

ASEN 3128 - Lab Deliverable 3

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

Contents

I Problem 3 Question 1	2
II Problem 3 Question 2	2
III Problem 5 Question 1	2
IV Figures	3
IV.A Problem 3.a and 4.a	3
IV.B Problem 3.b and 4.b	8
IV.C Problem 3.c and 4.c	13
IV.D Problem 3.d and 4.d	18
IV.E Problem 3.e and 4.e	23
IV.F Problem 3.f and 4.f	28
IV.G Problem 5	33
V Appendix	33
V.A MATLAB Code	33
VI Team Participation Table	41

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

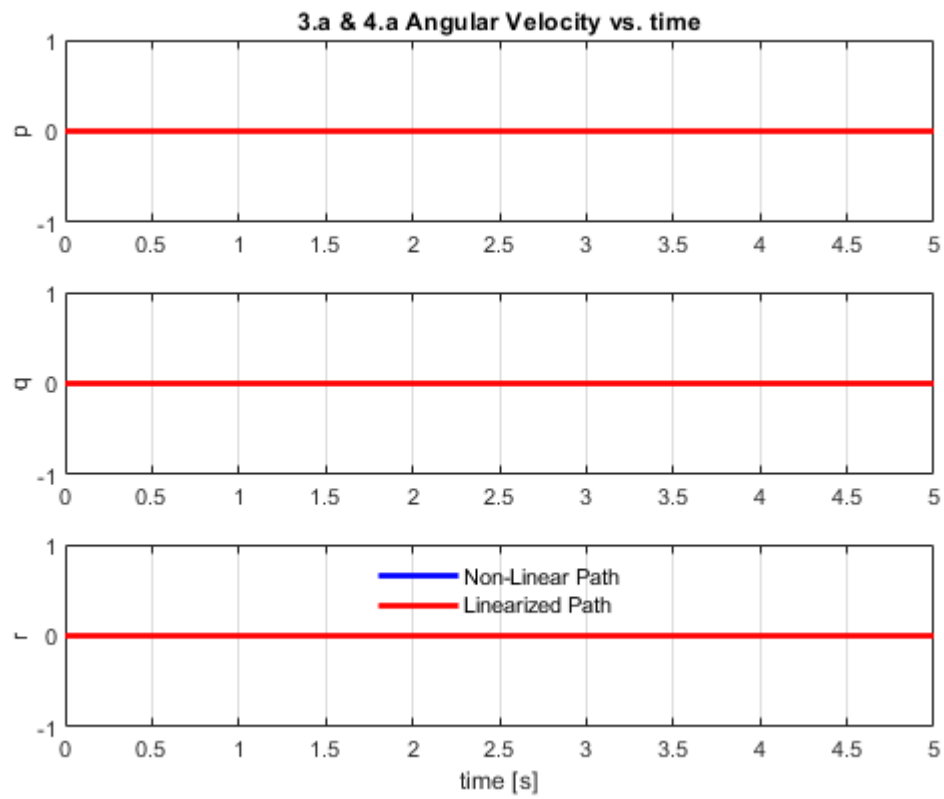


Fig. 1 Angular Velocity

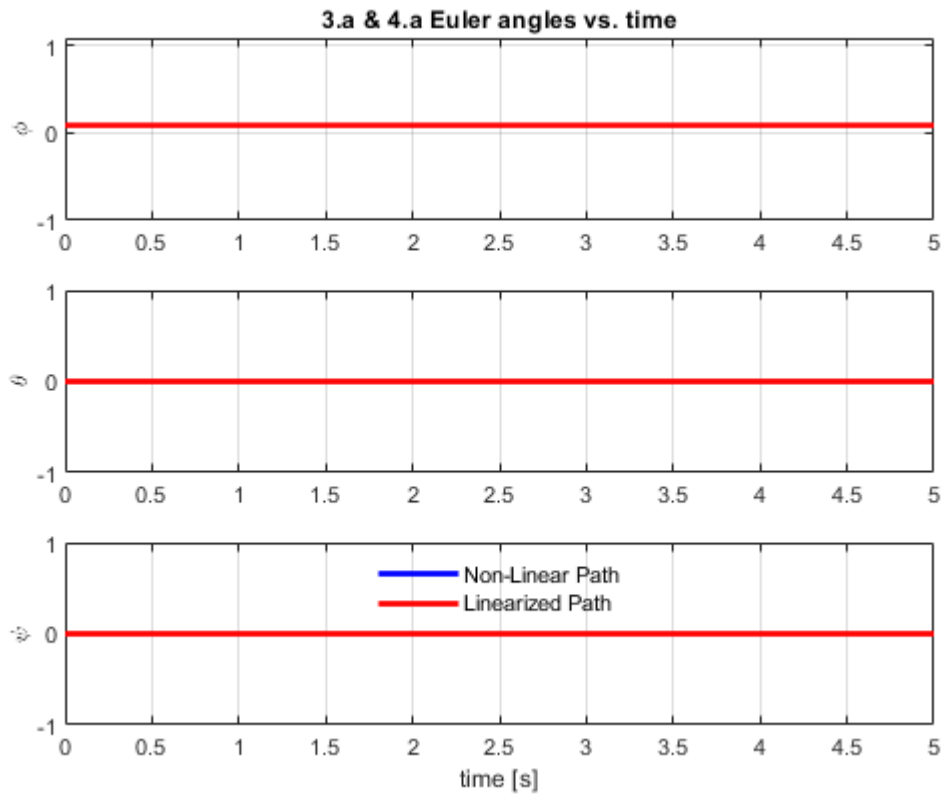


Fig. 2 Euler Angles

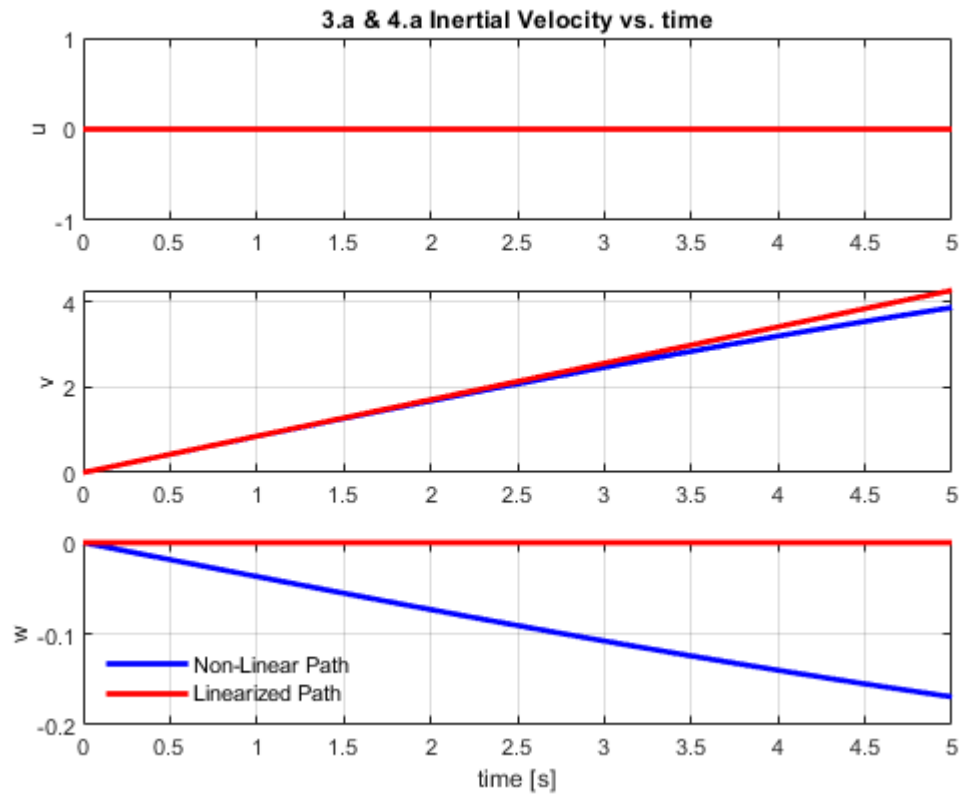


Fig. 3 Inertial Velocity

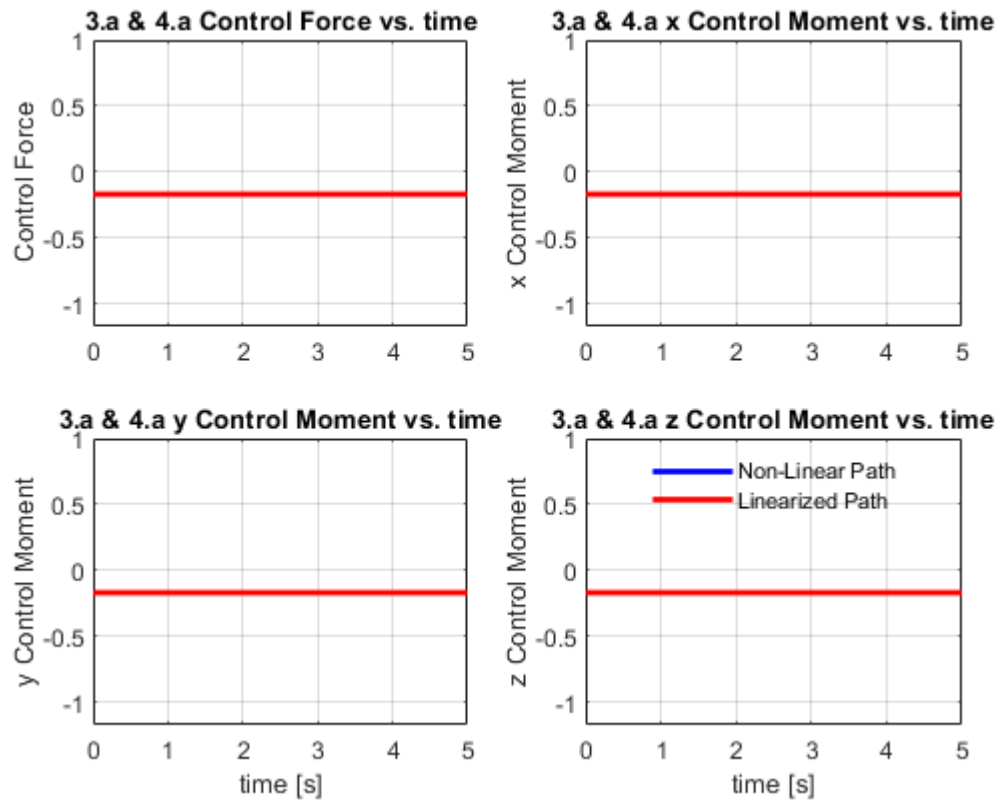


Fig. 4 Control Forces

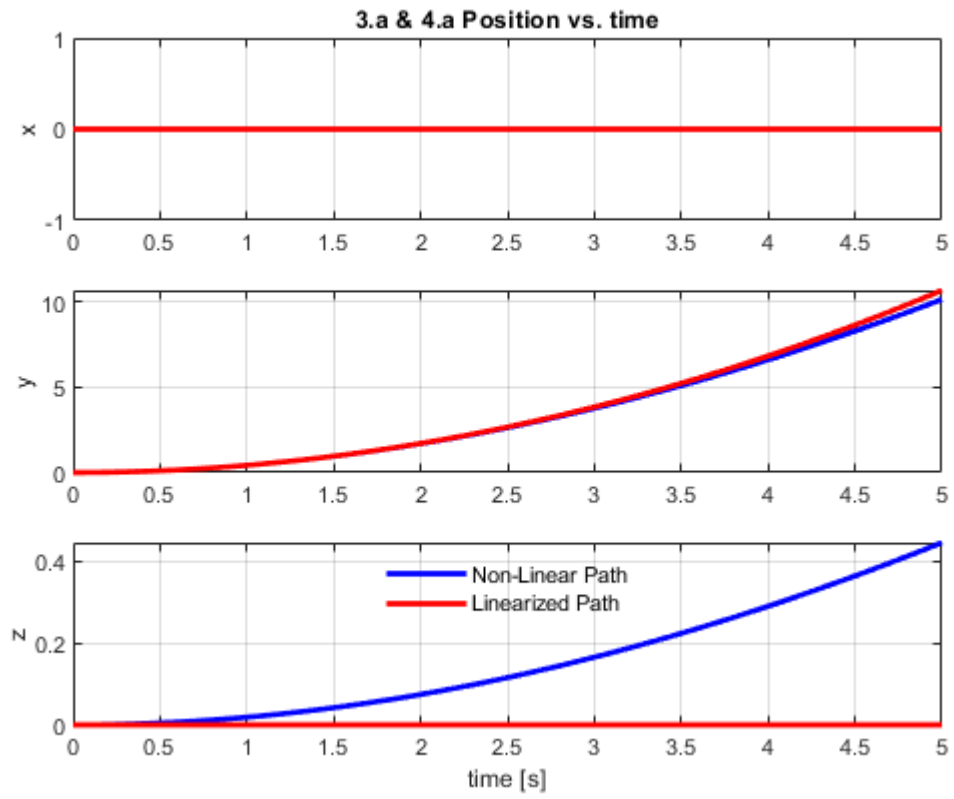


Fig. 5 Flight Path

B. Problem 3.b and 4.b

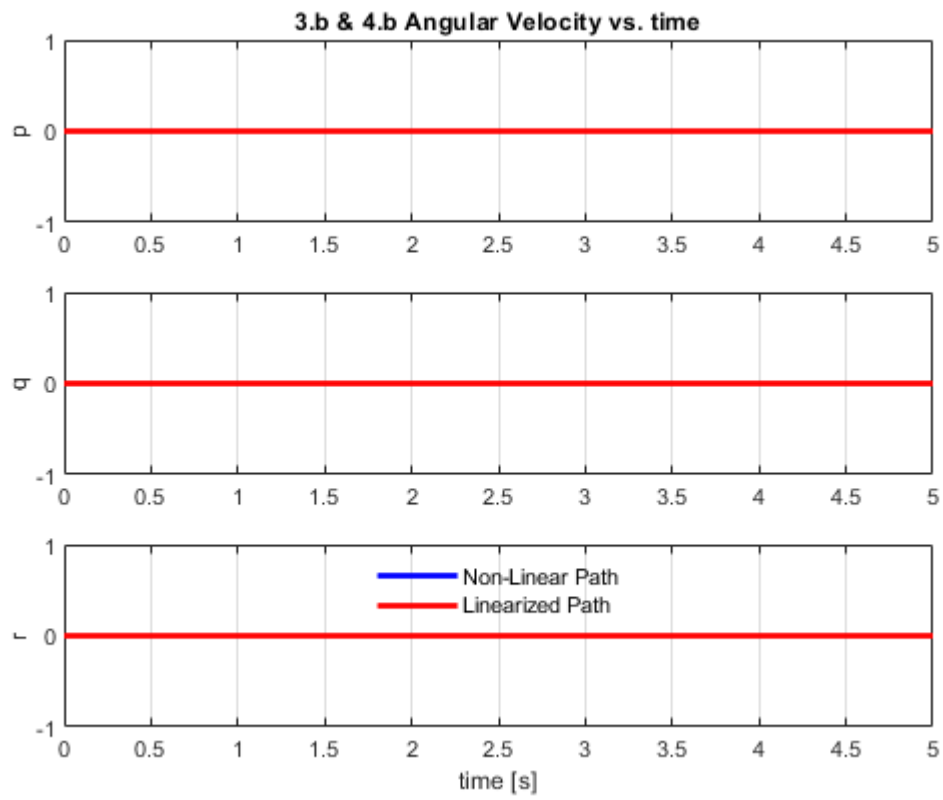


Fig. 6 Angular Velocity

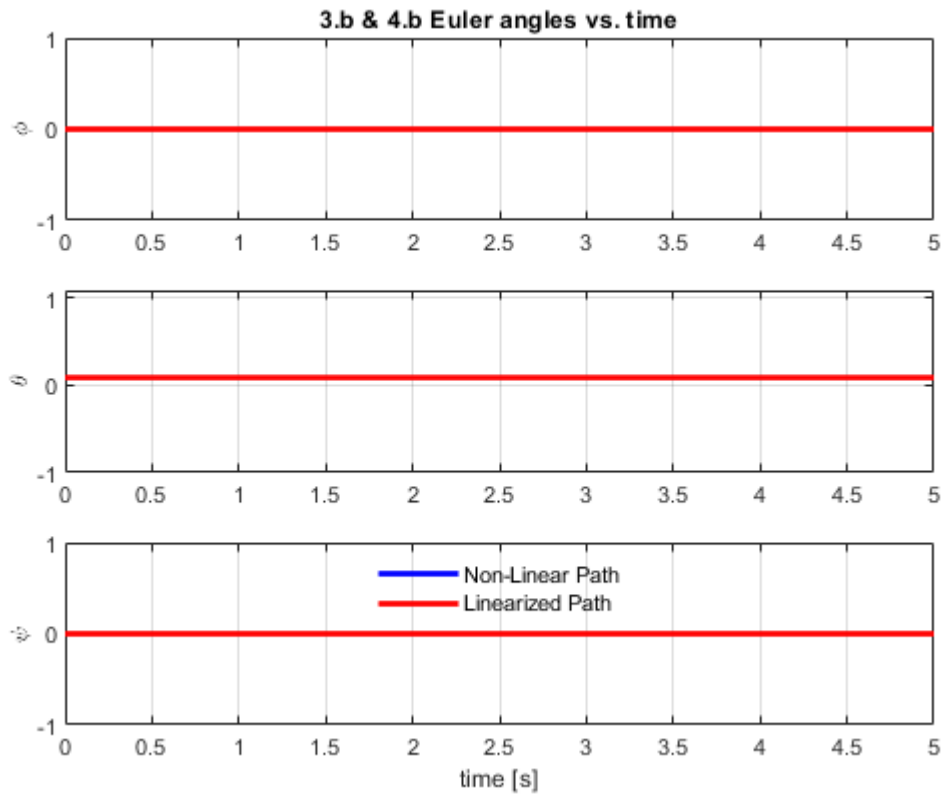


Fig. 7 Euler Angles

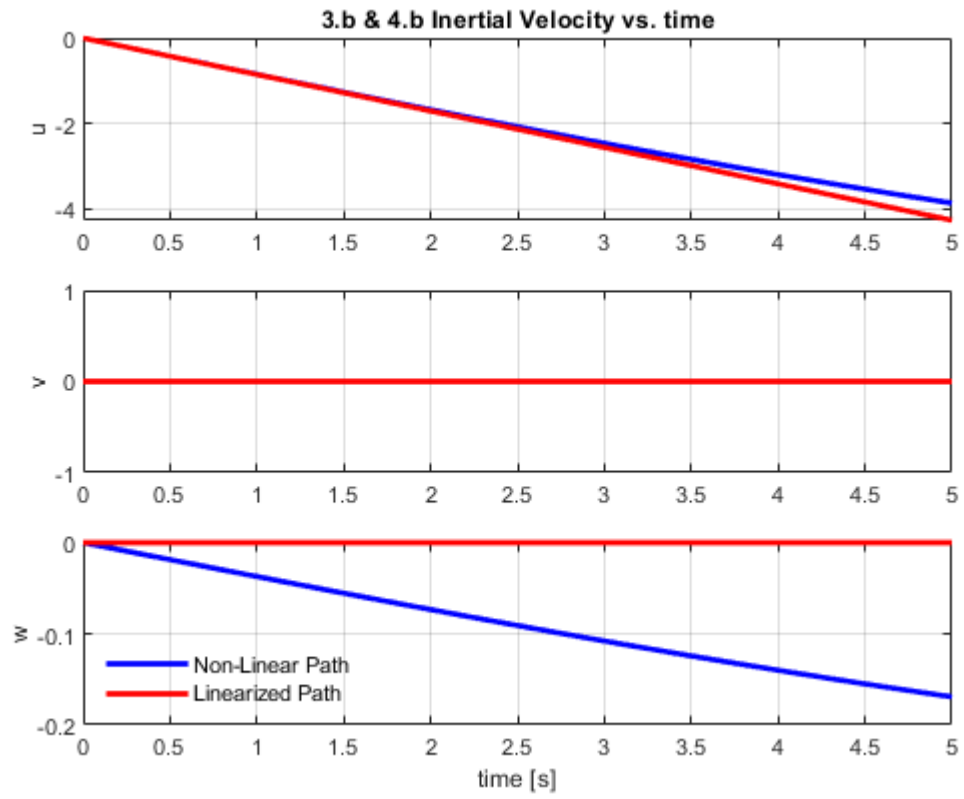


Fig. 8 Inertial Velocity

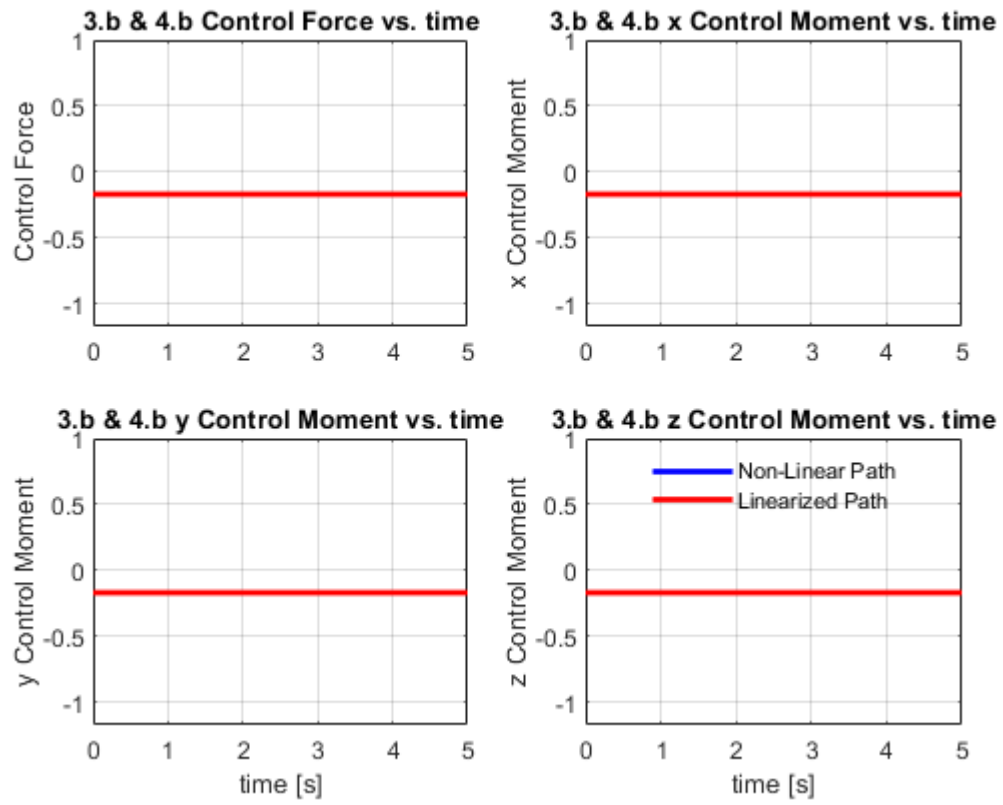


Fig. 9 Control Forces

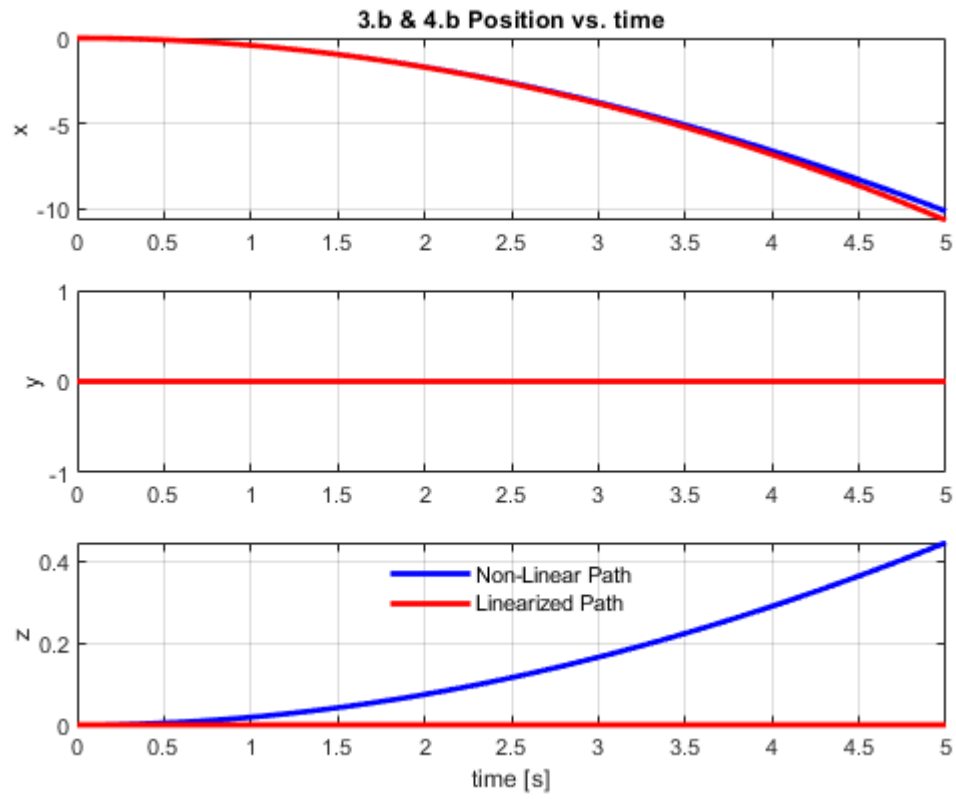


Fig. 10 Flight Path

