

UNIVERSITY OF COLORADO - BOULDER

ASEN 3128 : LAB 4

SUBMITTED OCTOBER 23RD, 2020

Lab 4: Quadrotor Control

Author:

Emerson BEINHAEUER^a

Author:

Luke ENGELKEN^b

Author:

Connor O'REILLY^c

Professor:

Professor FREW

Author:

Thyme ZUSCHLAG^d

^a108185329

^b107165781

^c107054811

^d109221211



College of Engineering & Applied Science
UNIVERSITY OF COLORADO **BOULDER**

Table of Contents

	Page
I Problem 1	2
II Problem 2 and 3 Plots and Discussion	3
II.A Deviation by +5 deg in roll	3
II.B Deviation by +5 deg in pitch	4
II.C Deviation by +0.1 rad/sec in roll rate	5
II.D Deviation by +0.1 rad/sec in pitch rate	6
III Problem 4	8
IV Problem 5	10
IV.A Lateral Model	10
IV.B Longitudinal Model	11
V Problem 6	12
Participation	14
Appendices	14
Code	14

I. Problem 1

Using the following equations the maximum eigenvalue for a given A state matrix can be solved for.

$$\tau = \frac{-1}{\lambda_i} \quad (1)$$

$$(\mathbf{A} - \lambda_i)\mathbf{v}_i = 0 \quad (2)$$

For questions 2 and 3 a time constant of 0.5 seconds is required which corresponds to a dominating eigenvalue term of -2. The matrix A can then be solved as it is only a function of k1 given an initial guess of k2 and using this process the following control laws were derived to stabilize roll and pitch attitude.

