University of Colorado Boulder

ASEN 3128 - Aircraft Dynamics

Assignment 4 - Afternoon Section

Assignment 4

Author:
Benjiman Smith
105979656

Professor:
D Lawrence

10 February, 2020

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Contents

List of Figures

The objective of this assignment was to utilize modified ODE functions from Assignment 2 and 3 to develop and design a linearized and non-linearized feedback control system for the lateral and longitudinal dynamics of the quad copter to stabilize elevation and bank attitude. Feedback gains were found through utilization of a time constant given in the assignment document, which were then applied to the linear and non-linear models. Through comparison of the two control models, it was determined that both the linearized and non-linearized models have similar results. The linearized model was also compared to experimental values to find both similarities and discrepancies in results.

I. Nomenclature

m	=	mass $[kg]$
r	=	body to motor distance $[m]$
I_x	=	moment of inertia in the body x direction $[kgm^2]$
I_y	=	moment of inertia in the body y direction $[kgm^2]$
I_z	=	moment of inertia in the body z direction $[kgm^2]$
u^E	=	inertial velocity in the x direction in body coordinates $[m/s]$
v^E	=	inertial velocity in the y direction in body coordinates $[m/s]$
w^E	=	inertial velocity in the z direction in body coordinates $[m/s]$
$\dot{u^E}$	=	derivative of the inertial in the x direction in body coordinates $[m/s^22]$
$\dot{v^E}$	=	derivative of the inertial velocity in the y direction in body coordinates $[m/s^2]$
$\dot{w^E}$	=	derivative of the inertial velocity in the z direction in body coordinates $[m/s^2]$
$\Delta\dot{u^E}$	=	derivative of the inertial in the body x direction deviated from trim $[m/s^2]$
$\Delta\dot{v^E}$	=	derivative of the inertial velocity in the body y direction deviated from trim $[m/s^2]$
$\dot{w^E}$	=	derivative of the inertial velocity in the body z direction deviated from trim $[m/s^2]$
p	=	inertial angular velocity component in the body x direction component in the inertial frame $[radian/s]$
q	=	inertial angular velocity component in the body y direction component in the inertial frame $[radian/s]$
r	=	inertial angular velocity component in the body z direction component in the inertial frame $[radian/s]$
$\dot{p}$	=	roll rate derivative $[radian/s^2]$
$\dot{q}$	=	pitch rate derivative $[radian/s^2]$
$\dot{r}$	=	yaw rate derivative $[radian/s^2]$
$\Delta\dot{p}$	=	roll rate derivative deviated from trim $[radian/s^2]$
$\Delta\dot{q}$	=	pitch rate derivative deviated from trim $[radian/s^2]$
$\Delta\dot{r}$	=	yaw rate derivative deviated from trim $[radian/s^2]$
ϕ	=	bank angle $[radian]$
θ	=	elevation angle $[radian]$
ψ	=	azimuth angle $[radian]$
$\dot{\phi}$	=	bank angle rate of change $[radian/s]$
$\dot{\theta}$	=	elevation angle rate of change $[rradian/s]$
$\dot{\psi}$	=	azimuth angle rate of change $[radian/s]$
$\Delta\dot{\phi}$	=	bank angle rate of change deviated from trim $[radian/s]$
$\Delta\dot{\theta}$	=	elevation angle rate of change deviated from trim $[radian/s]$
$\Delta\dot{\psi}$	=	azimuth angle rate of change deviated from trim $[radian/s]$
X_E	=	position vector in the x direction in inertial coordinates $[m]$
Y_E	=	position vector in the y direction in inertial coordinates $[m]$
Z_E	=	position vector in the z direction in inertial coordinates $[m]$
f_1	=	trim force exerted by motor 1 $[N]$
f_2	=	trim force exerted by motor 2 $[N]$
f_3	=	trim force exerted by motor 3 $[N]$
f_4	=	trim force exerted by motor 4 $[N]$

II. Question 1

For the first part of the lab, a feedback control system for the linearized lateral and longitudinal dynamics of the quad copter to stabilize elevation and bank attitude was designed. The gain values associated with this model were calculated as follows. The angular rate feedback control from Assignment 3 was kept for the azimuth portion of this system, which had a gain value of 0.004.

Problem 1, assignment 4

Thursday, March 21, 2019 12:01 PM

§ LATERAL CONTROL

$$\begin{bmatrix} \Delta\dot{P} \\ \Delta\dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{-k_1}{I_x} & -\frac{K_2}{I_x} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta P \\ \Delta\phi \end{bmatrix}$$

$$\dot{y} = A \cdot y$$

EQUATIONS OF MOTION FOR A CLOSED LOOP SYSTEM (K_1 & K_2 ARE LATERAL GAIN VALUES)

LET $A \cdot y = \lambda \cdot y$, WHERE λ IS A SCALAR

THEREFORE $(A - \lambda I)y = 0$

(I: IDENTITY MATRIX)

SINCE WE KNOW $y \neq 0$, AS WE HAVE REAL ΔP & $\Delta\phi$ VALUES, $\det(A - \lambda I) = 0$ IS TRUE, THEREFORE:

$$\det\begin{pmatrix} \frac{-K_1}{I_x} - \lambda & -\frac{K_2}{I_x} \\ 1 & -\lambda \end{pmatrix} = \left(-\frac{K_1}{I_x} - \lambda\right)(-\lambda) - \left(-\frac{K_2}{I_x}\right)(1) = 0$$

$$\Rightarrow \lambda^2 + \lambda\left(\frac{K_1}{I_x}\right) + \frac{K_2}{I_x} = 0$$

∘ THIS IS THE CHARACTERISTIC EQATION

FROM THE ASSIGNMENT #4 DOCUMENT ONE EIGENVALUE WAS SELECTED TO DOMINATE THE TIME CONSTANTS IN EACH SET WITH A VALUE OF 0.5 sec.

FROM THIS INITIAL CONDITION, THE DOMINANT EIGENVALUE CAN BE CALCULATED.

FROM CLASS, WE WERE GIVEN THE FOLLOWING EQUATION

$$e^{\lambda_1 t} = e^{nt} \cdot e^{j\omega t}$$

IN WHICH THE $e^{j\omega t}$ TERM IS OSCILLOTORY. SINCE THERE IS NO OSCILLATION PRESENT IN OUR SIMULATION, AS OUR CHARACTERISTIC EQUATION DOESN'T OUTPUT IMAGINARY NUMBERS, $\omega \Rightarrow 0$, THEREFORE THE EQUATION SIMPLIFIES TO:

$$e^{\lambda_1 t} = e^{nt} e^{0} \Rightarrow e^{\lambda_1 t} = e^{nt}$$

$$\Rightarrow \ln(e^{\lambda t}) = \ln(e^{nt}) \Rightarrow \lambda_1 t = nt$$

$$\therefore \boxed{\lambda_1 = n}$$

THIS ALLOWS FOR US TO SOLVE FOR λ_1 EASILY.

$\tau \equiv$ TIME CONSTANT

$\tau = 0.5$ sec (FROM ASSIGNMENT)

$e^{n\tau}$ ⇒ $e^{n\tau} \Rightarrow e^{n\tau} \Rightarrow$

$$e^{n\tau} = 0.37 \Rightarrow \ln(e^{n\tau}) = \ln(0.37)$$

$$\Rightarrow n\tau = -1 \Rightarrow \tau = -\frac{1}{n} \Rightarrow \tau = -\frac{1}{\lambda_1}$$

IN THIS CASE, $n = \lambda_1$

$$\Rightarrow \lambda_1 = -\frac{1}{\tau} = -\frac{1}{0.5\,\text{sec}} = \boxed{\lambda_1 = -2 \ [1/\text{sec}]}$$

IN ORDER TO MAKE THE 0.5 SEC TIME CONSTANT DOMINANT, AN EARLIER EQUATION WAS ANALYZED.

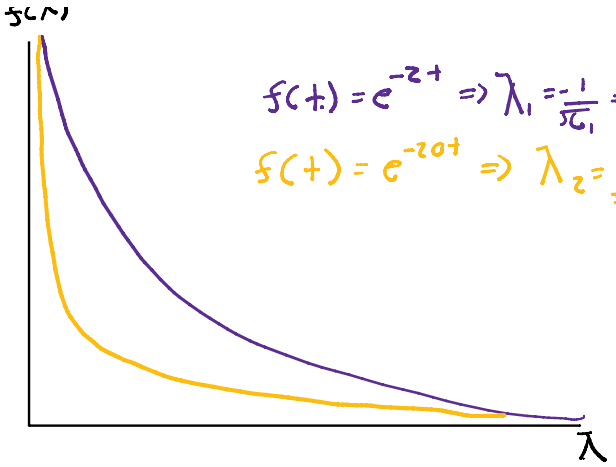
$n = -1/\tau$

$$e^{\lambda t} = e^{nt} \Rightarrow e^{\lambda t} = e^{-t/\tau} \Rightarrow \lambda = -\frac{1}{\tau}$$

IN ORDER TO SELECT THE SECOND EIGENVALUE TO MEET THIS ASSIGNMENT'S REQUIREMENTS, THE TIME CONSTANT FOR THE 2nd EIGENVALUE MUST BE MORE NEGATIVE THAN THE 1st EIGENVALUE'S TIME CONSTANT, AS THE TIME CONSTANTS CLOSER TO ZERO WILL BE THE LEAST DOMINANT, AS SHOWN BY THE FOLLOWING PLOT:

SCN

$f(t) = e^{-2t} \Rightarrow \lambda = -\frac{1}{\tau} \Rightarrow \tau = 0.5$



$f(t) = e^{-2t} \Rightarrow \lambda_1 = -\frac{1}{\tau_1} \Rightarrow \tau_1 = 0.5s$

$f(t) = e^{-20t} \Rightarrow \lambda_2 = \frac{1}{\tau_2} \Rightarrow \tau_2 = 0.05s$

AS SEEN FROM THE PLOT, THE MORE NEGATIVE THE 2ND EIGENVALUE IS, THE LESS DOMINATE IT IS. THEREFORE, FOR OUR SIMULATION, THE 2ND EIGENVALUE WAS SELECTED TO BE -20 [1/sec]

$$\boxed{\lambda_2 = -20 \ [1/sec]}$$

SINCE WE HAVE BOTH λ_1 & λ_2, WE CAN FIND K_1 & K_2 UTILIZING A SYSTEM OF EQUATIONS OF THE CHARACTERISTIC EQUATION FOR BOTH λ_1 & λ_2

