UNIVERSITY OF COLORADO - BOULDER

ASEN 3128 : LAB 4

SUBMITTED OCTOBER 23RD, 2020

Lab 4: Quadrotor Control

Author:

Emerson BEINHAUER[a]

Author:

Luke ENGELKEN[b]

Author:

Connor O'REILLY[c]

Author:

Thyme ZUSCHLAG[d]

Professor:

Professor FREW

[a]108185329
[b]107165781
[c]107054811
[d]109221211

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Table of Contents

I. Problem 1

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

$$\tau = \frac{-1}{\lambda_i} \tag{1}$$

$$(\mathbf{A} - \lambda_i)\mathbf{v_i} = 0 \tag{2}$$

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.

$$\Delta L_c = -0.004\Delta p - 0.0021\Delta\phi \tag{3}$$

$$\Delta M_c = -0.004\Delta q - 0.0021\Delta\theta \tag{4}$$

II. Problem 2 and 3 Plots and Discussion

A. Deviation by +5 deg in roll

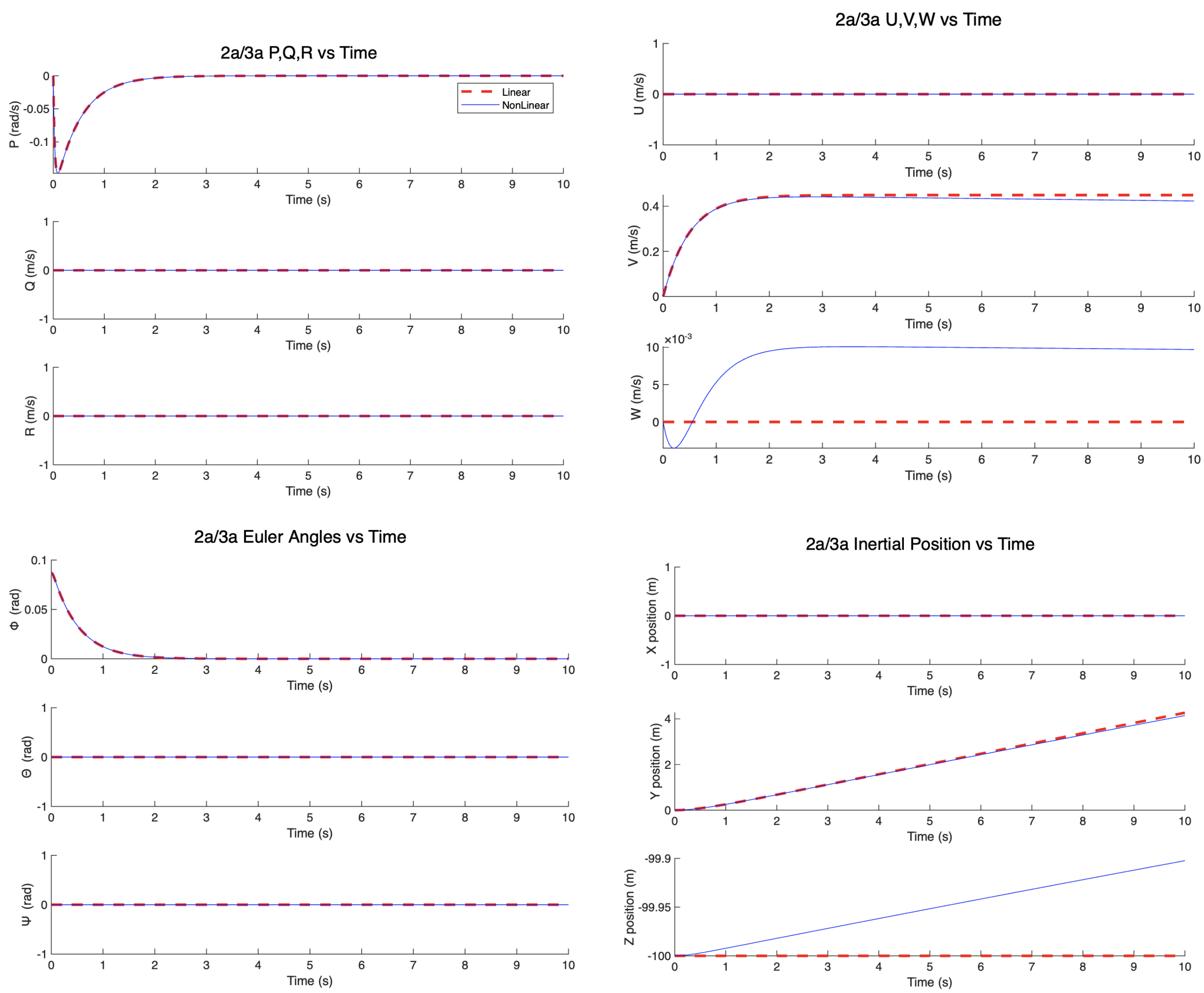


Fig. 1 Linearized vs. Nonlinearized Behaviour 2a/3a

B. Deviation by +5 deg in pitch

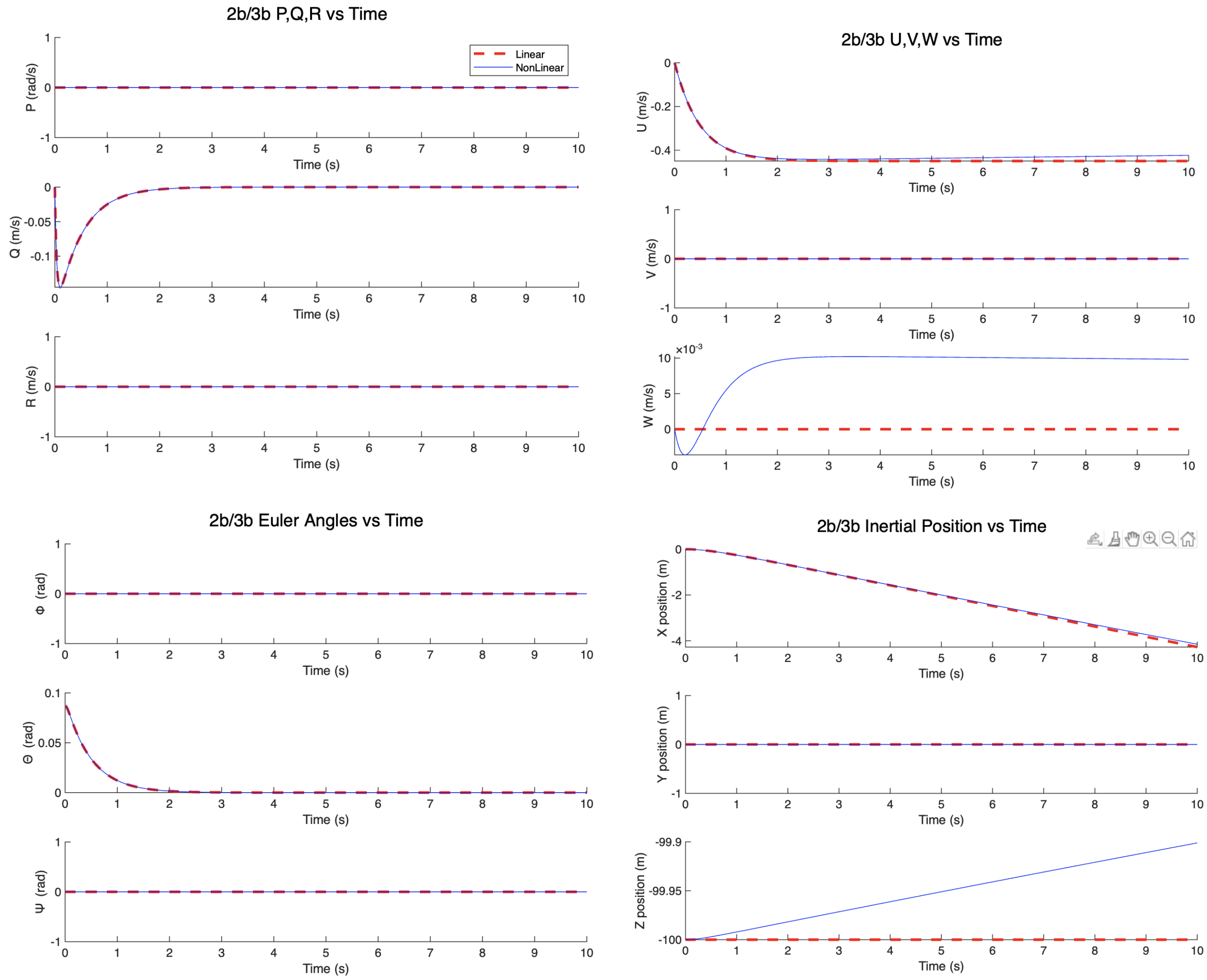


Fig. 2 Linearized vs. Nonlinearized Behaviour 2b/3b

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

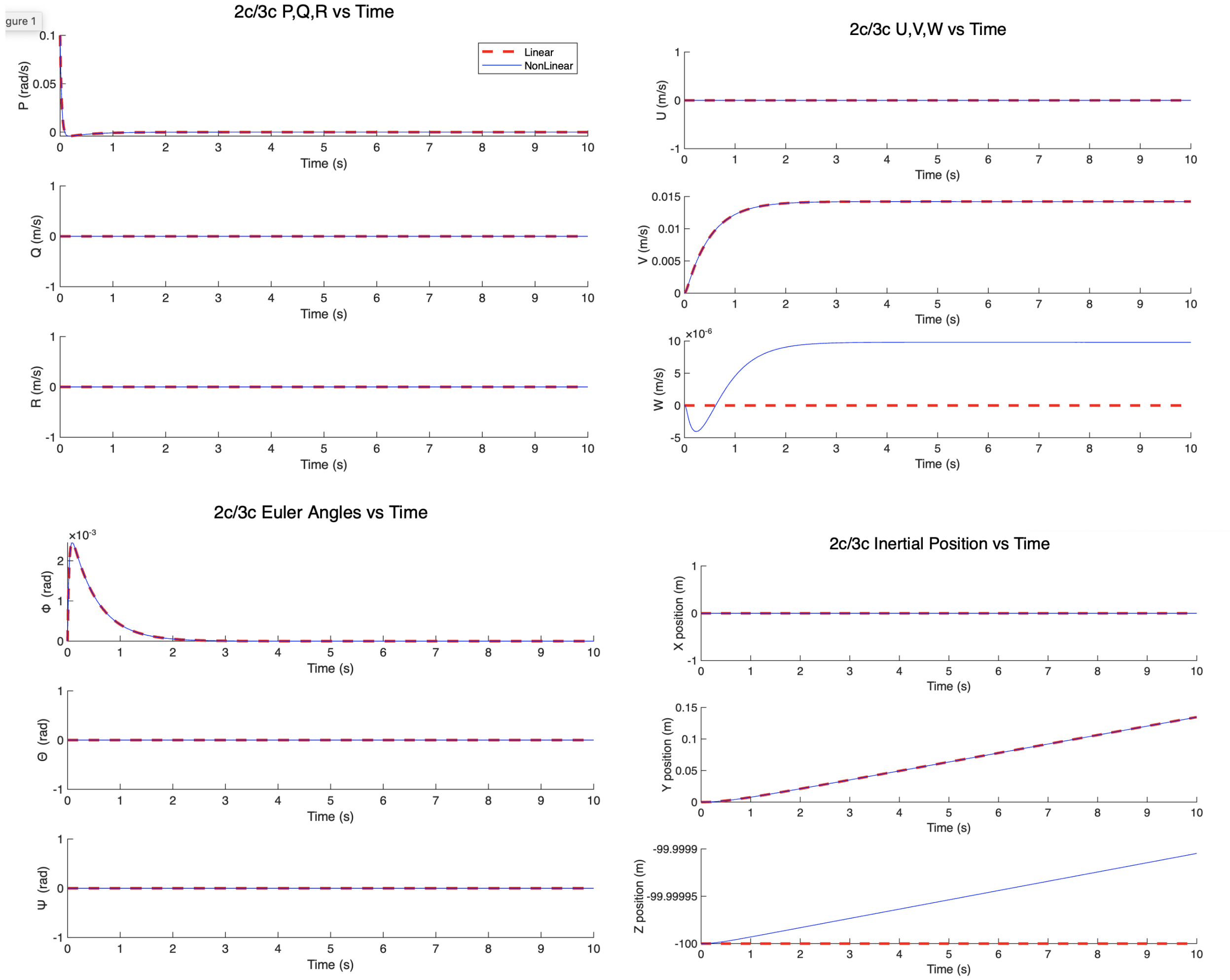


Fig. 3 Linearized vs. Nonlinearized Behaviour 2c/3c

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

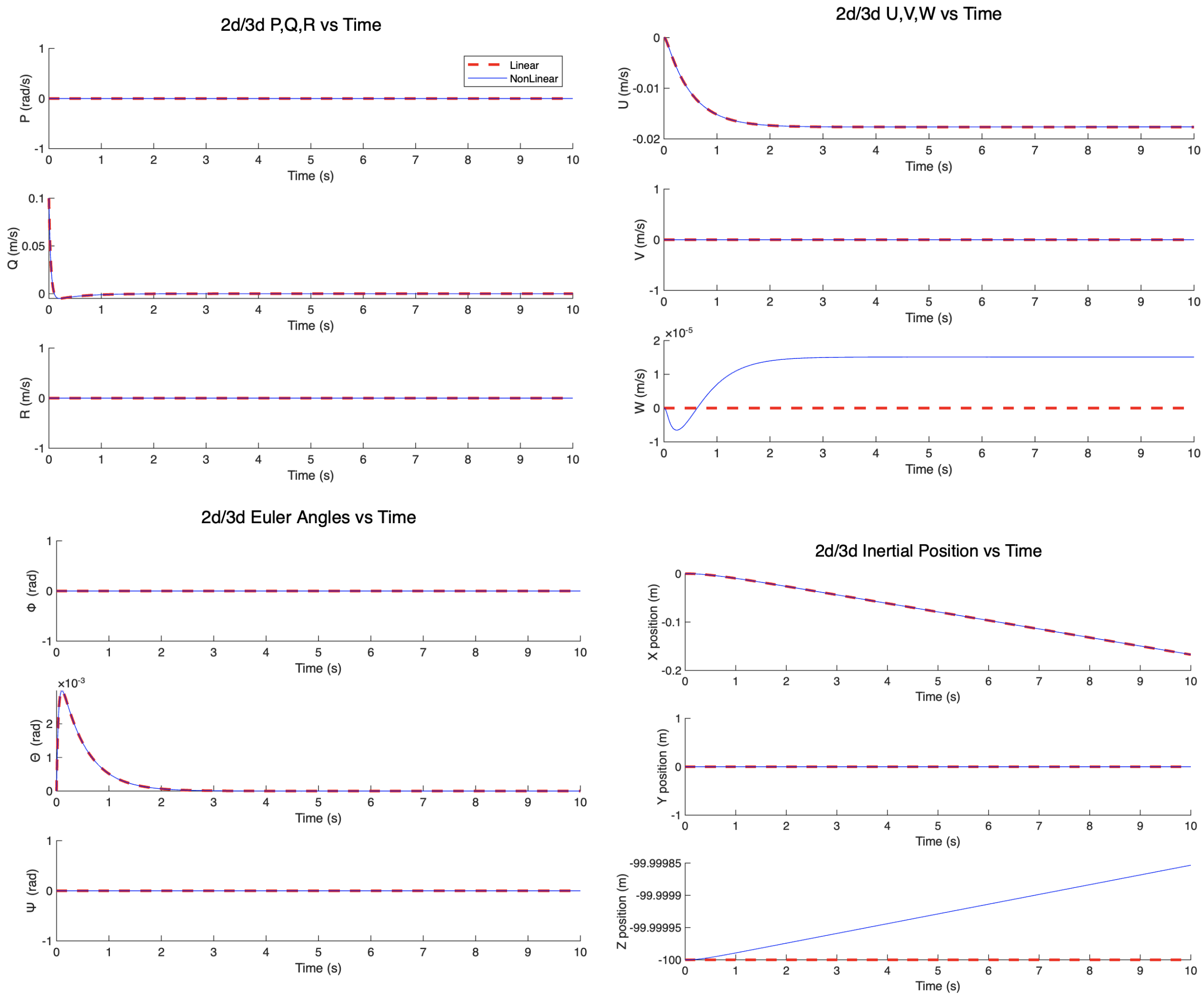


Fig. 4 Linearized vs. Nonlinearized Behaviour 2d/3d

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 tiled, 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

