ASEN 3128 Homework 8

Jack Lambert*
University of Colorado Boulder, ASEN 3128-012, Group 16
Submitted 4/9/2018

I. Question 1 - Open Loop:

A. Part A :

Implementing the conditions of a Boeing 747 flying at an altitude of 40,000 feet, using the values provided from page 165 of Etkin, the full A matrix is provided below:

$$A = \begin{vmatrix} -0.0069 & 0.0139 & 0 & -9.81 \\ -0.0905 & -0.3149 & 235.8933 & 0 \\ 3.8918E-4 & -0.0034 & -0.4281 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

To calculate the natural frequency, dampening ratio, and time constant - the following relations were used:

$$\omega_n = \sqrt{\omega^2 + n^2}, \qquad \zeta = -\frac{n}{\omega_n}, \qquad \tau = -\frac{1}{n} \tag{1}$$

Where n and ω are the real and imaginary parts of the eigenvalues, respectively. The numerical values for the variables above for both the phugoid and short period mode are provided below:

Short Period Mode:
Eigenvalues ($\lambda_{1,2}$): -0.3717 ± 0.8869i
Natural Frequency (ω_n): 0.9616 [rad/s]
Dampening Coefficient (ζ): 0.3865
Time Constant (τ): 2.6906 [s]

Phugoid Mode:
Eigenvalues ($\lambda_{1,2}$): -0.0033 ± 0.0672i
Natural Frequency (ω_n): 0.0673 [rad/s]
Dampening Coefficient (ζ): 0.0489
Time Constant (τ): 304.025 [s]

As can be seen from the open loop A matrix and it's corresponding eigenvalues, the Boeing 747 is stable as the the eigenvalues for both the short period mode and the phugoid mode have negative real parts, meaning that the envelope driving the imaginary part's ocsillatory motion decays, driving the perturbation to nominal conditions. It can also be seen that the damping coefficient is greater than 0, which also shows stability.

B. Part B:

To compute the matrix A_{PWD} and the vector B_{PWD} in terms of the linear vector-matrix differential equation, the following equations from class were implemented:

$$\mathbf{\Delta \dot{y}} = \begin{vmatrix} -0.0069 & 9.81 \\ 3.89E-4 & 0 \end{vmatrix} \mathbf{\Delta y} + \begin{vmatrix} -4.643 \\ -0.418 \end{vmatrix} \mathbf{\Delta u} \tag{2}$$

*SID: 104414093

Where:

$$\mathbf{\Delta\dot{y}} = \begin{vmatrix} \Delta\dot{u} \\ \Delta\dot{\theta} \end{vmatrix}, \qquad \mathbf{\Delta y} = \begin{vmatrix} \Delta u \\ \Delta\theta \end{vmatrix}, \qquad \mathbf{\Delta u} = \begin{vmatrix} \Delta\delta_e \end{vmatrix} \tag{3}$$

C. Part C:

To compare the accuracy of the PWD approximation with respect to the full A matrix, a comparison of the eigenvalues is needed. The eigenvalues for both modes of the full A matrix are compared to the eigenvalues of the PWD approximation are given below:

Eigenvalues:
A_{PWD} = -0.0034 ± 0.0611i
A_{full} = -0.0033 ± 0.0672i (Phugoid Mode)
A_{full} = -0.3717 ± 0.8869i (Short Period Mode)

Since the eigenvalues tell how the system will respond, the natural frequency and damping ratio are purely indications of how well the approximation predicted the eigenvalues. Since the A_{PWD} matrix is only 2x2, there is only one conjugate pair corresponding to one mode. To make a comparison to the full A matrix, the values of the phugoid mode were compared as they were much closer. The values are provided below:

Table 1. Dimensional Derivatives

	$\omega_n[\frac{rad}{s}]$	ζ
A_{full}	0.0673	0.0489
A_{PWD}	0.0611	0.0561
% Error	9.13	14.86

After analyzing the eigenvalues and the characteristics of these eigenvalues, namely, the natural frequency and the dampening ratio, it was noticed that the PWD approximation is relatively accurate as the deviations were small. Considering the simplifications of decoupling the 4x4 matrix to that of a 2x2 matrix and only getting a maximum error corresponding to the dampening ratio, which was found to be 14.89%, makes the PWD approximation very useful.

II. Question 2 - Closed Loop:

A. Part A :

When implementing a closed loop speed-to-elevator feedback loop, we first had to set a target zone for where we wanted our phugoid eigenvalues to ideally be. This target zone is defined by the ranges of our time constant and dampening ratio. The time constant (τ) has to be in the range - ($0 \leq \tau \leq 20$) [seconds], and the dampening ratio (ζ) has to be in the range - ($0.9 \leq \zeta \leq 0.95$). The values of the real values of the eigenvalues only depend on the time constant, so these values were found over the corresponding range first by using the relation: $n = -\frac{1}{\tau}$, where n is the real part of the eigenvalues. Once a vector of the real parts of our target zone were found, the imaginary parts of the eigenvector could be found by the relation: $\omega = n\sqrt{(\frac{1}{\zeta})^2 - 1}$, where we implemented each individual real part, n, and calculated the imaginary part over the range of ζ. After this was done over all n, inside it's corresponding vector, the bounds were found and are plotted below:

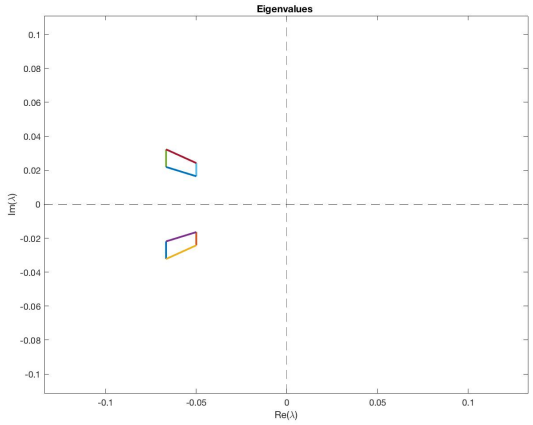

Figure 1. Target Zone Bounds

B. Part B:

To get a better understanding of what proportional gain we should use for a proportional only controller, a plot of the eigenvalues for the range of proportional gain values, $|k_1| = 0 : 0.00001 : 0.01$ was implemented. This was done by using the following relation and plugging into the linear vector-matrix differential equation from before:

$$\mathbf{\Delta}\delta_{\mathbf{e}} = \begin{vmatrix} -k_1 & 0 \end{vmatrix} \begin{vmatrix} \Delta u \\ \Delta\theta \end{vmatrix} \tag{4}$$

To find the ideal values for the proportional gain, k_1, we took the k_1 that corresponded to the eigenvalues with the closest distance to the target region. It should be noted that we took this distance for only the eigenvalues that had both imaginary and real parts since eigenvalues with only real or imaginary parts are unstable. Plots of the the eigenvalues corresponding to the shortest distance are plotted below. Since larger negative real parts have more stable behavior the largest negative real part that also has imaginary parts was also plotted in conjunction to the the closest eigenvalues.

