ESE 547 Homework 2 / Frequency Domain Analysis / Seif Elkhashab / 02-10-2024

SIMO Plant and Controller Dynamics

1. Consider the following SIMO plant and controller dynamics. Break the loop at the plant input.

$A_z\,(fps^2)$: **Acceleration along the z axis,** $q\,(rps)$ **: Pitch rate,**

$\delta_e\,(degrees)$ **: Defelction of the elevator control surface,** $\dot{\delta}_e\,(dps)$ **: Rate of elevartor deflection**

$$x = \begin{bmatrix} A_z \\ q \\ \delta_e \\ \dot{\delta}_e \end{bmatrix} \dot{x} = \begin{bmatrix} \dot{A}_z \\ \dot{q} \\ \dot{\delta}_e \\ \ddot{\delta}_e \end{bmatrix}$$

$$A_p = \begin{bmatrix} -0.576007 & -3255.07 & 4.88557 & 9.25796 \\ -0.0410072 & -0.488642 & -2.03681 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -8882.64 & -133.266 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8882.64 \end{bmatrix}$$

$$C_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, D_p = 0. \cdot C_p \cdot B_p$$

$$A_c = [0]\,, B_{c1} = [-1 \quad 0]\,, B_{c2} = [1]$$

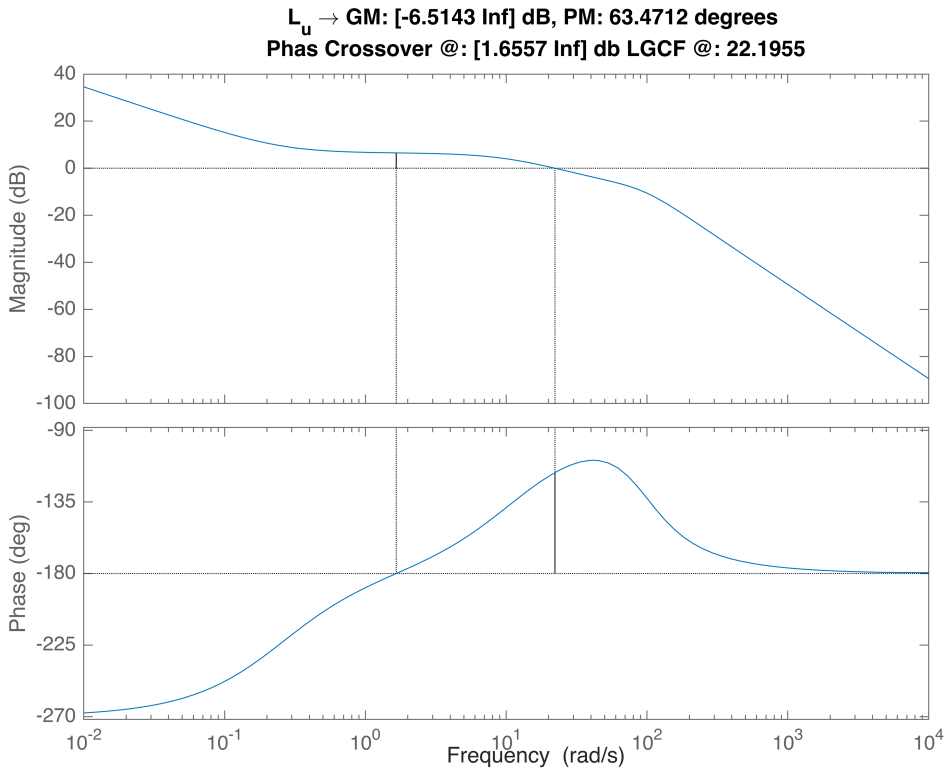
$$C_c = [0.0107349]\,, D_{c1} = [-0.0411729 \quad 11.4003]\,, D_{c2} = [0]$$

```
% for plotting
dd=0.:.001:2*pi;
xx1=cos(dd)-1;yy1=sin(dd);

% Plant Matricies
Ap = [ -0.576007  -3255.07   4.88557       9.25796;
       -0.0410072 -0.488642 -2.03681             0 ;
            0          0         0              1 ;
           0 0               -8882.64  -133.266 ];
 Bp = [ 0 ;
        0 ;
        0 ;
        8882.64];
 Cp = [ 1 0 0 0;
       0 1 0 0];
```

```
 Dp = 0.*Cp*Bp;

 % Controller Matricies
 Ac = 0;
 Bc1 = [ -1 0];
 Bc2 = 1;
 Cc = 0.0107349;
 Dc1 = [ -0.0411729 11.4003];
 Dc2 = 0;

% Closed Loop Dynamics
Zi = inv(eye(1) - Dc1*Dp);
Acl = [     Ap + Bp*Zi*Dc1*Cp                Bp*Zi*Cc;
         Bc1*(eye(2) + Dp*Zi*Dc1)*Cp    Ac + Bc1*Dp*Zi*Cc];
Bcl = [    Bp*Zi*Dc2;
       Bc2 + Bc1*Dp*Zi*Dc2];
Ccl = [(eye(2) + Dp*Zi*Dc1)*Cp Dp*Zi*Cc];
Dcl = Dp*Zi*Dc2;
sys_cl = ss(Acl,Bcl,Ccl,Dcl);

% Open Loop Dynamics at plant input
Alu = [Ap     zeros(4,1);
       Bc1*Cp     Ac];
Blu = [  Bp;
       Bc1*Dp];
Clu = -[Dc1*Cp Cc];
Dlu = -Dc1*Dp;
sys_u = ss(Alu,Blu,Clu,Dlu);

% Margins
lu_margins = allmargin(sys_u);
gm_lu = 20*log10(lu_margins.GainMargin);
pm_lu = lu_margins.PhaseMargin;
lgcf_lu = lu_margins.PMFrequency;
pc_lu = lu_margins.GMFrequency;
```

