the inner loop. Due to their similar physical characteristics and transfer functions, the design of the roll and pitch channels follow the same process. These are a little more complex than a standard PID design, due to a low frequency zero. Altitude and yaw may both follow a standard PID design procedure. All designs are tested using the non-linear simulation model, before they are deployed.

5.3.1 Roll & Pitch Control Law Design

Roll and pitch are inherently unstable, since they possess poles with positive real components. Therefore, the controller must be designed to ensure the closed-loop system becomes stable. In addition, system noise and delay limit the obtainable bandwidth of the system. Roll and pitch are designed to meet the following specifications:

- 45° phase margin
- $6\frac{rad}{s}$ gain crossover frequency

A 45° phase margin ensures stability and a reasonable transient response. A gain crossover frequency should be no more than $6\frac{rad}{s}$, because noise and transmission delays compromise performance at higher frequencies. For example, it was found through experimentation that high velocity feedback gains cause the system to

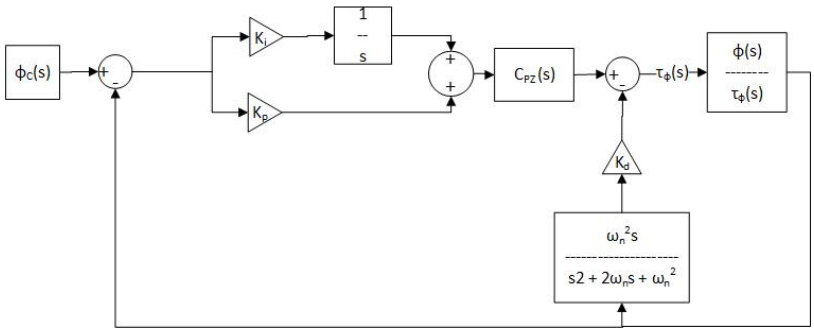

Figure 5.1: Roll & pitch controller structure

vibrate in flight. Roll and pitch control follow the same procedure so they will be presented in parallel. The controller for the roll and pitch will ultimately take the structure shown in Figure 5.1, including velocity feed back, a proportional and integral control, a pole-zero cancellation, and a pre-filter. The design procedure for roll (Φ) control law is described below.

1. The effect of motor time constant found in Section 4.2.5 is taken into account. This will result in the transfer function $G(s)$ below:

$$G(s) = \frac{\Phi(s)}{\tau_\phi^*(s)} \frac{1}{\frac{s}{21.5}+1} \tag{5.23}$$

   where $\frac{\Phi(s)}{\tau_\phi^*(s)}$ is the transfer function (5.20).

2. It is observed that roll and pitch have unstable poles. Thus, the velocity feedback gain must be used to help stabilize the system. The quadcopter does not use true angular velocity measurement, but rather an estimation based on the VICON Euler angles. Specifically, a velocity estimator with a corner frequency at 75 radians per second is applied and then converted to body rates (at nominal conditions the Body and Euler rates are the same). The velocity estimator's transfer function is $\frac{5625s}{s^2+75s+5625}$. The velocity feedback gain is designed using the root locus technique. The transfer function of the closed velocity feedback loop $G_{VFB}(s)$ shown below, where $G(s)$ is the open loop transfer function (5.23), is used to find the velocity feedback gain.

$$G_{VFB}(s) = \frac{G(s)}{1+K_d G(s) \frac{5625s}{s^2+75s+5625}} \tag{5.24}$$

   To design K_d, the characteristic equation of (5.24) is used to obtain the root loci shown in Figures 5.2

and 5.3 for roll and pitch, respectively. This characteristic equation is:

$$1 + K_d G(s) * \frac{5625s}{s^2 + 75s + 5625} = 0 \tag{5.25}$$

Based on the root loci as well as observations concerning the practical limits of K_d (it was found that a high value K_d tends to result in the system vibrating in flight), a K_d value of 10 was chosen.

3. Roll and Pitch dynamics possess a very low frequency zero at $-0.2149\frac{rad}{s}$ and $-0.2076\frac{rad}{s}$, respectively. This zero limits the settling time performance as well as significantly degrades steady state performance. A lag controller technique was used to move this zero to a higher frequency. A zero with higher frequency improves steady state performance, but also introduces a stronger lag effect, which reduces stability. As a compromise between these factors, the new zero is placed at $-0.6\frac{rad}{s}$ for both roll and pitch. Specifically the following transfer function ($C_{PZ}(s)$) is used for pole-zero cancellation:

$$C_{PZ}(s) = \frac{\frac{s}{0.6} + 1}{\frac{s}{0.2149} + 1} \tag{5.26}$$

While the location of the zero in the actual system is not exactly known, phase margin of the controlled system does not significantly change when the zero of the system is changed (keeping the controller pole constant). Details on the effects of variation of the plant zero may be seen in Appendix A.

4. The proportional gain (K_p) is designed via root locus. The characteristic equation of the closed-loop transfer function is used to find K_p. Specifically, the characteristic equation is shown below:

$$1 + K_p C_{PZ}(s) G_{VFB}(s) = 0 \tag{5.27}$$

where $C_{PZ}(s)$ is the pole-zero cancellation filter from (5.26), and $G_{VFB}(s)$ is the system transfer function with velocity feedback from (5.24). The root loci is shown in Figures 5.4 and 5.5 for roll and pitch, respectively. The Proportional Gain decreases the phase margin and overshoot, but improves system response speed. The proportional gain was designed for less than 35% overshoot. A K_p value of 100 is found to satisfy this specification for both roll and pitch.

5. The integral gain (K_i) is found last. The closed-loop transfer function, which includes velocity feedback, pole-zero cancellation, proportional gain, and integral gain, is found. The following characteristic

