ME 425 HW 3
Motion Control Of A Quadcopter

Arda Gencer
34124
Instructor: Mustafa Ünel

Fall 2025-2026
Due Date: 27.11.2025

1 Introduction

In this homework of ME 425, motion control for a quadcopter was developed. First, a dynamic model (plant) for the quadcopter was developed. After this, controllers for both hover control, which allow the drone to maintain its height at a desired altitude and then land safely, and also for trajectory tracking, which allows the drone to follow smooth trajectories while in the air were developed. Several feed-forward and feedback techniques were used to develop these controllers.

2 Dynamic Model Of The Quadcopter

The model of the quadcopter was designed as a subsytem inside simulink. The subsystem takes 4 inputs U1,2,3,4; and outputs X,Y,Z and also the three euler rates. Inside the block, the angular velocities; p,q,r are calculated inside a seperate block. These are then fed into a matrix multiply block and the outputs (ϕ, θ, ψ) are used in a loop to calculate the elements of the T_θ matrix. The euler rates are also used with U_1 input to calculate the position of the quadcopter inside a seperate block as well. The dynamic model is untouched after this point. Only the four control inputs are designed such that we can receive relevant outputs.

3 Hover Control

Approppriate FF+ FB controllers were developed to keep the quadrotor at a desired altitude and then to land it back down or change the altitude. To do this, for the angles, a single feedforward term were used in terms of desired angles second derivative. For the desired altitude, 2 feedforward terms were used: One to compensate for the gravity and other for the desired altitude's second derivative. Note that the second derivative terms (linear and angular accelerations) for this application is zero since our desired values are constant. For feedback terms, usual PID controllers were used with the error in each dynamic.

For results, the quadrotor performed well with no starting angle. It was able to hover and then land back to the ground. In this stage, it could also be controlled in the X-Y plane too. However, when tried with small and then bigger starting angles (i.e quadrotor is tilted), although the drone could still hover and land as intended, it could not be controlled in the X-Y plane as its coordinates increased (or decreased) continuously. For the control efforts, only U_1 was needed when no drone was not tilted initially. When the drone had small starting angles, small values of U_2, U_3, U_4 were also used but they dropped back to zero after the quadcoptor stabilized its attitude in the air. The results can be found in the Results section.

4 Trajectory Tracking

To implement trajectory tracking, the visual control inputs developed in the class were used to compensate for the underactuated state of the position controls (since all three positions are all controlled by U_1. To develop these virtual accelerations, the following formulas were used from lecture slides:

$$\mu_x = \ddot{X}_d + K_P e_x + K_D \dot{e_x} + K_I \int e_x dt$$

$$\mu_y = \ddot{Y}_d + K_P e_y + K_D \dot{e_y} + K_I \int e_y dt$$

$$\mu_z = \ddot{Z}_d + K_P e_z + K_D \dot{e_z} + K_I \int e_z dt$$

After this point, since desired accelerations were known, they can be used to determine the desired angle values. Here to be able to get a unique solution, the yaw angle needs to be fixed to a specific value. Then, the following equations were used to determine the desired angle values:

$$(\sin\psi^* \sin\phi_d + \cos\psi^* \sin\theta_d \cos\phi_d)\frac{U_1}{m} = \mu_x \tag{1}$$

$$(-\cos\psi^* \sin\phi_d + \sin\psi^* \sin\theta_d \cos\phi_d)\frac{U_1}{m} = \mu_y \tag{2}$$

$$(\cos\theta_d \cos\phi_d)\frac{U_1}{m} = \mu_z + g \tag{3}$$

Here, U_1 is set as:

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

so that we can also take into account the accelerations in the X and Y plane as well when determining the thrust.

Then, we can use these equations to solve for ϕ_d, θ_d and design these angle values accordingly. After this, we can use the same design principles for U inputs that we have used for hover control.

After the design process, the controller parameters were tuned so that the system would be stable and it would have as little overshoot and settling time as possible. These tunings were done experimentally.

