ASEN 3128 - Lab 4

Cole MacPherson, Joshua Schmitz, Nick Herrington, Panitnan Yuvanondha
March 21st, 2020

Problem 1

Below are the calculation by hands for finding the gain values for lateral and longitudinal motion.

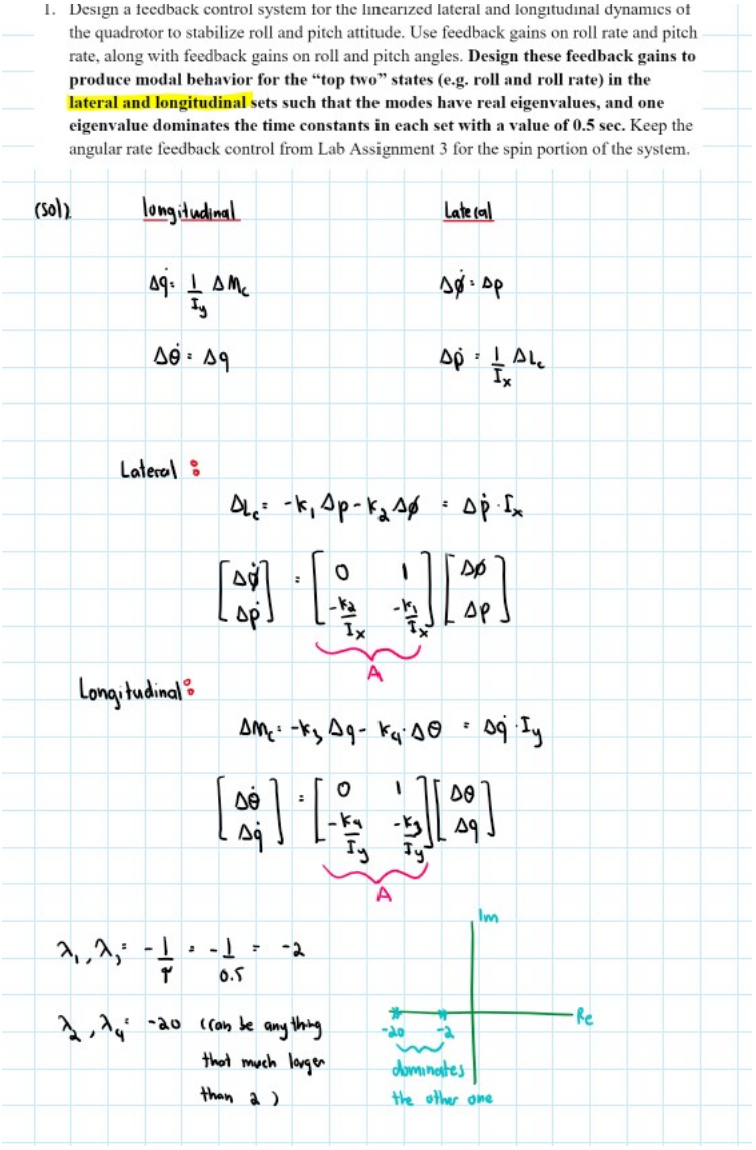

Fig. 1 Question 1 Calculation

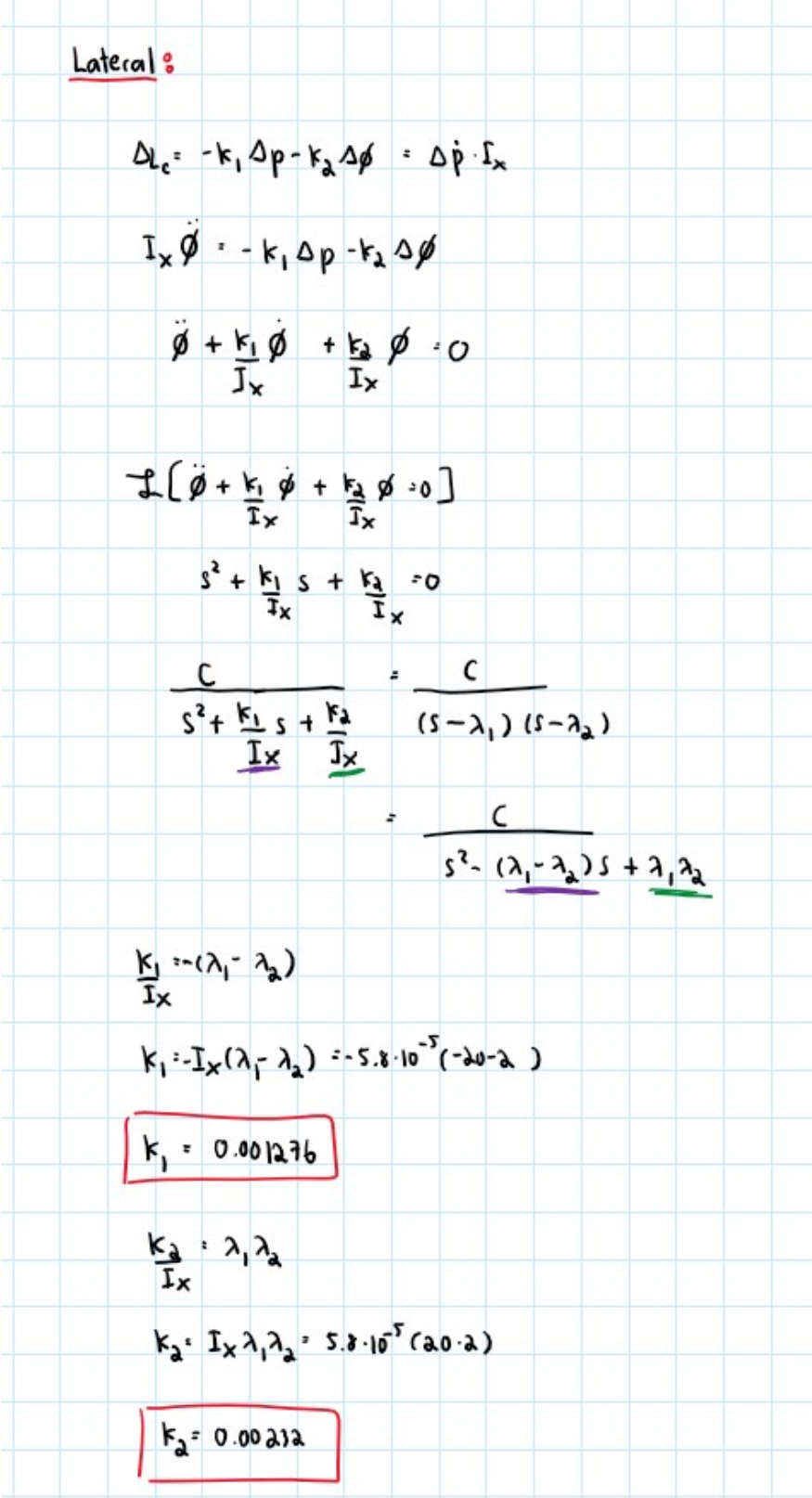

Fig. 2 Question 1 Calculation

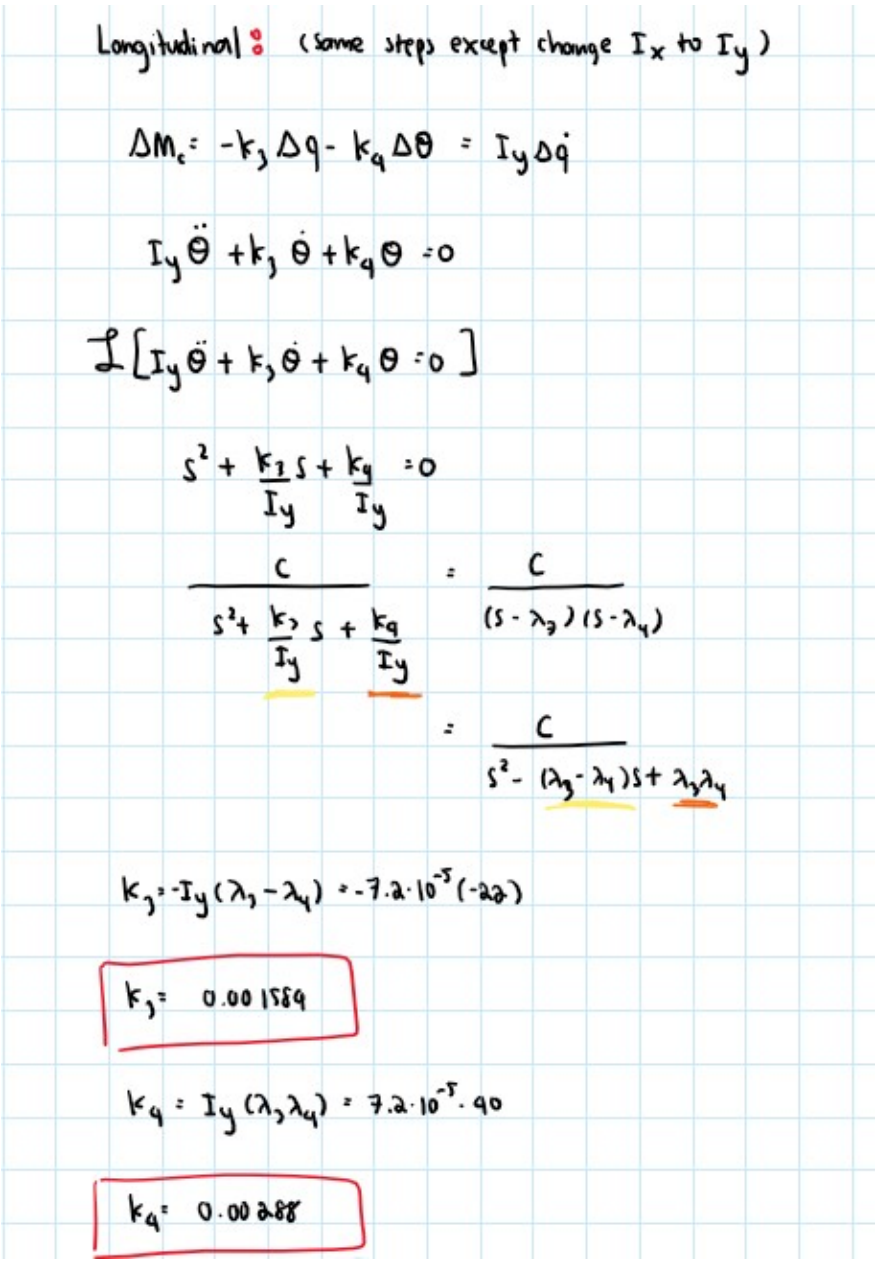

Fig. 3 Question 1 Calculation

Problem 2

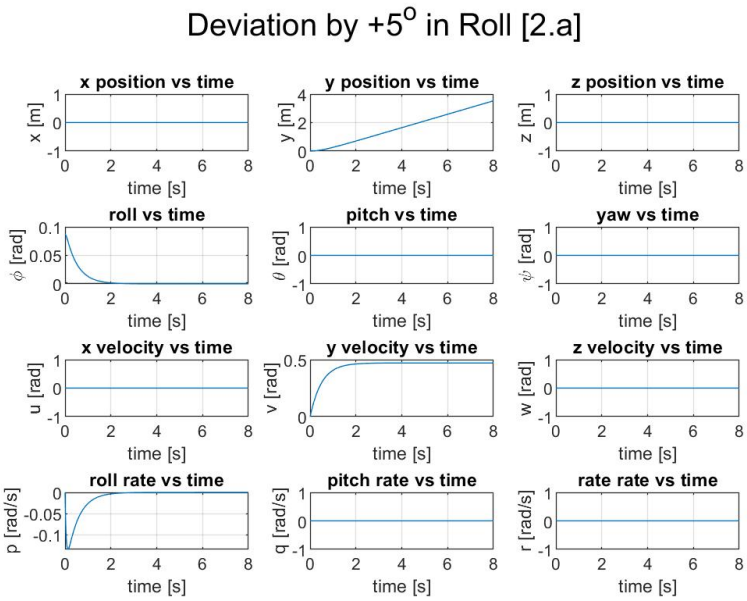

Fig. 4 Linearized Roll

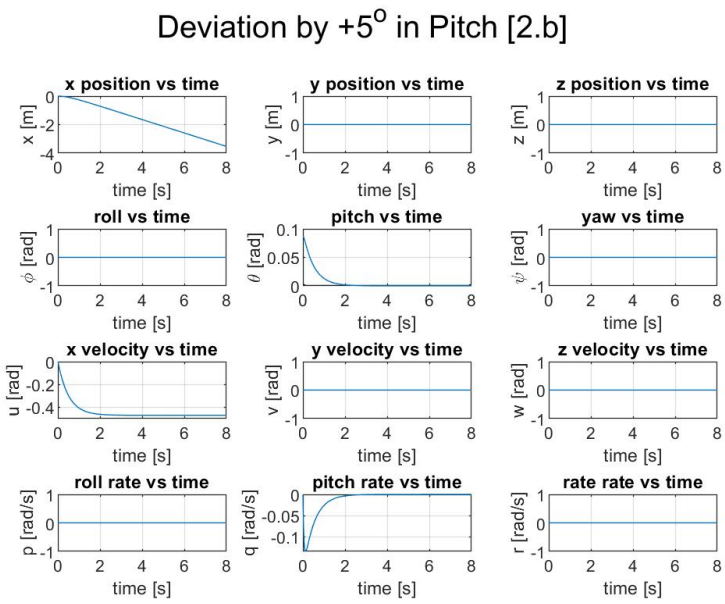

Fig. 5 Linearized Pitch

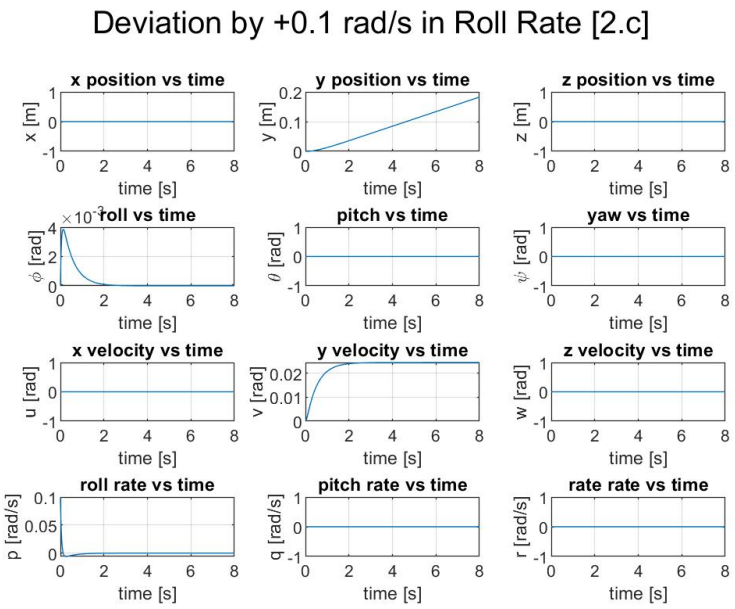