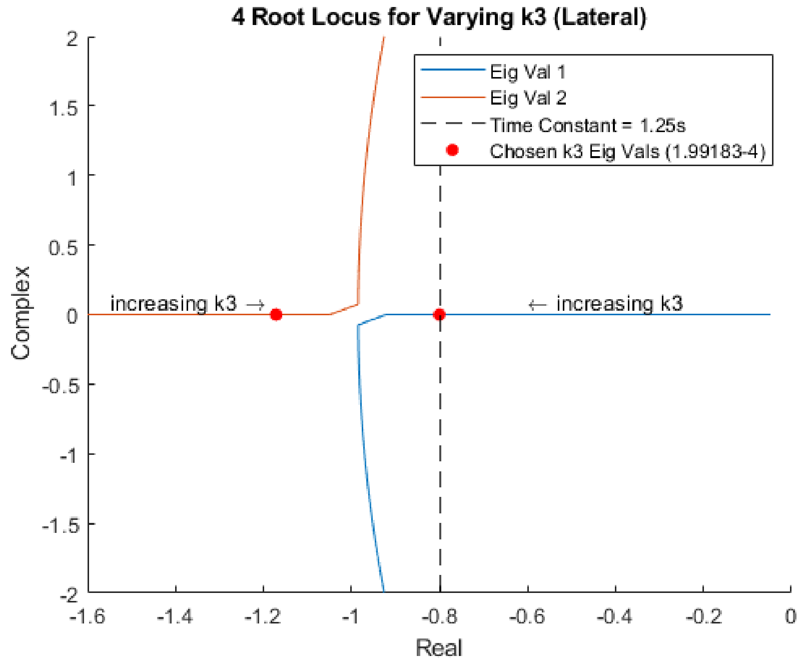


Fig. 5 Lateral

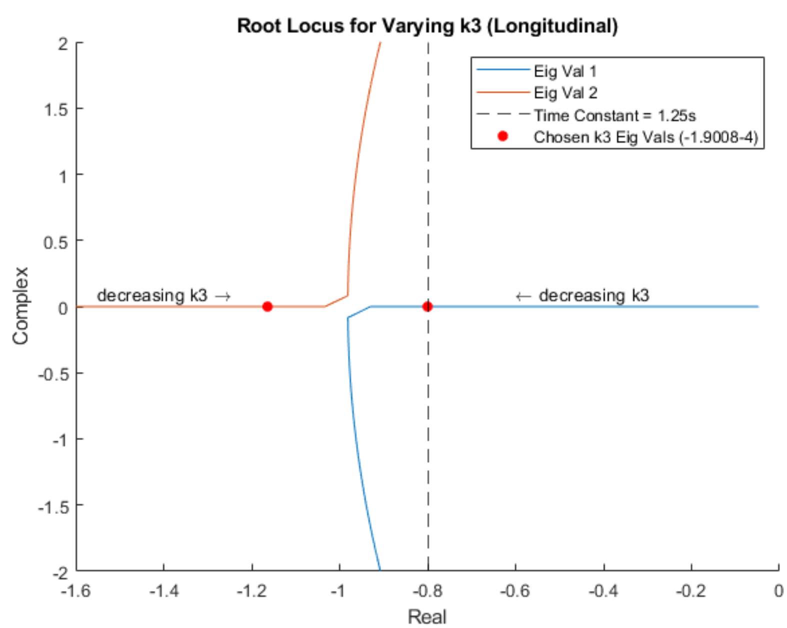


Fig. 6 Longitudinal

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

$$\tau = \frac{-1}{\lambda_i} \tag{5}$$

$$(\mathbf{A} - \lambda_i)\mathbf{v_i} = 0 \tag{6}$$

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:

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

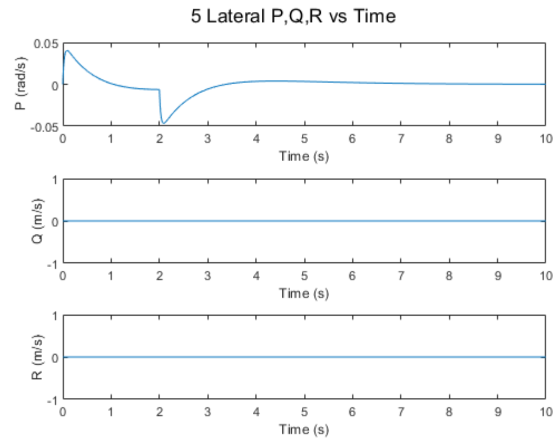
Lateral System:

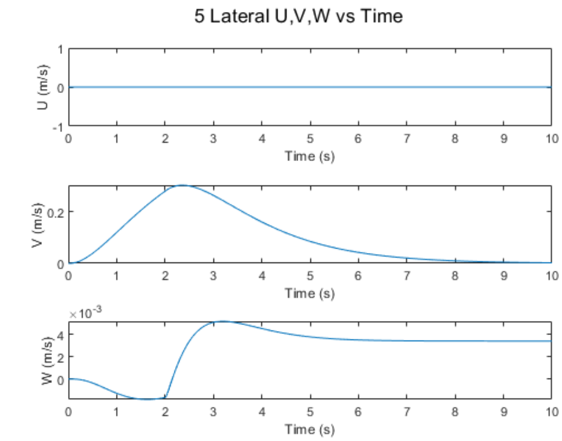
$$\Delta M_c = -0.004\Delta q - 0.0021\Delta\theta + (1.99183 \times 10^{-4})(\Delta u_r - \Delta u) \tag{8}$$

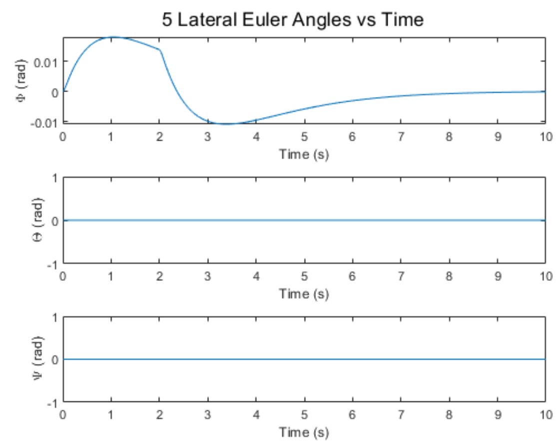
IV. Problem 5

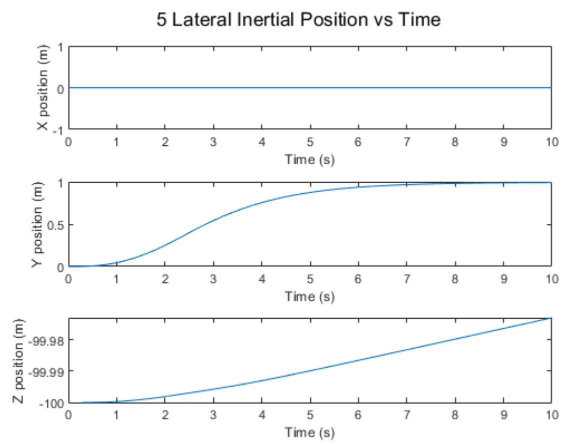
Using the lateral and longitudinal control laws designed in problem 4 the models are simulated and shown in the following plots.

