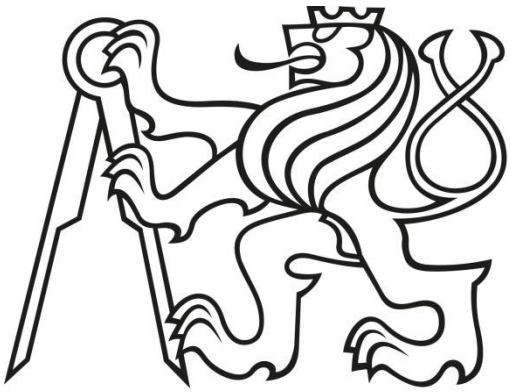
Czech Technical University

Space Masters Program

Faculty of Electrical Engineering

Department of Control Engineering



QUADCOPTER FLIGHT MECHANICS MODEL AND CONTROL ALGORITHMS

Eswarmurthi Gopalakrishnan

Prague, May 2016

Supervisor: **Prof. Dr. Martin Xhromcik**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Gopalakrishnan Eswar Murthi**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Diploma Thesis: **Quadcoptor flight mechanics model and control algorithms**

Guidelines:

The objective of the thesis is to develop a quadcopter flight mechanics nonlinear model in MATLAB/Simulink and - based on this - to design, implement in MATLAB/Simulink, and validate a set of basic and advanced control laws for its stabilization and guidance.

1. Deliver a literaure survey related specifically to the thesis tpic.
2. Implement the quadcopter flight mechanics nonlinear model in MATLAB/Simulink.
3. Perform linear analysis.
4. Design, implement and validate a set of basic and advanced control laws for stabilization and guidance.
5. Design, implement and validate a set of basic and advanced control laws for automatic guidance.

Bibliography/Sources:

Literature: Bryson Jr., Control Systems for Aircraft and Spacecraft, Wiley, 2002

Diploma Thesis Supervisor: doc.Ing. Martin Hromčík, Ph.D.

Valid until the summer semester 2015/2016

L.S.

Prof. Ing. Michael Šebek, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 4, 2015

Abstract:

In this thesis I am working on designing the controller for the non-linear Quadcopter model. The controller is finely tuned using gradient descent method in order to find the best parameters for the controller which guarantees the robustness of the model.

In the beginning of my work, the fundamental equations of motion and forces of the Quadcopter are derived and the design parameters for the given Quadcopter are chosen. I have created a non-linear model for the Quadcopter based on the equation of motion and forces of moment. Generally, the non-linear model will be linearized at a given point. In this thesis work, the model is made to hover at an altitude where the non-linear model is linearized. Based on the inputs, the response of the linear and non-linear model are analysed, it is made sure that both the models behaves in the same manner, if not then the non-linear model and the linearization condition is checked. Once the output for both the model is matched, the research work is further preceded. The controller for the non-linear model is designed and the results are analysed. Finally, the controller is finely tuned using 'Gradient Descent Method' for best controller parameters. I define the 'optimal' set of parameters as vector which minimizes the cost function. In my case, I would like to define a cost function that penalizes high error over an extended duration of time. The best parameter with the effective results for the given Quadcopter is fixed. For the given simulation time, the performance of the Quadcopter model with the optimal controller values are studied by analysing the 3-dimensional visualisation, the Angular velocity and Angular displacement of the model.

The controller is designed in such a way that even if there are any disturbances in the future, the model behaves well for the given set of inputs and the design parameters.

Proclamation:

I hereby declare that I have developed and written the enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

In Prague, May 27, 2016

……

Eswarmurthi Gopalakrishnan

Acknowledgment

Firstly, I would like to express my sincere thanks to my supervisor in Czech Technical University Dr. Prof. Martin Hromcik, who gave me the opportunity to work on this interesting project, for his guidance over the period of time.

I would like to dedicate this Master Thesis work to my late Mother Thriveni. Also, I would like to thank my Parents, Thamarai Selvi and my friends who stood by my side during hard time and gave me mental strength to complete my thesis successfully. Next I would like to thank Mr. Sanjeev Kubakaddi for his guidance throughout the thesis work. Last but not least, I would like to thank Mr. Sandesh GS, Mr. Sauradep Roy for their help in clearing my doubts whenever I approached them.

Contents

Chapter 1 Introduction

The aim of this work is to design a linearized simulation model for a Quadcopter and design a PID controller for the Quadcopter. The work progresses as follows, in the beginning a detailed introduction is given about the UAVs, their dynamics and applications. In Chapter 2, the simulation model for the Quadcopter is designed using the Quad-rotor dynamics. In Chapter 3, the non-linear model is linearized using Jacobian Matrix method assigning operating points for the Quadcopter. Then the controllability, observability of the linearized model is determined. Later, the Root locus analysis and the bode plot analysis of the linearized model is determined. In Chapter 4, a detailed description about the Simulink, the methods to follow to design the simulink model is given and finally a brief note on stability is given. In chapter 5, a brief note on the PID controller is given. The effect of each parameters of the PID controller is defined using a table. In chapter 6, the simulation of the model for the PD and PID controller is done and the performance of the system for the controller is studied. Then, the best parameter for PID gain values is determined using the gradient descent method and the plot for the cost to the iteration is plotted in this section. Finally in the Chapter 7, a detailed conclusion of the report, the controller chosen for the model is explained.

When developing the simulation model, first a detailed theoretical description of the problem is studied followed by a paper calculation for the model, later followed by simulation analysis for designing a PID controller.

1.1 Quadcopter

Quadcopter also known as Quad rotor Helicopter, Quad rotor is a multi-rotor helicopter that is lifted and propelled by four rotors. Quadcopter are classified as rotorcraft, as opposed to fixed-wing aircraft, because their lift is generated by a set of rotors (vertically oriented propellers).

Unlike most helicopters, Quadcopter uses two set of identical fixed pitched propellers: tow clock wise and two counters- clockwise. These use variation of RPM to control loft and torque. Control of the Quadcopter is achieved by altering the rotation rate of one or more rotor discs, thereby changing it torque load and thrust/lift characteristics.

A number of manned designs appeared in the 1920s and 1920s. These vehicles were among the first successful heavier- than- air vertical take-off and landing (VTOL) vehicles [1]. However, early prototypes suffered from poor performance [1], and latter prototypes required too much pilot work load, due to poor stability augmentation and limited control authority.

More recently Quadcopter designs have become popular in unmanned aerial vehicle (UAV) research. These vehicles use an electronic control system and electronic sensors to stabilize the aircraft. With their small size and agile manoeuvrability, these Quadcopter can be flown indoors as well as outdoors.

A typical Quadcopter is equipped with an inertial navigation unit (3 accelerometers, 3 gyroscopes and 3 magnetometers) for attitude determination, a barometer (outdoor) or an ultrasonic proximity sensor (indoor) for altitude measurements and optionally they come with a camera or GPS receiver.

1.1.1 Indoor Quadcopter

Indoor Quadcopter cannot use GPS for absolute positioning and magnetometers provide noisy measurements due to disturbed local magnetic field. However they take benefit from absence of wind gusts, from relatively stable light conditions and their mission duration is usually shorter than the outdoor Quadcopter. There are already many companies producing Indoor Quadcopter, for example, Ascending Technologies GmBH [2]. The best example for an Indoor Quadcopter is UDI U839, produced by UDI RC. The UDI U839 is shown below.

Figure 1: UDI U839- An Indoor Quadcopter

1.1.2 Outdoor Quadcopter

Outdoor Quadcopter are generally more durable, they take payload and can fly on longer missions than the Indoor Quadcopter. Absolute positioning is provided by GPS receiver. The best example for outdoor Quadcopter is Phantom 2 Vision+ [3]. The Phantom 2 Vision+ is shown in the figure below.

Figure 2: Phantom 2 Vision+ - An Outdoor Quadcopter

1.2 Advantages of Quadcopter over comparably scaled Helicopters

There are several advantages to Quadcopter over comparable- scaled helicopters. First, Quadcopter do not require mechanical linkages to vary the rotor blade pitch angle as they spin. This simplifies the design and maintenance of the vehicle [4]. Secondly, the use of four rotors allows each individual rotor to have a smaller diameter than the equivalent helicopter rotor, allowing them to possess less kinetic energy during flight. This reduces the damage caused should the rotors hit anything. For small-scale UAVs this makes the vehicle safer for close interaction. Some small-scale Quadcopter have frames that enclose the rotors, permitting flights through more challenging environments, with lower risk of damaging the vehicle or its surroundings [5].

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

1.3 Uses of Quadcopter

Research Platform: Quadcopter are a useful tool for university researchers to test and evaluate new ideas in a number of different fields, including flight control theory, navigation, real time systems, and robotics. In recent year many universities have shown Quadcopter performing increasingly complex aerial manoeuvers.

Military Law Enforcement: Quadcopter unmanned aerial vehicles are used for surveillance and reconnaissance by military and law enforcement agencies, as well as search and rescue missions in urban environments. One such example is the Aeryon Scout, created by Canadian company Aeryon Labs [6], which is a small UAV that can quietly hover in place and use a camera to observe people and objects on the ground.

Commercial Use: The largest use of Quadcopter has been in the field of aerial imagery. Quadcopter UAVs are suitable for this job because of their autonomous nature and huge cost

savings [7]. In December 2013, the Deutsche Post gathered international media attention with the project “Parcelcopter”, in which the company tested the shipment of medical products by drone- delivery. As Quadcopter are becoming less expensive media outlets and newspapers are using drones to capture photography of celebrities [8].

Investigating Purpose: Since Quadcopter is small in size and light in weight, they can get into places, where people cannot get into like caves, holes, tunnels, etc. In 2014, in Tamil Nadu, India Quadcopter was used to investigate the granite scam. Investigators used Quadcopter installed with camera and sensors to get in the tunnel and find the granite in it.

Chapter 2 Modelling of a Quadcopter dynamics

This section presents the basic Quadcopter dynamics, as well as control concept. It is based on [9] and [10]. The basic idea of the movement of the Quadcopter is shown in the following figure. It can be seen from the figure that the Quadcopter is simple in mechanical design compared to helicopters. Movement in horizontal frame is achieved by tilting the platform whereas vertical movement is achieved by changing the total thrust of the motors. But, Quadcopter arise certain difficulties with the control design.

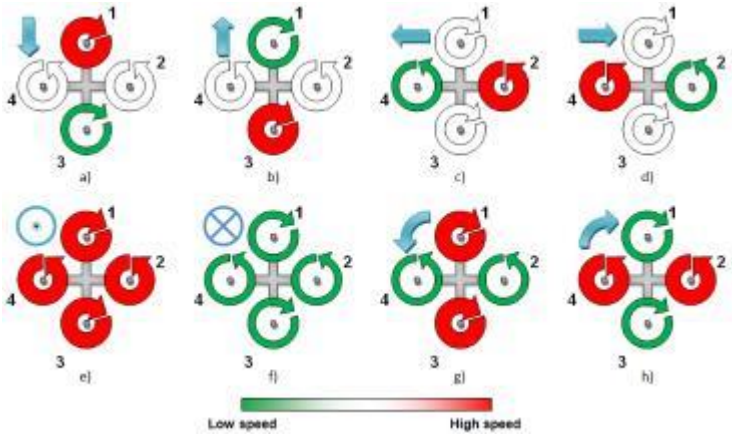


Figure 3: Basic Flight movements of a Quadcopter

A coordinate frame of the Quadcopter is shown in the figure below.

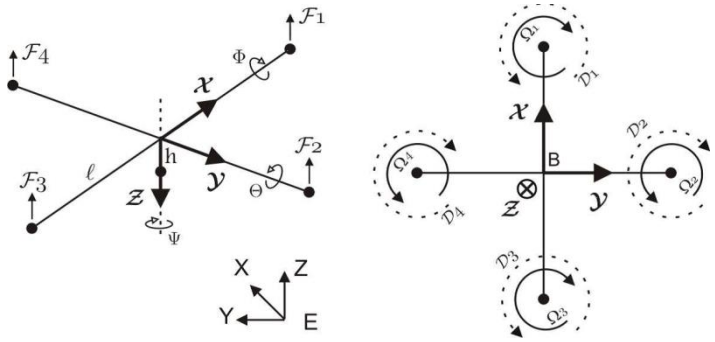


Figure 4: Quadcopter coordinate system

The Quadcopter is designed on the following assumptions [10]:

- The structure is supposed to be rigid
- The Centre of Gravity and the body fixed frame origin are assumed to coincide
- Thrust and drag are proportional to the square of the propeller's speed
- The propellers are supposed to be rigid

- The structure is supposed to be axis symmetrical
- Rotation matrix defined to transform the coordinates from Body to Earth co-ordinates using Euler angles ϕ – roll angle, θ- pitch angle, ψ- yaw angle
- About x by ϕ, y by θ and z by ψ

Special attention should be given in the difference between the body rate measured p, q, r in Body Fixed Frame and the Tait- Bryan angle rates expressed in Earth Fixed Frame. The transformation matrix from $[\phi \quad \dot{\theta} \quad \psi]^T$ to $[p \quad q \quad r]^T$ is given by [9],

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

Moreover, the rotation matrix of the Quadcopter's body must also be compensated during position control. The compensation is achieved using the transpose of the rotation matrix.

$$R(\phi,\theta,\phi) = R(x,\phi)R(y,\theta)R(z,\psi)$$

$$R(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$R(y,\theta) = \begin{bmatrix} \cos\theta & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(z,\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.1 General Moments and Forces

The forces acting upon a Quadcopter are provided below. J_r is a rotor inertia, T is thrust force, H is the hub force (sum of horizontal forces acting on blade elements), Q is a drag moment of a rotor (due to aerodynamic forces), R_m is a rolling moment of a rotor.

Rolling moments:

- Body gyro effect $\dot{\theta}\psi(I_{yy} - I_{zz})$
- Rolling moment due to forward flight $(-1)^{i+1}\sum_{i=1}^{4} R_{mxi}$
- Propeller gyro effect $J_r\dot{\theta}\Omega_r$
- Hub moment due to sideward flight $h\sum_{i-1}^{4} H_{yi}$
- Roll actuators action $l(-T_2 + T_4)$

Pitching moments:

- Body gyro effect $\dot{\phi}\dot{\psi}(I_{zz} - I_{xx})$
- Hub moment due to forward flight $h\sum_{i=1}^{4} H_{xi}$
- Propeller gyro effect $J_r\dot{\phi}\Omega_r$
- Rolling moment due to sideward flight $(-1)^{i+1}\sum_{i=1}^{4} R_{myi}$
- Pitch actuators action $l(T_1 - T_3)$

Yawing moments:

- Body gyro effect $\dot{\theta}\dot{\phi}(I_{zz} - I_{yy})$
- Hub force unbalance in forward flight $l(H_{x2} - H_{x4})$
- Inertial counter- torque $J_r\dot{\Omega}_r$
- Hub force unbalance in sideward flight $l(-H_{y1} + H_{y3})$
- Counter torque unbalance $(-1)^{i}\sum_{i=1}^{4} Q_i$

Forces along z Axis:

- Actuators action $\cos\psi\cos\phi\sum_{i=1}^{4} T_i$
- Weight mg

Forces along x Axis:

- Actuators action $(\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i$
- Hub force in x axis $-\sum_{i=1}^{4} H_{xi}$
- Friction $\frac{1}{2}C_x A_c \rho\dot{y}|\dot{y}|$

Forces along y Axis:

- Actuators action $(-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i$
- Hub force in y axis $-\sum_{i=1}^{4} H_{yi}$
- Friction $\frac{1}{2}C_y A_c \rho\dot{y}|\dot{y}|$

Where l stands for a distance between the propeller axis and COG, h is a vertical distance from centre of propeller to COG, Ω_r is an overall residual propeller angular speed and I is moment of inertia. Note that the DC motor dynamics is described by a first order transfer function.

2.2 Equations of Motion

The equations of motion are derived as follows, using moments and forces described in section 2.1. Note that g is a gravitational acceleration and m represents a mass of the rigid body.

$$I_{xx}\ddot{\phi} = \dot{\theta}\psi\left(I_{yy} - I_{zz}\right) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h\sum_{i=1}^{4} H_{yi} + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi}$$

$$I_{yy}\ddot{\theta} = \phi\psi(I_{zz} - I_{xx}) - J_r\phi\Omega_r + l(T_1 - T_3) + h\sum_{i=1}^{4} H_{xi} + (-1)^{i+1}\sum_{i=1}^{4} R_{myi}$$

$$I_{zz}\ddot{\psi} = \dot{\theta}\phi\left(I_{xx} - I_{yy}\right) + J_r\Omega_r + l(H_{x2} - H_{x4}) + l\left(-H_{y1} + H_{y3}\right) + (-1)^{i}\sum_{i=1}^{4} Q_i$$

$$m\ddot{z} = mg - (\cos\psi\cos\phi)\sum_{i=1}^{4} T_i$$

$$m\ddot{x} = (\sin\phi\sin\psi + \cos\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi}$$

$$m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi}$$

The main aerodynamic forces and moments acting on the Quadcopter, during a hovering flight segment, corresponds to the thrust (T), the hub force (H) and the drag moment (Q) because of vertical, horizontal and aerodynamic forces, respectively, followed by the rolling moment (R) related to the integration, over the entire rotor, of the lift of each section, acting at a given radius. An extended formulation of these forces and moments can be found in [11, 12]. The nonlinear dynamics of the system is described by the following equations.

$$\dot{X} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \\ \hline z \\ \dot{z} \\ x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}\frac{I_{yy}-I_{zz}}{I_{xx}} + \dot{\theta}\frac{J_r}{I_{xx}}\Omega_r + \frac{l_a}{I_{xx}}U_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}\frac{I_{zz}-I_{xx}}{I_{yy}} - \dot{\phi}\frac{J_r}{I_{yy}}\Omega_r + \frac{l_a}{I_{yy}}U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}\frac{I_{xx}-I_{yy}}{I_{zz}} + \frac{1}{I_{zz}}U_4 \\ \hline \dot{z} \\ g - (\cos\phi\cos\theta)\frac{U_1}{m_s} \\ \dot{x} \\ u_x\frac{U_1}{m_s} \\ \dot{y} \\ u_y\frac{U_1}{m_s} \end{bmatrix} + \begin{bmatrix} 0 \\ \widetilde{W}_1 \\ 0 \\ \widetilde{W}_2 \\ 0 \\ \widetilde{W}_3 \\ \hline 0 \\ \widetilde{W}_4 \\ 0 \\ \widetilde{W}_5 \\ 0 \\ \widetilde{W}_6 \end{bmatrix}$$

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ \Omega_r \end{bmatrix} = \begin{bmatrix} b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ b(-\Omega_2^2 + \Omega_4^2) \\ b(\Omega_1^2 - \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{bmatrix}$$

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \end{bmatrix}$$

Where,

- I_{xx}, I_{yy}, I_{zz} Moment of Inertia of the Quadcopter about E_x, E_y, E_z axis
- l_a Quadcopter arm length
- b, d thrust, drag coefficient
- J_r moment of inertia of the rotor about its axis of rotation
- m_s total mass of the Quadcopter
- g acceleration of gravity ($9.81\ m/s^2$)

U is the input vector consisting of U_1(total thrust), and U_2, U_3, U_4 which are related to the rotation of the Quadcopter, and Ω_r representing the overall residual propeller angular speed while $\Omega_1, \ldots, \Omega_4$ corresponds to the propeller's angular speeds, X is the state vector that consists of the following,

1) The translational components $\xi = [x,\ y,\ z]^T$ and their derivatives
2) The rotational components $\dot{\eta} = [\phi,\ \dot{\theta},\ \psi]^T$ and their derivatives

The effects of the external disturbances are accounted by the additive disturbance vector $\widehat{W}$.

