1 Method

1.1 Working principle

A quadcopter is a UAV with four propellers. It is a dynamic, under-actuated vehicle with six degrees of freedom (6DOF), four input forces, one for each propeller. A quad copter has four fixed-pitch and fixed-angle rotors instead of the variable pitch and fixed-angle rotors found in regular helicopters. The four individual propellers individual rotational speeds are changed to alter the lift and rotational forces, which controls the motion of a quadcopter in six degrees of freedom. The quadcopter rolls and pitches by tilting in the direction of the slowly rotating motor. Linear motion is accomplished by dividing the thrust into two directions because of the roll and pitch angles. To control the yaw caused by the drag force on the propellers, the rotors rotate in clockwise/anticlockwise pairs (Fig. 1).

1.2 Dynamic model of quadcopter

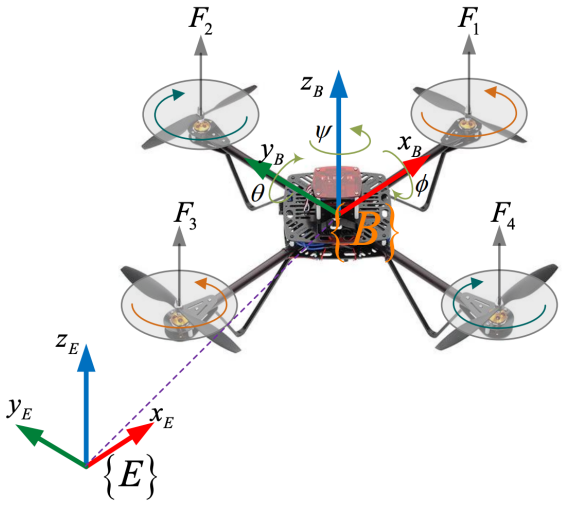

Figure 1 : Configuration of the Quadcopter.[2]

The Quadcopter dynamics are defined using two frame which are body frame and the world frame, all the states are with respect to the body frame. Here, using the Newton and Euler equations for the three-dimensional motion of a rigid body, The present a mathematical model of the quadrotor. This section's objectives are to gain a deeper understanding of the quadrotor's dynamics and to present a model that can be used to simulate and control its behavior [1].

Table 1: State variables of the Quadcopter

$State\ Variables$	Description
X	The position of the Quadcopter in the X direction in the body frame
Y	The position of the Quadcopter in the Y direction in the body frame
Z	The position of the Quadcopter in the Z direction in the body frame
$\dot{X}$	The Velocity of the Quadcopter in the X direction in the body frame
$\dot{Y}$	The Velocity of the Quadcopter in the Y direction in the body frame
$\dot{Z}$	The Velocity of the Quadcopter in the Z direction in the body frame
ϕ	Roll angle(Euler angle)
θ	Pitch angle(Euler angle)
ψ	Yaw angle(Euler angle)
$\dot{\phi}$(p)	Angular rate in the Roll direction
$\dot{\theta}$(q)	Angular rate in the Pitch direction
$\dot{\psi}$(r)	Angular rate in the Yaw direction

Table 2: Attributes of the dynamic system

$Properties$	Description
m	Mass of the quadcopter
l	Length between two propellers
I_{xx}	x moment of inertia
I_{yy}	y moment of inertia
I_{zz}	z moment of inertia
$\sum T$	Sum of all the trust force produced by the propeller
M_1	Moment along x direction
M_2	Moment along x direction
M_3	Moment along x direction

Three angles known as the Euler angles were developed by Leonhard Euler to describe how a rigid body is oriented. Three parameters are needed to describe such an orientation in the three-dimensional Euclidean space.

1.2.1 State space modelling

The state-space model is linear representation of the dynamic model. The dynamic model has four inputs (Eq 2), six outputs (Eq 3) and the states of twelve which are respected in a vector shown in Eq 1.Where $\vec{x}$ is the state vector table 1 describes the states , $\vec{u}$ is the control input vector table 2 describes the control inputs to the model and $\vec{y}$ is the output vector; A nonlinear states space model of a quadcopter can be expressed as

$$\vec{x} = [\, X \quad Y \quad Z \quad \dot{X} \quad \dot{Y} \quad \dot{Z} \quad \phi \quad \theta \quad \psi \quad \dot{\phi} \quad \dot{\theta} \quad \dot{\psi} \,] \text{ (1)}$$

$$\vec{u} = \lfloor u_1 \quad u_2 \quad u_3 \quad u_4 \rfloor = \lfloor \sum T \quad M_1 \quad M_2 \quad M_3 \rfloor \text{ (2)}$$

$$\vec{y} = [X \quad Y \quad Z \quad \phi \quad \theta \quad \psi \,] \text{ (3)}$$

