Position Control of Planar Quadrotor

Gaurav Dalmia (18110054)
B.Tech, Mechanical Engineering
IIT Gandhinagar
Gandhinagar, India
gaurav.dalmia@iitgn.ac.in

Navneet Kaur (18110106)
B.Tech, Mechanical Engineering
IIT Gandhinagar
Gandhinagar, India
navneet.kaur@iitgn.ac.in

Srujan Pandya (18110111)
B.Tech, Mechanical Engineering
IIT Gandhinagar
Gandhinagar, India
srujan.pandya@iitgn.ac.in

***Abstract*—Quadrotor is a type of Unmanned Aerial Vehicles (UAVs) with four rotors. It is the most common micro UAV due to its mechanical simplicity, both in geometry and maneuverability. It is typically in the shape of a cross with a rotor at each end on the upper side, capable of flying forward and backward, hover, etc. Planning and controlling trajectories for an autonomous flight is the most sought-after function of a quadrotor. The paper intends to discuss the design of a controller for the position control and planar motion stabilisation of a quadrotor.**

The paper carries out a mathematical motion analysis of the quadrotor which has been used to identify the equilibrium points. Furthermore, the state-space model has been developed to carry out the open loop analysis of the system and develop the transfer function for stability and position analysis.

The paper provides a brief analysis of the controller design where we have used PD controllers to enhance the transient response under a well-defined threshold of overshoot and settling time.

The analysis has been carried out for both linearized and non linearized systems, using two different approaches - one with SIMULINK Model and the second with a MATLAB code. Manual tuning of the controllers is done to achieve the gain values that result in the optimised behaviour of the step response and the differences between the linear and non-linear models are observed. Both the models are stable, with different values of overshoot and rise time, and the linear model meeting the desired specifications. The linear models satisfies the objectives of this paper, while the same controller does not produce the desired results. The future work includes the extension of the model to non-linear dynamics that achieves the desired specifications.

***Index Terms*—UAVs, controller, stabilisation, linearized, transient**

I. Introduction

In the 1920s, the concept of a manned quadrotor was introduced. Due to its large size and mechanical constraints, the idea was not developed. However, as time passed, engineers tried to develop smaller and smaller unmanned quadrotors. Over the last 30 years, the number of micro Unmanned Aerial Vehicles (UAVs) have increased exponentially, and the field of aerial robotics is rapidly evolving. Quadrotors are the most exploited micro-UAV due to its advantages over fixed-wing and flapped-wing mechanisms, both in terms of maneuverability and simplicity in design. A quadrotor has four rotors, embedded in either in a cross-configuration or a plus-configuration at the upper ends, weighing $0.18\ kg$ and the length of each span is $0.086\ m$ for the analysis in this paper. The most important aspect of the quadrotor is controlling its position. The following analysis intends to develop a mathematical model to understand the MIMO system and perform the position analysis of the planar quadrotor, where the inputs are defined as the net thrust and the net moment provided by the two rotors. The $x-y$ coordinates and the roll angle ϕ are the defined outputs for the system.

The first principles are used to derive the non-linear model from the equations of motion (EOM) which has been used to determine the equilibrium points. The model has been presented in the form of a state space model and linearized to identify the transfer function. The open loop analysis of the transfer function has been carried out for step response in which the stability has been checked.

Being a MIMO model, strongly coupled and underactuated with non-linearized characteristics, it is likely to be unstable at many points. Therefore, the paper further discusses the design of a control system for the model by adding PD controllers at the inputs. The PI and PID controllers are ruled out because it would affect the damping inappropriately and the steady state error can be controlled by the position and velocity error constants. Moreover, since it is a theoretical analysis, PD controller has been used as opposed to a Lead compensator. Finally, the PD controllers are tuned for different values of the position and velocity constants to yield the desired output.

The linearized model for the controller meets the desired specifications while the non-linear model does not satisfy the desired values of overshoot and rise time; however, both the models are stable with the achieved values of gains that optimize the step responses. The non-linear model can be tuned to get a better result, which can be taken up as future work, extending from the results obtained as part of this paper. The assumptions considered in this paper are in accordance with ignoring aerodynamic effects for small roll angles which are reasonable for hover and slow forward flight.

II. Dynamics of the Planar quadrotor

Following are the assumptions to the dynamic model of the quadrotor:

A. Assumptions

1) The quadrotor system is a rigid body system.
2) There is no air drag (or other kind of viscous/drag forces) acting on the quadrotor.
3) The body frame is attached to the center of mass of the quadrotor

4) The analysis of the mathematical model is done with small angle approximation.

The global reference frame or the world frame is the inertial reference frame in which the body is situated. The body frame is attached to the center of mass of the quadrotor which is shown in Fig. 1 as $x-y$ frame.
The control system has to satisfy the hovering condition at the equilibrium point in the body frame, while moving from an initial point to a desired point in the inertial frame.

B. Equations of Motion (EOMs)

The equations of motion are derived from force and moment balance of the quadrotor in the body frame.
Considering motion only in the $x-y$ plane with no yaw and pitch motion, we can write the equations of motion to be:

$$m\ddot{x} = -(F_1 + F_2)\sin\phi \tag{1}$$
$$m\ddot{y} = -mg + (F_1 + F_2)cos\phi \tag{2}$$
$$J\ddot{\phi} = (F_1 - F_2)L \tag{3}$$

The inputs to the system are defined as:

$$u_1 = F_1 + F_2 \tag{4}$$
$$u_2 = (F_1 - F_2)L \tag{5}$$

Rewriting the equations of motion in terms of the inputs to the system, we have:

$$m\ddot{x} = -u_1\sin\phi \tag{6}$$
$$m\ddot{y} = -mg + u_1\cos\phi \tag{7}$$
$$J\ddot{\phi} = u_2 \tag{8}$$