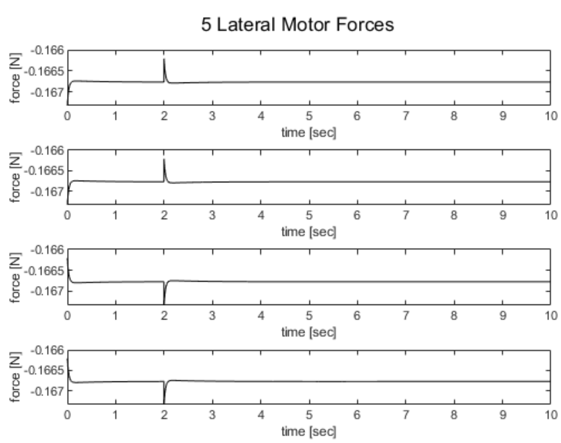
A. Lateral Model



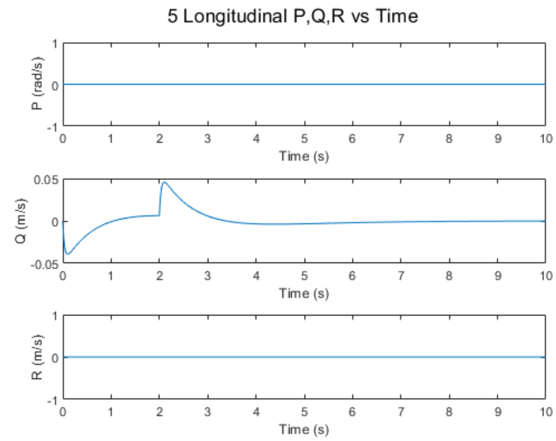


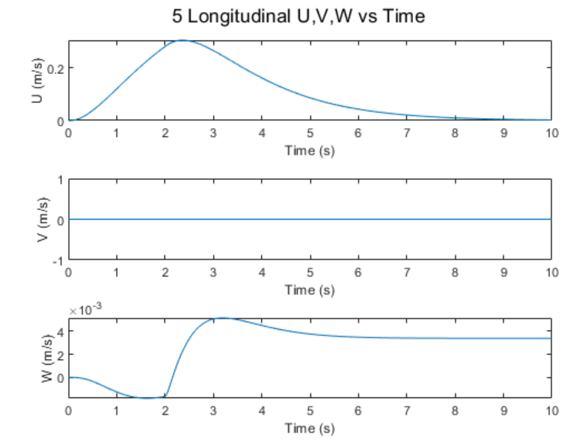


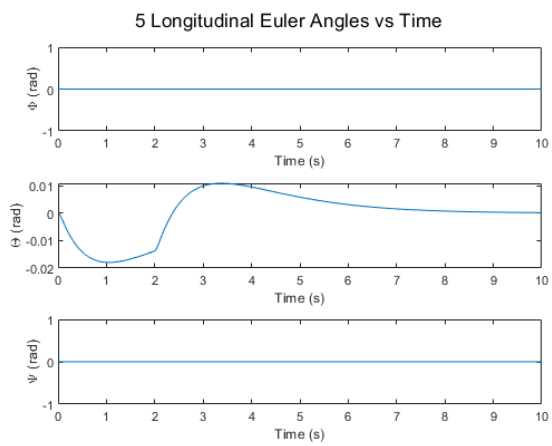


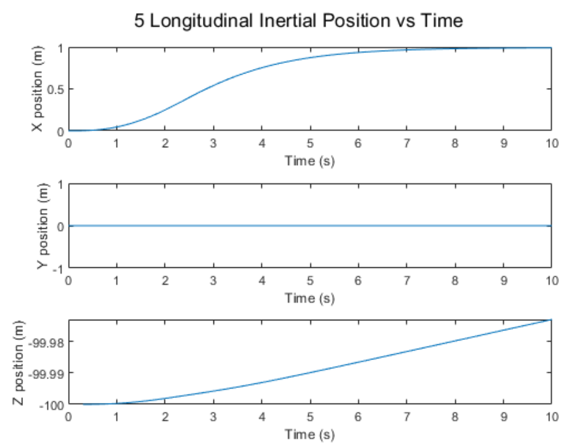


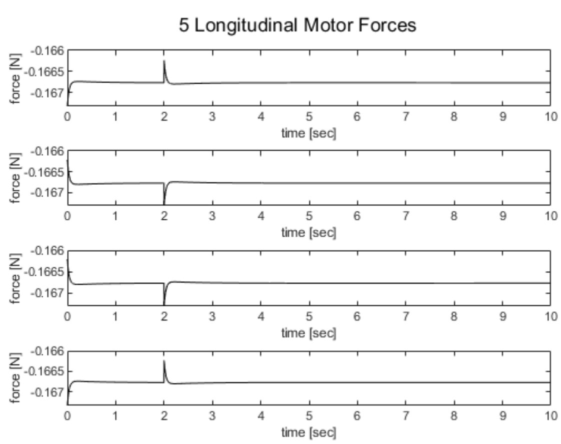
B. Longitudinal Model











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

V. Problem 6

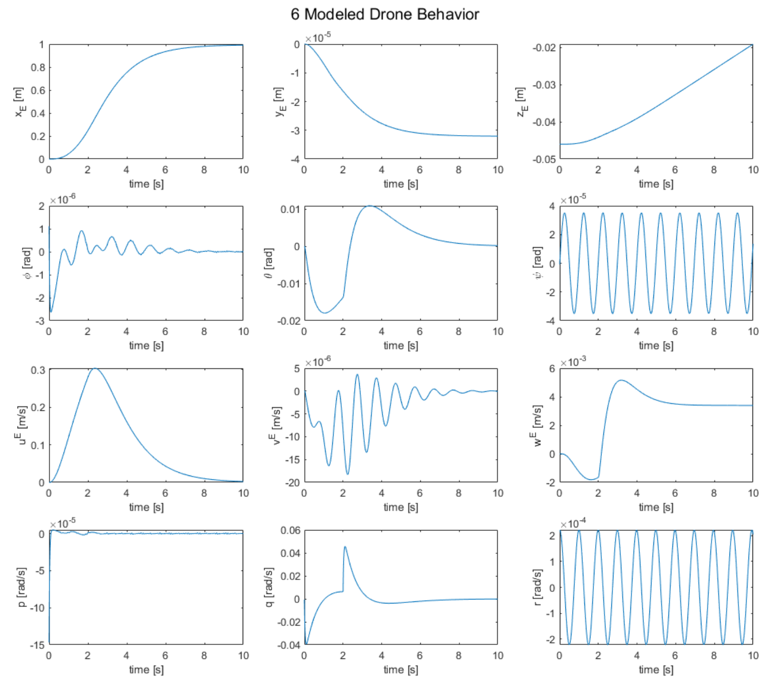


Fig. 7 Modeled Drone Behavior

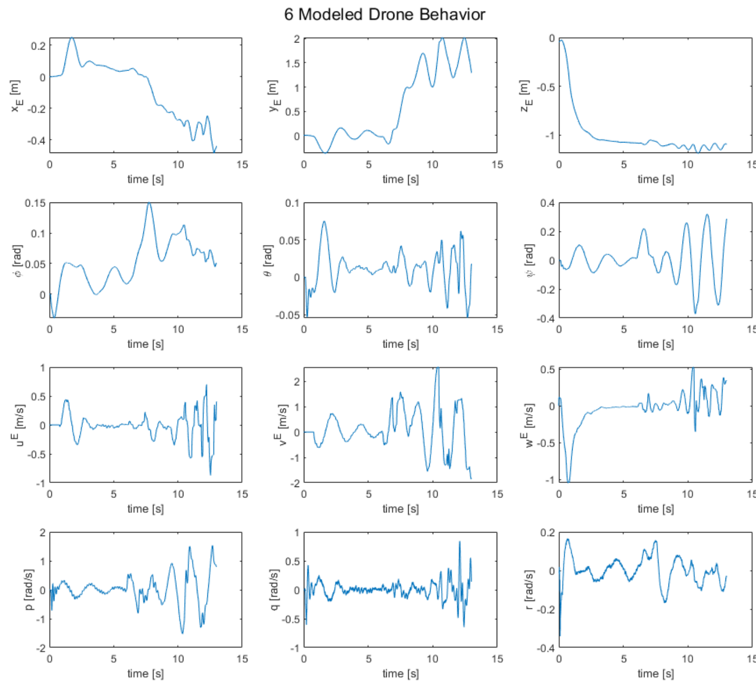


Fig. 8 Experimental Drone Behavior

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

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

```
% Luke Engelken
% ASEN 3128
% prob1.m
% Created: 9/11/20

%To run, in command window type "prob(<insert quetsion number)" and you
%will see plots
%ex: prob('1a')

function [] = drone(prob)
   %% Declare Constants
   m = 0.068; %mass of drone (kg)
   r = 0.06; %radial distance from cg to motor (m)
   km = 0.0024; %control moment coefficient (N*m/N)
   Ix = 5.8e-5; %x-axis moment of inertia (kg*m^2)
```

```
Iy = 7.2e-5; %y-axis moment of inertia (kg*m^2)
Iz = 1e-4; %z-axis moment of inertia (kg*m^2)
v = 1e-3; %aerodynamic force coefficient (N/(m/s^2))
mu = 2e-6; %N*m/(rad/s^2)
g = 9.81; %m/s^2
%xstate vector: [x; y; z; phi; theta; psi; xdot; ydot; zdot; phidot;
%thetadot; psidot] (integral of eom state vector)
tspan = [0 10];
switch prob
   case '1a'
      %Prob 1a - steady state hover, no drag
      f1 = g*m/4; %Assume -force up
      f2 = f1;
      f3 = f1;
      f4 = f1;
      pinit = [0; 0; 100; 0; 0; 0; 0; 0; 0; 0; 0; 0];
      [tx,x1] = ode45(@(t,x)
          objectEOMnodrag(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
   case '2a'
      %Prob 2a - steady state hover, with drag
      f1 = 2;%g*m/4; %Assume -force up
      f2 = f1;
      f3 = f1;
      f4 = f1;
      pinit = [0; 0; -10; 0; 0; 0; 0; 0; 0; 0; 0; 0];
      [tx,x1] = ode45(@(t,x)
          objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
   case '2b'
      %Prob 2b - constant y velocity = 5m/s, psi = 0
      syms phi
      %eqn = 5*v*5*cos(phi)+m*g*sin(phi)^3+m*g*sin(phi)*cos(phi)^2 == 0;
      eqn = atan(phi) == v*25/(m*g);
      phi = solve(eqn);
      phi = eval(phi(1,1)); %3.1041
      %phi = 2.12;
```

```
        pinit = [0; 0; 0; phi; 0; 0; 0; 5*cos(phi); -5*sin(phi); 0; 0; 0];
        Zc = m*g*(cos(pinit(4))+sin(pinit(4))^2/cos(pinit(4))); %-0.667080298221064
        %Zc = 25*v/sin(phi); %-.6681
        %Zc = .6665;
        f1 = Zc/4;
        f2 = f1;
        f3 = f2;
        f4 = f3;
        [tx,x1] = ode45(@(t,x)
            objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
    case '2c'
        %Prob 2c - constant y velocity = -5m/s, psi = 90deg
         syms phi
        %eqn = 5*v*5*cos(phi)+m*g*sin(phi)^3+m*g*sin(phi)*cos(phi)^2 == 0;
        eqn = atan(phi) == v*25/(m*g);
        phi = solve(eqn);
        phi = eval(phi(1,1)); %3.1041
        %phi = 2.12;
        pinit = [0; 0; 0; 0; -phi; pi/2; 5*cos(phi); 0; -5*sin(phi); 0; 0; 0];
        Zc = m*g*(cos(pinit(5))+sin(pinit(5))^2/cos(pinit(5))); %-0.667080298221064
        %Zc = 25*v/sin(phi); %-.6681
        %Zc = .6665;
        f1 = Zc/4;
        f2 = f1;
        f3 = f2;
        f4 = f3;
        [tx,x1] = ode45(@(t,x)
            objectEOM(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
    case '3'
        %Prob 2a - steady state hover, with drag
        f1 = g*m/4; %Assume -force up
        f2 = f1;
        f3 = f1;
        f4 = f1;
        pinit = [0; 0; -10; 0; 0; 0; 0; 0; 0; 0; 0; 0];
```

```
        [tx,x1] = ode45(@(t,x)
            objectEOM_disturb(t,x,m,r,km,Ix,Iy,Iz,v,mu,f1,f2,f3,f4),tspan,pinit);
    case '5'
%          f1 = -m*g/4; %Assume -force up
%          f2 = f1;
%          f3 = f1;
%          f4 = f1;
        pinit = [0; 0; -100; 5*pi/180; 0; 0; 0; 0; 0; 0; 0; 0];
        [tx,x1] = ode45(@(t,x)
            LinearQuadControlQ2(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);
end

xval = x1(:,1);
yval = x1(:,2);
zval = x1(:,3);
phival = x1(:,4);
thetaval = x1(:,5);
psival = x1(:,6);
uval = x1(:,7);
vval = x1(:,8);
wval = x1(:,9);
pval = x1(:,10);
qval = x1(:,11);
rval = x1(:,12);

figure(1)
drone_plot(x1,tx,'pqr',prob);

figure(2)
drone_plot(x1,tx,'uvw',prob);

figure(3)
drone_plot(x1,tx,'euler',prob);

figure(4)
```

```matlab
    drone_plot(x1,tx,'xyz',prob);

    figure(5)
    plot3(xval,yval,zval);
    grid on
end
```

```matlab
function [] = drone_plot(x1,tx,type,prob)
%DRONE_PLOT: Plots the 4 subplots of inertial position, Euler angles,
%body acceleration, and Euler rates
xval = x1(:,1);
    yval = x1(:,2);
    zval = x1(:,3);
    phival = x1(:,4);
    thetaval = x1(:,5);
    psival = x1(:,6);
    uval = x1(:,7);
    vval = x1(:,8);
    wval = x1(:,9);
    pval = x1(:,10);
    qval = x1(:,11);
    rval = x1(:,12);

switch type
    case 'pqr'
        subplot(3,1,1)
        plot(tx,pval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('P (rad/s)');
        subplot(3,1,2)
        plot(tx,qval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('Q (m/s)');
        subplot(3,1,3)
        plot(tx,rval,'LineWidth',5);
```

```