The expressions for $\vec{x}$ are given by

$$\dot{X} = \dot{X}$$

$$\dot{Y} = \dot{Y}$$

$$\dot{Z} = \dot{Z}$$

$$\ddot{X} = -\frac{1}{m}(sin\psi \,.\, sin\phi + \cos\psi \,.\, \sin\theta \,.\, \cos\phi).\sum T.$$

$$\ddot{Y} = -\frac{1}{m}(-cos\psi \,.\, sin\phi + \sin\psi \,.\, \sin\theta \,.\, \cos\phi).\sum T.$$

$$\ddot{Z} = -\frac{1}{m}(\cos\theta \,.\, \cos\phi).\sum T.$$

$$\dot{\phi} = p + s(\phi)t(\theta) \,.\, q + c(\phi)t(\theta).\, l$$

$$\dot{\theta} = c(\phi).\, q - s(\phi).\, l$$

$$\dot{\psi} = s(\phi)\sec(\theta) \,.\, q + c(\phi)\sec(\theta) \,.\, l$$

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} \,.\, qr + \frac{1}{I_{xx}} \,.\, M_1$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} . rp + \frac{1}{I_{yy}} . M_2$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} . pq + \frac{1}{I_{zz}} . M_3$$

The model has to be linearized at an equilibrium point which results in hovering which is given by $\vec{x} = [\,0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \psi \quad 0 \quad 0 \quad 0\,]$ and $\vec{u} = [mg \quad 0 \quad 0 \quad 0]$ which gives a linear equation of the state space as follows

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Were the final state space model is defined as with the A,B,C and D that define the dynamic model in the equilibrium is as follows

$$\dot{x} = Ax + Bu \quad (4)$$
$$y = Cx + Du \quad (5)$$

The output vector is defined as

$$y = [X \quad Y \quad Z \quad \phi \quad \theta \quad \psi\,]$$

1.3 Controller Design using pole placement

In this method, closed loop poles of a plant are placed in match in with desired performance parameters. The performance parameter that are choose in this case are settling time (t_s) of 5 sec and maximum overshoots (M_p) of 50%. Location of closed-loop poles equal to the Eigen values of the closed-loop dynamic system. The desired closed loop poles are set the in H matrix hence there are 12 states there must be twelve roots and the desired must be dominating among them. Based on those desired locations, the feedback gain K is calculated using the following MATLAB command.

$$K = place\ (A, B, \mathrm{H}) \quad (6)$$

The control input vector is given by

$$\vec{u}_{k-1} = -K\vec{x}_{k-1} \ (7)$$

And with refence state that modified control input is

$$\vec{u}_{k-1} = -K(\overrightarrow{x_{k-1}} - \overrightarrow{x_{r_k}}) \ (8)$$

The system behavior can be altered into a favorable behavior with help of the State-Feedback matrix using the pole placement control design. In this method be force the desired closed loop poles(H matrix) of the system by designing a feedback control system. It can be tuned by changing K, which we can do by changing the location of poles in H. If the closed loop transfer function of the system is represented by a state-space equation,

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

then the poles of the system are the roots of the characteristic equation given by

$$det\ (A - \lambda I) = 0 \ (9)$$

The roots of the characteristic equation of the pole placement-based system are given by

$$det\ ((A - KB) - \lambda I) = 0 \ (10)$$

However, the system needs to controllable to implement this method. When a system can be shifted from its initial state to any other state in a finite amount of time using an unrestricted control vector, it is said to be controllable at the initial time. For a system to be controllable, the following matrix is defined as

$$C = [B \quad BA \quad BA^2 \ . \ . \quad BA^{11}] \ (11)$$

which is called the controllability matrix (C) needs to be of rank 12 states has the dynamic system has twelves states. or should have n linearly independent column vectors. Weather to know the system controllable that rank of the controllability matrix must be equal to the number of states in the system. The MATLAB code for the rank of the controllability matrix is give as follows

$$\text{rank}(\ ctrb\ (A, B)) \quad (12)$$

In this case the rank of the controllability matrix matches with number of states in the system.

1.4 Observer for the linear model (Luenberger observer)

The problem with dynamics states is it need to be initialized, doesn't count for the disturbances in the system (disturbances can have impact such that the equation turns off) and as it an iterative model errors can accumulate. To know what is going on with the system, the system must a medium to accurately measure the dynamic states response which are the important variables that control the system. It is possible to represent concept of observability for a linear time-invariant dynamical system into linear systems of algebraic equations

Luenberger observer also known as state-prediction estimator was chosen to estimate the states of the Quadcopter. The definition of the estimator is given by

$$\vec{\hat{x}}_k = A\vec{\hat{x}}_{k-1} + B\vec{u}_{k-1} - L(\vec{\hat{y}}_{k-1} - \vec{y}_{k-1}) \quad (13)$$

$$\vec{\hat{y}}_k = C\vec{\hat{x}}_k + D\vec{u}_k \quad (14)$$

A ,B, C and D are state space model vectors. Where $\vec{\hat{x}}_k$ is the estimated states vector which contains the estimated states of the system $\vec{\hat{y}}_k$ is the estimated output , L is the state-feedback gain matrix of the estimator which was calculated using the following expression in the MATLAB. $z^*_{Estimator}$ are the poles selected for the estimator where the poles must settle three to six times faster than the closed-loop poles of the system. The calculation of the $z^*_{Estimator}$ was based on the performance parameter that are settling time of the estimator (t_{e_s}) which is one by third times of the settling time of the controller and maximum overshoots (M_{e_p}) of 30 %.