The position of the quadrotor is determined by x-y coordinates and the angle ϕ.
There are three second order differential equations. Since an n^{th} order differential equation can be resolved into n first order differential equations, there will be a total of six state variables (two for each equation).
Let the state variables x_1, x_2 for equation (6), y_1, y_2 for equation (7) and ϕ_1, ϕ_2 for equation (8).
Hence,

$$x_1 = x; x_2 = \dot{x} \tag{9}$$
$$\implies \dot{x_2} = \ddot{x} = -\frac{u_1}{m}\sin\phi \tag{10}$$
$$\implies \begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0 & 0\\ -\frac{\sin\phi}{m} & 0\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{11}$$

Similarly, from equations (7) and (8), we get:

$$\begin{bmatrix}\dot{y_1}\\ \dot{y_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} + g\begin{bmatrix}0\\ -1\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{\cos\phi}{m} & 0\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{12}$$

$$\begin{bmatrix}\dot{\phi_1}\\ \dot{\phi_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & \frac{1}{J}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{13}$$

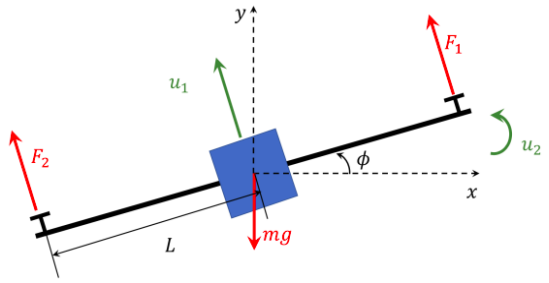

Fig. 1: Free-Body Diagram of Planar Quadrotor and Coordinate Axes attached with the Body Frame

Therefore, the state variables chosen for the system are x, y, ϕ, $\dot{x}$, $\dot{y}$, $\dot{\phi}$ (in the same order).
Thus, the state vector for the system can be defined as:

$$X = \begin{bmatrix}x_1\\ y_1\\ \phi_1\\ x_2\\ y_2\\ \phi_2\end{bmatrix} = \begin{bmatrix}x\\ y\\ \phi\\ \dot{x}\\ \dot{y}\\ \dot{\phi}\end{bmatrix} \implies \dot{X} = \begin{bmatrix}\dot{x_1}\\ \dot{y_1}\\ \dot{\phi_1}\\ \dot{x_2}\\ \dot{y_2}\\ \dot{\phi_2}\end{bmatrix} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \ddot{x}\\ \ddot{y}\\ \ddot{\phi}\end{bmatrix} \tag{14}$$

Hence, from equation (14), we have:

$$\begin{bmatrix}\dot{x_1}\\ \dot{y_1}\\ \dot{\phi_1}\\ \dot{x_2}\\ \dot{y_2}\\ \dot{\phi_2}\end{bmatrix} = \begin{bmatrix}x_2\\ y_2\\ \phi_2\\ 0\\ -g\\ 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{-\sin\phi}{m} & 0\\ \frac{\cos\phi}{m} & 0\\ 0 & \frac{1}{J}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{15}$$

Substituting $x_1 = x$ and $x_2 = \dot{x}$ and similarly for the other state variables, we get:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \ddot{x}\\ \ddot{y}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ 0\\ -g\\ 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{-\sin\phi}{m} & 0\\ \frac{\cos\phi}{m} & 0\\ 0 & \frac{1}{J}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{16}$$

The output equation is given by:

$$\begin{bmatrix}x\\ y\\ \phi\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ \phi\\ \dot{x}\\ \dot{y}\\ \dot{\phi}\end{bmatrix} \tag{17}$$

The dynamics are non-linear and the system is under-actuated i.e. the number of inputs are less than the number of outputs. Moreover, the system cannot be written in the form:

$$\dot{X} = AX + Bu$$

Where X is the state vector, A is the state matrix, B is the input matrix and u is the input to the system.
Therefore, the state-space model cannot be designed from the set of non-linear equations.
The gravity in y-direction is causing hindrance in the formation of the state space model, since it cannot be represented in terms of any state variable. Therefore, for our model, gravity is considered to be an external disturbance. The sinusoidal functions in the input matrix should also be linearized to further solve the differential equations.

C. Linearized EOMs and Equilibrium Points

For a small angle approximation ($\phi \longrightarrow 0$):

$$\cos\phi \approx 1 \text{ and } \sin\phi \approx \phi$$

Thus equations (6)-(8) become:

$$\ddot{x} = -\frac{u_1}{m}\phi \tag{18}$$

$$\ddot{y} = -g + \frac{u_1}{m} \tag{19}$$

$$\ddot{\phi} = \frac{u_2}{J} \tag{20}$$

Substituting zero acceleration values for equilibrium condition:

$$\ddot{x} = 0;\ \ddot{y} = 0;\ \ddot{\phi} = 0$$

Therefore, at equilibrium:

$$x, y, \phi = 0$$
$$u_1 = mg$$
$$u_2 = 0$$

Therefore, near the equilibrium point, $u_1 \approx mg$ and considering gravity as an external disturbance to the system (in y direction), we get the equations of motion as:

$$\ddot{x} = -g\phi \tag{21}$$

$$\ddot{y} = \frac{u_1}{m} \tag{22}$$

$$\ddot{\phi} = \frac{u_2}{J} \tag{23}$$

The state space equation of the above system can now be written as:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \ddot{x}\\ \ddot{y}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&-g&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}x\\ y\\ \phi\\ \dot{x}\\ \dot{y}\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0&0\\ 0&0\\ 0&0\\ 0&0\\ \frac{1}{m}&0\\ 0&\frac{1}{J}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{24}$$

