ESE 547 Homework 5 /

Seif Elkhashab / 03-01-2024 /

Static Output Feedback Projective Control Design

Describtion:

In this homework you will design 2 controllers, a new RSLQR state feedback controller, and a static projective control controller, and compare them with the results of homework 3 and 4. For the Static Projective Control (SPC) controller we want to use a different state space model for designing the RSLQR.

1. Use the same aircraft plant data from homework 3, build a longitudinal dynamics model that has states $\mathbf{x} = \begin{bmatrix} \mathbf{A_z} & \mathbf{q} & \boldsymbol{\delta_e} & \dot{\boldsymbol{\delta}}_\mathbf{e} \end{bmatrix}^\mathbf{T}$.

$$\begin{bmatrix} \dot{A}_z \\ \dot{q} \\ \dot{\delta}_e \\ \ddot{\delta}_e \end{bmatrix} = \begin{bmatrix} Z_\alpha/V & Z_\alpha & 0 & Z_\delta \\ M_\alpha/Z_\alpha & 0 & \left(M_\delta - M_\alpha Z_\delta/Z_\alpha\right) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} A_z \\ q \\ \delta_e \\ \dot{\delta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \omega_n^2 \end{bmatrix} \delta_c$$

Using this longitudinal dynamics model, design a RSLQR to track a constant acceleration command A_{zc}. The control is $\mathbf{u} = -\mathbf{K}_{\mathbf{x}_{\mathbf{lqr}}} \begin{bmatrix} \mathbf{e_I} & \mathbf{A_z} & \mathbf{q} & \boldsymbol{\delta_e} & \dot{\boldsymbol{\delta}}_\mathbf{e} \end{bmatrix}^\mathbf{T}$. Match the acceleration rise time from homework 3 RSLQR.

1.1 List out your relevant RSLQR design matrices = Awiggle, Bwiggle, Q_lqr, R_lqr, Kx_lqr.

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

1.2 List the controller matrices ( ) 1 2 1 2 Ac ,Bc ,Bc ,Cc ,Dc ,Dc implementing the 5-state RSLQR controller.

```
rslqr5.Kx_lqr = [0.00988495904662556    0.00267905414953521
-0.299814144855726    1.05846215076491    0.00501007920258213];
% populate the controller matrices
rslqr5.Ac = 0.;
rslqr5.Bc1 = [1. 0. 0. 0.];
rslqr5.Bc2 = -1;
rslqr5.Cc = -rslqr5.Kx_lqr(1);
rslqr5.Dc1 = [-rslqr5.Kx_lqr(2:5)];
rslqr5.Dc2 = 0;
```

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

$$Cc = (-0.00988) \qquad D_{c1} = (-0.00268 \quad 0.29981 \quad -1.05846 \quad -0.00501) \qquad D_{c2} = (0)$$

1.3 List out the closed loop eigenvalues and eigenvectors (5-state model).

```
Zi = inv(eye(1) - rslqr5.Dc1*Dp);
rslqr5.Acl = [     Ap + Bp*Zi*rslqr5.Dc1*Cp                Bp*Zi*rslqr5.Cc;
         rslqr5.Bc1*(eye(4) + Dp*Zi*rslqr5.Dc1)*Cp    rslqr5.Ac +
rslqr5.Bc1*Dp*Zi*rslqr5.Cc];
rslqr5.Bcl = [    Bp*Zi*rslqr5.Dc2;
       rslqr5.Bc2 + rslqr5.Bc1*Dp*Zi*rslqr5.Dc2];
rslqr5.Ccl = [(eye(4) + Dp*Zi*rslqr5.Dc1)*Cp Dp*Zi*rslqr5.Cc];
rslqr5.Dcl = Dp*Zi*rslqr5.Dc2;
sys_cl_5 = ss(rslqr5.Acl,rslqr5.Bcl,rslqr5.Ccl,rslqr5.Dcl);

info_rslqr5 = stepinfo(sys_cl_5,'RiseTimeThreshold', [0
0.63],'SettlingTimeThreshold', 0.05);

% Open loop at plant input
rslqr5.Alu = [Ap     zeros(4,1);
       rslqr5.Bc1*Cp     rslqr5.Ac];
rslqr5.Blu = [  Bp;
           rslqr5.Bc1*Dp];
rslqr5.Clu = -[rslqr5.Dc1*Cp rslqr5.Cc];
rslqr5.Dlu = 0;
sys_u_rslqr5 = ss(rslqr5.Alu,rslqr5.Blu,rslqr5.Clu,rslqr5.Dlu);

% Loop input break Info
rslqr5.lu_margins = allmargin(sys_u_rslqr5);
rslqr5.gm_lu = 20*log10(rslqr5.lu_margins.GainMargin);
rslqr5.pm_lu = rslqr5.lu_margins.PhaseMargin;
rslqr5.lgcf_lu = rslqr5.lu_margins.PMFrequency;
rslqr5.pc_lu = rslqr5.lu_margins.GMFrequency;

% Open Loop at plant output
rslqr5.Aly = [Ap           Bp*rslqr5.Cc;
       zeros(1,4)     rslqr5.Ac];
rslqr5.Bly = [Bp*rslqr5.Dc1;
        rslqr5.Bc1];
rslqr5.Cly = -[Cp Dp*rslqr5.Cc];
rslqr5.Dly = 0;
sys_y_rslqr5 = ss(rslqr5.Aly,rslqr5.Bly,rslqr5.Cly,rslqr5.Dly);

[V,D] = eig(Aw-Bw*rslqr5.Kx_lqr)
```

```
V = 5×5 complex
  -0.0096 - 0.0096i  -0.0096 + 0.0096i  -0.0613 - 0.0724i  -0.0613 + 0.0724i ···
   0.9421 + 0.0000i   0.9421 + 0.0000i   0.9954 + 0.0000i   0.9954 + 0.0000i
  -0.0045 - 0.0058i  -0.0045 + 0.0058i   0.0032 - 0.0067i   0.0032 + 0.0067i
  -0.0048 - 0.0006i  -0.0048 + 0.0006i  -0.0007 - 0.0006i  -0.0007 + 0.0006i
   0.2631 - 0.2071i   0.2631 + 0.2071i   0.0095 - 0.0010i   0.0095 + 0.0010i
D = 5×5 complex
 -48.8530 +48.8856i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i -48.8530 -48.8856i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -6.7762 + 8.0093i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -6.7762 - 8.0093i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

2. Design a SPC otput feedback controller to project out the actuator. Using the 5-state RSLQR state feedback controller from 1) above, design a SPC controller using $\mathbf{y} = [\mathbf{e_I} \quad \mathbf{A_z} \quad \mathbf{q}]^{\mathbf{T}}$, to keep the dominant RSLQR eigenstructure. (This design includes the int-error state which will then be put in the controller state space model). The control law is $\mathbf{u} = -\bar{\mathbf{K}}[\mathbf{e_I} \quad \mathbf{A_z} \quad \mathbf{q}]^{\mathbf{T}}$.

2.1 List out your projective control design matrices , r C X K .