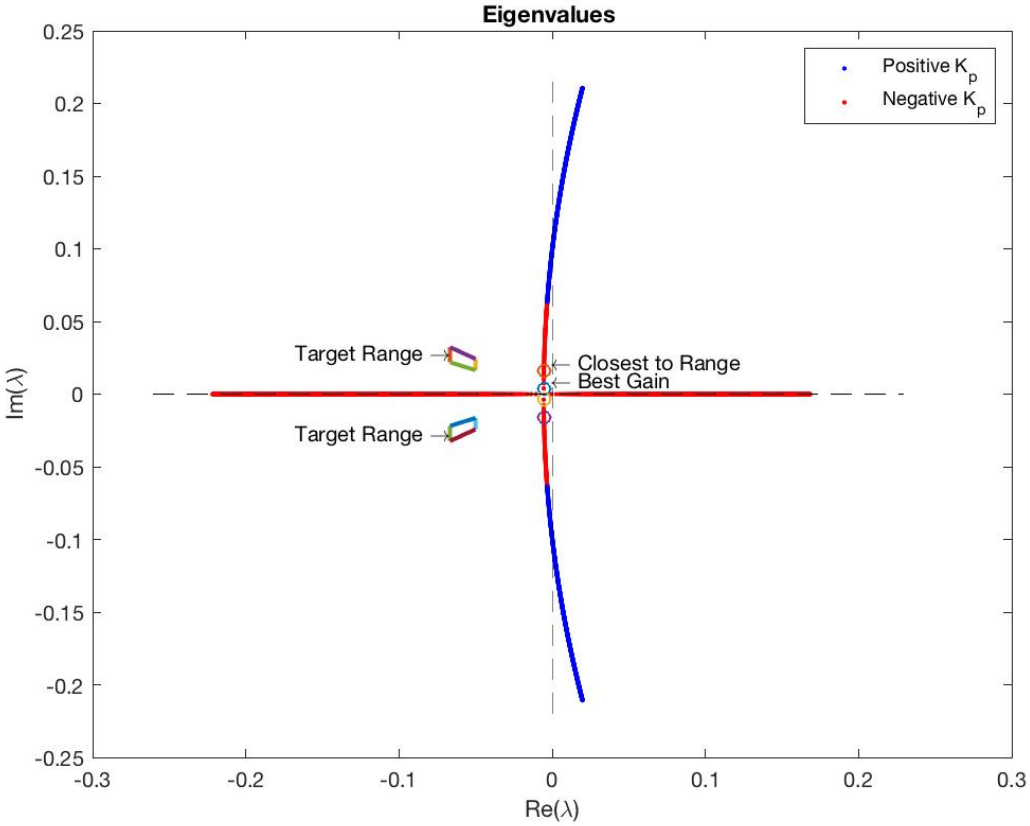

Figure 2. Finding Ideal Proportional Gain Values

As can be seen from the plots, negative k_1 values have values that represent a more stable system, where the positive k_1 values are less stable. Therefore, positive k_1 values are ideal since they have larger negative real parts of their eigenvectors, which drives the systems envelope to decay faster back to nominal conditions. The best k_1 values, found from the closest distance circled in the plots, and the k_1 values with the largest negative real parts, are given below:

k_1 Values:
k_1 = -0.0009 (Largest Negative Real Part)
k_1 = -0.00084 (Closest Distance)

C. Part C:

When implementing the closed-loop PWD model response for proportional-only control, the two different gain values of k_1 were used and plotted against each other for comparison. The plots for the variables of $\Delta u, \Delta\theta, and \Delta\delta_e$ are plotted versus time in the following plots for a perturbation of $\Delta u = 10\ \frac{m}{s}$:

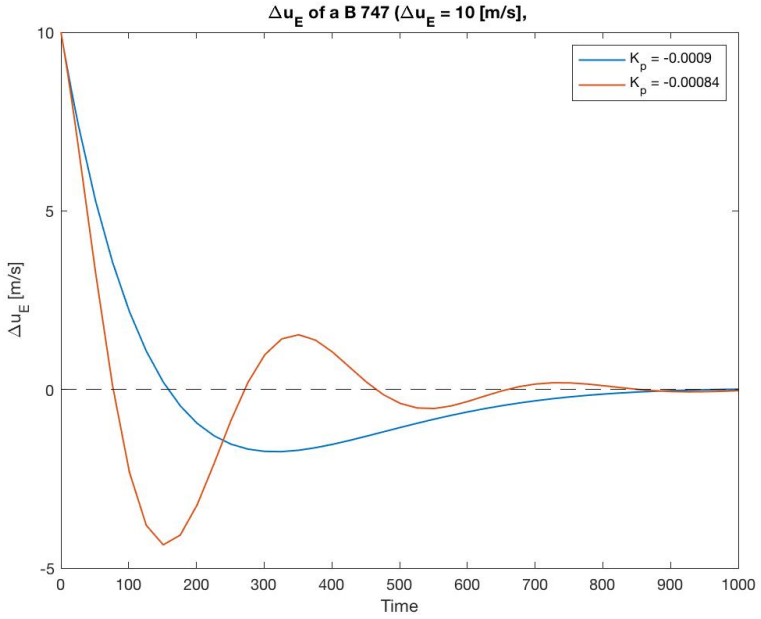

Figure 3. Close-Loop for P-Only Control

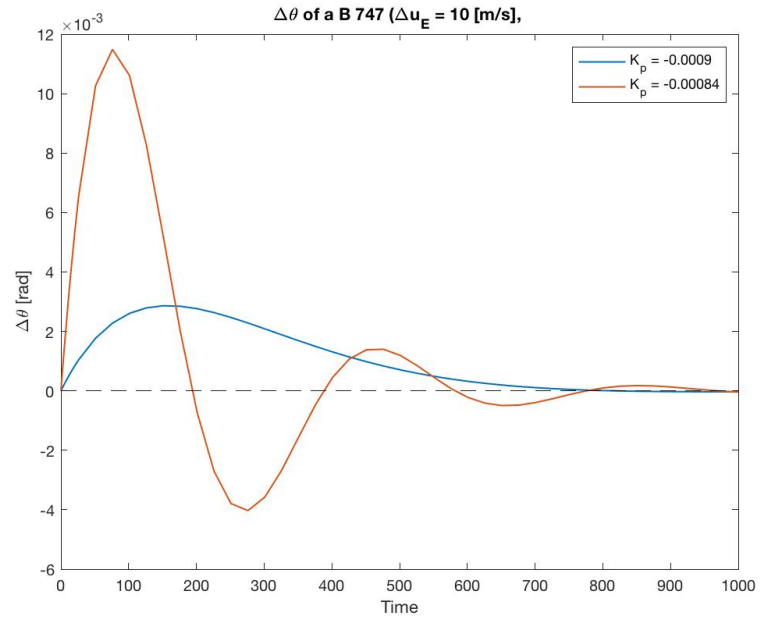

Figure 4. Close-Loop for P-Only Control

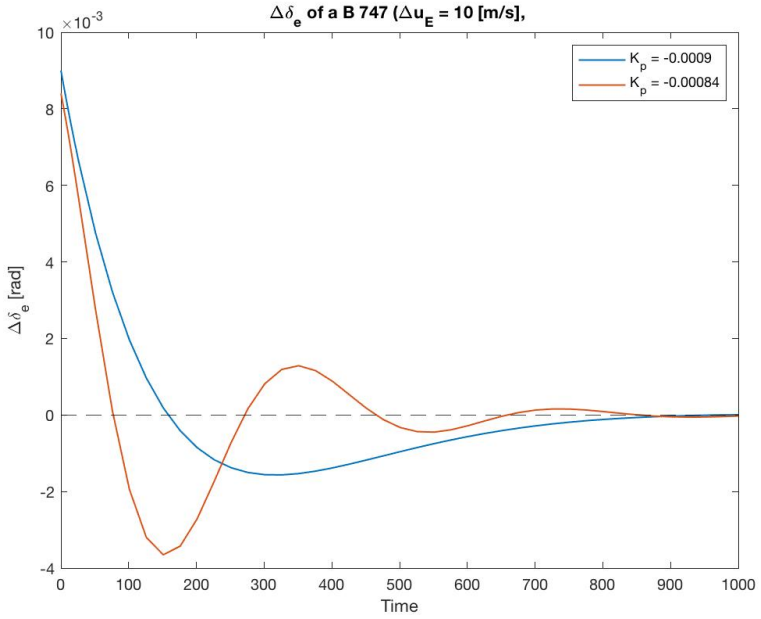

Figure 5. Close-Loop for P-Only Control

As can be seen from the plots, the realization about larger negative real parts ended up being valid as the k_1 value for this case provided greater dampening and stability as the state values returned to zero perturbation much quicker.

