ASEN 3128 - Lab Deliverable 3

Cole MacPherson, Donny Wolfe, Sam Packard, Dawei Zhao
March 5th, 2020

Contents

I. Problem 3 Question 1

Yes, the results found from problem 3 of the lab does make sense as the quad-rotor was to be modeled in steady equilibrium and that is what is seen in all of the linearized plots. For every plot with variables associated with steady equilibrium flight, those variables were found to be constant and don't vary with time. These variables that were found to be constant throughout time and need to be constant throughout time to ensure steady equilibrium flight include; Euler angles, angular velocity, and the control forces and moments. Thus, the behavior does make sense.

II. Problem 3 Question 2

Yes, steady hover is a stable flight condition, as the variables mentioned above (Euler angles, angular velocity, and the control forces and moments) are constant throughout time. While in a steady a quad-rotor is not changing its orientation at all and thus, its Euler angles are not changing. Additionally, it is not spinning about any of its axes and thus it has no angular velocity. Lastly, there is no no motion for the quad-rotor and thus the control forces and moments are constant. With all of the above knowledge it can be understood that the quad-rotor is in stable equilibrium flight while in a steady hover.

III. Problem 5 Question 1

The addition of the feedback control law to the non-linearized model, makes it so that the non-linearized model now acts the same as the linearized model in the sense that the results demonstrate steady equilibrium flight for the quad-rotor given the roll, pitch, and yaw rates. Without this control law in the non-linearized model the output would be such that that the quad-rotor would not be in equilibrium flight as demonstrated in our figures comparing the uncontrolled non-linearized model and the linearized model.