C. Problem 3.c and 4.c

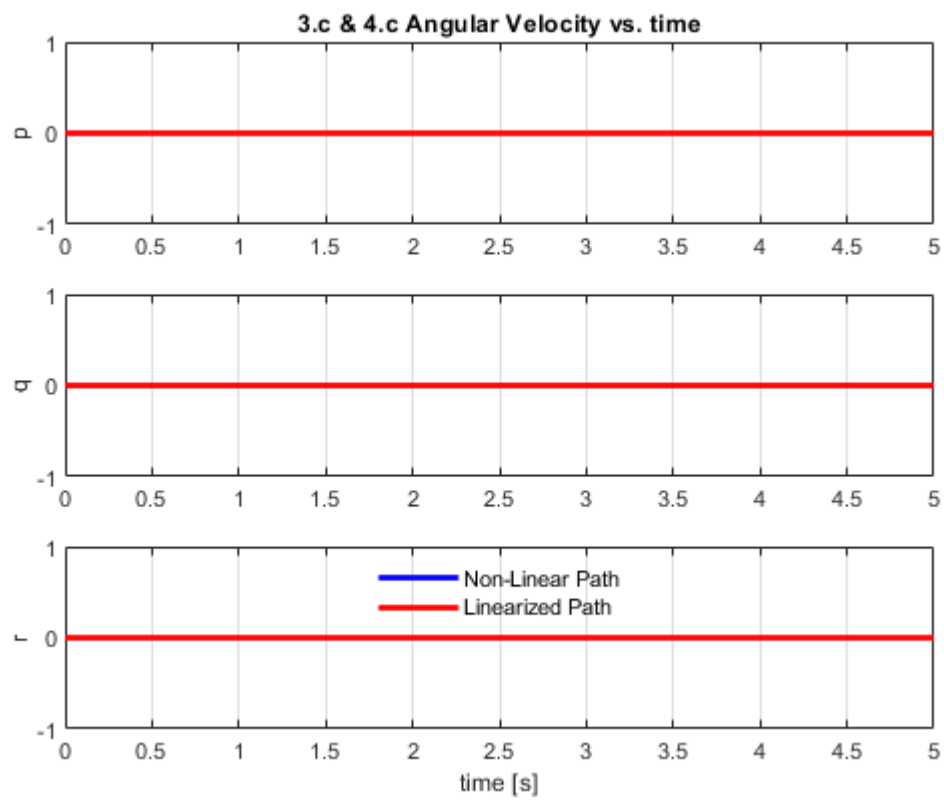


Fig. 11 Angular Velocity

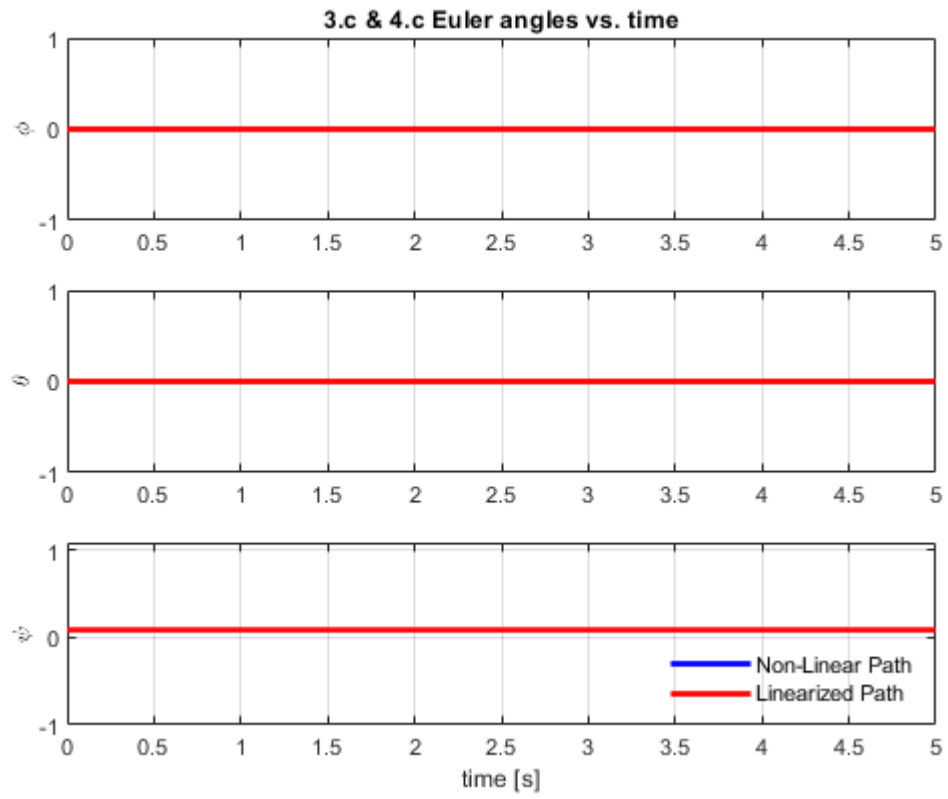


Fig. 12 Euler Angles

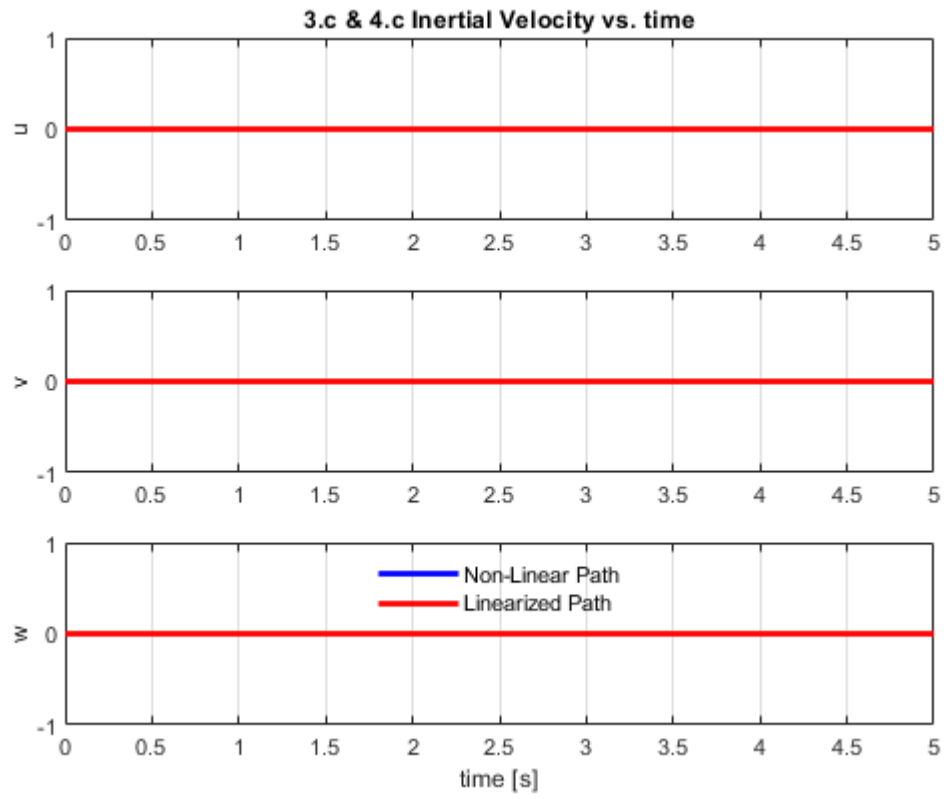


Fig. 13 Inertial Velocity

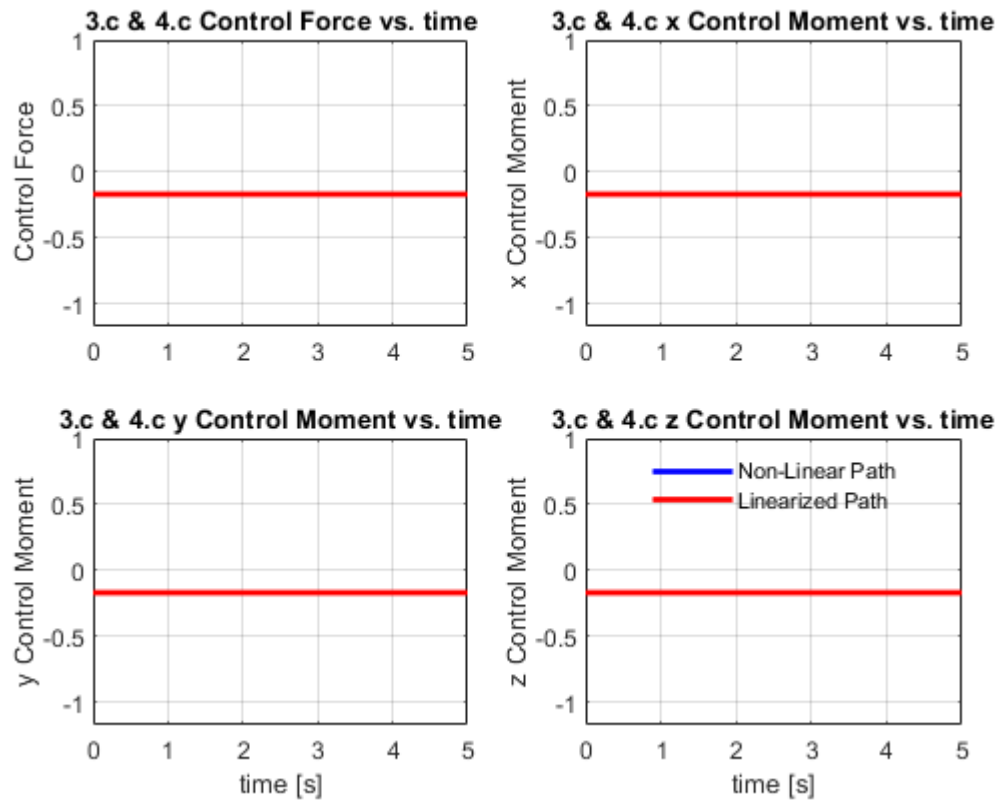


Fig. 14 Control Forces

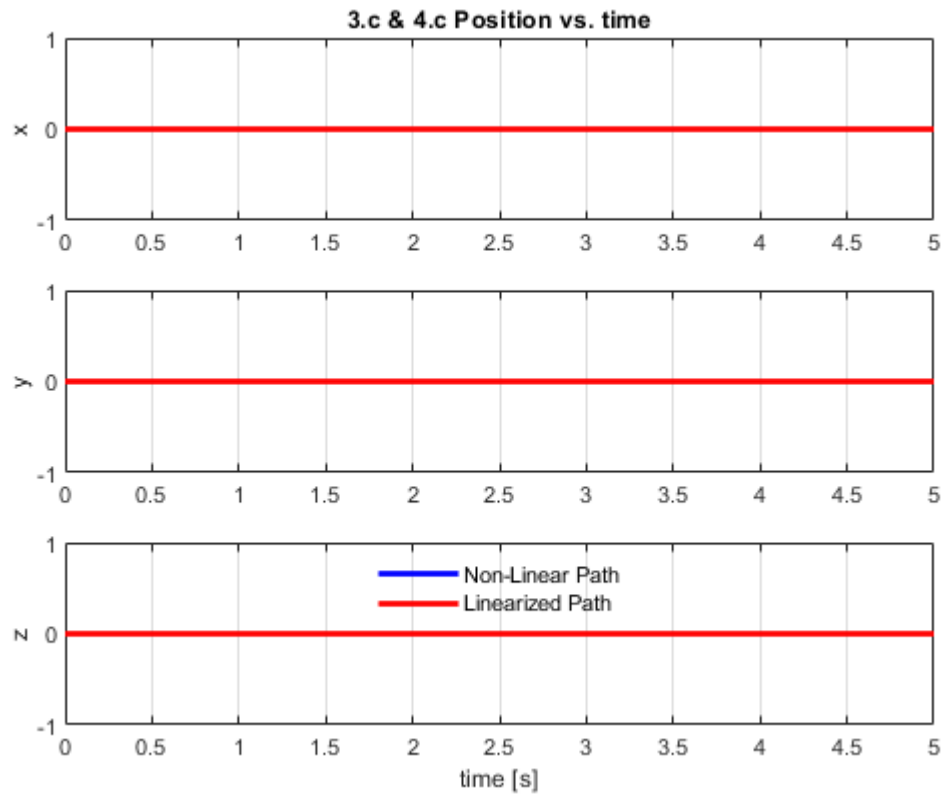


Fig. 15 Flight Path

D. Problem 3.d and 4.d

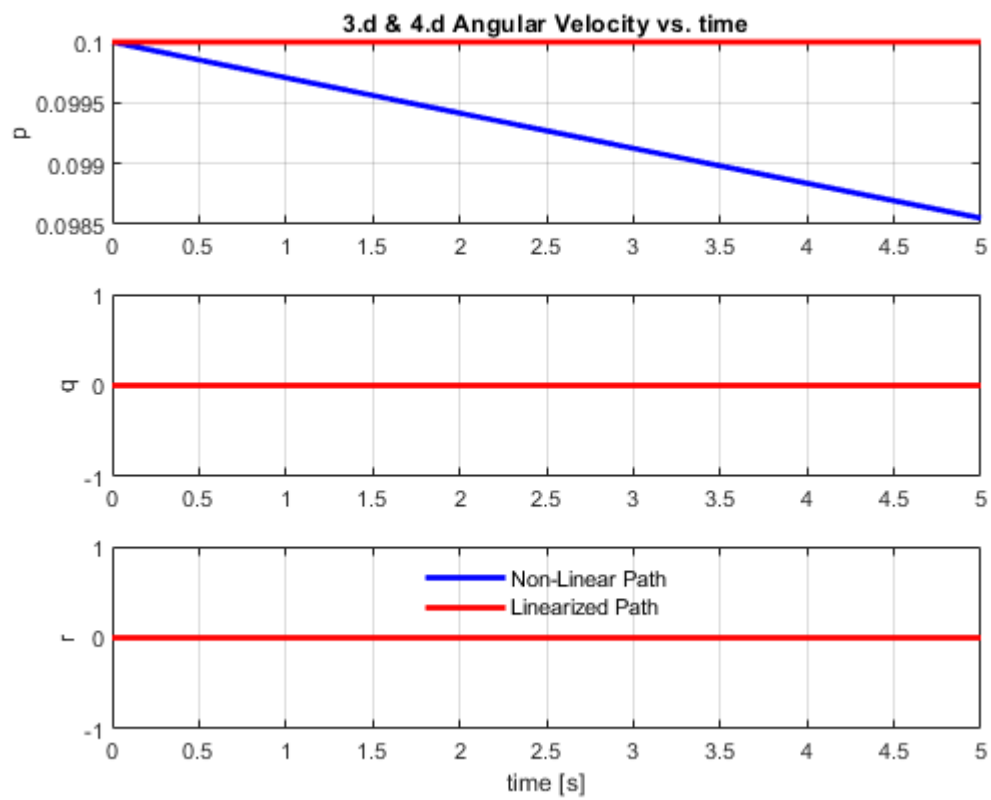


Fig. 16 Angular Velocity

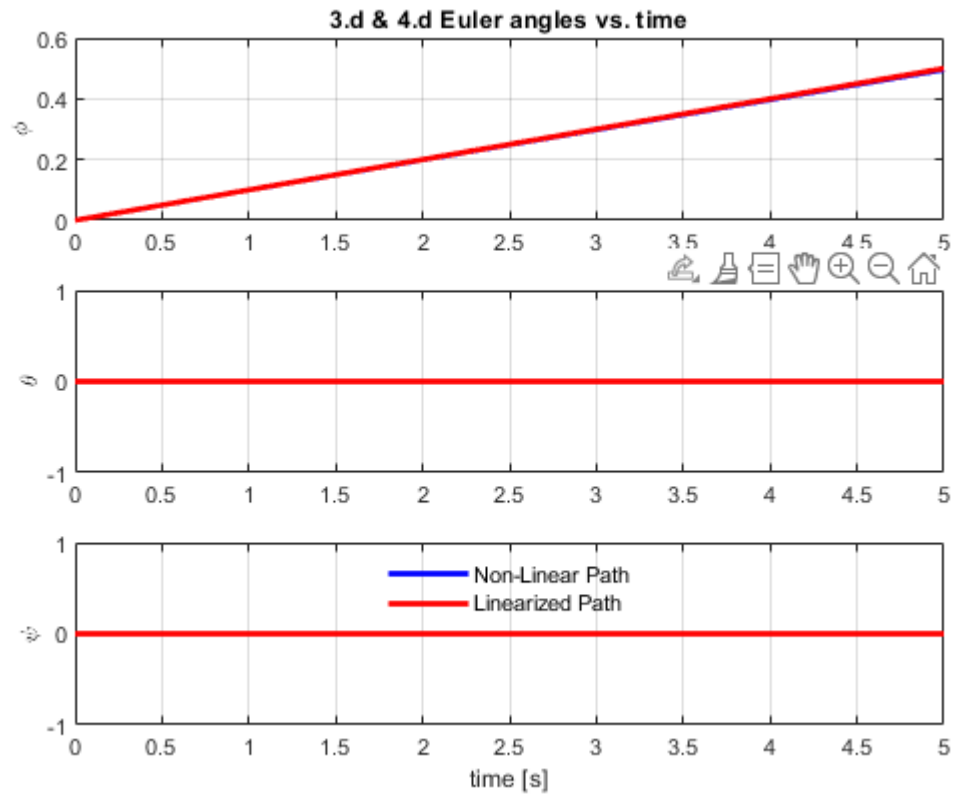


Fig. 17 Euler Angles

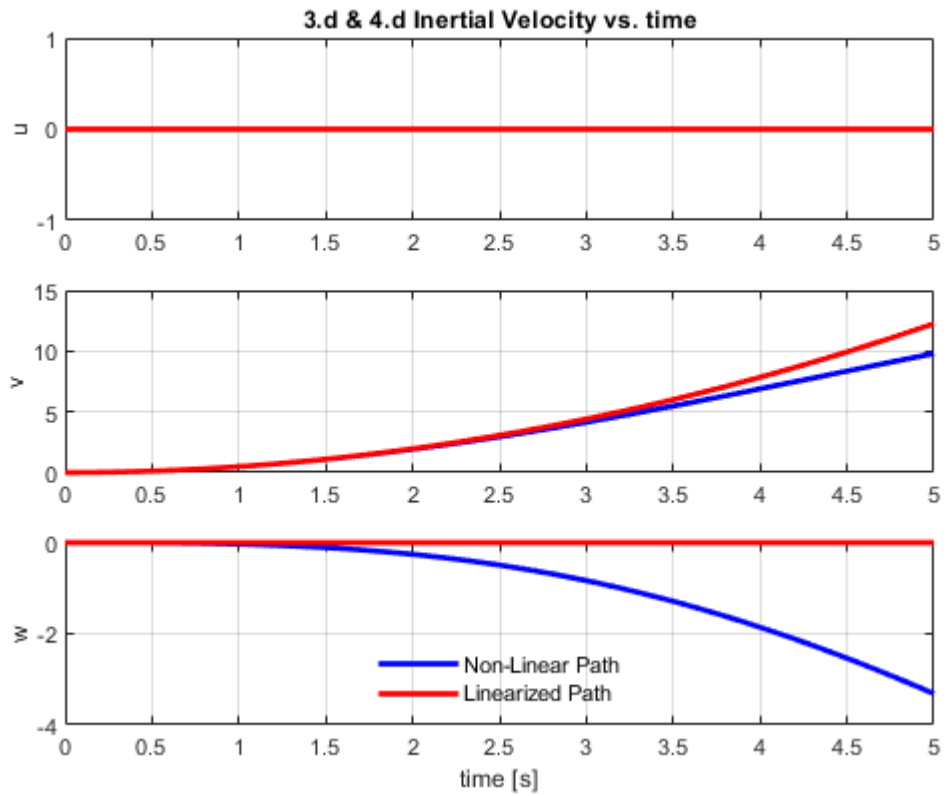


Fig. 18 Inertial Velocity

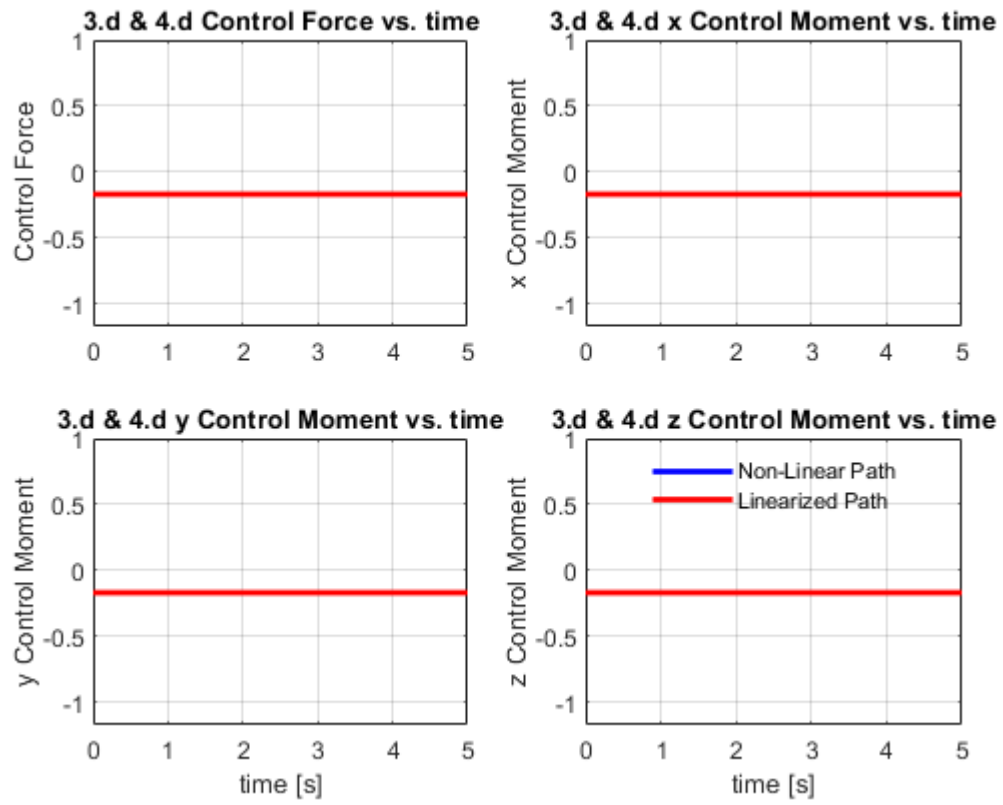


Fig. 19 Control Forces

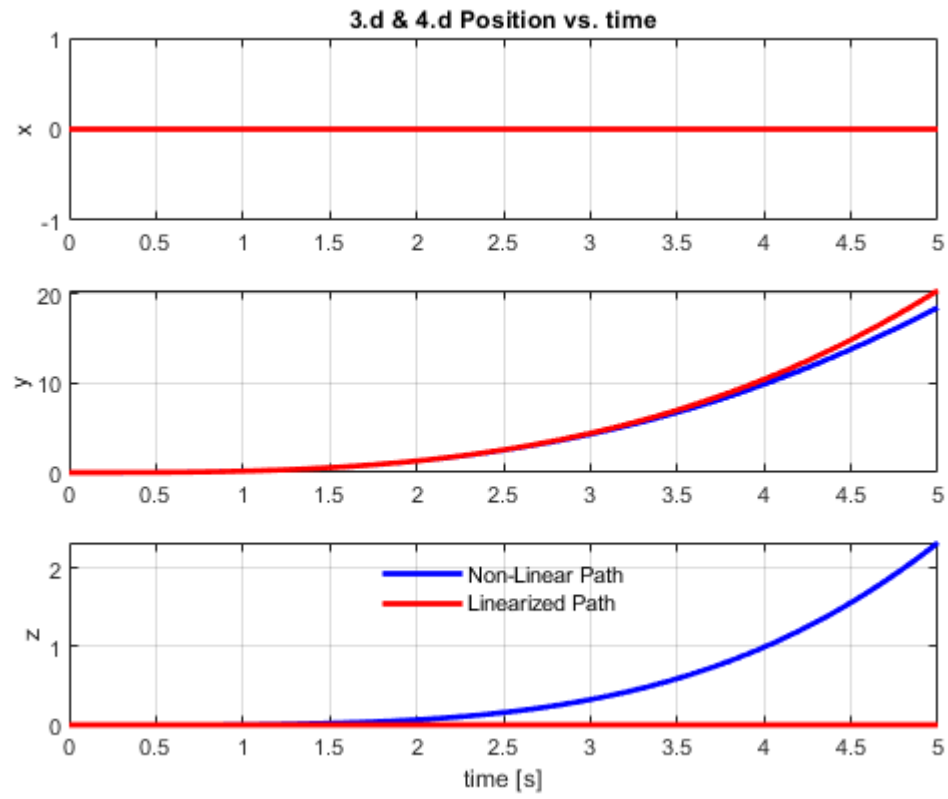


Fig. 20 Flight Path

E. Problem 3.e and 4.e

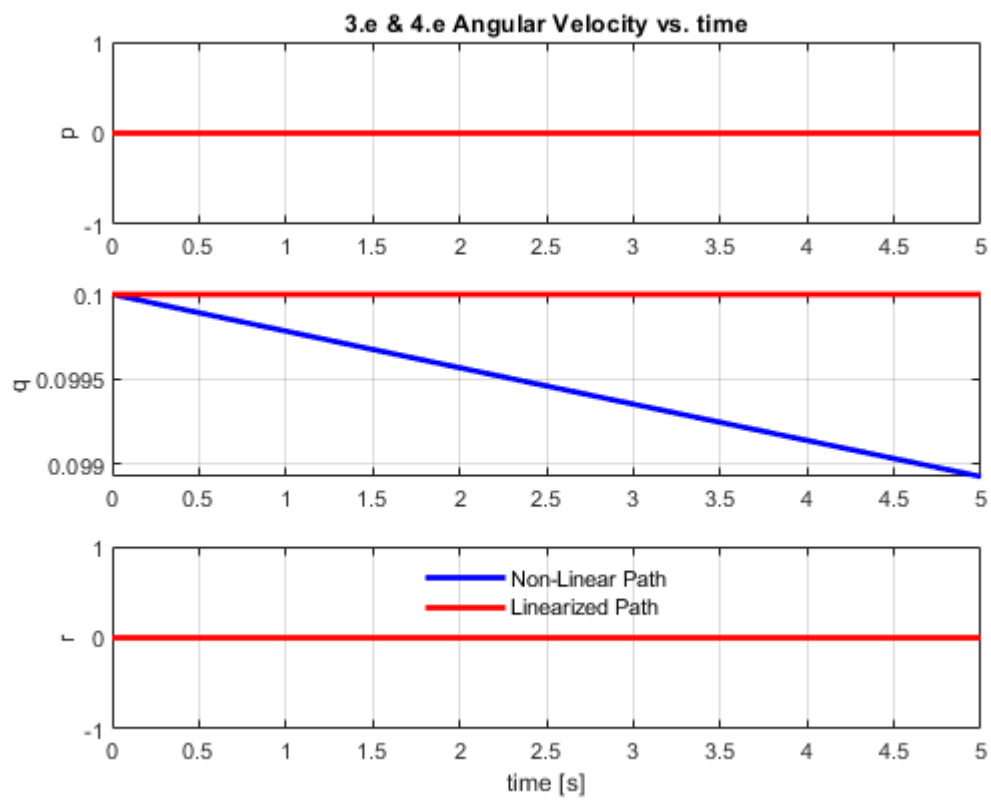


Fig. 21 Angular Velocity

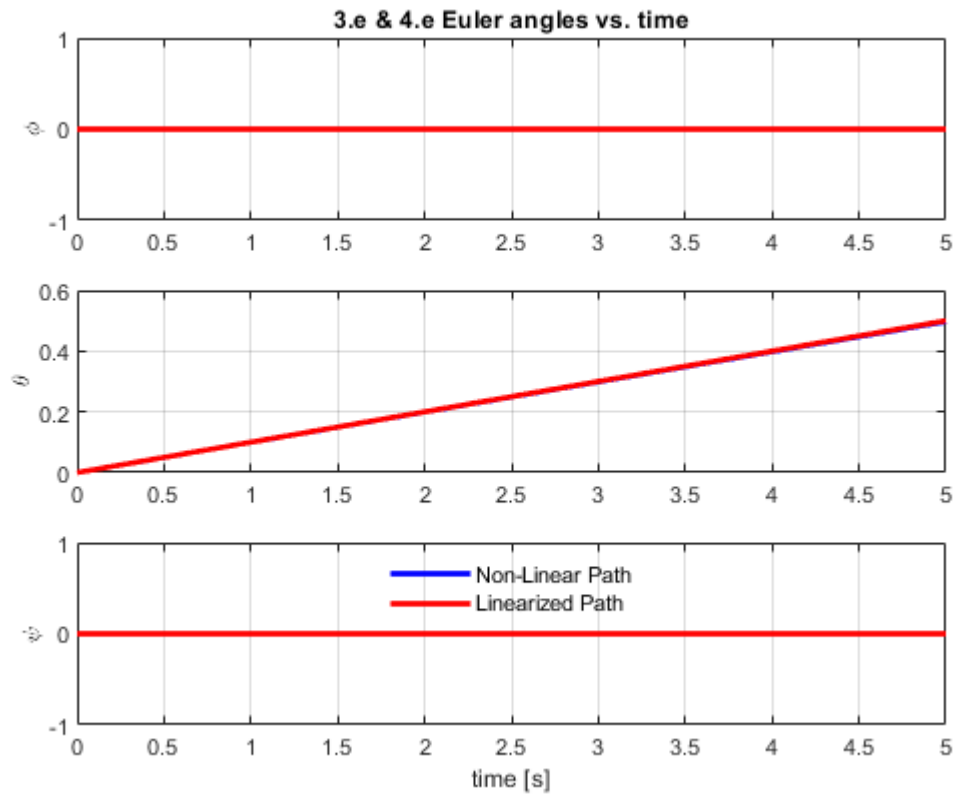


Fig. 22 Euler Angles

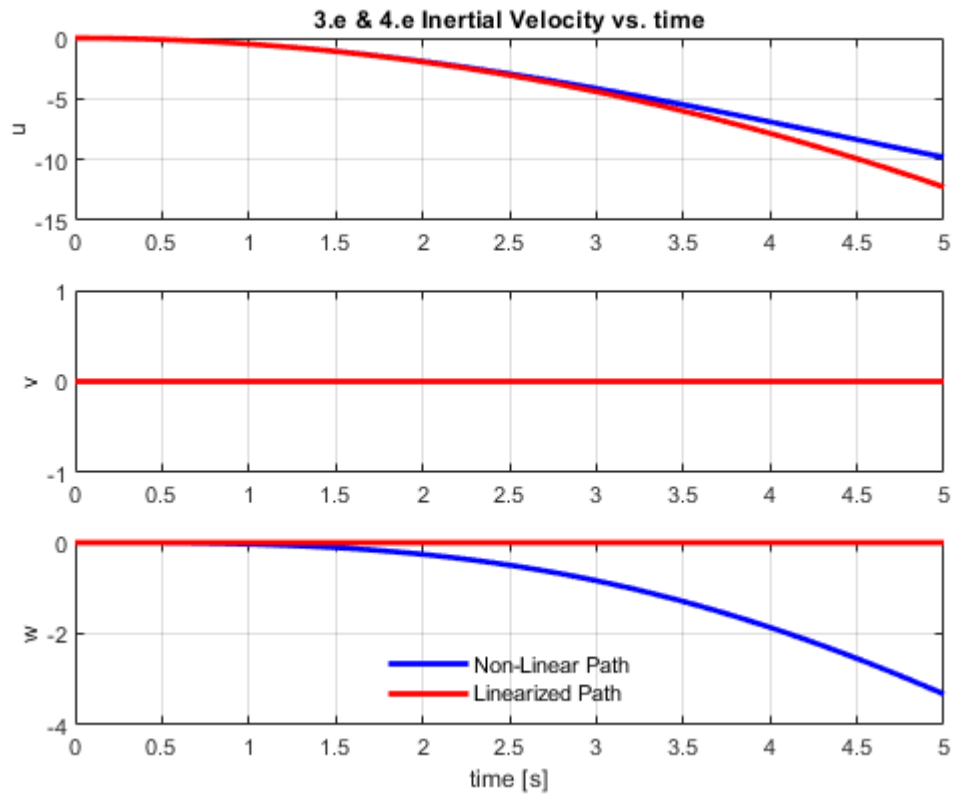


Fig. 23 Inertial Velocity

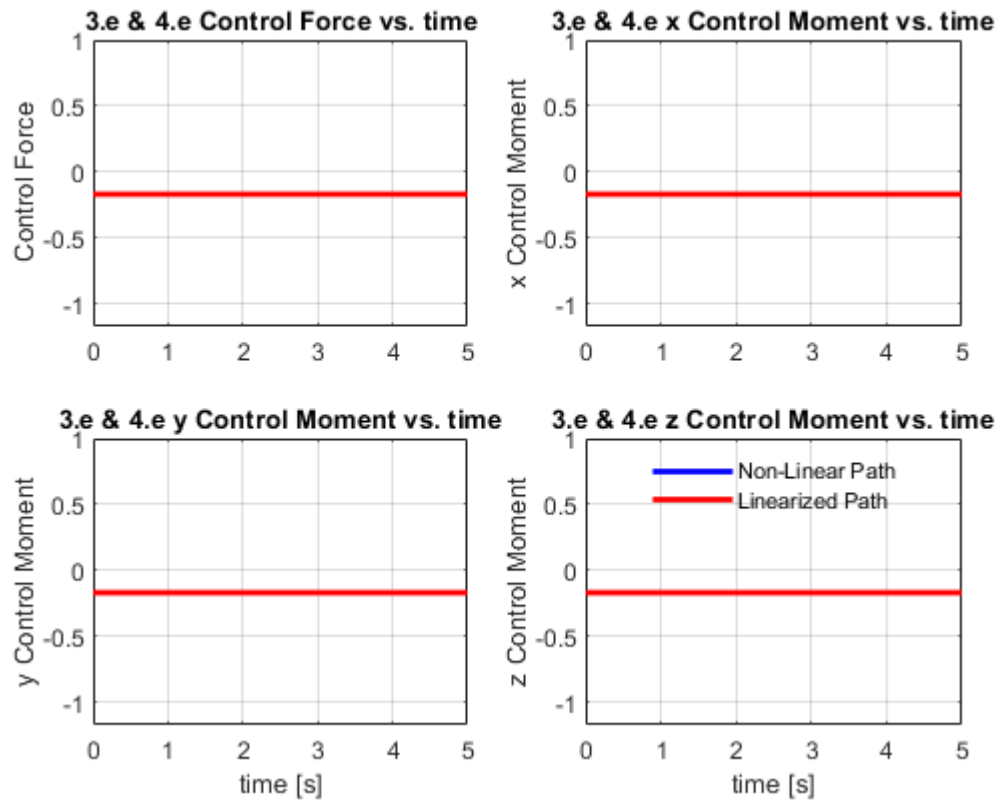


Fig. 24 Control Forces

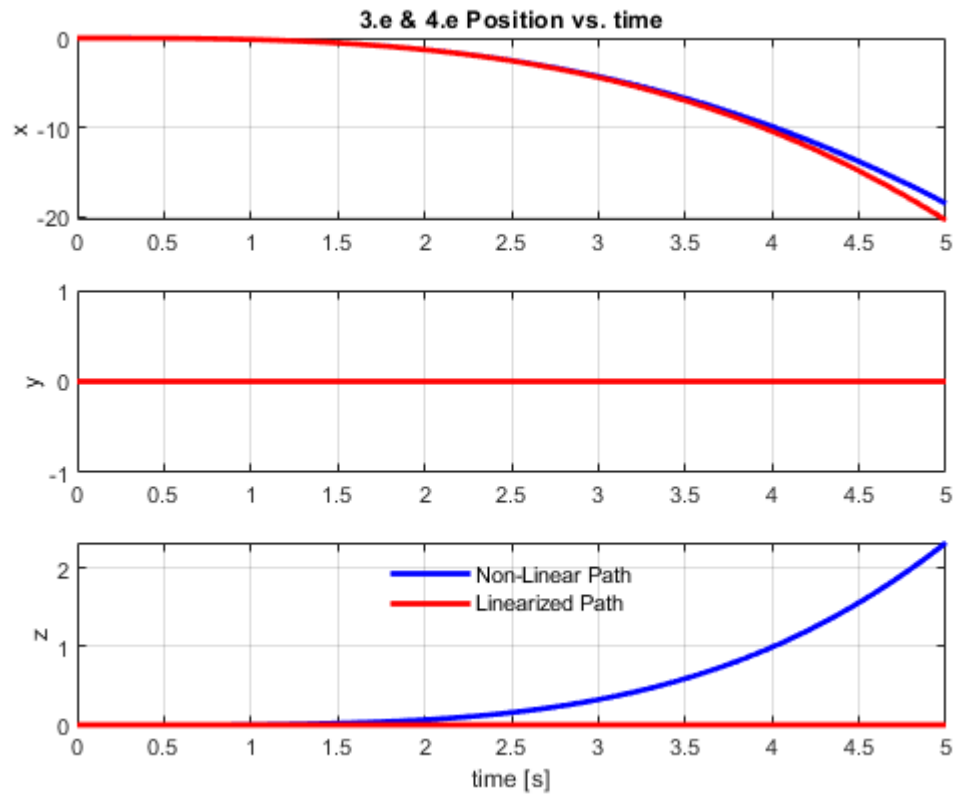


Fig. 25 Flight Path

F. Problem 3.f and 4.f

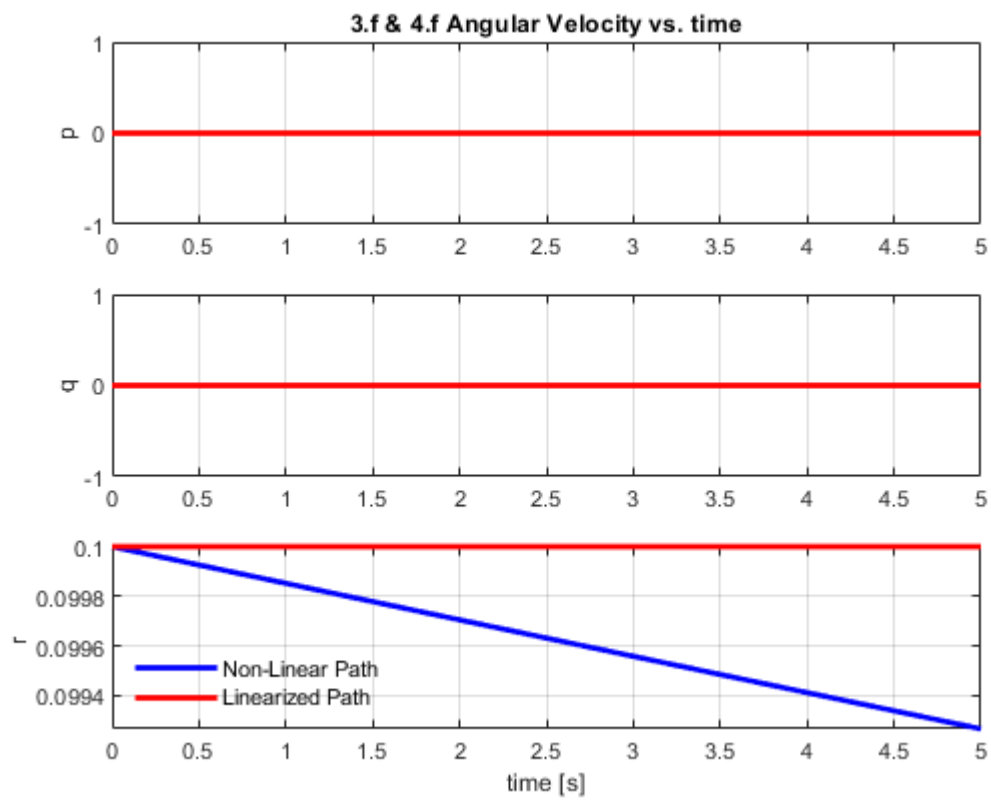


Fig. 26 Angular Velocity