D. Part D:

Designing a full PD controller for the linearized approximation, to again attempt to get in the target range of eigenvalues, the control input vector for a speed-to-elevator feedback loop can be represented by:

$$\mathbf{\Delta}\delta_\mathbf{e} = -k_1\Delta u - k_2\Delta\dot{u} \tag{5}$$

A range of k_1 and k_2 values were then varied over reasonable ranges until the eigenvalues were with in the target range. A graphical representation of this is provided below:

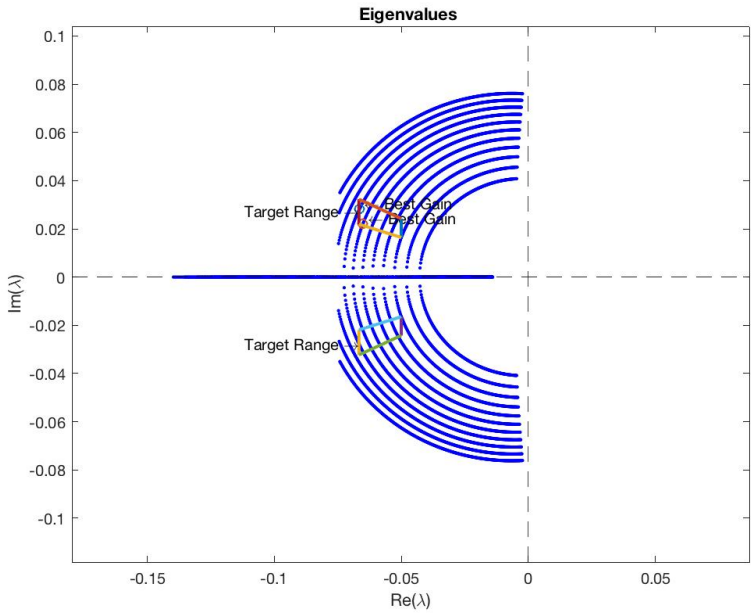

Figure 6. Eigenvalues Varying k_1 and k_2

To then find the ideal gains corresponding to the eigenvalues in this region, the values with the largest negative real parts were chosen as the larger the negative real part the more stable the system should be. A plot of the chosen eigenvalues and the corresponding gain values are provided below:

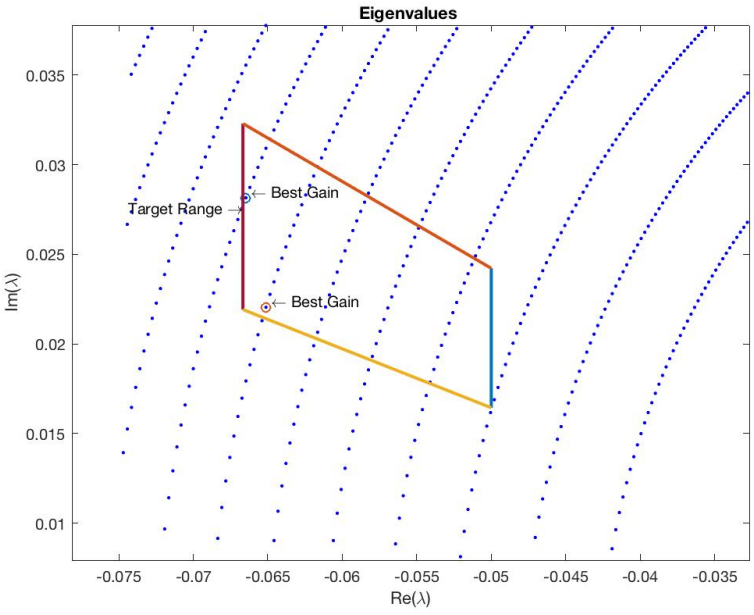

Figure 7. Largest Negative Real Part

Gain 1:
$k_1 = 0.0002$
$k_2 = 0.0269$

Gain 2:
$k_1 = 0.0001$
$k_2 = 0.0263$

E. Part E:

Implementing the closed-loop response for the PD controller and comparing the gains that we found for the two cases with the largest negative real parts for a perturbation of $\Delta u = 10$ [m/s], the following state variable response were found:

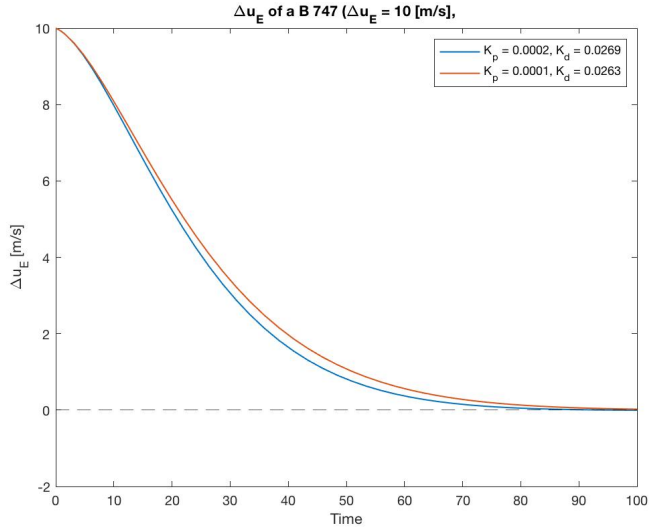

Figure 8. Close-Loop PWD Approximation for PD-Control

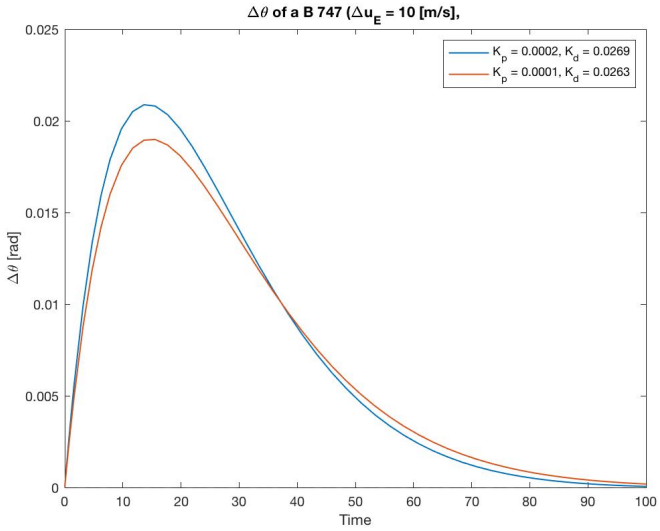

Figure 9. Close-Loop PWD Approximation for PD-Control

As illustrated in the plots above, the PD response to the perturbation about $\Delta u = 10$ [m/s], returned much faster to its nominal state in comparison to the P-only controller. This makes intuitive sense since implementing derivative control should increase the dampening of the system as we saw. After comparisons

of the state response, the two sets of gains gave very similar results. For the rest of this document we take the optimal gains to be $k_1 = 0.0001$ and $k_2 = 0.0263$, as the overshoot for $\Delta\theta$ was marginally better.

III. Question 3 - Full Linearized Longitudinal Dynamics:

A. Part A:

To calculate the eigenvalues of the closed loop PD-controlled phugoid eigenvalues for the full dynamics model, the following linear algebra was needed:

From Before we have the expression:

$$\boldsymbol{\Delta\dot{y}} = A\boldsymbol{\Delta y} - B\boldsymbol{\Delta u} \tag{6}$$

Where:

$$\boldsymbol{\Delta u} = \begin{vmatrix} -k_1\Delta u - k_2\Delta\dot{u} \\ 0 \end{vmatrix} = K_{mat1}\boldsymbol{\Delta y} + K_{mat2}\boldsymbol{\Delta\dot{y}} \tag{7}$$

$$K_{mat1} = \begin{vmatrix} -k_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}, \quad K_{mat2} = \begin{vmatrix} -k_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} \tag{8}$$

Further:

$$\boldsymbol{\Delta\dot{y}} = A\boldsymbol{\Delta y} + BK_{mat1}\boldsymbol{\Delta y} + BK_{mat2}\boldsymbol{\Delta\dot{y}} \tag{9}$$

Lastly, when simplified:

$$\boldsymbol{\Delta\dot{y}} = [(I_{4x4} - BK_{mat2})^{-}1(A + BK_{mat1})]\boldsymbol{\Delta y} \tag{10}$$

Now that we have the full linearized longitudinal dynamics model for PD-control, the eigenvalues for the phugoid mode can now be calculated by finding the eigenvalues with a smaller negative real part. We know this to be true because the decay of the envelope is smaller for a phugoind mode, as it lasts much longer. The phugoid eigenvalues are as follows:

Full Closed Loop PD-Controlled:
$\lambda_{1,2}$: -0.3078 $\pm$ 0.9174i (Phugoid Mode)

PWD Closed Loop PD-Controlled:
$\lambda_{1,2}$: -0.0651 $\pm$ 0.0220i

The eigenvalues of the PWD approximation are much different than the full linearized set. The coupled nature of the state variables inherently gives phugoid eigenvalues with much larger negative real parts than the approximation gives. This meaning the the full model is more damped than the PWD approximation, while also having different oscillatory motion due to the different imaginary eigenvalues. This is to be expected as decoupling variables that have derivative control will play a much larger role than just decoupling P-only control since the state variables are much more interrelated in the closed loop matrix.