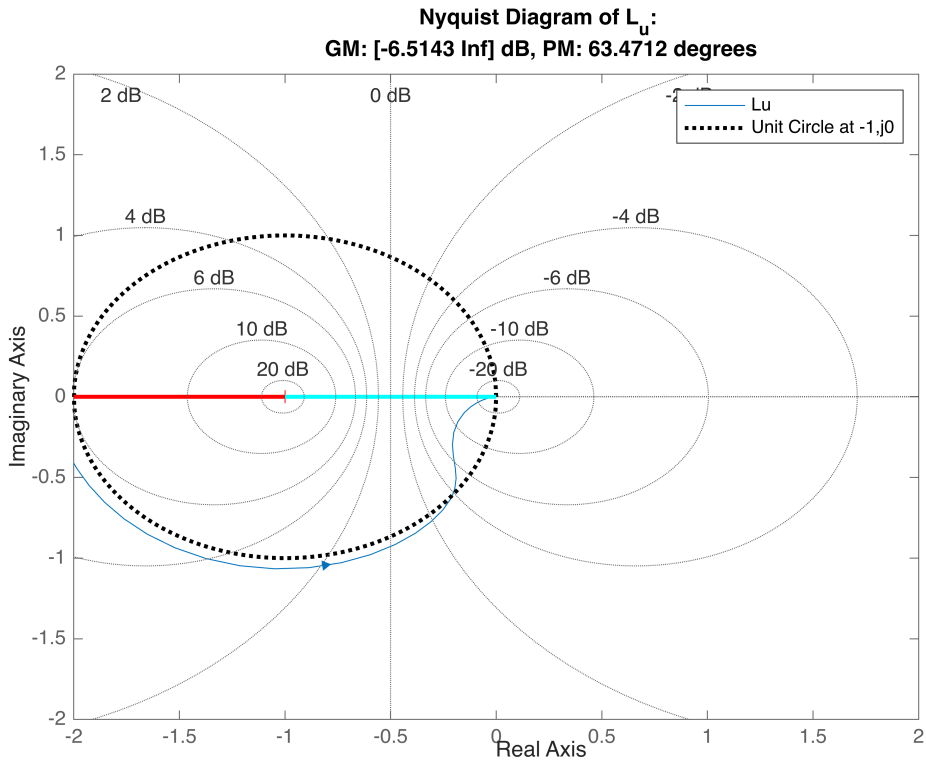
a) Plot a Bode plot and identify on the plot the gain and phase margins, loop gain crossover frequency and phase crossover frequency

```
margin(sys_u);
title(['L_u \rightarrow GM: [', num2str(gm_lu(1)), ' ',
num2str(gm_lu(2)),'] dB, ', 'PM: ',num2str(pm_lu(1)), ' degrees ',...
        newline 'Phas Crossover @: [', num2str(pc_lu(1)),'
',num2str(pc_lu(2)),'] db ','LGCF @: ',num2str(lgcf_lu(1))])
```



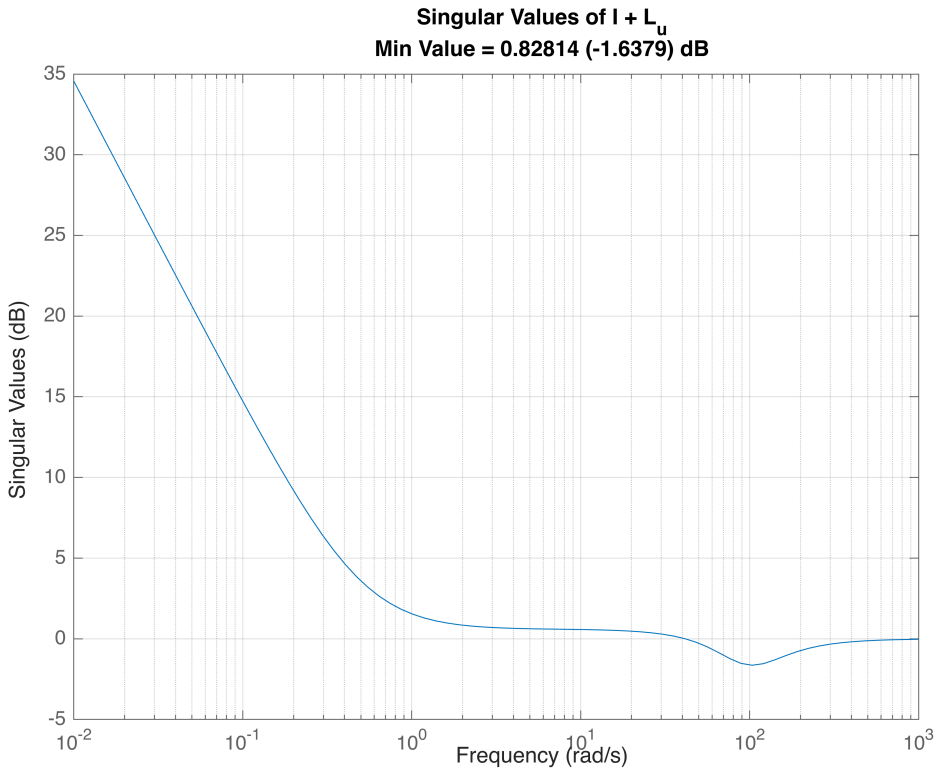
b) Plot a Nyquist plot and identify on the plot the gain and phase margins, loop gain and phase crossover frequencies.

```
n = nyquistplot(sys_u);
hold on;
plot(xx1,yy1,'k:',[-1/lu_margins.GainMargin(1) -1.],[0. 0.],'r',[-1 -1/
lu_margins.GainMargin(2)],[0. 0.],'c','LineWidth',2);grid
xlim([-2,2]), ylim([-2,2])
setoptions(n,'ShowfullContour','off'), grid on
title(['Nyquist Diagram of L_u: ',...
    newline 'GM: [', num2str(gm_lu(1)),' ', num2str(gm_lu(2)), '] dB,
' 'PM: ' num2str(pm_lu), ' degrees'])
legend('Lu', 'Unit Circle at -1,j0')
hold off;
```



c) Plot the minimum singular value of the return difference $I + L_u$ in dB vs frequency. Identify on the plot the minimum value (not in dB).

```
[a,b] = ss2tf(Alu,Blu,Clu,Dlu);
c = b + a;
sys_IL = tf(c,b);
alpha_0 = min(sigma(sys_IL));

sigma(sys_IL);
grid on, title(['Singular Values of I + L_u',...
                newline 'Min Value = ',num2str(alpha_0), ' (',
num2str(20*log10(alpha_0)),') dB'])
```




```
text = ['Minimum SV of (I +L_u): ',num2str(20*log10(alpha_0)), ' dB'];
disp(text)
```

```
Minimum SV of (I +L_u): -1.6379 dB
```

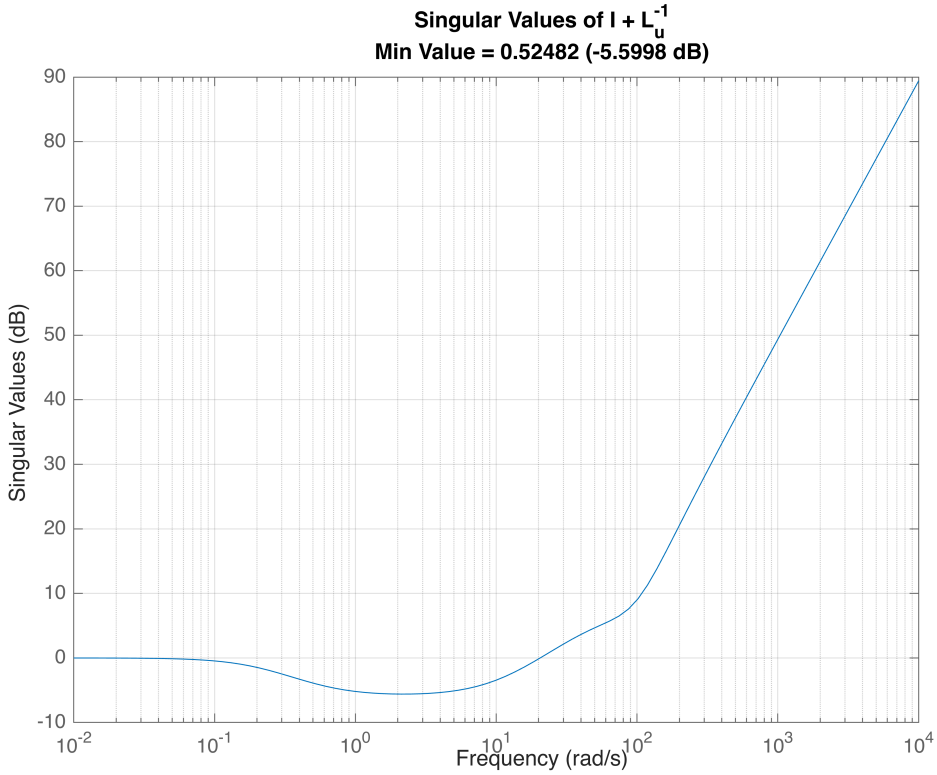
```
pm = 2*asin(alpha_0/2)*180/pi;
GM_l = 20*log10(1/(1 + alpha_0));
GM_u = 20*log10(1/(1 - alpha_0));
Gm_IL = [GM_l GM_u];
pm_IL = [-pm pm];
text = ['Gain Margin from (I+L_u):  [',num2str(Gm_IL(1)),',
',num2str(Gm_IL(2)),'] dB',...
         newline 'Phase Margin from (I+L_u): [',num2str(pm_IL(1)),',
',num2str(pm_IL(2)),'] degrees'];
disp(text);
```

```
Gain Margin from (I+L_u):  [-5.2402, 15.2967] dB
Phase Margin from (I+L_u): [-48.9218, 48.9218] degrees
```

