University of Colorado - Boulder

ASEN 3128: Aircraft Dynamics

Lab 7

Fixed-Wing Aircraft Linear Equations

Author:
Shawn Stone[a]

Author:
Anthony Danna[b]

Author:
Connor O'Reilly[c]

Author:
Ryan Collins[d]

[a] 109008271
[b] 810653532
[c] 107054811
[d] 104768223

Professor:
Eric Frew

December 6, 2020

Ann and H.J. Smead
Aerospace Engineering Sciences
UNIVERSITY OF COLORADO **BOULDER**

Contents

List of Figures

I. Problem 1: Modifying AircraftLinearModel.m

The Matlab function *AircraftLinearModel.m* was updated to include the control derivatives for the lateral and longitudal B matrices.

Lateral Equations:

$$\dot{\mathrm{x}}_{lat} = \mathbf{A}_{lat}\mathrm{x}_{lat} + \mathbf{B}_{lat}\mathbf{u}_{lat}$$

where,

$$\mathbf{B}_{\text{lat}} = \begin{pmatrix} 0 & \frac{Y_{\delta_r}}{m} \\ \Gamma_3 L_{\delta_a} + \Gamma_4 N_{\delta_a} & \Gamma_3 L_{\delta_r} + \Gamma_4 N_{\delta_r} \\ \Gamma_4 L_{\delta_a} + \Gamma_8 N_{\delta_a} & \Gamma_4 L_{\delta_r} + \Gamma_8 N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

To make computations easier,

$$\Gamma = I_x I_z - I_{xz}^2 \, , \Gamma_3 = \frac{I_z}{\Gamma} \, , \Gamma_4 = \frac{I_{xz}}{\Gamma} \, , \Gamma_8 = \frac{I_x}{\Gamma}$$

Longitudal Equations:

$$\dot{\mathrm{x}}_{lon} = \mathbf{\Lambda}_{lon}\mathrm{x}_{lon} + \mathbf{c}_{lon}$$

where,

$$\mathbf{B}_{\text{lon}} = \begin{pmatrix} \frac{X_{\delta_c}}{m} & \frac{X_{\delta_t}}{m} \\ \frac{Z_{\delta_c}}{m - Z_{\dot{w}}} & \frac{Z_{\delta_t}}{m - Z_{ii}} \\ \frac{M_{\delta_e}}{I_y} + \frac{M_{\dot{w}}}{I_y}\frac{Z_{\delta_s}}{(m - Z_{\dot{w}})} & \frac{M_{\delta_1}}{I_y} + \frac{M_w}{I_y}\frac{Z_{\delta_t}}{(m - Z_{\dot{w}})} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Both matrices where implemented into *AircraftLinearModel.m* and can be seen in Appendix B.

II. Problem 2: Yaw Damper

The modified closed loop state space matrix is shown below in equation 1, where **K** is a zeros vector with the kr gain in its 3rd position and $\mathbf{B}_{\mathbf{rud}}$ is the 2nd column of the $\mathbf{B}_{\mathbf{lat}}$ matrix.

$$(\mathbf{A}_{\text{lat}} - \mathbf{B}_{\text{rud}}\mathbf{K}) \tag{1}$$

Taking the eigenvalues of the new closed loop state space matrix resulted in the new poles of the system, which were then iterated with a kr gain ranging from 10 to -10 to find a resulting dutch roll pole with 50% greater damping ratio than the original. The result is the root locus plot shown below in figure 1. The red poles represent the dutch roll mode, the green represent spiral mode, and the blue represent roll mode. The black crosses represent the eigenvalues resulting from a kr gain that dampens the dutch roll mode by 50% more than the original. This kr gain was found to be -3.45.

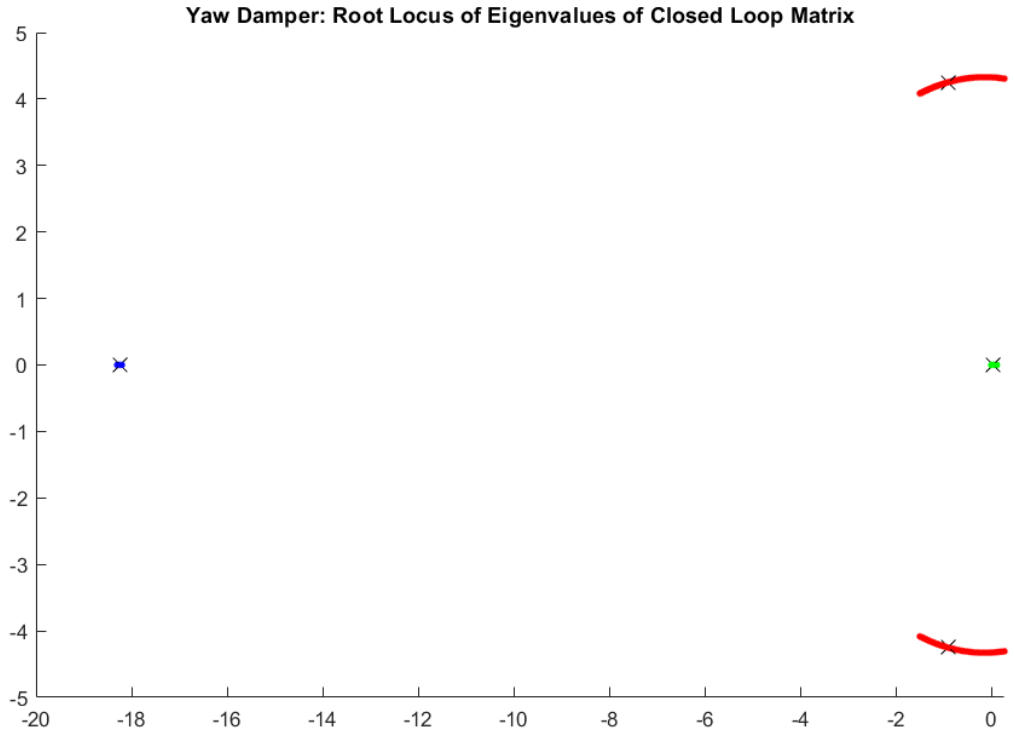

Figure 1. Root Locus of Closed Loop State Space Matrix

Looking at the impact of the yaw damper to each of the lateral modes, it is clear that the only significant change occurs with respect to the dutch roll mode, where a decrease in the kr gain (more negative) steadily increased both the damping and natural frequency of the mode. The roll and spiral modes see very minimal change in their poles, however, the spiral mode does become stable after a significant enough decrease in the kr gain (at approximately kr = -8).

A. Yaw Damper in the Full Nonlinear Dynamics

Using the kr gain of -3.45 from the linear design, the yaw damper was implemented into the full nonlinear aircraft dynamics. Shown below in figures 2 through 9 are the result of this implementation, where black lines represent the unmodified nonlinear dynamics and red represents the nonlinear dynamics with the yaw damper implemented. This simulation gave both systems an initial yaw rate disturbance of 5 deg/s. It can be seen that the yaw damper increases the damping of the system, although it is unable to stop the longer term spiral and phugoid behaviors (which is expected).