Fig. 6 Linearized Roll Rate

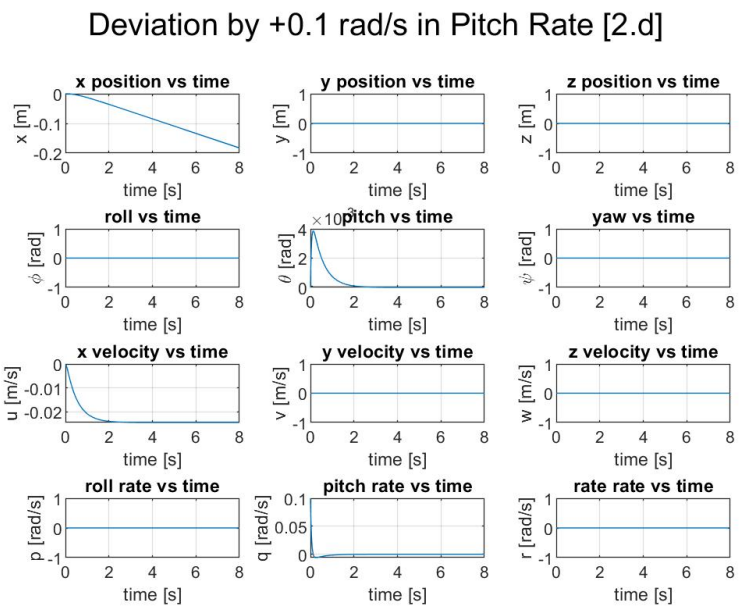

Fig. 7 Linearized Pitch Rate

The results for the linearized model make sense because when we add a perturbation to the quadcoptor, the altitude doesn't change. This can seen in the above graphs, the z-components do not change and remain zero for the duration of the ODE simulation. This is a result from the linearization of the equations of motion. Since we are looking at infinitesimal time steps, it appears that the quadcoptor's altitude is not changing with time.

Problem 3

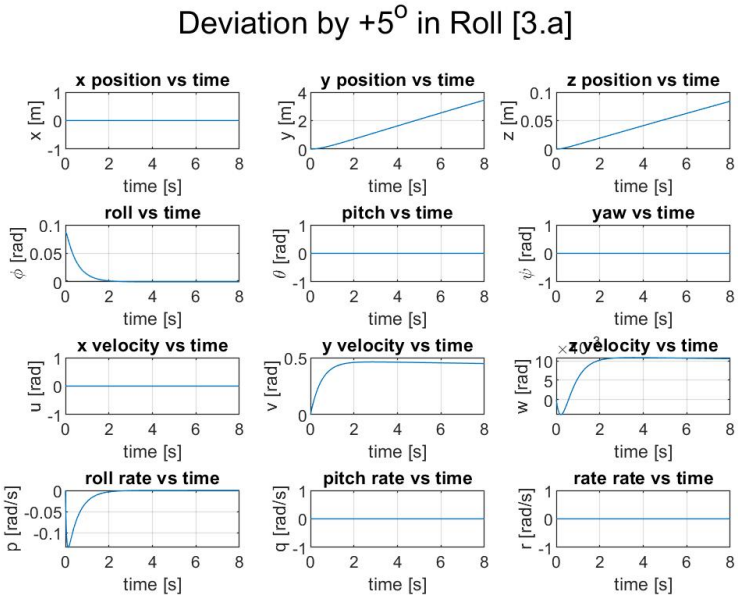