```
C = [1 0 0 0 0;
     0 1 0 0 0;
     0 0 1 0 0];

X_r = [V(:,3) V(:,4) V(:,5)];

SPC.K_bar = real(rslqr5.Kx_lqr*X_r*inv(C*X_r))
```

```
SPC = struct with fields:
    K_bar: [0.0036 0.0011 -0.1321]
```

2.2 List the controller matrices $(A_c, B_{c1}, B_{c2}, C_c, D_{c1}, D_{c2})$ implementing the output feedback SPC controller.

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

$$u = C_c x_c + D_{c_1} y + D_{c_2} r$$

```
SPC.Ac = 0;
SPC.Bc1 = [1 0];
SPC.Bc2 = -1;
SPC.Cc = -SPC.K_bar(1);
SPC.Dc1 = -SPC.K_bar(2:3);
SPC.Dc2 = 0;
```

2.3 List out the closed loop eigenvalues and eigenvectors (5-state model still includes actuator) using the output feedback SPC controller.

```
Cp = [ 1  0  0  0;
       0  1  0  0];
Dp = [0;
      0];

Zi = inv(eye(1) -SPC. Dc1*Dp);
SPC.Acl = [     Ap + Bp*Zi*SPC.Dc1*Cp                Bp*Zi*SPC.Cc;
```

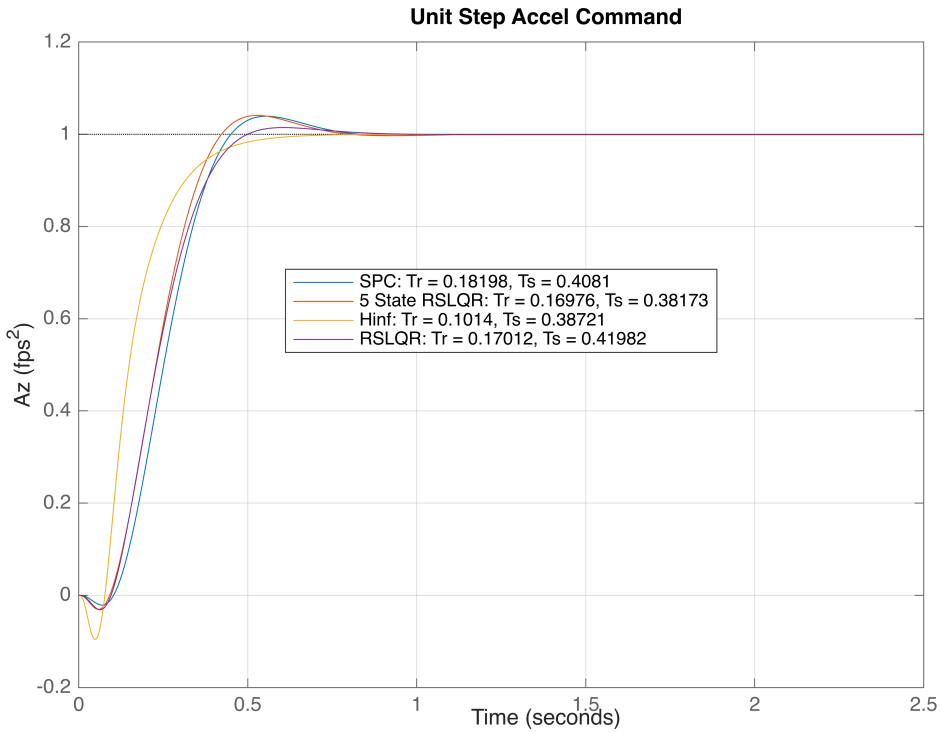
```
         SPC.Bc1*(eye(2) + Dp*Zi*SPC.Dc1)*Cp    SPC.Ac +
SPC.Bc1*Dp*Zi*SPC.Cc];
SPC.Bcl = [    Bp*Zi*SPC.Dc2;
       SPC.Bc2 + SPC.Bc1*Dp*Zi*SPC.Dc2];
SPC.Ccl = [(eye(2) + Dp*Zi*SPC.Dc1)*Cp Dp*Zi*SPC.Cc];
SPC.Dcl = Dp*Zi*SPC.Dc2;
sys_cl_spc = ss(SPC.Acl,SPC.Bcl,SPC.Ccl,SPC.Dcl);

info_spc = stepinfo(sys_cl_spc,'RiseTimeThreshold', [0
0.63],'SettlingTimeThreshold', 0.05);

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

% Loop input break Info
SPC.lu_margins = allmargin(sys_u_spc);
SPC.gm_lu = 20*log10(SPC.lu_margins.GainMargin);
SPC.pm_lu = SPC.lu_margins.PhaseMargin;
SPC.lgcf_lu = SPC.lu_margins.PMFrequency;
SPC.pc_lu = SPC.lu_margins.GMFrequency;

