ESE 547 Homework 3 / Robust Servo LQR /

Seif Elkhashab / 02-16-2024 /

SIMO Longitudinal Airframe Dynamics

1. Using the aircraft pitch-axis plant data from Example 5.2 (Eq. 5.58 page 119) without an actuator, DESIGN a RSLQR to command Az using a state feedback controller. Turn in LQR design charts: Figures 3.4 – 3.8.

$$\begin{bmatrix} A_p & B_p \\ C_p & D_p \end{bmatrix} = \left[\begin{bmatrix} -1.3046 & 1.0 & -0.21420 & 0 \\ 47.711 & 0 & -104.83 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & -1.2769.0 & -1.356 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 12769.0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} -1156.9 & 0 & -189.95 & 0 \\ 0 & 1.0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right] \tag{5.58}$$

In analyzing your design use a 11 Hz actuator model. Eq 5.58 has a actuator model included – you need to use an 11 Hz model. Assume that the elevon fin is rate limited to 100 deg/s for 1 g (32 fps2) commanded.

$$\ddot{\delta}_e = -2\zeta\omega_n\delta_e - \omega_n^2(\delta_e - \delta_c)$$

```
% constants
V = 886.78;
Md = -104.8346;
Ma = 47.7109;
Za = -1.3046*V;
Zd = -0.2142*V;

omega = 11*2*pi;
zeta = .707;

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

% Plant Matricies without the actuator
Ap = [Za/V 1;
       Ma 0];
Bp = [Zd/V;
       Md];
Cp = [Za 0;
       0  1];
Dp = [Zd;
```

```
       0];

% Wiggle System
Aw = [0 Za   0;
      0 Za/V 1;
      0 Ma   0];
Bw = [ Zd;
       Zd/V;
        Md];

% Plant Matricies with the actuator
Ap = [Za/V 1    Zd/V        0;
       Ma  0     Md         0;
       0   0     0          1;
       0   0 -omega^2 -2*zeta*omega];
Bp = [0;
      0;
      0
      omega^2];
Cp = [Za  0 Zd  0;
       1  0  0  0;
       0  1  0  0;
       0  0  1  0;
       0  0  0  1];
Dp = [0;
      0;
      0;
      0;
      0;];

Aw = [zeros(1,1) Cp(1,:);
     zeros(4,1) Ap];
Bw = [Dp(1,:);
      Bp];

% Setup range of penalties for the LQR
Q=0.*Aw; %sets up a matrix Q that is the size of Aw and is all 0s
R=1;

%qq is a vector which each element scales the LQR Q matrix
qq=logspace(-4, 0,200); %these are the varying values of q11
step_info = zeros(size(qq,2),8);
poles = zeros(size(qq,2),5);
margins = zeros(size(qq,2),4);
mins = zeros(size(qq,2),2);
gains = zeros(size(qq,2),5);
for ii = 1:200

    % The only penalty is the 1,1 element on the error state
    % Desing the gains
```

```
    Q(1,1)=qq(ii);
    [Kx_lqr,~,~]=lqr(Aw,Bw,Q,R);

    % populate the controller matrices
    Ac = 0.;
    Bc1 = [1. 0. 0. 0. 0.];
    Bc2 = -1;
    Cc = -Kx_lqr(1);
    Dc1 = [0. -Kx_lqr(2:5)];
    Dc2 = 0;

    Zi = inv(eye(1) - Dc1*Dp);
    Acl = [     Ap + Bp*Zi*Dc1*Cp                Bp*Zi*Cc;
             Bc1*(eye(5) + Dp*Zi*Dc1)*Cp    Ac + Bc1*Dp*Zi*Cc];
    Bcl = [    Bp*Zi*Dc2;
           Bc2 + Bc1*Dp*Zi*Dc2];
    Ccl = [(eye(5) + Dp*Zi*Dc1)*Cp Dp*Zi*Cc];
    Dcl = Dp*Zi*Dc2;
    sys_cl = ss(Acl,Bcl,Ccl,Dcl);

    % at plant input
    Alu = [Ap     zeros(4,1);
           Bc1*Cp     Ac];
    Blu = [  Bp;
           zeros(1,1)];
    Clu = -[Dc1*Cp Cc];
    Dlu = 0;
    sys_u = ss(Alu,Blu,Clu,Dlu);

    % Loop input break Info
    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;

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

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

    info = stepinfo(sys_cl,'RiseTimeThreshold', [0
0.63],'SettlingTimeThreshold', 0.05);
    step_info(ii,:) = [info(1).RiseTime info(1).SettlingTime
info(2).RiseTime, info(2).SettlingTime info(1).Overshoot info(1).Undershoot
info(4).Peak info(5).Peak];
```

```
    poles(ii,:) = eig(Acl);
    margins(ii,:) = [gm_lu pm_lu lgcf_lu pc_lu];
    mins(ii,:) = [alpha_0 beta_0];
    gains(ii,:) = Kx_lqr;

end
```

```
rise_time = step_info(:,1); % Rise time for the first output
settling_time = step_info(:,2); % Settling time for the first output
lgcf_lu = margins(:,3)./(2*pi); % lgcf in Hz
overshoot = step_info(:,5);
undershoot = step_info(:,6);
max_def = step_info(:,7);
max_defrate = step_info(:,8);
```

Figure 3.4 / RSLQR short-period dynamics root locus.

```
figure; % Creates a new figure

% Assuming 'poles' is a matrix with rows corresponding to each iteration
and columns to each pole
for ii = 1:size(poles, 1) % Loop through each set of poles
    % Extract real and imaginary parts of the poles for this iteration
    realParts = real(poles(ii, :));
    imagParts = imag(poles(ii, :));

    % Plotting
    scatter(realParts, imagParts, 50, 'filled', 'DisplayName',
sprintf('Step %d', ii)); % Plot with filled markers
    hold on; % Holds the current plot
end

xlabel('Real Part'); % X-axis label
ylabel('Imaginary Part'); % Y-axis label
title('Pole Locations for Each Step'); % Title of the plot
grid on; % Adds a grid to the plot for better readability
axis equal; % Sets equal scaling on both axes to preserve the aspect ratio

% Add lines to indicate the real and imaginary axes
xline(0, '--k'); % Real axis
yline(0, '--k'); % Imaginary axis
hold off;
```

