In this assignment, we take a closer look at the quadcopter's mathematical model. With the aid of simulink, we analyze the forces and moments acting on the quadcopter as well as inputs that allow us to control the dynamics of the quadcopter.

The following report is structured as follows: section 1 introduces the notations and mathematical backgrounds for the input; section 2 introduces the dynamics equations that govern the motion of the quadcopter. In section 3, we use the equations from section 2 to build an open-loop simulation of the drone, and analyze the results.

1 Mathematical Definition, Drone Dynamics Equations

1.1 States, and Variables

There are a total of 12 variables involved our differential equations.

- P describes the position of the drone.
- V describes the velocity of the drone.
- $\omega_{b/e}$ describes the rotational rate of the drone between Earth Frame and Body Frame
- $\dot{\omega_{b/e}}$ describes the angular(rotational) acceleration of the drone
- Superscript ^{b} or ^{e} denotes the corresponding frame the state variable is in
- Each variable can be interpreted as a vector, with x, y, and z components aligning with the body x, y and z or earth NED axes.

In addition to the state variables, we also use the following variables to describe the dynamics

- F describes the forces experienced by the drone. The subscripts describe the specific types
- M describe the moments experienced by the drone. The subscripts describe the nature of the force causing the moments.

Four combinations of forces and moments enable us to control the drone's dynamics. These four combinations are:

- Thrust, denoted by T
- Elevator, denoted by E
- Aileron, denoted by A
- Rudder, denoted by R

1.2 Rotation Matrices

In order to translate variables between body and earth(NED) frames, we use three different 3-D rotation matrices with corresponding rotational angles. Following the convention from our last assignment, we use the following three angles:

- Angle ψ describes the rotation of the $x-y$ plane around the $z-axis$. (yaw angle)
- Angle ϕ describes the rotation of the $y-z$ plane around the $x-axis$. (roll angle)
- Angle θ describes the rotation of the $x-z$ plane around the $y-axis$. (pitch angle)

Therefore, the three rotation matrices translate vectors in Earth(NED) frame to the body frame.

- $R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$
- $R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$
- $R(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

When translating from earth to body frame, the matrices are applied in the *yaw-roll-pitch* order. For example:

$$V^b = R(\phi)R(\theta)R(\psi)V^e \tag{1}$$

To translate vectors from body to earth frame, we can simply multiple the vector in body frame with the right inverse of all three rotation matrix (in *pitch-roll-yaw* order). Since all rotation matrices are orthonormal, their transpose matrices are also their inverse.

$$V^e = R(\psi)^T R(\theta)^T R(\phi)^T V^b \tag{2}$$

2 Drone Dynamics

A total of 4 equations model the dynamics of the quadcopter. Each subsection will describe one equation.

2.1 Position - Velocity

$$\dot{p}^E = V^E \tag{3}$$

This differential equation encapsulates the relationship between the drone's position and velocity in the earth frame. Note:

$$\dot{p}^E = R(\psi)^T R(\theta)^T R(\phi)^T V^b \tag{4}$$

2.2 Rotational Rate - Rotation

$$\dot{\Theta} = H(\Theta)^{-1}\omega_{b/E} \tag{5}$$

This differential equation shows the relationship between the drone's attitude angles and the respective rotational rates. It's important to note that $H(\Theta)^{-1}$ denotes the rotational matrix that translates the rates of euler angles into gyroscopic rotational rate measurements.

$$H(\Theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{6}$$

2.3 Newton's Second Law: Conservation of Forces

$$\dot{V}^b = \frac{1}{m}(F_T^b + F_A^b + F_G^b - \omega_{b/E} \times mV^b) \tag{7}$$

This equation sums up the relationship between the net force acting on the drone and the total translational acceleration of the drone. The equation contains three major forces, thrust (F_T^b), aerodynamic force (F_A^b), and gravity (F_G^b), as well as counter forces generated by speed and rotation.

It is important to note that these forces are all in body frame, so necessary translations are required during modelling. For example, in the earth frame, F_G has zero x and y components. Depending on the attitude of the drone, F_G^b can have non-zero x and y components.

To begin with, gravity in the body frame is calculated by:

$$F_G^b = R(\phi)R(\theta)R(\psi)F_G^E \tag{8}$$

where $R(\phi)$, $R(\theta)$, and $R(\psi)$ are previously defined rotation matrices.

The aerodynamic thrust F_T^b is calculated by:

$$F_T = C_T \rho n^2 D^4 \tag{9}$$

where n denotes motor speed in Hz, and D denotes the diameter of the propeller.