Fig. 8 Non-Linearized Roll

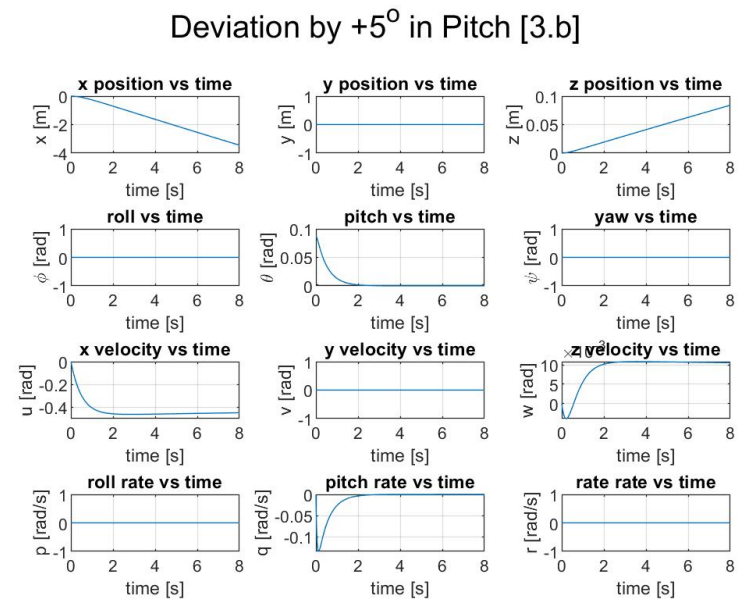

Fig. 9 Non-Linearized Pitch

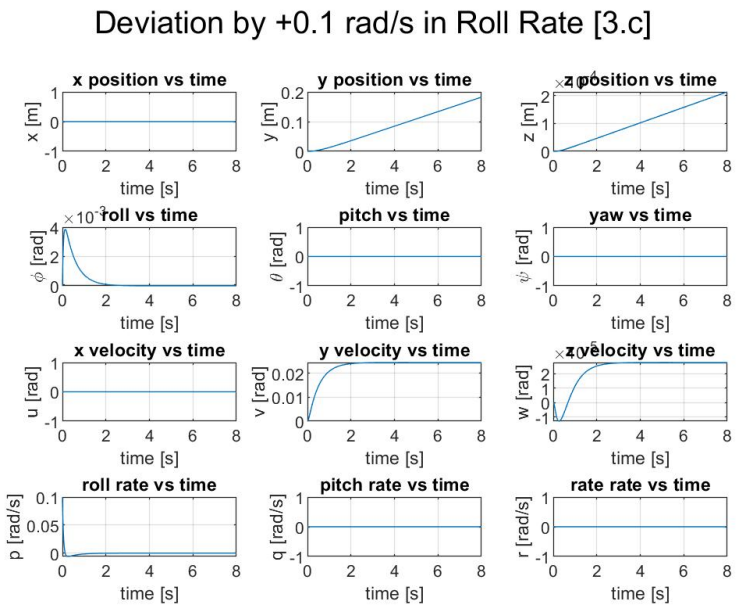

Fig. 10 Non-Linearized Roll Rate

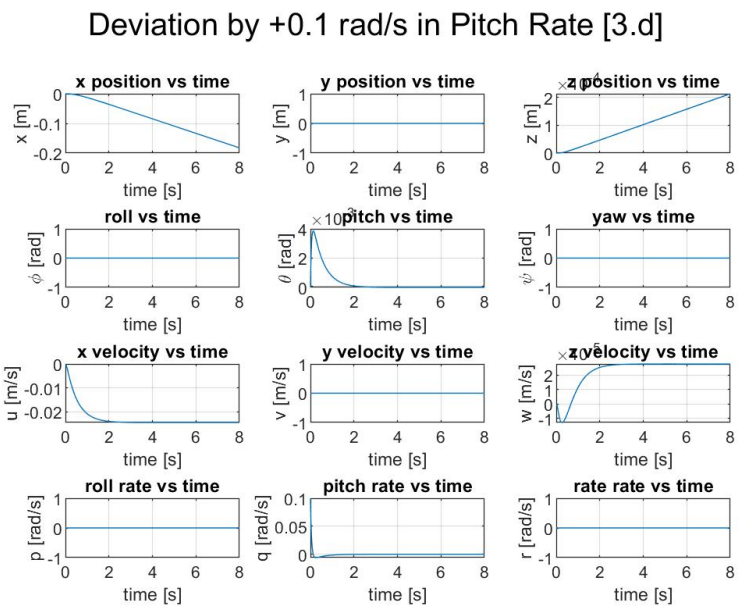

Fig. 11 Non-Linearized Pitch Rate

When modeling the motion of the quadcoptor with the non-linearized EOM's, we get similar results as to the linearized model except when get changes in the z-components which is expected. This model verifies that we correctly performed the calculations for the linearized model since the x-components and y-components followed similar changes in the graphs.

