Modelling, Simulation, and Implementation of PID Controller on Quadrotors

Abhishek Iyer
Department of Electrical and Electronics Engineering
BITS Pilani, Pilani Campus
Jhunjhunu, India
f20181105@pilani.bits-pilani.ac.in

Dr. Hari Om Bansal
Department of Electrical and Electronics Engineering
BITS Pilani, Pilani Campus
Jhunjhunu, India
hobansal@gmail.com

Abstract— **A quadrotor is a type of Unmanned Aerial Vehicle (UAV) that is popular for its vast applications due to its small size and maneuverability. UAVs fall under the category of drones. Drones are unmanned aircrafts or ships guided by remote control or on-board computers. They are high in demand because of their deployment for surveillance, product delivery, aerial photography and defence applications. Implementing a control system on a quadrotor can be very challenging due to its nonlinear functioning. Thus the motivation of this paper is to successfully implement the PID algorithm on the plant to achieve altitude and attitude control. This paper also compares a linearized plant system of the quadrotor with that of a nonlinear plant system.**

A mathematical model for the quadrotor, progressing from a uni dimensional to a three dimensional configuration has been derived. Newton-Euler equations have been used to develop the dynamic model. The simulation are carried out in MATLAB/SIMULINK environment and different PID controller tuning methods have been looked into for smoother control

Keywords- ***Modelling, Matlab Simulink, PID Controller, Quadrotors, Simulation***

I. INTRODUCTION

Quadrotors are a class of Unmanned Aerial Vehicles (UAVs) which are small and offer great manoeuvrability. They are also classified as Vertical Take-Off and Landing (VTOL) vehicles. Quadrotors are built for several different purposes encompassing carrying payloads and deliveries, surveillance, recreational purposes, photography, videography, and for defence applications to name a few. While the structure of the quadrotor changes according to its purpose, the general structure remains the same. A quadrotor, aptly suggested by its name, has 4 motors which are equidistant from its centre, which is where the centre of mass lies. They make use of the 4 motors for take-off, landing and other kinds of manoeuvring in the air.

As estimated in a study by Goldman Sachs, the drone market is estimated to be worth $100 billion by the end of 2020, with the defence sector being responsible for $70 billion, consumer market being worth $17 billion and the commercial and civil market being worth $13 billion. All major companies are trying to automate their processes with robotics for enhanced efficiency and accuracy. Quadrotors, being reliable, are being used in numerous sectors. Further details on the application and scope of the industry can be found in [1]. Below is a flow chart diagram explaining the approach taken to implement the proportional integral and derivative (PID) controller to successfully to achieve attitude and altitude control on the quadrotor.

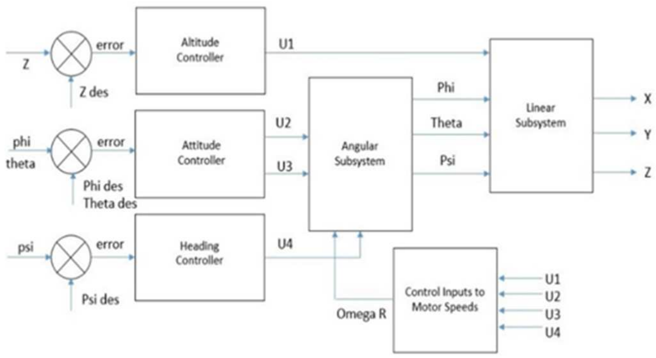

Fig. 1. Control System overview flowchart.

Relevant research papers have been referred to, for primary information along with other articles, videos and courses which have been referenced [2] [3] [4].

II. SCOPE OF THIS PAPER

The goal of this paper is to utilize existing work on the mechanics and dynamics of quadrotors [5] [6] to successfully model and simulate a quadrotor in Simulink Matlab [7]. We will be ignoring some of the more complicated effects while modelling the quadrotor. These can be found in [8] Furthermore, this paper will begin with modelling and simulating a quadrotor in 1-dimension and build to a 3-dimensional simulation to allow for proper understanding on which the PID [9] algorithm will be implemented to achieve attitude and altitude control in each case. Gain tuning will be discussed briefly for PD and PID models in each case. Furthermore, few graphs will be shown to highlight the difference between a nonlinear plant and a linearized one. Finally, desired outputs versus actual outputs will be looked at and the degree of success of the modelling and simulation will be determined.