d) Plot the minimum singular value of the $I + L_u^{-1}$ in dB vs frequency. Identify on the plot the minimum value (not in dB).

```
sys_ILi = tf(c,a);
beta_0 = (min(sigma(sys_ILi)));

sigma(sys_ILi)
```

```
grid on, title(['Singular Values of I + L_u^{-1}',...
                newline 'Min Value = ',num2str(beta_0), ' (',
num2str(20*log10(beta_0)),' dB)'])
```




```
text = ['Minimum SV of (I +inv(L_u)): ',num2str(20*log10(beta_0)), ' dB'];
disp(text)
```

```
Minimum SV of (I +inv(L_u)): -5.5998 dB
```

```
GM_u = 20*log10(1 + beta_0);
GM_l = 20*log10(1 - beta_0);
pm = 2*asin(beta_0/2)*180/pi;
Gm_ILi = [GM_l GM_u];
pm_ILi = [-pm pm];
text = ['Gain Margin from (I+inv(L_u)):  [',num2str(Gm_ILi(1)),',
',num2str(Gm_ILi(2)),'] dB',...
         newline 'Phase Margin from inv((I+L_u)): [',num2str(pm_ILi(1)),',
',num2str(pm_ILi(2)),'] degrees'];
disp(text);
```

```
Gain Margin from (I+inv(L_u)):  [-6.4628, 3.6644] dB
Phase Margin from inv((I+L_u)): [-30.4263, 30.4263] degrees
```

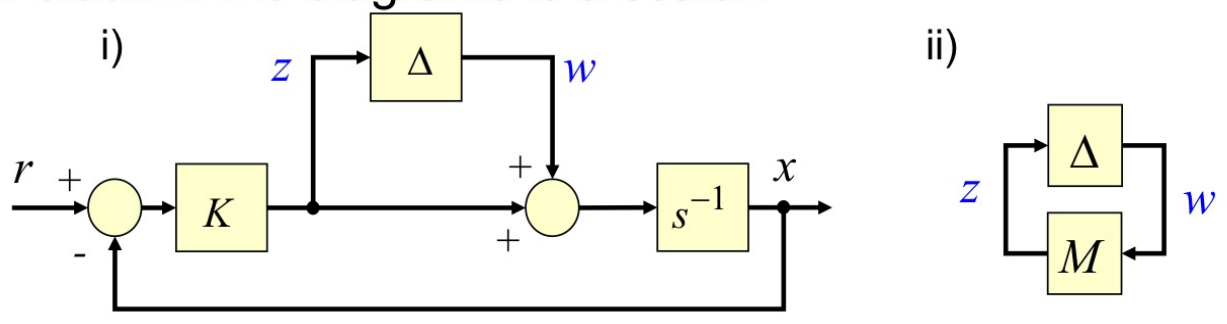
e) Compute the singular value gain and phase margins for the system (at the plant input) and compare with the classical values obtained in part a and b.

```
Gm_sv = [min(Gm_ILi(1),Gm_IL(1)),max(Gm_ILi(2),Gm_IL(2))];
Pm_sv = [min(pm_ILi(1),pm_IL(1)),max(pm_ILi(2),pm_IL(2))];

text = ['Classical Gain Margin:  [',num2str(gm_lu(1)),',
',num2str(gm_lu(2)),'] dB',...
        newline 'Classical Phase Margin: ',num2str(pm_lu),' degrees',...
        newline 'Singular Values Gain Margin:  [',num2str(Gm_sv(1)),',
',num2str(Gm_sv(2)),'] dB',...
        newline 'Singular Values Phase Margin: [',num2str(Pm_sv(1)),',
',num2str(Pm_sv(2)),'] degrees'];
disp(text);
```

```
Classical Gain Margin:  [-6.5143, Inf] dB
Classical Phase Margin: 63.4712 degrees
Singular Values Gain Margin:  [-6.4628, 15.2967] dB
Singular Values Phase Margin: [-48.9218, 48.9218] degrees
```

This follows the expected result, as the margins are similar, with the singular value margins being more conservative.

2. Exercise 5.2 This is an analytical problem (no Matlab). Consider the block diagrams shown below. Each block in the diagrams is a scalar.



a) Derive a state space model (A_M, B_M, C_M) for $M(s) = C_M (sI - A_M)^{-1} B_M$ (shown in ii)), modeling the system shown in i)

b) For $\Delta = 0$, what is the range of gain K that provides closed loop stability?

c) For $K = 1$, sketch the small gain theorem applied to the system in ii) using your model derived in part a). This should be a magnitude versus frequency plot.

d) What does the sketch in c) indicate for the system's robustness to uncertainties that are constant, i.e. $\Delta = \text{constant}$?

a.) M-Model: $\frac{z}{w} = M$, neglect refrence input for uncertainty analysis.