IV. Figures

A. Problem 3.a and 4.a

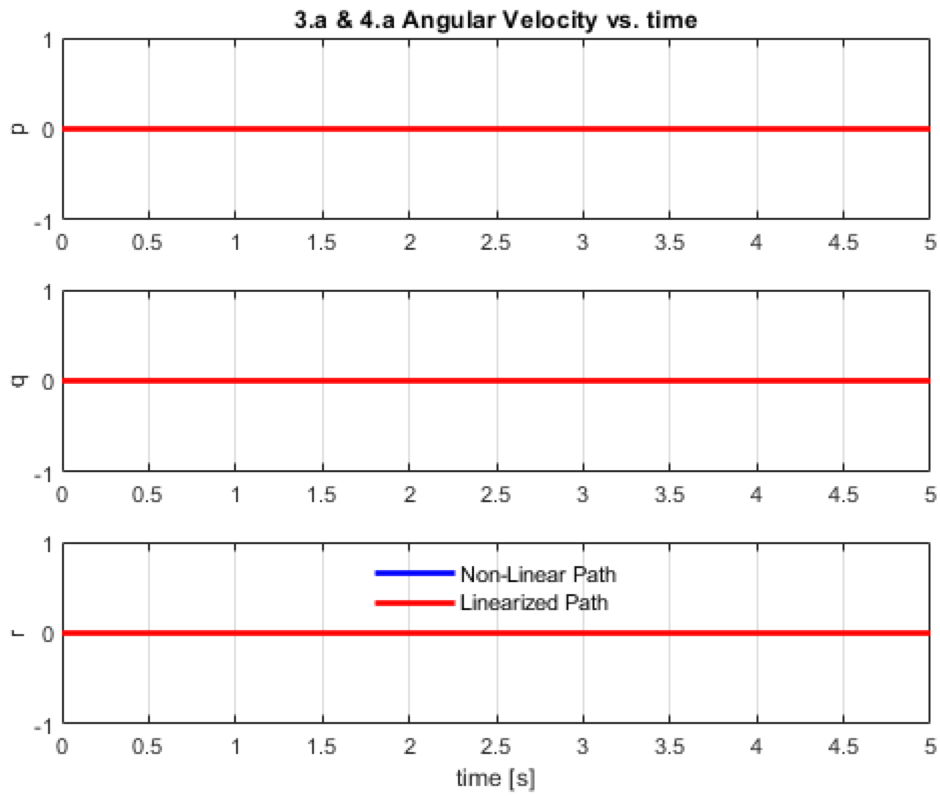

Fig. 1 Angular Velocity

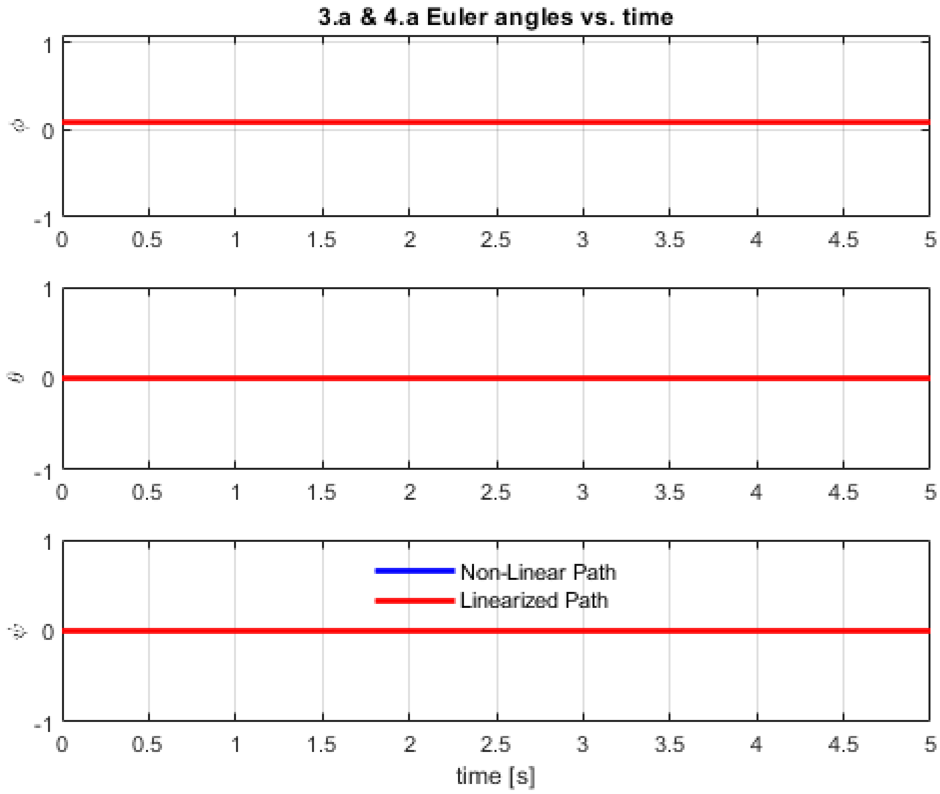

Fig. 2   Euler Angles

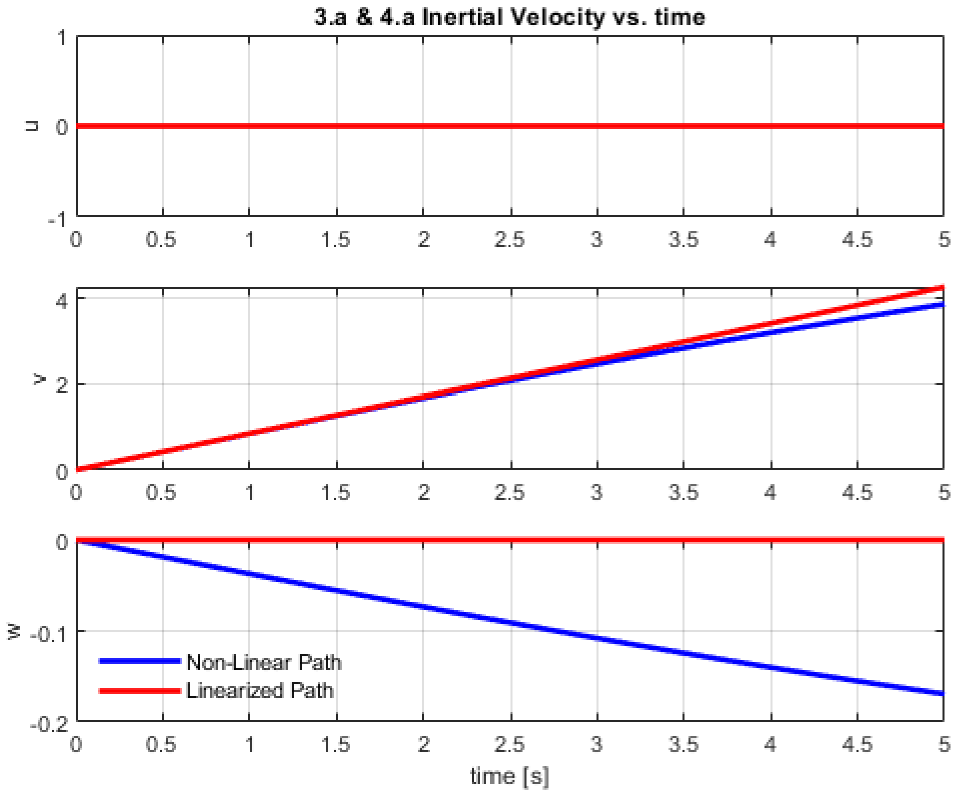

Fig. 3 Inertial Velocity

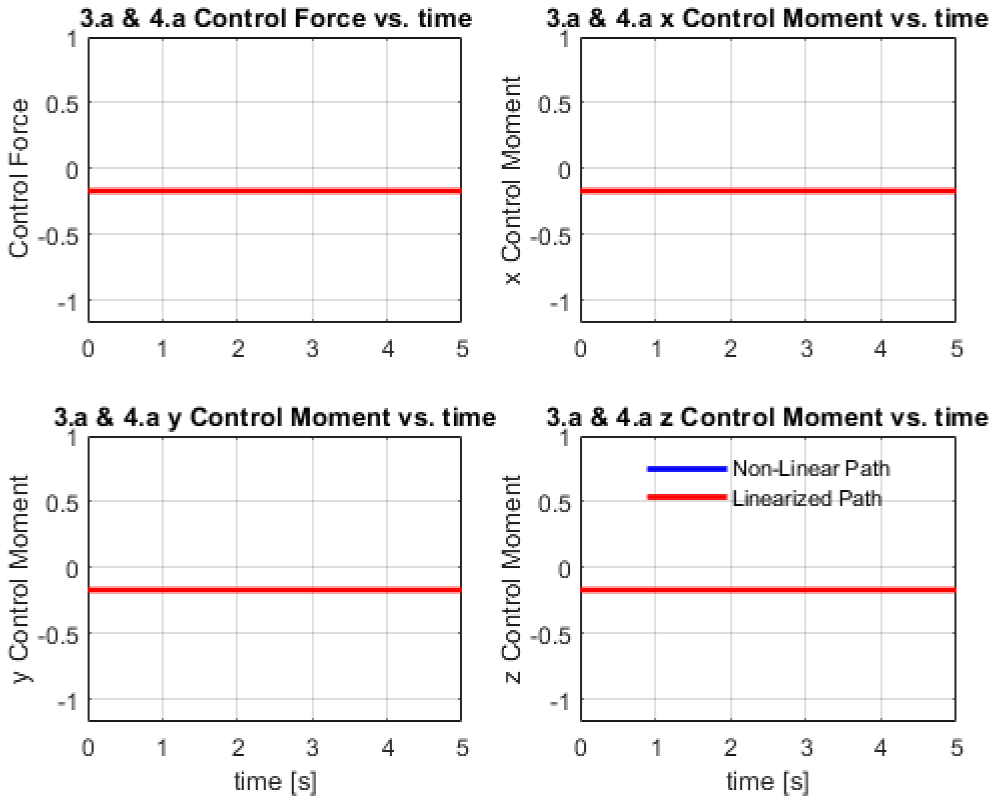

Fig. 4 Control Forces

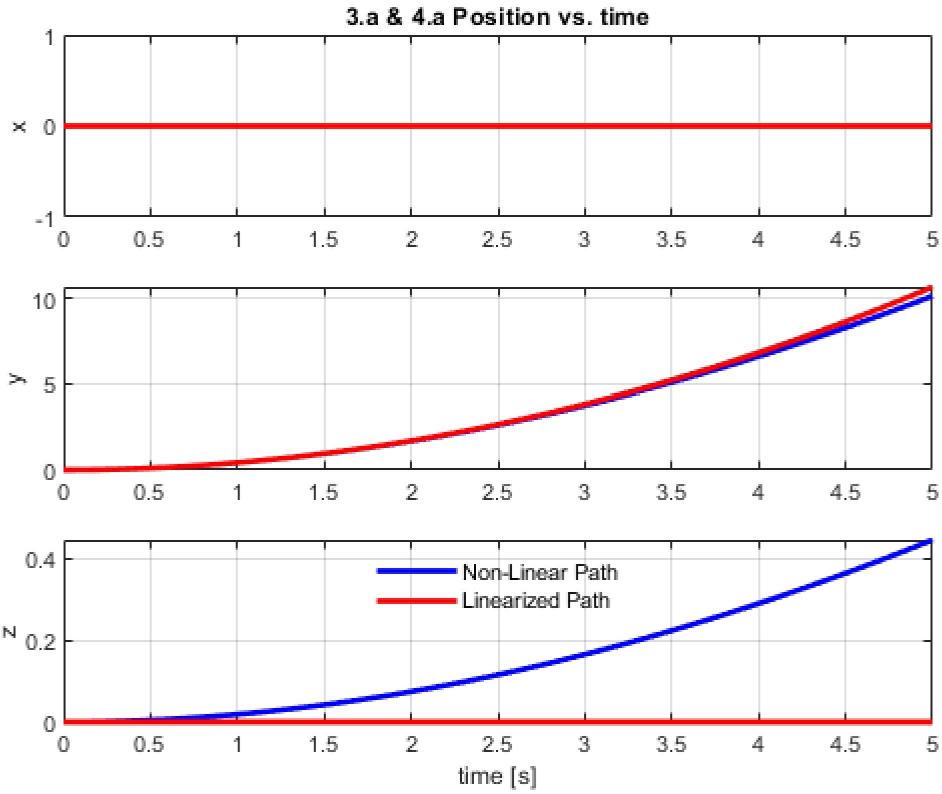

Fig. 5 Flight Path

B. Problem 3.b and 4.b

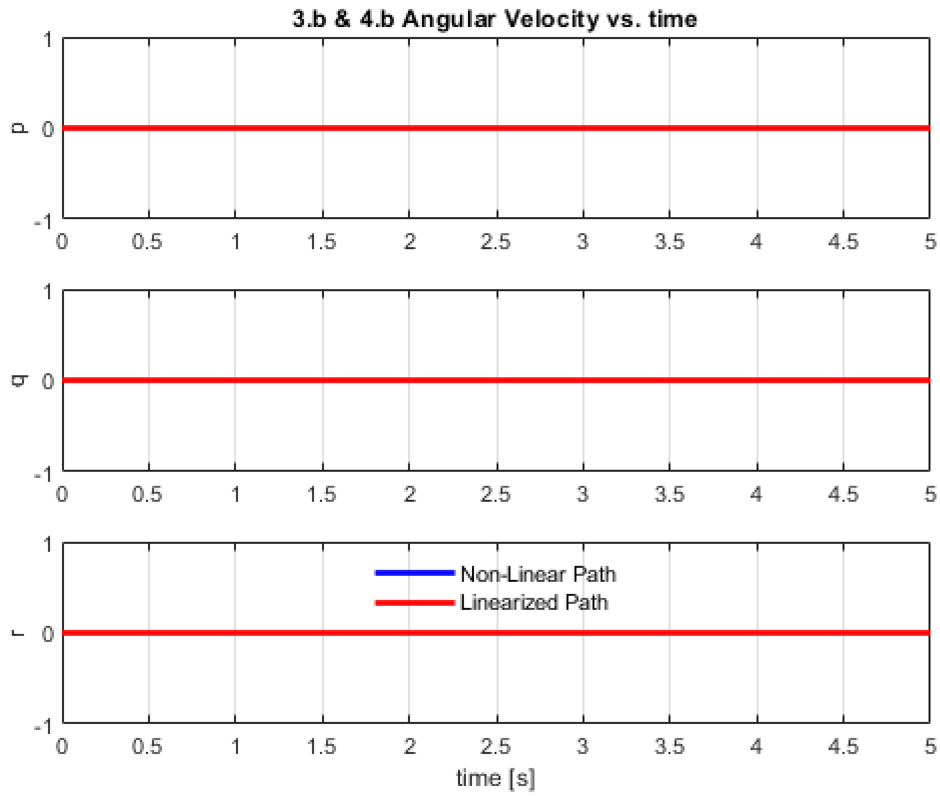

Fig. 6 Angular Velocity

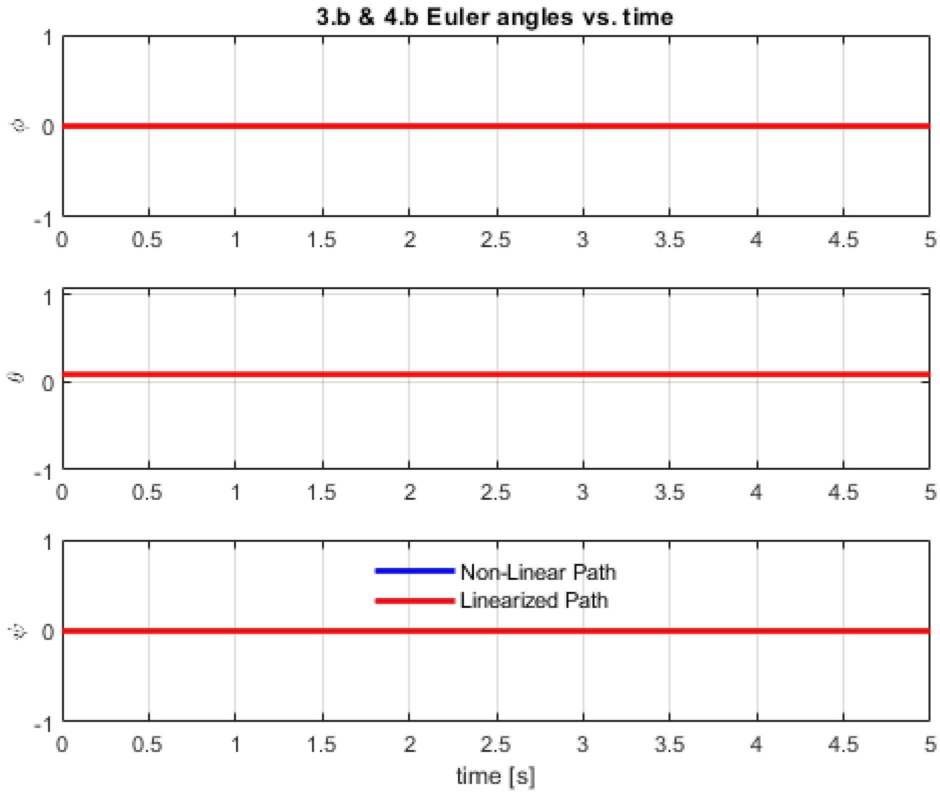

Fig. 7 Euler Angles

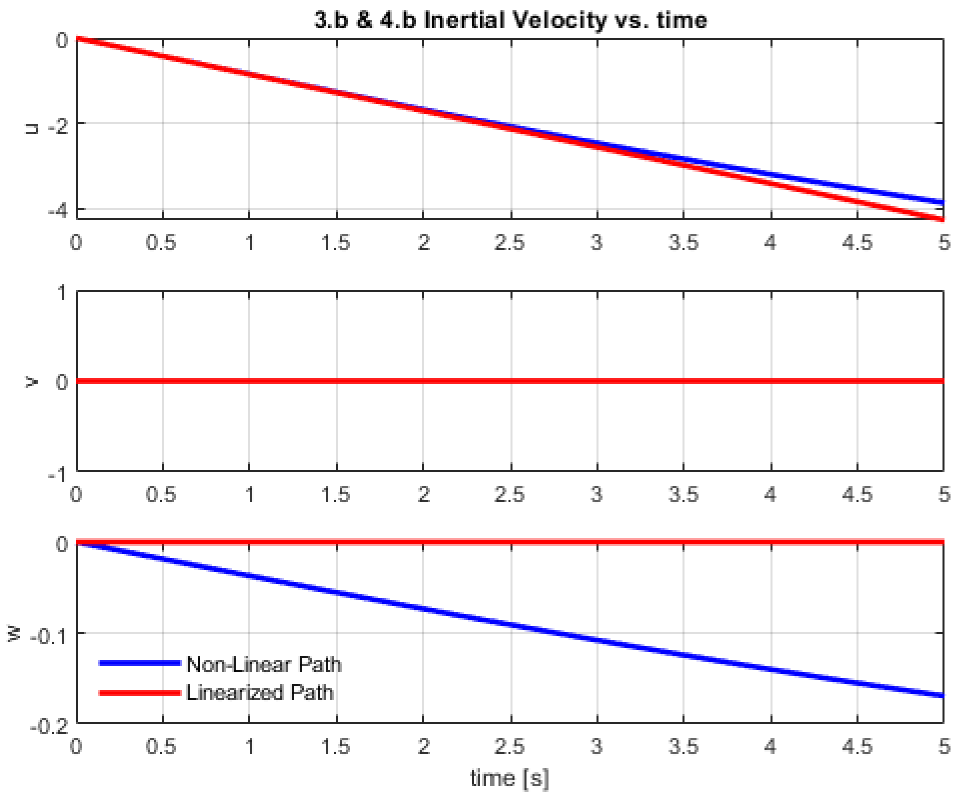

Fig. 8 Inertial Velocity

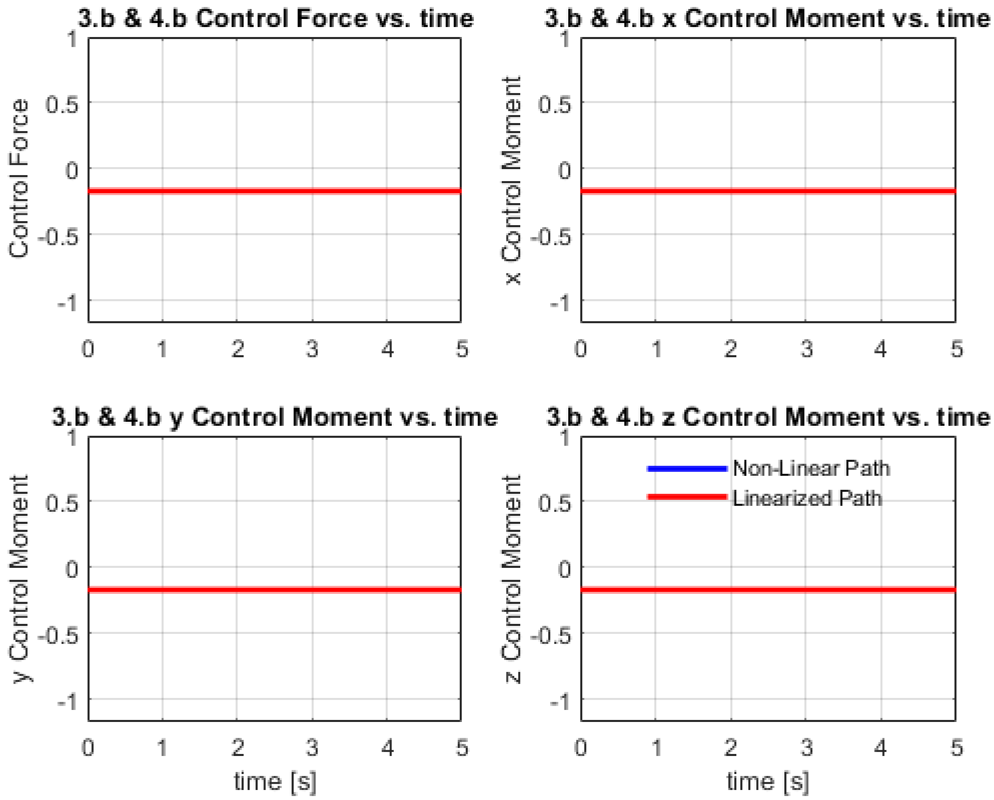

Fig. 9 Control Forces

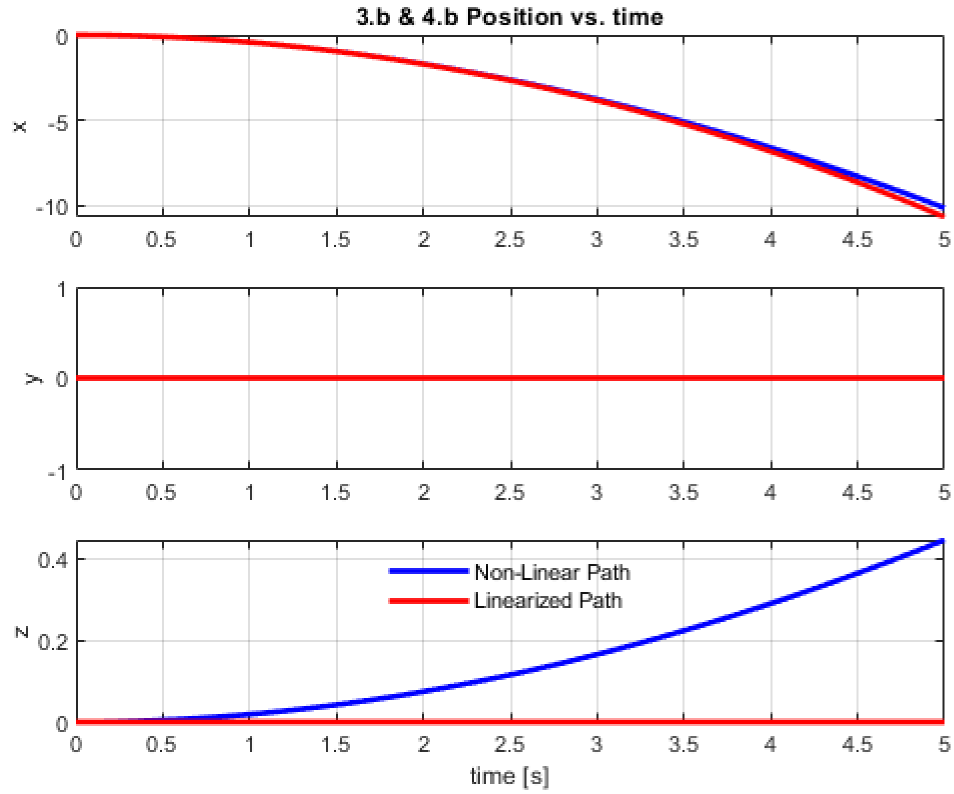

Fig. 10 Flight Path

C. Problem 3.c and 4.c

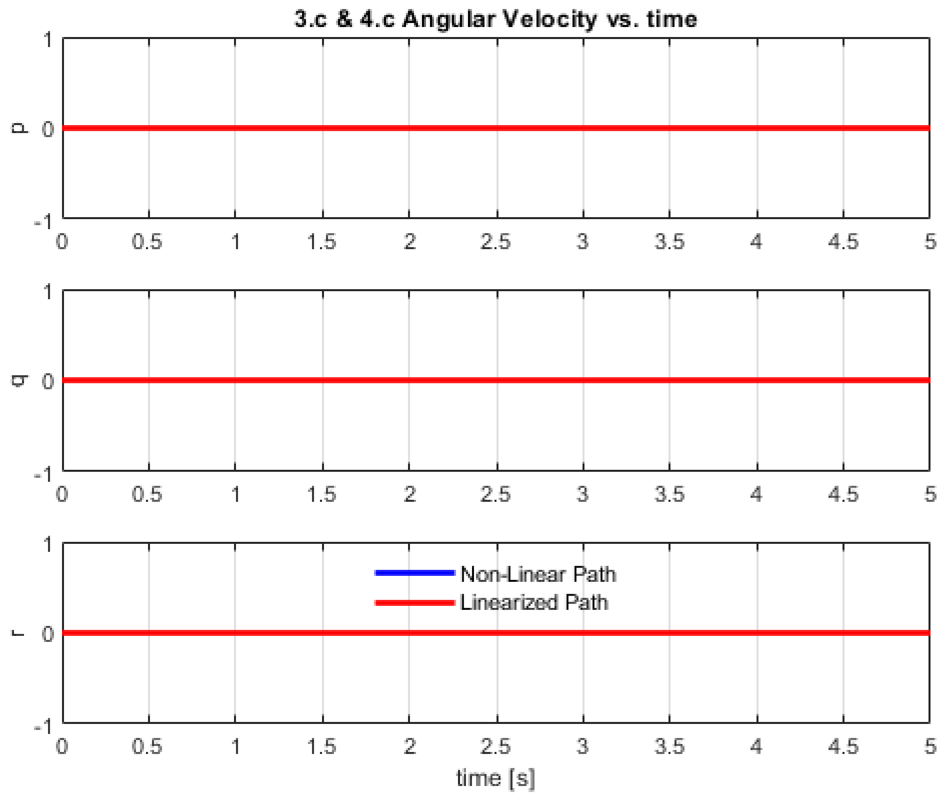

Fig. 11 Angular Velocity

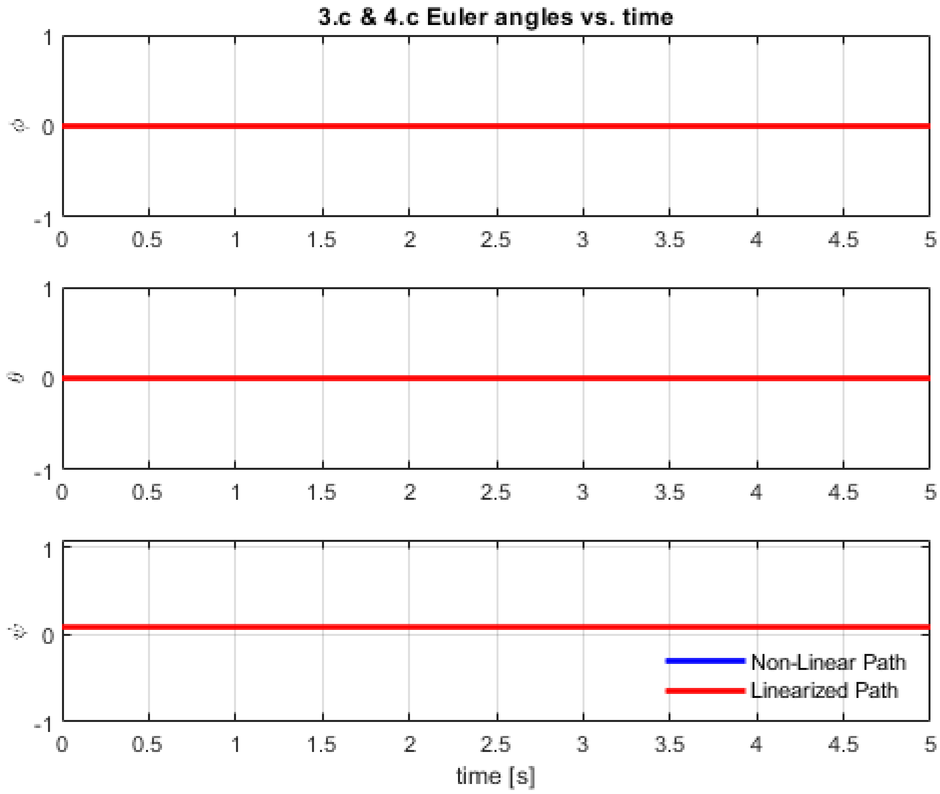

Fig. 12 Euler Angles

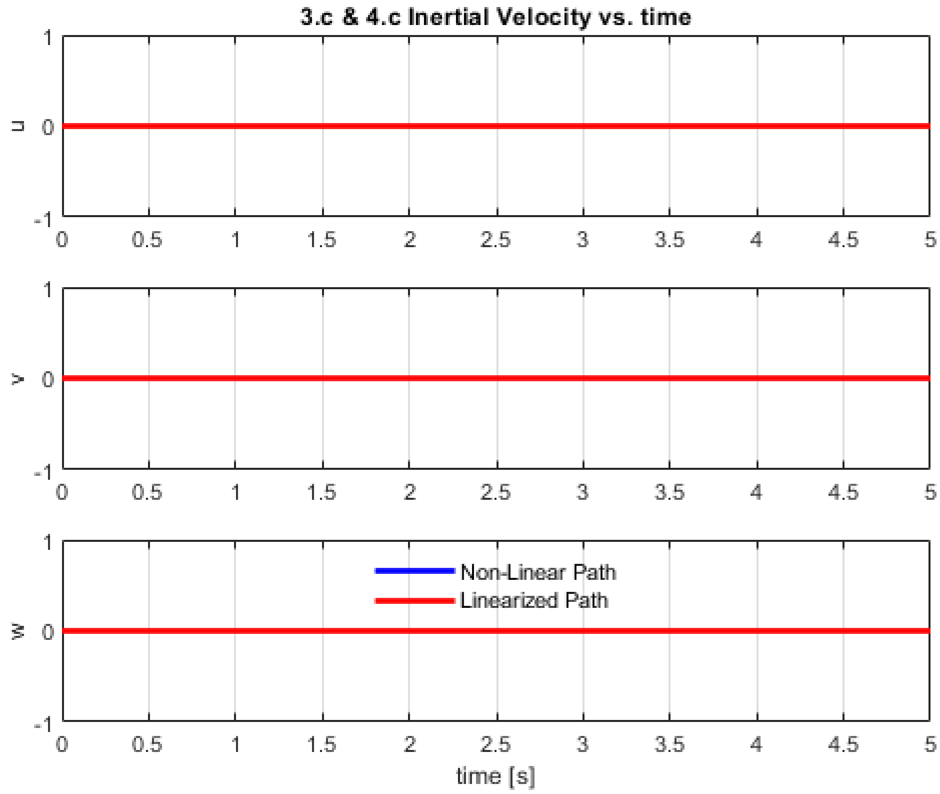

Fig. 13 Inertial Velocity

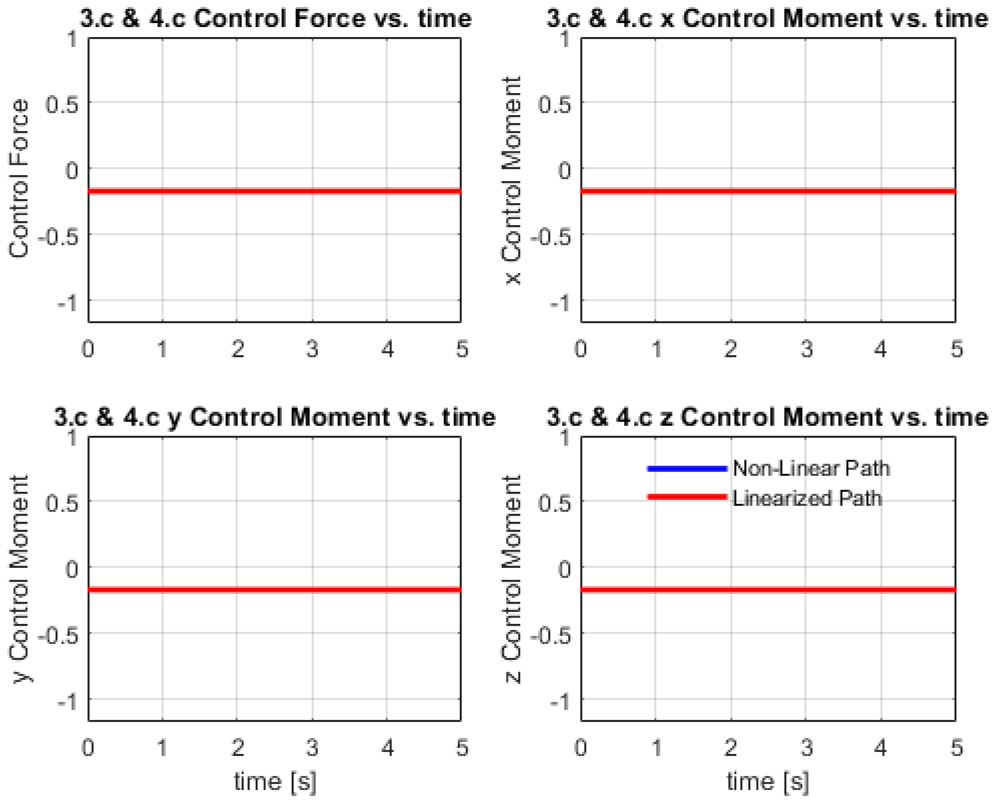

Fig. 14 Control Forces

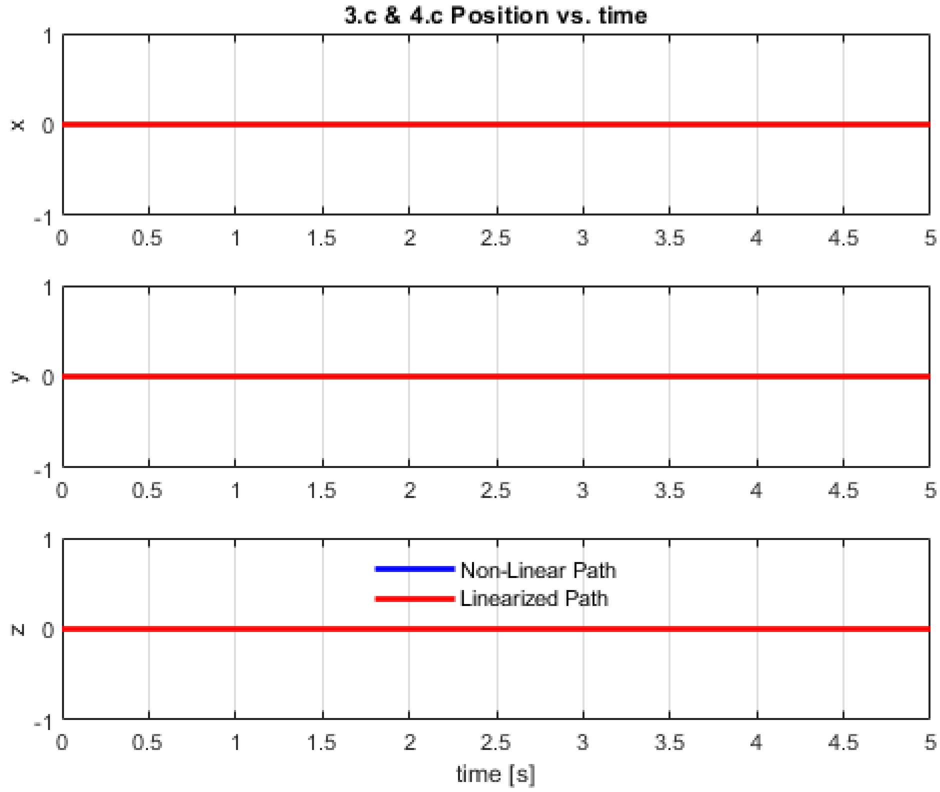

Fig. 15 Flight Path

D. Problem 3.d and 4.d

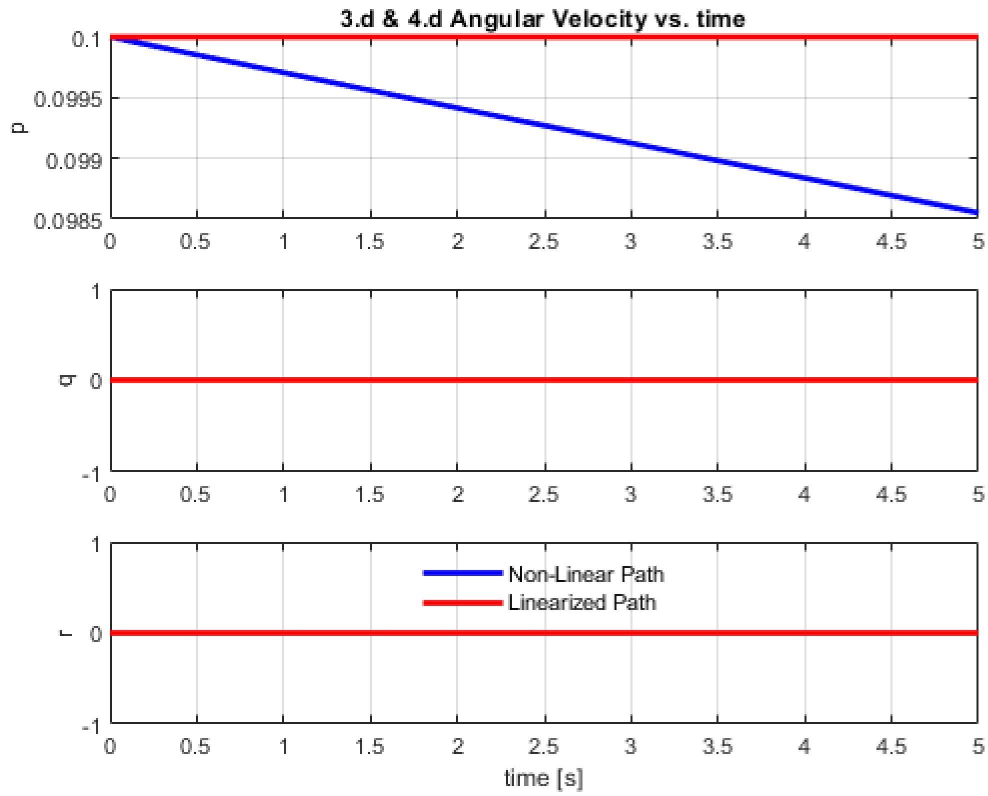

Fig. 16 Angular Velocity

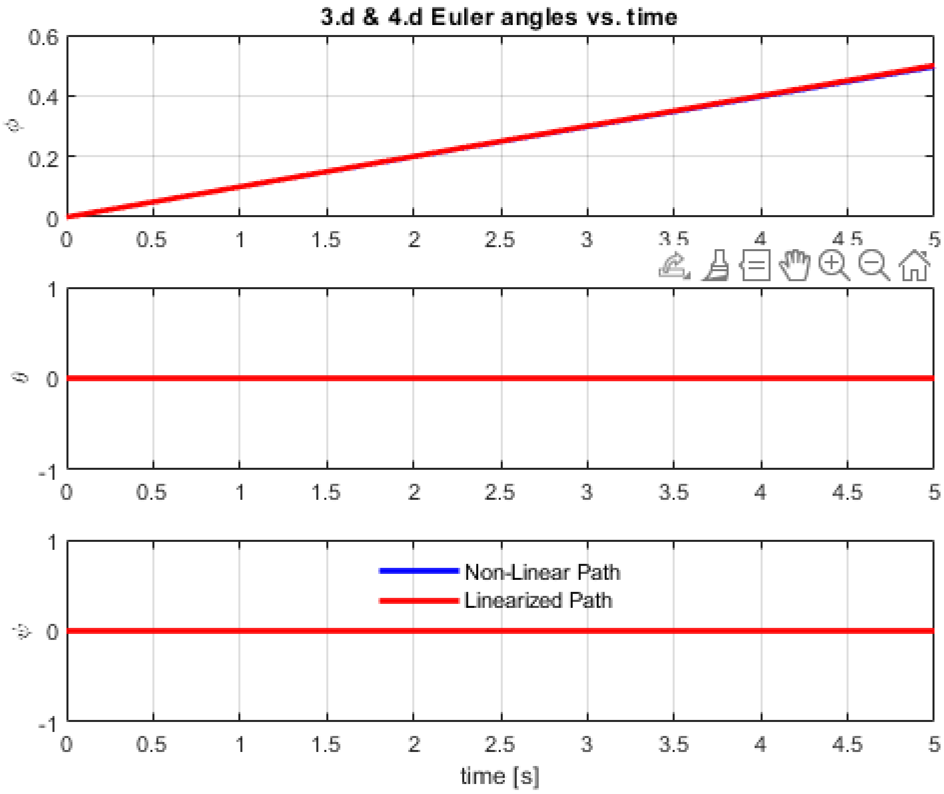

Fig. 17 Euler Angles

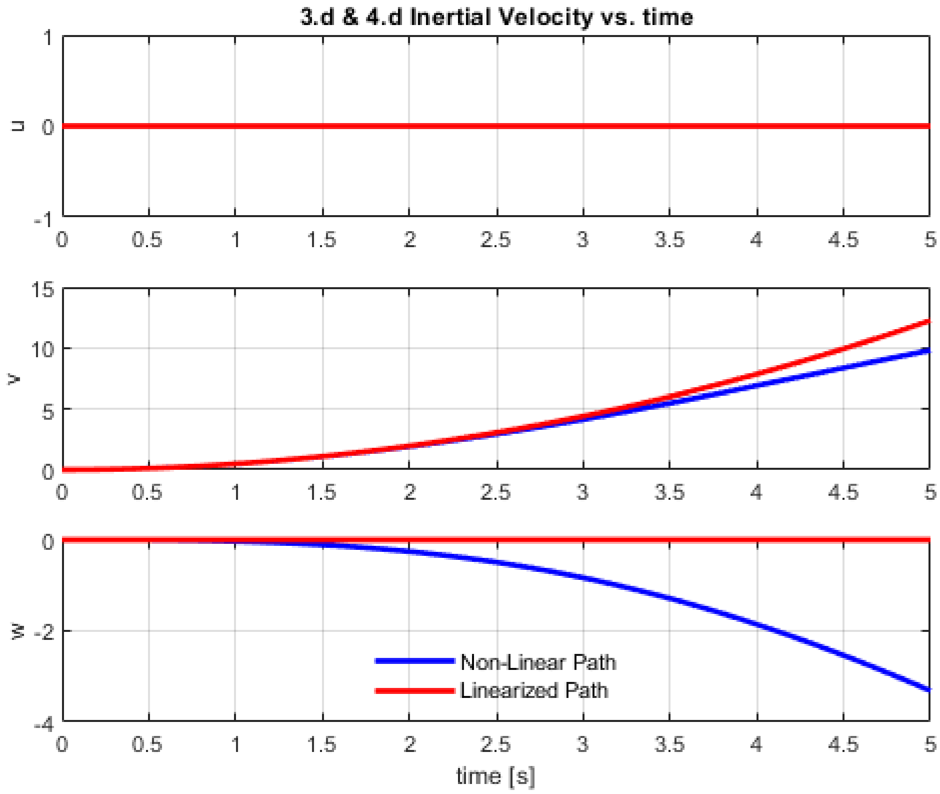

Fig. 18 Inertial Velocity

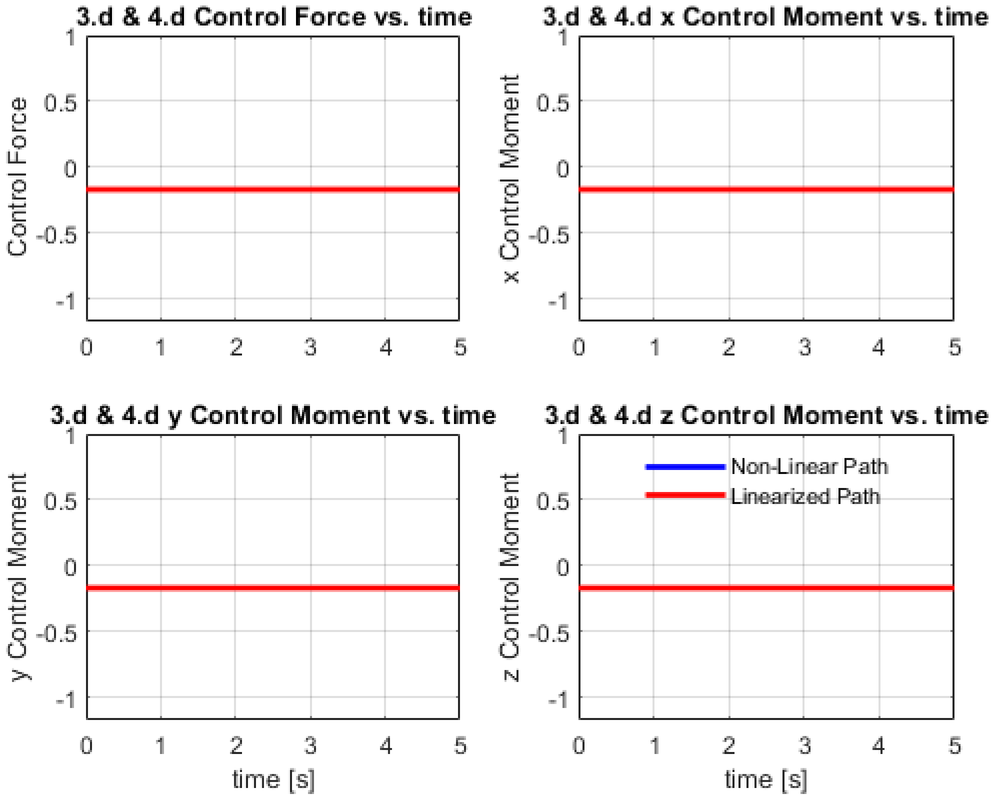

Fig. 19 Control Forces

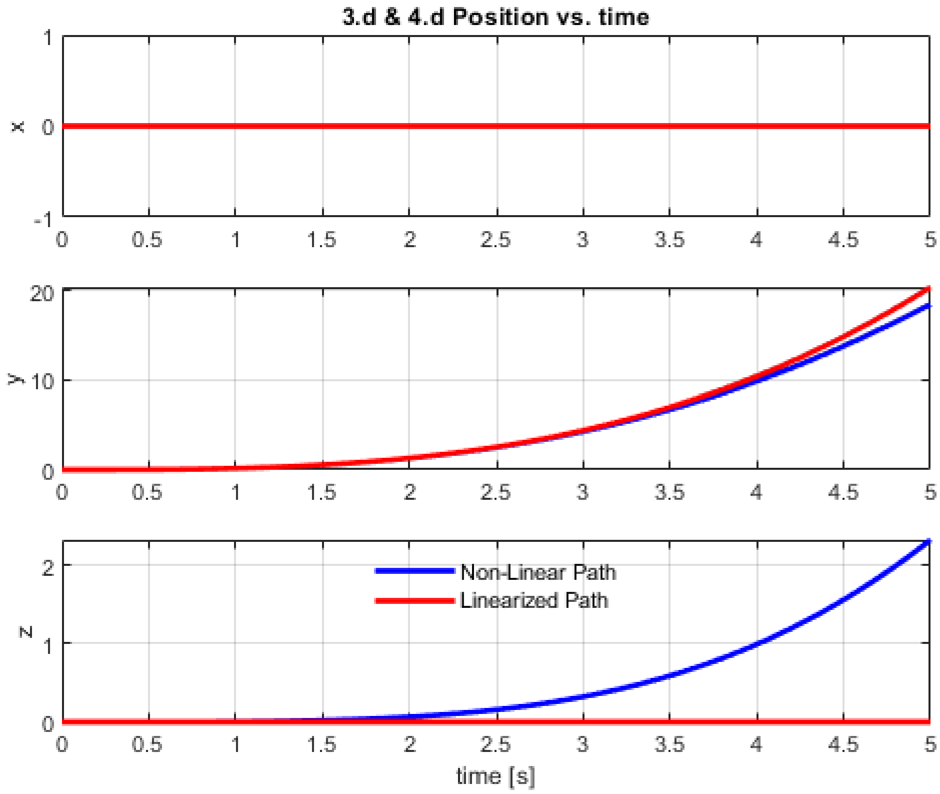

Fig. 20 Flight Path

E. Problem 3.e and 4.e

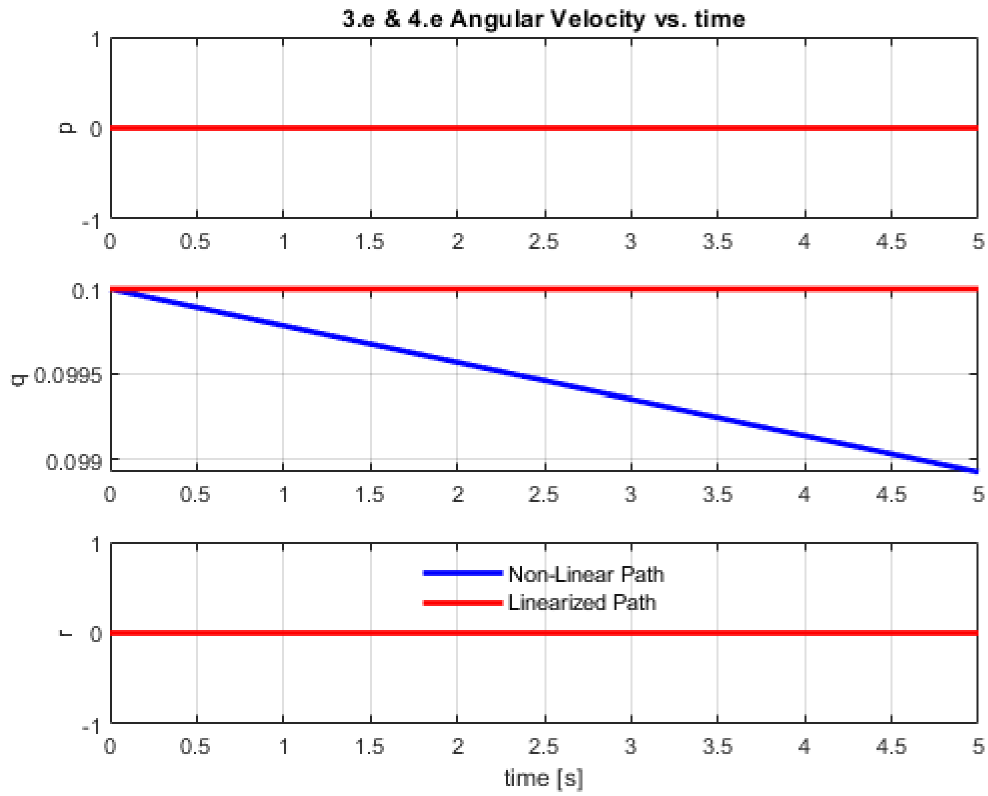

Fig. 21 Angular Velocity

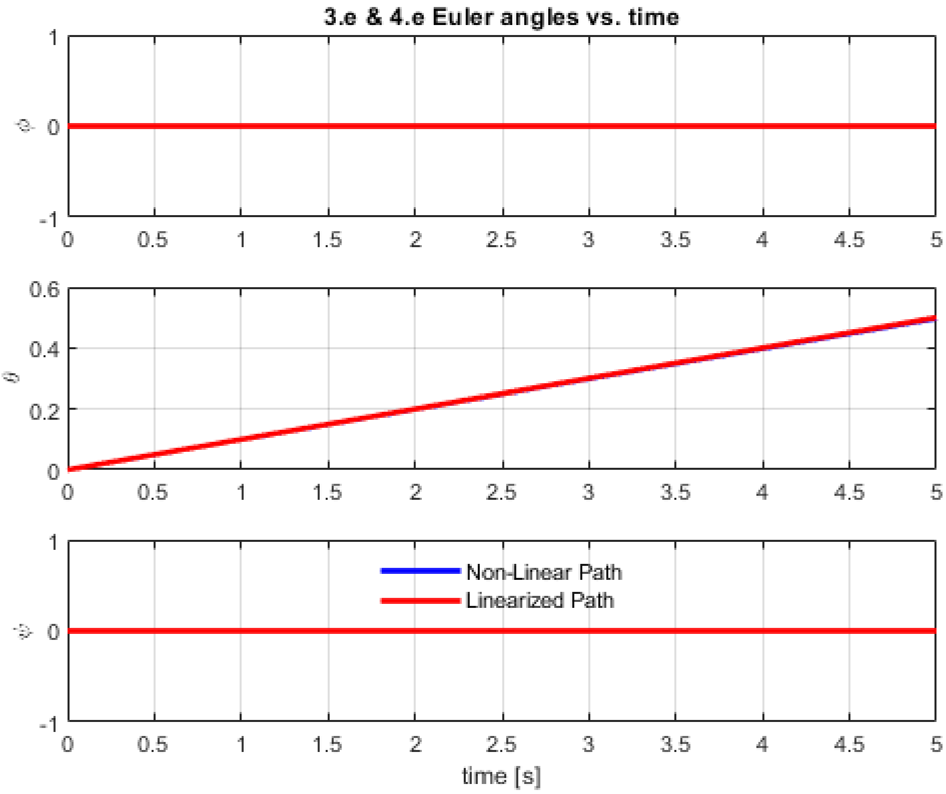