Then, the setup was tested with a test case where the drone would start on a flat surface and then take off and hover at an altitude of 2 meters. At the 10th second, the desired X value would be changed to 1 in a gradual manner (to avoid spikes in derivative blocks. All these changes were done like this instead of just giving a step input.). Then, at the 50th second, the psi value was changed to 2π to make the drone make a full rotation around itself. Finally, at the 60th second, the zDesired is set back to zero to make the drone land. The resulting plots of this process can be found at the Results section.

5 Results

5.1 Hover Control

5.1.1 No Starting Angle

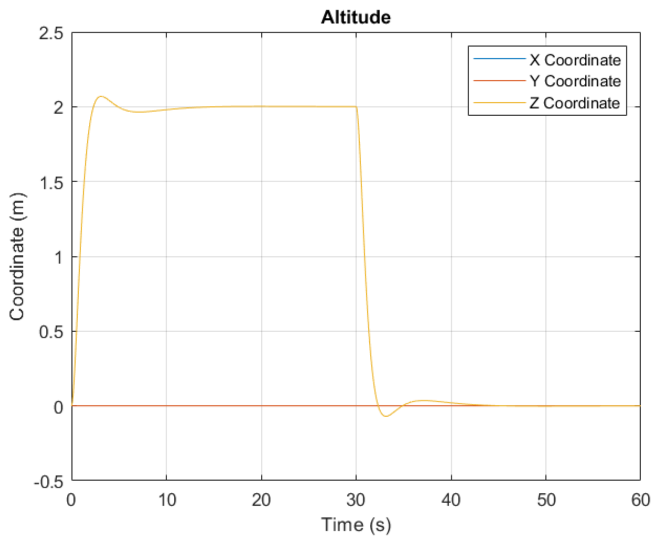

Figure 1: Position Dynamics

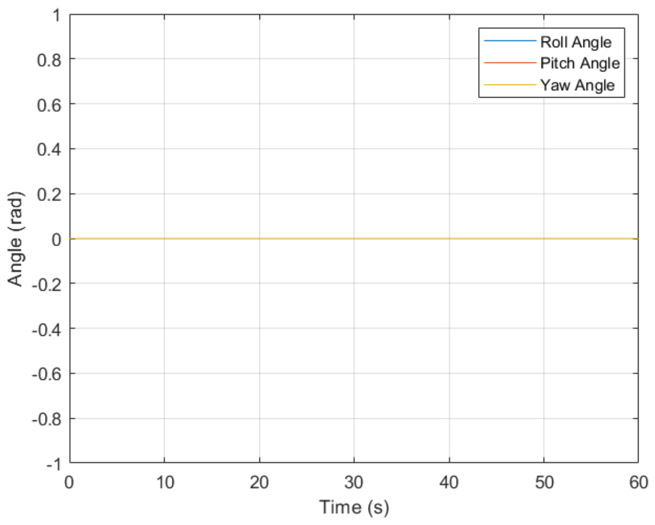

Figure 2: Euler Angles

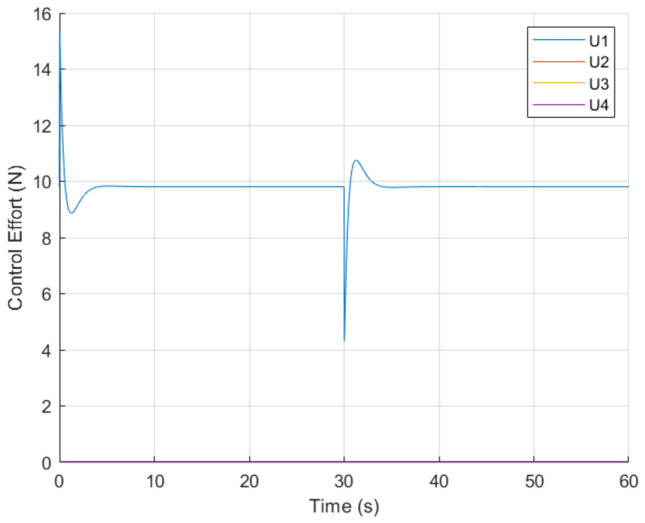

Figure 3: Control Efforts

5.1.2 Low Starting Angle (0.1 rad)

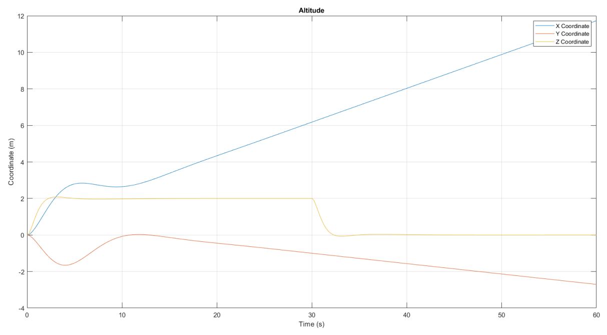

Figure 4: Position Dynamics