The output equation remains the same. Thus we get the state matrix A and input matrix B to the system.
The system is a Multiple-Input-Multiple-Output (MIMO) system. Hence a Transfer Matrix will be formed with each entry being a transfer function that relates a particular output to the particular input, as shown below:

$$\begin{bmatrix}x\\ y\\ \phi\end{bmatrix} = \begin{bmatrix}g_{11}&g_{12}\\ g_{21}&g_{22}\\ g_{31}&g_{32}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{25}$$

Which can be written as:

$$X = GU \tag{26}$$

Here G represents the Transfer Matrix where g_{ij} represents the transfer function that relates the i_{th} output to j_{th} input. The linearized equations of motion show that only output y is dependent on the input u_1 and the outputs x and ϕ are dependent on u_2.
The Laplace Transforms of equations (21)-(23) yield:

$$s^2X(s) = -g\Phi(s) \tag{27}$$

$$s^2Y(s) = \frac{U_1(s)}{m} \tag{28}$$

$$s^2\Phi(s) = \frac{U_2(s)}{J} \tag{29}$$

Substituting $\Phi(s)$ in (27), we get:

$$\frac{X(s)}{U_2(s)} = \frac{-g}{Js^4} \quad (g_{12}) \tag{30}$$

$$\frac{Y(s)}{U_1(s)} = \frac{1}{ms^2} \quad (g_{21}) \tag{31}$$

$$\frac{\Phi(s)}{U_2(s)} = \frac{1}{Js^2} \quad (g_{32}) \tag{32}$$

Therefore, equation (26) can be written as:

$$\begin{bmatrix}x\\ y\\ \phi\end{bmatrix} = \begin{bmatrix}0&\frac{-g}{Js^4}\\ \frac{1}{ms^2}&0\\ 0&\frac{1}{Js^2}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{33}$$

The above equation summarises the dependence of the output variables on the inputs in the state-space representation of the given MIMO system.
The same results can be obtained directly using the MATLAB function for a MIMO system.

D. Solving the Differential Equations Using ODE Solver

MATLAB's ODE Solver is used to solve the equations of motion eqn. (6)-(8), all of which are linearized to ordinary differential equations. Since the motion in the horizontal direction is dependent on the roll angle, the values of the roll angle ϕ from the solution of equation (20) become the right-hand side of equation (18).
ODE Solver gives results to the equations that equivalently represent the open-loop configuration in state-space representation.
Therefore, the system is evidently shown to be unstable (and unbounded) when the output variables are plotted against time with the help of ODE Solver. The system is actuated step inputs u_1 and u_2. The plots for x, y and ϕ are shown in Fig. 2, Fig. 3 and Fig. 4 respectively.

Notice that the response goes to negative infinity in the y-direction indicating that there is no thrust to balance the quadrotor in the hovering condition. Thus, the quadrotor will fall to the ground if not controlled by giving feedback. Also, x goes to the negative infinity since the motion in the horizontal direction is provided opposite to the positive x-axis defined in the body frame (since $\ddot{x} = \frac{-u_1}{m} \sin\phi$).

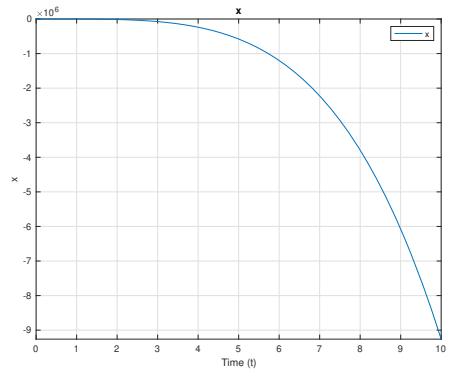

Fig. 2: $x(t)$

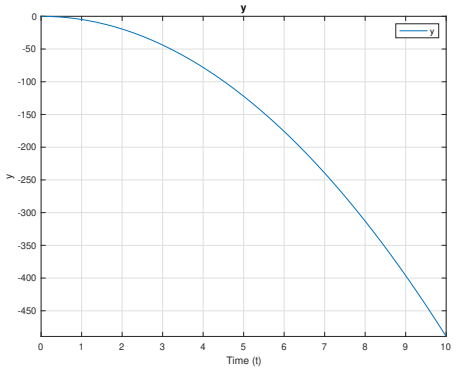

Fig. 3: $y(t)$

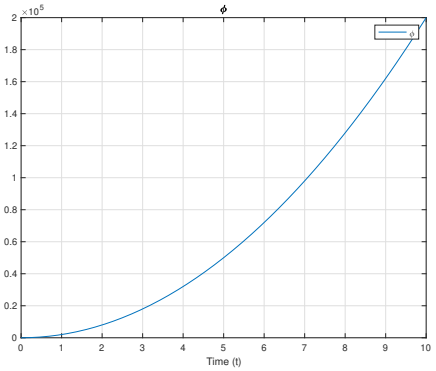

Fig. 4: $\phi(t)$

E. Open-Loop Analysis

Observing the open-loop transfer functions, we see that the poles of all the transfer function are located at $s = 0$ and none of the transfer functions have a zero.
The following table summarises the poles and zeros of the open-loop transfer functions:

Transfer Function	Poles	Zeros
$\frac{-g}{Js^4}$	$s = 0, 0, 0, 0$	–
$\frac{1}{ms^2}$	$s = 0, 0$	–
$\frac{1}{Js^2}$	$s = 0, 0$	–

TABLE I: Poles and Zeros of the Open-Loop Transfer Functions

The double poles at $s = 0$ (four poles for $X(s)/U_2(s)$) indicate that the system is unstable, and it involves (at least) two integrators between the input and the output signal in the open-loop. A total of 8 poles are present in the system, all at $s = 0$.
Also, none of the open-loop transfer functions have a zero. So the system will not have an undershoot or a faster response due to the absence of zeros.
The parameters m, g and J can only scale the magnitude of the plots. Hence, their values will not affect the stability of the open-loop system and the system will be unstable for all possible values of the parameters.
For the given system, the table below shows the parameters used:

Parameter	Symbol	Value
$Gravity$	g	$-9.8\ \frac{m}{s}$
$Mass$	m	$0.18\ kg$
$Moment\ of\ Inertia$	J	$0.00025\ \frac{kgm^2}{s}$
$Length\ of\ Arm$	L	$0.086\ m$

TABLE II: Parameters Used For the Given System

Using the parameters given in Table 2, the Transfer Matrix

becomes:

$$G = \begin{bmatrix} 0 & \frac{-39200}{s^4} \\ \frac{5.5556}{s^2} & 0 \\ 0 & \frac{4000}{s^2} \end{bmatrix} \tag{34}$$

Hence, from the Transfer Matrix, we can see that the parameters has no effect on the poles and zeros of the open-loop system. Hence, the poles and zeros of the system remain the same as shown in Table 1.

Transfer Function	Poles	Zeros
$\frac{-39200}{s^4}$	$s = 0, 0, 0, 0$	–
$\frac{5.556}{s^2}$	$s = 0, 0$	–
$\frac{4000}{s^2}$	$s = 0, 0$	–

TABLE III: Poles and Zeros of the Open-Loop Transfer Functions After Substituting Parameter Values

Table 3 shows the poles and zeros of the transfer function unaffected after entering the values of the parameters.

Fig. 5 shows the step response of the MIMO system.

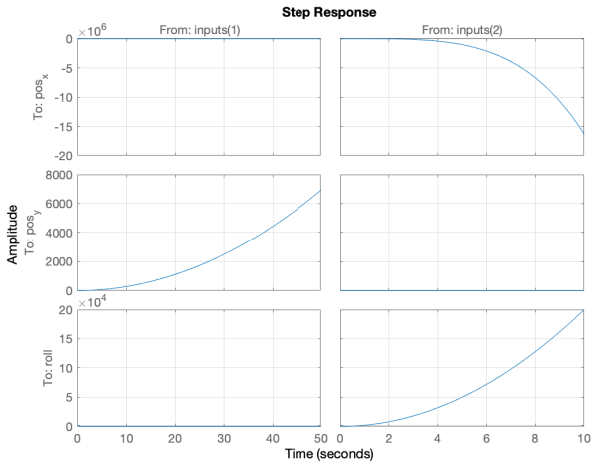

Fig. 5: Step Response of the Multiple-Input-Multiple Output System

It can be clearly seen that the plots represent the Transfer Matrix and the unstable nature of the step responses in the open-loop configuration.
The zero amplitude lines in the plots indicate the independence of the variable from the respective input.
The plots produced by the ODE Solver and the plots from the transfer function yield the same results, indicating that the linearized equations of motion are nothing but representative of the open-loop transfer functions.
In the ODE Solver, the disturbance due to gravity in the negative direction is considered while in the MIMO transfer function, the transfer function only includes the u_1/m part of the equation in y-direction.
However, both the plots indicate the unstable nature of the system in the open-loop configuration, without the stabilisation using a controller. Hence, this motivates the use of a controller which will reduce the error with a limited overshoot and a faster response (lower value of rise time) and stabilise the quadrotor.

III. Controller Design

Let the starting point of the quadrotor be (0,0) and the roll angle ϕ to be zero. At time $t = 0$, the quadrotor is taken to be in stable equilibrium and hence the initial values for $\ddot{x}$, $\ddot{y}$, $\ddot{\phi}$ $\dot{x}$, $\dot{y}$, $\dot{\phi}$ are all taken to be zero. Let the desired equilibrium orientation of the quadrotor be given by:

$$x_{des} = 1 \tag{35}$$
$$y_{des} = 1 \tag{36}$$
$$\phi_{des} = 0 \tag{37}$$

We now define the relative errors in position (e_p) and velocity (e_v) as:

$$e_p = r - r_{des} \tag{38}$$
$$e_v = \dot{r} - \dot{r}_{des} \tag{39}$$

Where r_{des} is the vector representing the desired position and r is the vector representing the actual position of the quadrotor in the inertial frame, as functions of time.
For hovering about the equilibrium point, the position and velocity error should exponentially go to zero in order to stabilise the planar motion. In other words, we want the motion of the quadrotor to follow the step input as closely as possible.

$$\ddot{r} - \ddot{r}_{des} + K_d(\dot{r} - \dot{r}_{des}) + K_p(r - r_{des}) = 0 \tag{40}$$

Where K_p and K_d are the derivative and proportional constants, or equivalently the position and velocity constants.
For the above conditions, a PD controller is the best suitable controller for the given system.
Increasing the proportionality constant reduces the steady-state error and the system response faster (rise time will reduce).
The derivative control increases the control loop performance and makes it more stable. It reduces the overshoot as well as the settling time, hence the response will be closed to the desired response. Therefore,

$$\ddot{x} - \ddot{x}_{des} + K_{dx}(\dot{x} - \dot{x}_{des}) + K_{px}(x - x_{des}) = 0 \tag{41}$$
$$\ddot{y} - \ddot{y}_{des} + K_{dy}(\dot{y} - \dot{y}_{des}) + K_{py}(y - y_{des}) = 0 \tag{42}$$
$$\ddot{\phi} - \ddot{\phi}_c + K_{d\phi}(\dot{\phi} - \dot{\phi}_c) + K_{p\phi}(\phi - \phi_c) = 0 \tag{43}$$

Here, ϕ_c denotes the commanding angle. The commanding angle can be thought of as the angle that is commanded to the quadrotor can be system