Fig. 22 Euler Angles

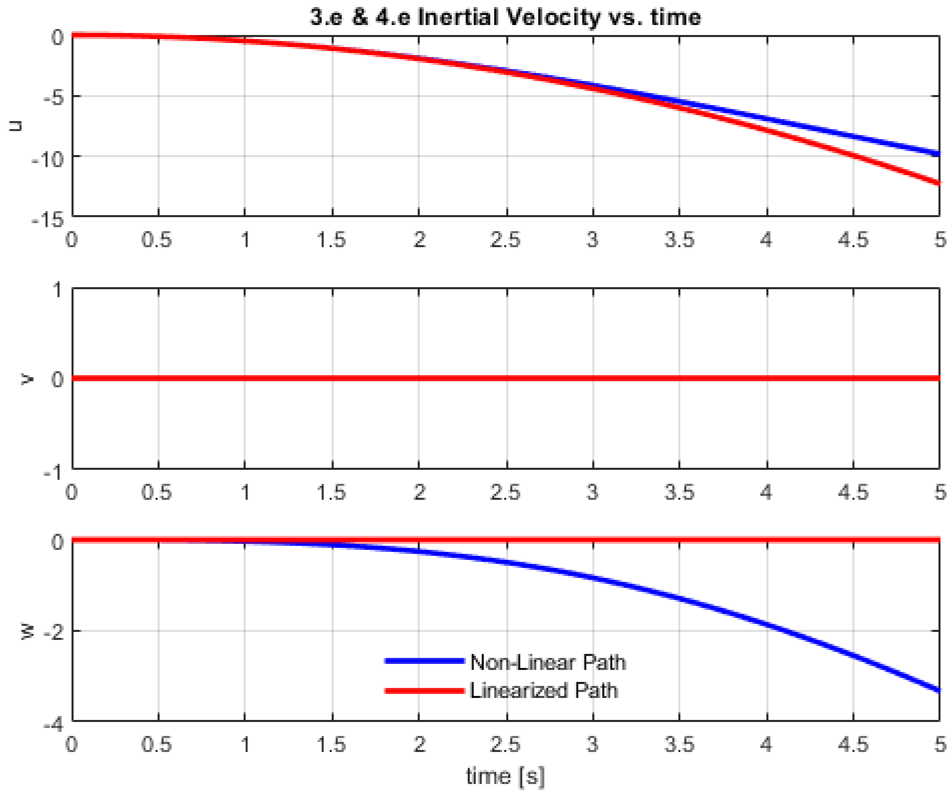

Fig. 23 Inertial Velocity

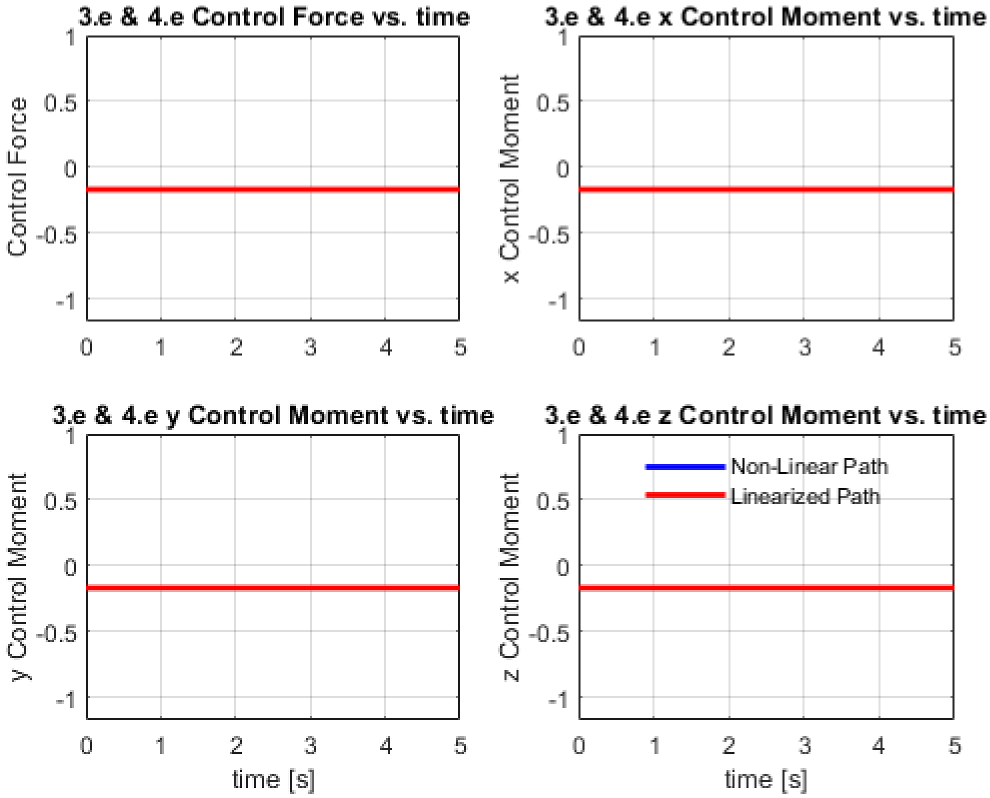

Fig. 24 Control Forces

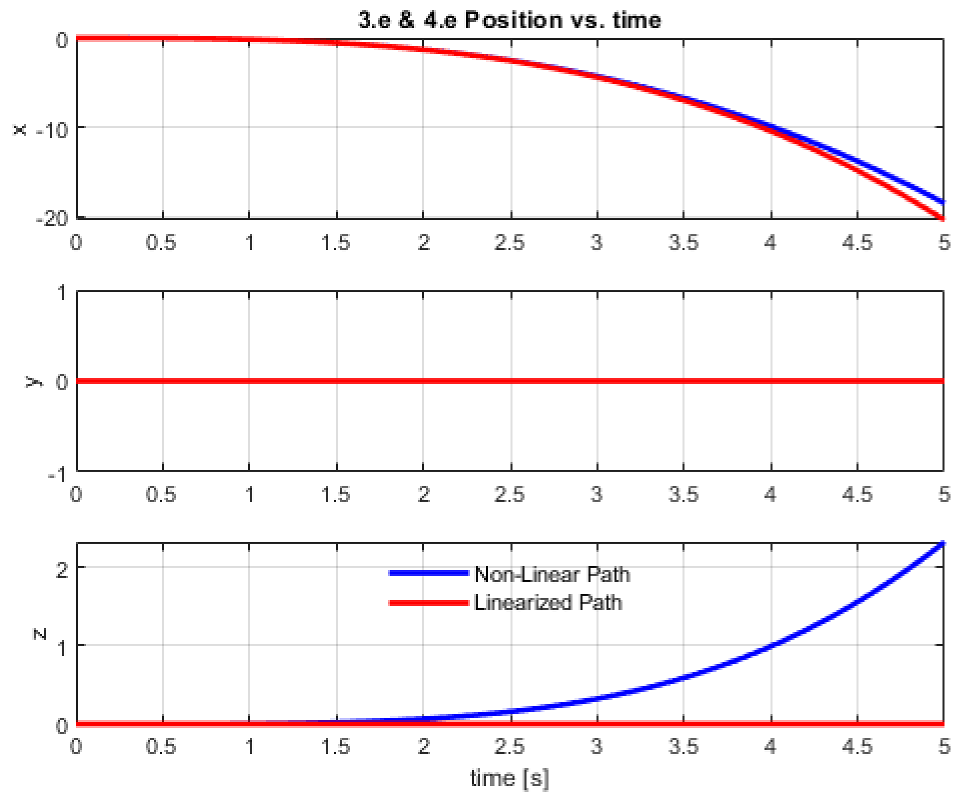

Fig. 25 Flight Path

F. Problem 3.f and 4.f

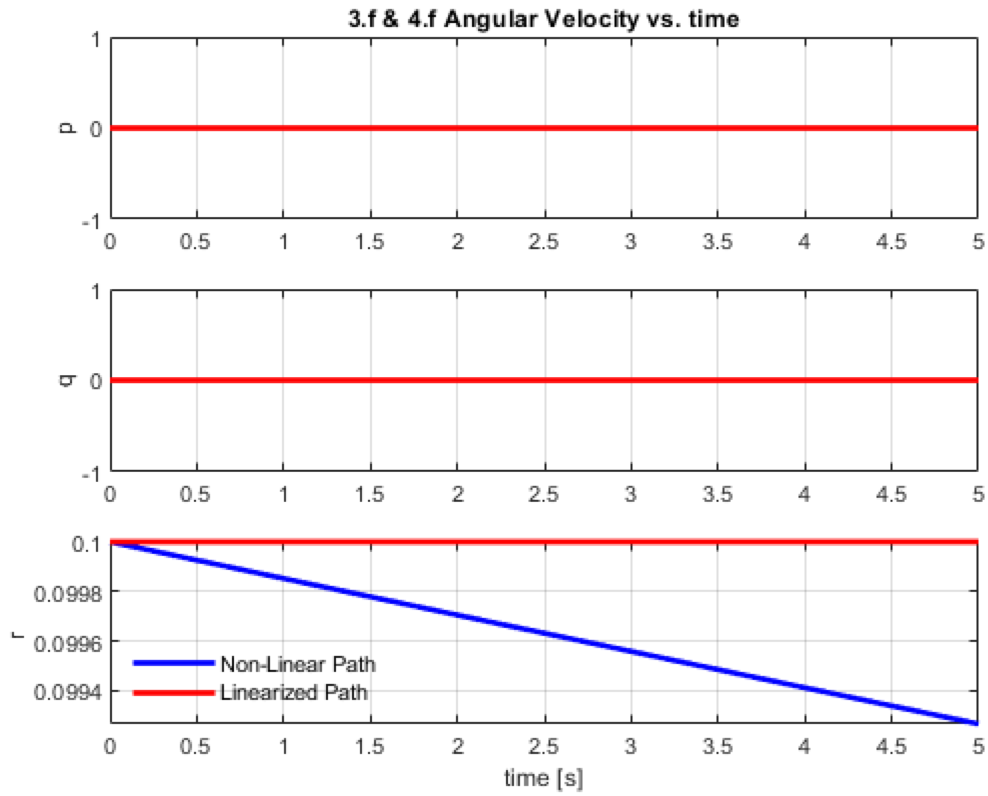

Fig. 26 Angular Velocity

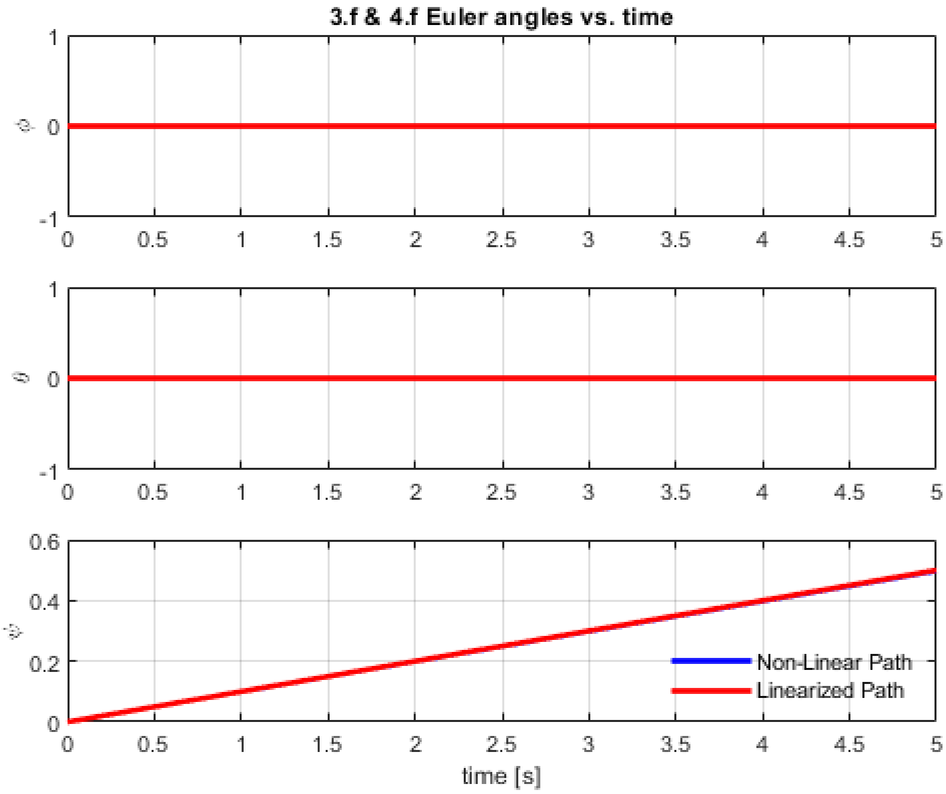

Fig. 27 Euler Angles

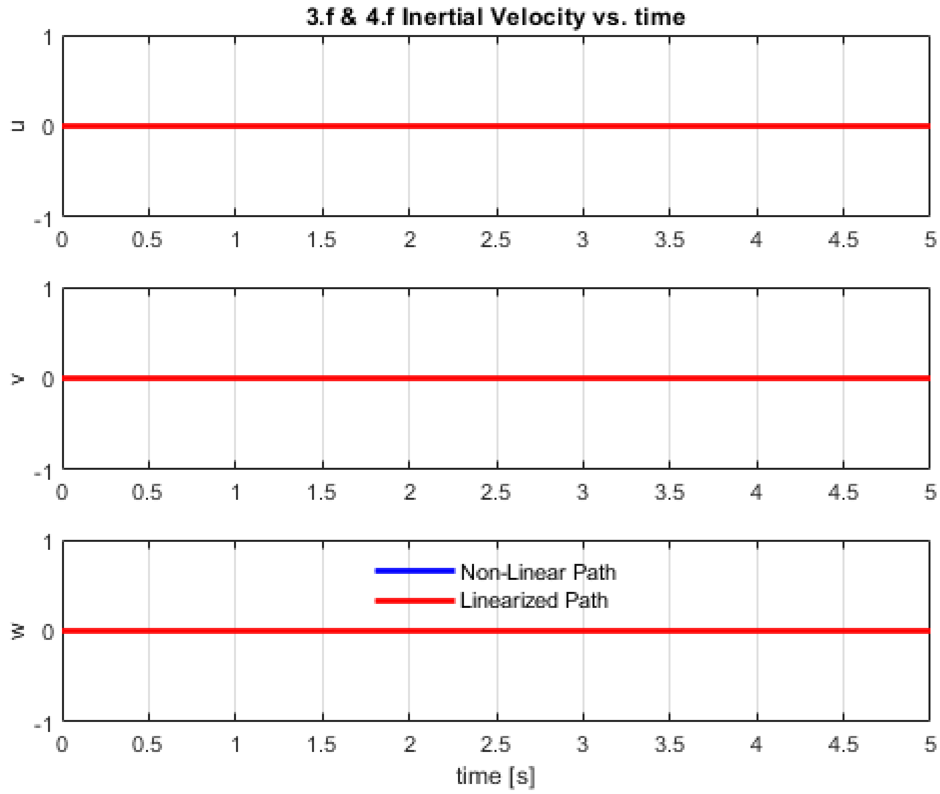

Fig. 28 Inertial Velocity

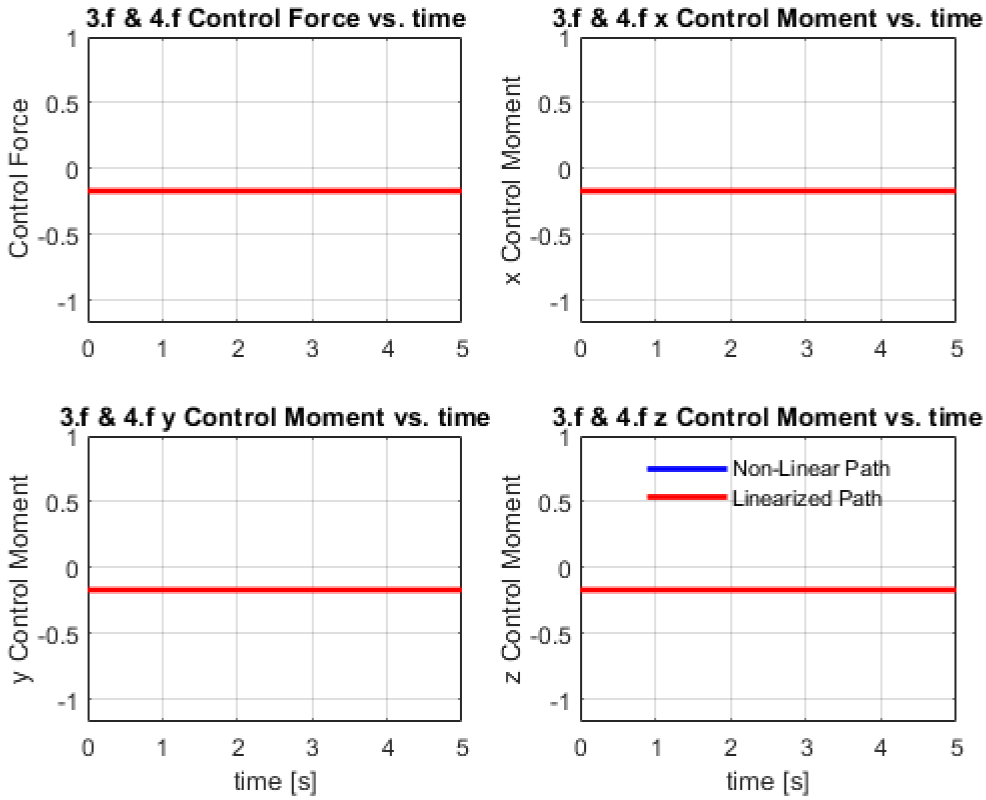

Fig. 29 Control Forces

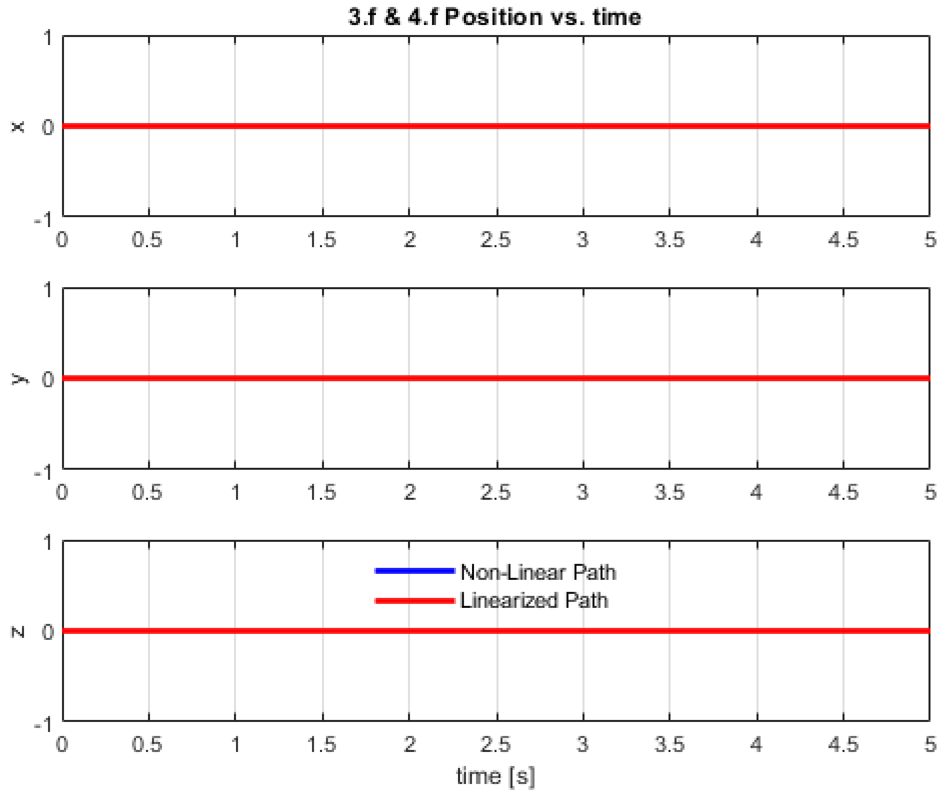

Fig. 30 Flight Path

G. Problem 5

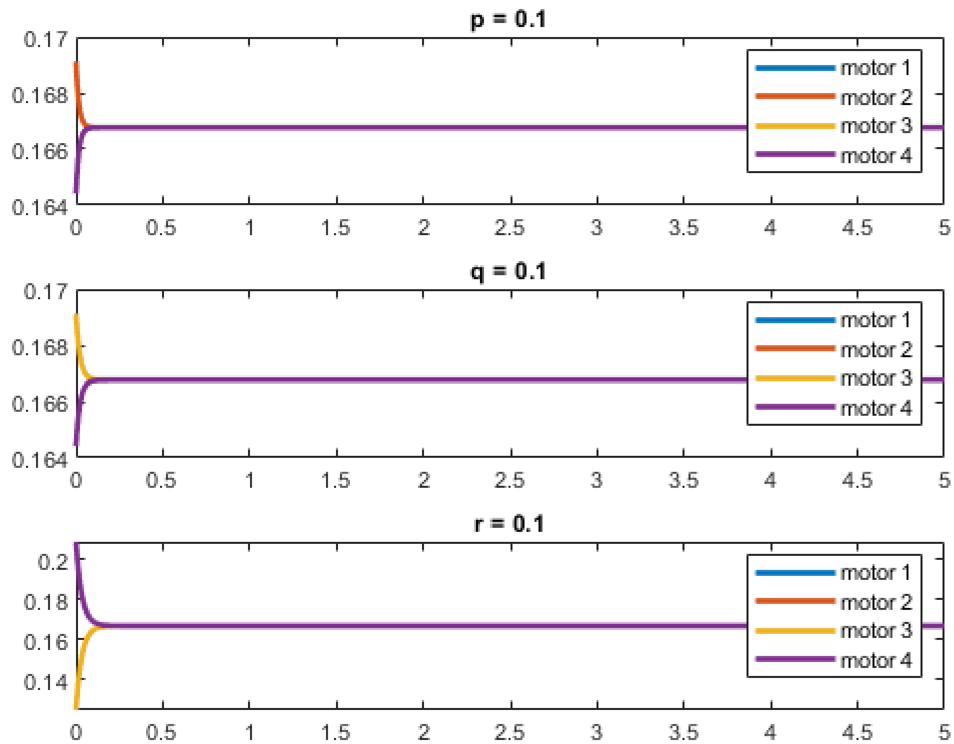

Fig. 31 Motor Forces with Feedback Control

V. Appendix

A. MATLAB Code

Main code

```
%% ASEN 3128 - LAb 3 - Main
% Script to compare the linearized and non-linearized models of a quadrotor
% in steady equilibrium flight. Additionally, control moments and forces
% are to be added to model how a quadrotor remains in steady hover flight
%
% Author: Cole MacPherson
% Collaborators: S. Packard, D. Wolfe, D. Zhao
% Date: 5th Mar 2021

%% Housekeeping
clc;
clear;
close all;
tic

%% declare constants
m = 0.068; % mass of the quadrotor [kg]
R = 0.06; % radial distance from CG to propeller [m]
k_m = 0.0024; % control moment coefficient [N*m/N]
I_x = 6.8e-5; % x-axis moment of inertia [kg*m^2]
I_y = 9.2e-5; % y-axis moment of inertia [kg*m^2]
```

```
I_z = 1.35e-4; % z-axis moment of inertia [kg*m^2]
nu = 1e-3; % aerodynamic force coefficient [N/(m/s)^2]
mu = 2e-6; % aerodynamic moment coefficient [N*m/(rad/s)^2]
