

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

I Nomenclature	2
II Question 1	3
III Question 2	11
IV Question 3	13
V Question 4	15
VI MATLAB Code	18

List of Figures

1 Change in initial Bank, Problem 2	11
2 Change in initial Elevation, Problem 2	12
3 Change in Roll Rate, Problem 2	12
4 Change in Pitch Rate, Problem 2	13
5 Change in initial Bank, Problem 3	14
6 Change in initial Elevation, Problem 3	14
7 Change in Roll Rate, Problem 3	15
8 Change in Pitch Rate, Problem 3	15
9 GUI Screenshot	16
10 Experimental data, 5° initial elevation	16

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^2]
\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]
Δu^E	= derivative of the inertial in the body x direction deviated from trim [m/s^2]
Δv^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 [radian/ 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

$$\begin{bmatrix} \Delta P \\ \Delta \dot{\phi} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} -K_1 & -K_2 \\ \frac{I_x}{I_x} & \frac{I_x}{I_x} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta P \\ \Delta \phi \end{bmatrix}$$

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

\uparrow
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 EQUATION

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

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

IN WHICH THE $e^{j\omega t}$ TERM IS OSCILLATORY. 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:

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

$$\Rightarrow \ln(e^{\lambda_1 t}) = \ln(e^{nt}) \Rightarrow \lambda_1 t = nt \\ \therefore \boxed{\lambda_1 = n}$$

THIS ALLOWS FOR US TO SOLVE FOR λ_1 EASILY.

τ = TIME CONSTANT

$\tau = 0.5 \text{ sec}$ (FROM ASSIGNMENT)

$$-n\tau \quad \sim \sim \quad . \quad -n\tau \quad , \quad \sim \sim \backslash$$

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

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

↑

IN THIS CASE, $n = \lambda_1$

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

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

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

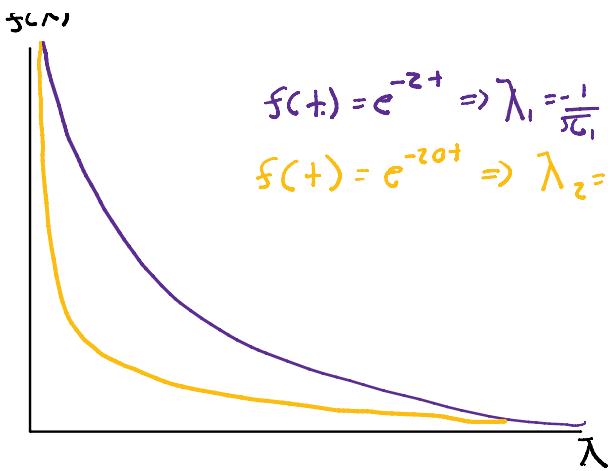
$n = -1/\tau$

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:

$\text{sc}\lambda$



$$f(t) = e^{-2t} \Rightarrow \lambda_2 = -2 \Rightarrow \tau_2 = 0.5 \text{ sec}$$



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]

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

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

$$\lambda_1 \text{ & } \lambda_2$$

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

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

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

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

$$\Rightarrow 400 - \frac{20}{6.8 \times 10^{-5}} \left[1.36 \times 10^{-4} + \frac{K_2}{2} \right] + \frac{K_2}{6.8 \times 10^{-5}} = 0$$

$$\Rightarrow 400 - \frac{0.00544 - 20K_2}{6.8 \times 10^{-5}} + \frac{K_2}{6.8 \times 10^{-5}} = 0 \Rightarrow 360 - \frac{9K_2}{6.8 \times 10^{-5}} = 0$$

$$\Rightarrow K_2 = 0.00272$$

$$\Rightarrow K_1 = 1.36 \times 10^{-4} + \frac{0.00272}{2} = 0.001496$$

$$\Rightarrow K_1 = 0.001496$$

$$K_2 = 0.00272$$

§ LONGITUDINAL CONTROL

SIMILAR TO LATERAL CONTROL, THE EQUATIONS OF MOTION

FOR A CLOSED LOOP ARE

$$\begin{bmatrix} \Delta \dot{\theta} \\ \Delta \dot{\alpha} \end{bmatrix} = \begin{bmatrix} -k_y & -\frac{k_y}{I_y} \\ \frac{1}{I_y} & 0 \end{bmatrix} \begin{bmatrix} \Delta \alpha \\ \Delta \epsilon \end{bmatrix}$$

$$\underline{x} = \underline{A} \cdot \underline{x}$$

LET $\underline{A}\underline{x} = \lambda \underline{x}$, WHERE λ IS A SCALAR

$$\text{THEREFORE } (\underline{A} - \lambda \underline{I}) \underline{x} = 0$$

\uparrow

IDENTITY
MATRIX

SINCE WE KNOW $\underline{x} \neq 0$, AS WE HAVE

$$\text{REAL } \Delta \alpha \text{ & } \Delta \theta \text{ VALUES, } \det(\underline{A} - \lambda \underline{I}) = 0$$

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 DERIVED 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

$$\lambda_1 \text{ & } \lambda_2 \qquad I_y = 9.2E-5 \text{ kgm}^2$$

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

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

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

$$\frac{L}{9.2 \times 10^{-5}} \quad z \quad \frac{9.2 \times 10^{-5}}{9.2 \times 10^{-5}}$$

$$\Rightarrow 400 - \frac{0.00736 - 20K_4}{z} + \frac{K_4}{9.2 \times 10^{-5}} = 0 \Rightarrow 360 - \frac{9K_4}{9.2 \times 10^{-5}} = 0$$

$$\Rightarrow K_4 = 0.00368$$

$$\Rightarrow K_3 = 1.84 \times 10^{-4} + \frac{(0.00368)}{z} = 0.002024$$

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

$K_1, K_2, K_3, \text{ and } K_4$ 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 $+5^\circ$ bank, $+5^\circ$ elevation, $+0.1$ radian/sec in roll rate (p), and $+0.1$ radian/sec in pitch rate (q).

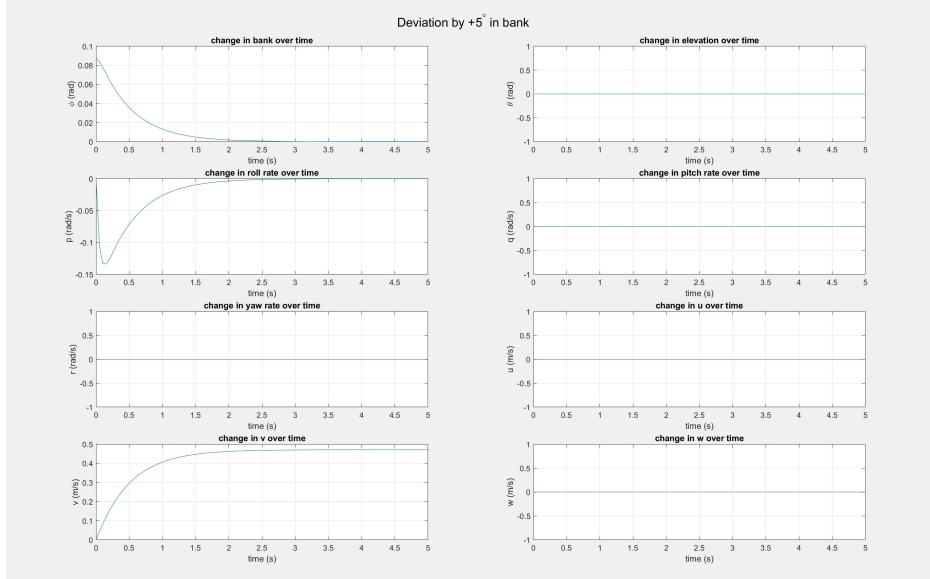


Fig. 1 Change in initial Bank, Problem 2

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.

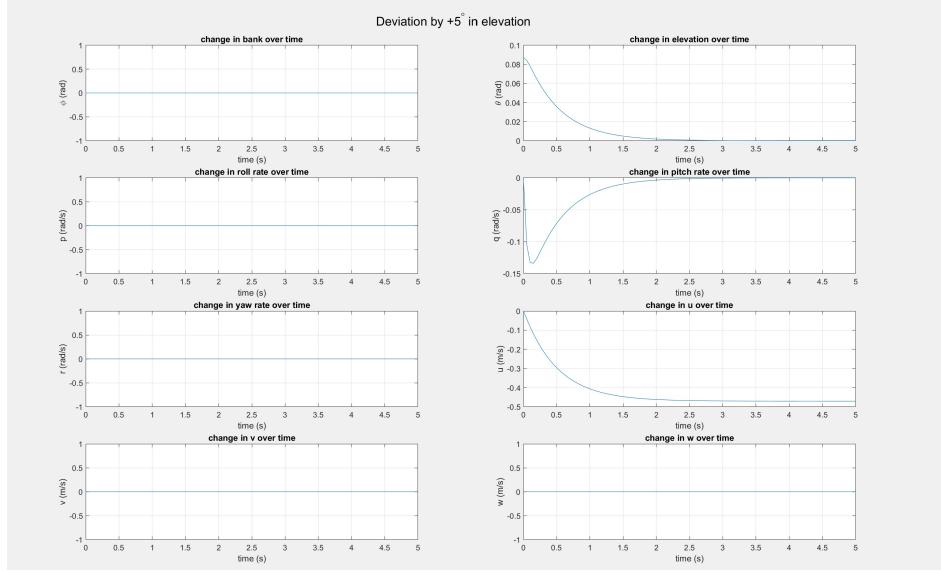


Fig. 2 Change in initial Elevation, Problem 2

as seen by figure 2, the values for v , w , p , r , \dot{r} , and ϕ , 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 implemented into the feedback system.