% Open Loop at plant output
SPC.Aly = [Ap           Bp*SPC.Cc;
       zeros(1,4)     SPC.Ac];
SPC.Bly = [Bp*SPC.Dc1;
        SPC.Bc1];
SPC.Cly = -[Cp Dp*SPC.Cc];
SPC.Dly = 0;
sys_y_spc = ss(SPC.Aly,SPC.Bly,SPC.Cly,SPC.Dly);
[eigenvectors,eigenvalues] = eig(SPC.Acl)
```

```
eigenvectors = 5×5 complex
  -0.9696 + 0.0000i  -0.9696 + 0.0000i  -0.9963 + 0.0000i   0.9954 + 0.0000i ···
   0.0099 + 0.0114i   0.0099 - 0.0114i  -0.0103 + 0.0000i   0.0032 - 0.0067i
   0.0055 + 0.0021i   0.0055 - 0.0021i  -0.0007 + 0.0000i  -0.0007 - 0.0006i
  -0.2417 + 0.0270i  -0.2417 - 0.0270i   0.0083 + 0.0000i   0.0095 - 0.0010i
   0.0208 + 0.0107i   0.0208 - 0.0107i   0.0851 + 0.0000i  -0.0613 - 0.0724i
eigenvalues = 5×5 complex
 -36.8867 +18.8683i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i -36.8867 -18.8683i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i -11.7074 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -6.7762 + 8.0093i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

Compare the 5-state RSLQR, SPC controller, homework 3 3-state RSLQR controller, and Homework 4 Hinf controller.

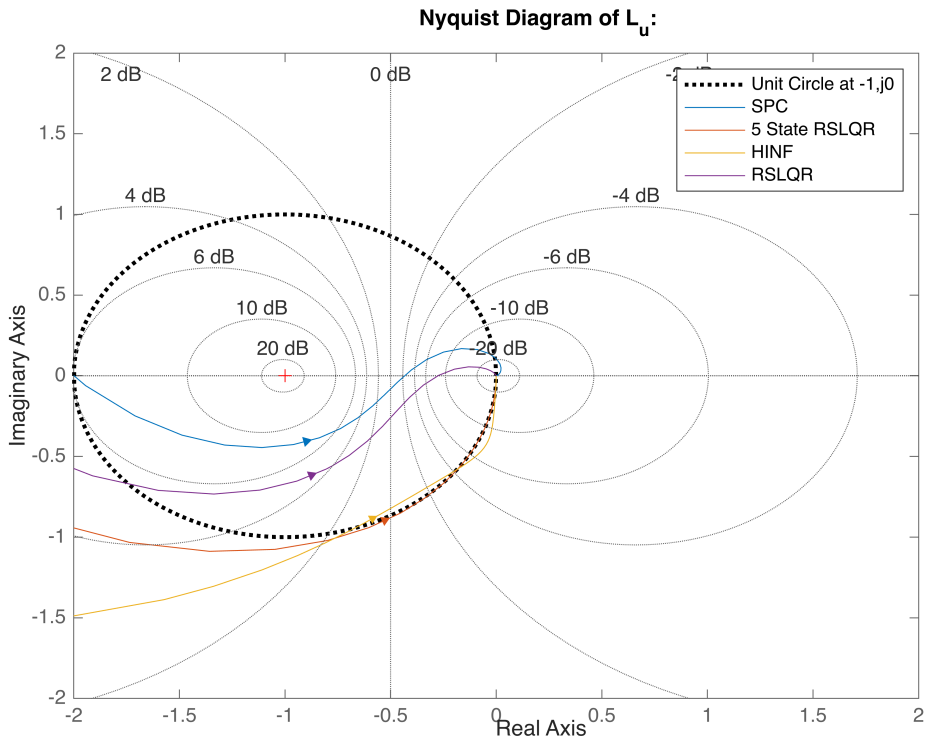
a) Simulate the closed loop system to a unit step Az command. Plot all of the acceleration response from each of the controllers. Compute the rise time and settling time for each. Label this on the plot.

```
% Hw 4 results for comparison
load("HWdata.mat")
sys_cl_hinf = ss(Hinf.Acl,Hinf.Bcl,Hinf.Ccl,Hinf.Dcl);
info_hinf = stepinfo(sys_cl_hinf,'RiseTimeThreshold', [0
0.63],'SettlingTimeThreshold', 0.05);
sys_u_hinf = ss(Hinf.Alu,Hinf.Blu,Hinf.Clu,Hinf.Dlu);
sys_y_hinf = ss(Hinf.Aly,Hinf.Bly,Hinf.Cly,Hinf.Dly);

sys_cl_rslqr = ss(rslqr.Acl,rslqr.Bcl,rslqr.Ccl,rslqr.Dcl);
info_rslqr = stepinfo(sys_cl_rslqr,'RiseTimeThreshold', [0
0.63],'SettlingTimeThreshold', 0.05);
sys_u_rslqr = ss(rslqr.Alu,rslqr.Blu,rslqr.Clu,rslqr.Dlu);
sys_y_rslqr = ss(rslqr.Aly,rslqr.Bly,rslqr.Cly,rslqr.Dly);

figure;
step(sys_cl_spc(1,1))
hold on;
step(sys_cl_5(1,1))
step(sys_cl_hinf(1,1));
step(sys_cl_rslqr(1,1))
xlim([0 2.5]), grid on;
legend(['SPC: Tr = ',num2str(info_spc(1).RiseTime),', Ts =
',num2str(info_spc(1).SettlingTime)], ...
       ['5 State RSLQR: Tr = ',num2str(info_rslqr5(1).RiseTime),', Ts =
',num2str(info_rslqr5(1).SettlingTime)], ...
       ['Hinf: Tr = ',num2str(info_hinf(1).RiseTime),', Ts =
',num2str(info_hinf(1).SettlingTime)], ...
       ['RSLQR: Tr = ',num2str(info_rslqr(1).RiseTime),', Ts =
',num2str(info_rslqr(1).SettlingTime)],'Location', 'best')
ylabel('Az (fps^2)'), title('Unit Step Accel Command')
hold off
```



