Modeling, Design and Control of a 6 D-O-F Quadcopter Fleet With Platooning Control

by

Anshuman Srinivasan

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

Approved December 2020 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Chair
Konstantinos Tsakalis
Jennie Si

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Vertical take-off and landing (VTOL) systems have become a crucial component of aeronautical and commercial applications alike. Quadcopter systems are rather convenient to analyze and design controllers for, owing to symmetry in body dynamics. In this work, a quadcopter model at hover equilibrium is derived, using both high and low level control. The low level control system is designed to track reference Euler angles (roll, pitch and yaw) as shown in previous work [1],[2]. The high level control is designed to track reference X, Y, and Z axis states [3].

The objective of this paper is to model, design and simulate platooning (separation) control for a fleet of 6 quadcopter units, each comprising of high and low level control systems, using a leader-follower approach. The primary motivation of this research is to examine the "accordion effect", a phenomenon observed in leader-follower systems due to which positioning or spacing errors arise in follower vehicles due to sudden changes in lead vehicle velocity. It is proposed that the accordion effect occurs when lead vehicle information is not directly communicated with the rest of the system [4][5].

In this paper, the effect of leader acceleration feedback is observed for the quadcopter platoon. This is performed by first designing a classical platoon controller for a nominal case, where communication within the system is purely ad-hoc (i.e from one quadcopter to it's immediate successor in the fleet). Steady state separation/-positioning errors for each member of the fleet are observed and documented during simulation. Following this analysis, lead vehicle acceleration is provided to the controller (as a feed forward term), to observe the extent of it's effect on steady state separation, specifically along tight maneuvers. Thus the key contribution of this work is a controller that stabilizes a platoon of quadcopters in the presence of the accordion effect, when employing a leader-follower approach.

The modeling shown in this paper builds on previous research to design a low cost quadcopter platform, the Mark 3 copter [1]. Prior to each simulation, model nonlinearities and hardware constants are measured or derived from the Mark 3 model, in an effort to observe the working of the system in the presence of realistic hardware constraints. The system is designed in compliance with Robot Operating System (ROS) and the Micro Air Vehicle Link (MAVLINK) communication protocol.

DEDICATION

I would like to dedicate this thesis research to my parents and grandfather.

ACKNOWLEDGMENTS

I would like to acknowledge and sincerely thank all people who have supported me to complete this thesis. I would like to express my gratitude to my advisor Dr. Armando Rodriguez for his continuous support of my studies and research, and his invaluable knowledge that I learned from throughout my journey at ASU. I also thank him for giving me the opportunity to work on exciting control topics; and for his help at all times for which I am grateful for.

I would also like to thank my thesis committee members, Dr. Konstantinos Tsakalis and Dr. Jennie Si for their support and help in my studies. Further, I would like to thank my colleagues at ASU - Shi Lu, Brent Wallace, Abdullah Altawaitan, Sai Shravan Manne and Kaustuv Mondal for their help and support throughout this journey.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Quadcopter research has been of great interest recently, given it's scope for greater maneuverability and control as compared to other models such as fixed wing aircraft. A standard quadcopter model for a hovering equillibrium (i.e without any movement) is fairly decoupled, meaning that the system can be visualised as multiple single input single output (SISO) systems operating in tandem. This makes control design and analysis of quadcopter models fairly simple, especially when using feedback linearization methods[2],[5]. Many conventional control design methods such as PID, LQR, backstepping control and sliding mode control) are applicable to quadcopter systems.

Quadcopter systems are inherently unstable and under-actuated ; there are four control inputs and 6 degrees of freedom, and as such a robust control system is required for complete input-output stability. As seen in contemporary research [2],[6],[7], modeling a single quadcopter to follow high level position commands is in itself a detailed and intriguing problem when considering that there are multiple time-varying states to stabilize, arising from both translational and rotational dynamics

With a single quadcopter acting as a virtual leader, this paper considers a leader-follower approach where only the leader is provided a trajectory. A separation control system is designed for follower quadcopters, such that a desired formation vector can be tracked with minimal deviation from assigned position, assuming that the only input provided is x, y, z position data from the quadcopter's nearest neighbor.

In this context, the 1992 seminal paper of Charles Desoer and Shahab Sheikholeslam poses an interesting question; what is the effect of leader information on the local stability of follower vehicles? In context, when considering leader follower

approaches, there is an accordion effect that arises within the quadrotor formation if data is simply communicated from one quadcopter to another within the formation.

Thus, the key contribution of this paper is the design of a controller to stabilize a leader-follower quadcopter fleet in the presence of the accordion effect, given a predetermined formation vector that specifies the desired vehicle spacing from a follower's immediate neighbours in formation.

Chapter 2

MOTIVATION AND OUTLINE OF WORK

2.1 Literature Survey

The quadrotor system must be designed well with respect to angular rate control (inner loop), angle control (outer loop), high level reference state tracking (for desired X,Y,Z references) and finally, separation control to ensure followers remain at desired spacing from each other. In this section (and in subsequent section in this thesis), these aspects are addressed in the following order 1. Modeling for a single Quadcopter 2. Low level control for a single quadcopter- Angular rate command following (inner loop) and Attitude angle command following (outer loop) 3. High level control for a single quadcopter- Translational position/path following control 4. Separation control for multiple quadcopters - control of spacing (Δ) between quad rotors in the fleet, in both X axis (longitudinal spacing) and Y axis (lateral spacing).

2.1.1 Modeling for a Single Quadcopter

In this work, each individual quadcopter is implemented using a kinematic model (to represent the Euler angle/body dynamics) and a dynamic model (comprising of rigid body and actuator dynamics). Figure 2.1 below shows the free body diagram for a single quadcopter.

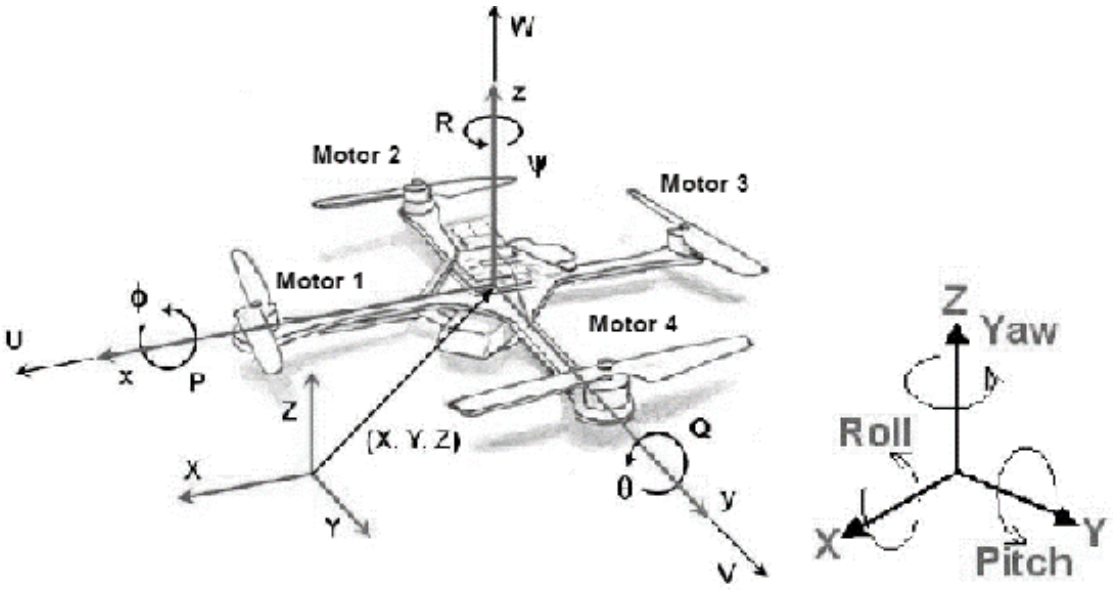

Figure 2.1: Quadcopter Free Body Diagram

Kinematic Model of Quadrotor In this work, the Cartesian coordinate system, i.e., Earth-centered Earth-fixed (ECEF) is considered as the global reference frame, while the Euler angles represent the vehicle's orientation in the body frame[1]. For modeling the controller, the body frame is used to model control inputs as shown in [1] and [2]. Rotation matrices are used to transform quadcopter states from the body frame to the global reference frame and vice versa; these are described further in Chapter 3

Dynamic Model of Quadrotor Each quadcopter is modeled using rigid and actuator dynamics, and it is assumed that forces and moments may be considered stable inputs (controls) for the system [3,11]. To map these forces and moments into corresponding motor speeds (in radians/second) we consider an inverse mapping function as described in [2]. Low level control is modeled by considering separate SISO (Single-Input-Single-Output) systems for roll-pitch-yaw angles and rates respectively, using

classical design concepts [8]. The angular rate control makes use of a first order transfer function, representing the actuator dynamics for the four BLDC (brushless DC) motors on the quadrotor. The choice of brushless motor is based on it's common use in contemporary quadcopter research platforms[9,10]

In this paper, it is assumed that all four model models are identical, each representing an ideal ESC (Electronic Speed Controller) mapping a PWM input signal to a desired motor speed. This model is derived using hardware testing results [1] to obtain a suitable transfer function representing the motor model.

2.1.2 Low Level Controller Design - Cascade Control Scheme

As discussed in the previous section, each angle in the inner-loop control is controlled using a SISO system. Classical SISO design ideas are described in [8], to model controllers which can achieve target specifications such as desired bandwidth,phase margin, settling time e.t.c; these are all important aspects to consider when designing a tracking controller, as described further in the modeling section of this thesis. Specifically ,the unity gain crossover frequency and the phase margin are taken into account when modeling the control system.

The objective of the low level control design is to accurate;y track desired angular rates, subject to actuator dynamics and motor constraints for a low cost quadrotor platform. The constraints on the model correspond to physical/hardware limitations for the quadcopter, which are obtained through reference material in previous work [1,2]. Based on the angular rate control, a hierarchical (Cascade) scheme is considered for angle control, which is performed to ensure faster dynamics for the rate control system, while maintaining slower dynamics for the angle control system. Different values of bandwidth are considered for the outer loop, specifically in order to test how aggressive the outer loop bandwidth can be made (given a fixed/constrained inner

loop model based on hardware specification). This cascade control scheme has proven to work well with quadcopter systems, as in [1],[12], and [13].

2.1.3 High Level Controller Design - Position Coordinate Tracking

For high level position/trajectory control, we assume full state feedback with differential flatness in states [14]. This forms a basis for using a Linear Quadratic Regulator (LQR) system to control position and velocity states. The formulation of the LQR equations to track X, Y and Z positions/velocities; this is implemented using model descriptions found in previous work [3] to obtain a zero steady-state error in trajectories. Thus, given a reference trajectory, we have a robust controller to track desired reference positions and velocities alike, and a framework is defined for translational position and velocity control of a single quadcopter.

2.1.4 Separation/Platooning Control

As the number of viable applications for quadcopters grows, the need for cooperation between multiple quadcopters within a system to perform complex operations becomes paramount. Applications such as surveillance, load transportation and terrain mapping (as in [15,16,17]) require large areas of coverage and ,specifically in the case of the latter, communication between deployed units in the fleet to maintain assigned separation.

With trajectory control described for a single leader,, the separation control is implemented by considering a leader-follower system, where follower quadcopters maintain fixed spacing from their immediate upstream predecessor in the fleet [18,19].

A leader-follower approach has the advantage of dynamic re-positioning i.e quadcopters can quite easily change formation for any desired time period for functions such as obstacle avoidance.[20,21]. This is because all units in the formation maintain

reference positions provided by a singular leader. However, using previous work as a basis [4],[5], the leader follower approach suffers from a major drawback, known as the accordion effect. In context of traffic flow stability and intelligent systems, this implies that any perturbation to the leader's trajectory will get amplified downstream, such that the positioning and velocity errors within each follower increase exponentially as described by Swaroop, Huandra, et. al[22].

Thus, It is imperative that in any platooning system, the specified separation is maintained for aggressive maneuvers that require constant change in leader velocity (in the Cartesian or global reference frame). In this paper, we consider a nominal case where all state information in the system is only communicated ad-hoc (i.e. passed from one quadcopter to the next in line). Subsequently, lead quadcopter acceleration is then fed forward to each quadcopter in the fleet, which is observed to reduce the accordion effect. Thus in this paper, the primary objective is to observe the extent to which this lead information reduces the accordion effect, given a robust control design for an ad-hoc communication case.

2.2 Contributions of Work

While each of the following aspects will be covered in detail subsequently in this thesis, the broad contributions/objectives of this work are as listed below

- Modeling a single 6 degree of freedom quadcopter based on contemporary low-cost hardware (with specifications and constraints)
- Low level control of a single quadcopter for accurate attitude and angular rate command following
- High level control of a single quadcopter for accurate path following given a lead trajectory
- Platoon controller design for a fleet of 6 quadcopters (1 leader and 5 followers) using a leader follower approach.
- Analysis of platoon controller performance in the presence of the accordion effect, with and without lead vehicle information in the system, as studied for ground vehicles in [4] (using a nonlinear quadcopter fleet model)

Chapter 3

MATHEMATICAL PRELIMINARIES

3.1 Overview

In order to define the kinematics for a single quadcopter (i.e each agent in the fleet) this section describes a convention that adequately describes each time-varying state of the quadcopter, by defining a body reference frame and Earth reference frame representation (as mentioned in Section 2) and deriving a relationship between the two coordinate frames using conventional Euler rotation matrices.

3.2 Body Reference Frame Representation

The body reference frame describes the motion of the quadrotor with respect to a local origin of the body frame (on the quadrotor itself). In aerial vehicles, the center of mass of the vehicle is often considered the origin point of the body-frame [3,]. The primary reason for using the body-frame representation in this work is that it is more intuitive to consider the orientation or attitude of the vehicle as a sequence of finite rotations around a fixed-point (rotations are considered to be counter-clockwise around the positive axis). This is very useful in context of the low level control for the quadcopter, specifically given the hierarchical (cascade) structure of the angular rate control : through use of an adequate mapping function (from control inputs to desired angles of rotation) we can establish robust control strategies for the body frame inputs, which in turn are based on the top level (Earth frame) reference commands.

3.3 Earth Reference Frame Representation

Given a definition for the body frame a global reference frame must be defined as well the Earth reference From a command perspective, the control law be able to interpret and track a reference position: the most commonly used notation for positioning is the Cartesian system, where the position of the quadcopter is represented by x, y, z position coordinates [insert reference for Cartesian].Both x and y represent the lateral distance from the origin (forming a plane tangent to the earth surface) while z is perpendicular to the surface. This is shown in Figure 3.1 below, with the body frame represented by the b vector, and the Earth frame represented by the e vector.

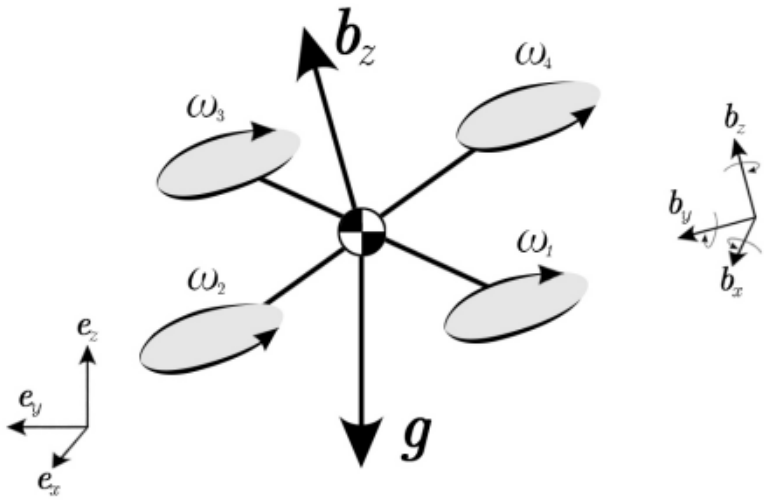

Figure 3.1: Representation of Earth Coordinate Frame and Body Reference Frame

3.4 Attitude Angles and Rotation Matrices (Quadrotor Kinematic Description)

To describe a convention for a quadcopter in the body frame, we require mathematical relations that transform the reference Earth-frame coordinates to the body coordinates, and vice versa. The angles of rotation for the quadcopter are represented by the Euler angle notation, as described below ϕ: Pitch angle is the counter clockwise rotation around the x-axis which is the angle between the b_x and (x, y) plane; θ : Roll angle is the counter clockwise rotation around the y-axis which is the angle between the b_y and (x, y) plane; ψ : Yaw angle is the counter clockwise rotation around the z-axis which is the angle between the projection of b_x in the (x, y) plane and e_x vector;

Within the body frame, the relationship between the angles (low level control outer loop) and desired angular rates (low level control inner loop) is described in 3.1.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sin\theta \\ 0 & \cos\phi & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.1}$$

Based on the above, we define rotation matrices [1, 3,23],which are used to transform the orientation coordinates of a vehicle from the body frame to the earth frame. These matrices are as defined in (3.2)-(3.4)

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.2}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.3}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Using these three matrices, we obtain a function for converting all Euler angles (body frame coordinates) to the Cartesian coordinates (Earth frame reference), shown in (3.5)-(3.7).

$$\Omega = R_x(\phi)R_y(\theta)R_z(\psi)\dot{\Theta} \tag{3.5}$$

where

$$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.6}$$

$$\Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.7}$$

Thus, we have the final coordinate transformation matrix given by (3.8)

$$R_{B->E} = \begin{bmatrix} \cos\psi\cos\theta & -\cos\phi\sin\psi + \cos\psi\sin\phi\sin\theta & -\cos\phi\cos\psi\sin\theta - \sin\phi\sin\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\psi\sin\phi - \cos\phi\sin\psi\sin\theta \\ \sin\theta & -\cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{3.8}$$

Chapter 4

MODELING FOR A SINGLE QUADCOPTER

In the previous section, the attitude representation (Body frame) of the quadcopter is defined, as well as a rotation matrix that establishes a mathematical relationship between the attitude angle and translational (Cartesian) position of the quadcopter in the Earth frame. With these preliminaries defined, the present section deals with the modeling of quadcopter dynamics.

Before proceeding to describe the quadcopter model (in state space form), measured constants for each quadcopter are fixed based on the hardware specifications of the Mark 3 quadcopter model [1]. The parameters are as described in Table 4.1

Table 4.1: Preliminary Hardware Constants for Modelling a Single Quadcopter

SI No.	Parameter	Value
1.	Mass (m)	0.647 Kg
2.	Gravity (g)	9.81 m/s
3.	Arm length(l)	0.125 m

The quadcopter is capable of VTOL (vertical-take off and landing) action, which is important to consider when comparing it with other unmanned aerial vehicle models (such as fixed-wing aircraft). The assumptions for the quadrotor model are listed below.

- The whole quadrotor is a rigid body.
- The quadrotor frame is symmetrical.
- The center of frame matches the center of mass.
- The inertia of motor is small and neglected.
- The range of pitch movement and roll movement is small.

Based on the assumptions above, we can define the system block diagram for the assembly of a single quadcopter unit, as shown in Figure 4.1 below.

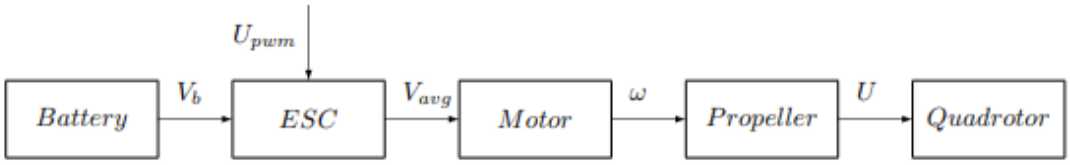

Figure 4.1: Assembly Level Block Diagram for a Single Quadcopter

The modeling of the ESC (Electronic Speed Controller) is complex and is not included within the scope of research. The dynamics of a single rotor thrust (ESC, Motor, Propeller) are modeled as a first-order system as presented in [3] and [11] . Therefore, this section will primarily deal with

1. The relationship between battery voltage and the voltage signal (PWM) put out by the ESC (V_avg), based on system identification tests from observed input-output data.

2. Modeling the motor for the quadcopter, including the actuator dynamics (based on measured hardware parameters for the motor) to obtain desired propeller speed

3. Airframe design - Moment of Inertia about each axis

4. Nonlinear dynamics for the quadcopter (based on the aforementioned points), and linearization at a given equilibrium.

4.1 Battery Voltage Mapping

From the assembly level block diagram (Figure 4.1), the first output signal is generated by the battery, represented as voltage V_b. From [1] and [11], when using

an ESC , the resulting motor rotation speed for a given PWM command (ESC input U_PWM from Figure 3) depends on the battery voltage. The PWM signal is to be generated by the flight controller.

In Figure 4.2, a 3-D curve is plotted between the PWM signal and motor speed value across different values of battery voltage (U_{PWM} vs. ω vs. V_b). This data is obtained through hardware testing.

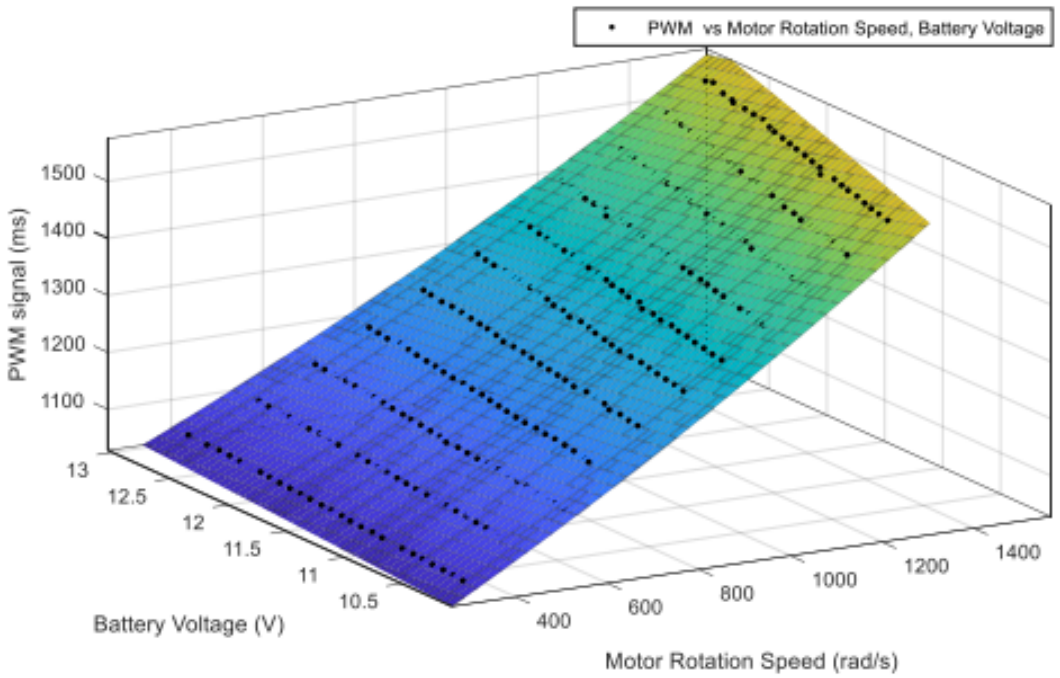

Figure 4.2: Mapping of PWM vs Motor Speed vs. Battery Voltage for Motor ESC Control

From this graph, we obtain a function for U_{PWM} in terms of ω and V_b, given in Equation (4.1) below [1]

$$U_{PWM} = \frac{\omega^2 + 5393\omega + 29960}{1166V_b + 1544} + 895 \tag{4.1}$$

4.2 Actuator Dynamics (Motor Modeling)

As in many conventional quadcopter designs, it is assumed that 4 identical brushless DC (BLDC) motors are mounted on each quadcopter, in order to generate necessary forces and moments. The primary reason for using a BLDC motor is the high torque ratings when compared to other DC motor types. High torques on all 4 motors allows for tight maneuver for the vehicle (as documented in [9],[10]) making BLDC motors a great choice for low-cost quadcopter platforms (great power-to-weight ratio). This is the same choice as in [3].

On each BLDC motor, it is assumed that three bladed propellers are used in the assembly. This design consideration is based on the Snail Propulsion System described in [1]. Figure 4.3 shows the actuator unit used in the Mark 3, which is the basis for modelling actuator dynamics in this paper.

Figure 4.3: Snail Propulsion System for Quadcopter Flight

Using the blade element momentum theory as proposed in [24], we obtain a relationship between the propeller speed, and the overall force and moment generated by the motor. Equation (4.2) shows the basic equation for the BLDC motor, derived using Kirchoff's Law (The propeller model is discussed in the next section).

$$L\frac{di}{dt} = u - RI - K_e\Omega_i \tag{4.2}$$

where L is the motor inductance, u is the voltage input, R is the equivalent motor resistance, K_e is the voltage coefficient and Ω_i is the propeller speed. The torque equillibrium equation for the motor during rotation ia shown in Equation (4.3)

$$L\frac{d\Omega_i}{dt} = k_m i - d\Omega_i^2 \tag{4.3}$$

Given that the BLDC motor is quite small, L is considered negligible. This yields the following approximate motor model shown in Equation (4.4) ([1],[9])

$$\dot{\Omega} = \frac{K_m K_e}{RJ}\Omega_i - \frac{d}{J}\Omega_i^2 + \frac{K_m}{RJ}u \tag{4.4}$$

From equation (4.4), using the Taylor series expansion all high order terms are removed, for a low frequency approximation of the motor model. This gives Equation (4.5) for the Taylor series approximation, and Equation (4.6) for the approximate motor model (Laplace transformation).

$$\dot{\Omega} = -A\Omega_i + Bu + C \tag{4.5}$$

$$\frac{\Omega(s)}{u} = \frac{z_{vol}}{(s+a)} \tag{4.6}$$

In Equation (4.6), while the z_{vol} represents voltage input to the ESC, but in actual practice this is a PWM signal, not a direct voltage. From the curve fitting shown in

the previous section, we have a mapping from input voltage to the PWM signal on the ESC, and in turn from the PWM signal to the desired motor speed. Thus, we require a transfer function from the desired motor speed, to the actual motor speed based on the actuator dynamics described in this section. This transfer function is considered as a first order low pass filter, shown in Equation (4.7)

$$\frac{\Omega(s)}{\Omega^*(s)} = \frac{a}{(s+a)} \tag{4.7}$$

Where $\Omega^*(s)$ is the desired motor speed, and $\Omega(s)$ is the actual motor speed.

4.3 Airframe Design

Having defined an appropriate motor model for the system, this section describes the parameters considered when modeling the airframe for each quadcopter. One of the most important components of the quadcopter hardware is the propeller, and the suitability of different propellers (in terms of blade count, material, e.t.c) is covered in literature such as [25]. In this design, as mentioned in the previous section, we have considered the use of a three bladed propeller made of carbon fiber, with a 250 mm carbon fibre frame chosen for the body of the quadcopter.

From [26], we obtain the relationship between propeller rotation speed ω, and the overall force (F_m) and moment (τ_m) generated by the motor. This is as shown in equations (4.8) and (4.9).

$$F_m = k_f \omega^2 \tag{4.8}$$

$$\tau_m = k_\tau \omega^2 \tag{4.9}$$

where k_f is the thrust coefficient, and k_τ is the moment coefficient of the motor. From the hardware testing in [1], these coefficients are obtained using curve fitting,

using the Dynamometer Series 1580 test stand to measure thrust and torque in real time for different motor speeds. Figures 4.4 and 4.5 show the curves plotted using data from the test stand.

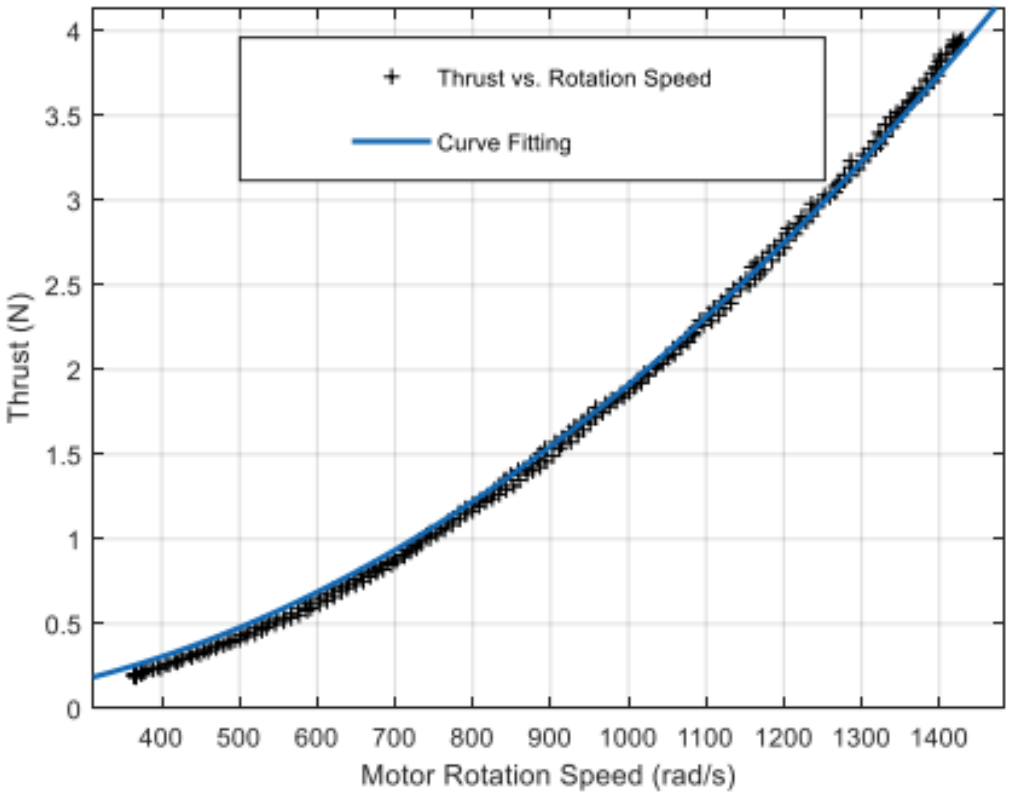

Figure 4.4: Thrust (Newton) vs Motor Rotation Speed (rad/s) for Single BLDC Motor

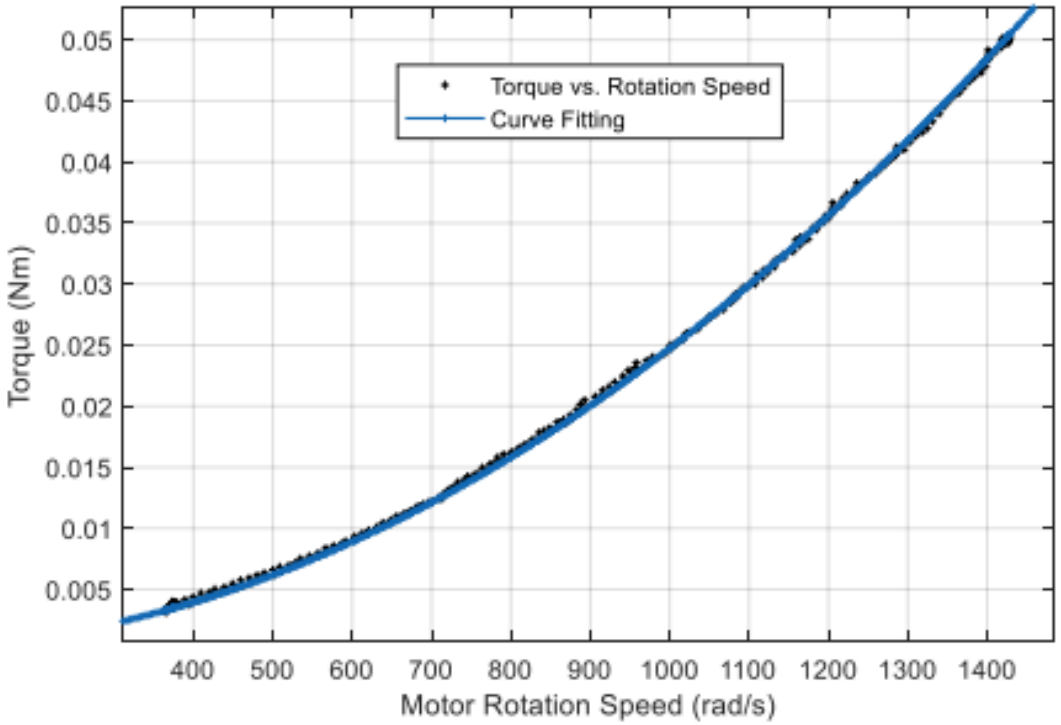

Figure 4.5: Torque (Newton-meter) vs Motor Rotation Speed (rad/s) for Single BLDC Motor

From the curve fitting shown in Figure 4.3 and 4.4, the value of k_f is $1.91 \times 10^{-6} N/rad^2$ and the value of k_τ is $2.47 \times 10^{-8} Nm/rad^2$. From the PWM curve fitting shown in section 4.1, we have a relationship between the voltage input to the PWM input as well as from the PWM input to motor rotation speed. Thus, with the mapping from rotation speed to force shown above, we obtain a fully described model for the quadrotor kinematics. The actuator constant a is chosen to be 9.79, as it shows 92% fitness with measured test stand data (using System Identification experiment)[1].

Finally, prior to modeling quadcopter dynamics, we require the moment of inertia of the airframe about the X,Y and Z axes. The bifilar pendulum experiment [27] is used to evaluate these parameters, as shown in [1]. In the experiment, the moment of

inertia about each axis is computed by measuring the twist oscillation period for the airframe about the respective axis, with the experiment setup as shown in Figure 4.5

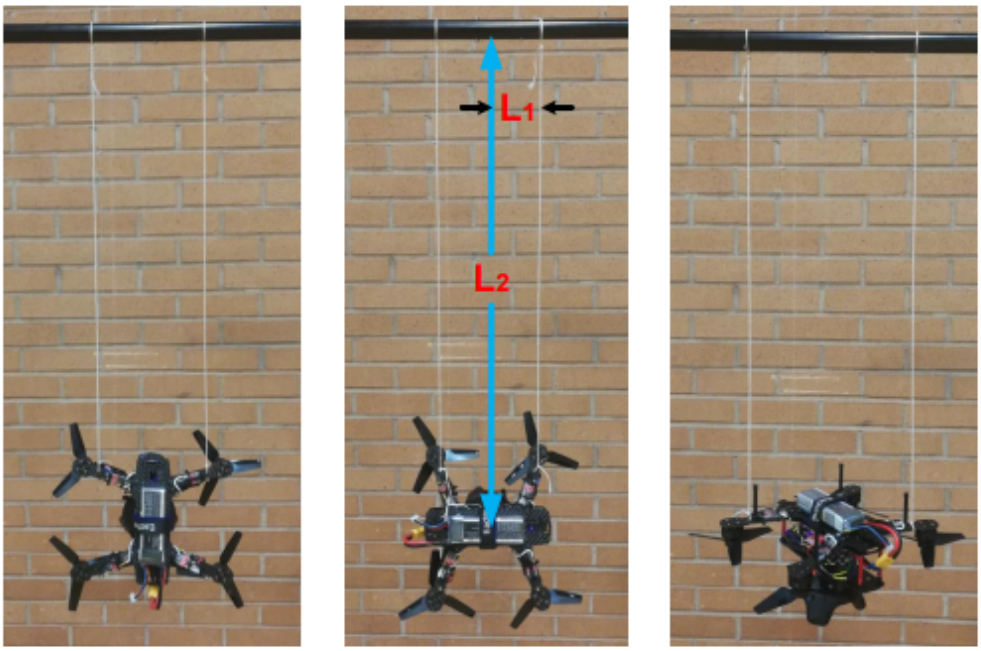

Figure 4.6: Bifilar Pendulum Experiment for Moment of Inertia Estimation (Single Quadcopter

With T as the time period of one twist oscillation, m as the mass of the quadrotor in kilogram, $L1$ and $L2$ as shown in Figure 4.5, and $g = 9.81$ (acceleration due to gravity), the formula used to find the moment of inertia is shown in Equation (4.10)

$$J = \frac{mgT^2L1^2}{4\pi^2 L2} \tag{4.10}$$

The resulting moment of inertia values are shown in Table 4.1

Table 4.2: Moment of Inertia Values for Quadcopter Airframe

SI No.	Parameter	Value
1.	J_x	0.0019 kgm^2
2.	J_y	0.00195 kgm^2
3.	J_z	0.00369 kgm^2

4.4 Rigid Body Dynamics for the Quadcopter

The dynamic model for the quadcopter describes the translational movement of the quadcopter in the global (Earth) frame of motion, when under the influence of external forces. Two primary design equations are considered to model quadcopter translational dynamics. The first is shown in Equation (4.11)[1]

$$m\ddot{\zeta} = T(R_{B->E})^T e_3 - mg \tag{4.11}$$

where m is the mass of the quadrotor (in kg), $\zeta = [x, y, z]^T$ is the Cartesian position of the quadrotor in the global frame, T is the total upward force in the body-frame (or the total thrust), and $g = [0, 0, 9.81]^T$ represents the gravity vector (downward acceleration). The second design equation is shown in Equation (4.12)[3].

$$J\dot{\Omega} = M - \Omega \times J\Omega \tag{4.12}$$

where $J = diag[J_{xx}, J_{yy}, J_{zz}]$ is the inertia matrix (described in section 4.3), $\Omega = [p, q, r]$ represents the angular rates of the quadcopter in all three axes, and $M = [M_x M_y M_z]^T$ represents the moment of the quadcopter about each axis.

When the quadcopter kinematic relations (derived in Section 3) are substituted into equations (4.11) and (4.12), we obtain the nonlinear translational dynamics for a single quadrotor, shown in equation (4.13)

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{T}{m}(-\cos\phi\cos\psi\sin\theta - \sin\phi\sin\psi) \\ \frac{T}{m}(\cos\psi\sin\phi - \cos\phi\sin\psi\sin\theta) \\ \frac{T}{m}(\cos\phi\cos\theta) - g \\ \frac{1}{Jxx}(M_x + J_{yy}qr - J_{zz}qr) \\ \frac{1}{Jyy}(M_y - J_{xx}pr + J_{zz}pr) \\ \frac{1}{Jzz}(M_z + J_{xx}pq - J_{yy}pq) \end{bmatrix} \tag{4.13}$$

The full rigid body dynamics (including integrator states) are shown in Equation 4.14.

$$X = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{V}_{\mathrm{x}} \\ \dot{V}_{\mathrm{y}} \\ \dot{V}_{\mathrm{z}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} V_{\mathrm{x}} \\ V_{\mathrm{y}} \\ V_{\mathrm{z}} \\ \frac{T}{m}(-\cos(\psi)\sin(\theta)\cos(\phi) - \sin(\psi)\sin(\phi)) \\ \frac{T}{m}(\cos(\psi)\sin(\phi) - \sin(\psi)\sin(\theta)\cos(\phi)) \\ \frac{T}{m}(\cos(\psi)\cos(\theta)) - g \\ p + q\sin(\psi)\tan(\theta) - r\cos(\phi)\tan(\theta) \\ q\cos(\phi) + r\sin(\phi) \\ -q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)} \\ qr\frac{J_{\mathrm{y}} - J_{\mathrm{z}}}{J_{\mathrm{x}}} + \frac{\tau\phi}{J_{\mathrm{x}}} \\ pr\frac{J_{\mathrm{z}} - J_{\mathrm{x}}}{J_{\mathrm{y}}} + \frac{\tau\theta}{J_{\mathrm{y}}} \\ pq\frac{J_{\mathrm{x}} - J_{\mathrm{y}}}{J_{\mathrm{z}}} + \frac{\tau\psi}{J_{\mathrm{z}}} \end{bmatrix} \tag{4.14}$$

Thus, the rigid body dynamics can be modeled as a state space system, with the form shown in Equation 4.15.

$$\dot{X_{rig}} = f(\dot{X_{rig}}, U_{rig}) \tag{4.15}$$

where X_{rig} is given by Equation 4.14, and U_{rig} is as shown in Equation 4.16

$$U = \begin{bmatrix} T \\ \tau\phi \\ \tau\theta \\ \tau\psi \end{bmatrix} \tag{4.16}$$

The quadcopter is assumed to have an equilibrium/operating point near hover position where the thrust magnitude is equal to the force of gravity, and the rotation angles are all close to zero. Substituting these into equation 4.14, we obtain the linearized dynamics shown in Equation 4.17 (same choice in [1] and [2])

$$X_{\text{lin}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{V}_{\text{x}} \\ \dot{V}_{\text{y}} \\ \dot{V}_{\text{z}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} V_{\text{x}} \\ V_{\text{y}} \\ V_{\text{z}} \\ -g\theta \\ g\phi \\ \frac{T}{m} - g \\ p \\ q \\ r \\ \frac{\tau\phi}{J_{\text{x}}} \\ \frac{\tau\theta}{J_{\text{y}}} \\ \frac{\tau\psi}{J_{\text{z}}} \end{bmatrix} \tag{4.17}$$

Thus, with the linearization at hover, the dynamics of a single quadcopter system can be represented as a nominal state space system as shown in Equations 4.18 through 4.20.

$$\dot{X}_{\text{rig}} = AX_{\text{rig}} + BU_{\text{rig}} \tag{4.18}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.19}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/m \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/J_x & 0 & 0 \\ 0 & 0 & 1/J_y & 0 \\ 0 & 0 & 0 & 1/J_z \end{bmatrix} \tag{4.20}$$

Chapter 5

LOW LEVEL CONTROL OF A SINGLE QUADCOPTER- ANGULAR RATE AND ANGLE TRACKING CONTROL

As in [1] and [2], a feedback control system is designed for low level control of the attitude of the quadcopter. The attitude of each quadcopter is given by it's Euler angle [ϕ, θ and ψ] orientation in the body frame. Therefore, reference angles must be tracked very accurately, given that the motion of a quadcopter in the Earth reference frame is entirely based on the direction of tilt, and in turn the rate of change of orientation angles in the body reference frame.