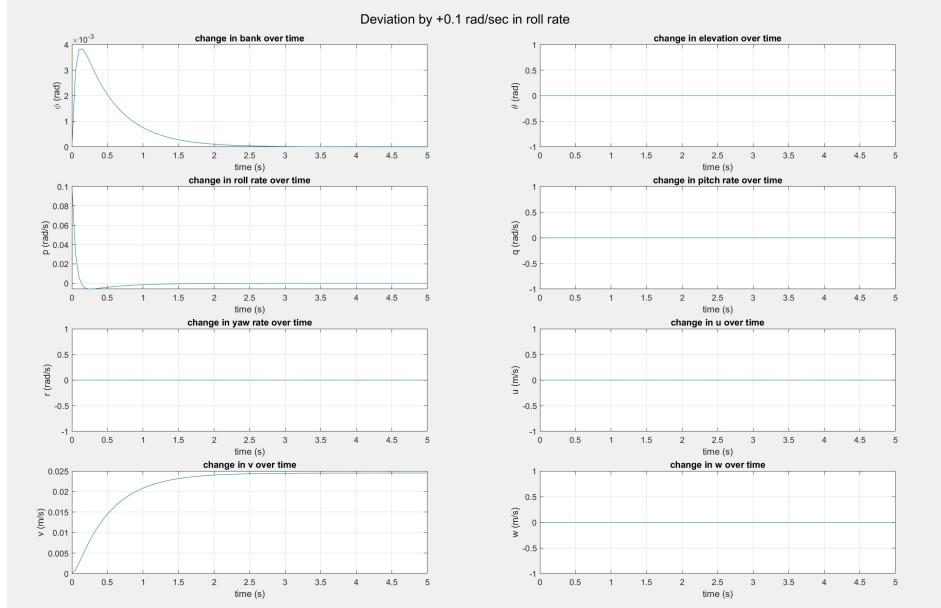


Fig. 3 Change in Roll Rate, Problem 2

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 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 due to the lack of altitude feedback control.

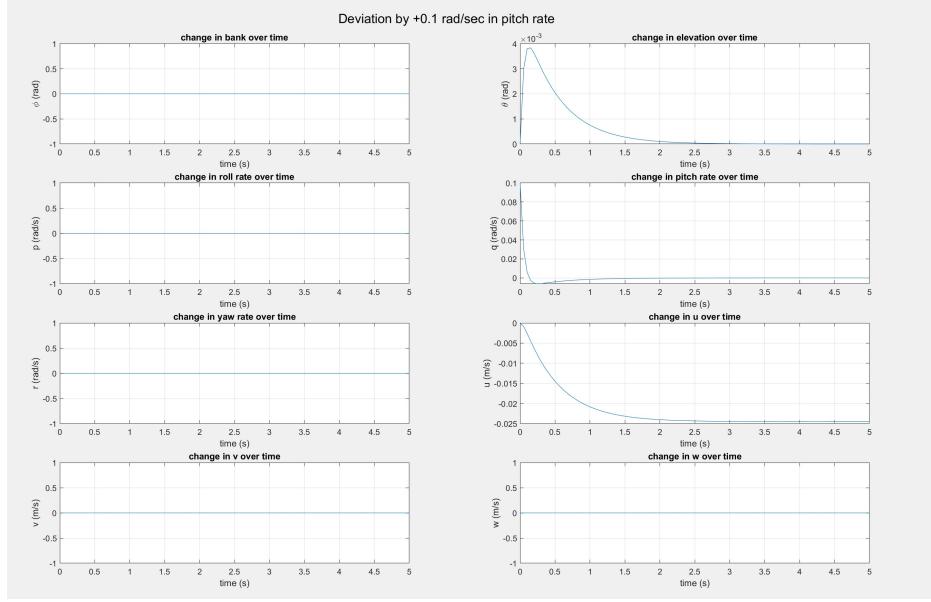


Fig. 4 Change in Pitch Rate, Problem 2

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 $+5^\circ$ bank, $+5^\circ$ pitch, $+0.1$ radian/sec in roll rate (p),and $+0.1$ radian/sec in pitch rate (q).

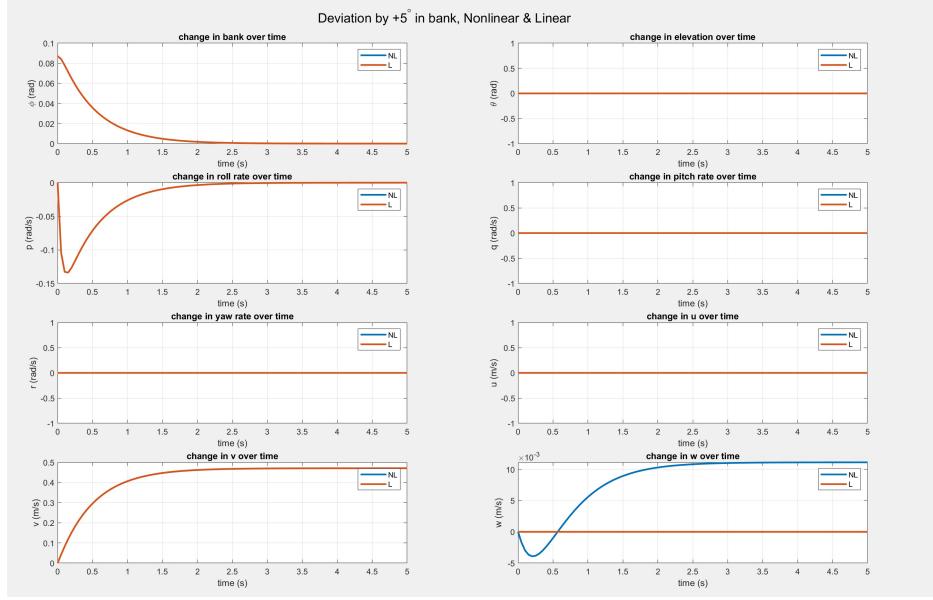


Fig. 5 Change in initial Bank, Problem 3

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 initially, 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 implemented into the model.

