Modeling, control, and simulation of quadcopters

Batuhan Toker

1 Introduction

In this report I applied hover and trajecktory tracking control on a quadrotor. The kinematic model is represented as below:

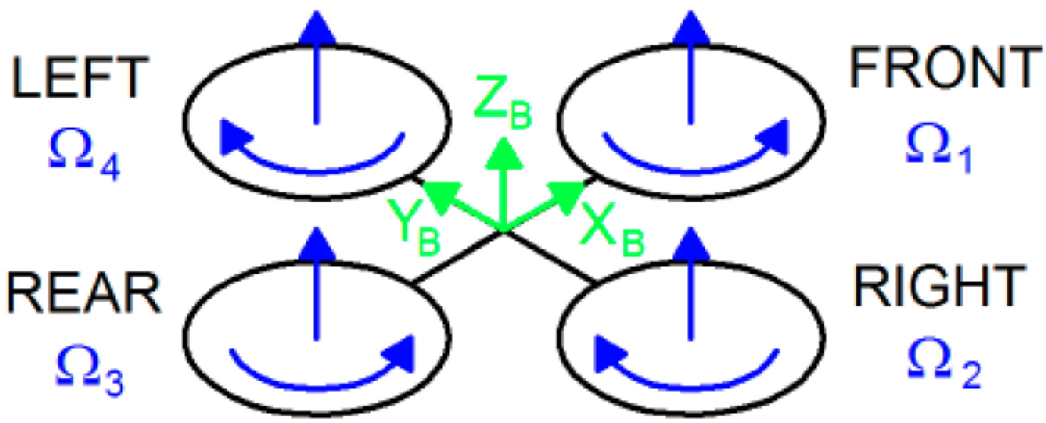


Figure 1: Simplified quadrotor model

This model can be dynamically modelled by following equations

$$\ddot{X} = (sin(\psi)sin(\phi) + cos(\psi)sin(\theta)cos(\phi))\frac{U1}{m} \tag{1}$$

$$\ddot{Y} = (-cos(\psi)sin(\phi) + cos(\psi)sin(\theta)cos(\phi))\frac{U1}{m} \tag{2}$$

$$\ddot{Z} = -g + (cos(\theta)cos(\phi))\frac{U1}{m} \tag{3}$$

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} qr + \frac{U_2}{I_{xx}} \tag{4}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{U_3}{I_{yy}} \tag{5}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{zz}} \tag{6}$$

The closed loop error of the system will be defined as

$$error_{pos} = pos_d - pos \tag{7}$$

A PID control will be applied both on position and angles. A general PID controller can be represented by the following equation

$$PID(e) = k_p e + k_d \dot{e} + k_i \int edt \tag{8}$$

Virtual control will calculate a virtual forces by using

$$\mu = \ddot{pos} + PID(error_{pos}) \tag{9}$$

Force input U_1 and desired angles will be

$$U_1 = m\sqrt{\mu_x^2 + \mu_y^2 + (g + \mu_z)^2} \tag{10}$$

$$\phi_d = asin(\frac{sin(\psi_d)\mu_x - cos(\psi_d)\mu_y}{\frac{U_1}{m}}) \tag{11}$$

$$\theta_d = asin(\frac{cos(\psi_d)\mu_x + sin(\psi_d)\mu_y}{cos(\phi_d)\frac{U_1}{m}}) \tag{12}$$

Other quadrotor inputs will be calculated by altitude control by using following equations

$$U_2 = I_{xx}(PID(e_\phi)) \tag{13}$$

$$U_3 = I_{yy}(PID(e_\theta)) \tag{14}$$

$$U_4 = I_{zz}(PID(e_\psi)) \tag{15}$$

Euler angles and angular velocity conversation is made by following equation system