Substituting $\ddot{x} = -g\phi$ from (18) in (41))and rearranging the equation we get the commanding angle to be:

$$\phi_c = -\frac{1}{g}(\ddot{x}_{des} + K_{vx}(\dot{x}_{des} - \dot{x}) + K_{px}(x_{des} - x)) \quad (44)$$

Similarly, from (19) and (20) we write the inputs u_1 and u_2 as:

$$u_1 = m(g + \ddot{y}_{des} + K_{vy}(\dot{y}_{des} - \dot{y}) + K_{py}(y_{des} - y)) \quad (45)$$

$$u_2 = J(\ddot{\phi_c} + K_{v\phi}(\dot{\phi_c} - \dot{\phi}) + K_{p\phi}(\phi_c - \phi)) \quad (46)$$

The equations (44)-(46) are the control equations for the system. The position and velocity errors in the x-direction of the quadrotor will give the value of the commanding angle for each instant, and this commanding value will eventually give feedback to the system to accordingly modify the input moment u_2 through (46).
Similarly, the position and velocity errors in the y-direction gives feedback to the thrust input to modify it using (45).
First of all, we design a subsystem of the controller that represents the open-loop configuration using the Transfer Matrix given in (33).

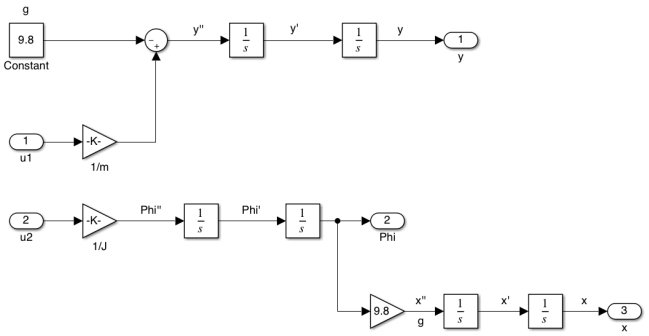

Fig. 6: Subsystem of the Controller

See the integrators in Fig. 6 as part of the subsystem, as discussed in the poles and zeros of the open-loop transfer functions. Also, gravity is added as an external disturbance in the y-direction, as discussed while linearizing and simplifying the equations of motion.
Now, we give feedback of the output signals to get the error signal, which will then enter the PD controller that controls the step response of the system to follow the desired hover equilibrium.
Fig. 7 shows the design of the controller for the linearized model.
The system will be become closed-loop. Step inputs are given as the desired coordinates (since $x_{des} = y_{des} = 1$). The system should control the step responses of the coordinates to reach the desired coordinates as soon as possible, while damping the oscillations of the roll angle ϕ.
A PD controller for each of the output variable x, y, ϕ has to be designed and the values of the proportionality and derivative constants have to be tuned such that the step responses behave as desired.

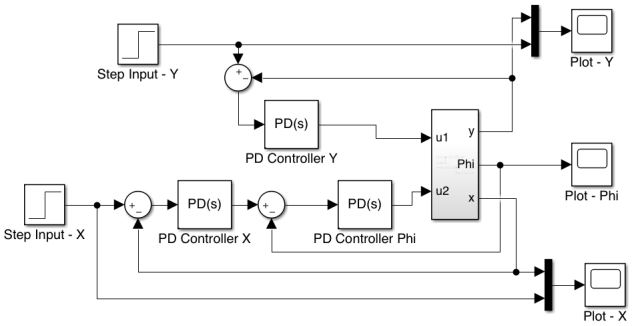

Fig. 7: Simulink Model of the Closed-Loop System with PD Controllers

IV. SIMULINK MODEL

A. Case 1: Linear Dynamics

Manual tuning of the PD controllers has been done for all the results using the relative effects on the overshoot and rise time. The controller has to be designed such that the overshoot is less than 16% (equivalently $\zeta > 0.05$) and rise time is less than $0.35s$. These conditions are satisfied and the controller is further tuned to the optimum solution. The plots comparing the step response of the outputs and the step input for x, y, ϕ are shown in Fig. 8, 9 and 10. The responses for x shows an almost zero steady state error, while the response for y shows some steady-state error. There is also an undershoot in the step response of y. This is due to the negative disturbance caused by gravity, which also leads to some finite but small error in y-direction.
The response of ϕ is zoomed in to see the oscillations being damped very quickly and the body oscillates with a minimal amplitude after 1.2 s

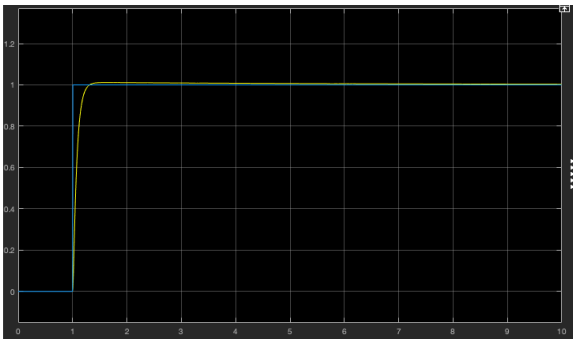

Fig. 8: Step Response of x(t) for the Linear Model

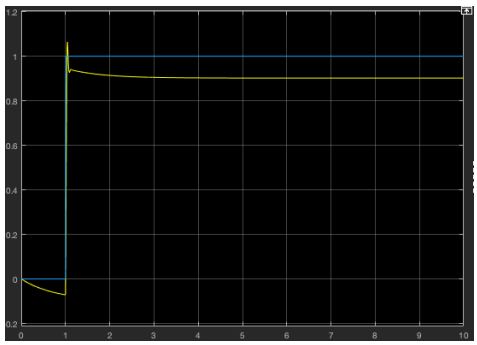

Fig. 9: Step Response of y(t) for the Linear Model

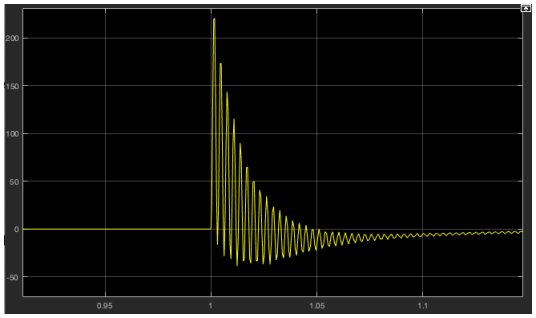

Fig. 10: Step Response of Roll Angle ϕ for Linear Model

Output	Rise Time	Overshoot
$x(t)$	$0.148\ s$	$0.505\ \%$
$y(t)$	$0.018\ s$	$6.5\ \%$