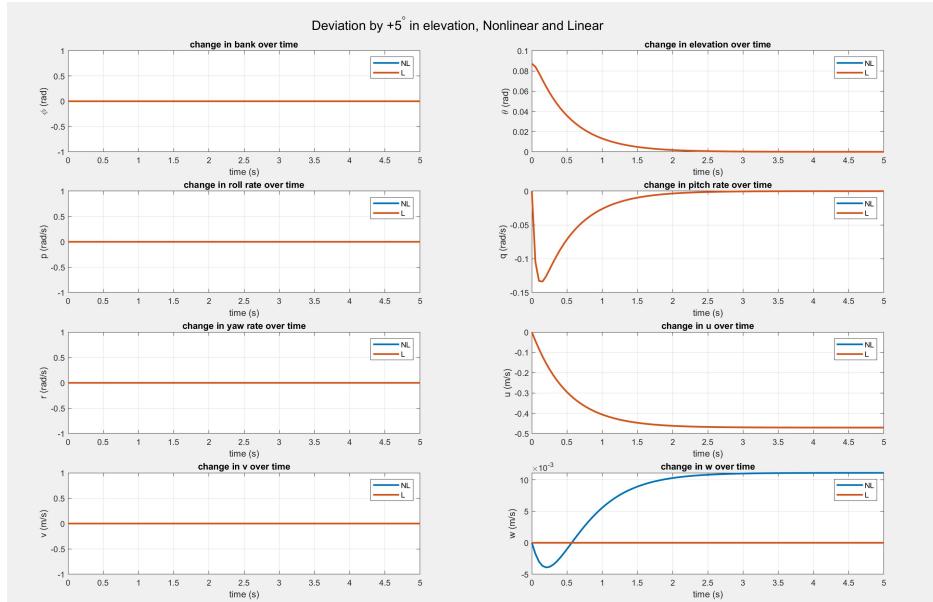


Fig. 6 Change in initial Elevation, Problem 3

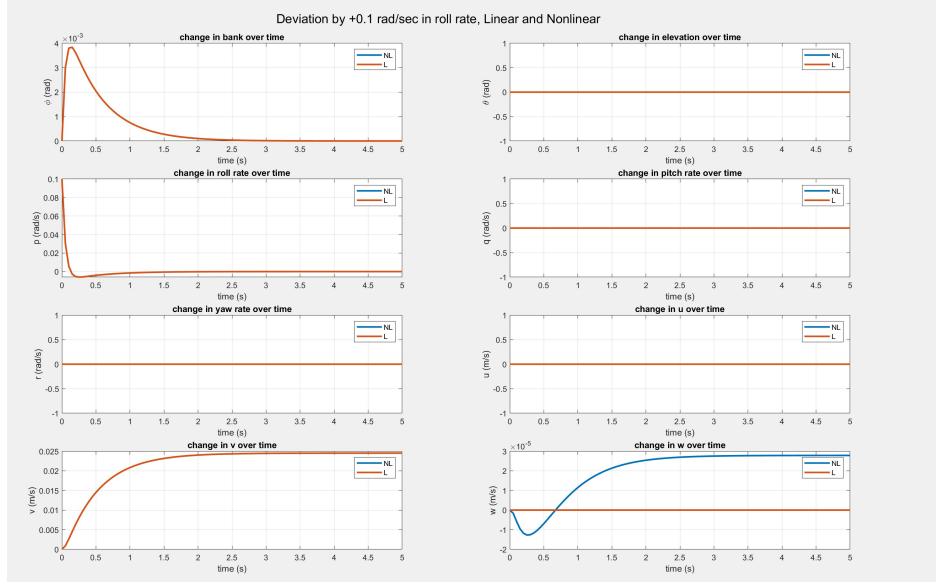


Fig. 7 Change in Roll Rate, Problem 3

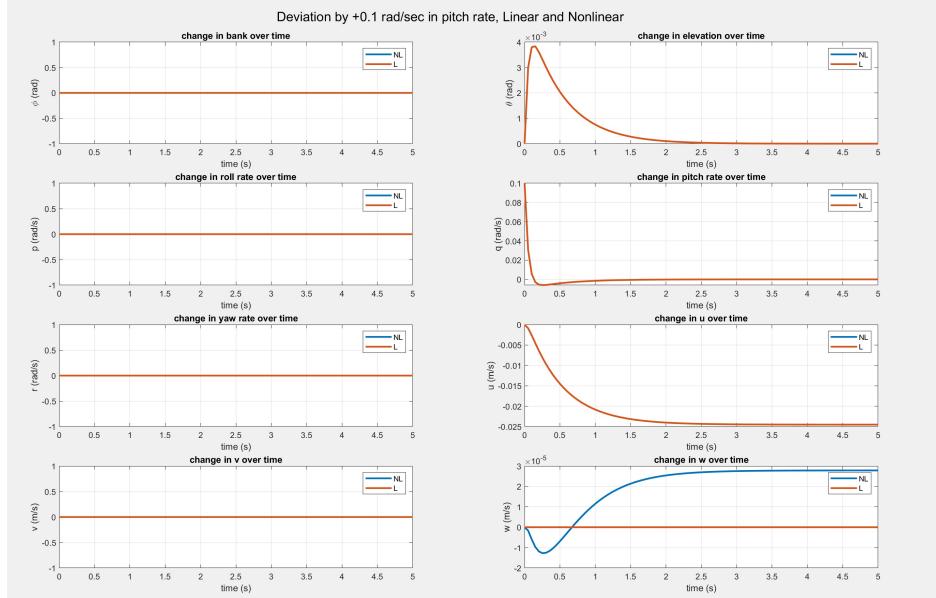


Fig. 8 Change in Pitch Rate, Problem 3

V. Question 4

Implementing 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.

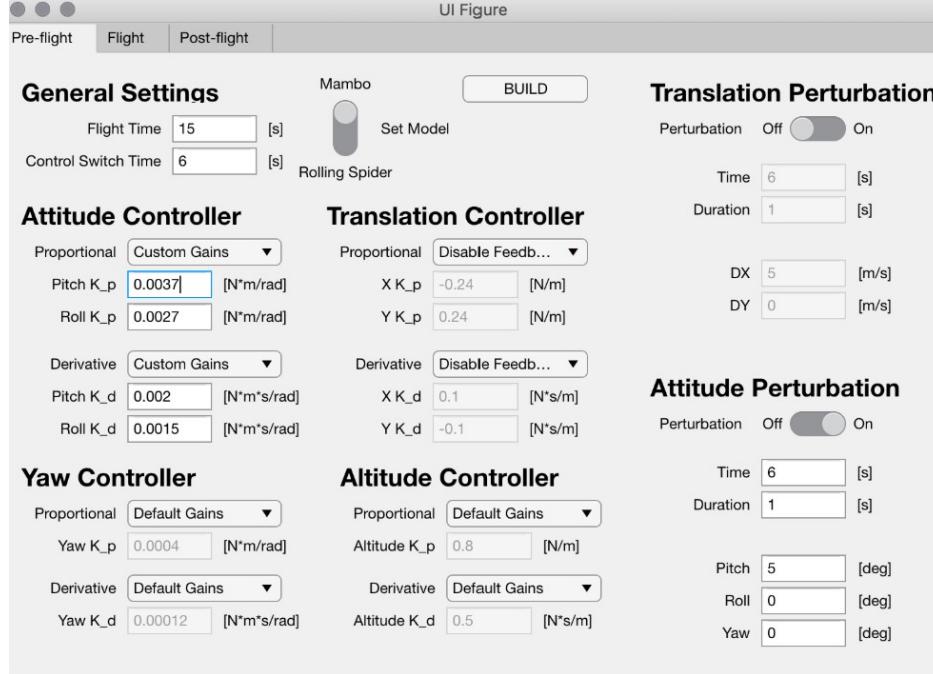


Fig. 9 GUI Screenshot

As seen from this picture, the gain values used were the same as derived 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.

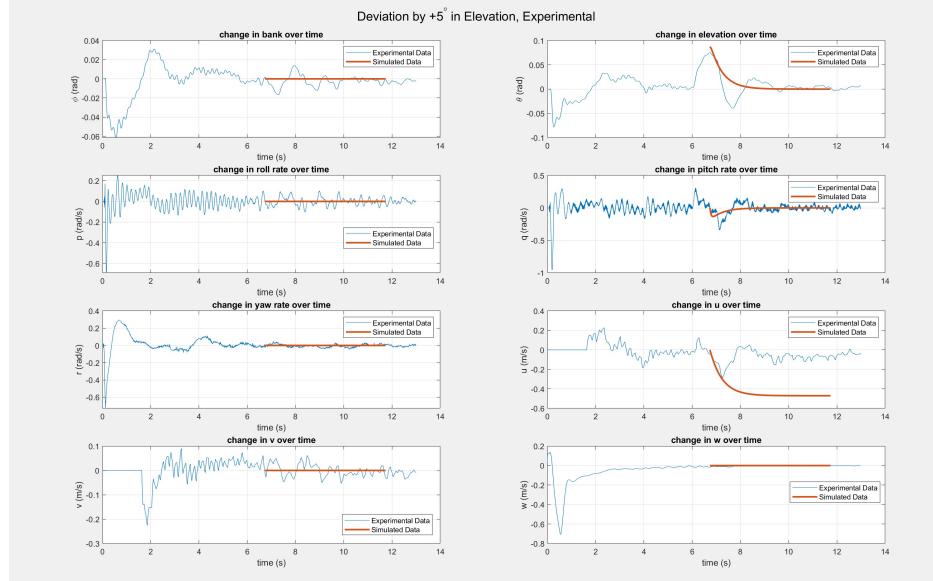


Fig. 10 Experimental data, 5° initial elevation

As seen from this figure, the experimental data matches with the experimental data roughly, but due 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 implmented 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

```
1 % This script finds the values for derivative gain controlling both
2 % the longitudinal and lateral rates of the drone
3 %
4 %
5 % Author: Benjiman Smith
6 % Collaborators: E. Owen, I. Quezada
7 % Date: 1/25/2020
8 %
9 %% lateral control
10 clear all
11 close all
12 Lambda1 = -2; % 1st lambda value, found from lab calculations
13 Lambda2 = -12;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
14 Ix = 6.8e-5; % moment of inertia in the x direction [kg m^2]
15 Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)
16
17 syms K1 K2 % symbollically solving for K1 and K2
18 eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
19 eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution
20
21
22 [A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
   into independent vectors
23
24 Solution = linsolve(A,B); % solve the system of equations
25 Solution = double(Solution); % convert symbolic solution vector to vector of doubles
26 K1 = Solution(1); % bank control
27 K2 = Solution(2); % bank control
28
29 %% Longitudinal Control
30 syms K3 K4 % symbollically solving for K3 and K4
31 eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
32 eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
33
34
35 [C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);
36
37 ForceVect2 = linsolve(C,D);
38 ForceVect2 = double(ForceVect2);
39 K3 = ForceVect2(1); % elevation control
40 K4 = ForceVect2(2); % elevation control
```