Robust control of attitude can be achieved by controlling the angular rate (inner loop control) and the angle (outer loop) in a cascade control scheme, as detailed in previous literature [12]. The first important aspect of control here is that the flight controller is digital, and thus the system must account for sampling time. From contemporary research in [3], we choose the sampling time of 0.0025 seconds (based on the hardware specification of a 400 Hz clock, on the Teensy 3.0 board).

The angular rate control system is modeled as a classical feedback loop as shown in Figure 5.1 below, where each input corresponds to a desired propeller angular rate [3].

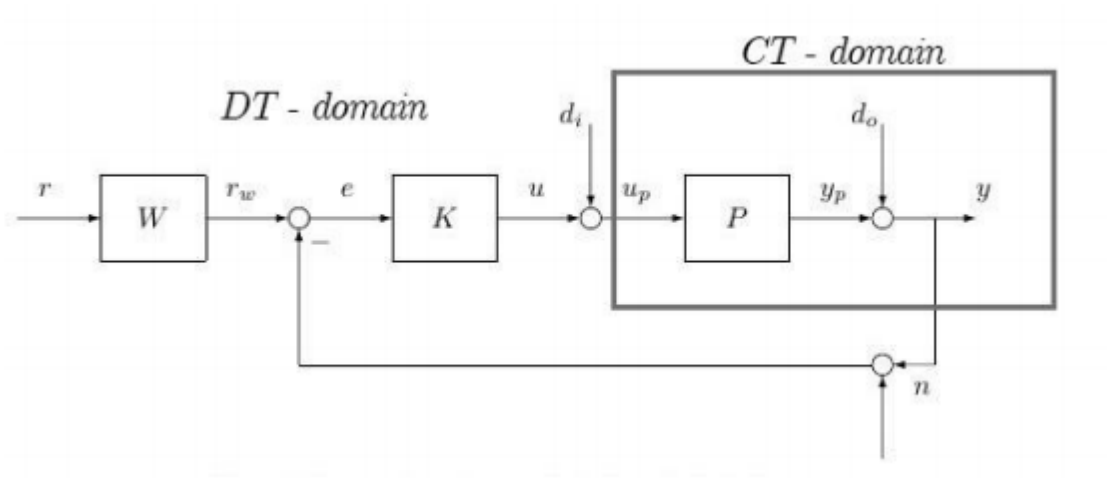

Figure 5.1: Inner Loop Block Diagram (Angular Rate Command Following)

With control inputs corresponding to desired angular rates, the low level angular rate control system can now be modeled.

5.1 Low Level Inner Loop Control: Angular Rate Command Following

The primary objective of the angular rate control system is to track desired p,q and r angular rate commands. From Chapter 4, the angular rate plant can be represented by the following transfer function matrix

$$P_{rate} = \begin{bmatrix} \frac{5151.28}{s(s+9.79)} & 0 & 0 \\ 0 & \frac{5011.26}{s(s+9.79)} & 0 \\ 0 & 0 & \frac{2653.55}{s(s+9.79)} \end{bmatrix} \tag{5.1}$$

Where $P_p = P_{rate}(1,1)$, $P_q = P_{rate}(2,2)$ and $P_r = P_{rate}(3,3)$. The frequency response for each approximated plant transfer function are shown in Figure 5.2.

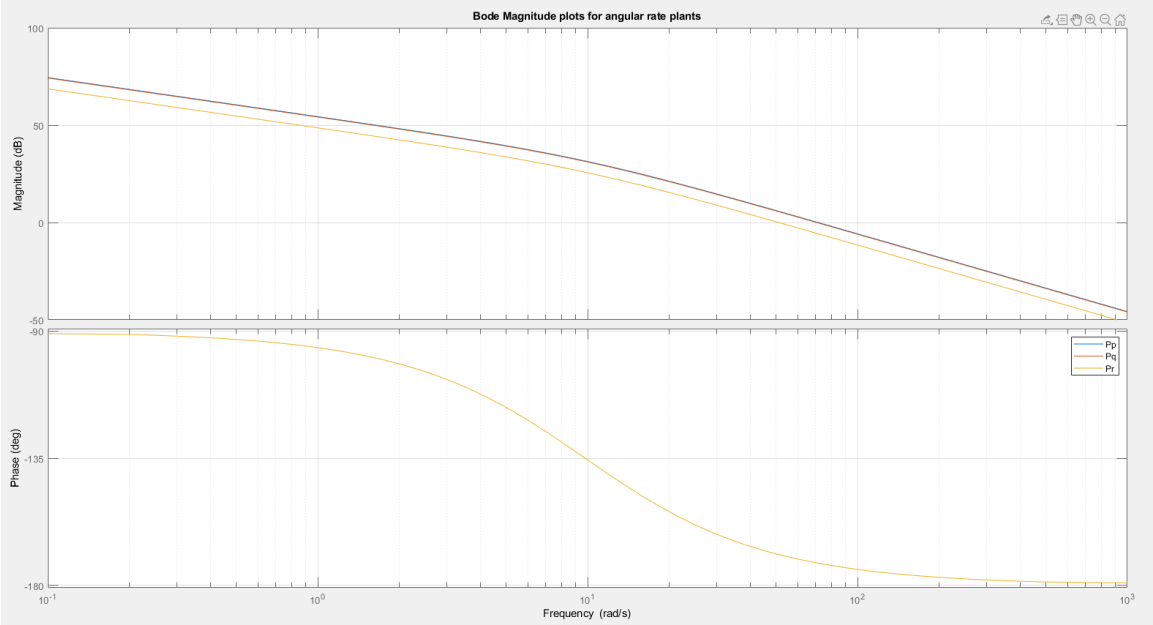

Figure 5.2: Low Level Inner Loop - Plant Frequency Response

While the pitch and roll angular rates (p, q) show similar behavior at all frequencies , it is noted (as in [1]) that yaw angular rate control requires more torque to generate equivalent angular acceleration. This is because the magnitude of the P_r plant (at low frequencies) is less than the magnitudes of P_p and P_q at those frequencies; this is due to the fact that the actuator torque coefficient of the BLDC motor is much smaller than it's combined thrust coefficient. As such, the quadcopter's moment of inertia about the Z-axis (J_{zz}) is nearly double that of the moment of inertia about the X-axis $(J_{xx}$ and Y axis (J_{yy}). Thus, the bandwidth of designed yaw rate control is chosen to be less than half of the bandwidth for pitch and roll control. This is the same choice in [1],[2] and [3]

Using the linearized dynamics at hover (from Chapter 4), we obtain generalized plant diagrams for angular rate feedback control, shown in Figure 5.3.

It is noted from [3] that the first propeller harmonic begins at 500 radians per second, and therefore the maximum bandwidth we can design for is 50 rad/s (i.e at

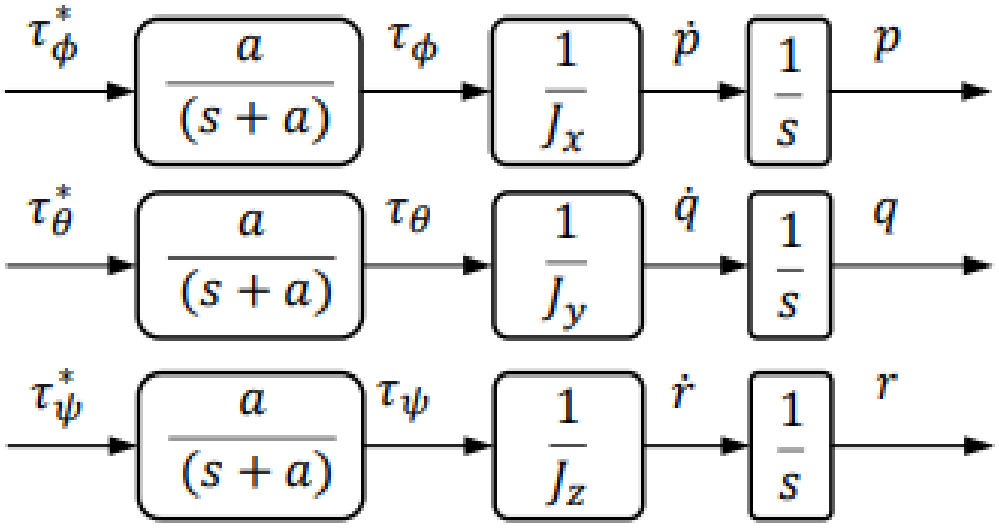

Figure 5.3: Angular Rate Plant - Block Diagram Visualization

least a decade below the first harmonic). Based on this specification, a bandwidth trade study for the angular rate plants is conducted, sweeping the unity gain crossover frequency (ω_g) from 7 rad/s to 35 rad/s. The phase margin is specified at 60 degrees, adequate for good closed loop command following properties. [8]

Bandwidth Sweep of Inner Loop - Comparisons and Observations

With phase margin fixed at 60 degrees, a bandwidth trade study is performed, by varying the unity gain crossover ω_g, to observe the effect of controller bandwidth on inner loop command following.

In designing the controller, the following assumptions are made.

1. The frequency of the digital controller is 400 Hz

2. A PD controller design is considered sufficient for accurate $[p, q, r]$ command following. This is based on common use of the PD design structure in literature[1].

3. A high frequency roll off term is included in each angular rate controller, to reduce the effects of high frequency noise on the angular control systems. Based on choices in contemporary literature [1][3] , this roll-off term should correspond to a frequency at least 10 times the controller bandwidth, in order to adequately attenuate high frequency noise

Based on the assumptions above, the controller structure is modeled as shown in Equation 5.2

$$K_{angrate} = g(s+z)\left(\frac{r}{s+r}\right)^2 \tag{5.2}$$

where r is the high frequency roll off term, chosen to be a decade above the w_g parameter.

The bandwidth sweep results are presented in the following subsections, for P and Q angular rate control respectively.

Plots and Key Observations for P Angular Rate Control

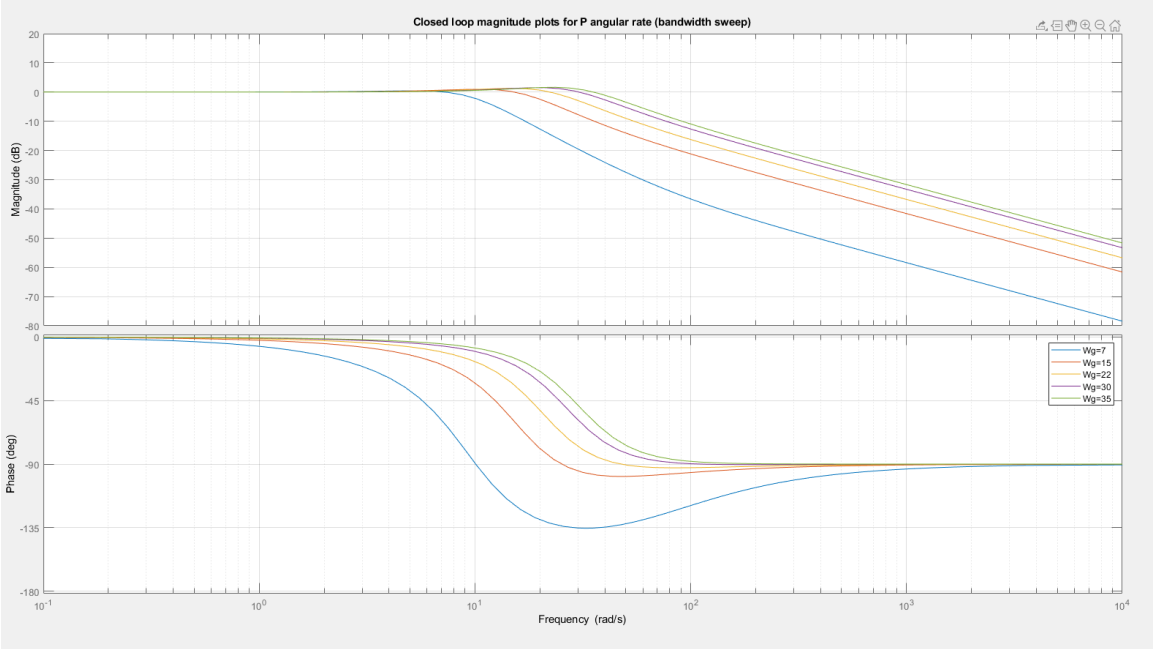

Figure 5.4: Closed Loop Frequency Response for P Command Following (Bandwidth Sweep)

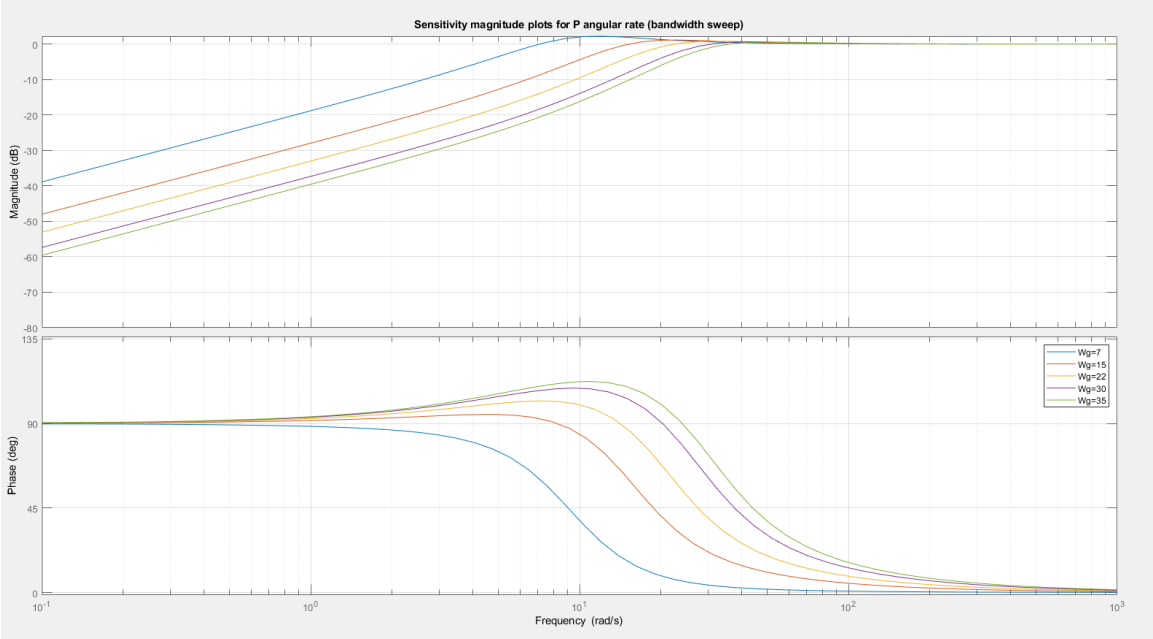

Figure 5.5: Closed Loop Sensitivity Response for P Command Following (Bandwidth Sweep)

From the plots shown in the previous page, the following inferences are made

- As ω_g is increased from 7 rad/s to 35 rad/s, the bandwidth of the system improves, as expected.
- It is noted that the frequency curve shows marginal improvement in high frequency response (i.e at 10 to 100 radians per second) from $\omega_g = 7$ rad/s to $\omega_g = 15$ rad/s,which is not as prevalent when increasing ω_g beyond 15 rad/s
- The peak sensitivity values (for each response) decrease as the bandwidth is increased. This means that the controller's input disturbance rejection properties improve, provided higher bandwidth.
- It is observed that the peak complementary sensitivities increase slightly with bandwidth. This is summarised in Table 5.1 below, with documented peak sensitivities and complementary sensitivities

Table 5.1: Bandwidth Sweep Results - Peak Sensitivity and Complementary Sensitivity Values for P Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**	**Peak comp. sensitivity (dB)**
1.	7	2.21	0.37
2.	15	1.46	0.818
3.	22	1.56	1.01
4.	30	1.93	1.03
5.	35	2.18	0.96

Figure 5.6 shows the command following response for P angular rate control.

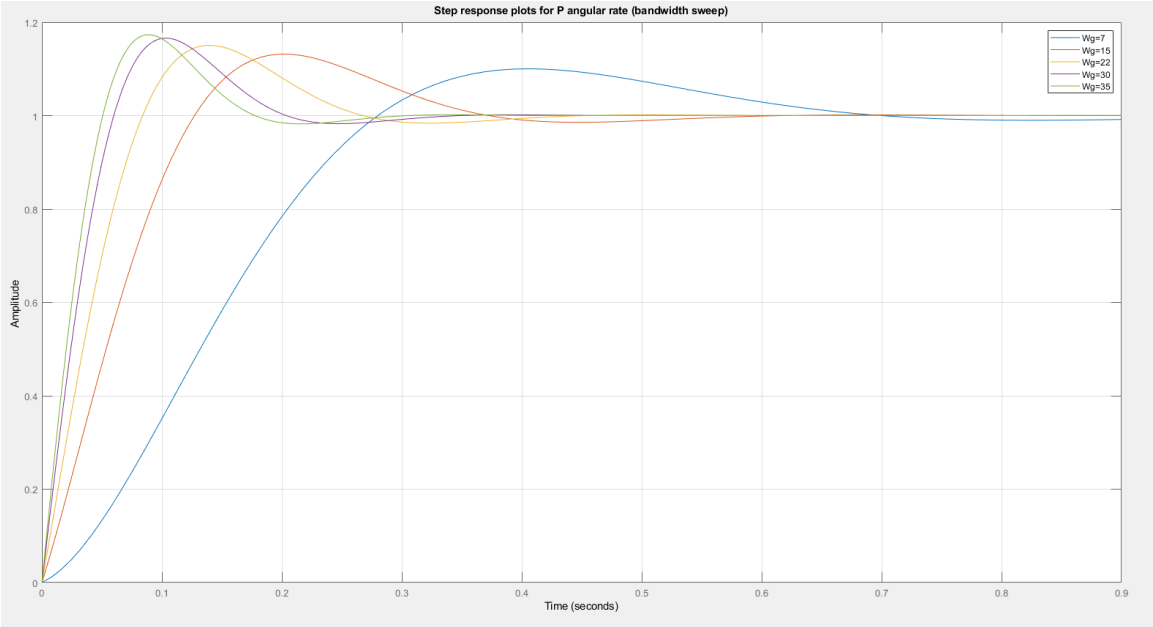

Figure 5.6: P Angular Rate Command Following- Time Domain Response

From the plots above, it can be inferred that even as bandwidth is increased beyond 30 rad/s, the peak overshoot and sensitivity remain relatively low. However, it is noted from [2] that the first propeller harmonics begin around 500 radians/sec. It is good practice to maintain a cut off frequency at least a decade below the first harmonic frequency.

With this condition in mind from the hardware, the design with $\omega_g = 22$ rad/s is chosen. It features a good balance between peak sensitivity (1.56dB), and peak complementary sensitivity (1.01dB), and the peak overshoot is around 12%, which is fairly low. This is the same choice for bandwidth as in [1] and [5].

Plots and Key Observations for Q Angular Rate Control

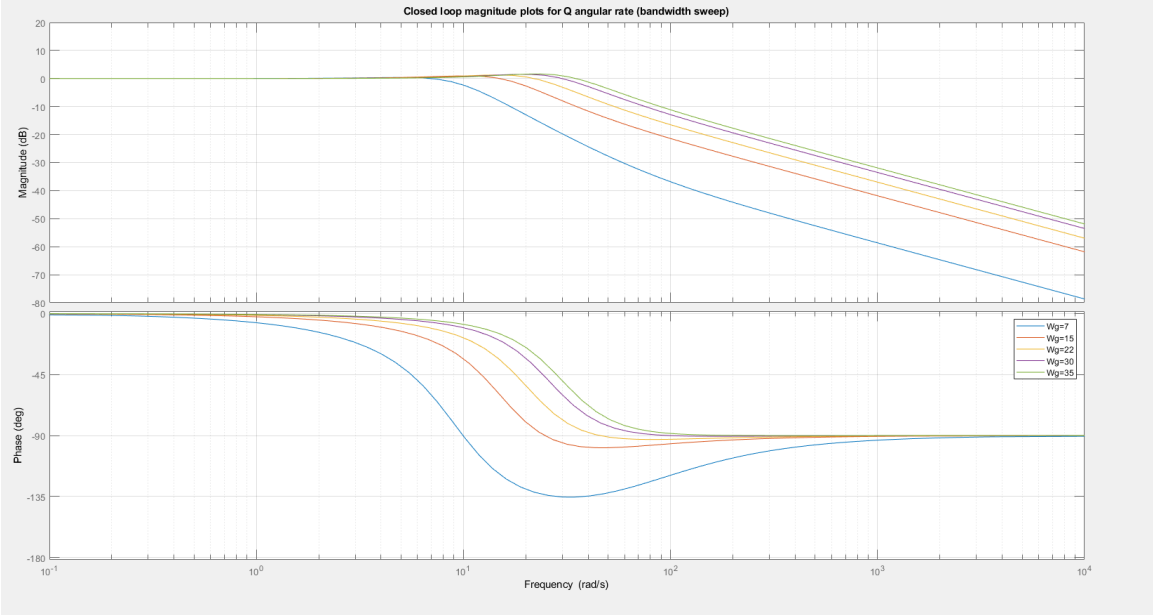

Figure 5.7: Closed Loop Frequency Response for Q Command Following (Bandwidth Sweep)

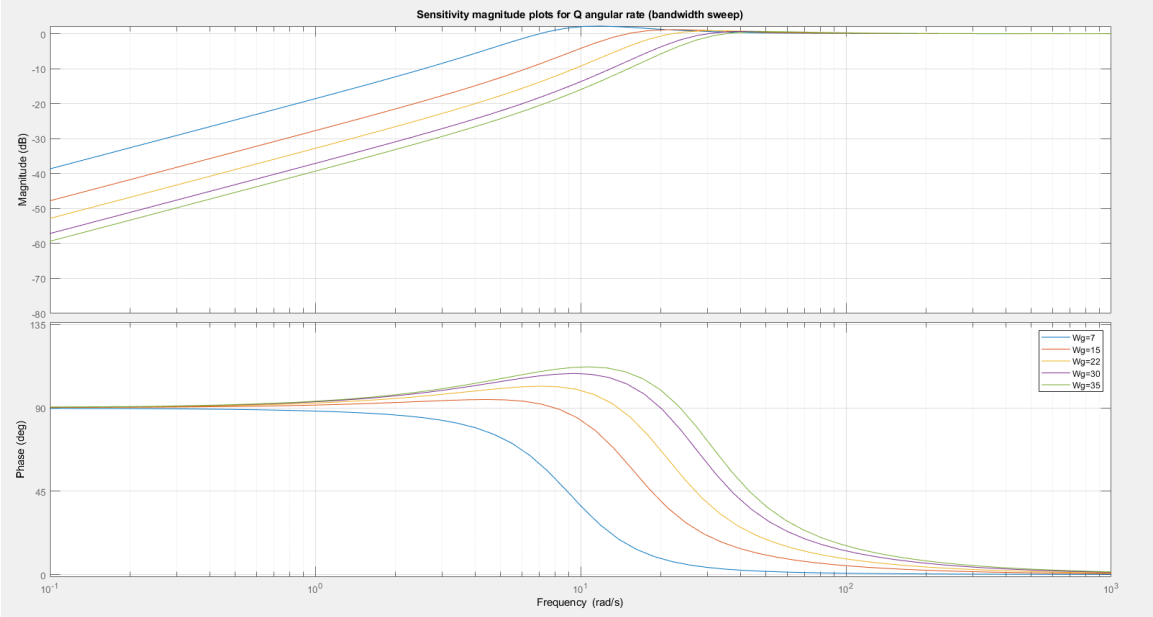

Figure 5.8: Closed Loop Sensitivity Response for Q Command Following (Bandwidth Sweep)

From the plots above, the following inferences are made

- As ω_g is increased from 7 rad/s to 35 rad/s, the bandwidth of the system improves, as expected.
- As in P angular rate control, the high frequency response is marginally improved (i.e at 10 to 100 radians per second), when increasing ω_g from 7 rad/s to 15 rad/s,which is not as prevalent as when ω_g is increased beyond 15 rad/s
- As presented in Table 5.2 below, the peak sensitivities and complementary sensitivities for Q angular rate control show nearly identical behavior with the P angular rate control values, on sweeping controller bandwidth.

Table 5.2: Bandwidth Sweep Results - Peak Sensitivity and Complementary Sensitivity Values for Q Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**	**Peak comp. sensitivity (dB)**
1.	7	2.21	0.372
2.	15	1.46	0.82
3.	22	1.56	1.01
4.	30	1.93	1.02
5.	35	2.18	0.964

Figure 5.9 shows the command following response for Q angular rate control.

From the plots above, it can be inferred that even as bandwidth is increased beyond 30 rad/s, the peak overshoot and sensitivity remain relatively low. However, it is noted from [2] that the first propeller harmonics begin around 500 radians/sec.

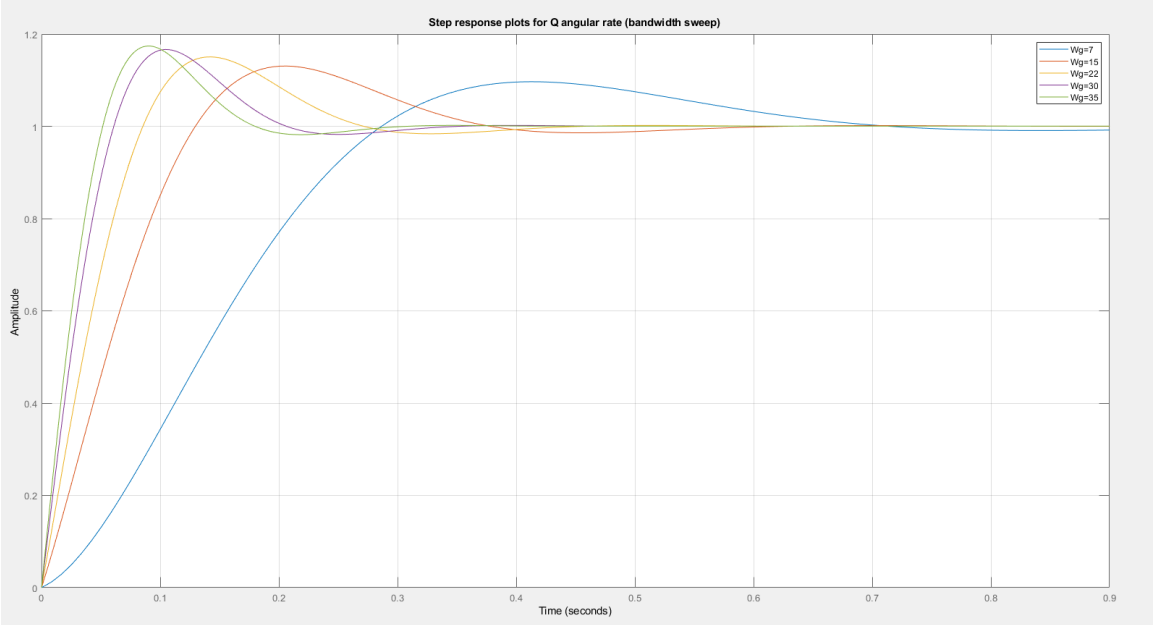

Figure 5.9: Q Angular Rate Command Following- Time Domain Response

It is good practice to maintain a cut off frequency at least a decade below the first harmonic frequency.

As in the design for P angular rate control, $\omega_g = 22$ rad/s is chosen. While the 35 rad/s design has slightly lower peak complementary sensitivity, it is too high for the first propeller harmonic. The $\omega_g = 22$ rad/s offers a good balance between peak sensitivity (1.56dB) and peak complementary sensitivity (1.01dB)

5.2 Low Level Outer Loop Control - Attitude Command Following

From the previous subsection, the inner loop angular rate control is fixed at 22 rad/s for P and Q angular rate command following. For R angular rate control, the bandwidth is fixed at 5 rad/s in accordance with the hardware requirement that the yaw rate bandwidth must be significantly less than pitch and roll rate bandwidth; this is for the following reasons

- The torque for yaw angular movement requires much more maximum motor speed than that for pitch angular movement, as the actuator torque coefficient is much smaller than the combined actuator thrust factor
- Moment of inertia in x and y body axis is smaller than that in z body axis.

To now address attitude (outer-loop) control in this design, it is assumed that all quadcopters perform maneuvers using robust pitch and roll control alone,and so *psi* dynamics are addressed separately in Chapter 6 as a part of high level control.

A proportional controller sufficient for attitude (Euler) angle command following, given a well designed inner loop (angular rate) control system [1]. Typically, in cascaded control system (inner-outer loop hierarchy) the outer loop is slower than the inner loop; at system limits, the inner loop is typically designed to be 2-4 times faster than the outer loop [28].

The cascade control scheme is shown in Figure 5.10, for phi angle control. Theta angle (Pitch) control is implemented using identical proportional controllers, as p and q rate control are nearly identical in nature.

From the previous section, the inner loop bandwidth is fixed at 22 rad/s for P and Q angular rate controllers. With this kept constant, a bandwidth sweep is performed for varying K_ϕ shown in Figure 5.10. It is to be noted that θ (pitch) angle control is

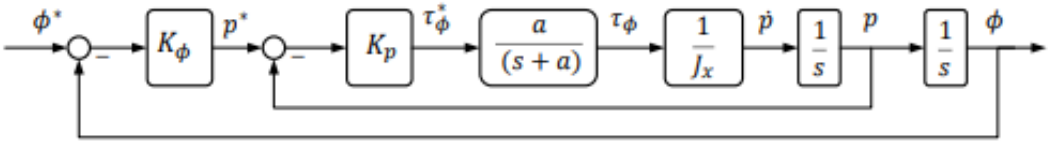

Figure 5.10: Attitude Command Following - Cascade Control Scheme

modeled identical to the structure to 5.10, with K_θ=K_ϕ; this is due to the fact the P and Q angular rate dynamics are virtually identical as well.

Assuming that the system is being tested at its limits, the inner loop should be 4 times faster than the outer loop. With this design requirement in mind, the minimum possible outer loop bandwidth at 5.5 rad/s, and the maximum possible outer loop bandwidth is 11 rad/s. With these considered boundary values, the bandwidth sweep is performed by setting K_ϕ =[5,6,8,10,11], (corresponding ω_g values are [5.1 6.2 8.6 11 12] rad/s).

Plots and Key Observations for Roll Angle (ϕ) Command Following

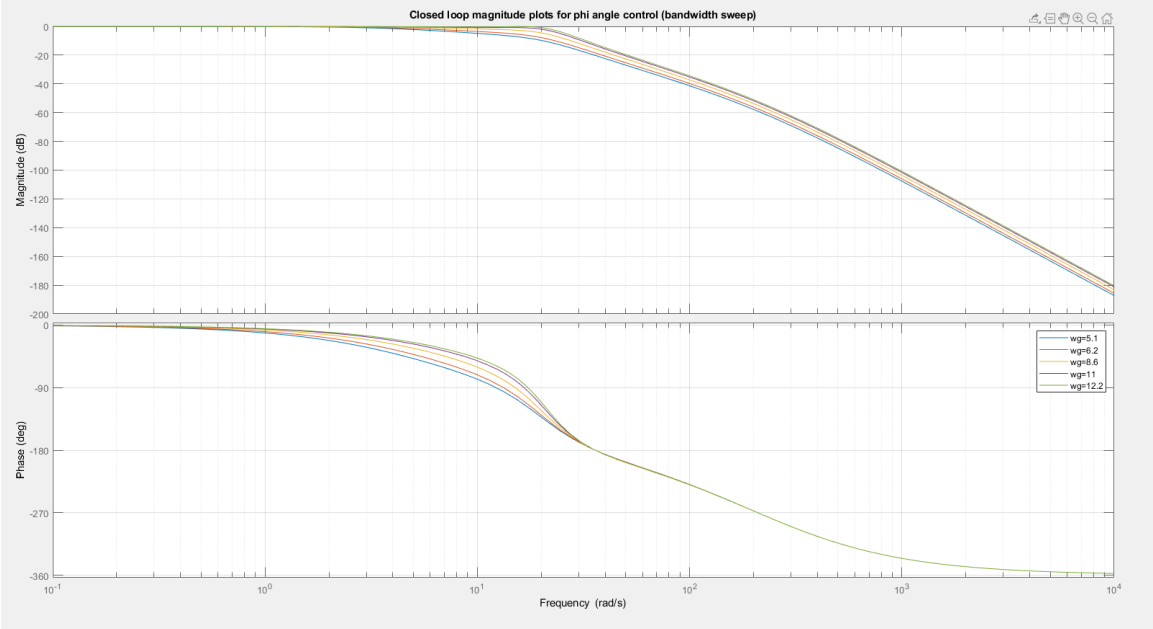

Figure 5.11: Closed Loop Frequency Response for Roll (ϕ) Angle Command Following (Bandwidth Sweep)

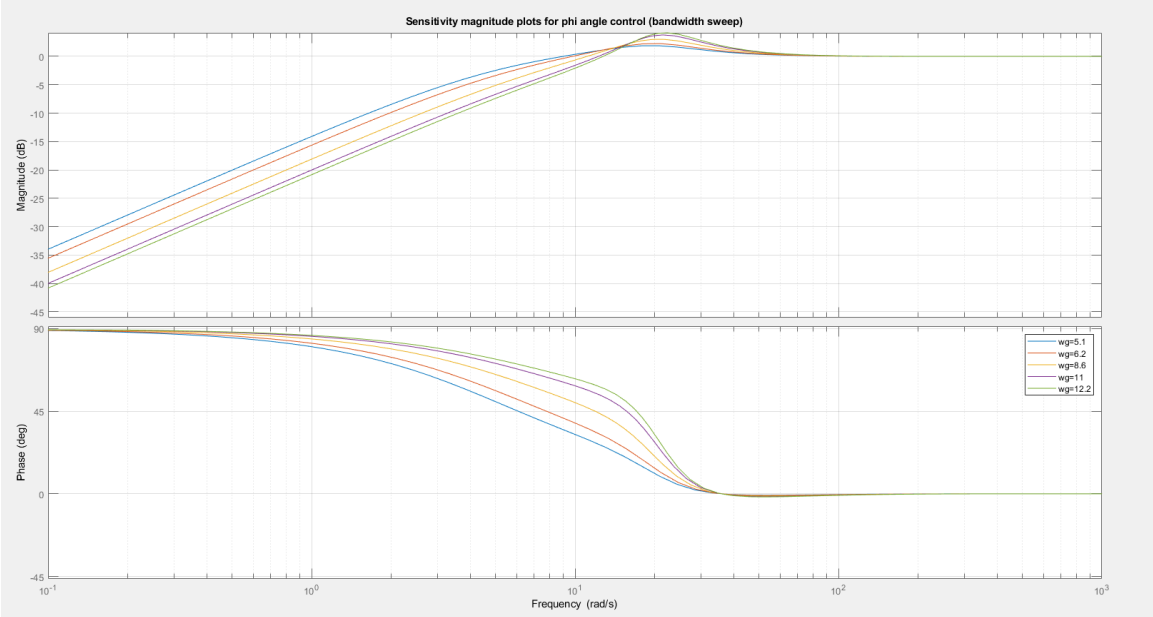

Figure 5.12: Closed Loop Sensitivity Response for Roll (ϕ) Angle Command Following (Bandwidth Sweep)

From the plots above, the following inferences are made

- As ω_g is increased from 5.1 rad/s to 12 rad/s, the bandwidth of the system improves, as expected.

- The peak sensitivity values (for each response) increases as the bandwidth is increased. This means that the angle controller's input disturbance rejection properties worsen at higher bandwidth, unlike the angular rate controller.

- It is observed that the peak complementary sensitivities remain below 0dB (constant with bandwidth increment. The sensitivity values are summarised in Table 5.3 below.

Table 5.3: Bandwidth Sweep Results - Peak Sensitivity and Complementary Sensitivity Values for ϕ Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**
1.	5.1	1.87
2.	6.2	2.25
3.	8.6	3
4.	11	3.78
5.	12	4.15

Figure 5.13 shows the command following response for phi angle control.

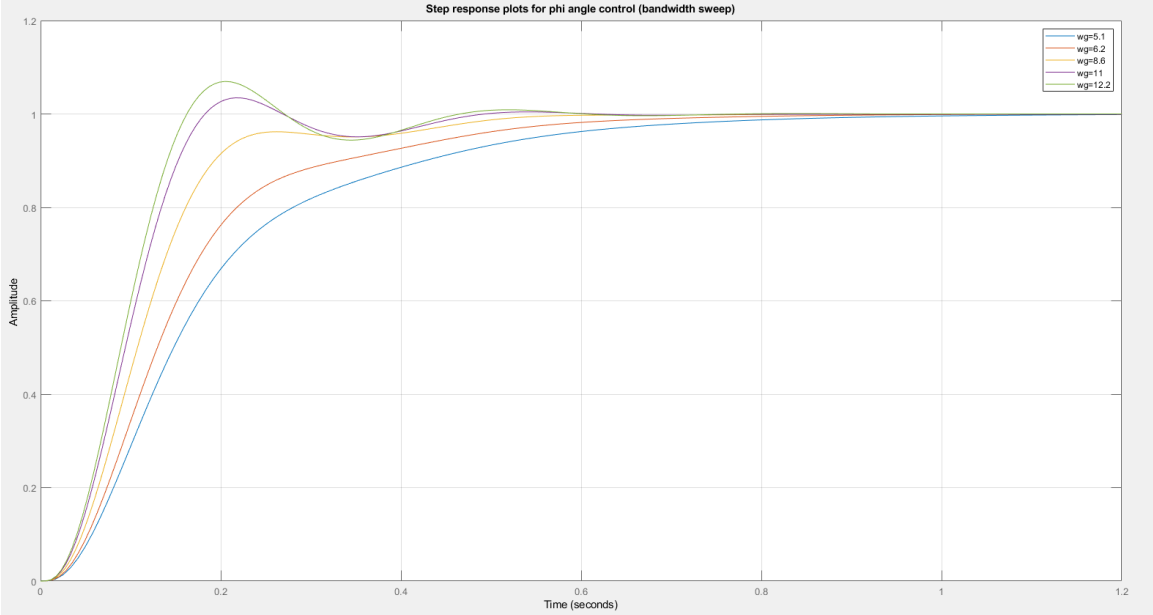

Figure 5.13: Roll (ϕ) Angle Command Following - Time Domain Response

From Figure 5.13, it is observed that the step response shows no overshoot, until the controller bandwidth is increased to 11 rad/s. At this stage, there is an overshoot in the response. In general, settling time reduces with increasing bandwidth of controller, although there is no significant difference in the settling time between ω_g=11 rad/s and ω_g=12 rad/s.