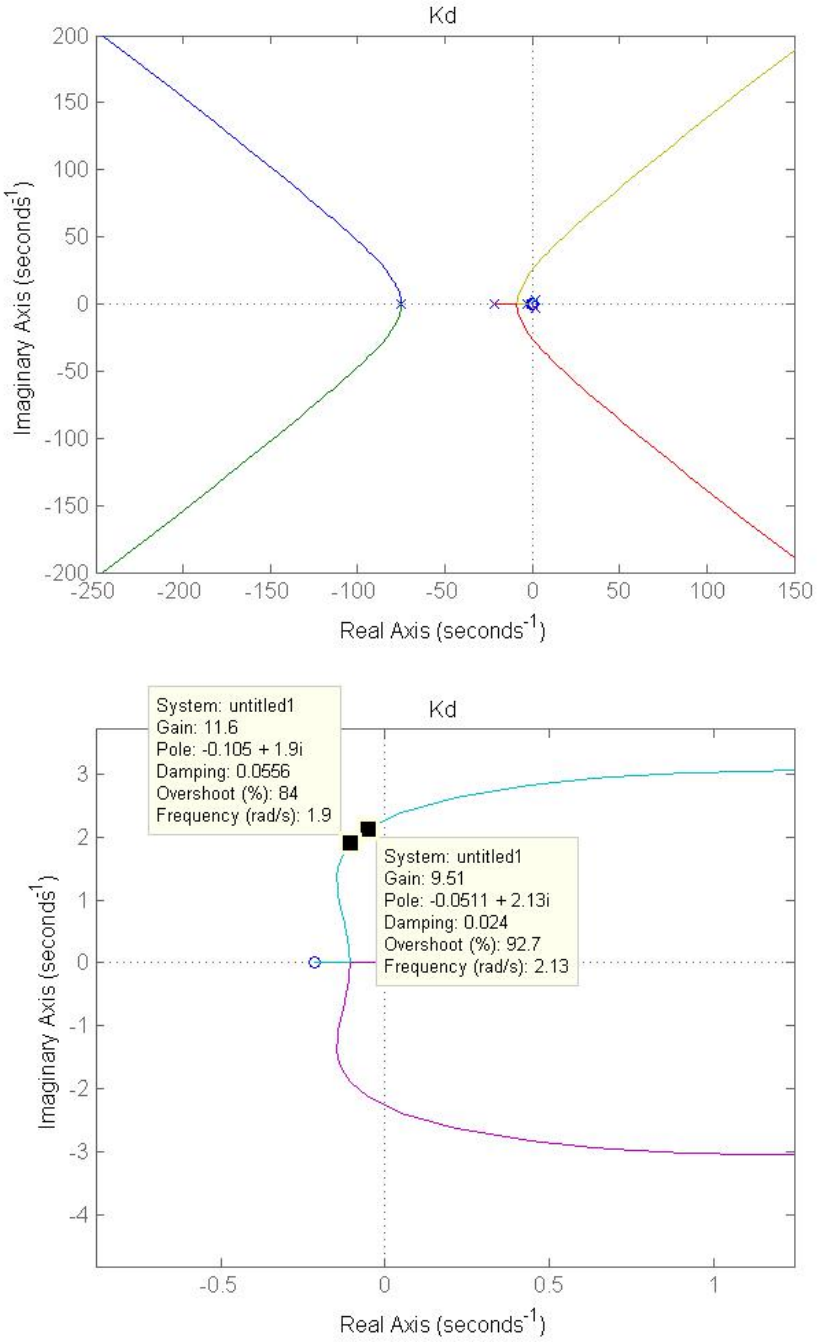

Figure 5.2: K_d design root locus for roll

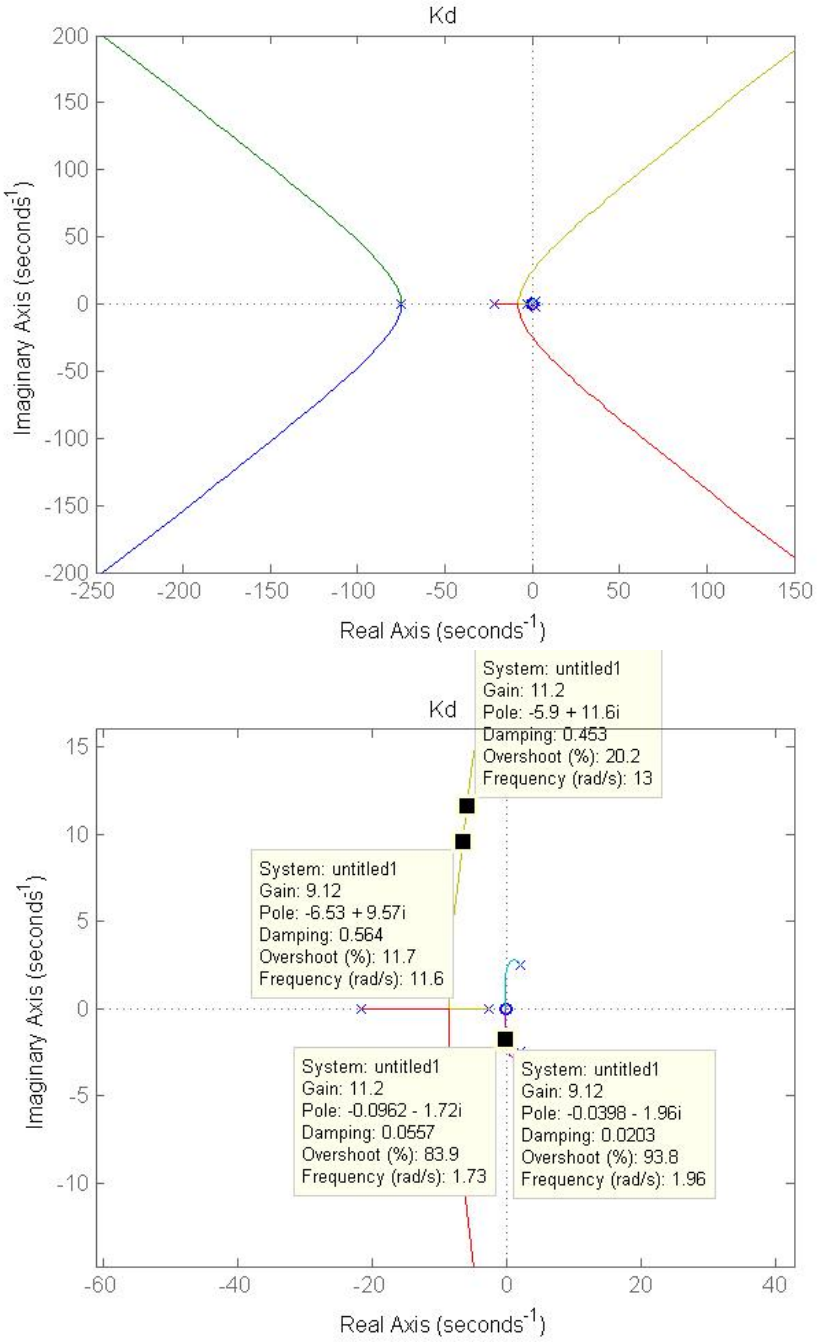

Figure 5.3: K_d design root locus for pitch

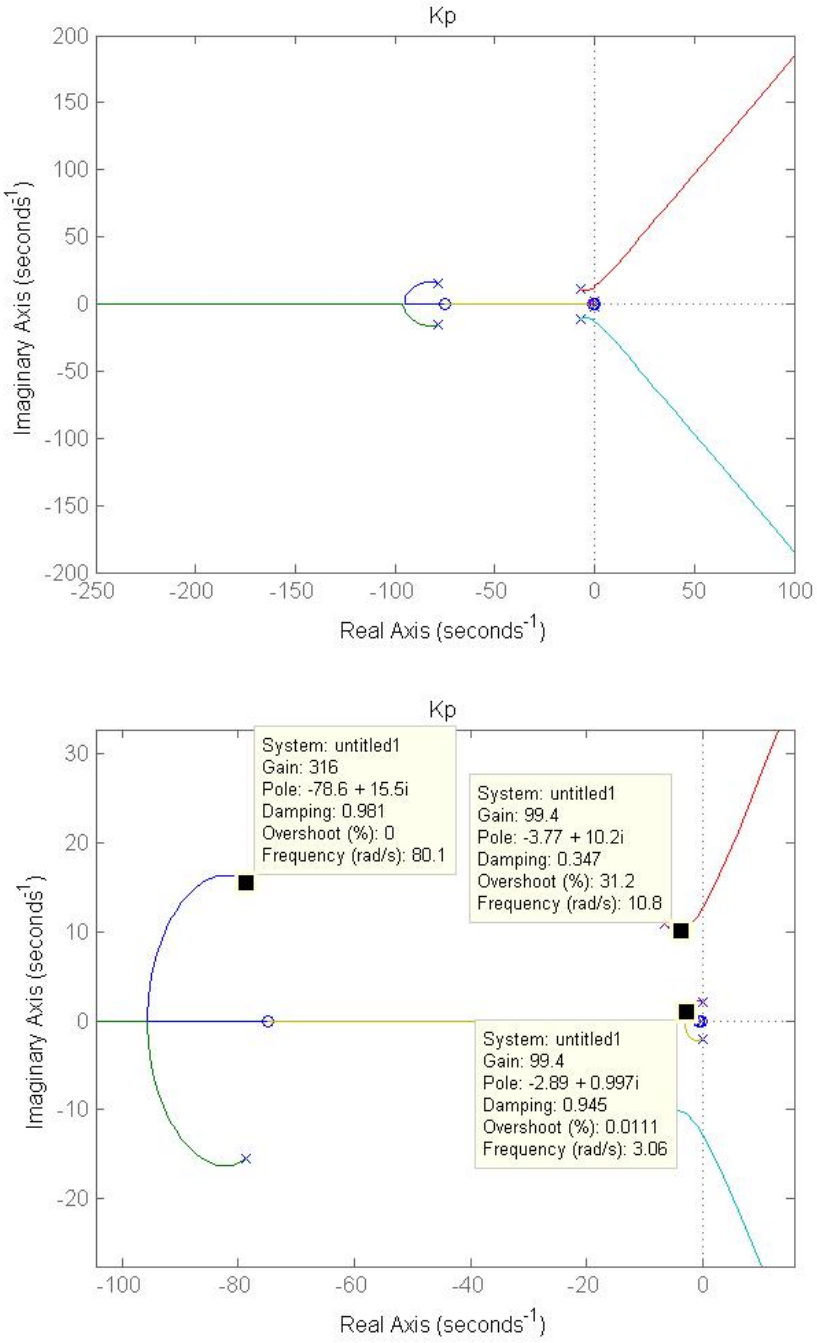

Figure 5.4: K_p design root locus for roll

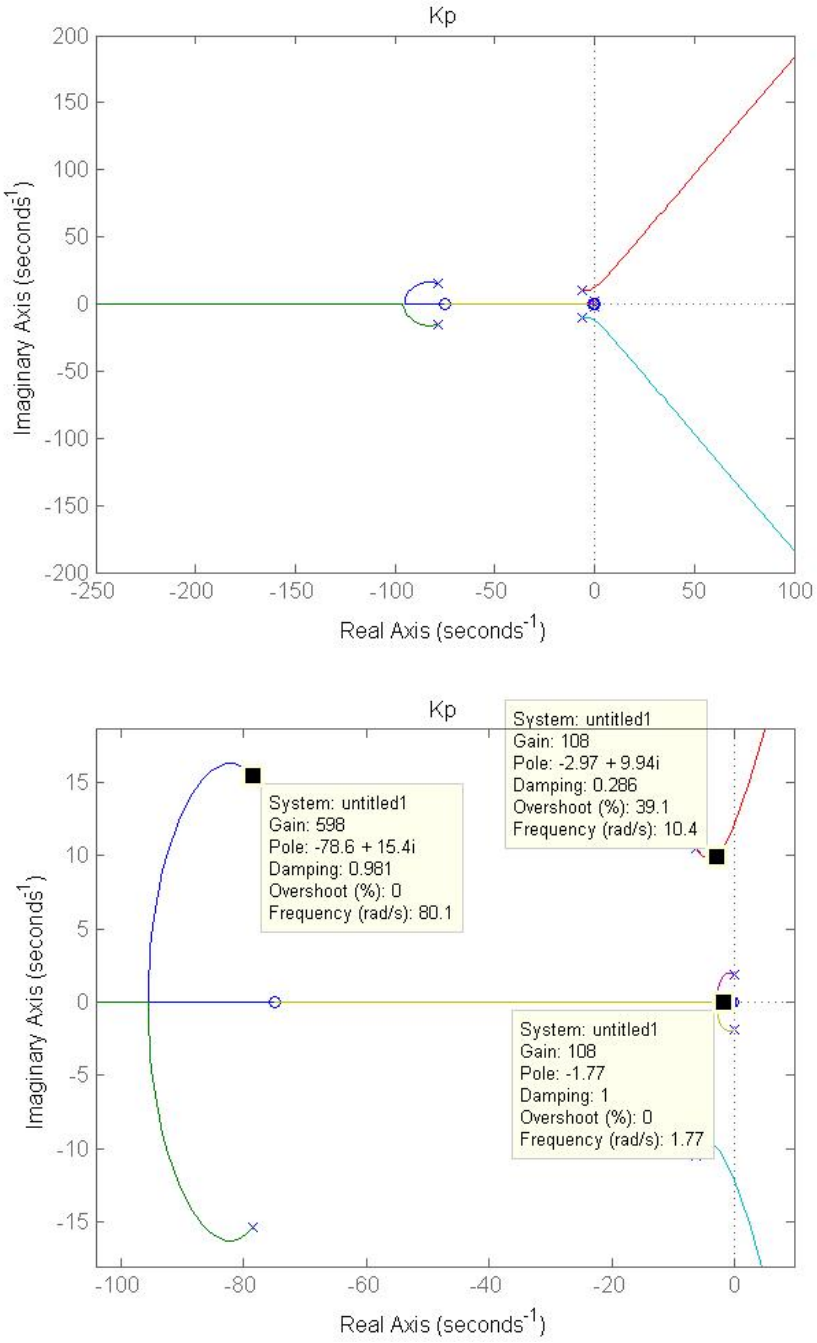

Figure 5.5: K_p design root locus for pitch

equation is manipulated to the form below to find a suitable K_i:

$$1 + K_i \frac{N}{K_p N s + D s} = 0 \tag{5.28}$$

where N and D are the numerator and denominator of the product of $C_{PZ}(s)$ defined in (5.26) and $G_{VFB}(s)$ defined in (5.24), respectively, and K_p is the proportional gain found previously. The design objective is to keep the overshoot less than 35%. A K_i value of 100 is found to satisfy this specification for both roll and pitch. The root loci used are shown in Figures 5.6 and 5.7 for roll and pitch, respectively.

6. The reasons for keeping the overshoot less than 35% are to try to ensure the following: (1) the phase margin remains within $45°$; (2) the slowest poles are less than $5\frac{rad}{s}$. The Bode plots in Figure 5.8 show that the phase margin and crossover frequency requirements are both met. However, this system has resonant peaks which may cause a large overshoot. This need to be suppressed via a pre-filter. The closed-loop Bode plot (shown in Figure 5.9) shows the resonant peak frequency is at around $4.5\frac{rad}{s}$. As some of this resonance is useful for a quicker response, the entire peak isn't fully suppressed. Specifically, a first-order low pass filter ($G_{PFrp}(s)$) with a corner frequency of $5\frac{rad}{s}$ is used to help suppress the peak. Since this pre-filter is not within the feedback loop, it will not affect reactions to disturbances.

Based on the above design, the final controller is a PID controller with a pole-zero canceler and velocity feedback. The step response to a filtered command input are shown in Figure 5.10. The controller's final structure is shown in Figure 5.1, and the parameters of the controller is given in Table 5.1. The controller is implemented digitally by applying the matched Z-transform to the velocity estimator, pre-filter, and lag controller using a $100Hz$ system sample frequency. The $100Hz$ sample frequency is assumed to be fast enough relative to the speed of the system that the gains do not need to be adjusted, and a simple and resetable digital integrator is used for the integral component.

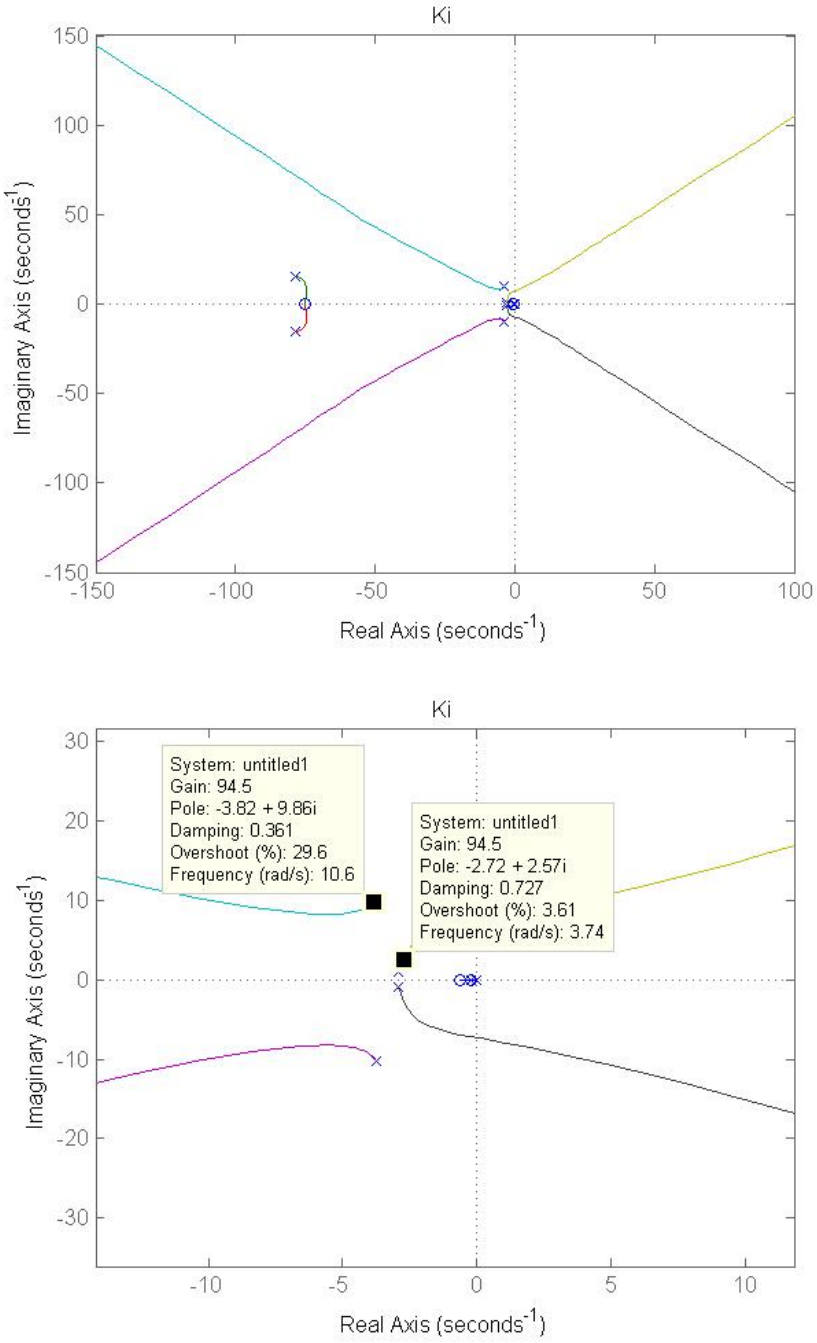

Figure 5.6: K_i design root locus for roll

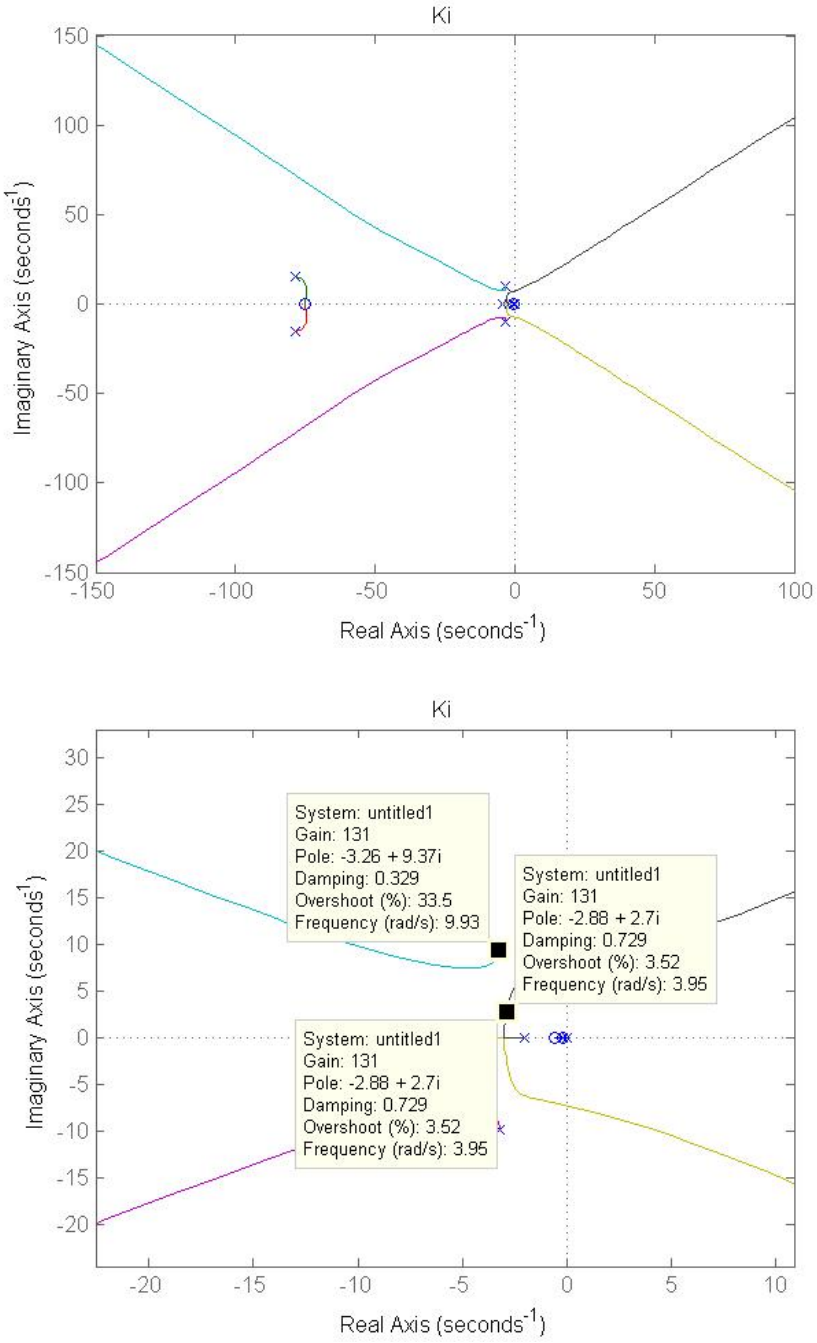

Figure 5.7: K_i design root locus for pitch

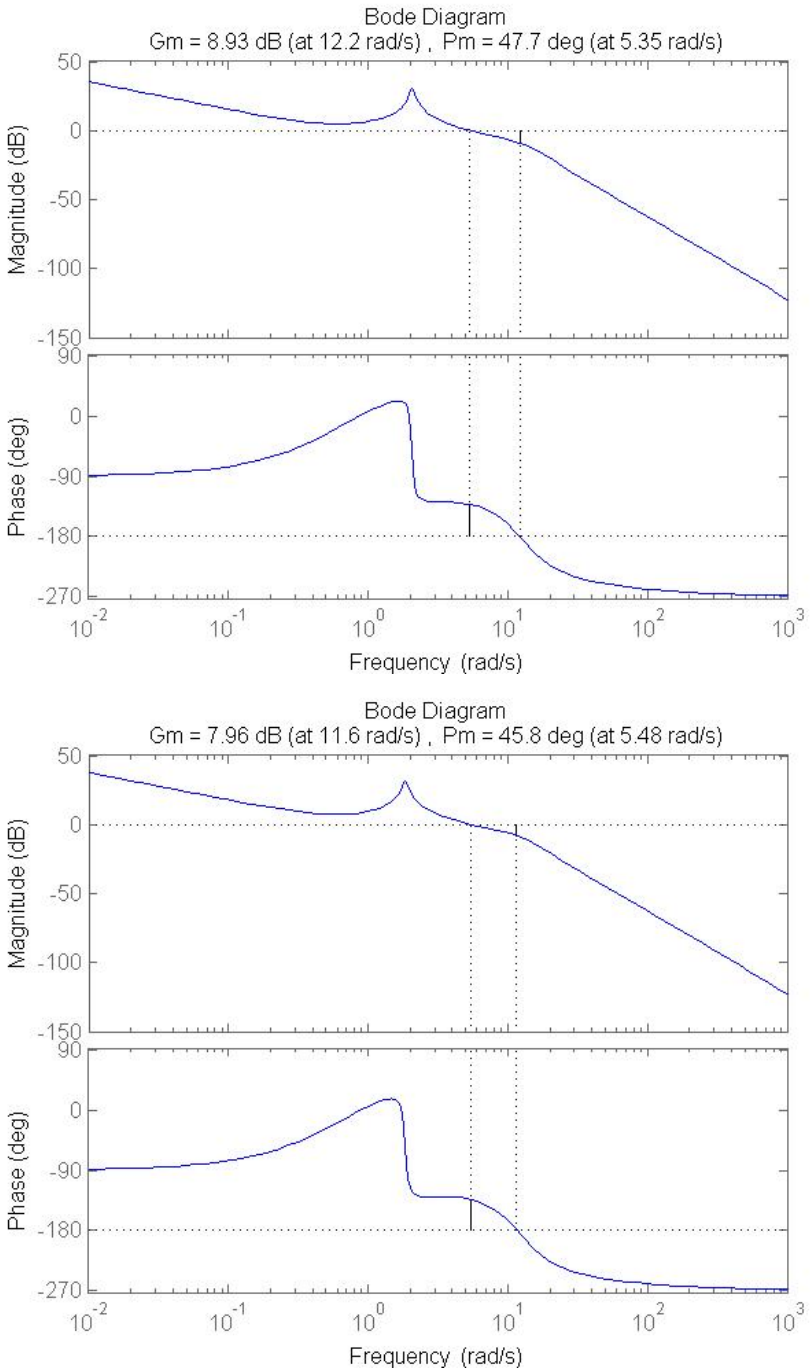

Figure 5.8: Bode and margins of roll(top) & pitch(bottom)