Question 2 Script

```
1 % This script initializes initial conditions for the trajectory of a drone
2 % in steady flight conditions (hovering) then calls the ODE45 function to
3 % numerically integrate the position of the drone with respect to the
4 % initial conditions, along with plotting the results answering Question
5 % 2 of Assignment 4
6 %
7 % Author: Benjiman Smith
8 % Collaborators: E. Owen, I. Quezada
9 % Date: 2/20/2020
10 %
11 clc;
12 clear all;
13 close all;
14 m = 0.068; % mass of the drone [kg]
15 r = 0.06; % body to motor distance [m]
16 k = 0.0024; % [Nm/N]
17 rad = r/sqrt(2); % [m]
18 g = 9.81; % gravity [m/s^2]
19 alpha = 2e-6; % [N/(m/s)^2]
```

```

20 eta = 1e-3;      % [N/(rad/s)^2]
21 Ix = 6.8e-5;    % moment of inertia in the x direction [kg m^2]
22 Iy = 9.2e-5;    % moment of inertia in the y direction [kg m^2]
23 Iz = 1.35e-4;   % moment of inertia in the z direction [kg m^2]
24 %% lateral control
25 Lambda1 = -2; % 1st lambda value, found from lab calculations
26 Lambda2 = -20;% 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
27 Ix = 6.8e-5;   % moment of inertia in the x direction [kg m^2]
28 Iy = 9.2e-5;   % Moment of inertia abt y (kgm^2)
29
30 syms K1 K2 % symbollically solving for K1 and K2
31 eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
32 eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution
33
34
35 [A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
36     into independent vectors
37
38 Solution = linsolve(A,B); % solve the system of equations
39 Solution = double(Solution); % convert symbolic solution vector to vector of doubles
40 K1 = Solution(1); % bank control
41 K2 = Solution(2); % bank control
42
43 %% Longitudinal Control
44 syms K3 K4 % symbollically solving for K3 and K4
45 eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
46 eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
47
48 [C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);
49
50 ForceVect2 = linsolve(C,D);
51 ForceVect2 = double(ForceVect2);
52 K3 = ForceVect2(1); % elevation control
53 K4 = ForceVect2(2); % elevation control
54
55 givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
56 F = m*g;
57
58 %% bank is 5
59 tspan = linspace(0,5); % time vector
60 Pertubations = zeros(1, 3); % No perturbation
61 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
62 conditions = zeros(1, 12); % initialize conditions vector (very large)
63 conditions(7) = deg2rad(5); % bank angle is 5 degrees
64 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
65 t1 = 0;
66 t2 = 0;
67 X = 0;
68 X2 = 0;
69 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
70 tolerance of 1e-8 is met
71 [t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
72 options); % ode call for linearized feedback control ode
73
74 %% plotting
75
76
77 sgtitle('Deviation by +5^{\circ} in bank');
78 subplot(4,2,1);
79 plot(t1, X(:,7));
80
81 grid on
82 xlabel('time (s)')
83 ylabel('phi (rad)')
84 title('change in bank over time')

```

```

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

```

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, givens), tspan, conditions,
    options); % ode call for linearized feedback control ode
169
170
171 %% plotting
172 figure()
173 sgttitle('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)')

```

```

219 title('change in yaw rate over time')
220
221 hold off
222 %
223 subplot(4,2,6);
224 plot(t1, X(:,1));
225
226 grid on
227 xlabel('time (s)')
228 ylabel('u (m/s)')
229 title('change in u over time')
230
231 hold off
232 %
233 subplot(4,2,7);
234 plot(t1, X(:,2));
235
236 grid on
237 xlabel('time (s)')
238 ylabel('v (m/s)')
239 title('change in v over time')
240
241 hold off
242 %
243 subplot(4,2,8);
244 plot(t1, X(:,3));
245
246 grid on
247 xlabel('time (s)')
248 ylabel('w (m/s)')
249 title('change in w over time')
250
251 hold off
252
253 %% azimuth is 5
254 tspan = linspace(0,5); % time vector
255 Pertubations = zeros(1, 3); % No perturbation
256 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
257 conditions = zeros(1, 12); % initialize conditions vector (very large)
258 conditions(9) = deg2rad(5); % azimuth angle is 5
259 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
260 t1 = 0;
261 t2 = 0;
262 X = 0;
263 X2 = 0;
264 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
               % tolerance of 1e-8 is met
265 [t1, X] = ode45(@(t, F) Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
               options); % ode call for linearized feedback control ode
266
267 figure()
268 sgttitle('Deviation by +5^{\circ} in azimuth'); subplot(4,2,1);
269 plot(t1, X(:,7));
270
271 grid on
272 xlabel('time (s)')
273 ylabel('phi (rad)')
274 title('change in bank over time')
275
276 hold off
277 %
278 subplot(4,2,2);
279 plot(t1, X(:,8));
280
281 grid on
282 xlabel('time (s)')
283 ylabel('theta (rad)')
284 title('change in elevation over time')

```

```

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

```

353 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
354 t1 = 0;
355 t2 = 0;
356 X = 0;
357 X2 = 0;
358 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
            tolerance of 1e-8 is met
359 [t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
            options);
360
361
362 figure()
363 sgtitle('Deviation by +0.1 rad/sec in roll rate');
364 subplot(4,2,1);
365 plot(t1, X(:,7));
366
367 grid on
368 xlabel('time (s)')
369 ylabel('\phi (rad)')
370 title('change in bank over time')
371
372 hold off
373 %
374 subplot(4,2,2);
375 plot(t1, X(:,8));
376
377 grid on
378 xlabel('time (s)')
379 ylabel('\theta (rad)')
380 title('change in elevation over time')
381
382 hold off
383 %
384 subplot(4,2,3);
385 plot(t1, X(:,4));
386
387 grid on
388 xlabel('time (s)')
389 ylabel('p (rad/s)')
390 title('change in roll rate over time')
391 hold off
392 %
393 subplot(4,2,4);
394 plot(t1, X(:,5));
395
396 grid on
397 xlabel('time (s)')
398 ylabel('q (rad/s)')
399 title('change in pitch rate over time')
400
401 hold off
402 %
403 subplot(4,2,5);
404 plot(t1, X(:,6));
405
406 grid on
407 xlabel('time (s)')
408 ylabel('r (rad/s)')
409 title('change in yaw rate over time')
410
411 hold off
412 %
413 subplot(4,2,6);
414 plot(t1, X(:,1));
415
416 grid on
417 xlabel('time (s)')
418 ylabel('u (m/s)')

```

```

419 title('change in u over time')
420
421 hold off
422 %
423 subplot(4,2,7);
424 plot(t1, X(:,2));
425
426 grid on
427 xlabel('time (s)')
428 ylabel('v (m/s)')
429 title('change in v over time')
430
431 hold off
432 %
433 subplot(4,2,8);
434 plot(t1, X(:,3));
435
436 grid on
437 xlabel('time (s)')
438 ylabel('w (m/s)')
439 title('change in w over time')
440
441 hold off
442
443 %% .01 pitch rate
444 tspan = linspace(0,5); % time vector
445 Pertubations = zeros(1, 3); % No perturbation
446 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
447 conditions = zeros(1, 12); % initialize conditions vector (very large)
448 conditions(5) = 0.1; % pitch rate is 0.1 rad/s
449 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
450 t1 = 0;
451 t2 = 0;
452 X = 0;
453 X2 = 0;
454 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
        tolerance of 1e-8 is met
455 [t1, X] = ode45(@(t, F) Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
        options);
456
457 %% plotting
458 figure()
459 sgttitle('Deviation by +0.1 rad/sec in pitch rate');
460 subplot(4,2,1);
461 plot(t1, X(:,7));
462
463 grid on
464 xlabel('time (s)')
465 ylabel('\phi (rad)')
466 title('change in bank over time')
467
468 hold off
469 %
470 subplot(4,2,2);
471 plot(t1, X(:,8));
472
473 grid on
474 xlabel('time (s)')
475 ylabel('\theta (rad)')
476 title('change in elevation over time')
477
478 hold off
479 %
480 subplot(4,2,3);
481 plot(t1, X(:,4));
482
483 grid on
484 xlabel('time (s)')