        xlabel('Time (s)');
        ylabel('R (m/s)');
        sgtitle(sprintf('%s P,Q,R vs Time',prob));
    case 'uvw'
        subplot(3,1,1)
        plot(tx,uval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('U (m/s)');
        if prob == '2c'
            axis([0 10 4.5 5.5]);
        end
        subplot(3,1,2)
        plot(tx,vval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('V (m/s)');
        if prob == '2b'
            axis([0 10 4.9 5.1]);
        end
        subplot(3,1,3)
        plot(tx,wval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('W (m/s)');
        if prob == '2b' | prob == '2c'
            axis([0 10 -1 0]);
        end
        sgtitle(sprintf('%s U,V,W vs Time',prob));
    case 'euler'
        subplot(3,1,1)
        plot(tx,phival,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('\Phi (rad)');
        subplot(3,1,2)
        plot(tx,thetaval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('\Theta (rad)');
```

```
        subplot(3,1,3)
        plot(tx,psival,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('\Psi (rad)');
        sgtitle(sprintf('%s Euler Angles vs Time',prob));
    case 'xyz'
        subplot(3,1,1)
        plot(tx,xval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('X position (m)');
        subplot(3,1,2)
        plot(tx,yval,'LineWidth',5);
        xlabel('Time (s)');
        ylabel('Y position (m)');
        subplot(3,1,3)
        plot(tx,zval,'LineWidth',5);
        if prob == '2b' | prob == '2c'
            axis([0 10 -1 1]);
        end
        xlabel('Time (s)');
        ylabel('Z position (m)');
        sgtitle(sprintf('%s Inertial Position vs Time',prob));


end
```

```
%Author: Thyme Zuschlag
%Class: 3128 Lab 4
%Date: 10/21
%Purpose: Plot nicely on subplots with appropiate titles

function [] = drone_plot2(x1,tx,x2, tx2, type,prob)
%DRONE_PLOT: Plots the 4 subplots of inertial position, Euler angles,
%body acceleration, and Euler rates
xval = x1(:,1);
```

```
    yval = x1(:,2);
    zval = x1(:,3);
    phival = x1(:,4);
    thetaval = x1(:,5);
    psival = x1(:,6);
    uval = x1(:,7);
    vval = x1(:,8);
    wval = x1(:,9);
    pval = x1(:,10);
    qval = x1(:,11);
    rval = x1(:,12);

    xval2 = x2(:,1);
    yval2 = x2(:,2);
    zval2 = x2(:,3);
    phival2 = x2(:,4);
    thetaval2 = x2(:,5);
    psival2 = x2(:,6);
    uval2 = x2(:,7);
    vval2 = x2(:,8);
    wval2 = x2(:,9);
    pval2 = x2(:,10);
    qval2 = x2(:,11);
    rval2 = x2(:,12);

switch type
    case 'pqr'
        subplot(3,1,1)
        hold on
        plot(tx,pval,'--r', "LineWidth", 2);
        plot(tx2,pval2, 'b');
        xlabel('Time (s)');
        ylabel('P (rad/s)');
        legend('Linear', 'NonLinear');
        subplot(3,1,2)
```

```
        hold on
        plot(tx,qval,'--r', "LineWidth", 2);
        plot(tx2,qval2, 'b');
        xlabel('Time (s)');
        ylabel('Q (m/s)');
        subplot(3,1,3)
        hold on
        plot(tx,rval,'--r', "LineWidth", 2);
        plot(tx2,rval2, 'b');
        xlabel('Time (s)');
        ylabel('R (m/s)');
        sgtitle(sprintf('%s 2d/3d P,Q,R vs Time',prob));

    case 'uvw'
        subplot(3,1,1)
        hold on
        plot(tx,uval,'--r', "LineWidth", 2);
        plot(tx2,uval2, 'b');
        xlabel('Time (s)');
        ylabel('U (m/s)');
        if prob == '2c'
            axis([0 10 4.5 5.5]);
        end
        subplot(3,1,2)
        hold on
        plot(tx,vval,'--r', "LineWidth", 2);
        plot(tx2,vval2, 'b');
        xlabel('Time (s)');
        ylabel('V (m/s)');
        if prob == '2b'
            axis([0 10 4.9 5.1]);
        end
        subplot(3,1,3)
        hold on
        plot(tx,wval,'--r', "LineWidth", 2);
```

```
      plot(tx2,wval2, 'b');
      xlabel('Time (s)');
      ylabel('W (m/s)');
      if prob == '2b' | prob == '2c'
         axis([0 10 -1 0]);
      end
      sgtitle(sprintf('%s 2d/3d U,V,W vs Time',prob));
   case 'euler'
      subplot(3,1,1)
      hold on
      plot(tx,phival,'--r', "LineWidth", 2);
      plot(tx2,phival2, 'b');
      xlabel('Time (s)');
      ylabel('\Phi (rad)');
      subplot(3,1,2)
      hold on
      plot(tx,thetaval,'--r', "LineWidth", 2);
      plot(tx2,thetaval2, 'b');
      xlabel('Time (s)');
      ylabel('\Theta (rad)');
      subplot(3,1,3)
      hold on
      plot(tx,psival,'--r', "LineWidth", 2);
      plot(tx2,psival2, 'b');
      xlabel('Time (s)');
      ylabel('\Psi (rad)');
      sgtitle(sprintf('%s 2d/3d Euler Angles vs Time',prob));
   case 'xyz'
      subplot(3,1,1)
      hold on
      plot(tx,xval,'--r', "LineWidth", 2);
      plot(tx2,xval2, 'b');
      xlabel('Time (s)');
      ylabel('X position (m)');
      subplot(3,1,2)
```

```
      hold on
      plot(tx,yval,'--r', "LineWidth", 2);
      plot(tx2,yval2, 'b');
      xlabel('Time (s)');
      ylabel('Y position (m)');
      subplot(3,1,3)
      hold on
      plot(tx,zval,'--r', "LineWidth", 2);
      plot(tx2,zval2, 'b');
      if prob == '2b' | prob == '2c'
         axis([0 10 -1 1]);
      end
      xlabel('Time (s)');
      ylabel('Z position (m)');
      sgtitle(sprintf('%s 2d/3d Inertial Position vs Time',prob));


end
```