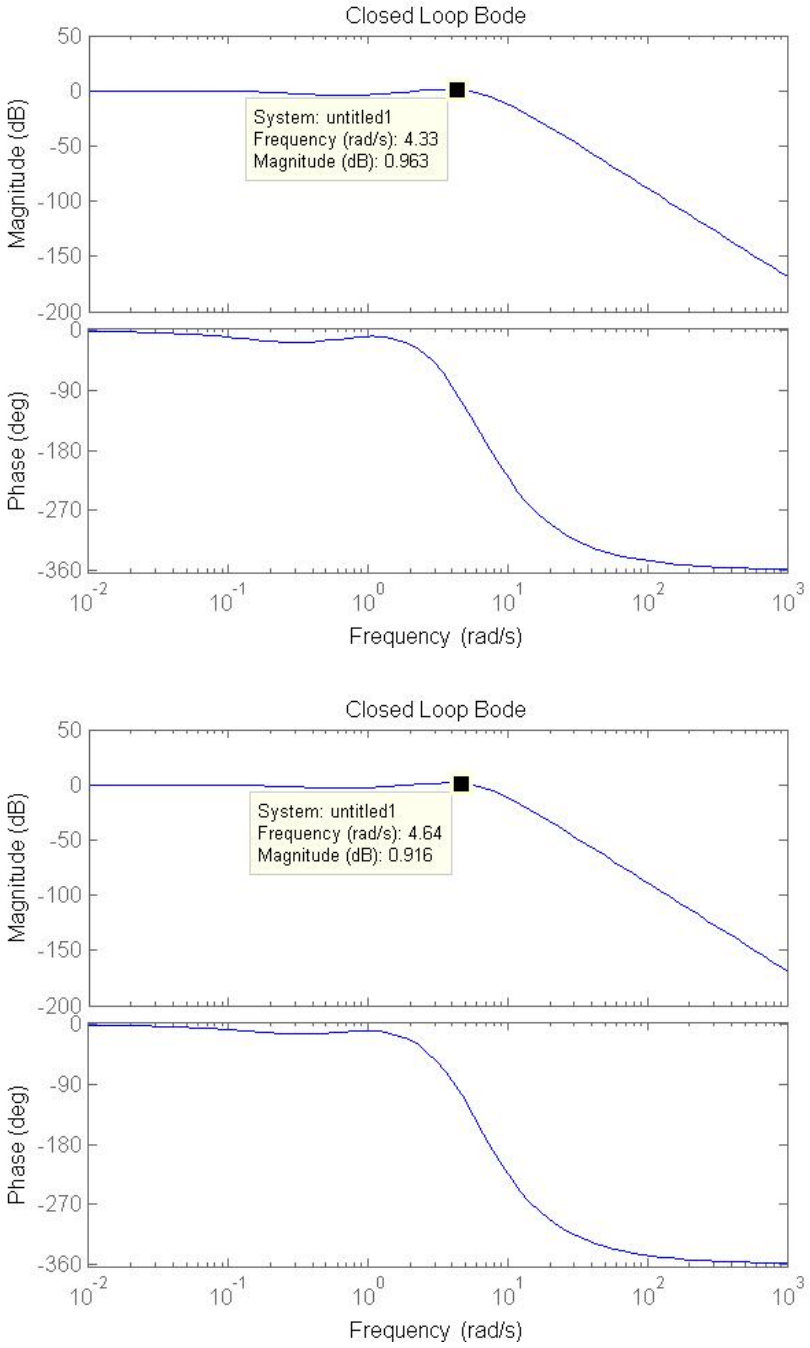

Figure 5.9: Bode and margins of roll(top) & pitch(bottom)

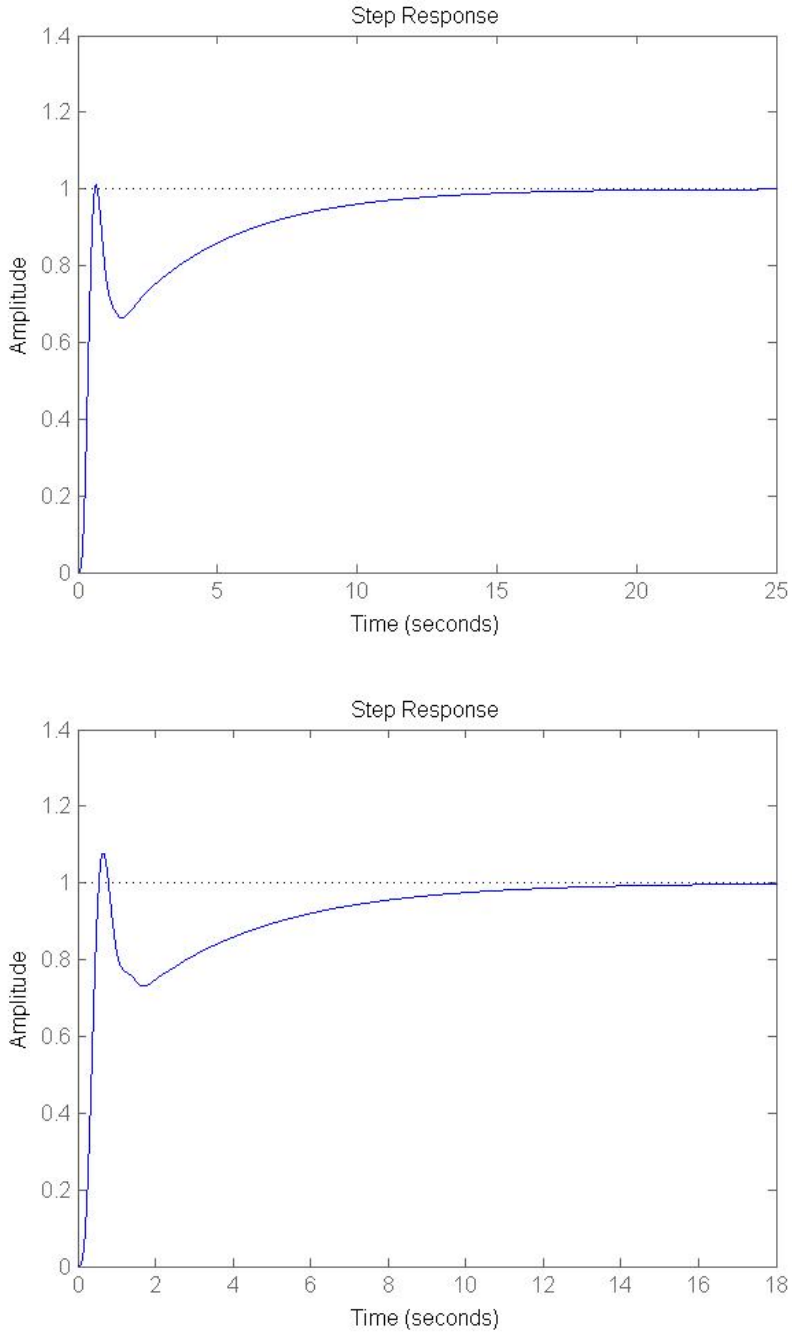

Figure 5.10: Step response of roll(top) & pitch(bottom)

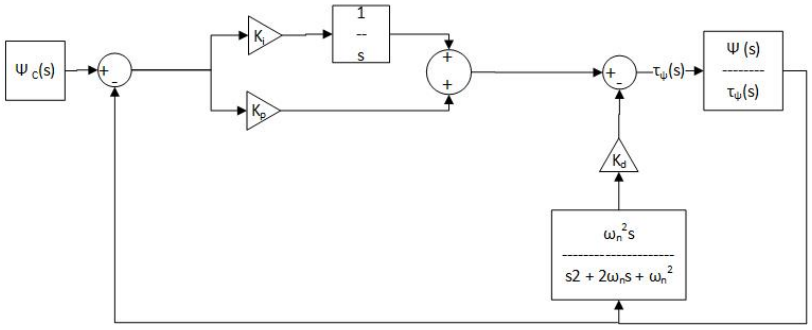

Figure 5.11: Yaw controller structure

Table 5.1: Gain values and filters for roll and pitch Controllers

	ϕ	θ
K_p	100	100
K_i	100	100
K_d	10	10
$C_{PZ}(s)$	$\frac{\frac{s}{0.6}+1}{\frac{s}{0.2149}+1}$	$\frac{\frac{s}{0.6}+1}{\frac{s}{0.2076}+1}$
$G_{PFrp}(s)$	$\frac{1}{\frac{s}{5}+1}$	$\frac{1}{\frac{s}{5}+1}$

5.3.2 Altitude & Yaw Control Law Design

Both altitude and yaw are Type 1 second-order systems. Therefore conventional PID design will be used (although they take into account the motor time constant and have velocity feedback and a pre-filter). The controller structure is shown in Figure 5.11. Both altitude and yaw are slower than roll and pitch, therefore, while the filter of the velocity estimator for the velocity feedback is used in the controller, the effects will be neglected and a pure derivative will be used while designing the control law. Altitude and yaw have the following design specifications, respectively,

Altitude design specifications:

- $\leq 5\%$ overshoot to command
- $\leq 1.5s$ rise time

Yaw design specifications:

- $\leq 5\%$ overshoot to command
- $\leq 1s$ rise time

The design for altitude and yaw controllers are performed using root loci. For each root loci the closed-loop transfer function is found, and the characteristic equation is manipulated, such that it becomes $1 + KG_K(s) = 0$, where K is the gain to be found. In this arrangement, $G_K(s)$ is the transfer function to be used for obtaining the root locus. For the design procedure, the transfer functions $G(s)$ represents $\frac{z_e(s)}{T_Z(s)}$ for altitude from (5.19) and $\frac{\Psi(s)}{\tau_\psi^*(s)}$ for yaw from (5.22) are used. The detailed design procedure is given below.

1. The velocity feedback gain (K_d) is designed to make the system more stable, by moving the slowest poles further into the left half plane. K_d is designed via a root locus using the characteristic equation below:

$$1 + K_d s G(s) \quad = \quad 0 \tag{5.29}$$

   where for altitude $G(s) = \frac{z_e(s)}{T_Z(s)}$ given by (5.19), and for yaw $G(s) = \frac{\Psi(s)}{\tau_\psi^*(s)}$ given by (5.22). The root loci of K_d design for altitude and yaw are shown in Figure 5.12. A K_d of 8 and 5 are used for altitude and yaw, respectively.

2. The proportional gain (K_p) is designed to increase the speed of the system. The characteristic equation for the root locus design of K_p is shown below:

$$1 + K_p \frac{G(s)}{1 + K_d s G(s)} \quad = \quad 0 \tag{5.30}$$

   where K_d is the previously designed velocity feedback gain. The gain K_p is designed to increase the bandwidth as much as possible, while keeping overshoot within overshoot specifications. Based on the root locus shown in 5.13, a gain K_p of 18 and 15 are used for altitude and yaw, respectively, to meet specifications.

3. As it is a Type 1 system, the steady state error to step input is zero. An integral controller is added so that the steady state response to a step disturbance (disturbances for this controller include interference from other control loops or to compensate for gravity not taken into account in the nominal solution, further details may be seen in Appendix B). The integral gain is designed as a trade-off, between a small overshoot and an acceptable settling time. The gain K_i is designed using a root locus which uses

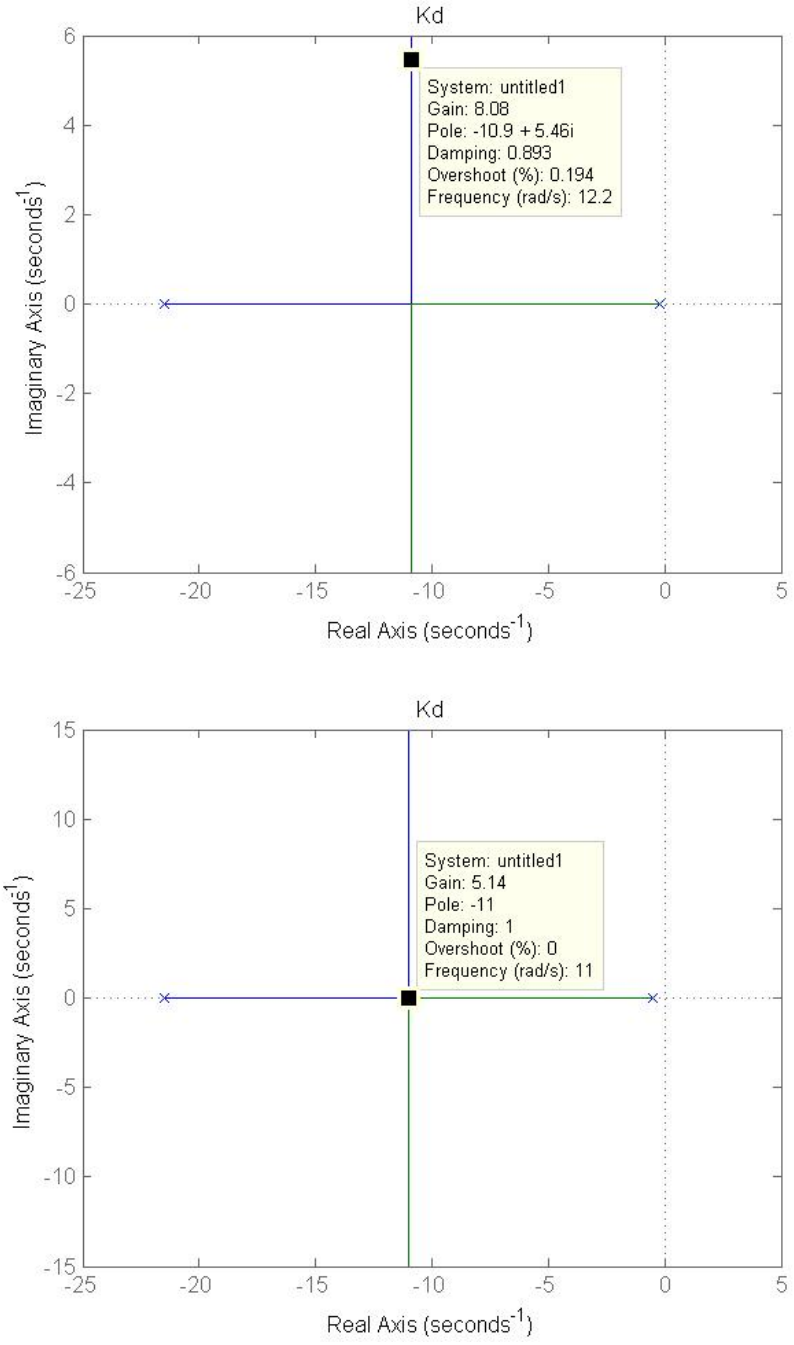

Figure 5.12: Root locus to design K_d for altitude(top) & yaw(bottom)

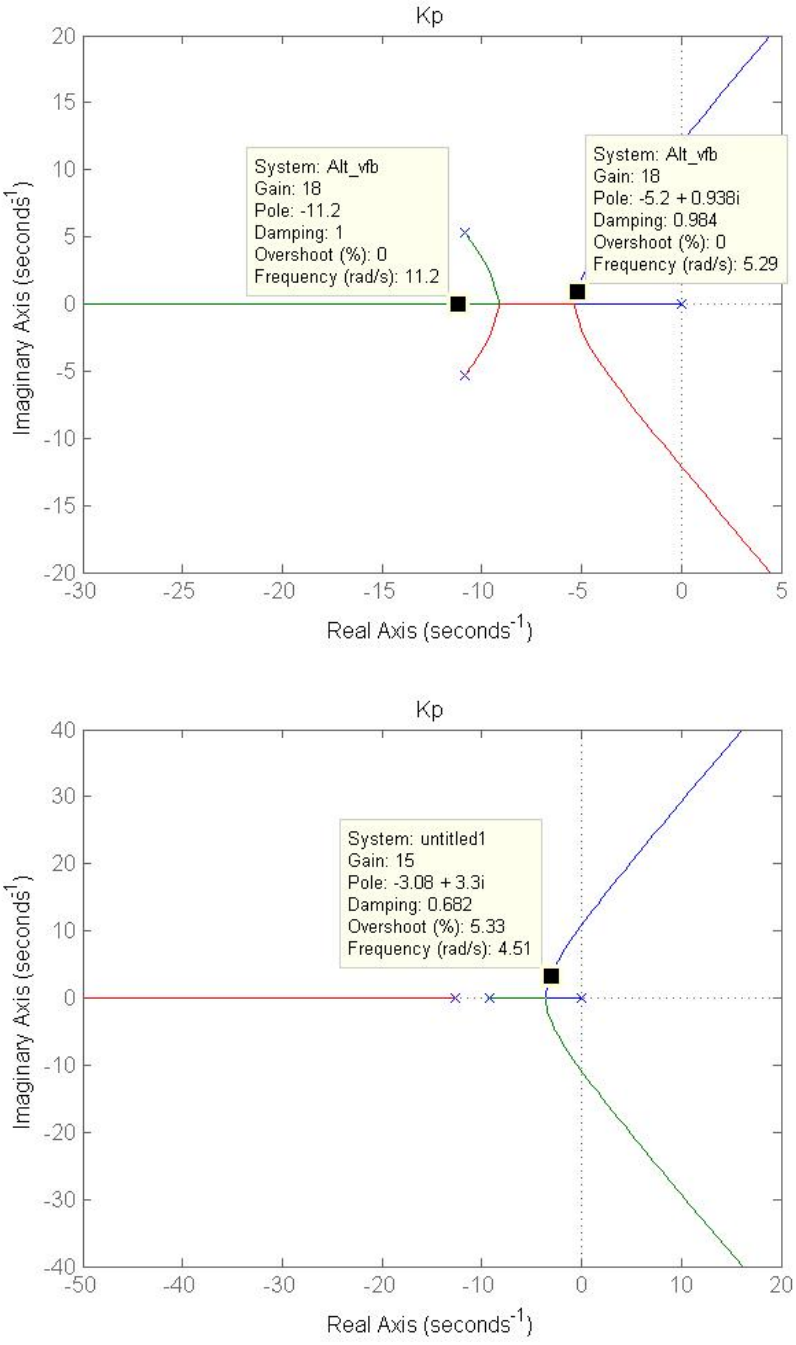

Figure 5.13: Root locus to design K_p for altitude(top) & yaw(bottom)

the following characteristic equation:

$$1 + K_i \frac{N}{(N+D)s} = 0 \tag{5.31}$$

where N and D are the numerator and denominator of the result of $K_p \frac{G(s)}{1+K_d s G(s)}$, respectively. Based on the root locus seen in Figure 5.14, a K_i value of 1 is used for both altitude and yaw to meet specifications.

