1: 20 points

Model Order Reduction Problems

(a) Let

$$G(s) = \sum_{i=1}^{5} \frac{\alpha^{2i}}{s + \alpha^{2i}}.$$

Find a balanced realization for each of the following α using Matlab: $\alpha = 2,\ 4,\ 20,\ 50$. Discuss the behavior of the Hankel singular values as $\alpha \to \infty$.

(b) Note that a time delay can be approximated as

$$e^{-\tau s} \approx \left(\frac{1 - \frac{\tau}{2n}s}{1 + \frac{\tau}{2n}s} \right)^n$$

for a sufficiently large n. Let a process model $\frac{e^{-s}}{1+Ts}$ be approximated by

$$G(s) = \left(\frac{1 - 0.05s}{1 + 0.05s} \right)^{10} \frac{1}{1 + sT}.$$

For each $T = 0,\ 0.01,\ 0.1,\ 1,\ 10$, find the smallest order model using balanced truncation such that the approximation error is no greater than 0.1. Use Matlab to solve.

2: 30 points

LMI Problems

(a) Consider the system

$$A = \begin{bmatrix} -5 & 1 & 2 \\ 1 & -9 & 1 \\ -1 & -10 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Compute the H_2 norm of the system using LMIs in Matlab.

(c) Consider the system

$$A = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Determine whether this system is stabilizable and detectable using LMIs in Matlab.

(d) Consider the system of form

$$\dot{x} = Ax + B_1 u + B_2 w$$
$$z = Cx + Du$$

with

$$A = \begin{bmatrix} -5 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.05 \\ 0 \\ 0.03 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 3 \\ 1 & 0 \end{bmatrix}.$$

Design a state feedback control law $u = Kx$ that solves the H_2 optimal control problem using LMIs in Matlab.

4: 30 points

Design Problem: LMI Synthesis

Consider the distillation column with plant model

$$G(s) = \frac{1}{75s+1} \begin{bmatrix} 87.8 & -86.4 \\ 108.2 & -109.6 \end{bmatrix}.$$

Assuming multiplicative input uncertainty of the form $W_I(s) = \frac{10s+10}{s+100} I_{2\times 2}$ and performance weight $W_P(s) = \frac{0.5s+10}{s+0.1} I_{2\times 2}$, design a 1.) an H_∞ optimal controller, 2.) a μ-synthesis controller, and 3.) a mixed H_2 / H_∞ controller that minimizes the H_2 norm such that robust stability is maintained. Plot the uncertain sensitivity functions that are obtained for all three design approaches. Which approach works "best"?