Problem 4

The following plots are the eigenvalues obtained from the MATLAB built-in function eig over the range of k3 values for both lateral and longitudinal motion in the complex plane. In order to meet the design objective, the real eigenvalue must be around 0.8 since the time constant given is 1.25 sec ($\tau = \frac{1}{1.25}$), therefore the k3 value is picked at the location where the real eigenvalue is or close to 0.8. From doing this, the k3 value is 0.009 for lateral and 0.0076 for longitudinal.

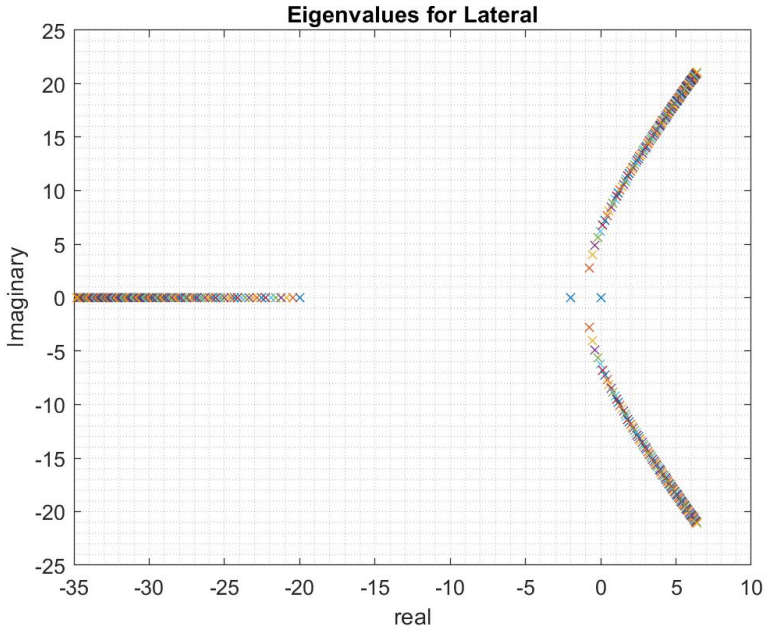

Fig. 12 Eigenvalues for Lateral

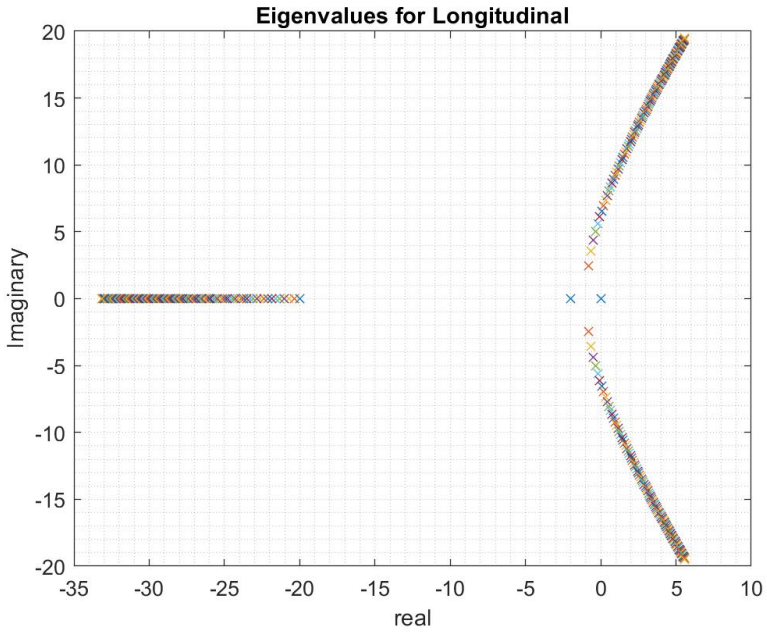

Fig. 13 Eigenvalues for Longitudinal

Problem 5

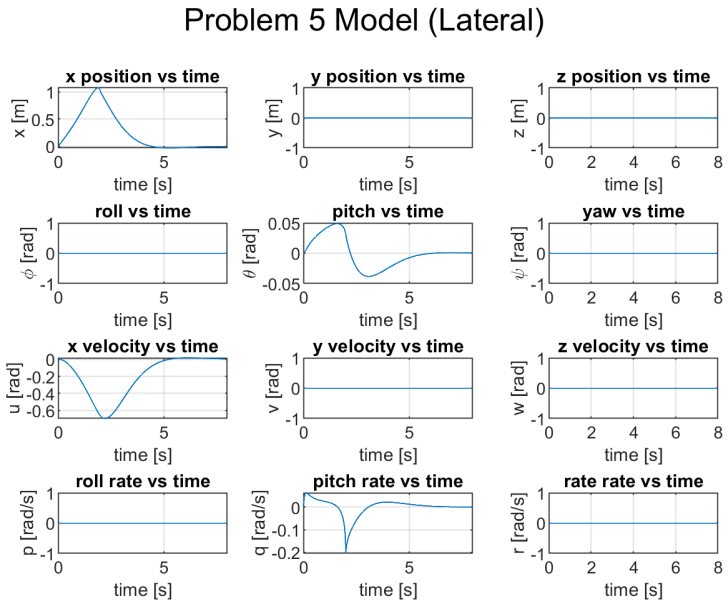

Fig. 14 Question 5 lateral Simulation

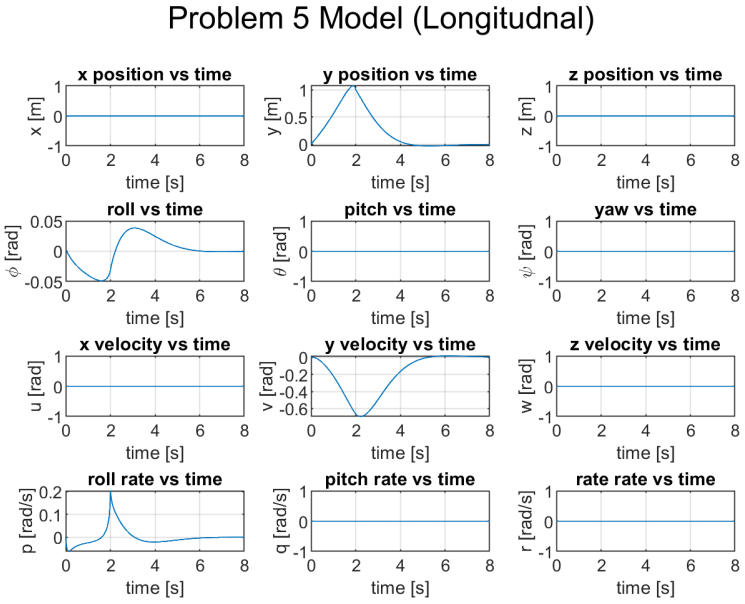

Fig. 15 Question 5 longitudinal Simulation

As seen in figures 14 and 15, the question 5 simulation takes in a steady state hover state vector in and moves the drone to a position of 1 meter in both the y and x directions for separate simulations. After this the drone then steadies itself out to go back to a steady hover state. This simulation can be confirmed as correct as the values presented in figures 14 and 15 act as expected through the simulation and the beginning and end states are steady hover states.