$$L = place(A', C', z^*_{Estimator}) \quad (15)$$

$\vec{u}_{k-1}$is the control input vector in the previous time step which is defined as follows in the Eq 16 where K is the state-feedback matrix that is defined in the Controller design section previously .

$$\vec{u}_{k-1} = -K\vec{\hat{x}}_{k-1} \text{ (16)}$$

With a reference state vector $\overrightarrow{x_r}_k$, the control vector is given by

$$\vec{u}_{k-1} = -K(\vec{\hat{x}}_{k-1} - \overrightarrow{x_r}_{k-1}) \text{ (17)}$$

$\vec{\tilde{x}}_k$ is the error between the estimated states vector and the state vector of the system model which is defined as follows.

$$\vec{\tilde{x}}_k = \vec{\hat{x}}_k - \vec{x}_k \text{ (18)}$$

$\vec{\tilde{x}}_k$ can be written as Eq 19 where eigen values $(A - LC)$ that is the closed-loop poles must be stable which means the estimated states goes to the true state values

$$\vec{\tilde{x}}_k = (A - LC)\vec{\tilde{x}}_{k-1} \text{ (19)}$$

The controllability matrix O seen in the Eq 20 must invertible so the estimator gain-matrix exist. if this is not invertible then it can be said that the system is non- observable.

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \cdot \\ \cdot \\ \cdot \\ CA^{11} \end{bmatrix} \text{ (20)}$$

It is commonly known that if and only if the rank of the system matrix is full, a solvable system of linear algebraic equations has a solution.

1.5 Hovering and Altitude control

For the hovering the reference is zero as there is no desired configuration. The only aim is taking the state vectors to zero which gives a steady state response.

For the Altitude control there is reference signal in the mathematical mode as there is desired configuration to be achieved.

2 Results

The table 3 illustrate the properties of the Quadcopter that was used in the simulation based on the reference [3]

Table 3: Properties of the Quadcopter

$Properties$	Value
m	0.468 Kg
l	0.225 m
I_{xx}	4.856*10^(-3) $Kg\,.m^2$
I_{yy}	4.856*10^(-3) $Kg\,.m^2$
I_{zz}	8.801*10^(-3) $Kg\,.m^2$

The sampling frequency is defined as the sensor capabilities but for this simulation. The frequency of 20 Hz was selected for the simulation. The MATLAB command c2d was used to convert the continuous State-Space system into a discrete State-Space system. The simulation was for a duration of 5 Sec.

2.1 Hovering

For the hovering stabilization the initial condition of that was assume was expressed in the MATLAB command format below

```
x(:,1)=[0;0;0;0;0;0;0;0;(pi/180)*45;0;0;0]; % initial states
```

An angle of 45 degrees has been applied to the dynamic system which the state-feedback controller was used to make this angle to zero that is stabilizing the system. The states were estimated using the estimator. The plots are represented in the Fig 3,4,5,6.