$$\Delta L_c = -0.004\Delta p - 0.0021\Delta\phi \quad (3)$$

$$\Delta M_c = -0.004\Delta q - 0.0021\Delta\theta \quad (4)$$

II. Problem 2 and 3 Plots and Discussion

A. Deviation by +5 deg in roll

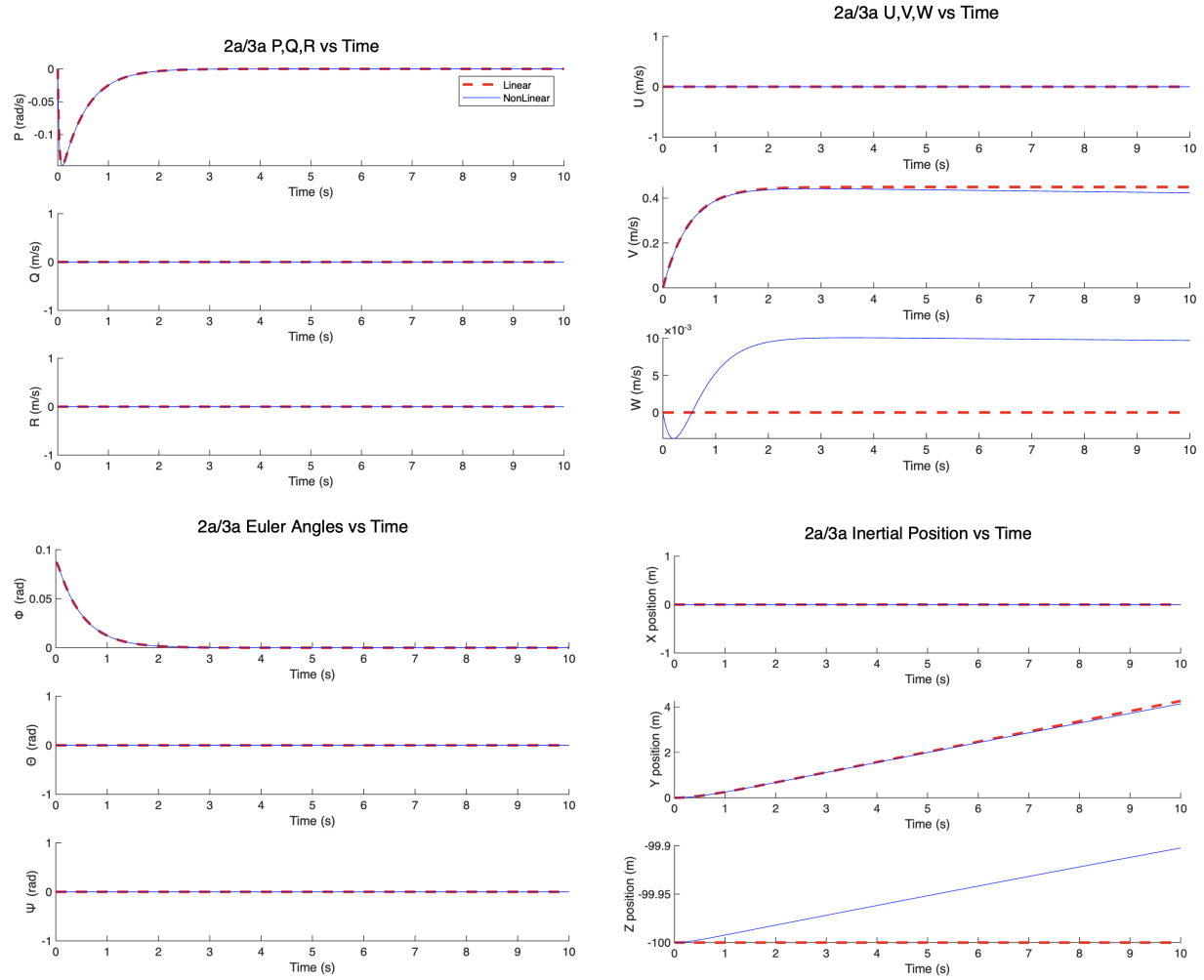


Fig. 1 Linearized vs. Nonlinearized Behaviour 2a/3a

B. Deviation by +5 deg in pitch

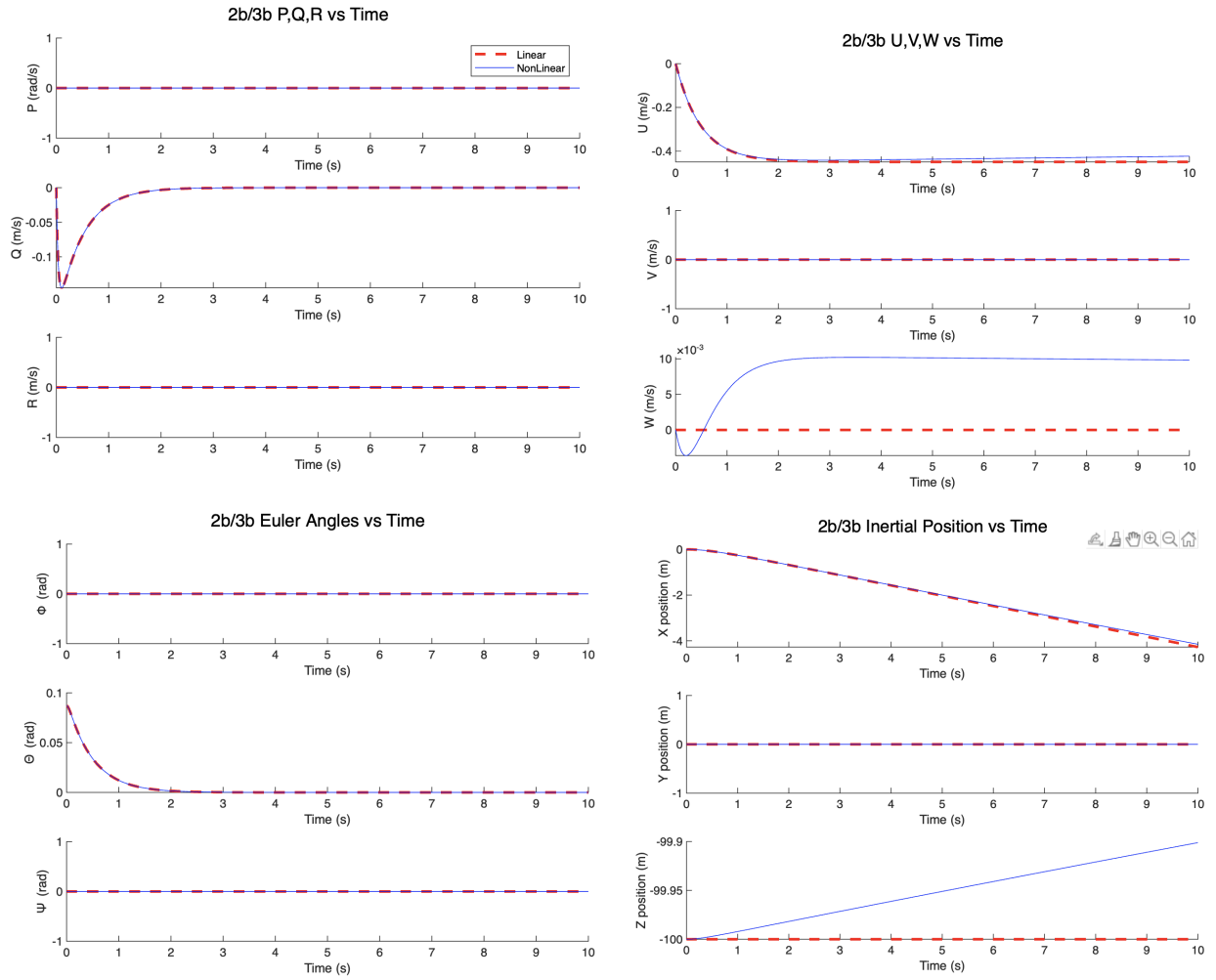


Fig. 2 Linearized vs. Nonlinearized Behaviour 2b/3b

C. Deviation by +0.1 rad/sec in roll rate

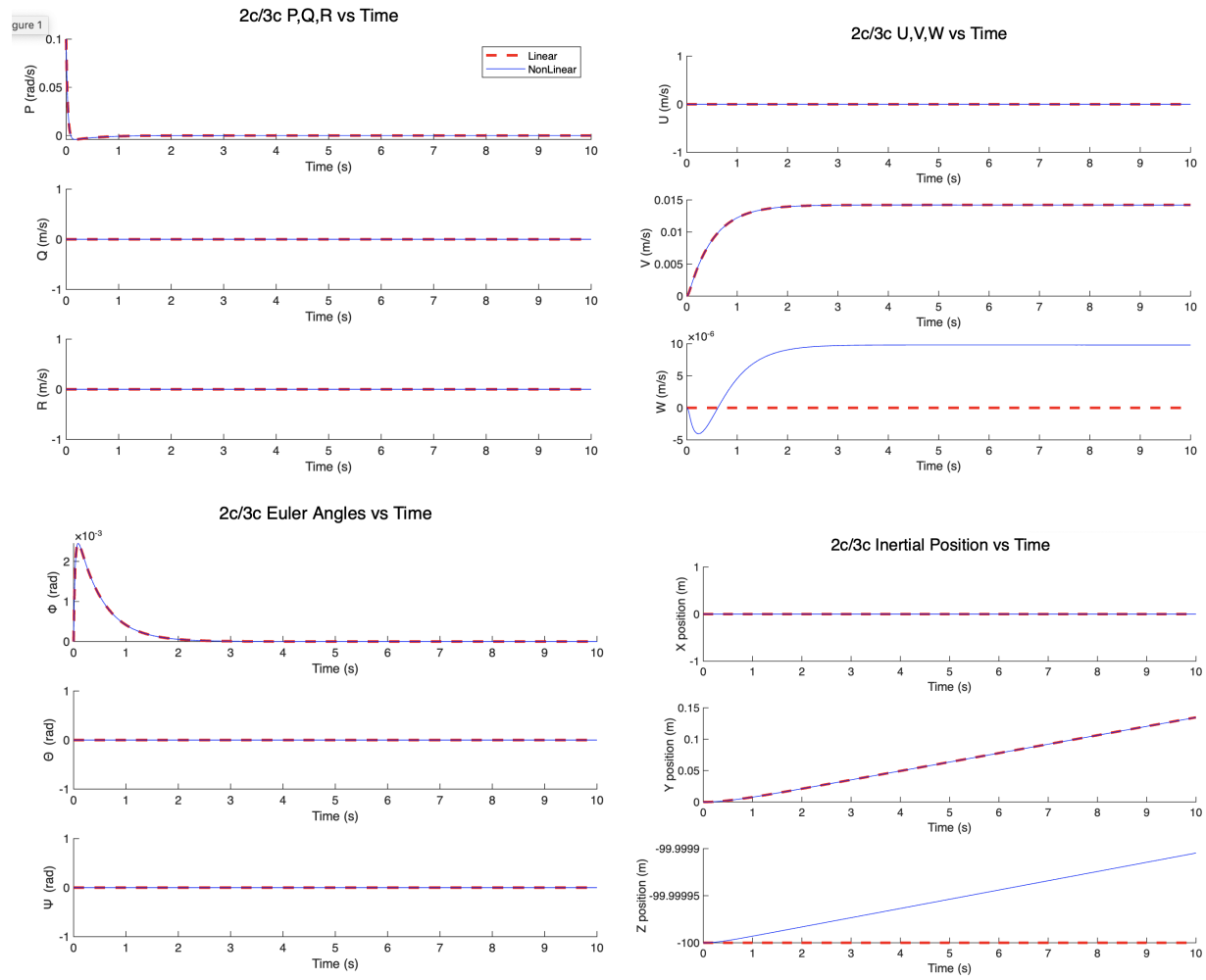


Fig. 3 Linearized vs. Nonlinearized Behaviour 2c/3c

D. Deviation by +0.1 rad/sec in pitch rate

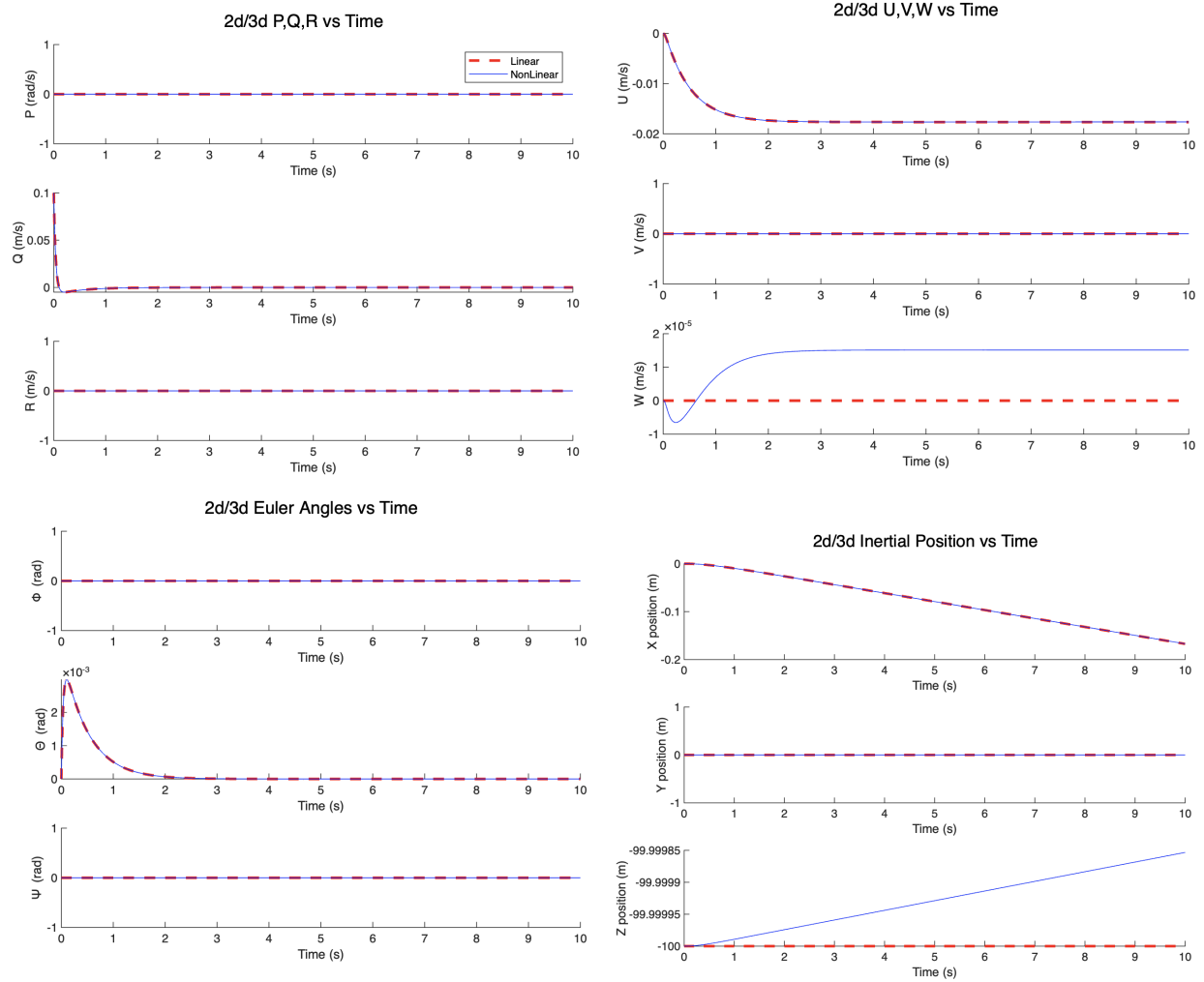


Fig. 4 Linearized vs. Nonlinearized Behaviour 2d/3d

For this particular model, the lateral variables are accounted for in the feedback control loop. The main purpose of feedback control is to keep the controlled variables close to their set point. So this means moving the system from some initial rate to zero.

As seen in the plots, the linear and non linear lines are almost identical with the exception of a few plots. This all corresponds to the expected behavior from the linearized modal response theory. The linearized equations are based on small angle approximations and due to the actual small angles used in this lab, the linearized model captures true behavior.

As depicted in the plots, the models for linear velocity and non linear produce very similar outputs. Roll and angle roll rate are controlled and driven back to zero, however neither lateral nor vertical speeds are zero. This is because the

feedback control is not implemented on speed or position.

For example, for scenarios a and b, uvw don't all go back to zero, which is why the y and z positions for a then x and z positions for b vary. As shown in Figures 1 and 2, the uvw terms converge on a value but do not converge back to zero because there is no feedback control implemented.

The quadrotor was also expected to fall for a deviation in pitch, because the sum of the four motor forces is set to equal the weight of the aircraft. So when it is tilted, the forces equal less than the total weight and the aircraft falls. So because the quadrotor was falling after it was tilted, it created speed downwards and sideways, and this is all expected motion.

When roll and pitch are deviated, the quadrotor drifts sideways as seen in v and y position plots in Figure 1 and u and x position plots in Figure 2. This drift is due to drag. We do account for drag in the nonlinear system, and this drag is so small that it takes a long time to the aircraft. These linear equations do not account for drag, which describes the small discrepancy between these V vs. Time, Y Position vs. Time, U vs. Time, and X Position vs. Time plots.