```

```

485 ylabel('p (rad/s)')
486 title('change in roll rate over time')
487
488 hold off
489 %
490 subplot(4,2,4);
491 plot(t1, X(:,5));
492
493 grid on
494 xlabel('time (s)')
495 ylabel('q (rad/s)')
496 title('change in pitch rate over time')
497
498 hold off
499 %
500 subplot(4,2,5);
501 plot(t1, X(:,6));
502
503 grid on
504 xlabel('time (s)')
505 ylabel('r (rad/s)')
506 title('change in yaw rate over time')
507
508 hold off
509 %
510 subplot(4,2,6);
511 plot(t1, X(:,1));
512
513 grid on
514 xlabel('time (s)')
515 ylabel('u (m/s)')
516 title('change in u over time')
517 hold off
518 %
519 subplot(4,2,7);
520 plot(t1, X(:,2));
521
522 grid on
523 xlabel('time (s)')
524 ylabel('v (m/s)')
525 title('change in v over time')
526 hold off
527 %
528 subplot(4,2,8);
529 plot(t1, X(:,3));
530
531 grid on
532 xlabel('time (s)')
533 ylabel('w (m/s)')
534 title('change in w over time')
535 hold off
536
537 %% .01 yaw rate
538 tspan = linspace(0,5); % time vector
539 Pertubations = zeros(1, 3); % No perturbation
540 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
541 conditions = zeros(1, 12); % initialize conditions vector (very large)
542 conditions(6) = 0.1; % yaw rate is 0.1 rad/s
543 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
544 t1 = 0;
545 t2 = 0;
546 X = 0;
547 X2 = 0;
548 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
      tolerance of 1e-8 is met
549 [t1, X] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
      options);
550

```

```

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

```

619 title('change in v over time')
620 hold off
621 %
622 subplot(4,2,8);
623 plot(t1, X(:,3));
624
625 grid on
626 xlabel('time (s)')
627 ylabel('w (m/s)')
628 title('change in w over time')
629 hold off

```

Question 3 Script

```

1 % This script initializes initial conditions for the trajectory of a drone
2 % in steady flight conditions (hovering) then calls the ODE45 function to
3 % numerically integrate the position of the drone with respect to the
4 % initial conditions, along with plotting the results answering Question
5 % 3 of Assignment 4
6 %
7 % Author: Benjiman Smith
8 % Collaborators: E. Owen, I. Quezada
9 % Date: 2/2/2020
10 %
11 clc;
12 clear all;
13 close all;
14 m = 0.068; % mass of the drone [kg]
15 r = 0.06; % body to motor distance [m]
16 k = 0.0024; % [Nm/N]
17 rad = r/sqrt(2); % [m]
18 g = 9.81; % gravity [m/s^2]
19 alpha = 2e-6; % [N/(m/s)^2]
20 eta = 1e-3; % [N/(rad/s)^2]
21 Ix = 6.8e-5; % moment of inertia in the x direction [kg m^2]
22 Iy = 9.2e-5; % moment of inertia in the y direction [kg m^2]
23 Iz = 1.35e-4; % moment of inertia in the z direction [kg m^2]
24
25 %% lateral control
26 Lambda1 = -2; % 1st lambda value, found from lab calculations
27 Lambda2 = -20; % 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
28 Ix = 6.8e-5; % moment of inertia in the x direction [kg m^2]
29 Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)
30
31 syms K1 K2 % symbollically solving for K1 and K2
32 eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
33 eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution
34
35
36 [A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
            into independent vectors
37
38 Solution = linsolve(A,B); % solve the system of equations
39 Solution = double(Solution); % convert symbolic solution vector to vector of doubles
40 K1 = Solution(1); % bank control
41 K2 = Solution(2); % bank control
42
43 %% Longitudinal Control
44 syms K3 K4 % symbollically solving for K3 and K4
45 eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
46 eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
47
48
49 [C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);
50
51 ForceVect2 = linsolve(C,D);
52 ForceVect2 = double(ForceVect2);

```

```

53 K3 = ForceVect2(1); % elevation control
54 K4 = ForceVect2(2); % elevation control
55
56 givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
57 F = m*g;
58
59 %% bank is 5
60 tspan = linspace(0,5); % time vector
61 Pertubations = zeros(1, 3); % No perturbation
62 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
63 conditions = zeros(1, 12); % initialize conditions vector (very large)
64 conditions(7) = deg2rad(5); % bank angle is 5 deg
65 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
66 % initialize variables
67 t1 = 0;
68 t2 = 0;
69 X = 0;
70 X2 = 0;
71
72 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
    tolerance of 1e-8 is met
73
74 [t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); % nonlinear ODE
75
76 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
    options); %Linear ODE
77
78 %sublotting of all 8 variables
79 figure()
80
81 sgtitle('Deviation by +5^{\circ} in bank, Nonlinear & Linear');
82 subplot(4,2,1);
83 plot(t1, X(:,7), 'linewidth', 2);
84 hold on
85 plot(t2, X2(:,7), 'linewidth', 2);
86 grid on
87 xlabel('time (s)')
88 ylabel('\phi (rad)')
89 title('change in bank over time')
90 legend( 'NL', 'L');
91 hold off
92 %
93 subplot(4,2,2);
94 plot(t1, X(:,8), 'linewidth', 2);
95 hold on
96 plot(t2, X2(:,8), 'linewidth', 2);
97 grid on
98 xlabel('time (s)')
99 ylabel('\theta (rad)')
100 title('change in elevation over time')
101 legend( 'NL', 'L');
102 hold off
103 %
104 subplot(4,2,3);
105 plot(t1, X(:,4), 'linewidth', 2);
106 hold on
107 plot(t2, X2(:,4), 'linewidth', 2);
108 grid on
109 xlabel('time (s)')
110 ylabel('p (rad/s)')
111 title('change in roll rate over time')
112 legend( 'NL', 'L');
113 hold off
114 %
115 subplot(4,2,4);
116 plot(t1, X(:,5), 'linewidth', 2);
117 hold on

```

```

118 plot(t2, X2(:,5), 'linewidth', 2);
119 grid on
120 xlabel('time (s)')
121 ylabel('q (rad/s)')
122 title('change in pitch rate over time')
123 legend( 'NL', 'L');
124 hold off
125 %
126 subplot(4,2,5);
127 plot(t1, X(:,6), 'linewidth', 2);
128 hold on
129 plot(t2, X2(:,6), 'linewidth', 2);
130 grid on
131 xlabel('time (s)')
132 ylabel('r (rad/s)')
133 title('change in yaw rate over time')
134 legend( 'NL', 'L');
135 hold off
136 %
137 subplot(4,2,6);
138 plot(t1, X(:,1), 'linewidth', 2);
139 hold on
140 plot(t2, X2(:,1), 'linewidth', 2);
141 grid on
142 xlabel('time (s)')
143 ylabel('u (m/s)')
144 title('change in u over time')
145 legend( 'NL', 'L');
146 hold off
147 %
148 subplot(4,2,7);
149 plot(t1, X(:,2), 'linewidth', 2);
150 hold on
151 plot(t2, X2(:,2), 'linewidth', 2);
152 grid on
153 xlabel('time (s)')
154 ylabel('v (m/s)')
155 title('change in v over time')
156 legend( 'NL', 'L');
157 hold off
158 %
159 subplot(4,2,8);
160 plot(t1, X(:,3), 'linewidth', 2);
161 hold on
162 plot(t2, X2(:,3), 'linewidth', 2);
163 grid on
164 xlabel('time (s)')
165 ylabel('w (m/s)')
166 title('change in w over time')
167 legend( 'NL', 'L');
168 hold off
169 %% elevation is 5
170 tspan = linspace(0,5); % time vector
171 Pertubations = zeros(1, 3); % No perturbation
172 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
173 conditions = zeros(1, 12); % initialize conditions vector (very large)
174 conditions(8) = deg2rad(5);
175 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
176 % initialize variables
177 t1 = 0;
178 t2 = 0;
179 X = 0;
180 X2 = 0;
181
182 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
   tolerance of 1e-8 is met
183 [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 sgttitle('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)')

```

```

251 ylabel('u (m/s)')
252 title('change in u over time')
253 legend( 'NL', 'L');
254 hold off
255 %
256 subplot(4,2,7);
257 plot(t1, X(:,2), 'linewidth', 2);
258 hold on
259 plot(t2, X2(:,2), 'linewidth', 2);
260 grid on
261 xlabel('time (s)')
262 ylabel('v (m/s)')
263 title('change in v over time')
264 legend( 'NL', 'L');
265 hold off
266 %
267 subplot(4,2,8);
268 plot(t1, X(:,3), 'linewidth', 2);
269 hold on
270 plot(t2, X2(:,3), 'linewidth', 2);
271 grid on
272 xlabel('time (s)')
273 ylabel('w (m/s)')
274 title('change in w over time')
275 legend( 'NL', 'L');
276 hold off
277
278 %% azimuth is 5
279 tspan = linspace(0,5); % time vector
280 Pertubations = zeros(1, 3); % No perturbation
281 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
282 conditions = zeros(1, 12); % initialize conditions vector (very large)
283 conditions(9) = deg2rad(5);
284 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
285 % initialize variables
286 t1 = 0;
287 t2 = 0;
288 X = 0;
289 X2 = 0;
290
291 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
   tolerance of 1e-8 is met
292
293 [t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
   options); % nonlinear ODE
294
295 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
   options); %Linear ODE
296
297 figure()
298 sgttitle('Deviation by +5^\circ in azimuth, Nonlinear and Linear'); subplot(4,2,1);
299 plot(t1, X(:,7), 'linewidth', 2);
300 hold on
301 plot(t2, X2(:,7), 'linewidth', 2);
302 grid on
303 xlabel('time (s)')
304 ylabel('\phi (rad)')
305 title('change in bank over time')
306 legend( 'NL', 'L');
307 hold off
308 %
309 subplot(4,2,2);
310 plot(t1, X(:,8), 'linewidth', 2);
311 hold on
312 plot(t2, X2(:,8), 'linewidth', 2);
313 grid on
314 xlabel('time (s)')
315 ylabel('\theta (rad)')

```