$I_x = 6.8E\text{-}5 \ Kgm^2$

$$\begin{cases} \lambda_1^2 + \lambda\left(\frac{K_1}{I_x}\right) + \frac{K_2}{I_x} = 0 \\ \lambda_2^2 + \lambda_2\left(\frac{K_1}{I_x}\right) + \frac{K_2}{I_x} = 0 \end{cases} \Rightarrow \begin{cases} 4 - \frac{2K_1}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0 \\ 400 - \frac{20K_1}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0 \end{cases}$$

$$\Rightarrow \begin{cases} K_1 = -\frac{6.8E\text{-}5}{2}\left(-4 - \frac{K_2}{6.8E\text{-}5}\right) \\ 400 - \frac{20K_1}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0 \end{cases} \Rightarrow \begin{cases} K_1 = 1.36E\text{-}4 + \frac{K_2}{2} \\ 400 - \frac{20K_1}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0 \end{cases}$$

$$\Rightarrow 400 - 20\left[1.36E\text{-}4 + \frac{K_2}{2}\right] + K_2 = 0$$

$$\Rightarrow \quad 400 - \frac{20\left[1.36E\text{-}4 + \frac{K_2}{2}\right]}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0$$

$$\Rightarrow \quad 400 - \frac{0.00544 - 20K_2}{6.8E\text{-}5} + \frac{K_2}{6.8E\text{-}5} = 0 \quad \Rightarrow \quad 360 - \frac{9K_2}{6.8E\text{-}5} = 0$$

$$\Rightarrow K_2 = 0.00272$$

$$\Rightarrow K_1 = 1.36E\text{-}4 + \frac{(0.00272)}{2} = 0.001496$$

$$\Rightarrow \boxed{\begin{aligned} K_1 &= 0.001496 \\ K_2 &= 0.00272 \end{aligned}}$$

---

§ LONGITUDINAL CONTROL

SIMILAR TO LATERAL CONTROL, THE EQUATIONS OF MOTION FOR A CLOSED LOOP ARE

$$\begin{bmatrix} \Delta\dot{q} \\ \Delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{k_3}{I_y} & -\frac{K_4}{I_y} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta\theta \end{bmatrix}$$

$$\dot{X} = A \cdot X$$

LET $AX = \lambda \cdot X$, WHERE λ IS A SCALAR

THEREFORE $(A - \lambda I)X = 0$

(I: IDENTITY MATRIX)

SINCE WE KNOW $X \neq 0$, AS WE HAVE REAL Δq & $\Delta\theta$ VALUES, $\det(A - \lambda I) = 0$ IS TRUE, THEREFORE:

$$\det\begin{pmatrix} \frac{-K_3}{I_y} - \lambda & -\frac{K_4}{I_y} \\ 1 & -\lambda \end{pmatrix} = \left(-\frac{K_3}{I_y} - \lambda\right)(-\lambda) - \left(-\frac{K_4}{I_y}\right)(1) = 0$$

$$\Rightarrow \lambda^2 + \lambda\left(\frac{K_3}{I_y}\right) + \frac{K_4}{I_y} = 0$$

- THIS IS THE CHARACTERISTIC EQUATION

AS WITH THE LATERAL CONTROL, $\lambda_1 = -2$ & $\lambda_2 = -20$, AS DERRIVED ABOVE. THIS IS TRUE, AS ONLY ONE TIME CONSTANT WAS GIVEN IN THE ASSIGNMENT.

ALSO AS IN THE LATERAL CONTROL, SINCE WE HAVE BOTH λ_1 & λ_2, WE CAN FIND K_1 & K_2 UTILIZING A SYSTEM OF EQUATIONS OF THE CHARACTERISTIC EQUATION FOR BOTH λ_1 & λ_2

$I_y = 9.2\text{E-}5\ Kgm^2$

$$\begin{cases} \lambda_1^2 + \lambda\left(\frac{K_3}{I_y}\right) + \frac{K_4}{I_y} = 0 \\ \lambda_2^2 + \lambda_2\left(\frac{K_3}{I_y}\right) + \frac{K_4}{I_y} = 0 \end{cases} \Rightarrow \begin{cases} 4 - \frac{2K_3}{9.2\,\text{E-}5} + \frac{K_4}{9.2\,\text{E-}5} = 0 \\ 400 - \frac{20K_3}{9.2\,\text{E-}5} + \frac{K_4}{9.2\text{E-}5} = 0 \end{cases}$$

$$\Rightarrow \begin{cases} K_3 = -\frac{9.2\,\text{E-}5}{2}\left(-4 - \frac{K_4}{9.2\text{E-}5}\right) \\ 400 - \frac{20K_3}{9.2\,\text{E-}5} + \frac{K_4}{9.2\text{E-}5} = 0 \end{cases} \Rightarrow \begin{cases} K_3 = 1.84\,\text{E-}4 + \frac{K_4}{2} \\ 400 - \frac{20K_3}{9.2\,\text{E-}5} + \frac{K_4}{9.2\text{E-}5} = 0 \end{cases}$$

$$\Rightarrow 400 - \frac{20\left[1.84\text{E-}4 + \frac{K_4}{2}\right]}{9.2\text{E-}5} + \frac{K_4}{9.2\,\text{E-}5} = 0$$

$$\frac{L \qquad ^{2}\big]}{9.2E-5} \qquad \frac{}{9.2}\,E-5$$

$$\Rightarrow 400 - \frac{\frac{0.00736 - 20K_4}{2}}{9.2E-5} + \frac{K_4}{9.2E-5} = 0 \quad \Rightarrow 360 - \frac{9K_4}{9.2E-5} = 0$$

$$\Rightarrow K_4 = 0.00368$$

$$\Rightarrow K_3 = 1.84E-4 + \frac{(0.00368)}{2} = 0.002024$$

$$\Rightarrow \boxed{\begin{aligned} K_3 &= 0.002024 \\ K_4 &= 0.00368 \end{aligned}}$$

$K_1, K_2, K_3, \& K_4$ ARE NOW USED IN MATLAB SIMULATION

III. Question 2

Responses were found for these calculated gain values by implementing a closed loop linearized system model to see the deviations in, u, v, w, p, q, r, ϕ, and θ while deviating input conditions by +5° bank, +5° elevation, +0.1 radian/sec in roll rate (p), and +0.1 radian/sec in pitch rate (q).

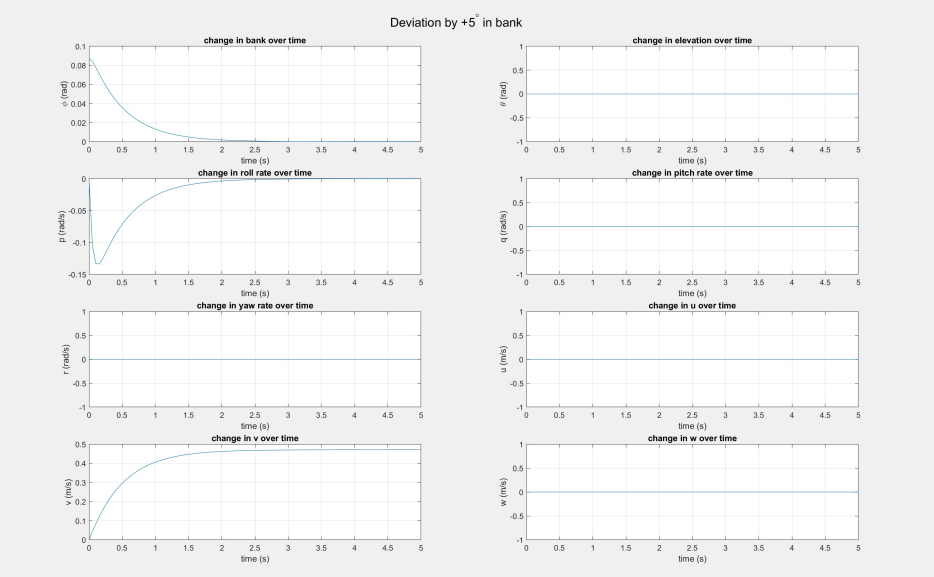


Fig. 1 Change in initial Bank, Problem 2

as seen by figure 1, the values for u, w, q, r, and θ are zero. With the initial bank angle deviation, it is seen that the feedback control implemented in the simulation brings the bank angle back to zero radians quickly, and the roll rate deviates from zero radians/sec in the negative direction quickly to initialize bringing the quadcopter back to a zero radian bank angle, then tapers back to zero radians/sec as the quadcopter recovers. v increases in a logarithmic fashion to a steady value. This is due to the quadcopter accelerating in the positive $\hat{y}$ direction initially due to the bank angle deviation, and the feedback control taking over to bring the acceleration down. The quadcopter will still have a $\hat{y}$ direction velocity after the bank angle is brought back to zero radians, as there is no altitude control implemented for this simulation.