As seen in the plots, the only time when there is a large discrepancy between the linear and nonlinear models are w and z position. This is consistent for all cases a-d. The linearized model is based off the small disturbance theory. Due to this, when the aircraft is disrupted and has to use control to make adjustments, these adjustments will be so minimal, they will fall under small disturbances, and will be taken to be zero. Also, this particular lab only focuses on longitudinal and lateral variables, so w and the z position are not controlled in this model.

Steady hover is now a stable flight condition due to the control feedback loop. For example, if the drone experiences a deviation in pitch angle instead of falling out of equilibrium the control law adjusts the angle and stabilizes the drone.

III. Problem 4

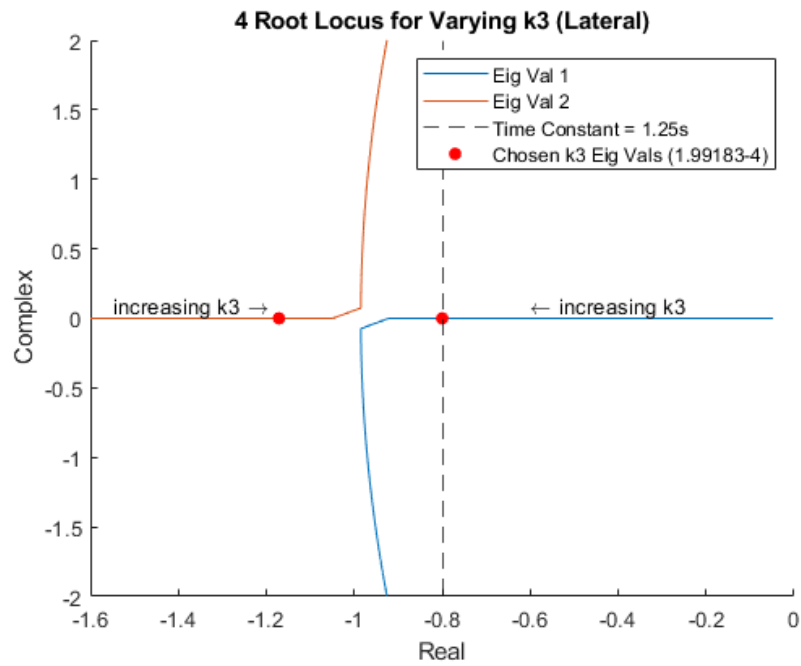


Fig. 5 Lateral

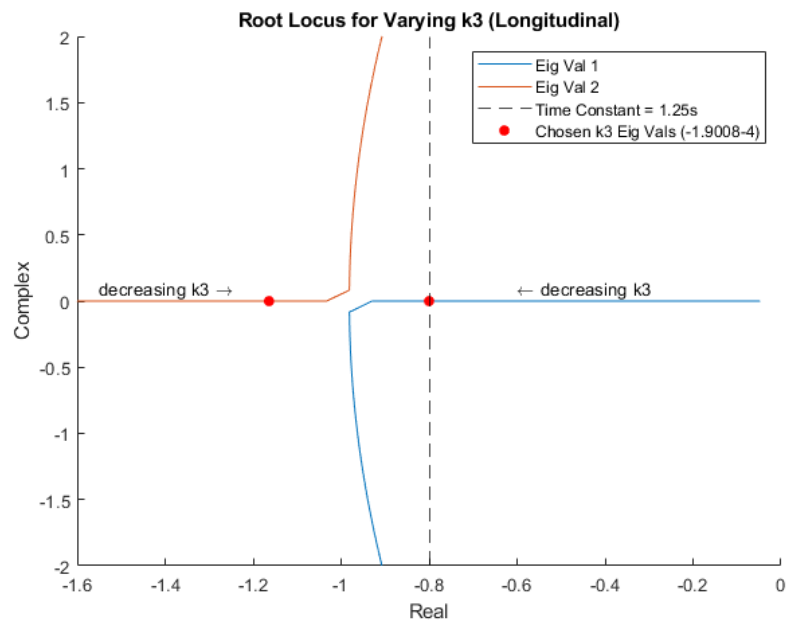


Fig. 6 Longitudinal

Following the same procedure as Problem 1, the maximum eigenvalue for our a given state matrix A can be determined,

$$\tau = \frac{-1}{\lambda_i} \quad (5)$$

$$(\mathbf{A} - \lambda_i)\mathbf{v}_i = 0 \quad (6)$$

a time constant less than 1.25 [sec] was required, corresponding to a dominating eigenvalue of -0.8. Using the feedback control gains $K_1 = 0.004$ and $K_2 = 0.0021$ designed in problem 1 the tracking control gain K_3 was determined by solving the state matrix A. For the closed loop three-state longitudinal system K_3 was computed to be -1.9008×10^{-4} and for the lateral system K_3 was determined to be 1.99183×10^{-4} . Following, the builtin MATLAB function *eig()* was used to plot the locus of the eigenvalues for the lateral and longitudinal systems for a range of K_3 values. For the longitudinal system, K_3 ranged from -1.9008×10^{-4} to -1.9008×10^{-2} and for the lateral system K_3 ranged from 1.99183×10^{-4} to 1.99183×10^{-2} . Using the root locus for both systems the determined gain values satisfied the design objectives. Using the determind gain values the following control laws were derived

Longitudinal System:

$$\Delta L_c = -0.004\Delta p - 0.0021\Delta\phi - (1.9008 \times 10^{-4})(\Delta v_r - \Delta v) \quad (7)$$

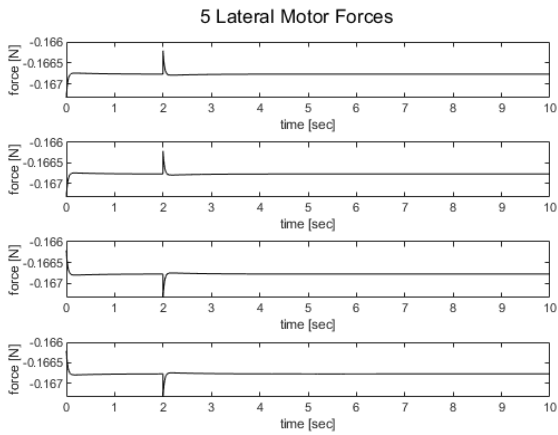
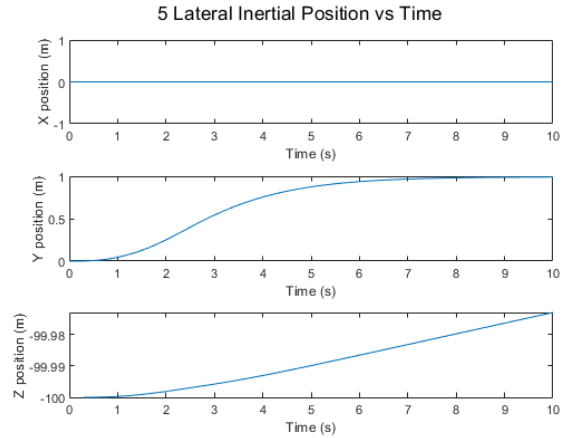
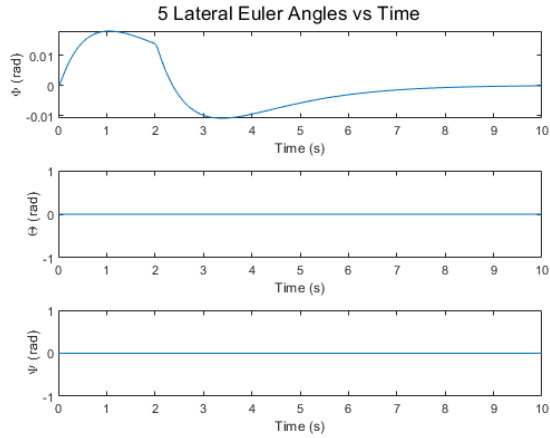
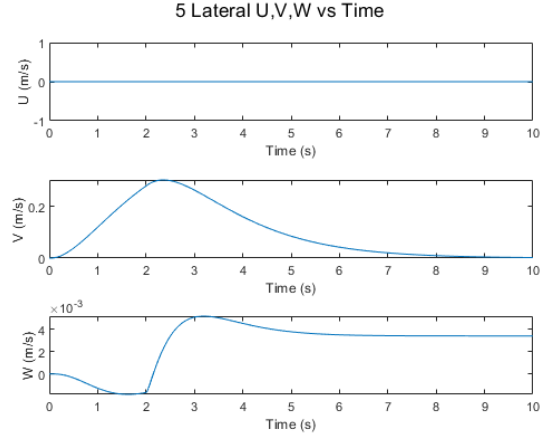
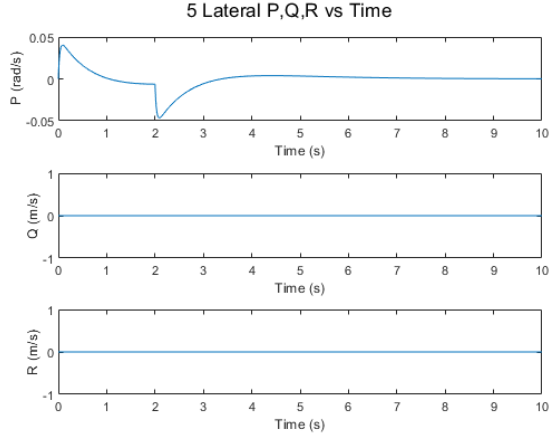
Lateral System:

$$\Delta M_c = -0.004\Delta q - 0.0021\Delta\theta + (1.99183 \times 10^{-4})(\Delta u_r - \Delta u) \quad (8)$$