```

316 title('change in elevation over time')
317 legend( 'NL', 'L');
318 hold off
319 %
320 subplot(4,2,3);
321 plot(t1, X(:,4), 'linewidth', 2);
322 hold on
323 plot(t2, X2(:,4), 'linewidth', 2);
324 grid on
325 xlabel('time (s)')
326 ylabel('p (rad/s)')
327 title('change in roll rate over time')
328 legend( 'NL', 'L');
329 hold off
330 %
331 subplot(4,2,4);
332 plot(t1, X(:,5), 'linewidth', 2);
333 hold on
334 plot(t2, X2(:,5), 'linewidth', 2);
335 grid on
336 xlabel('time (s)')
337 ylabel('q (rad/s)')
338 title('change in pitch rate over time')
339 legend( 'NL', 'L');
340 hold off
341 %
342 subplot(4,2,5);
343 plot(t1, X(:,6), 'linewidth', 2);
344 hold on
345 plot(t2, X2(:,6), 'linewidth', 2);
346 grid on
347 xlabel('time (s)')
348 ylabel('r (rad/s)')
349 title('change in yaw rate over time')
350 legend( 'NL', 'L');
351 hold off
352 %
353 subplot(4,2,6);
354 plot(t1, X(:,1), 'linewidth', 2);
355 hold on
356 plot(t2, X2(:,1), 'linewidth', 2);
357 grid on
358 xlabel('time (s)')
359 ylabel('u (m/s)')
360 title('change in u over time')
361 legend( 'NL', 'L');
362 hold off
363 %
364 subplot(4,2,7);
365 plot(t1, X(:,2), 'linewidth', 2);
366 hold on
367 plot(t2, X2(:,2), 'linewidth', 2);
368 grid on
369 xlabel('time (s)')
370 ylabel('v (m/s)')
371 title('change in v over time')
372 legend( 'NL', 'L');
373 hold off
374 %
375 subplot(4,2,8);
376 plot(t1, X(:,3), 'linewidth', 2);
377 hold on
378 plot(t2, X2(:,3), 'linewidth', 2);
379 grid on
380 xlabel('time (s)')
381 ylabel('w (m/s)')
382 title('change in w over time')
383 legend( 'NL', 'L');

```

```

384 hold off
385 %% .01 roll rate
386 tspan = linspace(0,5); % time vector
387 Pertubations = zeros(1, 3); % No perturbation
388 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
389 conditions = zeros(1, 12); % initialize conditions vector (very large)
390 conditions(4) = 0.1;
391 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
392
393 t1 = 0;
394 t2 = 0;
395 X = 0;
396 X2 = 0;
397
398 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
               % tolerance of 1e-8 is met
399
400 [t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
               options); % nonlinear ODE
401
402 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
               options); %Linear ODE
403
404 figure()
405 sgttitle('Deviation by +0.1 rad/sec in roll rate, Linear and Nonlinear');
406 subplot(4,2,1);
407 plot(t1, X(:,7), 'linewidth', 2);
408 hold on
409 plot(t2, X2(:,7), 'linewidth', 2);
410 grid on
411 xlabel('time (s)')
412 ylabel('\phi (rad)')
413 title('change in bank over time')
414 legend( 'NL', 'L');
415 hold off
416 %
417 subplot(4,2,2);
418 plot(t1, X(:,8), 'linewidth', 2);
419 hold on
420 plot(t2, X2(:,8), 'linewidth', 2);
421 grid on
422 xlabel('time (s)')
423 ylabel('\theta (rad)')
424 title('change in elevation over time')
425 legend( 'NL', 'L');
426 hold off
427 %
428 subplot(4,2,3);
429 plot(t1, X(:,4), 'linewidth', 2);
430 hold on
431 plot(t2, X2(:,4), 'linewidth', 2);
432 grid on
433 xlabel('time (s)')
434 ylabel('p (rad/s)')
435 title('change in roll rate over time')
436 legend( 'NL', 'L');
437 hold off
438 %
439 subplot(4,2,4);
440 plot(t1, X(:,5), 'linewidth', 2);
441 hold on
442 plot(t2, X2(:,5), 'linewidth', 2);
443 grid on
444 xlabel('time (s)')
445 ylabel('q (rad/s)')
446 title('change in pitch rate over time')
447 legend( 'NL', 'L');
448 hold off

```

```

449 %
450 subplot(4,2,5);
451 plot(t1, X(:,6), 'linewidth', 2);
452 hold on
453 plot(t2, X2(:,6), 'linewidth', 2);
454 grid on
455 xlabel('time (s)')
456 ylabel('r (rad/s)')
457 title('change in yaw rate over time')
458 legend( 'NL', 'L');
459 hold off
460 %
461 subplot(4,2,6);
462 plot(t1, X(:,1), 'linewidth', 2);
463 hold on
464 plot(t2, X2(:,1), 'linewidth', 2);
465 grid on
466 xlabel('time (s)')
467 ylabel('u (m/s)')
468 title('change in u over time')
469 legend( 'NL', 'L');
470 hold off
471 %
472 subplot(4,2,7);
473 plot(t1, X(:,2), 'linewidth', 2);
474 hold on
475 plot(t2, X2(:,2), 'linewidth', 2);
476 grid on
477 xlabel('time (s)')
478 ylabel('v (m/s)')
479 title('change in v over time')
480 legend( 'NL', 'L');
481 hold off
482 %
483 subplot(4,2,8);
484 plot(t1, X(:,3), 'linewidth', 2);
485 hold on
486 plot(t2, X2(:,3), 'linewidth', 2);
487 grid on
488 xlabel('time (s)')
489 ylabel('w (m/s)')
490 title('change in w over time')
491 legend( 'NL', 'L');
492 hold off
493 %
494 %% .01 pitch rate
495 tspan = linspace(0,5); % time vector
496 Pertubations = zeros(1, 3); % No perturbation
497 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
498 conditions = zeros(1, 12); % initialize conditions vector (very large)
499 conditions(5) = 0.1;
500 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
501
502 t1 = 0;
503 t2 = 0;
504 X = 0;
505 X2 = 0;
506
507 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
      tolerance of 1e-8 is met
508
509 [t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
      options); % nonlinear ODE
510
511 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
      options); %Linear ODE
512
513 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);

```

```

582 plot(t1, X(:,2), 'linewidth', 2);
583 hold on
584 plot(t2, X2(:,2), 'linewidth', 2);
585 grid on
586 xlabel('time (s)')
587 ylabel('v (m/s)')
588 title('change in v over time')
589 legend( 'NL', 'L');
590 hold off
591 %
592 subplot(4,2,8);
593 plot(t1, X(:,3), 'linewidth', 2);
594 hold on
595 plot(t2, X2(:,3), 'linewidth', 2);
596 grid on
597 xlabel('time (s)')
598 ylabel('w (m/s)')
599 title('change in w over time')
600 legend( 'NL', 'L');
601 hold off
602 %% .01 yaw rate
603 tspan = linspace(0,5); % time vector
604 Pertubations = zeros(1, 3); % No perturbation
605 TrimForces = ones(1, 4) * m * g / 4; % forces required by each motor to maintain hover
606 conditions = zeros(1, 12); % initialize conditions vector (very large)
607 conditions(6) = 0.1;
608 conditions(12) = -1; % set down direction to 1 to make signs correct for plotting
609
610 t1 = 0;
611 t2 = 0;
612 X = 0;
613 X2 = 0;
614
615 options = odeset('Events', @StopFnct, 'RelTol', 1e-8); % stop function that ends ODE when a
   tolerance of 1e-8 is met
616
617 [t1, X] = ode45(@(t, F)Specs2LB4NLC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
   options);
618
619 [t2, X2] = ode45(@(t, F)Specs2LB4LC(t, F, TrimForces, Pertubations, givens), tspan, conditions,
   options);
620
621
622 figure()
623 sgttitle('Deviation by +0.1 rad/sec in yaw rate, Linear and Nonlinear');
624 subplot(4,2,1);
625 plot(t1, X(:,7), 'linewidth', 2);
626 hold on
627 plot(t2, X2(:,7), 'linewidth', 2);
628 grid on
629 xlabel('time (s)')
630 ylabel('\phi (rad)')
631 title('change in bank over time')
632 legend( 'NL', 'L');
633 hold off
634 %
635 subplot(4,2,2);
636 plot(t1, X(:,8), 'linewidth', 2);
637 hold on
638 plot(t2, X2(:,8), 'linewidth', 2);
639 grid on
640 xlabel('time (s)')
641 ylabel('\theta (rad)')
642 title('change in elevation over time')
643 legend( 'NL', 'L');
644 hold off
645 %
646 subplot(4,2,3);

```