978-1-7281-5875-4/21/$31.00 ©2021 IEEE

III. 1-D Modelling and Simulation of Quadcopter

A. Mechanics and Quadrotor Dynamics

For one dimensional analysis of the drone, we shall investigate the altitude control such that the quadrotor is only allowed to move in the Z direction. Furthermore, the drone will be considered symmetrical around the centre of mass and the X and Y axes. The equation of motion along the Z axis for a quadrotor can be derived from the figure below.

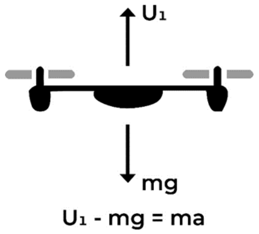

Fig. 2. 1 Dimensional quadrotor force body diagram

$$z'' = \frac{U_1}{m} - g \qquad (1)$$

Here U_1 is the thrust generated by the drone motors. Mg is the force applied by gravity and Ma is the net acceleration caused. z'' is the net acceleration in z direction.

B. Simulink Implementation

To successfully simulate a system, there are two parts one must take care of, the plant system, and its control. The plant describes the behaviour of the quadrotor and will be replicated by using equation (1), which will give the Z value for calculating the error and feeding it to the PID controller. The controller will be tuned to minimize error between the desired and actual Z value.

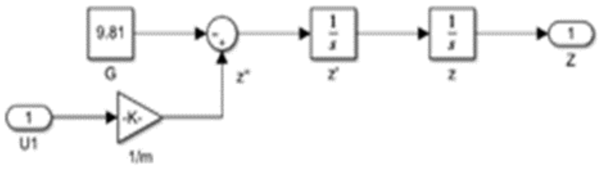

Fig. 3. Implementation of (1)

With the PID controller implemented, the overall system looks like this.

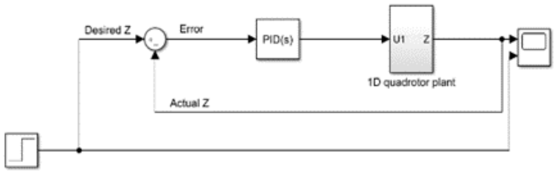

Fig. 4. 1D PID implementation on Quadrotor plant

C. Gains and PID

Equation (1) will be used as the plant equation and a control system will be developed around it to get the desirable output.

PID algorithm or the PID controller is a controller that uses the sum of the error multiplied by the coefficients Kp, Ki and Kd to elicit a response well within the design parameters of the system. MATLAB Simulink provides a readymade PID block, but one could easily make a block as per the applications.

We will use the provided block and after tweaking the gain coefficients a bit, the following results were the ones that lead to an acceptable response.

D. System Response using PD and PID controllers

TABLE I. PID Gains of system above

Constants	Gains
K_p	50
K_i	25
K_d	10

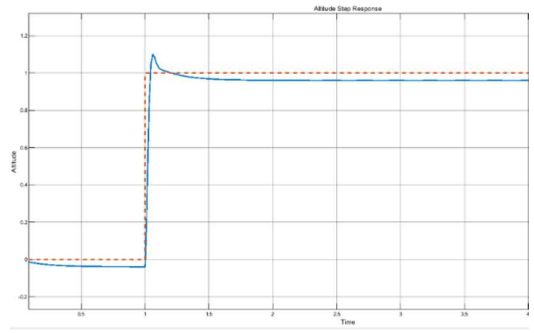

Fig. 5. Step response with PD implementation

The system response using PD and PID controllers is shown in figure 6 and 7 respectively. While the PD controller leaves a steady state error, a PID controller causes the peak overshoot to be higher and a decrease in the rise time. To compensate for this, often the PID controller has a lower P gain than a PD controller.

Compared to the PD graph above, the PID graph barely has any steady state error and a slightly higher peak overshoot which can be combated by using a lower P gain or a reduced D gain.