$z = -Kx$, $X = s^{-1}(-Kx + w) \longrightarrow X = s^{-1}(z + w) \longrightarrow z = -Ks^{-1}(z + w)$

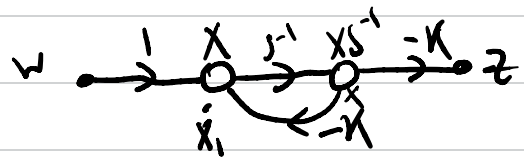
Solve for z:

$z = -Ks^{-1}z - Ks^{-1}w \rightarrow z + Ks^{-1}z = -Ks^{-1}w$, $z(1 + Ks^{-1}) = -Ks^{-1}w$

$\frac{z}{w} = \frac{-Ks^{-1}}{1 + Ks^{-1}}$, Simplify: $\underline{M = \frac{-K}{s + K}}$

- Deriving State space:

$\frac{z}{w} = \frac{-K}{s+K} \cdot \frac{Xs^{-1}}{Xs^{-1}} = \frac{-KXs^{-1}}{X + XKs^{-1}}$, $X = w - XKs^{-1}$, $z = -KXs^{-1}$



$\dot{X}_1 = w - Kx$

$y = -Kx$

$\dot{X} = [-K]X + [1]w$

$y = [-K]X + [0]w$

$\underline{A_M = [-K],\ B_M = [1],\ C_M = [-K],\ D_M = [0]}$,

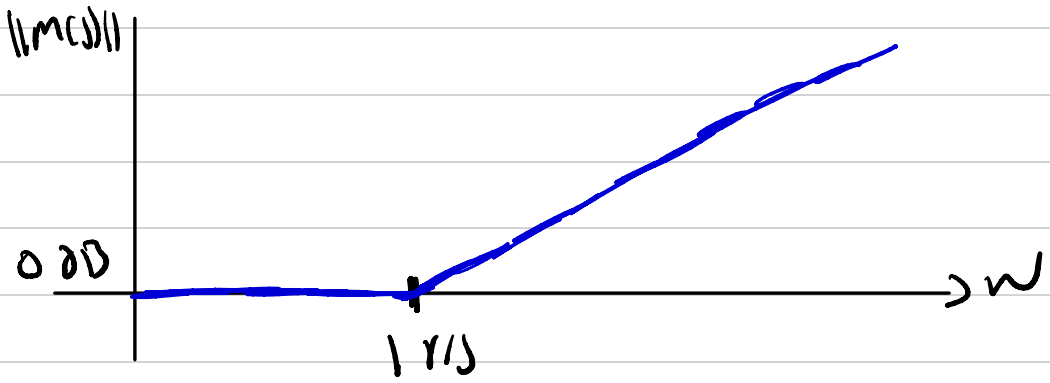
$M(s) = C_M(sI - A_M)D_M = -K(s + K)^{-1}1 = \frac{-K}{s+K}$ ✓

b.) For $\Delta = 0$, $M(s)$ is closed loop system and poles must be LHP,

$M(s) = \frac{-K}{s+K}$, $\Rightarrow$ $\underline{K > 0}$ is necessary for the system to be stable.

c.) Let $K=1$, $M(s) = \frac{-1}{s+1}$

Plotting $1/\bar{\sigma}(M)$



d.) Looking at the small gain theorem,

$\bar{\sigma}[\Delta] < 1/\bar{\sigma}[M]$, which is a sufficient test for stability

For any constant Δ, $\|\Delta\|$ has to be <1, $\underline{\|\Delta\| < 1}$ ensures Stability.

Adding uncertainty to transfer function $\rightarrow \frac{K(1+\Delta)}{s+K(1+\Delta)} \rightarrow \|\Delta\| < 1$ ensures Poles are LHP.

- Technically a positive Δ will not destabilize the system, but to ensure robustness $\|\Delta\|$ is used for the margin.

ESE 547 Homework 2 / Frequency Domain Analysis / Seif Elkhashab / 02-10-2024

SIMO Longitudinal Airframe Dynamics

Exercise 5.3 Consider the SIMO longitudinal airframe dynamics and classical control system described in example 5.1. The model data is for (5.14) and (5.8)

$K_a = -0.0015$	V = 886.78 fps
$K_q = -0.32$	Z_α / V = -1.3046
$a_z = 2.0$	Z_δ / V = -0.2142
$a_q = 6.0$	M_α = 47.7109
	M_δ = -104.8346

Setting up the model descdribed in the book symbolicaly

```
% Initializing symbolic Variables
syms alpha_dot q_dot Z_alpha M_alpha V alpha q Z_delta M_delta delta_e ...
a_z a_q K_q K_a alpha_dot_c q_dot_c alpha_c q_c Delta tau
```

Building the Plant State Space Model

```
A_p = [Z_alpha/V 1;M_alpha 0];
B_p = [Z_delta/V;M_delta];
C_p = [Z_alpha 0;0 1];
D_p = [Z_delta;0];
X_dot = [alpha_dot;q_dot];
X = [alpha;q];
y = [a_z;q];
u = delta_e;
% Displaying the Plant State Space Model
s = ["X_dot = A_p*X + B_p*u";"y = C_p*X + D_p*u"];
displayFormula(["'Plant State Space Model:'";s]);
```

Plant State Space Model:

$$\begin{pmatrix}\dot{\alpha}\\ \dot{q}\end{pmatrix} = \begin{pmatrix}\frac{Z_\alpha}{V} & 1\\ M_\alpha & 0\end{pmatrix}\begin{pmatrix}\alpha\\ q\end{pmatrix} + \begin{pmatrix}\frac{Z_\delta}{V}\\ M_\delta\end{pmatrix}\delta_e$$

$$\begin{pmatrix} a_z \\ q \end{pmatrix} = \begin{pmatrix} Z_\alpha & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ q \end{pmatrix} + \begin{pmatrix} Z_\delta \\ 0 \end{pmatrix} \delta_e$$

Building the Controller State Space Model

```
A_c = [0 0;K_a*a_z 0];
B_c1 = [-1 0;-K_a -1];
B_c2 = [1;K_a];
C_c = [K_q*K_a*a_z K_q*a_q];
D_c1 = [-K_q*K_a -K_q];
D_c2 = K_q*K_a;
X_dot_c = [alpha_dot_c;q_dot_c];
X_c = [alpha_c;q_c];
s = ["X_dot_c = A_c*X_c + B_c1*y + B_c2*r";"delta_e = C_c*X_c + D_c1*y +
D_c2*r"];
displayFormula(["'Controller State Space Model:'";s]);
```

Controller State Space Model:

$$\begin{pmatrix} \dot{\alpha}_c \\ \dot{q}_c \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ K_a\, a_z & 0 \end{pmatrix} \begin{pmatrix} \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} -1 & 0 \\ -K_a & -1 \end{pmatrix} \begin{pmatrix} a_z \\ q \end{pmatrix} + \begin{pmatrix} 1 \\ K_a \end{pmatrix} r$$

$$\delta_e = \begin{pmatrix} K_a\, K_q\, a_z & K_q\, a_q \end{pmatrix} \begin{pmatrix} \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} -K_a\, K_q & -K_q \end{pmatrix} \begin{pmatrix} a_z \\ q \end{pmatrix} + K_a\, K_q\, r$$

a) Form a closed loop state space model and simulate an acceleration step response to show the system is stable and correct.