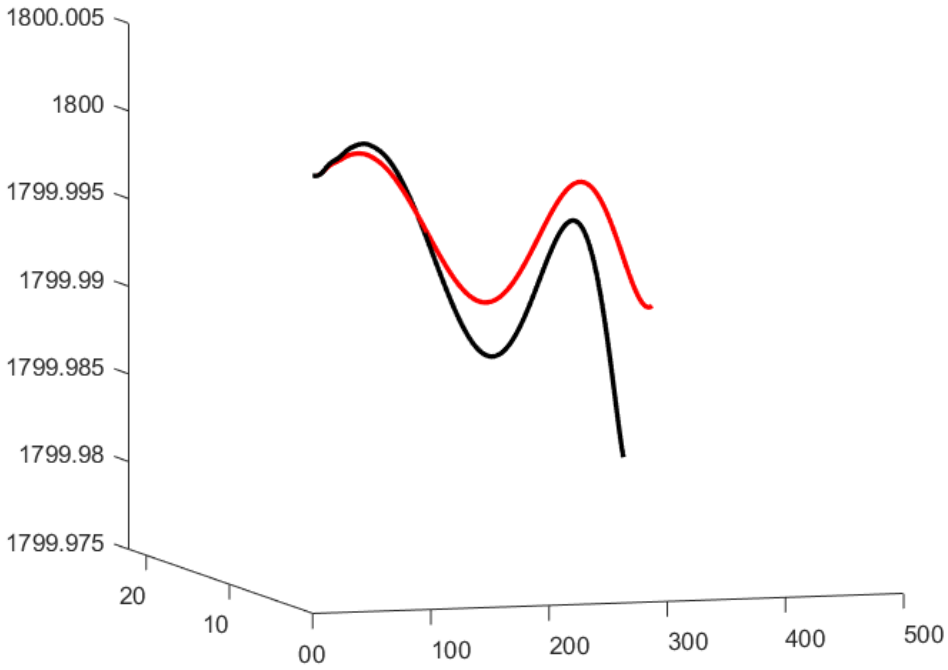

Figure 2. 3D Flight Path with Yaw Rate Disturbance

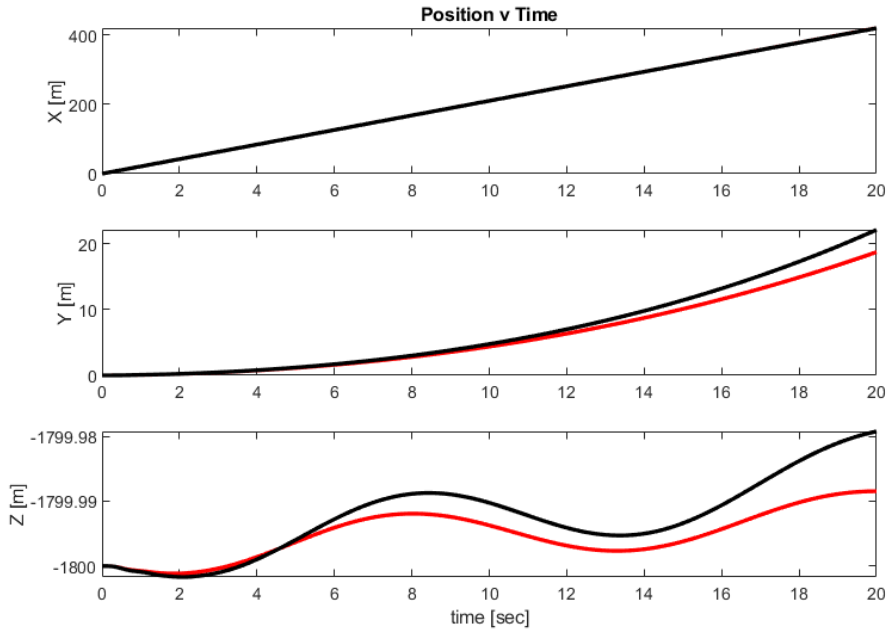

Figure 3. Position with Yaw Rate Disturbance

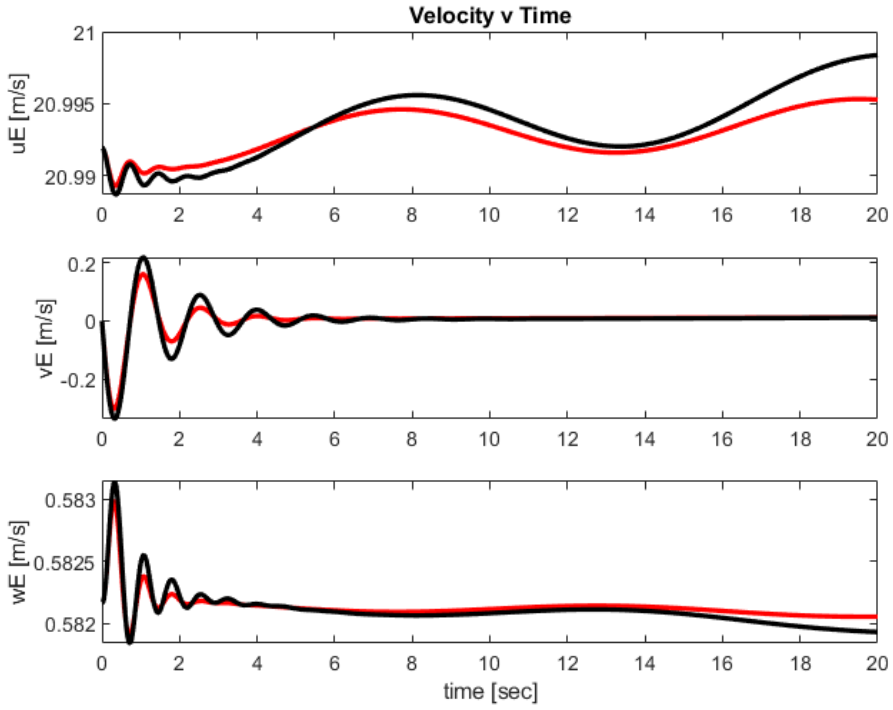

Figure 4. Velocity with Yaw Rate Disturbance

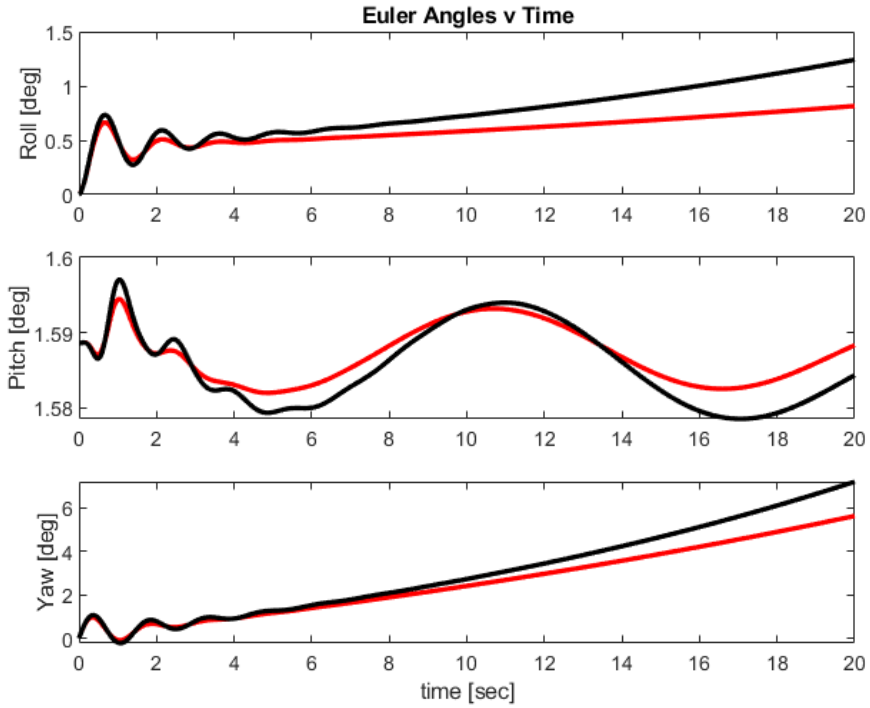

Figure 5. Euler Angles with Yaw Rate Disturbance

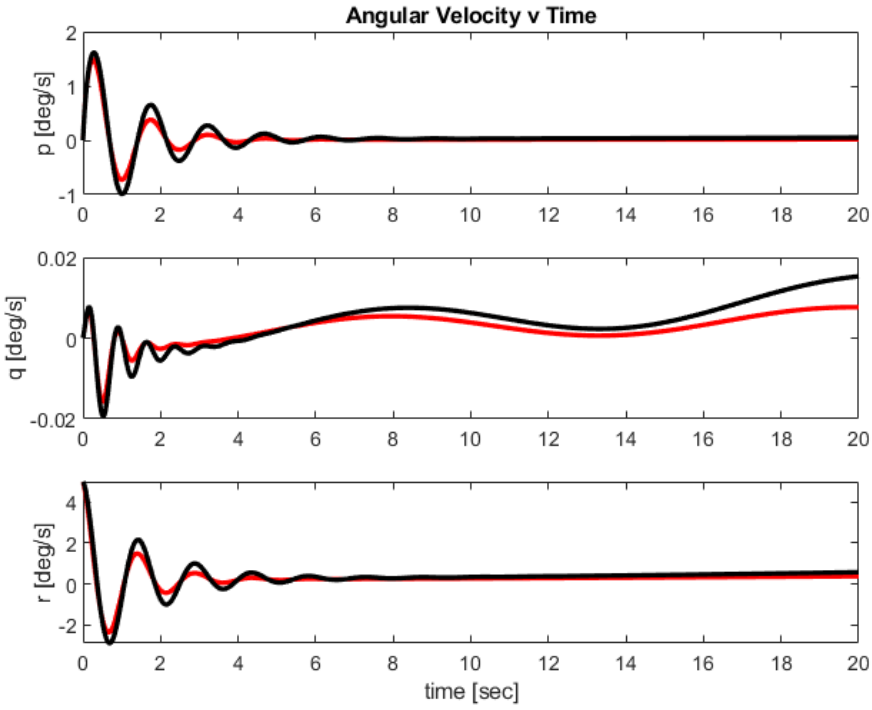

Figure 6. Angular Velocity with Yaw Rate Disturbance

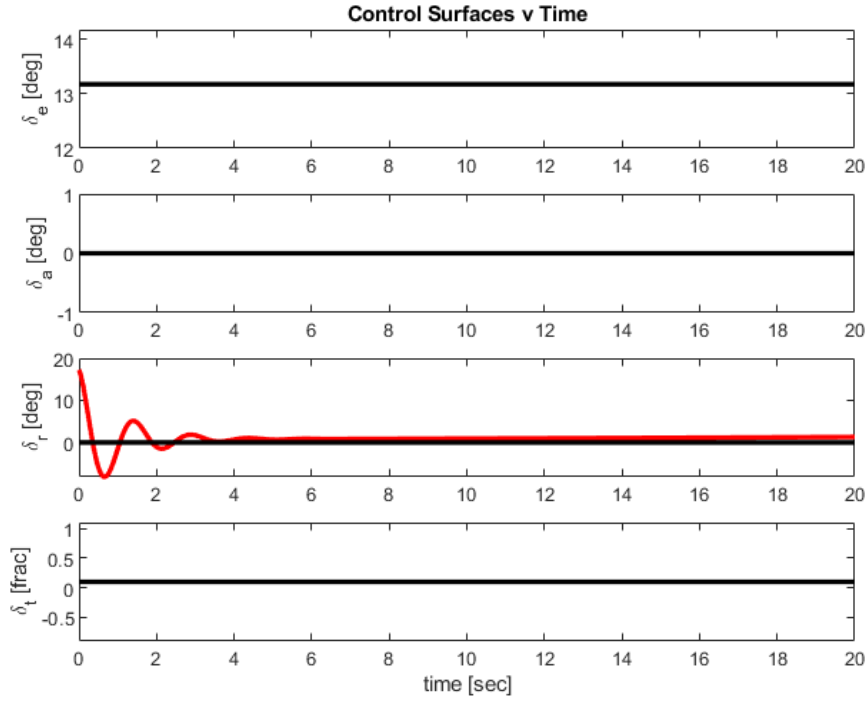

Figure 7. Control Surfaces with Yaw Rate Disturbance

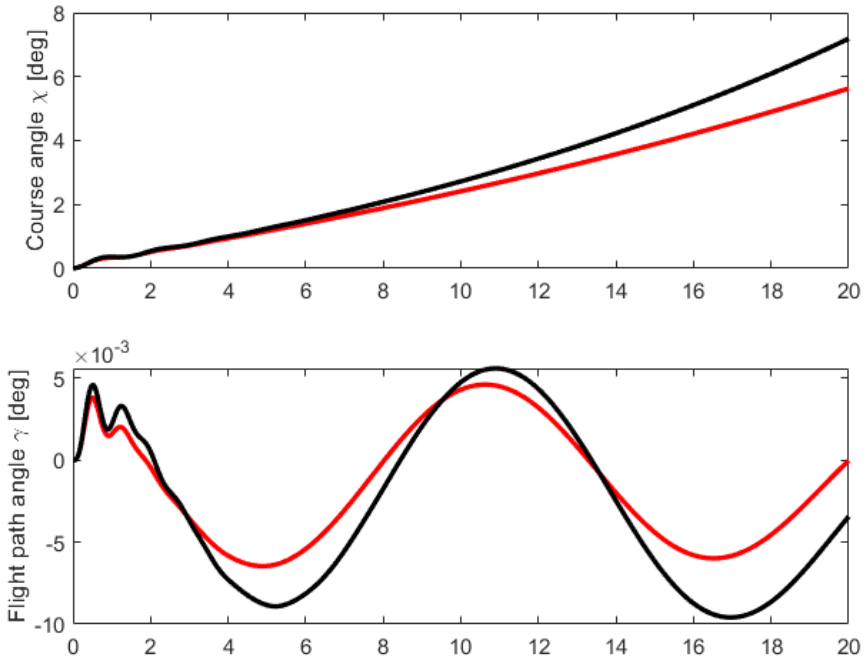

Figure 8. Course and Flight Path Angle with Yaw Rate Disturbance