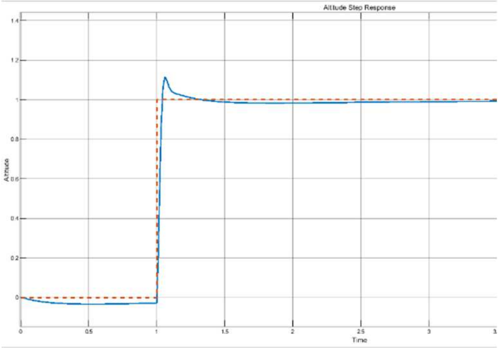

Fig. 6. Step response with PID implementation

IV. 2-D Modelling and Simulation of Quadrotor

A. Mechanics and Quadrotor Dynamics

In this section, the mechanics of a quadrotor fixed in the ZY plane will be explored. All our previous discussions about the symmetry and behaviour still hold. Although now, the quadrotor can move freely in the ZY plane which leaves us with three variables z, y and ϕ (Roll Angle).

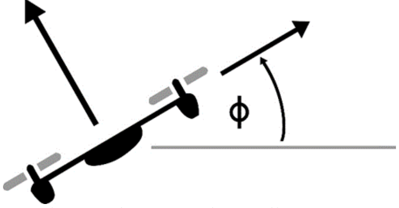

Fig. 7. Planar Quadrotor diagram

From the free body diagram in fig 7, one can derive the equations for y'', z'' and ϕ'' in which U_1 is the thrust input and U_2 is the moments input.

$$y'' = \frac{-U_1}{m}\sin(\phi) \tag{3}$$

$$z'' = -g + \frac{U_1}{m}\cos(\phi) \tag{4}$$

$$\phi'' = \frac{U_2}{I_{xx}} \tag{5}$$

$$\begin{pmatrix} y'' \\ z'' \\ \phi'' \end{pmatrix} = \begin{pmatrix} 0 \\ -g \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{-\sin(\phi)}{m} & 0 \\ \frac{\cos(\phi)}{m} & 0 \\ 0 & \frac{1}{I_{xx}} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} \tag{6}$$

Extending (3), (4) and (5) :

$$\frac{d^2y}{dt^2} = y'' = \frac{-U_1}{m}\sin\phi \tag{7}$$

$$\frac{d^2z}{dt^2} = z'' = -g + \frac{U_1}{m}\cos\phi \tag{8}$$

$$\frac{d^2\phi}{dt^2} = \phi'' = \frac{U_2}{I_{xx}} \tag{9}$$

B. Linearization

In the situation of the quadrotor hovering, the angle ϕ changes nominally. Hence one can linearize the above equations by applying the small angle approximations:

$$\frac{d^2y}{dt^2} = y'' = -g\phi \tag{10}$$

$$\frac{d^2z}{dt^2} = z'' = -g + \frac{U_1}{m} \tag{11}$$

Equation (9) is already in the linearized form.

C. Simulink Implementation

Now we shall compare with the linear model and will provide an insight into how well the approximation holds. For the linear plant system, (9), (10), and (11) will be implemented and for the nonlinear plant system, equations (7), (8), and (9) will be implemented. Below is the nonlinear implementation.

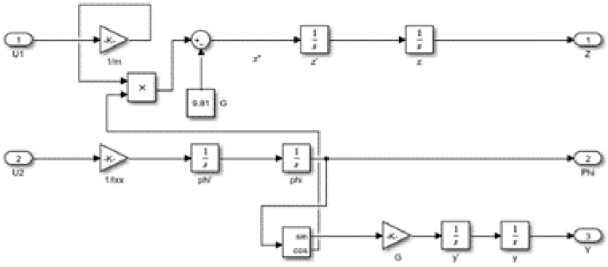

Fig. 8. Implementation of nonlinear plant system

Finally, after implementing the PID controller on the plant system we can observe the step responses.