Building the Closed Loop State Space Model

```
I = eye(size(D_c1*D_p));
z = (I-D_c1*D_p);
A_cl = [A_p+(B_p*(inv(z))*D_c1*C_p) B_p*inv(z)*C_c;...
B_c1*(eye(size(D_p*inv(z)*D_c1))+D_p*inv(z)*D_c1)*C_p
A_c+B_c1*D_p*inv(z)*C_c];
B_cl = [B_p*inv(z)*D_c2;B_c2+B_c1*D_p*inv(z)*D_c2];
C_cl = [(eye(size(D_p*inv(z)*D_c1))+D_p*inv(z)*D_c1)*C_p D_p*inv(z)*C_c];
D_cl = [D_p*inv(z)*D_c2];
X_dot_cl = [X_dot;X_dot_c];
X_cl = [X;X_c];
s = ["X_dot_cl = A_cl*X_cl + B_cl*r";"y= C_cl*X_cl + D_cl*r"];
displayFormula(["'Complete Closed Loop State Space Model:'";s]);
```

Complete Closed Loop State Space Model:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\alpha}_c \\ \dot{q}_c \end{pmatrix} = \begin{pmatrix} \frac{Z_\alpha}{V} - \frac{K_a K_q Z_\alpha Z_\delta}{V \sigma_2} & 1 - \frac{K_q Z_\delta}{V \sigma_2} & \frac{K_a K_q Z_\delta a_z}{V \sigma_2} & \frac{K_q Z_\delta a_q}{V \sigma_2} \\ M_\alpha - \frac{K_a K_q M_\delta Z_\alpha}{\sigma_2} & -\frac{K_q M_\delta}{\sigma_2} & \frac{K_a K_q M_\delta a_z}{\sigma_2} & \frac{K_q M_\delta a_q}{\sigma_2} \\ Z_\alpha \ (\sigma_1 - 1) & \frac{K_q Z_\delta}{\sigma_2} & -\frac{K_a K_q Z_\delta a_z}{\sigma_2} & -\frac{K_q Z_\delta a_q}{\sigma_2} \\ K_a Z_\alpha \ (\sigma_1 - 1) & \sigma_1 - 1 & K_a a_z - \frac{{K_a}^2 K_q Z_\delta a_z}{\sigma_2} & -\frac{K_a K_q Z_\delta a_q}{\sigma_2} \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} \frac{K_a K_q Z_\delta}{V \sigma_2} \\ \frac{K_a K_q M_\delta}{\sigma_2} \\ 1 - \sigma_1 \\ K_a - \frac{{K_a}^2 K_q Z_\delta}{\sigma_2} \end{pmatrix} r$$

where

$$\sigma_1 = \frac{K_a K_q Z_\delta}{\sigma_2}$$

$$\sigma_2 = K_a K_q Z_\delta + 1$$

$$\begin{pmatrix} a_z \\ q \end{pmatrix} = \begin{pmatrix} -Z_\alpha \left(\frac{K_a K_q Z_\delta}{\sigma_1} - 1 \right) & -\frac{K_q Z_\delta}{\sigma_1} & \frac{K_a K_q Z_\delta a_z}{\sigma_1} & \frac{K_q Z_\delta a_q}{\sigma_1} \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} \frac{K_a K_q Z_\delta}{\sigma_1} \\ 0 \end{pmatrix} r$$

where

$$\sigma_1 = K_a K_q Z_\delta + 1$$

Subbing in the Numerical Values to the Closed Loop Model

```
% V = 886.78; % Feet per second
% Z_alpha = -1.3046*V;
% Z_delta = -0.2142*V;
% M_alpha = 47.7109;
% M_delta = -104.8346;
% K_a = -0.0015;
% K_q = -0.32;
% a_z = 2.0;
% a_q = 6.0;
variables = [V;Z_alpha;Z_delta;M_alpha;M_delta;K_a;K_q;a_z;a_q];
variable_values =
[886.78;-1.3046*886.78;-0.2142*886.78;47.7109;-104.8346;-0.0015;-0.32;2.0;6.
0];
A_cl = double(subs(A_cl,variables,variable_values));
B_cl = double(subs(B_cl,variables,variable_values));
C_cl = double(subs(C_cl,variables,variable_values));
D_cl = double(subs(D_cl,variables,variable_values));
A_cl0 = round(vpa(A_cl),2);
B_cl0 = round(vpa(B_cl),2);
```

```
C_cl0 = round(vpa(C_cl),2);
D_cl0 = round(vpa(D_cl),2);

s = ["X_dot_cl = A_cl0*X_cl + B_cl0*r";"y= C_cl0*X_cl + D_cl0*r"];
displayFormula(["'Complete Closed Loop State Space Model:'";s]);
```

Complete Closed Loop State Space Model:

$$\begin{pmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\alpha}_c\\ \dot{q}_c\end{pmatrix}=\begin{pmatrix}-1.44 & 0.92 & 0 & 0.45\\ -16.34 & -36.91 & -0.11 & 221.48\\ 1272.96 & 66.88 & 0.2 & -401.29\\ -1.91 & -1.1 & 0 & 0.6\end{pmatrix}\begin{pmatrix}\alpha\\ q\\ \alpha_c\\ q_c\end{pmatrix}+\begin{pmatrix}0\\ -0.06\\ 1.1\\ 0\end{pmatrix}r$$

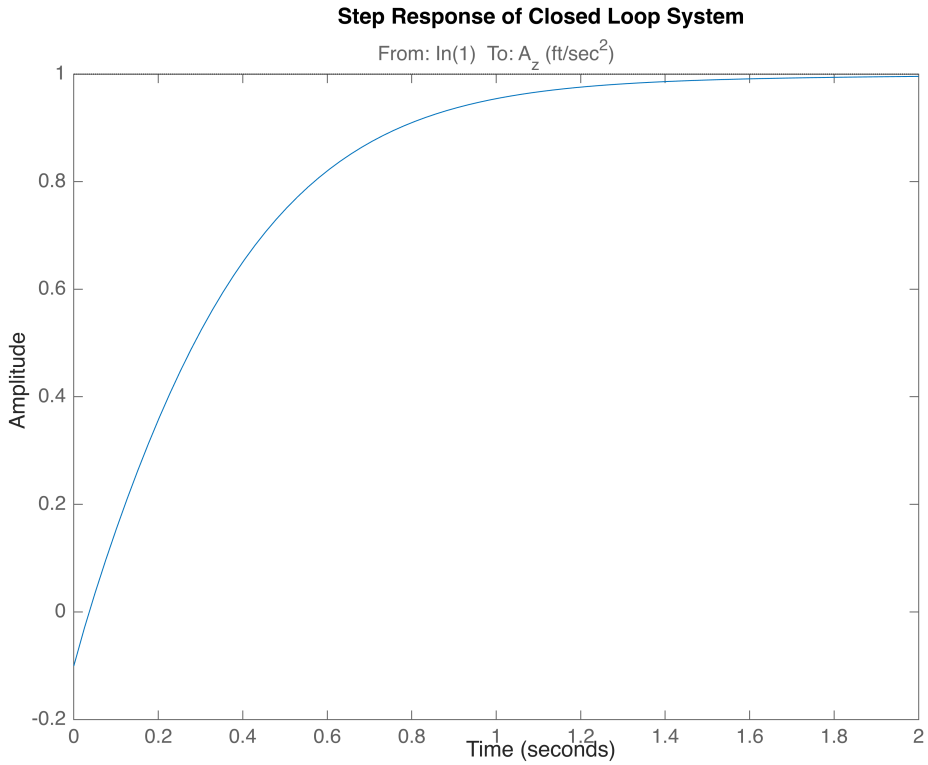
$$\begin{pmatrix}a_z\\ q\end{pmatrix}=\begin{pmatrix}-1272.96 & -66.88 & -0.2 & 401.29\\ 0 & 1.0 & 0 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ q\\ \alpha_c\\ q_c\end{pmatrix}+\begin{pmatrix}-0.1\\ 0\end{pmatrix}r$$

```
A_cl_eig = double(eig(A_cl));
disp(["Eigenvalues of Closed Loop System: "; num2str(A_cl_eig)]);
```

```
"Eigenvalues of Closed Loop System: "
"-1.81029+0i      "
"-3.65126+1.64868i"
"-3.65126-1.64868i"
"-28.4327+0i      "
```