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{16}$$

2 Control

The control approach on the defined problem can be represented as

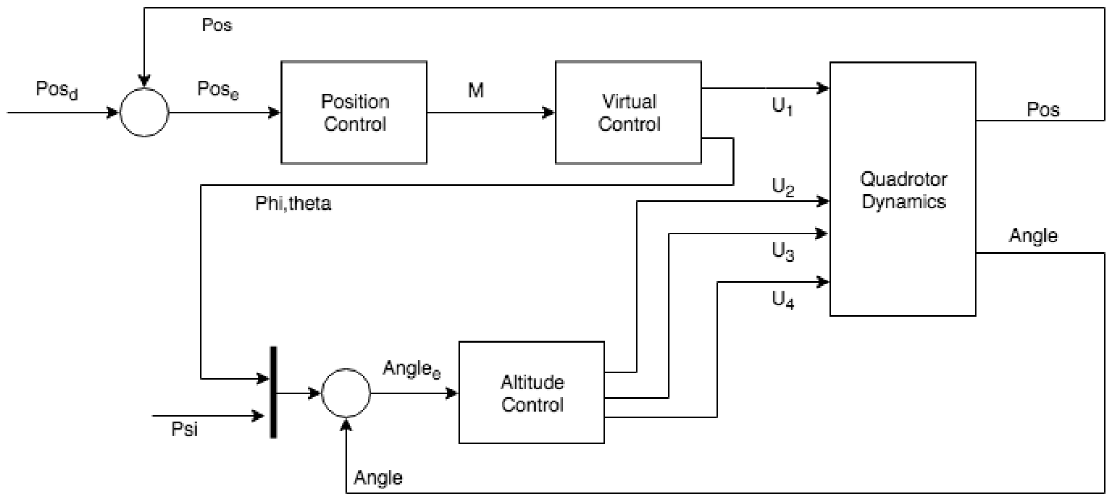


Figure 2: Overall closed loop control architecture of the quadrotor without disturbance observers

This architecture were realized through Simulink. The mass and inertias are defined as following

$$I_{xx} = 7.5 * 10^{-3} \tag{17}$$

$$I_{yy} = 7.5 * 10^{-3} \tag{18}$$

$$I_{zz} = 1.3 * 10^{-2} \tag{19}$$

$$m = 0.65 \tag{20}$$

To define a hover control trajectory the X_d and Y_d are defined as zero.So that their first and second derivatives are also zero. Trajecktory for the Z_d is defined with respect to time. Trajectory tracking control can be applied by defining X_d,Y_d and Z_d as 5^{th} order quintic polynomials.

3 Results

3.1 Hover Control

PID control were applied on both position and euler angles(roll, pitch, yaw) with coefficients of

$$k_{p,pos} = 25 \qquad k_{p,ang} = 10$$

$$k_{d,pos} = 0.6 \qquad k_{d,ang} = 8$$

$$k_{i,pos} = 0.5 \qquad k_{i,ang} = 0.5$$

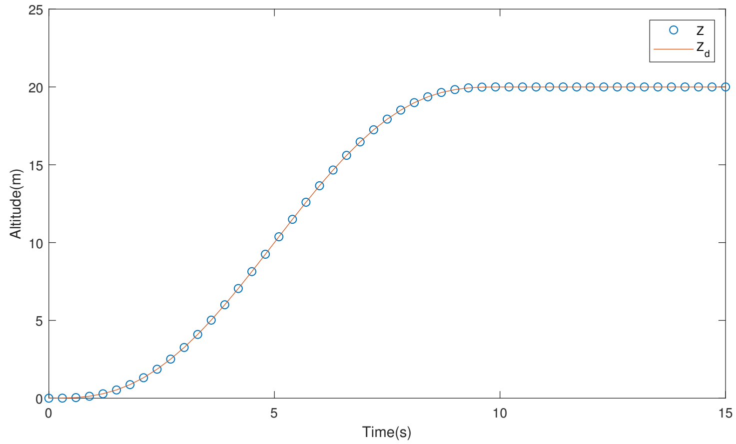


Figure 3: Trajecktory for hover control first case

The figure above represents the hovering motion of the quadrotor in which, the quadrotor goes to 20 meter altitude in 10 seconds and stays there for 5 seconds. Desired position and the results fits each other.