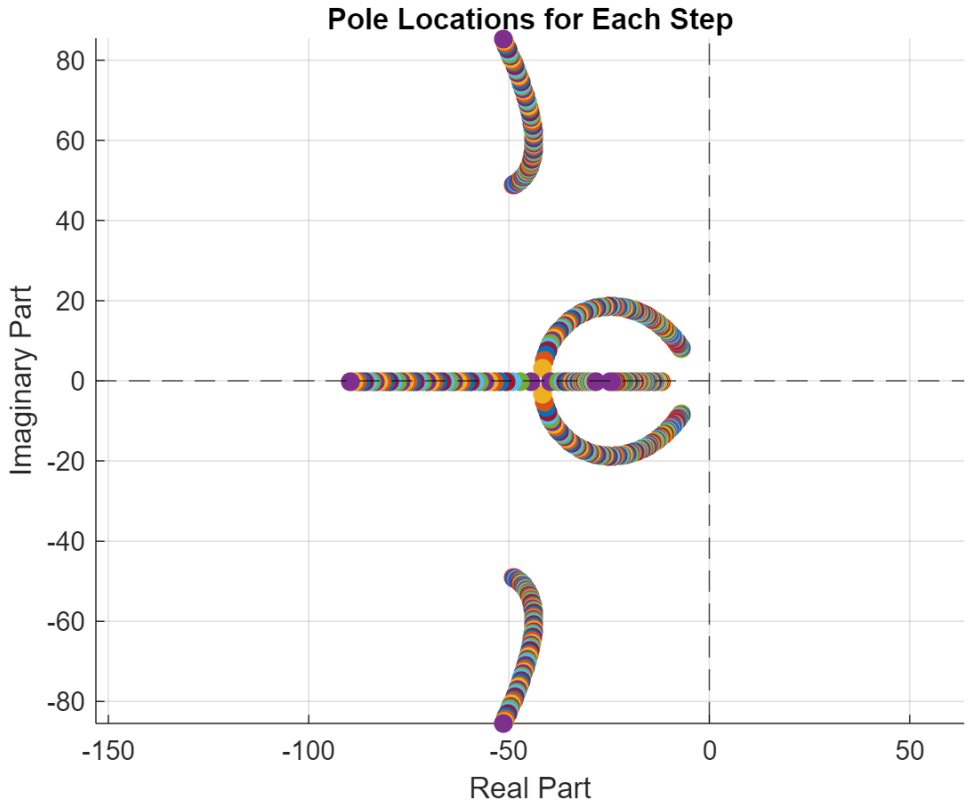


Fig. 3.5 Rise time (blue) and settling time (red) versus loop gain crossover frequency.

```
% Plotting
figure;
plot(lgcf_lu, rise_time, 'b'); % Plot Rise Time in blue
hold on; % Holds the current plot to allow multiple plots in the same figure
plot(lgcf_lu, settling_time, 'r'); % Plot Settling Time in red
xlabel('Loop Gain Crossover Frequency (Hz)'); % X-axis label
ylabel('Time (seconds)'); % Y-axis label
title('Rise Time and Settling Time vs. Loop Gain Crossover Frequency'); %
Title of the plot
legend('Rise Time', 'Settling Time'); % Legend to differentiate between the
two plots
set(gca, 'XScale', 'log');
grid on; % Adds a grid to the plot for better readability
hold off;
```

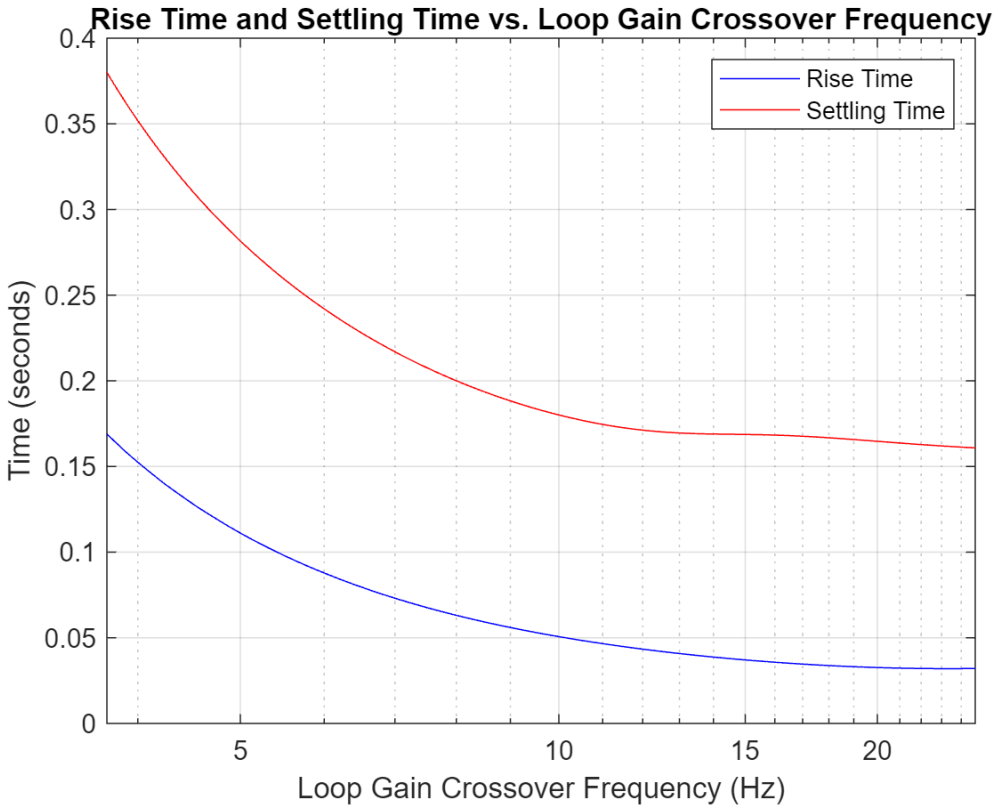


Fig. 3.6 Percent overshoot, undershoot, and max elevon deflection and rate versus loop gain crossover frequency ω_c

```
figure;
subplot(2,2,1),
plot(lgcf_lu, overshoot);
set(gca, 'XScale', 'log'), grid on;
ylabel('% Overshoot');
subplot(2,2,2),
plot(lgcf_lu, undershoot);
set(gca, 'XScale', 'log'), grid on;
ylabel('% Undershoot');
subplot(2,2,3),
plot(lgcf_lu, max_def);
set(gca, 'XScale', 'log'), grid on;
ylabel('Max Elevon (deg/g)');
xlabel('\omega_{c} (Hz)'); % X-axis label
subplot(2,2,4),
plot(lgcf_lu, max_defrate);
ylabel('Max Elevon Rate (dps/g)');
set(gca, 'XScale', 'log'), grid on;
xlabel('\omega_{c} (Hz)'); % X-axis label
```