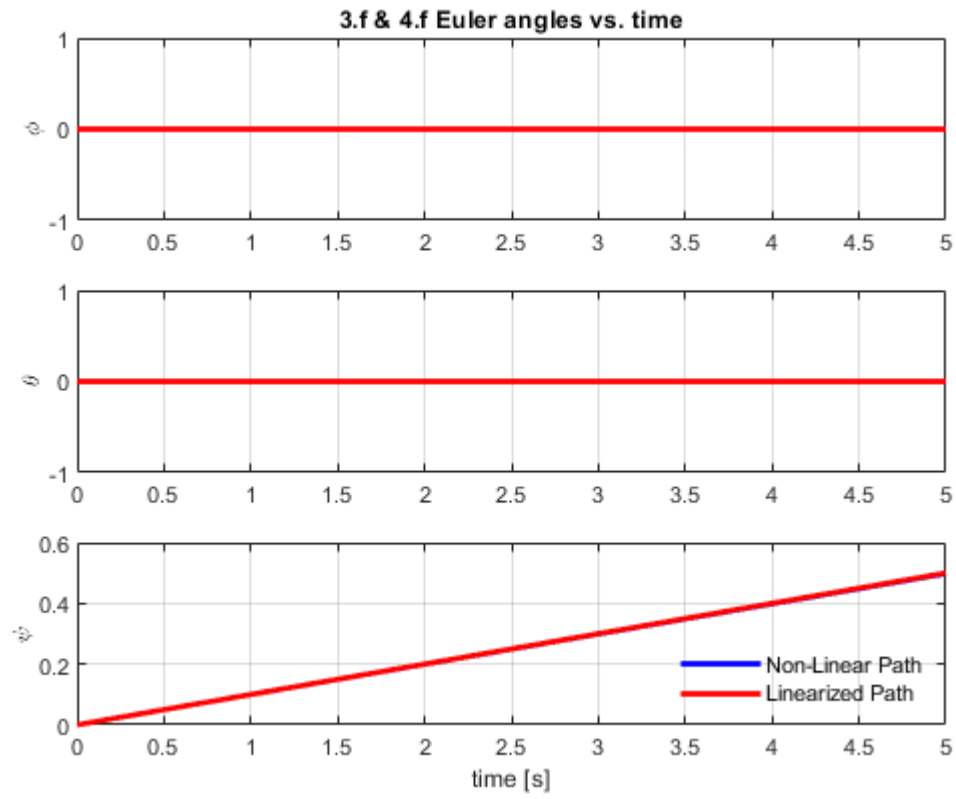


Fig. 27 Euler Angles

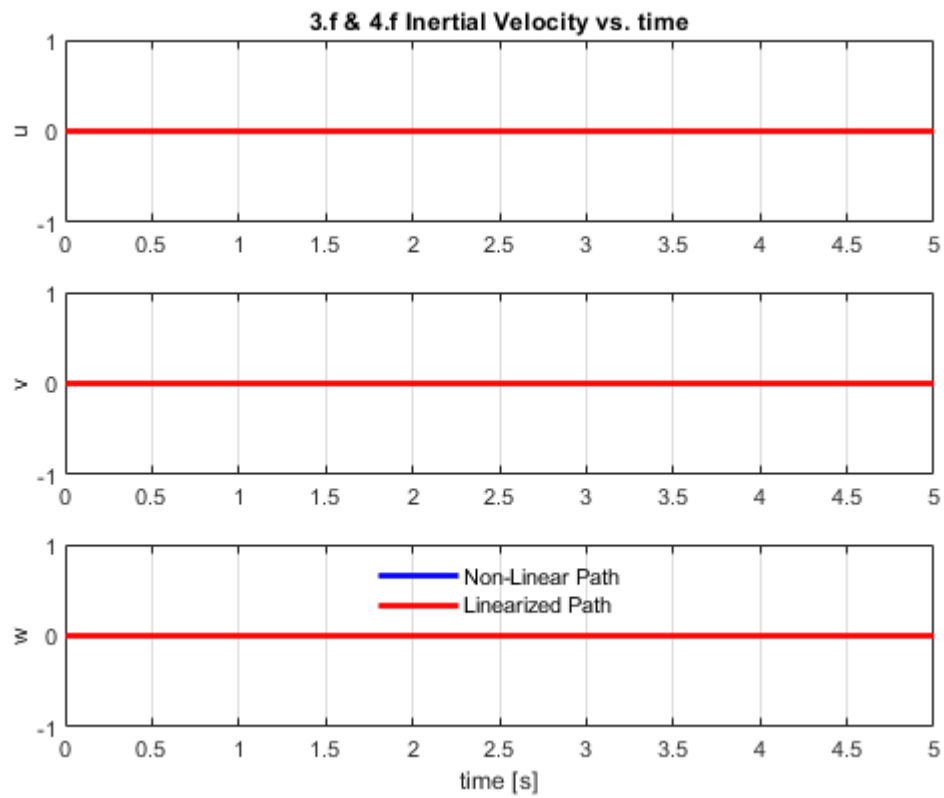


Fig. 28 Inertial Velocity

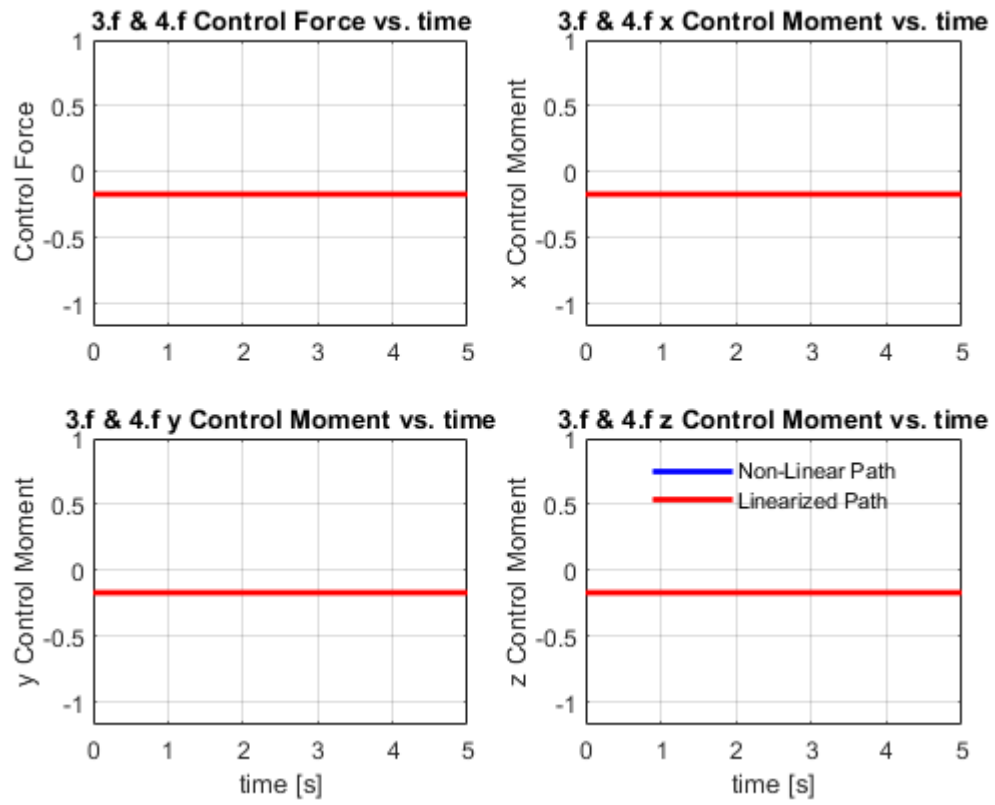


Fig. 29 Control Forces

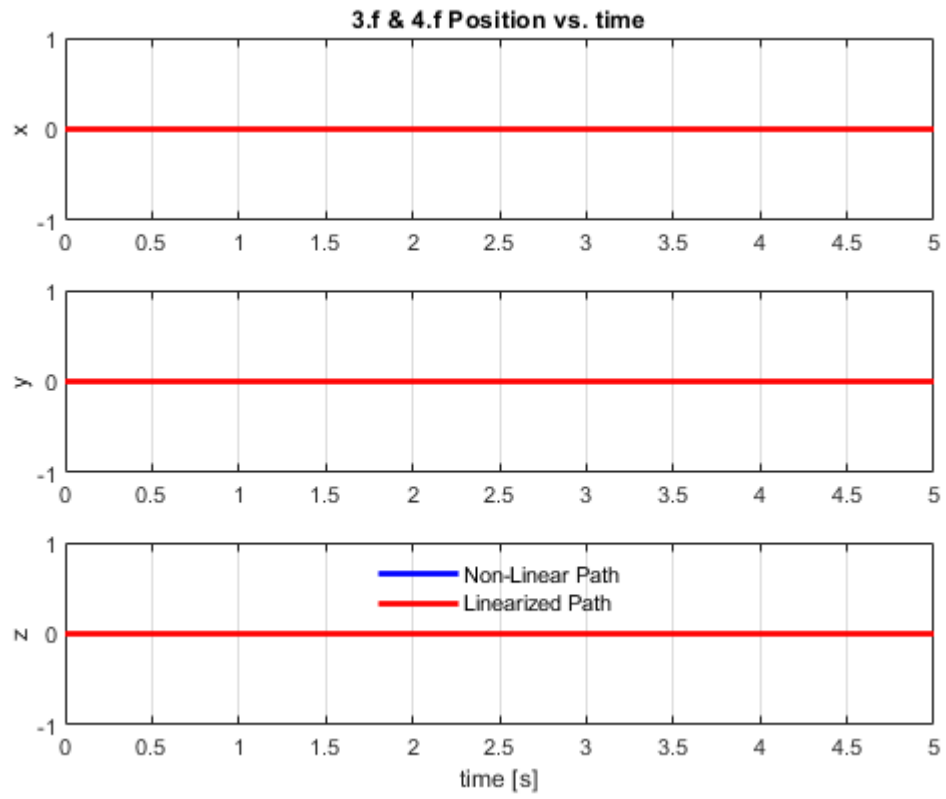


Fig. 30 Flight Path

G. Problem 5

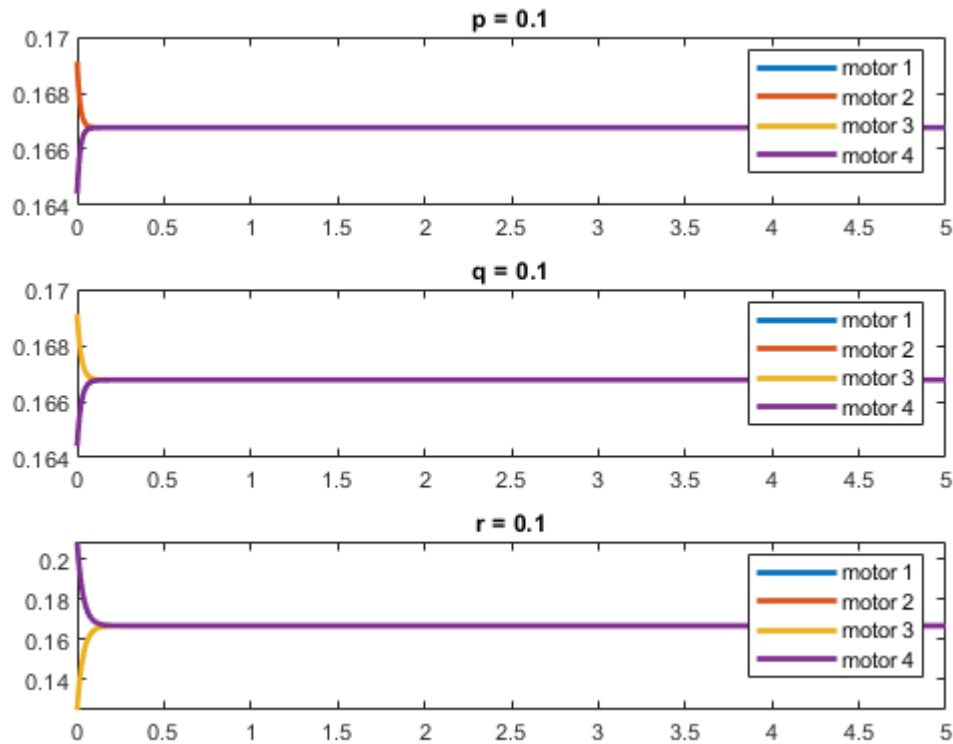


Fig. 31 Motor Forces with Feedback Control

V. Appendix

A. MATLAB Code

Main code

```

1 %% ASEN 3128 - Lab 3 - Main
2 % Script to compare the linearized and non-linearized models of a quadrotor
3 % in steady equilibrium flight. Additionally, control moments and forces
4 % are to be added to model how a quadrotor remains in steady hover flight
5 %
6 % Author: Cole MacPherson
7 % Collaborators: S. Packard, D. Wolfe, D. Zhao
8 % Date: 5th Mar 2021
9
10 %% Housekeeping
11 clc;
12 clear;
13 close all;
14 tic
15
16 %% declare constants
17 m = 0.068; % mass of the quadrotor [kg]
18 R = 0.06; % radial distance from CG to propeller [m]
19 k_m = 0.0024; % control moment coefficient [N*m/N]
20 I_x = 6.8e-5; % x-axis moment of inertia [kg*m^2]
21 I_y = 9.2e-5; % y-axis moment of inertia [kg*m^2]

```

```

22 I_z = 1.35e-4; % z-axis moment of inertia [kg*m^2]