D. Gains and PID

In the 2D system, there are 3 PID blocks to control the error of all the 3 variables that are being worked with – z, y and ϕ. The aim will be to control z and y and let ϕ vary independently to provide the desired y output since y and ϕ are dependent on each other. Another important thing to take care of in nested PID loops is that the internal loop should always have higher gains than the external ones to provide effective responsiveness. Below are the gains for the PID loops.

TABLE II. PID GAINS OF SYSTEM ABOVE

	ϕ	y	z
K_p	100	45	40
K_i	0	20	30
K_d	100	10	15

E. System Response of Linear and Nonlinear Systems

For the simulation described above, the system responses are shown. Below are the system responses for the altitude control of the linear system.

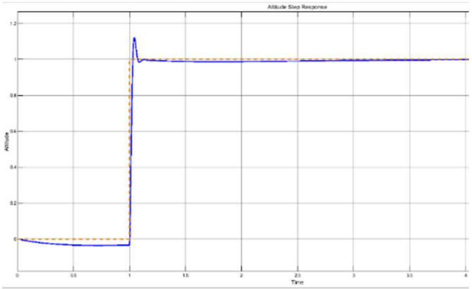

Fig. 9. Step response of z in linear system

We can compare this approximation to the actual nonlinear implementation. Below is the system response for altitude control for the nonlinear implementation of the plant. As we can see the settling time is higher and the responsiveness of the system seems a bit more delayed.

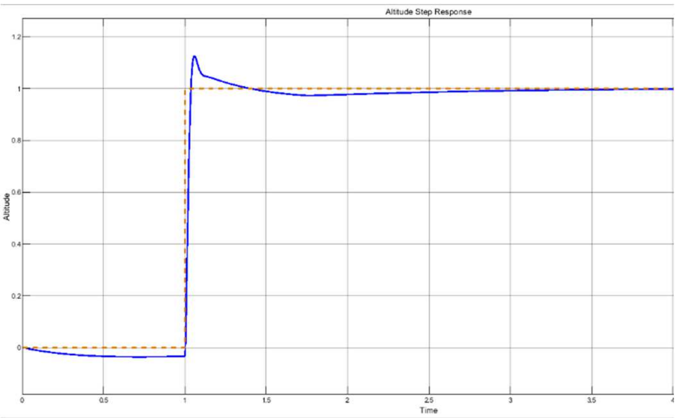

Fig. 10. Step response of z in nonlinear system

The step response of y is almost identical in both the nonlinear and linear implementation and for the sake of brevity is not included here.

V. 3-D Modelling and Simulation of Quadrotor

A. Mechanics and Quadrotor Dynamics

In this section, the mechanics of how a quadrotor functions in the 3-dimensional space is explored. The position of the quadrotor can be defined by [x y z] and the orientation of the quadrotor can be described by [ϕ θ ψ] which constitute the roll, pitch and yaw and represent rotations about the X, Y and Z axes, respectively. The distance from the body frame to the inertial frame will be represented by r = $[x\ y\ z]^T$. If one describes a movement with roll first, followed by pitch and finally yaw then the multiplication of each of the individual matrices will lead to the following matrix.

$$R = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \quad (12)$$

c is the shorthand for cosine and s is the shorthand for sine here. The rotational equation of motion can be written as follows

$$M_b = J\omega' + J\omega + Mg \quad (13)$$

Where:

J = Inertial Matrix which because of the symmetry around the centre is

$$\begin{pmatrix} Ixx & 0 & 0 \\ 0 & Iyy & 0 \\ 0 & 0 & Izz \end{pmatrix}$$

ω = Angular Rates of Velocity

Mg = $[0\ 0\ J r\Omega r]^T$ = Gyroscopic moments (14)

Ω_r = Relative motor speed.