The non-linearized model is defined in the Matlab simulink. The non-linear model is the set of equations represented in the section [2.2]. Those equations are represented as a model using Matlab simulink. The Drag moment is neglected since the effect of these forces inside a room is small compared to the thrust produced by the Quadcopter.

The overall residual angular speed is taken as $\Omega_r = 475\ rad/s$. This angular speed is the speed produced by the rotor used in Parrot drone [19], specified under the topic "Technical Specifications". Since a step input is given for the linearized model and the analysis is done between the non-linearized and linearized model, a step input is considered for the trust in non-linear model.

The design parameters of the model are given below.

DESIGN PARAMETER	VALUE
m	$0.5\ kg$
l	$0.25\ m$
$I_{xx} = I_{yy}$	$0.0196\ kg\ m^2$
I_{zz}	$0.0264\ kg\ m^2$
$T_{1,2,3,4}$	$5.39\ N$
b	$3.0 * 10^{-5}\ N\ s^2$
d	$1.1 * 10^{-6}\ Nms^2$
$R_{1,2,3,4}$	$sqrt\left(\frac{T_{1,2,3,4}}{b}\right)$
Ω_r	$sqrt\ (\ R_1^2 - R_2^2 + R_3^2 - R_4^2)$
ϕ_l, θ_l, ψ_l	$0\ {}^{rad}/_{s}$

Table 1: The design parameters for the Quadcopter model

The design parameters are substituted in the given matrices and the corresponding non-linearized model is obtained for the given system.

Chapter 3 Linearization of the model

In mathematics, linearization refers to finding the linear approximation to a function at a given point. In the study of dynamical systems, linearization is a method for assessing the local stability of an equilibrium point of a system of nonlinear differential equations or discrete dynamical systems [13]. A set of nonlinear ODE's (forces equations, moments equations) are parameterized by mass characteristics of the system as well as aerodynamics. The nonlinear equations are directly useful for simulations like computer games, flight trainers, flight simulators and validation of control law. They are not directly useable for development of control laws as the design methods rely on the linear systems and control theory. The nonlinear equations come from the geometric transformations and describing functions of aerodynamic coefficient. For this reason, the linearized system and control tools are attractive and viable options for FCS design. In order to design a controller for a Quadcopter it is suggested to have a linearized model of the system to have a precise controller [14]. In this chapter the nonlinear model of the Quadcopter is linearized using the Jacobian method.

3.1 Uses of linearization in Stability analysis

Linearization makes it possible to use tools for studying linear systems to analyse the behaviour of a non-linear function near a given point. The linearization function is the first order term of its Taylor expansion around the point of interest. For a system defined by the equation

$$\frac{dx}{dt} = F(x, t)$$

The Linearized systems can be written as

$$\frac{dx}{dt} \approx F\ (\ x_o, t) + DF\ (x_o, t)\ .(x - \ x_o)$$

Where x_o is the point of interest and $DF\ (x_o)$ is the Jacobian of $F(x)$ evaluated at x_o

Stability Analysis

In stability analysis of autonomous systems, one can use the eigenvalues of the Jacobian matrix evaluated at a hyperbolic equilibrium point to determine the nature of that equilibrium. This is the content of Linearization theorem. For time-varying systems, the linearization requires additional justification [15].

3.2 Jacobian Method

The Jacobian generalizes the gradient of a scalar- valued function of multiple variables, which itself generalizes the derivative of a scalar- valued function of a single variable. In other words, the Jacobian for a scalar- valued function of single variable is simply its derivative [16]. An example of how to solve a function using Jacobian method is given below,

Consider a system with '2' equations and '2' variables as given below,

$$F_1(x,y) = x^2 y$$

$$F_2(x,y) = 5x + \sin y$$

$$J_F(x,y) = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{bmatrix}$$

$$J_F(x,y) = \begin{bmatrix} 2xy & x^2 \\ 5 & \cos y \end{bmatrix}$$

A similar method is used to linearize the nonlinear model of the system. Firstly, the operating points for the system to be linearized are given and the Jacobian method is implemented to linearize the equation.

3.3 Linearization of the model

In order to derive the linearized representation of the Quadcopter's linearized attitude dynamics, small attitude perturbations δ_λ, with $\lambda \in \mathbb{Z}^+$, around the operating points $\left[0,\ \phi^{0,\lambda}, 0, \dot{\theta}^{0,\lambda}, 0, \psi^{0,\lambda}\right]^T$ are considered.

$$\dot{x}_\eta = A_\eta^\lambda x_\eta + B_\eta^\lambda u_\eta + \widetilde{W}_\eta$$

$$x_\eta = \left[\phi, \delta\phi^\lambda, \theta, \delta\dot{\theta}^\lambda, \psi, \delta\psi^\lambda\right]^T$$

$$u_\eta = [\delta U_1, \delta U_2, \delta U_3, \delta U_4, \delta\Omega_r]^T$$

$$\widetilde{W}_\eta = [0, \delta W_1, 0, \delta W_2, 0, \delta W_3]^T$$

The resulting linearized dynamics is an extension of the state space matrices represented in the following equations.

$$A_\eta^\lambda = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{I_{yy}-I_{zz}}{I_{xx}}\dot{\psi}^{0,\lambda} & 0 & \frac{I_{yy}-I_{zz}}{I_{xx}}\dot{\theta}^{0,\lambda} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{I_{zz}-I_{xx}}{I_{yy}}\dot{\psi}^{0,\lambda} & 0 & 0 & 0 & \frac{I_{zz}-I_{xx}}{I_{yy}}\dot{\phi}^{0,\lambda} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\theta}^{0,\lambda} & 0 & \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\phi}^{0,\lambda} & 0 & 0 \end{bmatrix}$$

$$B_\eta^\lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{l_a}{I_{xx}} & 0 & 0 & \frac{J_r}{I_{xx}}\dot{\theta}^{0,\lambda} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{l_a}{I_{yy}} & 0 & -\frac{J_r}{I_{xx}}\dot{\phi}^{0,\lambda} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} & 0 \end{bmatrix}$$

$$\mathrm{C}_\eta^\lambda = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{D}_\eta^\lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The above mentioned $A_\eta^\lambda, B_\eta^\lambda, C_\eta^\lambda, D_\eta^\lambda$ matrices are the linearized matrices for the given system. Where $\phi^{0,\lambda}, \dot{\theta}^{0,\lambda}, \psi^{0,\lambda}$the pitch, roll and yaw are rate at the linearization point.

The design parameters given in the previous section are substituted in the matrices above and obtained matrices are the linearized state space model for the system. The linearized model can be used to design the controller.

3.4 Response of Linearized and Non-linearized system

In this section, we shall see how both the non-linearized and linearized system react to the given input. For our consideration, the input given was a step input and the corresponding response is observed.

The input for the non-linearized as well as the linearized model is the propeller thrusts. The output depends on the model we observe, for the non-linearized model; the outputs are as follows- Roll angle, Roll rate, Pitch angle, Pitch rate, Yaw angle, Yaw rate, Velocity along $'z'$ direction, Velocity along 'x' direction, Velocity along 'y' direction.

Whereas the output from the linearized model; depends on the parameter we require for designing the controller. In my case, the output from the linearized model is as follows- Roll angle, Roll rate, Pitch angle, Pitch rate, Yaw angle, Yaw rate. The response of the non-linearized model and the linearized model for the given step input is given below.

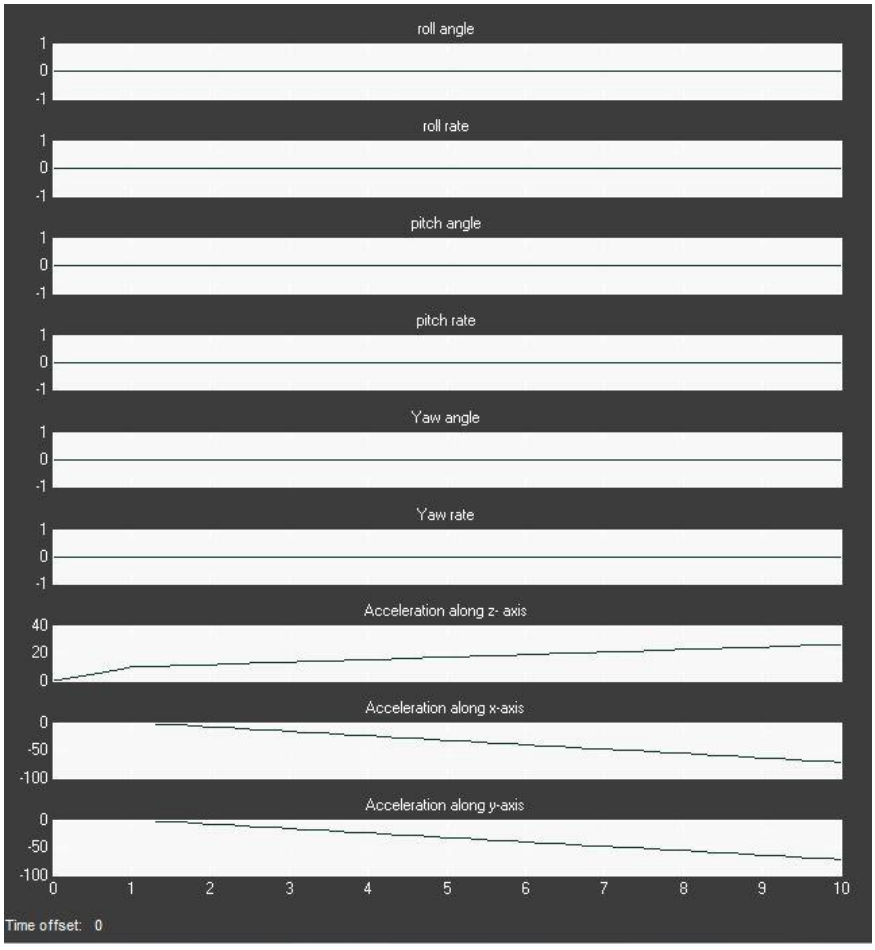


Figure 5: Response of the non-linearized model for the given input

The response of the non-linear Quadcopter is as given above. The reason for this kind of response is in my model equation of motion, in the angular displacement equations the thrust applied in the equation is cancelled out with each other. Hence the angular displacement is zero throughout. Whereas the acceleration of the model with respect to

particular direction is given by summation of all the forces (thrusts) acting on the model. Acceleration along the z- axis depends on the gravitational factors; hence the deviation is decreasing throughout the time. Whereas for the acceleration along the x and y direction depends only on the thrust acting along the direction and the hub force. Hence the acceleration of the model along the x and y direction increases by time. To summarize the result, the angular acceleration of the non-linear model is zero throughout the time and the acceleration along the directions varies throughout time.

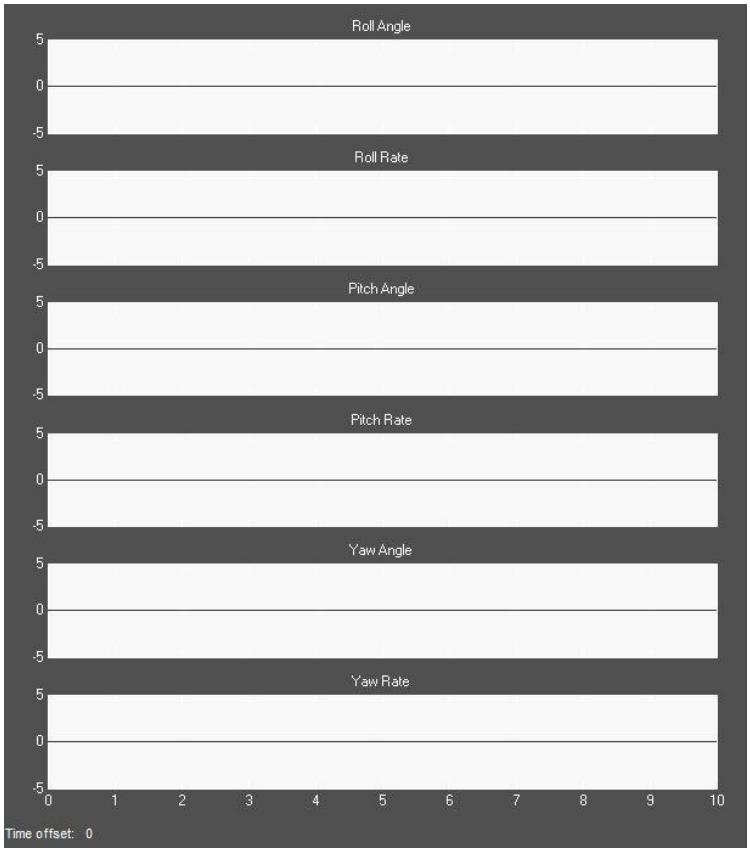


Figure 6: Response of the linearized model for the given step input

The response of the linearization model the reason for this response is because my linearization point for my model is that my model is hovering at a particular point. Since the model is hovering at a particular point, there is no deviation in any of its angle or angle rate. The thrust is the only factor is acting on the model. An equal thrust is applied by every 4 motors. According to the figure “Basic Flight movements of a Quadcopter” and the flight mechanics of the Quadcopter, for hovering, equal thrust produced by every motor. Hence there is no deviation in angle or angle rate of the Quadcopter.

For the case of simplicity, these situations are considered. There are many situations where the linearization of the model is considered. For example, the model is rotating in one particular direction or on every direction. In case of one particular direction, the rate of change of the angle for the particular direction is not zero and it implies that there is a change in the angle with respect to time throughout the time of flight.

For analysing the response of my linear and non-linear model, I have changed the thrusts acting on the model and the change in angular rate of the model. For example, I have changed the thrust acting on the rotor 2 and the rotor 4 and the change in phi angular rate. I have applied the thrust acting in rotor 2 as 1.51875 and for rotor 4 as 2.1875 and the change in phi angular rate as $1.6 * 10^{-5}$. The response of the non-linear and linear model is given below.

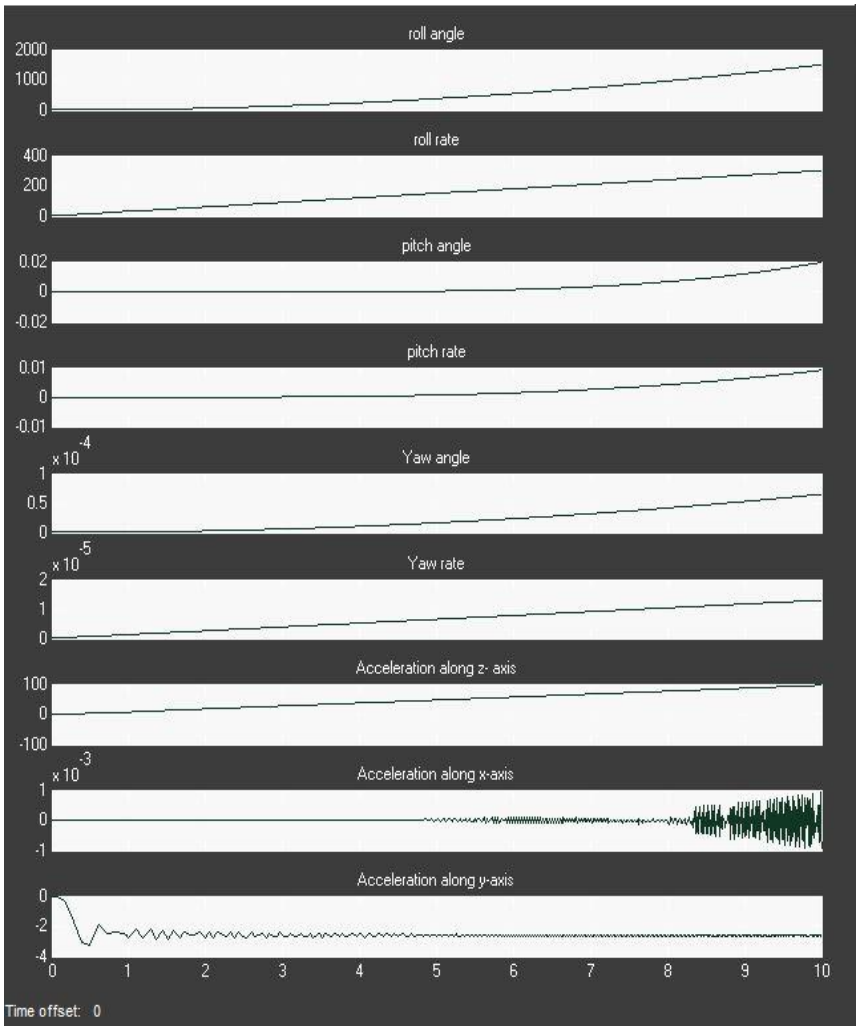


Figure 7: Response of the non-linear model for the change in given input

This is the response of the non-linear model for the change in input to my model. As we can see for the change in input, there is a rise in the angle and the angular rate for all the theta.