TABLE IV: Overshoot and Rise Time for Outputs of the Linear Model

Gain Type	X	Y	ϕ
K_p	0.2	18	30
K_d	1.3	14	10

TABLE V: Tuned Gain Values for the PD Controllers of Linear Model

B. Case 2: Non-Linear Model

Using the same gains tuned for the linear model, the control system for 'Non-Linear Dynamics' will have a different subsystem as compared to the one for 'Linear Dynamics'.
The subsystem modelled for this case is shown in Fig 11.

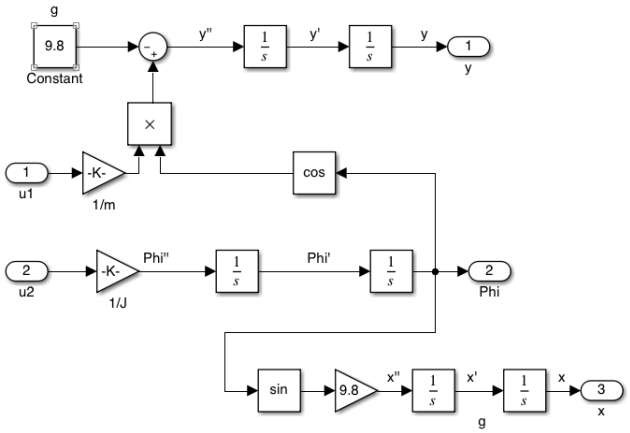

Fig. 11: Non-Linear Subsystem of the Controller

The controller design remains the same as in the linear case. The step responses of x, y and ϕ in the non-linear model are shown in Fig. 12, 13 and 14, respectively.

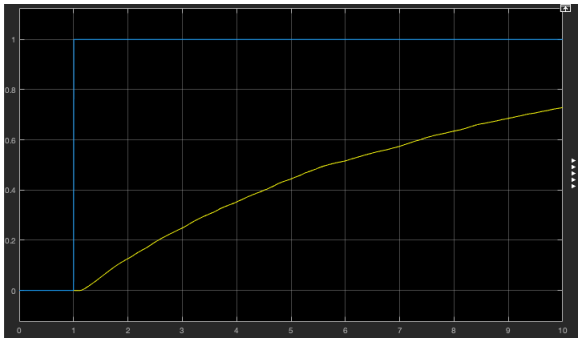

Fig. 12: Step Response of x(t) for Non-Linear Model

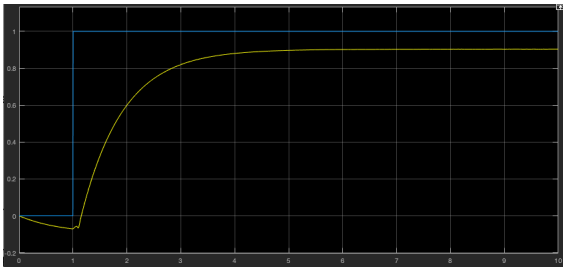

Fig. 13: Step Response of y(t) for Non-Linear Model

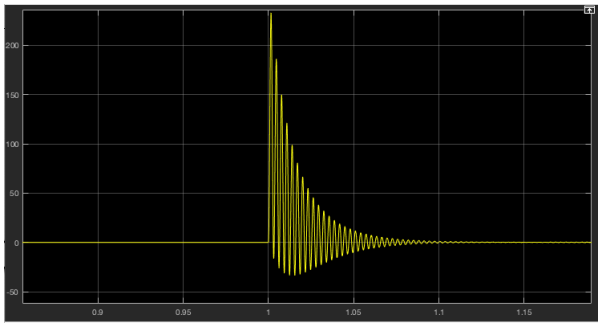

Fig. 14: Step Response of Roll Angle ϕ for Non-Linear Model

There are some differences that arise going from linear to non-linear dynamics. In case of x coordinate, the response becomes becomes sluggish (compared to the linear case) and it will take more time (more than 10 seconds) to reach the steady state.
The y coordinate response also becomes sluggish but has marginally more steady state error (almost equal) compared to the linear case.
The results of the non-linear model do not follow the objectives of the linear model, that is, they do not limit the overshoot below 16 % and neither do they limit the rise time to 0.35 s. However, none of the variables in the non-linear model have an unstable response, which indicates that further tuning of the controllers may result into desired responses and better results.

Output	Rise Time	Overshoot
$x(t)$	$0.148\ s$	$0.505\ \%$
$y(t)$	$0.018\ s$	$6.5\ \%$

TABLE VI: Overshoot and Rise Time for Outputs of the Non-Linear Model

V. Controller Code

In the MATLAB code written for the controller design, the results are obtained purely from the execution of the code and no modelling of the system is required. The code purely works on the governing equations of motion.
The 'Pseudo Code' is used to explain the MATLAB code used to plot the step response of the output variables.