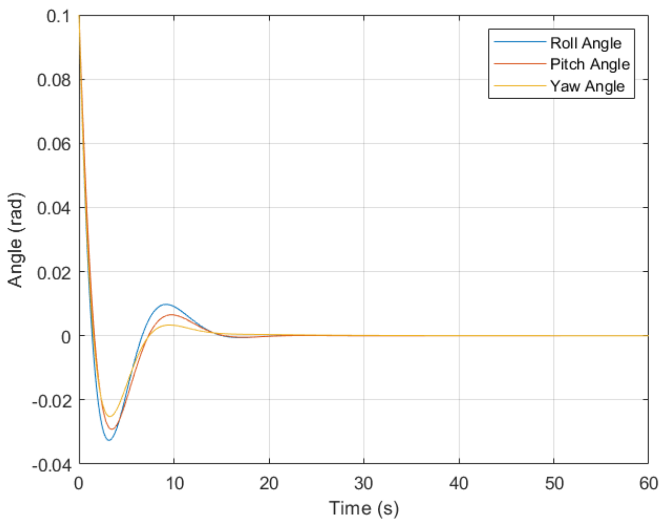

Figure 5: Euler Angles

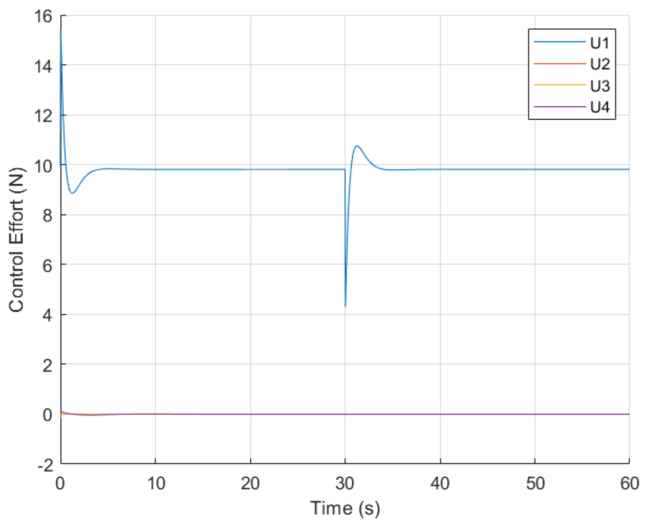

Figure 6: Control Efforts

5.1.3 High Starting Angle (1 rad)

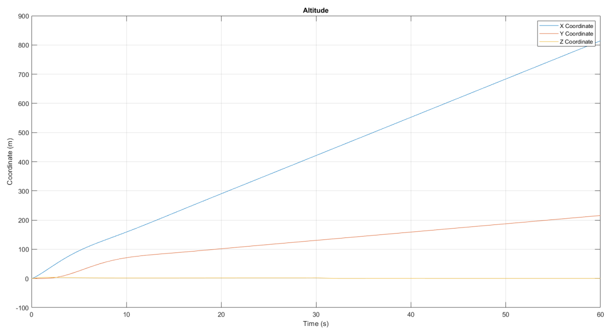

Figure 7: Position Dynamics

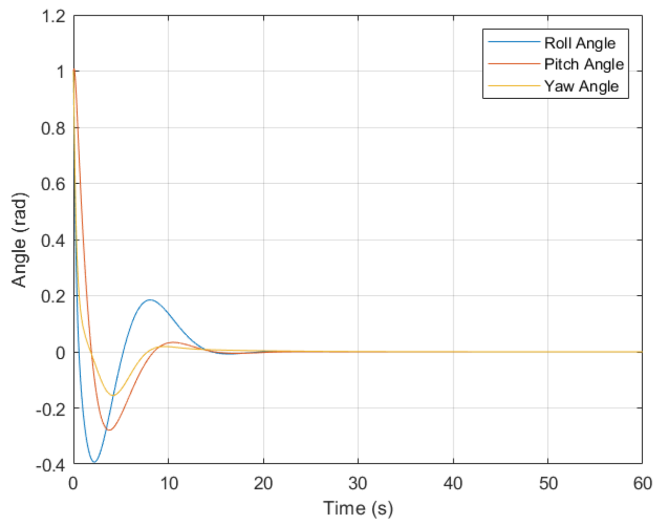

Figure 8: Euler Angles

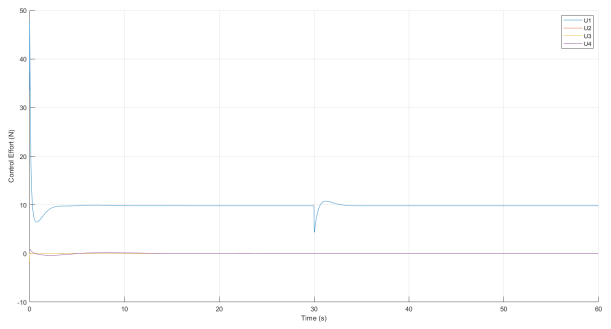

Figure 9: Control Efforts

5.1.4 Trajectory Tracking

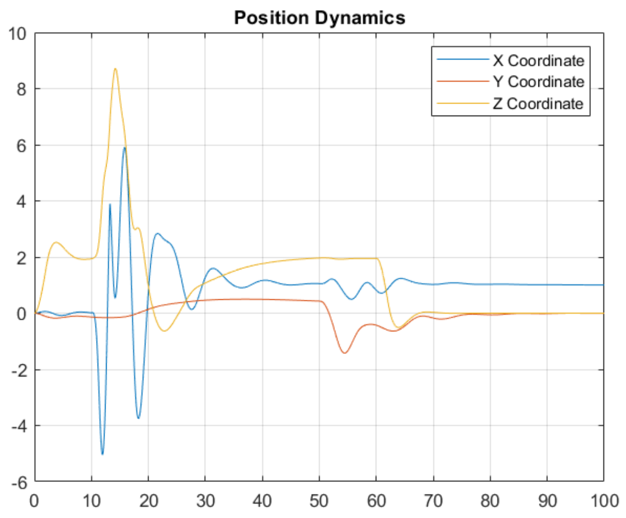

Figure 10: Position Values For Trajectory Tracking