Plots and Key Observations for Pitch Angle (θ) Command Following

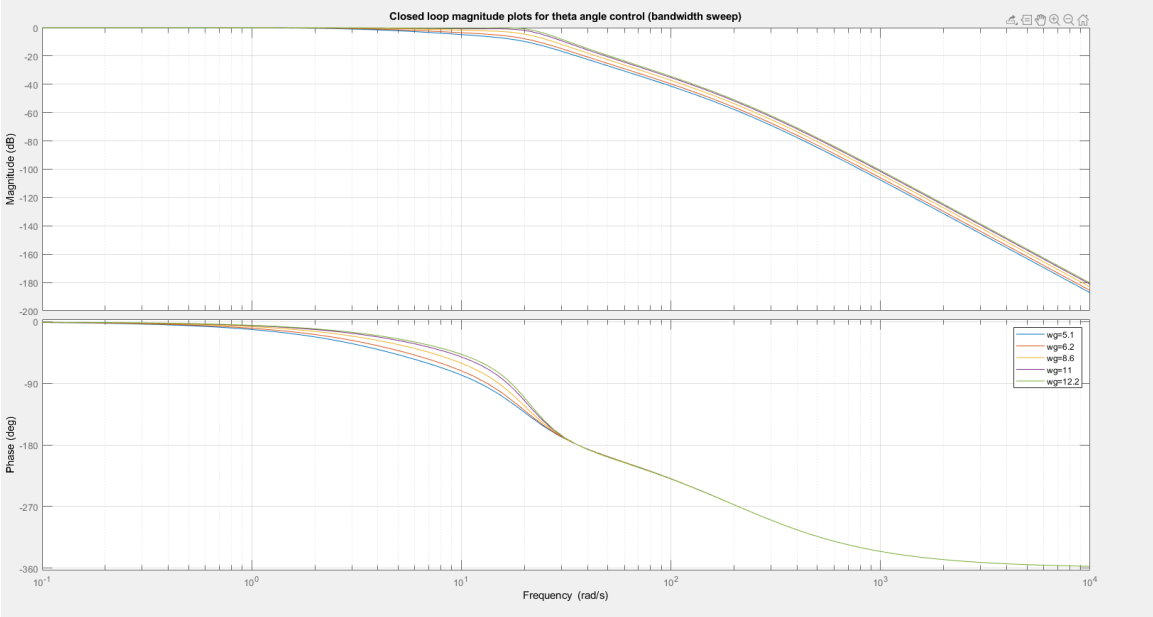

Figure 5.14: Closed Loop Frequency Response for Pitch (θ) Angle Command Following (Bandwidth Sweep)

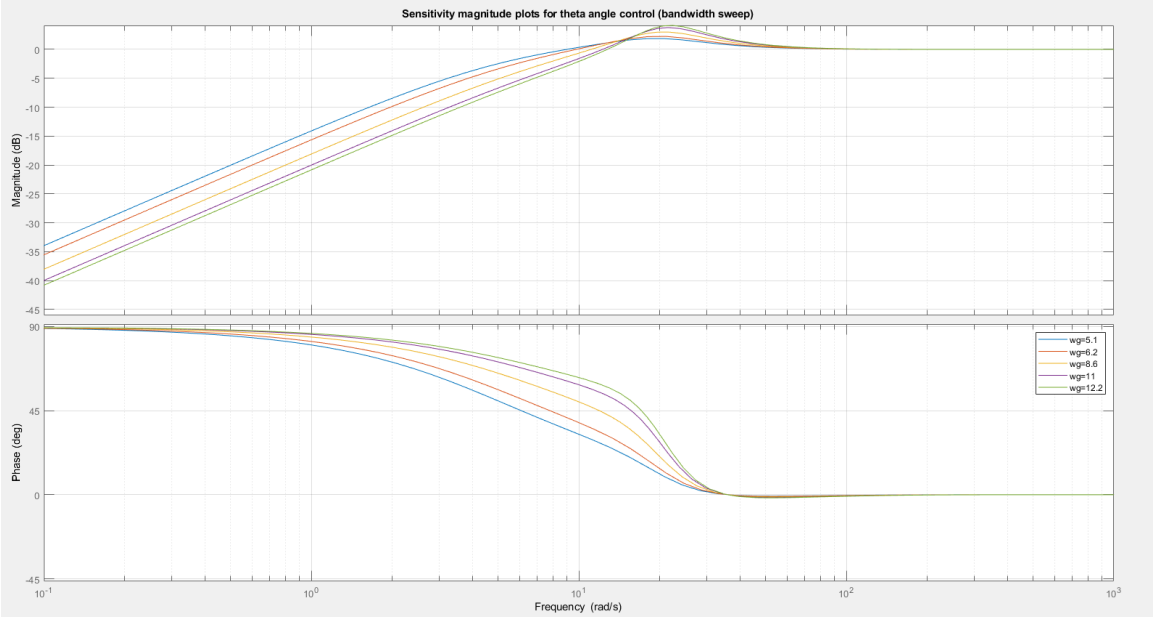

Figure 5.15: Closed Loop Sensitivity Response for Pitch (θ) Angle Command Following (Bandwidth Sweep)

From the plots above, the following inferences are made

- As ω_g is increased from 5.1 rad/s to 12 rad/s, the bandwidth of the system improves, as expected.

- The peak sensitivity values (for each response) increases with bandwidth as in the Phi control response

- It is observed that the peak complementary sensitivities remain below 0dB (constant with bandwidth increment. The peak sensitivity values for Theta angle control are identical to those of Phi angle control, and are summarised in Table 5.4 below.

Table 5.4: Bandwidth Sweep results - Peak Sensitivity Values for θ Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**
1.	5.1	1.87
2.	6.2	2.25
3.	8.6	3
4.	11	3.78
5.	12	4.15

Figure 5.16 shows the command following response for Theta angle control.

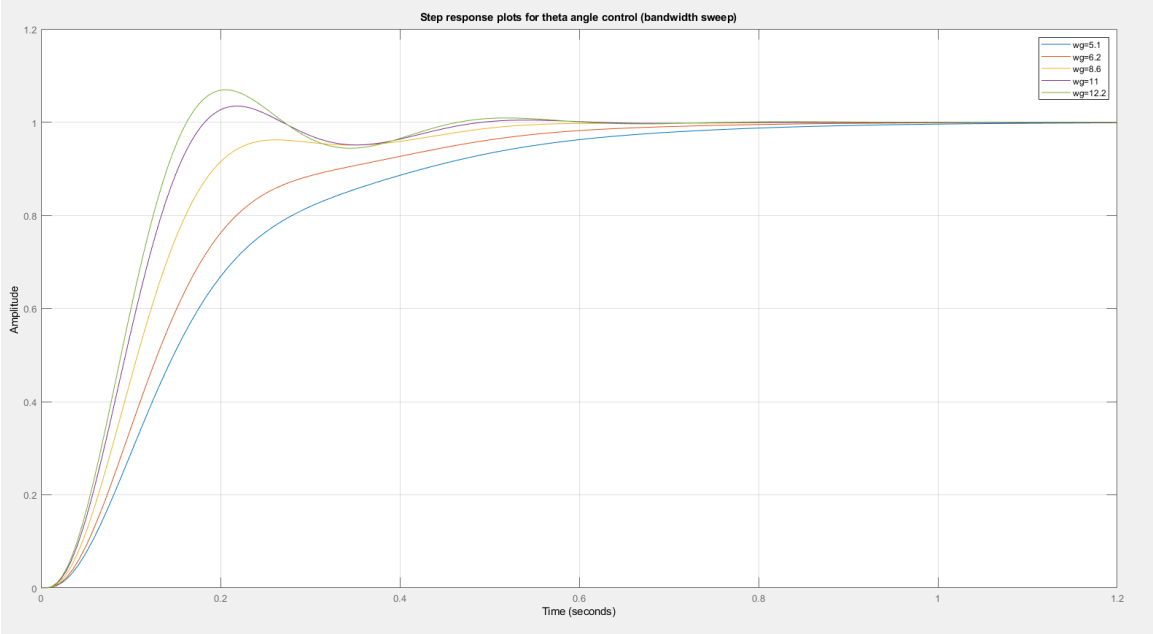

Figure 5.16: Pitch (θ) Angle Command Following - Time Domain Response

From Figure 5.16 it is observed that, as in the ϕ command following response, the response shows a distinct overshoot when control bandwidth is increased beyond 8.6 rad/s. The overshoot characteristics for ϕ and θ command following are nearly identical, and are as shown in Table 5.5.

Table 5.5: ϕ and θ Command Following - Step Response Characteristics

SI No.	ω_g**(rad/s)**	**Rise time (s)**	**Settling time (s)**	**Overshoot (%)**
1.	5.1	0.3695	0.7159	0
2.	6.2	0.2808	0.5865	0
3.	8.6	0.1468	0.4708	0
4.	11	0.1108	0.4354	3.4523
5.	12	0.1003	0.4247	6.9488

Moving forward, the design for $\omega_g = 8.6$ rad/s is used for roll and pitch angle command following. As seen in Table 5.5, there is no associated overshoot at this bandwidth, and a peak sensitivity of 3dB is adequately low, given that the maximum sensitivity is 6dB in most stable systems [8].

Chapter 6

HIGH LEVEL CONTROL OF A SINGLE QUADCOPTER- POSITION AND VELOCITY TRACKING

6.1 Overview

In Chapter 5, the low level control system is defined for angular rate control, as well roll/pitch angle command following. From Chapter 4, the quadcopter model has 12 states, with $X_{rig} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi.p, q, r]$ to stabilize. Having outlined control strategies for 5 of those states, (namely p ,q ,r , ϕ and θ), this section deals primarily with the control of the remaining 7 states of the single quadcopter, with the objective of establishing high level path following control for a fleet leader. This is implemented through use of an inverse mapping function and an LQ servo controller (as chosen in earlier literature [3]), which are described subsequently.

6.2 Inverse Mapping Function

It is desired that the leader quadcopter will be able to follow specified trajectories in the X, Y and Z axes. Given this requirement, the high level controller must be able to map trajectory commands as feasible low level control inputs for the motors. From contemporary literature [28], the concept of Differential Flatness can be used in order to achieve this mapping for trajectory generation.

Differential Flatness

A differentially flat system is one in which any state and input can be represented as a function of the output, without any derivation or integration required. The quadcopter system can be modeled as such, shown in Equations 6.1 through 6.3 [3][28]

$$y_p = y_p(x, u, \dot{u}, \ddot{u}, ...) \tag{6.1}$$

$$x_p = x_p(y, \dot{y}, \ddot{y}, ...) \tag{6.2}$$

$$u = u(y, \dot{y}, \ddot{y}, ...) \tag{6.3}$$

with $y_p = [x, y, z, \psi]^T$, $x_p = [y_p, \dot{y_p}]^T$, and $u = [y_p, \dot{y_p}, \ddot{y_p}]$.

This allows us to map control inputs for the x,y,z and yaw accelerations respectively, using the nonlinear quadcopter dynamic model obtained in Chapter 4. From Equations (4.13) and (4.14), we obtain a model for the desired control inputs $\ddot{x}$,$\ddot{y}$,$\ddot{z}$,and $\dot{\psi}$, shown in Equations 6.4 and 6.5

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\cos(\phi)\sin(\theta)cos(\psi) - \sin(\phi)\sin(\psi) \\ -\cos(\phi)\sin(\theta)sin(\psi) + \sin(\phi)\cos(\psi) \\ \cos(\phi)\cos(\theta) \end{bmatrix} \frac{T}{m} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{6.4}$$

$$\dot{\psi} = \frac{\sin(\phi)}{\cos(\theta)}p + \frac{\cos(\phi)}{\cos(\theta)}r \tag{6.5}$$

From [3] and [28], the (x, yz) acceleration terms can be used to map desired yaw (ψ) angle - in other words, a linear solution can be found for the system without

considering $\dot{\psi}$ as a constant [3]. Thus, we have Equation 6.6 and 6.7 representing the control law for high level trajectory control.

$$f_p(x,v) = R_z^T(\psi)R_y^T(\theta)R_x^T(\phi)\begin{bmatrix}0\\0\\1\end{bmatrix}\frac{T}{m} = u_p \tag{6.6}$$

$$f_\psi(x,v) = \frac{\sin(\phi)}{\cos(\theta)}p + \frac{\cos(\phi)}{\cos(\theta)}r = u_\psi \tag{6.7}$$

From the above, we obtain a state space representation for the system shown in Equations (6.8) through (6.11)

$$\dot{x} = Ax + Bu + hg \tag{6.8}$$

where

$$A = \begin{bmatrix}0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix} \tag{6.9}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

$$h = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{6.11}$$

The low level control input vector V can then be computed as shown in Equation 6.12

$$V = \begin{bmatrix} T \\ \phi \\ \theta \\ r \end{bmatrix} = \begin{bmatrix} f_p^{-1}(x, v) \\ f_\psi^{-1}(x, v) \end{bmatrix} \tag{6.12}$$

where $f_p(x, v)$ and $f_\psi(x, v)$ are the high level control inputs, as given in equation 6.6 and 6.7 respectively.

From these equations, the desired low level control inputs $[T, \phi, \theta, r]$ are obtained. To evaluate the thrust term T, we use equations 6.13 and 6.14 shown below.

$$u_p = R_z^T(\psi)R_y^T(\theta)R_x^T(\phi)\begin{bmatrix}0\\0\\1\end{bmatrix}\frac{T}{m} \tag{6.13}$$

from which we obtain

$$T = m\sqrt{u_{p1}^2 + u_{p2}^2 + u_{p3}^2} \tag{6.14}$$

To evaluate ϕ, θ and r commands, we multiply both sides of equation 6.13 by $R_z(\psi)\frac{m}{T}$ to obtain equation 6.15

$$R_z(\psi)u_p\frac{m}{T} = R_y^T(\theta)R_x^T(\phi)\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}z1\\z2\\z3\end{bmatrix} \tag{6.15}$$

From the values of $z1$,$z2$ and $z3$ computed, the ϕ and θ commands are computed using equations 6.16 and 6.17

$$\phi = \arcsin(z2) \tag{6.16}$$

$$\theta = \arctan(\frac{-z1}{z3}) \tag{6.17}$$

from which r is obtained using equation 6.18

$$r = u_\psi\frac{\cos(\theta)}{\cos(\phi)} + q\frac{\sin(\phi)}{\cos(\phi)} \tag{6.18}$$

Thus, the inverse mapping function is defined, using the concept of Differential Flatness, to map desired trajectory commands to low level control inputs.

6.3 Linearization at Hover

In order to define a linear state feedback system for trajectory following, the quadcopter model must be linearized about a chosen operating point. From Chapter 4 and in previous literature [29], each single quadcopter is linearized about the hover position where $\phi = \theta = \psi = 0^\circ$ and $T = mg$. This yields a state space model for the quadcopter, as shown in equations 6.19 and equation 6.20

$$\dot{x_p} = A_p x_p + B_p u_p \tag{6.19}$$

$$y_p = C_p x_p + D_p u_p \tag{6.20}$$

where $u_p = [T, \phi, \theta, r]^T$, $x_p = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \psi]^T$, $y_p = [x, y, z, \psi]^T$.

For finding the state space matrices $[A_p, B_p, C_p, D_p]$ we have the linear matrices shown in equations 6.21 and 6.22 without coordinate transform.

$$A_{p1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B_{p1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -9.81 & 0 & 0 & 0 \\ 0 & 9.81 & 0 & 0 \\ 0 & 0 & 1/m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.21}$$

$$C_{p1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, D_{p1} = \begin{bmatrix} 0_{4\times 7} \end{bmatrix} \tag{6.22}$$

Therefore, at low frequencies, the high level plant transfer matrix (for outputs of interest $[x, y, z, \psi]$ is shown in Equation 6.23. It is noted that for $[x, y, z]$ position control the respective plant transfer functions are all second order.

$$P(s) = C_p(sI - A_p)^{-1}B_p = \begin{bmatrix} \frac{-0.17122}{s^2} & 0 & 0 & 0 \\ 0 & \frac{0.17122}{s^2} & 0 & 0 \\ 0 & 0 & \frac{1.502}{s^2} & 0 \\ 0 & 0 & 0\frac{1}{s} & \end{bmatrix} \tag{6.23}$$

As described in [30], a coordinate transformation must be performed for the matrices in equation 6.21 and 6.22, to convert from degrees to radians. Transformation matrices are shown in Equation 6.24.

$$S_u = \begin{bmatrix} \frac{180}{\pi} & 0 & 0 & 0 \\ 0 & \frac{180}{\pi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{180}{\pi} \end{bmatrix}, S_x = \begin{bmatrix} I_{6\times 6} & 0_{6\times 1} \\ 0_{1\times 6} & \frac{180}{\pi} \end{bmatrix}, S_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{180}{\pi} \end{bmatrix} \tag{6.24}$$

from which the transformation is applied to obtain $[A_p, B_p, C_p, D_p]$ (equations 6.25-6.28)

$$A_p = S_x A_{p1} S_x^{-1} \tag{6.25}$$

$$B_p = S_x B_{p1} S_u^{-1} \tag{6.26}$$

$$C_p = S_y C_{p1} S_x^{-1} \tag{6.27}$$

$$D_p = S_y D_{p1} S_u^{-1} \tag{6.28}$$

Thus, a linear state space model is obtained for the system. Using this model, a state feedback control law can be defined using the LQR optimization method to achieve robust LQ servo control, which is described subsequently in this chapter. The high level feedback block diagram for the single quadcopter is shown in Figure 6.1 [3].

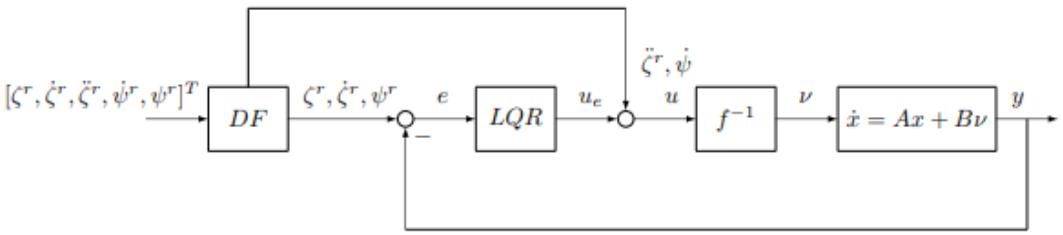

Figure 6.1: High Level Block Diagram - Lead Quadcopter Trajectory Generation

6.4 LQR Algorithm and LQ Servo Design - Key Plots and Observations

To model a high level controller the decoupled model (near hover) is used without the coordinate transformation. This is because the attitude (low level) control system was designed in units of radians, and as such position control must be designed using the same units.

To address LQ servo design, an integrator must be added to each state to guarantee zero steady state error in command following. Thus, we have state space matrices $[A, B, C, D]$ for the LQ servo as shown in Equation 6.29 (derived from [3])

$$A = \begin{bmatrix} A_p & 0_{7\times4} \\ C_p & 0_{4\times4} \end{bmatrix}, B = \begin{bmatrix} B_p \\ 0_{4\times4} \end{bmatrix}, C = \begin{bmatrix} I_{7\times7} \end{bmatrix}, D = [0_{7\times4}] \tag{6.29}$$

The objective of the algorithm is to find a stable solution u (corresponding to control input) using Equations 6.30 and 6.31

$$u = -Gx \tag{6.30}$$

$$G = R^{-1}B^{T}K \tag{6.31}$$

In Equation 6.26, K is the unique symmetric solution to the Control Algebraic Ricatti Equation (CARE) shown in Equation 6.32

$$0 = KA + A^{T}K + M^{T}M - KBR^{-1}B^{T}K \tag{6.32}$$

With $Q = diag(10, 10, 10, 10, 10, 10, 100, 100, 100, 100, 1)$ and $R = 0.2I_{4\times4}$, the G matrix obtained using the LQR optimization algorithm (CARE) is shown in Equation 6.33.

$$G = \begin{bmatrix} 30.93 & 0 & 0 & -20.28 & 0 & 0 & 0 & -22.36 & 0 & 0 & 0 \\ 0 & 30.93 & 0 & 0 & 20.28 & 0 & 0 & 0 & 22.36 & 0 & 0 \\ 0 & 0 & 21.32 & 0 & 0 & 9.05 & 0 & 0 & 0 & 22.36 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 22.46 & 0 & 0 & 0 & 2.24 \end{bmatrix} \tag{6.33}$$

The block diagram for LQ servo control is as shown in Figure 6.2 [3]

where

$$G_y = \begin{bmatrix} 30.93 & 0 & 0 & -20.28 & 0 & 0 & 0 \\ 0 & 30.93 & 0 & 0 & 20.28 & 0 & 0 \\ 0 & 0 & 21.33 & 0 & 0 & 9.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 22.46 \end{bmatrix}, \tag{6.34}$$

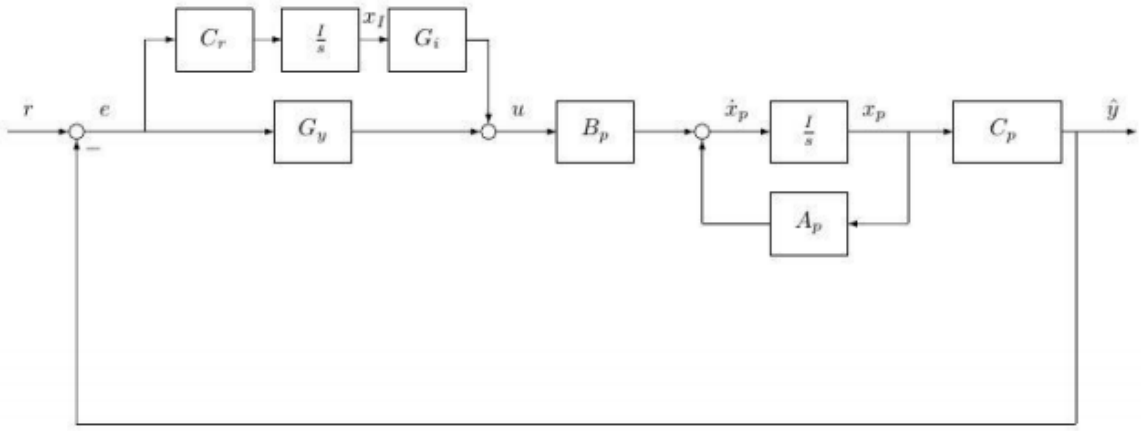

Figure 6.2: LQ Servo Control - State Space Block Diagram

$$G_z = \begin{bmatrix} -22.3607 & 0 & 0 & 0 \\ 0 & 22.3607 & 0 & 0 \\ 0 & 0 & 22.3607 & 0 \\ 0 & 0 & 0 & 2.2361 \end{bmatrix} \tag{6.35}$$

The LQ sensitivity and complementary sensitivity plots for this controller are shown in Figures 6.3 and 6.4 respectively. As shown on Figure 6.3, all frequencies below 0.7 radians per second are attenuated by at least 20 dB, and the system shows good low frequency disturbance attenuation. As peak complementary sensitivity is below 4dB (approximately 3.2 dB), the system shows good command following response at low frequencies, and rejects any output noise above 8 radians per second as seen in Figure 6.4 (a measure of the system bandwidth).

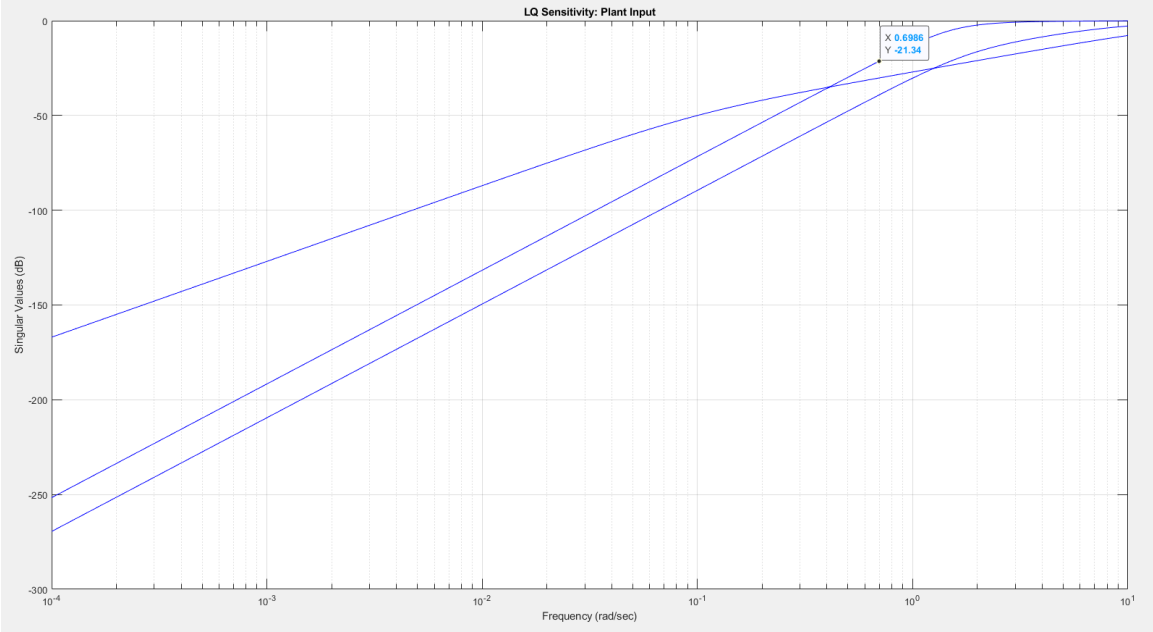

Figure 6.3: LQ Servo Control - Input Sensitivity Frequency Response

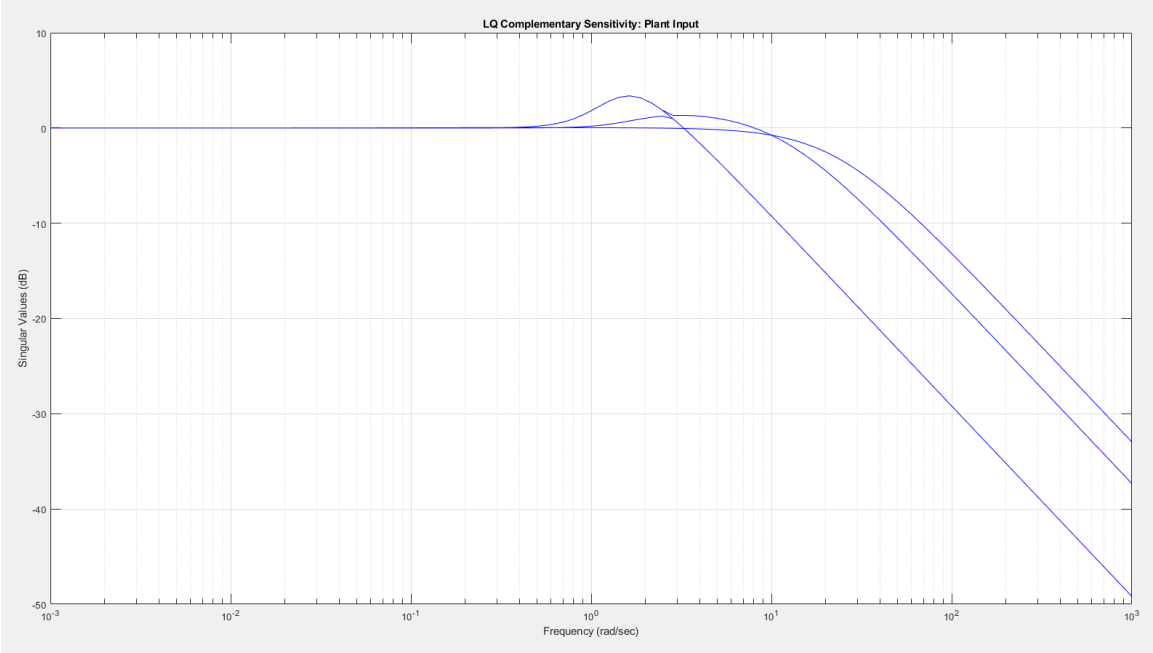

Figure 6.4: LQ Servo Control - Closed Loop Frequency Response

The command following (time domain) responses are shown on the facing pages, for x,y,z, and ψ step commands issued by the base station to the leader quadcopter, in Figures 6.5 through 6.8 respectively

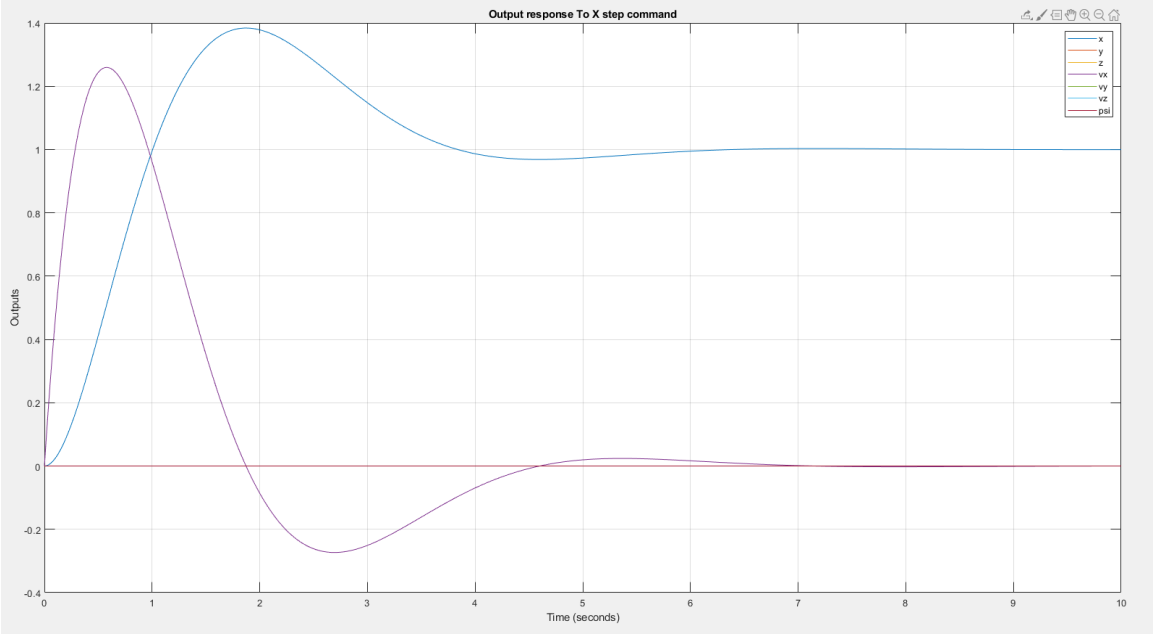

Figure 6.5: Time Domain Response - X-Axis Position Step Command (r=[1 0 0 0 0 0 0 0])

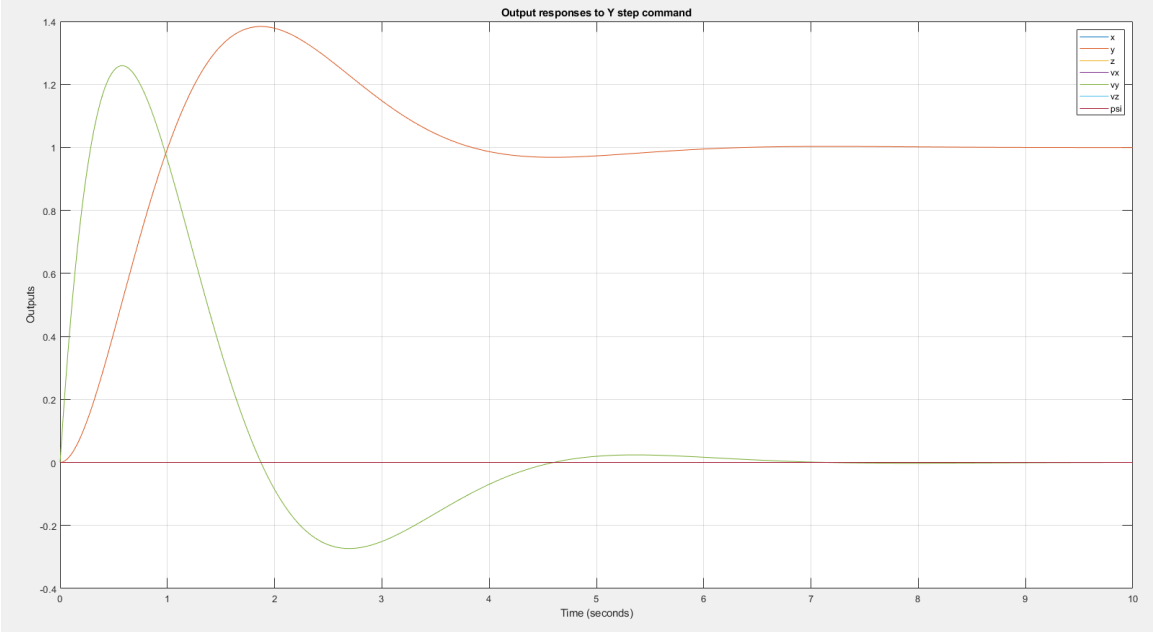

Figure 6.6: Time Domain Response - Y-Axis Position Step Command (r=[0 1 0 0 0 0 0 0])

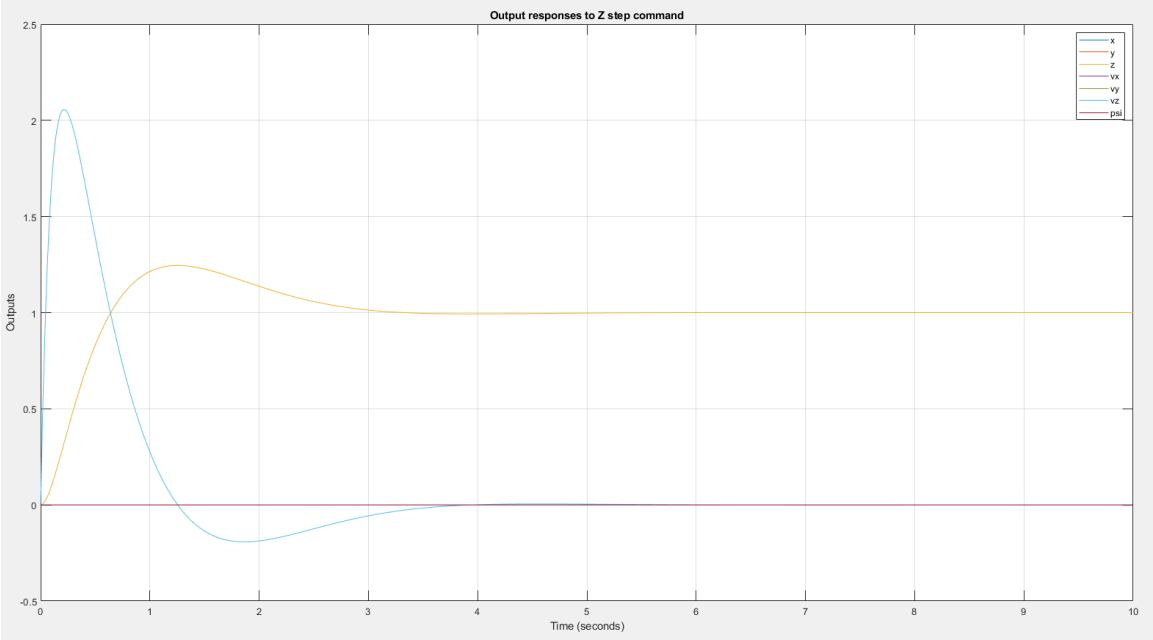

Figure 6.7: Time Domain Response - Z-Axis Position Step Command (r=[0 0 1 0 0 0 0 0])

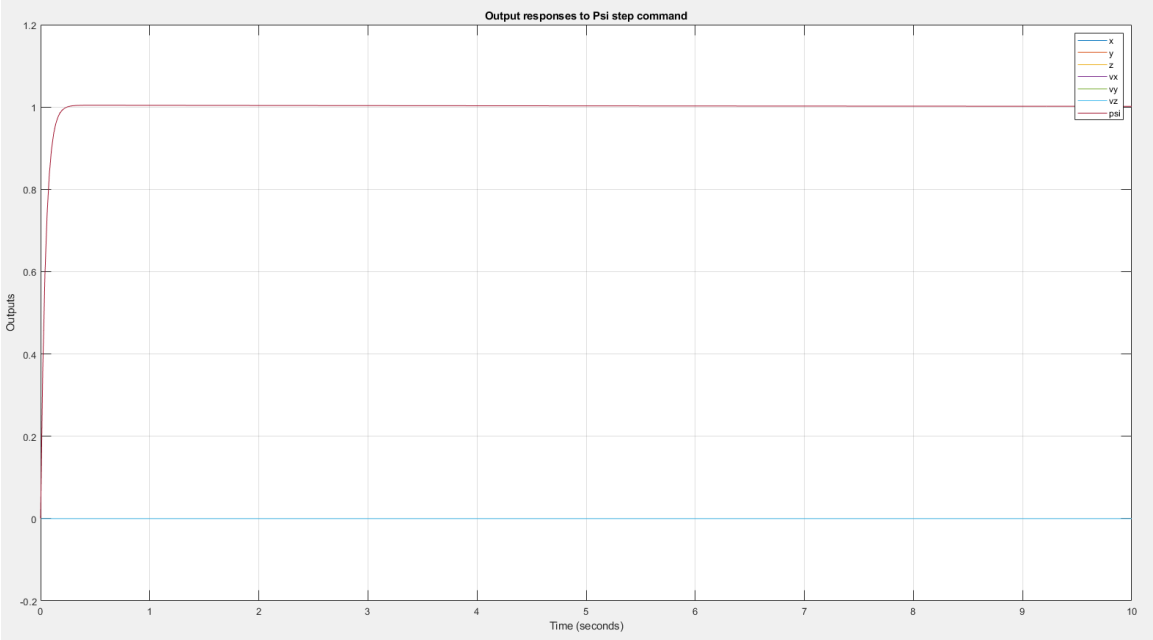

Figure 6.8: Time Domain Response - ψ Angle Step Command (r=[0 0 0 0 0 0 0 1])

The control input responses for x,y,z, and ψ are shown in Figures 6.9 through 6.12 respectively

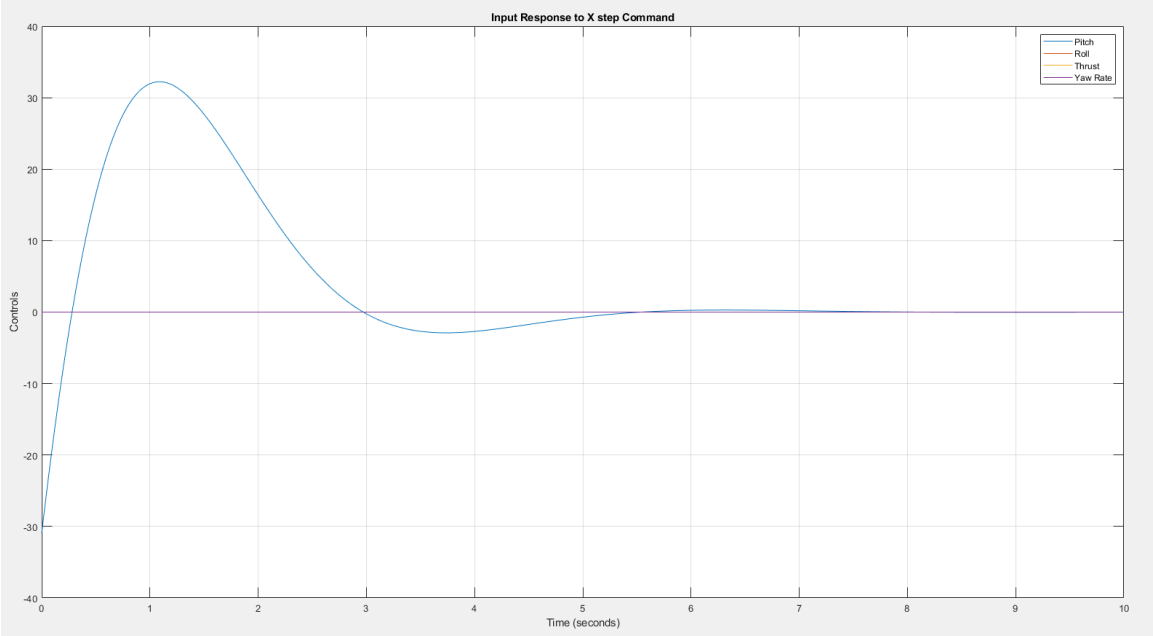

Figure 6.9: Control Input Response - X-Axis Position Step Command (r=[1 0 0 0 0 0 0 0])

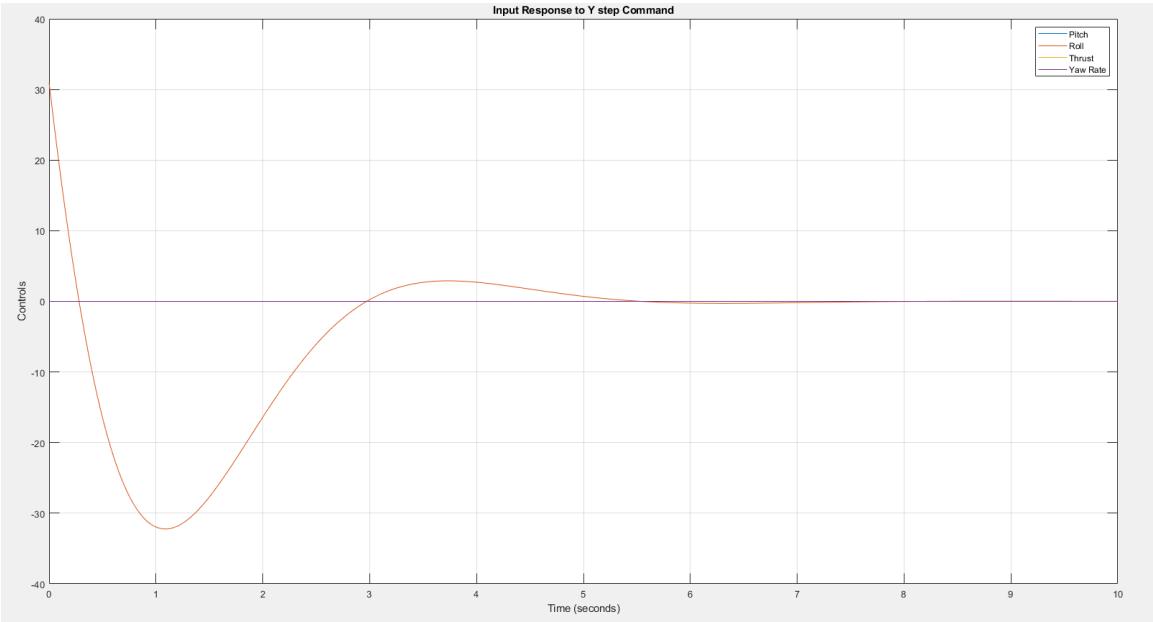

Figure 6.10: Control Input Response - Y-Axis Position Step Command (r=[0 1 0 0 0 0 0 0])

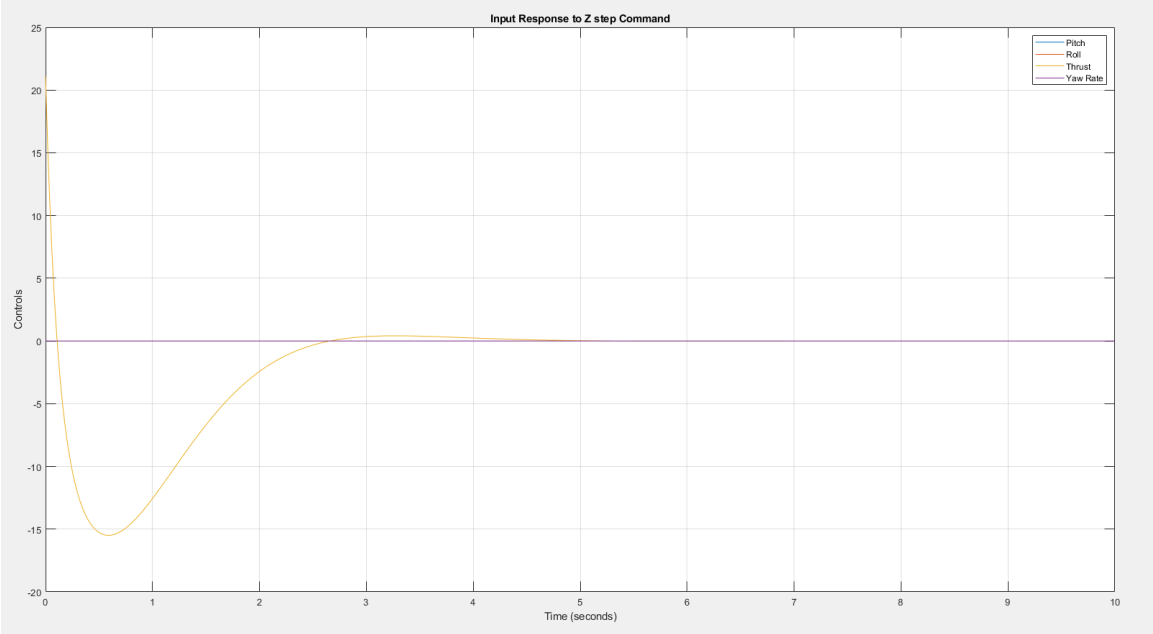

Figure 6.11: Control Input Response - Z-Axis Position Step Command (r=[0 0 1 0 0 0 0])

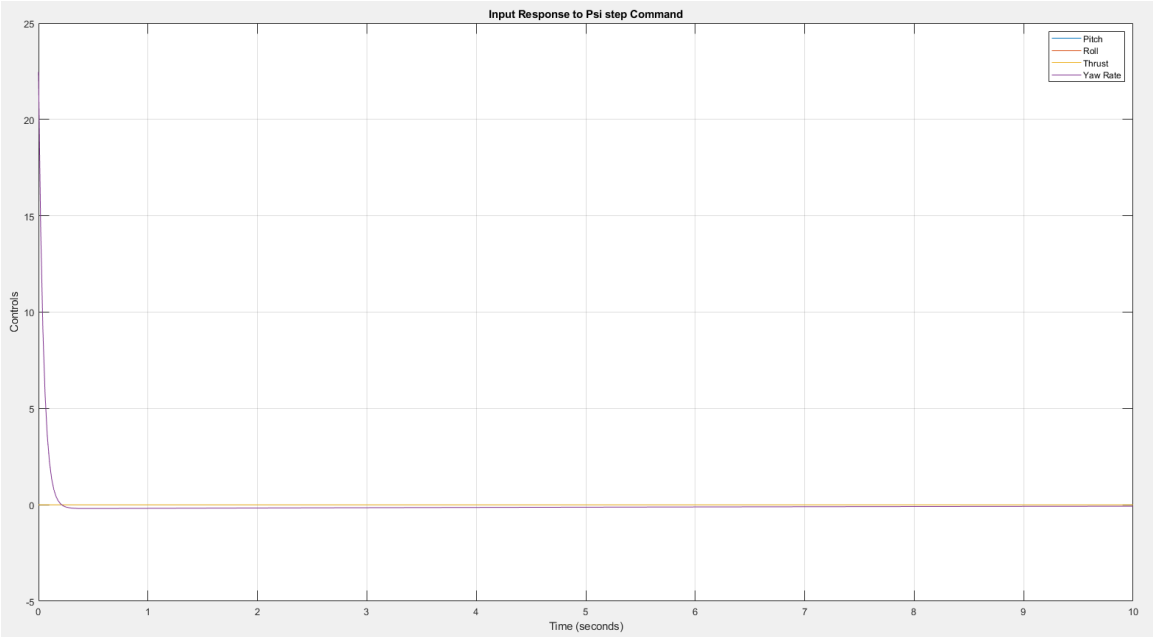

Figure 6.12: Control Input Response - ψ Step Command (r=[0 1 0 0 0 0 0])

From Figure 6.5 and 6.6, X and Y commands show an overshoot of around 38%, but it is noted that in both cases, the control response (pitch and roll values) is not aggressive where either angle goes from approximately -30° to 30°. For altitude (Z-axis) command, the output response has a relatively low overshoot of 23%. It is also noted that this design has proven to work well with quadcopter hardware (in [3]) and so this model is used as a basis for each agent for platoon control analysis.