B. Part B:

When simulating the the closed-loop response of the full dynamics model, for the initial conditions: $[\Delta u(0), \Delta w, \Delta q, \Delta\theta]^T = [10, 0, 0, 0]^T$, the following response for each of the state variables and the elevator response are as follows:

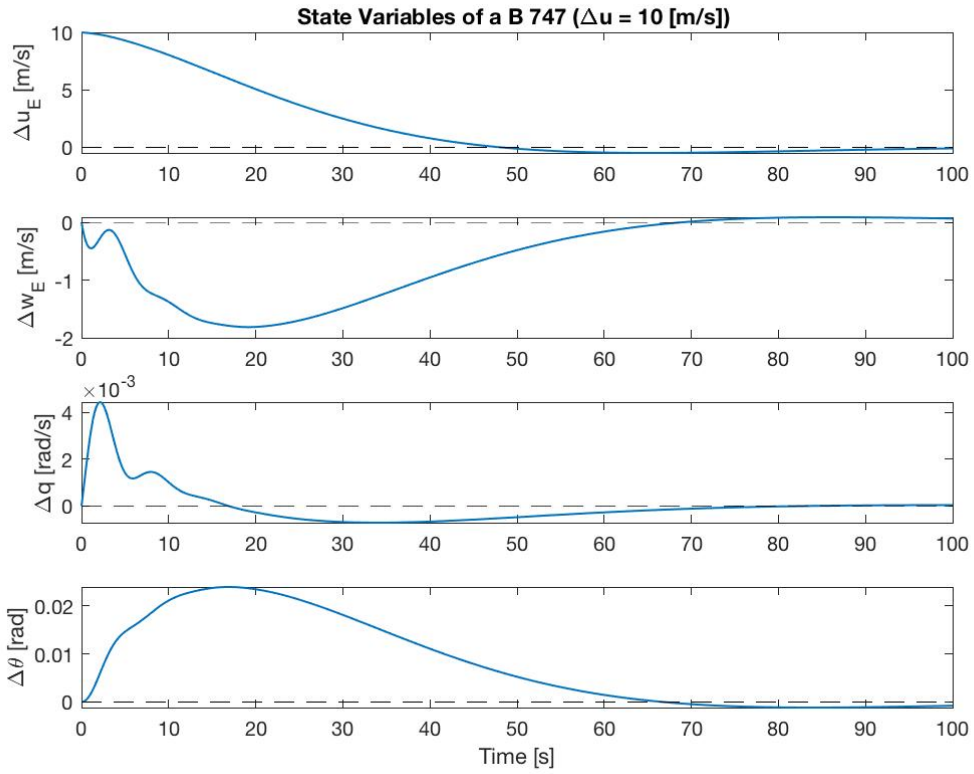

Figure 10. Full Linearized Longitudinal System

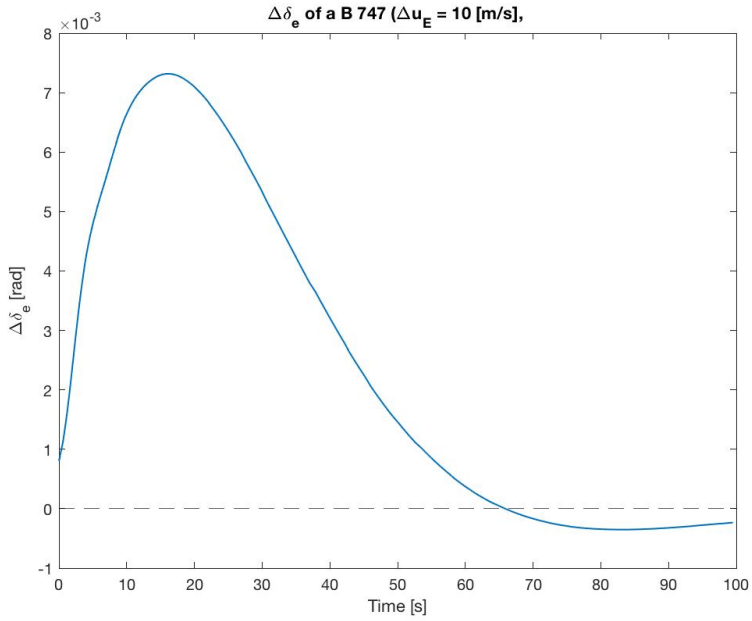

Figure 11. Full Linearized Longitudinal System

When comparing the plots of the state variable response to the perturbation about U_E, for the closed loop control for the PD-control of the full longitudinal set versus the P-only control of the PWD approximation

it was noticed that the time constants were much different. the dampening of the full set is much greater as the system returns much quicker to nominal conditions for the full set PD-control. This makes sense as the derivative gain should help the system's stability. The overshoot of the P-only system is also much larger due to this smaller dampening and stability, which also makes inherent sense.

