UNIVERSITY OF COLORADO - BOULDER

ASEN 3801

DECEMBER 1, 2023

ASEN 3801 Lab 4 - Quadrotor Simulation and Control

Author: *Race Pohlkamp*

Author: *Hayden Gebhardt*

Author: *Hector Rubi*

Author: *Manas Katragadda*

This lab involves analysis of quadrotor aircraft dynamics through the simulation of equations of motion (EOM). Specifically, the study focuses on the concepts of trim and static stability, exploring their implications in quadrotor flight dynamics. The experiment extends to both nonlinear and linearized quadrotor EOM, with thorough simulations and comparisons to explore the behavior of these systems.

In this assignment, we employ modal response theory to design and analyze closed-loop control laws tailored for the Parrot Mambo mini-drone. Modal analysis serves as the theoretical underpinning for deriving state space models resulting from these control laws. The ensuing analysis involves a wide variety of model responses by examining the eigenvalues and eigenvectors of resulting matrices, providing a nuanced understanding of the stability and dynamics of the quadrotor under various control scenarios.

Ann and H.J. Smead
Aerospace Engineering Sciences
UNIVERSITY OF COLORADO **BOULDER**

Contents

I. Task 1

A. 1.2 Plots

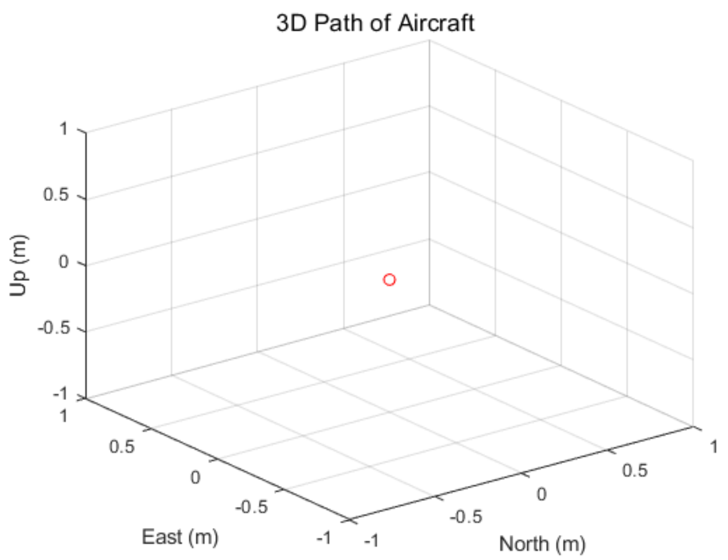


Fig. 1 3D Flight Path in Steady Hover

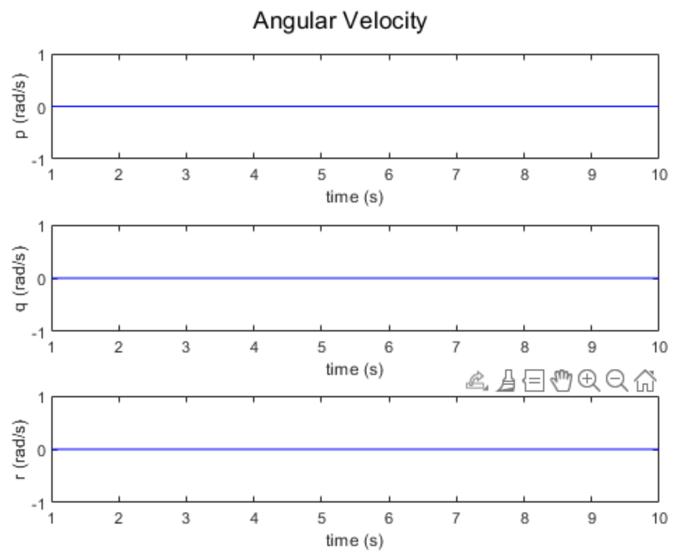


Fig. 2 Angular Velocity in Steady Hover

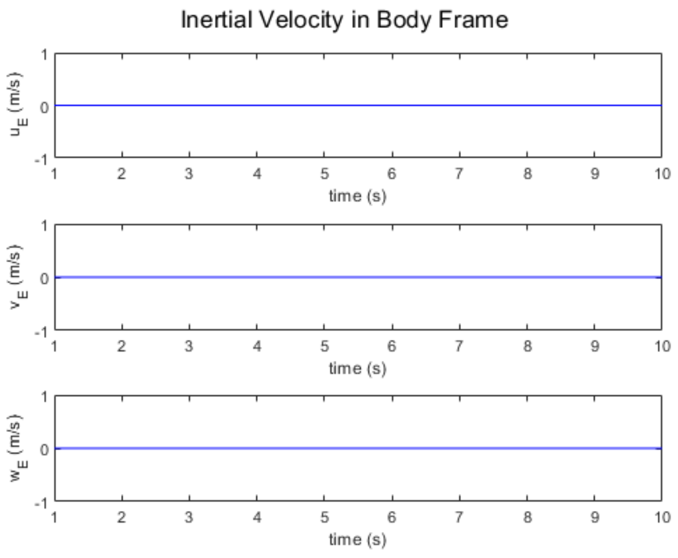


Fig. 3 Inertial Velocity in Steady Hover

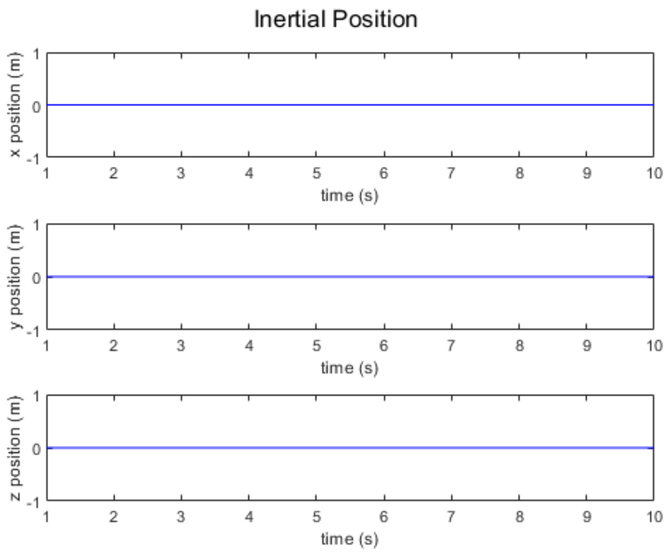


Fig. 4 Inertial Position in Steady Hover

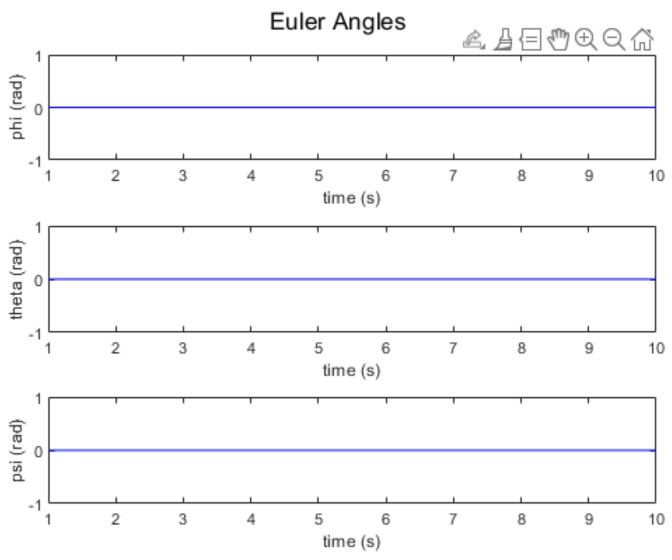


Fig. 5 Euler Angles in Steady Hover

For this task, we determined the trim thrusts for each of the four motors to be $\frac{1}{4}$th the mass of the drone. Below this is our work showing how we came to this result.

$$
\begin{aligned}
Z_c &= -\sum_{i=1}^{4} f_i = m \\
Z_c &= -(f_1 + f_2 + f_3 + f_4) \\
m &= (f_1 + f_2 + f_3 + f_4) \\
\therefore \frac{1}{4}m &= f_i
\end{aligned}
$$

B. 1.3 Plots

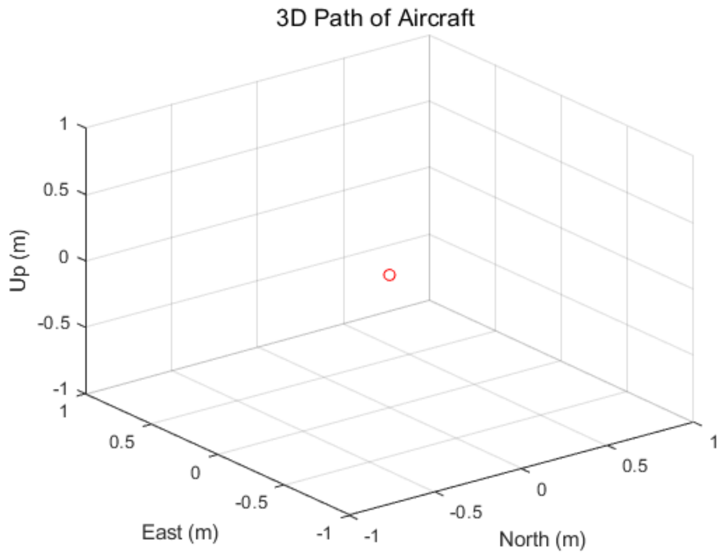


Fig. 6 3D Flight Path in Steady Hover

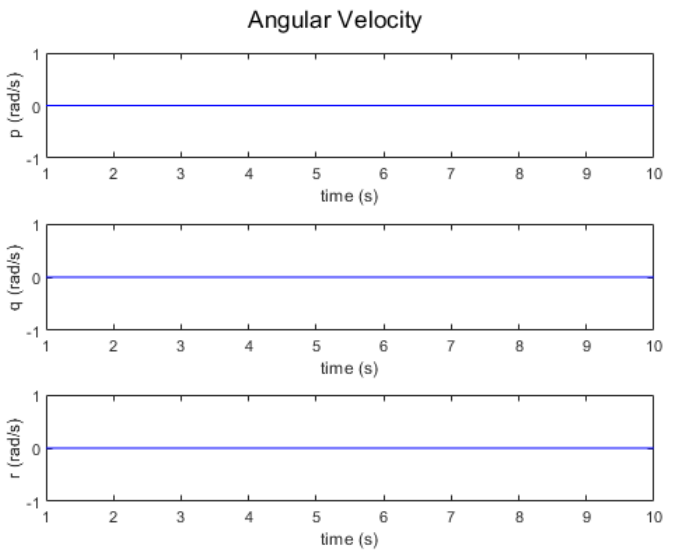


Fig. 7 Angular Velocity in Steady Hover

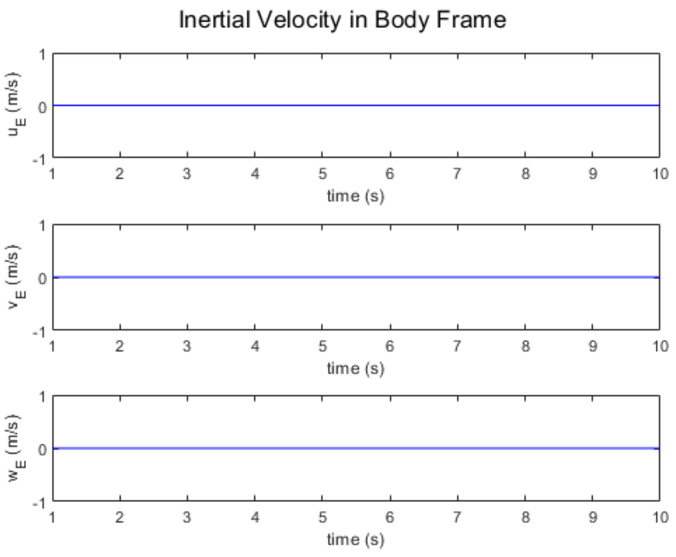


Fig. 8 Inertial Velocity in Steady Hover

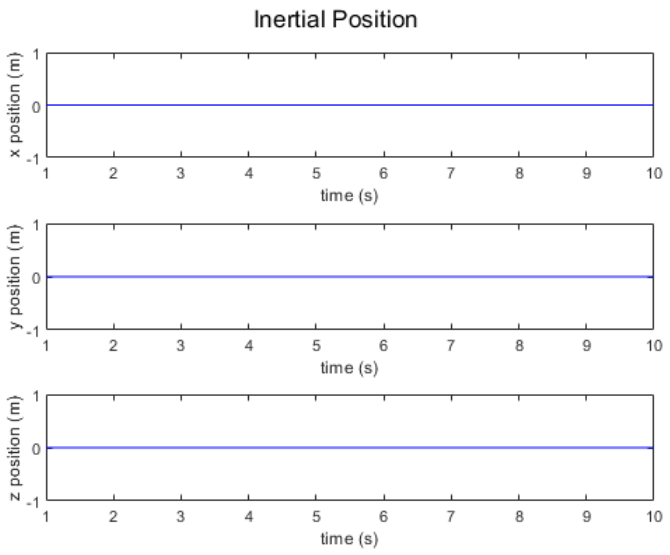


Fig. 9 Inertial Position in Steady Hover

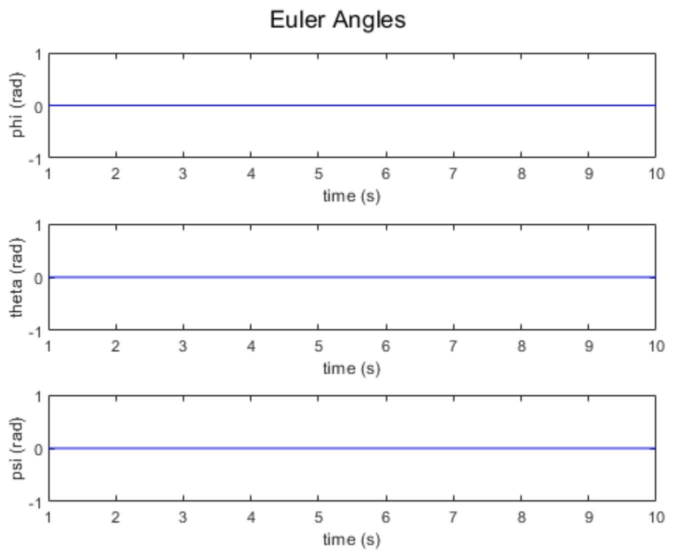


Fig. 10 Euler Angles in Steady Hover

As seen in the plots above, adding the aerodynamic forces and moments due to drag has not changed the trim state for steady hover.