A. Pseudo Code

Algorithm 1 Linear Case

Result: Rise time and Percentage Overshoot of x and y parameters along with the graphs of x, y, and ϕ

Constants g, m, J
Fix the values of duration and time step
$t_{max} \leftarrow 10$ ▷ Duration for which the code is to be run
$t_{step} \leftarrow 0.001$ ▷ Time step
Initialisation of Gains
$K_p \leftarrow [42, 32, 400]$
$K_v \leftarrow [10, 7, 58]$
For the initial position of (0,0) and the desired position of (1,1)
for *i = 2,3,...,n-1* **do**
  `// n is the length of the time list`
  Using the control equations, compute the values of ϕ_c, u_1, and u_2
  Using the linearized governing equations, compute the values of $\ddot{x}$, $\ddot{y}$, and $\ddot{\phi}$
    $x_{i+1} \leftarrow x_i + \dot{x} * t_{step} + 0.5 * \ddot{x} * (t^2_{step})$
    $\dot{x} \leftarrow \dot{x} + \ddot{x} * t_{step}$
    `// y and` ϕ `are updated in the same way`
  if $x_{i+1} > 0.1 * (x_{des} - x_{initial})$ *and Check1=True* **then**
    Rise time = Current time
  end
  else if $x_{i+1} > 0.9 * (x_{des} - x_{initial})$ *and Check2 = true* **then**
    Rise time = Current time - Rise time
  end
  `// Rise time of y is calculated in the same way`
end
$Overshoot_x \leftarrow (x_{max} - x_{des}) * 100$
$Overshoot_y \leftarrow (y_{max} - y_{des}) * 100$

Graphs are plotted for x, y and ϕ

For the Non-Linear case, the algorithm almost remains the same. We have to incorporate the following steps:

Algorithm 2 Non-Linear Case

Result: Rise time and Percentage Overshoot of x and y parameters along with the graphs of x, y, and ϕ

...
for *i = 2,3,...,n-1* **do**
  ...
  Using the non-linearized governing equations, compute the values of $\ddot{x}$, $\ddot{y}$, and $\ddot{\phi}$
  $\Delta\dot{x} = (x_i - x_{i-1})t_{step}$
  $\Delta\dot{y} = (y_i - y_{i-1})t_{step}$
  ...
end
...

B. Results and Discussions

The results obtained for both linear and non-linear models from the controller code are shown in Fig. 15-20.
The gain values used in both the model are shown in Table 5.

Gain Type	X	Y	ϕ
K_p	42	32	400
K_d	10	7	58

TABLE VII: Tuned Gain Values for the PD Controllers of Linear Model

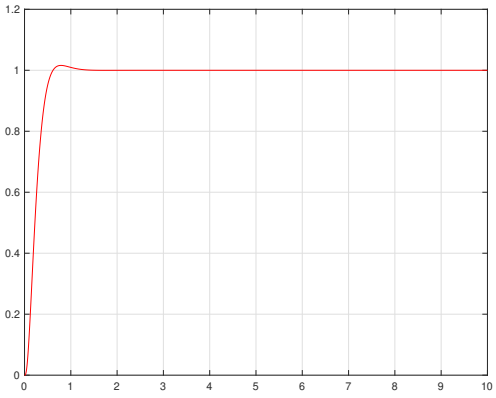

Fig. 15: Step Response of $x(t)$ for Linear Model

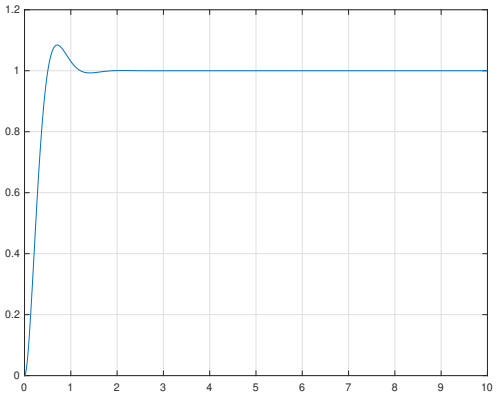

Fig. 16: Step Response of $y(t)$ for Linear Model

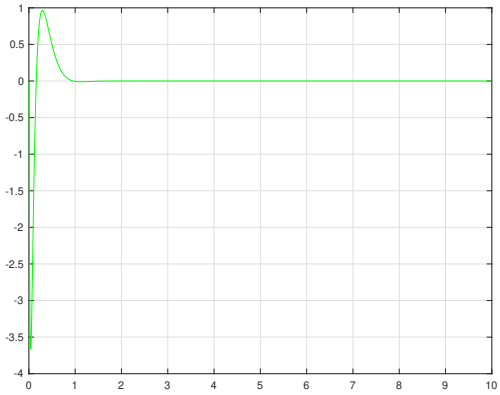

Fig. 17: Step Response of $\phi(t)$ for Linear Model

Table 6 shows the values of the overshoot and rise time for step responses of x and y in the Linear Model.