Finally, for our net aerodynamic force, we use a lumped aerodynamic model for all aerodynamic forces experienced by the drone:

$$F_A^b = -\rho C_T D^4 (n_1^2 + n_2^2 + n_3^2 + n_4^2) \begin{bmatrix} K_c & 0 & 0 \\ 0 & K_c & 0 \\ 0 & 0 & 0 \end{bmatrix} V^b \tag{10}$$

2.4 Conservation of Momentum

$$\dot{\omega}_{b/E} = J^{-1}(M_T^b + M_A^b + M_\Omega^b - \omega_{b/E} \times J\omega_{b/E}) \tag{11}$$

This differential equation sums up the net moment and torque experienced by the drone in relation to the angular acceleration of the drone. The thrust moment, M_T^b, is calculated by multiplying the thrust force with the respective lever arm length, i.e the length from the motor to the drone's center of mass.

$$M_T^b = F_T^b \cdot l \tag{12}$$

Since thrust does not directly affect torque in the z direction, we calculate the moment along the z axis by using the power from the propellers. Thrust power from the propellers is calculated by the following equation:

$$P = C_p \rho n^3 D^5 \tag{13}$$

We can thus calculate the moment generated by the thrust, since torque is proportional to power

$$P = \tau\omega, \omega = 2\pi n \tag{14}$$

In this assignment, we assume that there is no aerodynamic moments (e.g sideslip, drag), so

$$M_A^b = 0 \tag{15}$$

The last component to the net moment, M_Ω^b, is generated by the angular momentum of the propellers. It's calculated as follows:

$$M_{\Omega}^{b} = \omega_{b/e} \times \begin{bmatrix} 0 \\ 0 \\ J_p 2\pi(n_1 - n_2 + n_3 - n_4) \end{bmatrix} \tag{16}$$

where J_p is the moment of inertia of the propeller. Assuming our propeller is a cylinder and the spinning axis is located at the center of it's height, J_p is calculated as follows:

$$J_p = \frac{1}{12} m l^2 \tag{17}$$

where m is the mass of the propeller, and l is the length(height) of the propeller.

2.5 Mixer

For a quadcopter, the attitude control of the aircraft is done via adjusting the rotor speeds. By "mixing" different motor speeds, we can achieve thrust, aileron (rolling), elevator (pitching) and rudder (yawing) motion. Given a "x-quadcopter" motor layout, we label the motors in the following way:

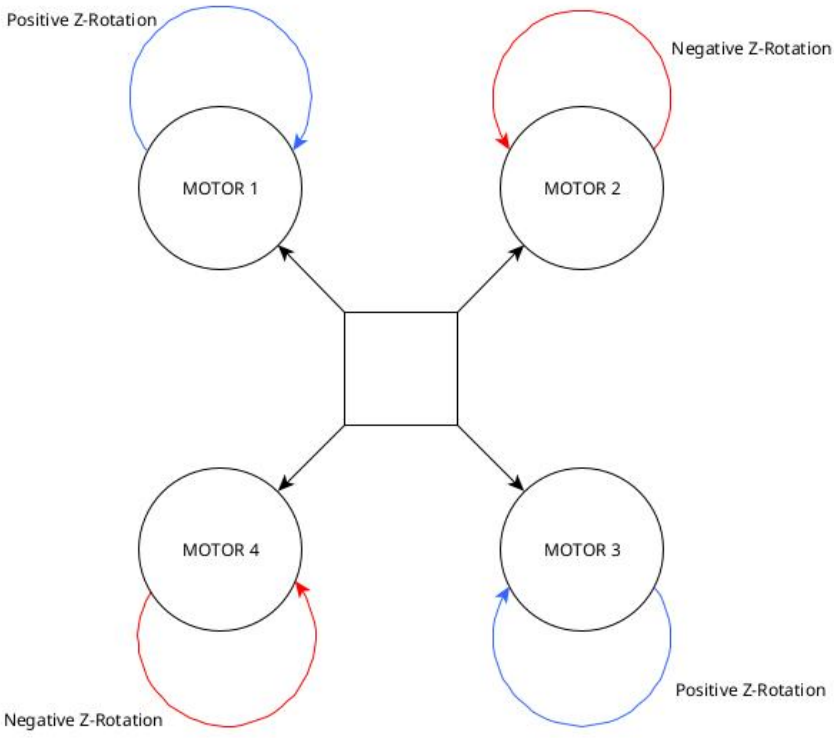

Figure 1: Motor Labelling

We can calculate the output thrust force, rolling moment, pitching moment and rudder moment with the following equation:

$$\begin{bmatrix} F_T \\ A_M \\ E_M \\ R_M \end{bmatrix} = \begin{bmatrix} a(n_1^2 + n_2^2 + n_3^2 + n_4^2) \\ b(n_1^2 - n_2^2 - n_3^2 + n_4^2) \\ c(n_1^2 + n_2^2 - n_3^2 - n_4^2) \\ d(-n_1^2 + n_2^2 - n_3^2 + n_4^2) \end{bmatrix} \tag{18}$$

where

$$a = C_T \rho D^4, b = a * l_x, c = a * l_y, d = \frac{C_P \rho D^5}{2\pi} \tag{19}$$

We can manipulate (14) into the product of two matrix and a motor speed vector as follows: we first define four new variables, v_1, v_2, v_3 and v_4 as the square of motor speeds n_1, n_2, n_3 and n_4