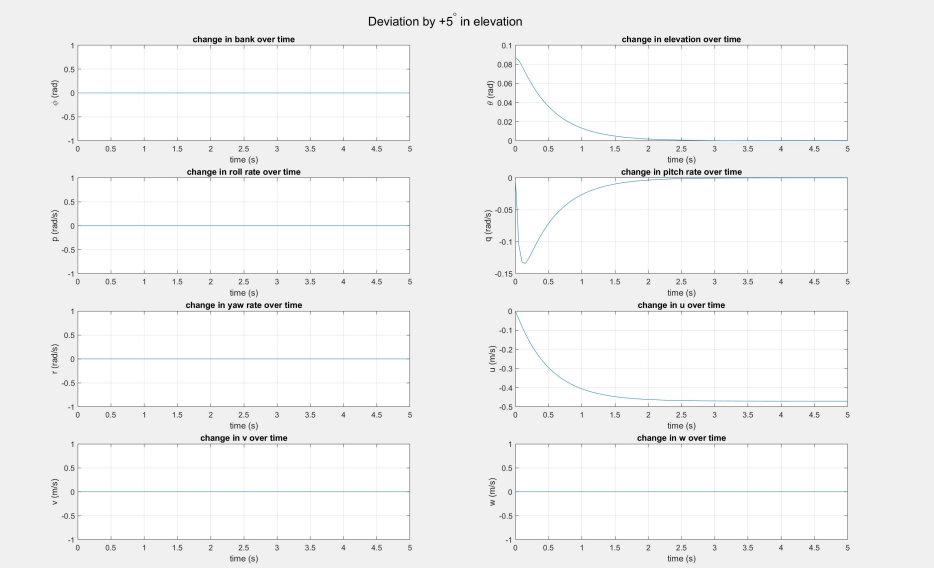


Fig. 2 Change in initial Elevation, Problem 2

as seen by figure 2, the values for v, w, p, r, ϕ, and are zero. With the initial elevation angle deviation, it is seen that the feedback control implemented in the simulation brings the elevation angle back to zero radians quickly, and the pitch rate deviates from zero radians/sec in the negative direction quickly to initialize bringing the quadcopter back to a zero radian elevation angle, then tapers back to zero radians/sec as the quadcopter recovers. u decreases in a logarithmic fashion to a steady value. This is due to the quadcopter accelerating in the negative $\hat{x}$ direction initially due to the positive bank angle deviation, and the feedback control taking over to bring the acceleration down. The quadcopter will still have a negative $\hat{x}$ direction velocity after the elevation angle is brought back to zero radians, as the initial acceleration brought the quadcopter to a new velocity. This occurs because there was no altitude control implimented into the feedback system.

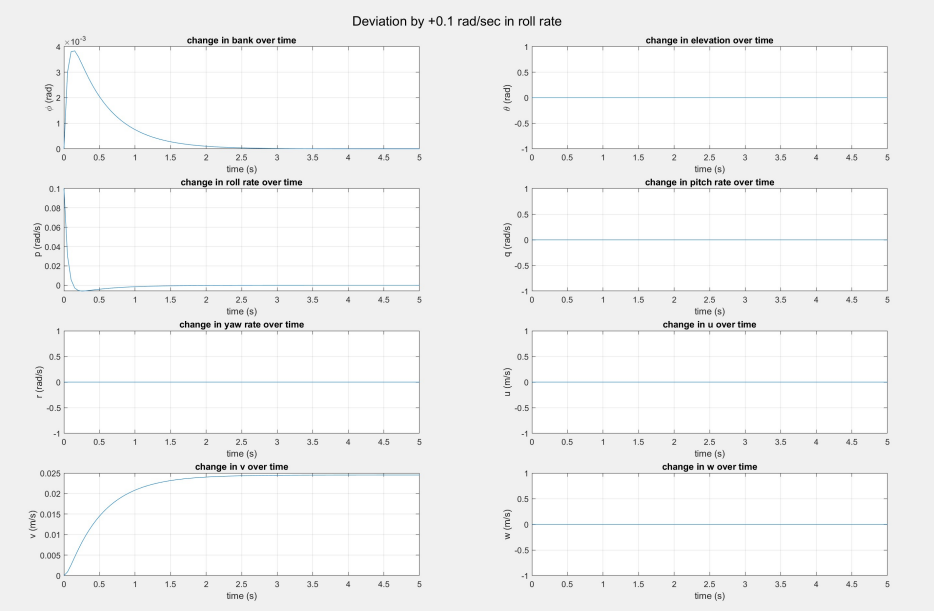


Fig. 3 Change in Roll Rate, Problem 2

as seen by figure 3, the values for u, w, q, r, and θ are zero. Due to the initial roll rate disturbance, the bank angle

quickly increases due to the disturbance, then begins to return to zero radians as the feedback control system corrects the disturbance. The roll rate of the quadcopter is initially 0.1 radians/sec, but as the feedback control activates, the rollrate quickly drops back to zero radian/sec with minimal overshoot. v increases in a logarithmic fashion to a steady value. This is due to the quadcopter accellerating in the positive $\hat{y}$ direction initially due to the bank angle deviation, and the feedback control taking over to bring the acceleration down. The quadcopter will still have a $\hat{y}$ direction velocity after the bank angle is brought back to zero radians due to the lack of altitude feedback control.

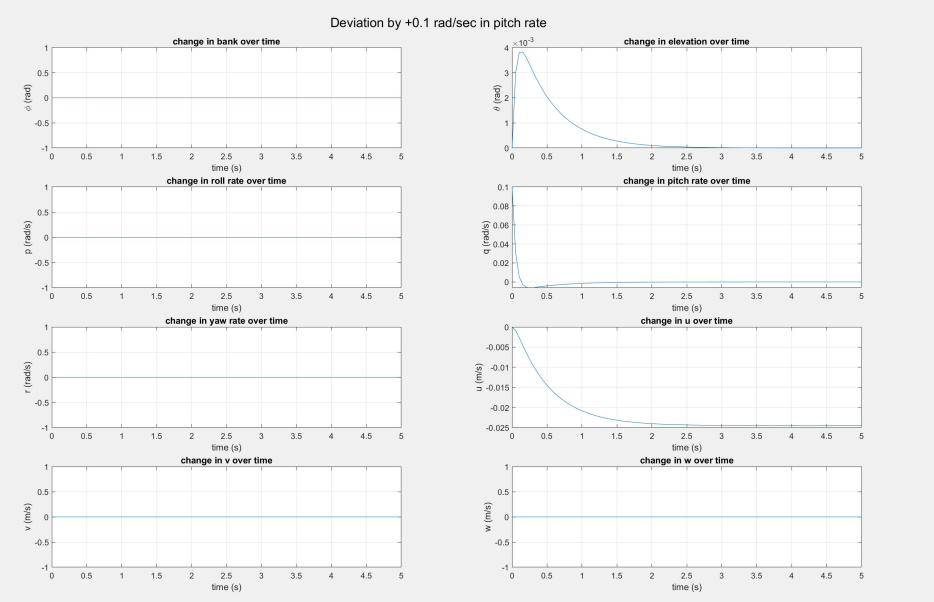


Fig. 4 Change in Pitch Rate, Problem 2

as seen by figure 4, the values for v, w, p, r, and ϕ are zero. Due to the initial pitch rate disturbance, the elevation angle quickly increases due to the disturbance, then begins to return to zero radians as the feedback control system corrects the disturbance. the pitch rate of the quadcopter is initially 0.1 radians/sec, but as the feedback control activates, the pitch rate quickly drops back to zero radians/sec with minimal overshoot. u decreases in a logarithmic fashion to a steady value. This is due to the quadcopter accelerating in the negative $\hat{x}$ direction initially due to the positive elevation angle deviation, and the feedback control taking over to bring the acceleration down. The quadcopter will still have a negative $\hat{x}$ direction velocity after the elevation angle is brought back to zero radians, as the initial acceleration brought the quadcopter to a new velocity. This occurs because there was no altitude control implimented into the feedback system.

Due to the response of the model in these disturbance conditions, the behavior of the model corresponds to the expected behavior from the linearized modal response theory, as the eigenvalues utilized to get the feedback gains were negative, which resulted in a return to system equilibrium, which is an expected characteristic of negative eigenvalues from the linearized modal response theory. Although the quadcopter does return to zero radians for both bank angle and elevation angle for all the tested disturbances, the quadcopter is still changing altitude due to body velocities present after the feedback control acts to return the system to a steady state, therefore steady hover is still not a stable flight condition.

IV. Question 3

Using the non-linearized model built in Assignment 2, a closed loop non-linearized system model was developed to simulate the Quadcopter's response to initial condition deviations, and compare results to that of the closed loop linearized system model. Plots were generated for both the nonlinear model and linear model to see the deviations in, u, v, w, p, q, r, ϕ, and θ while deviating input conditions by +5° bank, +5° pitch, +0.1 radian/sec in roll rate (p),and +0.1 radian/sec in pitch rate (q).

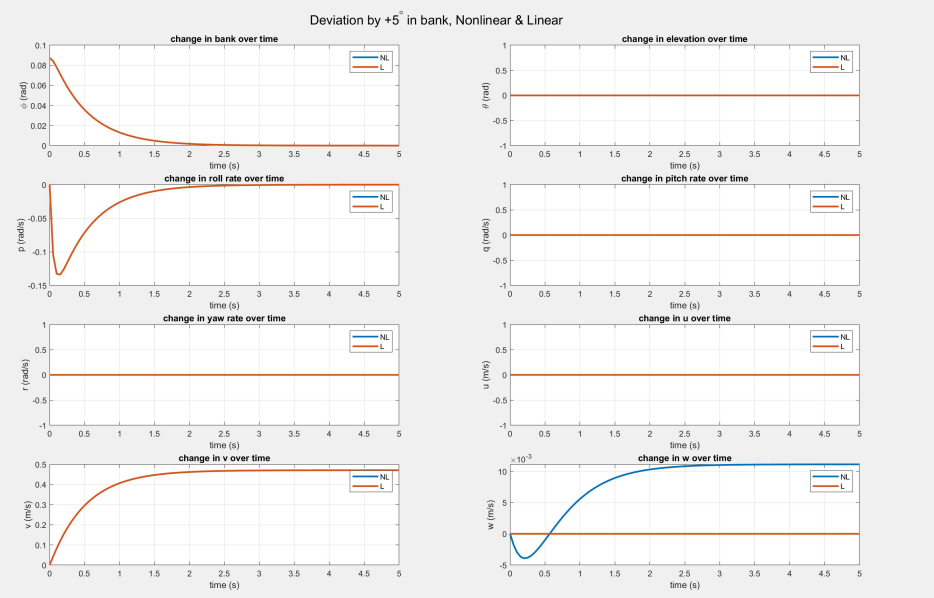


Fig. 5 Change in initial Bank, Problem 3

The key difference between the linear and non linear models for the closed loop feedback control was found in the $\hat{z}$ component of body frame velocity. As seen from the plots for each of the deviations, the non-linear model's $\hat{z}$ component of body frame velocity initially does down, as the disturbance from trim will make the copter translate down initally, then, as the feedback control takes over, the velocity increases logarithmically until the bank angle and elevation angle are zero radians. After this point, the velocity in the $\hat{z}$ direction is essentially constant, but still greater than zero. This shows that the quadcopter still has a $\hat{z}$ direction velocity component after the feedback control has brought the bank and elevation angles to zero, as there is no altitude control implimented into the model.