Computing the Closed Loop System Step Response, Rise Time & Settling Time

```
sys_cl = ss(A_cl,B_cl,C_cl,D_cl);
sys_cl(1,1).OutputName = 'A_z (ft/sec^2)';
sys_cl(2,1).OutputName = 'Pitch Rate (dps)';
step(sys_cl(1))
xlim([0,2])
title("Step Response of Closed Loop System")
```




```
info = stepinfo(sys_cl,'RiseTimeThreshold',[0
0.63],'SettlingTimeThreshold',0.05);text = ['Rise Time of Az (63%): ',
num2str(info(1).RiseTime), ' sec,',...
newline 'Settling Time of Az (95%): ', num2str(info(1).SettlingTime),
'sec',...
newline 'Rise Time of q (63%): ', num2str(info(2).RiseTime), 'sec,',...
newline 'Settling Time of q (95%): ', num2str(info(2).SettlingTime), 'sec'];
disp(text)
```

```
Rise Time of Az (63%): 0.34573 sec,
Settling Time of Az (95%): 0.97305sec
Rise Time of q (63%): 0.015932sec,
Settling Time of q (95%): 1.061sec
```

b) Form the loop transfer function matrix L_u at the plant input. Compute stability margins.

Loop Transfer Function at the Plant Input L_u

```
A_Lu = [A_p 0*A_p;B_c1*C_p A_c];
B_Lu = [B_p;B_c1*D_p];
C_Lu = -[D_c1*C_p C_c];
D_Lu = -[D_c1*D_p];
s = ["X_dot_cl = A_Lu*X_cl + B_Lu*u_in";"u_out= C_Lu*X_cl + D_Lu*u_in"];
displayFormula(["'Loop Opened at Plant Input:'";s]);
```

Loop Opened at Plant Input:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\alpha}_c \\ \dot{q}_c \end{pmatrix} = \begin{pmatrix} \frac{Z_\alpha}{V} & 1 & 0 & 0 \\ M_\alpha & 0 & 0 & 0 \\ -Z_\alpha & 0 & 0 & 0 \\ -K_a Z_\alpha & -1 & K_a a_z & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} \frac{Z_\delta}{V} \\ M_\delta \\ -Z_\delta \\ -K_a Z_\delta \end{pmatrix} u_{\text{in}}$$

$$u_{\text{out}} = \begin{pmatrix} K_a K_q Z_\alpha & K_q & -K_a K_q a_z & -K_q a_q \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} + K_a K_q Z_\delta u_{\text{in}}$$

Subbing in the Numerical Values to the $\mathbf{L_u}$ Model

```
A_Lu = double(subs(A_Lu,variables,variable_values));
B_Lu = double(subs(B_Lu,variables,variable_values));
C_Lu = double(subs(C_Lu,variables,variable_values));
D_Lu = double(subs(D_Lu,variables,variable_values));
A_Lu0 = round(vpa(A_Lu),2);
B_Lu0 = round(vpa(B_Lu),2);
C_Lu0 = round(vpa(C_Lu),2);
D_Lu0 = round(vpa(D_Lu),2);
s = ["X_dot_cl = A_Lu0*X_cl + B_Lu0*u_in";"u_out= C_Lu0*X_cl + D_Lu0*u_in"];
displayFormula(["'Loop Opened at Plant Input:'";s]);
```

Loop Opened at Plant Input:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\alpha}_c \\ \dot{q}_c \end{pmatrix} = \begin{pmatrix} -1.3 & 1.0 & 0 & 0 \\ 47.71 & 0 & 0 & 0 \\ 1156.89 & 0 & 0 & 0 \\ -1.74 & -1.0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} + \begin{pmatrix} -0.21 \\ -104.83 \\ 189.95 \\ -0.28 \end{pmatrix} u_{\text{in}}$$

$$u_{\text{out}} = \begin{pmatrix} -0.56 & -0.32 & 0 & 1.92 \end{pmatrix} \begin{pmatrix} \alpha \\ q \\ \alpha_c \\ q_c \end{pmatrix} - 0.09\, u_{\text{in}}$$

```
sys_Lu = ss(A_Lu,B_Lu,C_Lu,D_Lu);
[b,a] = ss2tf(A_Lu,B_Lu,C_Lu,D_Lu);
sys_Lu_TF = tf(b,a);
```

Calculating Stability Margins at L_u

```
lu = allmargin(sys_Lu);
text = ['Gain Margin: ', num2str(20*log10(lu.GainMargin(1))),' db to ',
num2str(20*log10(lu.GainMargin(2))), ' dB,',...
newline 'LGCF @: ', num2str(lu.PMFrequency), ' rad/s',...
```

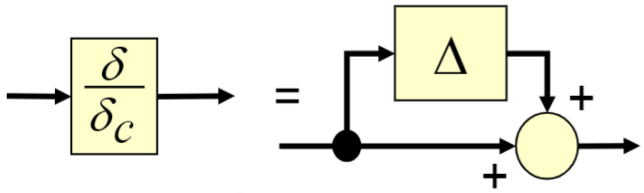
```
newline 'Phase margin: ' num2str(lu.PhaseMargin), ' degrees,',...
newline 'Phase Crossover @: ', num2str(lu.GMFrequency(1)), ' rad/s'];
disp(text)
```

```
Gain Margin: -12.1067 db to 20.8025 dB,
LGCF @: 32.9302 rad/s
Phase margin: 71.0668 degrees,
Phase Crossover @: 4.5341 rad/s
```

c) The actuator dynamics were neglected during the controller design. Derive a multiplicative error model for the neglected actuator dynamics, assuming that the actuator dynamics are modeled using the following transfer function):

$$\frac{\delta}{\delta_c} = \frac{1}{\tau s + 1}$$

We can rewrite the actuator using the model:



$$\frac{\delta}{\delta_c} = 1 + \Delta$$

This is modeling a delay modeled by the constant τ. $\frac{\delta}{\delta_c} = 1 + \Delta$, We need to solve this equation for Δ.

$$\frac{\delta}{\delta_c} = \frac{1}{\tau s + 1} = 1 + \Delta \Rightarrow \Delta = \frac{1}{\tau s + 1} - 1 \Rightarrow \Delta = -\frac{\tau s}{\tau s + 1}$$

For extra analysis between the two models:

```
% Placeholder
actuator = tf([-1],[5 1]);
[Aa,Ba,Ca,Da] = tf2ss([-tau 0],[tau 1]);
% Displaying the Actuator State Space Model
s = ["X_dot_a = Aa*X_a + Ba*u_a";"y_a = Ca*X_a + Da*u_a"];
displayFormula(["'Actuator Delta State Space Model:'";s]);
```