Appendix

A. Team Participation Table

Name	Plan	Model	Experiment	Results	Report	Code	Ack
Cole MacPherson	1	X	X	2	1	2	CM
Joshua Schmitz	1	X	X	1	1	2	JS
Nick Herrington	1	X	X	1	2	1	NH
Panitan Yuvanondha	1	X	X	1	2	1	PY

B. Code

This script is used for problems 2 through 5 and calls the required functions shown after this script:

```
%% ASEN 3128 - LAb 4 - Main
% Script to compare the lineared and non-linearized control responses of a
% quadrotor with a feedback loop incorperated into the equations of motion
%
%
% Author: Cole MacPherson
% Collaborators: N. Herrington, J. Schmitz, P. Yuvanondha
% 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 = 5.8e-5; % x-axis moment of inertia [kg*m^2]
I_y = 7.2e-5; % y-axis moment of inertia [kg*m^2]
I_z = 1.0e-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]
lambda_1 = -2; % 1st eigenvalue
lambda_2 = -20; % 2nd eigenvalue

%% Problem 1
% Solve for k_1 and k_2 (Lateral)
syms k_1 k_2 % declare k1 and k2 as symbolic variables
eqn1 = lambda_1^2 + lambda_1*(k_1/I_x) + (k_2/I_x) == 0; % 1st eigenvalue solution
eqn2 = lambda_2^2 + lambda_2*(k_1/I_x) + (k_2/I_x) == 0; % 2nd eigenvalue solution

[A,b] = equationsToMatrix([eqn1, eqn2],[k_1, k_2]); % convert equations to a matrix

x_lat = double(A\b); % solve the system of equations
k_1 = x_lat(1); % define k_1 value from system solutions
k_2 = x_lat(2); % define k_2 value from system solutions
k_3 = 0;

% Solve for k_3 and k_4 (Longitudinal)
syms k_4 k_5 % declare k1 and k2 as symbolic variables
eqn1 = lambda_1^2 + lambda_1*(k_4/I_y) + (k_5/I_y) == 0; % 1st eigenvalue solution
eqn2 = lambda_2^2 + lambda_2*(k_4/I_y) + (k_5/I_y) == 0; % 2nd eigenvalue solution

[C,d] = equationsToMatrix([eqn1, eqn2],[k_4, k_5]); % convert equations to a matrix
```

```
48
49 x_long = double(C\d); % solve the system of equations
50 k_4 = x_long(1); % define k_3 value from system solutions
51 k_5 = x_long(2); % define k_4 value from system solutions
52 k_6 = 0;
53
54 %% Problem 2 (Closed-Loop Linearized)
55 % define known vector to pass into function
56 knowns = [mu nu I_x I_y I_z m g k_1 k_2 k_3 k_4 k_5 k_6 R k_m 0];
57 % define time span for ode45
58 t_span = [0 8];
59 % define motor trim forces
60 forces = (1/4)*m*g.*ones(1,4);
61 % define perturbations
62 perturbations = zeros(1,3);
63
64 % a (+5 deg in roll)
65 state_vec = zeros(1,12);
66 state_vec(4) = deg2rad(5);
67
68 [t_2a, state_2a] = ode45(@(t_2a,state_2a) linearizedEOM(t_2a,state_2a,forces,perturbations,knowns
       ),t_span,state_vec);
69
70 % b (+5 deg in pitch)
71 state_vec = zeros(1,12);
72 state_vec(5) = deg2rad(5);
73
74 [t_2b, state_2b] = ode45(@(t_2b,state_2b) linearizedEOM(t_2b,state_2b,forces,perturbations,knowns
       ),t_span,state_vec);
75
76 % c (+0.1 rad/sec in roll rate)
77 [p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
78 state_vec = zeros(1,12);
79 state_vec(10:12) = [p,q,r];
80
81 [t_2c, state_2c] = ode45(@(t_2c,state_2c) linearizedEOM(t_2c,state_2c,forces,perturbations,knowns
       ),t_span,state_vec);
82
83 % d (+0.1 rad/sec in pitch rate)
84 [p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
85 state_vec = zeros(1,12);
86 state_vec(10:12) = [p,q,r];
87
88 [t_2d, state_2d] = ode45(@(t_2d,state_2d) linearizedEOM(t_2d,state_2d,forces,perturbations,knowns
       ),t_span,state_vec);
89
90 %% Problem 2 Plots
91 plotP2N3(t_2a,state_2a,t_2b,state_2b,t_2c,state_2c,t_2d,state_2d,'2');
92
93 %% Problem 3 (Non-Linearized)
94 % a (+5 deg in roll)
95 state_vec = zeros(1,12);
96 state_vec(4) = deg2rad(5);
97 perturbations(2) = 0.5;
98
99 [t_3a, state_3a] = ode45(@(t_3a,state_3a) quadrotorODE(t_3a,state_3a,forces,knowns),t_span,
       state_vec);
100
101 % b (+5 deg in pitch)
102 state_vec = zeros(1,12);
103 state_vec(5) = deg2rad(5);
104
105 [t_3b, state_3b] = ode45(@(t_3b,state_3b) quadrotorODE(t_3b,state_3b,forces,knowns),t_span,
       state_vec);
106
107 % c (+0.1 rad/sec in roll rate)
108 [p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
109 state_vec = zeros(1,12);
```

```
state_vec(10:12) = [p,q,r];

[t_3c, state_3c] = ode45(@(t_3c,state_3c) quadrotorODE(t_3c,state_3c,forces,knowns),t_span,
    state_vec);

% d (+0.1 rad/sec in pitch rate)
[p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
state_vec = zeros(1,12);
state_vec(10:12) = [p,q,r];

[t_3d, state_3d] = ode45(@(t_3d,state_3d) quadrotorODE(t_3d,state_3d,forces,knowns),t_span,
    state_vec);

%% Problem 3 Plots
plotP2N3(t_3a,state_3a,t_3b,state_3b,t_3c,state_3c,t_3d,state_3d,'3');

%% Problem 4
k_3 = -18.475e-5; % found through guess and check
k_6 = 22.95e-5; % found through guess and check

% define known vector to pass into function
knowns = [mu nu I_x I_y I_z m g k_1 k_2 k_3 k_4 k_5 k_6 R k_m 0];

% a (+5 deg in roll)
state_vec = zeros(1,12);
state_vec(4) = deg2rad(5);

[t_4a, state_4a] = ode45(@(t_4a,state_4a) linearizedEOM(t_4a,state_4a,forces,perturbations,knowns
    ),t_span,state_vec);

% b (+5 deg in pitch)
state_vec = zeros(1,12);
state_vec(5) = deg2rad(5);

[t_4b, state_4b] = ode45(@(t_4b,state_4b) linearizedEOM(t_4b,state_4b,forces,perturbations,knowns
    ),t_span,state_vec);

% c (+0.1 rad/sec in roll rate)
[p,q,r] = rollrate2pqr(0,0,0,0.1,0,0);
state_vec = zeros(1,12);
state_vec(10:12) = [p,q,r];

[t_4c, state_4c] = ode45(@(t_4c,state_4c) linearizedEOM(t_4c,state_4c,forces,perturbations,knowns
    ),t_span,state_vec);

% d (+0.1 rad/sec in pitch rate)
[p,q,r] = rollrate2pqr(0,0,0,0,0.1,0);
state_vec = zeros(1,12);
state_vec(10:12) = [p,q,r];

[t_4d, state_4d] = ode45(@(t_4d,state_4d) linearizedEOM(t_4d,state_4d,forces,perturbations,knowns
    ),t_span,state_vec);

%% Problem 4 Plots
plotP2N3(t_4a,state_4a,t_4b,state_4b,t_4c,state_4c,t_4d,state_4d,'4');

%% Problem 5

% define known vector to pass into function
knowns = [mu nu I_x I_y I_z m g k_1 k_2 k_3 k_4 k_5 k_6 R k_m 1 1 0 2];

% a lateral
state_vec = zeros(1,12);

[t_5a, state_5a] = ode45(@(t_5a,state_5a) linearizedEOM(t_5a,state_5a,forces,perturbations,knowns
    ),t_span,state_vec);

% define known vector to pass into function
```

```
knowns = [mu nu I_x I_y I_z m g k_1 k_2 k_3 k_4 k_5 k_6 R k_m 1 0 1 2];

% b longitudnal
state_vec = zeros(1,12);

[t_5b, state_5b] = ode45(@(t_5b,state_5b) linearizedEOM(t_5b,state_5b,forces,perturbations,knowns
    ),t_span,state_vec);

%% Problem 5 Plots
figure
sgtitle(['Problem 5 Model (Lateral)']);
    % x position vs time
    subplot(4,3,1);
    plot(t_5a, state_5a(:,1));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('x [m]');
    title('x position vs time');
    % y position vs time
    subplot(4,3,2);
    plot(t_5a, state_5a(:,2));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('y [m]');
    title('y position vs time');
    % z position vs time
    subplot(4,3,3);
    plot(t_5a, state_5a(:,3));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('z [m]');
    title('z position vs time');
    % roll vs time
    subplot(4,3,4);
    plot(t_5a, state_5a(:,4));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('\phi [rad]');
    title('roll vs time');
    % pitch vs time
    subplot(4,3,5);
    plot(t_5a, state_5a(:,5));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('\theta [rad]');
    title('pitch vs time');
    % yaw vs time
    subplot(4,3,6);
    plot(t_5a, state_5a(:,6));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('\psi [rad]');
    title('yaw vs time');
    % x velocity vs time
    subplot(4,3,7);
    plot(t_5a, state_5a(:,7));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('u [rad]');
    title('x velocity vs time');
    % y velocity vs time
```

```
    subplot(4,3,8);
    plot(t_5a, state_5a(:,8));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('v [rad]');
    title('y velocity vs time');
    % z velocity vs time
    subplot(4,3,9);
    plot(t_5a, state_5a(:,9));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('w [rad]');
    title('z velocity vs time');
    % roll rate vs time
    subplot(4,3,10);
    plot(t_5a, state_5a(:,10));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('p [rad/s]');
    title('roll rate vs time');
    % pitch rate vs time
    subplot(4,3,11);
    plot(t_5a, state_5a(:,11));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('q [rad/s]');
    title('pitch rate vs time');
    % yaw rate vs time
    subplot(4,3,12);
    plot(t_5a, state_5a(:,12));
    grid on;
    xlim([0 t_5a(end)]);
    xlabel('time [s]');
    ylabel('r [rad/s]');
    title('rate rate vs time');

figure
sgtitle(['Problem 5 Model (Longitudnal)']);
    % x position vs time
    subplot(4,3,1);
    plot(t_5b, state_5b(:,1));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('x [m]');
    title('x position vs time');
    % y position vs time
    subplot(4,3,2);
    plot(t_5b, state_5b(:,2));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('y [m]');
    title('y position vs time');
    % z position vs time
    subplot(4,3,3);
    plot(t_5b, state_5b(:,3));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('z [m]');
    title('z position vs time');
    % roll vs time
    subplot(4,3,4);
```

```
    plot(t_5b, state_5b(:,4));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('\phi [rad]');
    title('roll vs time');
    % pitch vs time
    subplot(4,3,5);
    plot(t_5b, state_5b(:,5));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('\theta [rad]');
    title('pitch vs time');
    % yaw vs time
    subplot(4,3,6);
    plot(t_5b, state_5b(:,6));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('\psi [rad]');
    title('yaw vs time');
    % x velocity vs time
    subplot(4,3,7);
    plot(t_5b, state_5b(:,7));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('u [rad]');
    title('x velocity vs time');
    % y velocity vs time
    subplot(4,3,8);
    plot(t_5b, state_5b(:,8));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('v [rad]');
    title('y velocity vs time');
    % z velocity vs time
    subplot(4,3,9);
    plot(t_5b, state_5b(:,9));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('w [rad]');
    title('z velocity vs time');
    % roll rate vs time
    subplot(4,3,10);
    plot(t_5b, state_5b(:,10));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('p [rad/s]');
    title('roll rate vs time');
    % pitch rate vs time
    subplot(4,3,11);
    plot(t_5b, state_5b(:,11));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
    ylabel('q [rad/s]');
    title('pitch rate vs time');
    % yaw rate vs time
    subplot(4,3,12);
    plot(t_5b, state_5b(:,12));
    grid on;
    xlim([0 t_5b(end)]);
    xlabel('time [s]');
```

```
    ylabel('r [rad/s]');
    title('rate rate vs time');


%% End Housekeeping
toc
```