The response of the linear model is given below.

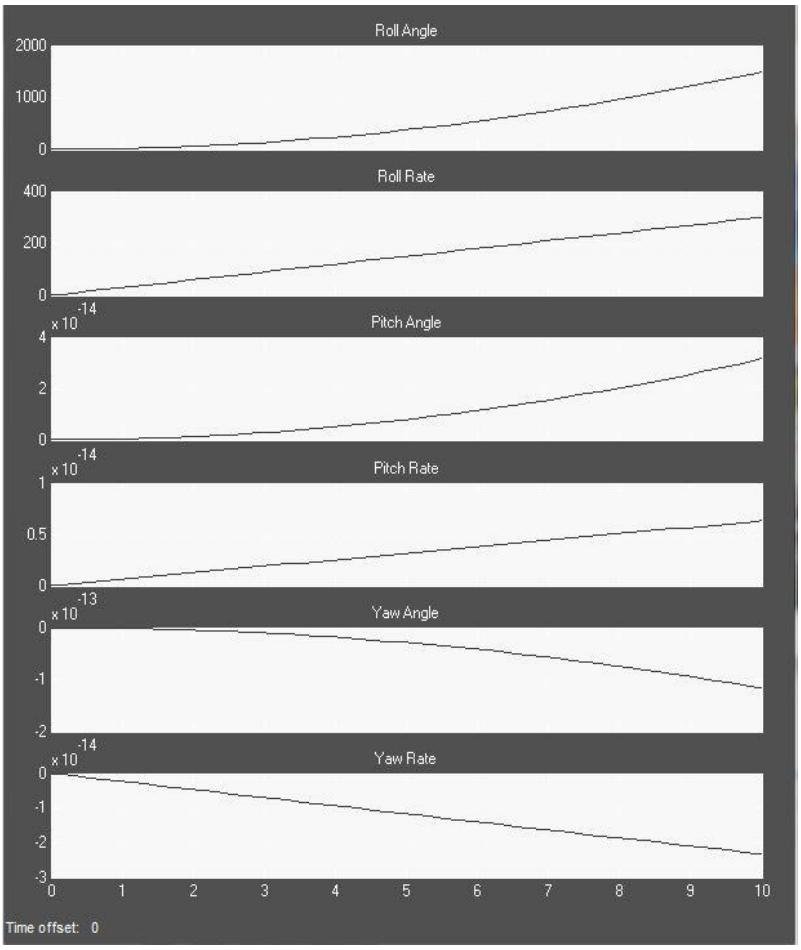


Figure 8: The response of linear model for the change in given input

3.5 Analysis of the Linearized model

In this section, we analyse the linearized model of the quad copter for the given input. The model is analysed based on the input given to the model. The response of the model is as follows- Roll angle, Roll rate, Pitch angle, Pitch rate, Yaw angle and Yaw rate. The model is analysed as, the Roll angel response for the different inputs given to the model, later the roll rate for the given inputs and so on. 5 different inputs are considered for the model, they are- Thrust from rotor 1, rotor 2, rotor 3 and rotor 4 lastly the overall angular rotor speed. The overall angular speed is nothing but the speed at which the rotor rotates. The overall residual angular speed is directly proportional to the thrust produced by the rotor. The response of the linearized model is studied based on the input given to the system. Hence, there are a total of 6 analysis studies, one for each degree of freedom of the system. A system is considered unstable if the poles are placed on the right half plane. The response of the model for the input is given below in the following figure.

3.5.1 Analysis using Controllability and the Observability:

The controllability and observability represents 2 major concepts of modern control system theory. It was introduced by R. Kalman in 1960 [17]. They can be roughly defined as follows:

Controllability; In order to be able to do whatever we want with the given dynamic system under control input, the system must be controllable.

Observability; In order to see what is going on inside the system under observation, the system must be observable.

Controllability of a Linear Time Invariant system:

Before determining the controllability of a LTI system, first let us understanding the Reachability of a system. A particular state X_1 is called reachable if there exists an input that transfers the state of the system from the initial state X_0 to X_1 in some finite time interval [t_0, t]. A system is reachable at time t_1, if every state X_1 in the state- space is reachable at time t_1. Similarly, a system is controllable at time t_0 if every state X_0 in the state- space is controllable at time t_0.

For the LTI system, a system is reachable if and only if its controllability matrix ζ has a full rank of p, where p is the dimension of matrix A and pxq is the dimension of B matrix. The controllability matrix is given by,

$$\zeta = [\, B \; AB \; A^2B \dots \dots \dots \; A^{p-1}B] \; \epsilon \; R^{p \; x \; pq}$$

A system is controllable or "Controllable to the origin" when any state X_1 can be driven to the zero state X=0 in a finite number of steps. A system is controllable when the rank of the system matrix A is p and the rank of the controllability matrix is equal to:

$$Rank \; (\zeta) = Rank \; (A^{-1} \; \zeta) = p$$

If the second equation is not satisfied, then the system is not controllable. If $Rank(A) < p$, then the controllability does not imply Reachability [18].

- Reachability always implies controllability
- Controllability only implies reachability when the transition matrix is non-singular.

Matlab allows one to create a controllability matrix with the $\boldsymbol{ctrb}$ command.

$$R = ctrb(A, B)$$

Then in order to find if the system is controllable or not we use the $\boldsymbol{rank}$ command to determine if it has full rank.

$$rank(R)$$

Observability of Linear Time Invariant system:

The observability of the system is dependant only on the system state and system output. The state space equation of a system is given by,

$$x'(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Therefore, we can show that observability of the system is dependant only on the co-efficient matrices A & C. We can show precisely how to determine whether the system is observable, using only these two matrices. The observability matrix Q is given by,

$$Q = [\ C\ CA\ CA^2 \dots\dots\dots CA^{p-1}]^T$$

We can show that the system is observable if and only if the Q matrix has a rank of p. Notice that the Q matrix has the dimension $pr\ x\ p$.

Matlab allows one to create the observability matrix with the ***obsv*** command as,

$$Q = obsv(A, C)$$

Then in order to determine if the system is observable or not, we can use the rank command to determine if it has full rank.

$$rank(Q)$$

Controllability and Observability of my model:

My model was checked for the controllability and observability using the same technique as above. The controllable and observable matrix has full rank that is 6. Hence the system is considered to be controllable and Observable.

3.5.2 Analysis using poles and zeros:

Poles and Zeros of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes zero respectively. The values of the poles and the zeros of a system determine whether the system is stable and how well the system performs. The locations of the poles and the values of the real and imaginary parts of the pole determine the response of the system. Real parts correspond to exponentials, and imaginary parts correspond to sinusoidal values. Addition of poles to the transfer function has the effect of pulling the root locus to the right, making the system less stable. Addition of zeros to the transfer function has the effect of pulling the root locus to the left, making the system more stable.

The transfer function provides a basis for determining important system response characteristics without solving the complete differential equation. As define, the transfer function is a rational function in the complex variable $s = \sigma + j\,\omega$, [19] that is

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$$

It is often convenient to the factor the polynomials in the numerator and denominator, and to write the transfer function in terms of those factors:

$$H(s) = \frac{N(s)}{D(s)} = K\,\frac{(s - z_1)(s - z_2)\ldots(s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2)\ldots(s - p_{n-1})(s - p_n)}$$

Where the numerator and denominator polynomials, N(s) and D(s), have real coefficients defined by the system's differential equation and $K = {b_m}/{a_n}$. As written in the previous equation, the $z_i's$ are the roots of the equation $N(s) = 0$, and are defined to be the system zeros, and the $p_i's$ are the roots of the equation $D(s) = 0$, and are defined to be the system poles.

A system is characterized by its pole and zeros in the sense that they allow reconstruction of the input or output differential equation. In general, the poles and zeros of a transfer function may be complex, and the system dynamics may be represented graphically by plotting their locations on the complex s-plane, whose axes represent the real and imaginary parts of the complex variable *s.* Such plots are known as pole-zero plots. It is usual to mark the zero location by a circle(o) and a pole location a cross(x). The location of the poles and zeros provide qualitative insights into the response characteristics of a system.

Since my model is a six degree of freedom model, there are 6 poles present in the root locus analysis plot.

The response of the model (as given in the above paragraph) for the thrust from rotor 2 to the Roll Angle and Roll Rate is shown in the following figure,

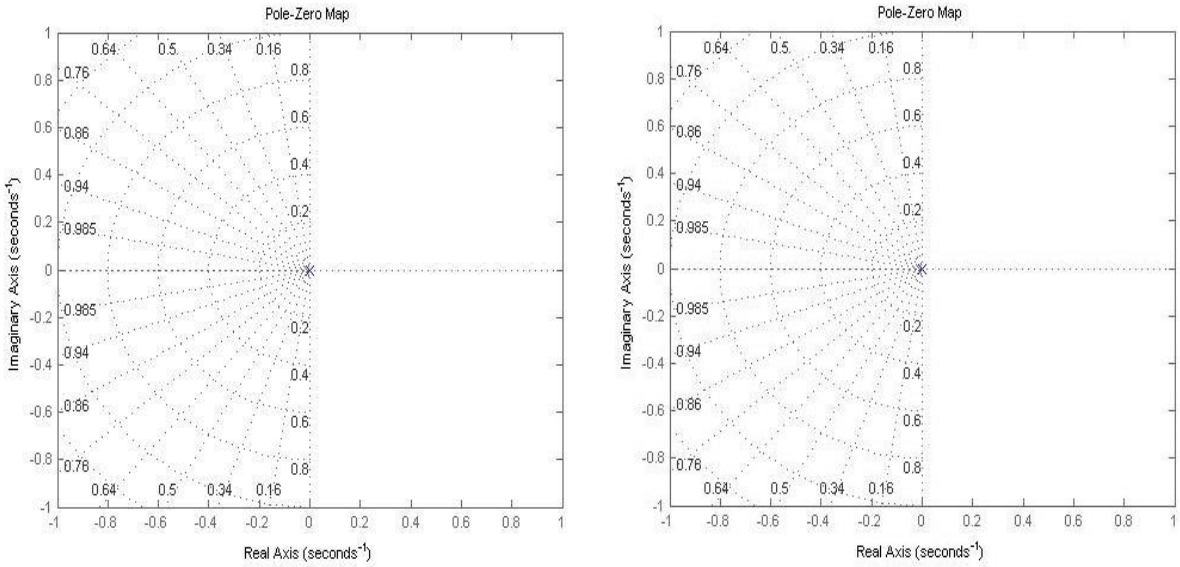


Figure 9: Pole zero position of Roll Angle and Roll Rate due to the effect of rotor 2

The response of the model for third input given to the system i.e. the thrust from the rotor 3 to the Pitch Angle and Pitch Rate is given in the below figure,

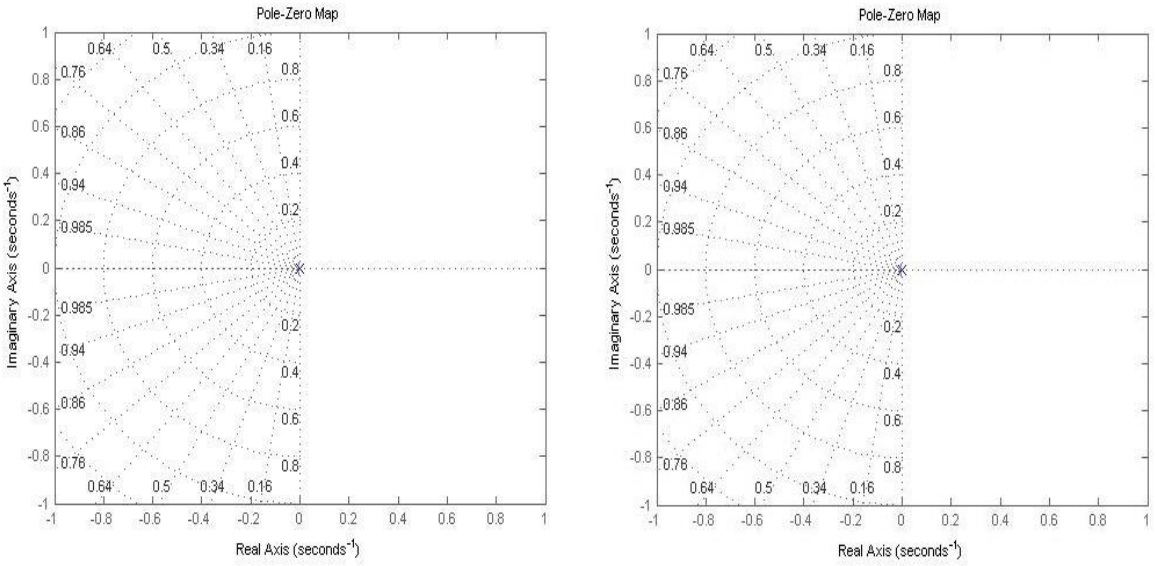


Figure 10: Pole- Zero position of Pitch Angle and Pitch Rate for the effect due to the rotor 3

The response for the thrust from the fourth rotor to the Yaw Angle and Yaw Rate is given below,

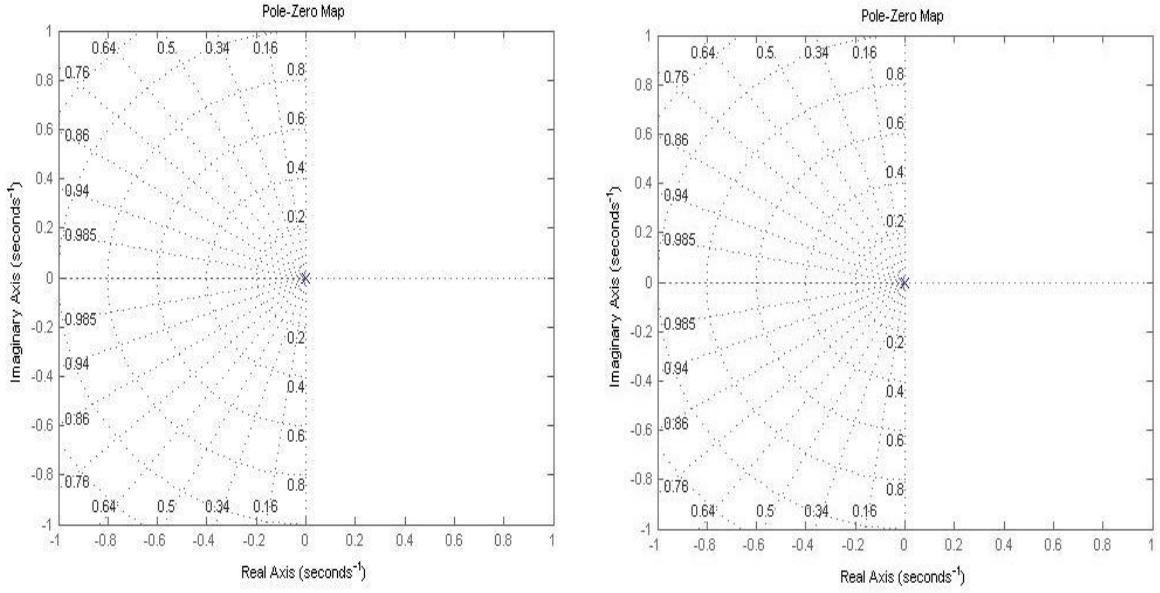


Figure 11: Pole-Zero position of the Yaw Angle and Yaw Rate for the effect of rotor 4

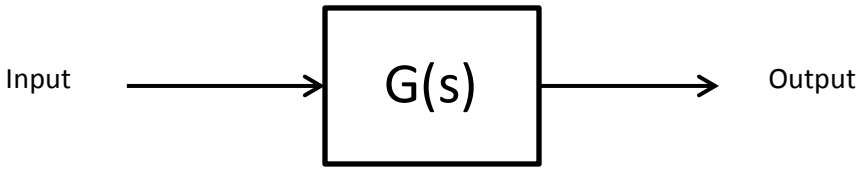
As we can see from the root locus plot, that there are 6 poles present since the model is a six degree of freedom model. Since all the poles are placed exactly at the origin, the model is considered to be marginally stable [20]. For this kind of a system, we need a much better controller to make sure that the model is stable throughout.

A Marginally Stable system is one that, if an impulse of finite magnitude as input is given, neither it will not "blow up" and give an unbounded output nor will the output returns to zero. If a continuous system is given an input at a frequency equal to the frequency of a pole with zero real part, the system's output will increase indefinitely. This explains why for a system to be BIBO stable, the real parts of the poles have to be strictly negative (and not just non positive). A system with a pole at the origin is also marginally stable but in this case there will be no oscillations in the response as the imaginary part is also zero.