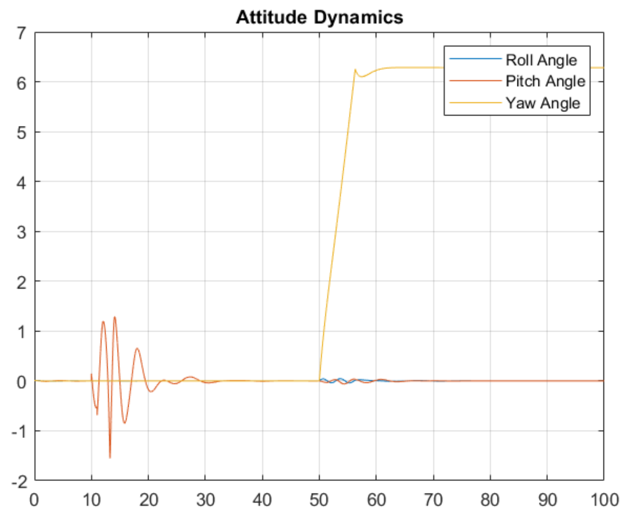

Figure 11: Angle Values For Trajectory Tracking

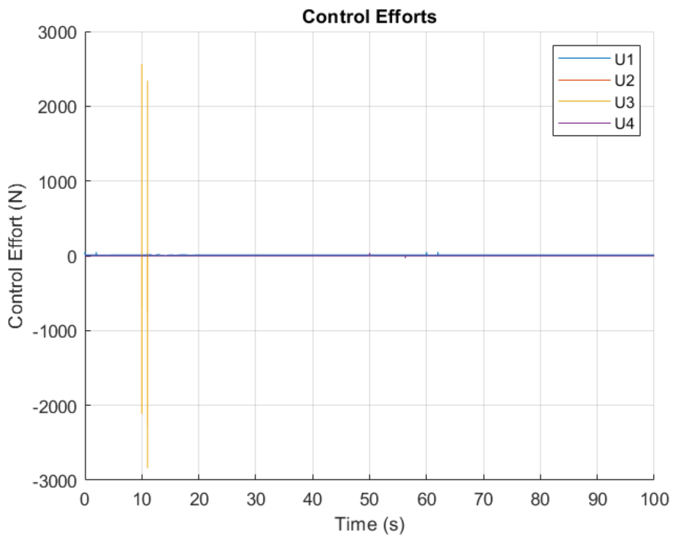

Figure 12: Control Efforts For Trajectory Tracking

6 Appendix (MATLAB Codes And Simulink Diagrams)

6.1 Dynamic Model

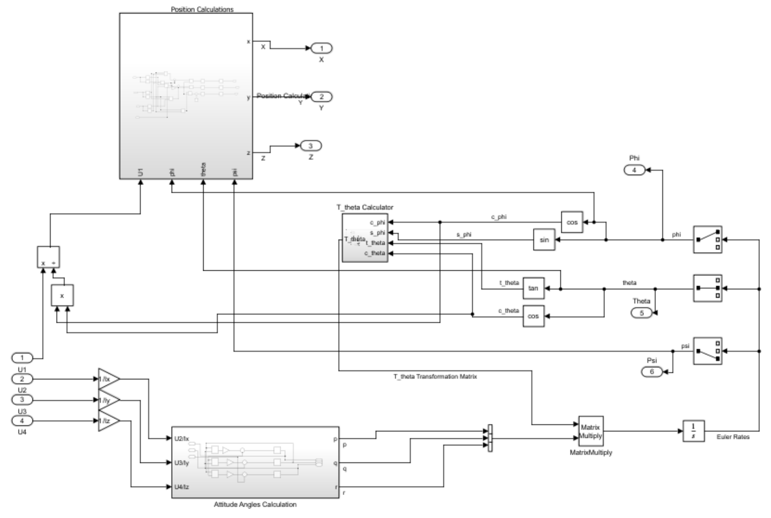

Figure 13: The whole model as a Simulink Model

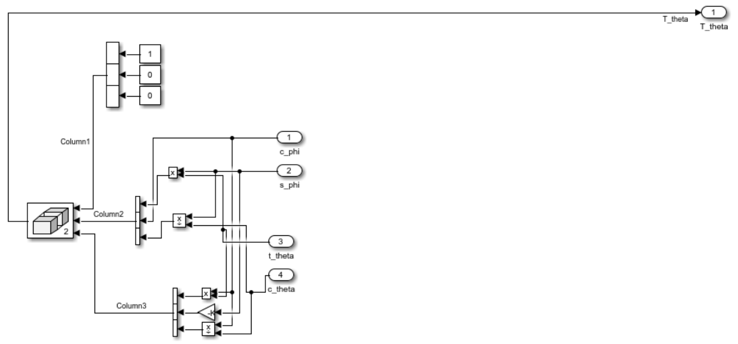

Figure 14: Calculation of the transformation matrix T_θ that is used to convert angular velocities to euler rates