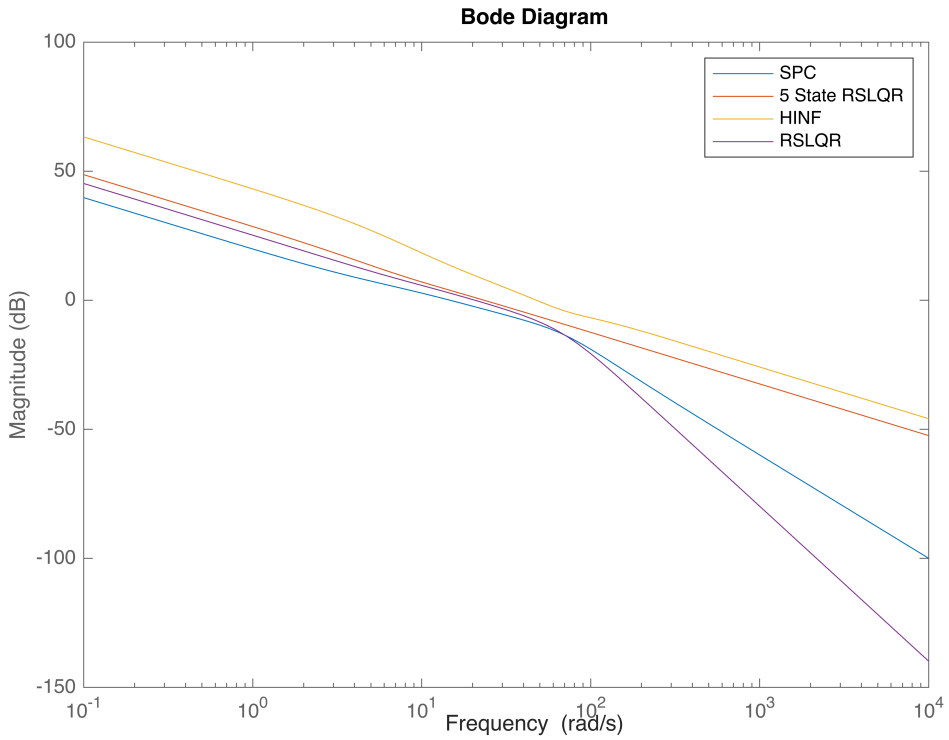
b) Plot a Nyquist plot comparing the frequency response for each of the controllers.

```
figure;
plot(xx1,yy1,'k:','LineWidth',2)
hold on;
n = nyquistplot(sys_u_spc);
n = nyquistplot(sys_u_rslqr5);
n = nyquistplot(sys_u_hinf);
n = nyquistplot(sys_u_rslqr);grid
xlim([-2,2]), ylim([-2,2])
setoptions(n,'ShowfullContour','off'), grid on
title('Nyquist Diagram of L_u: ')
legend('Unit Circle at -1,j0', 'SPC', '5 State RSLQR','HINF','RSLQR')
hold off;
```



c) Plot a Bode gain plot comparing the frequency response for each of the controllers.

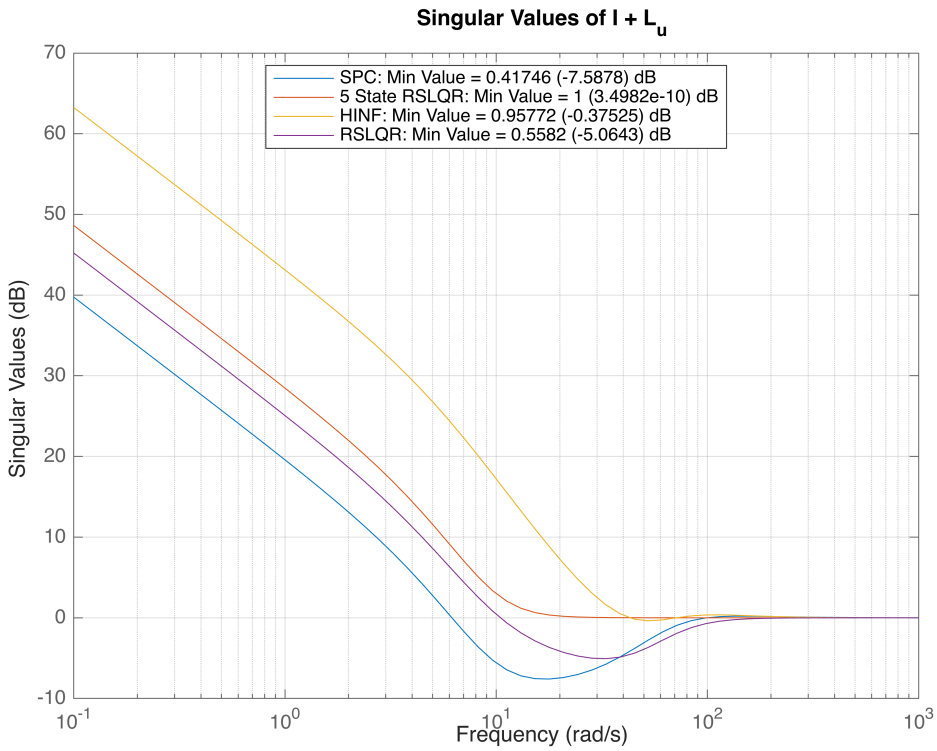
```
figure;
bodemag(sys_u_spc);
hold on;
bodemag(sys_u_rslqr5);
bodemag(sys_u_hinf);
bodemag(sys_u_rslqr)
legend('SPC', '5 State RSLQR', 'HINF', 'RSLQR')
hold off;
```



d) Plot the minimum singular value of the return difference matrix in dB vs frequency comparing each controller. Identify on the plot the minimum value of the return difference matrix (not in dB). Plot the RSLQR $\underline{\sigma}(I + L_u)$ including the actuator in the plant model with the Hinf $\underline{\sigma}(I + L_u)$ (both plant models should have the actuator in them).