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} n_1^2 \\ n_2^2 \\ n_3^2 \\ n_4^2 \end{bmatrix} \tag{20}$$

Therefore

$$\begin{bmatrix} T \\ A \\ E \\ R \end{bmatrix} = \begin{bmatrix} a(n_1^2 + n_2^2 + n_3^2 + n_4^2) \\ b(n_1^2 - n_2^2 - n_3^2 + n_4^2) \\ c(n_1^2 + n_2^2 - n_3^2 - n_4^2) \\ d(-n_1^2 + n_2^2 - n_3^2 + n_4^2) \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{21}$$

$$= CM \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{22}$$

$$\therefore \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = M^{-1}C^{-1} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & -0.25 \\ 0.25 & -0.25 & 0.25 & 0.25 \\ 0.25 & -0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & -0.25 & 0.25 \end{bmatrix} \begin{bmatrix} \frac{1}{a} & 0 & 0 & 0 \\ 0 & \frac{1}{b} & 0 & 0 \\ 0 & 0 & \frac{1}{c} & 0 \\ 0 & 0 & 0 & \frac{1}{d} \end{bmatrix} \begin{bmatrix} T \\ A \\ E \\ R \end{bmatrix} \tag{23}$$

The product of matrices $M^{-1}C^{-1}$ is therefore used as our mixer, which takes in desired control values and outputs respective motor speeds that produce those values.

3 Simulation

3.1 MATLAB Script

The MATLAB script for simulation is listed below:

Listing 1: Simulation Script

```
%% ESE 4481 Homework 3
% By Will Wu
clear, clc, close all

%% Constant Declaration:
m = 0.068;
Jxx = 0.69e-4;
Jyy = 0.775e-4;
Jzz = 1.5e-4;
g = 9.8;
motor_num = 4;

J = [Jxx 0 0; 0 Jyy 0; 0 0 Jzz];

F_b = zeros;
M_b = zeros;

x_dist = 0.047625; % prop to com x distance
y_dist = 0.047625; % prop to com y distance, in meters

prop_mass = 0.001; % in kg
prop_diam = 0.066; % meters
max_rpm = 29000; % rpm
max_rev = max_rpm/60;

air_density = 1.225; % kg/m^3
c_p = 0.041; % power coefficient, assume flat, constant coeff of power



%% Curve fit estimating thrust coefficient
rev = [11000 19000 23000 25000 26500 29000]'/60;
c_t = [0.08 0.083 0.087 0.09 0.091 0.093]';

% regressor matrix
regressor = cat(2, ones(size(rev)), rev);
k_estimate = (regressor' * regressor) \ regressor' * c_t;
offset = k_estimate(1);
slope = k_estimate(2);

residual = norm(c_t - regressor*k_estimate);
residual_cons = norm(c_t - 0.087*ones(size(c_t)));

% examine error
figure, hold on;
plot(rev, c_t, '*');
plot(rev, regressor*k_estimate);
legend('Real', 'Estimate');

%% Equilibrium Motor Speed, prop moment of inertia

thrust_eq = @(x) m*g/motor_num - (slope*x+offset)*air_density*x^2*...
    prop_diam^4;

```

```
motor_n_eq = fzero(@(x) thrust_eq(x), 300);
motor_omega_eq = motor_n_eq * 2 * pi;

% assuming a spinning cylinder about mid point
prop_j = 1/12*prop_mass*prop_diam^2;

%% Mixer Parameters

% sums the motor speeds into T, L, M, N
dynamics_matrix = [1 1 1 1;
                   1 -1 -1 1;
                   1 1 -1 -1;
                   -1 1 -1 1];

mixer_matrix = inv(dynamics_matrix);

%% Lumped aerodynamics model

K_c = 0.22;
lumped_matrix = [K_c 0 0; 0 K_c 0; 0 0 0];

%% Gyroscopic forces and moments

motor_mix_vector = [1 -1 1 -1];
m_b_a = [0;0;0];

%% Process Simulation
```

We used the least-squared method to attempt a curve fit when calculating the thrust coefficient. From the wind-tunnel data, we hypothesize that there is a linear relationship between motor speed and the coefficient of thrust C_T. Using the least square method, we are able to approximate a slope k and offset b that describes the linear relationship between motor speed and C_T:

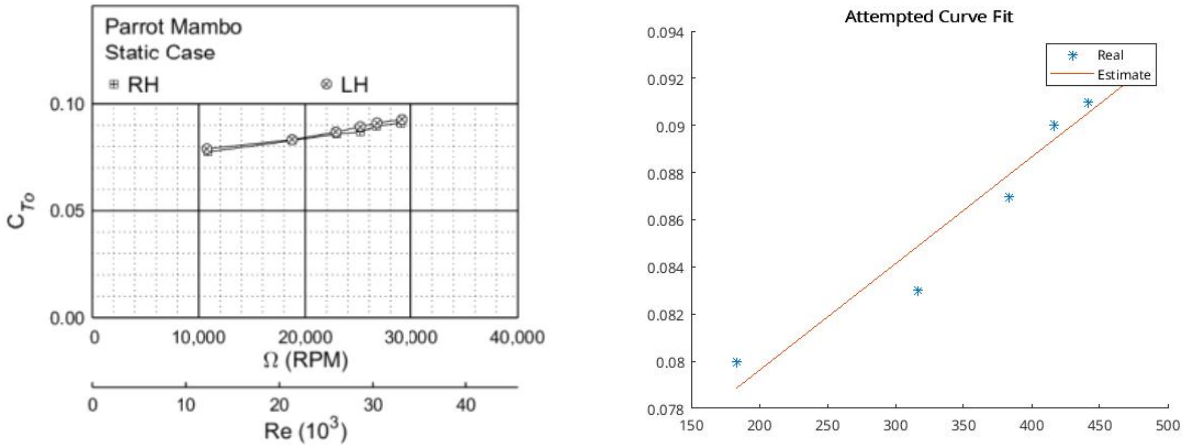

Figure 2: Aerodynamics Data vs Attempted Curve Fit

We also calculated the statistical residual of the curve fit (the total squared difference between data and fit), and obtained a residual of 0.0026. If we assume C_T remains constant,

we get a residual value of 0.0112. Thus, we consider our curve fit to be effective.

3.2 Simulink Simulation Results

3.2.1 Equilibrium Simulation

We first simulate the drone's dynamic response under equilibrium conditions. The initial conditions of the integrators are all set to zero (altitude under hover condition can be arbitrary). The input through the mixers are set to all zero except for throttle, which is set to equal the drone's own weight. We obtain the following results:

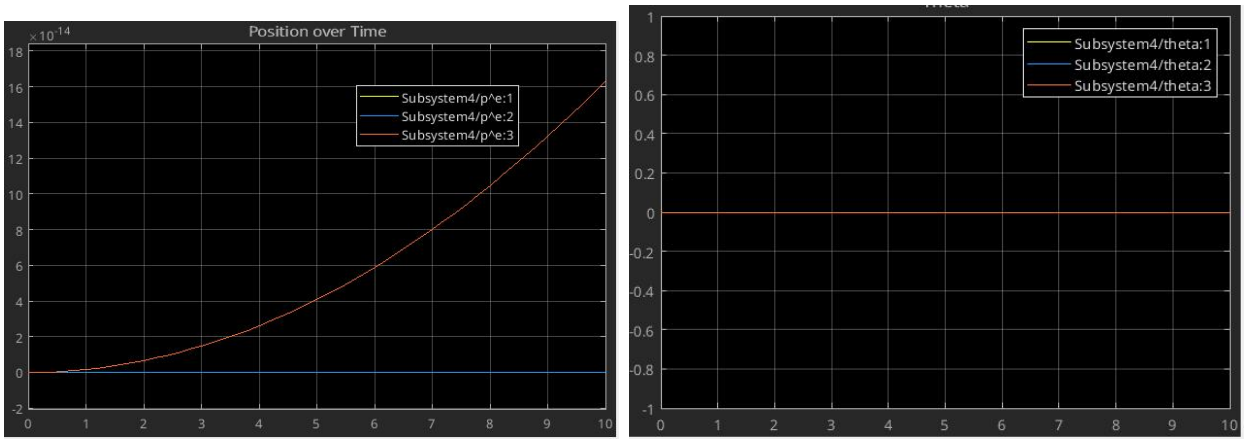

Figure 3: Translational Response of thrust input (Left: Position; Right: Velocity)

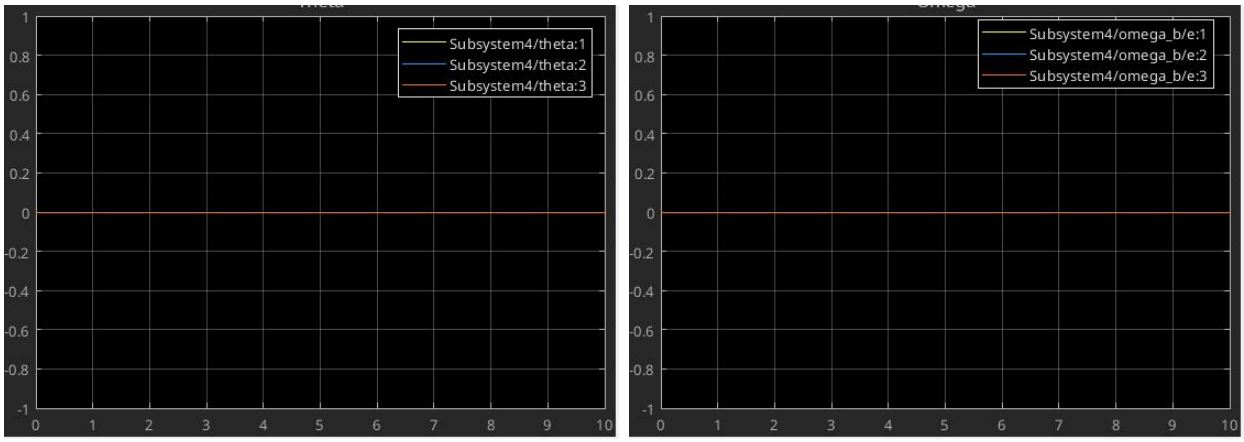

Figure 4: Rotational Response of thrust input (Left: Rotation; Right: Rotational Rate)