4. Pre-filters are used for both altitude and yaw to prevent rapid changes. First-order low-pass filters are used as the pre-filters, with corners frequencies of $2\frac{rad}{s}$ for altitude and $4\frac{rad}{s}$ for yaw. The step responses with pre-filtering are shown in Figure 5.15.

Based on the above design decisions, the gains for the altitude and yaw are summarized in Table 5.2.

Table 5.2: Gain values for altitude and yaw controllers

	z_e	ψ
K_p	18	15
K_i	1	1
K_d	8	5
$G_{PFay}(s)$	$\frac{1}{\frac{s}{2}+1}$	$\frac{1}{\frac{s}{4}+1}$

where $G_{PFay}(s)$ is the prefilter. The step responses of the designs are shown in Figure 5.15. as can be seen, both altitude and yaw response meet the specifications. Specifically, the altitude response has an overshoot of 2.15% and a rise time of $1.24s$ and the yaw response has an overshoot of 3.54% and a rise time of $0.665s$.

5.4 Outer Loop Control Laws Design

With the inner loop control law designed, it is now possible to design the outer loop control law for the X and Y position control. The successive loop closure and block diagram of the outer loop design for the Y position control is shown in Figure 5.16. The roll command to roll torque transfer function ($\frac{\tau_\phi(s)}{\Phi_C(s)}$) is given

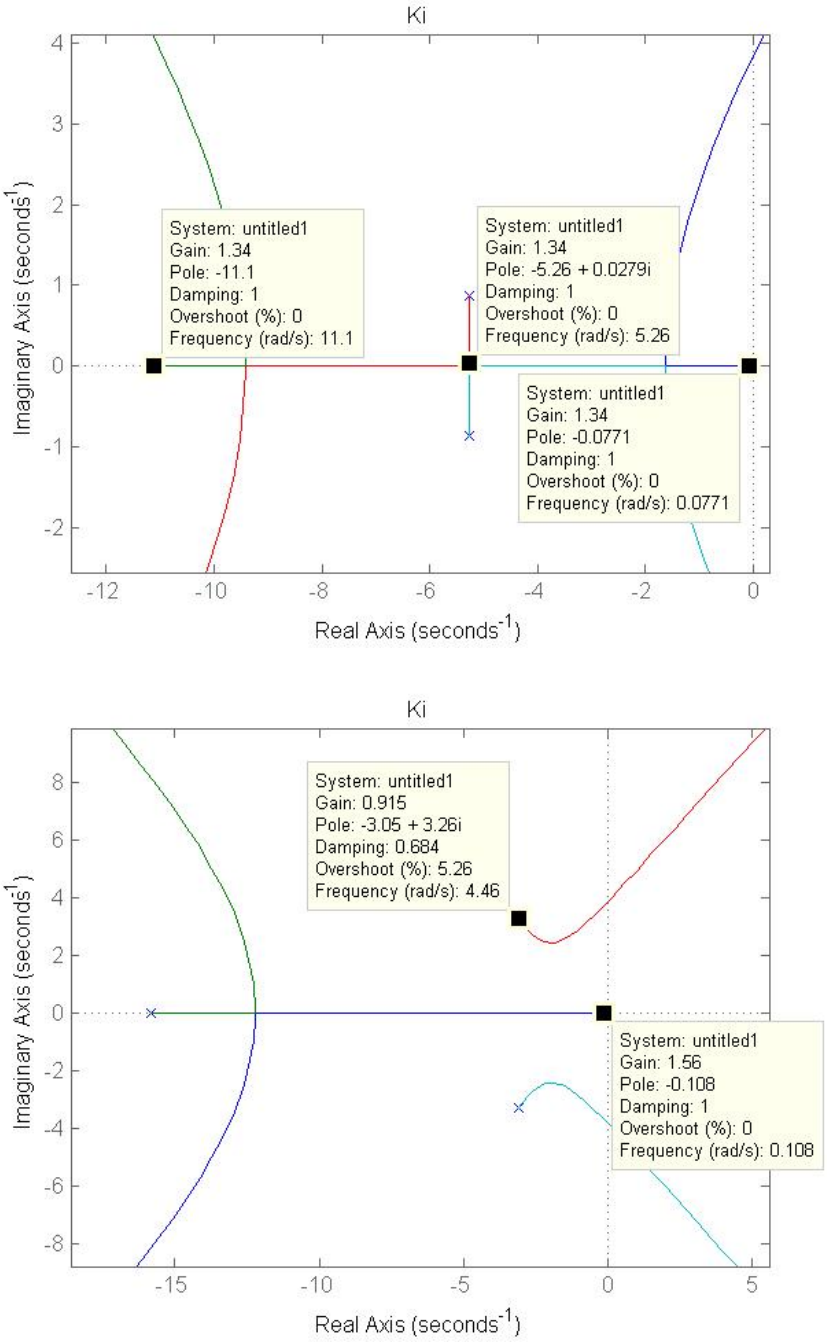

Figure 5.14: Root locus to design K_i for altitude(top) & yaw(bottom)

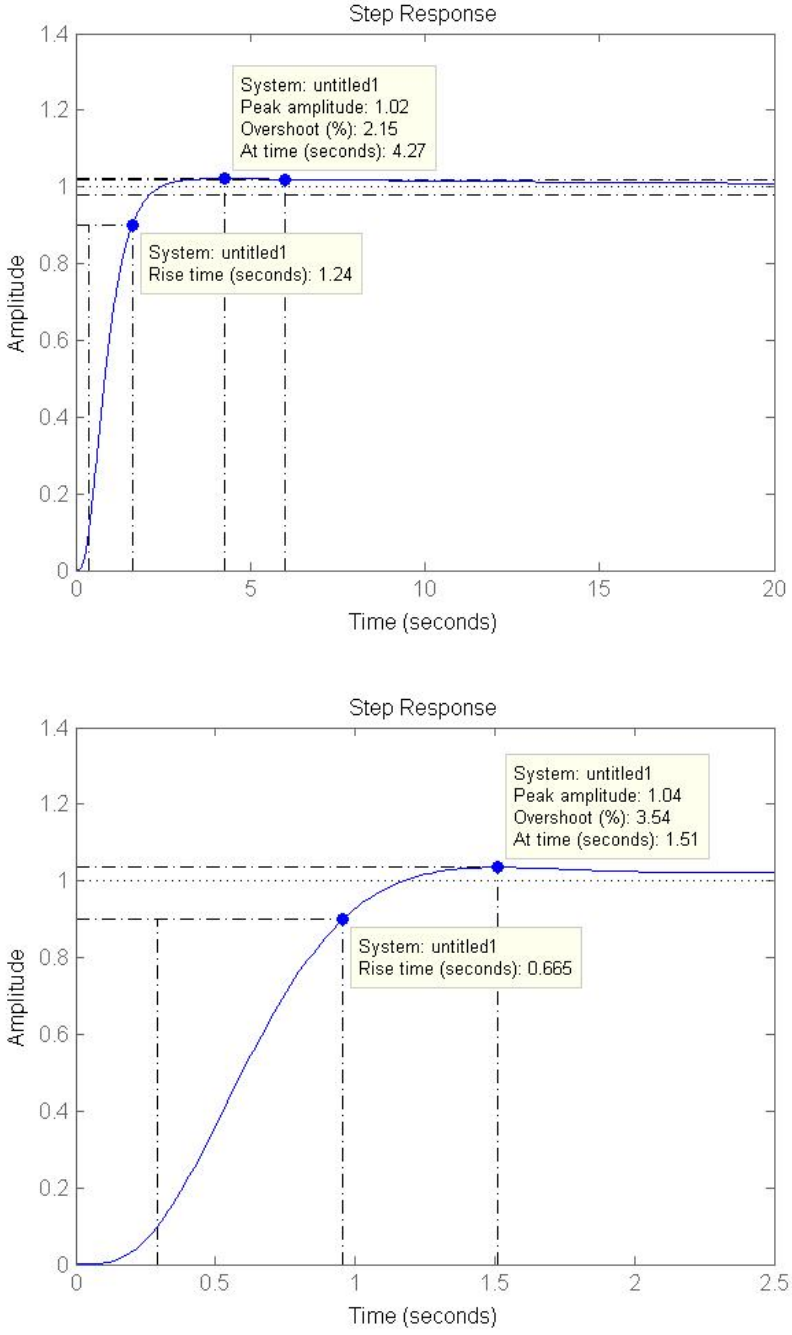

Figure 5.15: Step response of altitude(top) & yaw(bottom)

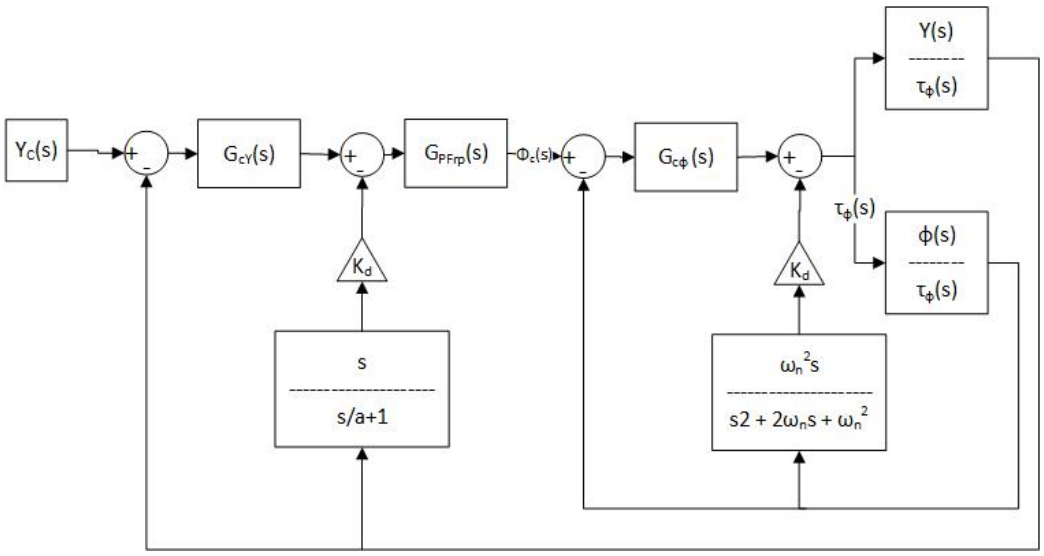

Figure 5.16: Block diagram of y_e control loop

below:

$$\frac{\tau_\phi(s)}{\Phi_C(s)} = \frac{G_{C\phi}(s)}{1 + (G_{C\phi}(s) + K_{d\phi}s)\frac{\phi(s)}{\tau_\phi(s)}} \tag{5.32}$$

where $G_{C\phi}(s)$ includes PI controller and pole-zero cancellation filter of the roll controller, and $K_{d\phi}$ is the velocity feedback gain of the roll controller (see Table 5.1, the velocity feedback filter is considered negligible in the outer-loop control design), and $\frac{\phi(s)}{\tau_\phi(s)}$ is given by the transfer function (5.20). The roll command to roll Torque transfer function is then used as part of the transfer function $G_{PY}(s)$, given below, in the control design:

$$G_{PY}(s) = G_{PFrp}(s)\frac{\tau_\phi(s)}{\Phi_C(s)}\frac{Y(s)}{\tau_\phi(s)} \tag{5.33}$$

where the $G_{PFrp}(s)$ is the prefilter for the roll controller (see Table 5.1), and $\frac{Y(s)}{\tau_\phi(s)}$ is (5.17). Note that the transfer function $G_{PY}(s)$ is a Type 1 system.

The X and Y position control is designed for the following specifications:

- $\geq 45°\%$ phase margin
- $\leq 1\frac{rad}{sec}$ gain crossover frequency

The primary design considerations for the outer loop is stability, specifically a target phase margin of $45°$. In addition, the bandwidth of the controller must be less than that of the inner loop, since the outer loop

cannot react faster than the inner loop. To account for this, a gain cross-over frequency of approximately $1\frac{rad}{sec}$ is desired, such that the inner loop is approximately five times faster than the outer loop. The X and Y position commands will be limited to $0.5\frac{m}{s}$ with a rate-limiter, in order to prevent large commands of roll and pitch. A PID control law design using velocity feedback, the same as for altitude and yaw, is used. The design procedure is given below.

1. Velocity feedback is used, estimating velocity with a first order high-pass filter, with a corner frequency of $10\frac{rad}{sec}$. The root locus is shown in Figure 5.17 and Figure 5.18 for X position, and Y position, respectively. A K_d value of .2 for both X and Y position was found to be sufficient.

2. The proportional gain is designed with a trade-off of speed versus overshoot. The root locus is shown in Figure 5.19 for X position, and Figure 5.20 for Y position, respectively. The rate limiter effectively limits the bandwidth of the system. As such the high overshoot for the high frequency pole pairs in Figures 5.19 and 5.20 is acceptable during design, since the overshoot will be attenuated by the rate limiter. A K_p value of .3 for both X and Y position is found to be sufficient.

3. This is a Type 1 system, so the steady state error to step input is zero. However, an integral controller is used to ensure zero steady state error from step disturbances. The integral gain is designed as a trade-off between the overshoot small and settling time of disturbances. By using the root locus in Figure 5.21 and Figure 5.22 for X position, and Y position, respectively, a K_i value of 0.02 is used for both X and Y position to meet specifications.

Figure 5.23 shows the gain and phase margins of the design. It shows that the phase margin for both designs is above 45° and the gain crossover frequencies for the designs are at $1.17\frac{rad}{s}$ and $1.12\frac{rad}{s}$ for the X Position and Y Position, respectively. This considered acceptable. The gain values for this design are given in Table 5.3. Figure 5.24 shows the response to a rate limited step, which shows that the high frequency pole pair is suppressed. However, it shows 15% and 17% overshoots, for X Position and Y Position, respectively.