```

647 plot(t1, X(:,4), 'linewidth', 2);
648 hold on
649 plot(t2, X2(:,4), 'linewidth', 2);
650 grid on
651 xlabel('time (s)')
652 ylabel('p (rad/s)')
653 title('change in roll rate over time')
654 legend( 'NL', 'L');
655 hold off
656 %
657 subplot(4,2,4);
658 plot(t1, X(:,5), 'linewidth', 2);
659 hold on
660 plot(t2, X2(:,5), 'linewidth', 2);
661 grid on
662 xlabel('time (s)')
663 ylabel('q (rad/s)')
664 title('change in pitch rate over time')
665 legend( 'NL', 'L');
666 hold off
667 %
668 subplot(4,2,5);
669 plot(t1, X(:,6), 'linewidth', 2);
670 hold on
671 plot(t2, X2(:,6), 'linewidth', 2);
672 grid on
673 xlabel('time (s)')
674 ylabel('r (rad/s)')
675 title('change in yaw rate over time')
676 legend( 'NL', 'L');
677 hold off
678 %
679 subplot(4,2,6);
680 plot(t1, X(:,1), 'linewidth', 2);
681 hold on
682 plot(t2, X2(:,1), 'linewidth', 2);
683 grid on
684 xlabel('time (s)')
685 ylabel('u (m/s)')
686 title('change in u over time')
687 legend( 'NL', 'L');
688 hold off
689 %
690 subplot(4,2,7);
691 plot(t1, X(:,2), 'linewidth', 2);
692 hold on
693 plot(t2, X2(:,2), 'linewidth', 2);
694 grid on
695 xlabel('time (s)')
696 ylabel('v (m/s)')
697 title('change in v over time')
698 legend( 'NL', 'L');
699 hold off
700 %
701 subplot(4,2,8);
702 plot(t1, X(:,3), 'linewidth', 2);
703 hold on
704 plot(t2, X2(:,3), 'linewidth', 2);
705 grid on
706 xlabel('time (s)')
707 ylabel('w (m/s)')
708 title('change in w over time')
709 legend( 'NL', 'L');
710 hold off

```

Question 4 Script

```

1 % This script answers question 4 by plotting the experimental data from
2 % 11:37, 2/14/2020, and plotting the simulated linearized feedback control
3 % simulation on top for comparison
4 % Author: Benjiman Smith
5 % Collaborators: E. Owen, I. Quezada
6 % Date: 2/20/2020
7 %
8
9 clear all
10 close all
11 clc
12 m = 0.068; % mass of the drone [kg]
13 r = 0.06; % body to motor distance [m]
14 k = 0.0024; % [Nm/N]
15 rad = r/sqrt(2); % [m]
16 g = 9.81; % gravity [m/s^2]
17 alpha = 2e-6; % [N/(m/s)^2]
18 eta = 1e-3; % [N/(rad/s)^2]
19 Ix = 6.8e-5; % moment of inertia in the x direction [kg m^2]
20 Iy = 9.2e-5; % moment of inertia in the y direction [kg m^2]
21 Iz = 1.35e-4; % moment of inertia in the z direction [kg m^2]
22 %% lateral control
23 Lambda1 = -2; % 1st lambda value, found from lab calculations
24 Lambda2 = -20; % 2nd lambda value is 10x Lambda 1, in order to make lambda 1 dominant
25 Ix = 6.8e-5; % moment of inertia in the x direction [kg m^2]
26 Iy = 9.2e-5; % Moment of inertia abt y (kgm^2)
27
28 syms K1 K2 % symbollically solving for K1 and K2
29 eqn1 = Lambda1^2 +Lambda1*(K1/Ix) + (K2/Ix) == 0; % 1st eigenvalue solution
30 eqn2 = Lambda2^2 +Lambda2*(K1/Ix) + (K2/Ix) == 0; % 2nd eigenvalue solution
31
32
33 [A,B] = equationsToMatrix([eqn1, eqn2], [K1, K2]); % convert both the equations and the unknowns
into independent vectors
34
35 Solution = linsolve(A,B); % solve the system of equations
36 Solution = double(Solution); % convert symbolic solution vector to vector of doubles
37 K1 = Solution(1); % bank control
38 K2 = Solution(2); % bank control
39
40 %% Longitudinal Control
41 syms K3 K4 % symbollically solving for K3 and K4
42 eqn3 = Lambda1^2 +Lambda1*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
43 eqn4 = Lambda2^2 +Lambda2*(K3/Iy) + (K4/Iy) == 0; % 1st eigenvalue solution
44
45
46 [C,D] = equationsToMatrix([eqn3, eqn4], [K3, K4]);
47
48 ForceVect2 = linsolve(C,D);
49 ForceVect2 = double(ForceVect2);
50 K3 = ForceVect2(1); % elevation control
51 K4 = ForceVect2(2); % elevation control
52
53 givens = [alpha eta Ix Iy Iz m r k rad g K1 K2 K3 K4]; % givens vector
54 F = m*g;
55
56
57 load('RSdata_1147.mat'); % load in data
58 % loading in of parameters
59 theta = rt_estim.signals.values(:,5);
60 phi = rt_estim.signals.values(:,6);
61 u = rt_estim.signals.values(:,7);
62 v = rt_estim.signals.values(:,8);
63 w = rt_estim.signals.values(:,9);
64 p = rt_estim.signals.values(:,10);
65 q = rt_estim.signals.values(:,11);
66 r = rt_estim.signals.values(:,12);
67 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 sgttitle('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)')

```

```

134 ylabel('r (rad/s)')
135 title('change in yaw rate over time')
136 legend('Experimental Data', 'Simulated Data');
137 hold off
138 subplot(4,2,6);
139 plot(times, u);
140 hold on
141 plot(t1 + subtracttime, X(:,1), 'linewidth', 2);
142 grid on
143 xlabel('time (s)')
144 ylabel('u (m/s)')
145 title('change in u over time')
146 legend('Experimental Data', 'Simulated Data');
147 hold off
148 subplot(4,2,7);
149 plot(times, v);
150 hold on
151 plot(t1 + subtracttime, X(:,2), 'linewidth', 2);
152 grid on
153 xlabel('time (s)')
154 ylabel('v (m/s)')
155 title('change in v over time')
156 legend('Experimental Data', 'Simulated Data');
157 hold off
158 subplot(4,2,8);
159 plot(times, w);
160 hold on
161 plot(t1 + subtracttime, X(:,3), 'linewidth', 2);
162 grid on
163 xlabel('time (s)')
164 ylabel('w (m/s)')
165 title('change in w over time')
166
167 legend('Experimental Data', 'Simulated Data');
168 hold off

```

linearized feedback control ODE function

```

1 % linear ODE Function
2 %Benjiman Smith
3 %02/13/2020
4 function dydt = Specs2LB4LC(t, Conditions, Force, Disturb, givens)
5 % define givens
6 m = givens(6); % mass of the drone [kg]
7 r = givens(7); % body to motor distance [m]
8 k = givens(8); % [Nm/N]
9 R = givens(9); % [m]
10 g = givens(10);% gravity [m/s^2]
11 alpha = givens(1); % [N/(m/s)^2]
12 eta = givens(2); % [N/(rad/s)^2]
13 I_x = givens(3); % moment of inertia in the x direction [kg m^2]
14 I_y = givens(4); % moment of inertia in the y direction[kg m^2]
15 I_z = givens(5); % moment of inertia in the z direction[kg m^2]
16 k_1 = givens(11);
17 k_2 = givens(12);
18 k_3 = givens(13);
19 k_4 = givens(14);
20
21 % define conditions vector
22 deltau = Conditions(1); % inertial velocity in the u direction in body coordinates [m/s]
23 deltav = Conditions(2); % inertial velocity in the v direction in body coordinates [m/s]
24 deltaw = Conditions(3); % inertial velocity in the w direction in body coordinates [m/s]
25 deltap = Conditions(4); % inertial angular velocity in the p direction in body coordinates [
26 % rad/s]
26 deltaq = Conditions(5); % inertial angular velocity in the q direction in body coordinates [
27 % rad/s]
27 deltar = Conditions(6); % inertial angular velocity in the r direction in body coordinates [

```