Note that even though the position and velocity graphs in Figure 3 display non-flat results, their magnitudes are extremely small (10 to the 14^{th} power). We believe that the non-zero

magnitude is resulted from residual/noise from the numerical differential equation solver. We therefore verify that at equilibrium, our simulation is accurate.

3.2.2 Small Deltas around Equilibrium

We test our simulation by inputting small delta inputs around the equilibrium inputs. The following graphs are simulation results of inputting a step signal from 0.1 to 0.2s into thrust, aileron and elevator, respectively.

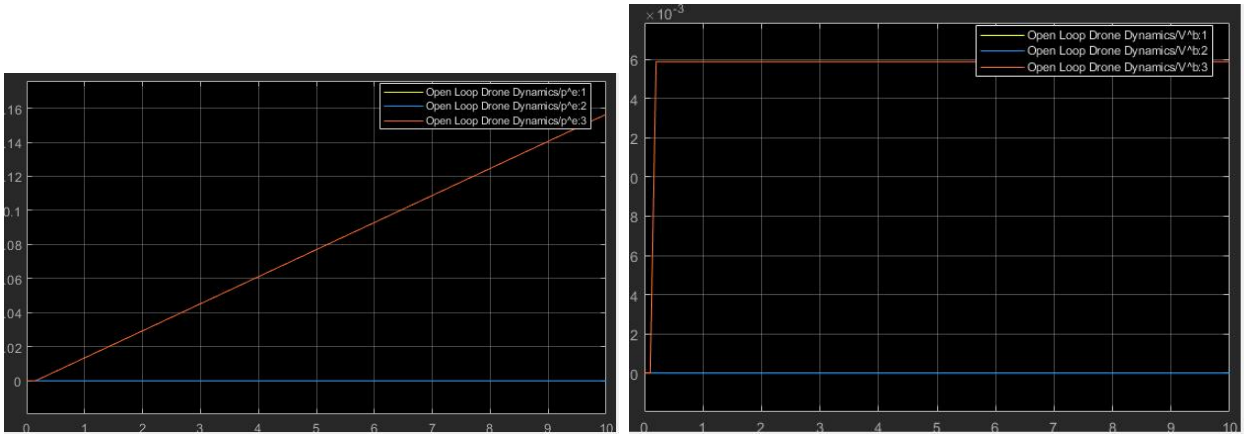

Figure 5: Translational Response of thrust input (Left: Position; Right: Velocity)

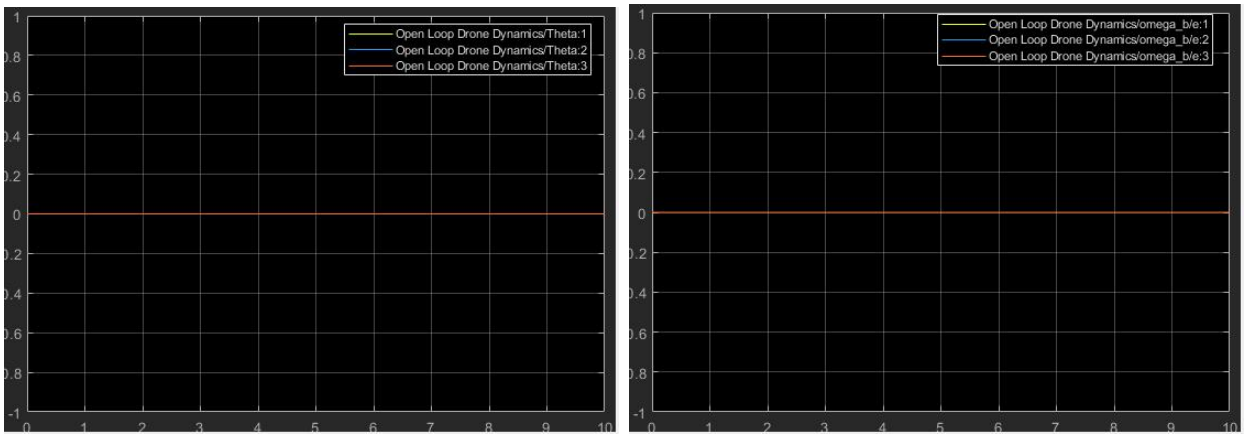

Figure 6: Rotational Response of thrust input (Left: Rotation; Right: Rotational Rate)

As we can see from the two sets of graphs, a sudden jerk in thrust input causes the drone to move upward. Due to the lack of air friction in the model, the drone continues upward with a linear, constant-speed motion. This confirms with our previous simulation.