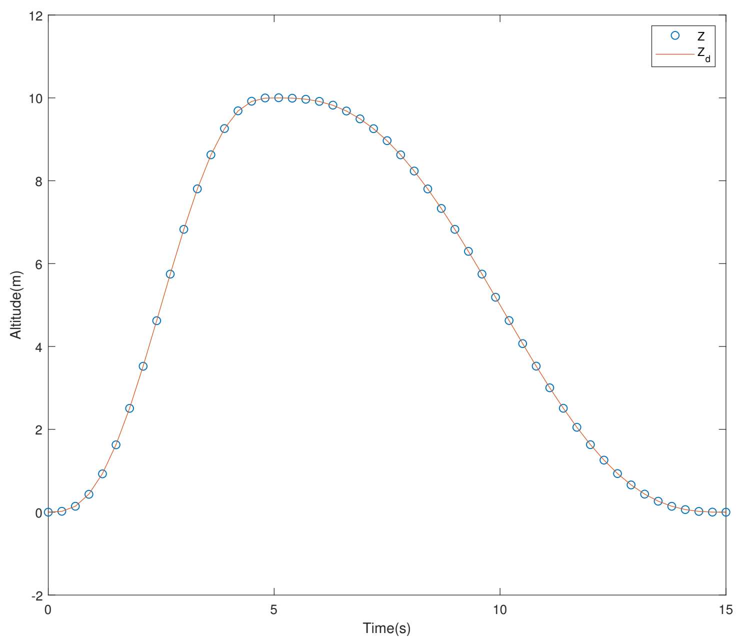


Figure 4: Trajecktory for hover control second case

The figure above represents hovering motion of the quadrotor in which, the quadrotor goes to 10 meter altitude in 5 seconds and goes to 0 meter in 10 seconds. Desired trajectory and the obtained results from quadrotor dynamics perfectly fits eacth other.

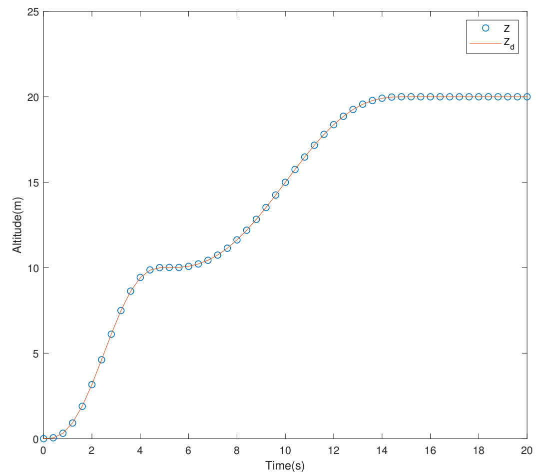


Figure 5: Trajecktory for hover control third case

The figure above represents hovering motion of the quadrotor in which, the quadrotor goes to 10 meter altitude in 5 seconds and stays there for a while, then goes to 20 meter altitude in a predefined time and stays there for another while. Desired trajectory and the obtained results from quadrotor dynamics perfectly fits eacth other.

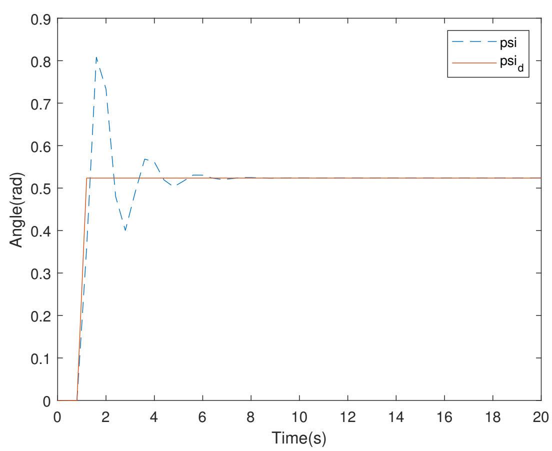


Figure 6: Control system's response on desired ψ value

The figure above represents hover control's response in case we define an angle for ψ.

3.2 Trajecktory Tracking Control

PID control on desired position is applied by using following controller gains:

$$\begin{bmatrix} k_{p,x} & k_{p,y} & k_{p,z} \\ k_{i,x} & k_{i,y} & k_{i,z} \\ k_{d,x} & k_{d,y} & k_{d,z} \end{bmatrix} = \begin{bmatrix} 25 & 25 & 25 \\ 0.6 & 0.5 & 0.5 \\ 0.1 & 0.7 & 0.6 \end{bmatrix} \tag{21}$$

The controller gains for the euler angles is the same with the hover control.