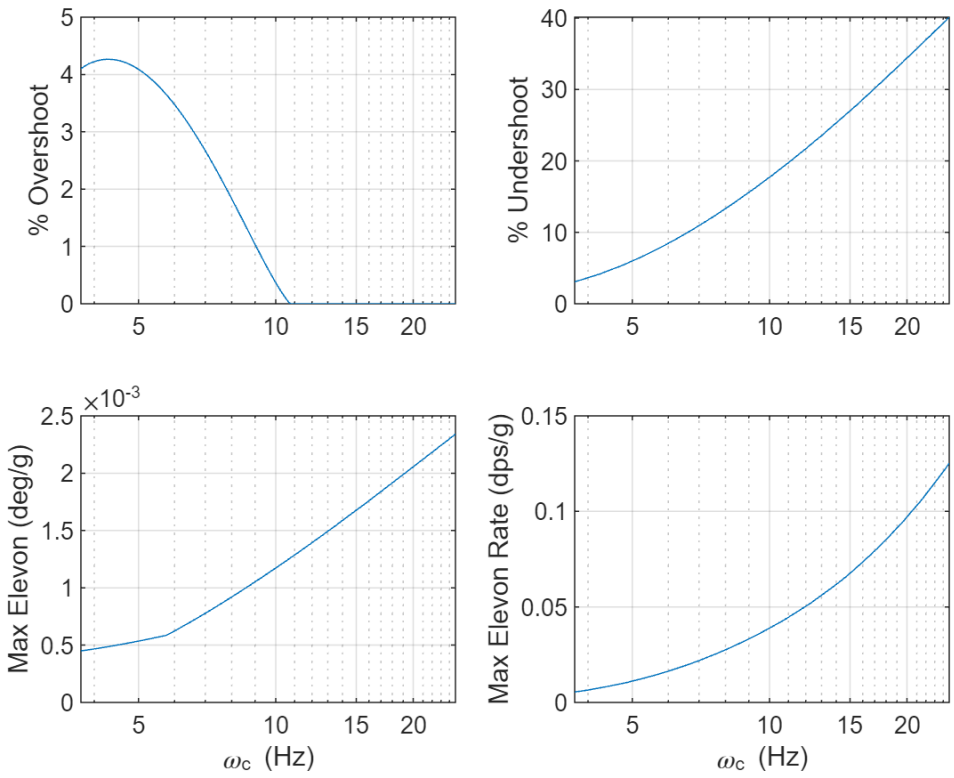
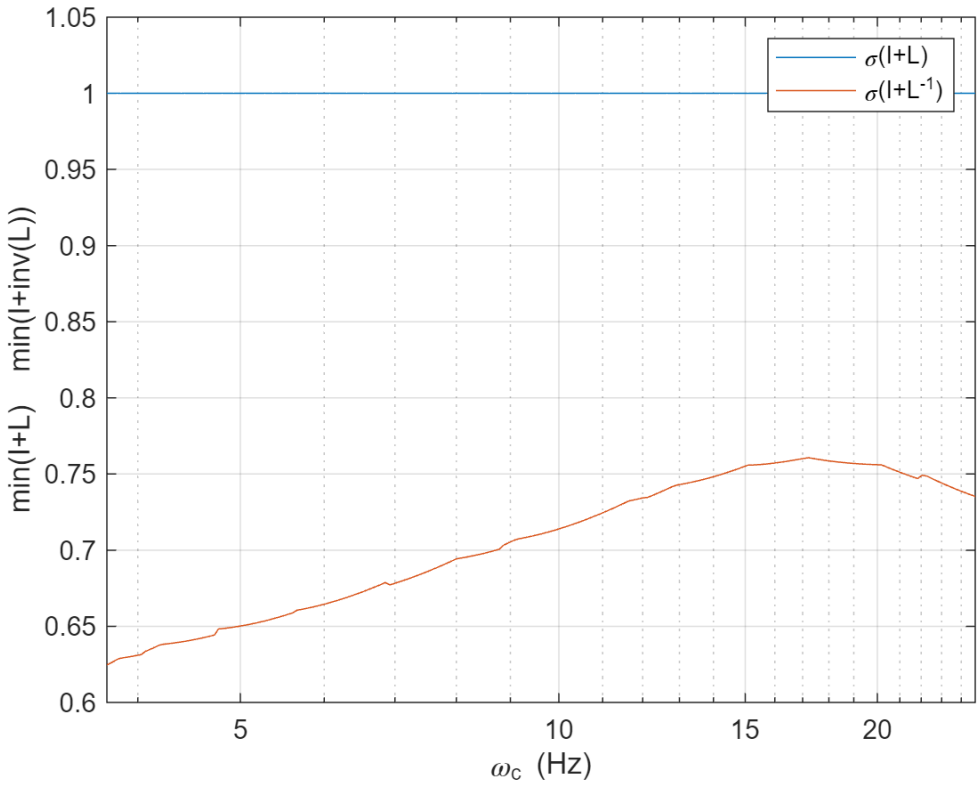


Fig. 3.7 Singular value frequency domain metrics versus loop gain crossover frequency.

```
figure;
plot(lgcf_lu,mins(:,1))
hold on;
plot(lgcf_lu,mins(:,2))
ylabel('min(I+L)    min(I+inv(L))');
legend('\sigma(I+L)','\sigma(I+L^{-1})'); % Using TeX interpreter with an
underline
set(gca, 'XScale', 'log'), grid on;
xlabel('\omega_{c} (Hz)'); % X-axis label
hold off;
```



Using the LQR design charts, show how you evaluated your design in the frequency domain and in the time domain. Describe how you selected the FINAL DESIGN (bandwidth) of your design.

As ω_c increases, the system responds more quickly to the step command. From Fig. 3.5, we can see there is a diminishing return in terms of speed of response as the bandwidth increases (especially after ~10 Hz). This is also evident from the root locus in Fig. 3.4, since the dominant poles approach the zero locations, the change in the pole location diminishes with the increasing gains. The poles headed toward infinity along the asymptotes continue to move, but its contribution to the response $e^{\lambda t}$ dies quickly as the eigenvalues get large and negative. This indicates that large gains are not needed to make the system respond quickly. Figure 3.6 shows that at lower values of ω_c, the response slightly overshoots the command (which we want to avoid).