C. Appendix A - MATLAB Code

1. *Calculates P-Control Eigenvector*

```
%% Author: Jack Lambert
% ASEN 3128
% Homework 8
% Purpose: This function find the target region for eigenvalues specified
% by a range of time constants and dampening coefficients. This code then
% finds the eigenvalues and mode and computes characterists of the the
% modes. This code then finds optimal proportional gain values that give
% eigenvalues close to the target range
close all;
clear all;
%% Airplane Parameters
% Nondimensional Derivatives
% Table 6.1 -
Cx = [-.108, .2193, 0, 0];
Cz = [-.106, -4.92, -5.921, 5.896];
Cm = [.1043, -1.023, -23.92, -6.314];

% Nondimensional Elevator Derivatives (Page 229 in Etkin)
C_x_de = -3.818*10^-6;
C_z_de = -0.3648;
C_m_de = -1.444;

% Table E.1 B747 Case 3
Alt = 40000*(0.3048); % Altitude [ft] -> [m]
rho = 0.3045; % Density [kg/m^3]
W = 2.83176*10^6; % Weight [N]
Ix = 0.247*10^8; % Moment of Interia x-SA [kg m^2]
Iy = 0.449*10^8; % Moment of Interia y-SA [kg m^2]
Iz = 0.673*10^8; % Moment of Interia z-SA [kg m^2]
Izx = -.212*10^7; % Moment of Interia zx-SA [kg m^2]
CD = .043; % Coefficient of Drag
cbar = 8.324; % Mean Chord Length [m]
S = 511; % Surface Area [m^2]
g = 9.81; % Gravity Constant [m/s^2]
m = W/g; % Mass of Plane [kg]


%% Trim States
Vel = 235.9;% Velocity [m/s]
u0 = Vel; % Initial Velocity in x-coord - Stability Axis Frame (Trim State)
theta0 = 0; % Initial Pitch Angle [deg]
Cw0 = W/(.5*rho*S*u0^2);
%% Function that Computes Dimensional Derivatives from Non-Dimenional derivatives
[X, Z, M, X_c, Z_c, M_c ] = NonDimLong(rho,u0,S,W,theta0,Cx,Cz,Cm,cbar,C_x_de,C_z_de,C_m_de);

%% State Variable Matrix A
row1 = [X(1)/m, X(2)/m, 0, -g*cosd(theta0)];
row2 = [Z(1)/(m-Z(4)), Z(2)/(m-Z(4)), (Z(3)+m*u0)/(m-Z(4)), (-W*sind(theta0))/(m-Z(4))];
row3 = [(1/Iy)*(M(1) + ((M(4)*Z(1))/(m-Z(4))) ),...
        (1/Iy)*(M(2) + ((M(4)*Z(2))/(m-Z(4))) ),...
        (1/Iy)*(M(3) + ((M(4)*(Z(3)+m*u0))/(m-Z(4))) ),...
        -((M(4)*W*sind(theta0))/(Iy*(m-Z(4))))];
row4 = [0, 0, 1, 0];

A = [row1;row2;row3;row4];

%% Input Matrix B
```

```
% Dimensionalizing Elevator Derivative

% Compenents of B Matrix
row1_C = [X_c(1)/m, X_c(2)/m];
row2_C = [Z_c(1)/(m-Z(4)), Z_c(2)/(m-Z(4))];
row3_C = [M_c(1)/Iy + (M(4)*Z_c(1))/(Iy*(m-Z(4))), M_c(2)/Iy + (M(4)*Z_c(2))...
    /(Iy*(m-Z(4)))];
row4_C = [0, 0];

B = [row1_C;row2_C;row3_C;row4_C];

%% Reduced PWD Model (Phugoid Mode)

row1_PWD = [X(1)/m, -g];
row2_PWD = [-Z(1)/(u0*m), 0];

A_PWD = [row1_PWD; row2_PWD];

B_PWD = [X_c(1)/m - (X(2)*M_c(1))/(m*M(2));
         (M_c(1)*Z(2)-M(2)*Z_c(1))/(m*u0*M(2))];

%% Copmaring the Eigen Values of the A matrix to the A_PWD
[eVA,eValA] = eig(A);
[eV_PWD,eVal_PWD] = eig(A_PWD);

modesA = diag(eValA);
modesA_PWD = diag(eVal_PWD);


max_real = max(abs(real(modesA)));
max_realz_PWD = max(abs(real(modesA_PWD)));

% Short Mode has Larger Real Part
 j = 1;
 k = 1;
for i = 1:length(modesA)
    if abs(real(modesA(i))) == max_real
        SP_Mode(j) = modesA(i); % Short Period Mode
        SP_vec(:,j) = eVA(:,i); % Short Period Eigen Vec
        j = j+1;
    else
        Phu_Mode(k) = modesA(i); % Phugoid Mode
        Phu_vec(k,:) = eVA(:,i); % Short Period Eigen Vec
        k = k+1;

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Natural Frequency and Dampeing Ratio

% Phugoid Mode

Wn_PM = ( real(Phu_Mode(1))^2+imag(Phu_Mode(1))^(2) )^(1/2); % Natural Frequency
zeta_PM = -real(Phu_Mode(1))/Wn_PM; % Dampening Coefficient
TimeConst_PM = -1/real(Phu_Mode(1)); % Time Constant (s)

% Short Period Mode

Wn_SP = ( real(SP_Mode(1))^2+imag(SP_Mode(1))^(2) )^(1/2); % Natural Frequency
zeta_SP = -real(SP_Mode(1))/Wn_SP; % Dampening Coefficient
TimeConst_SP = -1/real(SP_Mode(1)); % Time Constant (s)

% PWD
Wn_PWD = ( real(modesA_PWD(1))^2+imag(modesA_PWD(1))^(2) )^(1/2); % Natural Frequency
zeta_PWD = -real(modesA_PWD(1))/Wn_PWD; % Dampening Coefficient
TimeConst_PWD = -1/real(modesA_PWD(1)); % Time Constant (s)

% Percent Error ( PWD vs. Phugoid)
Wn_error = abs(Wn_PWD - Wn_PM) / (Wn_PM) * 100;
```

```
zeta_error = abs(zeta_PWD - zeta_PM) / (zeta_PM) * 100;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plotting the "Target Zone" of EigenValue
z = 0.9:.001:0.95;
tau = 15:.1:20;
n = (-1./tau);
for i = 1:length(tau)
    for j = 1:length(z)
        w(i,j) = n(i)*(((1/z(j))^2-1))^(1/2);
    end

end

%% Finding The Proportial Gain Needed to reach region
k1 =  0: 0.00001 : 0.01;
% Positive Proportional Gain Values
for i = 1:length(k1)
    K_mat = [k1(i), 0]; % K2 is zero for P control
    A_BK = A_PWD - B_PWD.*K_mat;
    eVal_A_BK = eig(A_BK);
    modesA_BK_pos(i,1) = eVal_A_BK(1);
    modesA_BK_pos(i,2) = eVal_A_BK(2);
end
% Negative Proportional Gain Values
for i = 1:length(k1)
    K_mat = [-k1(i), 0]; % K2 is zero for P control
    A_BK = A_PWD - B_PWD.*K_mat;
    eVal_A_BK = eig(A_BK);
    modesA_BK_neg(i,1) = eVal_A_BK(1);
    modesA_BK_neg(i,2) = eVal_A_BK(2);
        % Finding closest distance if there is Imaginary Values
    if imag(eVal_A_BK(1)) ~=0 || imag(eVal_A_BK(2)) ~=0
        dist1(i) = ( (n(end)-real(modesA_BK_neg(i,1)))^2 +...
            (w(end,1) - imag(modesA_BK_neg(i,1)))^2 )^(1/2);
        dist2(i) = ( (n(end)-real(modesA_BK_neg(i,2)))^2 +...
            (w(end,end) - imag(modesA_BK_neg(i,2)))^2 )^(1/2);
    end
end


% Gains Corresponding to these Points of Interest
ind1 = find(dist1 == min(dist1));
ind2 = find(dist2 == min(dist2));

k_1_1 = k1(ind1);
k_1_2 = k1(ind2);

% Plotting Eigenvalues
figure
plot(real(modesA_BK_pos(:,1)),imag(modesA_BK_pos(:,1)),'.B')
hold on
plot(real(modesA_BK_neg(:,1)),imag(modesA_BK_neg(:,1)),'.R')
plot(real(modesA_BK_pos(:,2)),imag(modesA_BK_pos(:,2)),'.B')
plot(real(modesA_BK_neg(:,2)),imag(modesA_BK_neg(:,2)),'.R')


% Closest Points to Target Region w/ Imag. Parts
plot(real(modesA_BK_neg(ind1,1)),imag(modesA_BK_neg(ind1,1)),'o')
text(real(modesA_BK_neg(ind1,1)),imag(modesA_BK_neg(ind1,1)),...
    ' \leftarrow Best Gain','VerticalAlignment','baseline')
plot(real(modesA_BK_neg(ind2,1)),imag(modesA_BK_neg(ind2,1)),'o')
text(real(modesA_BK_neg(ind2,1)),imag(modesA_BK_neg(ind2,1)),...
    ' \leftarrow Closest to Range','VerticalAlignment','baseline')
plot(real(modesA_BK_neg(ind1,2)),imag(modesA_BK_neg(ind1,2)),'o')
plot(real(modesA_BK_neg(ind2,2)),imag(modesA_BK_neg(ind2,2)),'o')

check = 1;
% Plotting Desired Region
% Plotting only the first and last values to show region
```

```
plot([n(1),n(1)],[w(1,1),w(1,end)],'Linewidth',2)
hold on
plot([n(end),n(end)],[w(end,1),w(end,end)],'Linewidth',2)
plot(n(:),w(:,1),'Linewidth',2)
plot(n(:),w(:,end),'Linewidth',2)
text(n(1),w(1,30),'  Target Range \rightarrow','HorizontalAlignment','right')

% For negative imaginary eigen values
plot([n(1),n(1)],-[w(1,1),w(1,end)],'Linewidth',2)
hold on
plot([n(end),n(end)],-[w(end,1),w(end,end)],'Linewidth',2)
plot(n(:),-w(:,1),'Linewidth',2)
plot(n(:),-w(:,end),'Linewidth',2)
text(n(1),-w(1,20),'  Target Range \rightarrow','HorizontalAlignment','right')

plot([0,0],[-.22,.22],'--k')
plot([-0.26,0.23],[0,0],'--k')
title('Eigenvalues ')
xlabel('Re(\lambda)')
ylabel('Im(\lambda)')
legend('Positive K_p','Negative K_p')

%% Finding k2 values for eigen values near range
```