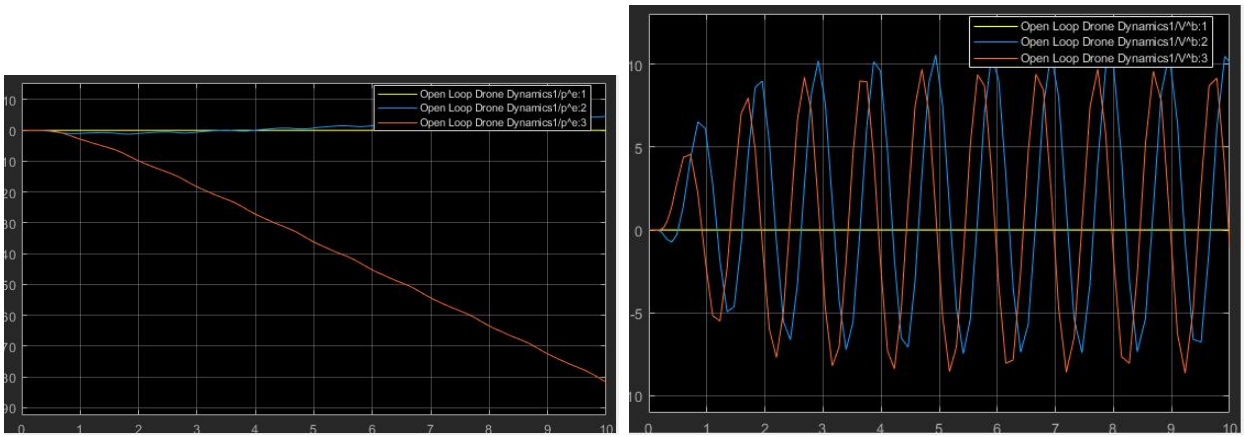

Figure 7: Translational Response of aileron input (Left: Position; Right: Velocity)

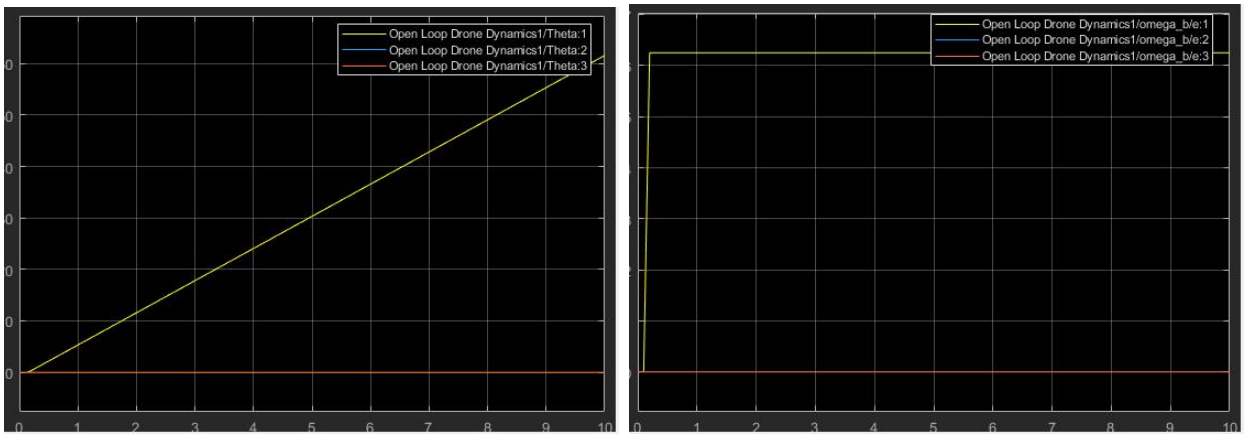

Figure 8: Rotational Response of aileron input (Left: Rotation; Right: Rotational Rate)