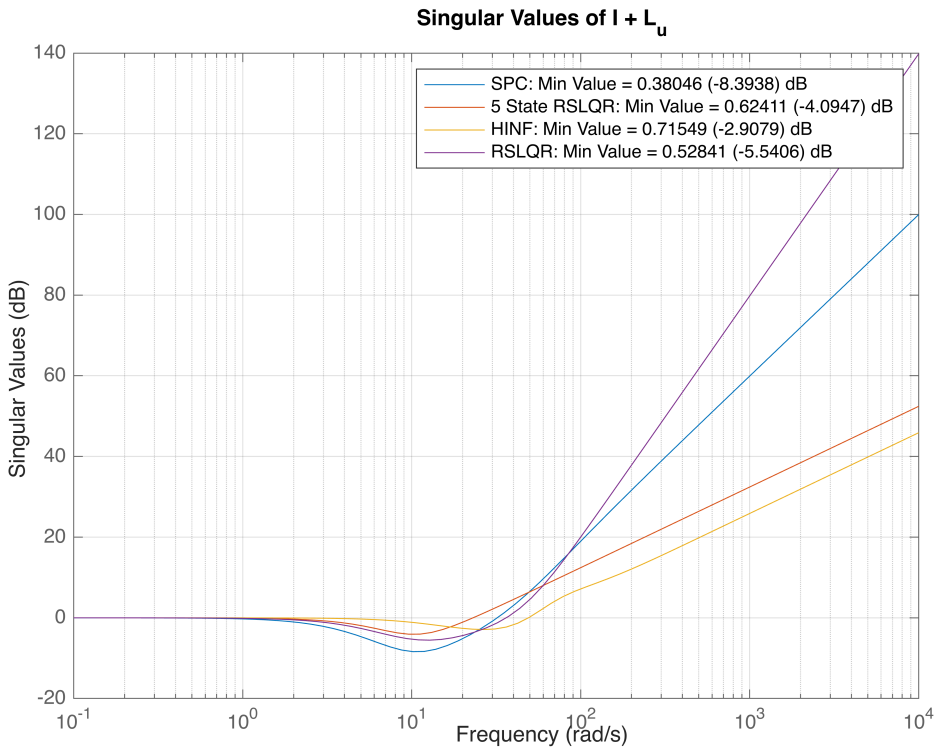
```
[a,b] = ss2tf(SPC.Alu,SPC.Blu,SPC.Clu,SPC.Dlu);
c = b + a;
sys_IL_spc = tf(c,b);
alpha_0_spc = min(sigma(sys_IL_spc));

sys_ILi_spc = tf(c,a);
beta_0_spc = (min(sigma(sys_ILi_spc)));

[a,b] = ss2tf(rslqr5.Alu,rslqr5.Blu,rslqr5.Clu,rslqr5.Dlu);
c = b + a;
sys_IL_rslqr5 = tf(c,b);
alpha_0_rslqr5 = min(sigma(sys_IL_rslqr5));

sys_ILi_rslqr5 = tf(c,a);
beta_0_rslqr5 = (min(sigma(sys_ILi_rslqr5)));

[a,b] = ss2tf(Hinf.Alu,Hinf.Blu,Hinf.Clu,Hinf.Dlu);
c = b + a;
```

```
sys_IL_hinf = tf(c,b);
alpha_0_hinf = min(sigma(sys_IL_hinf));

sys_ILi_hinf = tf(c,a);
beta_0_hinf = (min(sigma(sys_ILi_hinf)));

[a,b] = ss2tf(rslqr.Alu,rslqr.Blu,rslqr.Clu,rslqr.Dlu);
c = b + a;
sys_IL_rslqr = tf([c 0],[b 0]);
alpha_0_rslqr = min(sigma(sys_IL_rslqr));

sys_ILi_rslqr = tf(c,a);
beta_0_rslqr = (min(sigma(sys_ILi_rslqr)));

figure;
sigma(sys_IL_spc);
hold on;
sigma(sys_IL_rslqr5);
sigma(sys_IL_hinf);
sigma(sys_IL_rslqr);
legend(['SPC: Min Value = ',num2str(alpha_0_spc), ' (',
num2str(20*log10(alpha_0_spc)),') dB'],...
        ['5 State RSLQR: Min Value = ',num2str(alpha_0_rslqr5), ' (',
num2str(20*log10(alpha_0_rslqr5)),') dB'],...
        ['HINF: Min Value = ',num2str(alpha_0_hinf), ' (',
num2str(20*log10(alpha_0_hinf)),') dB'],...
       ['RSLQR: Min Value = ',num2str(alpha_0_rslqr), ' (',
num2str(20*log10(alpha_0_rslqr)),') dB'],'Location','best')
grid on, title('Singular Values of I + L_u')
hold off;
```



e) Plot the minimum singular value of the stability robustness matrix in dB vs frequency comparing each controller. Identify on the plot the minimum value of the stability robustness matrix (not in dB). Plot the RSLQR $\underline{\sigma}\left(I + L_u^{-1}\right)$ including the actuator in the plant model with the Hinf $\underline{\sigma}\left(I + L_u^{-1}\right)$ (both plant models should have the actuator in them).

```
figure;
sigma(sys_ILi_spc);
hold on
sigma(sys_ILi_rslqr5);
sigma(sys_ILi_hinf);
sigma(sys_ILi_rslqr);
legend(['SPC: Min Value = ',num2str(beta_0_spc), ' (',
num2str(20*log10(beta_0_spc)),') dB'],...
        ['5 State RSLQR: Min Value = ',num2str(beta_0_rslqr5), ' (',
num2str(20*log10(beta_0_rslqr5)),') dB'],...
        ['HINF: Min Value = ',num2str(beta_0_hinf), ' (',
num2str(20*log10(beta_0_hinf)),') dB'],...
       ['RSLQR: Min Value = ',num2str(beta_0_rslqr), ' (',
num2str(20*log10(beta_0_rslqr)),') dB'])
grid on, title('Singular Values of I + L_u')
hold off;
```