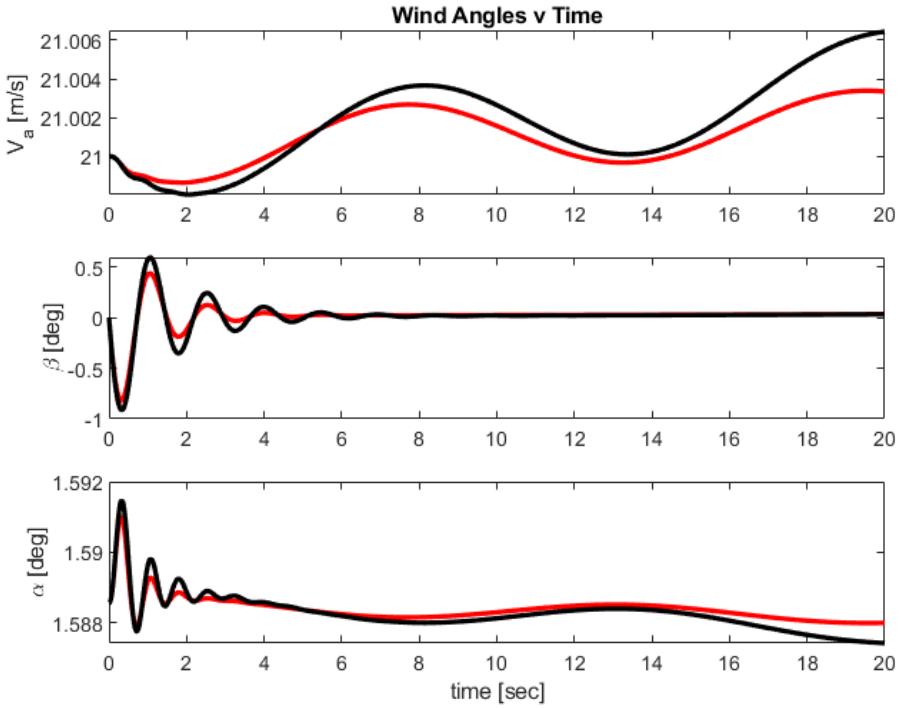

Figure 9. Wind Angles with Yaw Rate Disturbance

III. Problem 3: Pitch Controller

The general structure for the creation and optimization of gains for a pitch control was actually extremely similar to those seen in Problem 1. That being the exception of the transfer functions used to model the system and it's reaction to gain values for the pitch control. A transfer function was used and its poles were extracted and utilized as pseudo- eigenvalues. From here, a root locus was used in which both gain values were varied between -10 and 10. From there, the poles were plotted on a real and imaginary plot.