Chapter 7

PLATOONING CONTROL FOR A FLEET OF QUADCOPTERS

7.1 Overview

Having established high level trajectory tracking (using the LQR optimization detailed in section 6) we now address platooning (or separation) control for a fleet of 6 quadrotors, which is the focus of the modeling and simulations presented in this chapter. Specifically, in this section, the performance of different controllers (chosen for a bandwidth sweep) is analyzed, by observing the spacing between each follower quadcopter when the leader is commanded to perform complex maneuvers. Thus, a classical control law is proposed to minimize spacing errors between quadcopters in the fleet. The analysis is primarily focused on minimizing the accordion effect on follower units in the fleet.

7.2 Approach for Separation Control in Quadcopter Fleets - The Accordion Effect

Design of separation control laws is a major aspect of vehicle fleet management in aerial and terrestrial applications alike. In the 1992 seminal paper by Dr. Charles Desoer and Dr. Shahab Sheikholeslam [4], a controller is designed in order to maintain desired inter vehicular spacing between multiple cars in traffic. This is performed by considering a leader-follower approach, where the vehicle in front is considered the leader of traffic. All vehicles behind the leader are considered as followers, and controllers are designed for each follower in order to maintain a fixed spacing or separation from it's immediate neighbors in the fleet, based on a predetermined separation vector.

The major observation from this seminal paper was that any change in the leader vehicle's velocity would result in positioning errors between the follower vehicles. In subsequent and contemporary research, this phenomenon is referred to as the "accordion" effect. In leader-follower control, the accordion effect is found to arise mainly because of fluctuations in lead vehicle acceleration. Therefore, as the high level control inputs for the quadcopter correspond directly to desired acceleration, the separation control law for the quadcopter system must regulate the acceleration commanded of follower quadcopters, subject to fluctuations in lead quadcopter velocity along a specified trajectory.

In contemporary research ([3],[19],[20]) the leader follower approach has been implemented in different ways, with a focus on optimizing control input with state feedback. The optimization methods in these papers are described for models where leader state information is available to follower vehicles, with string stability assumed. Thus, the focus of this paper is to analyze and quantify the effect of leader feedback on string stability (as defined in [22]) within a quadcopter fleet, by assuming a nominal case where no leader feedback information is available. Leader acceleration information is then fed back to all quadcopters in the fleet, to observe the effect of lead vehicle acceleration on the separation controller output in line and circle trajectories.

7.3 Modeling of Platoon Dynamics - Separation Control Diagram and Equations

As introduced in the previous subsection, we first consider a nominal model wherein each quadcopter only receives position information from it's immediate predecessor in the fleet. The assumptions for this model are listed below.

- Each quadrotor is treated as identical for modeling the controller

- Each quadrotor transmits it's state information (x,y,z position and velocity data) to it's immediate neighbor in the fleet. It is envisaged that in current and evolving network technologies such as 5G, ad hoc wireless communication of states between quadrotors is feasible without significant packet loss-packet drop analysis is not included in the scope of this thesis.

- Communication of the position,velocity, and acceleration information is unidirectional: from the lead vehicle to each follower in succession

As described in chapter 2, the work presented by Dr. Desoer and Sheikholeslam [cite: accordion main paper 1, and no lead info paper 2] is used as a primary reference for modeling a separation controller for the quadcopter using classical PID control methods. In [4] and [5], the proposed platoon configuration is as shown in Figure 7.1.

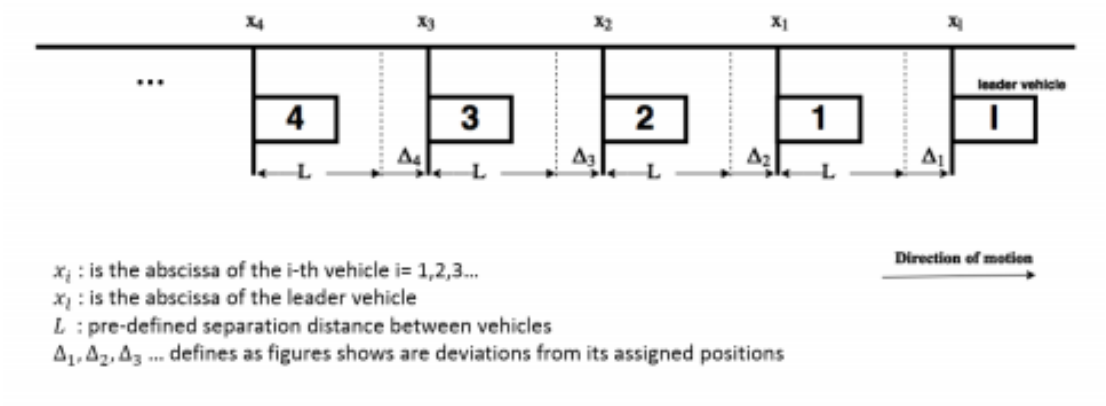

Figure 7.1: Platoon Configuration Model, with 1 Leader and 4 followers

It is assumed that in the quadcopter fleet, the distance L is assigned prior to take-off - in other words, each agent is aware of the spacing it must maintain from it's immediate neighbor. The Δ values shown in Figure 7.1 represent the deviation of the respective follower from it's assigned position (determined by L). Using Figure 7.1, the relevant kinematic equations for the platoon controller are shown in Equations 7.1 through 7.3 [4]

$$\Delta_i(t) = x_{i-1}(t) - x_i(t) - L \tag{7.1}$$

$$\dot{\Delta}_i(t) = \dot{x}_{i-1}(t) - \dot{x}_i(t) \tag{7.2}$$

$$\ddot{\Delta}_i(t) = \ddot{x}_{i-1}(t) - \ddot{x}_i(t) \tag{7.3}$$

With the assumption of ad hoc position data communication between the fleet (from one follower to the next), the value of $\Delta_i(t)$ can be directly computed by the i_{th} quadcopter using feedback of it's own position data. In this implementation, in order to maintain longitudinal and lateral spacing control within the fleet, two controllers are used to minimize deviation from the desired X axis coordinate (Δ_x)

as well as for the desired Y axis coordinate (Δ_y). Using the X and Y coordinates communicated by the previous quadcopter in the fleet, the Δ_x controller generates an X axis acceleration term and the Δ_y controller generates a Y axis acceleration term. These terms act as high level plant inputs for the inverse mapping function on each follower (as described in Chapter 6).

As the model is linearized about a hover position, it is assumed that lateral and longitudinal separation controller will be similar, as the roll and pitch dynamics of each respective quadcopter are nearly identical.

7.4 Nominal Model (Ad-hoc Communication)

For the quadcopter fleet, the analysis is first performed assuming no leader information feedback. In this case, the leader quadcopter follows the trajectory specified by the base station, using high level path following control described in Chapter 4, and the model is as shown in Figure 6.1. For the follower units, high level inputs (X and Y axis accelerations) are computed using the calculated X axis and Y axis position deviation respectively. The model for the i_{th} follower is shown in Figure 7.2.

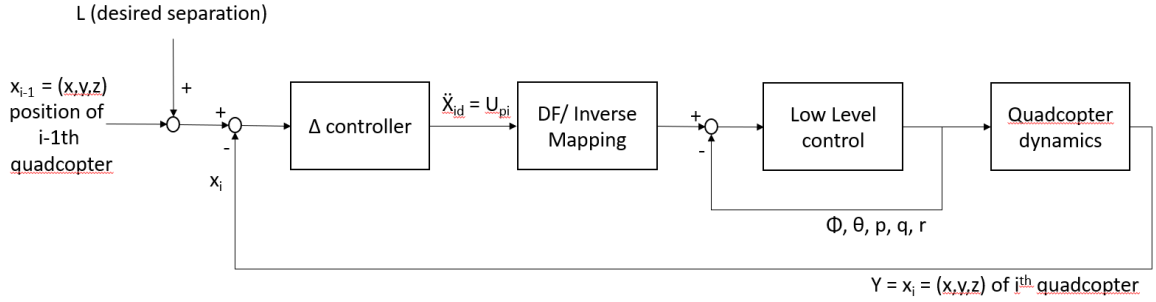

Figure 7.2: Model Diagram for i-th Follower

In Figure 7.2, the Δ (Separation) controller K_Δ is modeled as a PD design, with a first order high frequency roll-off term for noise attenuation and phase margin of 60 degrees. It is noted that the fuselage harmonics for each quadcopter begin around 30 rad/s, and so the separation controller bandwidth should ideally not exceed 3 rad/s for relative positioning/spacing commands (although slightly higher values are tested in simulation). The phase margin is maintained at 60° to maintain good closed loop properties [8].

The controller structure is as shown in Equation 7.4.

$$K_\Delta = K_{\Delta x} = K_{\Delta y} = g(s+z)\left(\frac{r}{(s+r)}\right) \tag{7.4}$$

Where $r = 10\omega_g$ (approximately a decade above desired bandwidth)

Using this model, a bandwidth sweep is performed for Δ_x and Δ_y control. A PD controller with a high frequency roll-off is used in all cases. The PD controller ensures that oscillations in the output are reduced, even at high gain. Moreover, as the high level system is 2nd order at low frequency(from $\ddot{x}$ and $\ddot{y}$, to x and y respectively, as seen in Equation 6.23), a high frequency roll off term is sufficient to ensure zero steady state errors in Δ_x and Δ_y response. Also, given that the inner loop has been designed well, Δ_x and Δ_y can be modeled as SISO systems, given that the nonlinear model for each quadcopter is fairly decoupled at low frequencies.

The closed loop magnitude response plots for Δ_x and Δ_y bandwidth sweep are shown in Figures 7.3 and 7.4 respectively.

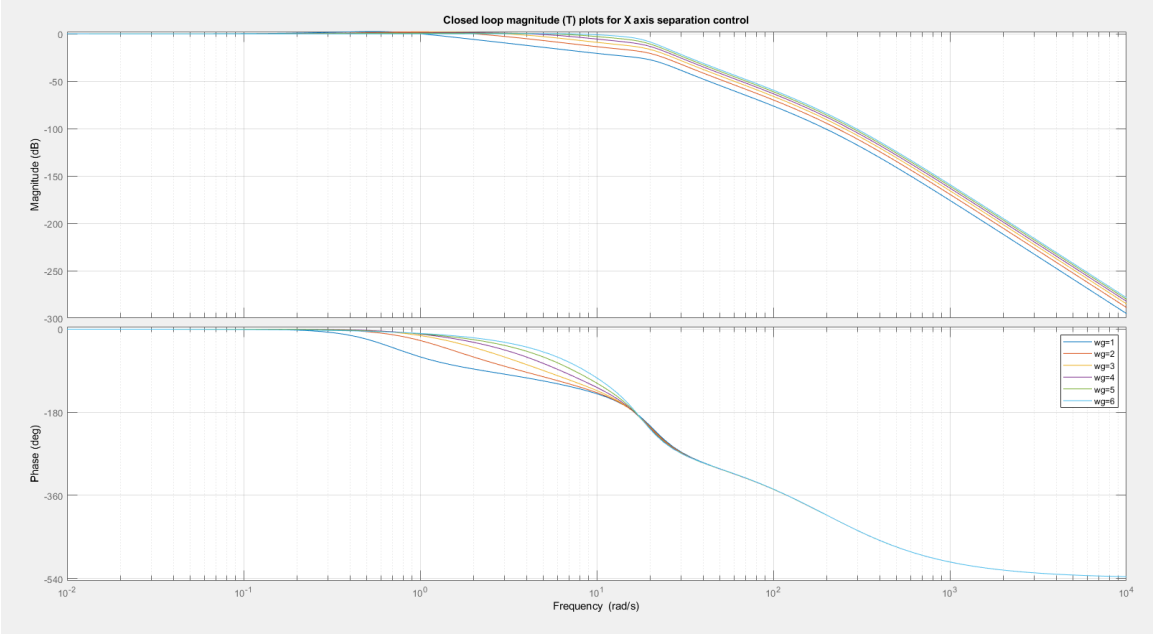

Figure 7.3: Closed Loop Frequency Response - Δ_x Control

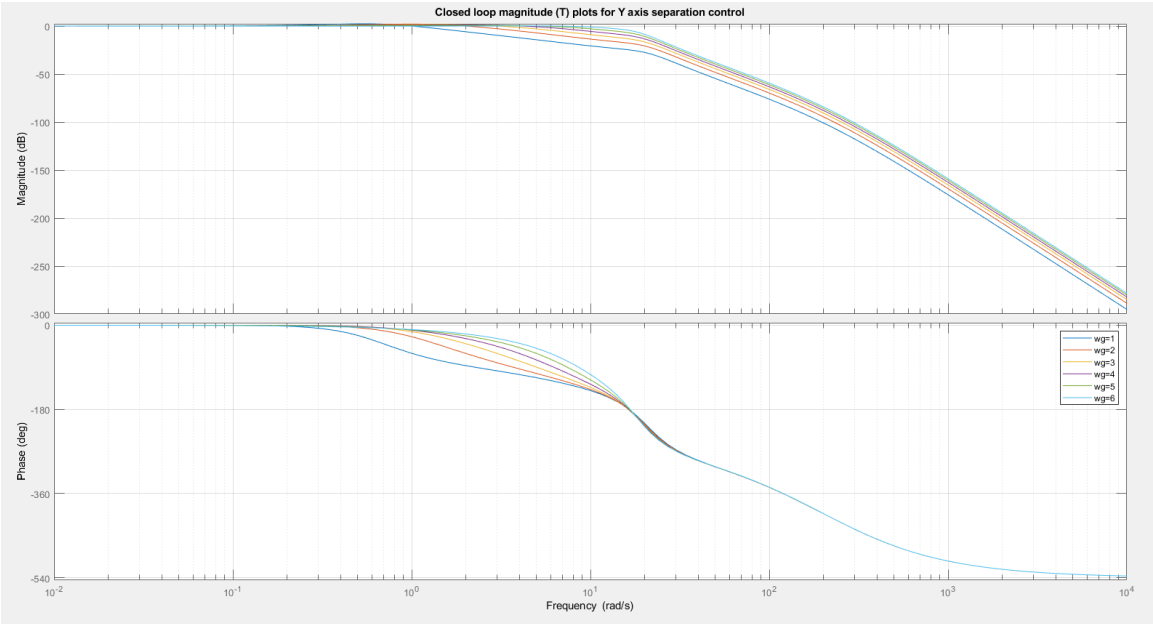

Figure 7.4: Closed Loop Frequency Response - Δ_y Control

From the closed loop Bode magnitude plot, the peak dB values are all near 0dB, which implies that all designs show good command following at low frequencies. The closed loop sensitivity functions are plotted in Figures 7.5 and 7.6.

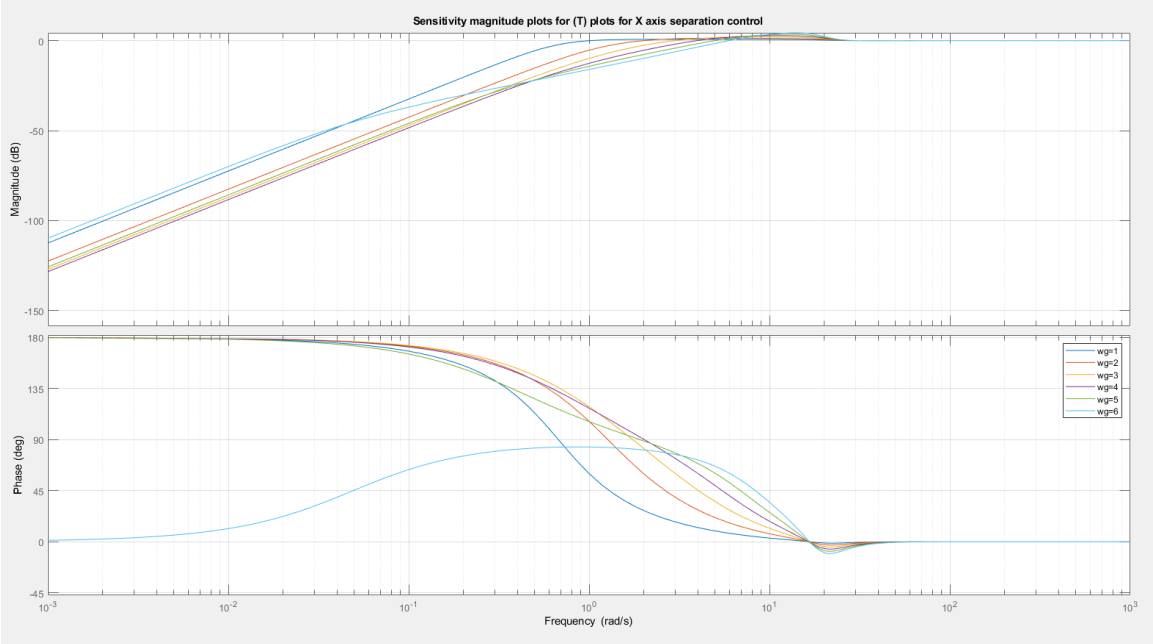

Figure 7.5: Closed Loop Frequency Response - Δ_x Control

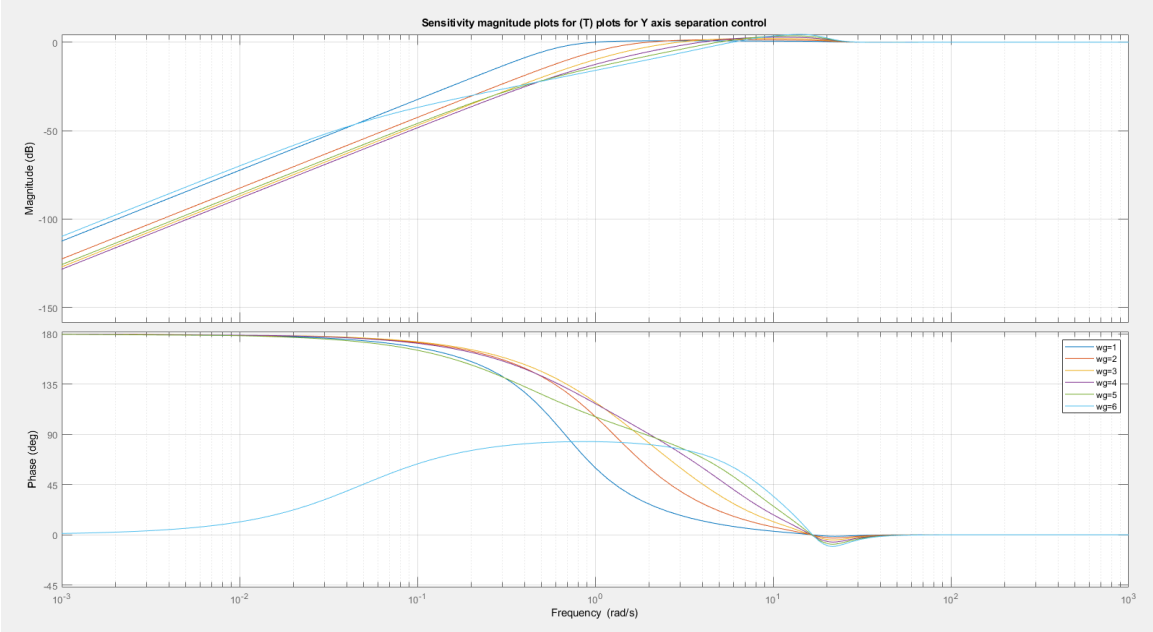

Figure 7.6: Closed Loop Frequency Response - Δ_y Control

It is observed that as bandwidth is increased, the peak sensitivity values for the closed loop Δx system increase as well. The peak sensitivity values are documented in Table 7.1 below.

Table 7.1: Bandwidth Sweep Results - Peak Sensitivity Values for Δx Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**
1.	1	0.75
2.	2	1.39
3.	3	2.03
4.	4	2.70
5.	5	3.47
6.	6	4.26

A similar pattern is observed in the $Delta_y$ sensitivity response, with peak values shown in Table 7.2 (similar to values in Table 7.1)

Table 7.2: Bandwidth Sweep Results - Peak Sensitivity Values for Δ_y Command Following

SI No.	ω_g**(rad/s)**	**Peak sensitivity (dB)**
1.	1	0.743
2.	2	1.38
3.	3	2
4.	4	2.67
5.	5	3.43
6.	6	4.22

The command following responses (in time domain) for Δ_x and Δ_y are shown in Figures 7.7 and 7.8

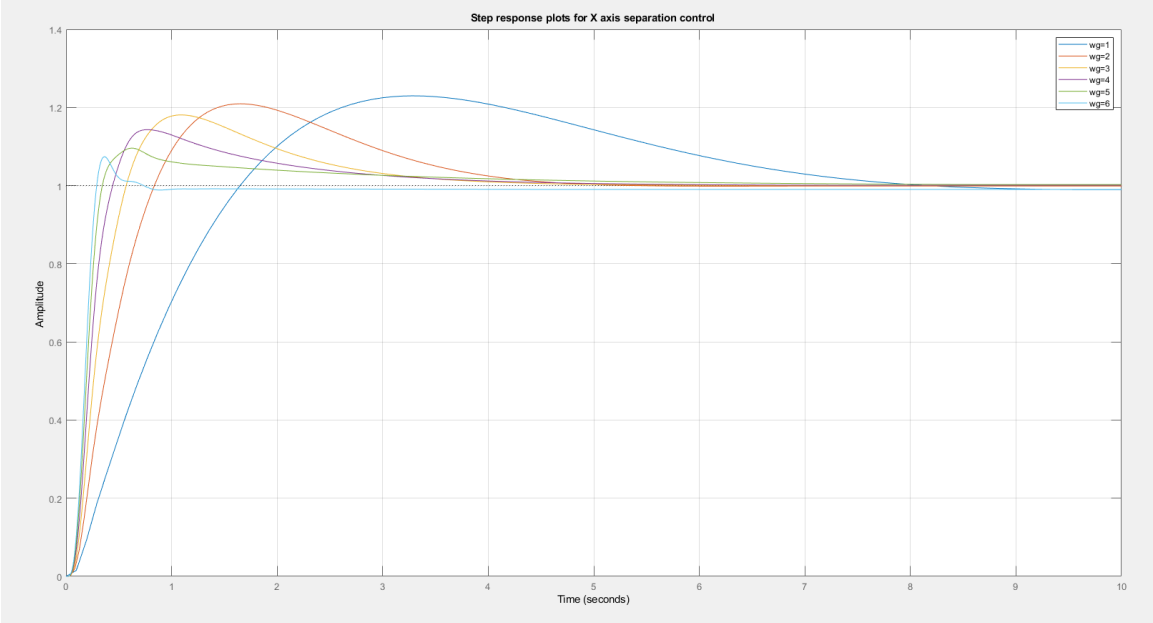

Figure 7.7: Δ_x Command Following Response

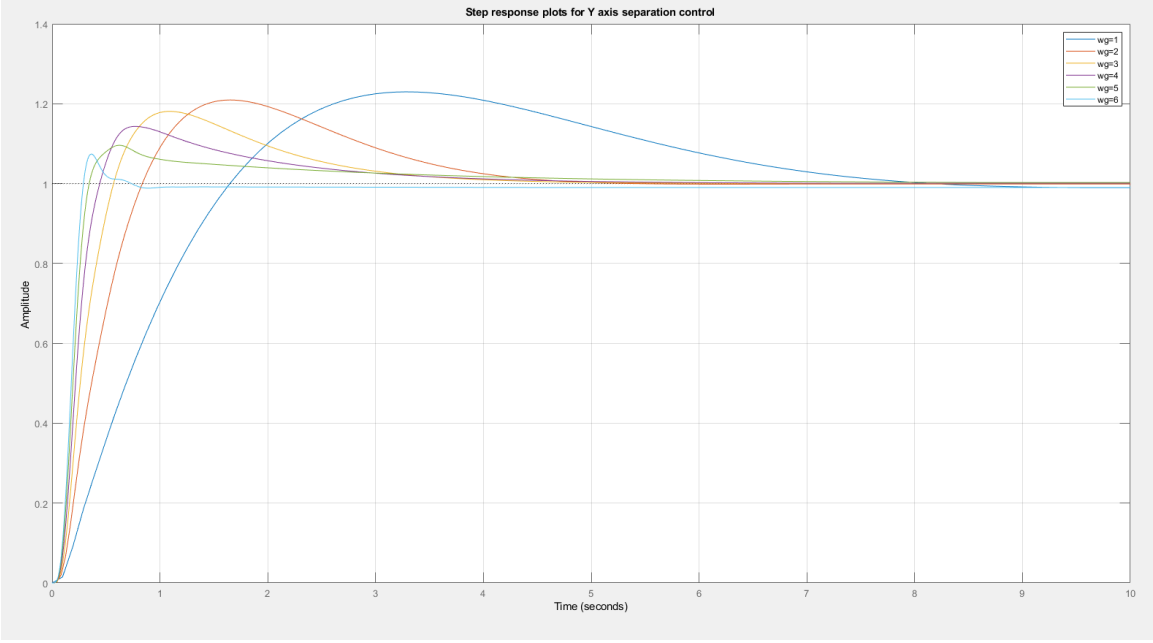

Figure 7.8: Δ_y Command Following Response

From Figures 7.7 and 7.8, the step response characteristics are nearly identical for X and Y command following. These characteristics are as shown in Table 7.3

Table 7.3: Δ_x and Δ_y Command Following - Step Response Characteristics

SI No.	ω_g**(rad/s)**	**Rise time (s)**	**Settling time (s)**	**Overshoot (%)**
1.	1	1.1916	7.2862	22.9260
2.	2	0.5645	4.1305	20.8512
3.	3	0.3637	3.3689	17.9774
4.	4	0.2561	3.3157	14.1498
5.	5	0.1927	3.6535	9.528
6.	6	0.1628	0.48(w/ error)	6.3

As noted in the table and seen in the figures, there is a steady state error in the step response when ω_g is increased beyond 5 rad/sec, with observed undershoot at $\omega_g = 6rad/s$. It is also noted that there is a slight increase in settling time (by 0.3 s) when moving from 4 rad/s to 5 rad/s, while rise time continues to reduce as bandwidth is increased upto 6 rad/s. Below 2 rad/s, there is high overshoot (¿20%) which is not acceptable to maintain accuracy in spacing when taking the accordion effect into account.

Thus, based on the observations above, analyses in future sections is restricted to ω_g = [234] rad/s.

7.4.1 Separation Control Along a Line, No Leader Feedback

With the bandwidth sweep presented in the previous section, Δ_x and Δ_y simulation results are presented for each controller. 6 quadcopters are arranged in a platoon configuration along the X axis, with desired separation of 2 m commanded between each of them. In this section, the leader is given a straight line path in the X-Y plane, with $\dot{x} = \dot{y} = 1m/s$, and $x = \dot{x} * t$,$y = \dot{y} * t$ (t = time since takeoff(s)

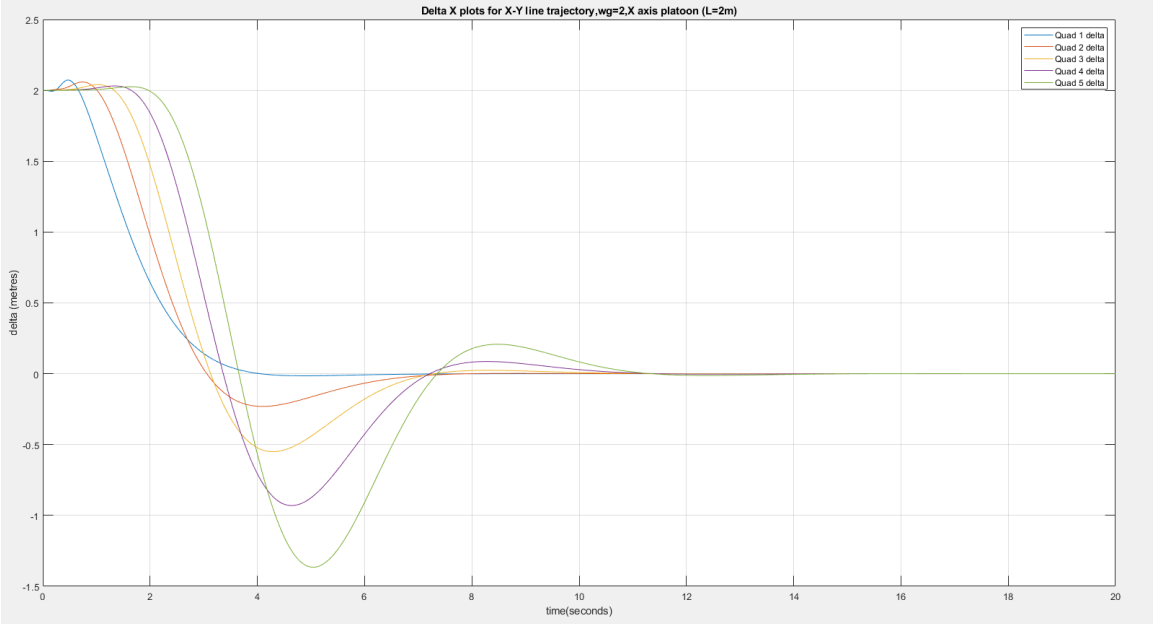

Figure 7.9: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=2 rad/s

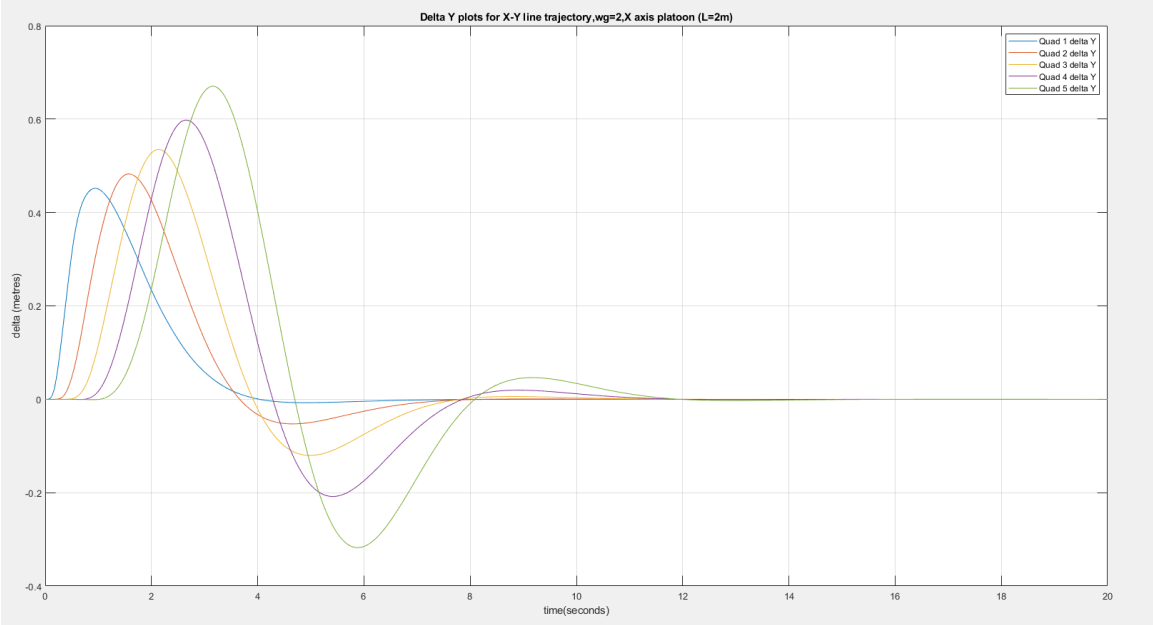

Figure 7.10: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=2 rad/s

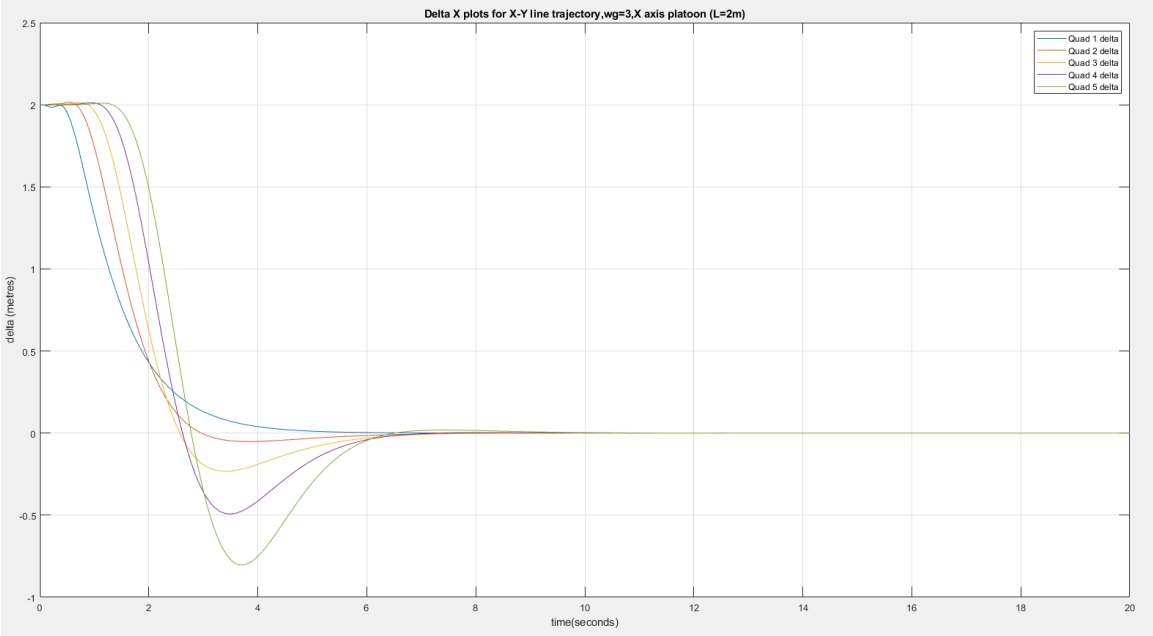

Figure 7.11: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=3 rad/s

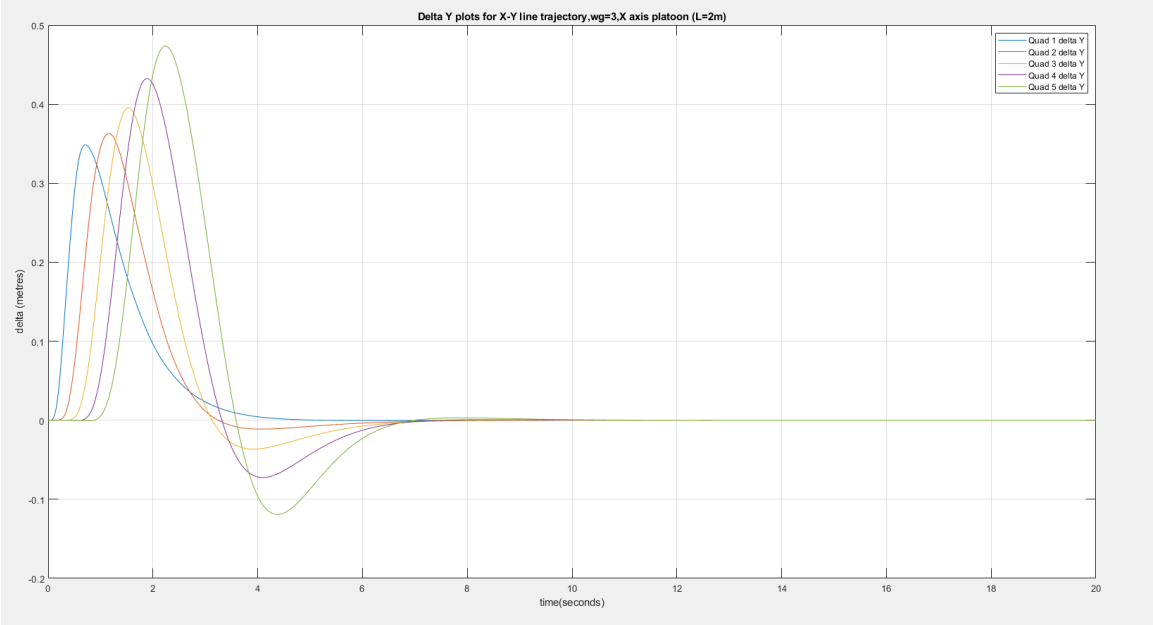

Figure 7.12: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=3 rad/s

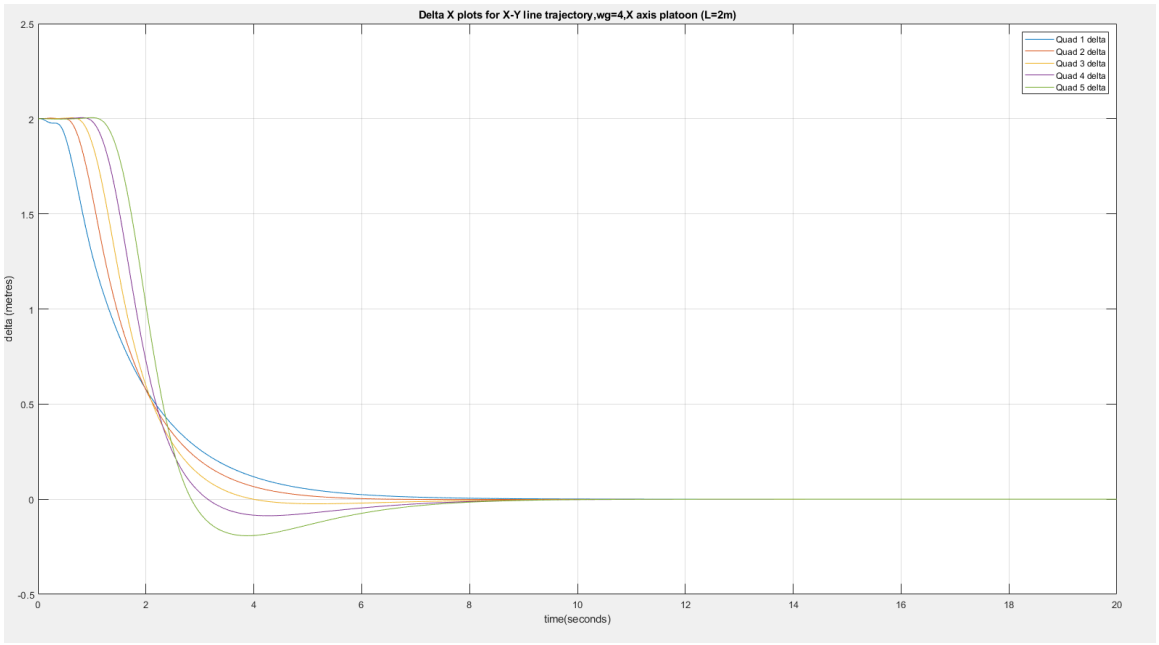

Figure 7.13: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=4 rad/s

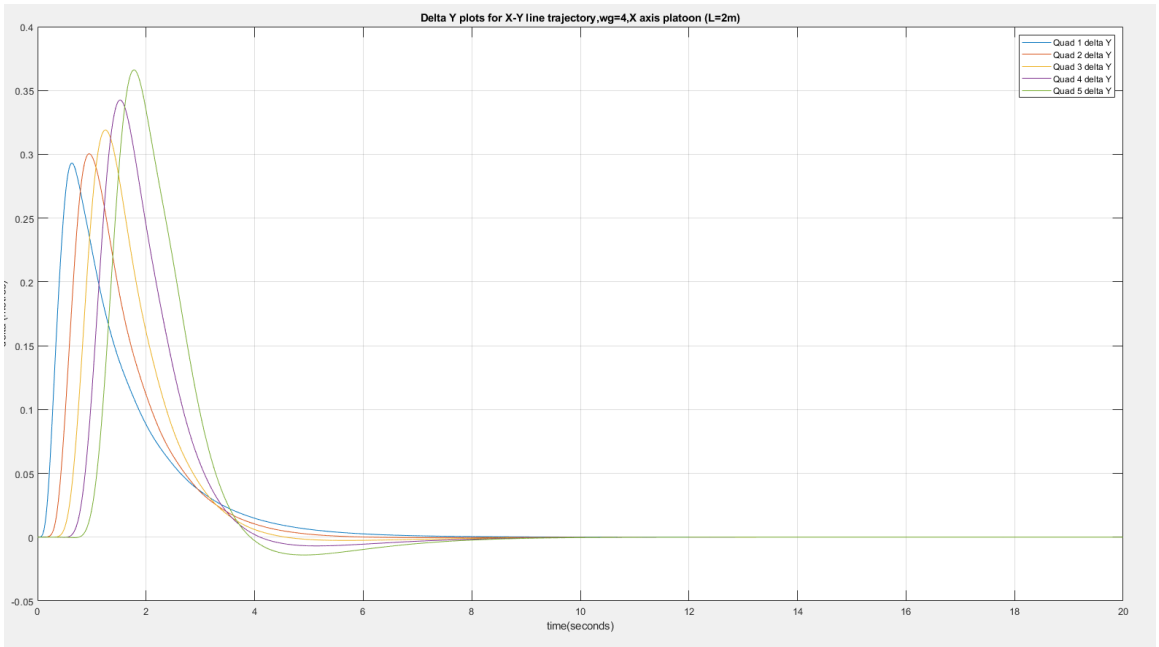

Figure 7.14: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Straight Line Trajectory, ω_g=4 rad/s

It is noted that along a line, Δ_x and Δ_y all show zero steady state error, and the initial accordion effect is stabilized without lead vehicle information. The settling time improves when ω_g is changed from 2 rad/s (settles in about 12 seconds) to 3 rad/s (settles in approximately 7 to 9 seconds), although there is little reduction in settling time when bandwidth is increased beyond 3 rad/s. The overshoot in Δ_y response (observed during initial seconds/platoon formation) reduces with increasing controller bandwidth, as expected.