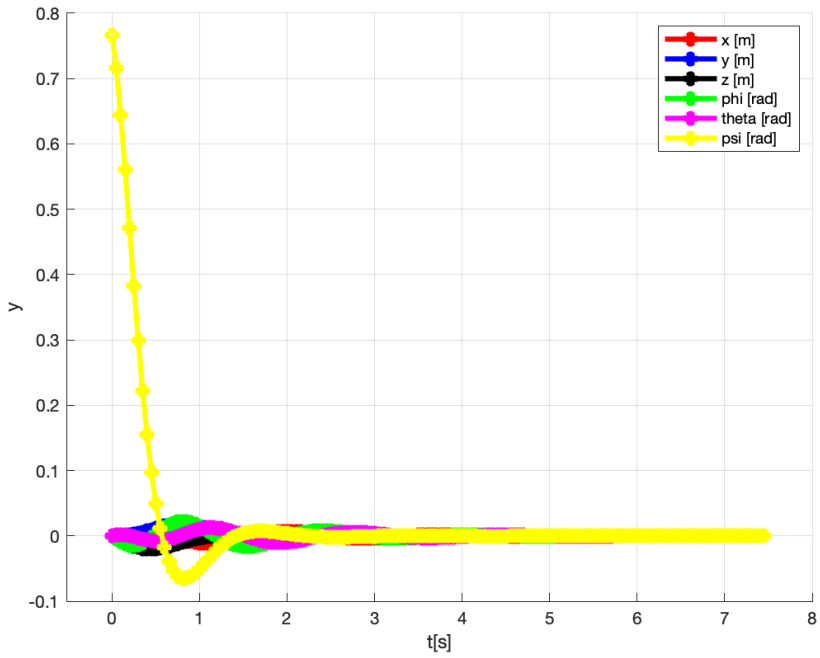

Figure 3 : The output of the Quadcopter for Hovering control

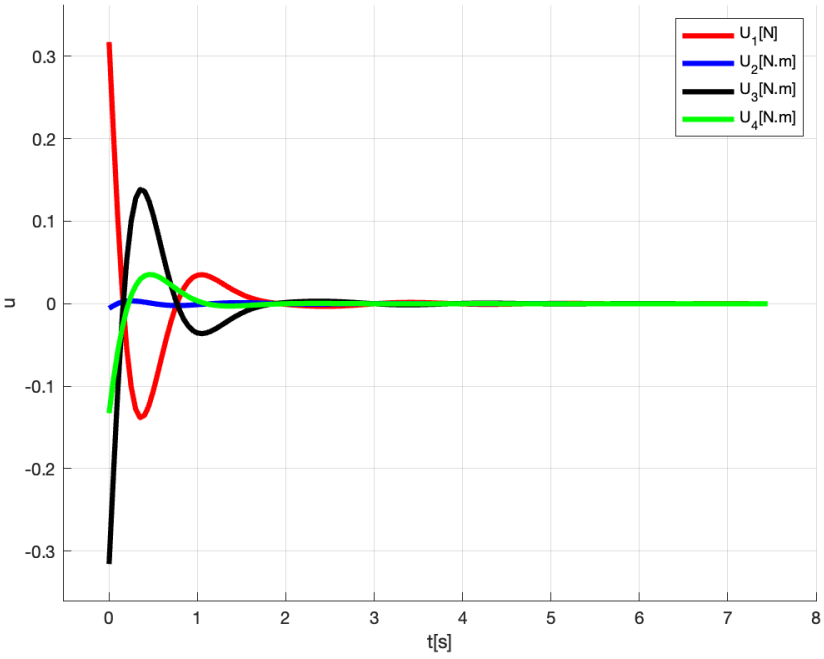

Figure 4 : The control inputs of the Quadcopter for Hovering control

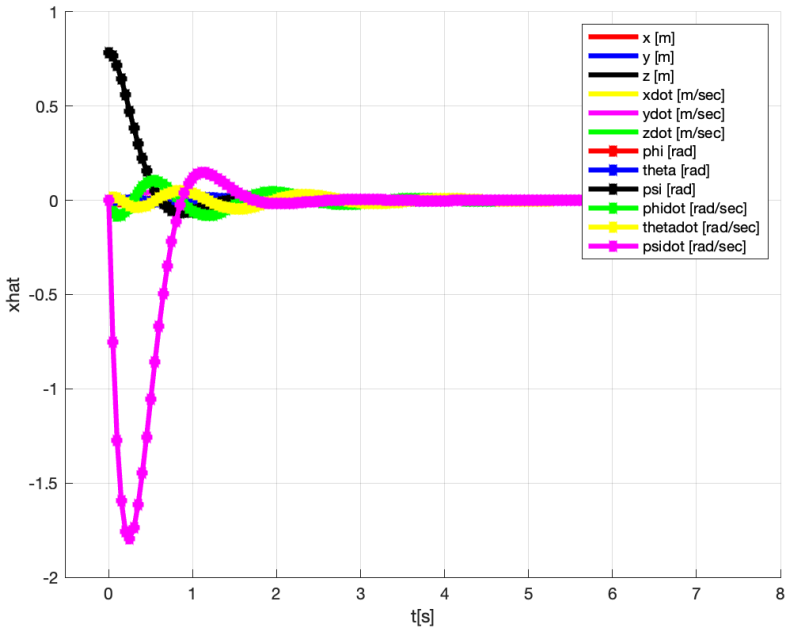

Figure 5 : The estimated states of the Quadcopter for Hovering control

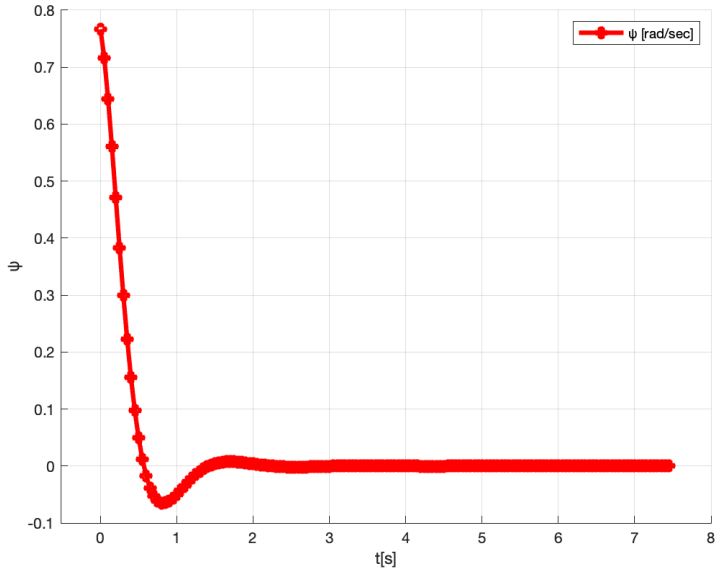

Figure 6 : The controlled state ψ of the Quadcopter for Hovering control

2.2 Altitude control

For controlling the altitude of the Quadcopter, the input is given only in the z direction as it is the direction in which the altitude is measured. So, there was a reference state used for the simulation with a sampling frequency of 20 Hz. The states were estimated using the estimator. Below are the reference states which are defined in MATLAB where z states which represents the altitude. The given reference is 4m.

```
xr=[0;0;4;0;0;0;0;0;0;0;0;0];% reference states
```