C. 1.4 Plots

1. Simulation 1

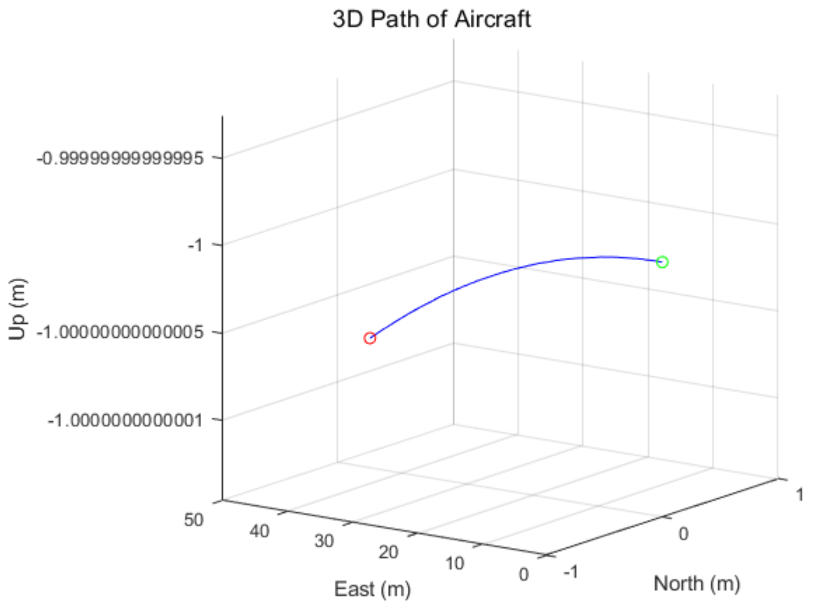


Fig. 11 3D Flight Path in Steady Hover

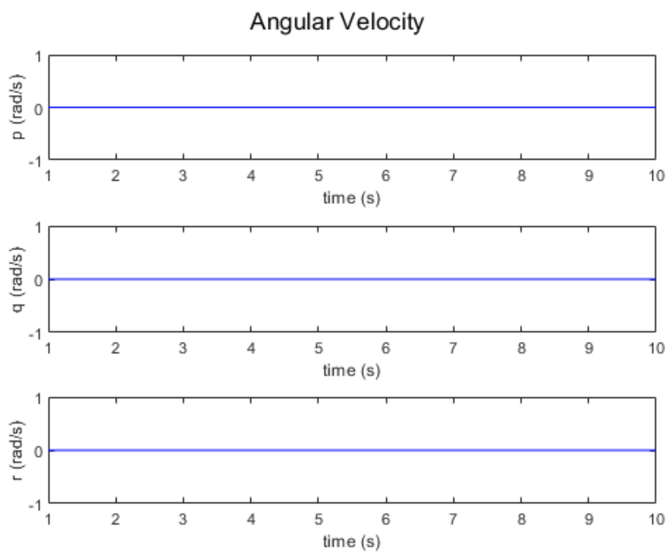


Fig. 12 Angular Velocity in Steady Hover

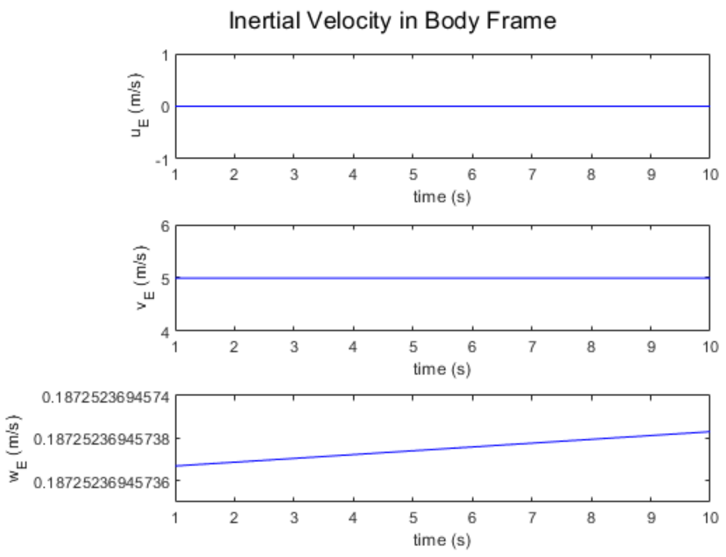


Fig. 13 Inertial Velocity in Steady Hover

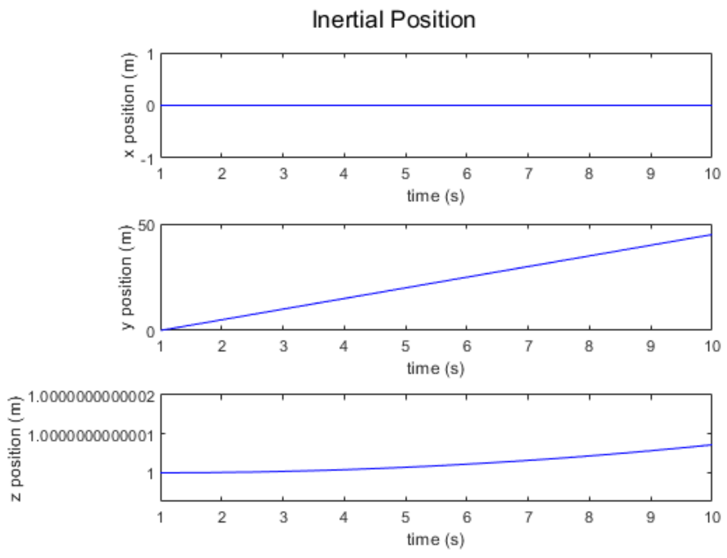


Fig. 14 Inertial Position in Steady Hover

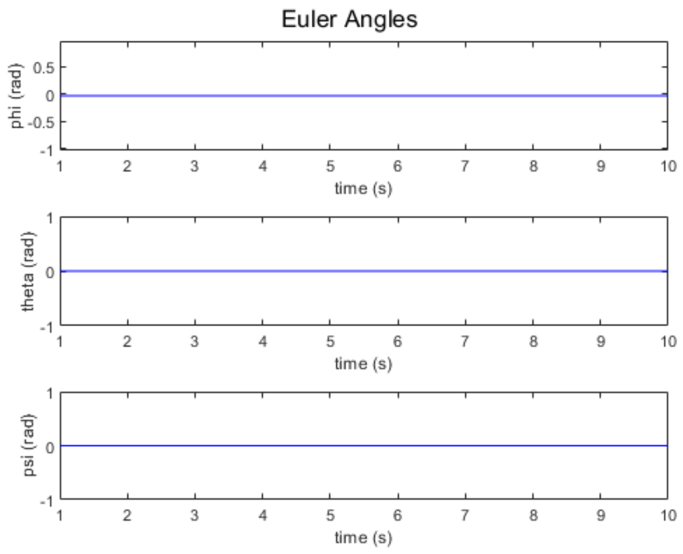


Fig. 15 Euler Angles in Steady Hover

2. *Simulation 2*

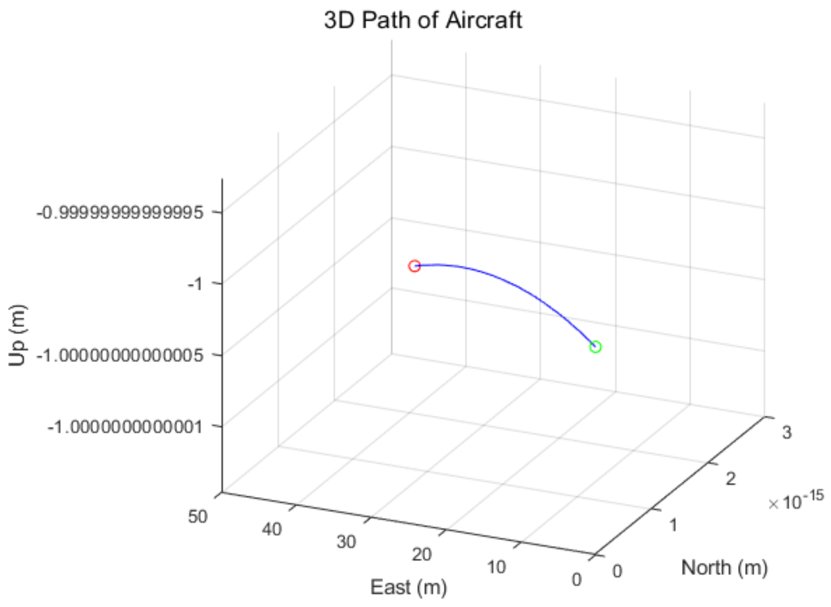


Fig. 16 3D Flight Path in Steady Hover

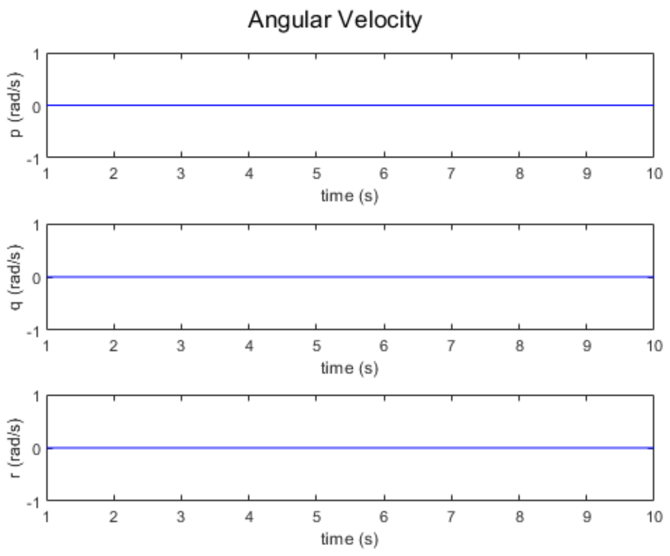


Fig. 17 Angular Velocity in Steady Hover

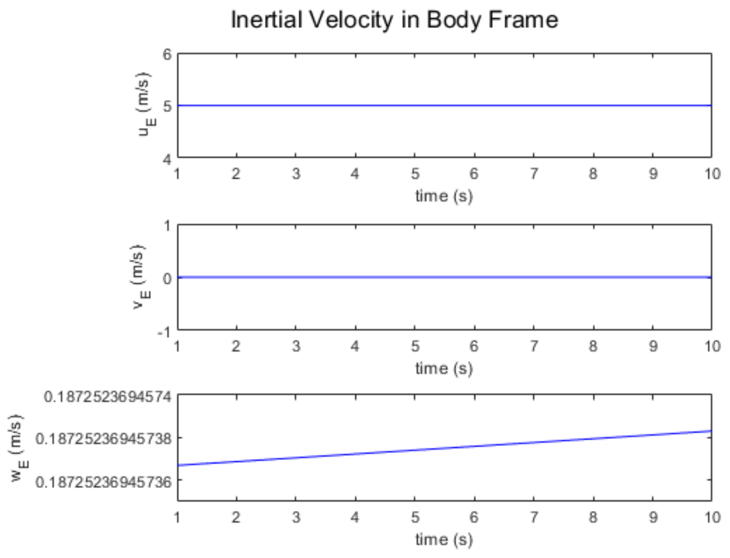


Fig. 18 Inertial Velocity in Steady Hover

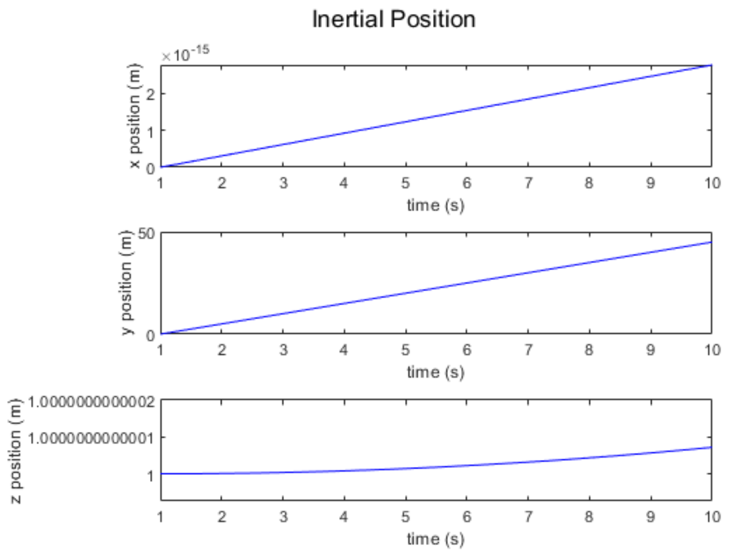


Fig. 19 Inertial Position in Steady Hover

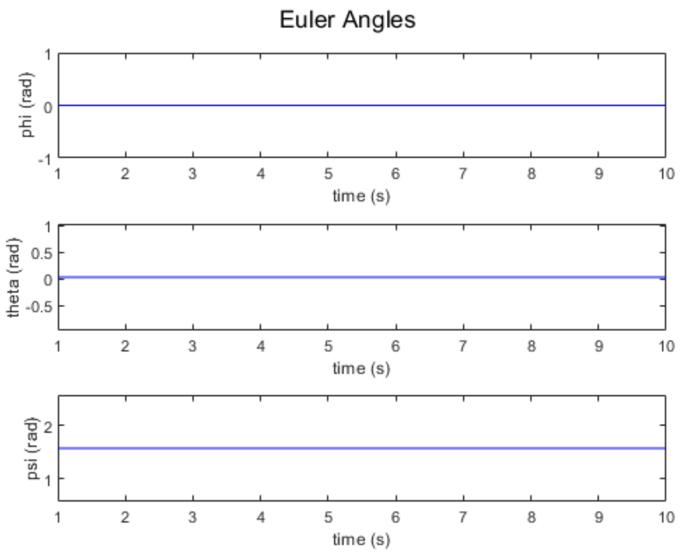


Fig. 20 Euler Angles in Steady Hover

D. 1.5

No, steady hovering flight is not stable for the quadrotor.

II. Task 2

A. 2.1 and 2.2

For each of the following figures, the linearized responses are represented in green and the non-linearized responses are represented in blue.