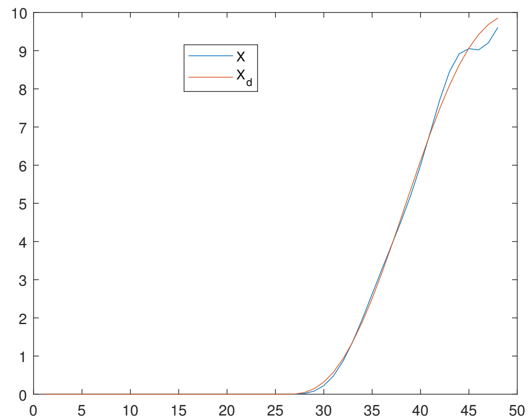


Figure 7: Control system's response on X_d value

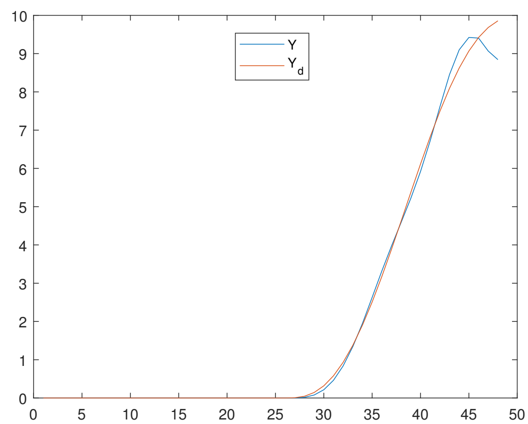


Figure 8: Control system's response on Y_d value

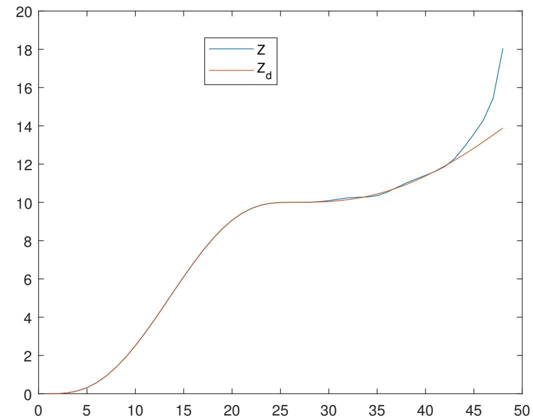


Figure 9: Control system's response on Z_d value

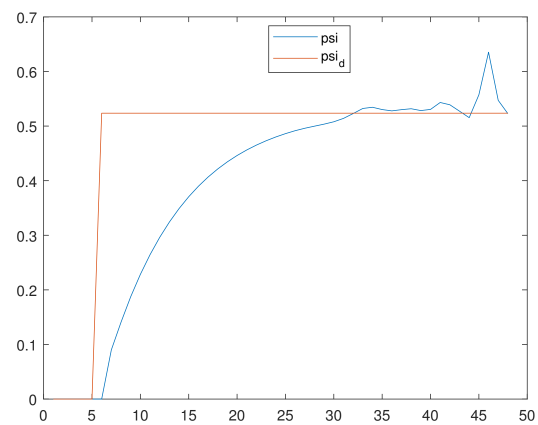


Figure 10: Control system's response on ψ_d value

4 Conclusion

In this report, I applied PID control on a quadrotor. Quadrotor and control dynamics are represented in the previous parts. In this work, there is no disturbance observer to control the quadrotor. The simulation is completed and the desired motions and obtained sensor values are fit each other.

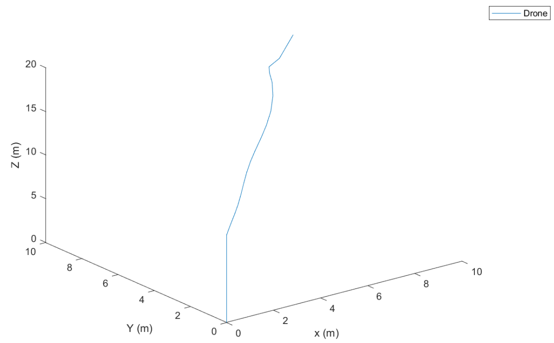


Figure 11: The trajectory followed by the quadrotor

5 Appendix

The MATLAB codes and Simulink model are provided in this section.