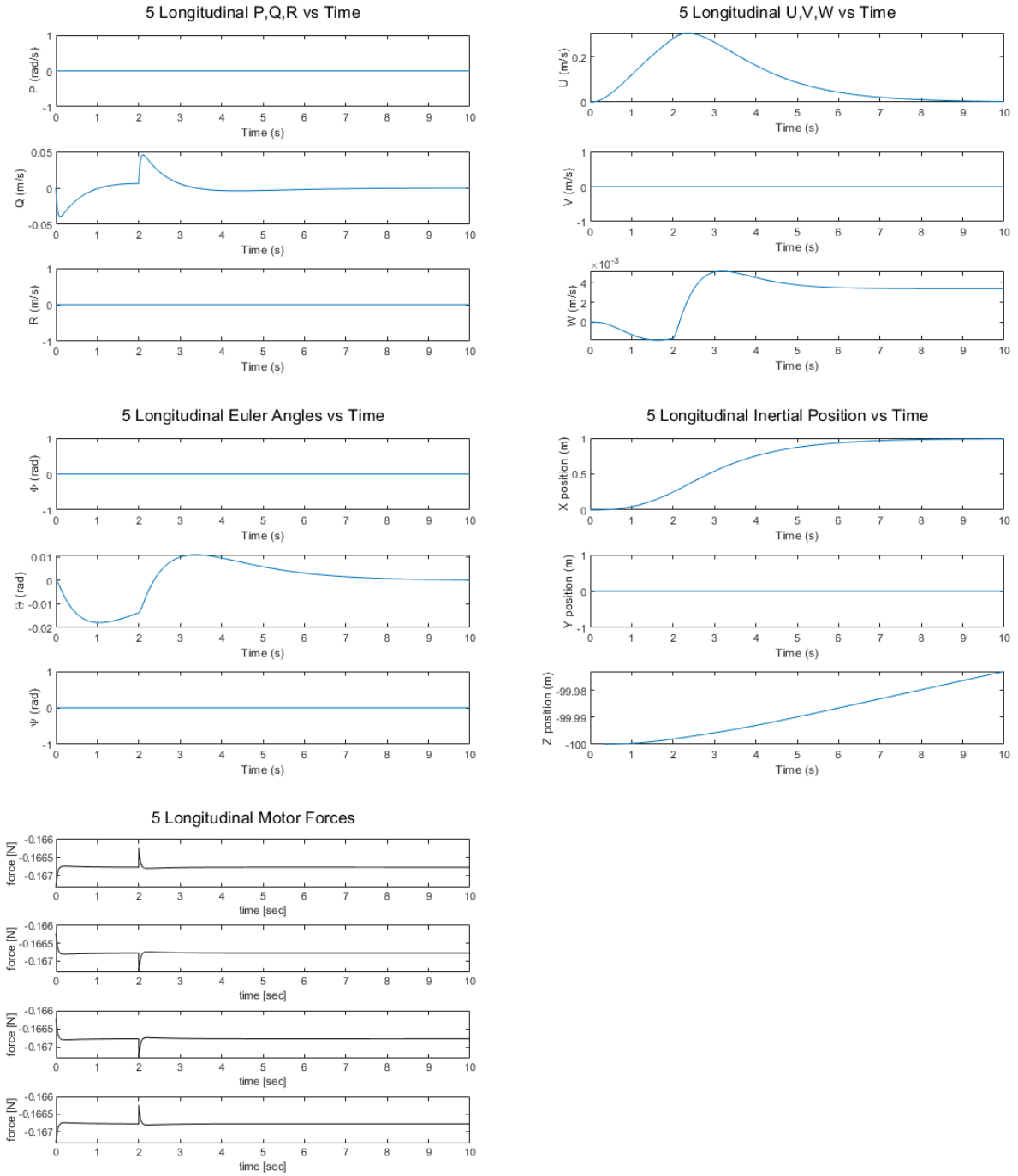
IV. Problem 5

Using the lateral and longitudinal control laws designed in problem 4 the models are simulated and shown in the following plots.

A. Lateral Model



B. Longitudinal Model



The simulated models behave as predicted because the u and v components of velocity approach the desired value of 0.5m/s for $t < 2\text{s}$ and then they go back to 0m/s as the control law no longer acts.

V. Problem 6

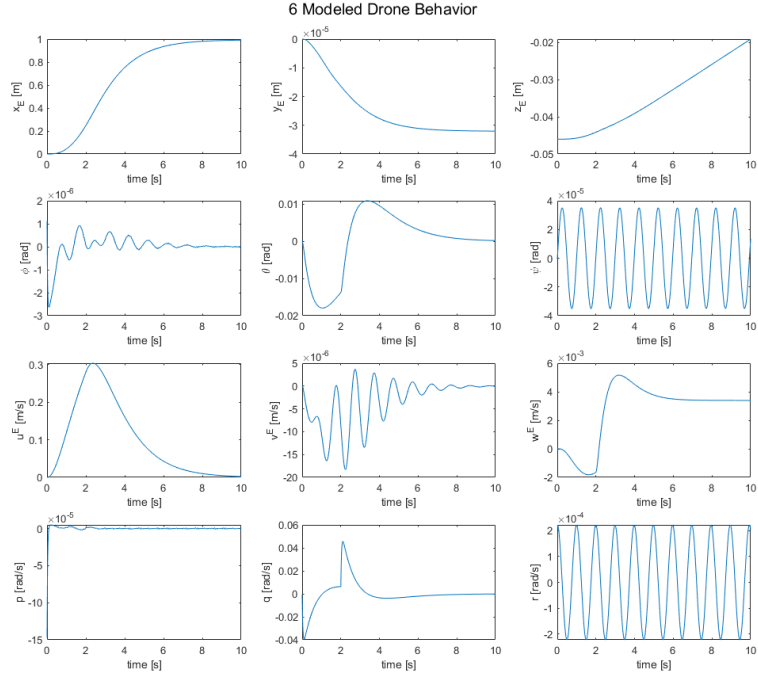


Fig. 7 Modeled Drone Behavior

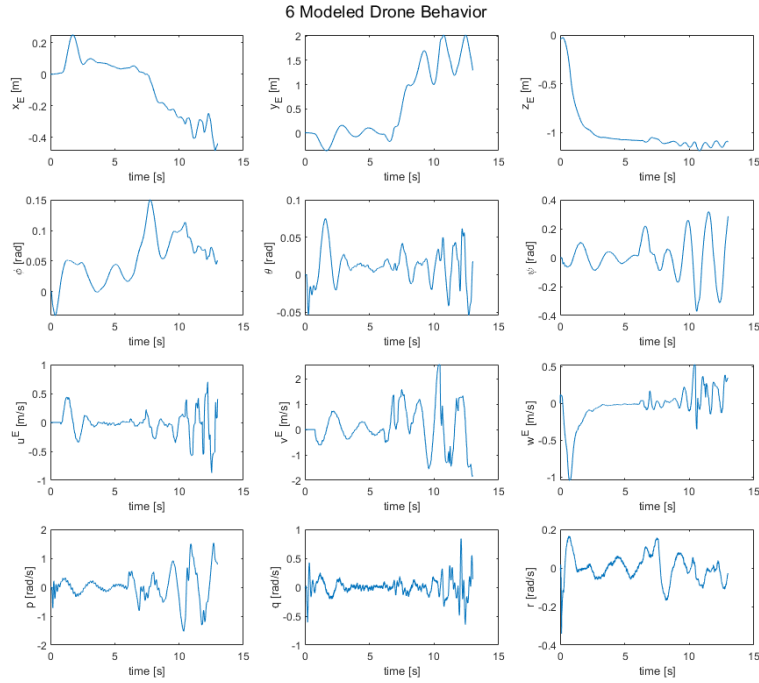


Fig. 8 Experimental Drone Behavior

The experimental data is quite different from the data simulated with the non-linear control model and this is due to the imperfect control forces applied by the motors. Drone motors can experience voltage spikes which could increase the error in the modeled drone behavior which then propagates through the simulation and results in more inaccuracy. Another error could stem from level of accuracy of the accelerometer and gyroscope readings. Without proper filtering these sensors could experience large amounts of error which again could propagate through the model and result in more error.

Participation Table

	Plan	Model	Experiment	Results	Report	Code	ACK
Emerson Beinhauer	2	1	N/A	1	1	1	EB
Luke Engelken	1	2	N/A	2	1	2	LE
Connor O'Rilly	2	1	N/A	1	1	1	CTO
Thyme Zuschlag	1	1	N/A	1	2	1	TZ

2 = Lead, 1 = Participate, 0 = Not involved, for each element. X = acknowledged by each team member.