g = 9.81; % graviational constant [m/s^2]

Z_c = -m*g; % control force
L_c = 0; % x control moment
M_c = 0; % y contorl moment
N_c = 0; % z control moment

tspan = [0 5]; % time to integrate over

%% initialize figure information
col = 'b';
fig_a = [1,2,3,4,5,6]; % figure a number vector
fig_b = [1,2,3,4,5,6] + 6; % figure b number vector
fig_c = [1,2,3,4,5,6] + 2*6; % figure c number vector
fig_d = [1,2,3,4,5,6] + 3*6; % figure d number vector
fig_e = [1,2,3,4,5,6] + 4*6; % figure e number vector
fig_f = [1,2,3,4,5,6] + 5*6; % figure f number vector

%% 3a
state_vec_0 = [0 0 0 deg2rad(5) 0 0 0 0 0 0 0 0]';
[t_a,state_vec_a] = ode45(@(t_a,state_vec_a) quadrotorODE(t_a,state_vec_a,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_a,state_vec_a,ones(length(t_a),4)*Z_c/4,fig_a,col,'3.a')

%% 3b
state_vec_0 = [0 0 0 0 deg2rad(5) 0 0 0 0 0 0 0]';
[t_b,state_vec_b] = ode45(@(t_b,state_vec_b) quadrotorODE(t_b,state_vec_b,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_b,state_vec_b,ones(length(t_b),4)*Z_c/4,fig_b,col,'3.b')

%% 3c
state_vec_0 = [0 0 0 0 0 deg2rad(5) 0 0 0 0 0 0]';
[t_c,state_vec_c] = ode45(@(t_c,state_vec_c) quadrotorODE(t_c,state_vec_c,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_c,state_vec_c,ones(length(t_c),4)*Z_c/4,fig_c,col,'3.c')

%% 3d
[p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[t_d,state_vec_d] = ode45(@(t_d,state_vec_d) quadrotorODE(t_d,state_vec_d,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_d,state_vec_d,ones(length(t_d),4)*Z_c/4,fig_d,col,'3.d')