Actuator Delta State Space Model:

$$\dot{X}_a = -\frac{X_a}{\tau} + 1\, u_a$$

$$y_a = \frac{X_a}{\tau} - u_a$$

Combining the Actuator with the Plant model

```
App = [Z_alpha/V 1;M_alpha 0];
Bpp = [Z_delta/V;M_delta];
Cpp = [Z_alpha 0;0 1];
Dpp = [Z_delta;0];

% Plant matrices after actuator
A_pa = [App Bpp*Ca;zeros(1,2) Aa];
B_pa = [0;0;Ba];
C_pa = [Cpp Dpp*Ca];
D_pa = [0;0];

% Displaying the Plant State Space Models
s = ["x_dot = A_pa*x + B_pa*u";"Y = C_pa*x + D_pa*u";
     "x_dot = A_p*x + B_p*u";"Y = C_p*x + D_p*u"];
displayFormula(["'Plant State Space Model With Actuator:'";s(1);s(2); ...
                "'Plant State Space Model With Out Actuator:'";s(3);s(4)]);
```

Plant State Space Model With Actuator:

$$\dot{x} = \begin{pmatrix} \frac{Z_\alpha}{V} & 1 & \frac{Z_\delta}{V\tau} \\ M_\alpha & 0 & \frac{M_\delta}{\tau} \\ 0 & 0 & -\frac{1}{\tau} \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \delta_e$$

$$Y = \begin{pmatrix} Z_\alpha & 0 & \frac{Z_\delta}{\tau} \\ 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \delta_e$$

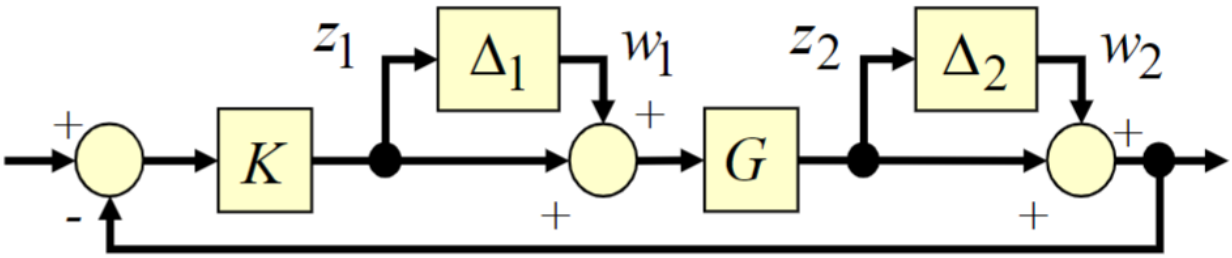
Plant State Space Model With Out Actuator:

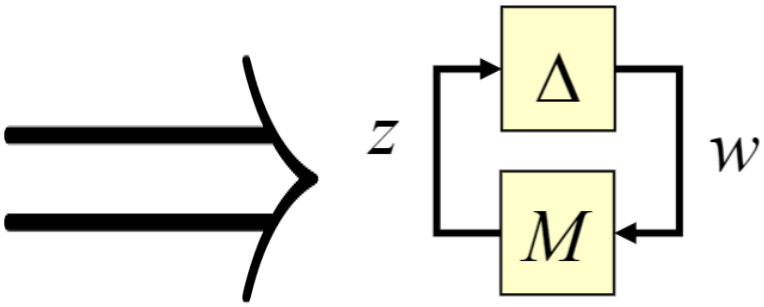
$$\dot{x} = \begin{pmatrix} \frac{Z_\alpha}{V} & 1 \\ M_\alpha & 0 \end{pmatrix} x + \begin{pmatrix} \frac{Z_\delta}{V} \\ M_\delta \end{pmatrix} \delta_e$$

$$Y = \begin{pmatrix} Z_\alpha & 0 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} Z_\delta \\ 0 \end{pmatrix} \delta_e$$

d) Form a ΔM robustness analysis model for analyzing these neglected actuator dynamics.

For robustness analysis, we represent the rest of the system as M. This is modified block diagram is displayed below (neglect the Δ_2 portion since we are only looking at the actuator for now)





By going around the loop, we can calculate M:

$$z = -K(G(z + w))$$

$$z + KGz = -KGw$$

$$\frac{z}{w} = M = \frac{-KG}{I + KG}$$

Now, we can solve for ΔM by multiplying the two:

$$\Delta M = -\frac{\tau s}{\tau s + 1} \cdot \frac{-KG}{I + KG}$$

Creating the closed loop model taking into account the actuator for extra analysis

```
I = eye(size(D_c1*D_pa));
z = (I-D_c1*D_pa);
A_cla = [A_pa+(B_pa*(inv(z))*D_c1*C_pa) B_pa*inv(z)*C_c;...
B_c1*(eye(size(D_pa*inv(z)*D_c1))+D_pa*inv(z)*D_c1)*C_pa
A_c+B_c1*D_pa*inv(z)*C_c];
B_cla = [B_pa*inv(z)*D_c2;B_c2+B_c1*D_pa*inv(z)*D_c2];
C_cla = [(eye(size(D_pa*inv(z)*D_c1))+D_pa*inv(z)*D_c1)*C_pa
D_pa*inv(z)*C_c];
D_cla = [D_pa*inv(z)*D_c2];
X_dot_cl = [X_dot;X_dot_c];
X_cl = [X;X_c];
```

```
s = ["x_dot = A_cla*x + B_cla*r";"y= C_cla*x + D_cla*r"];
displayFormula(["'Complete Closed Loop State Space Model With
Actuator:'";s]);
```

Complete Closed Loop State Space Model With Actuator:

$$\dot{x} = \begin{pmatrix} \frac{Z_\alpha}{V} & 1 & \frac{Z_\delta}{V\,\tau} & 0 & 0 \\ M_\alpha & 0 & \frac{M_\delta}{\tau} & 0 & 0 \\ -K_a\,K_q\,Z_\alpha & -K_q & -\frac{1}{\tau}-\frac{K_a\,K_q\,Z_\delta}{\tau} & K_a\,K_q\,a_z & K_q\,a_q \\ -Z_\alpha & 0 & -\frac{Z_\delta}{\tau} & 0 & 0 \\ -K_a\,Z_\alpha & -1 & -\frac{K_a\,Z_\delta}{\tau} & K_a\,a_z & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ K_a\,K_q \\ 1 \\ K_a \end{pmatrix} r$$

$$\begin{pmatrix} a_z \\ q \end{pmatrix} = \begin{pmatrix} Z_\alpha & 0 & \frac{Z_\delta}{\tau} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \end{pmatrix} r$$

Subbing in the Numerical Values to the Closed Loop Model

```
variables = [V;Z_alpha;Z_delta;M_alpha;M_delta;K_a;K_q;a_z;a_q;tau];
variable_values =
[886.78;-1.3046*886.78;-0.2142*886.78;47.7109;-104.8346;-0.0015;-0.32;2.0;6.
0;1];
A_cl_a = double(subs(A_cla,variables,variable_values));
B_cl_a = double(subs(B_cla,variables,variable_values));
C_cl_a = double(subs(C_cla,variables,variable_values));
D_cl_a = double(subs(D_cla,variables,variable_values));
A_cl0_a = round(vpa(A_cl_a),2);
B_cl0_a = round(vpa(B_cl_a),2);
C_cl0_a = round(vpa(C_cl_a),2);
D_cl0_a = round(vpa(D_cl_a),2);

s = ["x_dot = A_cl0_a*x + B_cl0_a*r";"y= C_cl0_a*x + D_cl0_a*r"];
displayFormula(["'Complete Closed Loop State Space Model With
Actuator:'";s]);
```