7.4.2 Separation Control Along a Curve, No Leader Feedback

Having observed results or platooning control along a line, simulations are now performed for curve path/circular trajectories. This curve trajectory has a radius of curvature = 2.8 m, with commanded translational velocity along the curve fixed at 1.4 m/s. The simulation results are as shown from facing page

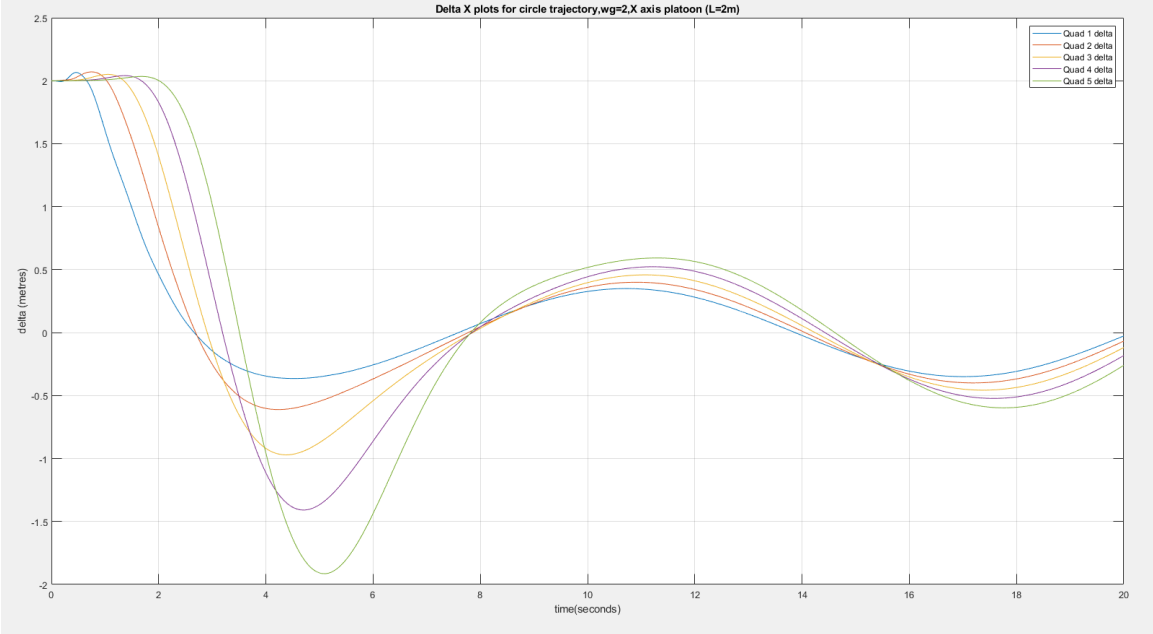

Figure 7.15: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=2 rad/s

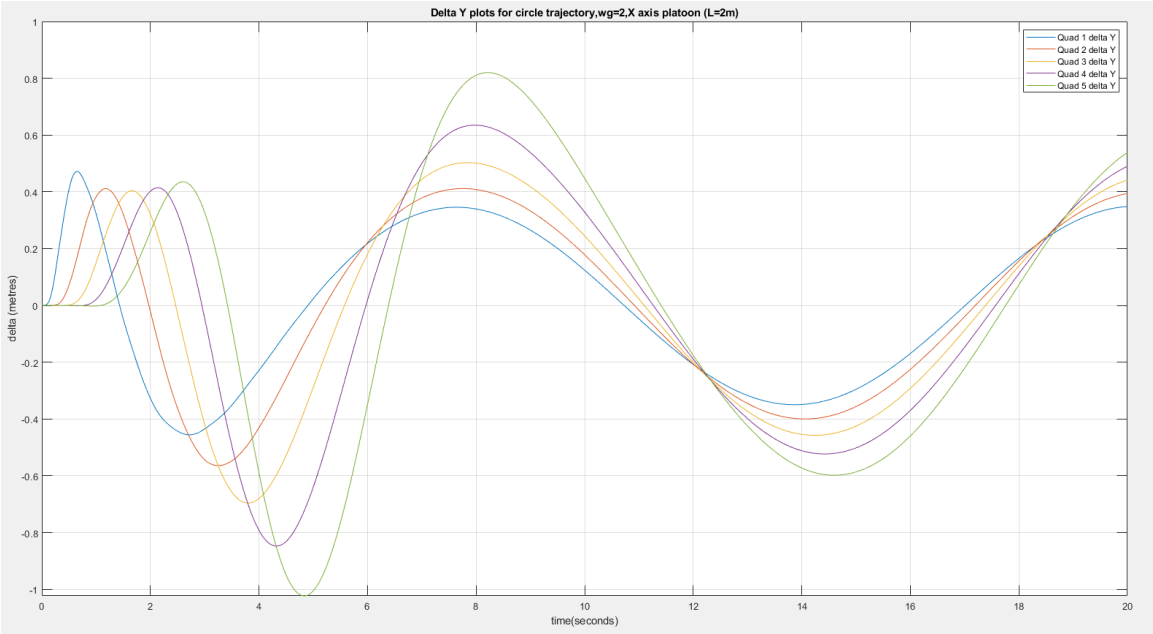

Figure 7.16: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=2 rad/s

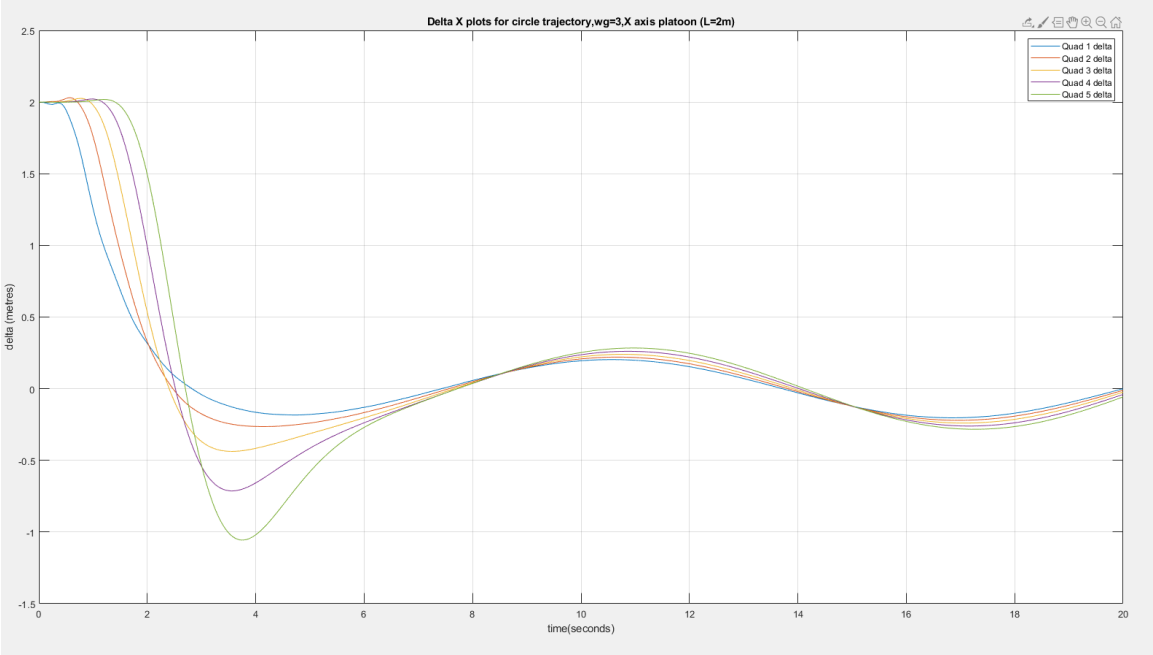

Figure 7.17: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=3 rad/s

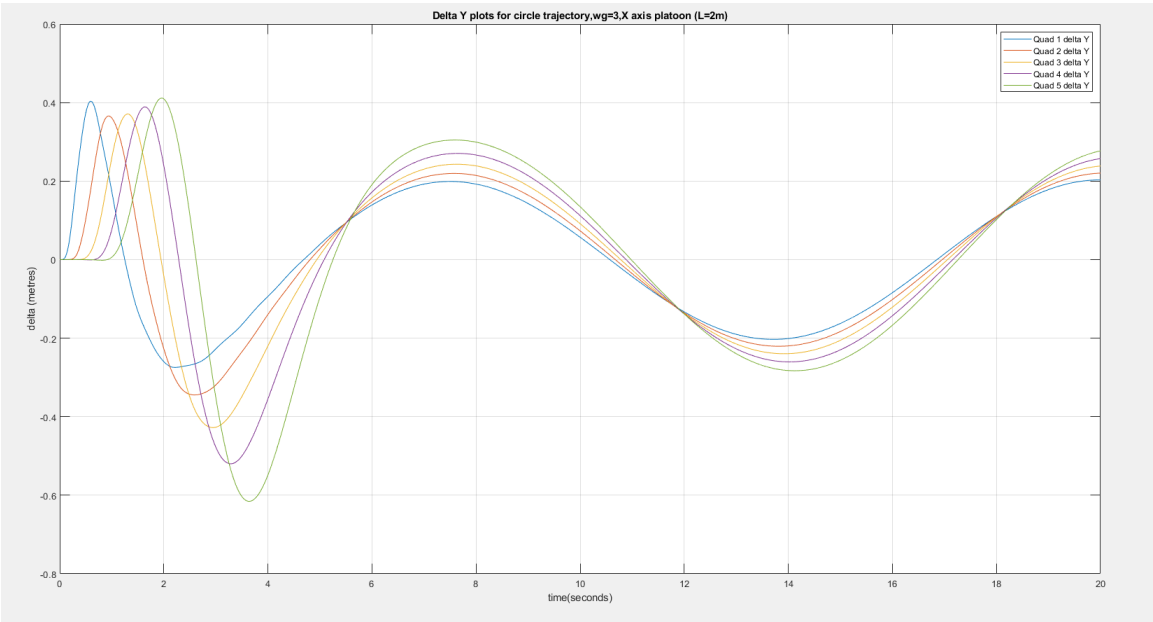

Figure 7.18: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=3 rad/s

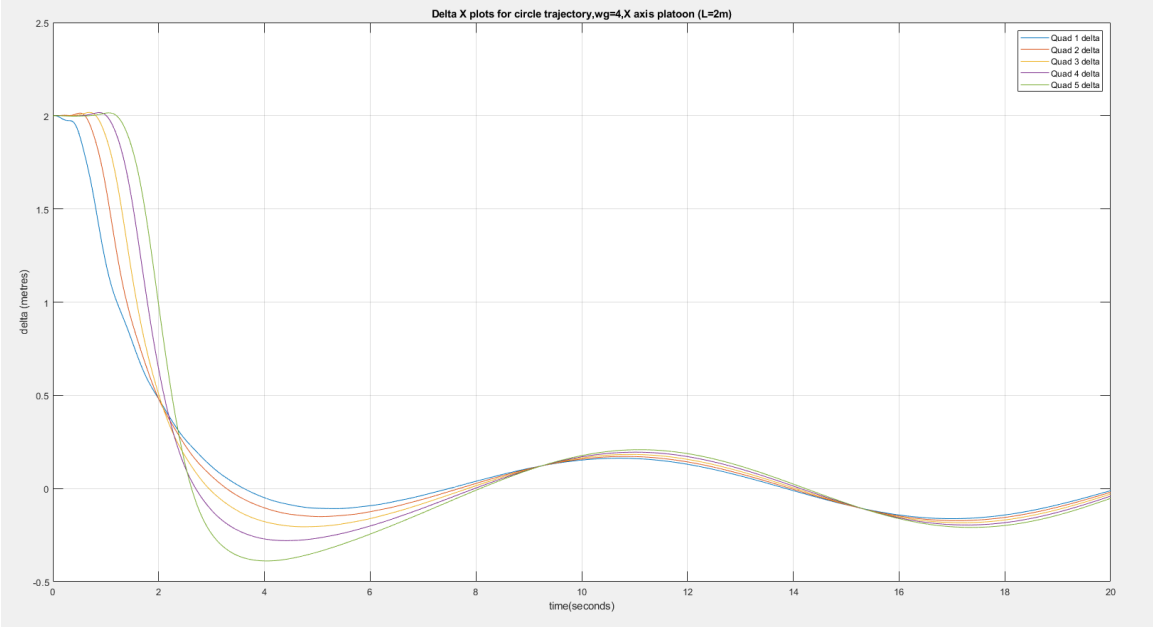

Figure 7.19: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=4 rad/s

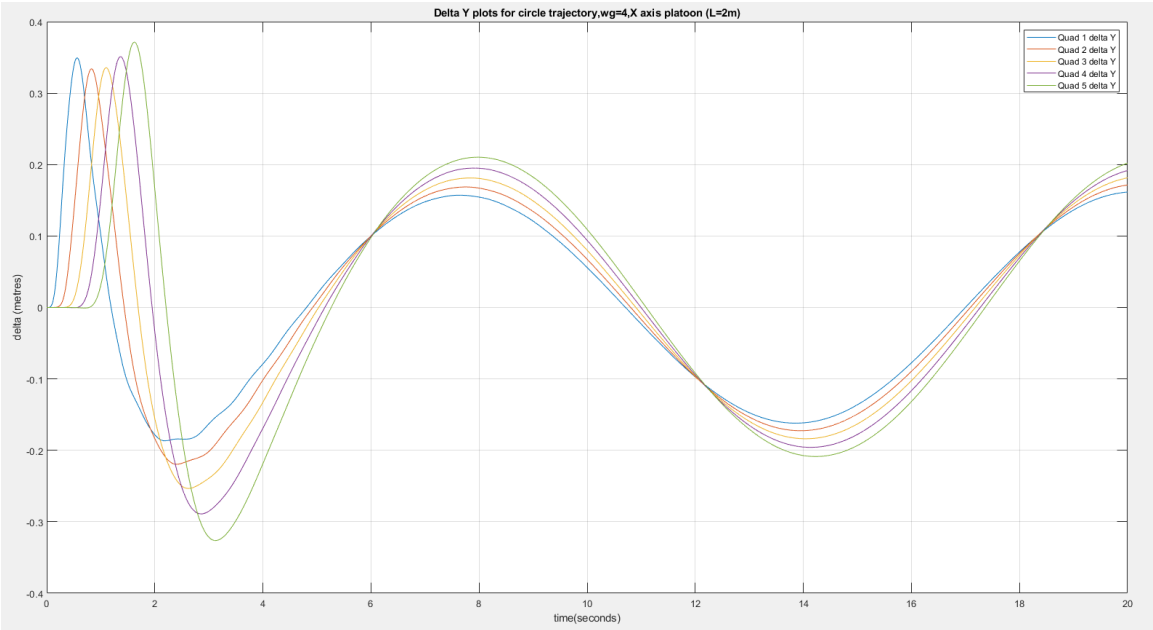

Figure 7.20: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Circular/Curved Trajectory, ω_g=4 rad/s

7.4.3 Analysis of Steady State Δ Error in Curve Path

From the figures presented in the previous section, it is observed that when moving along a curve, the controller does not drive steady state separation to 0 in the presence of the accordion effect. The steady state response shows oscillatory behavior, which is amplified downstream until the last follower. It is also observed that as the bandwidth of the controller is increased, the peak value of steady state oscillation decreases and convergence is achieved faster between all quadcopter trajectories.

Based on these observations, the Δ_x and Δ_y responses are observed when commanded velocity and radius of curvature for curved path are varied independently, to determine suitable controller bandwidth requirements in order to stabilize the accordion effect.

7.4.4 Separation Control Along a Curve, No Leader Feedback - Radius of Curvature Sweep with Velocity=1 m/s, w_g=2 rad/s

We first conduct a trade study for varying radius of curvature along a curve path for the quadcopter fleet, given a fixed velocity of 1 m/s. This analysis is conducted for $omega_g$ =2 rad/s. as well as for ω_g =3 rad/s to study the effect of controller bandwidth on peak steady state oscillation (for Δ_x and Δ_y respectively). The study is first conducted for ω_g =2 rad/s, with simulation results as presented from the facing page.

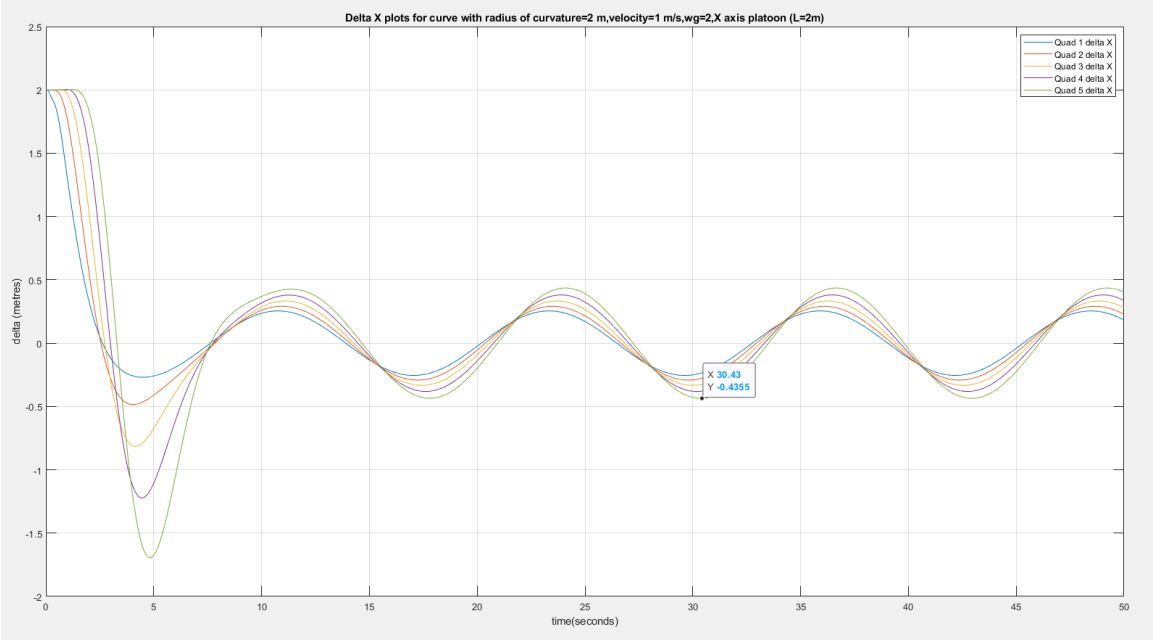

Figure 7.21: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

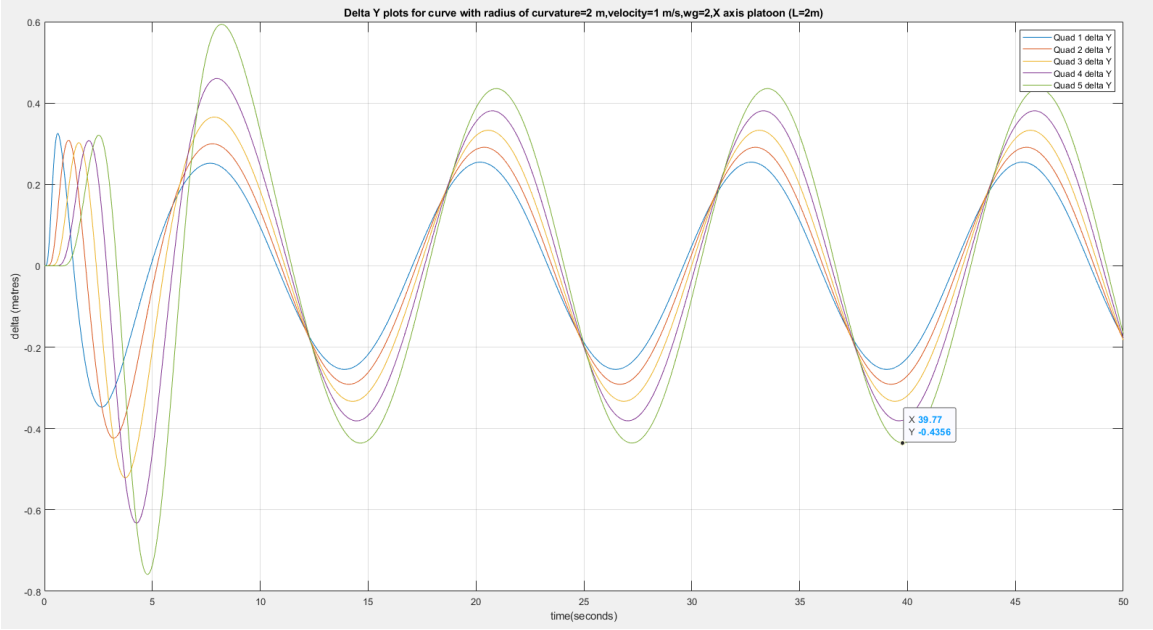

Figure 7.22: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

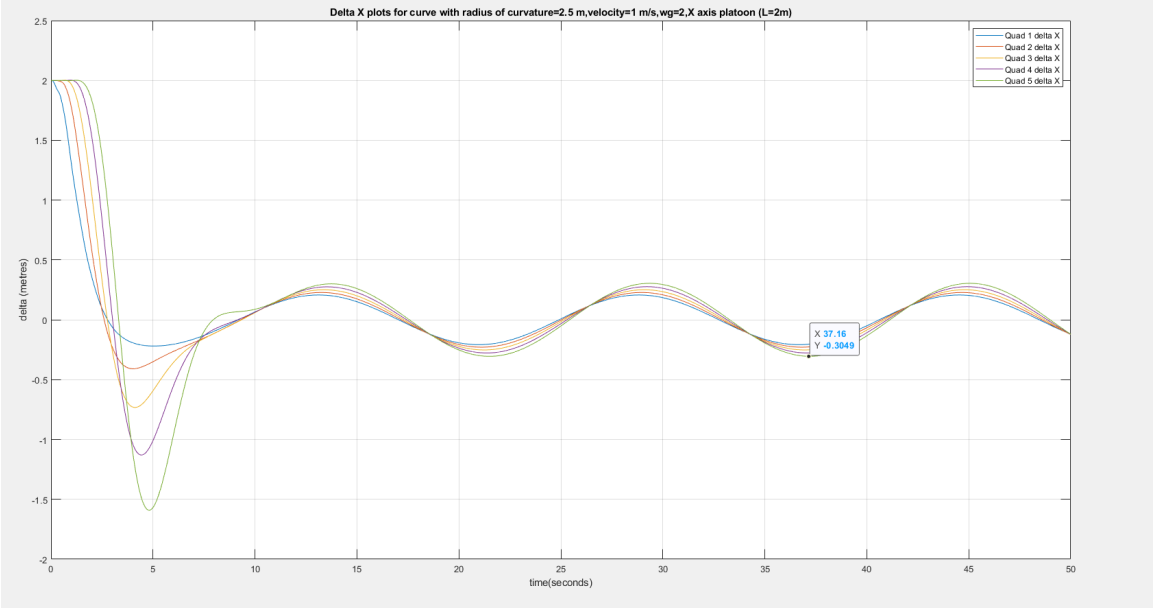

Figure 7.23: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2.5 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

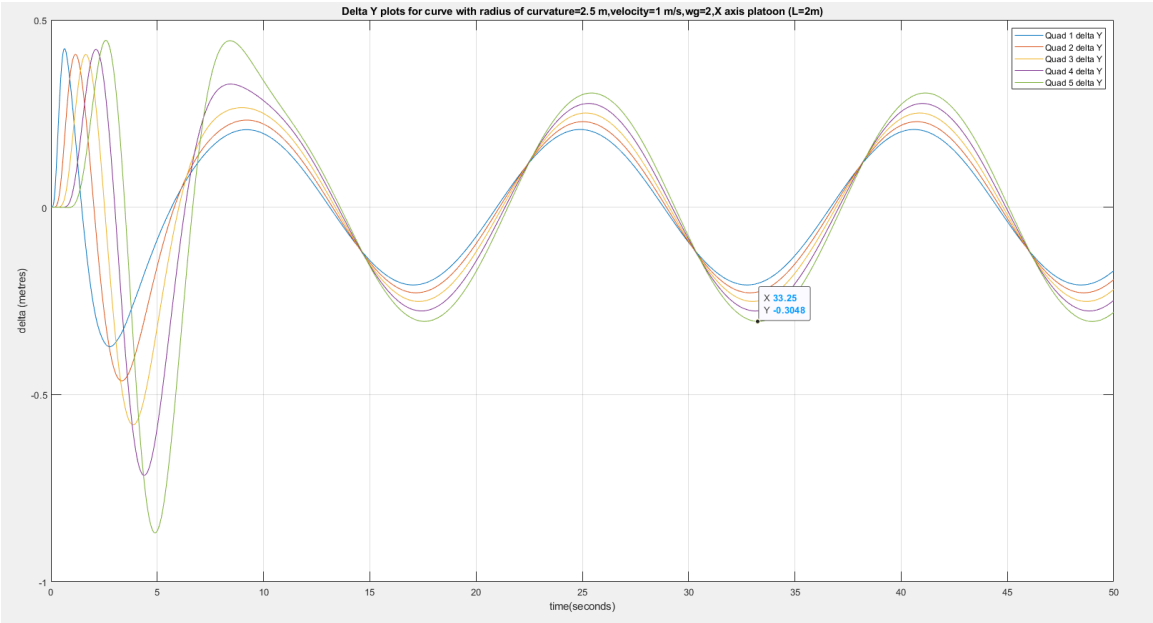

Figure 7.24: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2.5 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

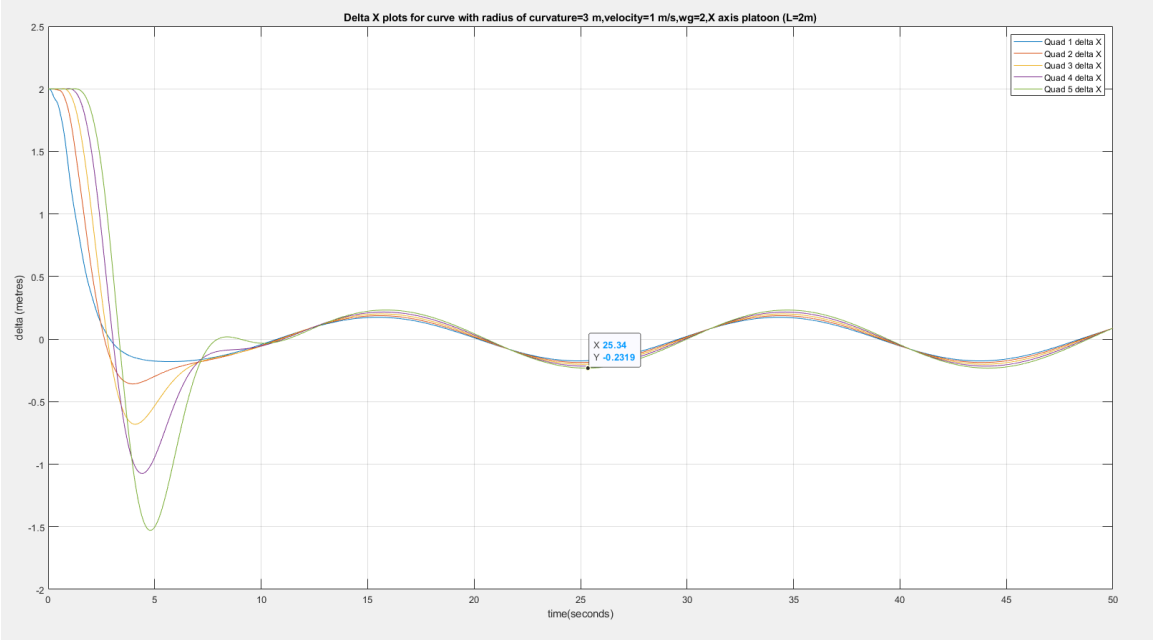

Figure 7.25: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

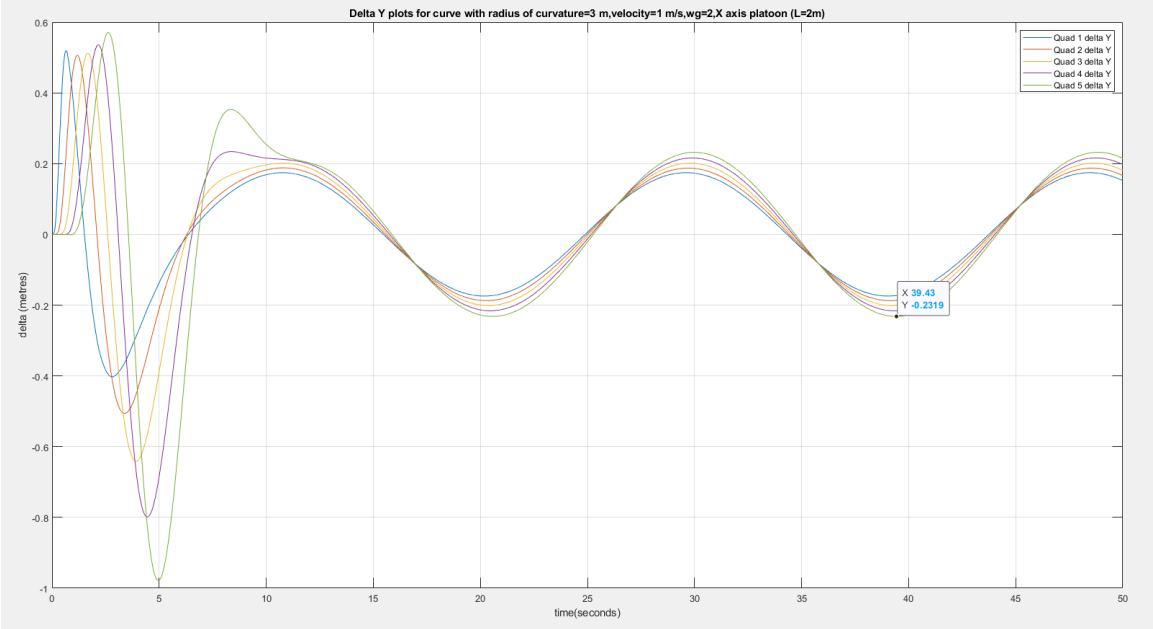

Figure 7.26: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

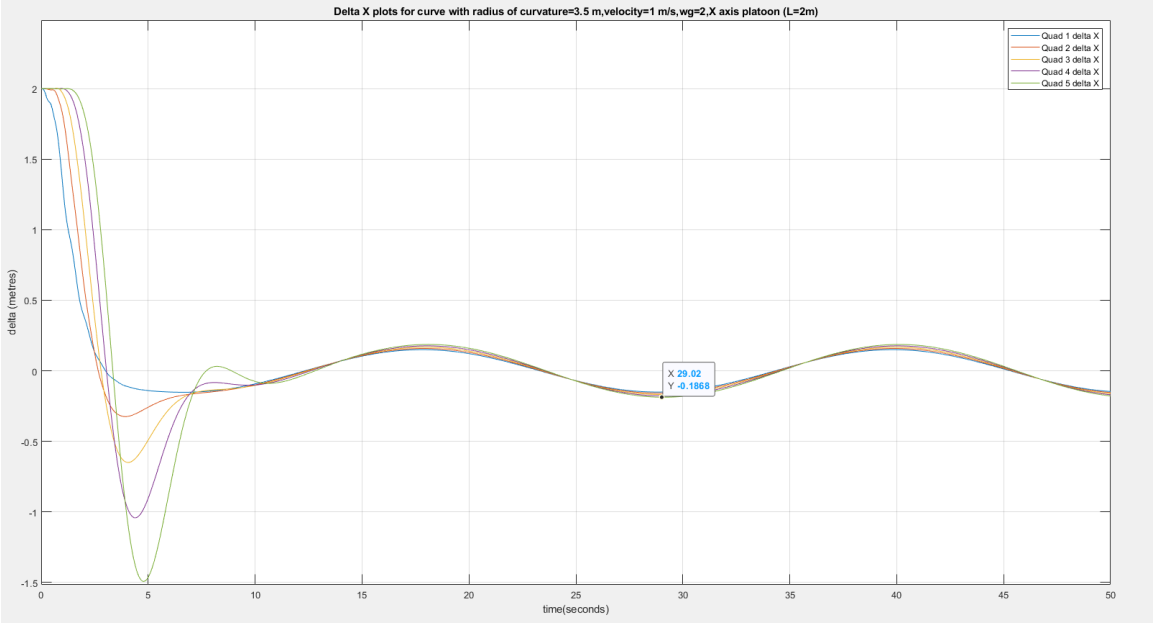

Figure 7.27: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3.5 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

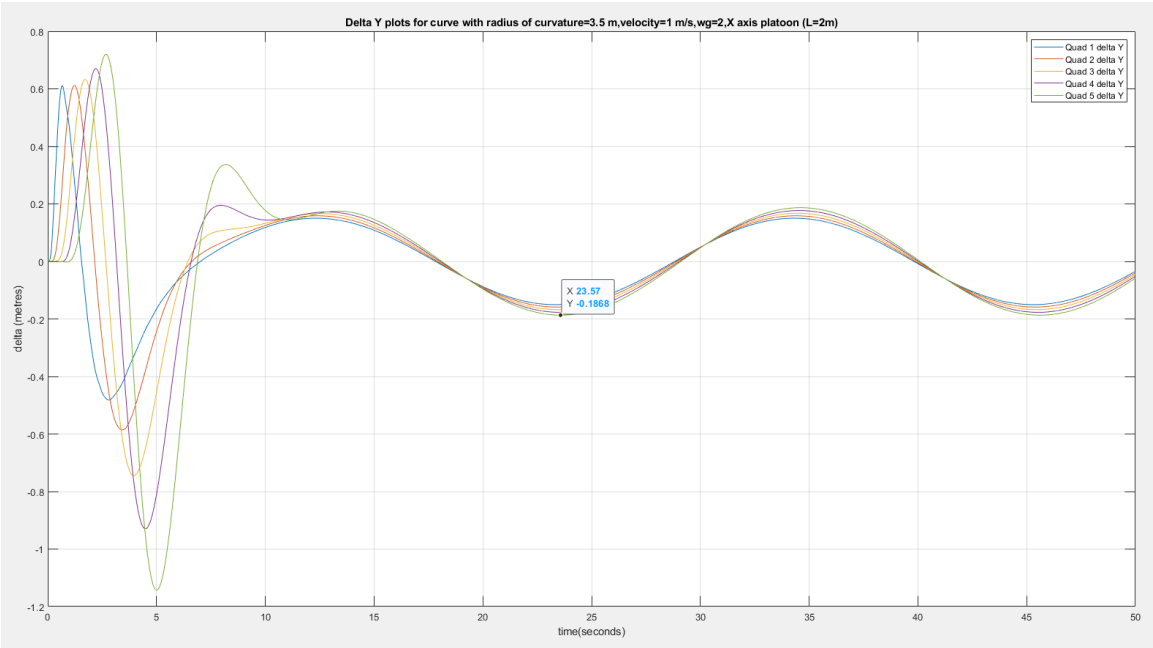

Figure 7.28: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3.5 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 2rad/s$

7.4.5 *Separation Control Along a Curve, No Leader Feedback - Radius of Curvature Sweep with velocity=1 m/s, w_g=3 rad/s*

Having presented the results for ω_g= 2 rad/s in the previous section, the same Radius of curvature sweep is presented for ω_g = 3 rad/s in this section, with plots shown below