f) Compute the singular value gain and phase margins for the system (at the plant input) and compare them with the other controllers.

```
pm = 2*asin(alpha_0_hinf/2)*180/pi;
GM_l = 20*log10(1/(1 + alpha_0_hinf));
GM_u = 20*log10(1/(1 - alpha_0_hinf));
Gm_IL_hinf = [GM_l GM_u];
pm_IL_hinf = [-pm pm];


pm = 2*asin(alpha_0_rslqr/2)*180/pi;
GM_l = 20*log10(1/(1 + alpha_0_rslqr));
GM_u = 20*log10(1/(1 - alpha_0_rslqr));
Gm_IL_rslqr = [GM_l GM_u];
pm_IL_rslqr = [-pm pm];

GM_u = 20*log10(1 + beta_0_hinf);
GM_l = 20*log10(1 - beta_0_hinf);
pm = 2*asin(beta_0_hinf/2)*180/pi;
Gm_ILi_hinf = [GM_l GM_u];
pm_ILi_hinf = [-pm pm];

GM_u = 20*log10(1 + beta_0_rslqr);
GM_l = 20*log10(1 - beta_0_rslqr);
pm = 2*asin(beta_0_rslqr/2)*180/pi;
```

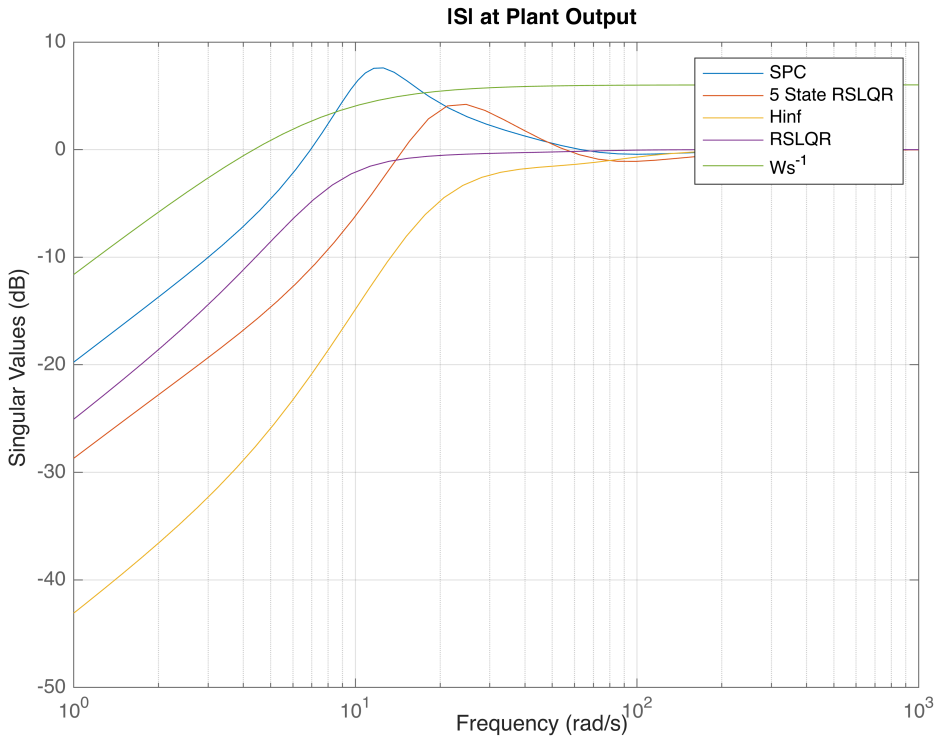
```
Gm_ILi_rslqr = [GM_l GM_u];
pm_ILi_rslqr = [-pm pm];

pm = 2*asin(alpha_0_rslqr5/2)*180/pi;
GM_l = 20*log10(1/(1 + alpha_0_rslqr5));
GM_u = 20*log10(1/(1 - alpha_0_rslqr5));
Gm_IL_rslqr5 = [GM_l GM_u];
pm_IL_rslqr5 = [-pm pm];


pm = 2*asin(alpha_0_spc/2)*180/pi;
GM_l = 20*log10(1/(1 + alpha_0_spc));
GM_u = 20*log10(1/(1 - alpha_0_spc));
Gm_IL_spc = [GM_l GM_u];
pm_IL_spc = [-pm pm];

GM_u = 20*log10(1 + beta_0_spc);
GM_l = 20*log10(1 - beta_0_spc);
pm = 2*asin(beta_0_spc/2)*180/pi;
Gm_ILi_spc = [GM_l GM_u];
pm_ILi_spc = [-pm pm];

GM_u = 20*log10(1 + beta_0_rslqr5);
GM_l = 20*log10(1 - beta_0_rslqr5);
pm = 2*asin(beta_0_rslqr5/2)*180/pi;
Gm_ILi_rslqr5 = [GM_l GM_u];
pm_ILi_rslqr5 = [-pm pm];


Gm_sv_spc =
[min(Gm_ILi_spc(1),Gm_IL_spc(1)),max(Gm_ILi_spc(2),Gm_IL_spc(2))];
pm_sv_spc =
[min(pm_ILi_spc(1),pm_IL_spc(1)),max(pm_ILi_spc(2),pm_IL_spc(2))];
Gm_sv_rslqr5 =
[min(Gm_ILi_rslqr5(1),Gm_IL_rslqr5(1)),max(Gm_ILi_rslqr5(2),Gm_IL_rslqr5(2))
];
pm_sv_rslqr5 =
[min(pm_ILi_rslqr5(1),pm_IL_rslqr5(1)),max(pm_ILi_rslqr5(2),pm_IL_rslqr5(2))
];


Gm_sv_hinf =
[min(Gm_ILi_hinf(1),Gm_IL_hinf(1)),max(Gm_ILi_hinf(2),Gm_IL_hinf(2))];
pm_sv_hinf =
[min(pm_ILi_hinf(1),pm_IL_hinf(1)),max(pm_ILi_hinf(2),pm_IL_hinf(2))];

Gm_sv_rslqr =
[min(Gm_ILi_rslqr(1),Gm_IL_rslqr(1)),max(Gm_ILi_rslqr(2),Gm_IL_rslqr(2))];
pm_sv_rslqr =
[min(pm_ILi_rslqr(1),pm_IL_rslqr(1)),max(pm_ILi_rslqr(2),pm_IL_rslqr(2))];
```

```
text = ['SPC SV Gain Margin:  [',num2str(Gm_sv_spc(1)),',
',num2str(Gm_sv_spc(2)),'] dB',...
        newline 'SPC SV Phase Margin: [',num2str(pm_sv_spc(1)),',
',num2str(pm_sv_spc(2)),'] degrees',...
        newline '5 State RSLQR SV Gain Margin:
[',num2str(Gm_sv_rslqr5(1)),', Inf] dB',...
        newline '5 State RSLQR SV Phase Margin:
[',num2str(pm_sv_rslqr5(1)),', ',num2str(pm_sv_rslqr5(2)),'] degrees',...
        newline 'HINF SV Gain Margin:  [',num2str(Gm_sv_hinf(1)),',
',num2str(Gm_sv_hinf(2)),'] dB',...
        newline 'HINF SV Phase Margin: [',num2str(pm_sv_hinf(1)),',
',num2str(pm_sv_hinf(2)),'] degrees',...
        newline 'RSLQR SV Gain Margin:  [',num2str(Gm_sv_rslqr(1)),',
',num2str(Gm_ILi_rslqr(2)),'] dB',...
        newline 'RSLQR SV Phase Margin: [',num2str(pm_sv_rslqr(1)),',
',num2str(pm_sv_rslqr(2)),'] degrees'];
disp(text);
```

```
SPC SV Gain Margin:  [-4.1586, 4.6934] dB
SPC SV Phase Margin: [-24.0956, 24.0956] degrees
5 State RSLQR SV Gain Margin:  [-8.4989, Inf] dB
5 State RSLQR SV Phase Margin: [-60, 60] degrees
HINF SV Gain Margin:  [-10.918, 27.4769] dB
HINF SV Phase Margin: [-57.2218, 57.2218] degrees
RSLQR SV Gain Margin:  [-6.5287, 3.6848] dB
RSLQR SV Phase Margin: [-32.4127, 32.4127] degrees
```