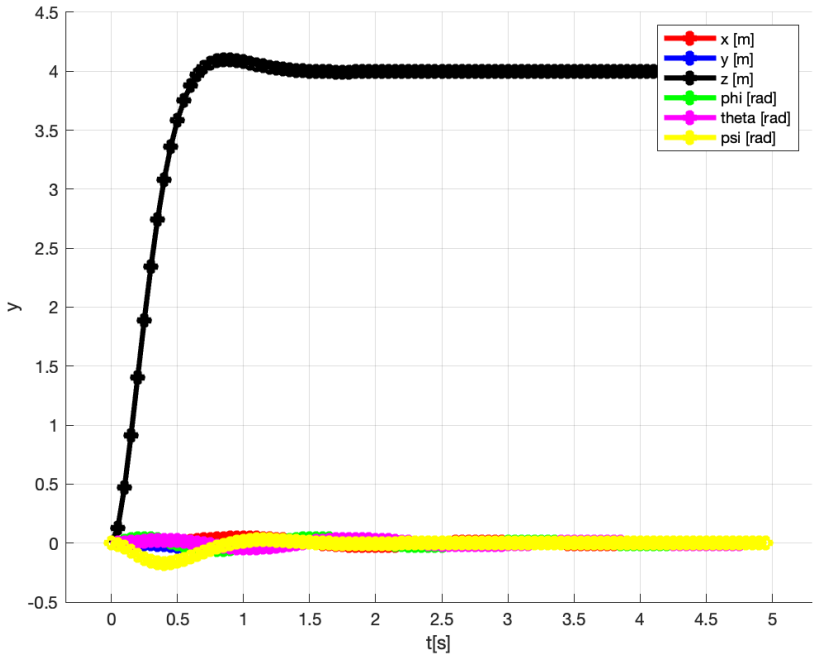

Figure 7 : The output of the Quadcopter for Altitude control

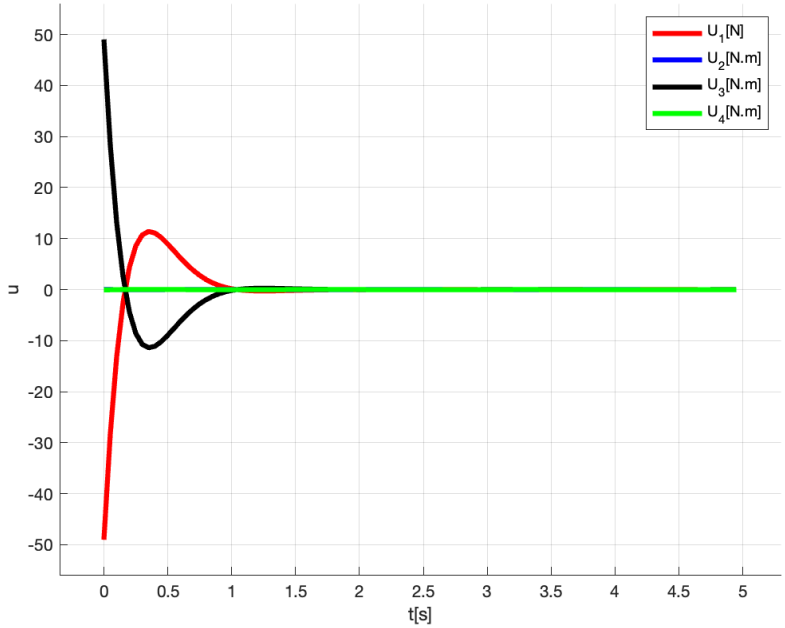

Figure 8 : The control inputs of the Quadcopter for Altitude control

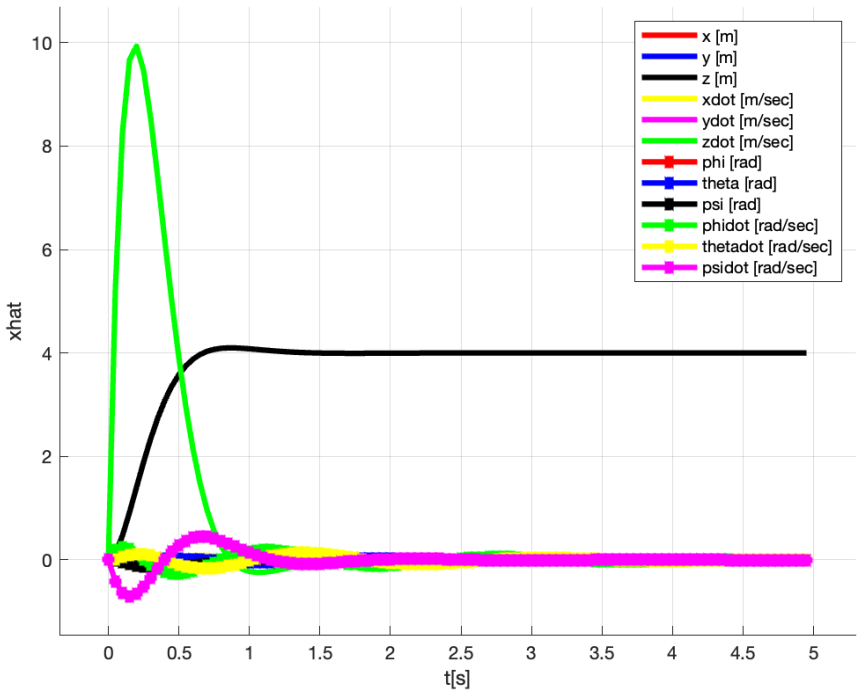

Figure 9 : The estimated states of the Quadcopter for Altitude control

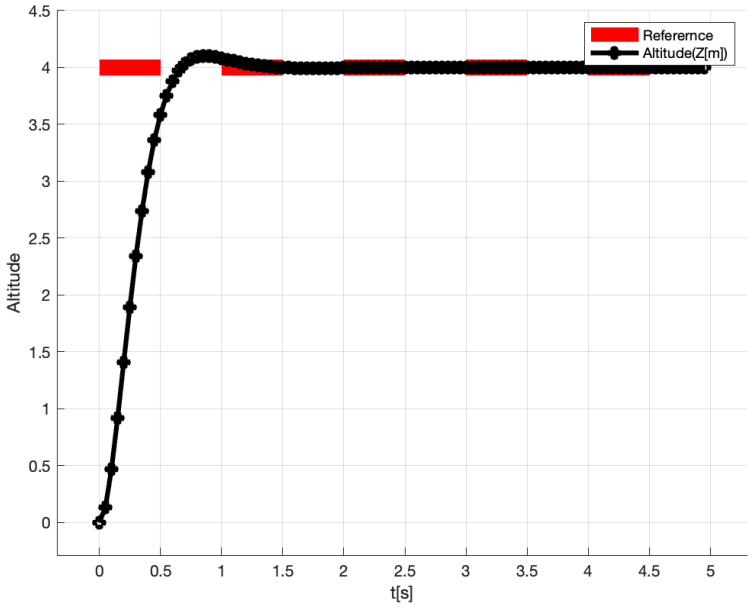

Figure 10 : The controlled altitude output of the Quadcopter for Altitude control

3 Discussion

The state-feedback controller has achieved the stabilization of the quadcopter by making the psi value to zero. It can be seen from Fig 3 other point worth mentioning is that there is not much variation in the remaining state apart from psi value. In Fig 4. There is thrust force from all the propellers and a moment generated in the y and z directions of the body frame for the system to be stable. It can be observed that there is no moment in x direction. The estimated states plot (Fig 5) show that while there is a change in angle psi the angle rate of psi which is psi dot, is also changing. This is due to the velocity at which of the quadcopter is moving itself in the psi direction which is varying with time. Fig 6 illustrates the stabilization of the quadcopter more clearly. Coming to the altitude control. In the Fig 7 the output z which is the altitude went to the desired value of 4m it can be noticed that there are few adjustments in the other states which are responses to the dynamics of the model. In the Fig 8 we can see that thrust forces of the quadcopter along with moment in the y direction .In this case unlike hovering there is no input of moment in x and z directions. It can be inferred from the Fig 9 since there is a movement in the z