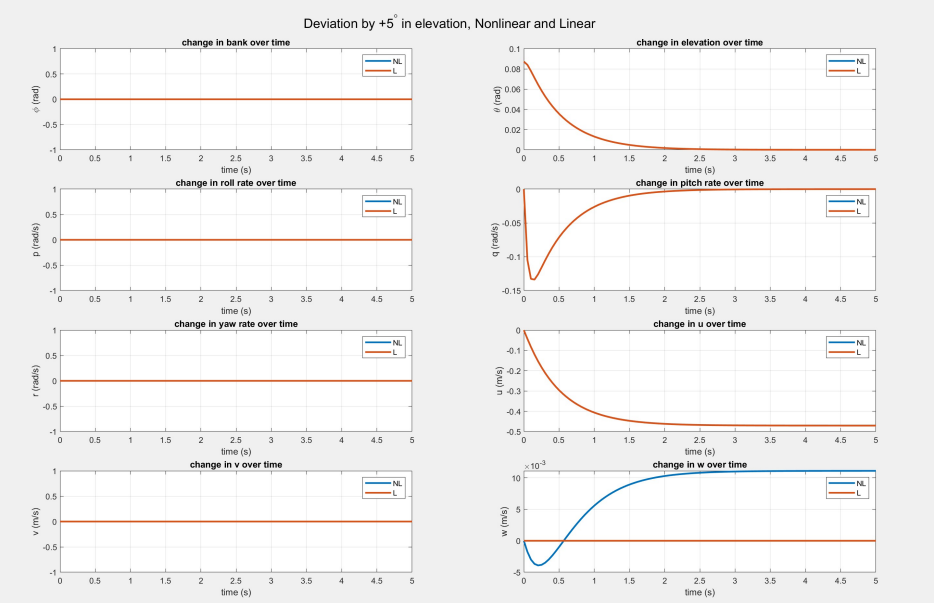


Fig. 6 Change in initial Elevation, Problem 3

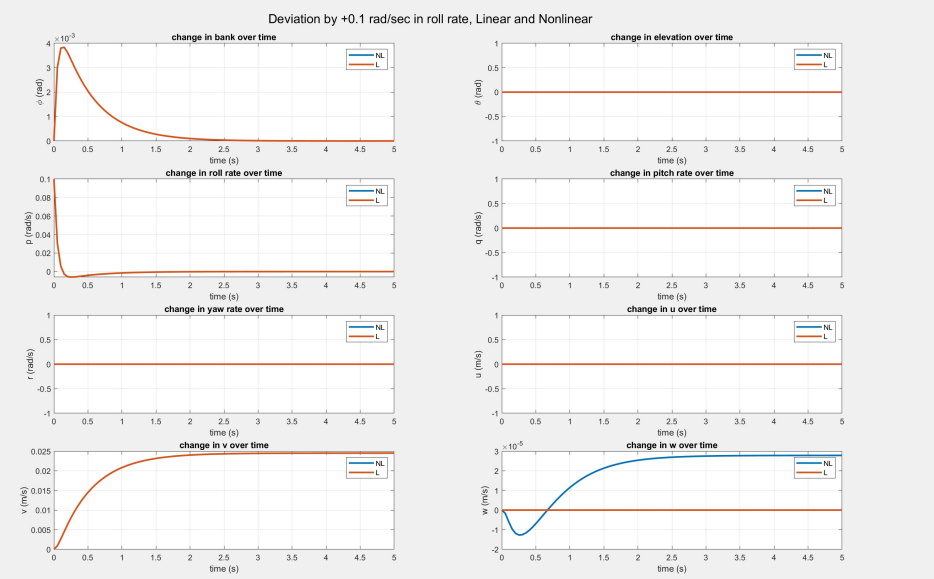


Fig. 7 Change in Roll Rate, Problem 3

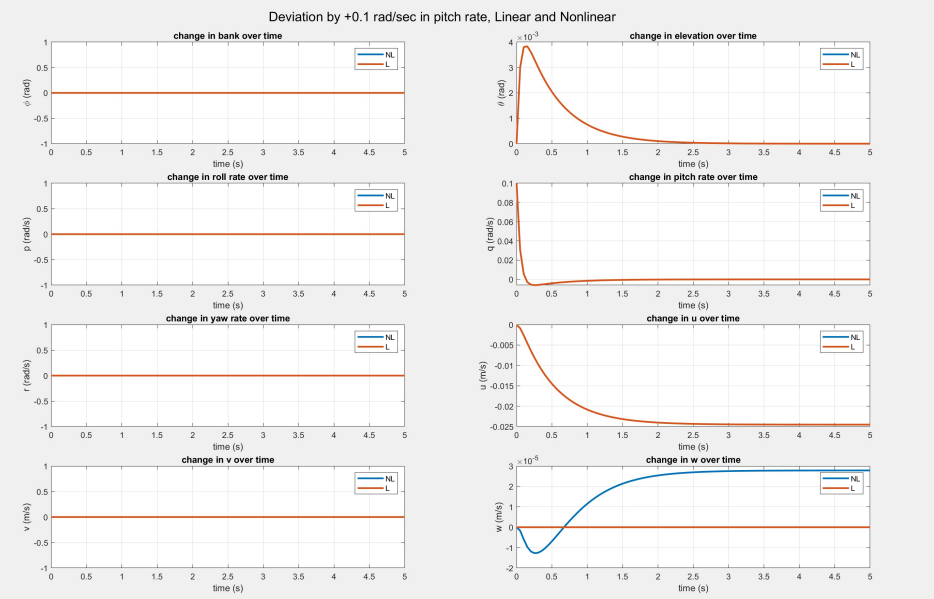


Fig. 8 Change in Pitch Rate, Problem 3

V. Question 4

Implimenting the gain values developed for the first part of this assignment to the Rolling Spider quad copter, experimental data was collected and compared to the linearized control feedback system results. The following picture shows the GUI input to the experimental quadcopter.

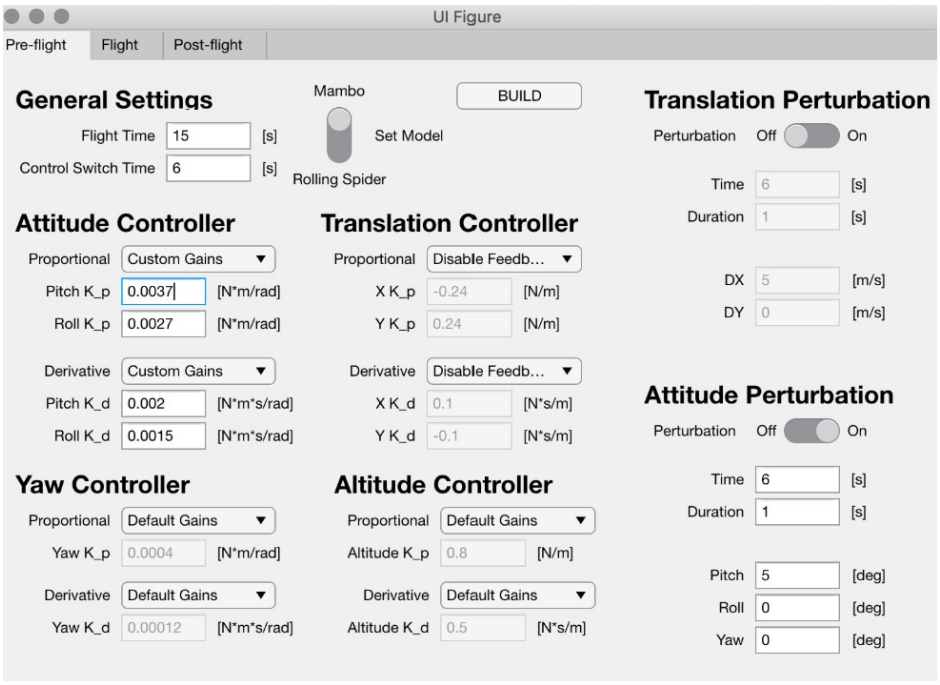


Fig. 9 GUI Screenshot

As seen from this picture, the gain values used were the same as derrived in Question 1 of this assignment ($k_1 = 0.0015$, $k_2 = 0.0027$, $k_3 = 0.0020$, and $k_4 = 0.0037$). The deviation used for this test was 5° elevation angle, as seen from the picture.

The data from the flight was then processed and plotted with the linearized control feedback system modeled results.

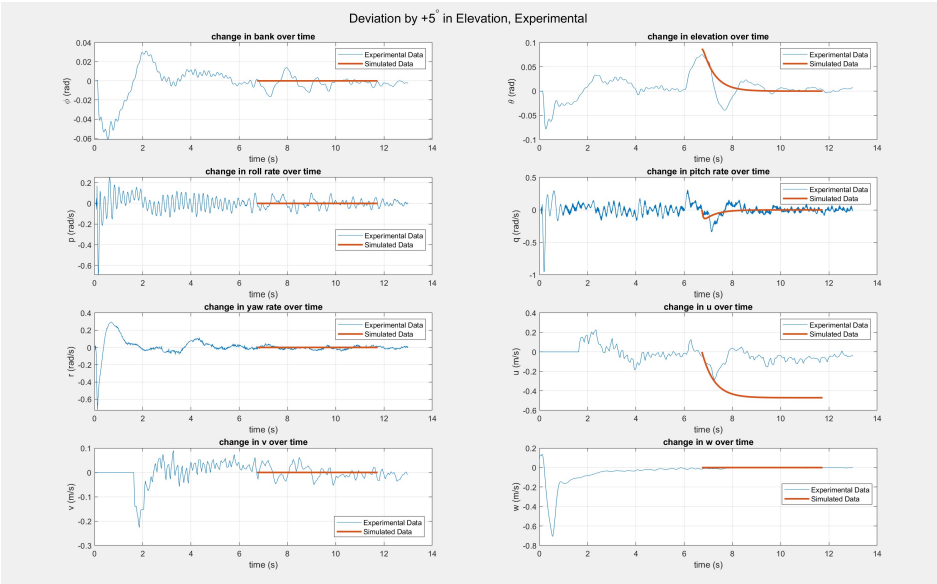


Fig. 10 Experimental data, 5° initial elevation

As seen from this figure, the experimental data matches with the experimental data roughly, but do to the broad assumptions of the linearized model, there are deviations in plots, as to be expected when comparing simulated data to experimental data.