2. *Calculates PD-Control Eigenvectors*

```
%% Author: Jack Lambert
% ASEN 3129
% Homework 8
% Purpose: This function find the target region for eigenvalues specified
% by a range of time constants and dampening coefficients. This code then
% finds the eigenvalues and mode and computes characterists of the the
% modes. This code then finds optimal proportional and derivative gain
% values that give eigenvalues with the largest negative real parts inside
% the specified target range
close all;
clear all;
%% Airplane Parameters
% Nondimensional Derivatives
% Table 6.1 -
Cx = [-.108, .2193, 0, 0];
Cz = [-.106, -4.92, -5.921, 5.896];
Cm = [.1043, -1.023, -23.92, -6.314];

% Nondimensional Elevator Derivatives (Page 229 in Etkin)
C_x_de = -3.818*10^-6;
C_z_de = -0.3648;
C_m_de = -1.444;

% Table E.1 B747 Case 3
Alt = 40000*(0.3048); % Altitude [ft] -> [m]
rho = 0.3045; % Density [kg/m^3]
W = 2.83176*10^6; % Weight [N]
Ix = 0.247*10^8; % Moment of Interia x-SA [kg m^2]
Iy = 0.449*10^8; % Moment of Interia y-SA [kg m^2]
Iz = 0.673*10^8; % Moment of Interia z-SA [kg m^2]
Izx = -.212*10^7; % Moment of Interia zx-SA [kg m^2]
CD = .043; % Coefficient of Drag
cbar = 8.324; % Mean Chord Length [m]
S = 511; % Surface Area [m^2]
g = 9.81; % Gravity Constant [m/s^2]
m = W/g; % Mass of Plane [kg]


%% Trim States
Vel = 235.9;% Velocity [m/s]
```

```
u0 = Vel; % Initial Velocity in x-coord - Stability Axis Frame (Trim State)
theta0 = 0; % Initial Pitch Angle [deg]
Cw0 = W/(.5*rho*S*u0^2);
%% Function that Computes Dimensional Derivatives from Non-Dimenional derivatives
[X, Z, M, X_c, Z_c, M_c ] = NonDimLong(rho,u0,S,W,theta0,Cx,Cz,Cm,cbar,C_x_de,C_z_de,C_m_de);
%% Defining the Target Region
z = 0.9:.001:0.95;
tau = 15:.1:20;
n = (-1./tau);
for i = 1:length(tau)
    for j = 1:length(z)
        w(i,j) = n(i)*(((1/z(j))^2-1))^(1/2);
    end

end
%% Finding The  Gain Combination Needed to reach region
k1 =  -0.0005 : 0.0001 : 0.0005;
k2 = 0 : 0.0001 : 0.03;
% Reduced PWD Model ( B Matrix )
B_PWD = [X_c(1)/m - (X(2)*M_c(1))/(m*M(2));
            (M_c(1)*Z(2)-M(2)*Z_c(1))/(m*u0*M(2))];
% Positive Proportional Gain Values
for i = 1:length(k1)
    for j = 1:length(k2)
        % Reduced PWD Model (A-B*K)
        r11 = (X(1)/m - B_PWD(1)*k1(i))/(1+B_PWD(1)*k2(j));
        r12 = -g/(1+B_PWD(1)*k2(j));
        r21 = (-Z(1) / (m*u0) - B_PWD(2)*k1(i)) - (B_PWD(2)*k2(j)/(1+B_PWD(1)*k2(j)))...
            *(X(1)/m - B_PWD(1)*k1(i));
        r22 = B_PWD(2)*k2(j)*g/(1+B_PWD(1)*k2(j));
        A_BK = [r11,r12;r21,r22];
        % Eigenvalues
        eVal_A_BK = eig(A_BK);
        modesA_BK_1(i,j) = eVal_A_BK(1);
        modesA_BK_2(i,j) = eVal_A_BK(2);
    end

end

% Graphically finding point w/ large negative real parts
dist1 = abs(real(modesA_BK_1(8,:))+0.06647);
ind1 =find(dist1 == min(dist1));

dist2 = abs(real(modesA_BK_1(7,:))+0.06512);
ind2 =find(dist2 == min(dist2));

% Gains Corresponding to these Values

k1_1 = k1(8);
k2_1 = k2(ind1);

k1_2 = k1(7);
k2_2 = k2(ind2);

% Plotting Eigenvalues
figure
plot(real(modesA_BK_1),imag(modesA_BK_1),'.B')
hold on
plot(real(modesA_BK_2),imag(modesA_BK_2),'.B')

% Closest Point to largest real (left corner of target region)
plot(real(modesA_BK_1(8,ind1)),imag(modesA_BK_1(8,ind1)),'o')
text(real(modesA_BK_1(8,ind1)),imag(modesA_BK_1(8,ind1)),...
    ' \leftarrow Best Gain','VerticalAlignment','baseline')
plot(real(modesA_BK_1(7,ind2)),imag(modesA_BK_1(7,ind2)),'o')
text(real(modesA_BK_1(7,ind2)),imag(modesA_BK_1(7,ind2)),...
    ' \leftarrow Best Gain','VerticalAlignment','baseline')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plotting the "Target Zone" of EigenValue
```

```
% Plotting only the first and last values to show region
plot([n(1),n(1)],[w(1,1),w(1,end)],'Linewidth',2)
plot([n(end),n(end)],[w(end,1),w(end,end)],'Linewidth',2)
plot(n(:),w(:,1),'Linewidth',2)
plot(n(:),w(:,end),'Linewidth',2)
text(n(1),w(1,25),'  Target Range \rightarrow','HorizontalAlignment','right')

% For negative imaginary eigen values
plot([n(1),n(1)],-[w(1,1),w(1,end)],'Linewidth',2)
plot([n(end),n(end)],-[w(end,1),w(end,end)],'Linewidth',2)
plot(n(:),-w(:,1),'Linewidth',2)
plot(n(:),-w(:,end),'Linewidth',2)
text(n(1),-w(1,25),'  Target Range \rightarrow','HorizontalAlignment','right')

plot([0,0],[-.22,.22],'--k')
plot([-0.26,0.23],[0,0],'--k')
title('Eigenvalues ')
xlabel('Re(\lambda)')
ylabel('Im(\lambda)')


%% Full Matrix Comparison
% Chosen Gains
k1 = 0.0001;
k2 = 0.0263;
% State Variable Matrix A
row1 = [X(1)/m, X(2)/m, 0, -g*cosd(theta0)];
row2 = [Z(1)/(m-Z(4)), Z(2)/(m-Z(4)), (Z(3)+m*u0)/(m-Z(4)), (-W*sind(theta0))/(m-Z(4))];
row3 = [(1/Iy)*(M(1) + ((M(4)*Z(1))/(m-Z(4))) ),...
        (1/Iy)*(M(2) + ((M(4)*Z(2))/(m-Z(4))) ),...
        (1/Iy)*(M(3) + ((M(4)*(Z(3)+m*u0))/(m-Z(4))) ),...
        -((M(4)*W*sind(theta0))/(Iy*(m-Z(4))))];
row4 = [0, 0, 1, 0];

A = [row1;row2;row3;row4];

% Input Matrix B
% Dimensionalizing Elevator Derivative

% Compenents of B Matrix
row1_C = [X_c(1)/m, X_c(2)/m];
row2_C = [Z_c(1)/(m-Z(4)), Z_c(2)/(m-Z(4))];
row3_C = [M_c(1)/Iy + (M(4)*Z_c(1))/(Iy*(m-Z(4))), M_c(2)/Iy + (M(4)*Z_c(2))...
    /(Iy*(m-Z(4)))];
row4_C = [0, 0];

B = [row1_C;row2_C;row3_C;row4_C];

I = eye(4); % 4x4 Identity Matrix
k1_mat = [-k1, 0, 0, 0; 0, 0, 0, 0];
k2_mat = [-k2, 0, 0, 0; 0, 0, 0, 0];
A_BK_full = inv(A+B*k1_mat)*(I-B*k2_mat);

%% PD Closed-Loop Matrix

% Reduced PWD Model (A-B*K)
r11 = (X(1)/m - B_PWD(1)*k1)/(1+B_PWD(1)*k2);
r12 = -g/(1+B_PWD(1)*k2);
r21 = (-Z(1) / (m*u0) - B_PWD(2)*k1) - (B_PWD(2)*k2/(1+B_PWD(1)*k2))...
    *(X(1)/m - B_PWD(1)*k1);
r22 = B_PWD(2)*k2*g/(1+B_PWD(1)*k2);
A_BK = [r11,r12;r21,r22];

% Copmaring the Eigen Values of the A matrix to the A_PWD
modesA_full = eig(A_BK_full);
modesA_PD = eig(A_BK);

max_real = max(abs(real(modesA_full)));
max_realz_PWD = max(abs(real(modesA_PD)));
```

```
% Short Mode has Larger Real Part
 j = 1;
 k = 1;
for i = 1:length(modesA_full)
    if abs(real(modesA_full(i))) == max_real
        SP_Mode_full(j) = modesA_full(i); % Short Period Mode

        j = j+1;
    else
        Phu_Mode_full(k) = modesA_full(i) % Phugoid Mode
        k = k+1;

    end
end
```