direction that is in the altitude direction the presence of the velocity in the z direction can be seen. The Fig 10 shows the reference of the altitude along with the response of the system that achieves the desired performance. Therefore, the results obtained show that state-feedback controller is capable of controlling the altitude and hovering of the quadcopter. The main limitation of the state-feedback controls if the there is any very significant disturbances in the system the equation will no longer work it acts only after the system has moved to the next point of the time it will take action. The future work on this includes testing different types of control strategies and optimizations to control the states of the quadcopter helping to choose the best control technique. Other work is the trajectory planning of the quadcopter flight path other than just altitude and hovering control. Finally, developing an accurate mathematical model which includes all the disturbances in the system and in the system environment.

References

[1] Tahir, Zaid & Jamil, Mohsin & Liaqat, Saad & Mubarak, Lubva & Tahir, Waleed & Gilani, Syed. (2016). State Space System Modeling of a Quad Copter UAV. Indian Journal of Science and Technology. 9. 10.48550/arXiv.1908.07401.

[2]Thanh, Ha Le Nhu Ngoc, Tuan Tu Huynh, Mai the Vu, Nguyen Xuan Mung, Nguyen Ngoc Phi, Sung Kyung Hong, and Truong Nguyen Luan Vu. 2022. "Quadcopter UAVs Extended States/Disturbance Observer-Based Nonlinear Robust Backstepping Control" *Sensors* 22, no. 14: 5082. https://doi.org/10.3390/s22145082

[3] Teppo Luukkonen , "Modelling and control of quadcopter" School of Science Mat-2.4108 Independent research project in applied mathematics Espoo, August 22, 2011.