%% 3e
[p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[t_e,state_vec_e] = ode45(@(t_e,state_vec_e) quadrotorODE(t_e,state_vec_e,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_e,state_vec_e,ones(length(t_e),4)*Z_c/4,fig_e,col,'3.e')

%% 3f
[p,q,r] = rollrate2pqr(0,0,0,0,0,0.1);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[t_f,state_vec_f] = ode45(@(t_f,state_vec_f) quadrotorODE(t_f,state_vec_f,m,I_x,I_y,I_z,nu,mu,Z_c
    ,L_c,M_c,N_c),tspan,state_vec_0);

PlotAircraftSim(t_f,state_vec_f,ones(length(t_f),4)*Z_c/4,fig_f,col,'3.f')

%% Redefine figure line color for problem 4
col = 'r';
```

```
%% 4a
t_final = 5;
state_vec_0 = [0 0 0 deg2rad(5) 0 0 0 0 0 0 0 0]';
[state_vec_a,t_a] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_a,state_vec_a,ones(length(t_a),4)*Z_c/4,fig_a,col,'3.a & 4.a')

%% 4b
state_vec_0 = [0 0 0 0 deg2rad(5) 0 0 0 0 0 0 0]';
[state_vec_b,t_b] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_b,state_vec_b,ones(length(t_b),4)*Z_c/4,fig_b,col,'3.b & 4.b')

%% 4c
state_vec_0 = [0 0 0 0 0 deg2rad(5) 0 0 0 0 0 0]';
[state_vec_c,t_c] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_c,state_vec_c,ones(length(t_c),4)*Z_c/4,fig_c,col,'3.c & 4.c')

%% 4d
[p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[state_vec_d,t_d] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_d,state_vec_d,ones(length(t_d),4)*Z_c/4,fig_d,col,'3.d & 4.d')

%% 4e
[p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[state_vec_e,t_e] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_e,state_vec_e,ones(length(t_e),4)*Z_c/4,fig_e,col,'3.e & 4.e')