3. Calculates Dimensional Derivatives

```
%% Author: Jack Lambert
% ASEN 3128
% Purpose: This function calculates the dimensional derivatives for the
% state matrix and the control matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X, Z, M, X_c, Z_c, M_c ] = NonDimLong(rho,u0,S,W,theta0,Cx,Cz,...
    Cm,cbar,C_x_de,C_z_de,C_m_de)

% Computing the Nondimesnional Inital Weight Derivative
Cw0 = W/ ((1/2)*rho*S*u0^2);

%% State Variable Derivatives
% X
Xu = rho*u0*S*Cw0*sind(theta0) + .5*rho*u0*S*Cx(1);
Xw = .5*rho*u0*S*Cx(2);
Xq = .25*rho*u0*cbar*S*Cx(3);
Xwdot = .25*rho*cbar*S*Cx(4);

X = [Xu, Xw, Xq, Xwdot]';

% Z
Zu = -rho*u0*S*Cw0*cosd(theta0) + .5*rho*u0*S*Cz(1);
Zw = .5*rho*u0*S*Cz(2);
Zq = .25*rho*u0*cbar*S*Cz(3);
Zwdot = .25*rho*cbar*S*Cz(4);

Z = [Zu, Zw, Zq, Zwdot]';

% M
Mu = .5*rho*u0*cbar*S*Cm(1);
Mw = .5*rho*u0*cbar*S*Cm(2);
Mq = .25*rho*u0*(cbar^2)*S*Cm(3);
Mwdot = .25*rho*(cbar^2)*S*Cm(4);

M = [Mu, Mw, Mq, Mwdot]';

%% Control Derivatives

% Elevator controls
X_c(1) = 1/2*rho*u0^2*S*C_x_de;
Z_c(1) = 1/2*rho*u0^2*S*C_z_de;
M_c(1) = 1/2*rho*u0^2*S*cbar*C_m_de;

% Thrust Controls
X_c(2) = 0;
Z_c(2) = 0;
M_c(2) = 0;
```

4. Calculates A, B, and K matrices

```
%% Author: Jack Lambert
% ASEN 3128
% Homework 8
% Purpose: To keep all constants in one function so they are not defined
% more than once and then Compute the constants for the state variable
% matrix A and the input matrix B. This function outputs the closed loop
% matrix given the optimal gains
% Date Modified: 4/9/2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [A_BK, A_BK_full] = Linearized(K_mat)
%% Airplane Parameters
% Nondimensional Derivatives
% Table 6.1 -
Cx = [-.108, .2193, 0, 0];
Cz = [-.106, -4.92, -5.921, 5.896];
Cm = [.1043, -1.023, -23.92, -6.314];

% Nondimensional Elevator Derivatives (Page 229 in Etkin)
C_x_de = -3.818*10^-6;
C_z_de = -0.3648;
C_m_de = -1.444;

% Table E.1 B747 Case 3
Alt = 40000*(0.3048); % Altitude [ft] -> [m]
rho = 0.3045; % Density [kg/m^3]
W = 2.83176*10^6; % Weight [N]
Ix = 0.247*10^8; % Moment of Interia x-SA [kg m^2]
Iy = 0.449*10^8; % Moment of Interia y-SA [kg m^2]
Iz = 0.673*10^8; % Moment of Interia z-SA [kg m^2]
Izx = -.212*10^7; % Moment of Interia zx-SA [kg m^2]
CD = .043; % Coefficient of Drag
cbar = 8.324; % Mean Chord Length [m]
S = 511; % Surface Area [m^2]
g = 9.81; % Gravity Constant [m/s^2]
m = W/g; % Mass of Plane [kg]


%% Trim States
Vel = 235.9;% Velocity [m/s]
u0 = Vel; % Initial Velocity in x-coord - Stability Axis Frame (Trim State)
theta0 = 0; % Initial Pitch Angle [deg]
Cw0 = W/(.5*rho*S*u0^2);
%% Function that Computes Dimensional Derivatives from Non-Dimenional derivatives
[X, Z, M, X_c, Z_c, M_c ] = NonDimLong(rho,u0,S,W,theta0,Cx,Cz,Cm,cbar,C_x_de,C_z_de,C_m_de);

%% State Variable Matrix A
row1 = [X(1)/m, X(2)/m, 0, -g*cosd(theta0)];
row2 = [Z(1)/(m-Z(4)), Z(2)/(m-Z(4)), (Z(3)+m*u0)/(m-Z(4)), (-W*sind(theta0))/(m-Z(4))];
row3 = [(1/Iy)*(M(1) + ((M(4)*Z(1))/(m-Z(4))) ),...
        (1/Iy)*(M(2) + ((M(4)*Z(2))/(m-Z(4))) ),...
        (1/Iy)*(M(3) + ((M(4)*(Z(3)+m*u0))/(m-Z(4))) ),...
        -((M(4)*W*sind(theta0))/(Iy*(m-Z(4))))];
row4 = [0, 0, 1, 0];

A = [row1;row2;row3;row4];

%% Input Matrix B
% Dimensionalizing Elevator Derivative

% Compenents of B Matrix
row1_C = [X_c(1)/m, X_c(2)/m];
row2_C = [Z_c(1)/(m-Z(4)), Z_c(2)/(m-Z(4))];
row3_C = [M_c(1)/Iy + (M(4)*Z_c(1))/(Iy*(m-Z(4))), M_c(2)/Iy + (M(4)*Z_c(2))...
    /(Iy*(m-Z(4)))];
row4_C = [0, 0];
```

```
B = [row1_C;row2_C;row3_C;row4_C];

%% Reduced PWD Model (Phugoid Mode)
k1 = K_mat(1);
k2 = K_mat(2);

B_PWD = [X_c(1)/m - (X(2)*M_c(1))/(m*M(2));
         (M_c(1)*Z(2)-M(2)*Z_c(1))/(m*u0*M(2))];

r11 = (X(1)/m - B_PWD(1)*k1)/(1+B_PWD(1)*k2);
r12 = -g/(1+B_PWD(1)*k2);
r21 = (-Z(1) / (m*u0) - B_PWD(2)*k1) - (B_PWD(2)*k2/(1+B_PWD(1)*k2))...
    *(X(1)/m - B_PWD(1)*k1);
r22 = B_PWD(2)*k2*g/(1+B_PWD(1)*k2);
A_BK = [r11,r12;r21,r22];

I = eye(4); % 4x4 Identity Matrix
k1_mat = [-k1, 0, 0, 0; 0, 0, 0, 0];
k2_mat = [-k2, 0, 0, 0; 0, 0, 0, 0];
A_BK_full = inv(I-B*k2_mat)*(A+B*k1_mat);

end
```