As seen from the experimental data after 6 seconds, the values for v, w, p, r, ϕ, and are roughly zero. With the initial elevation angle deviation, It is seen that the feedback control implemented in the experimental trial brings the elevation angle back to zero radians quickly, and the pitch rate deviates from zero radians/sec in the negative direction quickly to initialize bringing the quadcopter back to a zero radian elevation angle, then tapers back to zero as the quadcopter recovers, matching what is simulated in the linearized model. u decreases initially in the experimental data, but then returns to around zero m/s after 2 seconds, this experimental data is the largest deviation from the linearized data model, as the simulated data asymptotically approaches a negative u velocity rather than returning to zero. This is due to lack of some feedback control in the linear model which would keep the quadcopter from changing altitude. Since the linearized feedback control system developed for this assignment didn't include these control sequences, the drone will keep a constant negative $\hat{x}$ velocity in the simulation, while in the experiment there was additional control keeping the drone from changing altitude, as seen from the GUI image. This additional control factor implimented into the experimental trial adjusted the u value in order to keep the quadcopter at a constant altitude, causing the discrepancy seen between the experimental data and the simulated data.

VI. MATLAB Code

Question 1 Script

```
% This script finds the values for derrivative gain controlling both
% the longitudional and lateral rates of the drone
%
%
%   Author: Benjiman Smith
%   Collaborators: E. Owen, I. Quezada
%   Date: 1/25/2020
%
%% lateral control
clear all
close all
Lambda1 = -2; % 1st lambda value, found from lab calculations
Lambda2 = -12;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)

syms K1 K2 % symbollically solving for K1 and K2
eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution


[A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
    into independent vectors

Solution = linsolve(A,B); % solve the system of equations
Solution = double(Solution); % convert symbolic solution vector to vector of doubles
K1 = Solution(1); % bank control
K2 = Solution(2); % bank control

%% Longitudional Control
syms K3 K4 % symbollically solving for K3 and K4
eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution


[C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);

ForceVect2 = linsolve(C,D);
ForceVect2 = double(ForceVect2);
K3 = ForceVect2(1); % elevation control
K4 = ForceVect2(2); % elevation control
```

Question 2 Script

```
% This script initializes initial conditions for the trajectory of a drone
% in steady flight conditions (hovering) then calls the ODE45 function to
% numerically integrate the position of the drone with respect to the
% initial conditions, along with plotting the results answering Question
% 2 of Assignment 4
%
%   Author: Benjiman Smith
%   Collaborators: E. Owen, I. Quezada
%   Date: 2/20/2020
%
clc;
clear all;
close all;
m = 0.068; % mass of the drone [kg]
r = 0.06;  % body to motor distance [m]
k = 0.0024;  % [Nm/N]
rad = r/sqrt(2);  % [m]
g = 9.81; % gravity [m/s^2]
alpha = 2e-6;   % [N/(m/s)^2]
```

```
eta = 1e-3;    % [N/(rad/s)^2]
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5;   % moment of inertia in the y direction [kg m^2]
Iz = 1.35e-4;  % moment of inertia in the z direction [kg m^2]
%% lateral control
Lambda1 = -2; % 1st lambda value, found from lab calculations
Lambda2 = -20;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)

syms K1 K2 % symbollically solving for K1 and K2
eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution


[A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
    into independent vectors

Solution = linsolve(A,B); % solve the system of equations
Solution = double(Solution); % convert symbolic solution vector to vector of doubles
K1 = Solution(1); % bank control
K2 = Solution(2); % bank control

%% Longitudional Control
syms K3 K4 % symbollically solving for K3 and K4
eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution


[C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);

ForceVect2 = linsolve(C,D);
ForceVect2 = double(ForceVect2);
K3 = ForceVect2(1); % elevation control
K4 = ForceVect2(2); % elevation control

givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
F = m*g;

%% bank is 5
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(7) = deg2rad(5); % bank angle is 5 degrees
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;
options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
[t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % ode call for linearized feedback control ode

%% plotting

figure()


sgtitle('Deviation by +5^{\circ} in bank');
 subplot(4,2,1);
plot(t1, X(:,7));

grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')
```

```
85
86  hold off
87  %
88   subplot(4,2,2);
89  plot(t1, X(:,8));
90
91  grid on
92  xlabel('time (s)')
93  ylabel('\theta (rad)')
94  title('change in elevation over time')
95  hold off
96  %
97   subplot(4,2,3);
98  plot(t1, X(:,4));
99
100 grid on
101 xlabel('time (s)')
102 ylabel('p (rad/s)')
103 title('change in roll rate over time')
104
105 hold off
106 %
107  subplot(4,2,4);
108 plot(t1, X(:,5));
109
110 grid on
111 xlabel('time (s)')
112 ylabel('q (rad/s)')
113 title('change in pitch rate over time')
114
115 hold off
116 %
117  subplot(4,2,5);
118 plot(t1, X(:,6));
119
120 grid on
121 xlabel('time (s)')
122 ylabel('r (rad/s)')
123 title('change in yaw rate over time')
124
125 hold off
126 %
127  subplot(4,2,6);
128 plot(t1, X(:,1));
129
130 grid on
131 xlabel('time (s)')
132 ylabel('u (m/s)')
133 title('change in u over time')
134
135 hold off
136 %
137  subplot(4,2,7);
138 plot(t1, X(:,2));
139
140 grid on
141 xlabel('time (s)')
142 ylabel('v (m/s)')
143 title('change in v over time')
144
145 hold off
146 %
147  subplot(4,2,8);
148 plot(t1, X(:,3));
149
150 grid on
151 xlabel('time (s)')
152 ylabel('w (m/s)')
```

```
153 title('change in w over time')
154
155 hold off
156 %% elevation is 5
157 tspan = linspace(0,5); % time vector
158 Pertubations = zeros(1, 3); % No perturbation
159 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
160 conditions = zeros(1, 12); % initialize conditions vector (very large)
161 conditions(8) = deg2rad(5); % elevation of 5 degrees
162 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
163 t1 = 0;
164 t2 = 0;
165 X = 0;
166 X2 = 0;
167 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
        tolerance of 1e-8 is met
168 [t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
        options); % ode call for linearized feedback control ode
169
170
171 %% plotting
172 figure()
173 sgtitle('Deviation by +5^{\circ} in elevation'); subplot(4,2,1);
174 plot(t1, X(:,7));
175
176 grid on
177 xlabel('time (s)')
178 ylabel('\phi (rad)')
179 title('change in bank over time')
180
181 hold off
182 %
183  subplot(4,2,2);
184 plot(t1, X(:,8));
185
186 grid on
187 xlabel('time (s)')
188 ylabel('\theta (rad)')
189 title('change in elevation over time')
190
191 hold off
192 %
193  subplot(4,2,3);
194 plot(t1, X(:,4));
195
196 grid on
197 xlabel('time (s)')
198 ylabel('p (rad/s)')
199 title('change in roll rate over time')
200
201 hold off
202 %
203  subplot(4,2,4);
204 plot(t1, X(:,5));
205
206 grid on
207 xlabel('time (s)')
208 ylabel('q (rad/s)')
209 title('change in pitch rate over time')
210
211 hold off
212 %
213  subplot(4,2,5);
214 plot(t1, X(:,6));
215
216 grid on
217 xlabel('time (s)')
218 ylabel('r (rad/s)')
```

```
title('change in yaw rate over time')

hold off
%
 subplot(4,2,6);
plot(t1, X(:,1));

grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')

hold off
%
 subplot(4,2,7);
plot(t1, X(:,2));

grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')

hold off
%
 subplot(4,2,8);
plot(t1, X(:,3));

grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')

hold off

%% azimuth is 5
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(9) = deg2rad(5); % azimuth angle is 5
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;
options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
[t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % ode call for linearized feedback control ode

figure()
sgtitle('Deviation by +5^{\circ} in azimuth'); subplot(4,2,1);
plot(t1, X(:,7));

grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')

hold off
%
 subplot(4,2,2);
plot(t1, X(:,8));

grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')
```

```
285
286 hold off
287 %
288  subplot(4,2,3);
289 plot(t1, X(:,4));
290
291 grid on
292 xlabel('time (s)')
293 ylabel('p (rad/s)')
294 title('change in roll rate over time')
295
296 hold off
297 %
298  subplot(4,2,4);
299 plot(t1, X(:,5));
300
301 grid on
302 xlabel('time (s)')
303 ylabel('q (rad/s)')
304 title('change in pitch rate over time')
305
306 hold off
307 %
308  subplot(4,2,5);
309 plot(t1, X(:,6));
310
311 grid on
312 xlabel('time (s)')
313 ylabel('r (rad/s)')
314 title('change in yaw rate over time')
315
316 hold off
317 %
318  subplot(4,2,6);
319 plot(t1, X(:,1));
320
321 grid on
322 xlabel('time (s)')
323 ylabel('u (m/s)')
324 title('change in u over time')
325
326 hold off
327 %
328  subplot(4,2,7);
329 plot(t1, X(:,2));
330
331 grid on
332 xlabel('time (s)')
333 ylabel('v (m/s)')
334 title('change in v over time')
335
336 hold off
337 %
338  subplot(4,2,8);
339 plot(t1, X(:,3));
340
341 grid on
342 xlabel('time (s)')
343 ylabel('w (m/s)')
344 title('change in w over time')
345
346 hold off
347 %% 0.1 roll rate
348 tspan = linspace(0,5); % time vector
349 Pertubations = zeros(1, 3); % No perturbation
350 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
351 conditions = zeros(1, 12); % initialize conditions vector (very large)
352 conditions(4) = 0.1; % roll rate is 0.1 rad/s
```

```
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;
options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
[t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options);


figure()
sgtitle('Deviation by +0.1 rad/sec in roll rate');
 subplot(4,2,1);
plot(t1, X(:,7));

grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')

hold off
%
 subplot(4,2,2);
plot(t1, X(:,8));

grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')

hold off
%
 subplot(4,2,3);
plot(t1, X(:,4));

grid on
xlabel('time (s)')
ylabel('p (rad/s)')
title('change in roll rate over time')
hold off
%
 subplot(4,2,4);
plot(t1, X(:,5));

grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')

hold off
%
 subplot(4,2,5);
plot(t1, X(:,6));

grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')

hold off
%
 subplot(4,2,6);
plot(t1, X(:,1));

grid on
xlabel('time (s)')
ylabel('u (m/s)')
```

```
title('change in u over time')

hold off
%
 subplot(4,2,7);
plot(t1, X(:,2));

grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')

hold off
%
 subplot(4,2,8);
plot(t1, X(:,3));

grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')

hold off

%% .01 pitch rate
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(5) = 0.1; % pitch rate is 0.1 rad/s
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;
options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
[t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options);

%% plotting
figure ()
sgtitle('Deviation by +0.1 rad/sec in pitch rate');
 subplot(4,2,1);
plot(t1, X(:,7));

grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')

hold off
%
 subplot(4,2,2);
plot(t1, X(:,8));

grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')

hold off
%
 subplot(4,2,3);
plot(t1, X(:,4));

grid on
xlabel('time (s)')
```

```
ylabel('p (rad/s)')
title('change in roll rate over time')

hold off
%
 subplot(4,2,4);
plot(t1, X(:,5));

grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')

hold off
%
 subplot(4,2,5);
plot(t1, X(:,6));

grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')

hold off
%
 subplot(4,2,6);
plot(t1, X(:,1));

grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
hold off
%
 subplot(4,2,7);
plot(t1, X(:,2));

grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
hold off
%
 subplot(4,2,8);
plot(t1, X(:,3));

grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
hold off

%% .01 yaw rate
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(6) = 0.1; % yaw rate is 0.1 rad/s
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;
options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
[t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options);