1. Deviation by +5 deg in roll

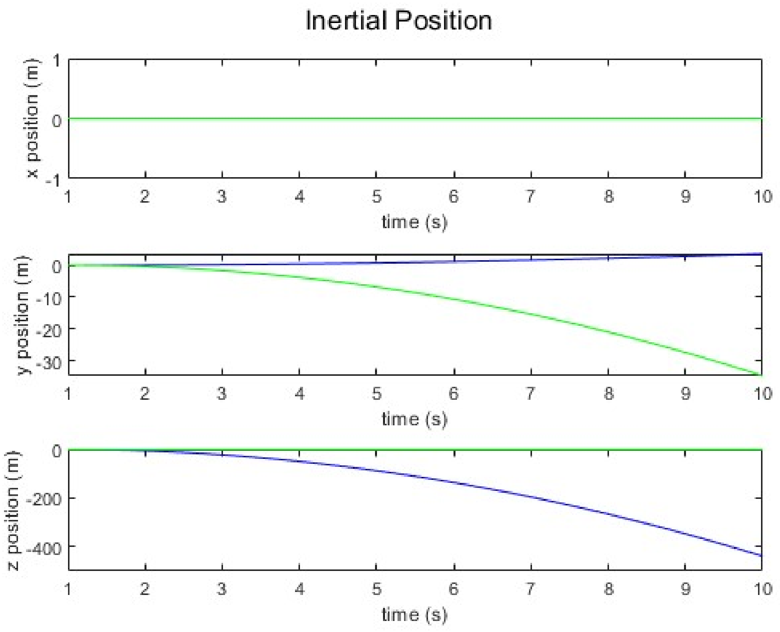


Fig. 21 Inertial Positions of Linearized and Non-linearized A/C

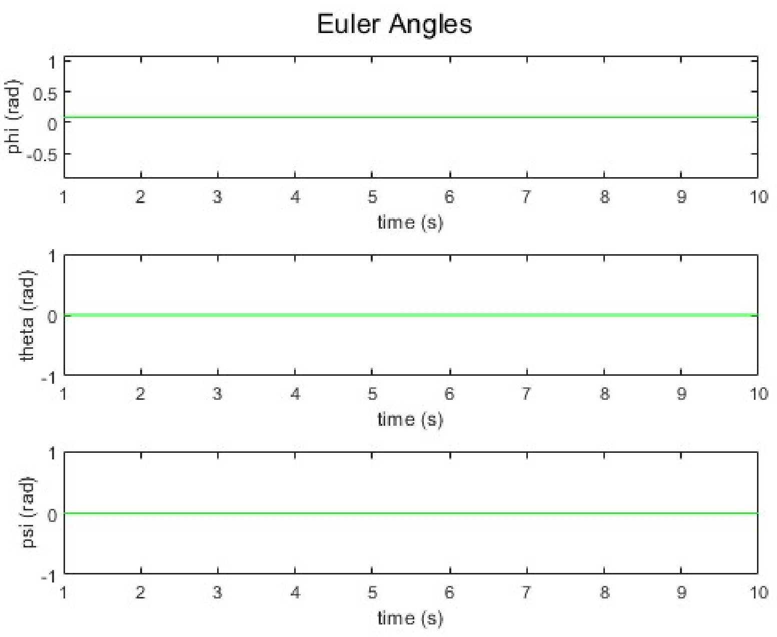


Fig. 22   Euler Angles of Linearized and Non-linearized A/C

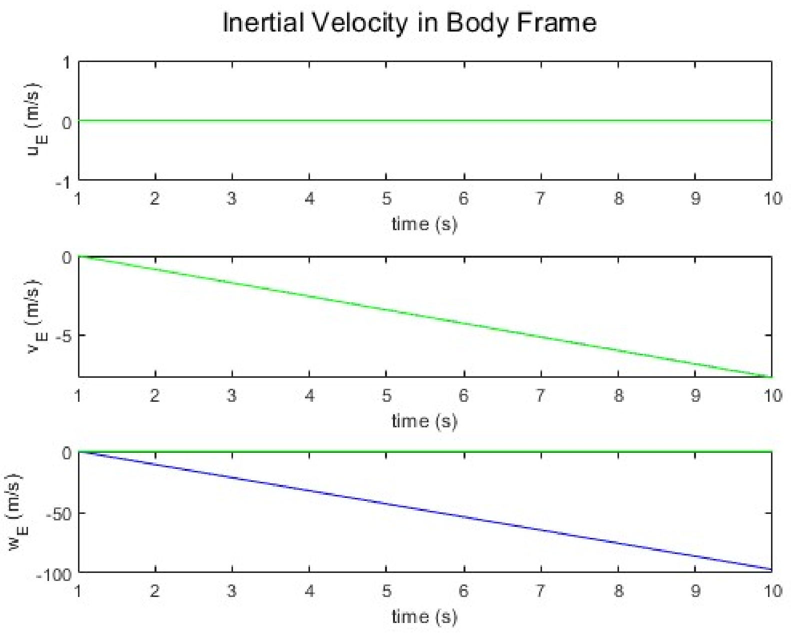


Fig. 23   Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

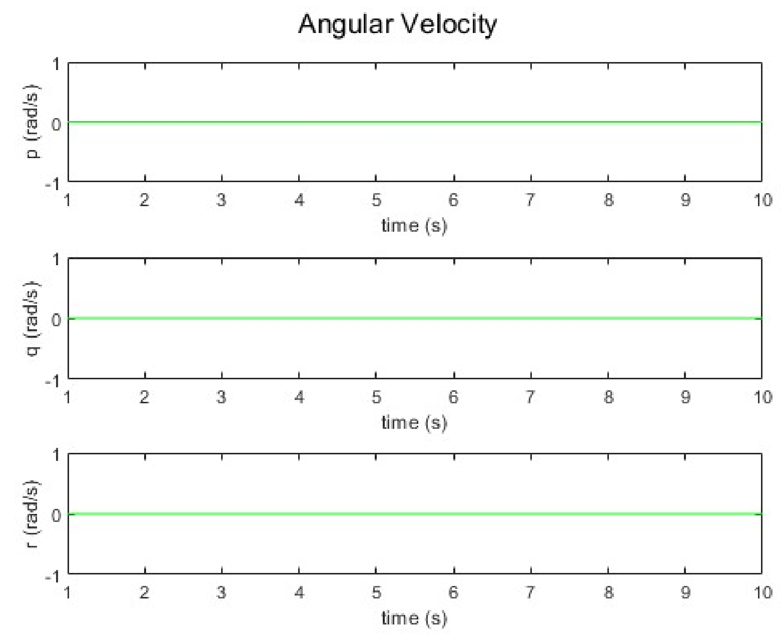


Fig. 24 Angular Velocity of Linearized and Non-linearized A/C

2. Deviation by +5 deg in pitch

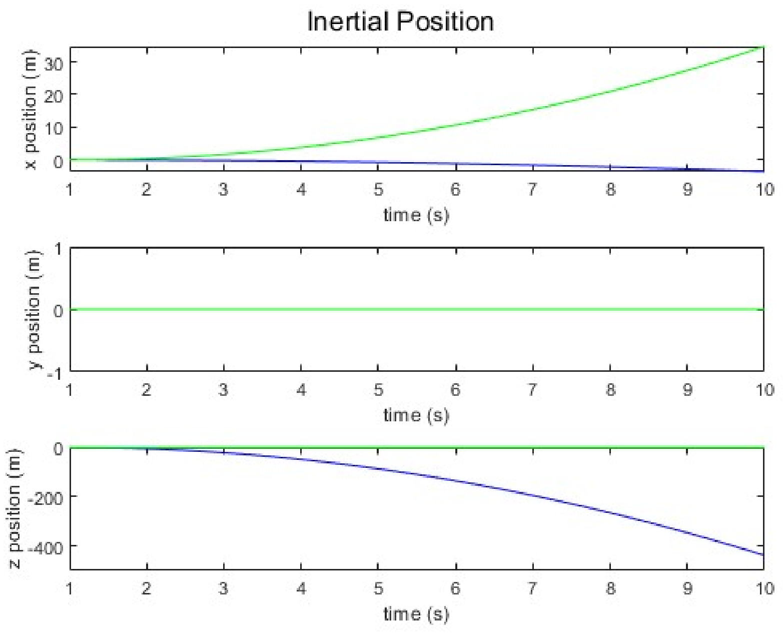


Fig. 25 Inertial Positions of Linearized and Non-linearized A/C

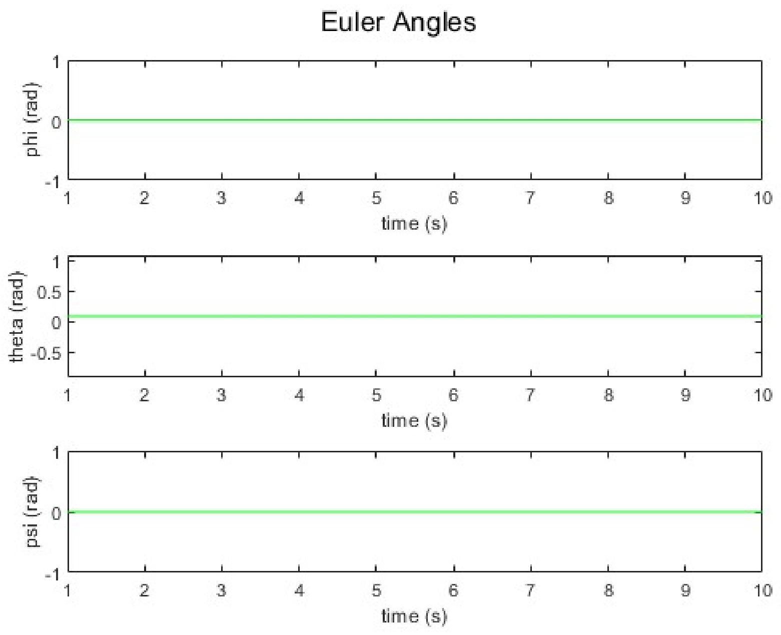


Fig. 26   Euler Angles of Linearized and Non-linearized A/C

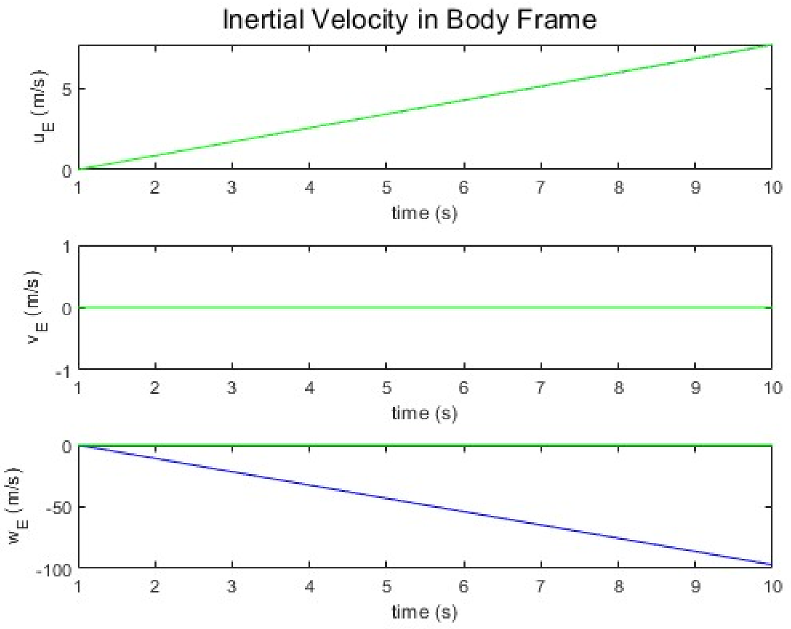


Fig. 27   Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

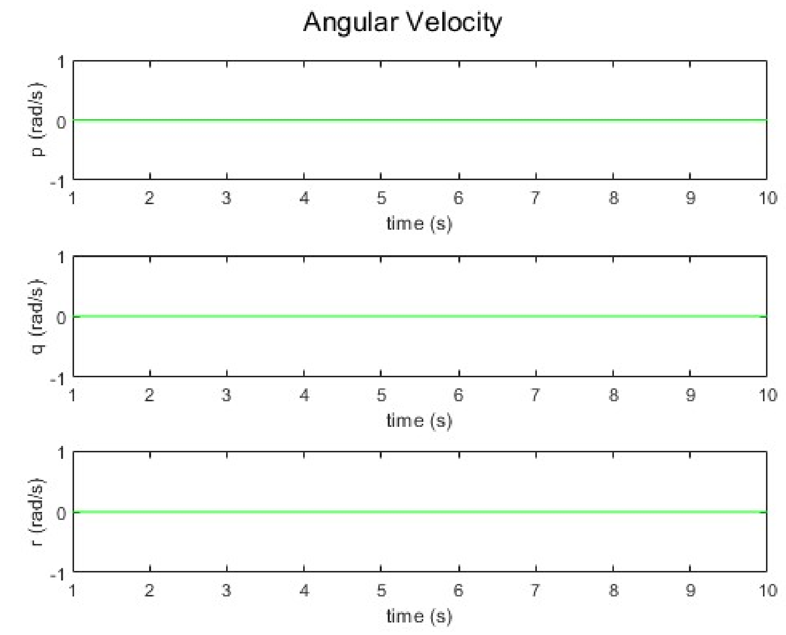


Fig. 28 Angular Velocity of Linearized and Non-linearized A/C

3. Deviation by +5 deg in yaw

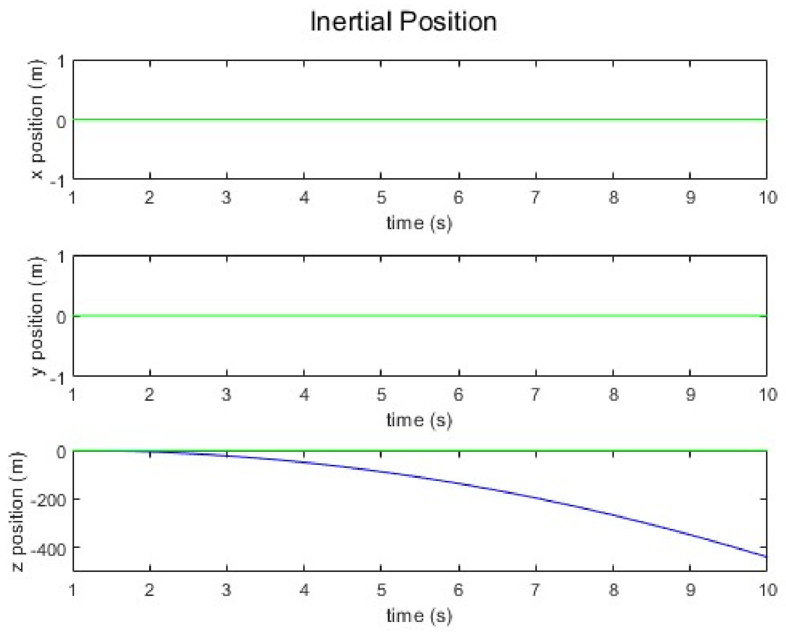


Fig. 29 Inertial Positions of Linearized and Non-linearized A/C

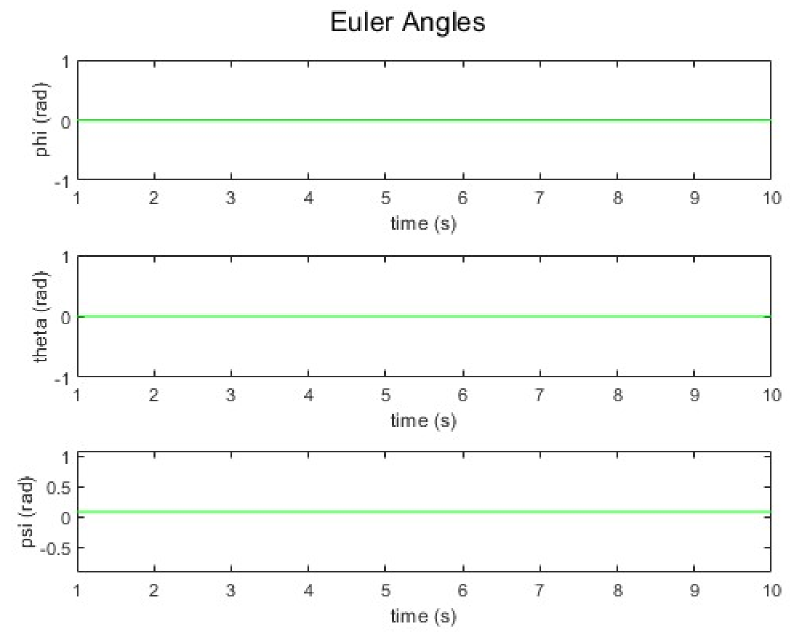


Fig. 30 Euler Angles of Linearized and Non-linearized A/C

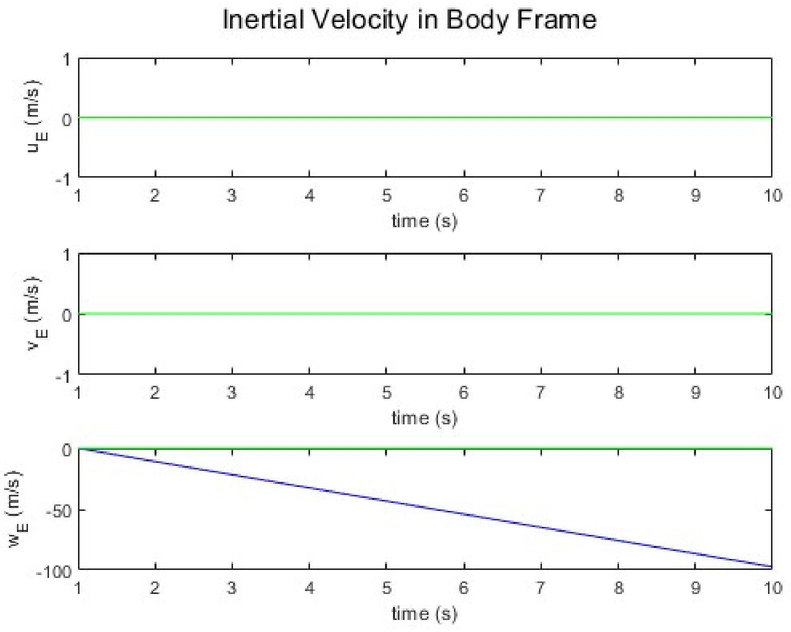


Fig. 31 Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

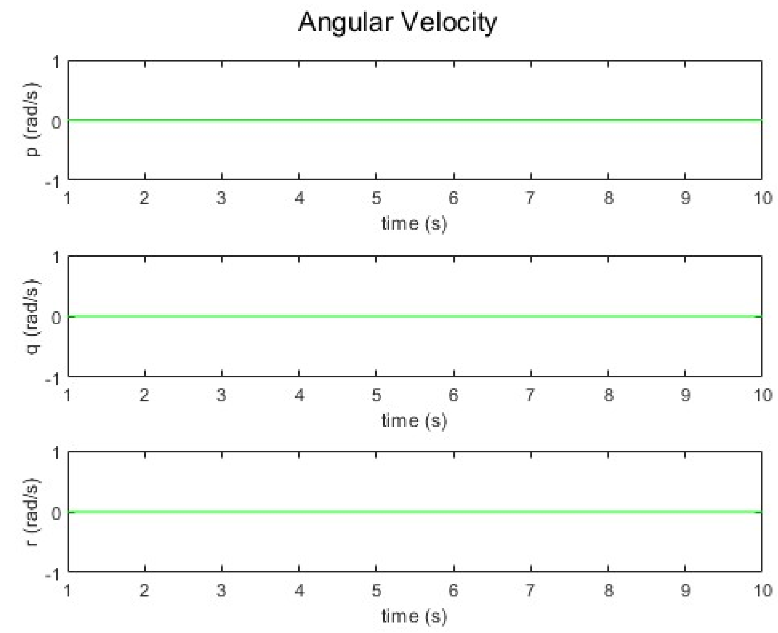


Fig. 32 Angular Velocity of Linearized and Non-linearized A/C

4. Deviation by +0.1 rad/sec in roll rate

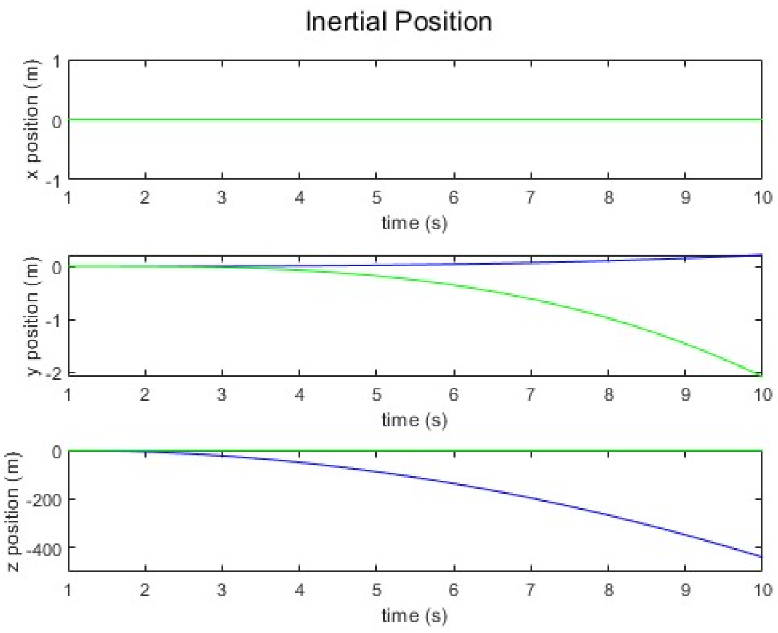


Fig. 33 Inertial Positions of Linearized and Non-linearized A/C

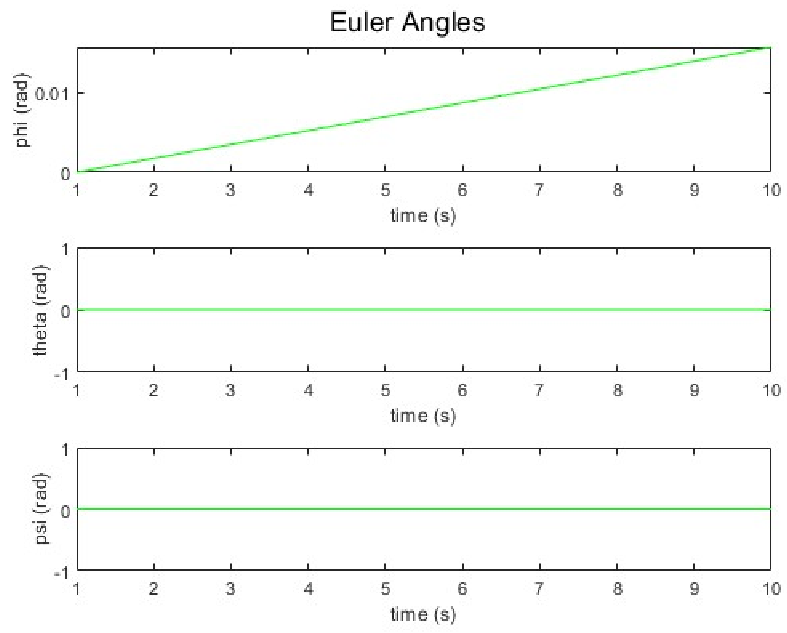


Fig. 34 Euler Angles of Linearized and Non-linearized A/C

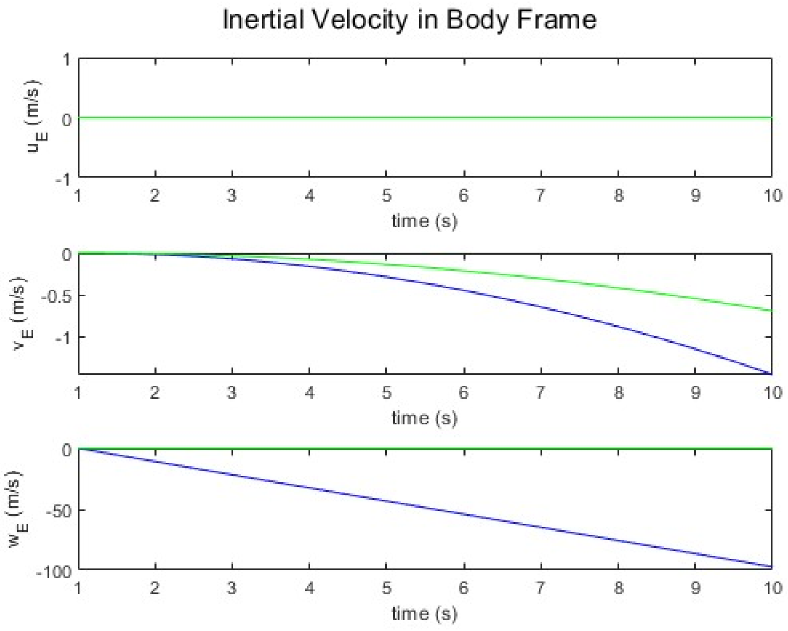