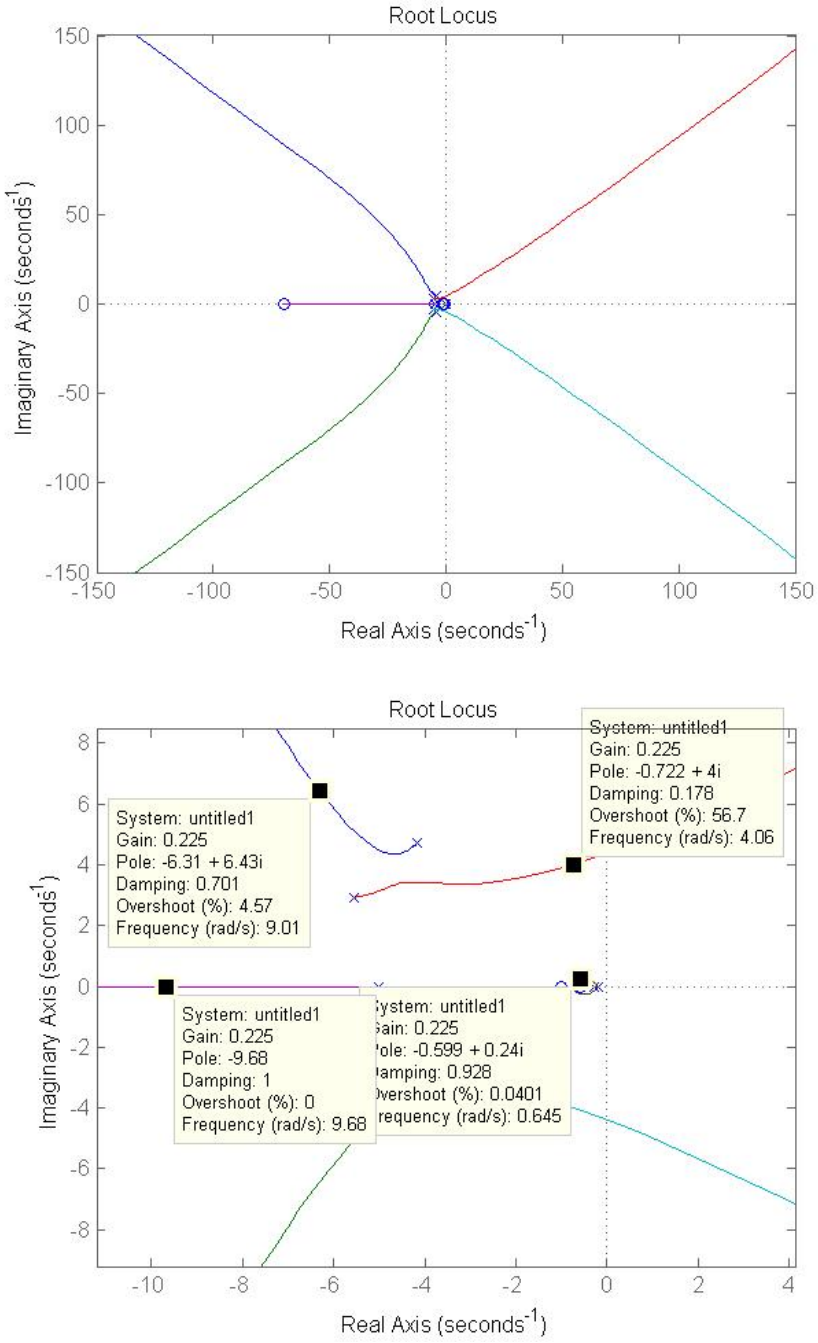

Figure 5.17: Root locus to design K_d for X position

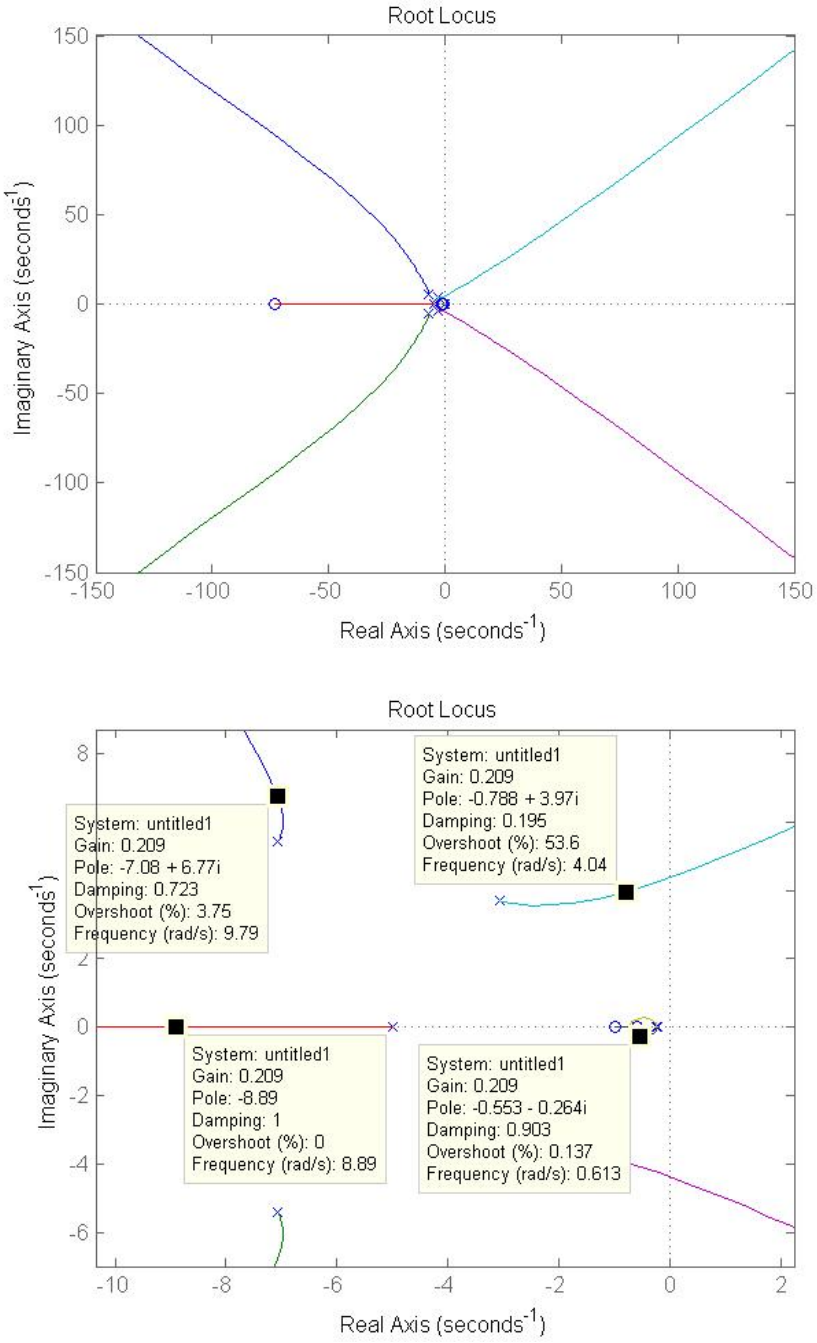

Figure 5.18: Root locus to design K_d for Y position

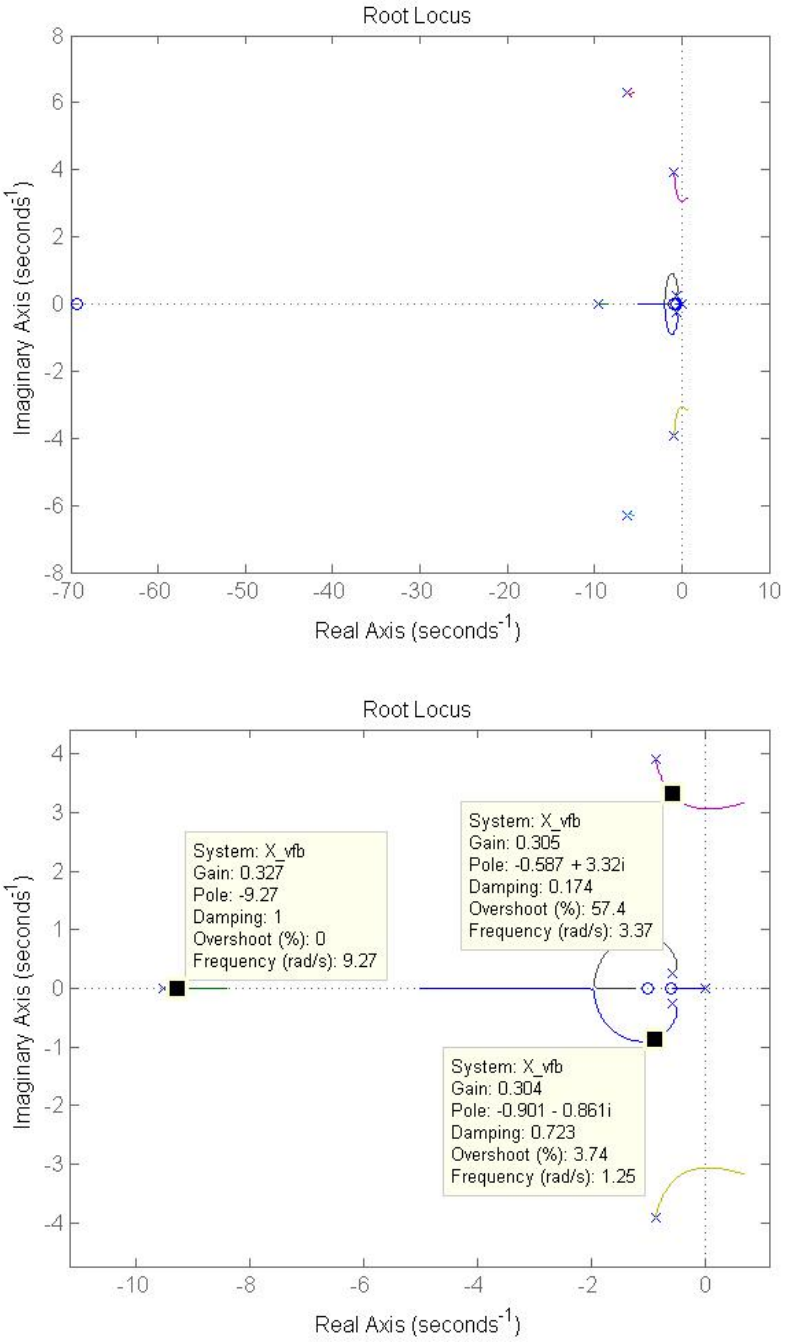

Figure 5.19: Root locus to design K_p for X position

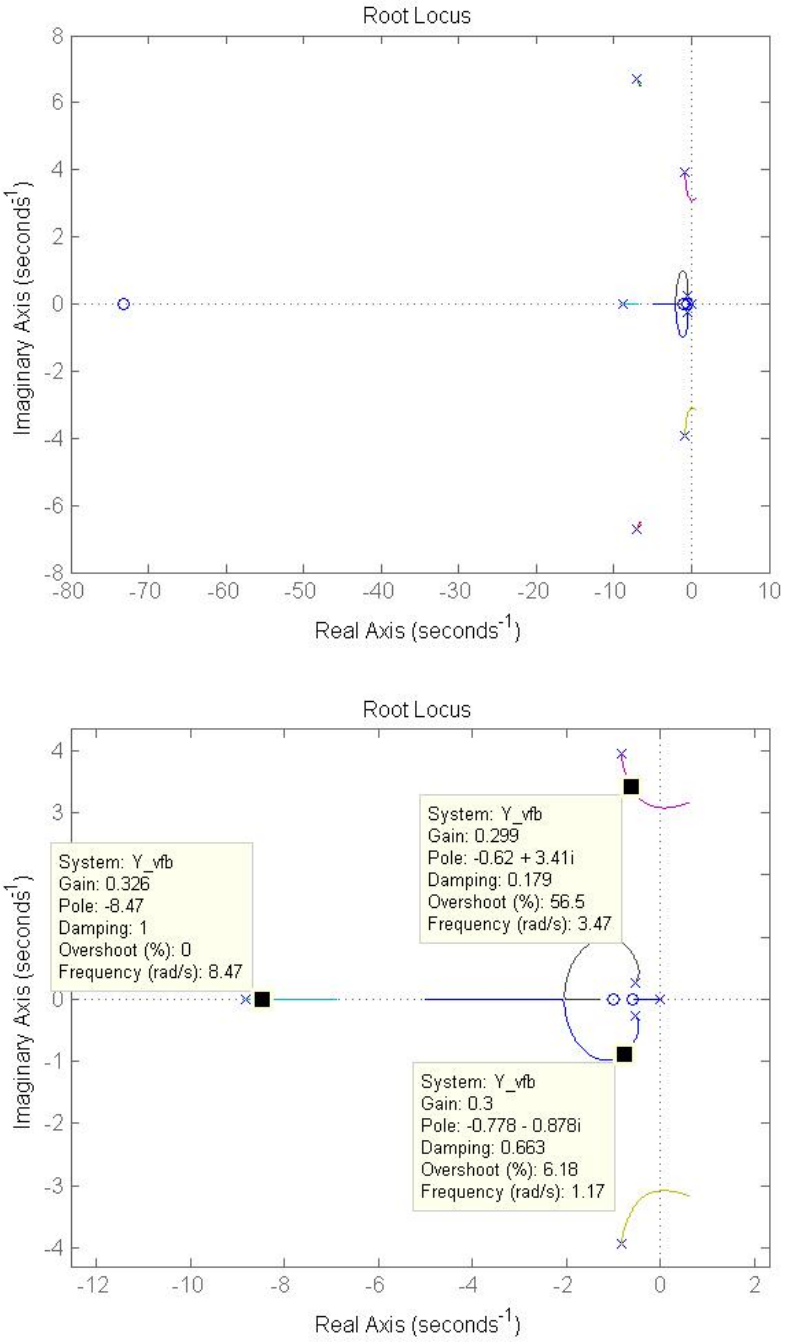

Figure 5.20: Root locus to design K_p for Y position

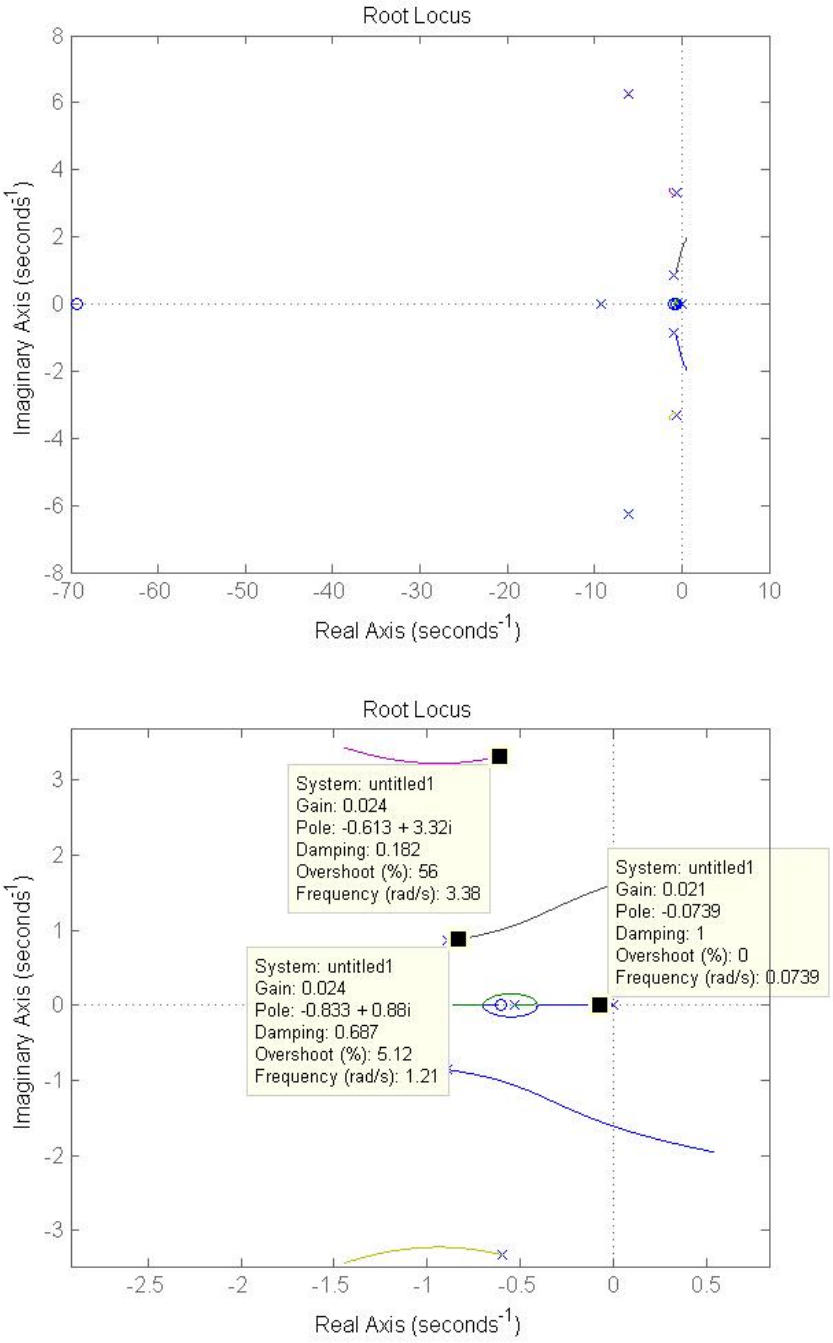

Figure 5.21: Root locus to design K_i for X position

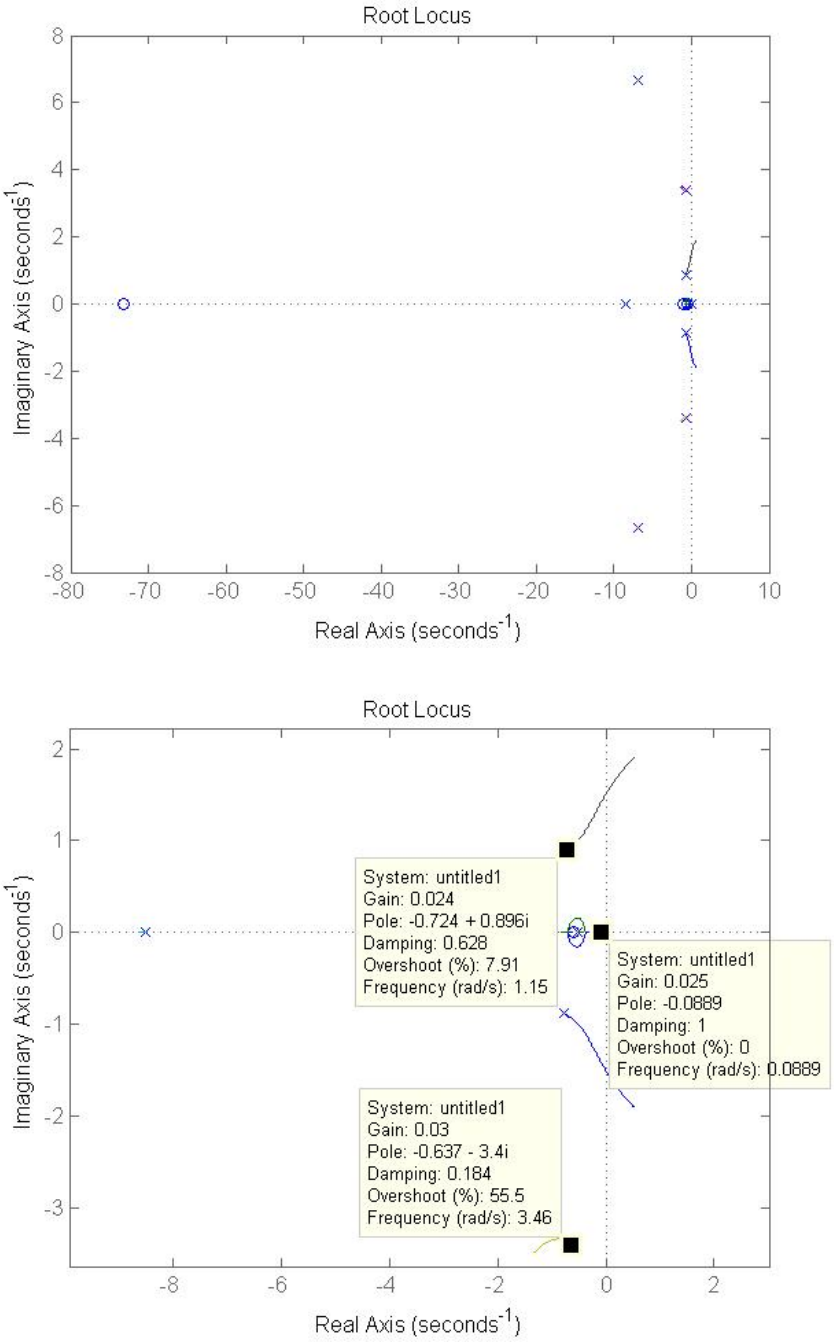

Figure 5.22: Root locus to design K_i for Y position

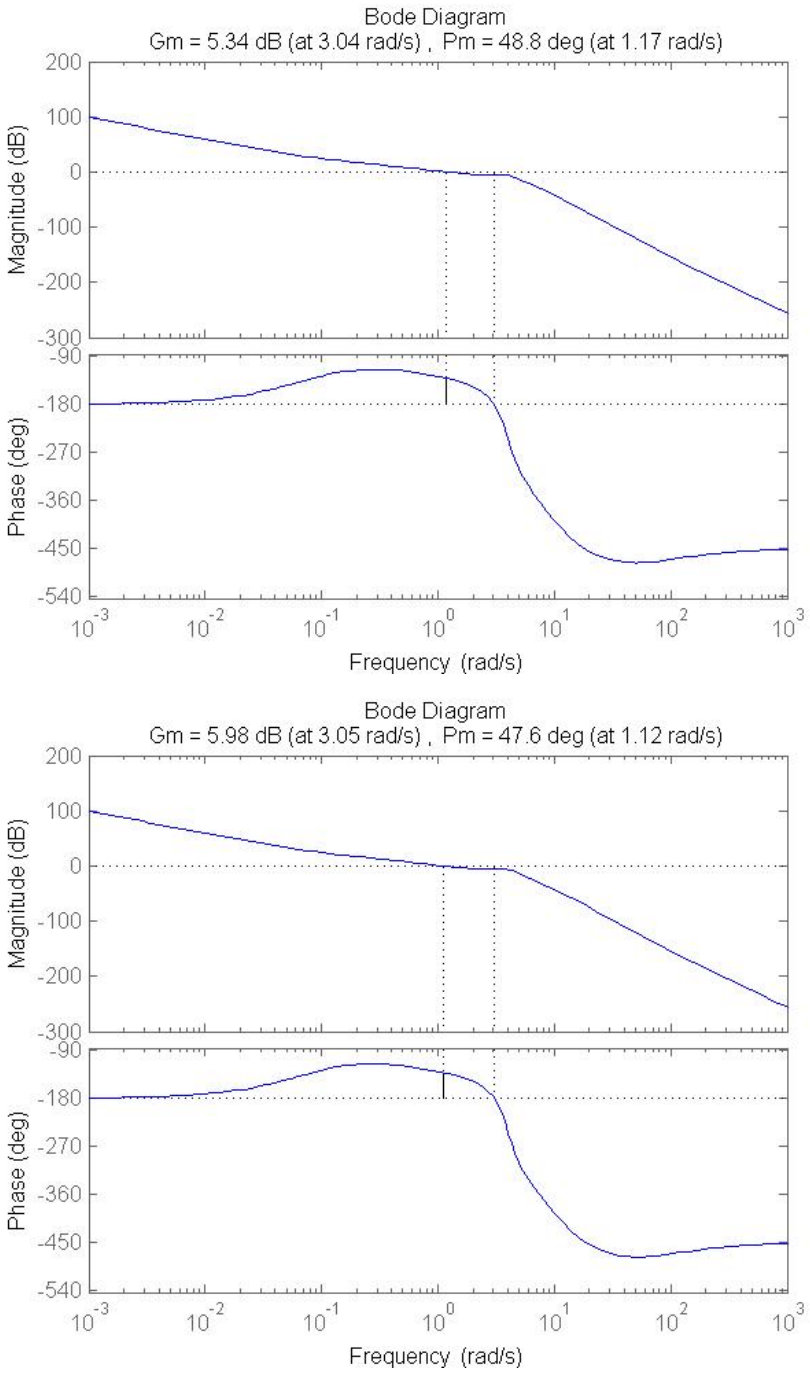

Figure 5.23: Open loop bode plot for X position (top) & Y position (bottom)

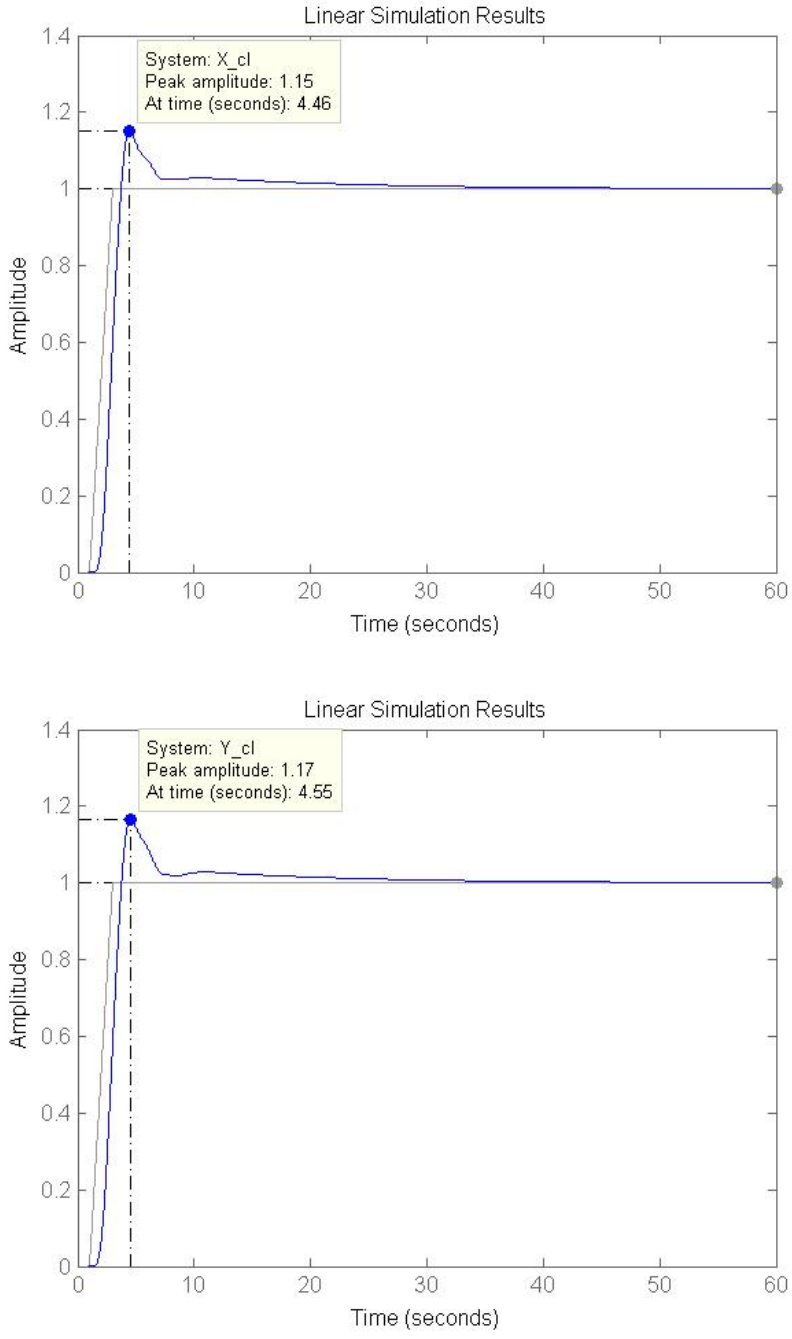

Figure 5.24: Input response plot for 'x' position (top) & 'y' position (bottom)

Table 5.3: Gain values for X position (left) & Y position (right) controllers

	x_e	y_e
K_p	.3	.3
K_i	.02	.02
K_d	.2	.2

5.5 Flight Simulation Results

To test the designs, a nonlinear simulation model of the system was prepared in Simulink. The controller send the PWM signals to the non-linear dynamic model, found previously. The simulation runs at the $100Hz$ sample frequency of the COM. In addition, noise and delay characteristics are added to the system, to help the simulation better model the quadcopter.

The noise and delay of the system are both approximated from the data of real flights. As the noise of the system did not appear to match additive white Gaussian noise, it was approximately modeled. To model the noise, flight data for the position and Euler angles were processed through the polynomial regression filter, and then the difference was taken between the un-filtered and filtered data. This difference was taken to be the noise (assuming that it was representative of the normal noise of the system). During the simulation, this noise was injected on a repeatedly. To model the transmission delay, a unity ramp function are transmitted from the ground station to the COM. The COM then transmits these signals back. The difference between the ramp functions is halved, and considered to be the one-way transmission delay. Then, the latency from the VICON block is added to become the total delay. On average, the total delay was found to be approximately 2 samples. The delay and noise were added to the feedback signals. The decay of the battery voltage was not modeled in the simulation. In addition, the ground effects was not modeled.

Figures 5.25-5.30 show the step responses in simulations. The responses for X, Y Positions, altitude, and yaw control are similar to the step responses of the linear model. Roll and pitch step responses, however, show a higher frequency ripple, which is not present in the linear step response. This is due to coupling effects of system states, as well as the non-linear effects. However, the simulation does show stability and acceptable performance, thus these controllers may be deployed to the quadcopter vehicle.