M_b = Moments acting on the body frame

Now one can look at the moments acting on the quadrotor (M_b)

$$F_i = \frac{1}{2}\rho A C_t R^2 {\Omega_i}^2 \quad (15)$$

$$M_i = \frac{1}{2}\rho A C_d R^2 {\Omega_i}^2 \quad (16)$$

Where:

ρ = Air Density

A = Area swept by the blades

C_t, C_d = Coefficients

R = Radius of the blades

Ω_i = Angular Velocity of motor

Equations (15) and (16) can be written as

$$F_i = K_f {\Omega_i}^2 \quad (17)$$

$$M_i = K_m {\Omega_i}^2 \quad (18)$$

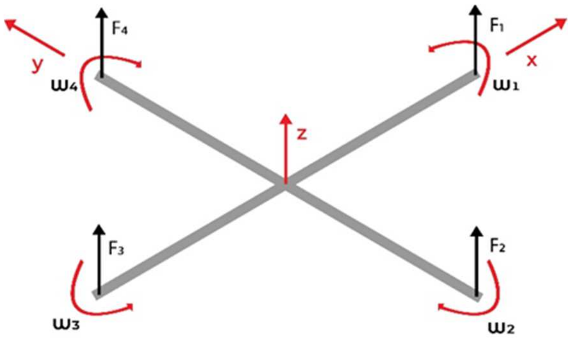

Fig. 11. Forces and Moments acting on the Quadrotor

Now calculating the moment about the X,Y and Z axes from Fig.13

$$M_x = L(F_4 - F_2) = LK_f({\Omega_4}^2 - {\Omega_2}^2) \quad (19)$$

$$M_y = L(F_1 - F_3) = LK_f({\Omega_1}^2 - {\Omega_3}^2) \quad (20)$$

$$M_z = M_1 - M_2 + M_3 - M_4 = K_m({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \quad (21)$$

Combining (19), (20), and (21)

$$M_b = \begin{pmatrix} LK_f({\Omega_4}^2 - {\Omega_2}^2) \\ LK_f({\Omega_1}^2 - {\Omega_3}^2) \\ K_m({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \end{pmatrix} \quad (22)$$

The translational motion of equation take the general form:

$$mr'' = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_b \quad (23)$$

F_b represents all the non-gravitational forces acting in the body frame and is given by

$$F_b = \begin{pmatrix} 0 \\ 0 \\ -K_f(\Omega_1{}^2 + \Omega_2{}^2 + \Omega_3{}^2 + \Omega_4{}^2) \end{pmatrix} \tag{24}$$

$$F_{bz} = F_1 + F_2 + F_3 + F_4 \tag{25}$$

which converts to the equation above. Positive z direction is assumed in direction of gravity.

B. State Space Representation

Now that the forces and moments acting on the body of the drone have been discussed, one can convert these equations into state space form which will be implemented to simulate the behaviour. State vector X can be represented as $[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}]^T$ which can be mapped to $X = [\phi\ \phi'\ \theta\ \theta'\ \psi\ \psi'\ x\ x'\ y\ y'\ z\ z']^T$

The control input U = [U1 U2 U3 U4] can be written as shown below

$$U_1 = K_f(\Omega_1{}^2 + \Omega_2{}^2 + \Omega_3{}^2 + \Omega_4{}^2) \tag{26}$$

$$U_2 = K_f(\Omega_4{}^2 - \Omega_2{}^2) \tag{27}$$

$$U_3 = K_f(\Omega_1{}^2 - \Omega_3{}^2) \tag{28}$$

$$U_4 = K_m(\Omega_1{}^2 - \Omega_2{}^2 + \Omega_3{}^2 - \Omega_4{}^2) \tag{29}$$

From equations (26) to (29), a coefficient matrix can be formed

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f \\ K_f & 0 & -K_f & 0 \\ K_m & -K_m & K_m & -K_m \end{pmatrix} \begin{pmatrix} \Omega_1{}^2 \\ \Omega_2{}^2 \\ \Omega_3{}^2 \\ \Omega_4{}^2 \end{pmatrix} \tag{30}$$

Equation (30) can be rearranged so that the coefficient matrix for motor speeds can be found from control vector U.

$$\begin{pmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{pmatrix} = \begin{pmatrix} 1/4K_f & 0 & 1/2K_f & 1/4K_m \\ 1/4K_f & -1/2K_f & 0 & -1/4K_m \\ 1/4K_f & 0 & -1/2K_f & 1/4K_m \\ 1/4K_f & 1/2K_f & 0 & -1/4K_m \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} \tag{31}$$