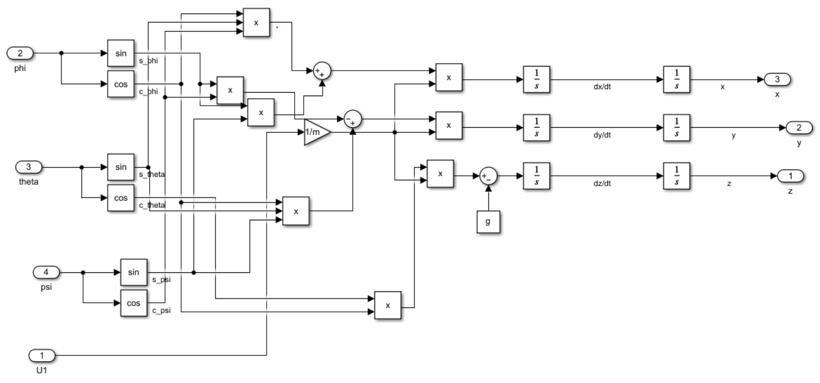

Figure 15: The position calculations of the drone (X,Y,Z)

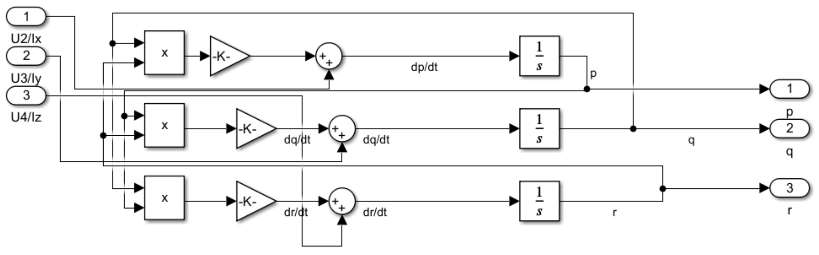

Figure 16: Calculation of angular velocities with inputs U_2, U_3, U_4

6.2 Hover Control

Listing 1: ME425HW3QuadcoptorMotionControl.m. Implementation of Hover control in MATLAB. Here, parameters for simulation and controller gains are defined. The model is then ran and results plotted.

```matlab
clc;clear;close all;
simTime = 60;
%System Parameters
Ix = 0.02;
Iy = 0.02;
Iz = 0.04;
m = 1;
b = 1e-3;
l = 0.35;
d = 1e-4;
g = 9.81;
```

```
%Controller Parameters
% Altitude
P_z = 0.75;
I_z = 0.1;
D_z = 2;

% Roll
P_phi = 0.5;
I_phi   = 0.1;
D_phi   = 0.95;

% Pitch
P_theta = 0.5;
I_theta = 0.1;
D_theta = 0.95;

% Yaw
P_psi = 0.5;
I_psi = 0.05;
D_psi = 0.95;

%Desired And Starting Values
xDesired  = 1;
yDesired = 1;
zDesired = 2;
initialAngles = [0.1 0.1 0.1];

%Running the simulation
out = sim("ME425QuadcoptorMotionControlV4");
%Getting the results
t = out.tout;
X = out.X;
Y = out.Y;
Z = out.Z;
phi = out.roll;
theta = out.pitch;
psi = out.yaw;
U1 = out.U1;
U2 = out.U2;
U3 = out.U3;
U4 = out.U4;

%Plotting
figure;
plot(t,X, 'DisplayName', "X Coordinate");
hold on;
plot(t,Y, 'DisplayName', 'Y␣Coordinate');
plot(t,Z, 'DisplayName', 'Z␣Coordinate');
title("Altitude");
xlabel("Time (s)");
ylabel("Coordinate (m)");
grid on;
legend;

figure;
plot(t, phi,'DisplayName', "Roll Angle");
hold on;
```

```
plot(t, theta,'DisplayName', "Pitch Angle");
plot(t, psi,'DisplayName', "Yaw Angle");
xlabel("Time (s)");
ylabel("Angle (rad)");
grid on;
legend;

figure;
hold on;
grid on;
xlabel("Time (s)");
ylabel("Control Effort (N)");
legend;
plot(t,U1,'DisplayName',"U1");
plot(t,U2,'DisplayName',"U2");
plot(t,U3,'DisplayName',"U3");
plot(t,U4,'DisplayName',"U4");
```