3.5.3 Analysis using Bode Plots:

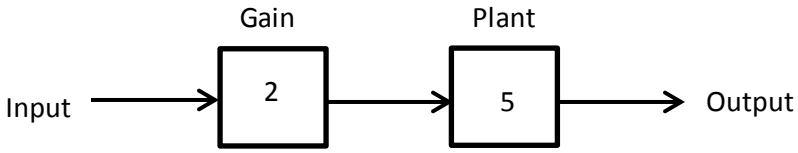
A Bode Plot is a useful tool that shows the gain and phase response of a given LTI system for different frequencies. Bode Plots are generally used with Fourier Transform of a given system. The frequency of the bode plots are plotted against a logarithmic frequency axis. Every tick mark on the frequency axis represents a power of 10 times the previous values (i.e.) the values of the markers go from 0.1, 1, 10, 100, 1000, … Each tick mark is referred to as a decade**.** The bode Magnitude plot measures the system Input / Output ratio in special units called decibels**.** The bode phase plot measures the phase shift in degrees.

Bode gain plots, or bode magnitude plots display the ratio of the system gain at each input frequency. Bode phase plots are plots of the phase shift to an input waveform dependent on the frequency characteristics of the system input [21].



The system G(s) is a Linear Time Variant system and let the input be a sine wave. When the sine wave is fed through G(s), the magnitude of the system can change and phase of the wave can change. Gain of the system is directly proportional to the magnitude of the output and the magnitude of the input at steady state. The output of the single input frequency is a combination of gain and phase shift. Gain and phase changes with frequency. Gain is a scalar term.

For example consider the following system,



Here the normal gain is 5, which is the plant and the scaling term is 2, which is the gain. Hence the total gain of the system is $2 * 5 = 10$.

Margin; Margin is an extra amount of something we can use if needed. Gain and Phase margin is the extra that protects us from instability. This means that margin value is directly proportional to the stability of the system. The Bode magnitude plot measures the system input to output ratio in decibels and phase plot measures the phase shift in degrees.

Procedure to find the gain and phase margin:

First of all to find out the gain margin and the phase margin, we need both the gain and phase cross over frequency. Then we need to find out the zero in magnitude scale, and draw a straight line to magnitude plot, the frequency where the zero meets is the gain cross over frequency. Find out the negative 180 degrees in phase scale and the corresponding frequency is phase cross over frequency.

Extend the phase line where the negative 180 degrees meet to the magnitude plot, so that it meets the gain plot somewhere, mark the point of intersection. Now we have the zero in gain scale, the distance between these two points in the magnitude scale is the gain margin.

Similarly, we have the gain line where zero meets the magnitude plot, extend the line to the phase plot, the distance between this point to the negative 180 degree phase line in phase scale is the phase margin.

In Matlab; to get the gain and phase margin, type in the command window, $\boldsymbol{margin}\,(\boldsymbol{model}\,(\boldsymbol{output}, \boldsymbol{input}))$ will give the bode plot. But notice there will be the Phase margin and gain margin. Also they are marked in dark colour on the bode plot.

Stability conditions of Bode Plots:

- **Stable System:** For a system to be stable, both the margins should be positive. Or the phase margin should be greater than the gain margin.
- **Marginally Stable System:** A system is considered to be marginally stable, if both the margins are zero or phase margin equal to the gain margin.
- **Unstable system:** For an unstable system, either the gain or the phase margin should be negative, or phase margin should be less than the gain margin.

Now for my model, the Bode plot for the 6 degree of freedom is given below.

The bode plot for the response of thrust provided to rotor 2 to the Roll Angle is,

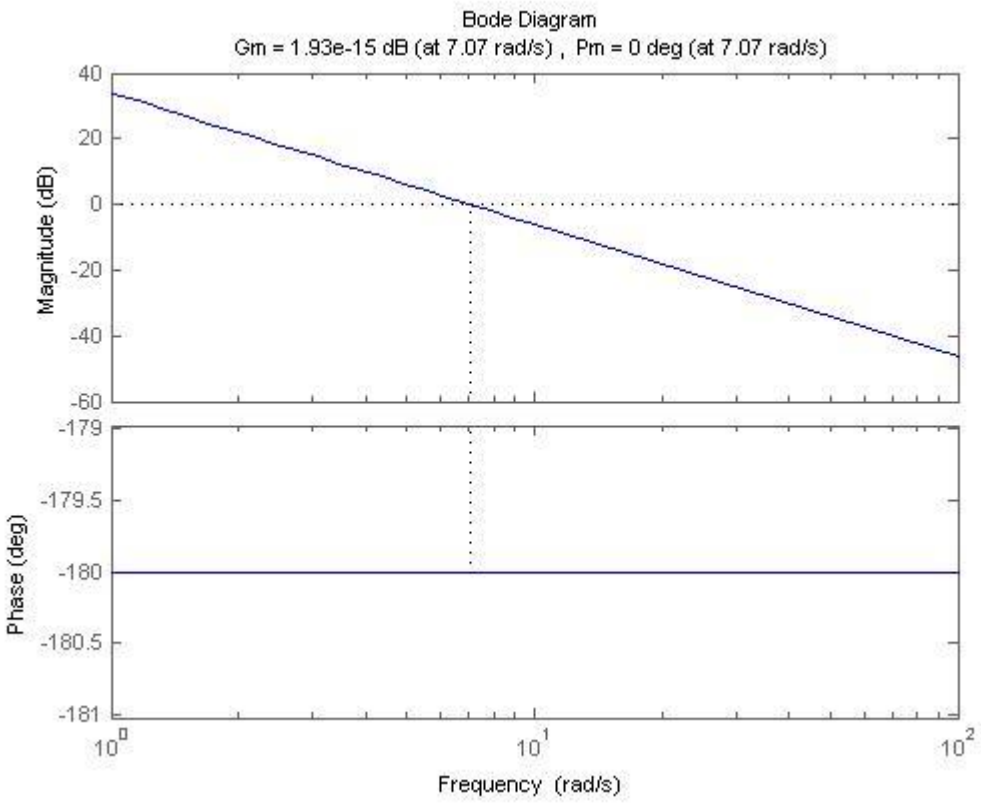


Figure 12: Bode response for the effect of Roll Angle due to rotor 2

As we can see the Gain margin for this response is $1.93 * 10^{-15} dB$ with Phase margin at 0 and the gain cross over frequency is $7.07\ rad/\ sec$. The system is close to marginally stable at $7.07\ rad/sec$. If disturbance occurs, which increases the gain, the system will go to an unstable zone, which is considered to be malfunctioning of the system.

Note:

- We can see that the phase cross over frequency, which determines the Gain Margin is a line (combination of points) here, rather than a point.
- So, for each point (which represents a frequency value) we can find a Gain margin at frequency value.

By assuming the above notes, we see that the system is working at ω less than $7.07\ rad/\ sec$ (Gain cross over frequency) and for value of ω more than $7.07\ rad/\ sec$, the system will not work properly because of having negative Gain Margin resonance frequency.

The bode plot for the response of thrust provided to rotor 2 to the Roll Rate is,

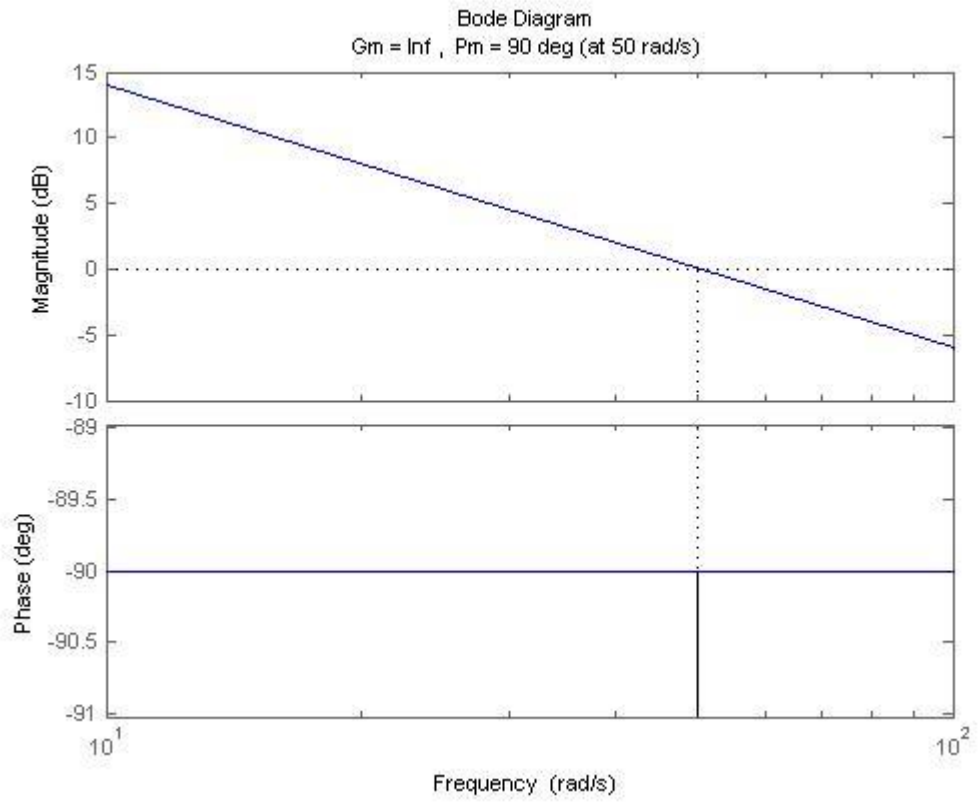


Figure 13: Bode plot response of the Roll Rate for the effect of rotor 2

The Gain margin for the response is infinity, whereas the Phase margin is at 90° and Gain cross over Frequency is $50\ rad/\ sec$. The system is highly stable because, the Gain Margin is greater than $12\ dB$ and the Phase Margin is greater than 45°. The system will be stable for any disturbance it faces.

The response of the thrust provided to rotor 3 to the Pitch angle is

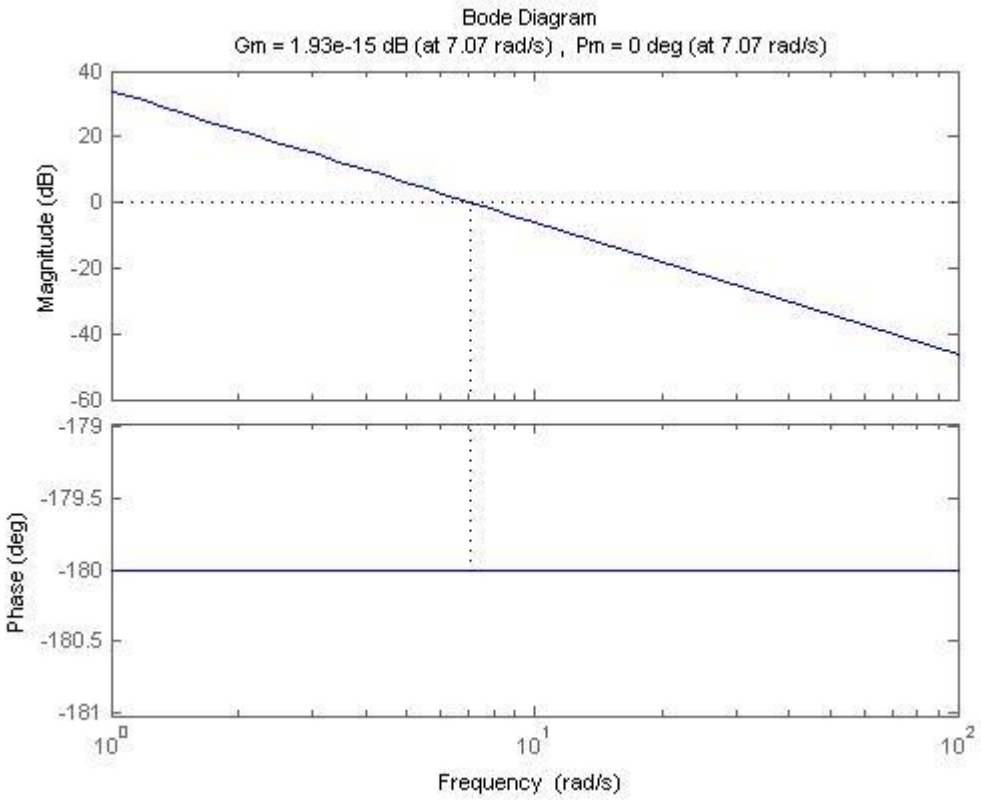


Figure 14: Bode plot of Pitch Angle due to response of rotor 3

The system response is similar to the first plot, which is that the system is close to marginally stable at 7.07 $rad/\ sec$, if there is any disturbance, the system would become unstable.

The response of the thrust produced by rotor 3 to the Pitch rate is given by,

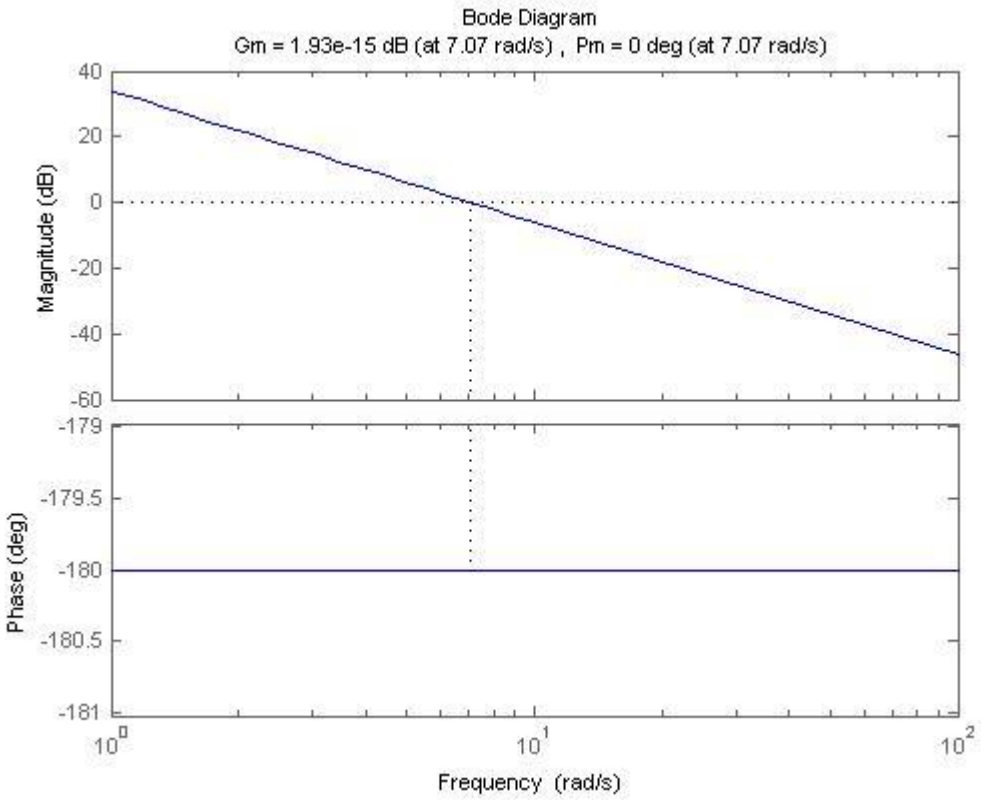


Figure 15: Bode plot response of Pitch Rate due to rotor 3

As per the second plot, the Gain margin is infinity and Phase margin is $90°$. So, the system is considered to be completely stable.

The response of the thrust produced by rotor 4 to the Yaw angle is given by,

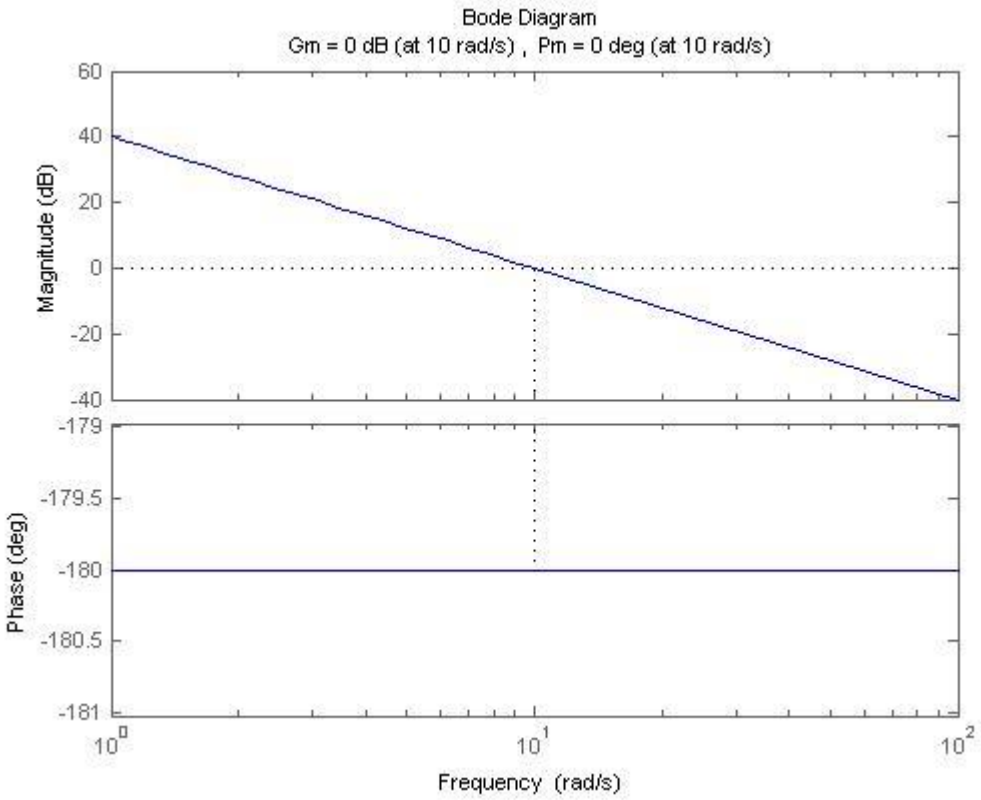


Figure 16: Bode Plot response of Yaw Angle due to rotor 4

The system's Gain margin is $0\ dB$, Phase margin is $0°$and the Gain cross over frequency is at $10\ rad/\ sec$. The frequencies as well as the gain are not good at all; neither the Gain Margin, nor the Phase Margin is good. The Gain Margin is less than $12\ dB$ even less than $6\ dB$, which means that the performance of the system will be bad. This means that the instability may occur anytime. The Phase Margin is less than $45°$ actually it is zero, which means that the system is marginally stable.