```

```
551
552 figure()
553 sgtitle('Deviation by +0.1 rad/sec in yaw rate ');
554  subplot(4,2,1);
555 plot(t1, X(:,7));
556
557 grid on
558 xlabel('time (s)')
559 ylabel('\phi (rad)')
560 title('change in bank over time')
561
562 hold off
563 %
564  subplot(4,2,2);
565 plot(t1, X(:,8));
566
567 grid on
568 xlabel('time (s)')
569 ylabel('\theta (rad)')
570 title('change in elevation over time')
571
572 hold off
573 %
574  subplot(4,2,3);
575 plot(t1, X(:,4));
576
577 grid on
578 xlabel('time (s)')
579 ylabel('p (rad/s)')
580 title('change in roll rate over time')
581
582 hold off
583 %
584  subplot(4,2,4);
585 plot(t1, X(:,5));
586
587 grid on
588 xlabel('time (s)')
589 ylabel('q (rad/s)')
590 title('change in pitch rate over time')
591
592 hold off
593 %
594  subplot(4,2,5);
595 plot(t1, X(:,6));
596
597 grid on
598 xlabel('time (s)')
599 ylabel('r (rad/s)')
600 title('change in yaw rate over time')
601
602 hold off
603 %
604  subplot(4,2,6);
605 plot(t1, X(:,1));
606
607 grid on
608 xlabel('time (s)')
609 ylabel('u (m/s)')
610 title('change in u over time')
611 hold off
612 %
613  subplot(4,2,7);
614 plot(t1, X(:,2));
615
616 grid on
617 xlabel('time (s)')
618 ylabel('v (m/s)')
```

```
title('change in v over time')
hold off
%
 subplot(4,2,8);
plot(t1, X(:,3));

grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
hold off
```

Question 3 Script

```
% This script initializes initial conditions for the trajectory of a drone
% in steady flight conditions (hovering) then calls the ODE45 function to
% numerically integrate the position of the drone with respect to the
% initial conditions, along with plotting the results answering Question
% 3 of Assignment 4
%
%   Author: Benjiman Smith
%   Collaborators: E. Owen, I. Quezada
%   Date: 2/2/2020
%
clc;
clear all;
close all;
m = 0.068; % mass of the drone [kg]
r = 0.06;  % body to motor distance [m]
k = 0.0024;  % [Nm/N]
rad = r/sqrt(2);  % [m]
g = 9.81; % gravity [m/s^2]
alpha = 2e-6;   % [N/(m/s)^2]
eta = 1e-3;    % [N/(rad/s)^2]
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5;   % moment of inertia in the y direction [kg m^2]
Iz = 1.35e-4;  % moment of inertia in the z direction [kg m^2]

%% lateral control
Lambda1 = -2; % 1st lambda value, found from lab calculations
Lambda2 = -20;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)

syms K1 K2 % symbollically solving for K1 and K2
eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution


[A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
    into independent vectors

Solution = linsolve(A,B); % solve the system of equations
Solution = double(Solution); % convert symbolic solution vector to vector of doubles
K1 = Solution(1); % bank control
K2 = Solution(2); % bank control

%% Longitudional Control
syms K3 K4 % symbollically solving for K3 and K4
eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution


[C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);

ForceVect2 = linsolve(C,D);
ForceVect2 = double(ForceVect2);
```

```
K3 = ForceVect2(1); % elevation control
K4 = ForceVect2(2); % elevation control

givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
F = m*g;

%% bank is 5
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(7) = deg2rad(5); % bank angle is 5 deg
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
% initialize variables
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % nonlinear ODE

[t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE

%sublotting of all 8 variables
figure()

sgtitle('Deviation by +5^{\circ} in bank, Nonlinear & Linear');
subplot(4,2,1);
plot(t1, X(:,7),'linewidth', 2);
hold on
plot(t2, X2(:,7),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,2);
plot(t1, X(:,8),'linewidth', 2);
hold on
plot(t2, X2(:,8),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,3);
plot(t1, X(:,4),'linewidth', 2);
hold on
plot(t2, X2(:,4),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('p (rad/s)')
title('change in roll rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,4);
plot(t1, X(:,5),'linewidth', 2);
hold on
```

```
plot(t2, X2(:,5),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,5);
plot(t1, X(:,6),'linewidth', 2);
hold on
plot(t2, X2(:,6),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,6);
plot(t1, X(:,1),'linewidth', 2);
hold on
plot(t2, X2(:,1),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,7);
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
hold off
%% elevation is 5
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(8) = deg2rad(5);
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
% initialize variables
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
```

```
    options); % nonlinear ODE
185
186 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE
187
188 figure()
189 sgtitle('Deviation by +5^{\circ} in elevation, Nonlinear and Linear');
190 subplot(4,2,1);
191 plot(t1, X(:,7),'linewidth', 2);
192 hold on
193 plot(t2, X2(:,7),'linewidth', 2);
194 grid on
195 xlabel('time (s)')
196 ylabel('\phi (rad)')
197 title('change in bank over time')
198 legend( 'NL', 'L');
199 hold off
200 %
201 subplot(4,2,2);
202 plot(t1, X(:,8),'linewidth', 2);
203 hold on
204 plot(t2, X2(:,8),'linewidth', 2);
205 grid on
206 xlabel('time (s)')
207 ylabel('\theta (rad)')
208 title('change in elevation over time')
209 legend( 'NL', 'L');
210 hold off
211 %
212 subplot(4,2,3);
213 plot(t1, X(:,4),'linewidth', 2);
214 hold on
215 plot(t2, X2(:,4),'linewidth', 2);
216 grid on
217 xlabel('time (s)')
218 ylabel('p (rad/s)')
219 title('change in roll rate over time')
220 legend( 'NL', 'L');
221 hold off
222 %
223 subplot(4,2,4);
224 plot(t1, X(:,5),'linewidth', 2);
225 hold on
226 plot(t2, X2(:,5),'linewidth', 2);
227 grid on
228 xlabel('time (s)')
229 ylabel('q (rad/s)')
230 title('change in pitch rate over time')
231 legend( 'NL', 'L');
232 hold off
233 %
234 subplot(4,2,5);
235 plot(t1, X(:,6),'linewidth', 2);
236 hold on
237 plot(t2, X2(:,6),'linewidth', 2);
238 grid on
239 xlabel('time (s)')
240 ylabel('r (rad/s)')
241 title('change in yaw rate over time')
242 legend( 'NL', 'L');
243 hold off
244 %
245 subplot(4,2,6);
246 plot(t1, X(:,1),'linewidth', 2);
247 hold on
248 plot(t2, X2(:,1),'linewidth', 2);
249 grid on
250 xlabel('time (s)')
```

```
ylabel('u (m/s)')
title('change in u over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,7);
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
hold off

%% azimuth is 5
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(9) = deg2rad(5);
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
% initialize variables
t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % nonlinear ODE

[t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE

figure()
sgtitle('Deviation by +5^{\circ} in azimuth, Nonlinear and Linear'); subplot(4,2,1);
plot(t1, X(:,7),'linewidth', 2);
hold on
plot(t2, X2(:,7),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,2);
plot(t1, X(:,8),'linewidth', 2);
hold on
plot(t2, X2(:,8),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\theta (rad)')
```

```
title('change in elevation over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,3);
plot(t1, X(:,4),'linewidth', 2);
hold on
plot(t2, X2(:,4),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('p (rad/s)')
title('change in roll rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,4);
plot(t1, X(:,5),'linewidth', 2);
hold on
plot(t2, X2(:,5),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,5);
plot(t1, X(:,6),'linewidth', 2);
hold on
plot(t2, X2(:,6),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,6);
plot(t1, X(:,1),'linewidth', 2);
hold on
plot(t2, X2(:,1),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,7);
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
```

```
hold off
%% .01 roll rate
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(4) = 0.1;
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting

t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % nonlinear ODE

[t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE

figure()
sgtitle('Deviation by +0.1 rad/sec in roll rate, Linear and Nonlinear');
subplot(4,2,1);
plot(t1, X(:,7),'linewidth', 2);
hold on
plot(t2, X2(:,7),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,2);
plot(t1, X(:,8),'linewidth', 2);
hold on
plot(t2, X2(:,8),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,3);
plot(t1, X(:,4),'linewidth', 2);
hold on
plot(t2, X2(:,4),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('p (rad/s)')
title('change in roll rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,4);
plot(t1, X(:,5),'linewidth', 2);
hold on
plot(t2, X2(:,5),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')
legend( 'NL', 'L');
hold off
```

```
%
subplot(4,2,5);
plot(t1, X(:,6),'linewidth', 2);
hold on
plot(t2, X2(:,6),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,6);
plot(t1, X(:,1),'linewidth', 2);
hold on
plot(t2, X2(:,1),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,7);
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
hold off

%% .01 pitch rate
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(5) = 0.1;
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting

t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % nonlinear ODE

[t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE

figure ()
```

```
514 sgtitle('Deviation by +0.1 rad/sec in pitch rate, Linear and Nonlinear');
515 subplot(4,2,1);
516 plot(t1, X(:,7),'linewidth', 2);
517 hold on
518 plot(t2, X2(:,7),'linewidth', 2);
519 grid on
520 xlabel('time (s)')
521 ylabel('\phi (rad)')
522 title('change in bank over time')
523 legend( 'NL', 'L');
524 hold off
525 %
526 subplot(4,2,2);
527 plot(t1, X(:,8),'linewidth', 2);
528 hold on
529 plot(t2, X2(:,8),'linewidth', 2);
530 grid on
531 xlabel('time (s)')
532 ylabel('\theta (rad)')
533 title('change in elevation over time')
534 legend( 'NL', 'L');
535 hold off
536 %
537 subplot(4,2,3);
538 plot(t1, X(:,4),'linewidth', 2);
539 hold on
540 plot(t2, X2(:,4),'linewidth', 2);
541 grid on
542 xlabel('time (s)')
543 ylabel('p (rad/s)')
544 title('change in roll rate over time')
545 legend( 'NL', 'L');
546 hold off
547 %
548 subplot(4,2,4);
549 plot(t1, X(:,5),'linewidth', 2);
550 hold on
551 plot(t2, X2(:,5),'linewidth', 2);
552 grid on
553 xlabel('time (s)')
554 ylabel('q (rad/s)')
555 title('change in pitch rate over time')
556 legend( 'NL', 'L');
557 hold off
558 %
559 subplot(4,2,5);
560 plot(t1, X(:,6),'linewidth', 2);
561 hold on
562 plot(t2, X2(:,6),'linewidth', 2);
563 grid on
564 xlabel('time (s)')
565 ylabel('r (rad/s)')
566 title('change in yaw rate over time')
567 legend( 'NL', 'L');
568 hold off
569 %
570 subplot(4,2,6);
571 plot(t1, X(:,1),'linewidth', 2);
572 hold on
573 plot(t2, X2(:,1),'linewidth', 2);
574 grid on
575 xlabel('time (s)')
576 ylabel('u (m/s)')
577 title('change in u over time')
578 legend( 'NL', 'L');
579 hold off
580 %
581 subplot(4,2,7);
```

```
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
hold off
%% .01 yaw rate
tspan = linspace(0,5); % time vector
Pertubations = zeros(1, 3); % No perturbation
TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
conditions = zeros(1, 12); % initialize conditions vector (very large)
conditions(6) = 0.1;
conditions(12) = -1; % set down direction to 1 to make signs correct for plotting

t1 = 0;
t2 = 0;
X = 0;
X2 = 0;

options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met

[t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options);

[t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options);


figure()
sgtitle('Deviation by +0.1 rad/sec in yaw rate, Linear and Nonlinear');
subplot(4,2,1);
plot(t1, X(:,7),'linewidth', 2);
hold on
plot(t2, X2(:,7),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\phi (rad)')
title('change in bank over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,2);
plot(t1, X(:,8),'linewidth', 2);
hold on
plot(t2, X2(:,8),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('\theta (rad)')
title('change in elevation over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,3);
```

```
plot(t1, X(:,4),'linewidth', 2);
hold on
plot(t2, X2(:,4),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('p (rad/s)')
title('change in roll rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,4);
plot(t1, X(:,5),'linewidth', 2);
hold on
plot(t2, X2(:,5),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('q (rad/s)')
title('change in pitch rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,5);
plot(t1, X(:,6),'linewidth', 2);
hold on
plot(t2, X2(:,6),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('r (rad/s)')
title('change in yaw rate over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,6);
plot(t1, X(:,1),'linewidth', 2);
hold on
plot(t2, X2(:,1),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,7);
plot(t1, X(:,2),'linewidth', 2);
hold on
plot(t2, X2(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
legend( 'NL', 'L');
hold off
%
subplot(4,2,8);
plot(t1, X(:,3),'linewidth', 2);
hold on
plot(t2, X2(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')
legend( 'NL', 'L');
hold off
```