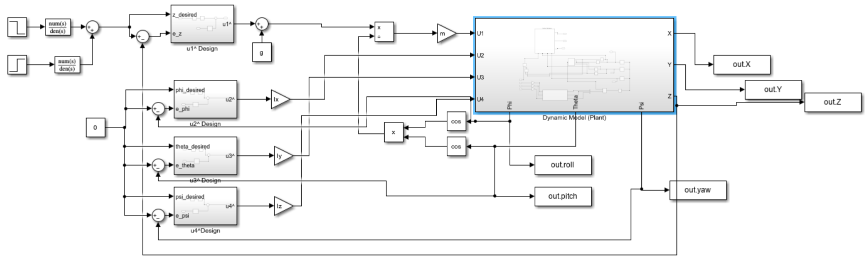

Figure 17: The whole model with the plant and also the FF+FB designed control inputs

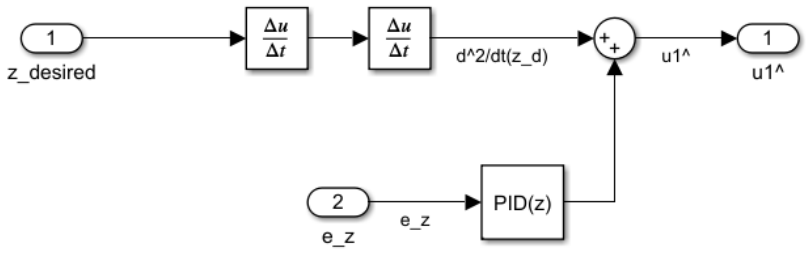

Figure 18: How the controllers for inputs are designed. In this figure, the design is only showed for U1, but the process is similar for the other inputs as well

6.3 Trajectory Tracking

Listing 2: ME425HW3QuadcoptorTrajectoryTrack.m. Implementation of trajectory tracking in MATLAB. Here, parameters for simulation and controller gains are defined. The model is then ran and results plotted.

```
clc;clear;close all;
simTime = 100;
%System Parameters
Ix = 0.02;
Iy = 0.02;
Iz = 0.04;
m = 1;
b = 1e-3;
```

```
l = 0.35;
d = 1e-4;
g = 9.81;

%Controller Parameters
%X Coord
P_x = 0.75;
D_x = 1.5;
I_x = 0.075;

%Y Coord
P_y = 0.75;
D_y = 1.1;
I_y = 0.075;

% Altitude
P_z = 0.75;
D_z = 1.1;
I_z = 0.085;

% Roll
P_phi = 0.75;
I_phi   = 0.075;
D_phi   = 0.6;

% Pitch
P_theta = 0.75;
I_theta = 0.075;
D_theta = 0.6;

% Yaw
P_psi = 0.8;
I_psi = 0.05;
D_psi = 1.5;

%Desired And Starting Values
xDesired  = 1;  %Will Be Applied After the robot hovers
yDesired = 0;
zDesired = 2;
psiAngle = 0;
initialAngles = [0.01 0.01 0.01];

%Running the simulation
out = sim("ME425QuadcoptorTrajectoryTrackingV1.slx");
%Getting the results
t = out.tout;
X = out.X;
Y = out.Y;
Z = out.Z;
phi = out.roll;
theta = out.pitch;
psi = out.yaw;
U1 = out.U1;
U2 = out.U2;
U3 = out.U3;
U4 = out.U4;

```

```
%Plotting
figure;
plot(t,X, 'DisplayName', "X Coordinate");
hold on;
plot(t,Y, 'DisplayName', 'Y␣Coordinate');
plot(t,Z, 'DisplayName', 'Z␣Coordinate');
title("Position Dynamics");
grid on;
legend;

figure;
plot(t, phi,'DisplayName', "Roll Angle");
hold on;
plot(t, theta,'DisplayName', "Pitch Angle");
plot(t, psi,'DisplayName', "Yaw Angle");
grid on;
title("Attitude Dynamics");
legend;

figure;
hold on;
grid on;
xlabel("Time (s)");
ylabel("Control Effort (N)");
title("Control Efforts");
legend;
plot(t,U1,'DisplayName',"U1");
plot(t,U2,'DisplayName',"U2");
plot(t,U3,'DisplayName',"U3");
plot(t,U4,'DisplayName',"U4");
```