23 nu = 1e-3; % aerodynamic force coefficient [N/(m/s)^2]
24 mu = 2e-6; % aerodynamic moment coefficient [N*m/(rad/s)^2]
25 g = 9.81; % gravitational constant [m/s^2]
26
27 Z_c = -m*g; % control force
28 L_c = 0; % x control moment
29 M_c = 0; % y control moment
30 N_c = 0; % z control moment
31
32 tspan = [0 5]; % time to integrate over
33
34 %% initialize figure information
35 col = 'b';
36 fig_a = [1,2,3,4,5,6]; % figure a number vector
37 fig_b = [1,2,3,4,5,6] + 6; % figure b number vector
38 fig_c = [1,2,3,4,5,6] + 2*6; % figure c number vector
39 fig_d = [1,2,3,4,5,6] + 3*6; % figure d number vector
40 fig_e = [1,2,3,4,5,6] + 4*6; % figure e number vector
41 fig_f = [1,2,3,4,5,6] + 5*6; % figure f number vector
42
43 %% 3a
44 state_vec_0 = [0 0 0 deg2rad(5) 0 0 0 0 0 0 0]';
45 [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);
46
47 PlotAircraftSim(t_a,state_vec_a,ones(length(t_a),4)*Z_c/4,fig_a,col,'3.a')
48
49 %% 3b
50 state_vec_0 = [0 0 0 0 deg2rad(5) 0 0 0 0 0 0]';
51 [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);
52
53 PlotAircraftSim(t_b,state_vec_b,ones(length(t_b),4)*Z_c/4,fig_b,col,'3.b')
54
55 %% 3c