Appendix A: MATLAB Code

```

1 function [motor_forces] = ComputeMotorForces(Zc, Lc, Mc, Nc, R, km)
2 %Function explanation stuff
3 A = [-1 -1 -1 -1; -R/sqrt(2) -R/sqrt(2) R/sqrt(2) R/sqrt(2);...
4      R/sqrt(2) -R/sqrt(2) -R/sqrt(2) R/sqrt(2); km -km km -km];
5 motor_forces = inv(A) * [Zc; Lc; Mc; Nc];
6 % motor_forces = motor_forces';
7 end

1 % Luke Engelken
2 % ASEN 3128
3 % probl.m
4 % Created: 9/11/20
5
6 %To run, in command window type "prob(<insert quetsion number)" and you
7 %will see plots
8 %ex: prob('1a')
9
10 function [] = drone(prob)
11     %% Declare Constants
12     m = 0.068; %mass of drone (kg)
13     r = 0.06; %radial distance from cg to motor (m)
14     km = 0.0024; %control moment coefficient (N*m/N)
15     Ix = 5.8e-5; %x-axis moment of inertia (kg*m^2)

```

```

16  Iy = 7.2e-5; %y-axis moment of inertia (kg*m^2)
17  Iz = 1e-4; %z-axis moment of inertia (kg*m^2)
18  v = 1e-3; %aerodynamic force coefficient (N/(m/s^2))
19  mu = 2e-6; %N*m/(rad/s^2)
20  g = 9.81; %m/s^2
21  %xstate vector: [x; y; z; phi; theta; psi; xdot; ydot; zdot; phidot;
22  %thetadot; psidot] (integral of eom state vector)
23  tspan = [0 10];
24  switch prob
25      case '1a'
26          %Prob 1a - steady state hover, no drag
27          f1 = g*m/4; %Assume -force up
28          f2 = f1;
29          f3 = f1;
30          f4 = f1;
31          pinit = [0; 0; 100; 0; 0; 0; 0; 0; 0; 0; 0; 0];
32          [tx,x1] = ode45(@(t,x)
                        objectEOMnodrag(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
33      case '2a'
34          %Prob 2a - steady state hover, with drag
35          f1 = 2;%g*m/4; %Assume -force up
36          f2 = f1;
37          f3 = f1;
38          f4 = f1;
39          pinit = [0; 0; -10; 0; 0; 0; 0; 0; 0; 0; 0; 0];
40          [tx,x1] = ode45(@(t,x)
                        objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
41      case '2b'
42          %Prob 2b - constant y velocity = 5m/s, psi = 0
43          syms phi
44          %eqn = 5*v*5*cos(phi)+m*g*sin(phi)^3+m*g*sin(phi)*cos(phi)^2 == 0;
45          eqn = atan(phi) == v*25/(m*g);
46          phi = solve(eqn);
47          phi = eval(phi(1,1)); %3.1041
48          %phi = 2.12;

```



```

49 pinit = [0; 0; 0; phi; 0; 0; 0; 5*cos(phi); -5*sin(phi); 0; 0; 0];
50 Zc = m*g*(cos(pinit(4))+sin(pinit(4))^2/cos(pinit(4))); %-0.667080298221064
51 %Zc = 25*v/sin(phi); %-.6681
52 %Zc = .6665;
53 f1 = Zc/4;
54 f2 = f1;
55 f3 = f2;
56 f4 = f3;
57 [tx,x1] = ode45(@ (t,x)
    objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
58 case '2c'
59 %Prob 2c - constant y velocity = -5m/s, psi = 90deg
60 syms phi
61 %eqn = 5*v*5*cos(phi)+m*g*sin(phi)^3+m*g*sin(phi)*cos(phi)^2 == 0;
62 eqn = atan(phi) == v*25/(m*g);
63 phi = solve(eqn);
64 phi = eval(phi(1,1)); %3.1041
65 %phi = 2.12;
66 pinit = [0; 0; 0; 0; -phi; pi/2; 5*cos(phi); 0; -5*sin(phi); 0; 0; 0];
67 Zc = m*g*(cos(pinit(5))+sin(pinit(5))^2/cos(pinit(5))); %-0.667080298221064
68 %Zc = 25*v/sin(phi); %-.6681
69 %Zc = .6665;
70 f1 = Zc/4;
71 f2 = f1;
72 f3 = f2;
73 f4 = f3;
74 [tx,x1] = ode45(@ (t,x)
    objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
75 case '3'
76 %Prob 2a - steady state hover, with drag
77 f1 = g*m/4; %Assume -force up
78 f2 = f1;
79 f3 = f1;
80 f4 = f1;
81 pinit = [0; 0; -10; 0; 0; 0; 0; 0; 0; 0; 0; 0];

```

```

82     [tx,x1] = ode45(@(t,x)
        objectEOM_disturb(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
83
84     case '5'
85         %     f1 = -m*g/4; %Assume -force up
86         %     f2 = f1;
87         %     f3 = f1;
88         %     f4 = f1;
89         pinit = [0; 0; -100; 5*pi/180; 0; 0; 0; 0; 0; 0; 0; 0];
90         [tx,x1] = ode45(@(t,x)
            LinearQuadControlQ2(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);
91
92     end
93
94     xval = x1(:,1);
95     yval = x1(:,2);
96     zval = x1(:,3);
97     phival = x1(:,4);
98     thetaval = x1(:,5);
99     psival = x1(:,6);
100    uval = x1(:,7);
101    vval = x1(:,8);
102    wval = x1(:,9);
103    pval = x1(:,10);
104    qval = x1(:,11);
105    rval = x1(:,12);
106
107    figure(1)
108    drone_plot(x1,tx,'pqr',prob);
109
110    figure(2)
111    drone_plot(x1,tx,'uvw',prob);
112
113    figure(3)
114    drone_plot(x1,tx,'euler',prob);

```

```

115     drone_plot(x1,tx,'xyz',prob);
116
117     figure(5)
118     plot3(xval,yval,zval);
119     grid on
120 end

1 function [] = drone_plot(x1,tx,type,prob)
2 %DRONE_PLOT: Plots the 4 subplots of inertial position, Euler angles,
3 %body acceleration, and Euler rates
4 xval = x1(:,1);
5     yval = x1(:,2);
6     zval = x1(:,3);
7     phival = x1(:,4);
8     thetaval = x1(:,5);
9     psival = x1(:,6);
10    uval = x1(:,7);
11    vval = x1(:,8);
12    wval = x1(:,9);
13    pval = x1(:,10);
14    qval = x1(:,11);
15    rval = x1(:,12);
16
17 switch type
18     case 'pqr'
19         subplot(3,1,1)
20         plot(tx,pval,'LineWidth',5);
21         xlabel('Time (s)');
22         ylabel('P (rad/s)');
23         subplot(3,1,2)
24         plot(tx,qval,'LineWidth',5);
25         xlabel('Time (s)');
26         ylabel('Q (m/s)');
27         subplot(3,1,3)
28         plot(tx,rval,'LineWidth',5);

```

```

29     xlabel('Time (s)');
30     ylabel('R (m/s)');
31     sgtitle(sprintf('%s P,Q,R vs Time',prob));
32 case 'uvw'
33     subplot(3,1,1)
34     plot(tx,uval,'LineWidth',5);
35     xlabel('Time (s)');
36     ylabel('U (m/s)');
37     if prob == '2c'
38         axis([0 10 4.5 5.5]);
39     end
40     subplot(3,1,2)
41     plot(tx,vval,'LineWidth',5);
42     xlabel('Time (s)');
43     ylabel('V (m/s)');
44     if prob == '2b'
45         axis([0 10 4.9 5.1]);
46     end
47     subplot(3,1,3)
48     plot(tx,wval,'LineWidth',5);
49     xlabel('Time (s)');
50     ylabel('W (m/s)');
51     if prob == '2b' | prob == '2c'
52         axis([0 10 -1 0]);
53     end
54     sgtitle(sprintf('%s U,V,W vs Time',prob));
55 case 'euler'
56     subplot(3,1,1)
57     plot(tx,phival,'LineWidth',5);
58     xlabel('Time (s)');
59     ylabel('\Phi (rad)');
60     subplot(3,1,2)
61     plot(tx,thetaval,'LineWidth',5);
62     xlabel('Time (s)');
63     ylabel('\Theta (rad)');

```

```

64     subplot(3,1,3)
65     plot(tx,psival,'LineWidth',5);
66     xlabel('Time (s)');
67     ylabel('\Psi (rad)');
68     sgtitle(sprintf('%s Euler Angles vs Time',prob));
69 case 'xyz'
70     subplot(3,1,1)
71     plot(tx,xval,'LineWidth',5);
72     xlabel('Time (s)');
73     ylabel('X position (m)');
74     subplot(3,1,2)
75     plot(tx,yval,'LineWidth',5);
76     xlabel('Time (s)');
77     ylabel('Y position (m)');
78     subplot(3,1,3)
79     plot(tx,zval,'LineWidth',5);
80     if prob == '2b' | prob == '2c'
81         axis([0 10 -1 1]);
82     end
83     xlabel('Time (s)');
84     ylabel('Z position (m)');
85     sgtitle(sprintf('%s Inertial Position vs Time',prob));
86
87
88 end

```

```

1 %Author: Thyme Zuschlag
2 %Class: 3128 Lab 4
3 %Date: 10/21
4 %Purpose: Plot nicely on subplots with appropriate titles
5
6 function [] = drone_plot2(x1,tx,x2, tx2, type,prob)
7 %DRONE_PLOT: Plots the 4 subplots of inertial position, Euler angles,
8 %body acceleration, and Euler rates
9 xval = x1(:,1);