Question 4 Script

```
% This script answers question 4 by plotting the experimental data from
% 11:37, 2/14/2020, and plotting the simulated linearized feedback control
% simulation on top for comparison
%   Author: Benjiman Smith
%   Collaborators: E. Owen, I. Quezada
%   Date: 2/20/2020
%

clear all
close all
clc
m = 0.068; % mass of the drone [kg]
r = 0.06;  % body to motor distance [m]
k = 0.0024;  % [Nm/N]
rad = r/sqrt(2);  % [m]
g = 9.81; % gravity [m/s^2]
alpha = 2e-6;    % [N/(m/s)^2]
eta = 1e-3;     % [N/(rad/s)^2]
Ix = 6.8e-5;    % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5;    % moment of inertia in the y direction [kg m^2]
Iz = 1.35e-4;  % moment of inertia in the z direction [kg m^2]
%% lateral control
Lambda1 = -2; % 1st lambda value, found from lab calculations
Lambda2 = -20;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)

syms K1 K2 % symbollically solving for K1 and K2
eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution


[A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
    into independent vectors

Solution = linsolve(A,B); % solve the system of equations
Solution = double(Solution); % convert symbolic solution vector to vector of doubles
K1 = Solution(1); % bank control
K2 = Solution(2); % bank control

%% Longitudional Control
syms K3 K4 % symbollically solving for K3 and K4
eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution


[C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);

ForceVect2 = linsolve(C,D);
ForceVect2 = double(ForceVect2);
K3 = ForceVect2(1); % elevation control
K4 = ForceVect2(2); % elevation control

givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
F = m*g;


load('RSdata_1147.mat'); % load in data
% loading in of parameters
theta = rt_estim.signals.values(:,5);
phi = rt_estim.signals.values(:,6);
u = rt_estim.signals.values(:,7);
v = rt_estim.signals.values(:,8);
w = rt_estim.signals.values(:,9);
p = rt_estim.signals.values(:,10);
q = rt_estim.signals.values(:,11);
r = rt_estim.signals.values(:,12);
times =rt_estim.time(:);
```

```
68
69
70  startpoint = max(theta); % find maximum theta angle, beginning of disturbance
71  timestart = find(theta==startpoint); % find the index of this maximum value
72  subtracttime = times(timestart); % find the time at which the maximum occurs
73
74  tspan = linspace(0,5); % time vector
75  Pertubations = zeros(1, 3); % No perturbation
76  TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
77  conditions = zeros(1, 12); % initialize conditions vector (very large)
78  conditions(8) = deg2rad(5);
79  conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
80  t1 = 0; % initialize time
81  X = 0;% initialize x
82
83  options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
        tolerance of 1e-8 is met
84  [t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
        options); % linear controlled ode call
85
86  %% plotting
87  sgtitle('Deviation by +5^{\circ} in Elevation, Experimental'); % subplot title
88   subplot(4,2,1);
89  plot(times, phi); % plot experimental data
90  hold on
91  plot(t1 + subtracttime, X(:,7), 'linewidth', 2) % plot simulator data
92  grid on
93  xlabel('time (s)')
94  ylabel('\phi (rad)')
95  title('change in bank over time')
96   legend('Experimental Data', 'Simulated Data');
97  hold off
98   subplot(4,2,2);
99  plot(times, theta);
100 hold on
101 plot(t1 + subtracttime, X(:,8),'linewidth', 2);
102 grid on
103 xlabel('time (s)')
104 ylabel('\theta (rad)')
105 title('change in elevation over time')
106 legend('Experimental Data', 'Simulated Data');
107 hold off
108  subplot(4,2,3);
109 plot(times, p);
110 hold on
111 plot(t1 + subtracttime, X(:,4),'linewidth', 2);
112 grid on
113 xlabel('time (s)')
114 ylabel('p (rad/s)')
115 title('change in roll rate over time')
116  legend('Experimental Data', 'Simulated Data');
117 hold off
118  subplot(4,2,4);
119 plot(times, q);
120 hold on
121 plot(t1 + subtracttime, X(:,5),'linewidth', 2);
122 grid on
123 xlabel('time (s)')
124 ylabel('q (rad/s)')
125 title('change in pitch rate over time')
126  legend('Experimental Data', 'Simulated Data');
127 hold off
128  subplot(4,2,5);
129 plot(times, r);
130 hold on
131 plot(t1 + subtracttime, X(:,6),'linewidth', 2);
132 grid on
133 xlabel('time (s)')
```

```
ylabel('r (rad/s)')
title('change in yaw rate over time')
 legend('Experimental Data', 'Simulated Data');
hold off
 subplot(4,2,6);
plot(times, u);
hold on
plot(t1 + subtracttime, X(:,1),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('u (m/s)')
title('change in u over time')
 legend('Experimental Data', 'Simulated Data');
hold off
 subplot(4,2,7);
plot(times, v);
hold on
plot(t1 + subtracttime, X(:,2),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('v (m/s)')
title('change in v over time')
 legend('Experimental Data', 'Simulated Data');
hold off
 subplot(4,2,8);
plot(times, w);
hold on
plot(t1 + subtracttime, X(:,3),'linewidth', 2);
grid on
xlabel('time (s)')
ylabel('w (m/s)')
title('change in w over time')

 legend('Experimental Data', 'Simulated Data');
hold off
```