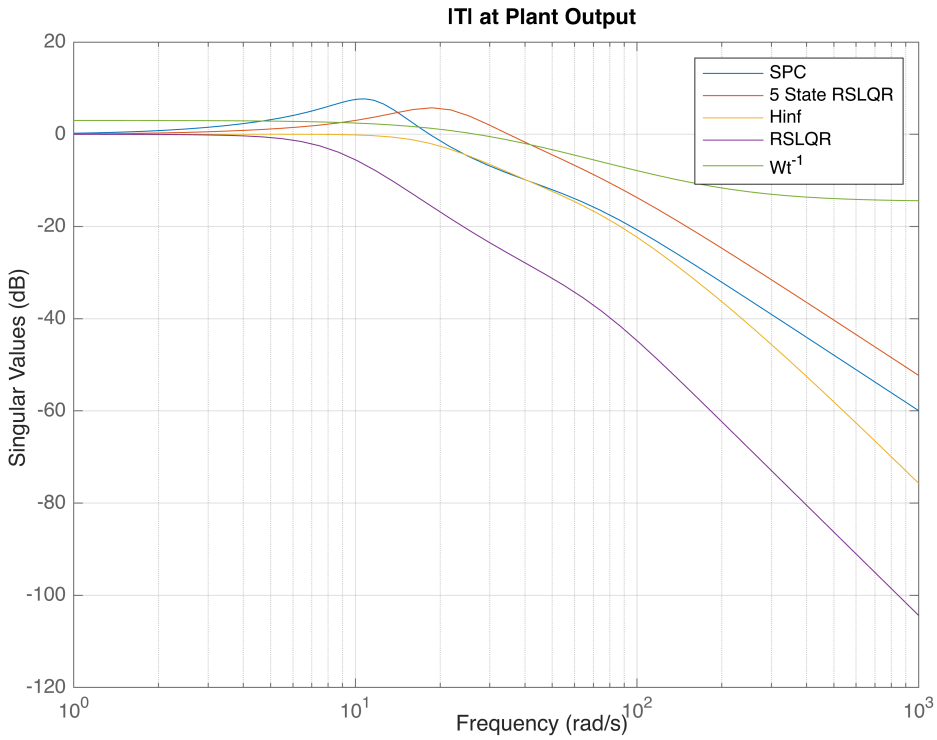
g) Compute the SISO sensitivity function $e/r = S$ (at the output) for the commandedvariable comparing each controller. Identify on the plot the $||S||_\infty$ for each controller. Plot the RSLQR $e/r = S$ including the actuator in the plant model with the Hinf $e/r = S$ (both plant models should have the actuator in them).

```
S_y_spc = 1/(1+sys_y_spc(1,1));
S_y_rslqr5 = 1/(1+sys_y_rslqr5(1,1));
S_y_hinf = 1/(1+sys_y_hinf(1,1));
S_y_rslqr = 1/(1+sys_y_rslqr(1,1));

sigma(S_y_spc),
hold on;
sigma(S_y_rslqr5),
sigma(S_y_hinf),
sigma(S_y_rslqr),
sigma(inv(Ws.sys))
grid on, title('|S| at Plant Output')
legend('SPC', '5 State RSLQR','Hinf', 'RSLQR', 'Ws^{-1}')
hold off
```



h) Compute the SISO complementary sensitivity $y/r = T$ (at the output) for the commanded variable comparing each controller. Identify on the plot the $||S||_\infty$ for each controller. Plot the RSLQR $A_z/A_{zc} = T$ including the actuator in the plant model with the Hinf $A_z/A_{zc} = T$ (both plant models should have the actuator in them). Plot your Weighting filter WT with these two frequency responses.