MATLAB Functions

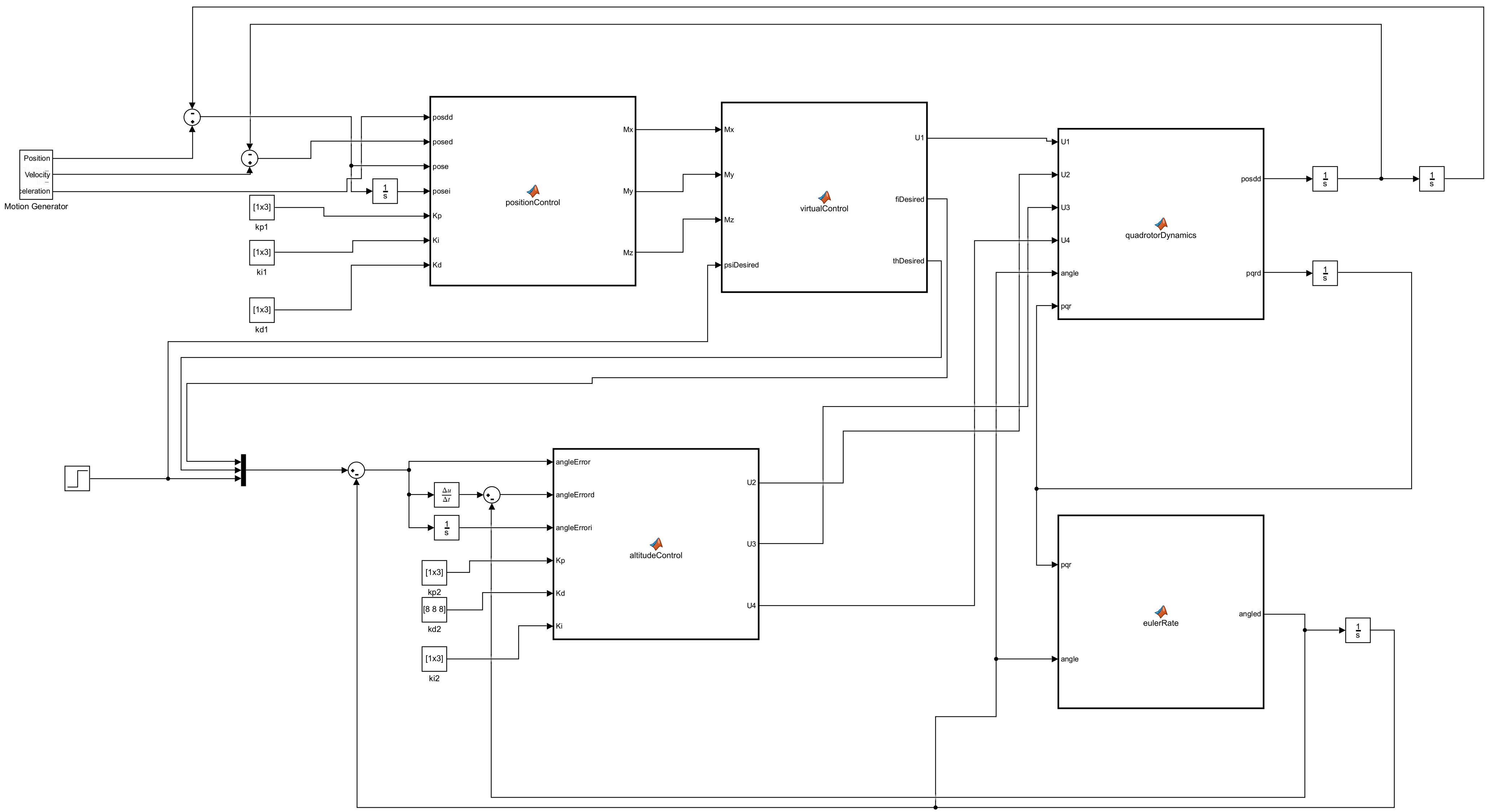
```
function [Mx,My,Mz] = positionControl(posdd,posed,pose,posei,Kp,Ki,Kd)
Mx=posdd(1)+Kp(1)*pose(1)+Kd(1)*posed(1)+Ki(1)*posei(1)
My=posdd(2)+Kp(2)*pose(2)+Kd(2)*posed(2)+Ki(2)*posei(2)
Mz=posdd(3)+Kp(3)*pose(3)+Kd(3)*posed(3)+Ki(3)*posei(3)
end
function [U1,fiDesired,thDesired] = virtualControl(Mx,My,Mz,psiDesired)
Ix = 7.5*10^(-3);  % Quadrotor moment of inertia around X axis
Iy = 7.5*10^(-3);  % Quadrotor moment of inertia around Y axis
Iz = 1.3*10^(-2);  % Quadrotor moment of inertia around Z axis
Jr = 6.5*10^(-5);  % Total rotational moment of inertia around the propeller axis
b = 3.13*10^(-5);  % Thrust factor
d = 7.5*10^(-7);  % Drag factor
l = 0.23;  % Distance to the center of the Quadrotor
m = 0.65;  % Mass of the Quadrotor in Kg
g = 9.81;   % Gravitational acceleration
U1= m*sqrt(Mx^2+My^2+(Mz+g)^2);
fiDesired=asin((sin(psiDesired)*Mx-cos(psiDesired)*My)/sqrt(Mx^2+My^2+(Mz+g^2)));
thDesired=asin((cos(psiDesired)*Mx+sin(psiDesired)*My)/(cos(fiDesired)*sqrt(Mx^2+My^2+(Mz+g)^2)));
end
function [posdd,pqrd] = quadrotorDynamics(U1,U2,U3,U4,angle,pqr)
```

```
% Quadrotor constants
Ix = 7.5*10^(-3);  % Quadrotor moment of inertia around X axis
Iy = 7.5*10^(-3);  % Quadrotor moment of inertia around Y axis
Iz = 1.3*10^(-2);  % Quadrotor moment of inertia around Z axis
Jr = 6.5*10^(-5);  % Total rotational moment of inertia around the propeller axis
b = 3.13*10^(-5);  % Thrust factor
d = 7.5*10^(-7);  % Drag factor
l = 0.23;  % Distance to the center of the Quadrotor
m = 0.65;  % Mass of the Quadrotor in Kg
g = 9.81;   % Gravitational acceleration
fi=angle(1);
th=angle(2);
psi=angle(3);
p=pqr(1);
q=pqr(2);
r=pqr(3);
xdd=(sin(psi)*sin(fi)+cos(psi)*sin(th)*cos(fi))*U1/m;
ydd=(-cos(psi)*sin(fi)+sin(psi)*sin(th)*cos(fi))*U1/m;
zdd=-g+(cos(th)*cos(fi))*U1/m;
pd=(Iy-Iz)/Ix*q*r+U2/Ix;