```

```

10     yval = x1(:,2);
11     zval = x1(:,3);
12     phival = x1(:,4);
13     thetaval = x1(:,5);
14     psival = x1(:,6);
15     uval = x1(:,7);
16     vval = x1(:,8);
17     wval = x1(:,9);
18     pval = x1(:,10);
19     qval = x1(:,11);
20     rval = x1(:,12);
21
22     xval2 = x2(:,1);
23     yval2 = x2(:,2);
24     zval2 = x2(:,3);
25     phival2 = x2(:,4);
26     thetaval2 = x2(:,5);
27     psival2 = x2(:,6);
28     uval2 = x2(:,7);
29     vval2 = x2(:,8);
30     wval2 = x2(:,9);
31     pval2 = x2(:,10);
32     qval2 = x2(:,11);
33     rval2 = x2(:,12);
34
35 switch type
36     case 'pqr'
37         subplot(3,1,1)
38         hold on
39         plot(tx,pval,'--r', "LineWidth", 2);
40         plot(tx2,pval2, 'b');
41         xlabel('Time (s)');
42         ylabel('P (rad/s)');
43         legend('Linear', 'NonLinear');
44         subplot(3,1,2)

```

```

45     hold on
46     plot(tx,qval,'--r', "LineWidth", 2);
47     plot(tx2,qval2, 'b');
48     xlabel('Time (s)');
49     ylabel('Q (m/s)');
50     subplot(3,1,3)
51     hold on
52     plot(tx,rval,'--r', "LineWidth", 2);
53     plot(tx2,rval2, 'b');
54     xlabel('Time (s)');
55     ylabel('R (m/s)');
56     sgtitle(sprintf('%s 2d/3d P,Q,R vs Time',prob));
57
58 case 'uvw'
59     subplot(3,1,1)
60     hold on
61     plot(tx,uval,'--r', "LineWidth", 2);
62     plot(tx2,uval2, 'b');
63     xlabel('Time (s)');
64     ylabel('U (m/s)');
65     if prob == '2c'
66         axis([0 10 4.5 5.5]);
67     end
68     subplot(3,1,2)
69     hold on
70     plot(tx,vval,'--r', "LineWidth", 2);
71     plot(tx2,vval2, 'b');
72     xlabel('Time (s)');
73     ylabel('V (m/s)');
74     if prob == '2b'
75         axis([0 10 4.9 5.1]);
76     end
77     subplot(3,1,3)
78     hold on
79     plot(tx,wval,'--r', "LineWidth", 2);

```

```

80     plot(tx2,wval2, 'b');
81     xlabel('Time (s)');
82     ylabel('W (m/s)');
83     if prob == '2b' | prob == '2c'
84         axis([0 10 -1 0]);
85     end
86     sgtitle(sprintf('%s 2d/3d U,V,W vs Time',prob));
87 case 'euler'
88     subplot(3,1,1)
89     hold on
90     plot(tx,phival,'--r', "LineWidth", 2);
91     plot(tx2,phival2, 'b');
92     xlabel('Time (s)');
93     ylabel('\Phi (rad)');
94     subplot(3,1,2)
95     hold on
96     plot(tx,thetaval,'--r', "LineWidth", 2);
97     plot(tx2,thetaval2, 'b');
98     xlabel('Time (s)');
99     ylabel('\Theta (rad)');
100    subplot(3,1,3)
101    hold on
102    plot(tx,psival,'--r', "LineWidth", 2);
103    plot(tx2,psival2, 'b');
104    xlabel('Time (s)');
105    ylabel('\Psi (rad)');
106    sgtitle(sprintf('%s 2d/3d Euler Angles vs Time',prob));
107 case 'xyz'
108     subplot(3,1,1)
109     hold on
110     plot(tx,xval,'--r', "LineWidth", 2);
111     plot(tx2,xval2, 'b');
112     xlabel('Time (s)');
113     ylabel('X position (m)');
114     subplot(3,1,2)

```



```

115     hold on
116     plot(tx,yval,'--r', "LineWidth", 2);
117     plot(tx2,yval2, 'b');
118     xlabel('Time (s)');
119     ylabel('Y position (m)');
120     subplot(3,1,3)
121     hold on
122     plot(tx,zval,'--r', "LineWidth", 2);
123     plot(tx2,zval2, 'b');
124     if prob == '2b' | prob == '2c'
125         axis([0 10 -1 1]);
126     end
127     xlabel('Time (s)');
128     ylabel('Z position (m)');
129     sgtitle(sprintf('%s 2d/3d Inertial Position vs Time',prob));
130
131
132 end

```

```

1 function [xstate] = LinearQuadControlQ2(t,x,m,r,km,Ix,Iy,Iz,v,mu)
2 %
3 %Inputs:
4 % t = time
5 % x = 12-dimension state vector includes the inertial velocity in
6 % inertial coordinates and the inertial position in inertial coordinates
7 % [x; y; z; phi; theta; psi; u; v; w; p; q; r]
8 % m = mass of drone (kg)
9 % r = radius of frame from motor to cg
10 % km = control moment coefficient
11 % Ix,Iy,Iz = moments about axis
12 % v = drag coefficient
13 % mu = moment coefficient
14 % f1,f2,f3,f4 = forces from motors
15 %
16 %Outputs:

```

```

17 % xdot = 12-dimension state vector includes inertial velocity in inertial
18 %   coordinates and the inertial acceleration in inertial coordinates
19 %   [xdot; ydot; zdot; phidot; thetadot; psidot; udot; vdot; wdot;
20 %   pdot; qdot; rdot]
21 %
22 %Methodology: Use Newton's second law F=ma to calculate the acceleration
23 %   and velocity at each point in time for ode45 to integrate to find
24 %   position. Drag, gravity and motor thrust are only forces acting on drone.
25 %
26
27 g = 9.81;
28
29 %Get IV's
30 x1 = x(1);
31 y1 = x(2);
32 z1 = x(3);
33 phil = x(4);
34 theta1 = x(5);
35 psil = x(6);
36 u1 = x(7);
37 v1 = x(8);
38 w1 = x(9);
39 p1 = x(10);
40 q1 = x(11);
41 r1 = x(12);
42 Zc = -m*g;
43
44 %Calculate moments
45 k1Lat = 0.0021;
46 k2Lat = 0.004;
47 k3Lat = 1.9118e-4;
48 k1Lon = 0.0021;
49 k2Lon = 0.004;
50 k3Lon = -1.9008e-4;
51 m_ctl(1) = -k1Lat*p1-k2Lat*phil;

```

```

52 m_ctl(2) = -k1Lon*q1-k2Lon*thetal;
53 m_ctl(3) = -0.004*psil;
54
55 Lc = m_ctl(1);
56 Mc = m_ctl(2);
57 Nc = m_ctl(3);
58
59 %Put back into ode45
60 xstate(1) = u1; %xdot
61 xstate(2) = v1; %ydot
62 xstate(3) = w1; %zdot
63 xstate(4) = p1; %phidot
64 xstate(5) = q1; %thetadot
65 xstate(6) = r1; %psidot
66 xstate(7) = -g*thetal; %udot
67 xstate(8) = g*phil; %vdot
68 xstate(9) = 0; %wdot
69 xstate(10) = 1/Ix*(Lc); %pdot
70 xstate(11) = 1/Iy*(Mc); %qdot
71 xstate(12) = 1/Ix*(Nc); %rdot
72 xstate = xstate';
73 end

1 function [xstate] = LinearQuadControlQ4(t,x,m,r,km,Ix,Iy,Iz,v,mu,vref)
2 %
3 %Inputs:
4 % t = time
5 % x = 12-dimension state vector includes the inertial velocity in
6 % inertial coordinates and the inertial position in inertial coordinates
7 % [x; y; z; phi; theta; psi; u; v; w; p; q; r]
8 % m = mass of drone (kg)
9 % r = radius of frame from motor to cg
10 % km = control moment coefficient
11 % Ix,Iy,Iz = moments about axis
12 % v = drag coefficient

```

```

13 % mu = moment coefficient
14 % f1,f2,f3,f4 = forces from motors
15 %
16 %Outputs:
17 % xdot = 12-dimension state vector includes inertial velocity in inertial
18 %   coordinates and the inertial acceleration in inertial coordinates
19 %   [xdot; ydot; zdot; phidot; thetadot; psidot; udot; vdot; wdot;
20 %   pdot; qdot; rdot]
21 %
22 %Methodology: Use Newton's second law  $F=ma$  to calculate the acceleration
23 %   and velocity at each point in time for ode45 to integrate to find
24 %   position. Drag, gravity and motor thrust are only forces acting on drone.
25 %
26
27 g = 9.81;
28
29 %Get IV's
30 x1 = x(1);
31 y1 = x(2);
32 z1 = x(3);
33 phi1 = x(4);
34 theta1 = x(5);
35 psi1 = x(6);
36 u1 = x(7);
37 v1 = x(8);
38 w1 = x(9);
39 p1 = x(10);
40 q1 = x(11);
41 r1 = x(12);
42 Zc = -m*g;
43 ur = vref(1);
44 vr = vref(2);
45
46 %K vals from q1
47 k1Lat = 0.0021;

```

```

48 k2Lat = 0.004;
49 k3Lat = 1.9118e-4;
50 k1Lon = 0.0021;
51 k2Lon = 0.004;
52 k3Lon = -1.9008e-4;
53 m_ctl(1) = -k1Lat*p1-k2Lat*phi1+k3Lat*(vr-v1);
54 m_ctl(2) = -k1Lon*q1-k2Lon*theta1+k3Lon*(ur-u1);
55 m_ctl(3) = -0.004*psi1;
56
57 Lc = m_ctl(1);
58 Mc = m_ctl(2);
59 Nc = m_ctl(3);
60
61 %Put back into ode45
62 xstate(1) = u1; %xdot
63 xstate(2) = v1; %ydot
64 xstate(3) = w1; %zdot
65 xstate(4) = p1; %phidot
66 xstate(5) = q1; %thetadot
67 xstate(6) = r1; %psidot
68 xstate(7) = -g*theta1; %udot
69 xstate(8) = g*phi1; %vdot
70 xstate(9) = 0; %wdot
71 xstate(10) = 1/Ix*(Lc); %pdot
72 xstate(11) = 1/Iy*(Mc); %qdot
73 xstate(12) = 1/Ix*(Nc); %rdot
74 xstate = xstate';
75 end

1 clc
2 clear all
3 close all
4
5 %% Q2 and 3: Find k vals using the A matrix
6 %time constant of 0.5s -> eig = -2