Expanding and solving (13),(14), and (22)

$$\phi'' = \frac{1}{I_{xx}}(LU_2 - J_r x_4 \Omega_r + (I_{yy} - I_{zz}) x_4 x_6) \tag{32}$$

$$\theta'' = \frac{1}{I_{yy}}(LU_3 - J_r x_2 \Omega_r + (I_{zz} - I_{xx}) x_2 x_6) \tag{33}$$

$$\psi'' = \frac{1}{I_{zz}}(U_4 + (I_{xx} - I_{yy}) x_2 x_4) \tag{34}$$

These provide the rotational equations. The translational equations can be derived from expanding (23) and using (12),

$$= \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -U_1 \end{pmatrix}$$

Expanding and rewriting,

$$x'' = \frac{-U_1}{m}(cos\phi sin\theta cos\psi + sin\phi sin\psi) \tag{36}$$

$$y'' = \frac{-U_1}{m}(cos\phi sin\theta sin\psi + sin\theta sin\psi) \tag{37}$$

$$z'' = g - \frac{U_1}{m}(cos\phi cos\theta) \tag{38}$$

This can be rewritten in state space form as,

$$x'' = \frac{-U_1}{m}(cosx_1 sinx_3 cosx_5 + sinx_1 sinx_5) \tag{39}$$

$$y'' = \frac{-U_1}{m}(cosx_1 sinx_3 sinx_5 + sinx_3 sinx_5) \tag{40}$$

$$z'' = g - \frac{U_1}{m}(cosx_1 cosx_3) \tag{41}$$

C. Simulink Implementation

The parameters used for the simulation are listed below.

Parameter	Value	Unit
L	0.24	m
M	1	kg
g	9.81	m/s^2
I_{xx}	0.01	kg m^2
I_{yy}	0.01	kg m^2
I_{zz}	0.02	kg m^2
K_f	1.87 x 10^{-6}	-
K_m	1.87 x 10^{-6}	-

Equations (32), (33), and (34) will be implemented to achieve the roll pitch and yaw accelerations which in turn will provide the roll, pitch and yaw angles.

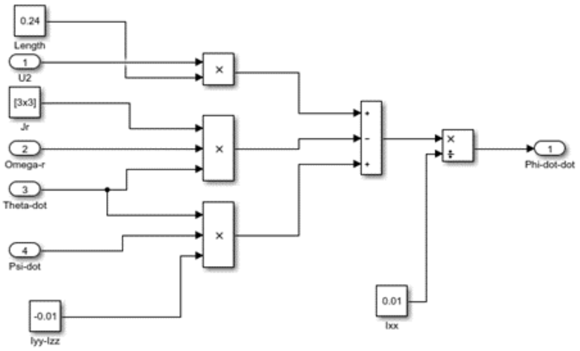

Fig. 12. Implementation to get roll acceleration, ϕ’’.

Similarly one can implement the other equations for angular and linear accelerations. Equations (36), (37), and (38) will be implemented to give us the outputs x, y and z accelerations which can be integrated twice to provide the actual positions.

Note that the following are the nonlinear implementations and to implement a linearized linear accelerations subsystem, one can apply the small angle approximations described previously.

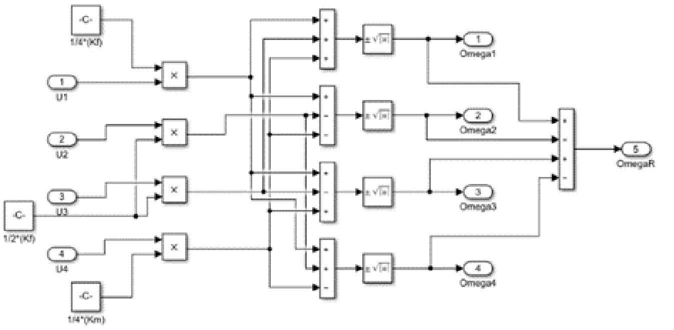

Fig. 14. Implementation to get relative motor speed, Ωr from control inputs, Ui.