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

```
% xdot = 12-dimension state vector includes inertial velocity in inertial
%   coordinates and the inertial acceleration in inertial coordinates
%   [xdot; ydot; zdot; phidot; thetadot; psidot; udot; vdot; wdot;
%   pdot; qdot; rdot]
%
%Methodology: Use Newton's second law F=ma to calculate the acceleration
%  and velocity at each point in time for ode45 to integrate to find
%  position. Drag, gravity and motor thrust are only forces acting on drone.
%

g = 9.81;

%Get IV's
x1 = x(1);
y1 = x(2);
z1 = x(3);
phi1 = x(4);
theta1 = x(5);
psi1 = x(6);
u1 = x(7);
v1 = x(8);
w1 = x(9);
p1 = x(10);
q1 = x(11);
r1 = x(12);
Zc = -m*g;

%Calculate moments
k1Lat = 0.0021;
k2Lat = 0.004;
k3Lat = 1.9118e-4;
k1Lon = 0.0021;
k2Lon = 0.004;
k3Lon = -1.9008e-4;
m_ctl(1) = -k1Lat*p1-k2Lat*phi1;
```

```
m_ctl(2) = -k1Lon*q1-k2Lon*theta1;
m_ctl(3) = -0.004*psi1;

Lc = m_ctl(1);
Mc = m_ctl(2);
Nc = m_ctl(3);

%Put back into ode45
xstate(1) = u1; %xdot
xstate(2) = v1; %ydot
xstate(3) = w1; %zdot
xstate(4) = p1; %phidot
xstate(5) = q1; %thetadot
xstate(6) = r1; %psidot
xstate(7) = -g*theta1; %udot
xstate(8) = g*phi1; %vdot
xstate(9) = 0; %wdot
xstate(10) = 1/Ix*(Lc); %pdot
xstate(11) = 1/Iy*(Mc); %qdot
xstate(12) = 1/Ix*(Nc); %rdot
xstate = xstate';
end
```