As the integrator gain increases, above ~11 Hz ω_c, the response has no command overshoot. This metric indicates a desire for larger gains. The percent undershoot characteristic of nonminimum phase responses continues to increase with increasing ω_c which is undesirable. Both the max deflection and max rate increase with increasing ω_c.

As seen in this figure, some of the metrics tend toward increasing the gains, and some tend toward decreasing the gains. Figure 3.7 shows two frequency response metrics: the minimum of the minimum singular value of the

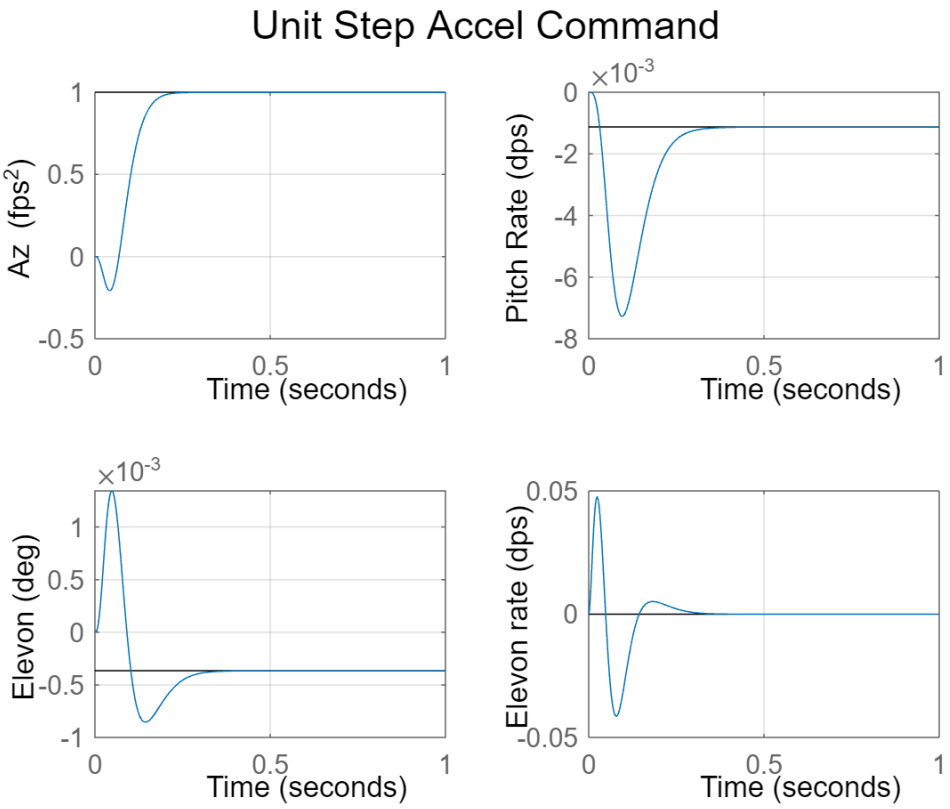
return difference dynamics $\sigma(I + L)$ and the minimum of the minimum singular value of the stability robustness matrix $\sigma(I + L^{-1})$. We would like to maximize $\sigma(I + L^{-1})$. The figure shows that this metric tends to favor larger gains.

With all of these facts in consideration, a bandwidth of 11.5 Hz was selected.This is because of several facts. First, this bandwidth lies in the region where the response has 0 overshoot, which is paramount for a flight control system. With that, this choice also attempts to maximize the minimum singular value of our stability robustness matrix, without increasing our undershoot and max elevon deflection rate beyond reasonable bounds. At a bandwidth of 11.5 Hz, the percent undershoot only reaches 20.7% and the max elevon deflection rate has a value of 0.047 dps/g (which is well within spec). The states of the system responding to a unit step command are shown in Fig 3.8, which is found in the section below.

Fig. 3.8 States of the system responding to a unit acceleration step command.

```
Kx_lqr = gains(124,:); % location where bandwidth is ~11.5 Hz
% populate the controller matrices
Ac = 0.;
Bc1 = [1. 0. 0. 0. 0.];
Bc2 = -1;
Cc = -Kx_lqr(1);
Dc1 = [0. -Kx_lqr(2:5)];
Dc2 = 0;

Zi = inv(eye(1) - Dc1*Dp);
Acl = [     Ap + Bp*Zi*Dc1*Cp                Bp*Zi*Cc;
         Bc1*(eye(5) + Dp*Zi*Dc1)*Cp    Ac + Bc1*Dp*Zi*Cc];
Bcl = [    Bp*Zi*Dc2;
       Bc2 + Bc1*Dp*Zi*Dc2];
Ccl = [(eye(5) + Dp*Zi*Dc1)*Cp Dp*Zi*Cc];
Dcl = Dp*Zi*Dc2;
sys_cl = ss(Acl,Bcl,Ccl,Dcl);

figure;
sgtitle('Unit Step Accel Command')
subplot(2,2,1);
step(sys_cl(1,1));
xlim([0 1]), grid on;
ylabel('Az (fps^2)'), title('')

subplot(2,2,2);
step(sys_cl(3,1));
xlim([0 1]), grid on;
ylabel('Pitch Rate (dps)'), title('')

subplot(2,2,3);
step(sys_cl(4,1));
xlim([0 1]), grid on;
```

```
ylabel('Elevon (deg)'), title('')

subplot(2,2,4);
step(sys_cl(5,1));
xlim([0 1]), grid on;
ylabel('Elevon rate (dps)'), title('')
```