Output	Rise Time	Overshoot
$x(t)$	0.349 s	1.600 %
$y(t)$	0.336 s	8.441 %

TABLE VIII: Overshoot and Rise Time for Outputs of the Linear Model

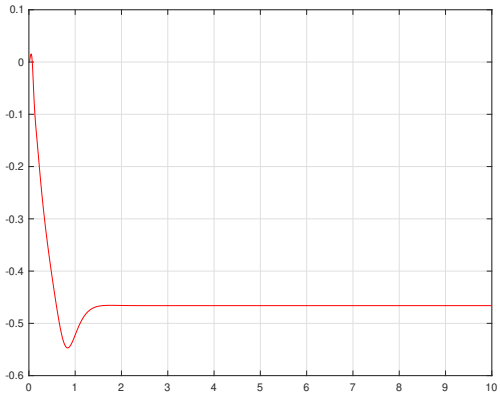

Fig. 18: Step Response of $x(t)$ for Non-Linear Model

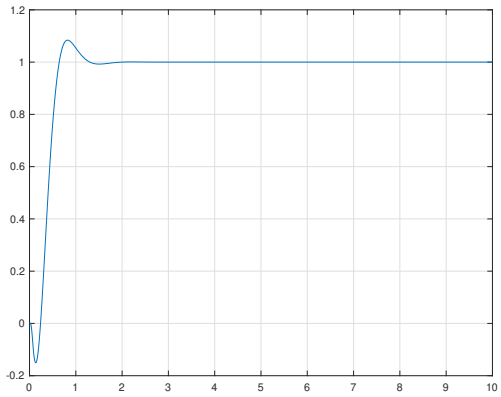

Fig. 19: Step Response of $y(t)$ for Non-Linear Model

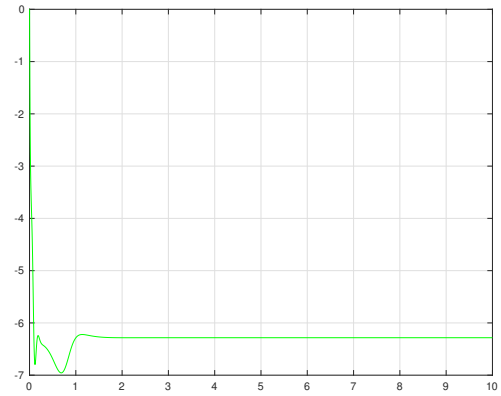

Fig. 20: Step Response of $\phi(t)$ for Non-Linear Model

The response from the controller code are much more neat and provides better insight than the results from the SIMULINK model; however, they also have a disadvantage.
The results from the controller code for the linear model show similar trends in overshoot as those in SIMULINK model, but the rise time for the step responses from the code are very close to the upper limit of $0.35\ s$.
The interesting results from the code show up in the responses of the non-linear model. The most important similarity with the SIMULINK model in the non-linear model is the stable nature of all the response after a certain period of time. Here, just like before, none of the responses show unstbale nature.
Also, in case of the non-linear results from the code, see tha the stabilisation of every variable is at a different value. This indicates the final position of the quadrotor does not align with the final desired positon (y stabilises at $y = 1$ while x stabilises at $x \approx -0.47$). Also, the angle ϕ is stabilising at a non-zero value. Physically, this means that the quadrotot reaches a different desired position with an orientation at some non-zero roll angle ϕ.
Notice that the gain values for angle ϕ in both the code and SIMULINK models are much greatere than those of x. This follows the principle that the inner loop dynamics (inner loop w.r.t controller design) have to be faster than the outer loop dynamics and hence the gain values of the attitude controller needs to be larger than the position controllers.
The differences in the results and the gain values of the SIMULINK model and the controller code are due to the different approaches taken in both the methods. The code works directly on the governing equations which includes numerical simulation and iterative analysis while the SIMULINK model includes transfer funtion analysis.

VI. Conclusion

This paper tries to achieve the desired specifications of overshoot and rise time for the step responses of a planar quadrotor via a controller design for linear as well as non-linear dynamics.
Two different approaches have been followed: one with the SIMULINK model and the other is purely a MATLAB code. The values of gains of the PD controllers are different for the two different approaches, however both the approaches design a stable controller for the linearized model, but fail to achiever the objectives for the non-linearized model.
On a broader basis, the paper tries to solve for the hovering stability of a quadrotor while tracking a desired trajectory. Although the model is linearized, the linearized model produces stable results even when extended to the non-linear model. The tuning of the gains is the most important and crucial part of designing the controller (including the fact that the inner loop dynamics have to be faster than the outer loop dynamics). Tuning the non-linear model may also result in achieving the desired results. Thus, the linearized model will work for sufficiently good range of motions, especially if we keep the roll angle in the small angle approximation range.
Overall, there are a lot of limitations to the above designed model such as avoiding air drag, small angle approximation, linearizing the equations, etc. However, the model produces satisfactory results which provides insight into the mathematical as well as physical understanding of the given problem. Extending the controller design for the non-linear model is the future direction in which the work can be extended.

References

[1] X. Zhang, X. Li, K. Wang, and Y. Lu, "A Survey of Modelling and Identification of Quadrotor Robot," Abstract and Applied Analysis, vol. 2014, p. e320526, Oct. 2014, doi: 10.1155/2014/320526.
[2] J. Li and Y. Li, "Dynamic analysis and PID control for a quadrotor," in 2011 IEEE International Conference on Mechatronics and Automation, Beijing, China, Aug. 2011, pp. 573–578. doi: 10.1109/ICMA.2011.5985724.

VII. Author's Contribution

Gaurav Dalmia: The working principles of the quadrotor, its history and motivation, set as a reference for the sections to follow (that delve into the mathematical models) has been formulated by me. In addition to that, the abstract and the introduction have been penned down by me which intends to provide the reader with the entire summary of the methods, results and conclusions. I contributed to the mathematical modelling of the controller design. Lastly, I worked on the PID Tuning of the controllers in the SIMULINK model for different values of K_p (proportional constant) and K_d (derivative constant). In addition to that, the concluding remarks were also penned down by me.

Navneet Kaur: My contribution to this paper has been focused on the controller code (explained as a pseudo code) for both the linear and nonlinear models, which is the second method . I performed an iterative analysis in the code for the governing kinematic equations of the quadrotor. The results and discussions obtained from the code and its comparison with the results of the SIMULINK model was also penned down by me.

Srujan Pandya: I worked on the equations of motion (EOM) and developed the space-state model after carrying out the linearization of the governing equations. Moreover, I worked with ODE Solver (code) to solve the EOM as part of the open-loop analysis. I also worked on the formulation of the Transfer Matrix for extracting the transfer functions of the MIMO system. Lastly, I modelled the controller design in SIMULINK for both the linear and nonlinear cases.