Appendix

Hovering

```
clc;
clear all;
close all;
%% HOVERING
m=0.468;
g=-9.81;
r=0.225;
Ixx=4.856*10^(-3);
Iyy=4.856*10^(-3);
Izz=8.801*10^(-3);

Ac=[0 0 0 1 0 0 0 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0 0 0 0;
   0 0 0 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 -g 0 0 0 0;
   0 0 0 0 0 0 g 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 0 0 0 1;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0];
Bc=[0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   -1/m 0 0 0;
    0 0 0 0;
    0 0 0 0;
    0 0 0 0;
```

```
    0 1/Ixx 0 0;
    1/Iyy 0 1/Iyy 0;
     0 0 0 1/Izz];
C=[1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0;
   0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0];
D=zeros(6,4);
%% Continous to discrete
sysC=ss(Ac,Bc,C,D);
f=20;
T=1/f;
sysD=c2d(sysC,T);
A=sysD.A;
B=sysD.B;
rank(ctrb(A,B))
%% pole placement controller
ts=5;
Mp=0.5;
e=sqrt((log(Mp).^2)./(log(Mp).^2 + pi^2));
wn=4./(e.*ts);
wd=wn.*sqrt(1-e.^2);
Croots=-e.*wn+wd*1i;
Droots=exp(Croots*T);
H=[Droots,conj(Droots),0.9*Droots,0.9*conj(Droots),0.95*Droots,0.95*conj(Droo
ts),0.8*Droots,0.8*conj(Droots),0.85*Droots,0.85*conj(Droots),0.825*Droots,0.
825*conj(Droots)];
K= place(A,B,H);

%% Estimator 3X faster
Mp_E=0.3;
zeta_E=sqrt((log(Mp_E)^2)/(log(Mp_E)^2 + pi^2));
ts_E=ts/3;
wn_E=4/(zeta_E*ts_E);
wd_E=wn_E*sqrt(1-zeta_E^2);
lambdac_star_E=-zeta_E*wn_E+wd_E*1i;
lambdad_star_E=exp(lambdac_star_E*T);
beta_star_E=[lambdad_star_E,conj(lambdad_star_E),0.2,-0.2,0.3,-0.3,0.4,-
0.4,0.5,-0.5,0.6,-0.6];
L=place(A',C',beta_star_E)';
eig(A-L*C);
%% Simulation the response
N=150;
x=zeros(12,N);
y=zeros(6,N);
t=0:T:(N-1)*T;
u=zeros(4,N); %input
xhat=zeros(12,N);% estimated states
yhat=zeros(6,N); % estimated output
x(:,1)=[0;0;0;0;0;0;0;0;(pi/180)*45;0;0;0]; % initial states
xhat(:,1)=x(:,1);
 u(:,1)=-K*(x(:,1));
for k=2:N
```

```
    x(:,k)=A*x(:,k-1)+B*u(:,k-1);
    xhat(:,k)=A*x(:,k-1)+B*u(:,k-1)-L*(yhat(:,k-1)-y(:,k-1));
    y(:,k-1)=C*x(:,k)+D*u(:,k);
    yhat(:,k-1)=C*xhat(:,k)+D*u(:,k);
   u(:,k) = -K*(x(:,k));
end

figure
hold on

% outputs
plot(t,yhat(1,:),'ro-','LineWidth',3)
plot(t,yhat(2,:),'bo-','LineWidth',3)
plot(t,yhat(3,:),'ko-','LineWidth',3)
plot(t,yhat(4,:),'go-','LineWidth',3)
plot(t,yhat(5,:),'mo-','LineWidth',3)
plot(t,y(6,:),'yo-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('y')
legend('x [m]','y [m]','z [m]','phi [rad]','theta [rad]','psi [rad]')

% control input plot
figure
hold on
plot(t,u(1,:),'r','LineWidth',3)
plot(t,u(2,:),'b','LineWidth',3)
plot(t,u(3,:),'k','LineWidth',3)
plot(t,u(4,:),'g','LineWidth',3)
legend('U_1[N]','U_2[N.m]','U_3[N.m]','U_4[N.m]')
grid on
axis padded
xlabel('t[s]'),ylabel('u')

% states  plot
figure
hold on
plot(t,xhat(1,:),'r','LineWidth',3)
plot(t,xhat(2,:),'b','LineWidth',3)
plot(t,xhat(3,:),'k','LineWidth',3)
plot(t,xhat(4,:),'y','LineWidth',3)
plot(t,xhat(5,:),'m','LineWidth',3)
plot(t,xhat(6,:),'g','LineWidth',3)
plot(t,xhat(7,:),'r*-','LineWidth',3)
plot(t,xhat(8,:),'b*-','LineWidth',3)
plot(t,xhat(9,:),'k*-','LineWidth',3)
plot(t,xhat(10,:),'g*-','LineWidth',3)
plot(t,xhat(11,:),'y*-','LineWidth',3)
plot(t,xhat(12,:),'m*-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('xhat')
legend('x [m]','y [m]','z [m] ','xdot [m/sec]','ydot [m/sec]','zdot
[m/sec]','phi [rad]','theta [rad] ','psi [rad]','phidot [rad/sec]','thetadot
[rad/sec]','psidot [rad/sec]')
```

```
figure
hold on
plot(t,y(6,:),'ro-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('ψ')
legend('ψ [rad/sec]')
```