%% 4f
[p,q,r] = rollrate2pqr(0,0,0,0,0,0.1);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
[state_vec_f,t_f] = linearizedEOM(state_vec_0,t_final);

PlotAircraftSim(t_f,state_vec_f,ones(length(t_f),4)*Z_c/4,fig_f,col,'3.f & 4.f')

%% 5d
[p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r Z_c -0.004*p -0.004*q -0.004*r]';
[t_d,state_vec_d] = ode45(@(t_d,state_vec_d) quadrotorODE_controlled(t_d,state_vec_d,m,I_x,I_y,
    I_z,nu,mu,Z_c,L_c,M_c,N_c),tspan,state_vec_0);

% compute forces
F_5d = ComputeMotorForces(state_vec_d(:,13),state_vec_d(:,14),state_vec_d(:,15),state_vec_d(:,16)
    ,R,k_m);

% plot force
figure
subplot(3,1,1)
plot(t_d,F_5d(1,:),'linewidth',2); hold on;
plot(t_d,F_5d(2,:),'linewidth',2); hold on;
plot(t_d,F_5d(3,:),'linewidth',2); hold on;
plot(t_d,F_5d(4,:),'linewidth',2); hold on;
title(['p = ' num2str(p)]);
legend('motor 1','motor 2','motor 3','motor 4');
hold on;

%% 5e
[p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r Z_c -0.004*p -0.004*q -0.004*r]';
[t_e,state_vec_e] = ode45(@(t_e,state_vec_e) quadrotorODE_controlled(t_e,state_vec_e,m,I_x,I_y,
    I_z,nu,mu,Z_c,L_c,M_c,N_c),tspan,state_vec_0);
```

```
% compute forces
F_5e = ComputeMotorForces(state_vec_e(:,13),state_vec_e(:,14),state_vec_e(:,15),state_vec_e(:,16)
    ,R,k_m);

% plot force
subplot(3,1,2)
plot(t_e,F_5e(1,:),'linewidth',2); hold on;
plot(t_e,F_5e(2,:),'linewidth',2); hold on;
plot(t_e,F_5e(3,:),'linewidth',2); hold on;