This is the non-linearized function used in problem 3:

```
function state_dot = quadrotorODE(t,state_vec,forces,knowns)

% Function to calculate the respose of a non-linearized quadrotor model
% with feedback loops intergrated
%
% Inputs:
%         t             -> time
%         state_vec     -> state vector
%         forces        -> motor forces
%         knowns        -> vector of known values
% Outputs:
%         state_dot     -> derivative of the state vector

    %% Define known variables
    mu = knowns(1); % [N/(m/s)^2]
    nu = knowns(2); % [N/(rad/s)^2]
    I_x = knowns(3); % x moment of inertia [kg*m^2]
    I_y = knowns(4); % y moment of inertia [kg*m^2]
    I_z = knowns(5); % z moment of inertia [kg*m^2]
    m = knowns(6); % mass [kg]
    g = knowns(7); % gravity [m/s^2]
    k_1 = knowns(8); % lateral k_1
    k_2 = knowns(9); % lateral k_2
    k_3 = knowns(10); % lateral k_3
    k_4 = knowns(11); % longitudinal k_1
    k_5 = knowns(12); % longitudinal k_2
    k_6 = knowns(13); % longitudinal k_3
    k_m = knowns(15); % moment coefficient

    %% 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 rate
    q = state_vec(11); % pitch rate
    r = state_vec(12); % yaw rate
    f_1 = forces(1); % motor force 1
    f_2 = forces(2); % motor force 2
    f_3 = forces(3); % motor force 3
    f_4 = forces(4); % motor force 4

    %% Define Aerodynamic/Control Forces/Moments
    M_aero = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
    F_aero = -nu*norm([u;v;w])*[u;v;w]; % aerodynamic force
    M_cntl = [-k_1*p-k_2*phi;...
        -k_4*q-k_5*theta;...
        k_m*(f_1-f_2+f_3-f_4)]; % control moment
    F_cntl = [0; 0; -sum(forces)]; % control force

    %% Non-Linear Equations of Motion
    % velocity
    xyz_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);...
        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)]...
```

```
        * [u;v;w];
    % rate of change of the euler angles
    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];
    % acceleration
    v_dot = [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_aero + (1/m)*F_cntl;
    % angular acceleration
    v_angular_dot = [((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_aero(1); (1/I_y)*M_aero(2); (1/I_z)*M_aero(3)]...
        + [(1/I_x)*M_cntl(1); (1/I_y)*M_cntl(2); (1/I_z)*M_cntl(3)];

    state_dot = [xyz_dot; euler_dot; v_dot; v_angular_dot]; % vector output for the integrals

end
```