56 state_vec_0 = [0 0 0 0 0 deg2rad(5) 0 0 0 0 0 0]';
57 [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);
58
59 PlotAircraftSim(t_c,state_vec_c,ones(length(t_c),4)*Z_c/4,fig_c,col,'3.c')
60
61 %% 3d
62 [p,q,r] = rollrate2pqr(0,0,0,0,0.1,0,0);
63 state_vec_0 = [0 0 0 0 0 0 0 0 p q r]';
64 [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);
65
66 PlotAircraftSim(t_d,state_vec_d,ones(length(t_d),4)*Z_c/4,fig_d,col,'3.d')
67
68 %% 3e
69 [p,q,r] = rollrate2pqr(0,0,0,0,0,0.1,0);
70 state_vec_0 = [0 0 0 0 0 0 0 0 p q r]';
71 [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);
72
73 PlotAircraftSim(t_e,state_vec_e,ones(length(t_e),4)*Z_c/4,fig_e,col,'3.e')
74
75 %% 3f
76 [p,q,r] = rollrate2pqr(0,0,0,0,0,0,0.1);
77 state_vec_0 = [0 0 0 0 0 0 0 0 p q r]';
78 [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);
79
80 PlotAircraftSim(t_f,state_vec_f,ones(length(t_f),4)*Z_c/4,fig_f,col,'3.f')
81
82 %% Redefine figure line color for problem 4
83 col = 'r';

```

```

84
85 %% 4a
86 t_final = 5;
87 state_vec_0 = [0 0 0 deg2rad(5) 0 0 0 0 0 0 0]';
88 [state_vec_a,t_a] = linearizedEOM(state_vec_0,t_final);
89
90 PlotAircraftSim(t_a,state_vec_a,ones(length(t_a),4)*Z_c/4,fig_a,col,'3.a & 4.a')