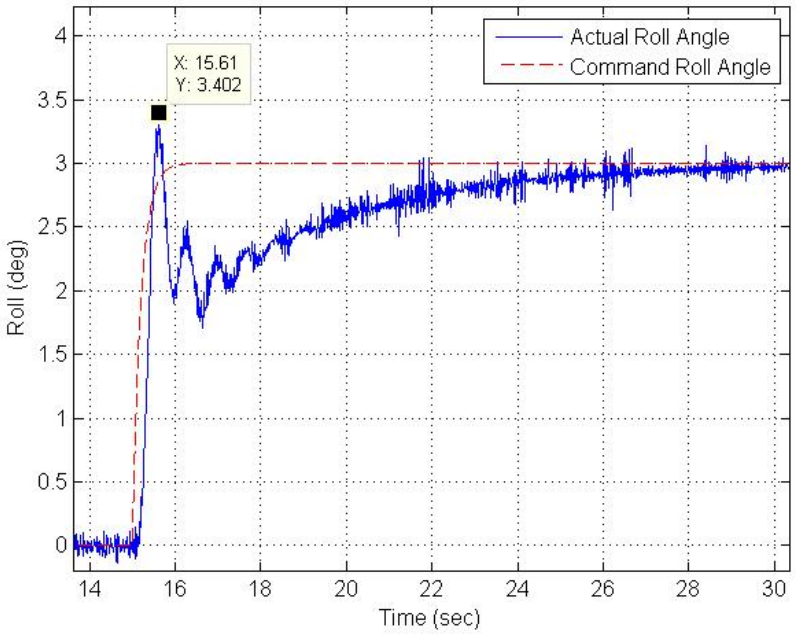

Figure 5.25: Simulation results for a step roll input

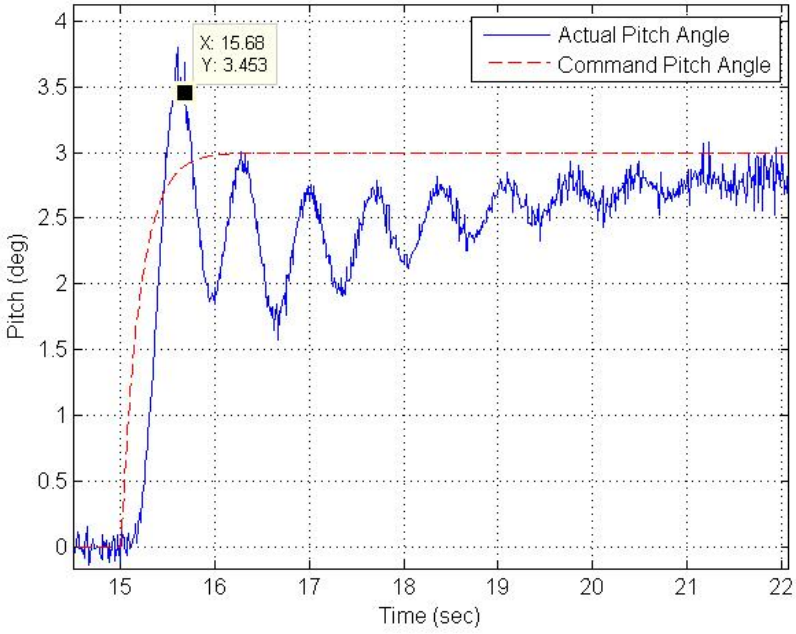

Figure 5.26: Simulation results for a step pitch input

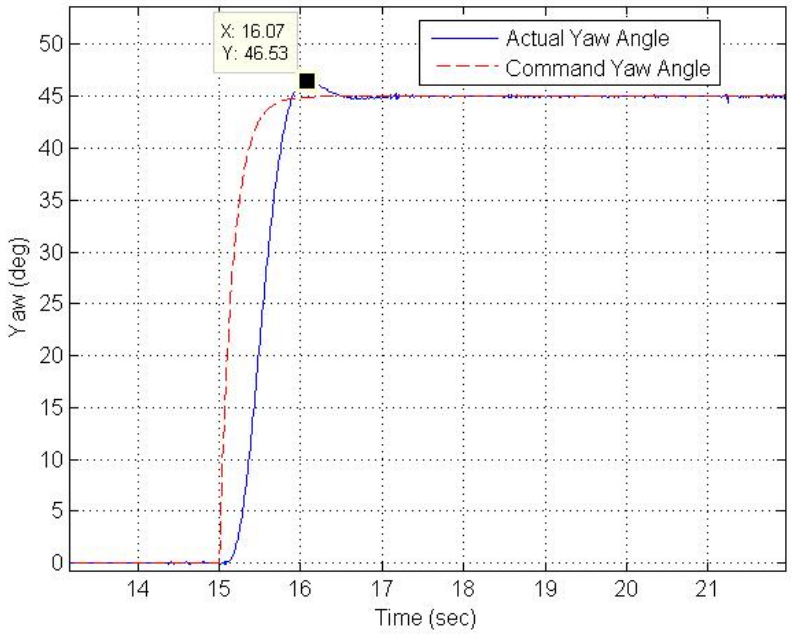

Figure 5.27: Simulation results for a step yaw input

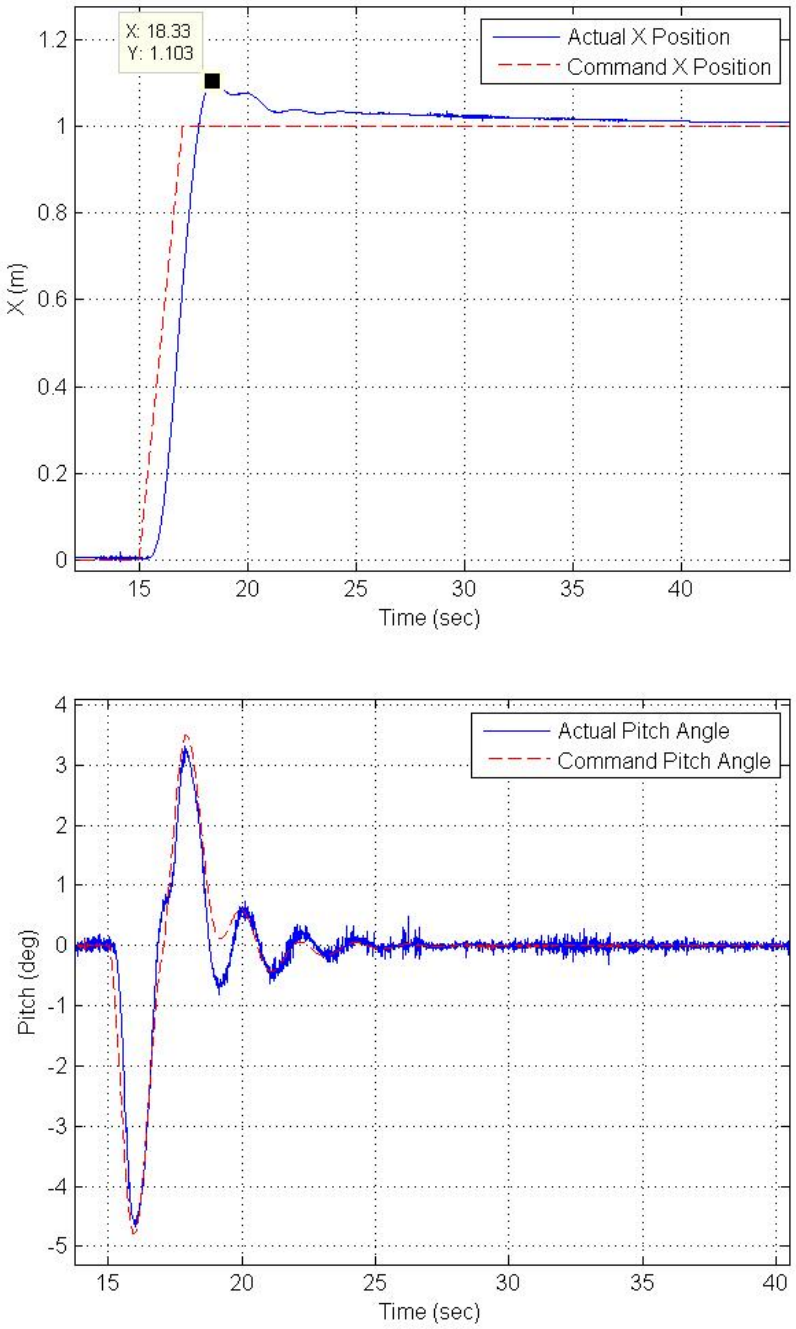

Figure 5.28: Simulation results for a rate limited step X input for X position (outer loop) and pitch (inner loop)

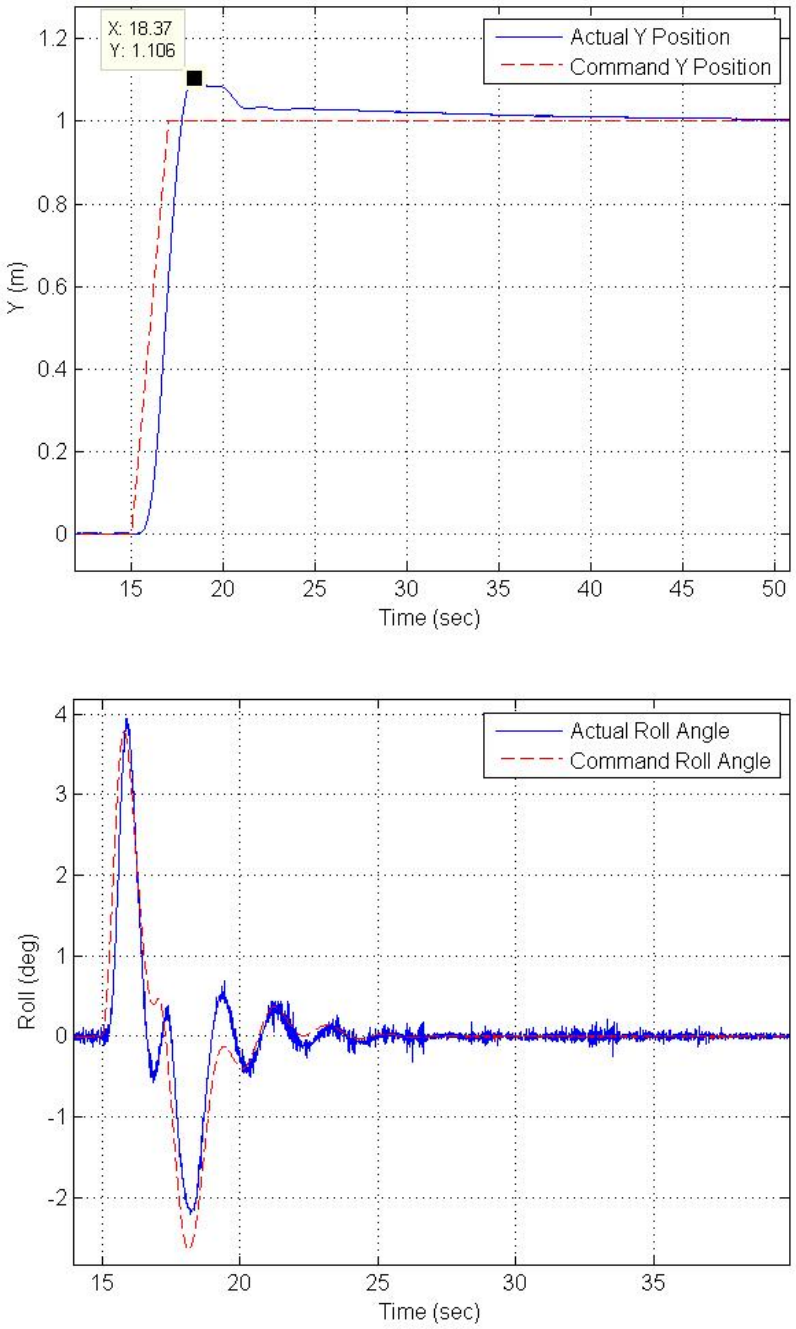

Figure 5.29: Simulation results for a rate limited step Y input for Y position (outer loop) and roll (inner loop)

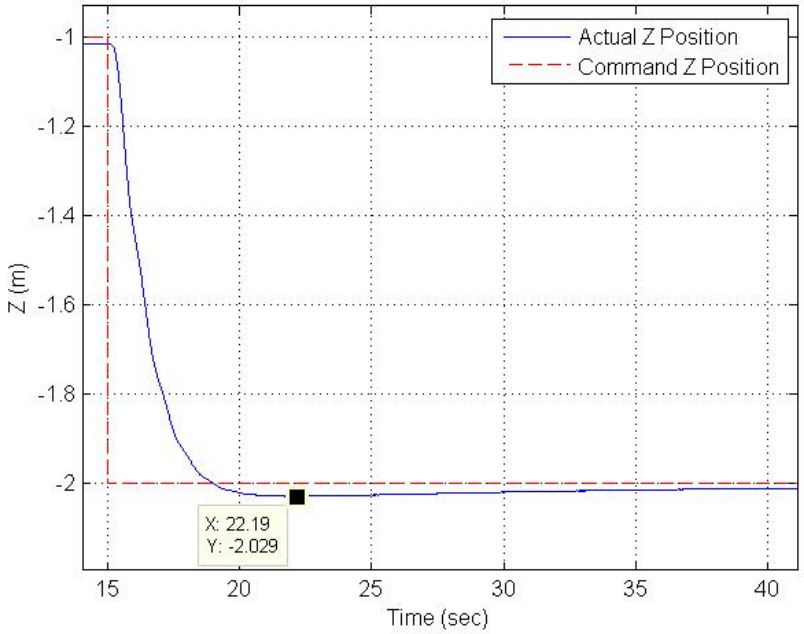

Figure 5.30: Simulation results for a step Z input

5.6 Controller Flight Test Results

Once the controller is verified in simulation, the controllers may be deployed to the quadcopter vehicle. The parameters in Tables 3.1, 4.2, 4.4, 4.6, 4.8, 4.10, and 4.12, and in (5.11) and (5.14), are implemented in the flight control law. The routes for X and Y positions are mapped, with a repeating sequence block in Simulink, during position control. Altitude and attitude commands are generated using a RC controller, so pure steps are not feasible in execution.

Figures 5.31 - 5.36 show the results of the flight tests. The tracking error is larger than what are shown in the simulation. This is because certain characteristics aren't captured in simulation model, for example the coupling between states. However, overall control performances are acceptable.

Flight tests show that the controller performance of the vehicle is heavily dependent on battery voltage, which the simulation did not take into account. This indicates that the motor battery mapping, as well as the battery decay needs to be more accurately modeled. In addition, even though controller gains that worked in simulation also worked in flight, some gains that worked in flight did not work in simulation. This indicates the real system is more stable than the model indicates. While the differences in flight and simulation show that the model could be improved, the results show that this model is sufficient to create a stable model-based classical control law design for the quadcopter.