```

    rad/s]
28 deltaphi = Conditions(7); % bank [rad]
29 deltatheta = Conditions(8); % elevation [rad]
30 psi = Conditions(9); % azimuth [rad]
31 x_E = Conditions(10); % position vector in the x direction in inertial coordinates [m]
32 y_E = Conditions(11); % position vector in the y direction in inertial coordinates [m]
33 z_E = Conditions(12); % position vector in the z direction in inertial coordinates [m]
34 f1 = Force(1); % trim force exerted by motor 1 [N]
35 f2 = Force(2); % trim force exerted by motor 1 [N]
36 f3 = Force(3); % trim force exerted by motor 1 [N]
37 f4 = Force(4); % trim force exerted by motor 1 [N]
38
39 % Aerodynamic/Control Moments and Forces
40 Laero = - alpha^2 * deltap^2 * sign(deltap); % p component of the aerodynamic moments
41 Maero = - alpha^2 * deltaq^2 * sign(deltaq); % q component of the aerodynamic moments
42 Naero = - alpha^2 * deltar^2 * sign(deltar); % w component of the aerodynamic moments
43 Lcontrol = ((f1 + f2) - (f3 + f4)) * R; % p component of the control moments
44 Mcontrol = ((f3 + f2) - (f1 + f4)) * R; % q component of the control moments
45 Ncontrol = -0.004*deltar; % w component of the aerodynamic moments
46 deltaL = Laero + Lcontrol; % L moment sum (abt x axis)
47 deltaM = Maero + Mcontrol; % M moment sum (abt y axis)
48 deltaN = Naero + Ncontrol; % N moment sum (abt Z axis)
49
50 if t > 2 && t < 2.5
51     deltaL = deltaL + Disturb(2); % Disturbed L moment (abt x axis)
52     deltaM = deltaM + Disturb(1); % Disturbed M moment (abt y axis)
53     deltaN = deltaN + Disturb(3); % Disturbed N moment (abt z axis)
54 end
55 Xaero = - eta^2 * deltau^2 * sign(deltau); % x component of aerodynamic force
56 Yaero = - eta^2 * deltav^2 * sign(deltav); % y component of aerodynamic force
57 Zaero = - eta^2 * deltar^2 * sign(deltar); % z component of aerodynamic force
58 Xcontrol = 0; % x control
59 Ycontrol = 0; % y control
60 Zcontrol = -sum(Force); % gravitational force counteraction
61 deltaX = Xaero + Xcontrol; % sum of x forces
62 deltaY = Yaero + Ycontrol; % sum of y forces
63 deltaZ = 0; % sum of z forces
64 deltapdot = ((-k_1*deltap)/I_x) - ((k_2*deltaphi)/I_x);
65 deltaqdot = ((-k_3*deltaq)/I_y) - ((k_4*deltatheta)/I_y); % pitch rate derivative
66 deltardot = deltaN/I_z; % yaw rate derivative
67
68 dOmega_bdt = [deltapdot, deltaqdot, deltardot]';
69 deltaudot = -g*deltatheta; % 4.7,1 acceleration in the x axis
70 deltavdot = g*deltaphi; % acceleration in the y axis
71 deltawdot = deltaZ/m; % acceleration in the z axis
72 dVbdt = [deltaudot, deltavdot, deltawdot]';
73 xdot = deltau;
74 ydot = deltav;
75 zdot = deltar;
76 dVEdt = [xdot, ydot, zdot]';
77 phidot = deltap; % bank roc (pg 104)
78 thetadot = deltaq; % elevation roc (pg 104)
79 psidot = deltar; % azimuth roc (pg 104)
80 dEuldt = [phidot, thetadot, psidot]';
81 dydt = [dVbdt; dOmega_bdt; dEuldt; dVEdt];
82
83
84
85 end

```

non-linearized feedback control ODE function

```

1 % Nonlinear ODE Function with control
2 %Benjiman Smith
3 %02/13/2020
4 function dydt = Specs2LB4NLC(t, Conditions, Force, Disturb, givens)
5 % define givens

```

```

6      m = givens(6);          % mass of the drone [kg]
7      r = givens(7);          % body to motor distance [m]
8      k = givens(8);          % [Nm/N]
9      R = givens(9);          % [m]
10     g = givens(10);% gravity [m/s^2]
11     alpha = givens(1);    % [N/(m/s)^2]
12     eta = givens(2);    % [N/(rad/s)^2]
13     I_x = givens(3);    % moment of inertia in the x direction [kg m^2]
14     I_y = givens(4);    % moment of inertia in the y direction[kg m^2]
15     I_z = givens(5);    % moment of inertia in the z direction[kg m^2]
16     k_1 = givens(11);
17     k_2 = givens(12);
18     k_3 = givens(13);
19     k_4 = givens(14);
20     % define conditions vector
21     u = Conditions(1); % inertial velocity in the u direction in body coordinates [m/s]
22     v = Conditions(2); % inertial velocity in the v direction in body coordinates [m/s]
23     w = Conditions(3); % inertial velocity in the w direction in body coordinates [m/s]
24     p = Conditions(4); % inertial angular velocity in the p direction in body coordinates [rad/s]
25     q = Conditions(5); % inertial angular velocity in the q direction in body coordinates [rad/s]
26     r = Conditions(6); % inertial angular velocity in the r direction in body coordinates [rad/s]
27     phi = Conditions(7); % bank [rad]
28     theta = Conditions(8); % elevation [rad]
29     psi = Conditions(9); % azimuth [rad]
30     x_E = Conditions(10); % position vector in the x direction in inertial coordinates [m]
31     y_E = Conditions(11); % position vector in the y direction in inertial coordinates [m]
32     z_E = Conditions(12); % position vector in the z direction in inertial coordinates [m]
33
34
35     % Aerodynamic/Control Moments and Forces
36     Laero = - alpha^2 * p^2 * sign(p); % p component of the aerodynamic moments
37     Maero = - alpha^2 * q^2 * sign(q); % q component of the aerodynamic moments
38     Naero = - alpha^2 * r^2 * sign(r); % w component of the aerodynamic moments
39     Lcontrol = -k_1*p - k_2*phi;
40     Mcontrol = -k_3*q - k_4*theta;
41     Ncontrol = -0.004*r; % w component of the aerodynamic moments
42     L = Laero + Lcontrol; % L moment sum (abt x axis)
43     M = Maero + Mcontrol; % M moment sum (abt y axis)
44     N = Naero + Ncontrol; % N moment sum (abt Z axis)
45
46     %           L = p*.004 - L ; % Disturbed L moment (abt x axis)
47     %           M = q*.004 - M; % Disturbed M moment (abt y axis)
48     %           N = r*0.004 - N; % Disturbed N moment (abt z axis)
49
50     Xaero = - eta^2 * u^2 * sign(u); % x component of aerodynamic force
51     Yaero = - eta^2 * v^2 * sign(v); % y component of aerodynamic force
52     Zaero = - eta^2 * w^2 * sign(w); % z component of aerodynamic force
53     Xcontrol = 0; % x control
54     Ycontrol = 0; % y control
55     Zcontrol = -sum(Force); % gravitational force counteraction
56     X = Xaero + Xcontrol; % sum of x forces
57     Y = Yaero + Ycontrol; % sum of y forces
58     Z = Zaero + Zcontrol; % sum of z forces
59     pdot = p*q*(I_y - I_z)/I_x + (1/I_x * L); % roll rate derrivative
60     qdot = p * r*(I_z - I_x)/I_y + (1/I_y * M); % pitch rate derrivative
61     rdot = p * q*(I_x - I_y)/I_z + (1/I_z * N); % yaw rate derrivative
62
63     dOmega_bdt = [pdot, qdot, rdot]';
64     udot = r*v - q*w - g*sin(theta) + 1/m * X; % 4.7,1 acceleration in the x axis
65     vdot = p*w - r*u + g*sin(phi)*cos(theta) + 1/m * Y; % acceleration in the y axis
66     wdot = q*u - p*v + g*cos(phi)*cos(theta) + 1/m * Z; % acceleration in the z axis
67     dVbdt = [udot, vdot, wdot]';
68     xdot = u * cos(theta)*cos(psi) + v * (sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi)) + w *
69         (cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi));
70     ydot = u * cos(theta)*sin(psi) + v * (sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi)) + w *
71         (cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi));
72     zdot = -u * sin(theta) + v * sin(phi)*cos(theta) + w * cos(phi)*cos(theta);
73     dVEdt = [xdot, ydot, zdot]';

```

```

72     phidot = p + (q*sin(phi)+r*cos(phi))*tan(theta); % bank roc (pg 104)
73     thetadot = q*cos(phi) - r*sin(phi); % elevation roc (pg 104)
74     psidot = q*sin(phi)*sec(theta) + r*cos(phi)*sec(theta); % azimuth roc (pg 104)
75
76     dEuldt = [phidot, thetadot, psidot]';
77     dydt = [dvbdt; domega_bdt; dEuldt; dvEdt];
78
79
80
81 end

```

Stop Function

```

1 % This function is used by ODE45 in options to end the drones
2 % path after a certain tolerance of values is reached
3 %
4 % Author: Benjiman Smith
5 % Collaborators: E. Owen, I. Quezada
6 % Date: 1/26/2020
7 %
8 function [value, isTerm, direction] = StopFnct(t, F)
9 value = F(12);
10 isTerm = 1;
11 direction = [];
12 end

```