Note that, if we increase the Gain Margin, hence the Gain stability by operating the system at low frequency less than $10\ rad/\ sec$.

The bode response for the thrust produced by rotor 4 to the Yaw rate is given by,

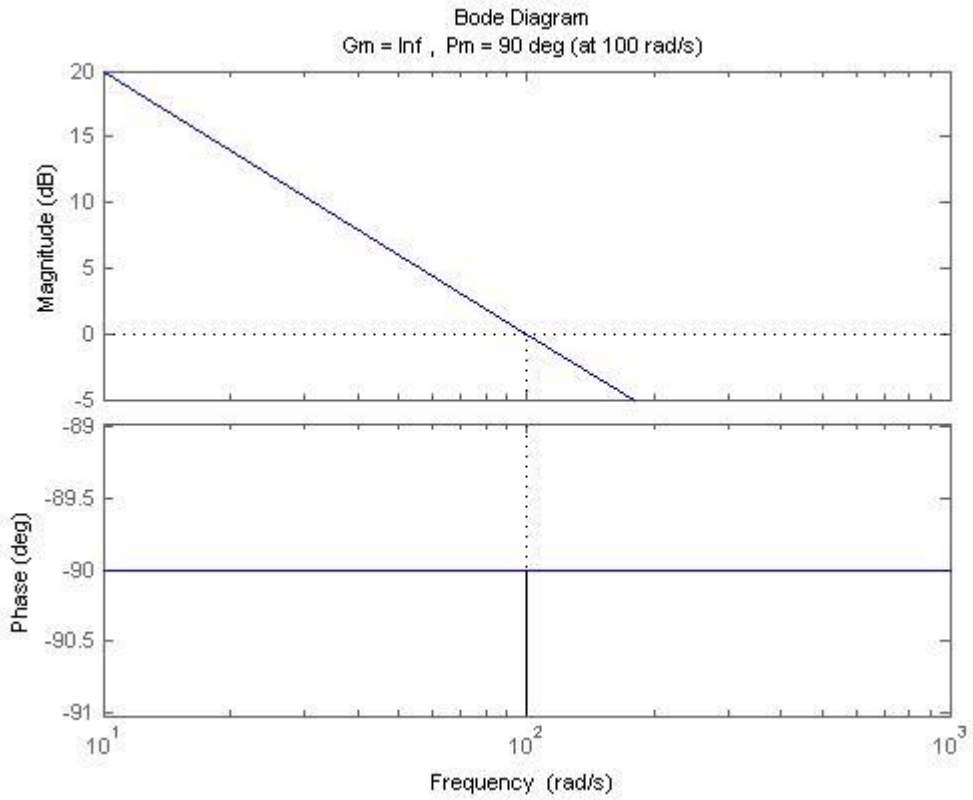


Figure 17: Bode Plot response of Yaw Rate due to rotor 4

The Gain Margin is infinity, Phase Margin is 90 degrees and the Gain cross over frequency is at $100\ rad/\ sec$. It means that the system has very good frequency specifications. That is, the Gain Margin is greater than $12\ dB\ (\infty)$, so gain can be increased as high as we want without disturbing the performance of the system. The Phase Margin is greater than 45°, so the system is very stable. The condition for the system is very good, that is even in case of any disturbance the system can maintain stability and work properly.

Chapter 4 Simulink:

Simulink is a block diagram environment for multi domain simulation and Model Based Design. It supports system level design, simulation, automatic cod generation and continuous test and verifications of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modelling and simulating dynamic systems. It is integrated with Matlab, enabling us to incorporate Matlab algorithms into model and export simulation results into Matlab for further analysis.

The advantages of using Matlab Simulink over other programming software are [22]:

- A very large database of built-in algorithms for image processing and computer vision applications.
- Matlab allows us to test algorithms immediately without recompilation. We can type something at the command line or execute a section in the editor and immediately see the results, greatly facilitating algorithms development.
- The ability to auto- generate C code using Matlab Coder, for a large subset of image processing and mathematical functions, which we could then use in other environments, such as embedded systems or as a component in other software.

4.1 Model Based Design:

Model- based design is a process that enables fast and cost effective development of dynamic systems; including control systems, signal processing and communications systems [23]. In Model- Based design, a system model is at the centre of the development process, from requirements development through design, implementation and testing. After model development, simulation shows whether the model works correctly.

4.1.1 Modelling, Simulation and Analysis with Simulink:

With Simulink, we can move beyond idealized linear models to explore realistic nonlinear models, factoring in friction, air resistance, and other parameters that describe real- world phenomena. Simulink enables us to think of the development environment as a laboratory for modelling and analysing systems that would not be possible or practical otherwise.

After we define the model, we can simulate its dynamic behaviour using a choice of mathematical integration methods either interactively in Simulink or by entering commands in the Matlab Command Window. Commands are particularly useful for running a batch of simulations. Using scopes and other display blocks, we can see the simulation results while a simulation runs. We can then change parameters and see what happens for the changes made. We can save simulation results in the Matlab workspace for post processing and visualization. From the various Analysis tools available, we can linearize the model for further use. Since Matlab and Simulink are integrated, we can simulate, analyse and revise our models in either environment.

4.1.2 System:

Identify the components of a system, determine the physical characteristics, and define dynamic behaviour with equations. We perform these steps outside of the Simulink software environment and before we begin building our model.

4.1.3 Determining Modelling goals:

Before designing a model, we need to understand our goals and requirements. We need to ask ourselves these questions to help plan our model design:

- What problems does the model help us solve?
- What questions can it answer?
- How accurately must it represent the system?

4.1.4 Identifying System Components:

Once we understand our modelling requirements, we can begin to identify the components of the system.

- Identify the components that correspond to structural parts of the systems. Creating a model that reflects the physical structure of a system.
- Identify functional parts that we can independently model and test.
- Describe the relationships between components, for example, data, energy, and force transfer.

4.1.5 Defining System Equations:

After we identify the components in a system, we can describe the system mathematically with equations. Derive the equations using scientific principles or from the input-output response of measured data. Many of the system equations fall into three categories:

- For continuous systems, differential equations describe the rat of change for variables with the equations defined for all values of time. For example, velocity of a car is given by the second order differential equation

$$\frac{dv(t)}{dt} = -\frac{b}{m}v(t) + u(t)$$

- For discrete systems, difference equations describe the rate of change for variables, but the equations are defined only at a specific time. For example, the control signal from a discrete propositional derivative controller is given by the difference equation

$$pd[n] = (e[n] - e[n-1])K_d + e[n]K_p$$

- Equations without derivatives are algebraic equations. For example, the total current in a parallel circuit with two components is given by the algebraic equation. Most of the equations used in the thesis come under algebraic equation. For example, an algebraic formula used in the thesis is given below.

$$I_t = I_a + I_b$$

4.1.6 Collect Parameter Data:

Firstly, create a list of equation variables and constant coefficient, and then determine the coefficient values from published sources or by performing experiments on the system. Then use the measured data from the system to define equation coefficient and parameters in our model.

- Identify the parts that are measurable in a system
- Measure physical characteristics or use published property values. Manufacturer data sheets are a good source for hardware values.

4.1.7 Model System:

Build individual model components that implement the system equations, and define the interfaces for passing data between components.

4.1.8 Model Top- Level Structure:

A model in Simulink is defined as a graphical representation of a system using blocks and connections between links. Once we finish defining a system, its components and equations, we can begin to build our model.

- Use system equations to build a graphical model of a system with the Simulink editor.
- If we place all the models in one level of a diagram, it will be difficult to read and understand if an error occurs or another scenario. One way to organize our model is to make use of Subsystems.
- Identify the inputs and outputs connections between subsystems (for example feedback connection, etc.) The input/ output values change dynamically during a simulation.
- Find out the constants for each components and the values that do not change unless, we change it.
- The variables for ach components and the values that change over time
- The state variables that components have.

After we build a model component, we can simulate to validate the design.

- Predict the expected output of the integrated model components.
- Add blocks to approximate the actual inputs and control value.
- Validate the model design by comparing the simulation output and our expected output.
- If the result does not match our prediction, change our model to improve the accuracy of our prediction.

4.1.9 Connecting Model components:

After we build and validate each model components, we can connect them into a complete model, simulate the model, and analyse the results.

- Integrate the model components by first connecting two of them (for example, the plant and the controller). After validating the pair by simulation, continue connecting components until our model I complete.
- We must take care of how each component we add affects the other parts of the model.

4.1.10 Simulating Connected Components:

Validating our model determines if it accurately represents the physical characteristics of the modelled dynamic system.

- Predict the expected simulation results and outputs of the sub systems.
- Simulate the subsystems and compare the simulated results without expected results.

4.2 Simulation:

Simulation is a process in which we validate and verify the model by comparing simulation results with:

- Data collected from a real system.
- Functionality describes in the mode requirements.

4.2.1 Determine Simulation Goals:

Before we simulate a mode we need to understand our goals and requirements. Some of the possible Simulation goals are,

- Understand input to output causality- I.e. for a given input set and nominal part values, look at how the input flow through the system to the output.
- Verify mode- Compare the simulation results with collected data from the model system. Iteratively debug and improve design.
- Optimize parameters- Change parameters and compare simulation runs.

4.2.2 Collect Data:

Collect input and output data from an actual system. Use measured input data to drive the simulation. Use measured output data to compare with the simulation results from our model.

We will use the measured input and output values to validate the model.

4.2.3 Prepare model:

Preparing the model for simulation includes defining the external interfaces for input data and control signals, and output signals for viewing and recording simulation results.

4.2.4 Set Parameters:

For the first simulation, use model parameters from the validated model. After comparing the simulation result with measured output data, change model parameters to more accurately represent the modelled systems.

4.2.5 Run and Evaluate Simulation:

Simulate our model and verify that the simulation results match the measured data from the modelled system.

4.2.6 Import data:

Simulink enables us to import data into our model. For large data sets, use a MATLAB MAT i.e. with an import block.

4.2.7 Run Simulation:

Using measured input data, run a simulation and save results.

4.2.8 Evaluate Result:

Evaluate the differences between simulated output and measured output data. Use the evaluation to verify the accuracy of our model and how well it represents the system behaviour. Decide if the accuracy of our model adequately represents the dynamic system behaviour. Decide if the accuracy of our model adequately represents the dynamic system we are modelling.

4.2.9 Change Model:

Determine the changes to improve our model,

- Parameters- some parameters were initially estimated and approximated. Optimize and update parameters.
- Adding structure- Some parts or details of the system were not modelled. Add missing details.

4.3 Stability:

Stability in the general term is defined as the ability of the object to return to its equilibrium if disturbed. Static stability is defined as an object's initial tendency upon displacement. An object with the initial tendency to return to its equilibrium position is said to have positive

static stability. Dynamic Stability of a system or object is defined as the ability to return to a previously established steady motion, after being perturbed. Used, for instance, to refer to the ability of a walking robot to maintain its balance while moving. The static and Dynamic stability is illustrated in the following figure.

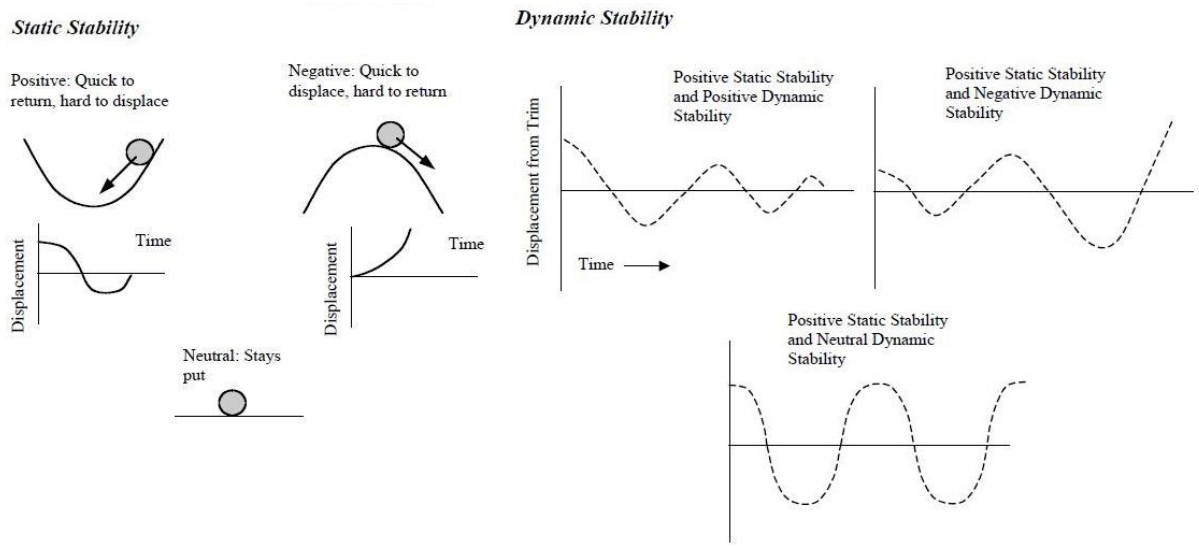


Figure 18: Static and Dynamic Stability

Period is time per cycle. Frequency, is inversely proportional to period, is cycles per unit of time. Amplitude is the difference between the crest or the trough and the original equilibrium condition. Damping is the force that decreases the amplitude of the oscillation with each cycle. The damping ratio is the time for one cycle divided by the total time it takes for the oscillation to subside. The higher the damping ratio, the more quickly the motion disappears. There are two modes of pitch oscillation: the heavily damped short period mode (damping ratio about 0.3 or greater), followed by the lightly damped, and more familiar, long period Phugoid mode.

Short Period mode is excited by a change in angle of attack. The change could be caused by a sudden gust or by a longitudinal displacement of the stick. The lightly damped, long period, or Phugoid, oscillation can take minutes to play out. But it doesn't get to very often, unlike the short mode. During the Phugoid the aircraft maintains essentially a constant angle of attack.

Chapter 5 Controller

Proportional-integral-derivative controller is a control loop feedback mechanism widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired set-point. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable. The PID controller algorithm involves three separate constant parameters and is accordingly: the proportional, the integral and the derivative values, denoted by P, I, D [24].

5.1 SISO approach

A single-input and single-output (SISO) system is a simple single variable control system with one input and one output. As the linear model of the Quadcopter shows, it is possible to use SISO approach for controlling attitude components. SISO systems are typically less complex compared to MIMO systems [25]. Hence a SISO approach is advised for designing a PID controller for the system.

5.2 PID Controller

PID (Proportional- Integral- Derivative) is a closed loop control system that tries to get the actual result closer to the desired result by adjusting the input. Quadcopter or multicopters use PID to achieve stability. Tuning of the PID controllers has been attracting interest for six decades. Numerous methods have been suggested so far try to accomplish the task by making use of different representations of the essential aspects of the process behaviour [26]. Among the well- known formulas are the Ziegler- Nicolas rule, the Cohen-Coon method, IAE, ITAE, and the internal model control. Control parameters are usually tuned so that the closed- loop system meets the following three objectives:

1. Stability and stability robustness, usually measured in frequency domain.
2. Transient response, including rise time, overshoot, and settling time.
3. Steady state accuracy [27].

The figure [31] below shows control block diagram that can be used for each one of ϕ, θ, ψ components. As shown in the figure, one controller should be designed for each one ofϕ, θ, ψ.

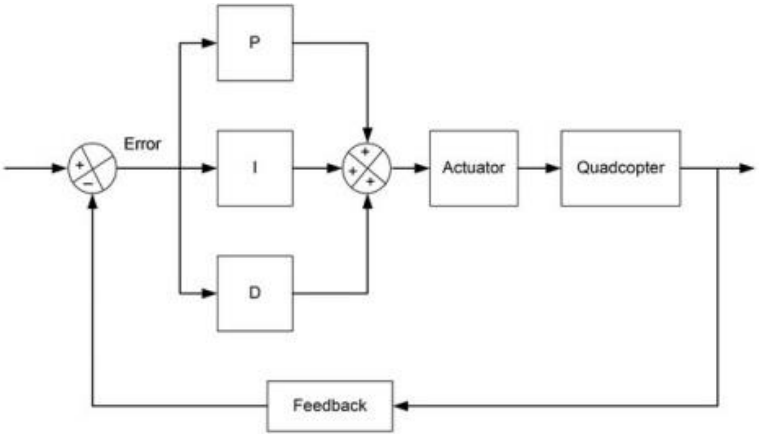


Figure 19: Simulink structure of PID Controller

Where,

Quadcopter is the plant for which the controller has to be designed. Here it is the linearized state space model.

The "PID" block is the PID controller for the system. The generalized transfer function of the PID controller is given by,

$$K_p + \frac{K_i}{s} + K_d s$$

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d\, s^2 + K_p s + K_i}{s}$$

K_p = Proportional Gain.

K_i = Integral Gain

K_d = Derivative Gain

There are 3 algorithms in a PID controller; they are P, I and D respectively. P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. These controller algorithms are translated into software code lines.

To have any kind of control over the Quadcopter or multicopters, we need to be able to measure the Quadcopter sensor output (for example the pitch angle), so we can estimate the error (how far we are from the desired pitch angle, e.g. horizontal, 0 degree). We can then apply the 3 control algorithms to the error, to get the next outputs for the motors aiming to correct the error.