91
92 %% 4b
93 state_vec_0 = [0 0 0 0 deg2rad(5) 0 0 0 0 0 0]';
94 [state_vec_b,t_b] = linearizedEOM(state_vec_0,t_final);
95
96 PlotAircraftSim(t_b,state_vec_b,ones(length(t_b),4)*Z_c/4,fig_b,col,'3.b & 4.b')
97
98 %% 4c
99 state_vec_0 = [0 0 0 0 0 deg2rad(5) 0 0 0 0 0]';
100 [state_vec_c,t_c] = linearizedEOM(state_vec_0,t_final);
101
102 PlotAircraftSim(t_c,state_vec_c,ones(length(t_c),4)*Z_c/4,fig_c,col,'3.c & 4.c')
103
104 %% 4d
105 [p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
106 state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
107 [state_vec_d,t_d] = linearizedEOM(state_vec_0,t_final);
108
109 PlotAircraftSim(t_d,state_vec_d,ones(length(t_d),4)*Z_c/4,fig_d,col,'3.d & 4.d')
110
111 %% 4e
112 [p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
113 state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
114 [state_vec_e,t_e] = linearizedEOM(state_vec_0,t_final);
115
116 PlotAircraftSim(t_e,state_vec_e,ones(length(t_e),4)*Z_c/4,fig_e,col,'3.e & 4.e')
117
118 %% 4f
119 [p,q,r] = rollrate2pqr(0,0,0,0,0,0.1);
120 state_vec_0 = [0 0 0 0 0 0 0 0 0 p q r]';
121 [state_vec_f,t_f] = linearizedEOM(state_vec_0,t_final);