A. Pitch Control Law from Linear Model

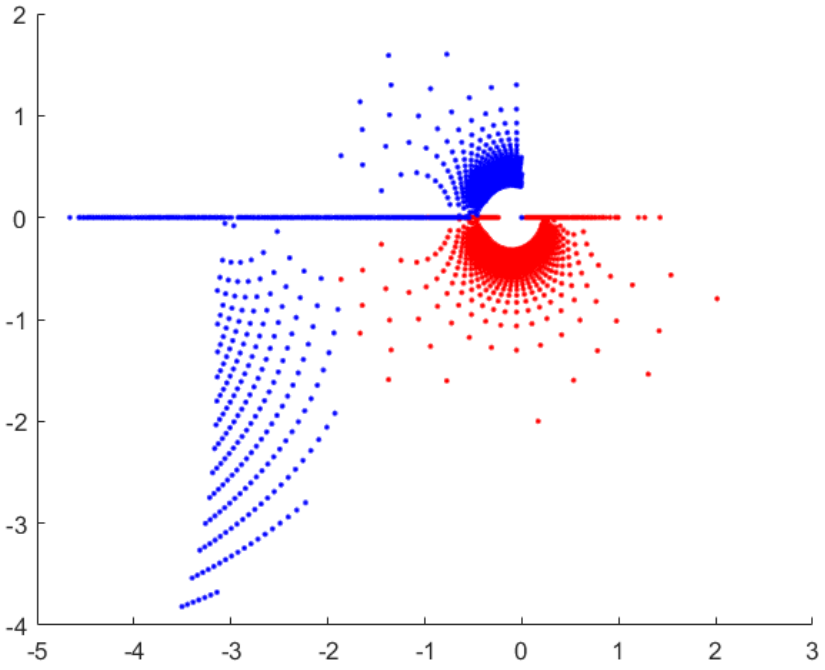

Figure 10. Pitch Gain Dual Root Locus

As seen in the plot, both the blue and red represent the phugoid mode of the simulated eigenvalues. From here an optimum value was determined by comparing the magnitude of the damping ratio and natural frequency. The gain values with the largest corresponding differences between the non-controlled system are considered to be the ideal cases. In our circumstance the gain values turned out to be K1=9.9 and K2=-7.8 for the q and theta "values" respectively.

B. New State Space Matrix

As mentioned previously these values of the initial system without a pitch control were compared to that of the natural frequency and damping ratios of the controlled system. The difference in the two values was a natural frequency increase of 2.361 and a damping ratio increase of 0.167. If the gain value "range" was expanded it is believed that damping could further be improved, although it was determined that it was not overly necessary to over-"damp" the system as it is a phugoid perturbation, and more aggressive pitch controllers are not necessary.

C. Pitch Controller in the Full Nonlinear Dynamics

In the below state space models, the red line simulates the controlled system, and the black represents the nonlinear system. The nonlinear system will eventually reach a settled state, but is not nearly as effective as the linearized control method.

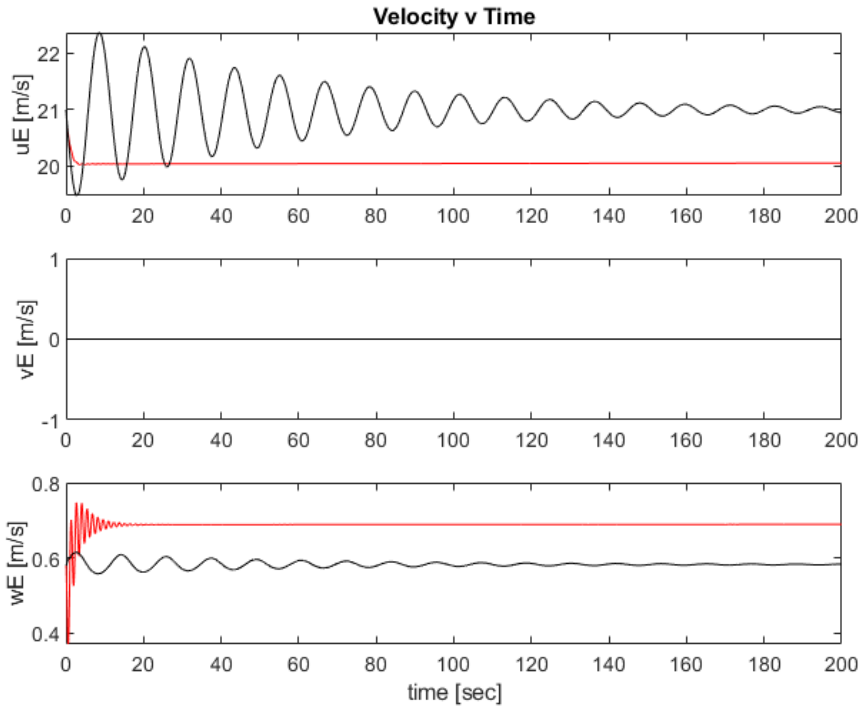

Figure 11. Pitch Controller vs Nonlinear - Velocity vs Time

Naturally the linear system is not able to model the changes in the forward velocity, but is able to comprehend changes in the vertical velocity.

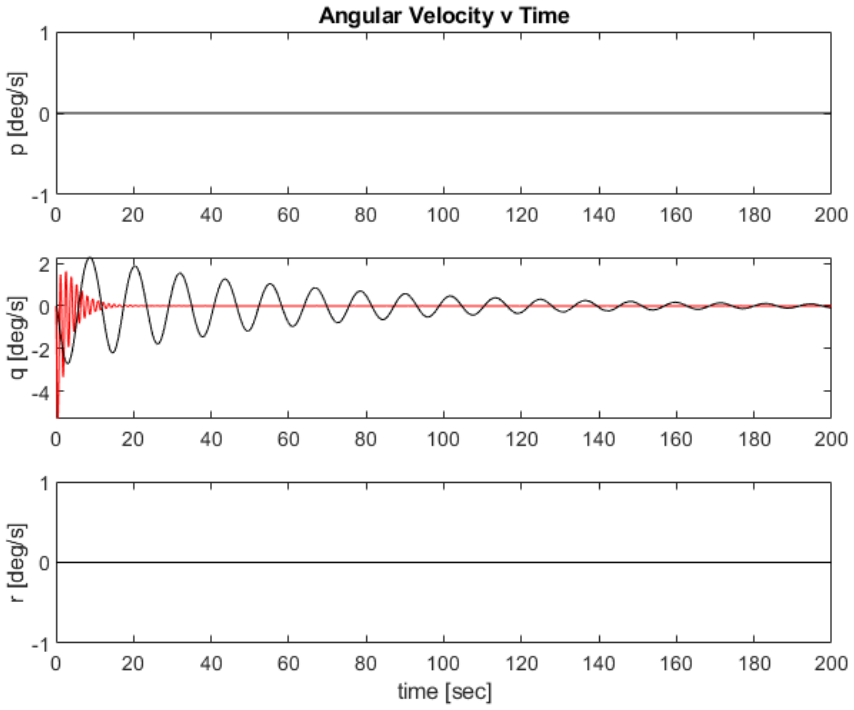

Figure 12. Pitch Controller vs Nonlinear - Angular Velocity vs Time

The above image acting as the "primary" plot of concern, we notice that the controlled plot is almost fully settled by the time that the non-linear system makes its first period.

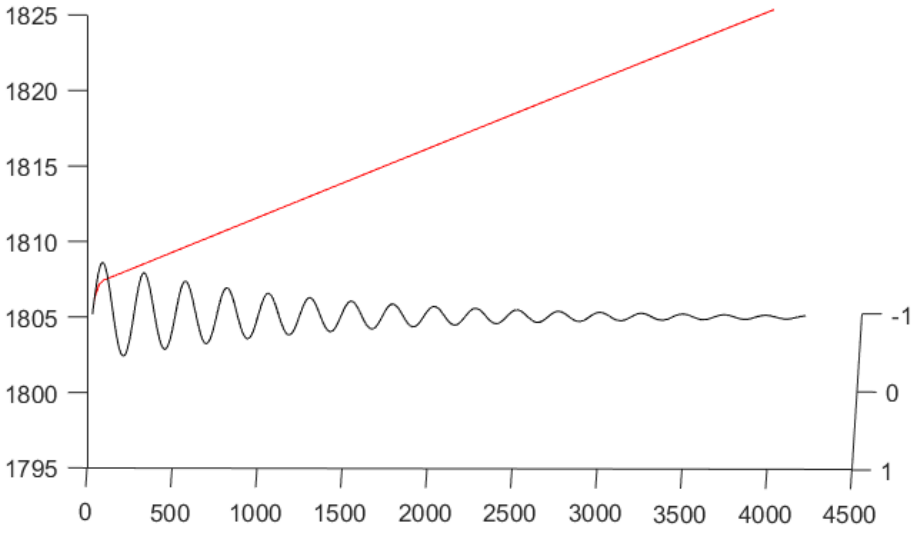

Figure 13. Pitch Controller vs Nonlinear - 3D Plot

It is worth noting that similar to the previous lab, the vertical position does not see a correction similar to how the angular velocities do. It is worth noting that the change in position given the time step of 200 seconds is rather inconsequential.