5. *Main that calls both ODE45 functions*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Author: Jack Lambert
% Dale Lawrence
% Aircraft Dynmaics Homework 8
% Purpose: Sets Initial Conditions for each Pertubation Case and Calls ODE45
% to plot the State Variables vs time
% Date Modefied: 4/9/18
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ODE45 Variable Allocation
%                       u_dot = z(1); % x-component of Velocity, Body Frame
%                       z_dot = z(2); % x-component of Velocity, Body Frame
%                       q_dot = z(3); % Angular Velocity about the y-axis [rad/s]
%                       theta_dot = z(4); % Pitch Angle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions
c1 = 10; % Delta U: x-comp, BF Interial Velocity [m/s]
c2 = 0; % Delta Theta: Pitch Angle
condition = [c1 c2];
%% State Variables vs. Time

% P-control
t = [0 1000]; % Larger times to see phugoid mode, shorter for short period mode
k1 = [-9.0*10^(-4),-8.4*10^(-4)];
K_mat1 = [k1(1), 0]; % Gain Vector for K1_1
K_mat2 = [k1(2), 0]; % Gain Vector for K1_2

[t,z1] = ode45(@(t,y) ODEcall_PWD(t,y,K_mat1),t,condition);
[t,z2] = ode45(@(t,y) ODEcall_PWD(t,y,K_mat2),t,condition);

% Change in U_E vs time
figure
plot(t ,z1(:,1),'Linewidth',1)
hold on
plot(t ,z2(:,1),'Linewidth',1)
plot(t,0*t,'--k')
tit = sprintf('%s %s %s','\Deltau_E of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Deltau_E [m/s]')
xlabel('Time')
legend('K_p = -0.0009','K_p = -0.00084')
```

```
hold off

% Change in Theta vs time
figure
plot(t ,z1(:,2),'Linewidth',1)
hold on
plot(t ,z2(:,2),'Linewidth',1)
plot(t,0*t,'--k')
tit = sprintf('%s %s %s','\Delta\theta of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Delta\theta [rad]')
xlabel('Time')
legend('K_p = -0.0009','K_p = -0.00084')
hold off

% Change in Elevator Response

de(:,1) = -k1(1)*z1(:,1);
de(:,2) = -k1(2)*z2(:,1);

figure
plot(t, de(:,1),'Linewidth',1)
hold on
plot(t ,de(:,2),'Linewidth',1)
plot(t,0*t,'--k')
tit = sprintf('%s %s %s','\Delta\delta_e of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Delta\delta_e [rad]')
xlabel('Time')
legend('K_p = -0.0009','K_p = -0.00084')
hold off

% PD-control

t = [0 100]; % Larger times to see phugoid mode, shorter for short period mode
k1 = [2*10^(-4),1*10^(-4)];
k2 = [0.0269,0.0263];
K_mat1 = [k1(1), k2(1)]; % Gain Vector for K1_1
K_mat2 = [k1(2), k2(2)]; % Gain Vector for K1_2

[t,z1] = ode45(@(t,y) ODEcall_PWD(t,y,K_mat1),t,condition);
[t,z2] = ode45(@(t,y) ODEcall_PWD(t,y,K_mat2),t,condition);

% Change in U_E vs time
figure
plot(t ,z1(:,1),'Linewidth',1)
hold on
plot(t ,z2(:,1),'Linewidth',1)
plot(t,0*t,'--k')
tit = sprintf('%s %s %s','\Deltau_E of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Deltau_E [m/s]')
xlabel('Time')
legend('K_p = 0.0002, K_d = 0.0269 ','K_p = 0.0001, K_d = 0.0263')
hold off

% Change in Theta vs time
figure
plot(t ,z1(:,2),'Linewidth',1)
hold on
plot(t ,z2(:,2),'Linewidth',1)
plot(t,0*t,'--k')
tit = sprintf('%s %s %s','\Delta\theta of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Delta\theta [rad]')
xlabel('Time')
legend('K_p = 0.0002, K_d = 0.0269 ','K_p = 0.0001, K_d = 0.0263')
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Full Matrix Response
```

```
% Initial Conditions
c1 = 10; % Delta U: x-comp, BF Interial Velocity [m/s]
c2 = 0; % Delta W: z-comp, BF Interial Velocity [m/s]
c3 = 0; % Delta q: y-comp, BF Angular Velocity [rad/s]
c4 = 0; % Delta Theta: Pitch Angle

condition = [c1 c2 c3 c4];

% PD Control
t = [0 100]; % Larger times to see phugoid mode, shorter for short period mode
k1 = 1*10^(-4);
k2 = 0.0263;
K_mat = [k1(1), k2(1)]; % Gain Vector for K1_1

% Phugoid Response (Longer Time)

% Calling ODE45
[t,z] = ode45(@(t,y) ODEcall_Full(t,y,K_mat),t,condition);


% U_E vs time
figure
subplot(4,1,1)
plot(t ,z(:,1),'Linewidth',1)
hold on
plot(t,0*t,'--k')
hold off
tit = sprintf('%s %s','State Variables of a B 747 (\Deltau = 10 [m/s])');
title(tit)
ylabel('\Deltau_E [m/s]')


% W_E vs time
subplot(4,1,2)
plot(t ,z(:,2),'Linewidth',1)
hold on
plot(t,0*t,'--k')
hold off
ylabel('\Deltaw_E [m/s]')

% q vs time
subplot(4,1,3)
plot(t ,z(:,3),'Linewidth',1)
hold on
plot(t,0*t,'--k')
hold off
ylabel('\Deltaq [rad/s]')

% Theta vs time
subplot(4,1,4)
plot(t ,z(:,4),'Linewidth',1)
ylabel('\Delta\theta [rad]')
hold on
plot(t,0*t,'--k')
hold off
xlabel('Time [s]')


% Elevator Response vs time
delta_u_dot = diff(z(:,1))./diff(t); % Differentating delta u
de_full = -k1*z(1:end-1,1) - k2*delta_u_dot; % Accounting for shorter vector
figure
plot(t(1:end-1) ,de_full,'Linewidth',1)
hold on
plot(t,0*t,'--k')
hold off
tit = sprintf('%s %s %s','\Delta\delta_e of a B 747 (\Deltau_E = 10 [m/s],');
title(tit)
ylabel('\Delta\delta_e [rad]')
```

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

6. ODE call for PWD Approximation

```
%% Author: Jack Lambert
% ASEN 3128
% Purpose: Function for ODE45 to call to calculate the State variables
% u_dot, w_dot, q_dot, and theta_dot for the PWD Approximation. This function
% uses the simplified assumptions for the Linearized Longitudinal Dynamics Set
% Last Edited: 4/9/2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dydt] = ODEcall_PWD(t,y,K_mat)

u_dot = y(1); % x-component of Velocity, Body Frame
theta_dot = y(2); % Pitch Angle

%% State Variable Matrix for Linearized Longitudinal Set
[A_BK, ~ ] = Linearized(K_mat); % A matrix function based on plane and parameters
State = [u_dot, theta_dot]'; % Couple State Variables in Long. Set
var = A_BK*State; % Couple State Variables in Long. Set
%% Solving for State Variables in the Linearized Longitudinal Set
dydt(1) = var(1); % uE
dydt(2) = var(2); % theta

dydt = dydt'; % Inverts for ODE45
end
```

7. ODE call for Full Linearized Longitudinal Model

```
%% Author: Jack Lambert
% ASEN 3128
% Purpose: Function for ODE45 to call to calculate the State variables
% u_dot, w_dot, q_dot, and theta_dot for the full linearized longitudinal
% set
% Last Edited: 4/9/2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dydt] = ODEcall_Full(t,y,K_mat)

u_dot = y(1); % x-compoinent of Inertial velocity, Body Frame
w_dot = y(2); % z-compoinent of Inertial velocity, Body Frame
q_dot = y(3); % y-component of Angular Velocity, Body Frame
theta_dot = y(4); % Pitch Angle

%% State Variable Matrix for Linearized Longitudinal Set
[ ~, A_BK_full] = Linearized(K_mat); % A matrix function based on plane and parameters
State = [u_dot, w_dot, q_dot, theta_dot]'; % Couple State Variables in Long. Set
var = A_BK_full*State; % Couple State Variables in Long. Set
%% Solving for State Variables in the Linearized Longitudinal Set
dydt(1) = var(1); % uE
dydt(2) = var(2); % wE
dydt(3) = var(3); % q
dydt(4) = var(4); % theta

dydt = dydt'; % Inverts for ODE45
end
```