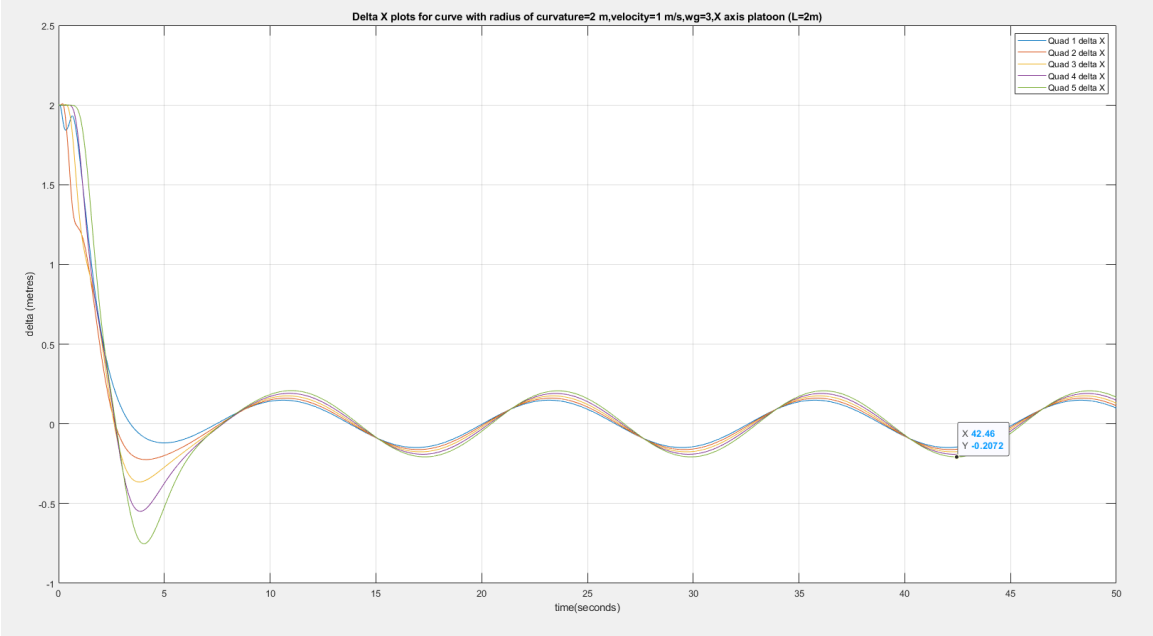

Figure 7.29: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

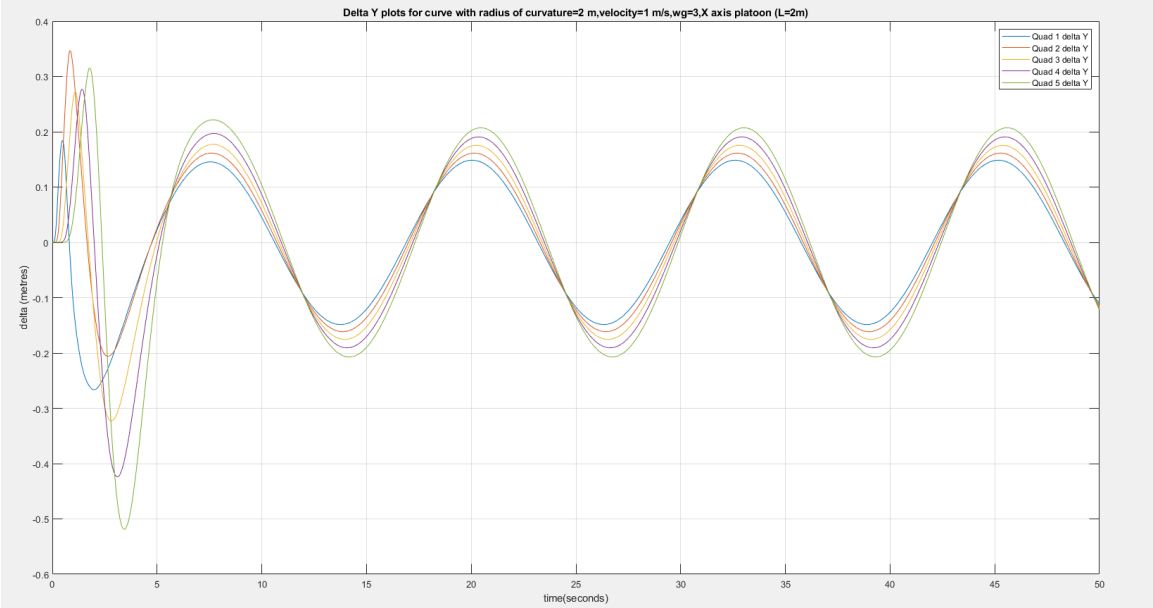

Figure 7.30: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2 m, Velocity = 1 m/s, L (X-axis) = 2 m, $\omega_g = 3rad/s$

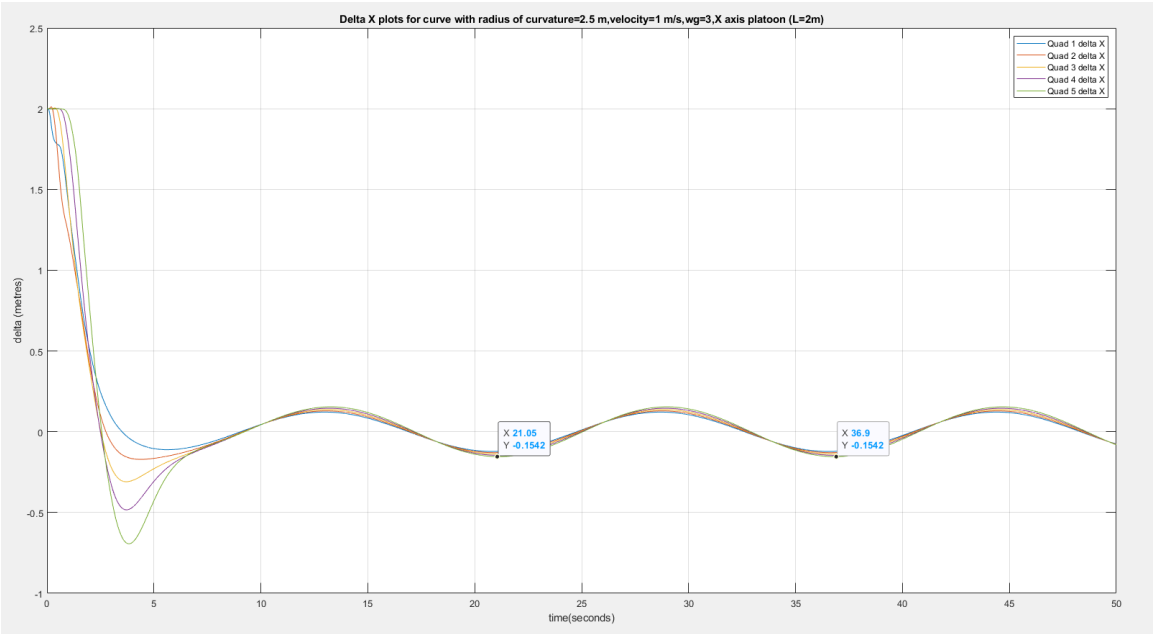

Figure 7.31: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2.5 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

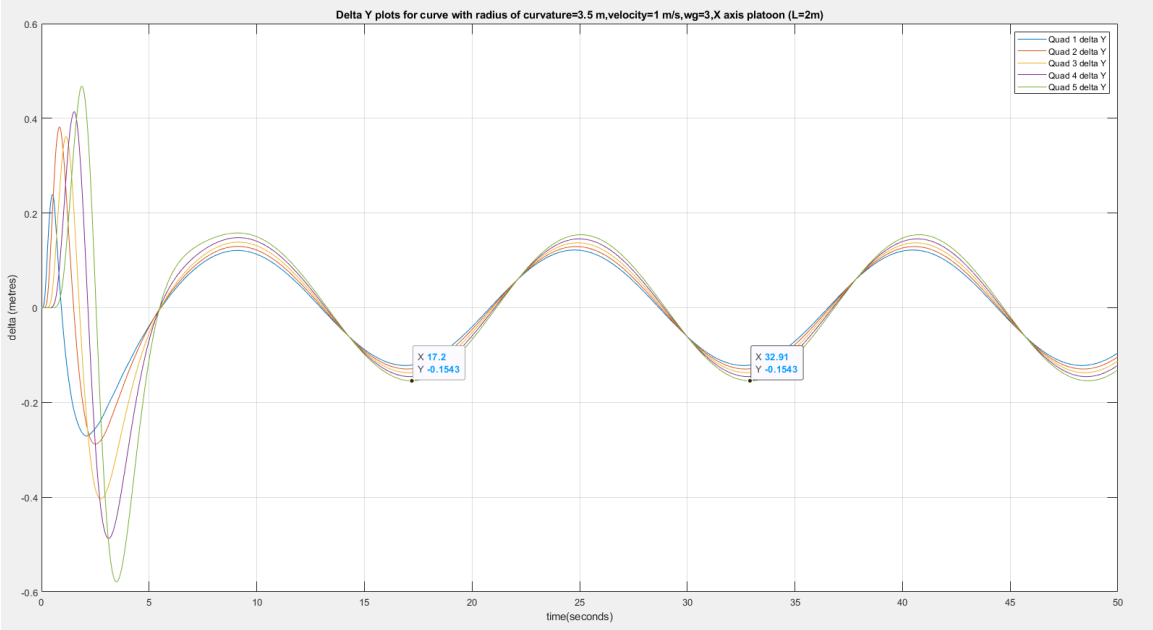

Figure 7.32: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 2.5 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

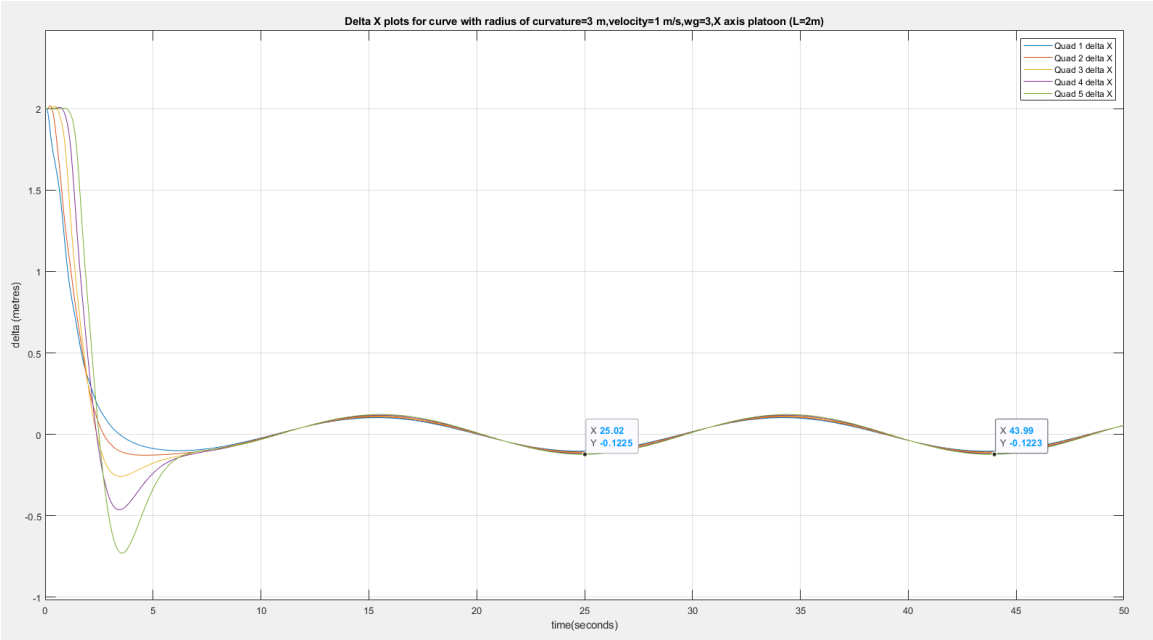

Figure 7.33: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

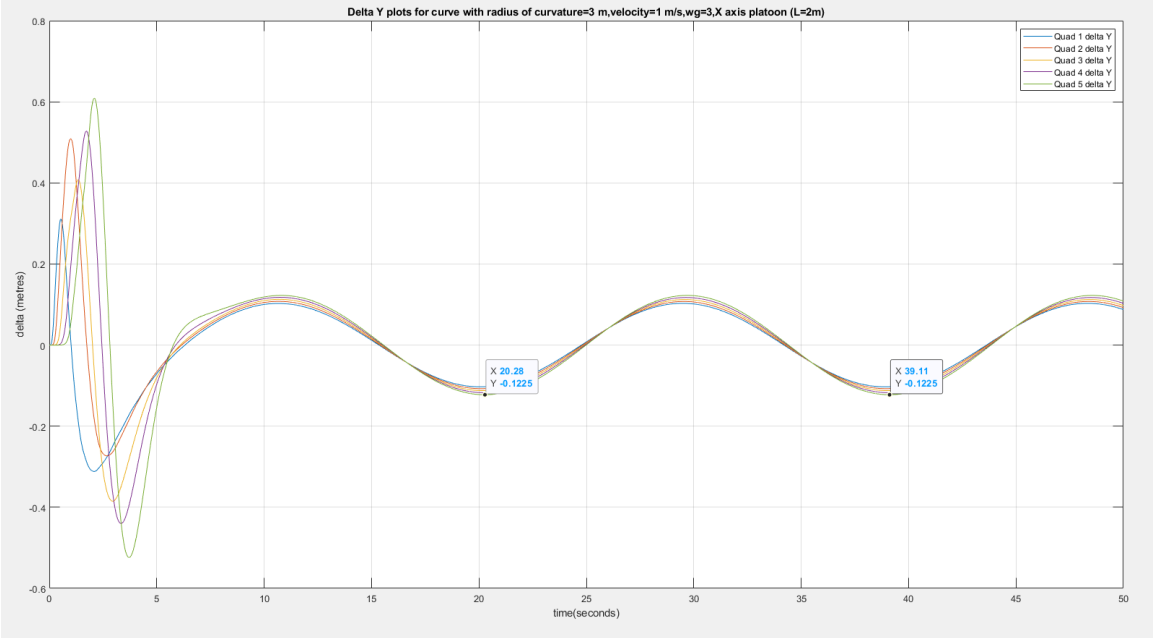

Figure 7.34: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

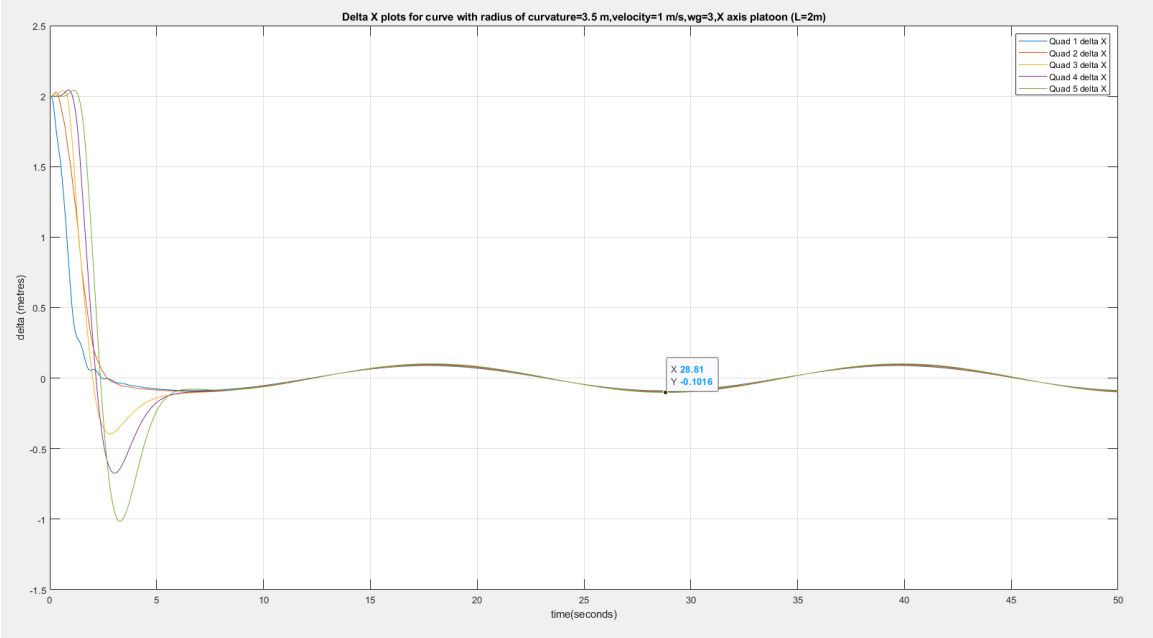

Figure 7.35: PPlot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3.5 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

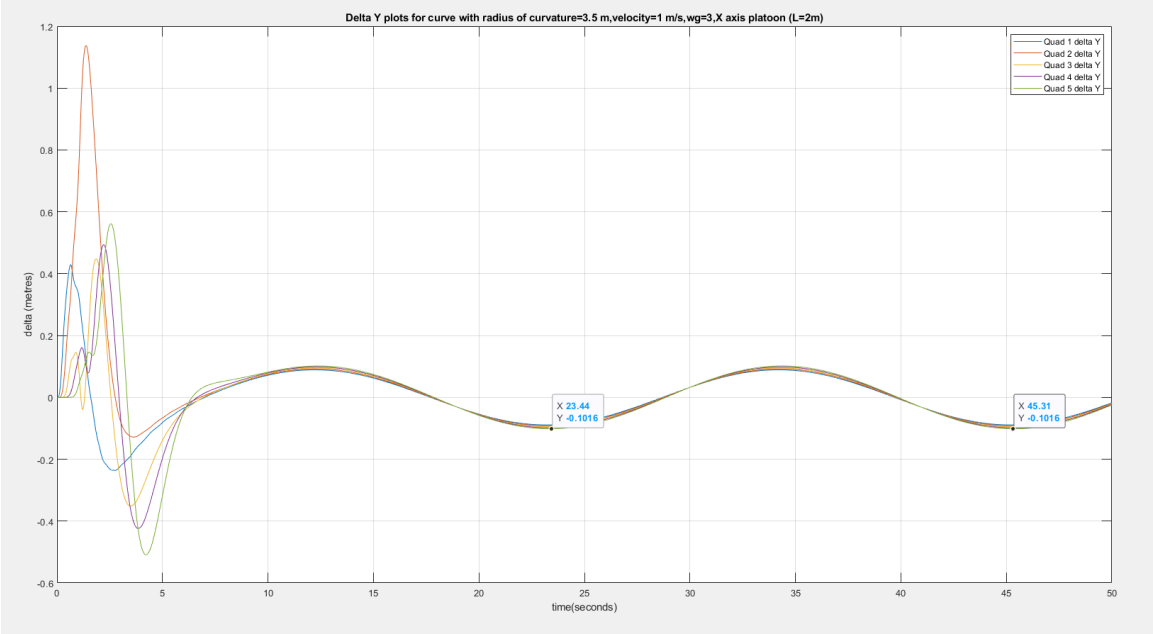

Figure 7.36: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3.5 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

Radius of Curvature Sweep -Key Observations

From the figure presented, it is observed that along a curve, as radius of curvature decreases, steady state delta oscillation peak values increase when velocity is kept constant at 1 m/s(at both controller bandwidth values). The peak values of Δ_x and Δ_y oscillation are nearly identical given a fixed bandwidth and v/R; these values are as documented in Table 7.4

From the table, it is observed that peak Δ oscillation values are reduced by factor of around 0.5 when bandwidth is increased by 1.5 times (from 2 to 3 radians per second). As radius of curvature is increased in 0.5 metre increments, the steady state Δ error reduces by a factor of approximately 0.75.

In order to further analyze curve path oscillation, a velocity sweep is presented for curve trajectories commanded of the lead vehicle.

Table 7.4: Radius of Curvature Sweep - Peak Steady State Δ Values

ROC (m)	**Peak Δ at ω_g = 2 rad/s (m)**	**Peak Δ at ω_g = 3 rad/s (m)**
2	0.43	0.2
2.5	0.3	0.15
3	0.2319	0.1225
3.5	0.1868	0.1016

7.4.6 Separation Control Along a Curve, No Leader Feedback - Velocity Sweep with Radius of Curvature=3 m, w_g=2 rad/s

To perform a velocity sweep, the radius of curvature for the curve path is fixed at 3 m, with velocity varied from 0.4 m/s to 1 m/s. The simulation results are presented in this section using the controller with $\omega_g = 2$ rad/s

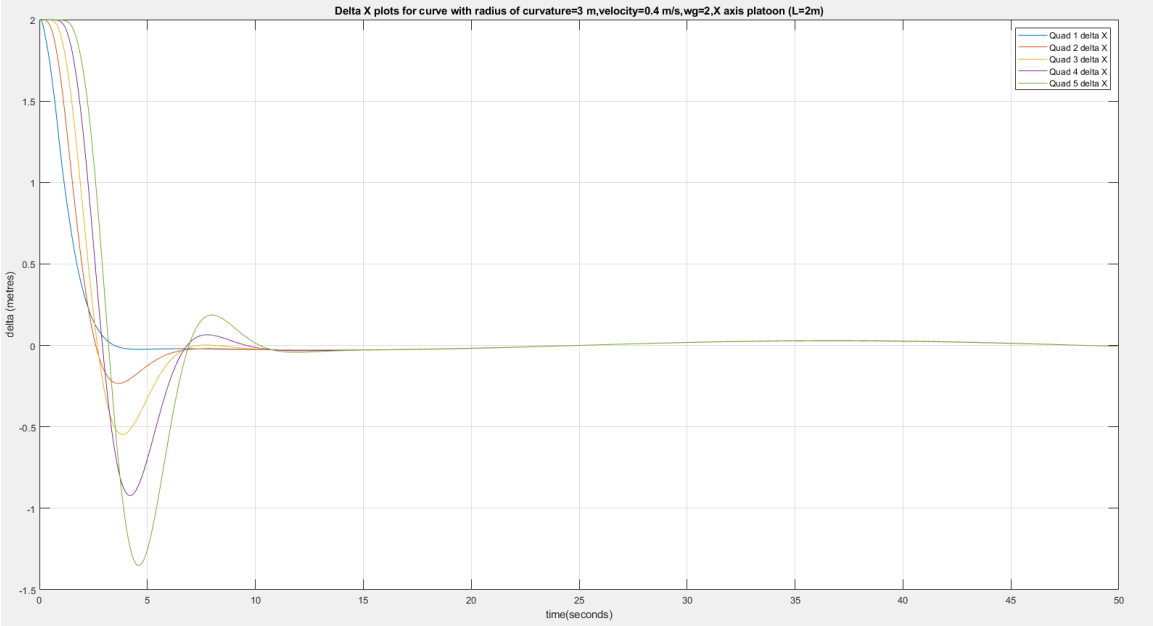

Figure 7.37: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.4 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

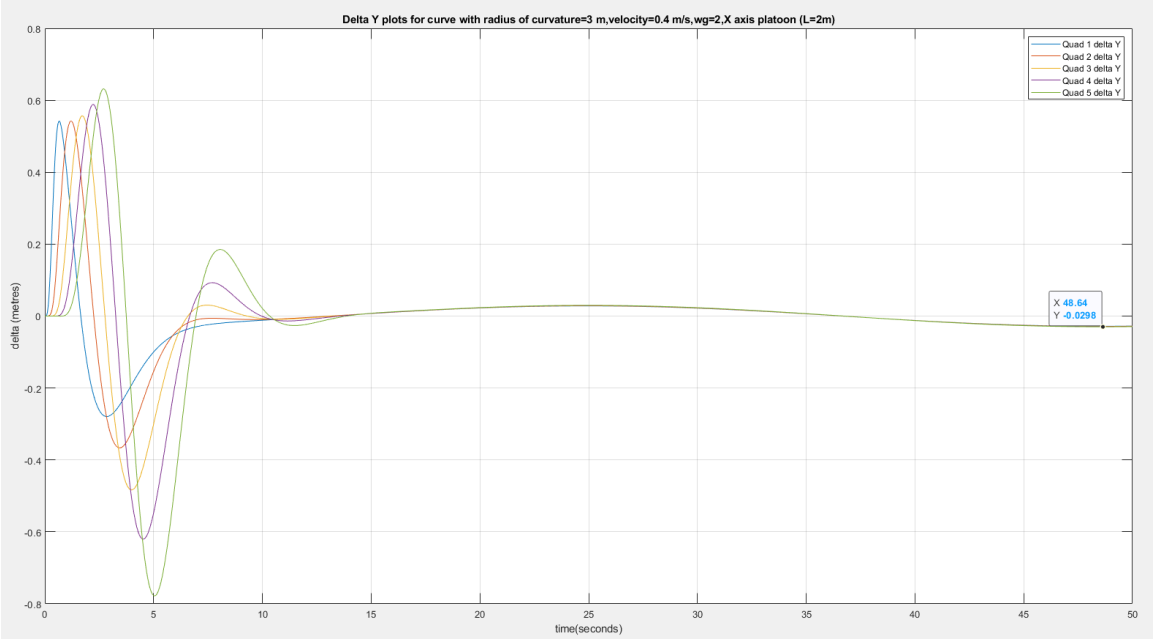

Figure 7.38: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.4 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

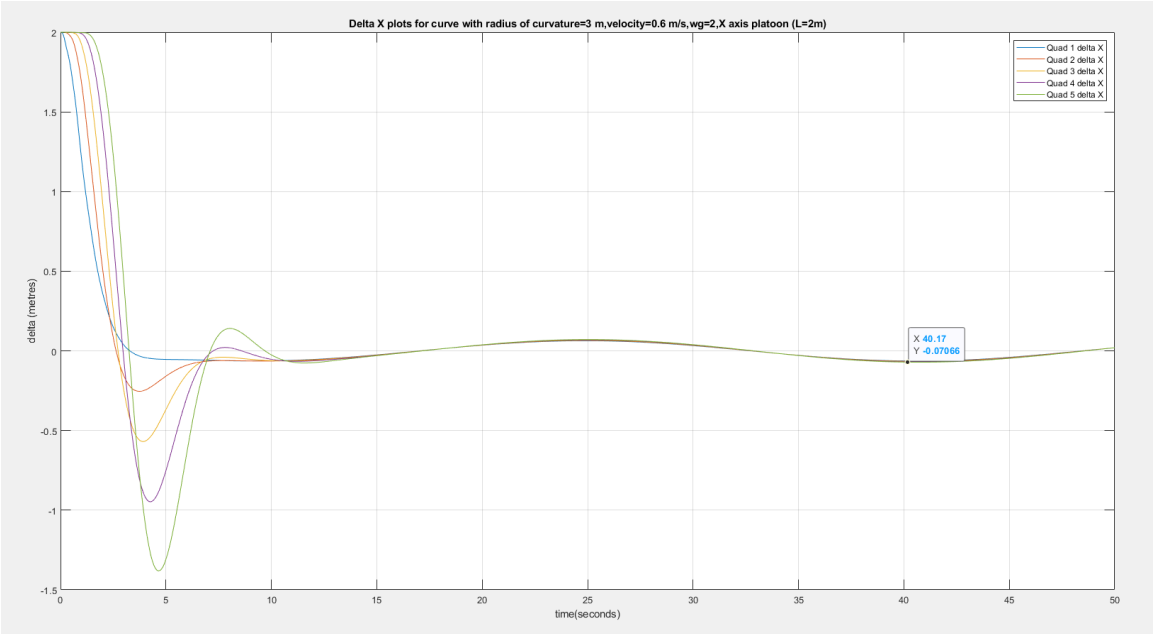

Figure 7.39: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.6 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

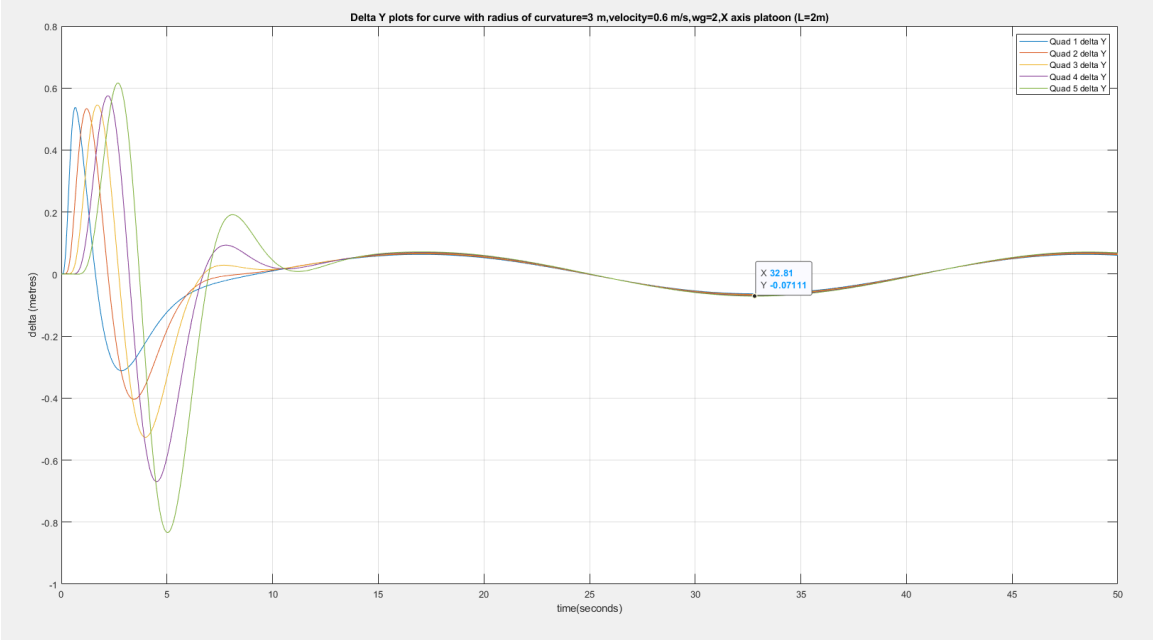

Figure 7.40: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.6 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

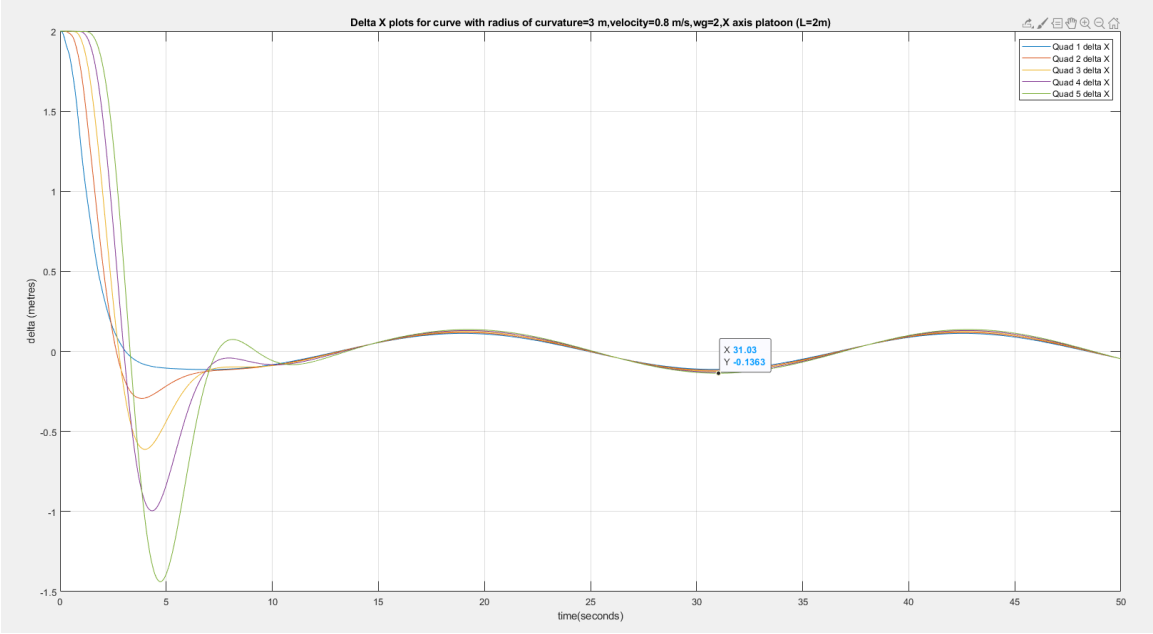

Figure 7.41: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.8 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

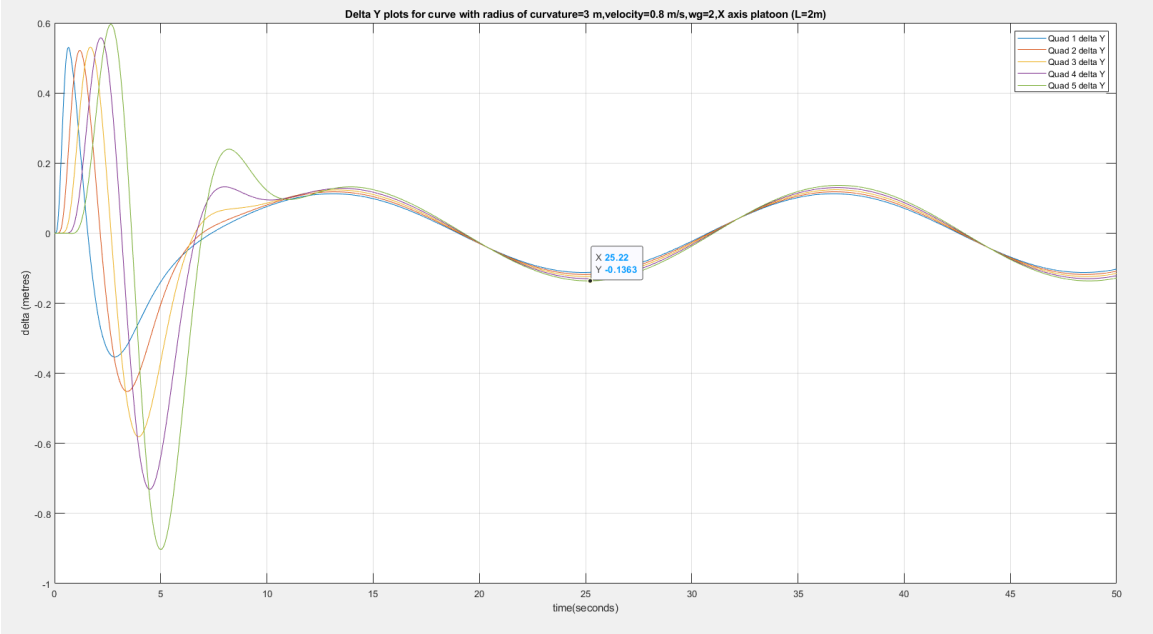

Figure 7.42: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.8 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

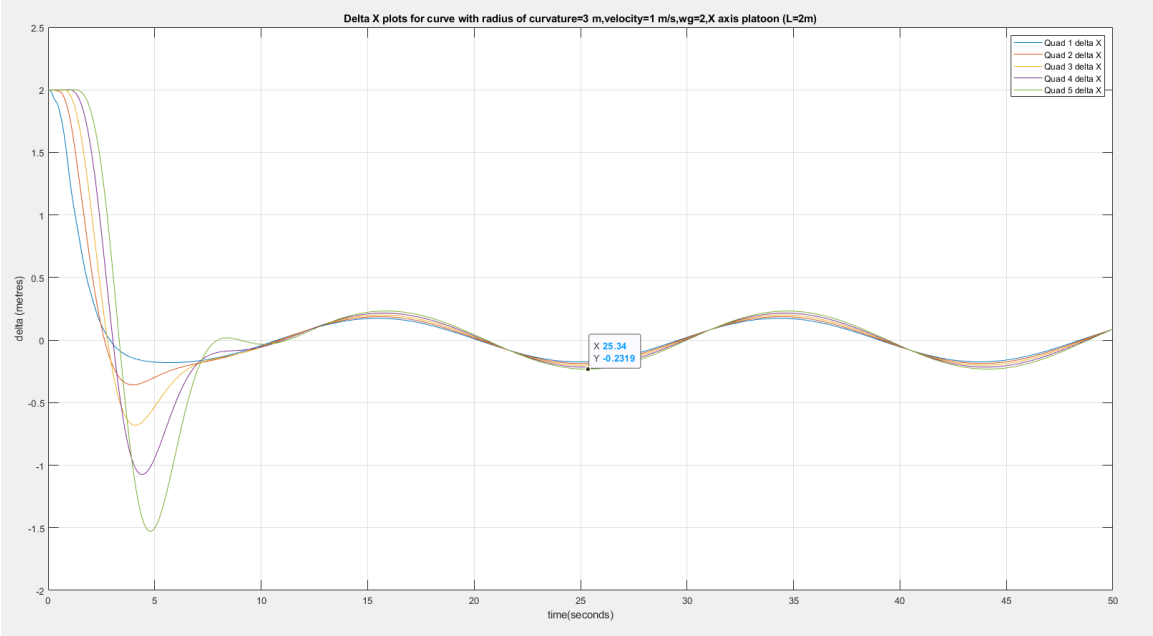

Figure 7.43: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

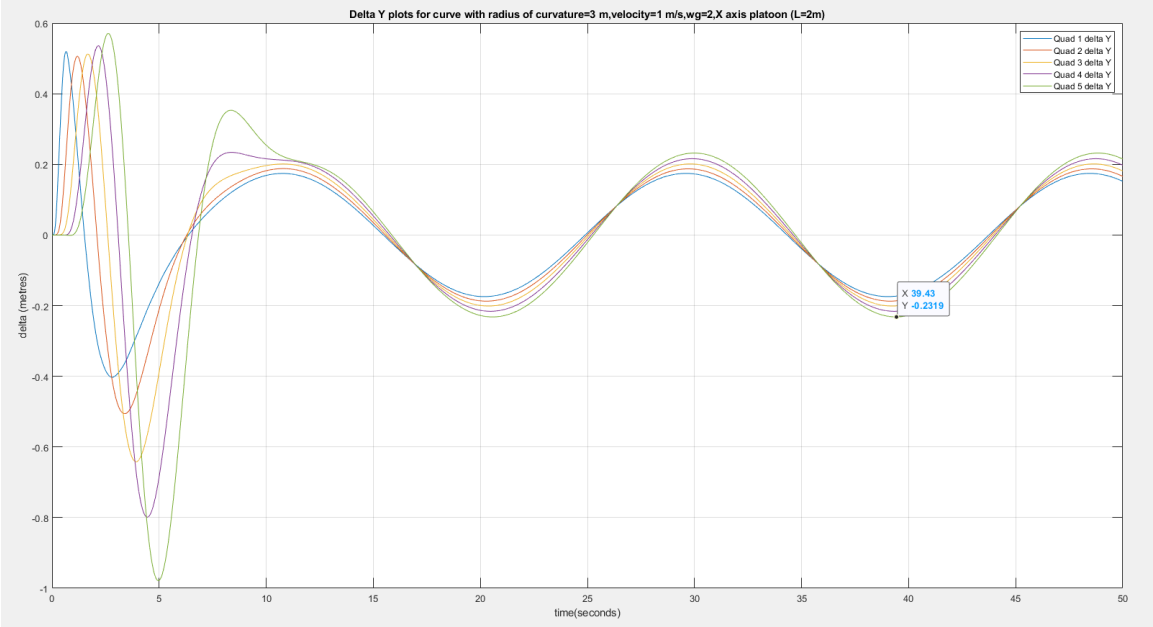

Figure 7.44: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 2rad/s$

7.4.7 Separation Control Along a Curve, No Leader Feedback - Velocity Sweep with Radius of Curvature=3 m, w_g=3 rad/s

The velocity sweep from the previous section is performed again using the controller with $\omega_g = 3$ rad/s, with results presented from the facing page on.