```

```

7  Ix = 5.8e-5;
8  Iy = 7.2e-5;
9  k2Latreal = 0.004;
10 k1Latreal = (-k2Latreal/Ix - 4)*-Ix/2;
11 % k2Lat = linspace(k2Latreal/100,k2Latreal,100);
12 k2Lonreal = 0.004;
13 k1Lonreal = (-k2Lonreal/Iy - 4)*-Iy/2;
14 g = 9.81;
15 %%Lat k vals
16 % for i = 1:100
17 %   k1Lat = (-k2(i)/Ix - 4)*-Ix/2;
18 %   A = [0 1; -k2(i)/Ix -k1Lat/Ix];
19 %   eigval(i,:) = eig(A)';
20 % end
21 % A1 = [0 1; -k2Latreal/Ix -k1Latreal/Ix];
22 % eigreal = eig(A1)
23 %Lin k vals
24 % A1 = [0 1; -k2Lonreal/Iy -k1Lonreal/Iy];
25 % eigreal = eig(A1)
26
27 % %% Q4
28 bLat = -k2Latreal/Ix;
29 cLat = -k1Latreal/Ix;
30 bLon = -k2Lonreal/Iy;
31 cLon = -k1Lonreal/Iy;
32 k3Latreal = -Ix*(0.8*bLat-0.64*cLat-0.512)/9.81;
33 k3Lonreal = Iy*(0.8*bLon-0.64*cLon-0.512)/9.81;
34 aLatreal = -k3Latreal/Ix;
35 aLonreal = -k3Lonreal/Iy;
36 ALatreal = [0 g 0; 0 0 1; aLatreal bLat cLat];
37 ALonreal = [0 -g 0; 0 0 1; aLonreal bLon cLon];
38 eigsLat = eig(ALatreal);
39 eigsLon = eig(ALonreal)
40 k3Lat = linspace(k3Latreal/10,k3Latreal*10,1000);
41 aLat = -k3Lat/Ix;

```

```

42 for i = 1:1000
43     A = [0 g 0; 0 0 1; aLat(i) bLat cLat];
44     eigs = eig(A);
45     eigvalLat(i,:) = eigs';
46 end
47 figure(1)
48 hold on
49 axis([-1.6 0 -2 2]);
50 % plot(eigvalLat(:,1),0);
51 plot(eigvalLat(:,2));
52 plot(eigvalLat(:,3));
53 xline(-0.8,'--k');
54 plot(eigsLat(1),0,'r.','MarkerSize',20);
55 plot(eigsLat(2),0,'r.','MarkerSize',20);
56 plot(eigsLat(3),0,'r.','MarkerSize',20);
57 text(-1.55,.1,'increasing k3 \rightarrow');
58 text(-.6,.1,'\leftarrow increasing k3 ');
59 xlabel('Real');
60 ylabel('Complex');
61 title('4 Root Locus for Varying k3 (Lateral)');
62 legend('Eig Val 1','Eig Val 2','Time Constant = 1.25s',...
63     'Chosen k3 Eig Vals (1.99183-4)');
64
65 % k3Lon = linspace(k3Lonreal/10,k3Lonreal*10,1000);
66 % aLon = -k3Lon/Iy;
67 % for i = 1:1000
68 %     A = [0 -g 0; 0 0 1; aLon(i) bLon cLon];
69 %     eigs = eig(A);
70 %     eigvalLon(i,:) = eigs';
71 % end
72 % figure(2)
73 % hold on
74 % axis([-1.6 0 -2 2]);
75 % % plot(eigvalLat(:,1),0);
76 % plot(eigvalLon(:,2));

```

```

77 % plot(eigvalLon(:,3));
78 % xline(-0.8,'--k');
79 % plot(eigsLon(1),0,'r.','MarkerSize',20);
80 % plot(eigsLon(2),0,'r.','MarkerSize',20);
81 % plot(eigsLon(3),0,'r.','MarkerSize',20);
82 % text(-1.55,.1,'decreasing k3 \rightarrow');
83 % text(-.6,.1,'\leftarrow decreasing k3 ');
84 % xlabel('Real');
85 % ylabel('Complex');
86 % title('4 Root Locus for Varying k3 (Longitudinal)');
87 % legend('Eig Val 1','Eig Val 2','Time Constant = 1.25s',...
88 % 'Chosen k3 Eig Vals (-1.9008-4)');
89
90 %% Q5
91 % k2Latreal = 0.004;
92 % k1Latreal = (-k2Latreal/Ix - 4)*-Ix/2;
93 % k2Lonreal = 0.004;
94 % k1Lonreal = (-k2Lonreal/Iy - 4)*-Iy/2;
95 % bLat4 = -k2Latreal/Ix;
96 % cLat4 = -k1Latreal/Ix;
97 % bLon4 = -k2Lonreal/Iy;
98 % cLon4 = -k1Lonreal/Iy;
99 % k3Latreal = -Ix*(0.8*bLat4-0.64*cLat4-0.512)/9.81;
100 % k3Lonreal = Iy*(0.8*bLon4-0.64*cLon4-0.512)/9.81;
101
102 % ALatreal = [0 1 0 0; 0 0 g 0; 0 0 0 1; aLatreal bLat cLat dLat];
103 % ALonreal = [0 1 0 0; 0 0 -g 0; 0 0 0 1; aLonreal bLon cLon dLon];

1 function [REB] = R_eb(phi,theta,psi,units)
2 %switch if input is either rad or deg
3 switch units
4     case 'deg'
5         REB(1,1) = cosd(theta)*cosd(psi);
6         REB(1,2) = cosd(theta)*sind(psi);
7         REB(1,3) = -sind(theta);

```



```

8     REB(2,1) = sind(phi)*sind(theta)*cosd(psi)-cosd(phi)*sind(psi);
9     REB(2,2) = sind(phi)*sind(theta)*sind(psi)+cosd(phi)*cosd(psi);
10    REB(2,3) = sind(phi)*cosd(theta);
11    REB(3,1) = cosd(phi)*sind(theta)*cosd(psi)+sind(phi)*sind(psi);
12    REB(3,2) = cosd(phi)*sind(theta)*sind(psi)-sind(phi)*cosd(psi);
13    REB(3,3) = cosd(phi)*cosd(theta);
14    case 'rad'
15        phi = phi * (180/pi);
16        theta = theta * (180/pi);
17        psi = psi * (180/pi);
18        REB(1,1) = cosd(theta)*cosd(psi);
19        REB(1,2) = cosd(theta)*sind(psi);
20        REB(1,3) = -sind(theta);
21        REB(2,1) = sind(phi)*sind(theta)*cosd(psi)-cosd(phi)*sind(psi);
22        REB(2,2) = sind(phi)*sind(theta)*sind(psi)+cosd(phi)*cosd(psi);
23        REB(2,3) = sind(phi)*cosd(theta);
24        REB(3,1) = cosd(phi)*sind(theta)*cosd(psi)+sind(phi)*sind(psi);
25        REB(3,2) = cosd(phi)*sind(theta)*sind(psi)-sind(phi)*cosd(psi);
26        REB(3,3) = cosd(phi)*cosd(theta);
27    end
28    REB = inv(REB);
29
30 end

```

```

1 function [Tmat] = T(phi,theta,psi,units)
2 %T Summary of this function goes here
3 % Detailed explanation goes here
4 switch units
5     case 'deg'
6         Tmat(1,1) = 1;
7         Tmat(1,2) = sind(phi)*tand(theta);
8         Tmat(1,3) = cosd(phi)*tand(theta);
9         Tmat(2,1) = 0;
10        Tmat(2,2) = cosd(phi);
11        Tmat(2,3) = -sind(phi);

```

```

12     Tmat(3,1) = 0;
13     Tmat(3,2) = sind(phi)*secd(theta);
14     Tmat(3,3) = cosd(phi)*secd(theta);
15
16     case 'rad'
17         phi = phi * (180/pi);
18         theta = theta * (180/pi);
19         psi = psi * (180/pi);
20         Tmat(1,1) = 1;
21         Tmat(1,2) = sind(phi)*tand(theta);
22         Tmat(1,3) = cosd(phi)*tand(theta);
23         Tmat(2,1) = 0;
24         Tmat(2,2) = cosd(phi);
25         Tmat(2,3) = -sind(phi);
26         Tmat(3,1) = 0;
27         Tmat(3,2) = sind(phi)*secd(theta);
28         Tmat(3,3) = cosd(phi)*secd(theta);
29     end

```

```

1 % Luke Engelken
2 % ASEN 3128
3 % prob1.m
4 % Created: 9/11/20
5
6 %To run, in command window type "prob(<insert quetsion number)" and you
7 %will see plots
8 %ex: prob('1a')
9
10 function [] = updated_drone()
11     %% Declare Constants
12     m = 0.068; %mass of drone (kg)
13     r = 0.06; %radial distance from cg to motor (m)
14     km = 0.0024; %control moment coefficient (N*m/N)
15     Ix = 5.8e-5; %x-axis moment of inertia (kg*m^2)
16     Iy = 7.2e-5; %y-axis moment of inertia (kg*m^2)