Now one can connect these subsystems with each other so that the behaviour of the drone i.e., the plant system, is now entirely modelled. Once the plant system is ready, the PID control system will be implemented on the plant to achieve altitude and attitude control. The gains used are mentioned in the next section.

D. Gains and PID

To achieve altitude control a PID controller has been applied on U1 and to achieve attitude control 3 PID controllers have been applied on U2, U3 and U4 to get the desired ϕ, θ, and ψ values. The gains for all the PID controller loops have been shown below. These gains can be different based on what parameters are acceptable to the user. These controllers can be just PD controllers as well, which was demonstrated earlier in 2D simulation.

TABLE III. PID GAINS OF SYSTEM ABOVE

	z	ϕ	θ	ψ
K_p	800	110	16	8
K_i	300	50	10	10
K_d	25	3	10	2

E. System Response for Attitude and Altitude Control

This section contains the system responses from the implementation of the 3D plant and one can observe the efficiency of the implemented control system used to achieve altitude and attitude control of the quadrotor.

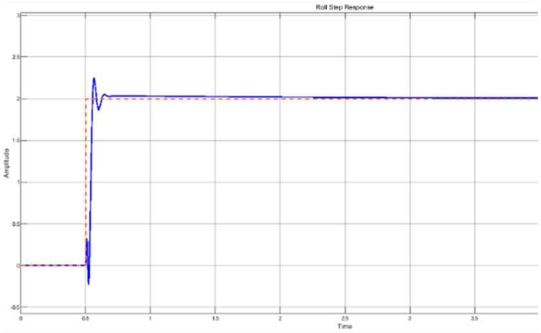

Fig. 15. Step response of roll angle, ϕ

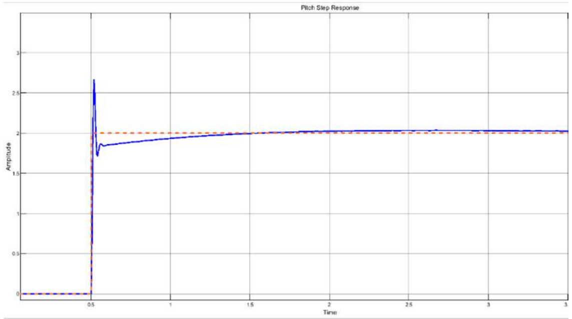

Fig. 16. Step response of pitch angle, θ

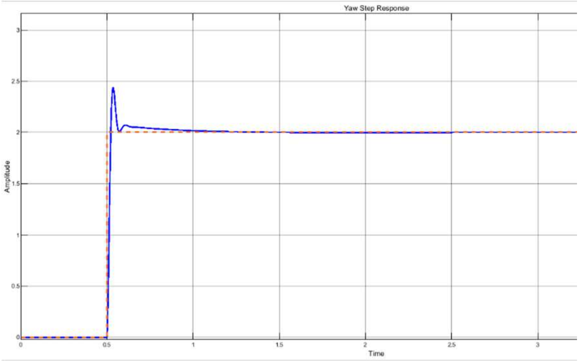

Fig. 17 Step response of yaw angle, ψ

The responses for altitude control are given in the figures 18 and 19 for linear and nonlinear implementation, respectively.