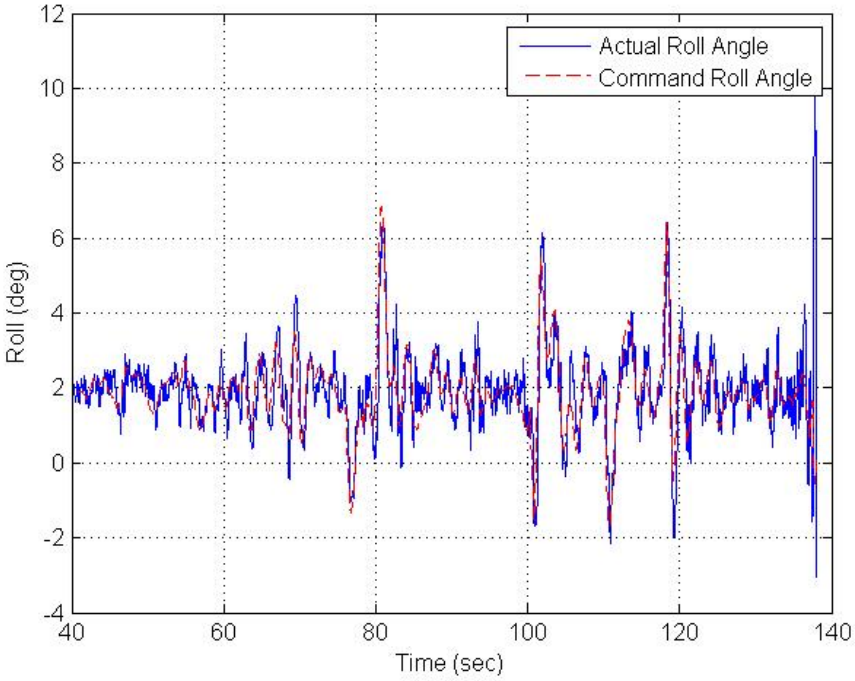

Figure 5.31: Flight test roll results

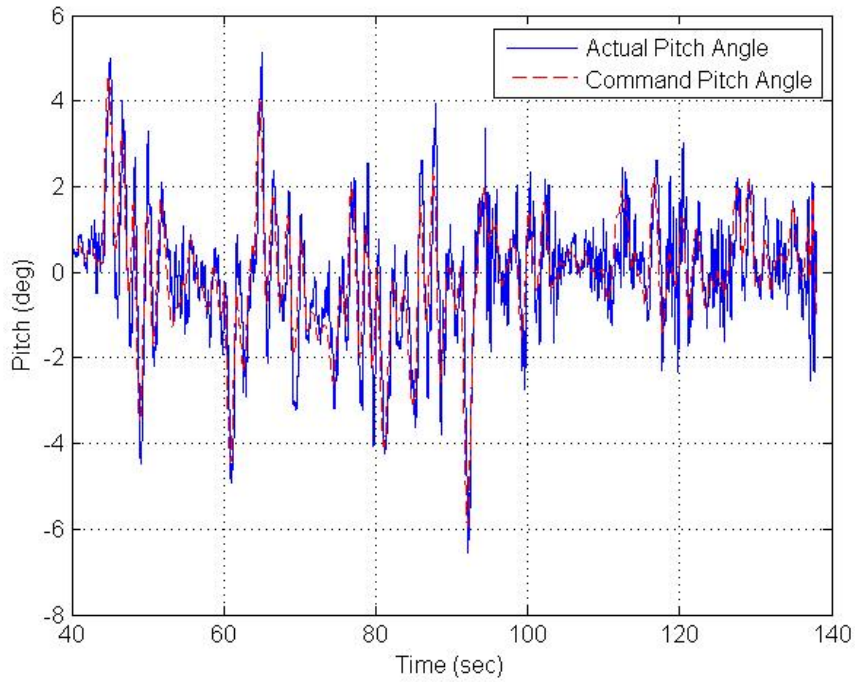

Figure 5.32: Flight test pitch results

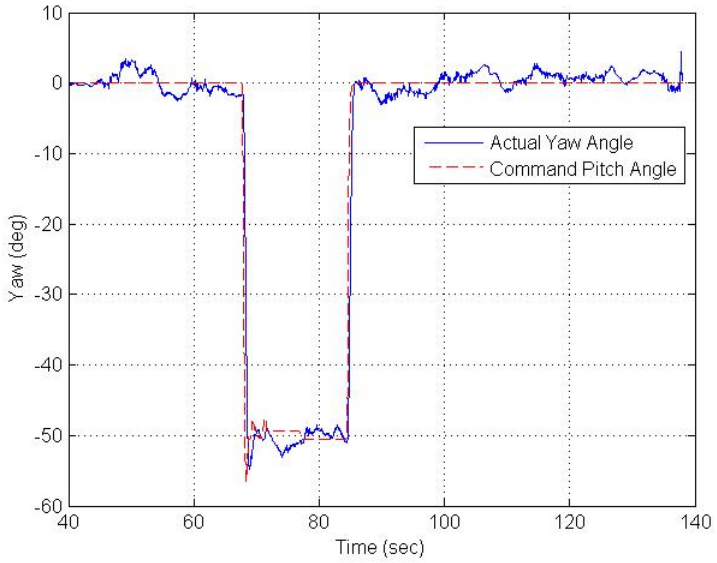

Figure 5.33: Flight test yaw results

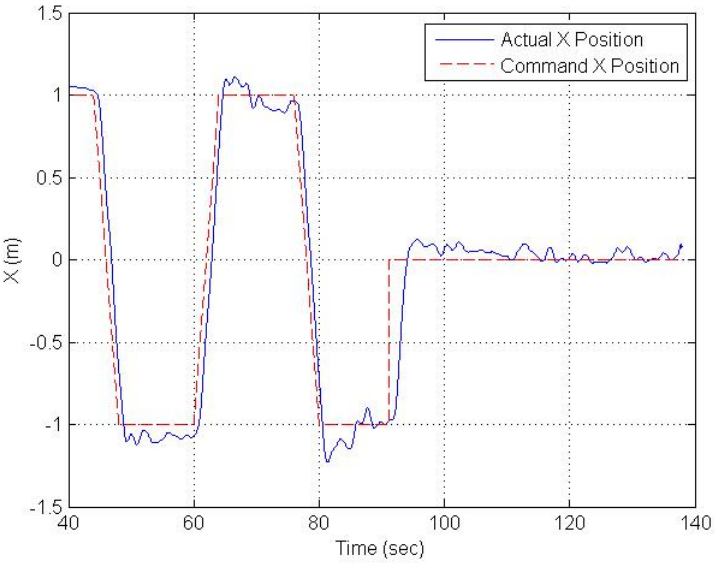

Figure 5.34: Flight test X position results

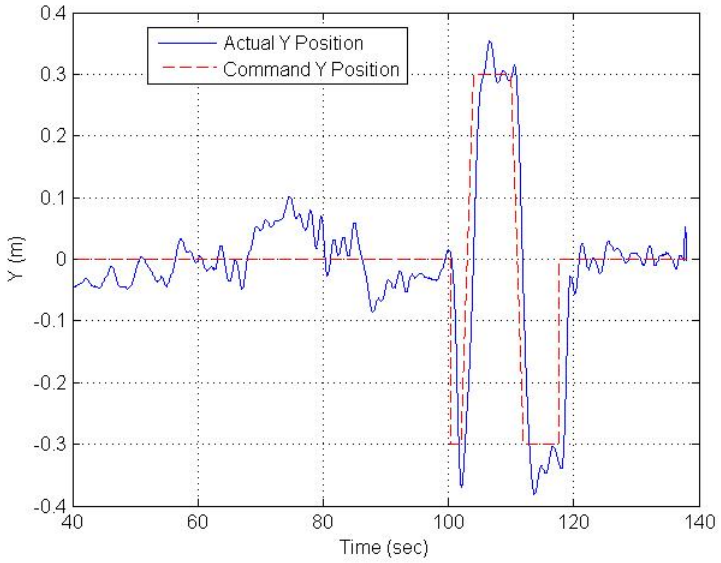

Figure 5.35: Flight Y position results

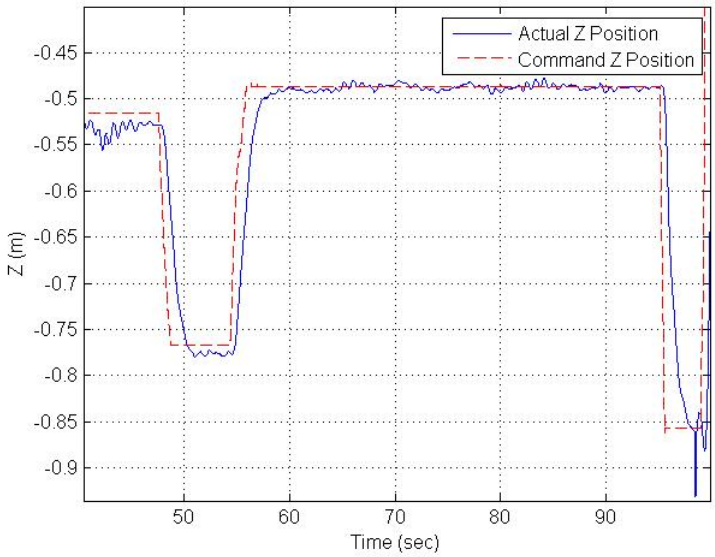

Figure 5.36: Flight Z position/Altitude results

Chapter 6: Conclusion

In this thesis, a non-linear dynamic model of a quadcopter vehicle is developed using Linear Least Squares Error method of parameter estimation. The RT-MaG powered quadcopter platform developed for this research is flown in an indoor test environment equipped with a VICON visual tracking camera system and excited to create frequency rich data sets. The in-flight data is then processed offline to estimate model parameters. Once the model was created it was then implemented in a simulation environment, and linearized to perform a model-based classical control law designs for both attitude and position. These control laws were simulated and then deployed on the physical quadcopter for flight tests. The results show the ability of this dynamic model to serve for both simulation as well as model-based control design.

This research provides various opportunities for future research. A more accurate dynamic model will help model-based desgin of advanced algorithms including adaptive and intelligent systems, fault diagnosis, Kalman filtering, or neural network controllers. In addition, model-based classical controllers designs for quadcopters is useful for educational purposes and training students.

One of the benefits of the Linear Least Square Error method of parameter estimation is that new models can be easily tested. Thus this method can be further used to create more accurate models using extra terms, in addition to those used in this research, including more cross or squared terms in the proposed model. In addition, different methods of excitation may be used to obtain a more frequency rich flight data, such as uncorrelated sum of sinusoids. This method was investigated but not implemented in this research.

The quadcopter platform used in this project will need further work if will be used for future research. The Gumstix Overo AIRSTORM used has been discontinued by Gumstix, and replaced with the Overo AIRSTORM-Y, which possesses additional RAM and a different WiFi module. And new Linux images must be created to support the new COM.

In addition, there is currently an unsolved bug which may affect the internal clock of the Gumstix COM. Specifically the clock may increase faster than desired (which throws off the control laws, sample rate, and integrator). Fortunately, occurrence of the bug is very rare, having only been definitively seen twice in over a year of research. Each time, however, the glitch appeared and then stopped affecting the system without any

change in code. Heat was suspected to be a cause of the glitch, because the glitch usually gets cleared after a prolonged period of inactivity and unplugged from the battery. The glitch has so far not been able to be replicated, nor any cause or solution found. The glitch was not detected in the flight tests used for research.

Furthermore, the APM only provides the Gumstix COM with IMU data and battery voltage. Currently, control laws do no use the IMU data in favor of using VICON data for attitude. The APM could be replaced with an IMU and a power regulator/monitor directly on on the circuit board which holds the Gumstix. This could reduce the cost of the platform, improve the quality of the signals, and decrease the data lag. Also, the circuit board currently in use must be created by hand soldering. Designing and creating a printed circuit board, or using of an off-the-shelf board for this quadcopter platform will enable it to be more easily replicated.

Bibliography

[1] A. Manecy, N. Marchand, and S. Viollet, "RT-MaG: an open-source SIMULINK Toolbox for Real-Time Robotic Applications," *IEEE International Conference on Robotics and Biomimetics*, Dec. 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00943294

[2] S. Bouabdalla, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," *IEEE International Conference on Robotics and Automation*, 2004.

[3] G. Hoffmann, W. Huang, Haomiao, Steven, and C. Tomlin, "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment," *AIAA Guidance, Navigation and Control Conference and Exhibit*, vol. 2, p. 4, 2007.

[4] W. Wei and K. Cohen, "Development of a Model based Fuzzy-PID Controller for the AeroQuad Cyclone Quad-copter," *AIAA Infotech @ Aerospace*, Feb 2015.

[5] D. Rotondo, F. Nejjari, and V. Puig, "Robust Quasi–LPV Model Reference FTC of a Quadrotor Uav Subject to Actuator Faults," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 1, Jan 2015.

[6] Y. Zhang, A. Chamseddine, C. Rabbath, B. Gordon, C.-Y. Su, S. Rakheja, C. Fulford, J. Apkarian, and P. Gosselin, "Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed," *Journal of the Franklin Institute*, vol. 350, no. 9, pp. 2396 – 2422, 2013.

[7] F. X. Johan, E. Joelianto, A. Widyotriatmo, and Salmah, "Experiment of model based non linear control design for altitude control of quadrotor using vision-based localization system," *International Conference on Robotics, Biomimetics, Intelligent Computational Systems*, 2013.

[8] E. Ekawati, A. Widyotriatmo, and I. Askandari, "Quadrotor position control based on model identification and proportional-derivative algorithm," *The 2nd International Conference on Technology, Informatics, Management, Engineering and Environment*, 2014.

[9] K. N. Dang, G. Lee, and T. Kang, "Linear quadrotor modelling and attitude controller design based on experimental data," *The 15th International Conference on Control, Automation and Systems (ICCAS)*, 2015.

[10] J. Villbrandt, "The Quadrotor's Coming of Age," http://illumin.usc.edu/162/the-quadrotors-coming-of-age/, accessed: 2016-09-29. [Online]. Available: http://illumin.usc.edu/162/the-quadrotors-coming-of-age/

[11] C. of Aviation Museum and E. Center, "Convertawings Model A Quadrotor," http://www.cradleofaviation.org/history/permanent_exhibits/the_jet_age/convertawings_model_a_quadrotor.html, accessed: 2016-09-29. [Online]. Available: http://www.cradleofaviation.org/history/permanent_exhibits/the_jet_age/convertawings_model_a_quadrotor.html

[12] R. N. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition)*. Springer US, 2010.

[13] R. Stengel, "Aircraft Equations of Motion," https://www.princeton.edu/~stengel/MAE331Lecture9.pdf, 2014, accessed: 2016-08-01. [Online]. Available: https://www.princeton.edu/~stengel/MAE331Lecture9.pdf

[14] R. C. Avram, X. Zhang, J. Muse, and M. Clark, "Nonlinear adaptive control design and controller integrity monitoring for quadrotor UAVs," *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016.

[15] N. K. Dalwadi, *Mathematical Modeling of a DC Motor, Technical Report*. Wright State University, 2016.

[16] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. PTR Prentice-Hall, 1993.

[17] V. Klein and E. A. Morelli, *Aircraft System Identification: Theory and Practice*. American Institute of Aeronautics and Astronautics, 2006.

[18] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems (9th Edition)*. John Wiley & Sons, INC., 2010.

Appendix A: Effect of Deviation of Zero in Lag Controller

Cancellation of a zero in a real system carries a risk that the frequency of the zero in the model may deviate from the zero's actual location in the system, particularly after linearization. Deviation does decrease the "cancellation" effect, however so long as the deviation is not large, it still mitigates the effect of the zero. This effect any deviation is modeled by varying the system zero of the Laplace transfer function while keeping the controller constant. The difference of performance is shown in Figures A.1 and A.2, the variation of in phase margin of the system is given in Table A.1. From the step response performance, it may be estimated that so long as the frequency of the zero is not less than $0.1\frac{rad}{s}$ or greater than $0.5\frac{rad}{s}$ the performance is still acceptable.

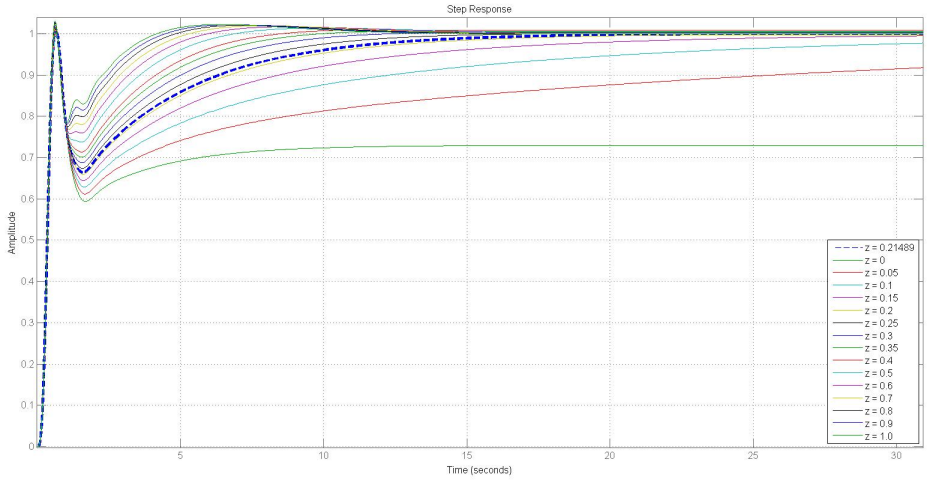

Figure A.1: Step Response While Deviating Roll System Zero Frequency

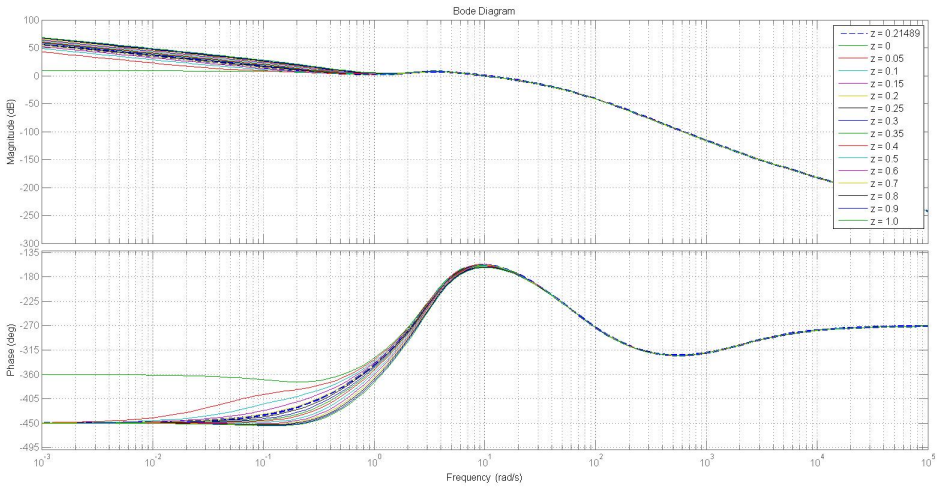

Figure A.2: Bode Plot While Deviating Roll System Zero Frequency

Table A.1: Variation of Phase Margin While Deviating Roll System Zero Frequency

Zero Frequency	Phase Margin (°)	Gain Cross-Over Frequency $\left(\frac{rad}{s}\right)$
0.2149	47.6555	5.3531
0.0000	47.6618	5.2146
0.0500	47.6598	5.2463
0.1000	47.6582	5.2783
0.1500	47.6570	5.3106
0.2000	47.6558	5.3433
0.2500	47.6546	5.3764
0.3000	47.6531	5.4098
0.3500	47.6510	5.4436
0.4000	47.6483	5.4777
0.5000	47.6400	5.5472
0.6000	47.6262	5.6182
0.7000	47.5912	5.6936
0.8000	47.5626	5.7674
0.9000	47.5305	5.8416
1.0000	47.4733	5.9198

Appendix B: Use of PI Controller for Disturbance Rejection

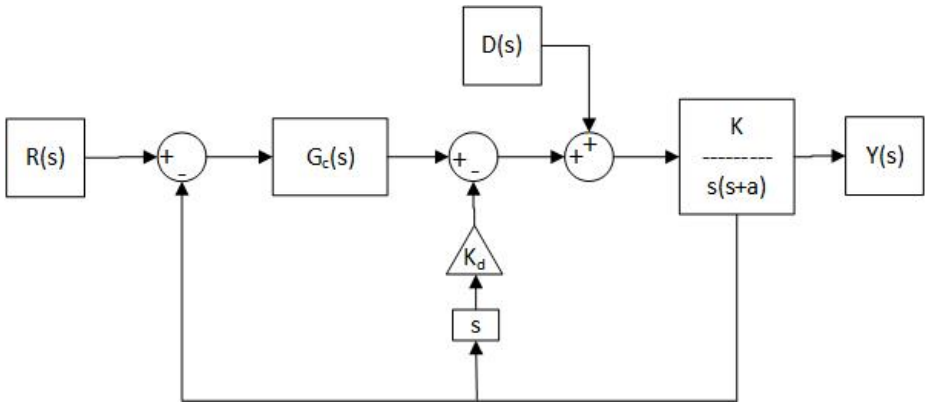

Figure B.1: Example Type 1 System

An Integral component of a PI controller is often used to eliminate steady state error to a step input. This feature is unnecessary for Type-1 systems, however integral components can be implemented to eliminate the steady state error to a step disturbance input. Figure B.1, shows an example Type 1 system with velocity feedback. The loop of interest is the disturbance input $(D(s))$ to the system output $(Y(s))$ response. The $\left(\frac{Y(s)}{D(s)}\right)$ transfer function is shown below, first with a proportional controller $(G_c(s) = K_p)$, then with a PI controller $(G_c(s) = K_p + K_i s)$:

$$\frac{Y(s)}{D(s)} = \frac{\frac{K}{s(s+a+K_dK)}}{1+K_p\frac{K}{s(s+a+K_dK)}} = \frac{K}{s^2+(a+K_dK)s+K_pK} \tag{B.1}$$

$$\frac{Y(s)}{D(s)} = \frac{\frac{K}{s(s+a+K_dK)}}{1+(K_p+\frac{K_i}{s})\frac{K}{s(s+a+K_dK)}} = \frac{Ks}{s^3+(a+K_dK)s^2+K_pKs+K_iK} \tag{B.2}$$

(B.1) is the closed loop transfer function to with a proportional controller, (B.2) is the closed loop transfer function to with a PI controller. The steady state value of each system can be found using the final value theorem with a step disturbance applied.

$$\lim_{t\to\infty} y(t)\Big|_{u(t)} = \lim_{s\to 0} s\frac{Y(s)}{D(s)}\frac{1}{s} = \lim_{s\to 0}\frac{Y(s)}{D(s)}$$

$$\lim_{s\to 0} \frac{K}{s^2 + (a + K_d K)s + K_p K} = \frac{K}{K_p K} \tag{B.3}$$

$$\lim_{s\to 0} \frac{Ks}{s^3 + (a + K_d K)s^2 + K_p K s + K_i K} = \frac{0}{K_i K} = 0 \tag{B.4}$$

(B.3) is the final value theorem with a proportional controller, which can be observed to have a steady state value, (B.4) is the final value theorem with a PI controller which has zero steady state value. This feature of the PI controller is useful in system which have various known disturbance, such as gravity or other control loops acting on the system.