122
123 PlotAircraftSim(t_f,state_vec_f,ones(length(t_f),4)*Z_c/4,fig_f,col,'3.f & 4.f')
124
125 %% 5d
126 [p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
127 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]';
128 [t_d,state_vec_d] = ode45(@(t_d,state_vec_d) quadrotorODE_controlled(t_d,state_vec_d,m,I_x,I_y,
129     I_z,nu,mu,Z_c,L_c,M_c,N_c),tspan,state_vec_0);
130
131 % compute forces
132 F_5d = ComputeMotorForces(state_vec_d(:,13),state_vec_d(:,14),state_vec_d(:,15),state_vec_d(:,16),
133     R,k_m);
134
135 % plot force
136 figure
137 subplot(3,1,1)
138 plot(t_d,F_5d(1,:), 'linewidth',2); hold on;
139 plot(t_d,F_5d(2,:), 'linewidth',2); hold on;
140 plot(t_d,F_5d(3,:), 'linewidth',2); hold on;
141 plot(t_d,F_5d(4,:), 'linewidth',2); hold on;
142 title(['p = ' num2str(p)]);
143 legend('motor 1','motor 2','motor 3','motor 4');
144 hold on;
145
146 %% 5e
147 [p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
148 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]';
149 [t_e,state_vec_e] = ode45(@(t_e,state_vec_e) quadrotorODE_controlled(t_e,state_vec_e,m,I_x,I_y,
150     I_z,nu,mu,Z_c,L_c,M_c,N_c),tspan,state_vec_0);