plot(t_e,F_5e(4,:),'linewidth',2); hold on;
title(['q = ' num2str(q)]);
legend('motor 1','motor 2','motor 3','motor 4');
hold on;

%% 5f
[p,q,r] = rollrate2pqr(0,0,0,0,0,0.1);
state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r Z_c -0.004*p -0.004*q -0.004*r]';
[t_f,state_vec_f] = ode45(@(t_f,state_vec_f) quadrotorODE_controlled(t_f,state_vec_f,m,I_x,I_y,
    I_z,nu,mu,Z_c,L_c,M_c,N_c),tspan,state_vec_0);

% compute forces
F_5f = ComputeMotorForces(state_vec_f(:,13),state_vec_f(:,14),state_vec_f(:,15),state_vec_f(:,16)
    ,R,k_m);

% plot force
subplot(3,1,3)
plot(t_f,F_5f(1,:),'linewidth',2); hold on;
plot(t_f,F_5f(2,:),'linewidth',2); hold on;
plot(t_f,F_5f(3,:),'linewidth',2); hold on;
plot(t_f,F_5f(4,:),'linewidth',2); hold on;
title(['r = ' num2str(r)]);
legend('motor 1','motor 2','motor 3','motor 4');
hold on;

%% Plot Housekeeping
% make legends for each plot
for N = 1:6
    figure(6*N-5)
    legend('Non-Linear Path','Linearized Path','location','best');
    legend('boxoff');

    figure(6*N-4)
    legend('Non-Linear Path','Linearized Path','location','best');
    legend('boxoff');

    figure(6*N-3)
    legend('Non-Linear Path','Linearized Path','location','best');
    legend('boxoff');

    figure(6*N-2)
    legend('Non-Linear Path','Linearized Path','location','best');
    legend('boxoff');

    figure(6*N-1)
    legend('Non-Linear Path','Linearized Path','location','best');
    legend('boxoff');

    figure(6*N)
    legend('Non-Linear Path','Start','End','Linearized Path');
end

%% End Housekeeping
toc
```