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

```
% mu = moment coefficient
% f1,f2,f3,f4 = forces from motors
%
%Outputs:
% xdot = 12-dimension state vector includes inertial velocity in inertial
%   coordinates and the inertial acceleration in inertial coordinates
%   [xdot; ydot; zdot; phidot; thetadot; psidot; udot; vdot; wdot;
%   pdot; qdot; rdot]
%
%Methodology: Use Newton's second law F=ma to calculate the acceleration
%  and velocity at each point in time for ode45 to integrate to find
%  position. Drag, gravity and motor thrust are only forces acting on drone.
%

g = 9.81;

%Get IV's
x1 = x(1);
y1 = x(2);
z1 = x(3);
phi1 = x(4);
theta1 = x(5);
psi1 = x(6);
u1 = x(7);
v1 = x(8);
w1 = x(9);
p1 = x(10);
q1 = x(11);
r1 = x(12);
Zc = -m*g;
ur = vref(1);
vr = vref(2);

%K vals from q1
k1Lat = 0.0021;
```

```
k2Lat = 0.004;
k3Lat = 1.9118e-4;
k1Lon = 0.0021;
k2Lon = 0.004;
k3Lon = -1.9008e-4;
m_ctl(1) = -k1Lat*p1-k2Lat*phi1+k3Lat*(vr-v1);
m_ctl(2) = -k1Lon*q1-k2Lon*theta1+k3Lon*(ur-u1);
m_ctl(3) = -0.004*psi1;

Lc = m_ctl(1);
Mc = m_ctl(2);
Nc = m_ctl(3);

%Put back into ode45
xstate(1) = u1; %xdot
xstate(2) = v1; %ydot
xstate(3) = w1; %zdot
xstate(4) = p1; %phidot
xstate(5) = q1; %thetadot
xstate(6) = r1; %psidot
xstate(7) = -g*theta1; %udot
xstate(8) = g*phi1; %vdot
xstate(9) = 0; %wdot
xstate(10) = 1/Ix*(Lc); %pdot
xstate(11) = 1/Iy*(Mc); %qdot
xstate(12) = 1/Ix*(Nc); %rdot
xstate = xstate';
end
```

```
clc
clear all
close all

%% Q2 and 3: Find k vals using the A matrix
%time constant of 0.5s -> eig = -2
```

```
Ix = 5.8e-5;
Iy = 7.2e-5;
k2Latreal = 0.004;
k1Latreal = (-k2Latreal/Ix - 4)*-Ix/2;
% k2Lat = linspace(k2Latreal/100,k2Latreal,100);
k2Lonreal = 0.004;
k1Lonreal = (-k2Lonreal/Iy - 4)*-Iy/2;
g = 9.81;
%%Lat k vals
% for i = 1:100
%    k1Lat = (-k2(i)/Ix - 4)*-Ix/2;
%    A = [0 1; -k2(i)/Ix -k1Lat/Ix];
%    eigval(i,:) = eig(A)';
% end
% A1 = [0 1; -k2Latreal/Ix -k1Latreal/Ix];
% eigreal = eig(A1)
%Lin k vals
% A1 = [0 1; -k2Lonreal/Iy -k1Lonreal/Iy];
% eigreal = eig(A1)

% %% Q4
bLat = -k2Latreal/Ix;
cLat = -k1Latreal/Ix;
bLon = -k2Lonreal/Iy;
cLon = -k1Lonreal/Iy;
k3Latreal = -Ix*(0.8*bLat-0.64*cLat-0.512)/9.81;
k3Lonreal = Iy*(0.8*bLon-0.64*cLon-0.512)/9.81;
aLatreal = -k3Latreal/Ix;
aLonreal = -k3Lonreal/Iy;
ALatreal = [0 g 0; 0 0 1; aLatreal bLat cLat];
ALonreal = [0 -g 0; 0 0 1; aLonreal bLon cLon];
eigsLat = eig(ALatreal);
eigsLon = eig(ALonreal)
k3Lat = linspace(k3Latreal/10,k3Latreal*10,1000);
aLat = -k3Lat/Ix;
```

```
for i = 1:1000
   A = [0 g 0; 0 0 1; aLat(i) bLat cLat];
   eigs = eig(A);
   eigvalLat(i,:) = eigs';
end
figure(1)
hold on
axis([-1.6 0 -2 2]);
% plot(eigvalLat(:,1),0);
plot(eigvalLat(:,2));
plot(eigvalLat(:,3));
xline(-0.8,'--k');
plot(eigsLat(1),0,'r.','MarkerSize',20);
plot(eigsLat(2),0,'r.','MarkerSize',20);
plot(eigsLat(3),0,'r.','MarkerSize',20);
text(-1.55,.1,'increasing k3 \rightarrow');
text(-.6,.1,'\leftarrow increasing k3 ');
xlabel('Real');
ylabel('Complex');
title('4 Root Locus for Varying k3 (Lateral)');
legend('Eig Val 1','Eig Val 2','Time Constant = 1.25s',...
   'Chosen k3 Eig Vals (1.99183-4)');

% k3Lon = linspace(k3Lonreal/10,k3Lonreal*10,1000);
% aLon = -k3Lon/Iy;
% for i = 1:1000
%    A = [0 -g 0; 0 0 1; aLon(i) bLon cLon];
%    eigs = eig(A);
%    eigvalLon(i,:) = eigs';
% end
% figure(2)
% hold on
% axis([-1.6 0 -2 2]);
% % plot(eigvalLat(:,1),0);
% plot(eigvalLon(:,2));
```

```
% plot(eigvalLon(:,3));
% xline(-0.8,'--k');
% plot(eigsLon(1),0,'r.','MarkerSize',20);
% plot(eigsLon(2),0,'r.','MarkerSize',20);
% plot(eigsLon(3),0,'r.','MarkerSize',20);
% text(-1.55,.1,'decreasing k3 \rightarrow');
% text(-.6,.1,'\leftarrow decreasing k3 ');
% xlabel('Real');
% ylabel('Complex');
% title('4 Root Locus for Varying k3 (Longitudinal)');
% legend('Eig Val 1','Eig Val 2','Time Constant = 1.25s',...
%    'Chosen k3 Eig Vals (-1.9008-4)');

%% Q5
% k2Latreal = 0.004;
% k1Latreal = (-k2Latreal/Ix - 4)*-Ix/2;
% k2Lonreal = 0.004;
% k1Lonreal = (-k2Lonreal/Iy - 4)*-Iy/2;
% bLat4 = -k2Latreal/Ix;
% cLat4 = -k1Latreal/Ix;
% bLon4 = -k2Lonreal/Iy;
% cLon4 = -k1Lonreal/Iy;
% k3Latreal = -Ix*(0.8*bLat4-0.64*cLat4-0.512)/9.81;
% k3Lonreal = Iy*(0.8*bLon4-0.64*cLon4-0.512)/9.81;

% ALatreal = [0 1 0 0; 0 0 g 0; 0 0 0 1; aLatreal bLat cLat dLat];
% ALonreal = [0 1 0 0; 0 0 -g 0; 0 0 0 1; aLonreal bLon cLon dLon];
```