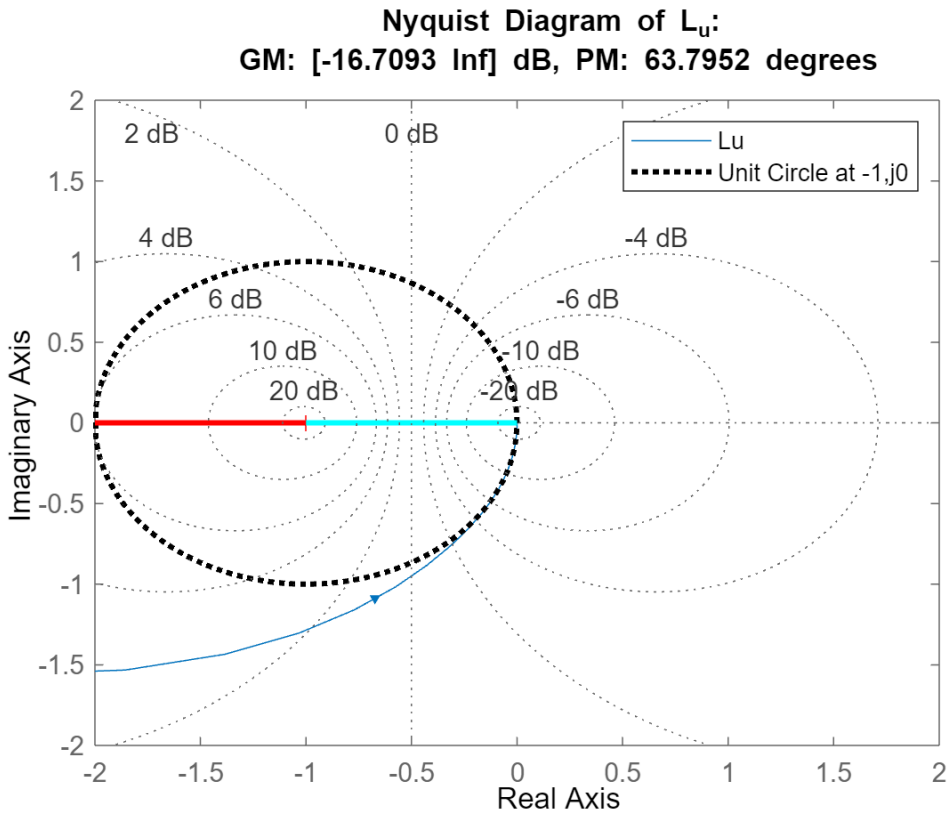


<u>Final Design Analysis</u> ,Break the loop at the plant input and evaluate the design from in the frequency domain.

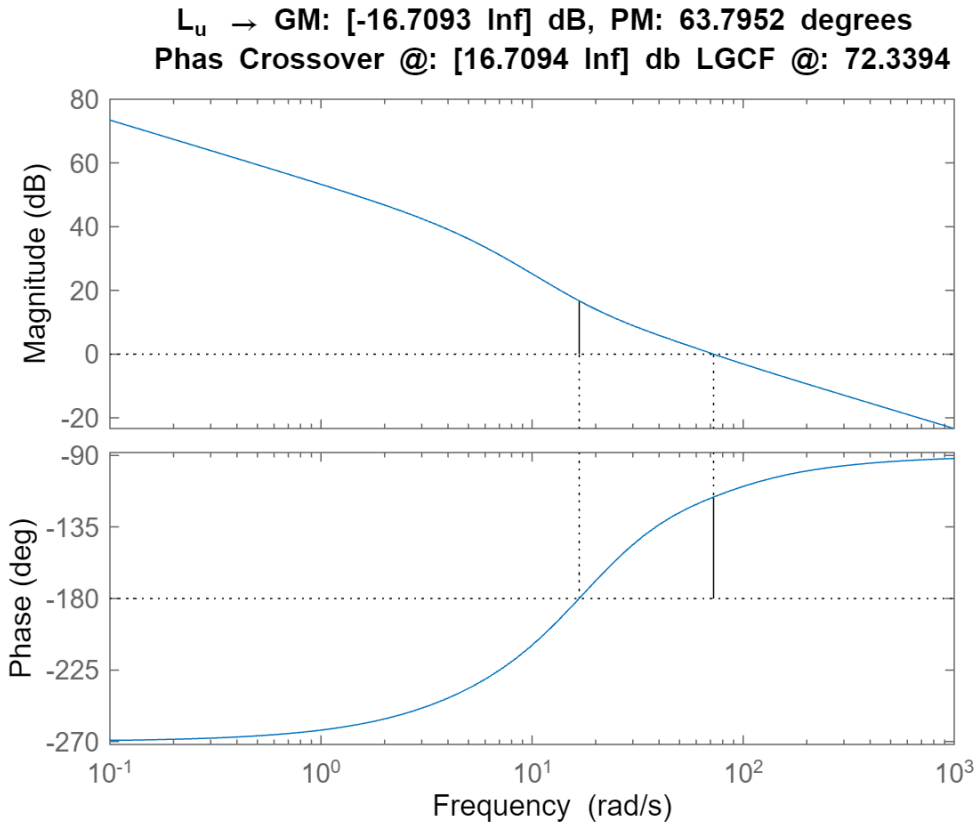
```
% at plant input
Alu = [Ap     zeros(4,1);
       Bc1*Cp     Ac];
Blu = [  Bp;
       zeros(1,1)];
Clu = -[Dc1*Cp Cc];
Dlu = 0;
sys_u = ss(Alu,Blu,Clu,Dlu);

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;
```

i) Plot a Nyquist plot and Bode plot for u L . Identify the LGCF, Phase crossover frequency.

```
figure;
n = nyquistplot(sys_u);
hold on;
plot(xx1,yy1,'k:',[-1/lu_margins.GainMargin(1) -1.],[0. 0.],'r',[-1 0],[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)),' Inf] dB, ' 'PM: ' num2str(pm_lu),
' degrees'])
legend('Lu', 'Unit Circle at -1,j0')
hold off;
```