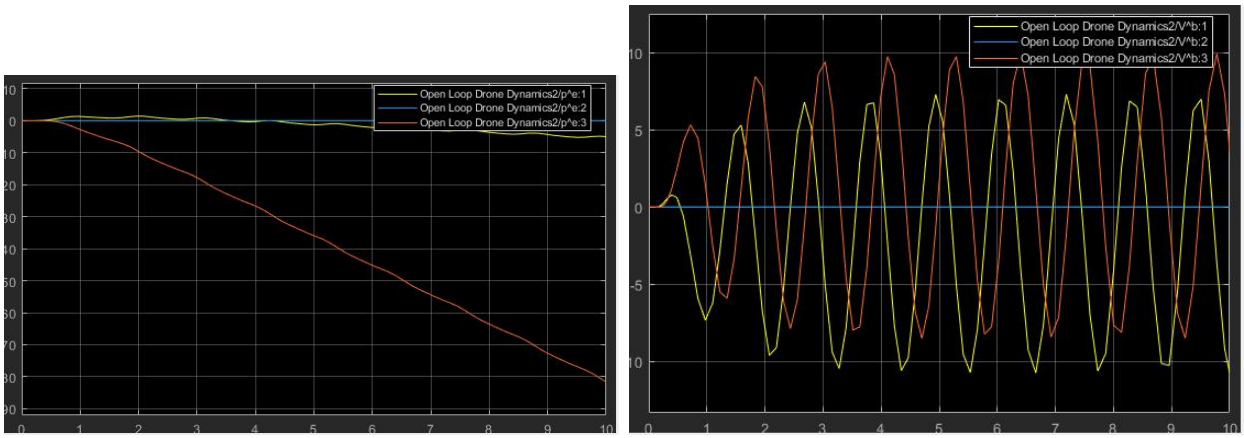

Figure 9: Translational Response of elevator input (Left: Position; Right: Velocity)

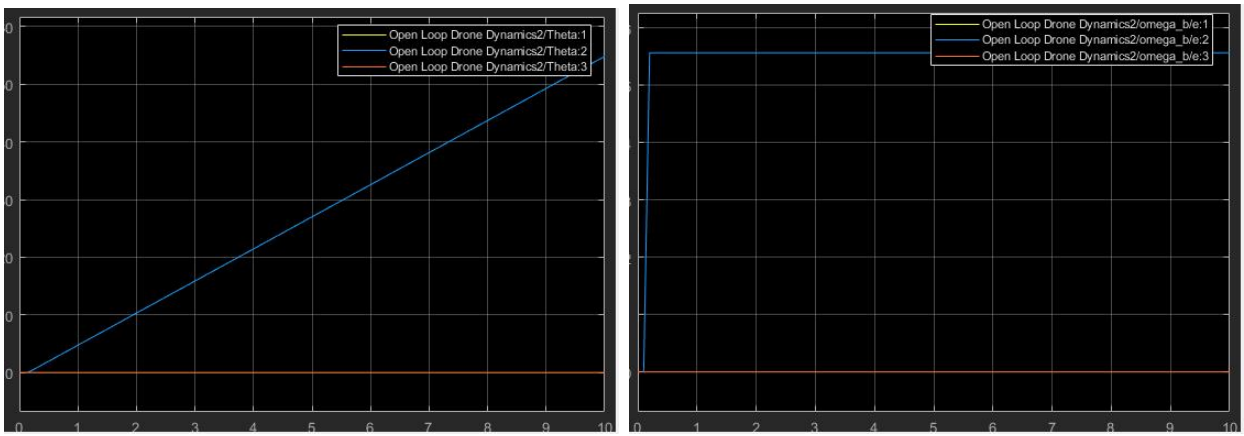

Figure 10: Rotational Response of elevator input (Left: Rotation; Right: Rotational Rate)

Our aileron and elevator input simulations show more details compared to previous simulations. A jerk in aileron and elevator result in changes in roll/pitch angle respectively, which result in changes in psi/theta rotation and changes in the x/y direction. The drone starts moving because of the attitude change. What our simulation also showed is that the drone losses altitude. After the drone rotates, thrust forces produced by the motors no longer aligns with the z-axis in earth frame. The original equilibrium input no longer suffices the equilibrium condition (the force vectors now have non-zero horizontal components, resulting in decreases in vertical components that counters gravity). Hence, we can see a decrease of altitude in the position graphs.

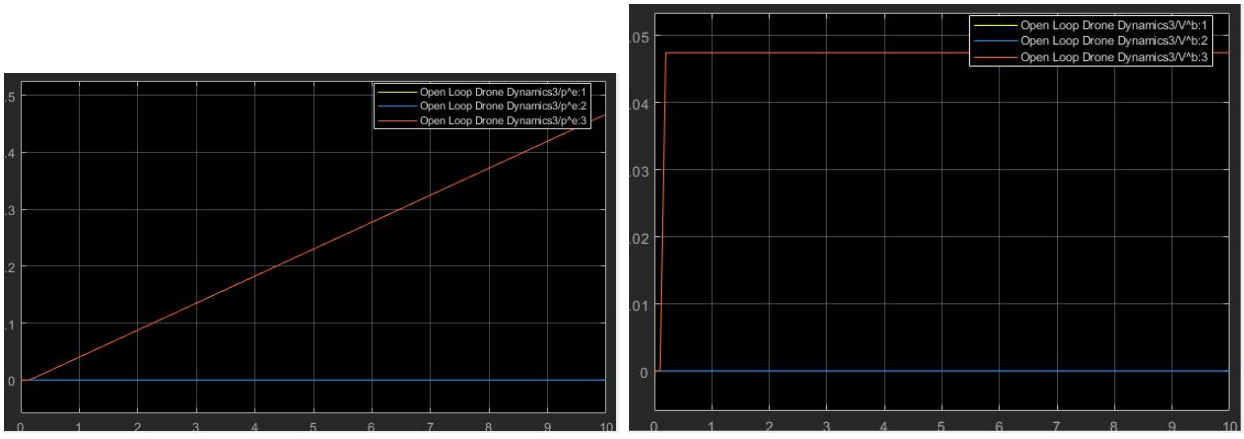

Figure 11: Translational Response of elevator input (Left: Position; Right: Velocity)