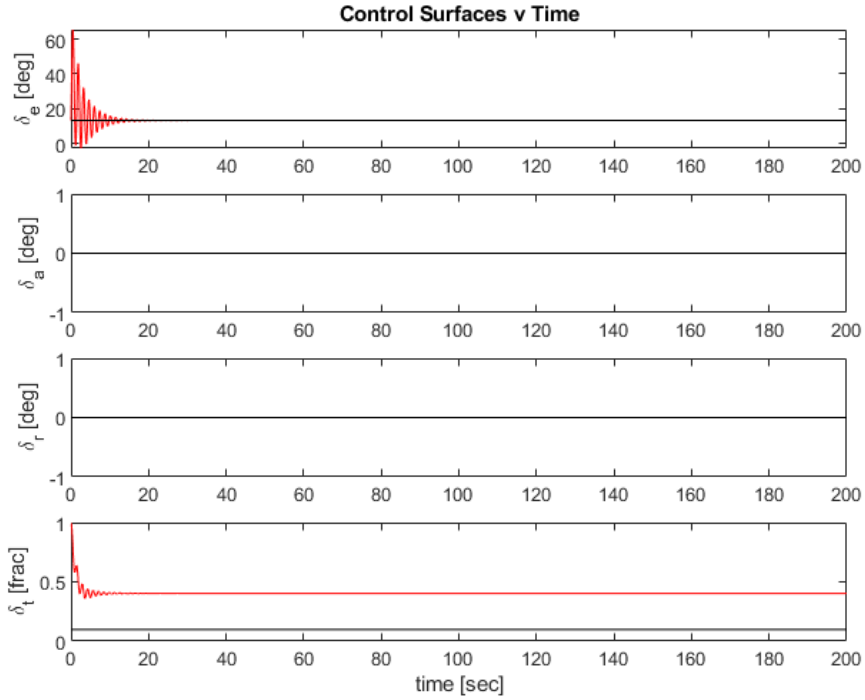

Figure 14. Pitch Controller vs Nonlinear - Control Surfaces

Quite notably in the above figure, it is important to not over fit your gains in order to create elevator deflection that is implausible for the system. While perhaps on the steeper end, a 60 degree deflection is "possible" in the correct configurations of aircraft.

IV. Problem 4: Roll Controller

A Roll controller was designed based on the pure roll approximation. The controller consisted of an inner loop and outer loop.

Pure Roll Approximation:

$$\Delta\dot{\mathrm{p}} = \mathbf{L_p}\Delta\mathrm{p}$$

$$\Delta\dot{\phi} = \Delta p$$

A. Inner-Loop Controller Design

The inner-loop of the controller fed back the roll rate to the aileron.

Inner Loop Controller:

$$\delta_a = k_a(p_c - \Delta p)$$

B. Outer-Loop Controller and Gains

The outer loop of the controller fed back the roll angle to the commanded roll rate (p_c).

Outer-Loop Controller:

$$p_c = k_p(\phi_c - \Delta\phi)$$

To determine the aileron gain value k_a first the natural dynamics of the system were analyzed to determine the Time-to-Half (t_{half}). This was determined to be 0.038 s. The response of this system could then be improved by using the time constant and solving for k_a with an improved time to half value.

k_a Time Constant:

$$k_a = \frac{ln(0.5) - L_p t}{-L_{\delta a} t}$$

At first we sought to improve this by an order of magnitude to 0.004 s. This resulted in vastly improved response to perturbations in roll angle and roll rate, but also in impossibly high aileron deflections. After some trial and error it was determined that a 20 % reduction in the time to half resulted in improved response while maintaining realistic aileron deflections.

A similar program was followed in selecting k_p for the outer loop using the equation below. The time to half of this loop was selected to be 5X that of the inner-loop.

k_p Time Constant:

$$k_p = -\frac{ln(0.5)}{t}$$

This process resulted in a selection of $k_a = -1.5979$ and $k_p = 4.5640$.

In order to ensure that reasonable performance was attained k_a and k_p gain values were tested to ensure that aileron deflections were reasonable. We chose 45° of deflection as the upper bound. Additionally, we compared the response of the controller to that of the natural system to ensure that the controller provided faster, and better damping of roll and roll rate perturbations. In figure 15 below the blue lines represent the drone's natural response to a 5 degree perturbation in the roll angle ϕ, while the red lines indicate the performance of the drone with the roll controller. From these graphs it can be seen that the controlled system recovers quickly to a trim condition, whereas the uncontrolled system diverges from the trim state and enters a death spiral.

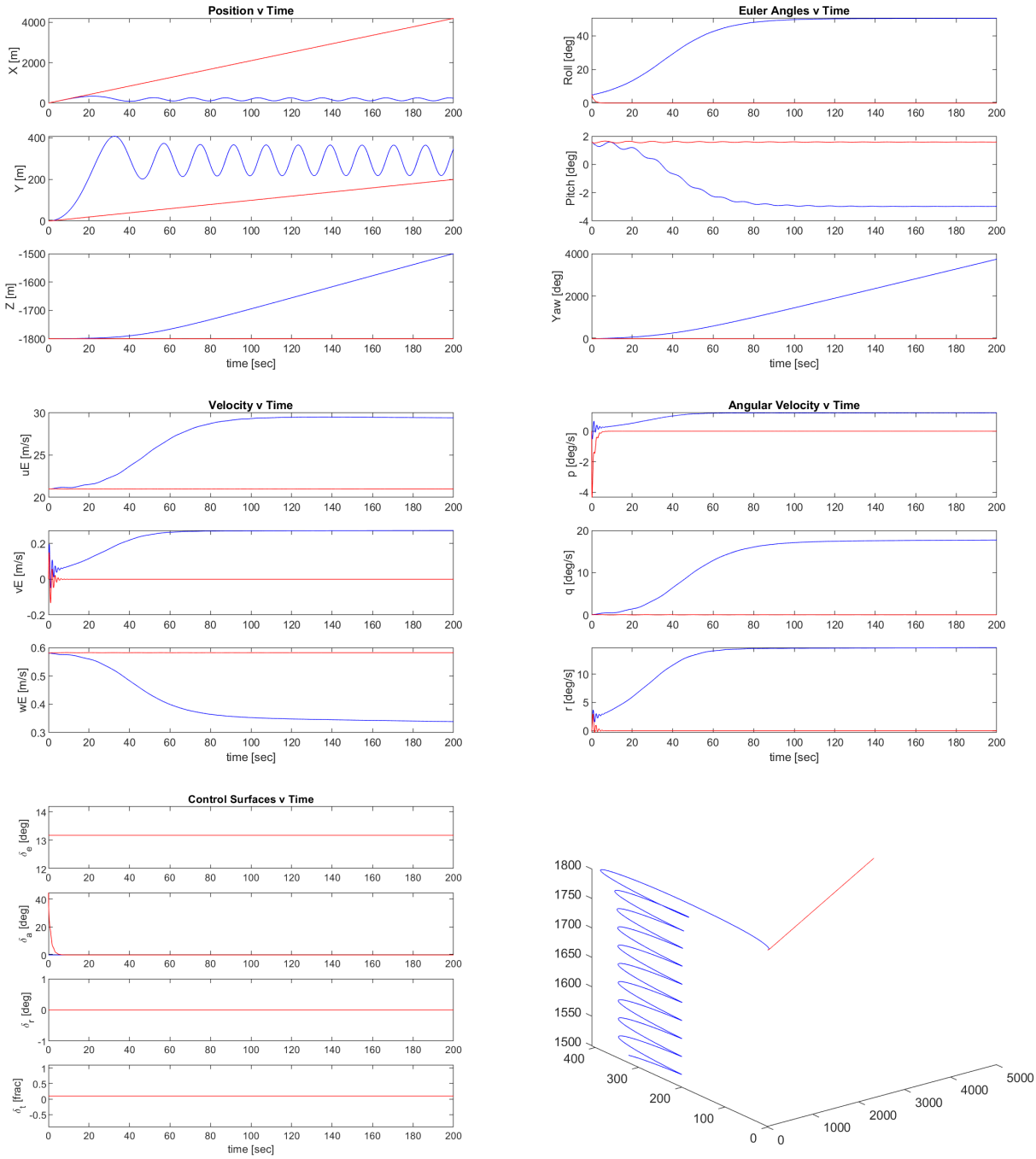

Figure 15. Controlled and Uncontrolled Response to Roll Angle Perturbation

C. Roll Control for full nonlinear dynamics

Once the roll controller had been tested, it was combined with the Yaw Damper from Part 2 above. With the combined controllers the drone was put into a coordinated turn with radius 1000m from a trim state at 21 m/s and 1800m altitude. At these conditions the proper bank angle was determined to be 2.57° using the equations below.

$$\phi = atan(\frac{\omega u_0}{gcos(\theta)})$$

where

$$\omega = \frac{u_0}{R_{turn}}$$

In the figure below the results of the coordinated turn with 2.57° bank angle are presented with the blue lines showing the system without the yaw damper and the red lines showing the system with the yaw damper.