Plotting (Problem 1)

```
function PlotAircraftSim(t,state,control,fig,col,num)

```

```matlab
%% Plot position vs time
figure(fig(1))
subplot(3,1,1)
plot(t,state(:,1),col,'linewidth',2); hold on;
grid on
ylabel('x');
title([num ' Position vs. time']);
subplot(3,1,2)
plot(t,state(:,2),col,'linewidth',2); hold on;
grid on
ylabel('y');
subplot(3,1,3)
plot(t,state(:,3),col,'linewidth',2); hold on;
grid on
ylabel('z');
xlabel('time [s]');

%% Plot Euler andgles vs time
figure(fig(2))
subplot(3,1,1)
plot(t,state(:,4),col,'linewidth',2); hold on;
grid on
ylabel('\phi');
title([num ' Euler angles vs. time']);
subplot(3,1,2)
plot(t,state(:,5),col,'linewidth',2); hold on;
grid on
ylabel('\theta');
subplot(3,1,3)
plot(t,state(:,6),col,'linewidth',2); hold on;
grid on
ylabel('\psi');
xlabel('time [s]');

%% Plot velocity vs time
figure(fig(3))
subplot(3,1,1)
plot(t,state(:,7),col,'linewidth',2); hold on;
grid on
ylabel('u');
title([num ' Inertial Velocity vs. time']);
subplot(3,1,2)
plot(t,state(:,8),col,'linewidth',2); hold on;
grid on
ylabel('v');
subplot(3,1,3)
plot(t,state(:,9),col,'linewidth',2); hold on;
grid on
ylabel('w');
xlabel('time [s]');

%% Plot angular velocity vs time
figure(fig(4))
subplot(3,1,1)
plot(t,state(:,10),col,'linewidth',2); hold on;
grid on
ylabel('p');
title([num ' Angular Velocity vs. time']);
subplot(3,1,2)
plot(t,state(:,11),col,'linewidth',2); hold on;
grid on
ylabel('q');
subplot(3,1,3)
plot(t,state(:,12),col,'linewidth',2); hold on;
grid on
ylabel('r');
xlabel('time [s]');

```

```
    %% Plot control forces and moments vs time
    figure(fig(5))
    subplot(2,2,1)
    plot(t,control(:,1),col,'linewidth',2); hold on;
    grid on
    title([num ' Control Force vs. time']);
    ylabel('Control Force');
    xlim([t(1) t(end)]);
    subplot(2,2,2)
    plot(t,control(:,2),col,'linewidth',2); hold on;
    grid on
    title([num ' x Control Moment vs. time']);
    ylabel('x Control Moment');
    xlim([t(1) t(end)]);
    subplot(2,2,3)
    plot(t,control(:,3),col,'linewidth',2); hold on;
    grid on
    title([num ' y Control Moment vs. time']);
    ylabel('y Control Moment');
    xlabel('time [s]');
    xlim([t(1) t(end)]);
    subplot(2,2,4)
    plot(t,control(:,4),col,'linewidth',2); hold on;
    grid on
    title([num ' z Control Moment vs. time']);
    xlabel('time [s]');
    ylabel('z Control Moment');
    xlim([t(1) t(end)]);

    %% Plot flight path of the simulated quadrotor
    figure(fig(6))
    plot3(state(:,1),state(:,2),state(:,3),col,'linewidth',2); hold on;
    plot3(state(1,1),state(1,2),state(1,3),'g.','markersize',20); hold on;
    plot3(state(end,1),state(end,2),state(end,3),'r.','markersize',20); hold on;
    set(gca, 'YDir','reverse')
    set(gca, 'ZDir','reverse')
    grid on
    title([num ' Flight Path']);
    xlabel('x');
    ylabel('y');
    zlabel('z');

end
```

Coord frame conversions

```
function [p,q,r] = rollrate2pqr(phi,theta,psi,phi_dot,theta_dot,psi_dot)

    %% Sovle for pqr vector
    pqr = [phi_dot,theta_dot,psi_dot]...
        *inv([1, sin(phi)*tan(theta), cos(phi)*tan(theta);...
        0, cos(phi), sin(phi);...
        0, sin(phi)*sec(theta), cos(phi)*sec(theta)]);

    %% Define p, q, and r based on previous calculations
    p = pqr(1);
    q = pqr(2);
    r = pqr(3);

end
```

Non feedback control ODE for problem 3

```
function xdot = quadrotorODE(t,state_vec,m,I_x,I_y,I_z,n,mu,Z_c,L_c,M_c,N_c)

    phi = state_vec(4); % roll angle
    theta = state_vec(5); % pitch angle
    psi = state_vec(6); % yaw angle
```

```
    u = state_vec(7); % x velocity
    v = state_vec(8); % y velocity
    w = state_vec(9); % z velocity
    p = state_vec(10); % roll velocity
    q = state_vec(11); % pitch velocity
    r = state_vec(12); % yaw velocity

    g = 9.81; %Gravity m/s^2
    M = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
    V = norm([u;v;w]); % velocity
    F = -n*V*[u;v;w]; % aerodynamic force

    MomCntl = [L_c; M_c; N_c]; % control moment

    % transformation matrix
    R = [cos(theta)*cos(psi) sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi) cos(phi)*sin(theta)*
    cos(psi)+sin(phi)*sin(psi);...
        cos(theta)*sin(psi) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) cos(phi)*sin(theta)*
    sin(psi)-sin(phi)*cos(psi);...
        -sin(theta) sin(phi)*cos(theta) cos(phi)*cos(theta)];

    % equations of motion
    v_inertial = R * [u;v;w];
    euler_dot = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);...
        0 cos(phi) -sin(phi);...
        0 sin(phi)*(1/cos(theta)) cos(phi)*(1/cos(theta))]...
        * [p;q;r];
    a_inertial = [r*v-q*w; p*w-r*u; q*u-p*v]...
        + g*[-sin(theta);cos(theta)*sin(phi);cos(theta)*cos(phi)]...
        + (1/m)*F + (1/m)*[0;0;Z_c];
    a_angular = [((I_y-I_z)/I_x)*q*r; ((I_z-I_x)/I_y)*q*r; ((I_x-I_y)/I_z)*q*r]...
        + [(1/I_x)*M(1); (1/I_y)*M(2); (1/I_z)*M(3)]...
        + [(1/I_x)*MomCntl(1); (1/I_y)*MomCntl(2); (1/I_z)*MomCntl(3)];

    xdot = [v_inertial; euler_dot; a_inertial; a_angular]; % vecotr output for the integrals

end
```

Motor Forces (Problem 2)

```
function motor_forces = ComputeMotorForces(Z_c,L_c,M_c,N_c,r,k_m)

    %% Moment to force matrix
    m2f = [-1, -1, -1, -1;...
        -r/sqrt(2), -r/sqrt(2), r/sqrt(2), r/sqrt(2);...
        r/sqrt(2), -r/sqrt(2), -r/sqrt(2), r/sqrt(2);...
        k_m, -k_m, k_m, -k_m];

    %% Vecotr of control moments and force
    f_vec = [Z_c, L_c, M_c, N_c];

    %% Solve for motor forces
    motor_forces = m2f\f_vec';

end
```

Linearization for problem 4

```
function [x,t] = linearizedEOM(x0,t_f)

    g = 9.81; %acceleration due to gravity

    %% Defien state space matrix
    A = zeros(12,12);
    A(1,7) = 1;
    A(2,8) = 1;
    A(3,9) = 1;
    A(4,10) = 1;
```

```
    A(5,11) = 1;
    A(6,12) = 1;
    A(7,5) = -g;
    A(8,4) = g;

    %% Define state space system
    sys = ss(A,zeros(12,1),eye(12),0);

    %% Simulate state space system
    [x,t] = initial(sys,x0,t_f);

end
```

Feedback control ODE for problem 5

```
function xdot = quadrotorODE_controlled(t,state_vec,m,I_x,I_y,I_z,n,mu,Z_c,L_c,M_c,N_c)

    %% Define variables
    phi = state_vec(4); % roll angle
    theta = state_vec(5); % pitch angle
    psi = state_vec(6); % yaw angle
    u = state_vec(7); % x velocity
    v = state_vec(8); % y velocity
    w = state_vec(9); % z velocity
    p = state_vec(10); % roll velocity
    q = state_vec(11); % pitch velocity
    r = state_vec(12); % yaw velocity
    g = 9.81; %Gravity m/s^2
    M = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
    V = norm([u;v;w]); % velocity
    F = -n*V*[u;v;w]; % aerodynamic force
    MCntl = -0.004*[p;q;r]; % control moment

    %% Transformation matrix
    R = [cos(theta)*cos(psi) sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi) cos(phi)*sin(theta)*
    cos(psi)+sin(phi)*sin(psi);...
        cos(theta)*sin(psi) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) cos(phi)*sin(theta)*
    sin(psi)-sin(phi)*cos(psi);...
        -sin(theta) sin(phi)*cos(theta) cos(phi)*cos(theta)];

    %% Equations of motion
    v_inertial = R*[u;v;w];
    euler_dot = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);...
        0 cos(phi) -sin(phi);...
        0 sin(phi)*(1/cos(theta)) cos(phi)*(1/cos(theta))]...
        * [p;q;r];
    a_inertial = [r*v-q*w; p*w-r*u; q*u-p*v]...
        + g*[-sin(theta);cos(theta)*sin(phi);cos(theta)*cos(phi)]...
        + (1/m)*F + (1/m)*[0;0;Z_c];
    a_angular = [((I_y-I_z)/I_x)*q*r; ((I_z-I_x)/I_y)*q*r; ((I_x-I_y)/I_z)*q*r]...
        + [(1/I_x)*M(1); (1/I_y)*M(2); (1/I_z)*M(3)]...
        + [(1/I_x)*MCntl(1); (1/I_y)*MCntl(2); (1/I_z)*MCntl(3)];
    deltaF = [0; -0.004*a_angular];

    xdot = [v_inertial; euler_dot; a_inertial; a_angular; deltaF]; % vecotr output for the
    integrals

end
```

VI. Team Participation Table

Name	Plan	Model	Experiment	Results	Report	Code	Ack
Cole MacPherson	1	2	X	2	1	2	CM
Donny Wolfe	1	1	X	1	2	1	DW
Samuel Packard	1	1	X	1	2	1	SP
Dawei Zhao	2	1	X	1	1	1	DZ