Fig. 35 Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

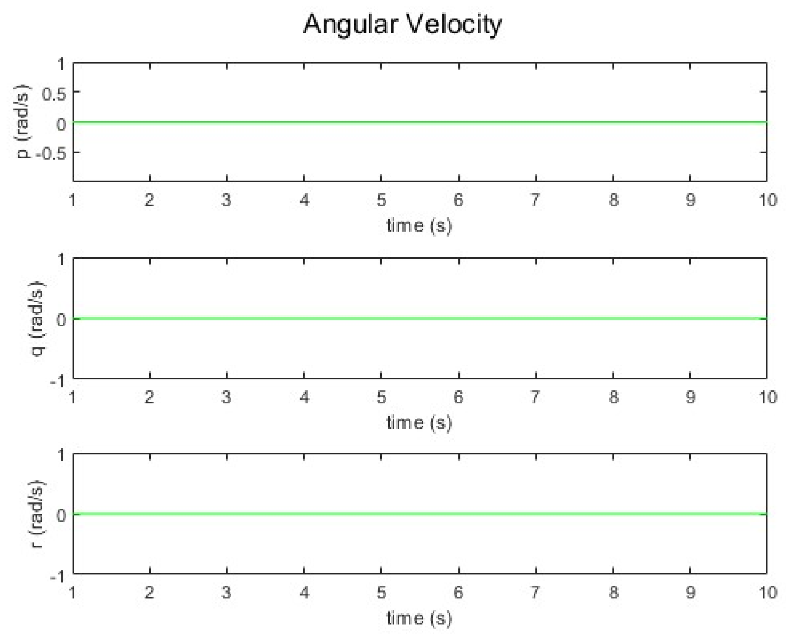


Fig. 36 Angular Velocity of Linearized and Non-linearized A/C

5. Deviation by +0.1 rad/sec in pitch rate

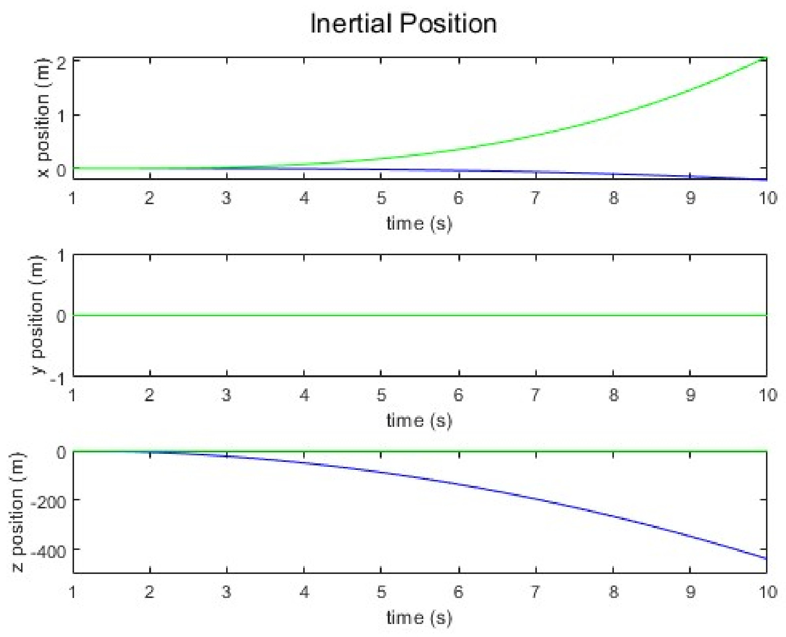


Fig. 37 Inertial Positions of Linearized and Non-linearized A/C

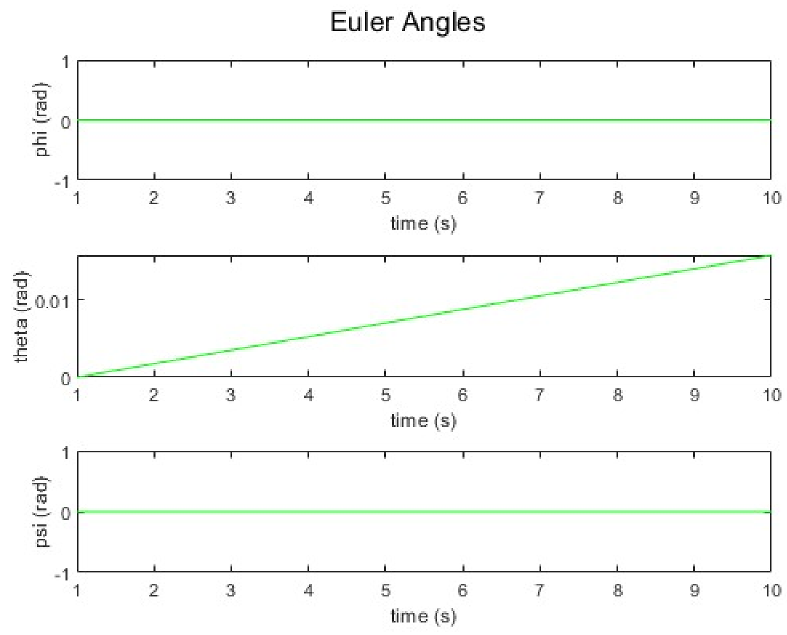


Fig. 38   Euler Angles of Linearized and Non-linearized A/C

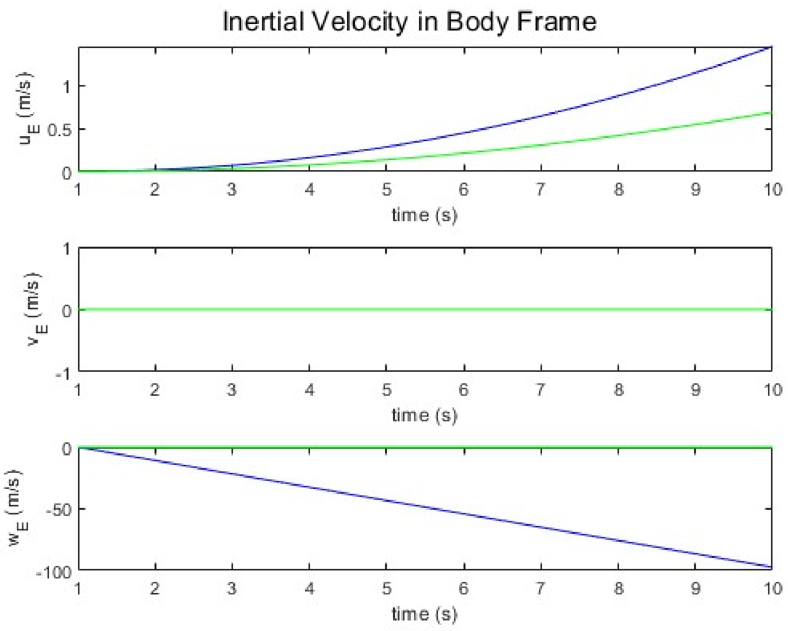


Fig. 39   Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

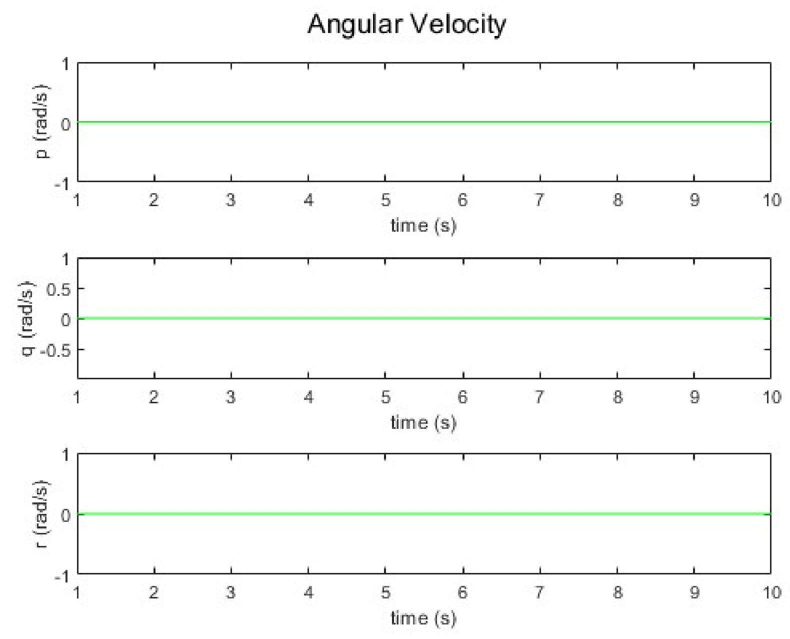


Fig. 40 Angular Velocity of Linearized and Non-linearized A/C

6. Deviation by +0.1 rad/sec in yaw rate

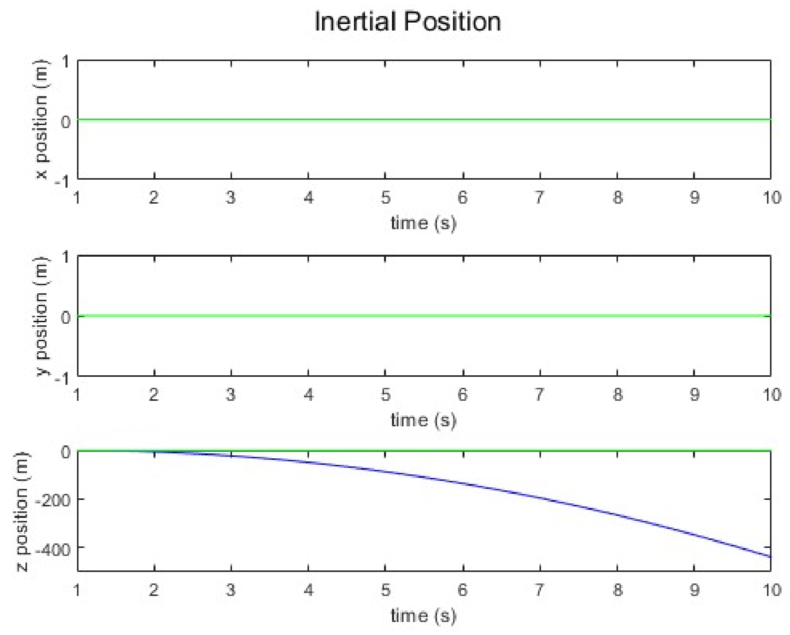


Fig. 41 Inertial Positions of Linearized and Non-linearized A/C

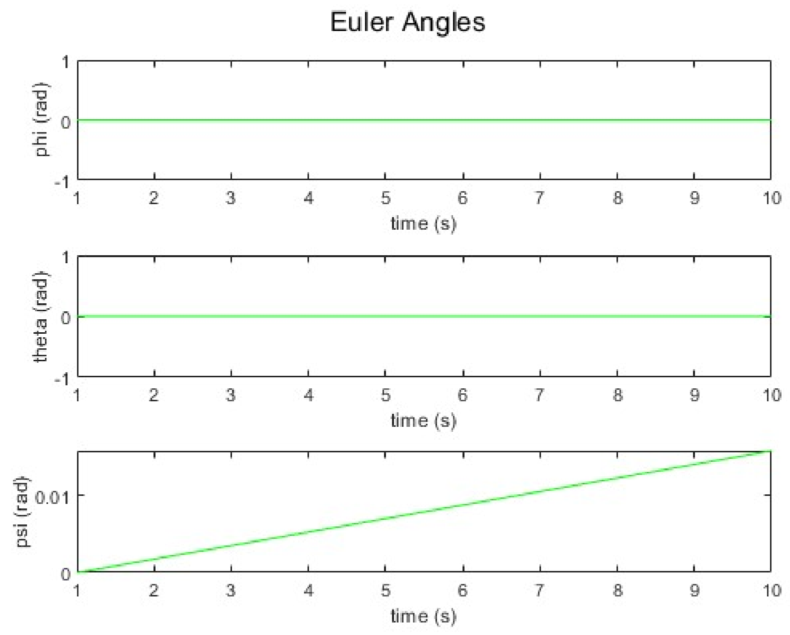


Fig. 42 Euler Angles of Linearized and Non-linearized A/C

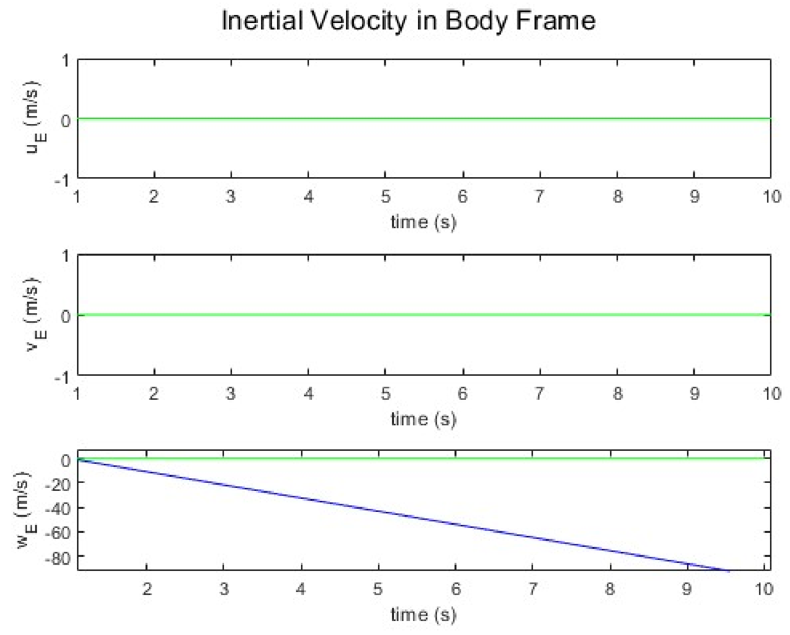


Fig. 43 Inertial Velocity in Body Coordinates of Linearized and Non-linearized A/C

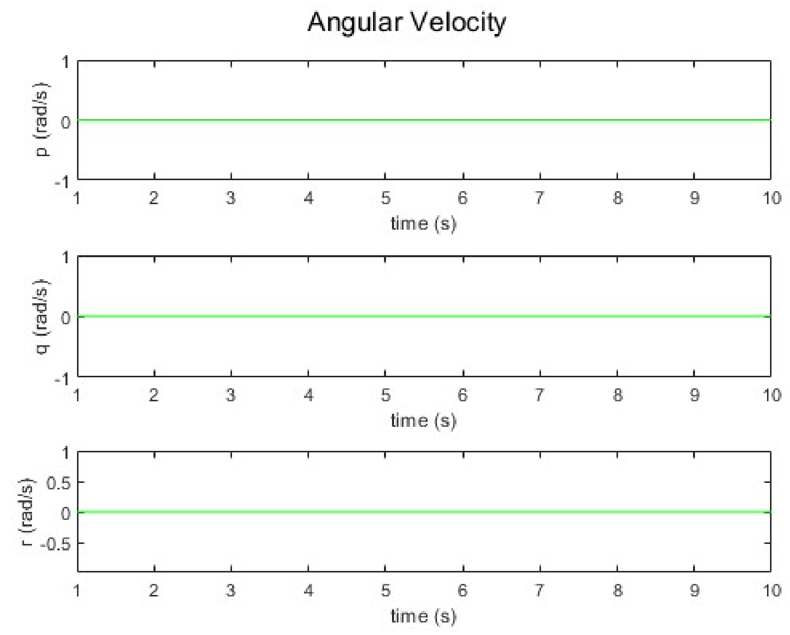


Fig. 44 Angular Velocity of Linearized and Non-linearized A/C

When simulating these different scenarios, we noticed that our results did seem to make physical sense. The nonlinear equations of motion are more sensitive to the perturbations and that was shown within our plots. Our results agreed with our previous conclusion because the flights were not stable once the perturbations were added.

B. 2.3
See MATLAB code in Appendix: **Reference here**

C. 2.4
See MATLAB code in Appendix: **Reference here**

D. 2.5
For each of the following figures the controlled responses are represented in green and the uncontrolled responses are represented in blue.