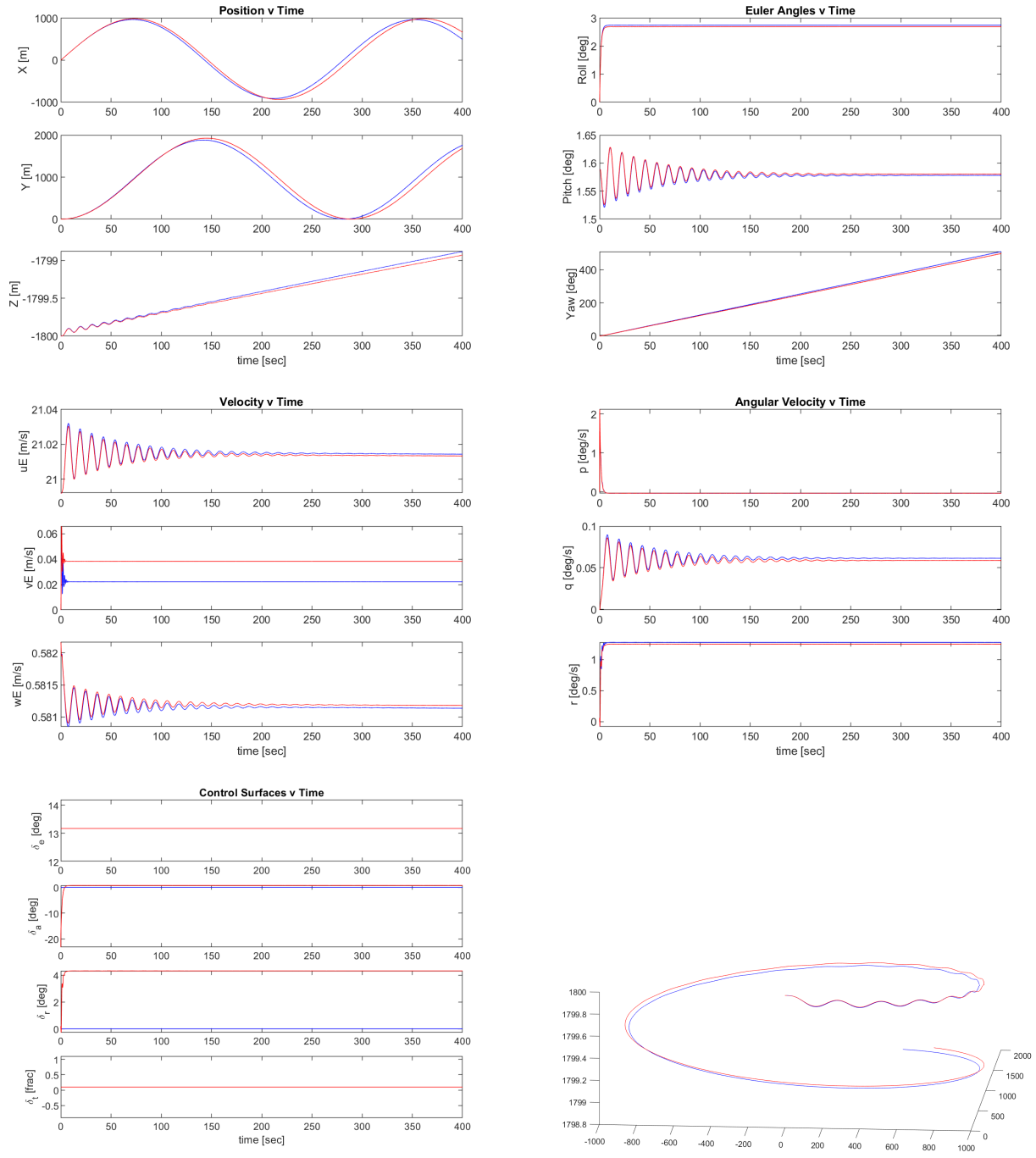

Figure 16. Coordinated Turn with and without Yaw Damper

In both cases the drone achieves a coordinated turn with a slight excitation of the phugoid mode evident at the beginning of the turn. Additionally, the drone with and without the yaw damper loses only about 1 meter of altitude after a 360° turn. From the Euler angle plot it can be seen that the phugoid pitch oscillations are slightly greater in magnitude and that the roll angle is slightly larger in the case without the yaw damper. This occurs despite the aileron deflections being identical in each case. Finally, in the case without the yaw damper the aircraft descends, and turns at a slightly

greater rate than with the yaw damper. This is expected as rudder deflections should act to reduce air speed via greater drag.

The coordinated turn condition was verified by examining the ω_E vector as well as by using the condition below:

$$tan(\phi) = \frac{\omega u_0}{g cos(\theta)}$$

In a coordinated turn $\omega_E = [0, 0, \omega]$ where ω is the magnitude of ω_E. This can be seen below in figure 17. Although at the start p_E and q_e are not exactly 0 they quickly approach it within a few seconds.

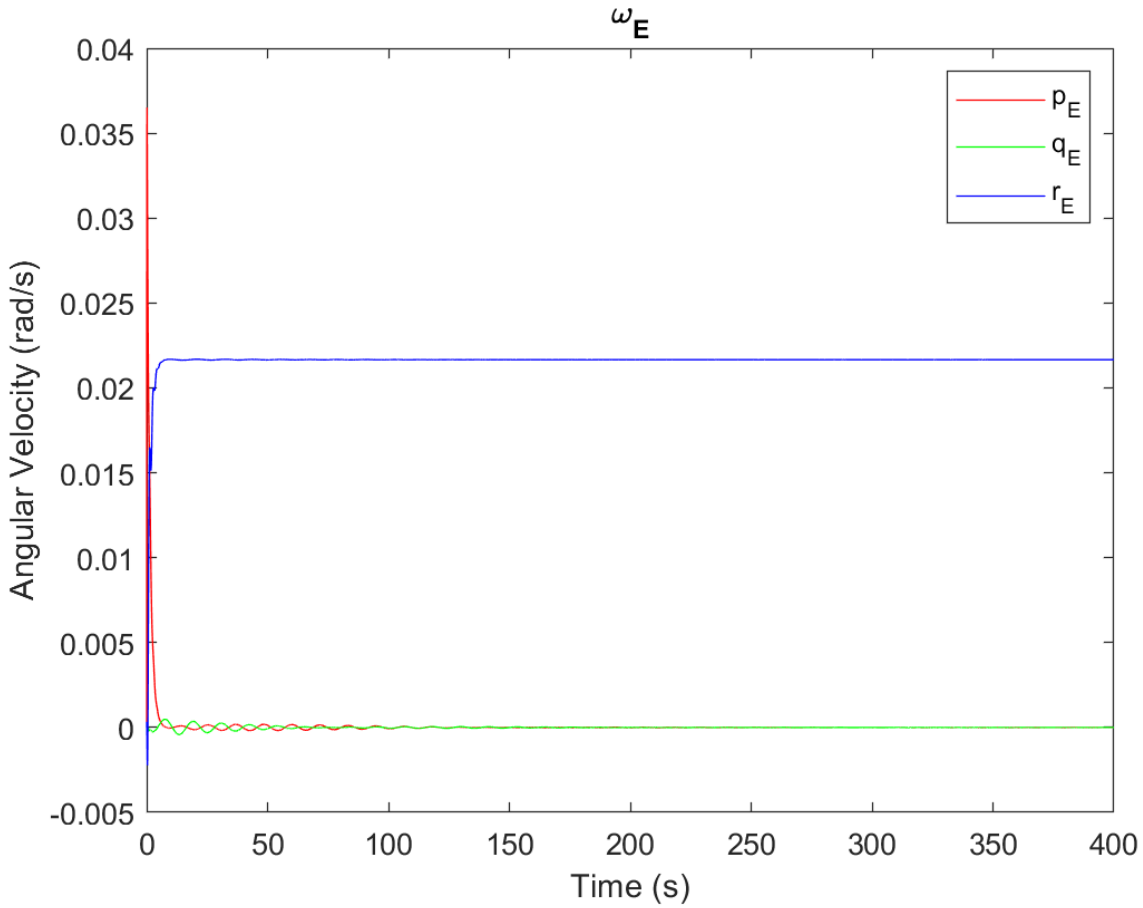

Figure 17. Angular Velocity in Inertial Frame

In figure 18 the percent difference between the left and right sides of $tan(\phi)$ equation above is plotted. Again, the difference is large in the beginning but soon settles near 3.5% difference.