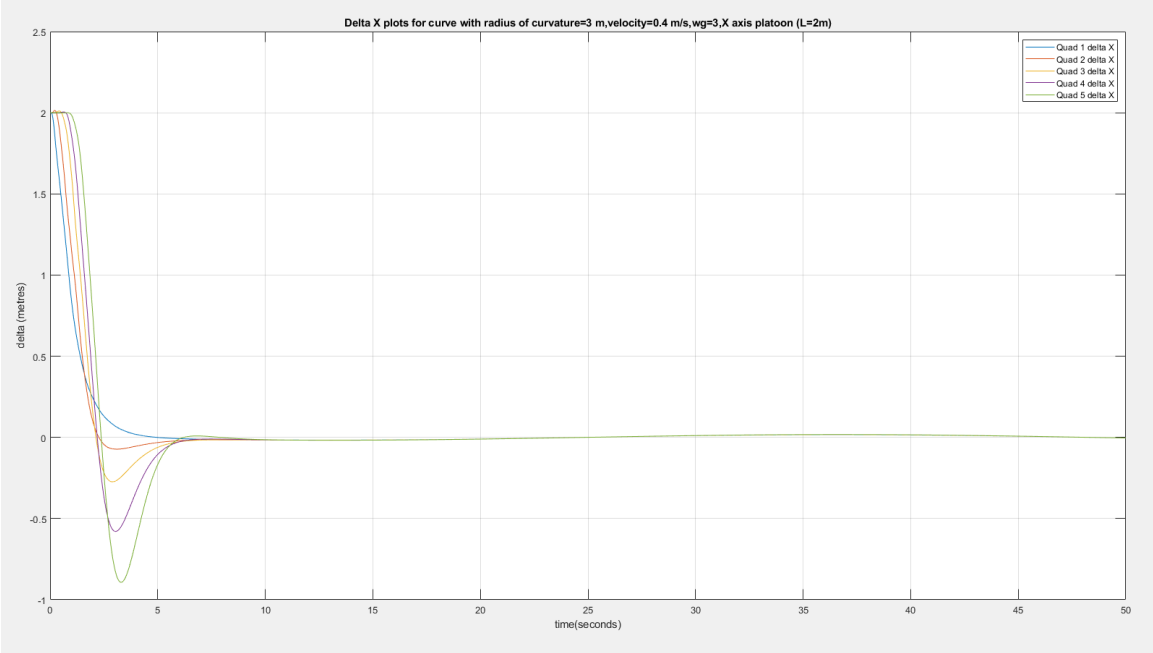

Figure 7.45: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.4 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

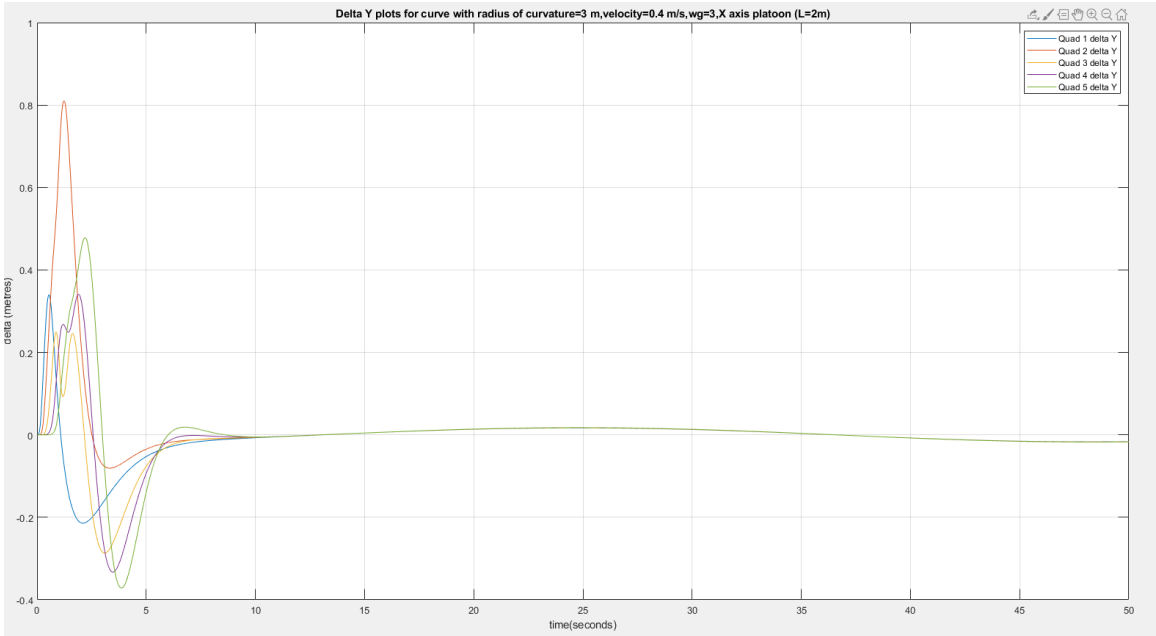

Figure 7.46: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.4 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

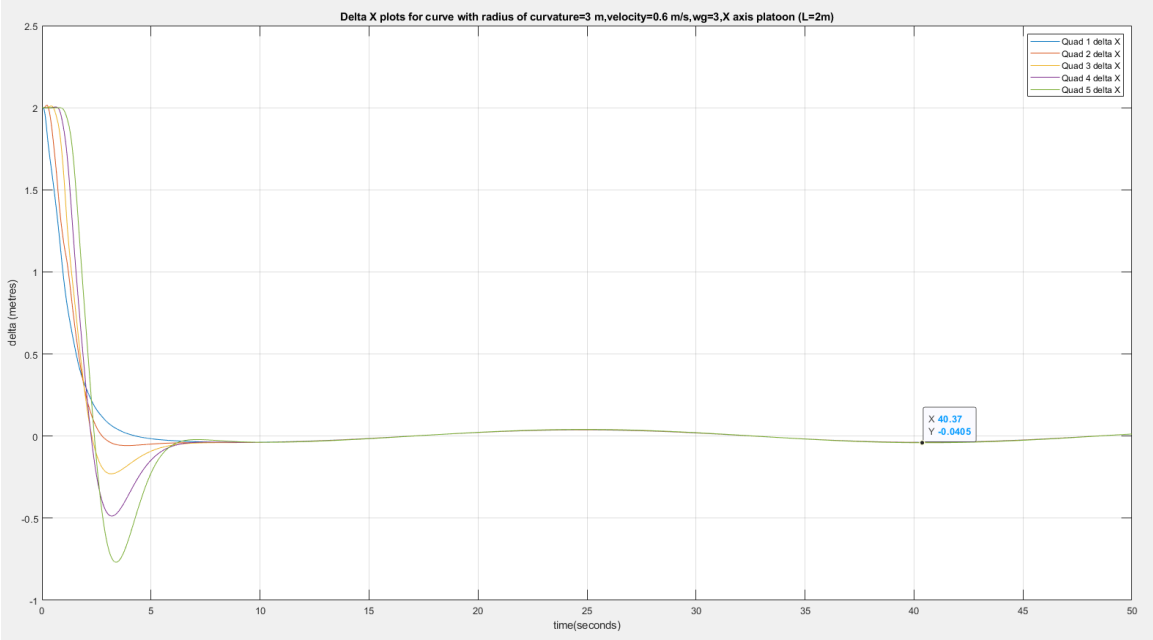

Figure 7.47: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.6 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

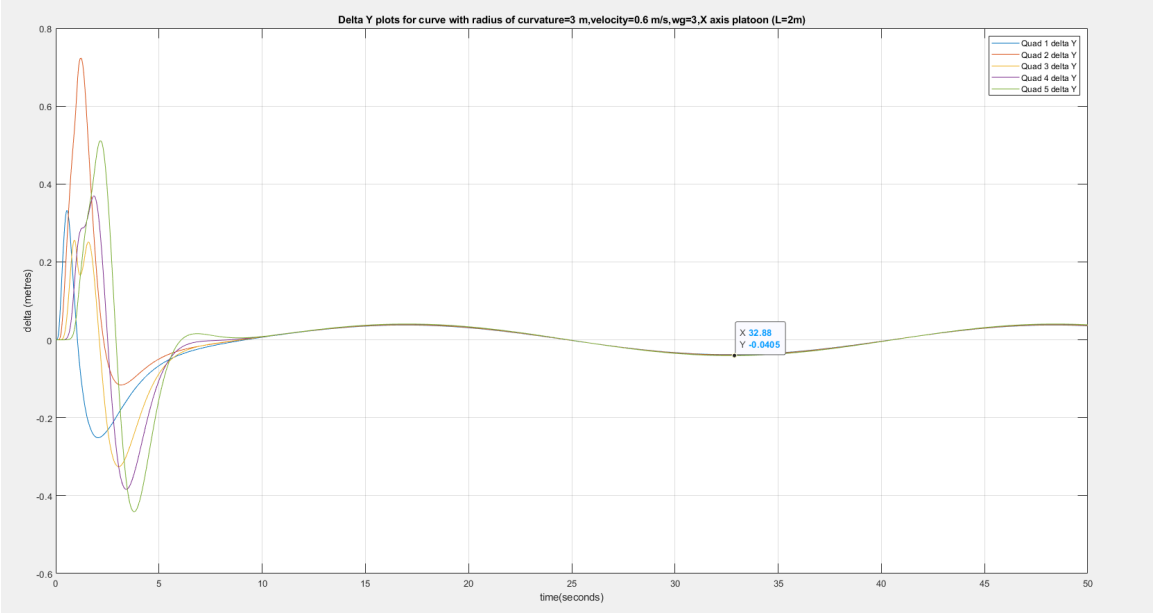

Figure 7.48: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.6 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

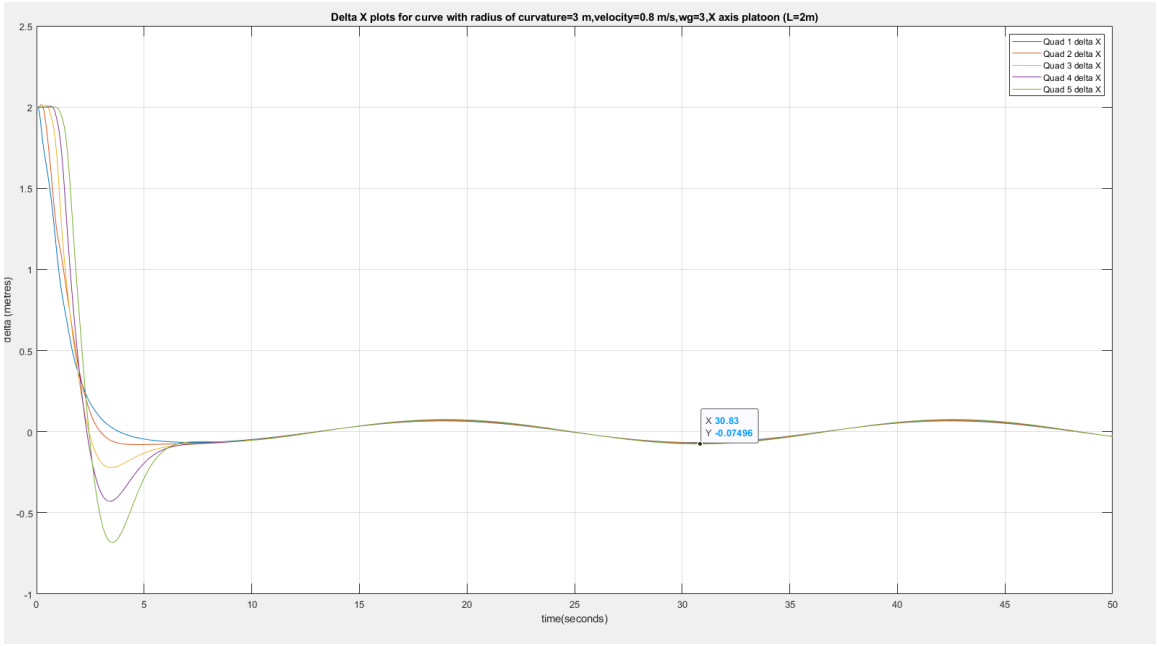

Figure 7.49: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.8 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

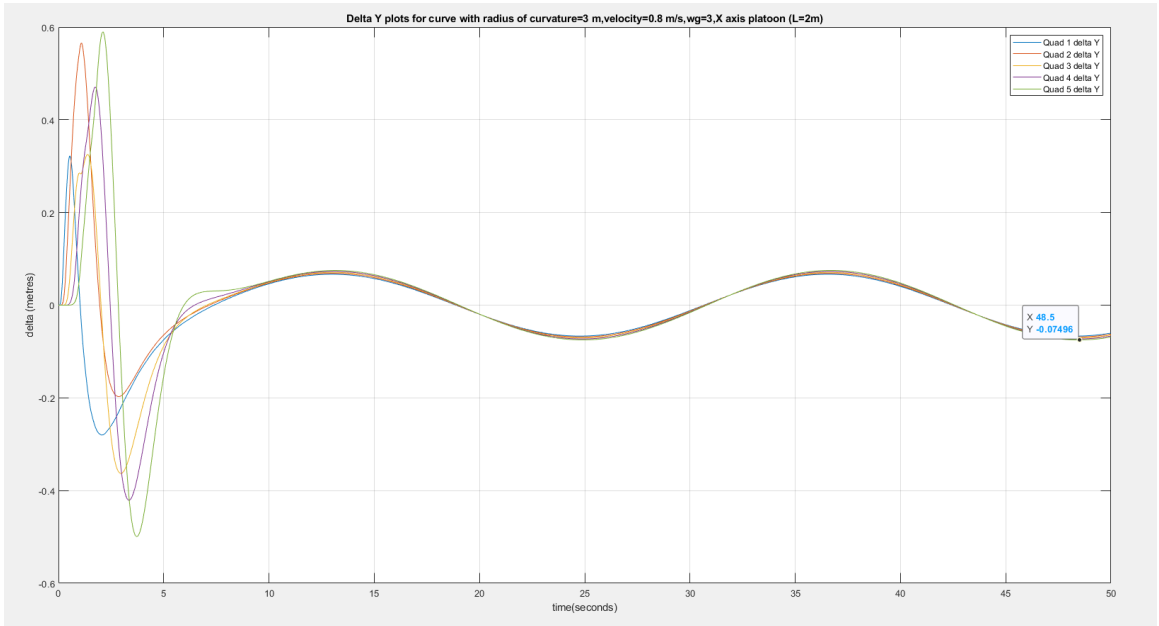

Figure 7.50: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 0.8 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

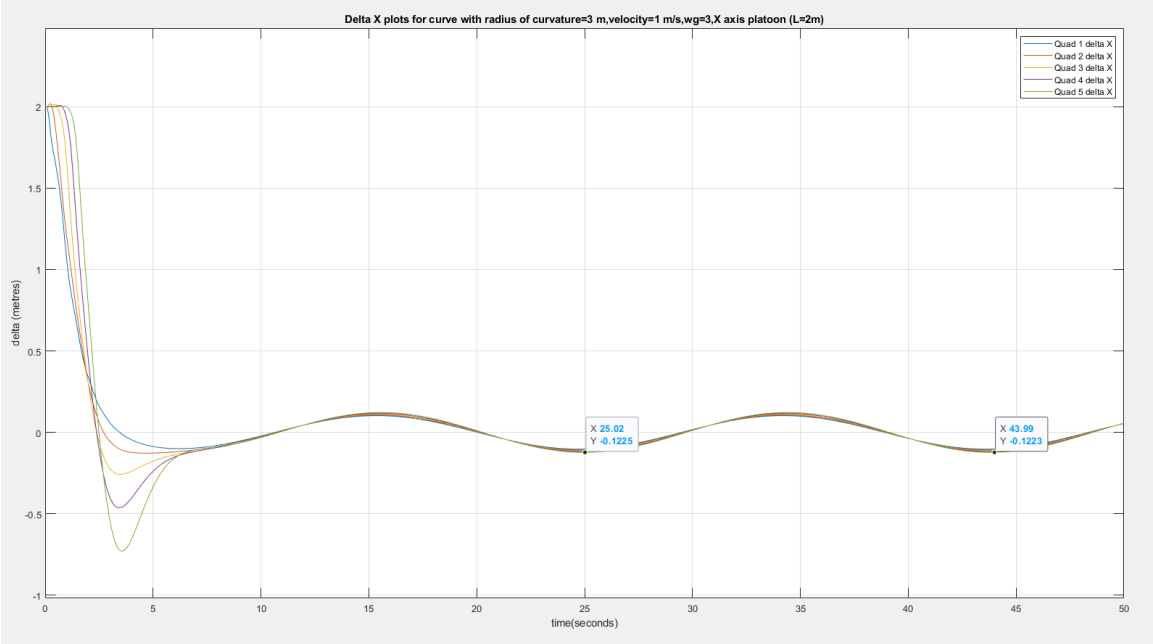

Figure 7.51: Plot of Δ_x (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

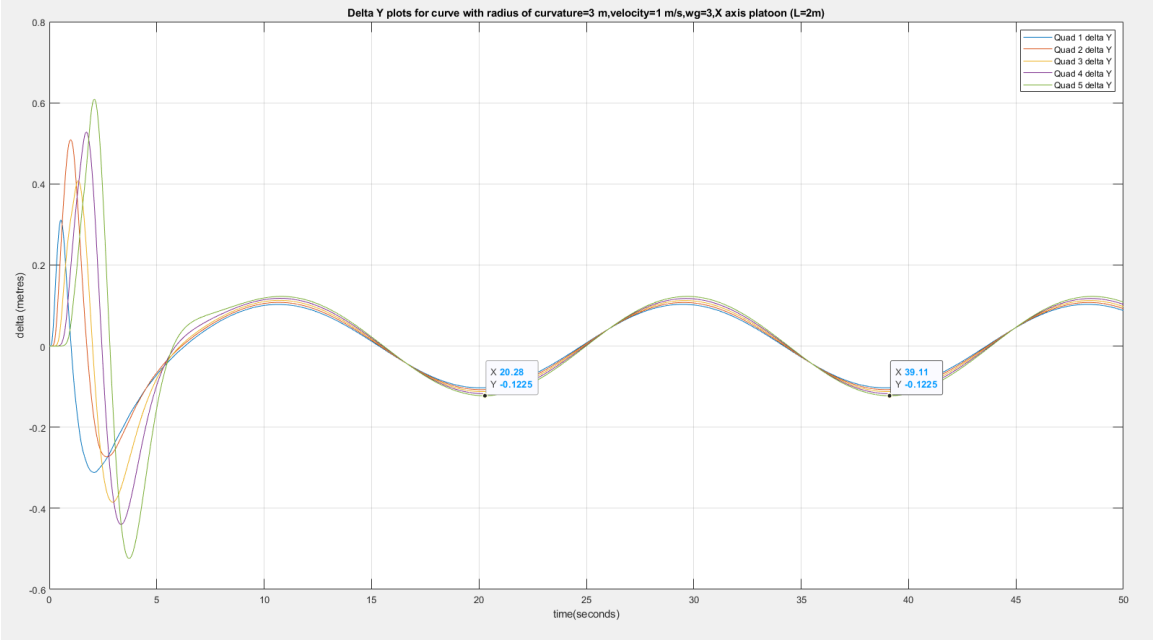

Figure 7.52: Plot of Δ_y (m) vs. Time (s) for 6 Quadcopter Platoon, Curve Trajectory with Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m $\omega_g = 3rad/s$

Velocity Sweep - Key Observations

From the figure presented, it is observed that along a curve, as velocity increases, steady state delta oscillation peak values increase as well when the radius of curvature is kept constant at 3 m (for both controller bandwidth values). The peak values of Δ_x and Δ_y oscillation are nearly identical, and are documented in Table 7.5

Table 7.5: Velocity Sweep - Peak Steady State Δ Values

Velocity (m/s)	**Peak Δ (m), ω_g = 2 rad/s**	**Peak Δ (m), ω_g = 3 rad/s**
1	0.232	0.1225
0.8	0.1363	0.075
0.6	0.07	0.04
0.4	0.03	0.017

From the table, it is observed that peak Δ oscillation values are reduced by factor of around 0.5 when bandwidth is increased by 1.5 times (as in the radius of curvature sweep) but the effect of changing velocity on steady state Δ appears more significant. As velocity is decreased by 0.2 m/s, the steady state Δ error reduces by a factor of approximately 0.5 per decrement, which is a larger factor of reduction proportional to change in velocity (as compared to 0.75 factor for 0.5 m decrements in radius of curvature). In general, a relationship is observed and established between the desired v/R moving along a curve, and the bandwidth required to maintain a low steady state deviation. An attempt is made to quantify this relationship in the next section.

7.5 Summary of Nominal Platoon Model - Comparison with Leader Feedback Model

From the previous section, it is observed that the PD controller design works well in stabilizing the accordion effect for a quadcopter platoon, when a straight line path is commanded of the leader, with steady state Δ_x and Δ_y converging to zero irrespective of bandwidth. Increasing controller bandwidth upto 4 radians per second is found to decrease the convergence time for all quadcopters to reach steady state Δ_x of 0.

When commanding a curve path for the leader, as described at the end of the previous section, it is observed that a steady state Δ arises in all follower vehicles, with largest deviation observed in the last quadcopter. This is due to the fact that along a curve, while translational velocity may be constant, the actual (x, y, z) acceleration components are periodic, and as such there are constant changes in lead vehicle acceleration specifically. The value of steady state delta is observed to be inversely proportional to the radius of curvature and the bandwidth of the system (approximately), but proportional to the velocity of the platoon along the curve. It is noted that along a straight line path the radius of curvature is effectively infinite, and steady state delta converges to zero with no oscillation even at higher velocities.

Based on these results, a trade study is conducted by varying $\frac{v}{R}$ where v is the velocity along a curve and R is the radius of curvature. As$\frac{v}{R}$ corresponds to a frequency term, the objective of the analysis is to determine a possible relationship between this term and the frequency of ω_g (or the control bandwidth). It is noted that as v is measured in m/s and R is measured in m, the $\frac{v}{R}$ term has unit of Hertz, and so it is converted to rad/s in order to compare with ω_g frequency.

With $\frac{v}{R}$ varied from 0.7 to 2 (in radians per second), the minimum bandwidth

required for a maximum steady state Δ error of 5% is recorded; in this case, with L = 2 m, it is assumed that the maximum error bound is $[-0.1, 0.1]$. The bandwidth values obtained for each $\frac{v}{R}$ are documented in Table 7.6, and visualized in Figure 7.53 .

Table 7.6: $\frac{v}{R}$ Sweep - Required Bandwidth for 5% Peak Steady State Δ Error

SI No.	$\frac{v}{R}$**(rad/s)**	ω_g **for 5% maximum error (rad/s)**
1.	0.7	0.8
2.	0.8	0.9
3.	1.0	1.2
4.	1.2	1.5
5.	1.4	1.8
6.	1.6	2.2
7.	1.8	2.6
8.	2.0	3
9.	2.1	3.2

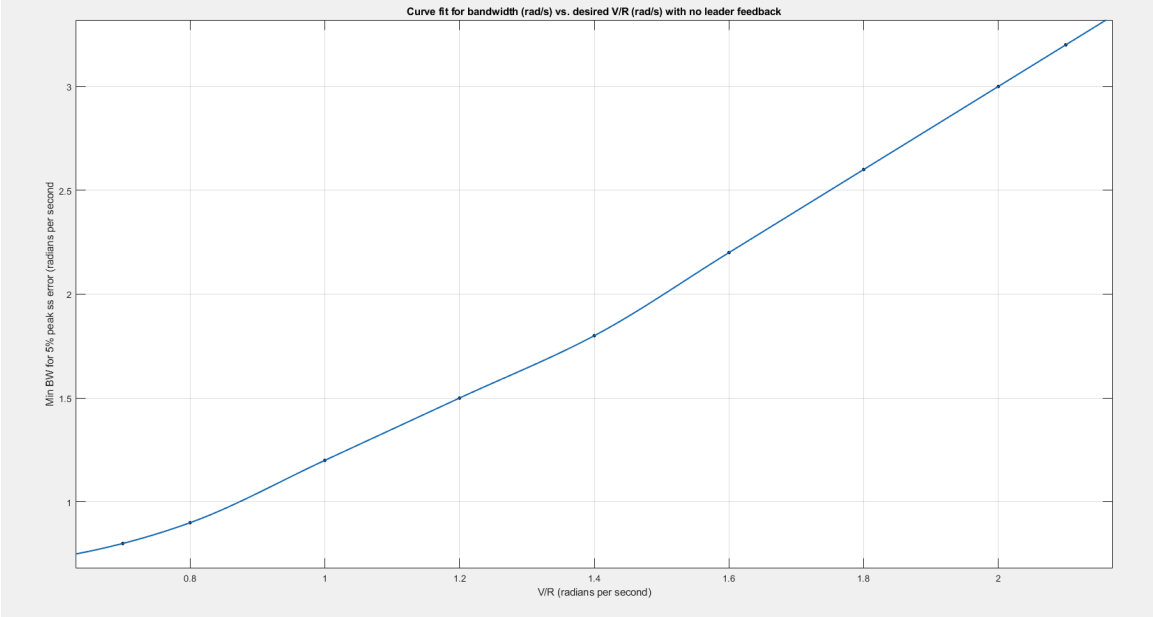

Figure 7.53: $\frac{v}{R}$ vs Bandwidth with No Leader Feedback- Visualization

From Figure 7.53, it is observed that a curvilinear relationship can be observed between required separation control bandwidth for the quadcopter fleet, and the $\frac{v}{R}$ ratio for the curve trajectory commanded of the leader.As $\frac{v}{R}$ is increased from 1 to 2, the required bandwidth for 5% peak error goes from around 1.1 times the desired $\frac{v}{R}$ (near $\frac{v}{R} = 1$) to around 1.5 times the desired $\frac{v}{R}$ (near $\frac{v}{R} = 2$) In general, the required bandwidth is approx. 1.3 times desired V/R with no lead feedback information

7.5.1 Leader Feedback Comparison

Having addressed the desired nominal case (with no leader feedback information), we simulate the $\frac{v}{R}$ lead quadcopter acceleration is provided to each quadcopter in the fleet. It is assumed that with lead feedback information, the steady state Δ oscillations reduce and in turn, the required control bandwidth for 5% peak Δ error also will reduce. To test this, lead quadcopter (x, y, z) accelerations are fed forward directly to the PD separation controller on each follower, and reduced steady state errors are observed when acceleration information is fed forward directly (i.e with

no control action), as proportional/PD control is observed to increase Δ convergence time. The general model is shown in Figure 7.54, with $\ddot{x}_l = [\ddot{x}, \ddot{y}, \ddot{z}]$ of the leader vehicle (added to control input)

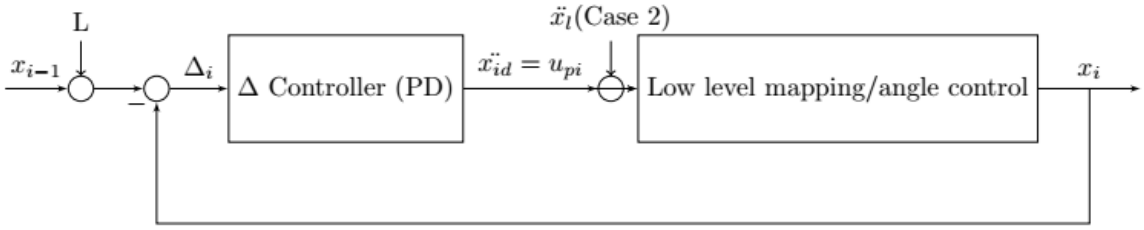

Figure 7.54: Lead Feedback Model- Visualization of i-th Follower

Plots for Lead Acceleration Feedback Model (Comparison with Nominal Model)

With ω_g fixed at 3 rad/s on each separation controller, the Δ_x and Δ_y simulation results are presented below, for a curve path with radius of curvature 3 m and velocity = 1 m/s. These results are shown in parallel with the simulation outputs of the nominal model when using the same control bandwidth and lead vehicle trajectory, in order to compare responses.

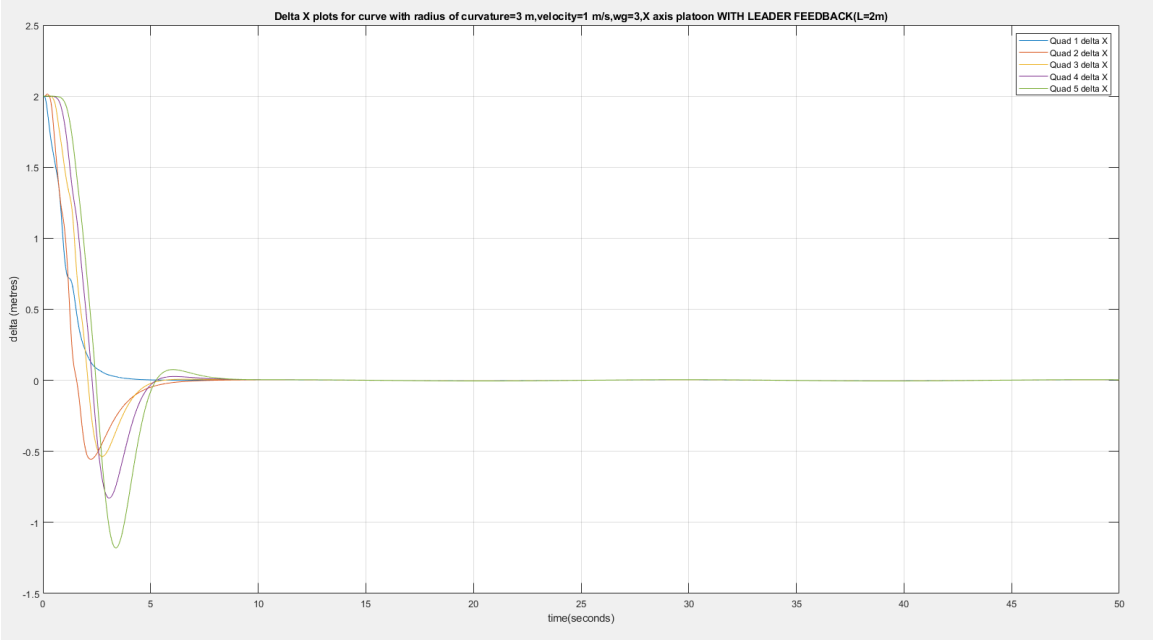

Figure 7.55: Plot of Δ_x (m) vs. time (s) for 6 Quadcopter Platoon, Curve Trajectory with Leader Feedback (Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m)

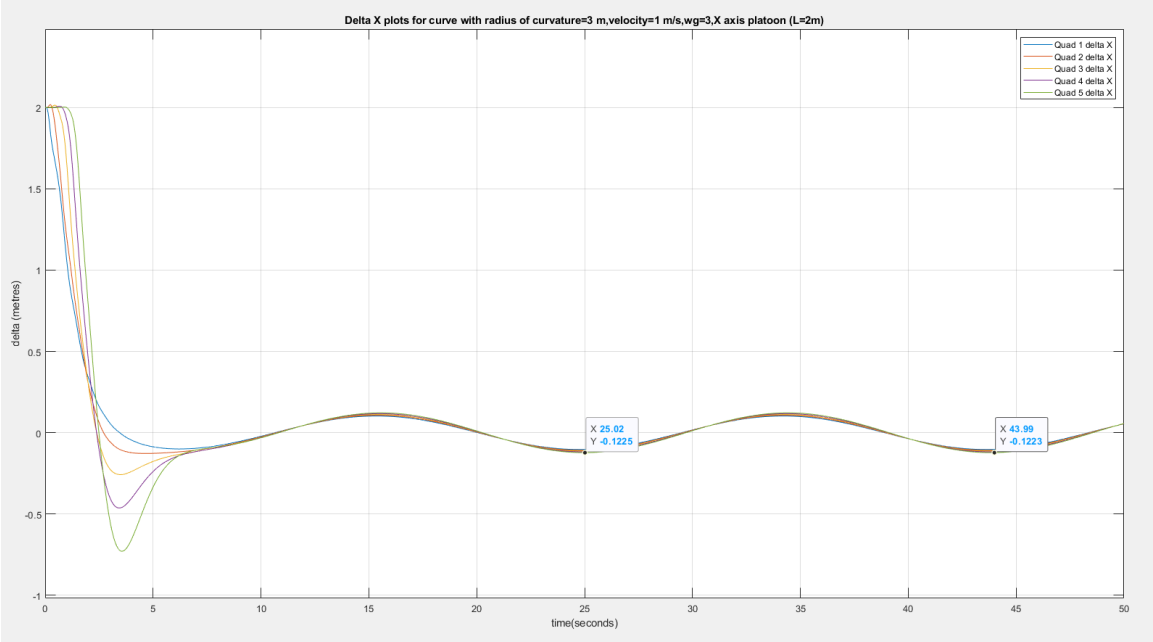

Figure 7.56: Plot of Δ_x (m) vs. time (s) for 6 Quadcopter Platoon, Curve Trajectory with No Leader Feedback (Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m)

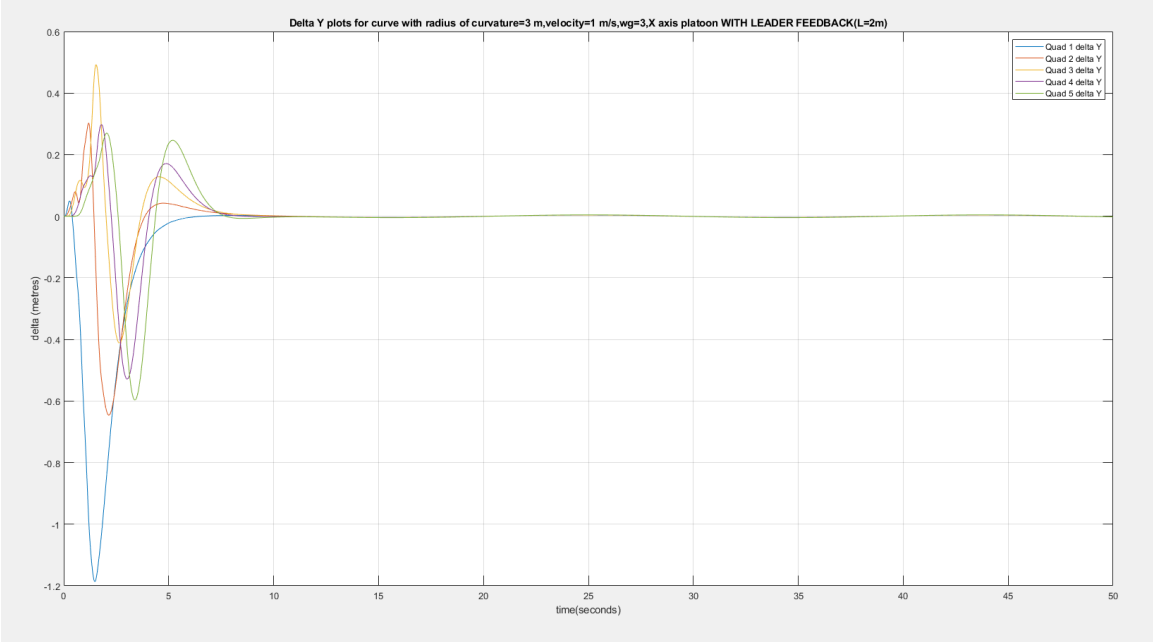

Figure 7.57: Plot of Δ_y (m) vs. time (s) for 6 Quadcopter Platoon, Curve Trajectory with Leader Feedback (Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m)

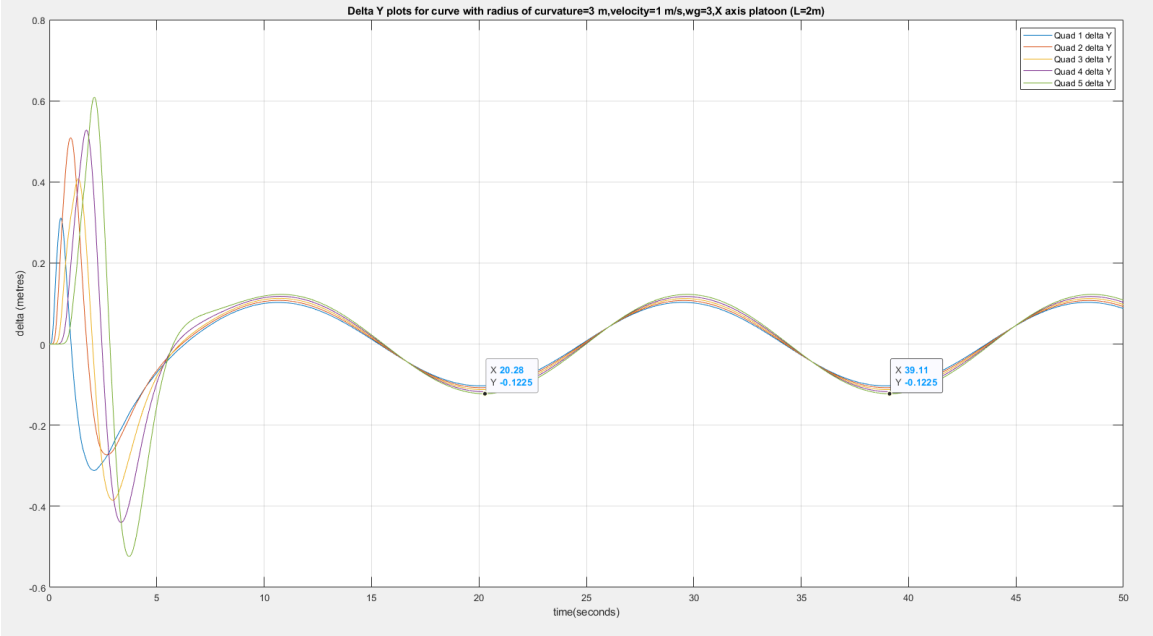

Figure 7.58: Plot of Δ_y (m) vs. time (s) for 6 Quadcopter Platoon, Curve Trajectory with No Leader Feedback (Radius of Curvature = 3 m, Velocity = 1 m/s, L (X-axis) = 2 m)

As observed in the figures presented in pages 105 and 106, provision of lead vehicle acceleration is found to stabilize the accordion effect, with zero steady state error in separation achieved within 7 to 8 seconds of simulation. It is to be noted that undershoot is observed in the first follower separation response within the initial few seconds of simulation, while subsequent followers show less undershoot due to the initial accordion effect. All followers converge to desired spacing with little to no delay.

As analysed for the nominal model, $\frac{v}{R}$ is varied from 0.7 to 2 (in radians per second)in order to find minimum bandwidth required for a maximum steady state Δ error of 5% given available leader acceleration information. The bandwidth values obtained for each $\frac{v}{R}$ for the lead feedback model are documented in Table 7.7, and visualized in Figure 7.59.

Table 7.7: $\frac{v}{R}$ Sweep - Required Bandwidth for 5% Peak Steady State Δ Error, Lead Acceleration Feedback Model

SI No.	$\frac{v}{R}$**(rad/s)**	ω_g **for 5% maximum error (rad/s)**
1.	0.7	0.13
2.	0.8	0.14
3.	1.0	0.2
4.	1.2	0.3
5.	1.4	0.4
6.	1.6	0.48
7.	1.8	0.57
8.	2.0	0.68
9.	2.1	0.72

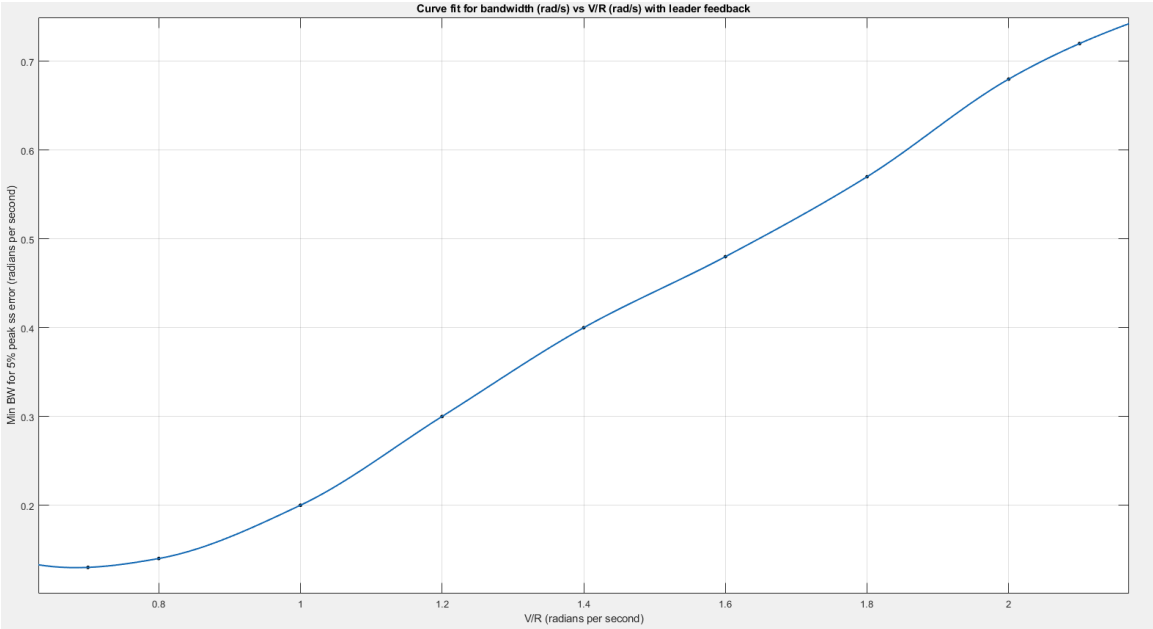

Figure 7.59: $\frac{v}{R}$ vs Bandwidth with Leader Feedback - Visualization

From Figure 7.59 and Table 7.7, it is observed that lead vehicle information significantly reduces the required controller bandwidth in the presented model. As $\frac{v}{R}$ is increased from 1 to 2, the required bandwidth for 5% peak error goes from around 0.2 times the desired $\frac{v}{R}$ (near $\frac{v}{R} = 1$) to around 0.4 times the desired $\frac{v}{R}$ (near $\frac{v}{R} = 2$). For comparison, the curves from Figure 7.53 and 7.59 are plotted together, shown in Figure 7.60.

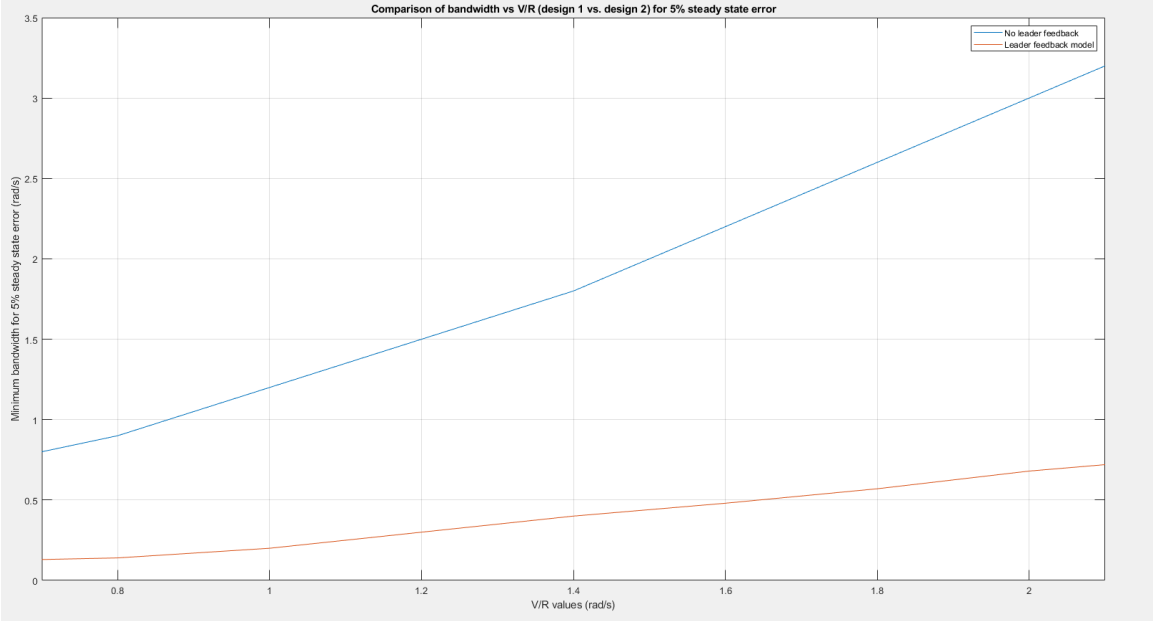

Figure 7.60: $\frac{v}{R}$ vs Bandwidth - Nominal (Ad-hoc) Model vs. Lead Feedback Model

It can be observed that the bandwidth requirements significantly reduce for a desired v/R when lead vehicle acceleration feedback is available in the fleet. With lead feedback, the required bandwidth for a maximum of 5% steady state error is approximately 0.33 times desired v/R, a far smaller factor than in the nominal case (where bandwidth required is 1.3 times desired v/R). In both curves, the relationship between required bandwidth and desired v/R is found to be fairly linear between $v/R = 1$ and $v/R = 2$. relationship observed between v/R = 1.2 and v/R = 2.2

Chapter 8

SUMMARY

8.1 Conclusion

In this thesis, a model for quadcopter rotational and translational dynamics has been designed and analyzed with trade-off studies. Classical controls are used to design low-level control for angular rate and Euler angle command following, using a cascade control scheme, and LQ Servo design is used to model high-level control (path following) for a single quadcopter. Following this analysis, a model for platooning control is derived and analyzed, including a study of the string stability (accordion effect) within each quadcopter. A classical control approach is used to design spacing control for each follower vehicle, and results are simulated on MATLAB using a nonlinear quadcopter model based on the Mark 3 quadcopter design. From the analysis, trade studies are conducted for curved paths (with constant change in lead vehicle acceleration), and the controller is observed to perform well in minimizing steady-state separation errors to 0. As expected, the controller performance is improved by feeding forward lead quadcopter acceleration, as observed for multiple curved paths with varying velocity and radius of curvature. With lead quadcopter acceleration feedback in fleet, the reduce in required separation control bandwidth is observed and quantified (given a desired v/R) for curved path trajectory following.