1. Deviation by +0.1 rad/sec in roll rate

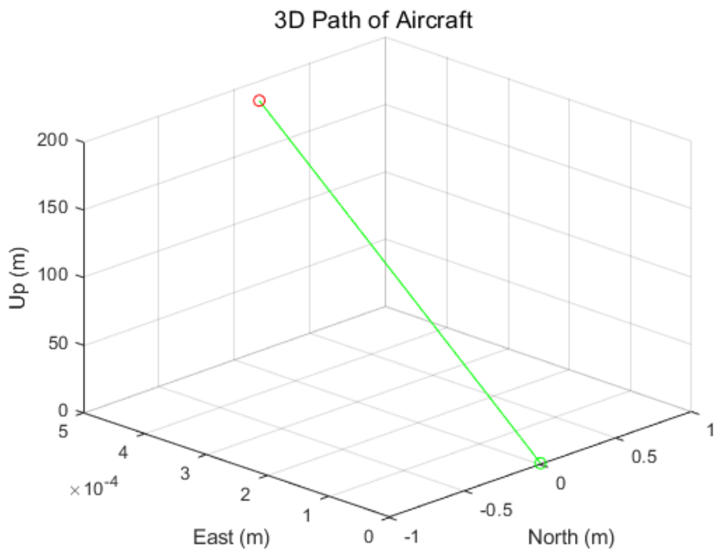


Fig. 45 3D Flight Path

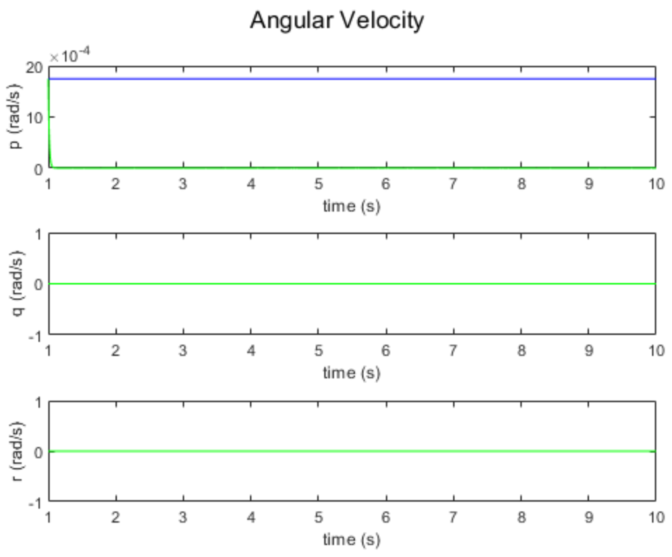


Fig. 46 Angular Velocity

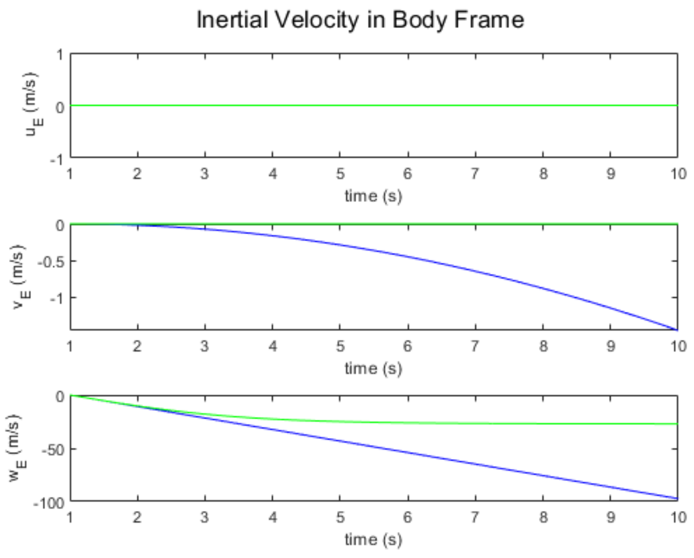


Fig. 47 Inertial Velocity

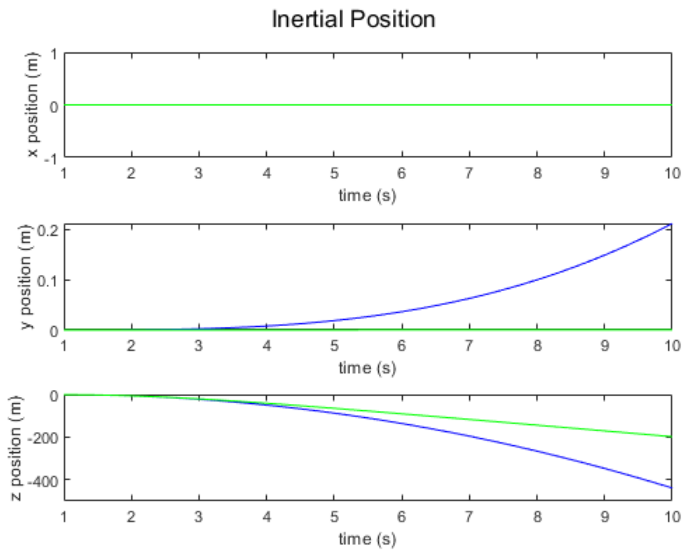


Fig. 48 Inertial Position

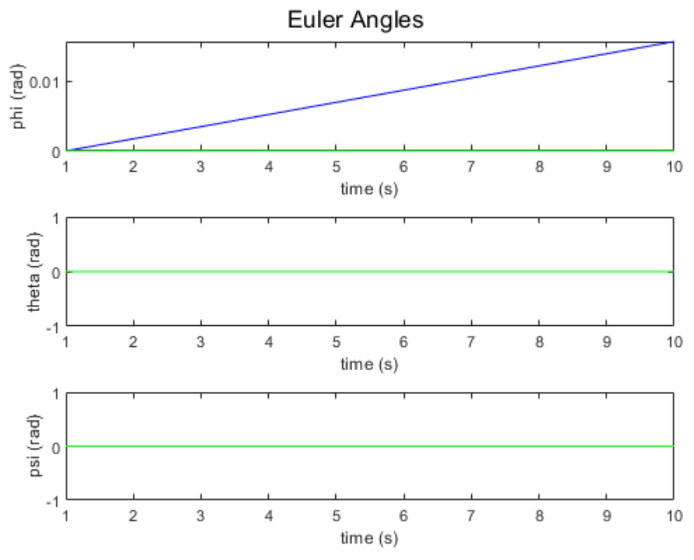


Fig. 49 Euler Angles

2. Deviation by +0.1 rad/sec in pitch rate

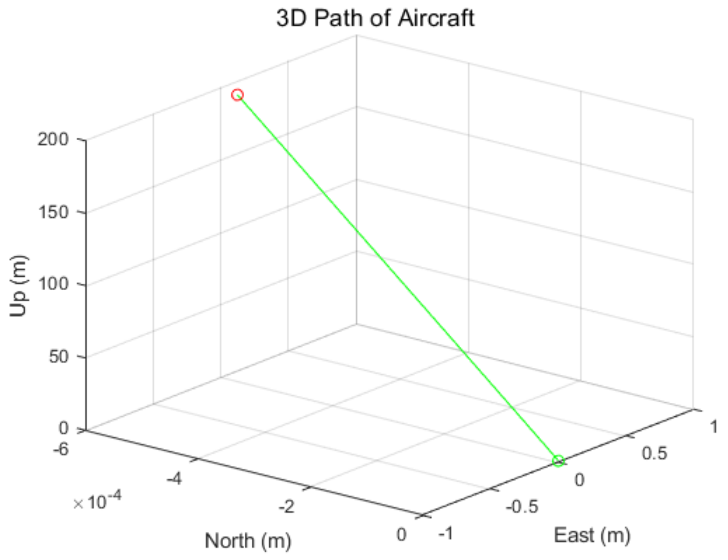


Fig. 50 3D Flight Path

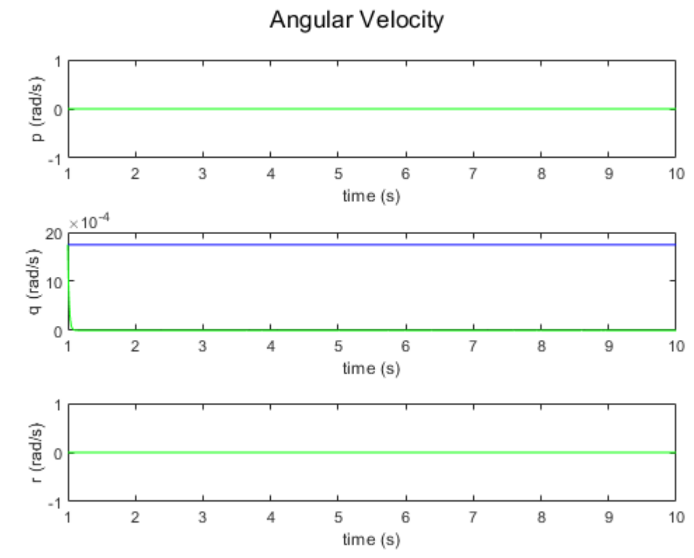


Fig. 51 Angular Velocity

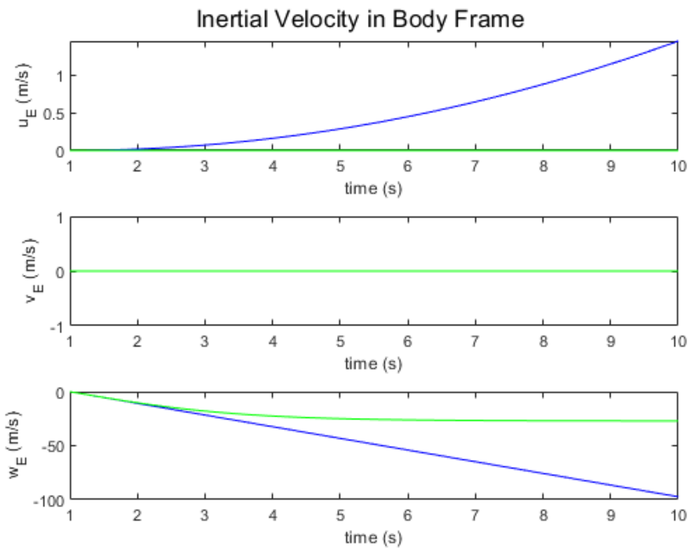


Fig. 52 Inertial Velocity

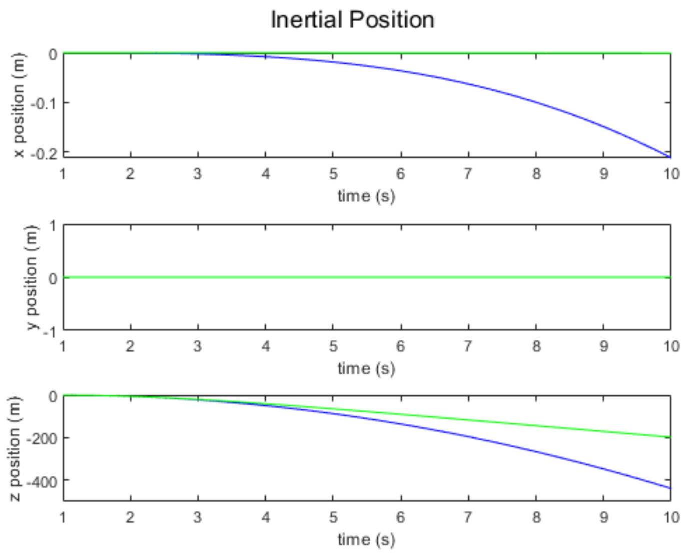


Fig. 53 Inertial Position

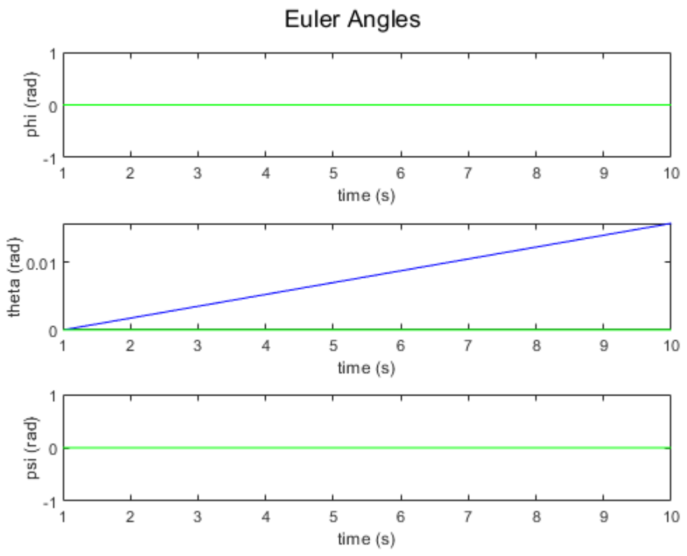


Fig. 54 Euler Angles

3. Deviation by +0.1 rad/sec in yaw rate

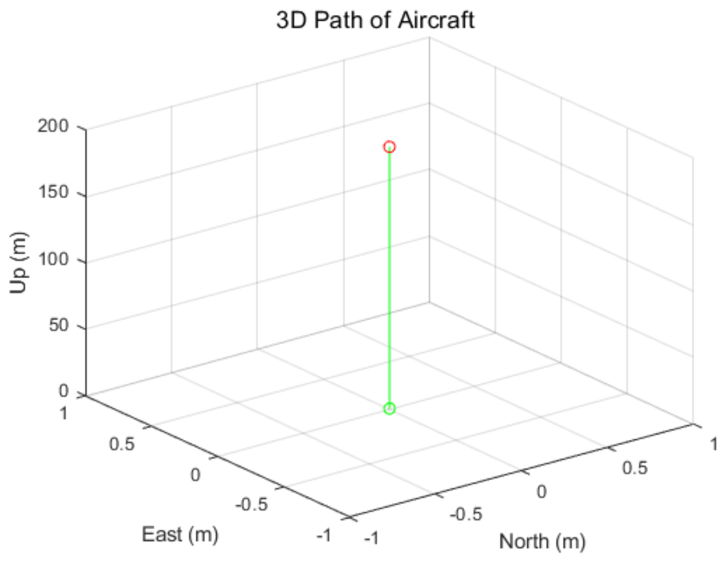