This is the linearized function used in problems 2, 4, and 5:

```
function state_dot = linearizedEOM(t, state_vec, forces, perturbations, knowns)

% Function to calculate the respose of a linearized quadrotor model with
% feedback loops intergrated
%
% Inputs:
%         t             -> time
%         state_vec     -> state vector
%         forces        -> motor forces
%         perturbations -> vector to describe any perturbations
%         knowns        -> vector of known values
% Outputs:
%         state_dot     -> derivative of the state vector

    %% Define known variables
    mu = knowns(1); % [N/(m/s)^2]
    I_x = knowns(3); % x moment of inertia [kg*m^2]
    I_y = knowns(4); % y moment of inertia [kg*m^2]
    I_z = knowns(5); % z moment of inertia [kg*m^2]
    g = knowns(7); % gravity [m/s^2]
    k_1 = knowns(8); % lateral k_1
    k_2 = knowns(9); % lateral k_2
    k_3 = knowns(10); % lateral k_3
    k_4 = knowns(11); % longitudinal k_1
    k_5 = knowns(12); % longitudinal k_2
    k_6 = knowns(13); % longitudinal k_3
    R = knowns(14); % radius from CG to motor force [m]
    k_m = knowns(15); % control moment coefficient

    %% Define variables
    x = state_vec(1); % x position
    y = state_vec(2); % y position
    z = state_vec(3); % z position
    phi = state_vec(4); % roll angle
    theta = state_vec(5); % pitch angle
    u = state_vec(7); % x velocity
    v = state_vec(8); % y velocity
    w = state_vec(9); % z velocity
    p = state_vec(10); % roll rate
    q = state_vec(11); % pitch rate
    r = state_vec(12); % yaw rate
    f_1 = forces(1); % motor force 1
    f_2 = forces(2); % motor force 2
    f_3 = forces(3); % motor force 3
    f_4 = forces(4); % motor force 4

    %% Define Aerodynamic/Control Forces/Moments
```

```
    M_aero = -mu*norm([p;q;r])*[p;q;r]; % aerodynamic moment
    M_cntl = [(R/sqrt(2))*(-f_1-f_2+f_3+f_4);...
        (R/sqrt(2))*(f_1-f_2-f_3+f_4);...
        k_m*(f_1-f_2+f_3-f_4)]; % control moment

    %% Linear Equations of Motion
    % rate of change of euler angles
    euler_dot = [p; q; r];
    % acceleration
    v_dot = g*[-theta; phi; 0];

    if knowns(16) == 1 && (t >= 0 && t <= 2)
        % time and distance requiered for required travel
        t_r = knowns(19);
        x_r = knowns(17);
        y_r = knowns(18);
        % velocity to meet requirements
        u_r = (x_r-x)/(t_r-t);
        v_r = (y_r-y)/(t_r-t);
        % angular acceleration
        v_angular_dot = [-((k_1*p)/I_x) - ((k_2*phi)/I_x) + (k_3*(v_r-v))/I_x;...
            -((k_4*q)/I_y) - ((k_5*theta)/I_y) + (k_6*(u_r-u))/I_y;...
            (M_aero(3) + M_cntl(3) + perturbations(3))/I_z];
        % velocity
        xyz_dot = [u_r-u; v_r-v; w];
    else
        % angular acceleration
        v_angular_dot = [-((k_1*p)/I_x) - ((k_2*phi)/I_x) + (k_3*v)/I_x;...
            -((k_4*q)/I_y) - ((k_5*theta)/I_y) + (k_6*u)/I_y;...
            (M_aero(3) + M_cntl(3) + perturbations(3))/I_z];
        % velocity
        xyz_dot = [u; v; w];
    end

    state_dot = [xyz_dot; euler_dot; v_dot; v_angular_dot]; % vector output for the integrals

end
```