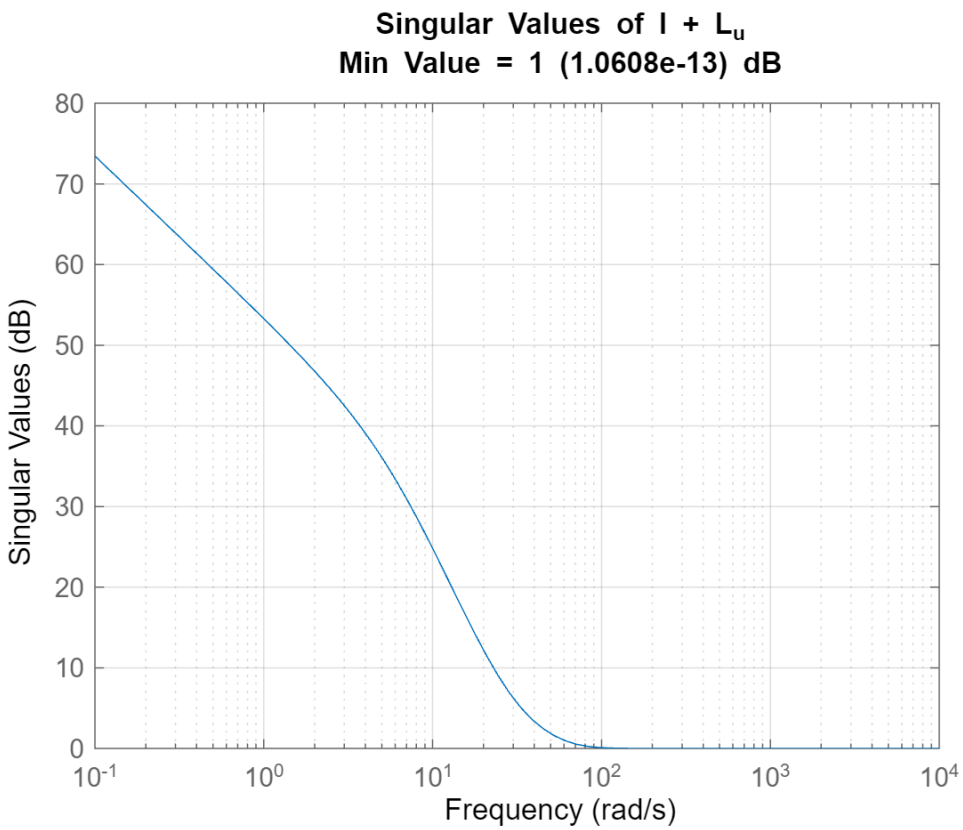



```
figure;
margin(sys_u);
title(['L_u \rightarrow GM: [', num2str(gm_lu(1)), ' Inf] dB, ', 'PM:
',num2str(pm_lu(1)), ' degrees ',...
        newline 'Phas Crossover @: [', num2str(pc_lu(1)),' Inf] db ','LGCF
@: ',num2str(lgcf_lu(1))])
```



ii) Plot $\underline{\sigma}(I + L_u)$ and $\underline{\sigma}\left(I + L_u^{-1}\right)$.

```
[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)
```

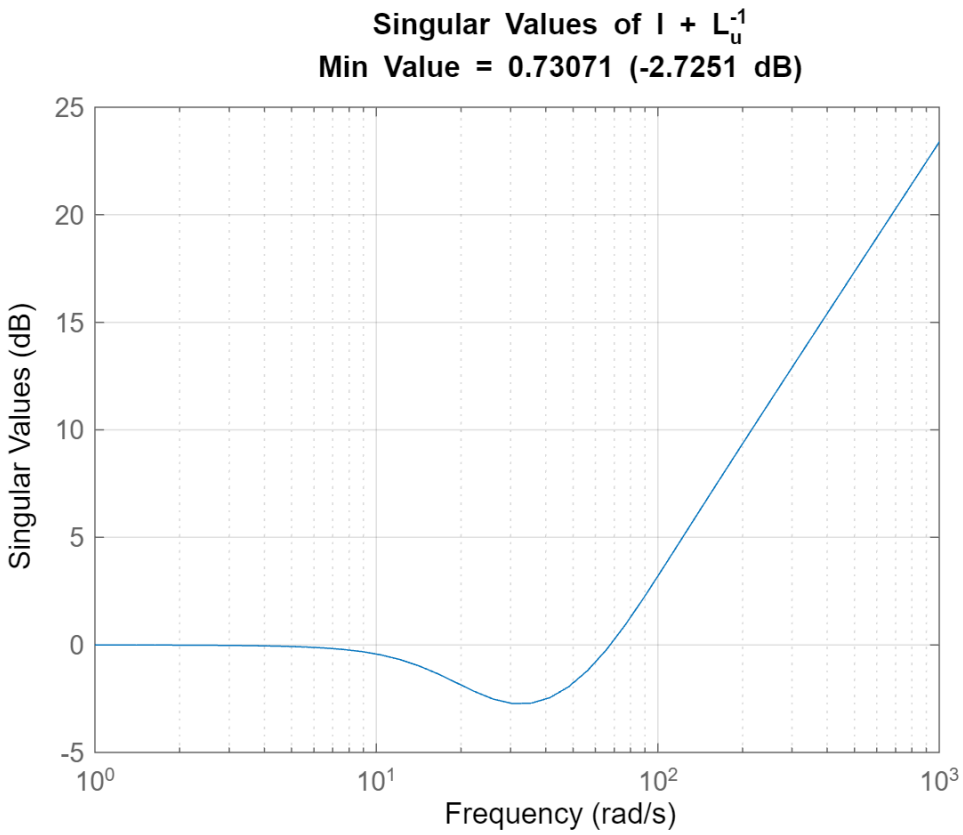
```
Minimum SV of (I +L_u): 1.0608e-13 dB
```

```
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):  [-6.0206, 278.2639+27.28753i] dB
Phase Margin from (I+L_u): [-60, 60] degrees
```

```
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)): -2.7251 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)):  [-11.3956, 4.7645] dB
Phase Margin from inv((I+L_u)): [-42.8588, 42.8588] degrees
```

iii) Compute classical stability margins and singular value stability margins at the plant input.