Fig. 55 3D Flight Path

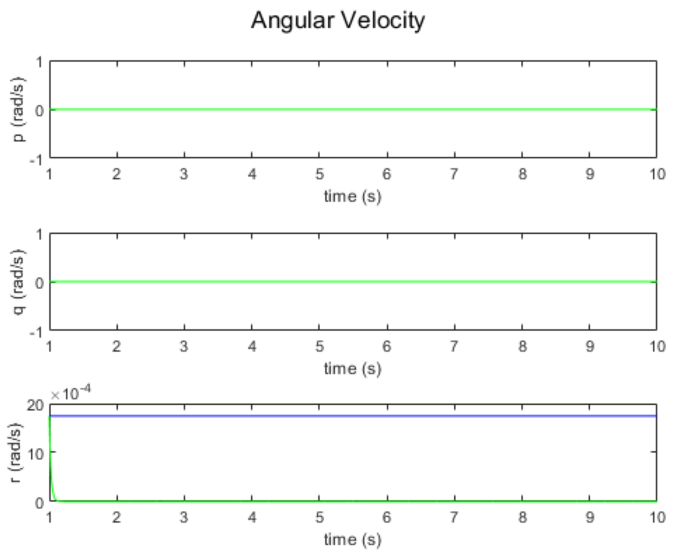


Fig. 56 Angular Velocity

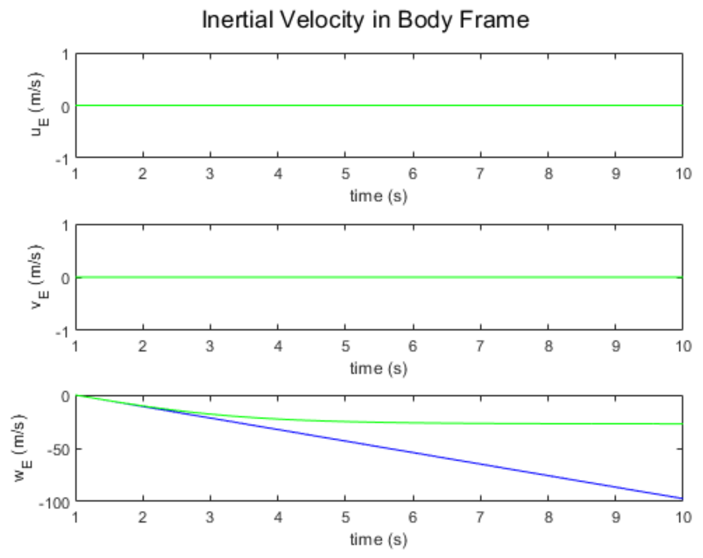


Fig. 57 Inertial Velocity

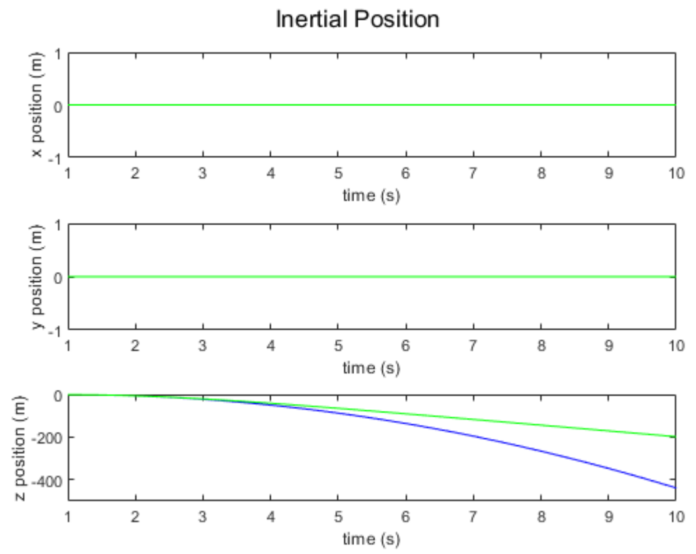


Fig. 58 Inertial Position

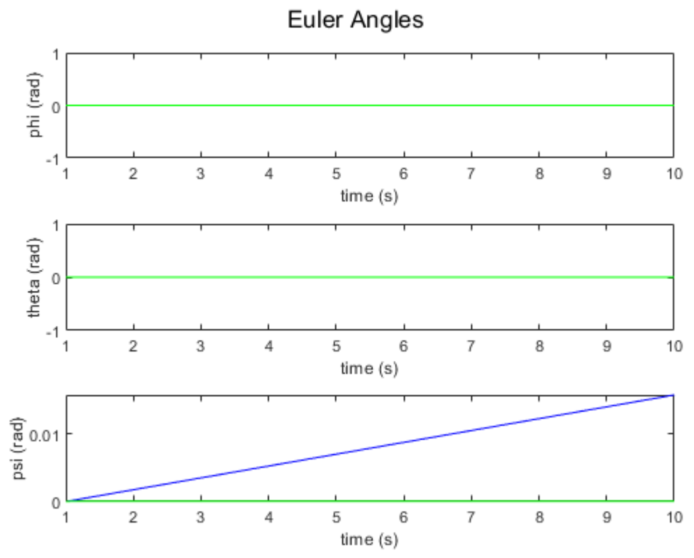


Fig. 59 Euler Angles

The control laws act to stabilize the flight of the quadrotor and it can be seen that it is being attempted, but with our simulations we still see that we are not reaching a truly stable flight despite attempting to do so.

III. Task 3

A. 3.3

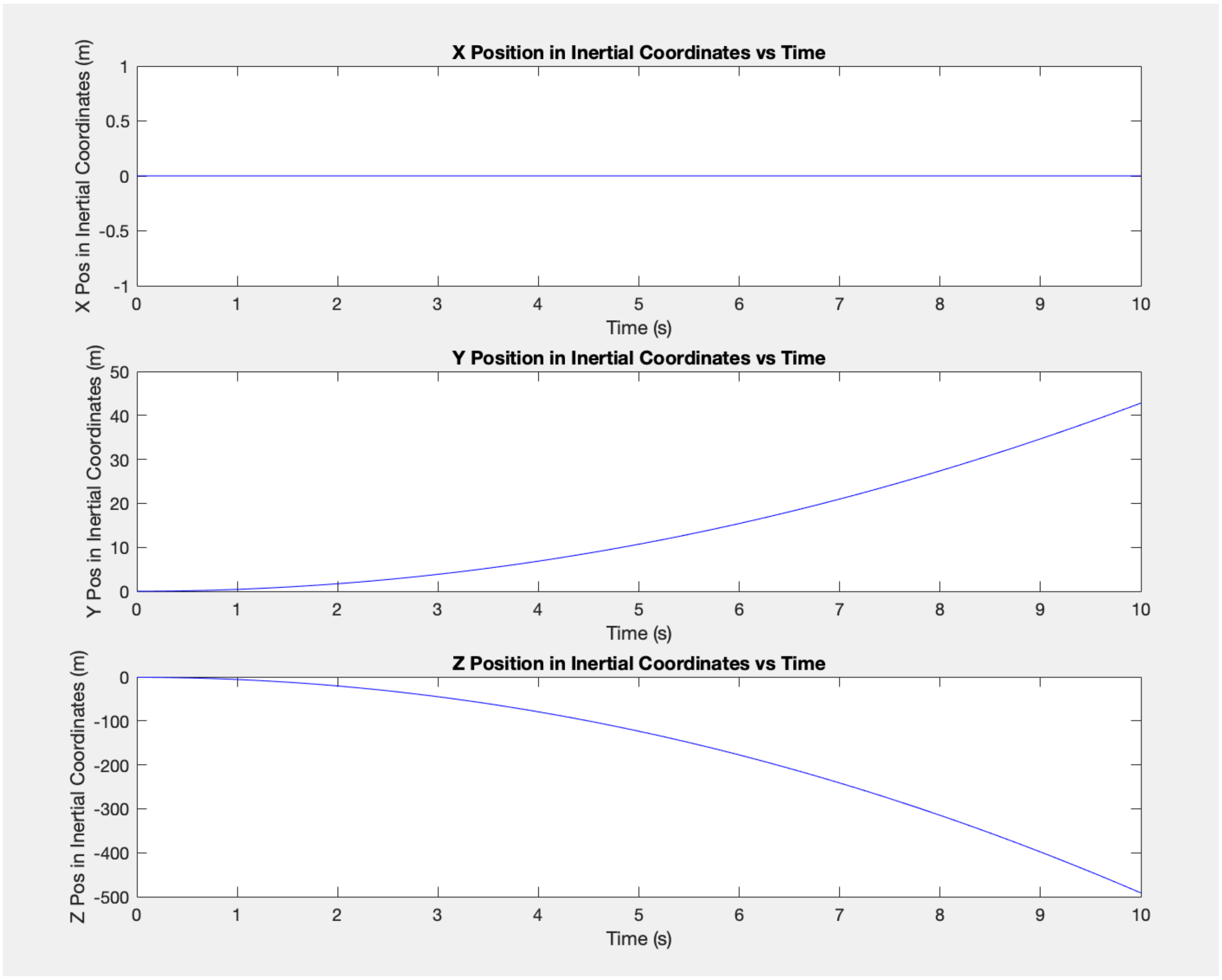


Fig. 60

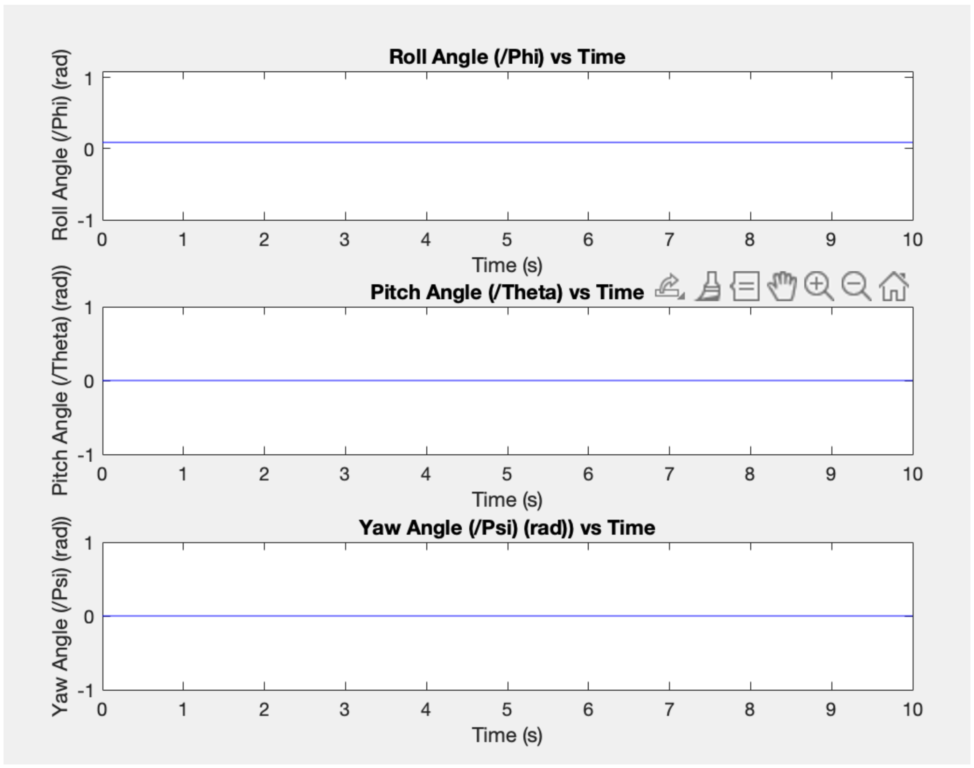

Roll Angle (/Phi) vs Time
Roll Angle (/Phi) (rad)
Time (s)
Pitch Angle (/Theta) vs Time
Pitch Angle (/Theta) (rad))
Time (s)
Yaw Angle (/Psi) (rad)) vs Time
Yaw Angle (/Psi) (rad))
Time (s)