First, let's see how the PID controller works in a closed- loop system using the schematic shown above. The variable (e) represent the tracking error, the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID

controller, and the controller computes both the derivative and the integral of this error signal. The signal (U) just past the controller is now equal to the proportional gain (K_p) ties the magnitude f the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error.

$$u = K_p e + K_i \int e\, dt + K_d \frac{de}{dt}$$

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computed its derivative and it is integral again. This process goes on and on.

5.2.1 Effect of each parameter

The variation of each of these parameters alters the effectiveness of the stabilization. Generally there are 3 PID loops with their own"PID"coefficients, one per axis, so you will have to set P, I and D values for each axis ($pitch\ \theta, roll\ \phi and\ yaw\ \psi$).

To a Quadcopter, these parameters can cause this behaviour.

- **Proportional Gain coefficient –** Your Quadcopter can fly relatively stable without other parameters but this one. This coefficient determines which is more important, human control or the values measured by the gyroscopes. The higher the coefficient, the higher the Quadcopter seems more sensitive and reactive to angular change. If it is too low, the Quadcopter will appear sluggish and will be harder to keep steady. You might find the Quadcopter starts to oscillate with a high frequency when P gain is too high.
- **Integral Gain coefficient –** This coefficient can increase the precision of the angular position. For example, when the Quadcopter is disturbed and its angle changes from $20°$, in theory it remembers how much the angle has changed and will return $20°$. In practise if you make your Quadcopter go forward and the force it to stop, the Quadcopter will continue for some time to counteract the action. Without this term, the opposition does not last as long. This term is especially useful with irregular wind, and ground effect (turbulence from motors). However, when the I value get too high your Quadcopter might begin to have slow reaction and a decrease effect of the proportional gain as consequence, it will also start to oscillate like having high P gain, but with a lower frequency.

- **Derivative Gain coefficient –** This coefficient allows the Quadcopter to reach more quickly the desired attitude. Some people call it the accelerator parameter because it amplifies the user input. It also decrease control action fast when the error is decreasing fast. In particle it will increase the reaction speed and in certain cases an increase the effect of the P gains.

5.2.2 The characteristics of P, I and D controllers:

A proportional controller (K_p) will have the effect of reducing the rise time and will reduce but never eliminate the steady-state error. An integral controller (K_i) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative controller (K_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Effects of each of controller K_p, K_i, K_d on a closed-loop system are summarized in the table shown below,

Controller	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decreases	Increases	Small Change	Decreases	Degrade
K_i	Decreases	Increases	Increases	Eliminate	Degrade
K_d	Small Change	Decreases	Decreases	No effect in theory	Improve if K_d is small

Table 2: Response of the different parameters of the Proportional Integral and Derivative gain

The performance chart for the K_p, K_i, K_d are taken from the [28] reference mentioned below.

Note that these correlations may not be exactly the same, because K_p, K_i, K_d are dependent on each other. In fact, changing one of these variables can change the effect of the other two.

5.2.3 How to tune Quadcopter PID Gains

I usually tune one parameter at a time, start with P, I and then D gain. We can also go back to fine tune the values I need.

For P gain, I fist start low and work my way up, until I notice it is producing oscillations. Fine tune it until you get to a point it is not sluggish & there is no oscillation.

For I gain, again start low and increase slowly. Roll and Pitch our Quadcopter left and right, pay attention to the how long does it take to stop and stabilize. We want to get to a point where it stabilizes very quickly as we release the stick & it does not wander around for too low. We might also want to test it under windy condition to get a reliable I value.

For D gain, it can get into a complicated interaction with P and I values. When using D gain, we need to go back and fine tune P and I to keep the plant well stabilized.

Quadcopter are symmetric so we can set the same gain values for Pitch and Roll. The values for Yaw is not very important as those of Pitch and Roll so it is probably OK to set the same values as for Pitch/ Roll to start with (even it might not be the best). After our multi-copter is relatively stable, we can start alter the Yaw gain. For non-symmetric multi-copter like, Hexa-

copter, Tri-copter, we might want to fine tune the pitch, Roll separately, after we might want to fine tune the Pitch and Roll separately, after we have some flight experience.

Consider, the following unity feedback system,

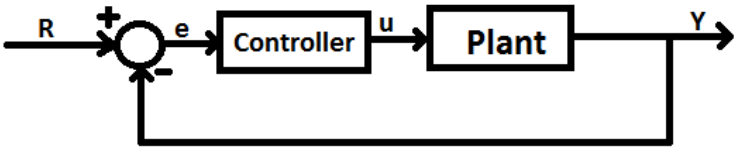


Figure 20: Controller-Plant Simulink Structure

Plant: A system to be controlled

Controller: Provides the excitation for the plant, designed to control the overall system behaviour.

5.2.4 Proportional Controller:

From the table shown above, we see that the proportional controller (Kp) reduces the rise time, increases the overshoot and reduces the steady- state error. The closed loop transfer function of the proportional controller is given by,

$$TF(s) = \frac{K_p\ b_0}{s^2 + a_1 s + (a_0 + K_p b_0\ )}$$

5.2.5 Proportional Derivative Controller:

From the table, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed loop transfer function of the PD controller is given by,

$$TF(s) = \frac{K_d\ b_1 s + K_p b_0}{s^2 + (a_1 + K_d b_1)s + (a_0 + K_p b_0)}$$

5.2.6 Proportional Integral Controller:

We see that the Integral controller (Ki) decreases the rise-time, increases both the overshoot and the settling time, and eliminates the steady – state error. The corresponding closed loop transfer function of a PI controller is given by,

$$TF(s) = \frac{b_0(K_p s + K_i)}{s^3 + a_1 s^2 + \left(a_0 + K_p b_0\right)s + K_i b_0}$$

5.2.7 Proportional Integral and Derivative Controller:

The closed loop transfer function of a PID controller is given by,

$$TF(s) = \frac{b_0(K_p s + K_i + K_d s^2)}{s^3 + (a_1 + K_d b_0)s^2 + (a_0 + K_p b_0)s + K_i b_0}$$

The closed loop transfer function for the controllers are taken from the reference [29]

For example, consider an example shown below,

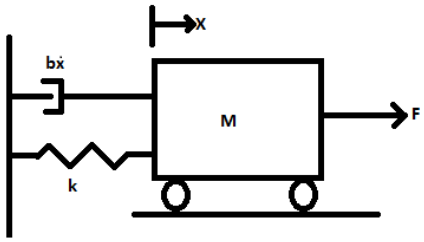


Figure 21: A simple Mechanical model

The modelling equation of the system is

$$M\ddot{x} + b\dot{x} + kx = F$$

Taking the Laplace transform of the modelling equation, we get

$$Ms^2X(s) + bsX(s) + kX(s) = F(s)$$

The transfer function between the displacement X(s) and the input F(s) then becomes

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

Let,

- $M = 1\ kg$
- $b = 10\ Ns/m$
- $k = 20\ N/m$
- $F(s) = 1$

The corresponding transfer function is given by,

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

Depending on the required goal, the best suitable controller can be chosen. Consider for an example the response of a system; we have the following goals to be satisfied:

- Fast rise time
- Minimum overshoot
- No steady- state error

Add a new m-fie, and type in a code to find the step response of the system, the code for the system and the step response of the above system is given by,

$$num = 1;$$

$$den = [1\ 10\ 20];$$

$$plant = tf\,(num, den);$$

$$step(plant)$$

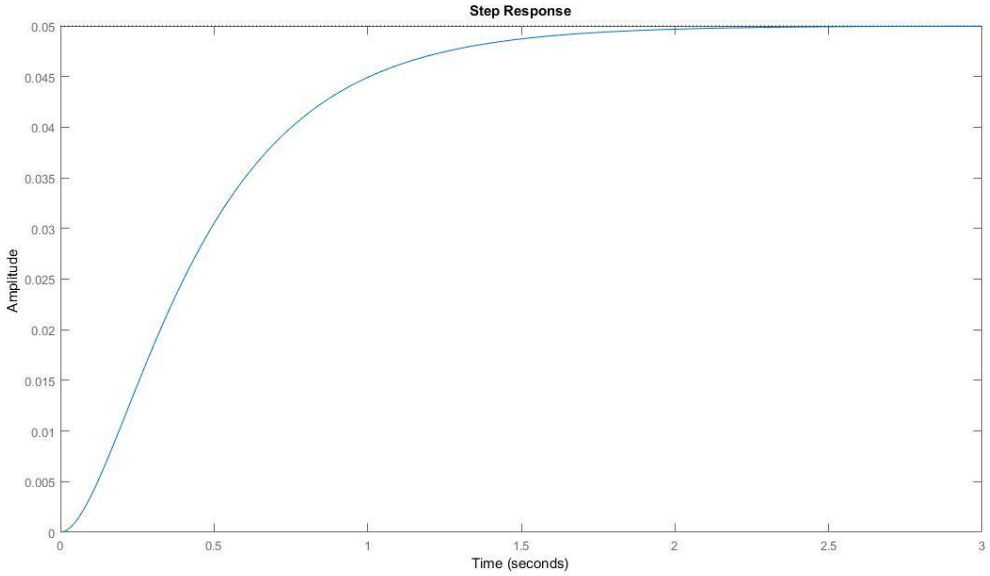


Figure 22: Step response of the Mechanical model

For the above response we see that, the DC gain of the plant transfer function is 1/20, so 0.05 is the final value of the output to a unit step input. This corresponds to the steady state error of 0.95 (which is quite large). Furthermore, the rise time is about one second, and the settling time is about 1.5 seconds. Let's design a controller that will reduce the rise time, reduce the settling time, and eliminates the steady- state error.

First let's consider the Proportional controller, let the proportional gain (Kp) equals 300 and change the m-file to the following:

$$Kp = 300;$$

$$contr = Kp;$$

$$sys\_cl = feedback\,(contr * plant, 1);$$

$$t = 0{:}0.01{:}2;$$

$$step(sys\_cl, t)$$

Running the above m-file, we get the following response,

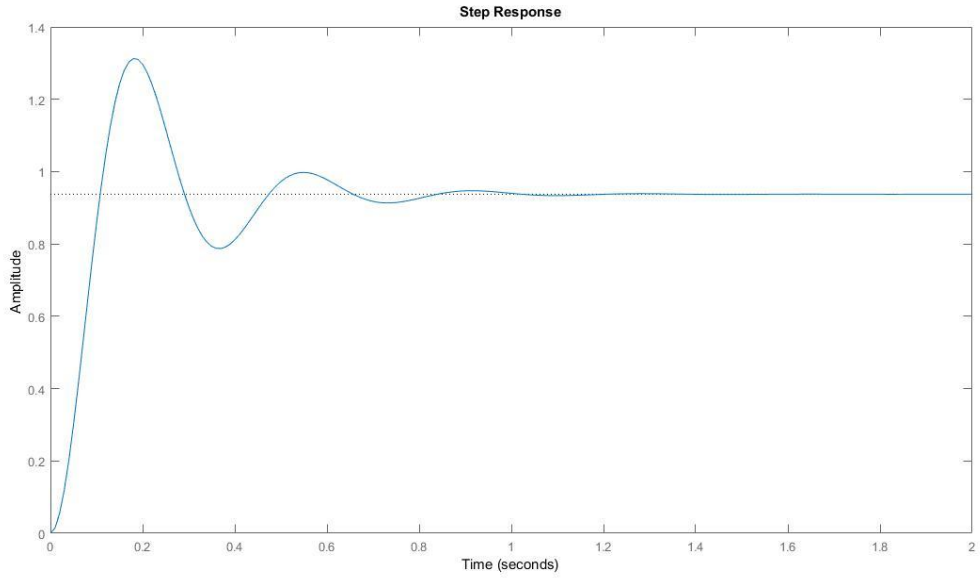


Figure 23: Proportional controller response of the Mechanical Model

(Note: The MATLAB function called feedback was used to obtain a closed-loop transfer function directly from the open-loop transfer function)

The above plot shows that the proportional controller reduced both the rise time and the steady state error, increased the overshoot and the decreased the settling time by small amount.

Let's now consider the Proportional- Derivative controller, the proportional gain is equal as before (I.e. 300) and let K_d equal to 10. The above m-file is modified according to the gain we have considered and the m-file is run and the response is taken.

$$Kp = 300;$$

$$K_d = 10;$$

$$contr = tf([Kd\ Kp], 1);$$

$$sys\_cl = feedback(contr * plant, 1);$$

$$t = 0{:}\,0.01{:}\,2;$$

$$step(sys\_cl, t)$$

This plot shows that the derivative controller reduced both the overshoot and the settling time, and had a small effect on the rise time and the steady state error.

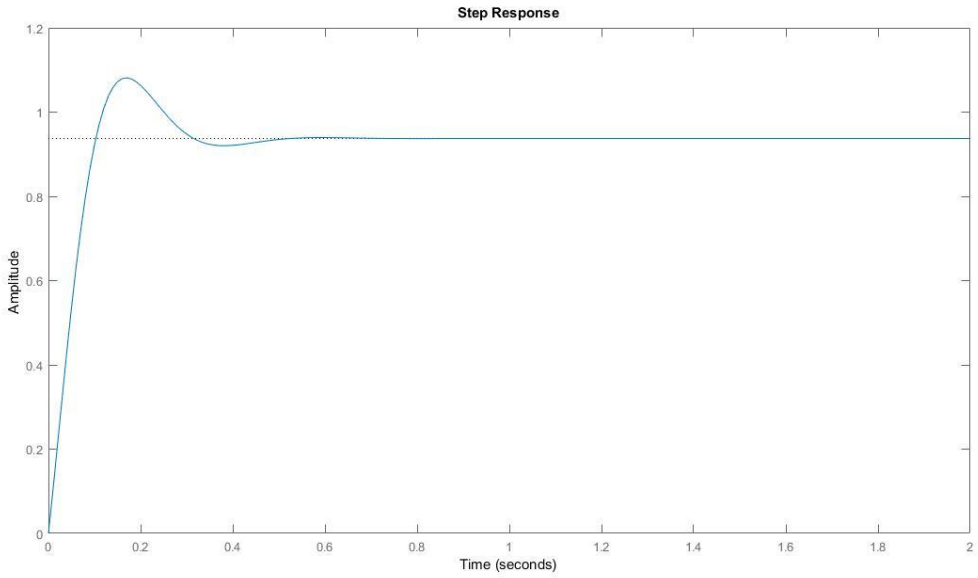


Figure 24: Response of the model due to Proportional Derivative controller

Now, let us consider the Proportional Integral Controller. Let us reduce the proportional gain to 30, and let integral gain be equal to 70. The m-file is edited according to the gain we have changed.

$$Kp = 30;$$

$$Ki = 0;$$

$$contr = tf([Kp\ Ki], [1\ 0]);$$

$$sys\_cl = feedback(contr * plant, 1);$$

$$t = 0{:}\,0.01{:}\,2;$$

$$step(sys\_cl, t)$$

The following response of the system with the PI controller is shown below,

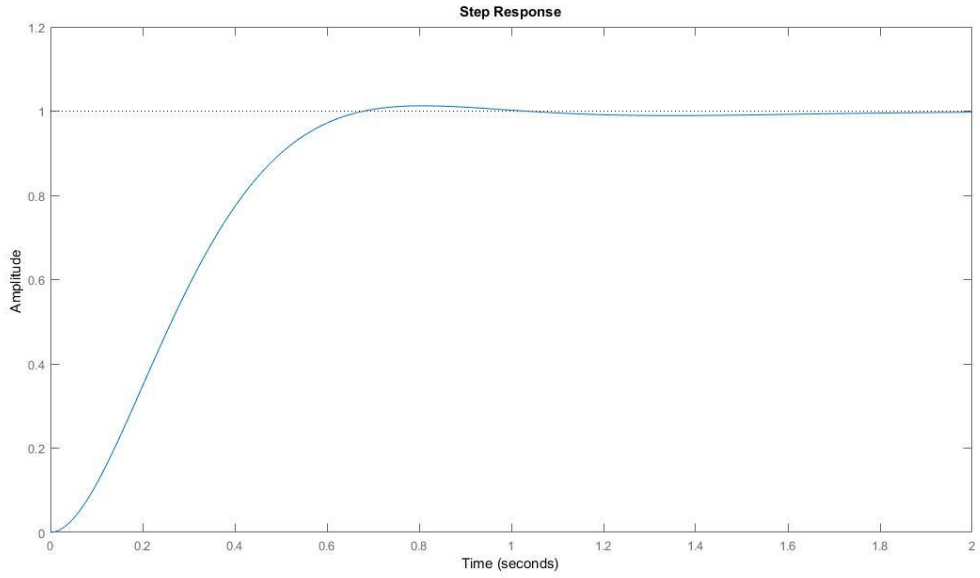


Figure 25: Response of the model due to the Proportional Integral controller

From the above plot, we see that the proportional gain (Kp) because the integral controller also reduces the rise time and increases the overshoot as the proportional controller does (double effect). The above response shows that the integral controller eliminated the steady-state error.

Now, let's consider the PID controller. After several trial and error runs, the gains $Kp = 350, Ki = 300\ and\ Kd = 50$ provided the desired response. To confirm, enter the following commands to an m-file and run it in the command window.

$$Kp = 350;$$

$$Ki = 300;$$

$$Kd = 50;$$

$$contr = tf([Kd\ Kp\ Ki],[1\ 0]);$$

$$sys\_cl = feedback(contr * plant, 1);$$

$$t = 0{:}\,0.01{:}\,2;$$

$$step(sys\_cl, t)$$

For the above command, we have obtained the following step response.