8.2 Directions for Future Research

Based on the analyses conducted in this research, the following are directions for building on the observed results

- Comparative modeling and trade studies for dynamic formations with variable spacing, requiring increased cooperation and communication between followers

- Analysis of control strategies to stabilize steady state separation along a curve without use of lead feedback information, including trade studies for phase margins greater than 60°

- Study of communication requirements (latency, frequency, e.t.c) to enable lead vehicle information feedback within a fleet, in the context of developing 5G environments.

- Study of optimal formation topologies to minimize the accordion effect and improve scalability in leader-follower quadcopter fleets.

REFERENCES

[1] - Lu, Shi, "Modeling, Control and Design of a Quadrotor Platform for Indoor Environments," Master's Thesis.

[2] - Wallace, Brent, "Modeling, Analysis, Control and Design of Highly Maneuverable Quadcopters", Honors Thesis

[3] -Altawaitan, Abdullah, "Modeling, Design, and Control of Multiple Quadrotors," Master's Thesis

[4] - S. Sheikholeslam and C. A. Desoer, "Longitudinal Control of a Platoon of Vehicles," 1990 American Control Conference, San Diego, CA, USA, 1990, pp. 291-296, doi: 10.23919/ACC.1990.4790743.

[5] - S. Sheikholeslam and C. A. Desoer, "Longitudinal control of a platoon of vehicles with no communication of lead vehicle information: a system level study," in IEEE Transactions on Vehicular Technology, vol. 42, no. 4, pp. 546-554, Nov. 1993, doi: 10.1109/25.260756.

[6] - E. Abbasi, M. Ghayour, M. Danesh, P. Amiri and M. H. Yoosefian, "Formation flight control and path tracking of a multi-quadrotor system in the presence of measurement noise and disturbances," 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM), Tehran, Iran, 2018, pp. 273-279, doi: 10.1109/ICRoM.2018.8657620.

[7] - D. A. Mercado, R. Castro and R. Lozano, "Quadrotors flight formation control using a leader-follower approach," 2013 European Control Conference (ECC), Zurich, 2013, pp. 3858-3863, doi: 10.23919/ECC.2013.6669637.

[8] - A. A. Rodriguez, "Analysis and Design of Feedback Control Systems," Tempe, AZ: CONTROL3D, L.L.C., 2002.

[9] - M. Bangura, "Aerodynamics and control of quadrotors," M.S. Thesis, The Australian National University, Canberra, 2017.

[10] - Baru, Penerbit and Sabikan, Sulaiman and Nawawi, Sophan, "Open-Source Project (OSPs) Platform for Outdoor Quadcopter" ,Journal of Advanced Research Design(2016).

[11] - Matthias Faessler and Flavio Fontana, "Theory and Math Behind RPG Quadrotor Control", University of Zurich, 2018

[12] - Praveen, V. and Pillai, Anju, "Modeling and simulation of quadcopter using PID controller", 2016

[13] - L. M. Argentim, W. C. Rezende, P. E. Santos and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," 2013 International Conference on Informatics, Electronics and Vision (ICIEV), Dhaka, 2013, pp. 1-6, doi: 10.1109/ICIEV.2013.6572698.

[14] - Sreenath, Koushil and Kumar, Vijay, "Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots", June 2013, 10.15607/RSS.2013.IX.011

[15] - Recchiuto, CT, Sgorbissa, "A Post-disaster assessment with unmanned aerial vehicles: A survey on practical implementations and research approaches", J Field Robotics, 2018; 35: 459– 490.

[16] - Milhouse, Mark," Framework for Autonomous Delivery Drones",2015, 1-4. 10.1145/2808062.2808075.

[17] - Berrahal, Sarra and Kim, Jong-Hoon and Rekhis, Slim and Boudriga, N. and Wilkins, Deon and Acevedo, Jaime, "Border surveillance monitoring using Quadcopter UAV-Aided Wireless Sensor Networks", Journal of Communications Software and Systems, 2016 12. 67-82. 10.24138/jcomss.v12i1.92.

[18]- Rafifandi, R., Asri, D.L., Ekawati, E. et al, "Leader–follower formation control of two quadrotor UAVs", SN Appl. Sci. 1, 532, 2019

[19] - H. Huang, C. Yu and Q. Wu, "Distributed LQR design for multi-agent formations," 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, 2010, pp. 4535-4540, doi: 10.1109/CDC.2010.5716988.

[20] - E. Kuantama, I. Tarca and R. Tarca, "Feedback Linearization LQR Control for Quadcopter Position Tracking," 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, 2018, pp. 204-209, doi: 10.1109/CoDIT.2018.8394911.

[21] - J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," in IEEE Transactions on Robotics and Automation, vol. 18, no. 5, pp. 837-846, Oct. 2002, doi: 10.1109/TRA.2002.803458

[22] - Darbha, Swaroop and Huandra, R, " Intelligent Cruise Control System Design based on a Traffic Flow Specification. Vehicle System Dynamics", November 1998, 30. 319-344. 10.1080/00423119808969455.

[23] - D. Gheorghiţă, I. Vîntu, L. Mirea and C. Brăescu, "Quadcopter control system," 2015 19th International Conference on System Theory, Control and Computing (ICSTCC), Cheile Gradistei, 2015, pp. 421-426, doi: 10.1109/ICSTCC.2015.7321330.

[24] - Evdokimov, Ilya, "BLADE ELEMENT THEORY IN THE UAV MULTIROTOR BLADE OPTIMIZATION",2015

[25] - A. Michael Harrington, “Optimal propulsion system design for a micro quadrotor”, Master’s Thesis (University of Maryland), 2011

[26] - S. BOUABDALLAH, “Design and control of quadrotors with application to autonomous flying,” Ph.D. Dissertation, EPFL, 2007

[27] - Paul E. Klopsteg “The Bifilar Pendulum”, Review of Scientific Instruments 1, 3 (1930)

[28] - J. Ferrin, R. Leishman, R. Beard, and T. McLain, “Differential flatness based control of a rotorcraft for aggressive maneuvers,” 2011

[29] - A. E. C. D. Cunha, “Benchmark: Quadrotor attitude control,” in ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ser. EPiC Series in Computing, G. Frehse and M. Althoff, Eds., vol. 34. EasyChair, 2015

[30] - A. A. Rodriguez, Analysis and Design of Multivariable Feedback Control Systems. Tempe, AZ: CONTROL3D, L.L.C., 2002.

[31] - A. A. Rodriguez, Kaustuv Mondal et. al ”Modeling, design and control of low-cost differential-drive robotic ground vehicles: Part II — Multiple vehicle study,” 2017 IEEE Conference on Control Technology and Applications (CCTA), Mauna Lani, HI, 2017, pp. 161-166, doi: 10.1109/CCTA.2017.8062457.

[32] - Kaustav Mondal, A.A Rodriguez et al, ”Comparison of Kinematic and Dynamic Model Based Linear Model Predictive Control of Non-Holonomic Robot for Trajectory Tracking: Critical Trade-offs Addressed”, IASTED International Conference on Mechatronics and Control, 2019.

APPENDIX A

SELECTED MATLAB CODE

```
function y = Traj(t)
%% Following a Straight Line
% h = 1;
% if (t < 10)

%     x_r_acc = 0;
%     y_r_acc = 0;
%     z_r_acc = 0;
%     x_r_vel = 1;
%     y_r_vel = 1;
%     z_r_vel = 0;
%     x_r = x_r_vel*t;
%     y_r = y_r_vel*t;
%     z_r = 1;
%     psi = 0;
%     psi_vel = 0;
%     psi_acc = 0;


%% Curved path
% Parameters

b = 2.1213;
a = b;
h = 1;%default height
w1=0.3333;
w2=w1;



   x_r=a*sin(w1*t);
   y_r =b*cos(w2*t);
   z_r = h;
   x_r_vel =a*w1*cos(w1*t);
   y_r_vel =-b*w2*sin(w2*t);
   z_r_vel = 0;
   x_r_acc = -a*(w1^2)*sin(w1*t);
   y_r_acc = -b*(w2^2)*cos(w2*t);
   z_r_acc = 0;
   psi = 0;
   psi_vel = 0;
   psi_acc = 0;

% % % % %


%% Output
% y (12 x 1)
y = [x_r; y_r; z_r; x_r_vel; y_r_vel; z_r_vel; x_r_acc; y_r_acc;
    z_r_acc; psi; psi_vel];

end
```

2. PD Control Design (Bandwidth and Robustness)

```
%% Roll off term included (for inner loop BW=22 rad/s)

function [gp,zp]=InnerLoopPDControllerDesignFinal(magPx,angPx,PM,wg)
zp=wg/(tand(PM-180-angPx+2*atand(wg/220)));
gp=((wg^2)+(220^2))/((magPx)*(220^2)*sqrt((wg^2) + (zp^2)));
```

3. Altitude Control Design (PD Control)

```
function [gp,zp]=PDControllerDesignAlt(magPx,angPx,PM,wg)

rolloff=100;
zp=wg/(tand(PM-180-angPx+atand(wg/rolloff)));
gp=sqrt((wg^2)+(rolloff^2))/(magPx)*(rolloff)*sqrt((wg^2)+(zp^2));
```

4. Low Level Control Simulation File

```
s=tf('s');
Ixx = 0.0019; % Moment of interia (x-axis) in [kg m^2]
Iyy = 0.00195; % Moment of interia (y-axis) in [kg m^2]
Izz = 0.00369; % Moment of interia (z-axis) in [kg m^2]

Pp=(1/Ixx)*(1/s);
Pq=(1/Iyy)*(1/s);
Pr=(1/Izz)*(1/s);

a=9.79;%Sconstant from actuator testing in Mark 3 design (Shi Lu)
ActDynamics=a/(s+a);


%% Angular rate control(Continuous)
PpActDynamicsCont=ActDynamics*Pp;
PqActDynamicsCont=ActDynamics*Pq;
PrActDynamicsCont=ActDynamics*Pr;
bode(PpActDynamics);
hold on;bode(PqActDynamics);
hold on;bode(PrActDynamics);
grid on;legend('Pp','Pq','Pr');
title('Bode Magnitude plots for angular rate plants')

%% Discretization (for DAC, given digital controller)

Ts=0.0025; %400 Hz Controller

PpActDynamicsDisc=c2d(PpActDynamicsCont,Ts,'zoh');
PqActDynamicsDisc=c2d(PqActDynamicsCont,Ts,'zoh');
PrActDynamicsDisc=c2d(PrActDynamicsCont,Ts,'zoh');

PpActDynamics=d2c(PpActDynamicsDisc,'zoh');
PqActDynamics=d2c(PqActDynamicsDisc,'zoh');
PrActDynamics=d2c(PrActDynamicsDisc,'zoh');
```

```
%% Inner loop angular rate control (bandwidth sweep results)
PM=60;
% wg=7;
% wg=15;
wg=22;
% wg=30;
% wg=35;
[magPp,angPp]=bode(PpActDynamics,wg);
[magPq,angPq]=bode(PqActDynamics,wg);
[magPr,angPr]=bodemag(PrActDynamics,wg/4);


[gpp,zpp]=InnerLoopPDControllerDesign(magPp,angPp,PM,wg);
[gpq,zpq]=InnerLoopPDControllerDesign(magPq,angPq,PM,wg);
[gpr,zpr]=InnerLoopPDControllerDesign(magPr,angPp,PM,wg/4);

Kp=gpp*(s+zpp)*((220/(s+220))^2);
Kq=gpq*(s+zpq)*((220/(s+220))^2);
Kr=gpr*(s+zpr)*((50/(s+50))^2);

Lp=Kp*PpActDynamics;
Lq=Kq*PqActDynamics;

%% Outer loop angle control

%First order roll off
[gppF,zppF]=InnerLoopPDControllerDesignFinal(magPp,angPp,PM,wg);
[gpqF,zpqF]=InnerLoopPDControllerDesignFinal(magPq,angPq,PM,wg);

KpRollOff=gppF*(s+zppF)*((220^2)/(s+220)^2);
KqRollOff=gpqF*(s+zpqF)*((220^2)/(s+220)^2);
% Kr=gpr*(s+zpr);

LpRollOff=KpRollOff*PpActDynamics;
LqRollOff=KqRollOff*PqActDynamics;

Tp=feedback(LpRollOff,1);
Tq=feedback(LqRollOff,1);


%Kphi=5;

%Kphi=6;

%Kphi=8;

Kphi=9;

%Kphi=10;


Ktheta=Kphi;

Lphi=Kphi*(Tp)*(1/s);
Ltheta=Ktheta*(Tq)*(1/s);
```

```
s = tf('s');
g = 9.81;
m = 0.647;
Ap1 = [zeros(3,3) eye(3,3) zeros(3,1); zeros(4,3) zeros(4,3) ...
zeros(4,1)];
Bp1 = [zeros(3,3) zeros(3,1); -g 0 0 0; 0 g 0 0; 0 0 1/m 0;
    0 0 0 1];
Cp1 = [eye(3,3) zeros(3,4); zeros(1,3) zeros(1,3) 1];
Dp1 = zeros(4,4);
% Changing Units from radians to degrees
r2d = 180/pi;
su = diag( [ r2d, r2d, 1, r2d ] );
sx = diag( [ 1, 1, 1, 1, 1, 1, r2d ] );
sy = diag( [ 1, 1 , 1, r2d ] );
Ap1 = sx*Ap1*inv(sx);
Bp1 = sx*Bp1*inv(su);
Cp1 = sy*Cp1*inv(sx);
Dp1 = sy*Dp1*inv(su);
%*****************************************************
% Linearization Around Hover with Linear Drag
% beta = rho*Cd*vx e*Ss*
% vx e = 1 (m/s), Ss = 0.015625 (m^2), rho = 1.225, Cd = 0.47

beta = 0.0090;
Ap2 = [zeros(3,3) eye(3,3) zeros(3,1); zeros(3,3) -beta*eye(3,3) ...
zeros(3,1); zeros(1,7)];
Bp2 = [zeros(3,3) zeros(3,1); -g 0 0 0; 0 g 0 0; 0 0 1/m 0;
    0 0 0 1];
Cp2 = [eye(3,3) zeros(3,4); zeros(1,3) zeros(1,3) 1];
Dp2 = zeros(4,4);
% Changing Units from radians to degrees
r2d = 180/pi;
su = diag( [ r2d, r2d, 1, r2d ] );
sx = diag( [ 1, 1, 1, 1, 1, 1, r2d ] );
sy = diag( [ 1, 1 , 1, r2d ] );
Ap2 = sx*Ap2*inv(sx);
Bp2 = sx*Bp2*inv(su);
Cp2 = sy*Cp2*inv(sx);
Dp2 = sy*Dp2*inv(su);
% -------------------------------------------------
% Plant Dimensions
%
[ns,nc] = size(Bp1); % Number of States, Number of ...
% Controls
[no,~] = size(Cp1);
% First System with no drag
[ns1,nc1] = size(Bp1); % Number of States, Number ...
% of Controls
[no1,~] = size(Cp1); % Number of Outputs
% Second System with drag
[ns2,nc2] = size(Bp2); % Number of States, Number ...
% of Controls
[no2,~] = size(Cp2); % Number of Outputs
% -------------------------------------------------
```

```
% Natural Modes: Poles (Eigenvalues), Eigenvectors
%
% First System with no drag
[evec1,eval1] = eig(Ap1) % evec contains eigenvectors
% eval contains poles or eigenvalues
% Second System with drag
[evec2,eval2] = eig(Ap2)
% ------------------------------------------------
% Transmission Zeros
%
% First System with no drag
plantzeros1 = tzero(ss(Ap1,Bp1,Cp1,Dp1)) % transmission zeros
% System has no finite transmission zeros
% Second System with drag
plantzeros2 = tzero(ss(Ap2,Bp2,Cp2,Dp2)) % transmission zeros
% System has no finite transmission zeros
% 209
% ------------------------------------------------
% SYSTEM TRANSFER FUNCTIONS: From u i to x j
%
% First System with no drag
% Plant zpk1 = zpk(ss(Ap1,Bp1,Cp1,Dp1)) % Zeros, Poles, and Gains
% from u i to x j
% Second System with drag
% Plant zpk2 = zpk(ss(Ap2,Bp2,Cp2,Dp2)) % Zeros, Poles, and Gains
% from u i to x j
% ------------------------------------------------
% Controllability
%
% First System with no drag
% cm1 = [Bp1 Ap1*Bp1 (Ap1^2)*Bp1 (Ap1^3)*Bp1 (Ap1^4)*Bp1
% (Ap1^5)*Bp1  (Ap1^6)*Bp1]; % Controllability Matrix
% rcm1 = rank(cm1) % Rank of Controllability Matrix
% % Second System with drag
% cm2 = [Bp2 Ap2*Bp2 (Ap2^2)*Bp2 (Ap2^3)*Bp2 (Ap2^4)*Bp2
% (Ap2^5)*Bp2  (Ap2^6)*Bp2]; % Controllability Matrix
% rcm2 = rank(cm2) % Rank of Controllability Matrix
% % ------------------------------------------------
% % Observability
% %
% % First System with no drag
% om1 = [Cp1; Cp1*Ap1; Cp1*(Ap1^2); Cp1*(Ap1^3); Cp1*(Ap1^4); ...
% Cp1*(Ap1^5); Cp1*(Ap1^6)]; % Observability Matrix
% rom1 = rank(om1) % Rank of Observability Matrix
% % Second System with drag
% om2 = [Cp2; Cp2*Ap2; Cp2*(Ap2^2); Cp2*(Ap2^3); Cp2*(Ap2^4); ...
% Cp2*(Ap2^5); Cp2*(Ap2^6)]; % Observability Matrix
% rom2 = rank(om2) % Rank of Observability Matrix
% % FREQUENCY RESPONSE: Singular Values
% %
% % u = [ theta (rad) phi (rad) T (N) r (rad/s) ]
% % x = [ x (m/s) y (m/s) z (m/s) vx (m/s) ...
% vy (m/s) vz (m/s) psi (rad) ]
% % y = [ x (m/s) y (m/s) z (m/s) psi (rad) ]
% %
% winit = -4;
% wfin = 1;
```

```
% nwpts = 200;
% w = logspace(winit,wfin,nwpts); % Form vector of ...
% logarithmically spaced freq points
% sv = sigma(ss(Ap1, Bp1, Cp1, Dp1),w);
% sv = 20*log10(sv);
% figure; semilogx(w, sv, 'b')
% %clear sv
% title('Outputs: x, y, z (m), \psi (deg); Inputs: \theta,
%\phi (deg), T (N), r (deg/s)')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(Ap2, Bp2, Cp2, Dp2),w);
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% 210
% pause
% PLANT SVD ANALYSIS at Low Frequencies, w = 0.01 (rad/s)
%
% First System with no drag
% w0 = 0.01;
% P w0 = Cp1*inv(w0*eye(7,7)-Ap1)*Bp1;
% [udc1,sdc1,vdc1] = svd(P w0)
% Second System with drag
% P w0 = Cp2*inv(w0*eye(7,7)-Ap2)*Bp2;
% [udc2,sdc2,vdc2] = svd(P w0)
%% ...
% ***************************************************
%
% First Design for Linear Quadratic Regulator (LQR)
% Augment Plant with Integrators
% For Zero Steady Error to Step Commands
% This follows from the Internal Model Priciple
% State x = [xp xI]
% where
% xp is the state
% xI is the integrator state
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First System with no drag
A1 = [Ap1 zeros(7,4); Cp1 zeros(4,4)];
B1 = [Bp1; zeros(4,4)];
C1 = [Cp1 zeros(4,4)];
D1 = zeros(4,4);
Cr1 = [0 0 0 1 0 0 0;
0 0 0 0 1 0 0;
0 0 0 0 0 1 0];
% Second System with drag
A2 = [Ap2 zeros(7,4); Cp2 zeros(4,4)];
B2 = [Bp2; zeros(4,4)];
C2 = [Cp2 zeros(4,4)];
D2 = zeros(4,4);
Cr2 = [0 0 0 1 0 0 0;
0 0 0 0 1 0 0;
0 0 0 0 0 1 0];
% LQR Design Parameters
rho = 0.2;
```

```
Q = diag([10,10,10,10,10,10,100,100,100,100,1]);
R = rho * eye(4,4);
[G1, K1, clpoles1] = lqr(A1,B1,Q,R);
[G2, K2, clpoles2] = lqr(A2,B2,Q,R);
%*******************************************************
%
% LQ OPEN LOOP FREQUENCY RESPONSE
%
gy1 = [G1(:,1:3) G1(:,7)];
gr1 = G1(:,4:6);
gz1 = G1(:,8:11);
gy2 = [G2(:,1:3) G2(:,7)];
gr2 = G2(:,4:6);
gz2 = G2(:,8:11);
% 211
% First System with no drag
aol1 = [ Ap1-Bp1*gr1*Cr1 Bp1*gz1;
zeros(4,7) zeros(4,4) ];
bol1 = [ Bp1*gy1;
eye(4,4) ];
col1 = [ Cp1 zeros(4,4) ];
dol1 = zeros(4,4);
ols1 = ss(aol1, bol1, col1, dol1);
% Second System with drag
aol2 = [ Ap2-Bp2*gr2*Cr2 Bp2*gz2;
zeros(4,7) zeros(4,4) ];
bol2 = [ Bp2*gy2;
eye(4,4) ];
col2 = [ Cp2 zeros(4,4) ];
dol2 = zeros(4,4);
ols2 = ss(aol2, bol2, col2, dol2);
w = logspace(-3,3,100);
% sv = sigma(ss(A1, B1, G1, 0*ones(4,4)),w);
% sv = 20*log10(sv);
% figure;semilogx(w, sv, 'b')
% %clear sv
% title('Open Loop Singular Values: Plant Input')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(A2, B2, G2, 0*ones(4,4)),w);
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% hold off
% pause
% w = logspace(-3,3,100);
% sv = sigma(ols1,w);
% sv = 20*log10(sv);
% figure;semilogx(w, sv, 'b')
% %clear sv
% title('Open Loop Singular Values: Error Signal')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ols2,w);
```

```
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% pause
%return
%*********************************************************
%
% LQ CLOSED LOOP FREQUENCY RESPONSE
%
% First System with no drag
acl1 = aol1 - bol1*col1;
% 212
bcl1 = bol1;
ccl1 = col1;
dcl1 = dol1;
cls1 = ss(acl1,bcl1,ccl1,dcl1);
% Second System with drag
acl2 = aol2 - bol2*col2;
bcl2 = bol2;
ccl2 = col2;
dcl2 = dol2;
cls2 = ss(acl2,bcl2,ccl2,dcl2);
% Closed Loop Poles
% First System with no drag
[Wn1,zeta1, clpoles1] = damp(cls1)
% Second System with drag
[Wn2,zeta2, clpoles2] = damp(cls2)
sv = sigma(ss(A1-B1*G1, B1, -G1, eye(4,4)-0*ones(4,4)),w);
sv = 20*log10(sv);
% figure(1);semilogx(w, sv, 'b')
% %clear sv
% title('LQ Sensitivity: Plant Input')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(A2-B2*G2, B2, -G2, eye(4,4)-0*ones(4,4)),w);
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% hold off
% pause
% sv = sigma(ss(acl1, bcl1, -ccl1, eye(4,4)-dcl1),w);
% sv = 20*log10(sv);
% figure;semilogx(w, sv, 'b')
% %clear sv
% title('LQ Sensitivity: Error Signal')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(acl2, bcl2, -ccl2, eye(4,4)-dcl2),w);
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% pause
% sv = sigma(ss(acl1, bcl1, ccl1, dcl1),w);
% sv = 20*log10(sv);
% figure;semilogx(w, sv, 'b')
% %clear sv
```

```
% title('LQ Complementary Sensitivity: Plant Output')
% grid
% xlabel('Frequency (rad/sec)')
% ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(acl2, bcl2, ccl2, dcl2),w);
% sv = 20*log10(sv);
% 213
% semilogx(w, sv, 'r')
% hold off
% pause
% %return
sv = sigma(ss(A1-B1*G1, B1, G1, 0*ones(4,4)),w);
sv = 20*log10(sv);
figure;semilogx(w, sv, 'b')
%clear sv
title('LQ Complementary Sensitivity: Plant Input')
grid
xlabel('Frequency (rad/sec)')
ylabel('Singular Values (dB)')
% hold on
% sv = sigma(ss(A2-B2*G2, B2, G2, 0*ones(4,4)),w);
% sv = 20*log10(sv);
% semilogx(w, sv, 'r')
% hold off
% pause
% %*****************************************************
%
% CLOSED LOOP COMMAND FOLLOWING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First System
%
% t = [0:0.02:10];
% [y, t, x] = step(cls1,t);
% %
% % POSITION IN X-AXIS COMMAND
% %
% % x: r = [1 0 0 0] x-axis position Command
% %
% figure(1);
% % subplot(2,1,1)
% % plot(t,y(:,:,1))
% % grid
% % title('Output Response To r = [1 0 0 0] Command')
% % ylabel('Outputs')
% % xlabel('Time (seconds)')
% % legend('x', 'y', 'z', 'psi')
% %
% % Controls: r = [1 0 0 0] x-axis position Command
% %
% u10 = [-G1 gy1]*[x(:,:,1)'
% ones(1, size(x(:,:,1)')*[0 1]')
% 0*ones(1, size(x(:,:,1)')*[0 1]')
% 0*ones(1, size(x(:,:,1)')*[0 1]')
% 0*ones(1, size(x(:,:,1)')*[0 1]')];
% plot(t,u10)
% grid
```

```
% title('Input Response to X step Command')
% ylabel('Controls')
% xlabel('Time (seconds)')
% legend('Pitch', 'Roll', 'Thrust', 'Yaw Rate')
% % 214
% % pause
% %
% % POSITION IN Y-AXIS COMMAND
% %
% % y: r = [0 1 0 0] y-axis position Command
% %
% figure(2);
% % subplot(2,1,1)
% % plot(t,y(:,:,2))
% % grid
% % title('Output Response To r = [0 1 0 0] Command')
% % ylabel('Outputs')
% % xlabel('Time (seconds)')
% % legend('x', 'y', 'z', 'psi')
% %
% % Controls: r = [0 1 0 0] y-axis position Command
% %
% u20 = [-G1 gy1]*[x(:,:,2)'
% 0*ones(1, size(x(:,:,2)')*[0 1]')
% ones(1, size(x(:,:,2)')*[0 1]')
% 0*ones(1, size(x(:,:,2)')*[0 1]')
% 0*ones(1, size(x(:,:,2)')*[0 1]')];
% % subplot(2,1,2)
% plot(t,u20)
% grid
% title('Input Response to Y step Command')
% % ylabel('Pitch, Roll (deg), Thrust (N), Yaw (deg/s)')
% xlabel('Time (seconds)')
% ylabel('Controls')
% legend('Pitch', 'Roll', 'Thrust', 'Yaw Rate')
% % pause
% %
% % POSITION IN Z-AXIS COMMAND
% %
% % z: r = [0 0 1 0] z-axis position Command
% %
% figure(3);
% % subplot(2,1,1)
% % plot(t,y(:,:,3))
% % grid
% % title('Output Response To r = [0 0 1 0] Command')
% % ylabel('Outputs')
% % xlabel('Time (seconds)')
% % legend('x', 'y', 'z', 'psi')
% %
% % Controls: r = [0 0 1 0] z-axis position Command
% %
% u30 = [-G1 gy1]*[x(:,:,3)'
% 0*ones(1, size(x(:,:,3)')*[0 1]')
% 0*ones(1, size(x(:,:,3)')*[0 1]')
% ones(1, size(x(:,:,3)')*[0 1]')
% 0*ones(1, size(x(:,:,3)')*[0 1]')];
```

```
% % subplot(2,1,2)
% plot(t,u30)
% % 215
% grid
% title('Input Response to Z step Command')
% ylabel('Controls')
% xlabel('Time (seconds)')
% legend('Pitch', 'Roll', 'Thrust', 'Yaw Rate')
% % pause
% %
% % YAW COMMAND
% %
% % Yaw: r = [0 0 0 1] yaw Command
% %
% figure(4);
% % subplot(2,1,1)
% % plot(t,y(:,:,4))
% % grid
% % title('Output Response To r = [0 0 0 1] Command')
% % ylabel('Outputs')
% % xlabel('Time (seconds)')
% % legend('x', 'y', 'z', 'psi')
% %
% % Controls: r = [0 0 0 1] Yaw Command
% %
% u40 = [-G1 gy1]*[x(:,:,4)'
% 0*ones(1, size(x(:,:,4)')*[0 1]')
% 0*ones(1, size(x(:,:,4)')*[0 1]')
% 0*ones(1, size(x(:,:,4)')*[0 1]')
% ones(1, size(x(:,:,4)')*[0 1]')];
% % subplot(2,1,2)
% plot(t,u40)
% grid
% title('Input Response to Psi step Command')
% ylabel('Controls')
% xlabel('Time (seconds)')
% legend('Pitch', 'Roll', 'Thrust', 'Yaw Rate')
```

6. Inverse mapping - Trajectory generation file

```
function v = inverseMapping(u)
    m = 0.647;
    up1 = u(1);
    up2 = u(2);
    up3 = u(3);
    up4 = u(4);
    phi = u(5);
    theta = u(6);
    psi = u(7);
    p = u(8);
    q = u(9);
    r = u(10);
    t = u(11);

    R_psi = [cos(psi) sin(psi) 0;
             -sin(psi) cos(psi) 0;
```

```
            0 0 1];
    T_d = m*sqrt(up1^2+up2^2+up3^2);
%     T_d=m*(up3);
    z = R_psi*[up1; up2; up3]*m/(T_d);
    phi_d = asin(z(2));
%     phi_d = up2/9.81;
    theta_d = atan(-z(1)/z(3));
%     theta_d= -up1/9.81;

    r_d = cos(theta)/cos(phi)*up4+q*tan(phi);
%     r_d= up4;


    v = [T_d; phi_d; theta_d; r_d];
%     v = [T_dlin; phi_dlin; theta_dlin; r_dlin];
end
```

7. Platooning Control - Simulation File

```
%% Plant model
% based on Inverse mapping and linearization at hover


Px=(-1/9.81)*TthetaFinal*(-9.81)*(1/s)*(1/s);
Py=(-1/9.81)*TphiFinal*(-9.81)*(1/s)*(1/s);


%Bandwidth sweep design
wg1=1;
wg2=2;
wg3=3;
wg4=4;
wg5=5;
wg6=6;
wg10=10;


[magPx0,angPx0]=bode(Px,wg1);
[magPx1,angPx1]=bode(Px,wg2);
[magPx2,angPx2]=bode(Px,wg3);
[magPx3,angPx3]=bode(Px,wg4);
[magPx4,angPx4]=bode(Px,wg5);
[magPx5,angPx5]=bode(Px,wg6);
[magPx6,angPx6]=bode(Px,wg10);



[magPy0,angPy0]=bode(Py,wg1);
[magPy1,angPy1]=bode(Py,wg2);
[magPy2,angPy2]=bode(Py,wg3);
[magPy3,angPy3]=bode(Py,wg4);
[magPy4,angPy4]=bode(Py,wg5);
[magPy5,angPy5]=bode(Py,wg6);
[magPy6,angPy6]=bode(Px,wg10);
```

```
PM=60; %sufficient for closed loop properties, AAR book

% PD controller design

[gpd0,zpd0]=PDControllerDesign(magPx0,angPx0,PM,wg1);
[gpd1,zpd1]=PDControllerDesign(magPx1,angPx1,PM,wg2);
[gpd2,zpd2]=PDControllerDesign(magPx2,angPx2,PM,wg3);
[gpd3,zpd3]=PDControllerDesign(magPx3,angPx3,PM,wg4);
[gpd4,zpd4]=PDControllerDesign(magPx4,angPx4,PM,wg5);
[gpd5,zpd5]=PDControllerDesign(magPx5,angPx5,PM,wg6);
[gpd6,zpd6]=PDControllerDesign(magPx6,angPx6,PM,wg10);

% PID controller design

% [gpd0,zpd0]=PIDControllerDesign(magPx0,angPx0,PM,wg1);
% [gpd1,zpd1]=PIDControllerDesign(magPx1,angPx1,PM,wg2);
% [gpd2,zpd2]=PIDControllerDesign(magPx2,angPx2,PM,wg3);
% [gpd3,zpd3]=PIDControllerDesign(magPx3,angPx3,PM,wg4);
% [gpd4,zpd4]=PIDControllerDesign(magPx4,angPx4,PM,wg5);
% [gpd5,zpd5]=PIDControllerDesign(magPx5,angPx5,PM,wg6);
% [gpd6,zpd6]=PIDControllerDesign(magPx6,angPx6,PM,wg10);

% PD controller design
[gpdy0,zpdy0]=PDControllerDesign(magPy0,angPy0,PM,wg1);
[gpdy1,zpdy1]=PDControllerDesign(magPy1,angPy1,PM,wg2);
[gpdy2,zpdy2]=PDControllerDesign(magPy2,angPy2,PM,wg3);
[gpdy3,zpdy3]=PDControllerDesign(magPy3,angPy3,PM,wg4);
[gpdy4,zpdy4]=PDControllerDesign(magPy4,angPy4,PM,wg5);
[gpdy5,zpdy5]=PDControllerDesign(magPy5,angPy5,PM,wg6);
[gpdy6,zpdy6]=PDControllerDesign(magPy6,angPy6,PM,wg10);

% PID controller design

% [gpdy0,zpdy0]=PIDControllerDesign(magPy0,angPy0,PM,wg1);
% [gpdy1,zpdy1]=PIDControllerDesign(magPy1,angPy1,PM,wg2);
% [gpdy2,zpdy2]=PIDControllerDesign(magPy2,angPy2,PM,wg3);
% [gpdy3,zpdy3]=PIDControllerDesign(magPy3,angPy3,PM,wg4);
% [gpdy4,zpdy4]=PIDControllerDesign(magPy4,angPy4,PM,wg5);
% [gpdy5,zpdy5]=PIDControllerDesign(magPy5,angPy5,PM,wg6);
% [gpdy6,zpdy6]=PIDControllerDesign(magPy6,angPy6,PM,wg10);


% PD Controller Design
Kpd0=gpd0*(s+zpd0)*(500/(s+500));
Kpd1=gpd1*(s+zpd1)*(500/(s+500));
Kpd2=gpd2*(s+zpd2)*(500/(s+500));
Kpd3=gpd3*(s+zpd3)*(500/(s+500));
Kpd4=gpd4*(s+zpd4)*(500/(s+500));
Kpd5=gpd5*(s+zpd5)*(500/(s+500));
Kpd6=gpd6*(s+zpd6)*(500/(s+500));

%PID Controller Design
% Kpd0=(gpd0*(s+zpd0)/s);
% Kpd1=(gpd1*(s+zpd1)/s);
% Kpd2=(gpd2*(s+zpd2)/s);
% Kpd3=(gpd3*(s+zpd3)/s);
% Kpd4=(gpd4*(s+zpd4)/s);
```

```
% Kpd5=(gpd5*(s+zpd5)/s);
% Kpd6=(gpd6*(s+zpd6)/s);


% PD Controller
Kpdy0=gpdy0*(s+zpdy0)*(1000/(s+1000));
Kpdy1=gpdy1*(s+zpdy1)*(1000/(s+1000));
Kpdy2=gpdy2*(s+zpdy2)*(1000/(s+1000));
Kpdy3=gpdy3*(s+zpdy3)*(1000/(s+1000));
Kpdy4=gpdy4*(s+zpdy4)*(1000/(s+1000));
Kpdy5=gpdy5*(s+zpdy5)*(1000/(s+1000));
Kpdy6=gpdy6*(s+zpdy6)*(1000/(s+1000));

% % PID Controller
% % Kpdy0=(gpdy0*(s+zpdy0)/s);
% % Kpdy1=(gpdy1*(s+zpdy1)/s);
% % Kpdy2=(gpdy2*(s+zpdy2)/s);
% % Kpdy3=(gpdy3*(s+zpdy3)/s);
% % Kpdy4=(gpdy4*(s+zpdy4)/s);
% % Kpdy5=(gpdy5*(s+zpdy5)/s);
% % Kpdy6=(gpdy6*(s+zpdy6)/s);

L0PD=Kpd0*Px;
L1PD=Kpd1*Px;
L2PD=Kpd2*Px;
L3PD=Kpd3*Px;
L4PD=Kpd4*Px;
L5PD=Kpd5*Px;
L6PD=Kpd6*Px;
%
%
L0PDy=Kpdy0*Py;
L1PDy=Kpdy1*Py;
L2PDy=Kpdy2*Py;
L3PDy=Kpdy3*Py;
L4PDy=Kpdy4*Py;
L5PDy=Kpdy5*Py;
L6PDy=Kpdy6*Py;
%
% % L1PID=Kpid1*Px;
% % L2PID=Kpid2*Px;
% % L3PID=Kpid3*Px;
% % L4PID=Kpid4*Px;
% % L5PID=Kpid5*Px;
%
T0PD=feedback(L0PD,1);
T1PD=feedback(L1PD,1);
T2PD=feedback(L2PD,1);
T3PD=feedback(L3PD,1);
T4PD=feedback(L4PD,1);
T5PD=feedback(L5PD,1);
%
T0PDy=feedback(L0PDy,1);
T1PDy=feedback(L1PDy,1);
T2PDy=feedback(L2PDy,1);
T3PDy=feedback(L3PDy,1);
T4PDy=feedback(L4PDy,1);
```

```
T5PDy=feedback(L5PDy,1);

S0PD=1-T0PD;
S1PD=1-T1PD;
S2PD=1-T2PD;
S3PD=1-T3PD;
S4PD=1-T4PD;
S5PD=1-T5PD;

S0PDy=1-T0PDy;
S1PDy=1-T1PDy;
S2PDy=1-T2PDy;
S3PDy=1-T3PDy;
S4PDy=1-T4PDy;
S5PDy=1-T5PDy;
%% Closed loop (T) and Sensitivity (S) Plots
figure(1);
bodemag(T0PD);
hold on;bodemag(T1PD);
hold on;bodemag(T2PD);
hold on;bodemag(T3PD);
hold on;bodemag(T4PD);
hold on;bodemag(T5PD);
grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
title('T magnitude plots,X axis separation control');


figure(2);
bodemag(T0PDy);
hold on;bodemag(T1PDy);
hold on;bodemag(T2PDy);
hold on;bodemag(T3PDy);
hold on;bodemag(T4PDy);
hold on;bodemag(T5PDy);
grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
title('T magnitude plots,X axis separation control');


figure(3);
bodemag(S0PD);
hold on;bodemag(S1PD);
hold on;bodemag(S2PD);
hold on;bodemag(S3PD);
hold on;bodemag(S4PD);
hold on;bodemag(S5PD);
grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
title('S magnitude plots,X axis separation control');

figure(4);
bodemag(S0PDy);
hold on;bodemag(S1PDy);
hold on;bodemag(S2PDy);
hold on;bodemag(S3PDy);
hold on;bodemag(S4PDy);
hold on;bodemag(S5PDy);
grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
title('S magnitude plots,Y axis separation control');
```

```
% figure(5);
% step(T0PD);
% hold on;step(T1PD);
% hold on;step(T2PD);
% hold on;step(T3PD);
% hold on;step(T4PD);
% hold on;step(T5PD);
% grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
% title('Step response plots for X axis separation control')
%
% figure(6);
% step(T0PDy);
% hold on;step(T1PDy);
% hold on;step(T2PDy);
% hold on;step(T3PDy);
% hold on;step(T4PDy);
% hold on;step(T5PDy);
% grid on;legend('wg=1','wg=2','wg=3','wg=4','wg=5','wg=6');
% title('Step response plots for Y axis separation control')
```

8. Curve trajectory generation- ROC and Velocity sweep

```
%Curve trajectory generation

b = 2.1213;
a = b;
w1=0.3333;
w2=w1;
velocity=sqrt(((a*w1)^2)+((b*w2)^2));
accel=sqrt(((a*(w1^2))^2)+((b*(w2^2))^2));
ROC=(velocity^2)/accel;


%%Test function, to compute a and b for desired V/R
ROCd=3;
% VoverRradpersec=1.8;

% velocityd=(VoverRradpersec*ROCd)/(2*pi);

velocityd=1;
a1d=ROCd/sqrt(2);
w1d=velocityd/ROCd;
```