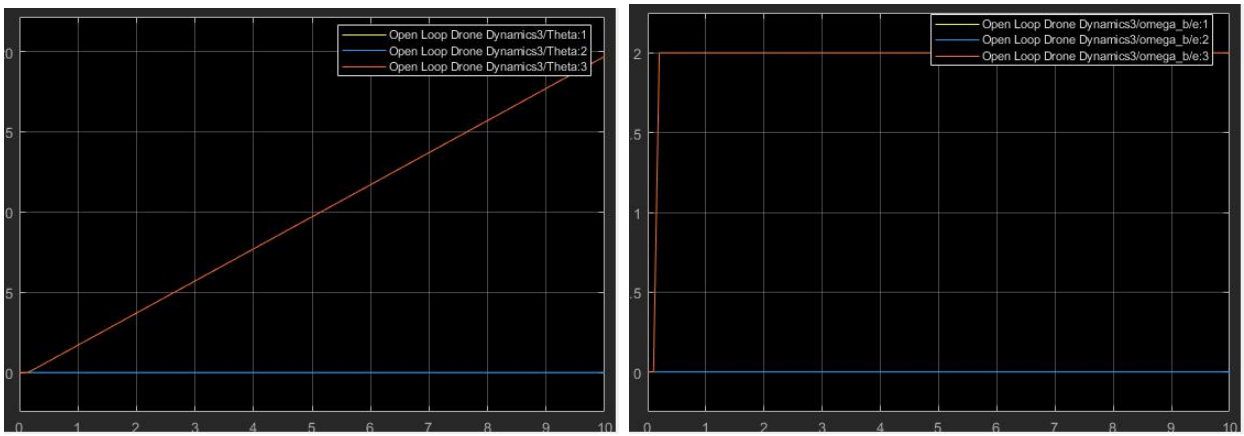

Figure 12: Rotational Response of elevator input (Left: Rotation; Right: Rotational Rate)

There are two important conclusions to be drawn *Figure 11* and *Figure 12*. First, the simulation results for θ and ω conform with our previous simulations. As we can see on from the graph, the jerk in rudder input produces a constant rotation along the $z - axis$. The drone remains constant angular velocity after the jerk in rudder input. The constant angular velocity keeps the drone rotating. Second, our simulations shows a jerk in vertical velocity and increase in altitude. This result may seem bizarre at first, since only thrust input can produce changes in altitude. Examining our mixer control signal (shown below) yields an explanation.

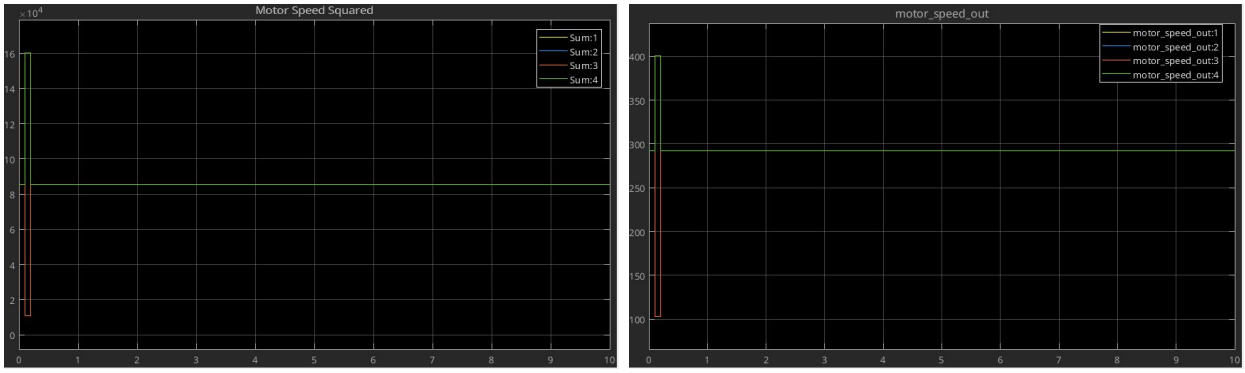

Figure 13: Mixer Signal Compared: Squared Motor Speed vs Motor Speed (Hz)

As shown by *Figure 13*, the mixer correctly calculates the gain/loss in motor speed (squared) required to produce the desired yawing moment delta around the equilibrium input. Although the squared motor-speed deltas share the same magnitude (absolutely value), their square rooted counterparts do not. Hence, the squared-rooted deltas produce an uneven gain/loss around the equilibrium input. In addition, since we have hypothesized a linear relationship between motor speed and C_T, our coefficient of thrust, the uneven gain/loss of motor speeds results in uneven thrust gain.

References

[1] Aircraft Control and Simulation, Stevens, Lewis & Johnson

[2] Homework 2 Report

[3] Lecture Slides