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

```
        REB(2,1) = sind(phi)*sind(theta)*cosd(psi)-cosd(phi)*sind(psi);
        REB(2,2) = sind(phi)*sind(theta)*sind(psi)+cosd(phi)*cosd(psi);
        REB(2,3) = sind(phi)*cosd(theta);
        REB(3,1) = cosd(phi)*sind(theta)*cosd(psi)+sind(phi)*sind(psi);
        REB(3,2) = cosd(phi)*sind(theta)*sind(psi)-sind(phi)*cosd(psi);
        REB(3,3) = cosd(phi)*cosd(theta);
    case 'rad'
        phi = phi * (180/pi);
        theta = theta * (180/pi);
        psi = psi * (180/pi);
        REB(1,1) = cosd(theta)*cosd(psi);
        REB(1,2) = cosd(theta)*sind(psi);
        REB(1,3) = -sind(theta);
        REB(2,1) = sind(phi)*sind(theta)*cosd(psi)-cosd(phi)*sind(psi);
        REB(2,2) = sind(phi)*sind(theta)*sind(psi)+cosd(phi)*cosd(psi);
        REB(2,3) = sind(phi)*cosd(theta);
        REB(3,1) = cosd(phi)*sind(theta)*cosd(psi)+sind(phi)*sind(psi);
        REB(3,2) = cosd(phi)*sind(theta)*sind(psi)-sind(phi)*cosd(psi);
        REB(3,3) = cosd(phi)*cosd(theta);
end
REB = inv(REB);

end
```

```
function [Tmat] = T(phi,theta,psi,units)
%T Summary of this function goes here
%   Detailed explanation goes here
switch units
    case 'deg'
        Tmat(1,1) = 1;
        Tmat(1,2) = sind(phi)*tand(theta);
        Tmat(1,3) = cosd(phi)*tand(theta);
        Tmat(2,1) = 0;
        Tmat(2,2) = cosd(phi);
        Tmat(2,3) = -sind(phi);
```

```
        Tmat(3,1) = 0;
        Tmat(3,2) = sind(phi)*secd(theta);
        Tmat(3,3) = cosd(phi)*secd(theta);

    case 'rad'
        phi = phi * (180/pi);
        theta = theta * (180/pi);
        psi = psi * (180/pi);
        Tmat(1,1) = 1;
        Tmat(1,2) = sind(phi)*tand(theta);
        Tmat(1,3) = cosd(phi)*tand(theta);
        Tmat(2,1) = 0;
        Tmat(2,2) = cosd(phi);
        Tmat(2,3) = -sind(phi);
        Tmat(3,1) = 0;
        Tmat(3,2) = sind(phi)*secd(theta);
        Tmat(3,3) = cosd(phi)*secd(theta);
end
```

```
% Luke Engelken
% ASEN 3128
% prob1.m
% Created: 9/11/20

%To run, in command window type "prob(<insert quetsion number)" and you
%will see plots
%ex: prob('1a')

function [] = updated_drone()
    %% Declare Constants
    m = 0.068; %mass of drone (kg)
    r = 0.06; %radial distance from cg to motor (m)
    km = 0.0024; %control moment coefficient (N*m/N)
    Ix = 5.8e-5; %x-axis moment of inertia (kg*m^2)
    Iy = 7.2e-5; %y-axis moment of inertia (kg*m^2)
```

```
Iz = 1e-4; %z-axis moment of inertia (kg*m^2)
v = 1e-3; %aerodynamic force coefficient (N/(m/s^2))
mu = 2e-6; %N*m/(rad/s^2)
g = 9.81; %m/s^2
%xstate vector: [x; y; z; phi; theta; psi; xdot; ydot; zdot; phidot;
%thetadot; psidot] (integral of eom state vector)
tspan = [0 10];
pinit = [0; 0; -100; 0; 0; 0; 0; 0; 0; 0; 0.1; 0];
[tx,x1] = ode45(@(t,x) LinearQuadControlQ2(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);
[tx2,x2] = ode45(@(t,x)
    QuadrotorEOM_controlQ3(t,x,m,r,km,Ix,Iy,Iz,v,mu),tspan,pinit);


xval = x1(:,1);
yval = x1(:,2);
zval = x1(:,3);
phival = x1(:,4);
thetaval = x1(:,5);
psival = x1(:,6);
uval = x1(:,7);
vval = x1(:,8);
wval = x1(:,9);
pval = x1(:,10);
qval = x1(:,11);
rval = x1(:,12);

xval2 = x2(:,1);
yval2 = x2(:,2);
zval2 = x2(:,3);
phival2 = x2(:,4);
thetaval2 = x2(:,5);
psival2 = x2(:,6);
uval2 = x2(:,7);
vval2 = x2(:,8);
wval2 = x2(:,9);
```

```
    pval2 = x2(:,10);
    qval2 = x2(:,11);
    rval2 = x2(:,12);

    prob = 0;


    figure(1)
    hold on
    %drone_plot(x1,tx,'pqr', prob);
    drone_plot2(x1,tx, x2,tx2, 'pqr', prob);
    hold off

    hold on
    figure(2)
    %drone_plot(x1,tx,'uvw',prob);
    drone_plot2(x1,tx,x2,tx2,'uvw',prob);
    hold off

    hold on
    figure(3)
    %drone_plot(x1,tx,'euler',prob);
    drone_plot2(x1,tx,x2,tx2,'euler',prob);
    hold off

    hold on
    figure(4)
    %drone_plot(x1,tx,'xyz',prob);
    drone_plot2(x1, tx,x2,tx2,'xyz',prob);
    hold off

    figure(5)
    plot3(xval,yval,zval);
    grid on
end
```

```
function [Output_Plot] = Plot12(Time,State)

t=Time;%time

x=State(:,1);%populate variables
y=State(:,2);%populate variables
z=State(:,3);%populate variables
phi=State(:,4);%populate variables
theta=State(:,5);%populate variables
psi=State(:,6);%populate variables
u=State(:,7);%populate variables
v=State(:,8);%populate variables
w=State(:,9);%populate variables
p=State(:,10);%populate variables
q=State(:,11);%populate variables
r=State(:,12);%populate variables



sgtitle('6 Experimental Drone Behavior')%main title
subplot(4,3,1)%sub plot call
plot(t,x)
ylabel('x_E [m]')
xlabel('time [s]')

subplot(4,3,2)%sub plot call
plot(t,y)
ylabel('y_E [m]')
xlabel('time [s]')


subplot(4,3,3)%sub plot call
plot(t,z)
ylabel('z_E [m]')
xlabel('time [s]')
```

```
subplot(4,3,4)%sub plot call
plot(t,phi)
ylabel('\phi [rad]')
xlabel('time [s]')


subplot(4,3,5)%sub plot call
plot(t,theta)
ylabel('\theta [rad]')
xlabel('time [s]')


subplot(4,3,6)%sub plot call
plot(t,psi)
ylabel('\psi [rad]')
xlabel('time [s]')


subplot(4,3,7)%sub plot call
plot(t,u)
ylabel('u^E [m/s]')
xlabel('time [s]')


subplot(4,3,8)%sub plot call
plot(t,v)
ylabel('v^E [m/s]')
xlabel('time [s]')


subplot(4,3,9)%sub plot call
plot(t,w)
ylabel('w^E [m/s]')
```

```
xlabel('time [s]')


subplot(4,3,10)%sub plot call
plot(t,p)
ylabel('p [rad/s]')
xlabel('time [s]')


subplot(4,3,11)%sub plot call
plot(t,q)
ylabel('q [rad/s]')
xlabel('time [s]')


subplot(4,3,12)%sub plot call
plot(t,r)
ylabel('r [rad/s]')
xlabel('time [s]')




Output_Plot=1;%lazy

end
```

```
%Part 6 Data read in and plot
%Emerson and group
%10/22/2020

clear;close;clc

data=load('RSdata_10_21_1451');% load file

```

```
%---
A=data.rt_sensor;%get data
B=data.rt_optical;%get data
C=data.rt_calib;%get data
D=data.rt_cmd;%get data
G1=data.rt_estim;%get data
%G2=data.rt_estimatedStates; %obsolete

%----
MotorForce=data.rt_motor;%motor forces get data
MotorForceTime=MotorForce.time;%motor forces get data
MotorForceStrut=MotorForce.signals;%motor forces get data
MotorForcesValues=MotorForceStrut.values;%motor forces get data
% ---

time=G1.time(:);%time call
xval_data=G1.signals.values(:,1);%get x data
yval_data=G1.signals.values(:,2);%get y data
zval_data=G1.signals.values(:,3);%get z data
phival_data=G1.signals.values(:,4);%get angle data
thetaval_data=G1.signals.values(:,5);%get angle data
psival_data=G1.signals.values(:,6);%get angle data
uval_data=G1.signals.values(:,7);%get u data
vval_data=G1.signals.values(:,8);%get v data
wval_data=G1.signals.values(:,9);%get w data
pval_data=G1.signals.values(:,10);%get p data
qval_data=G1.signals.values(:,11);%get q data
rval_data=G1.signals.values(:,12);%get r data
%---

%--
LLL=length(xval_data);%get length
x1_data=zeros(LLL,12);%make blank matrix
x1_data(:,1)=xval_data;%populate matrix
x1_data(:,2)=yval_data;%populate matrix
```

```
x1_data(:,3)=zval_data;%populate matrix
x1_data(:,4)=phival_data;%populate matrix
x1_data(:,5)=thetaval_data;%populate matrix
x1_data(:,6)=psival_data;%populate matrix
x1_data(:,7)=uval_data;%populate matrix
x1_data(:,8)=vval_data;%populate matrix
x1_data(:,9)=wval_data;%populate matrix
x1_data(:,10)=pval_data;%populate matrix
x1_data(:,11)=qval_data;%populate matrix
x1_data(:,12)=rval_data;%populate matrix


plot=Plot12(time,x1_data)%plot call





















%
```

```
% t=2600;
% State=x1_data;
%
%
% x=State(:,1);
% y=State(2);
% z=State(3);
% phi=State(4);
% theta=State(5);
% psi=State(6);
% u=State(7);
% v=State(8);
% w=State(9);
% p=State(10);
% q=State(11);
% r=State(12);
%
%
% subplot(4,3,1)
% plot(t,x)
%
% subplot(4,3,2)
% plot(t,y)
%
% subplot(4,3,3)
% plot(t,z)
%
% subplot(4,3,4)
% plot(t,phi)
%
% subplot(4,3,5)
% plot(t,theta)
%
% subplot(4,3,6)
% plot(t,psi)
```

```
%
% subplot(4,3,7)
% plot(t,u)
%
% subplot(4,3,8)
% plot(t,v)
%
% subplot(4,3,9)
% plot(t,w)
%
% subplot(4,3,10)
% plot(t,p)
%
% subplot(4,3,11)
% plot(t,q)
%
% subplot(4,3,12)
% plot(t,r)
%
%
%
% % plot=Plot12(time,x1_data)
%
% % subplot(4,3,1)
%  % plot(time,xval_data)
% %
% % subplot(4,3,2)
% % plot(time,xval_data)
% %
% % subplot(4,3,3)
% % plot(time,xval_data)
% %
% % subplot(4,3,4)
% % plot(time,xval_data)
%
```

```
%
% %Plot Function(time,matrix12)
```