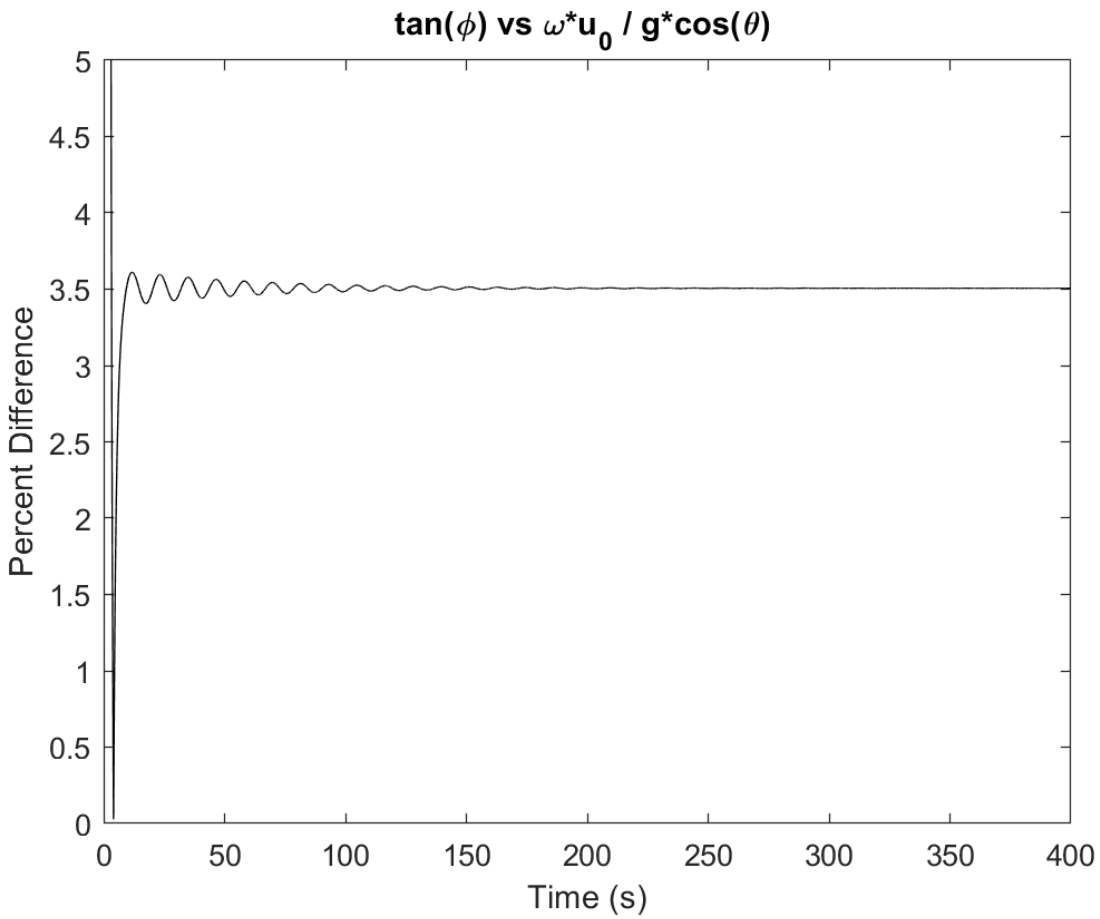

Figure 18. Percent Difference

From both of these graphs it is evident that the turn is mostly coordinated after an initial 3 second period of uncoordinated turning.

D. Improvement of Yaw Damper

In order to improve the yaw damper a washout filter could be added to the system. This would prevent the yaw damper from fighting against the roll controller. This filter would allow slower inputs from the pilot through, while damping out higher frequency oscillations.

V. Appendix A: Team Participation

	Plan	Model	Experiments	Results	Report	Code	ACK
Shawn Stone	1	2	1	1	2	1	SS
Anthony Danna	2	1	1	1	1	2	AD
Connor O'Reilly	1	1	2	1	1	1	CB
Ryan Collins	1	1	1	2	1	1	RC