```

```

149 % compute forces
150 F_5e = ComputeMotorForces(state_vec_e(:,13),state_vec_e(:,14),state_vec_e(:,15),state_vec_e(:,16)
    ,R,k_m);
151
152 % plot force
153 subplot(3,1,2)
154 plot(t_e,F_5e(1,:), 'linewidth',2); hold on;
155 plot(t_e,F_5e(2,:), 'linewidth',2); hold on;
156 plot(t_e,F_5e(3,:), 'linewidth',2); hold on;
157 plot(t_e,F_5e(4,:), 'linewidth',2); hold on;
158 title(['q = ' num2str(q)]);
159 legend('motor 1','motor 2','motor 3','motor 4');
160 hold on;
161
162 %% 5f
163 [p,q,r] = rollrate2pqr(0,0,0,0,0,0.1);
164 state_vec_0 = [0 0 0 0 0 0 0 0 p q r Z_c -0.004*p -0.004*q -0.004*r]';
165 [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);
166
167 % compute forces
168 F_5f = ComputeMotorForces(state_vec_f(:,13),state_vec_f(:,14),state_vec_f(:,15),state_vec_f(:,16)
    ,R,k_m);
169
170 % plot force
171 subplot(3,1,3)
172 plot(t_f,F_5f(1,:), 'linewidth',2); hold on;
173 plot(t_f,F_5f(2,:), 'linewidth',2); hold on;
174 plot(t_f,F_5f(3,:), 'linewidth',2); hold on;
175 plot(t_f,F_5f(4,:), 'linewidth',2); hold on;
176 title(['r = ' num2str(r)]);
177 legend('motor 1','motor 2','motor 3','motor 4');
178 hold on;
179
180 %% Plot Housekeeping
181 % make legends for each plot
182 for N = 1:6
183     figure(6*N-5)
184     legend('Non-Linear Path','Linearized Path','location','best');
185     legend('boxoff');
186
187     figure(6*N-4)
188     legend('Non-Linear Path','Linearized Path','location','best');
189     legend('boxoff');
190
191     figure(6*N-3)
192     legend('Non-Linear Path','Linearized Path','location','best');
193     legend('boxoff');
194
195     figure(6*N-2)
196     legend('Non-Linear Path','Linearized Path','location','best');
197     legend('boxoff');
198
199     figure(6*N-1)
200     legend('Non-Linear Path','Linearized Path','location','best');
201     legend('boxoff');
202
203     figure(6*N)
204     legend('Non-Linear Path','Start','End','Linearized Path');
205 end
206
207 %% End Housekeeping
208 toc

```

Plotting (Problem 1)

```

1 function PlotAircraftSim(t,state,control,fig,col,num)
2

```

```

3  %% Plot position vs time
4  figure(fig(1))
5  subplot(3,1,1)
6  plot(t,state(:,1),col,'linewidth',2); hold on;
7  grid on
8  ylabel('x');
9  title([num ' Position vs. time']);
10 subplot(3,1,2)
11 plot(t,state(:,2),col,'linewidth',2); hold on;
12 grid on
13 ylabel('y');
14 subplot(3,1,3)
15 plot(t,state(:,3),col,'linewidth',2); hold on;
16 grid on
17 ylabel('z');
18 xlabel('time [s]');
19
20 %% Plot Euler andgles vs time
21 figure(fig(2))
22 subplot(3,1,1)
23 plot(t,state(:,4),col,'linewidth',2); hold on;
24 grid on
25 ylabel('\phi');
26 title([num ' Euler angles vs. time']);
27 subplot(3,1,2)
28 plot(t,state(:,5),col,'linewidth',2); hold on;
29 grid on
30 ylabel('\theta');
31 subplot(3,1,3)
32 plot(t,state(:,6),col,'linewidth',2); hold on;
33 grid on
34 ylabel('\psi');
35 xlabel('time [s]');
36
37 %% Plot velocity vs time
38 figure(fig(3))
39 subplot(3,1,1)
40 plot(t,state(:,7),col,'linewidth',2); hold on;
41 grid on
42 ylabel('u');
43 title([num ' Inertial Velocity vs. time']);
44 subplot(3,1,2)
45 plot(t,state(:,8),col,'linewidth',2); hold on;
46 grid on
47 ylabel('v');
48 subplot(3,1,3)
49 plot(t,state(:,9),col,'linewidth',2); hold on;
50 grid on
51 ylabel('w');
52 xlabel('time [s]');
53
54 %% Plot angular velocity vs time
55 figure(fig(4))
56 subplot(3,1,1)
57 plot(t,state(:,10),col,'linewidth',2); hold on;
58 grid on
59 ylabel('p');
60 title([num ' Angular Velocity vs. time']);
61 subplot(3,1,2)
62 plot(t,state(:,11),col,'linewidth',2); hold on;
63 grid on
64 ylabel('q');
65 subplot(3,1,3)
66 plot(t,state(:,12),col,'linewidth',2); hold on;
67 grid on
68 ylabel('r');
69 xlabel('time [s]');
70

```

```

71  %% Plot control forces and moments vs time
72  figure(fig(5))
73  subplot(2,2,1)
74  plot(t,control(:,1),col,'linewidth',2); hold on;
75  grid on
76  title([num ' Control Force vs. time']);
77  ylabel('Control Force');
78  xlim([t(1) t(end)]);
79  subplot(2,2,2)
80  plot(t,control(:,2),col,'linewidth',2); hold on;
81  grid on
82  title([num ' x Control Moment vs. time']);
83  ylabel('x Control Moment');
84  xlim([t(1) t(end)]);
85  subplot(2,2,3)
86  plot(t,control(:,3),col,'linewidth',2); hold on;
87  grid on
88  title([num ' y Control Moment vs. time']);
89  ylabel('y Control Moment');
90  xlabel('time [s]');
91  xlim([t(1) t(end)]);
92  subplot(2,2,4)
93  plot(t,control(:,4),col,'linewidth',2); hold on;
94  grid on
95  title([num ' z Control Moment vs. time']);
96  xlabel('time [s]');
97  ylabel('z Control Moment');
98  xlim([t(1) t(end)]);
99
100 %% Plot flight path of the simulated quadrotor
101 figure(fig(6))
102 plot3(state(:,1),state(:,2),state(:,3),col,'linewidth',2); hold on;
103 plot3(state(1,1),state(1,2),state(1,3),'g.','markersize',20); hold on;
104 plot3(state(end,1),state(end,2),state(end,3),'r.','markersize',20); hold on;
105 set(gca, 'YDir','reverse')
106 set(gca, 'ZDir','reverse')
107 grid on
108 title([num ' Flight Path']);
109 xlabel('x');
110 ylabel('y');
111 zlabel('z');
112
113 end

```

Coord frame conversions

```

1  function [p,q,r] = rollrate2pqr(phi,theta,psi,phi_dot,theta_dot,psi_dot)
2
3  %% Sovle for pqr vector
4  pqr = [phi_dot,theta_dot,psi_dot]...
5        *inv([1, sin(phi)*tan(theta), cos(phi)*tan(theta);...
6             0, cos(phi), sin(phi);...
7             0, sin(phi)*sec(theta), cos(phi)*sec(theta)]);
8
9  %% Define p, q, and r based on previous calculations
10 p = pqr(1);
11 q = pqr(2);
12 r = pqr(3);
13
14 end

```

Non feedback control ODE for problem 3

```

1  function xdot = quadrotorODE(t,state_vec,m,I_x,I_y,I_z,n,mu,Z_c,L_c,M_c,N_c)
2
3  phi = state_vec(4); % roll angle
4  theta = state_vec(5); % pitch angle
5  psi = state_vec(6); % yaw angle

```

```

6   u = state_vec(7); % x velocity
7   v = state_vec(8); % y velocity
8   w = state_vec(9); % z velocity
9   p = state_vec(10); % roll velocity
10  q = state_vec(11); % pitch velocity
11  r = state_vec(12); % yaw velocity
12
13  g = 9.81; %Gravity m/s^2
14  M = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
15  V = norm([u;v;w]); % velocity
16  F = -n*V*[u;v;w]; % aerodynamic force
17
18  MomCntl = [L_c; M_c; N_c]; % control moment
19
20  % transformation matrix
21  R = [cos(theta)*cos(psi) sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi) cos(phi)*sin(theta)*
22        cos(psi)+sin(phi)*sin(psi);...
23        cos(theta)*sin(psi) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) cos(phi)*sin(theta)*
24        sin(psi)-sin(phi)*cos(psi);...
25        -sin(theta) sin(phi)*cos(theta) cos(phi)*cos(theta)];
26
27  % equations of motion
28  v_inertial = R * [u;v;w];
29  euler_dot = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);...
30              0 cos(phi) -sin(phi);...
31              0 sin(phi)*(1/cos(theta)) cos(phi)*(1/cos(theta))];...
32  * [p;q;r];
33  a_inertial = [r*v-q*w; p*w-r*u; q*u-p*v]...
34  + g*[-sin(theta);cos(theta)*sin(phi);cos(theta)*cos(phi)]...
35  + (1/m)*F + (1/m)*[0;0;Z_c];
36  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]...
37  + [(1/I_x)*M(1); (1/I_y)*M(2); (1/I_z)*M(3)]...
38  + [(1/I_x)*MomCntl(1); (1/I_y)*MomCntl(2); (1/I_z)*MomCntl(3)];
39
40  xdot = [v_inertial; euler_dot; a_inertial; a_angular]; % vecotr output for the integrals
41 end

```

Motor Forces (Problem 2)

```

1 function motor_forces = ComputeMotorForces(Z_c,L_c,M_c,N_c,r,k_m)
2
3 %% Moment to force matrix
4 m2f = [-1, -1, -1, -1;...
5        -r/sqrt(2), -r/sqrt(2), r/sqrt(2), r/sqrt(2);...
6        r/sqrt(2), -r/sqrt(2), -r/sqrt(2), r/sqrt(2);...
7        k_m, -k_m, k_m, -k_m];
8
9 %% Vecotr of control moments and force
10 f_vec = [Z_c, L_c, M_c, N_c];
11
12 %% Solve for motor forces
13 motor_forces = m2f\f_vec';
14
15 end

```

Linearization for problem 4

```

1 function [x,t] = linearizedEOM(x0,t_f)
2
3 g = 9.81; %acceleration due to gravity
4
5 %% Defien state space matrix
6 A = zeros(12,12);
7 A(1,7) = 1;
8 A(2,8) = 1;
9 A(3,9) = 1;
10 A(4,10) = 1;

```

```

11 A(5,11) = 1;
12 A(6,12) = 1;
13 A(7,5) = -g;
14 A(8,4) = g;
15
16 %% Define state space system
17 sys = ss(A,zeros(12,1),eye(12),0);
18
19 %% Simulate state space system
20 [x,t] = initial(sys,x0,t_f);
21
22 end

```

Feedback control ODE for problem 5

```

1 function xdot = quadrotorODE_controlled(t,state_vec,m,I_x,I_y,I_z,n,mu,Z_c,L_c,M_c,N_c)
2
3 %% Define variables
4 phi = state_vec(4); % roll angle
5 theta = state_vec(5); % pitch angle
6 psi = state_vec(6); % yaw angle
7 u = state_vec(7); % x velocity
8 v = state_vec(8); % y velocity
9 w = state_vec(9); % z velocity
10 p = state_vec(10); % roll velocity
11 q = state_vec(11); % pitch velocity
12 r = state_vec(12); % yaw velocity
13 g = 9.81; %Gravity m/s^2
14 M = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
15 V = norm([u;v;w]); % velocity
16 F = -n*V*[u;v;w]; % aerodynamic force
17 MCntl = -0.004*[p;q;r]; % control moment
18
19 %% Transformation matrix
20 R = [cos(theta)*cos(psi) sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi) cos(phi)*sin(theta)*
21      cos(psi)+sin(phi)*sin(psi);...
22      cos(theta)*sin(psi) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) cos(phi)*sin(theta)*
23      sin(psi)-sin(phi)*cos(psi);...
24      -sin(theta) sin(phi)*cos(theta) cos(phi)*cos(theta)];
25
26 %% Equations of motion
27 v_inertial = R*[u;v;w];
28 euler_dot = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);...
29             0 cos(phi) -sin(phi);...
30             0 sin(phi)*(1/cos(theta)) cos(phi)*(1/cos(theta))];...
31             * [p;q;r];
32 a_inertial = [r*v-q*w; p*w-r*u; q*u-p*v]...
33             + g*[-sin(theta);cos(theta)*sin(phi);cos(theta)*cos(phi)]...
34             + (1/m)*F + (1/m)*[0;0;Z_c];
35 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]...
36             + [(1/I_x)*M(1); (1/I_y)*M(2); (1/I_z)*M(3)]...
37             + [(1/I_x)*MCntl(1); (1/I_y)*MCntl(2); (1/I_z)*MCntl(3)];
38 deltaF = [0; -0.004*a_angular];
39
40 xdot = [v_inertial; euler_dot; a_inertial; a_angular; deltaF]; % vecotr output for the
41 integrals
42 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