qd=(Iz-Ix)/Iy*p*r+U3/Iy;
rd=(Ix-Iy)/Iz*p*q+U4/Iz;
posdd=[xdd;ydd;zdd];
pqrd=[pd;qd;rd]
end
function [U2,U3,U4] = altitudeControl(angleError,angleErrord,angleErrori,Kp,Kd,Ki)
Ix = 7.5*10^(-3);  % Quadrotor moment of inertia around X axis
Iy = 7.5*10^(-3);  % Quadrotor moment of inertia around Y axis
Iz = 1.3*10^(-2);  % Quadrotor moment of inertia around Z axis
Jr = 6.5*10^(-5);  % Total rotational moment of inertia around the propeller axis
b = 3.13*10^(-5);  % Thrust factor
d = 7.5*10^(-7);  % Drag factor
l = 0.23;  % Distance to the center of the Quadrotor
m = 0.65;  % Mass of the Quadrotor in Kg
g = 9.81;   % Gravitational acceleration
I=[Ix;Iy;Iz];
U2=I(1)*(Kp(1)*angleError(1)+Kd(1)*angleErrord(1)+Ki(1)*angleErrori(1))
U3=I(2)*(Kp(2)*angleError(2)+Kd(2)*angleErrord(2)+Ki(2)*angleErrori(2))
U4=I(3)*(Kp(3)*angleError(3)+Kd(3)*angleErrord(3)+Ki(3)*angleErrori(3))
```

```
end
function angled = eulerRate(pqr,angle)
fi=angle(1);
th=angle(2);
psi=angle(3);
angled=[1 sin(fi)*tan(th) cos(fi)*tan(th);
        0 cos(fi)         -sin(fi);
        0 sin(fi)/cos(th) cos(fi)/cos(th)]*pqr;
end
%%Data simulation code
figure
axis vis3d
xlabel('X axis (m)')
ylabel('Y axis (m)')
zlabel('Z axis (m)')
cam=[1:length(x.data)]
for i=1:length(x.data)
    scatter3([x.data(i)],[y.data(i)],[z.data(i)],80,'k','x','
        DisplayName','drone')
    xlim([0 max(x.data)+1])
    ylim([0 max(y.data)+1])
    zlim([0 max(z.data)+1])
    view(cam(i)*0.3-75,cam(i)*0.1+40)
    pause(0.2)
end
```

Simulink

The Simulink model is given in the following page.



Position
Velocity
celeration
Motion Generator
posdd
posed
pose
posei
Kp
Ki
Kd
positionControl
Mx
My
Mz
[1x3]
kp1
ki1
kd1
virtualControl
psiDesired
U1
fiDesired
thDesired
quadrotorDynamics
U2
U3
U4
angle
pqr
pqrd
1/s
altitudeControl
angleError
angleErrord
angleErrori
[8 8 8]
kp2
kd2
ki2
eulerRate
angled