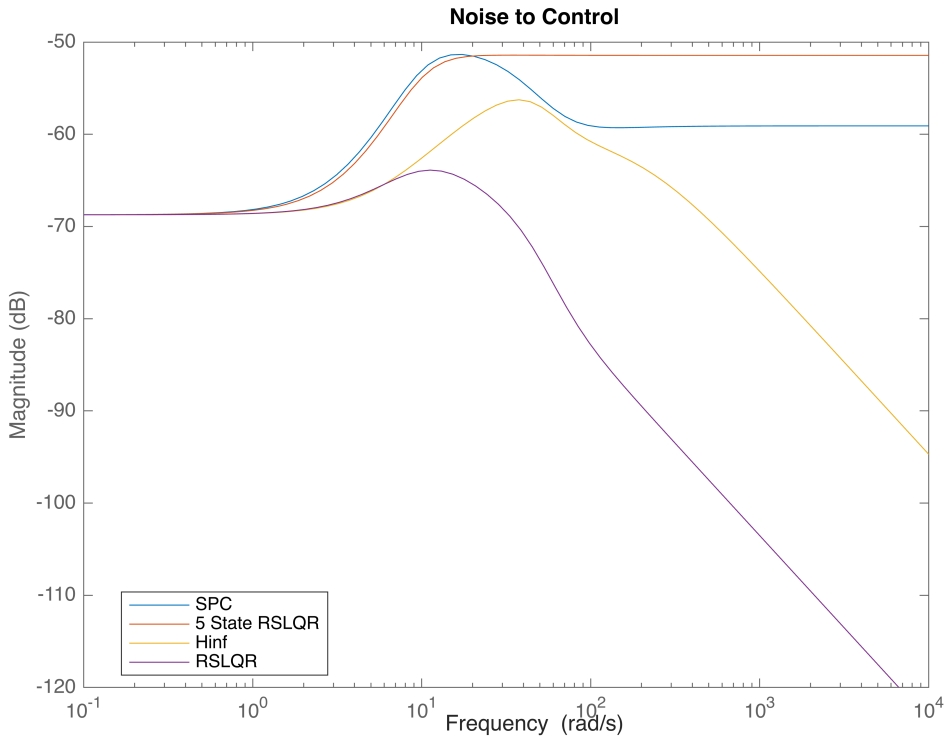
```
T_y_spc = sys_y_spc(1,1)/(1+sys_y_spc(1,1));
T_y_rslqr5 = sys_y_rslqr5(1,1)/(1+sys_y_rslqr5(1,1));
T_y_hinf = sys_y_hinf(1,1)/(1+sys_y_hinf(1,1));
T_y_rslqr = sys_y_rslqr(1,1)/(1+sys_y_rslqr(1,1));

sigma(T_y_spc),
hold on;
sigma(T_y_rslqr5),
sigma(T_y_hinf),
sigma(T_y_rslqr),
sigma(inv(Wt.sys))
legend('SPC', '5 State RSLQR','Hinf', 'RSLQR', 'Wt^{-1}')
grid on, title('|T| at Plant Output')
xlim([1 1000]), hold off
```



i) Plot the SISO Bode noise-to-control and noise-to-control rate frequency responses for each controller.

```
% Compute the noise-to-control TF
Cp = [ 1  0  0  0;
       0  1  0  0];
Dp = [0;
      0];

SPC.Bv = [ Bp*Zi*SPC.Dc1;
        (SPC.Bc1+SPC.Bc1*Dp*Zi*SPC.Dc1)];
SPC.Cv = [ Zi*SPC.Dc1*Cp Zi*SPC.Cc];
SPC.Cvv = [ SPC.Cv ; SPC.Cv*SPC.Acl ];
SPC.Dv = Zi*SPC.Dc1;
SPC.Dvv = [ SPC.Dv; SPC.Cv*SPC.Bv];
sys_noise_spc = ss(SPC.Acl,SPC.Bv,SPC.Cvv,SPC.Dvv);

Cp = [ 1  0  0  0;
       0  1  0  0;
       0  0  1  0;
       0  0  0  1];
Dp = [0;
      0;
      0;
      0];
```

```
% Compute the noise-to-control TF
rslqr5.Bv = [ Bp*Zi*rslqr5.Dc1;
        (rslqr5.Bc1+rslqr5.Bc1*Dp*Zi*rslqr5.Dc1)];
rslqr5.Cv = [ Zi*rslqr5.Dc1*Cp Zi*rslqr5.Cc];
rslqr5.Cvv = [ rslqr5.Cv ; rslqr5.Cv*rslqr5.Acl ];
rslqr5.Dv = Zi*rslqr5.Dc1;
rslqr5.Dvv = [ rslqr5.Dv; rslqr5.Cv*rslqr5.Bv];
sys_noise_rslqr5 = ss(rslqr5.Acl,rslqr5.Bv,rslqr5.Cvv,rslqr5.Dvv);


% Compute the noise-to-control TF
sys_noise_rslqr = ss(rslqr.Acl,rslqr.Bv,rslqr.Cvv,rslqr.Dvv);
sys_noise_hinf = ss(Hinf.Acl,Hinf.Bv,Hinf.Cvv,Hinf.Dvv);

figure;
bodemag(sys_noise_spc(1,1))
hold on;
bodemag(sys_noise_rslqr5(1,1))
bodemag(sys_noise_hinf(1,1))
bodemag(sys_noise_rslqr(1,1))
legend('SPC', '5 State RSLQR','Hinf', 'RSLQR','Location','best')
title('Noise to Control')
hold off
```




```
figure;
bodemag(sys_noise_spc(2,1))
```

```
hold on;
bodemag(sys_noise_rslqr5(2,1))
bodemag(sys_noise_hinf(2,1))
bodemag(sys_noise_rslqr(2,1))
legend('SPC', '5 State RSLQR','Hinf', 'RSLQR','Location','best')
title('Noise to Control Rate')
hold off
```