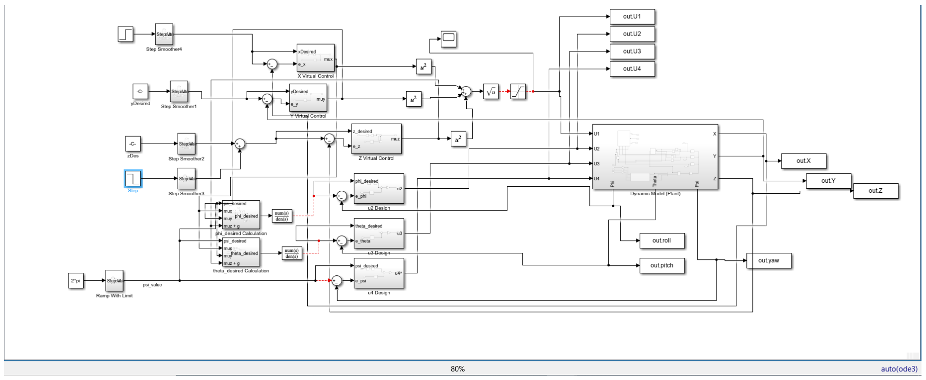

Figure 19: The entire model for trajectory tracking

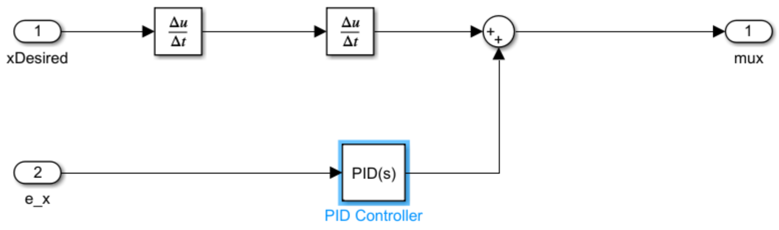

Figure 20: How the virtual control for accelerations are designed. Y and Z are designed similarly as well.

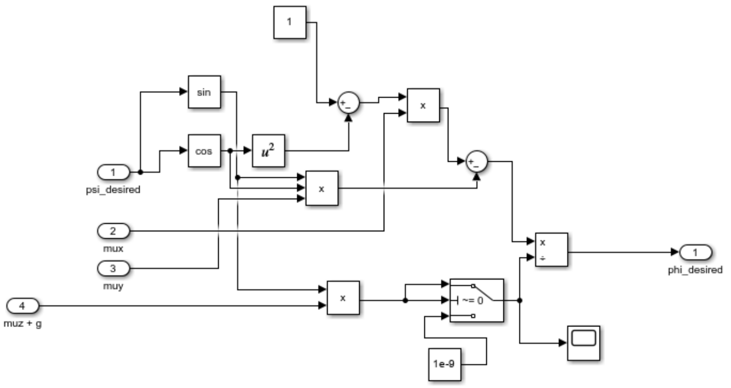

Figure 21: The design process of ϕ_d. The design was made after solving the three equations for the angle values.

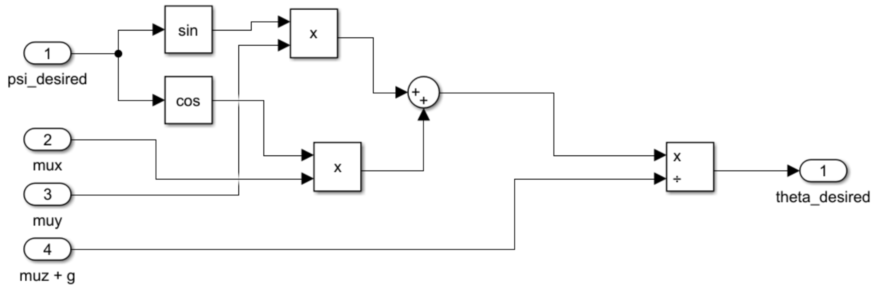

Figure 22: The design process of θ_d. The design was made after solving the three equations for the angle values.