Fig. 61

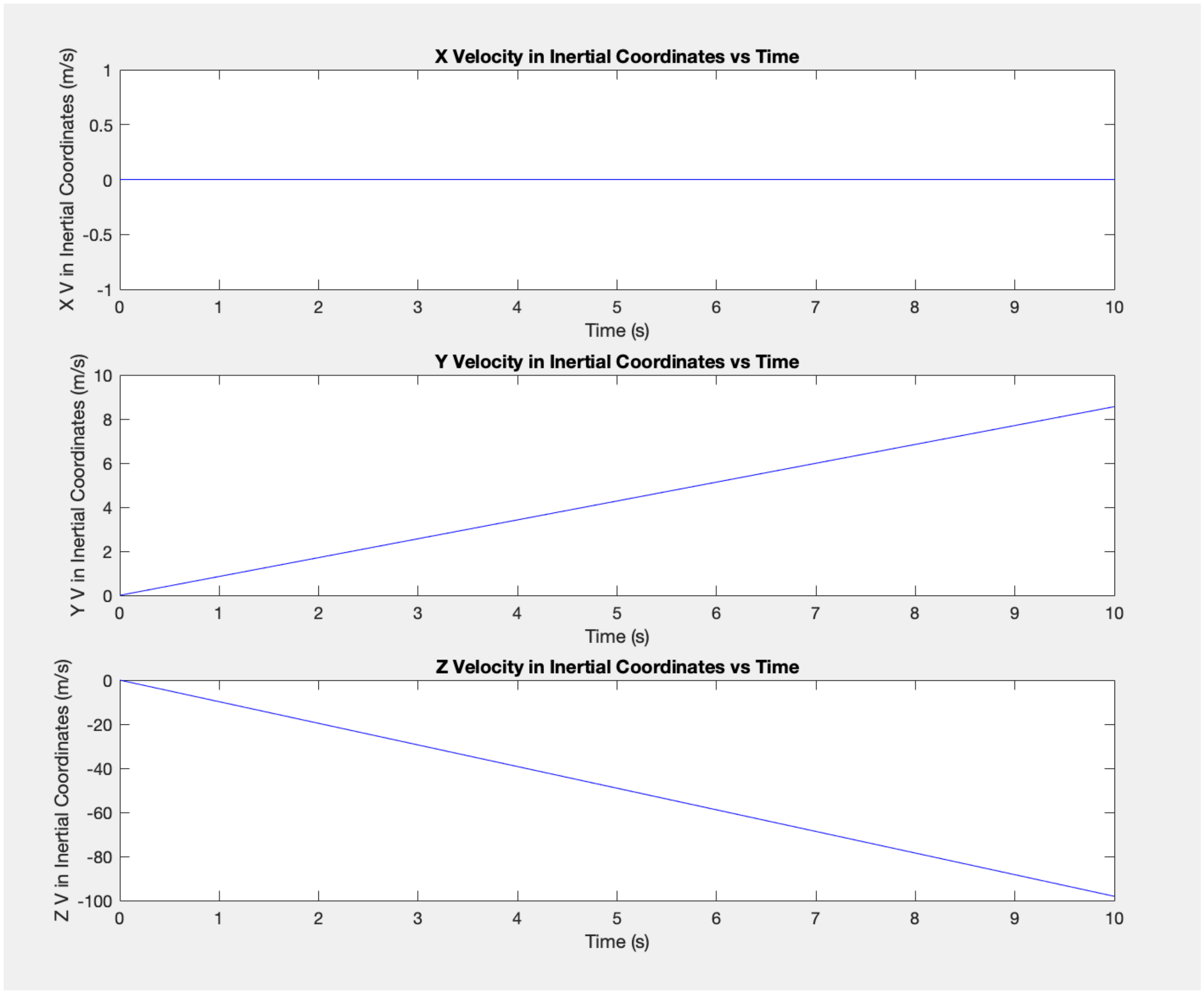

X Velocity in Inertial Coordinates vs Time
X V in Inertial Coordinates (m/s)
Time (s)
Y Velocity in Inertial Coordinates vs Time
Y V in Inertial Coordinates (m/s)
Time (s)
Z Velocity in Inertial Coordinates vs Time
Z V in Inertial Coordinates (m/s)
Time (s)

Fig. 62

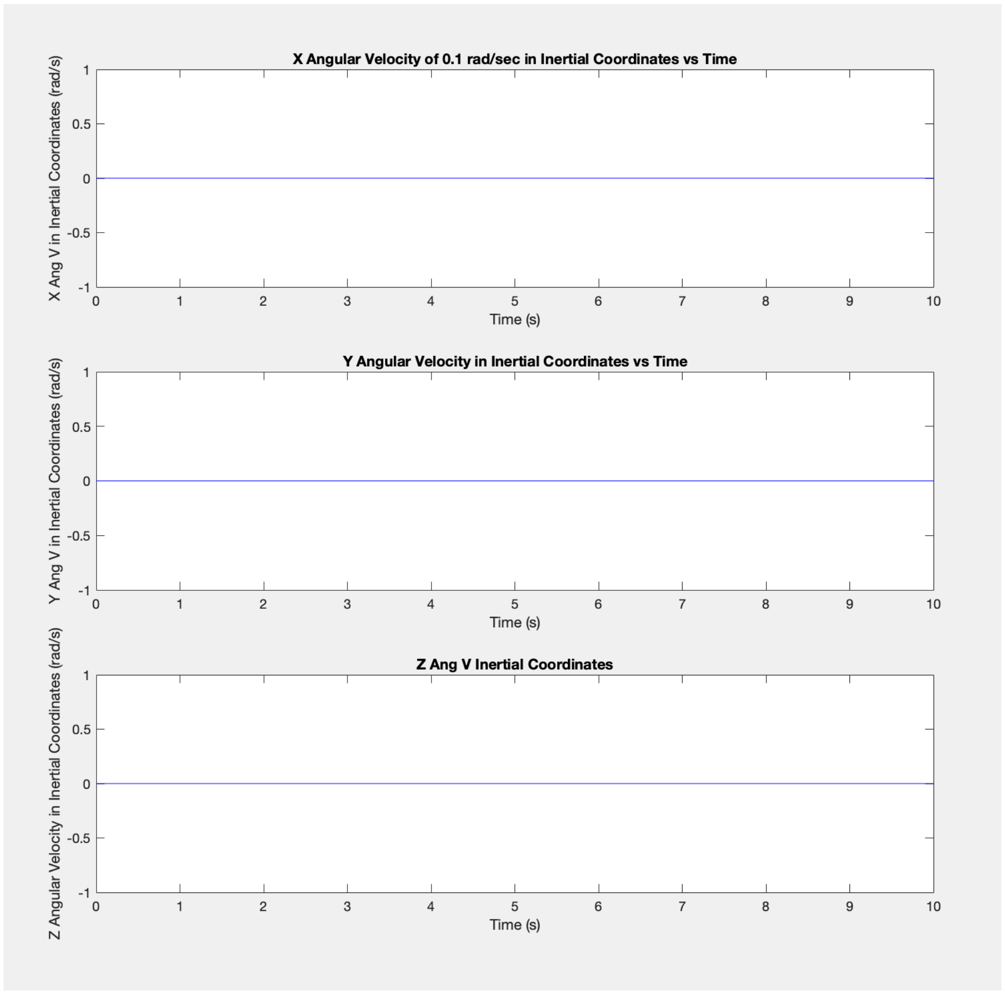
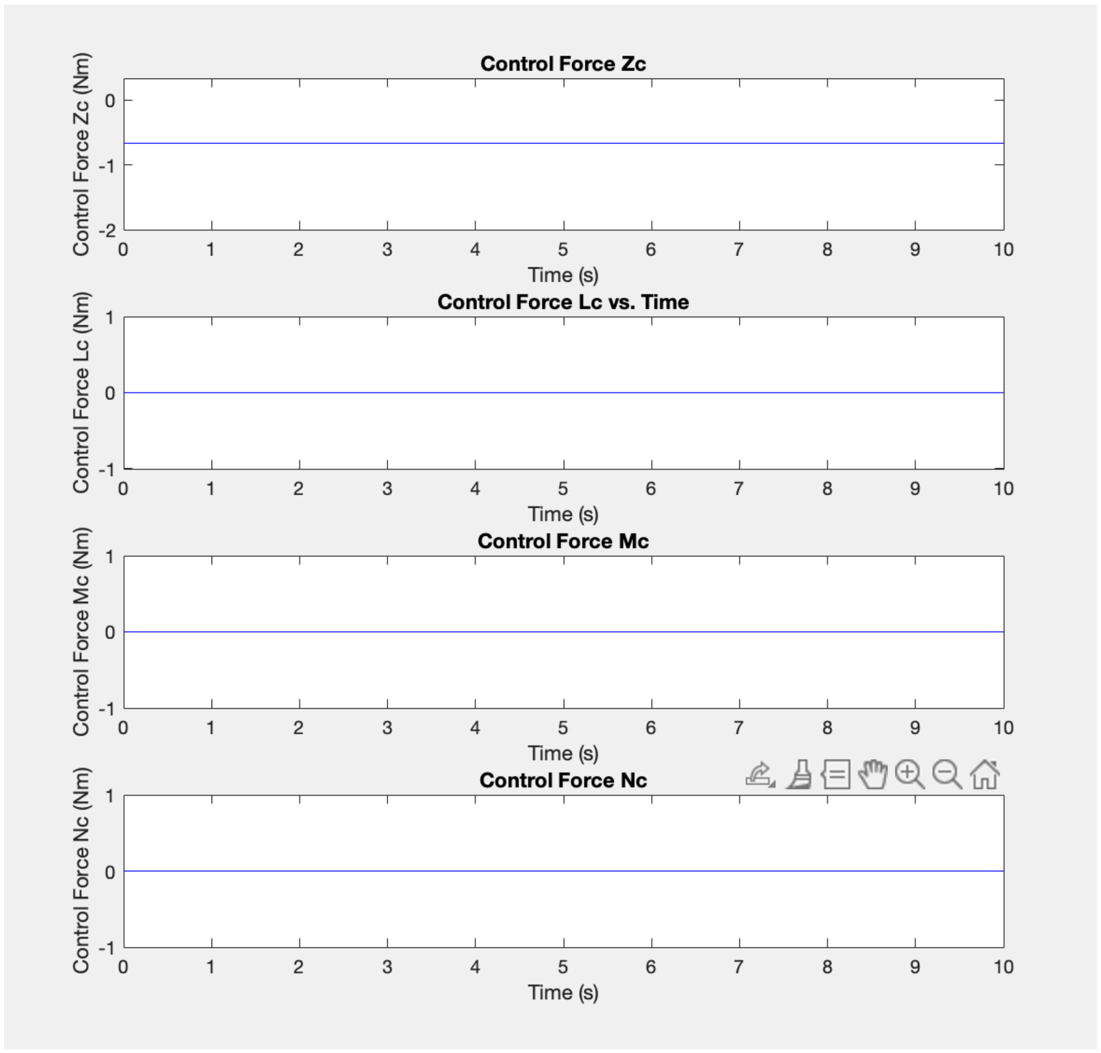


Fig. 63

This corresponds somewhat to what we expected because we expected instability but it acts in a sort of unstable stable fashion, as once its perturbed it will continue into that direction once creating equilibrium with the forces.

B. 3.5



Control Force Zc
Control Force Zc (Nm)
0
-1
-2
0
1
2
3
4
5
6
7
8
9
10
Time (s)
Control Force Lc vs. Time
Control Force Lc (Nm)
1
0
-1
Time (s)
Control Force Mc
Control Force Mc (Nm)
1
0
-1
Time (s)
Control Force Nc
Control Force Nc (Nm)
1
0
-1
Time (s)

Fig. 64

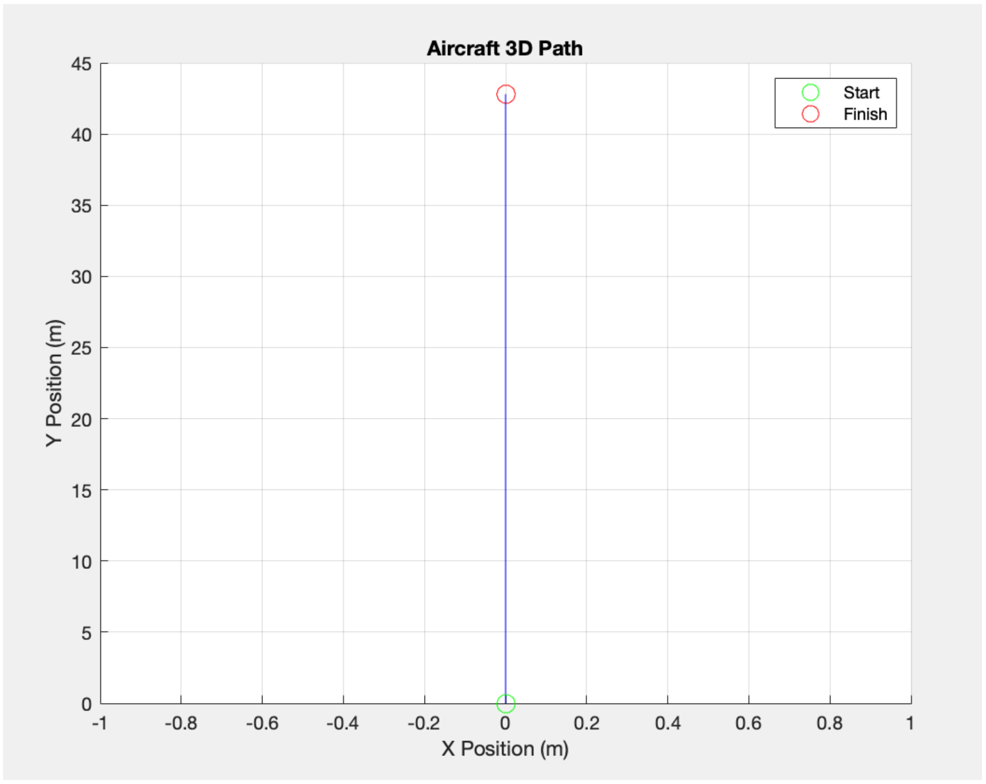


Fig. 65

The difference between our calculated results and the real results are due to the fact that there are many assumptions and simplifications to our model than what happens in real life. There are lots of things that could potentially go wrong or work imperfectly in a real life situation and our model really has no way to account for it.

IV. Acknowledgements

	Plan	Model	Experiment	Results	Report	Code	ACK
Race	1	1	1	1	2	1	RP
Hayden	1	1	1	1	2	2	HG
Hector	1	1	1	1	1	1	Hr
Manas	1	1	2	1	1	1	MK

V. Code

A. PlotAircraftSim

```
function PlotAircraftSim(time, aircraft_state_array, control_input_array, fig,
    col)

    figure(fig(1)); hold on;
    sgtitle('Inertial␣Position')

    subplot(3, 1, 1);
    plot(time, aircraft_state_array(:, 1), col); hold on;
    xlabel('time␣(s)')
    ylabel('x␣position␣(m)')
    subplot(3, 1, 2);
    plot(time, aircraft_state_array(:, 2), col); hold on;
    xlabel('time␣(s)')
    ylabel('y␣position␣(m)')
    subplot(3, 1, 3);
    plot(time, aircraft_state_array(:, 3), col); hold on;
    xlabel('time␣(s)')
    ylabel('z␣position␣(m)')
    hold off;

    figure(fig(2)); hold on;
    sgtitle('Euler␣Angles')

    subplot(3, 1, 1);
    plot(time, aircraft_state_array(:, 4), col); hold on;
    xlabel('time␣(s)')
    ylabel('phi␣(rad)')
    subplot(3, 1, 2);
    plot(time, aircraft_state_array(:, 5), col); hold on;
    xlabel('time␣(s)')
    ylabel('theta␣(rad)')
    subplot(3, 1, 3);
    plot(time, aircraft_state_array(:, 6), col); hold on;
    xlabel('time␣(s)')
    ylabel('psi␣(rad)')
    hold off;

    figure(fig(3)); hold on;
    sgtitle('Inertial␣Velocity␣in␣Body␣Frame')

    subplot(3, 1, 1);
    plot(time, aircraft_state_array(:, 7), col); hold on;
    xlabel('time␣(s)')
    ylabel('u_E␣(m/s)')
    subplot(3, 1, 2);
    plot(time, aircraft_state_array(:, 8), col); hold on;
    xlabel('time␣(s)')
    ylabel('v_E␣(m/s)')
    subplot(3, 1, 3);
    plot(time, aircraft_state_array(:, 9), col); hold on;
```

```
xlabel('time␣(s)')
ylabel('w_E␣(m/s)')
hold off;

figure(fig(4)); hold on;
sgtitle('Angular␣Velocity')

subplot(3, 1, 1);
plot(time, aircraft_state_array(:, 10), col); hold on;
xlabel('time␣(s)')
ylabel('p␣(rad/s)')
subplot(3, 1, 2);
plot(time, aircraft_state_array(:, 11), col); hold on;
xlabel('time␣(s)')
ylabel('q␣(rad/s)')
subplot(3, 1, 3);
plot(time, aircraft_state_array(:, 12), col); hold on;
xlabel('time␣(s)')
ylabel('r␣(rad/s)')
hold off;

% figure(fig(5)); hold on;
% sgtitle('Control Input Variables')
%
% subplot(4, 1, 1);
% plot(time, control_input_array(:, 1), col); hold on;
% xlabel('time (s)')
% ylabel('Z_c (N)')
% subplot(4, 1, 2);
% plot(time, control_input_array(:, 2), col); hold on;
% xlabel('time (s)')
% ylabel('L_c (N)')
% subplot(4, 1, 3);
% plot(time, control_input_array(:, 3), col); hold on;
% xlabel('time (s)')
% ylabel('M_c (N)')
% subplot(4, 1, 4);
% plot(time, control_input_array(:, 4), col); hold on;
% xlabel('time (s)')
% ylabel('N_c (N)')
% hold off;

figure(fig(5));
sgtitle('3D␣Path␣of␣Aircraft')
plot3(aircraft_state_array(:, 1), aircraft_state_array(:, 2), -
    aircraft_state_array(:, 3), col)
xlabel('North␣(m)')
ylabel('East␣(m)')
zlabel('Up␣(m)')
grid ON
hold on;
scatter3(aircraft_state_array(1, 1), aircraft_state_array(1, 2), -
    aircraft_state_array(1, 3), 'green');
    scatter3(aircraft_state_array(end, 1), aircraft_state_array(end, 2), -
        aircraft_state_array(end, 3), 'red');
    hold off;

end
```