VI. Appendix B: Code

```
function aileron_perturb = RollAttitudeControl(ka, kp, phi_c, phi, delta_p)

    Pc = kp * (phi_c - phi);

aileron_perturb = ka*(Pc - delta_p);



function xdot = AircraftEOMControl(t,aircraft_state,aircraft_surfaces0,wind_inertial,aircraft_parameters)


pos_inertial = aircraft_state(1:3,1);
euler_angles = aircraft_state(4:6,1);
vel_body = aircraft_state(7:9,1);
omega_body = aircraft_state(10:12,1);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Control law
%%% [This shows implementation of the pitch control. STUDENTS EDIT for
%%% other controllers]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
theta_c = 5*pi/180;
kq = 1; %NOT REASONABLE VALUES
kth = 1; %NOT REASONABLE VALUES

% %YAW DAMPER CONTROL
kr = -3.45; %determined from root locus
dr = -kr*omega_body(3); %YawDamper(kr, omega_body(3));
aircraft_surfaces = aircraft_surfaces0 + [0; 0; dr; 0];

%%%%%%%%%ROLL%%%%%%%%%VARIABLEs%%%%%%%%%%%%%%
%Vars for calculating a coord turn of radius r_turn
r_turn = 1000;
u0 = norm(vel_body);
w = u0/r_turn;

phi_c = deg2rad( 2.6 ); %Command roll angle degrees
phi_c = atan( (u0*w) / (aircraft_parameters.g*cos(euler_angles(2)))    ); %Corrd turn bank angle
ka = -1.9579; %Aileron Control gain
kp = 4.5640; %Roll rate control gain

delta_p = omega_body(1); %From trim condition, delta_p IS the roll rate (p), bc at trim p = 0;
phi = euler_angles(1);
aileron_perturb = RollAttitudeControl(ka, kp, phi_c, phi, delta_p); % put this into aircraft surfaces

% elev_perturb = PitchAttitudeControl(theta_c, aircraft_state(5), aircraft_state(11), kth, kq);
elev_perturb = 0;

aircraft_surfaces = aircraft_surfaces0 + [elev_perturb; aileron_perturb; dr; 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%
%%% Kinematics
vel_inertial = TransformFromBodyToInertial(vel_body, euler_angles);
euler_rates = EulerRatesFromOmegaBody(omega_body, euler_angles);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aerodynamic force and moment
density = stdatmo(-pos_inertial(3,1));
```

```
[fa_body, ma_body, wind_angles] = AeroForcesAndMoments_BodyState_WindCoeffs(aircraft_state, aircraft_surfaces, wi


%%% Gravity
fg_body = (aircraft_parameters.g*aircraft_parameters.m)*[-sin(euler_angles(2));sin(euler_angles(1))*cos(euler_ang


%%% Dynamics
vel_body_dot = -cross(omega_body, vel_body) + (fg_body + fa_body)/aircraft_parameters.m;


inertia_matrix = [aircraft_parameters.Ix 0 -aircraft_parameters.Ixz;...
                    0 aircraft_parameters.Iy 0;...
                    -aircraft_parameters.Ixz 0 aircraft_parameters.Iz];

omega_body_dot = inv(inertia_matrix)*(-cross(omega_body, inertia_matrix*omega_body) + ma_body);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% State derivative
xdot = [vel_inertial; euler_rates; vel_body_dot; omega_body_dot];

end



%Shawn Stone, Anthony Danna, Connor Baldwin, Ryan Collins
%ASEN 3128 - Lab 6: Fixed-Wing Aircraft Linear Equations
%Created: 11/5/20
%Last Edited: 11/5/20
clc; clear all; close all;

recuv_tempest;
ap = aircraft_parameters;

Va_trim = 21;
h_trim = 1800;

wind_inertial = [0;0;0];

trim_definition = [Va_trim; h_trim];

%%% Use full minimization to determine trim
[trim_variables, fval] = CalculateTrimVariables(trim_definition, aircraft_parameters);
[trim_state, trim_input]= TrimStateAndInput(trim_variables, trim_definition);
[Alon, Blon, Alat, Blat] = AircraftLinearModel(trim_definition, trim_variables, aircraft_parameters);


%% %%%%%%%%%%%%%%%%%%%%%%%%%%Problem 3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Helper code for Problem 3. Look inside the AircraftEOMControl function
% for hints on how to set up controllers for Problem 2 and Problem 4.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % Problem 3.1 Solve for Gain values using iterative looping
[num_elev2pitch, den_elev2pitch] = ss2tf(Alon(1:4,1:4), Blon(1:4,1), [0 0 0 1],0);

%% Controller
% matx=[];
% AllPitch=[];
%
% tic
%
% for (i=1:50)
%     for (j=1:50)
%         kq = (25-i);
%         kth = (25-j);
```

```
%         num_c = [kq kth];
%         den_c = 1;
%
%         %%% Closed loop transfer function
%         pitch_cl = feedback(1, tf(conv(num_c, num_elev2pitch), conv(den_c, den_elev2pitch)));
%         AllPitch=[AllPitch ; pitch_cl];
%         [num_cl, den_cl] = tfdata(pitch_cl,'v');
%
%         %%% Poles of the closed loop (linear) system. Now do the same with the
%         %%% state space model.
%         matx=[matx , [roots(den_cl);kq;kth] ];
%     end
% end
% damp=[];
% Wn=[];
% TTst=[];
% for(i=1:length(matx))
%     if(real(matx(3,i))>0)
%        matx(3,i)=0;
%     end
%     wn=sqrt(real(matx(3,i))^2+imag(matx(3,i))^2);
%     Wn=[Wn , wn];
%
%     dmp=sqrt(1-(imag(matx(3,i))/wn)^2);
%     if dmp~=1
%         test=wn/imag(matx(3,i));
%         TTst=[TTst , test];
%         damp=[damp , dmp];
%     else
%         TTst=[TTst , 0];
%         damp=[damp , 0];
%     end
% end
%
% toc
%
% [max, indx]=min(TTst);
%
% k1_ideal=matx(5,indx);
% k2_ideal=matx(6,indx);
%
%
% figure(10)
% hold on
% scatter(real(matx(4,:)), imag(matx(4,:)), 'r.')
% scatter(real(matx(3,:)), imag(matx(3,:)), 'b.')
%


%Temp Values to test 3.b
kq_ideal= 9.9;
kth_ideal= -7.8;
%% Part 3.b - Apply Controll laws to 'large matrix'

Test=eig(Alon);
Wn_orgnl=sqrt(real(Test(5))^2+imag(Test(5))^2);
dmp_orgnl=real(Test(5))/Wn_orgnl;

%Composed K matrix
K=[0,0,kq_ideal, kth_ideal,0,0].';
test=zeros(6,4);
modMatx=[Blon.*K , test];
closed_loop_matrix = Alon-modMatx; %Construct Closed loop matrix
eig_clsd=eig(closed_loop_matrix);
Wn_idl=sqrt(real(eig_clsd(5))^2+imag(eig_clsd(5))^2);
dmp_idl=real(eig_clsd(5))/Wn_idl;

delta_lon_wn= Wn_idl - Wn_orgnl;
```

```
delta_lon_zeta=dmp_idl-dmp_orgnl;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Full sim in ode45
aircraft_state0_perturb = trim_state + [0, 0, 0, 0, deg2rad(5), 0, 0, 0, 0, 0, 0, 0]';
aircraft_state0 = trim_state;
control_input0 = trim_input;

tfinal = 200;
TSPAN = [0 tfinal];
[TOUT2,YOUT2] = ode45(@(t,y) AircraftEOMControl(t,y,control_input0,wind_inertial,aircraft_parameters),TSPAN,aircr
[TOUT,YOUT] = ode45(@(t,y) AircraftEOM(t,y,control_input0,wind_inertial,aircraft_parameters),TSPAN,aircraft_state

%trunc=length(TOUT);
%TOUT2=TOUT2(1:trunc,1);
%YOUT2=YOUT2(1:trunc,1:12);

for i=1:length(TOUT2)
     control_input_perturb = control_input0 + [ -kq_ideal*YOUT2(i,11), 0, 0, -kth_ideal*YOUT2(i,5)]';
    UOUT2(i,:) = control_input_perturb';
end

for i=1:length(TOUT)
    UOUT(i,:) = control_input0';
end

PlotAircraftSim(TOUT2,YOUT2,UOUT2,wind_inertial,'r') %Control
PlotAircraftSim(TOUT,YOUT,UOUT,wind_inertial,'k') %nonlin



function xdot = AircraftEOMControl(t,aircraft_state,aircraft_surfaces0,wind_inertial,aircraft_parameters)


pos_inertial = aircraft_state(1:3,1);
euler_angles = aircraft_state(4:6,1);
vel_body = aircraft_state(7:9,1);
omega_body = aircraft_state(10:12,1);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Control law
%%% [This shows implementation of the pitch control. STUDENTS EDIT for
%%% other controllers]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
theta_c = 5*pi/180;
kq = 1; %NOT REASONABLE VALUES
kth = 1; %NOT REASONABLE VALUES

% %YAW DAMPER CONTROL
% kr = -3.45; %determined from root locus
% dr = -kr*omega_body(3); %YawDamper(kr, omega_body(3));
% aircraft_surfaces = aircraft_surfaces0 + [0; 0; dr; 0];
dr = 0;
%%%%%%%%%ROLL%%%%%%%%%VARIABLEs%%%%%%%%%%%%%%

%Vars for calculating a coord turn of radius r_turn
r_turn = 1000;
u0 = norm(vel_body);
w = u0/r_turn;

phi_c = deg2rad( 2.6 ); %Command roll angle degrees
phi_c = atan( (u0*w) / (aircraft_parameters.g*cos(euler_angles(2)))    ); %Corrd turn bank angle
```

```
ka = -1.9579; %Aileron Control gain
kp = 4.5640; %Roll rate control gain

delta_p = omega_body(1); %From trim condition, delta_p IS the roll rate (p), bc at trim p = 0;
phi = euler_angles(1);
aileron_perturb = RollAttitudeControl(ka, kp, phi_c, phi, delta_p); % put this into aircraft surfaces

% elev_perturb = PitchAttitudeControl(theta_c, aircraft_state(5), aircraft_state(11), kth, kq);
elev_perturb = 0;

aircraft_surfaces = aircraft_surfaces0 + [elev_perturb; aileron_perturb; dr; 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%
%%% Kinematics
vel_inertial = TransformFromBodyToInertial(vel_body, euler_angles);
euler_rates = EulerRatesFromOmegaBody(omega_body, euler_angles);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aerodynamic force and moment
density = stdatmo(-pos_inertial(3,1));

[fa_body, ma_body, wind_angles] = AeroForcesAndMoments_BodyState_WindCoeffs(aircraft_state, aircraft_surfaces, wi


%%% Gravity
fg_body = (aircraft_parameters.g*aircraft_parameters.m)*[-sin(euler_angles(2));sin(euler_angles(1))*cos(euler_ang


%%% Dynamics
vel_body_dot = -cross(omega_body, vel_body) + (fg_body + fa_body)/aircraft_parameters.m;


inertia_matrix = [aircraft_parameters.Ix 0 -aircraft_parameters.Ixz;...
                  0 aircraft_parameters.Iy 0;...
                  -aircraft_parameters.Ixz 0 aircraft_parameters.Iz];

omega_body_dot = inv(inertia_matrix)*(-cross(omega_body, inertia_matrix*omega_body) + ma_body);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% State derivative
xdot = [vel_inertial; euler_rates; vel_body_dot; omega_body_dot];

end
```