Altitude control

```
clc;
clear all;
close all;

%% ALTITUDE CONTROL

m=0.468;
g=-9.81;
r=0.225;
Ixx=4.856*10^(-3);
Iyy=4.856*10^(-3);
Izz=8.801*10^(-3);


Ac=[0 0 0 1 0 0 0 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0 0 0 0;
   0 0 0 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 -g 0 0 0 0;
   0 0 0 0 0 0 g 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 0 0 0 1;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0];
Bc=[0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   0 0 0 0;
   -1/m 0 0 0;
    0 0 0 0;
    0 0 0 0;
    0 0 0 0;
    0 1/Ixx 0 0;
    1/Iyy 0 1/Iyy 0;
     0 0 0 1/Izz];
C=[1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0 0 0 0;
```

```
   0 0 0 0 0 0 1 0 0 0 0 0;
   0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0];
D=zeros(6,4);
sysC=ss(Ac,Bc,C,D);
f=20;
T=1/f;
sysD=c2d(sysC,T);
A=sysD.A;
B=sysD.B;
%% Pole placement controller
ts=5;
Mp=0.5;
e=sqrt((log(Mp).^2)./(log(Mp).^2 + pi^2));
wn=4./(e.*ts);
wd=wn.*sqrt(1-e.^2);
Croots=-e.*wn+wd*1i;
Droots=exp(Croots*T);
H=[Droots,conj(Droots),0.9*Droots,0.9*conj(Droots),0.95*Droots,0.95*conj(Droo
ts),0.8*Droots,0.8*conj(Droots),0.85*Droots,0.85*conj(Droots),0.825*Droots,0.
825*conj(Droots)];
K= place(A,B,H);
%% Estimator 3X faster
Mp_E=0.3;
zeta_E=sqrt((log(Mp_E)^2)/(log(Mp_E)^2 + pi^2));
ts_E=ts/3;
wn_E=4/(zeta_E*ts_E);
wd_E=wn_E*sqrt(1-zeta_E^2);
lambdac_star_E=-zeta_E*wn_E+wd_E*1i;
lambdad_star_E=exp(lambdac_star_E*T);
beta_star_E=[lambdad_star_E,conj(lambdad_star_E),0.2,-0.2,0.3,-0.3,0.4,-
0.4,0.5,-0.5,0.6,-0.6];
L=place(A',C',beta_star_E)';
eig(A-L*C);
%% Simulation of teh response
N=100;
x=zeros(12,N);
y=zeros(6,N);
t=0:T:(N-1)*T;
u=zeros(4,N); %input
xhat=zeros(12,N);% estimated states
yhat=zeros(6,N); % estimated output
x(:,1)=[0;0;0;0;0;0;0;0;0;0;0;0];% intital condition for states
xhat(:,1)=[0;0;0;0;0;0;0;0;0;0;0;0];% intital condition for estimated states
xr=[0;0;4;0;0;0;0;0;0;0;0;0];% reference states
 u(:,1)=-K*(x(:,1)-xr);
for k=2:N
    x(:,k)=A*x(:,k-1)+B*u(:,k-1);
    xhat(:,k)=A*x(:,k-1)+B*u(:,k-1)-L*(yhat(:,k-1)-y(:,k-1));
    y(:,k)=C*x(:,k)+D*u(:,k);
    yhat(:,k)=C*xhat(:,k)+D*u(:,k);
   u(:,k) = -K*(xhat(:,k)-xr);
end

figure
```

```
hold on

% outputs
plot(t,yhat(1,:),'ro-','LineWidth',3)
plot(t,yhat(2,:),'bo-','LineWidth',3)
plot(t,yhat(3,:),'ko-','LineWidth',3)
plot(t,yhat(4,:),'go-','LineWidth',3)
plot(t,yhat(5,:),'mo-','LineWidth',3)
plot(t,y(6,:),'yo-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('y')
legend('x','y','z','phi','theta','psi')

% control input plot
figure
hold on
plot(t,u(1,:),'r','LineWidth',3)
plot(t,u(2,:),'b','LineWidth',3)
plot(t,u(3,:),'k','LineWidth',3)
plot(t,u(4,:),'g','LineWidth',3)
legend('U_1','U_2','U_3','U_4')
grid on
axis padded
xlabel('t[s]'),ylabel('u')

% states  plot
figure
hold on
plot(t,xhat(1,:),'r','LineWidth',3)
plot(t,xhat(2,:),'b','LineWidth',3)
plot(t,xhat(3,:),'k','LineWidth',3)
plot(t,xhat(4,:),'y','LineWidth',3)
plot(t,xhat(5,:),'m','LineWidth',3)
plot(t,xhat(6,:),'g','LineWidth',3)
plot(t,xhat(7,:),'r*-','LineWidth',3)
plot(t,xhat(8,:),'b*-','LineWidth',3)
plot(t,xhat(9,:),'k*-','LineWidth',3)
plot(t,xhat(10,:),'g*-','LineWidth',3)
plot(t,xhat(11,:),'y*-','LineWidth',3)
plot(t,xhat(12,:),'m*-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('xhat')
legend('x','y','z','xdot','ydot','zdot','phi','theta','psi','phidot','thetado
t','psidot')

figure
hold on
plot(t,xr(3).*ones(length(t),1),'r--','LineWidth',10)
plot(t,yhat(3,:),'ko-','LineWidth',3)
grid on
axis padded
xlabel('t[s]'),ylabel('Altitude')
legend('Refererrnce','Altitude(Z[m])')
```