B. QuadrotorEOM

```
function var_dot = QuadrotorEOM(t, var, g, m, I, d, km, nu, mu, motor_forces)

    %[F_c, G_c] = feedback_fn(var);
    v = vars;

    x = var(1);
    y = var(2);
    z = var(3);

    phi = var(4);
    theta = var(5);
    psi = var(6);

    u_E = var(7);
    v_E = var(8);
    w_E = var(9);

    p = var(10);
    q = var(11);
    r = var(12);

    V_a = sqrt(u_E^2 + v_E^2 + w_E^2);

    X = -v.nu * V_a *u_E;
    Y = -v.nu * V_a * v_E;
    Z = -v.nu * V_a * w_E;

    F_mag = sqrt(p^2 + q^2 + r^2);

    % L = -v.mu * F_mag * p; problem 2
    % M = -v.mu * F_mag * q;
    % N = -v.mu * F_mag * r;

    L = 0;
    M = 0;
    N = 0;

    % L_c = G_c(1);
    % M_c = G_c(2);
    % N_c = G_c(3);

    L_c = 0;
    M_c = 0;
    N_c = 0;

    Z_c = -sum(motor_forces);

    % EOM

    x_dot = [cos(theta) * cos(psi), sin(phi) * sin(theta) * cos(psi) - cos(phi
        ) * sin(psi), cos(phi) * sin(theta) * cos(psi) + sin(phi) * sin(psi)];
    y_dot = [cos(theta) * sin(psi), sin(phi) * sin(theta) * sin(psi) + cos(phi
        ) * cos(psi), cos(phi) * sin(theta) * sin(psi) - sin(phi) * cos(psi)];
    z_dot = [-sin(theta), sin(phi) * cos(theta), cos(phi) * cos(theta)];

    v_dot = [x_dot ; y_dot; z_dot] * [u_E; v_E; w_E];

    phi_dot =   [1, sin(phi) * tan(theta), cos(phi) * tan(theta)];
    theta_dot = [0, cos(phi), -sin(phi)];
    psi_dot =   [0, sin(phi) * sec(theta), cos(phi) * sec(theta)];

    euler_rates = [phi_dot; theta_dot; psi_dot] * [p; q; r];

    accel = [(r * v_E) - (q * w_E); (p * w_E) - (r * u_E); (q * u_E) - (p *
       v_E)];
    grav = g * [-sin(theta); cos(theta) * sin(phi); cos(theta) * cos(phi)];
    %aero = (1/m) * [X; Y; Z];
    control = (1/m) * [0; 0; Z_c];

    %vel_rates = accel + grav + aero + control; problem 2
    vel_rates = accel + grav + control;

    omega_1 = [((I(2) - I(3)) / I(1)) * q * r; ((I(3) - I(1)) / I(2)) * p * r;
        ((I(1) - I(2)) / I(3)) * p * q];
    omega_2 = [L/I(1); M/I(2); N/I(3)];
    omega_3 = [L_c/I(1); M_c/I(2); N_c/I(3)];

    angular_rates = omega_1 + omega_2 + omega_3;

    var_dot = [v_dot(1); v_dot(2); v_dot(3); euler_rates(1); euler_rates(2);
       euler_rates(3); vel_rates(1); vel_rates(2); vel_rates(3);
       angular_rates(1); angular_rates(2); angular_rates(3)];

end
```

C. Specific Code for 2.1/2.2

```
% ASEN 3801
% Lab 4
% names

%% Housekeeping

clc; clear; close all;

%% Variable declarations

v = vars;

f1 = v.m / 4;
f2 = f1;
f3 = f1;
f4 = f1;

motor_forces = [f1; f2; f3; f4];

Z_c = -sum(motor_forces);

tspan = [1 10];
time = tspan;

F_c = [0, 0, Z_c];
G_c = [0, 0, 0];

Z_c = Z_c * ones(length(time), 1);
L_c = G_c(1) * ones(length(time), 1);
M_c = G_c(2) * ones(length(time), 1);
N_c = G_c(3) * ones(length(time), 1);

deltaFc = [0, 0, 0]';
deltaGc = [0, 0, 0]';

fig = 1:1:30;

col = 'b-';
col2 = 'g-';

%% Lab Task 1

initial_state = zeros(12, 1);
initial_state_1 = [0, 0, -1, deg2rad(5), 0, 0, 0, 0, 0, 0, 0, 0];
initial_state_2 = [0, 0, -1, 0, deg2rad(5), 0, 0, 0, 0, 0, 0, 0];
initial_state_3 = [0, 0, -1, 0, 0, deg2rad(5), 0, 0, 0, 0, 0, 0];
initial_state_4 = [0, 0, -1, 0, 0, 0, 0, 0, 0, deg2rad(0.1), 0, 0];
initial_state_5 = [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, deg2rad(0.1), 0];
initial_state_6 = [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, deg2rad(0.1)];

[t1, y1] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_1);
[t2, y2] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_2);
[t3, y3] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_3);
[t4, y4] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_4);
[t5, y5] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_5);
[t6, y6] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_6);

[t7, y7] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy;
   v.Iz], deltaFc, deltaGc), tspan, initial_state_1);
[t8, y8] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy;
   v.Iz], deltaFc, deltaGc), tspan, initial_state_2);
[t9, y9] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy;
   v.Iz], deltaFc, deltaGc), tspan, initial_state_3);
[t10, y10] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy
   ; v.Iz], deltaFc, deltaGc), tspan, initial_state_4);
[t11, y11] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy
   ; v.Iz], deltaFc, deltaGc), tspan, initial_state_5);
[t12, y12] = ode45(@(t, y) QuadrotorEOM_Linearized(t, y, v.g, v.m, [v.Ix; v.Iy
   ; v.Iz], deltaFc, deltaGc), tspan, initial_state_6);

% a
hold on;
PlotAircraftSim(t1, y1, [Z_c; 0; 0; 0], fig(1:5), col)
PlotAircraftSim(t7, y7, [Z_c; L_c; M_c; N_c], fig(1:5), col2)
hold off;

% b
hold on;
PlotAircraftSim(t2, y2, [Z_c; 0; 0; 0], fig(6:10), col)
PlotAircraftSim(t8, y8, [Z_c; L_c; M_c; N_c], fig(6:10), col2)
hold off;

% c
hold on;
PlotAircraftSim(t3, y3, [Z_c; 0; 0; 0], fig(11:15), col)
PlotAircraftSim(t9, y9, [Z_c; L_c; M_c; N_c], fig(11:15), col2)
hold off;

% d
hold on;
PlotAircraftSim(t4, y4, [Z_c; 0; 0; 0], fig(16:20), col)
PlotAircraftSim(t10, y10, [Z_c; L_c; M_c; N_c], fig(16:20), col2)
hold off;

% e
hold on;
PlotAircraftSim(t5, y5, [Z_c; 0; 0; 0], fig(21:25), col)
PlotAircraftSim(t11, y11, [Z_c; L_c; M_c; N_c], fig(21:25), col2)
hold off;

% f
```

```
hold on;
PlotAircraftSim(t6, y6, [Z_c; 0; 0; 0], fig(26:30), col)
PlotAircraftSim(t12, y12, [Z_c; L_c; M_c; N_c], fig(26:30), col2)
hold off;
```

D. QuadrotorEOM Linearized

```
function var_dot = QuadrotorEOM_Linearized(t, var, g, m, I, deltaFc, deltaGc)

 x = var(1);
 y = var(2);
 z = var(3);

 phi = var(4);
 theta = var(5);
 psi = var(6);

 u = var(7);
 v = var(8);
 w = var(9);

 p = var(10);
 q = var(11);
 r = var(12);

 L_c = deltaGc(1);
 M_c = deltaGc(2);
 N_c = deltaGc(3);

 Z_c = deltaFc(3);

 var_dot(1) = u;
 var_dot(2) = v;
 var_dot(3) = w;
 var_dot(4) = p;
 var_dot(5) = q;
 var_dot(6) = r;
 var_dot(7) = g * -theta;
 var_dot(8) = g * phi;
 var_dot(9) = (1 / m) * Z_c;
 var_dot(10) = 1/I(1) * L_c;
 var_dot(11) = 1/I(2) * M_c;
 var_dot(12) = 1/I(3) * N_c;

 var_dot = var_dot';

end
```

E. Rotation Derivative Feedback

```
function [Fc, Gc] = RotationDerivativeFeedback(var, m, g)
    v = vars;
    Zc = m * g;
    Fc = [0, 0, Zc];
    Gc = -v.ang_gain * [var(10), var(11), var(12)]';

end
```

F. Compute Motor Forces

```
function motor_forces = ComputeMotorForces(Fc, Gc, d, km)

A = [ -1, -1, -1, -1;
        -d/sqrt(2), -d/sqrt(2), d/sqrt(2), d/sqrt(2);
        d/sqrt(2), -d/sqrt(2), -d/sqrt(2), d/sqrt(2);
        km, km, km, km;
      ];

X = [Fc; Gc(1); Gc(2); Gc(3);];

motor_forces = A' * X;

end
```

G. QuadrotorEOM with Rate Feedback

```
function var_dot = QuadrotorEOMwithRateFeedback(t, var, g, m, I, nu, mu)
     v = vars;
    [F_c, G_c] = RotationDerivativeFeedback(var, v.m, v.g);

    x = var(1);
    y = var(2);
    z = var(3);

    phi = var(4);
    theta = var(5);
    psi = var(6);

    u_E = var(7);
    v_E = var(8);
    w_E = var(9);

    p = var(10);
    q = var(11);
    r = var(12);

    V_a = sqrt(u_E^2 + v_E^2 + w_E^2);

    X = -v.nu * V_a *u_E;
    Y = -v.nu * V_a * v_E;
    Z = -v.nu * V_a * w_E;

    F_mag = sqrt(p^2 + q^2 + r^2);

    % L = -v.mu * F_mag * p; problem 2
    % M = -v.mu * F_mag * q;
    % N = -v.mu * F_mag * r;

    L = 0;
    M = 0;
    N = 0;

    L_c = G_c(1);
    M_c = G_c(2);
    N_c = G_c(3);

    f1 = v.m / 4;
    f2 = f1;
    f3 = f1;
    f4 = f1;
    motor_forces = [f1; f2; f3; f4];

    Z_c = -sum(motor_forces);

    % EOM

    x_dot = [cos(theta) * cos(psi), sin(phi) * sin(theta) * cos(psi) - cos(phi
        ) * sin(psi), cos(phi) * sin(theta) * cos(psi) + sin(phi) * sin(psi)];
    y_dot = [cos(theta) * sin(psi), sin(phi) * sin(theta) * sin(psi) + cos(phi
        ) * cos(psi), cos(phi) * sin(theta) * sin(psi) - sin(phi) * cos(psi)];
    z_dot = [-sin(theta), sin(phi) * cos(theta), cos(phi) * cos(theta)];

    v_dot = [x_dot ; y_dot; z_dot] * [u_E; v_E; w_E];

    phi_dot =   [1, sin(phi) * tan(theta), cos(phi) * tan(theta)];
    theta_dot = [0, cos(phi), -sin(phi)];
    psi_dot =   [0, sin(phi) * sec(theta), cos(phi) * sec(theta)];

    euler_rates = [phi_dot; theta_dot; psi_dot] * [p; q; r];

    accel = [(r * v_E) - (q * w_E); (p * w_E) - (r * u_E); (q * u_E) - (p *
        v_E)];
    grav = g * [-sin(theta); cos(theta) * sin(phi); cos(theta) * cos(phi)];
    aero = (1/m) * [X; Y; Z];
    control = (1/m) * [0; 0; Z_c];

    vel_rates = accel + grav + aero + control;

    omega_1 = [((I(2) - I(3)) / I(1)) * q * r; ((I(3) - I(1)) / I(2)) * p * r;
         ((I(1) - I(2)) / I(3)) * p * q];
    omega_2 = [L/I(1); M/I(2); N/I(3)];
    omega_3 = [L_c/I(1); M_c/I(2); N_c/I(3)];

    angular_rates = omega_1 + omega_2 + omega_3;

    var_dot = [v_dot(1); v_dot(2); v_dot(3); euler_rates(1); euler_rates(2);
        euler_rates(3); vel_rates(1); vel_rates(2); vel_rates(3);
        angular_rates(1); angular_rates(2); angular_rates(3)];

end
```

H. Specific Code for 2.5

```
% ASEN 3801
% Lab 4
% names

%% Housekeeping

clc; clear; close all;

%% Variable declarations

v = vars;

f1 = v.m / 4;
f2 = f1;
f3 = f1;
f4 = f1;

motor_forces = [f1; f2; f3; f4];

Z_c = -sum(motor_forces);

tspan = [1 10];
time = tspan;

F_c = [0, 0, Z_c];
G_c = [0, 0, 0];

Z_c = Z_c * ones(length(time), 1);
L_c = G_c(1) * ones(length(time), 1);
M_c = G_c(2) * ones(length(time), 1);
N_c = G_c(3) * ones(length(time), 1);

deltaFc = [0, 0, 0]';
deltaGc = [0, 0, 0]';

fig = 1:1:30;

col = 'b-';
col2 = 'g-';

%% Lab Task 1

initial_state = zeros(12, 1);
initial_state_1 = [0, 0, -1, 0, 0, 0, 0, 0, 0, deg2rad(0.1), 0, 0];
initial_state_2 = [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, deg2rad(0.1), 0];
initial_state_3 = [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, deg2rad(0.1)];


[t1, y1] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_1);
[t2, y2] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_2);
[t3, y3] = ode45(@(t, y) QuadrotorEOM(t, y, v.g, v.m, [v.Ix; v.Iy; v.Iz], v.d,
    v.km, v.nu, v.mu, motor_forces), tspan, initial_state_3);

[t4, y4] = ode45(@(t, y) QuadrotorEOMwithRateFeedback(t, y, v.g, v.m, [v.Ix; v
    .Iy; v.Iz], deltaFc, deltaGc), tspan, initial_state_1);
[t5, y5] = ode45(@(t, y) QuadrotorEOMwithRateFeedback(t, y, v.g, v.m, [v.Ix; v
    .Iy; v.Iz], deltaFc, deltaGc), tspan, initial_state_2);
[t6, y6] = ode45(@(t, y) QuadrotorEOMwithRateFeedback(t, y, v.g, v.m, [v.Ix; v
    .Iy; v.Iz], deltaFc, deltaGc), tspan, initial_state_3);

% 2.1d
% hold on;
% PlotAircraftSim(t1, y1, [Z_c; 0; 0; 0], fig(1:5), col)
% PlotAircraftSim(t4, y4, [Z_c; L_c; M_c; N_c], fig(1:5), col2)
% hold off;

% 2.1e
% hold on;
% PlotAircraftSim(t2, y2, [Z_c; 0; 0; 0], fig(6:10), col)
% PlotAircraftSim(t5, y5, [Z_c; L_c; M_c; N_c], fig(6:10), col2)
% hold off;

% 2.1f
hold on;
PlotAircraftSim(t3, y3, [Z_c; 0; 0; 0], fig(11:15), col)
PlotAircraftSim(t6, y6, [Z_c; L_c; M_c; N_c], fig(11:15), col2)
hold off;
```

I. Vars Class

```
classdef vars
    % Aircraft properties

    properties
        m = 0.068; % [kg]
        g = -9.81; % [m/s^2]
        d = 0.060; % [m]
        km = 0.0024; % [N*M/N]
        Ix = 5.8e-5; % [kg*m^2]
        Iy = 7.2e-5; % [kg*m^2]
        Iz = 1e-4;  % [kg*m^2]
        %I = diag([Ix; Iy; Iz]);
        nu = 1e-3; % [N/(m/s)^2]
        mu = 2e-6; % [N*m/(rad/s)^2]
        ang_gain = 0.004; % [N*m/(rad/s)]
    end

    methods
        %%
    end
end
```