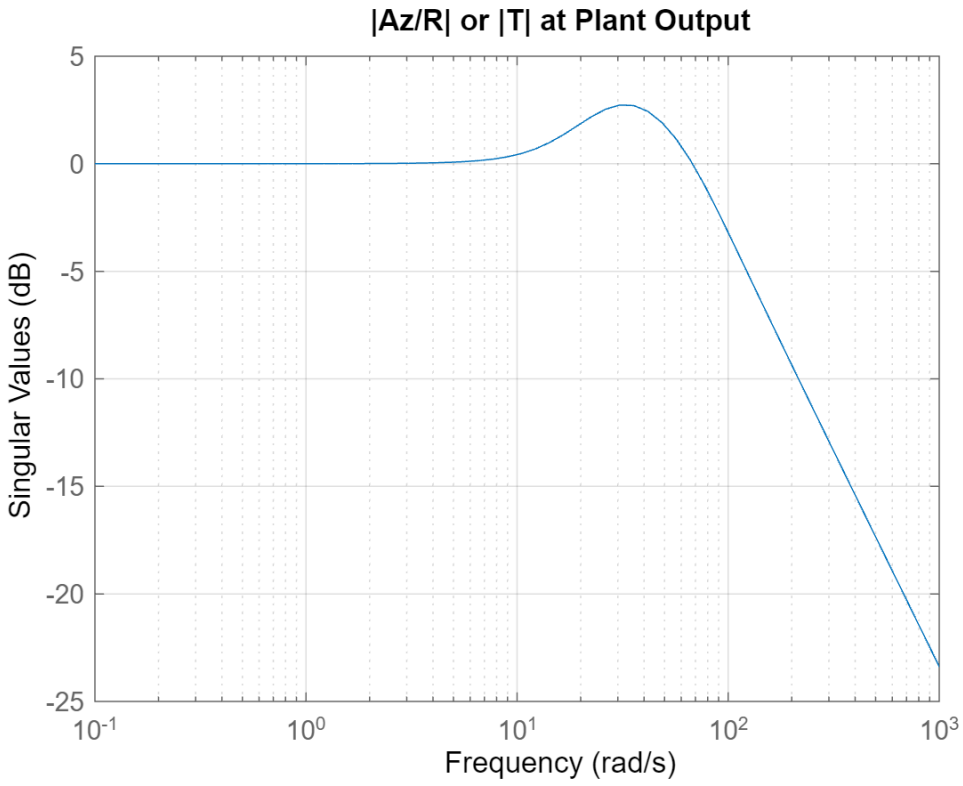
```
Gm_sv = [min(Gm_ILi(1),Gm_IL(1)),max(Gm_ILi(2),Gm_IL(2))];
Pm_sv = [min(pm_ILi(1),pm_IL(1)),max(pm_ILi(2),pm_IL(2))];

text = ['Classical Gain Margin:  [',num2str(gm_lu(1)),', Inf] 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);
```

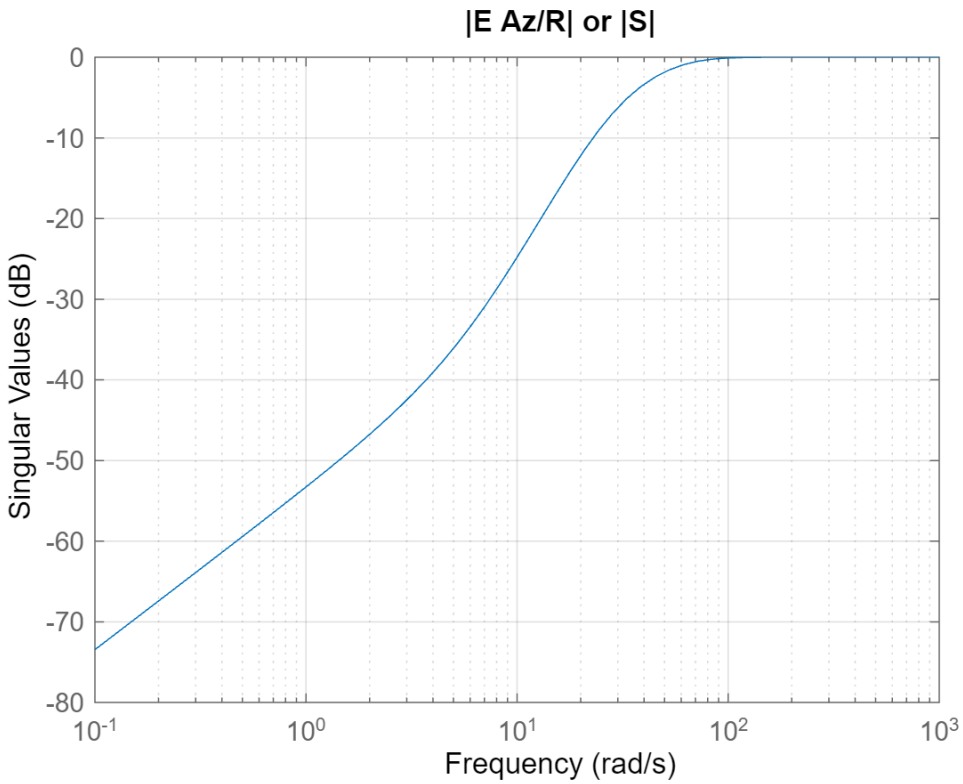
```
Classical Gain Margin:  [-16.7093, Inf] dB
Classical Phase Margin: 63.7952 degrees
Singular Values Gain Margin:  [-11.3956, 278.2639+27.28753i] dB
Singular Values Phase Margin: [-60, 60] degrees
```

iv) Plot the sensitivity S and comp sensitivity T for the commanded variable.

```
T_y = sys_u(1,1)/(1+sys_u(1,1));
sigma(T_y)
grid on, title('|Az/R| or |T| at Plant Output')
xlim([.1 1000])
```




```
S_y = 1/(1+sys_u(1,1));
sigma(S_y), grid on, title('|E Az/R| or |S|')
xlim([.1 1000])
```