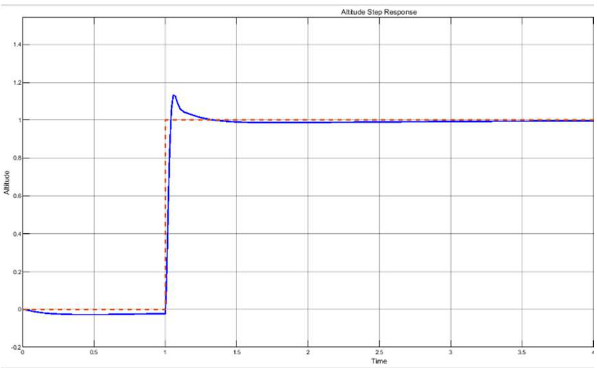

Fig. 18. Step response of Z in linear case

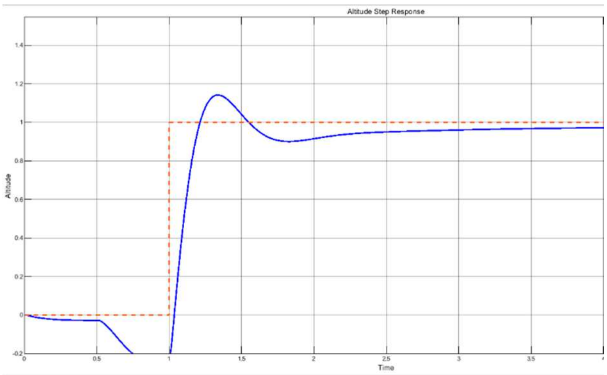

Fig. 19. Step response of Z in non-linear case

It is observed that attitude control implementations are identical in both the linear and nonlinear models. However, altitude control has a huge difference and so the small angle approximation is not one that is appropriate here. Further methods or techniques must be looked into to provide a smoother and more robust altitude controller for the 3D nonlinear plant system. The results from the simulation are tabulated below.

TABLE IV. RESULTS OF SIMULATIONS

Attribute	Settling Time (ms)	Rise Time (ms)	Peak Overshoot (%)
Nonlinear Altitude	413.0	179.7	14.2
Linear Altitude	214.9	35.8	13.0
Roll	195.7	52.5	12.1
Pitch	837.2	9.3	30.2
Yaw	499.5	18.8	21.7

VI. CONCLUSION AND FUTURE RESEARCH

As seen, one can quite accurately simulate the behavior of a quadrotor discounting a few effects that would unnecessarily complicate the physics. While the simulations are fairly accurate and the PID algorithm is good enough to elicit the desired response from the drone, this only provides us with a starting point while building and testing a drone. Furthermore, the nonlinear altitude control needs to be improved. It would require a lot of further tweaking of the gains as the physics that has been used for the simulation involve approximations. Upcoming research will be based on the different optimization tools to precisely predict the gain values such as reinforced learning, genetic algorithms, and swarm optimization etc. Lastly, motion planning and environment tracking remain areas that need to be explored further for nuanced control of the drone in its surrounding environment.

REFERENCES

[1] L. Camilli, “Emerging technologies, applications, regulations, and market challenges in the consumer aerial drone industry,” Technical Report, San Francisco State University College of Business, 2015.

[2] O. M. Hamdy, and A. O. El-Nady, “Modeling, Simulation and Control of Quadcopter using PID controller,” 4th International under-Graduate Research Conference, IUGRC 2019, Cairo, 2019.

[3] A. T. Bayisa, “Controlling Quadcopter Altitude using PID-Control System,” in International Journal of Engineering Research & Technology (IJERT), Vol 8, pp. 195-199, 2019.

[4] Q. Huang, “Mathematical Modeling of Quadcopter Dynamics,” Report, Rose Human Institute of Technology, 30 May 2016.

[5] “MathWorks,” [Online]. Available: www.mathworks.com [Accessed May 2020]

[6] University of Pennsylvania, “Robotics: Aerial Robotics,” Coursera, [Online]. Available: https://www.coursera.org/learn/robotics-flight?specialization=robotics. [Accessed June 2020].

[7] V. Desh, “MATLAB & Simulink Tutorial: Quadrotor UAV Trajectory and Control Design (PID + Cascaded),” YouTube, [Online]. Available: https://www.youtube.com/channel/UCRaiKJHsh_Ii4H-86uqXT_g. [Accessed July 2020]

[8] T. Lukkonen, “Modelling and Control of quadcopter,” Thesis, Aalto University - School of Science, 22 August 2011.

[9] A. Zulu, and S. John “A Review of Control Algorithms for Autonomous Quadrotors,” Open Journal of Applied Sciences, pp. 547-556, 2014.

[10] J. Anand, "Automatic Traffic Control Technologies for Remote Monitoring of Unmanned Railway Gates," Journal of Electronics, pp. 30-37, 2020.

[11] E. Kamel, and K. Kamel, "PLC Batch Process Control Design and Implementation Fundamentals," Journal of Electronics, pp. 155-161, 2020.