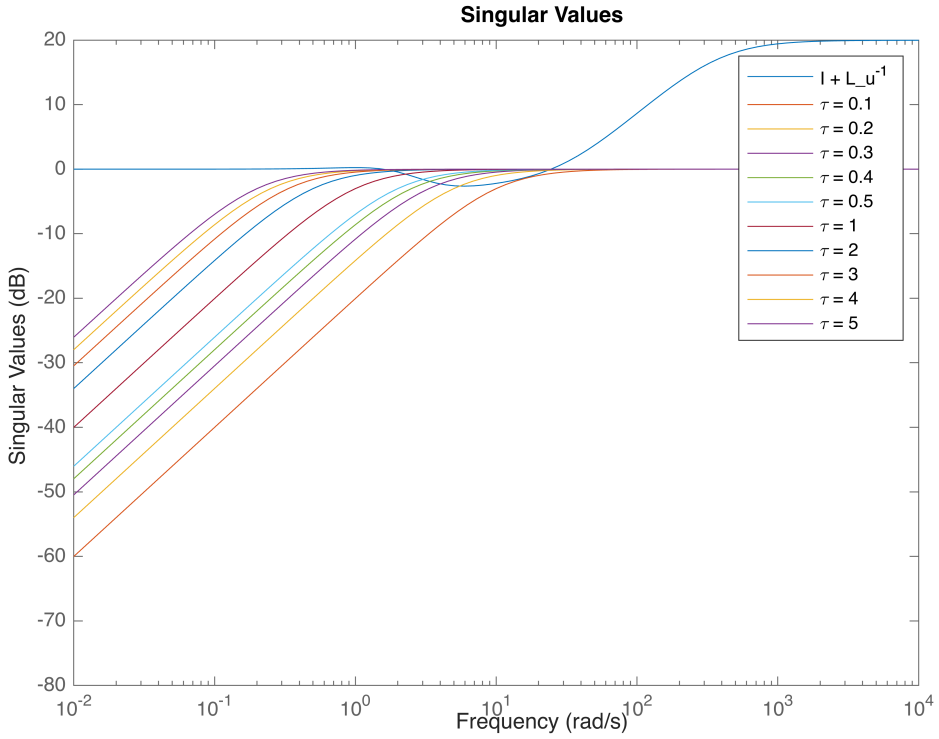
Complete Closed Loop State Space Model With Actuator:

$$\dot{x} = \begin{pmatrix} -1.3 & 1.0 & -0.21 & 0 & 0 \\ 47.71 & 0 & -104.83 & 0 & 0 \\ 0.56 & 0.32 & -0.91 & 0 & -1.92 \\ 1156.89 & 0 & 189.95 & 0 & 0 \\ -1.74 & -1.0 & -0.28 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1.0 \\ 0 \end{pmatrix} r$$

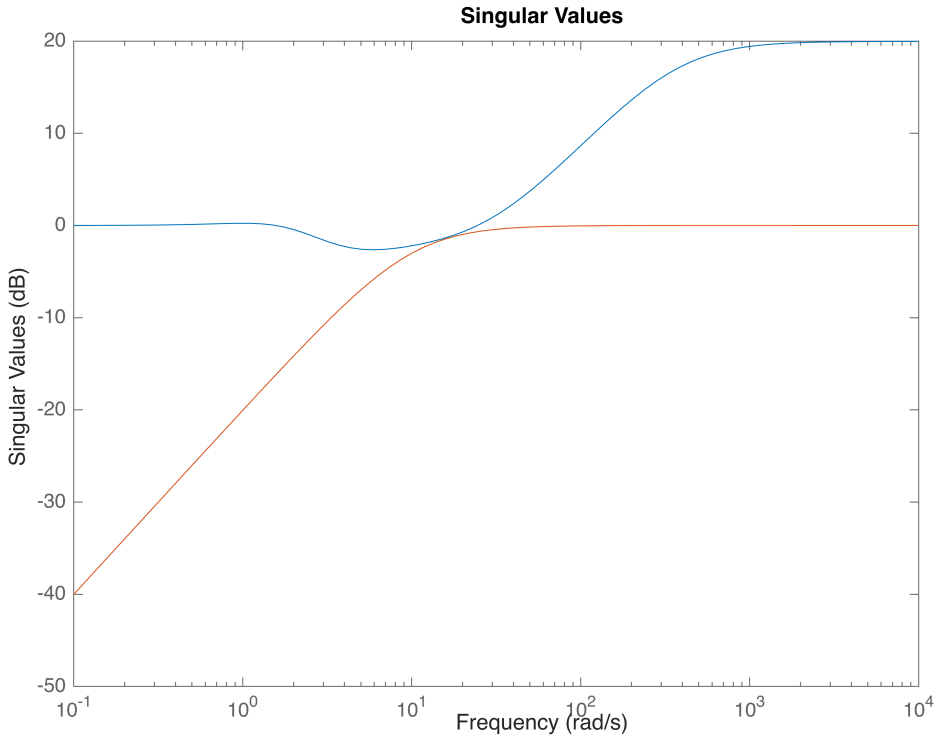
$$\begin{pmatrix} a_z \\ q \end{pmatrix} = \begin{pmatrix} -1156.89 & 0 & -189.95 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \end{pmatrix} r$$

e) Determine the largest actuator time constant τ that results in a stable closed loop system using the small gain theorem as the robustness test.

```
% Testing multiple Tau values
tau = [.1 .2 .3 .4 .5 1 2 3 4 5]; % arbitrary for now
M = -(1+sys_Lu)^-1 * sys_Lu;

sigma(1+sys_Lu^-1)
for i = 1:10
    tau_n = tau (i);
    Delta = tf([-tau_n 0],[tau_n 1]);
    hold on;
    sigma(Delta)
end
hold off;
legend('I + L_u^{-1}',['\tau = ', num2str(tau(1))],...
                      ['\tau = ', num2str(tau(2))],...
                      ['\tau = ', num2str(tau(3))],...
                      ['\tau = ', num2str(tau(4))],...
                      ['\tau = ', num2str(tau(5))],...
                      ['\tau = ', num2str(tau(6))],...
                      ['\tau = ', num2str(tau(7))],...
                      ['\tau = ', num2str(tau(8))],...
                      ['\tau = ', num2str(tau(9))],...
                      ['\tau = ', num2str(tau(10))])
```



```
% With Tau = 0.1
M = -(1+sys_Lu)^-1 * sys_Lu;
tow = 0.1;
Delta = tf([-tow 0],[tow 1]);
sigma(1+sys_Lu^-1)
hold on;
sigma(Delta)
```



With τ = 0.1 the sigma plots do not intersect proving stability, with a higher τ stability would be lost. The system is not very robust with this τ = 0.1, but it is stable according to the small gain theorem.