List out all matrices used in the final design and analysis, closed loop eigenvalues and eigenvectors.

```
[eigenvectors,eigenvalues] = eig(Acl)
```

```
eigenvectors = 5×5 complex
  -0.0003 + 0.0001i  -0.0003 - 0.0001i  -0.0009 + 0.0027i  -0.0009 - 0.0027i ···
   0.0028 - 0.0217i   0.0028 + 0.0217i  -0.0316 - 0.0912i  -0.0316 + 0.0912i
  -0.0096 - 0.0108i  -0.0096 + 0.0108i  -0.0247 - 0.0168i  -0.0247 + 0.0168i
   0.9988 + 0.0000i   0.9988 + 0.0000i   0.9792 + 0.0000i   0.9792 + 0.0000i
   0.0002 - 0.0414i   0.0002 + 0.0414i  -0.1446 - 0.0997i  -0.1446 + 0.0997i
eigenvalues = 5×5 complex
 -45.8198 +51.6283i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i -45.8198 -51.6283i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i -27.1233 +18.4675i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i -27.1233 -18.4675i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

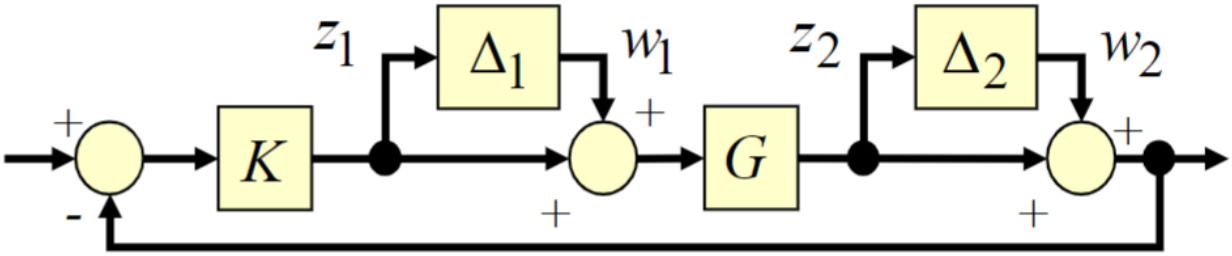
$$\begin{bmatrix} A_p & B_p \\ C_p & D_p \end{bmatrix} = \left[\begin{bmatrix} -1.3046 & 1.0 & -0.21420 & 0 \\ 47.711 & 0 & -104.83 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & -1.2769.0 & -1.356 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 12769.0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} -1156.9 & 0 & -189.95 & 0 \\ 0 & 1.0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right] \tag{5.58}$$

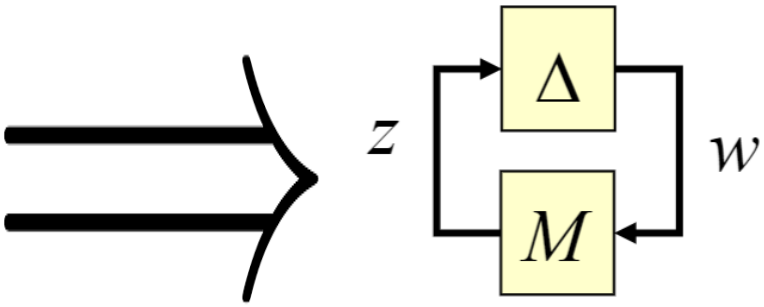
$A_c = (0) \qquad B_{c1} = (1 \quad 0 \quad 0 \quad 0 \quad 0) \qquad B_{c2} = (-1)$

$Cc = (-0.1723) \qquad D_{c1} = (0 \quad 24.49 \quad 1.41 \quad -1.87 \quad -0.01) \qquad D_{c2} = (0)$

Using the small gain theorem, (robustness theory from Chapter 5), show your design is robust to the actuator dynamics (use a 11 Hz actuator with 0.707 damping).

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:

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

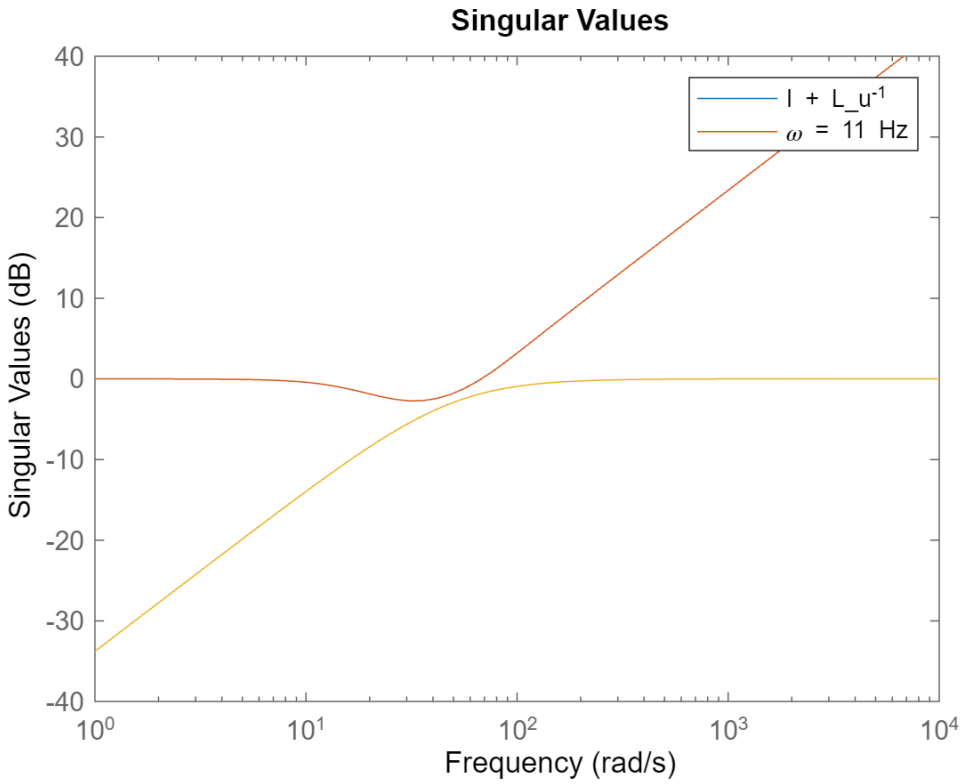
$$z + KGz = -KGw$$

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

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

$$\Delta M = \frac{-s^2 + 2\zeta\omega s}{s^2 - 2\zeta\omega s - \omega^2} \cdot \frac{-KG}{I + KG}$$

```
sigma(1+sys_u^-1);
hold on;
Delta = tf([-1 2*omega*zeta 0],[1 omega*.707 -omega^2]);
sigma(Delta);
legend('I + L_u^{-1}','\omega = 11 Hz ')
hold off;
```



2. List the controller matrices $(A_c, B_{c1}, B_{c2}, C_c, D_{c1}, D_{c2})$ implementing the RSLQR . $y = x_p$

$$\dot{x}_c = A_c x_c + B_{c_1} y + B_{c_2} r$$
$$u = C_c x_c + D_{c_1} y + D_{c_2} r$$

$$\begin{bmatrix} A_p & B_p \\ C_p & D_p \end{bmatrix} = \left[\begin{bmatrix} -1.3046 & 1.0 & -0.21420 & 0 \\ 47.711 & 0 & -104.83 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & -1.2769.0 & -1.356 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 12769.0 \end{bmatrix} \right.$$
$$\left. \begin{bmatrix} -1156.9 & 0 & -189.95 & 0 \\ 0 & 1.0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right] \tag{5.58}$$

$$A_c = (0) \qquad B_{c1} = (1 \quad 0 \quad 0 \quad 0 \quad 0) \qquad B_{c2} = (-1)$$

$$Cc = (-0.1723) \qquad D_{c1} = (0 \quad 24.49 \quad 1.41 \quad -1.87 \quad -0.01) \qquad D_{c2} = (0)$$