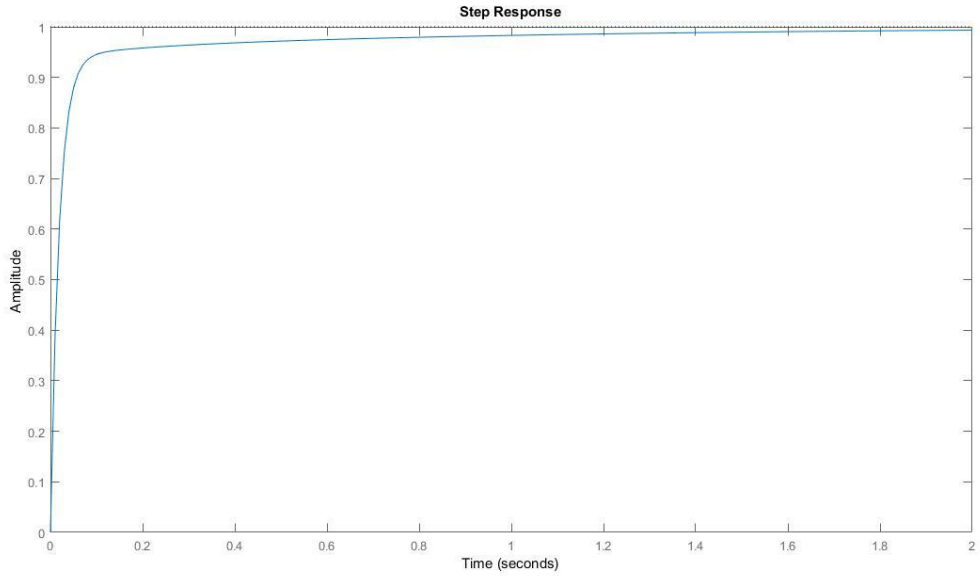


Figure 26: Response of the model due to the Proportional Integral and Derivative controller

Now, we have obtained a closed-loop system with no overshoot, fast rise time and no steady-state error.

Chapter 6 Simulation

6.1 Simulation:

Now that we have the complete equations of motion describing the dynamics of the system, we can create a simulation environment in which to test and view results of various inputs and controllers. Here Euler's method is used for solving the differential equations to evolve the system state. I have used Matlab Simulink and script to create a simulation environment and test the controller for my Quadcopter.

The idea for creating the simulation is to define the time variable for the simulation and determining the iterations for the complete time duration. The initial simulation, velocity and the angular displacement is defined to zero state. Some disturbances are also defined in the angular velocity and the magnitude of the deviation is in radians per second. The input from the controller is obtained for the time variables. The linear and angular acceleration for the model; is obtained for the input obtained from controllers. The function for the all the physical forces and torques and defined separately in the simulation program as a separate function variable.

6.2 Control:

The mathematical model of a Quadcopter is derived so that it is easier for developing a controller for the model. Since we can only control the voltage across the motors, the inputs to our system consist of the angular velocities of each rotor. Note that in our model, we can use the square of the angular velocities,ω_i^2, and not the angular velocity, ω_i. For the notational simplicity, let us introduce the inputs $\gamma_i = \omega_i^2$. Let x_1 be the position of the Quadcopter in space, x_2 be the Quadcopter linear velocity, x_3 be the Roll, Pitch, Yaw angles, and x_4 be the angular velocity vector. (Note that all these are 3-vectors i.e. along, X, Y and Z axis) With these being our state, the state space equations for evolution of our state is written as,

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} RT_B + \frac{1}{m} F_D$$

$$\dot{x}_3 = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix}^{-1} x_4$$

$$\dot{x}_4 = \begin{bmatrix} \tau_\phi {I_{xx}}^{-1} \\ \tau_\theta {I_{yy}}^{-1} \\ \tau_\psi {I_{zz}}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy} - I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz} - I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx} - I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix}$$

Note that our inputs are not used in these equations directly. But we will be able to solve for γ_i by choosing the values of τ and T, and then solve for values for γ_i.

6.3 PD Control:

First we will try controlling the model using a PD Controller, with a component proportional to the error between our desired trajectory and the observed trajectory, and a component proportional to the derivative of the error. As the name suggests PD controller is a combination of proportional and a derivative controller the output (also called the actuating signal) is equals to the summation of proportional and derivative of the error signal. Writing this mathematically we have,

$$A(t)\ \alpha\ \frac{de(t)}{dt} + A(t)\ \alpha\ e(t)$$

Removing the sign of proportionality we have,

$$A(t) = K_d \frac{de(t)}{dt} + K_p e(t)$$

Where K_d and K_p proportional constant and derivative constant respectively. [30]

Our Quadcopter will only have a gyro, so we will only be able to use the angle derivatives $\dot{\phi}, \dot{\theta}, \dot{\psi}$ in our controller; these measured values will give us the derivative of our error, and their integral will provide us with the actual error. We would like to stabilize the Quadcopter in a hovering condition, so our required velocities and angles will all be zero. Torques are related to our angular velocities by $\tau = I\ddot{\theta}$, so we would like to set the torques proportional to the output of our controller, with $\tau = Iu(t)$. Thus,

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} -I_{xx}\left(K_d \dot{\phi} + K_p \int_0^T \dot{\phi}\ dT\right) \\ -I_{yy}\left(K_d \dot{\theta} + K_p \int_0^T \dot{\theta}\ dT\right) \\ -I_{zz}\left(K_d \dot{\psi} + K_p \int_0^T \dot{\psi}\ dT\right) \end{bmatrix}$$

We have precisely derived the relationship between torque and our inputs, so we know that

$$\tau_B = \begin{bmatrix} LK\,(\gamma_1 - \gamma_3) \\ LK(\gamma_2 - \gamma_4) \\ b(\gamma_1 - \gamma_2 + \gamma_3 - \gamma_4) \end{bmatrix} = \begin{bmatrix} -I_{xx}\left(K_d\dot{\phi} + K_p\int_0^T \phi\ dT\right) \\ -I_{yy}\left(K_d\dot{\theta} + K_p\int_0^T \dot{\theta}\ dT\right) \\ -I_{zz}\left(K_d\dot{\psi} + K_p\int_0^T \psi\ dT\right) \end{bmatrix}$$

This gives us a set of three equations with four unknowns. We can constraint this by enforcing the constraint that our input must keep the Quadcopter loft:

$$T = mg$$

Note that this equation ignores the fact that the thrust will not be pointed directly up. This will limit the applicability of our controller, but should not cause major problems for small deviations from stability. If we had a way of determining the current angle accurately, we could compensate. If our gyro is precise enough, we can integrate the values obtained from the gyro to get the angles $\theta\ and\ \phi$. In this case, we can calculate the thrust necessary to keep the Quadcopter aloft by projecting the thrust mg onto the inertial z axis. We find that,

$$T_{proj} = mg\cos\theta\cos\phi$$

Therefore, with a precise angle measurement, we can instead enforce the requirement that the thrust be equal to

$$T = \frac{mg}{\cos\theta\cos\phi}$$

In which case the component of the thrust pointing along the positive z axis will be equal to mg. We know that the thrust is proportional to a weighted sum of the inputs:

$$T = \frac{mg}{\cos\theta\cos\phi} = K\sum\gamma_i \Rightarrow \sum\gamma_i = \frac{mg}{K\cos\theta\cos\phi}$$

With this extra constraint, we have a set of four linear equations with four unknowns γ_i. We can simulate this controller using our simulation environment. The controller drives the angular velocities and angles to zero.

The response of my model for the PD controller is given below. Here we are analysing the Angular Velocity and the Angular Displacement of the model for the given input and the Proportional and Derivative controller.

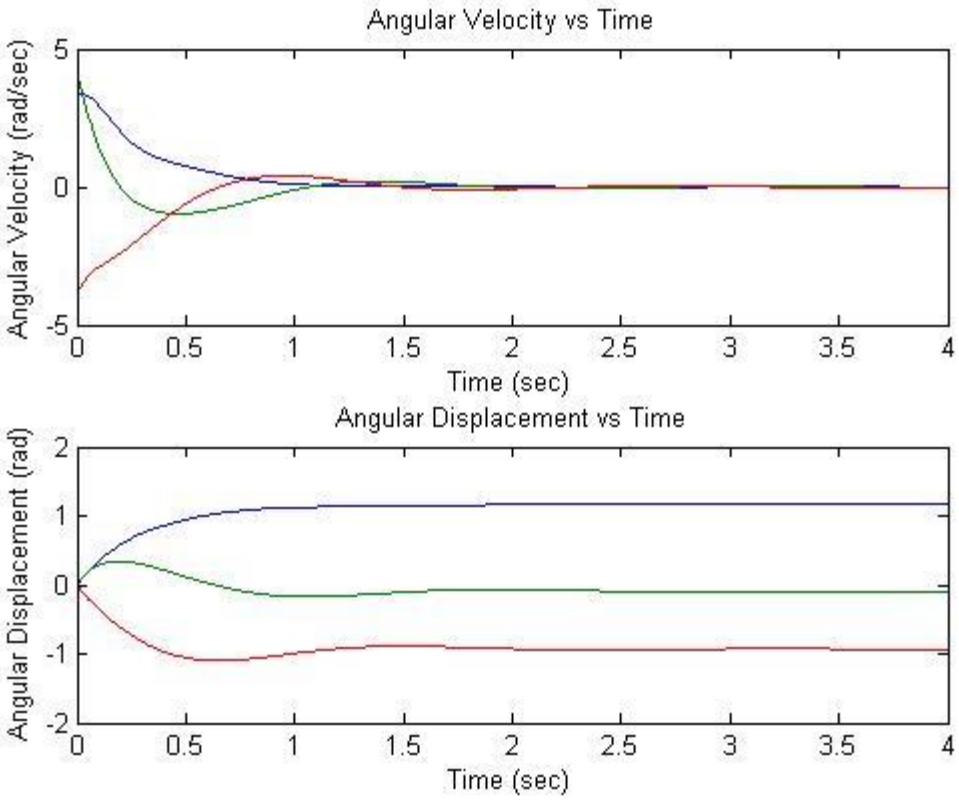


Figure 27: Response of Quadcopter for the Proportional Derivative controller

The plot on the top is the response of the Angular Velocity in radians per second for time in seconds. The other plot is the response of Angular Displacement in radians for the time in seconds. The variables ϕ, θ, ψ are coded as red, green and blue.

Note that the angles from the result are not completely driven to zero. The average steady state error (error after 4 seconds of simulation) is approximately0.3°. This is a common problem with using PD controllers for mechanical systems, which can be partially alleviated with a PID controller, as we will discuss in the next section.

In addition, note that since we are only controlling angular velocities, our positions and linear velocities do not converge to zero. However, the z position will remain constant, because we have constrained the total vertical thrust to be such that it keeps the Quadcopter perfectly aloft, without ascending or descending. However, this is really nothing more than a curiosity. While in theory we could compute the linear velocities and positions from the angular velocities, in practice the values will be so noisy as to be completely useless. Thus we will restrict ourselves to just stabilizing the Quadcopter angle and angular velocity.

We have implemented this PD control for use in our simulation. The controller is implemented as a function which is given some state (corresponding to controller state, not system state) and the sensor inputs, and must compute the inputs and the updated state.

6.4 PID Control

PID control stands for proportional plus derivative plus integral control. PID control is a feedback mechanism which is used in control system. This type of control is also termed as three term control. By controlling the three parameters- proportional, integral and derivative we can achieve different control actions for specific work.

PD controller holds advantages of simplicity and ease of implementation, but they are often inadequate for controlling mechanical systems. Especially if there are noise and disturbances, PD controllers will often lead to steady state error. A PID control is a PD control with another term added, which is proportional to the integral of the process variable. Adding an integral term causes any remaining steady state error to build up and enact a change, so a PID controller should be able to track our trajectory (and stabilize the Quadcopter) with a significantly smaller steady state error. The equations remain the same as PD controller but with an integral term in error:

$$e_{\phi} = K_d \dot{\phi} + K_p \int_0^T \phi \, dt + K_i \int_0^T \int_0^T \phi \, dt \, dt$$

$$e_{\theta} = K_d \dot{\theta} + K_p \int_0^T \dot{\theta} \, dt + K_i \int_0^T \int_0^T \dot{\theta} \, dt \, dt$$

$$e_{\psi} = K_d \dot{\psi} + K_p \int_0^T \psi \, dt + K_i \int_0^T \int_0^T \psi \, dt \, dt$$

However, PID controls come with their own shortcomings. However there is a dis advantage in using PID controller, which is Integral Windup.

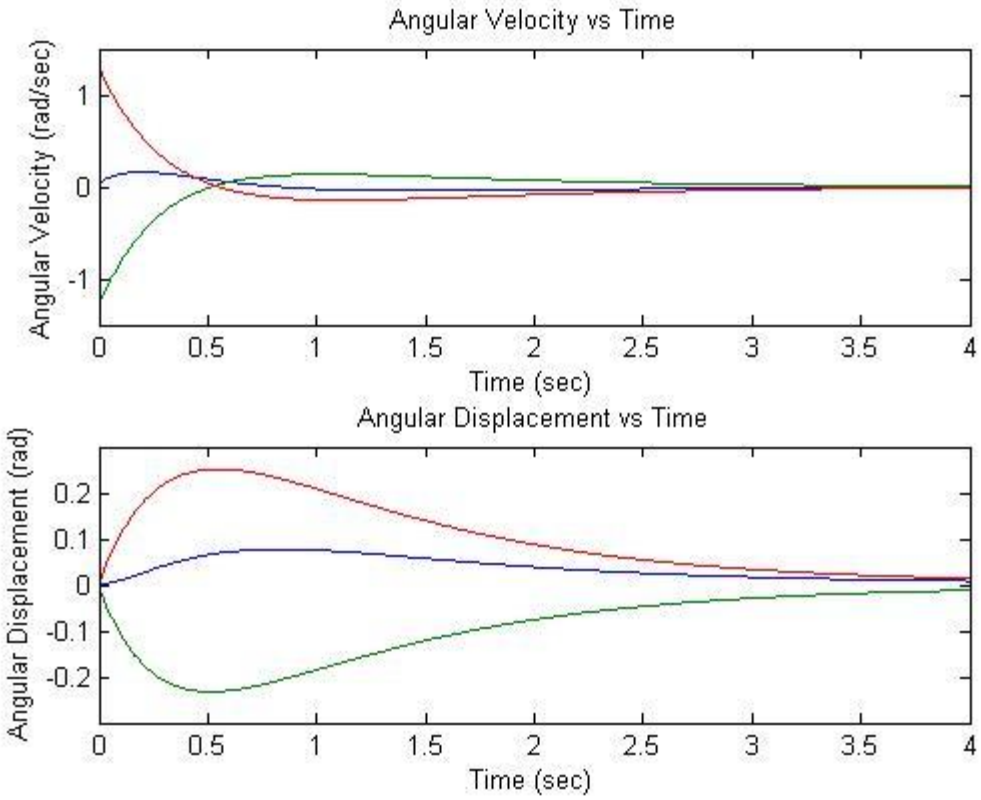


Figure 28: Response of the Quadcopter due to the Proportional Integral and Derivative Controller

Here the Angular Velocity and the Angular Displacement response of the model for the given input and the properly implemented PID gain values are given. As in the PD controller case, the above plot is the response of Angular Velocity and the Angular Displacement for time in seconds. As we can see, the displacement error is in the range of ±1 radians per seconds and ±0.2 radians in Angular Displacement.

We have implemented this PID control for use in simulation, in the same way as with the PD controller shown earlier. Note that there is an additional parameter to tune in a PID. The disturbances used for all the test cases are identical, shown to compare the controllers.

6.5 Automated PID Tuning

Although PID control has the potential to perform very well, it turns out that the quality of the controller is highly dependent on the gain parameters. Tuning the parameters by hand may be quite difficult, as the ratios of the parameters is as important as the magnitudes of the parameters themselves; often, tuning parameters requires detailed knowledge of the systems and an understanding of the conditions in which the PID control will be used. That is in case of presence of unexpected disturbances or noise; the chosen parameters should be good enough to overcome the disturbances. Hence we use automatic PID tuning to obtain an optimal set of parameters. The parameters we chose previously were tuned by hand for good performance, simply by running simulations with many possibly disturbance and parameter

values, and choosing something that worked reasonable well. This method is clearly suboptimal, not only because the resulting gains are not in any way guaranteed to be optimal or even close to optimal.

Ideally, we would be able to use an algorithm to analyse the system and find the "optimal" PID gains, for some reasonable definition of optimal. This problem has been studied in depth, and many methods have been proposed. Many of these methods require detailed knowledge of the system being modelled, and some rely on properties of the system, such as stability or linearity. The method we will use for choosing PID parameters is a method known as extremum seeking.

In Extremum seeking, we defined the "optimal" set of parameters as some vector $\vec{\theta} = (K_p, K_i, K_d)$which minimizes some cost function $J(\vec{\theta})$. In our case, we would like to define a cost function that penalizes high error and error over extended duration of time. The cost function is given by,

$$J(\vec{\theta}) = \frac{1}{t_f - t_0}\int_{t_0}^{t_f} e(t,\vec{\theta})^2\, dt$$

Where $e(t,\vec{\theta})$is the error in following some reference trajectory with some initial disturbance using a set of parameters $\vec{\theta}$. Suppose we were able to somehow compute the gradient of this cost function, $\nabla J(\vec{\theta})$. In that case, we could iteratively improve our parameter vector by defining a parameter update rule for the iteration domain

$$\vec{\theta}(k+1) = \vec{\theta}(k) - \alpha\nabla J(\vec{\theta})$$

Where $\vec{\theta}(k)$ is the parameter vector after k iterations and α is some step size which dictates how much we adjust our parameter vector at each step of the iteration. As $k \to \infty$, the cost function $J(\vec{\theta})$will approach a local minimum in the space of PID parameters.