```

```

17 Iz = 1e-4; %z-axis moment of inertia (kg*m^2)
18 v = 1e-3; %aerodynamic force coefficient (N/(m/s^2))
19 mu = 2e-6; %N*m/(rad/s^2)
20 g = 9.81; %m/s^2
21 %xstate vector: [x; y; z; phi; theta; psi; xdot; ydot; zdot; phidot;
22 %thetadot; psidot] (integral of eom state vector)
23 tspan = [0 10];
24 pinit = [0; 0; -100; 0; 0; 0; 0; 0; 0; 0; 0.1; 0];
25 [tx,x1] = ode45(@(t,x) LinearQuadControlQ2(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);
26 [tx2,x2] = ode45(@(t,x)
    QuadrotorEOM_controlQ3(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);
27
28
29 xval = x1(:,1);
30 yval = x1(:,2);
31 zval = x1(:,3);
32 phival = x1(:,4);
33 thetaval = x1(:,5);
34 psival = x1(:,6);
35 uval = x1(:,7);
36 vval = x1(:,8);
37 wval = x1(:,9);
38 pval = x1(:,10);
39 qval = x1(:,11);
40 rval = x1(:,12);
41
42 xval2 = x2(:,1);
43 yval2 = x2(:,2);
44 zval2 = x2(:,3);
45 phival2 = x2(:,4);
46 thetaval2 = x2(:,5);
47 psival2 = x2(:,6);
48 uval2 = x2(:,7);
49 vval2 = x2(:,8);
50 wval2 = x2(:,9);

```

```

51     pval2 = x2(:,10);
52     qval2 = x2(:,11);
53     rval2 = x2(:,12);
54
55     prob = 0;
56
57
58     figure(1)
59     hold on
60     %drone_plot(x1,tx,'pqr', prob);
61     drone_plot2(x1,tx, x2,tx2, 'pqr', prob);
62     hold off
63
64     hold on
65     figure(2)
66     %drone_plot(x1,tx,'uvw',prob);
67     drone_plot2(x1,tx,x2,tx2,'uvw',prob);
68     hold off
69
70     hold on
71     figure(3)
72     %drone_plot(x1,tx,'euler',prob);
73     drone_plot2(x1,tx,x2,tx2,'euler',prob);
74     hold off
75
76     hold on
77     figure(4)
78     %drone_plot(x1,tx,'xyz',prob);
79     drone_plot2(x1, tx,x2,tx2,'xyz',prob);
80     hold off
81
82     figure(5)
83     plot3(xval,yval,zval);
84     grid on
85 end

```

```

1 function [Output_Plot] = Plot12(Time,State)
2
3 t=Time;%time
4
5 x=State(:,1);%populate variables
6 y=State(:,2);%populate variables
7 z=State(:,3);%populate variables
8 phi=State(:,4);%populate variables
9 theta=State(:,5);%populate variables
10 psi=State(:,6);%populate variables
11 u=State(:,7);%populate variables
12 v=State(:,8);%populate variables
13 w=State(:,9);%populate variables
14 p=State(:,10);%populate variables
15 q=State(:,11);%populate variables
16 r=State(:,12);%populate variables
17
18
19
20 sgtitle('6 Experimental Drone Behavior')%main title
21 subplot(4,3,1)%sub plot call
22 plot(t,x)
23 ylabel('x_E [m]')
24 xlabel('time [s]')
25
26 subplot(4,3,2)%sub plot call
27 plot(t,y)
28 ylabel('y_E [m]')
29 xlabel('time [s]')
30
31
32 subplot(4,3,3)%sub plot call
33 plot(t,z)
34 ylabel('z_E [m]')
35 xlabel('time [s]')

```

```

36
37
38 subplot(4,3,4)%sub plot call
39 plot(t,phi)
40 ylabel('\phi [rad]')
41 xlabel('time [s]')
42
43
44 subplot(4,3,5)%sub plot call
45 plot(t,theta)
46 ylabel('\theta [rad]')
47 xlabel('time [s]')
48
49
50 subplot(4,3,6)%sub plot call
51 plot(t,psi)
52 ylabel('\psi [rad]')
53 xlabel('time [s]')
54
55
56 subplot(4,3,7)%sub plot call
57 plot(t,u)
58 ylabel('u^E [m/s]')
59 xlabel('time [s]')
60
61
62 subplot(4,3,8)%sub plot call
63 plot(t,v)
64 ylabel('v^E [m/s]')
65 xlabel('time [s]')
66
67
68 subplot(4,3,9)%sub plot call
69 plot(t,w)
70 ylabel('w^E [m/s]')

```

```

71 xlabel('time [s]')
72
73
74 subplot(4,3,10)%sub plot call
75 plot(t,p)
76 ylabel('p [rad/s]')
77 xlabel('time [s]')
78
79
80 subplot(4,3,11)%sub plot call
81 plot(t,q)
82 ylabel('q [rad/s]')
83 xlabel('time [s]')
84
85
86 subplot(4,3,12)%sub plot call
87 plot(t,r)
88 ylabel('r [rad/s]')
89 xlabel('time [s]')
90
91
92
93
94 Output_Plot=1;%lazy
95
96 end

1 %Part 6 Data read in and plot
2 %Emerson and group
3 %10/22/2020
4
5 clear;close;clc
6
7 data=load('RSdata_10_21_1451');% load file
8

```

```

9  %---
10 A=data.rt_sensor;%get data
11 B=data.rt_optical;%get data
12 C=data.rt_calib;%get data
13 D=data.rt_cmd;%get data
14 G1=data.rt_estim;%get data
15 %G2=data.rt_estimatedStates; %obsolete
16
17 %----
18 MotorForce=data.rt_motor;%motor forces get data
19 MotorForceTime=MotorForce.time;%motor forces get data
20 MotorForceStrut=MotorForce.signals;%motor forces get data
21 MotorForcesValues=MotorForceStrut.values;%motor forces get data
22 % ---
23
24 time=G1.time(:);%time call
25 xval_data=G1.signals.values(:,1);%get x data
26 yval_data=G1.signals.values(:,2);%get y data
27 zval_data=G1.signals.values(:,3);%get z data
28 phival_data=G1.signals.values(:,4);%get angle data
29 thetaval_data=G1.signals.values(:,5);%get angle data
30 psival_data=G1.signals.values(:,6);%get angle data
31 uval_data=G1.signals.values(:,7);%get u data
32 vval_data=G1.signals.values(:,8);%get v data
33 wval_data=G1.signals.values(:,9);%get w data
34 pval_data=G1.signals.values(:,10);%get p data
35 qval_data=G1.signals.values(:,11);%get q data
36 rval_data=G1.signals.values(:,12);%get r data
37 %---
38
39 %--
40 LLL=length(xval_data);%get length
41 x1_data=zeros(LLL,12);%make blank matrix
42 x1_data(:,1)=xval_data;%populate matrix
43 x1_data(:,2)=yval_data;%populate matrix

```



```

44 x1_data(:,3)=zval_data;%populate matrix
45 x1_data(:,4)=phival_data;%populate matrix
46 x1_data(:,5)=thetaval_data;%populate matrix
47 x1_data(:,6)=psival_data;%populate matrix
48 x1_data(:,7)=uval_data;%populate matrix
49 x1_data(:,8)=vval_data;%populate matrix
50 x1_data(:,9)=wval_data;%populate matrix
51 x1_data(:,10)=pval_data;%populate matrix
52 x1_data(:,11)=qval_data;%populate matrix
53 x1_data(:,12)=rval_data;%populate matrix
54
55
56 plot=Plot12(time,x1_data)%plot call
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78 %

```

```

79 % t=2600;
80 % State=x1_data;
81 %
82 %
83 % x=State(:,1);
84 % y=State(2);
85 % z=State(3);
86 % phi=State(4);
87 % theta=State(5);
88 % psi=State(6);
89 % u=State(7);
90 % v=State(8);
91 % w=State(9);
92 % p=State(10);
93 % q=State(11);
94 % r=State(12);
95 %
96 %
97 % subplot(4,3,1)
98 % plot(t,x)
99 %
100 % subplot(4,3,2)
101 % plot(t,y)
102 %
103 % subplot(4,3,3)
104 % plot(t,z)
105 %
106 % subplot(4,3,4)
107 % plot(t,phi)
108 %
109 % subplot(4,3,5)
110 % plot(t,theta)
111 %
112 % subplot(4,3,6)
113 % plot(t,psi)

```

```

114 %
115 % subplot(4,3,7)
116 % plot(t,u)
117 %
118 % subplot(4,3,8)
119 % plot(t,v)
120 %
121 % subplot(4,3,9)
122 % plot(t,w)
123 %
124 % subplot(4,3,10)
125 % plot(t,p)
126 %
127 % subplot(4,3,11)
128 % plot(t,q)
129 %
130 % subplot(4,3,12)
131 % plot(t,r)
132 %
133 %
134 %
135 % % plot=Plot12(time,x1_data)
136 %
137 % % subplot(4,3,1)
138 % % plot(time,xval_data)
139 % %
140 % % subplot(4,3,2)
141 % % plot(time,xval_data)
142 % %
143 % % subplot(4,3,3)
144 % % plot(time,xval_data)
145 % %
146 % % subplot(4,3,4)
147 % % plot(time,xval_data)
148 %

```

```
149 | %  
150 | % %Plot Function(time,matrix12)
```