This is the function used to plot problems 2, 3, and 4:

```
function plotP2N3(t_2a,state_2a,t_2b,state_2b,t_2c,state_2c,t_2d,state_2d,N)

    %% Problem 2 Plots
    % a (+5 deg in roll)
    figure
    sgtitle(['Deviation by +5^{o} in Roll [' N '.a]']);
        % x position vs time
        subplot(4,3,1);
        plot(t_2a, state_2a(:,1));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('x [m]');
        title('x position vs time');
        % y position vs time
        subplot(4,3,2);
        plot(t_2a, state_2a(:,2));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('y [m]');
        title('y position vs time');
        % z position vs time
        subplot(4,3,3);
        plot(t_2a, state_2a(:,3));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
```

```
        ylabel('z [m]');
        title('z position vs time');
        % roll vs time
        subplot(4,3,4);
        plot(t_2a, state_2a(:,4));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('\phi [rad]');
        title('roll vs time');
        % pitch vs time
        subplot(4,3,5);
        plot(t_2a, state_2a(:,5));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('\theta [rad]');
        title('pitch vs time');
        % yaw vs time
        subplot(4,3,6);
        plot(t_2a, state_2a(:,6));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('\psi [rad]');
        title('yaw vs time');
        % x velocity vs time
        subplot(4,3,7);
        plot(t_2a, state_2a(:,7));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('u [rad]');
        title('x velocity vs time');
        % y velocity vs time
        subplot(4,3,8);
        plot(t_2a, state_2a(:,8));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('v [rad]');
        title('y velocity vs time');
        % z velocity vs time
        subplot(4,3,9);
        plot(t_2a, state_2a(:,9));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('w [rad]');
        title('z velocity vs time');
        % roll rate vs time
        subplot(4,3,10);
        plot(t_2a, state_2a(:,10));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('p [rad/s]');
        title('roll rate vs time');
        % pitch rate vs time
        subplot(4,3,11);
        plot(t_2a, state_2a(:,11));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('q [rad/s]');
        title('pitch rate vs time');
        % yaw rate vs time
        subplot(4,3,12);
```

```
        plot(t_2a, state_2a(:,12));
        grid on;
        xlim([0 t_2a(end)]);
        xlabel('time [s]');
        ylabel('r [rad/s]');
        title('rate rate vs time');

    % b (+5 deg in pitch)
    figure
    sgtitle(['Deviation by +5^{o} in Pitch [' N '.b]']);
        % x position vs time
        subplot(4,3,1);
        plot(t_2b, state_2b(:,1));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('x [m]');
        title('x position vs time');
        % y position vs time
        subplot(4,3,2);
        plot(t_2b, state_2b(:,2));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('y [m]');
        title('y position vs time');
        % z position vs time
        subplot(4,3,3);
        plot(t_2b, state_2b(:,3));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('z [m]');
        title('z position vs time');
        % roll vs time
        subplot(4,3,4);
        plot(t_2b, state_2b(:,4));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('\phi [rad]');
        title('roll vs time');
        % pitch vs time
        subplot(4,3,5);
        plot(t_2b, state_2b(:,5));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('\theta [rad]');
        title('pitch vs time');
        % yaw vs time
        subplot(4,3,6);
        plot(t_2b, state_2b(:,6));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('\psi [rad]');
        title('yaw vs time');
        % x velocity vs time
        subplot(4,3,7);
        plot(t_2b, state_2b(:,7));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('u [rad]');
        title('x velocity vs time');
        % y velocity vs time
        subplot(4,3,8);
```

```
        plot(t_2b, state_2b(:,8));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('v [rad]');
        title('y velocity vs time');
        % z velocity vs time
        subplot(4,3,9);
        plot(t_2b, state_2b(:,9));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('w [rad]');
        title('z velocity vs time');
        % roll rate vs time
        subplot(4,3,10);
        plot(t_2b, state_2b(:,10));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('p [rad/s]');
        title('roll rate vs time');
        % pitch rate vs time
        subplot(4,3,11);
        plot(t_2b, state_2b(:,11));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('q [rad/s]');
        title('pitch rate vs time');
        % yaw rate vs time
        subplot(4,3,12);
        plot(t_2b, state_2b(:,12));
        grid on;
        xlim([0 t_2b(end)]);
        xlabel('time [s]');
        ylabel('r [rad/s]');
        title('rate rate vs time');

    % c (+0.1 rad/s in roll rate)
    figure
    sgtitle(['Deviation by +0.1 rad/s in Roll Rate [' N '.c]']);
        % x position vs time
        subplot(4,3,1);
        plot(t_2c, state_2c(:,1));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('x [m]');
        title('x position vs time');
        % y position vs time
        subplot(4,3,2);
        plot(t_2c, state_2c(:,2));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('y [m]');
        title('y position vs time');
        % z position vs time
        subplot(4,3,3);
        plot(t_2c, state_2c(:,3));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('z [m]');
        title('z position vs time');
        % roll vs time
        subplot(4,3,4);
```

```
        plot(t_2c, state_2c(:,4));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('\phi [rad]');
        title('roll vs time');
        % pitch vs time
        subplot(4,3,5);
        plot(t_2c, state_2c(:,5));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('\theta [rad]');
        title('pitch vs time');
        % yaw vs time
        subplot(4,3,6);
        plot(t_2c, state_2c(:,6));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('\psi [rad]');
        title('yaw vs time');
        % x velocity vs time
        subplot(4,3,7);
        plot(t_2c, state_2c(:,7));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('u [rad]');
        title('x velocity vs time');
        % y velocity vs time
        subplot(4,3,8);
        plot(t_2c, state_2c(:,8));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('v [rad]');
        title('y velocity vs time');
        % z velocity vs time
        subplot(4,3,9);
        plot(t_2c, state_2c(:,9));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('w [rad]');
        title('z velocity vs time');
        % roll rate vs time
        subplot(4,3,10);
        plot(t_2c, state_2c(:,10));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('p [rad/s]');
        title('roll rate vs time');
        % pitch rate vs time
        subplot(4,3,11);
        plot(t_2c, state_2c(:,11));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
        ylabel('q [rad/s]');
        title('pitch rate vs time');
        % yaw rate vs time
        subplot(4,3,12);
        plot(t_2c, state_2c(:,12));
        grid on;
        xlim([0 t_2c(end)]);
        xlabel('time [s]');
```

```
        ylabel('r [rad/s]');
        title('rate rate vs time');

    % d (+0.1 rad/s in pitch rate)
    figure
    sgtitle(['Deviation by +0.1 rad/s in Pitch Rate [' N '.d]']);
        % x position vs time
        subplot(4,3,1);
        plot(t_2d, state_2d(:,1));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('x [m]');
        title('x position vs time');
        % y position vs time
        subplot(4,3,2);
        plot(t_2d, state_2d(:,2));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('y [m]');
        title('y position vs time');
        % z position vs time
        subplot(4,3,3);
        plot(t_2d, state_2d(:,3));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('z [m]');
        title('z position vs time');
        % roll vs time
        subplot(4,3,4);
        plot(t_2d, state_2d(:,4));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('\phi [rad]');
        title('roll vs time');
        % pitch vs time
        subplot(4,3,5);
        plot(t_2d, state_2d(:,5));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('\theta [rad]');
        title('pitch vs time');
        % yaw vs time
        subplot(4,3,6);
        plot(t_2d, state_2d(:,6));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('\psi [rad]');
        title('yaw vs time');
        % x velocity vs time
        subplot(4,3,7);
        plot(t_2d, state_2d(:,7));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('u [m/s]');
        title('x velocity vs time');
        % y velocity vs time
        subplot(4,3,8);
        plot(t_2d, state_2d(:,8));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
```

```
        ylabel('v [m/s]');
        title('y velocity vs time');
        % z velocity vs time
        subplot(4,3,9);
        plot(t_2d, state_2d(:,9));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('w [m/s]');
        title('z velocity vs time');
        % roll rate vs time
        subplot(4,3,10);
        plot(t_2d, state_2d(:,10));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('p [rad/s]');
        title('roll rate vs time');
        % pitch rate vs time
        subplot(4,3,11);
        plot(t_2d, state_2d(:,11));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('q [rad/s]');
        title('pitch rate vs time');
        % yaw rate vs time
        subplot(4,3,12);
        plot(t_2d, state_2d(:,12));
        grid on;
        xlim([0 t_2d(end)]);
        xlabel('time [s]');
        ylabel('r [rad/s]');
        title('rate rate vs time');

end
```

This is the function used to convert the roll rate to p, q, and r values

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

% Function to calculate the roll, pitch, and yaw from a given position with
% a known roll rate, pitch rate, and yaw rate
%
% Inputs:
%         phi       -> yaw
%         theta     -> pitch
%         psi       -> yaw
%         phi_dot   -> roll rate
%         theta_dot -> pitch rate
%         psi_dot   -> yaw rate
% Outputs:
%         p         -> roll velocity
%         q         -> pitch velocity
%         r         -> yaw velocity


    %% 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
```

This is the eigenvalues plot :

```
% Lab 4
% Nick Herrington, Cole MacPherson, Joshua Schmitz, Panitnan Yuvanondha

clc;clear;close all

%% 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 = 5.8e-5; % x-axis moment of inertia [kg*m^2]
I_y = 7.2e-5; % y-axis moment of inertia [kg*m^2]
I_z = 1.0e-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]
k1=0.001276;
k2=0.00232;
k3=0.001584;
k4=0.00288;



%% Problem 4
% Lateral
k3Lat = 0:0.001:0.1;
for i = 1:length(k3Lat)
A_lat = [0 g 0;0 0 1; -(k3Lat(i)/I_x) -(k2/I_x) -(k1/I_x)];
lam_lat(:,i) = eig(A_lat);
figure(1)
plot(real(lam_lat(:,i)),imag(lam_lat(:,i)),'x')
hold on
grid minor
title('Eigenvalues for Lateral')
xlabel('real')
ylabel('Imaginary')
end
% for lateral at index = 10 , the real eigenvalue is about 0.8 (tau=1/1.25)
 k3lat=k3Lat(10)



% Longitudinal
k3lon= 0:0.001:0.1;
for i = 1:length(k3Lat)
 A_lon = [0 -g 0;0 0 1; (k3lon(i)/I_y) -(k4/I_y) -(k3/I_y)];
 lam_lon(:,i) = eig(A_lon);
figure(2)
plot(real(lam_lon(:,i)),imag(lam_lon(:,i)),'x')
hold on
grid minor
title('Eigenvalues for Longitudinal')
xlabel('real')
ylabel('Imaginary')
end

% for longitudinal at index = 77  , the real eigenvalue is about 0.8 (tau=1/1.25)
 k3Lon=k3lon(77)
```

References

[1] Smead Department of Aerospace Engineering. ASEN 3128 Aircraft Dynamics - Spring 2021 LABORATORY 4 University of Colorado at Boulder. Lab Document and Supplemental Resources.