linearized feedback control ODE function

```
% linear ODE Function
%Benjiman Smith
%02/13/2020
function dydt = Specs2LB4LC(t, Conditions, Force, Disturb, givens)
    % define givens
    m = givens(6);      % mass of the drone [kg]
    r = givens(7);       % body to motor distance [m]
    k = givens(8);     % [Nm/N]
    R = givens(9);  % [m]
    g = givens(10);% gravity [m/s^2]
    alpha = givens(1);   % [N/(m/s)^2]
    eta = givens(2);    % [N/(rad/s)^2]
    I_x = givens(3);   % moment of inertia in the x direction [kg m^2]
    I_y = givens(4);   % moment of inertia in the y direction[kg m^2]
    I_z = givens(5);  % moment of inertia in the z direction[kg m^2]
    k_1 = givens(11);
    k_2 = givens(12);
    k_3 = givens(13);
    k_4 = givens(14);

    % define conditions vector
    deltau = Conditions(1); % inertial velocity in the u direction in body coordinates [m/s]
    deltav = Conditions(2); % inertial velocity in the v direction in body coordinates [m/s]
    deltaw = Conditions(3); % inertial velocity in the w direction in body coordinates [m/s]
    deltap = Conditions(4); % inertial angular velocity in the p direction in body coordinates [
        rad/s]
    deltaq = Conditions(5); % inertial angular velocity in the q direction in body coordinates [
        rad/s]
    deltar = Conditions(6); % inertial angular velocity in the r direction in body coordinates [
```

```
        rad/s]
    deltaphi = Conditions(7); % bank [rad]
     deltatheta = Conditions(8); % elevation [rad]
     psi = Conditions(9); % azimuth [rad]
     x_E = Conditions(10); % position vector in the x direction in inertial coordinates [m]
     y_E = Conditions(11); % position vector in the y direction in inertial coordinates [m]
     z_E = Conditions(12); % position vector in the z direction in inertial coordinates [m]
     f1 = Force(1); % trim force exerted by motor 1 [N]
     f2 = Force(2); % trim force exerted by motor 1 [N]
     f3 = Force(3); % trim force exerted by motor 1 [N]
     f4 = Force(4); % trim force exerted by motor 1 [N]

     % Aerodynamic/Control Moments and Forces
     Laero = - alpha^2 * deltap^2 * sign(deltap); % p component of the aerodynamic moments
     Maero = - alpha^2 * deltaq^2 * sign(deltaq); % q component of the aerodynamic moments
     Naero = - alpha^2 * deltar^2 * sign(deltar); % w component of the aerodynamic moments
  Lcontrol = ((f1 + f2) - (f3 + f4)) * R; % p component of the control moments
     Mcontrol = ((f3 + f2) - (f1 + f4)) * R; % q component of the control moments
     Ncontrol = -0.004*deltar; % w component of the aerodynamic moments
     deltaL = Laero + Lcontrol; % L moment sum (abt x axis)
     deltaM = Maero + Mcontrol; % M moment sum (abt y axis)
     deltaN = Naero + Ncontrol; % N moment sum (abt Z axis)

     if t > 2 && t < 2.5
         deltaL = deltaL + Disturb(2); % Disturbed L moment (abt x axis)
         deltaM = deltaM + Disturb(1); % Disturbed M moment (abt y axis)
         deltaN = deltaN + Disturb(3); % Disturbed N moment (abt z axis)
     end
     Xaero = - eta^2 * deltau^2 * sign(deltau); % x component of aerodynamic force
     Yaero = - eta^2 * deltav^2 * sign(deltav); % y component of aerodynamic force
     Zaero = - eta^2 * deltaw^2 * sign(deltaw); % z component of aerodynamic force
     Xcontrol = 0; % x control
     Ycontrol = 0; % y control
     Zcontrol = -sum(Force); % gravitational force counteraction
     deltaX = Xaero + Xcontrol; % sum of x forces
     deltaY = Yaero + Ycontrol; % sum of y forces
     deltaZ = 0; % sum of z forces
     deltapdot = ((-k_1*deltap)/I_x) -((k_2*deltaphi)/I_x);
     deltaqdot = ((-k_3*deltaq)/I_y) -((k_4*deltatheta)/I_y); % pitch rate derrivative
     deltardot = deltaN/I_z; % yaw rate derrivative

     dOmega_bdt = [deltapdot, deltaqdot, deltardot]';
     deltaudot = -g*deltatheta; % 4.7,1 acceleration in the x axis
     deltavdot = g*deltaphi; % acceleration in the y axis
     deltawdot = deltaZ/m; % acceleration in the z axis
     dVbdt = [deltaudot, deltavdot, deltawdot]';
     xdot =  deltau;
     ydot =  deltav;
     zdot = deltaw;
     dVEdt = [xdot, ydot, zdot]';
     phidot  = deltap; % bank roc (pg 104)
     thetadot = deltaq; % elevation roc (pg 104)
     psidot   = deltar; % azimuth roc (pg 104)
     dEuldt = [phidot, thetadot, psidot]';
     dydt = [dVbdt; dOmega_bdt; dEuldt; dVEdt];



end
```

non-linearized feedback control ODE function

```
% Nonlinear ODE Function with control
%Benjiman Smith
%02/13/2020
function dydt = Specs2LB4NLC(t, Conditions, Force, Disturb, givens)
    % define givens
```

```
    m = givens(6);      % mass of the drone [kg]
    r = givens(7);       % body to motor distance [m]
    k = givens(8);     % [Nm/N]
    R = givens(9);  % [m]
    g = givens(10);% gravity [m/s^2]
    alpha = givens(1);   % [N/(m/s)^2]
    eta = givens(2);    % [N/(rad/s)^2]
    I_x = givens(3);   % moment of inertia in the x direction [kg m^2]
    I_y = givens(4);   % moment of inertia in the y direction[kg m^2]
    I_z = givens(5);  % moment of inertia in the z direction[kg m^2]
    k_1 = givens(11);
    k_2 = givens(12);
    k_3 = givens(13);
    k_4 = givens(14);
    % define conditions vector
    u = Conditions(1); % inertial velocity in the u direction in body coordinates [m/s]
    v = Conditions(2); % inertial velocity in the v direction in body coordinates [m/s]
    w = Conditions(3); % inertial velocity in the w direction in body coordinates [m/s]
    p = Conditions(4); % inertial angular velocity in the p direction in body coordinates [rad/s]
    q = Conditions(5); % inertial angular velocity in the q direction in body coordinates [rad/s]
    r = Conditions(6); % inertial angular velocity in the r direction in body coordinates [rad/s]
    phi = Conditions(7); % bank [rad]
    theta = Conditions(8); % elevation [rad]
    psi = Conditions(9); % azimuth [rad]
    x_E = Conditions(10); % position vector in the x direction in inertial coordinates [m]
    y_E = Conditions(11); % position vector in the y direction in inertial coordinates [m]
    z_E = Conditions(12); % position vector in the z direction in inertial coordinates [m]


    % Aerodynamic/Control Moments and Forces
    Laero = - alpha^2 * p^2 * sign(p); % p component of the aerodynamic moments
    Maero = - alpha^2 * q^2 * sign(q); % q component of the aerodynamic moments
    Naero = - alpha^2 * r^2 * sign(r); % w component of the aerodynamic moments
    Lcontrol = -k_1*p - k_2*phi;
    Mcontrol = -k_3*q - k_4*theta;
    Ncontrol = -0.004*r; % w component of the aerodynamic moments
    L = Laero + Lcontrol; % L moment sum (abt x axis)
    M = Maero + Mcontrol; % M moment sum (abt y axis)
    N = Naero + Ncontrol; % N moment sum (abt Z axis)

%         L = p*.004 - L  ; % Disturbed L moment (abt x axis)
%         M = q*.004 - M; % Disturbed M moment (abt y axis)
%         N = r*0.004 - N; % Disturbed N moment (abt z axis)

    Xaero = - eta^2 * u^2 * sign(u); % x component of aerodynamic force
    Yaero = - eta^2 * v^2 * sign(v); % y component of aerodynamic force
    Zaero = - eta^2 * w^2 * sign(w); % z component of aerodynamic force
    Xcontrol = 0; % x control
    Ycontrol = 0; % y control
    Zcontrol = -sum(Force); % gravitational force counteraction
    X = Xaero + Xcontrol; % sum of x forces
    Y = Yaero + Ycontrol; % sum of y forces
    Z = Zaero + Zcontrol; % sum of z forces
    pdot = p*q*(I_y - I_z)/I_x  + (1/I_x * L); % roll rate derrivative
    qdot = p * r*(I_z - I_x)/I_y  + (1/I_y * M); % pitch rate derrivative
    rdot =  p * q*(I_x - I_y)/I_z  + (1/I_z * N); % yaw rate derrivative

    dOmega_bdt = [pdot, qdot, rdot]';
    udot = r*v - q*w - g*sin(theta) + 1/m * X; % 4.7,1 acceleration in the x axis
    vdot = p*w - r*u + g*sin(phi)*cos(theta) + 1/m * Y; % acceleration in the y axis
    wdot = q*u - p*v + g*cos(phi)*cos(theta) + 1/m * Z; % acceleration in the z axis
    dVbdt = [udot, vdot, wdot]';
    xdot =  u * cos(theta)*cos(psi) +v * (sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi)) + w *
         (cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi));
    ydot =  u * cos(theta)*sin(psi) + v * (sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi)) + w
        * (cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi));
    zdot = -u * sin(theta) + v * sin(phi)*cos(theta) + w * cos(phi)*cos(theta);
    dVEdt = [xdot, ydot, zdot]';
```

```
    phidot   = p + (q*sin(phi)+r*cos(phi))*tan(theta); % bank roc (pg 104)
    thetadot = q*cos(phi) - r*sin(phi); % elevation roc (pg 104)
    psidot   = q*sin(phi)*sec(theta) + r*cos(phi)*sec(theta); % azimuth roc (pg 104)

    dEuldt = [phidot, thetadot, psidot]';
    dydt = [dVbdt; dOmega_bdt; dEuldt; dVEdt];



end
```

Stop Function

```
% This function is used by ODE45 in options to end the drones
% path after a certain tolerance of values is reached
%
%   Author: Benjiman Smith
%   Collaborators: E. Owen, I. Quezada
%   Date: 1/26/2020
%
function [value, isTerm, direction] = StopFnct(t, F)
value = F(12);
isTerm = 1;
direction = [];
end
```