The question remains as to how we can estimate $\nabla J(\vec{\theta})$. By definition,

$$\nabla J(\vec{\theta}) = \left(\frac{\partial}{\partial K_p}J(\vec{\theta}), \frac{\partial}{\partial K_i}J(\vec{\theta}), \frac{\partial}{\partial K_d}J(\vec{\theta})\right)$$

We know how to compute $J(\vec{\theta})$. Using this, we can approximate the derivative with respect to any of the gains numerically, simply by computing

$$\frac{\partial}{\partial K}J(\vec{\theta}) \approx \frac{J(\vec{\theta} + \delta.\hat{u}_K) - J(\vec{\theta} - \delta.\hat{u}_K)}{2\delta}$$

Where $\hat{u}_K$is the unit vector in the K direction. As $\delta \to 0$, this approximation becomes better. Using this approximation, we can minimize our cost function and achieve locally optimal PID parameters. We can start with randomly initialized errors i.e. positive weights, disturb the

system in some set manner, evaluate $J(\vec{\theta})$ by simulating the system for different PID parameters, and then compute the gradient. Then using the method of gradient descent, we can iteratively optimize the gains until we have some form of convergence.

The gradient descent method does, however, have many disadvantages. First of all, although it finds a local minimum, that minimum is only guaranteed to be a local minimum- there may be other minima which are better global minima. In order to avoid choosing suboptimal local minima in the cost function, we repeat our optimization several times, and choose the best result. We initialize our PID parameters randomly, so each time we run the optimization we will get a different result. In addition, instead of choosing disturbance and optimizing the response to that disturbance, we use random disturbance for all iteration and use the average response to compute costs and gradients. This ensures that our parameters are general and not optimized for a specific disturbance. In addition, we vary the step size and the number of disturbances to try per iteration, in order to increase the sensitivity of our results as our iteration continues. We stop iterations when we detect a steady state, which we do by computing a linear regression on the most recent costs and iterating until the slope is almost zero using a 99% confidence interval.

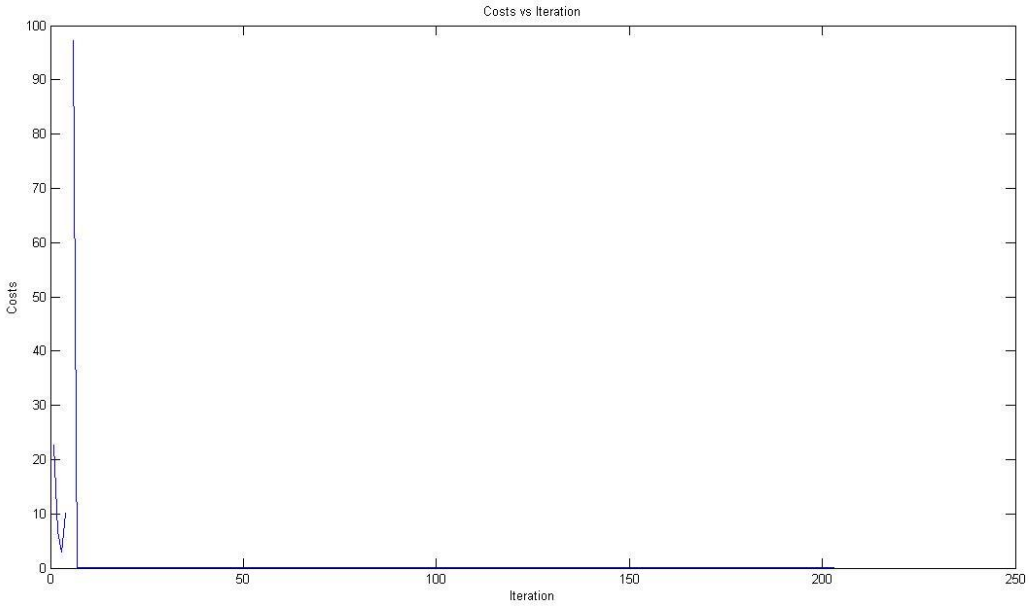


Figure 29: Plot for the Cost vs Iteration

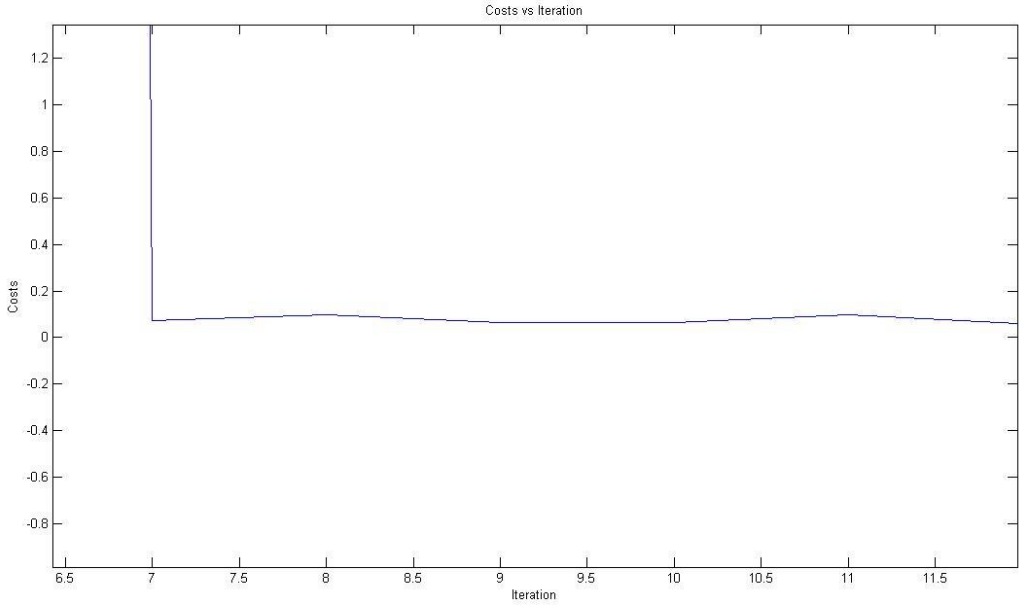


Figure 30: Plot showing the steady level of the cost

Using our Quadcopter simulation, we can define a function that computes the cost for a given set of PID parameters. We can use this function to approximate a derivative with respect to a gain. We can then use our gradient descent method with all the modifications described above to find the minimized cost function and obtain a good set of PID parameters. We can verify that our tuning method is working by visualizing the cost function versus the iteration and seeing that the cost function is indeed going down and stabilizing at a local minimum. We can finally compare the simulation result for the manually chosen PID parameters with those designed by the Automatic PID tuning.

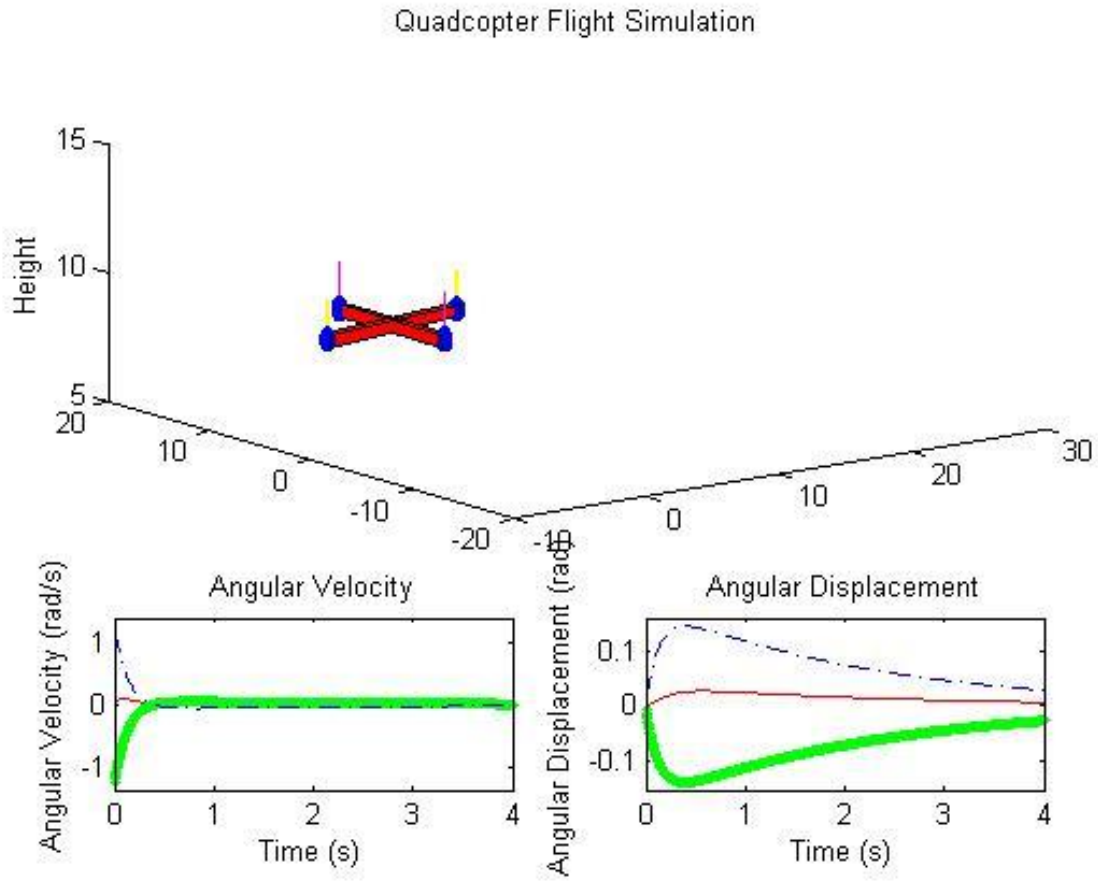


Figure 31: 3- Dimensional visualisation of the Quadcopter for the manually tuned Proportional Integral Derivative controller

This figure describes the performance of the system, using automatically tuned PID controller

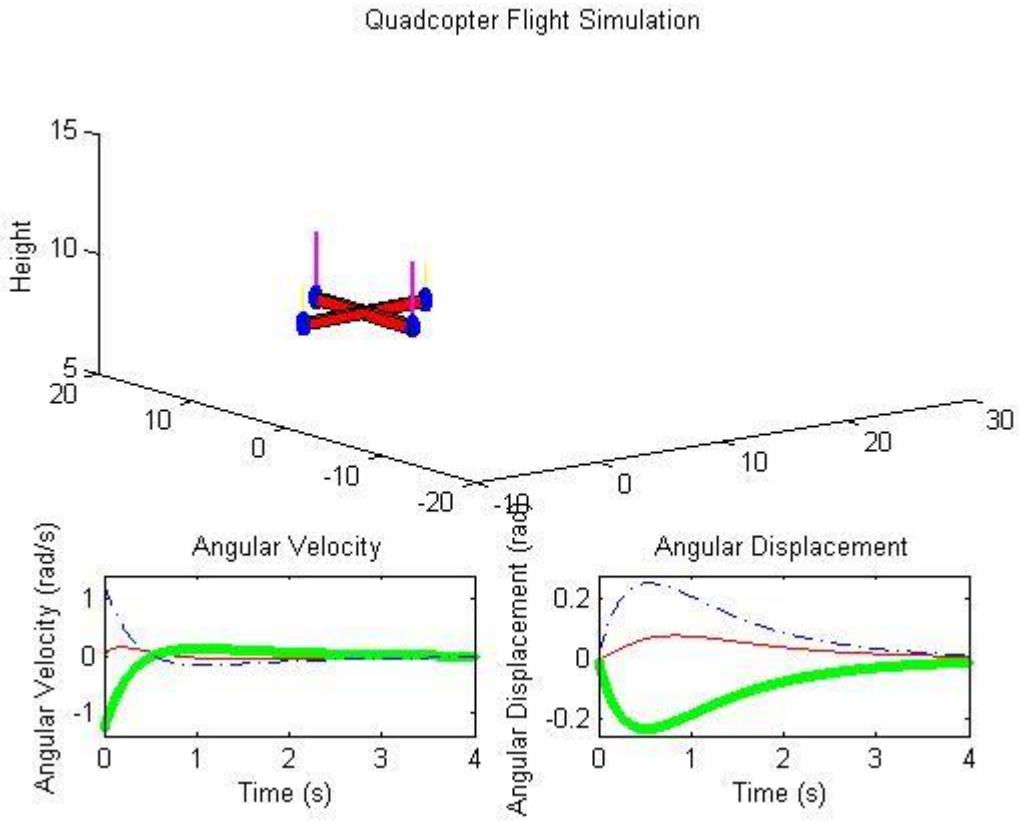


Figure 32: 3- Dimensional visualization of the Quadcopter for the automatically tune Proportional Integral and Derivative gain using Gradient Descent method

The automatically chosen PID parameters do significantly better overall. They have significantly smaller oscillation in value, overshoot is significantly less, and converge faster. However, the error in the angular displacement takes longer to converge to zero with the automatically tuned parameters than with the manually turned parameters, although the initial convergence is much better when the parameters are chosen via gradient descent. This is due to the fact that our cost function emphasizes squared error, and thus gives priority to minimizing overall error magnitude rather than long term convergence. Since, the flight is controlled by Humans, the long term convergence is not given prior importance. We could easily modify our cost function to give higher priority to long term error, in which case the automatically tuned parameters are likely to do better.

Chapter 7 Conclusion

In this paper, the introduction about the Quadcopter is given followed by the advantages of the Quadcopter over comparable scaled helicopter and the application of Quadcopter in day to day life. Later the dynamics of the Quadcopter is briefly explained theoretically. I derived the equation of motion of the Quadcopter, starting with the Voltage-Torque relation for the brushless motor and working through the Quadcopter kinematics and dynamics. I created the non-linearized model of the Quadcopter is using Matlab simulink for the equation of motion derived. This simulator was used to test and visualize Quadcopter control mechanics. I ignored the aerodynamically effects such as blade- flapping and the non-zero free stream velocity. But, the air friction was included as a linear drag forces in all directions. Then, the non-linearized model is linearized for the given operating point using the Jacobian method. Later, the non-linearized and the linearized model are studied for a given set of step input. Then, the response of the linearized model using the root locus plot, bode plot is analysed for the given input and the controllability and observability of the model is determined.

A detailed explanation about the PID control is given, with the PID plant structure, the transfer function and finally the tuning using a SISO tool approach. I began with a simple PD controller, even though the response of the model was good there was a significant steady-state error. In order to decrease the significant steady state error, I introduced the integral term to the controller to create the PID controller. I tested the PID controller with minor modifications to prevent integral wind up. I found out that the steady state error prevented in PID controller was much better that the PD controller when presented with the same disturbances and using the same Proportional and Derivative gains.

I found out that the tuning of PID controller was difficult, and would often lead to an unstable system for unknown disturbances. In order to avoid the difficulties of tuning the PID parameters and find the optimal set of gain values, I used the automatic PID tuning using a gradient descent based extremum seeking method in order to numerically estimate gradient of a cost function in PID- parameter space and iteratively choose a set of parameters to minimize the cost function. I found out that the controller designed using the Automatic PID tuning is performing much better than the one using manually tuned parameter even in case of presence of random disturbances.

References

1. Leishman, J.G. (2000). Principles of Helicopter Aerosynamics. New Yorl, NY: Cambridge University Press.

2. www.asctec.de

3. http://www.dji.com/product/phantom-2-vision-plus

4. Pounds, P.;Mahony, R,; Corke, P. (December 2006). "In the Proceedings of the Australasian Conference on Robotics and Automation". Auckland, New Zealand.

5. Hoffman, G,; Haung, H.; Waslander, S.L.; Tomlin, C.J. (20-23 August 2007). "In the Conference of the Anerican Institute of Aeronautics and Astronautics". Head Hilton, South California.

6. "Aeryon Labs Inc."

7. Illumin- The Quadcoptor's Coming of Age"

8. "Paparazzi Agency - We've Used Drones For A Long Time". www.tmz.com

9. Model Predictive quadrotor control: attitude, altitude and position experimental studies by K. Alexis, G. Nikolakopoulos and A. Tzes

10. Design and Control of Quadrotors with application to Autonomous Flying by Samir Bouabdallah.

11. "Full control of a Quadrotor" by Bouabdallah, S, Siegwart, R.

12. "Design and control of quadrotors with application to autonomous flying" by Bouabdallah,

19. http://ardrone2.parrot.com/

13. http://www.scholarpedia.org/article/Siegel_disks/Linearization - The Linearization problem in complex dimension one dynamical systems

14. CSAS_flightmechanics1 lecture slides in CVUT by Martin Xhromcik- department of control Engineering.

15. G.A. Leonov, N.V. Kuznetsov, Time-Varying Linearization and the Perron effects, International Journal of Bifurcation and Chaos, Vol. 17, No. 4, 2007

16. http://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

17. http://www.ece.rutgers.edu/~gajic/psfiles/chap5traCO.pdf

18. https://en.wikibooks.org/wiki/Control_Systems/Controllability_and_Observability

19. Understanding Poles and Zeros by Massachusetts Institute of Technology, Department of Mechanical Engineering

20. https://en.wikipedia.org/wiki/Marginal_stability

21. https://en.wikibooks.org/wiki/Control_Systems/Bode_Plots

22. http://www.mathworks.com/matlabcentral/answers/82408-the-advantages-of-matlab-over-other-programing-languges-for-image-processing

23. Simulink Getting Started Tutorial by Mathworks

24. http://en.wikipedia.org/wiki/PID_controller

25. http://en.wikipedia.org/wiki/Single-input_single-output_system

26. "Improving Disturbance Rejection of PID controllers by Means of the Magnitude Optimum Method", D. Vrancic, S. Strmcnik, J. Kocijan and P. B. M. Oliveira

27. "On stabilizing PI Controller Ranges for Multi- variable Systems" pp. 620- 625 by, C. Lin, Q.G. Wang, Y. He, G. Wen, X. Han and G. Z. H. Z. Li.

28. https://en.wikipedia.org/wiki/PID_controller

29. https://courses.engr.illinois.edu/ece486/documents/lecture_notes/pid.pdf

30.http://www.electrical4u.com/types-of-controllers-proportional-integral-derivative-controllers/

31. https://oscarliang.com/quadcopter-pid-explained-tuning/