1a

```
s = tf("s");
alpha = 2;
G = 0;
for i = 1:5
    G = G + (alpha^(2*i))/(s+alpha^(2*i));
end
[sysb,g] = balreal(G)
```

```
sysb =

  A = 
            x1      x2      x3      x4      x5
   x1   -219.5   262.5  -178.9  -92.16   36.04
   x2    262.5  -381.4   315.6   180.4  -73.27
   x3   -178.9   315.6  -352.6  -257.6   117.3
   x4   -92.16   180.4  -257.6  -268.6   160.5
   x5    36.04  -73.27   117.3   160.5    -142

  B = 
            u1
   x1   -26.27
   x2    22.12
   x3   -12.13
   x4    -5.73
   x5     2.18

  C = 
            x1      x2      x3      x4      x5
   y1   -26.27   22.12  -12.13   -5.73    2.18

  D = 
        u1
   y1    0

Continuous-time state-space model.
 g = 5x1
      1.5721
      0.6413
      0.2087
      0.0611
      0.0167
```

```
alpha = 4;
G = 0;
for i = 1:5
    G = G + (alpha^(2*i))/(s+alpha^(2*i));
end
[sysb,g] = balreal(G)
```

```
sysb =

  A = 
                x1          x2          x3          x4          x5
   x1   -1.271e+05  -1.955e+05   1.917e+05   1.411e+05  -7.235e+04
   x2   -1.955e+05  -3.109e+05   3.183e+05   2.436e+05  -1.281e+05
   x3    1.917e+05   3.183e+05  -3.465e+05   -2.82e+05    1.55e+05
   x4    1.411e+05   2.436e+05   -2.82e+05  -2.462e+05   1.432e+05
   x5   -7.235e+04  -1.281e+05    1.55e+05   1.432e+05  -8.766e+04

  B = 
            u1
   x1    502.1
   x2    654.4
   x3   -538.5
   x4   -347.6
   x5    165.4

  C = 
            x1      x2      x3      x4      x5
   y1    502.1   654.4  -538.5  -347.6   165.4

  D = 
        u1
   y1    0

Continuous-time state-space model.
 g = 5x1
      0.9917
      0.6885
      0.4185
      0.2453
      0.1560
```

```
alpha = 20;
G = 0;
for i = 1:5
    G = G + (alpha^(2*i))/(s+alpha^(2*i));
end
[sysb,g] = balreal(G)
```

```
sysb =

  A = 
                x1          x2          x3          x4          x5
   x1  -8.977e+11   1.536e+12  -1.744e+12   1.485e+12   8.471e+11
   x2   1.536e+12   -2.63e+12   2.992e+12  -2.553e+12  -1.458e+12
   x3  -1.744e+12   2.992e+12  -3.411e+12   2.918e+12    1.67e+12
   x4   1.485e+12  -2.553e+12   2.918e+12  -2.502e+12  -1.435e+12
   x5   8.471e+11  -1.458e+12    1.67e+12  -1.435e+12  -8.237e+11

  B = 
                u1
   x1  -1.029e+06
   x2   1.699e+06
   x3  -1.841e+06
   x4   1.499e+06
   x5   8.283e+05

  C = 
                x1          x2          x3          x4          x5
   y1  -1.029e+06   1.699e+06  -1.841e+06   1.499e+06   8.283e+05

  D = 
       u1
   y1   0

Continuous-time state-space model.
 g = 5x1
      0.5894
      0.5485
      0.4967
      0.4490
      0.4165
```

```
alpha = 50;
G = 0;
for i = 1:5
    G = G + (alpha^(2*i))/(s+alpha^(2*i));
end
[sysb,g] = balreal(G)
```

```
sysb =

  A = 
                x1          x2          x3          x4          x5
   x1   -8.29e+15    1.43e+16  -1.642e+16   1.414e+16   8.128e+15
   x2    1.43e+16  -2.467e+16   2.833e+16   -2.44e+16  -1.403e+16
   x3  -1.642e+16   2.833e+16  -3.255e+16   2.805e+16   1.614e+16
   x4   1.414e+16   -2.44e+16   2.805e+16  -2.418e+16  -1.391e+16
   x5   8.128e+15  -1.403e+16   1.614e+16  -1.391e+16  -8.008e+15

  B = 
                u1
   x1   9.419e+07
   x2  -1.601e+08
   x3   1.803e+08
   x4  -1.523e+08
   x5  -8.638e+07

  C = 
                x1          x2          x3          x4          x5
   y1   9.419e+07  -1.601e+08   1.803e+08  -1.523e+08  -8.638e+07

  D = 
       u1
   y1   0

Continuous-time state-space model.
 g = 5x1
      0.5351
      0.5198
      0.4995
      0.4798
      0.4658
```

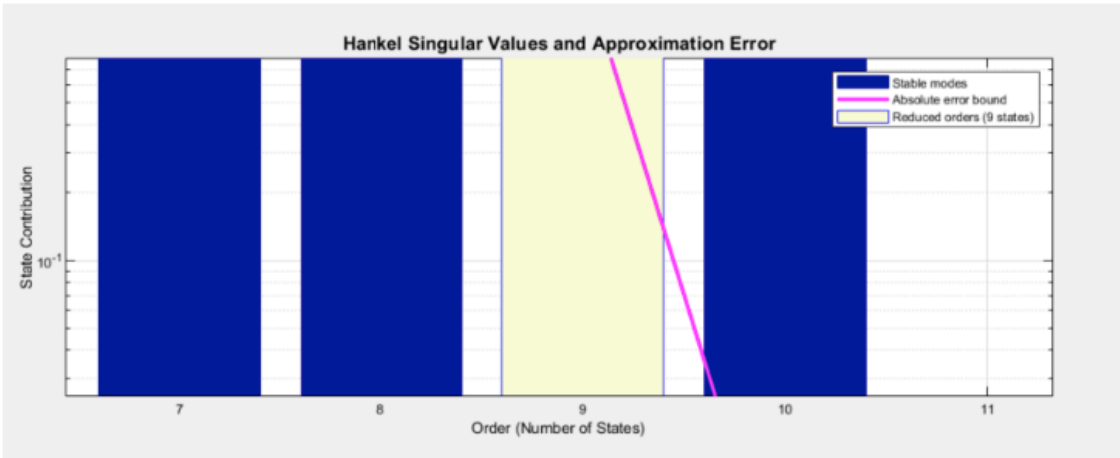
The Hankel singular values will approach to 0.5 when alpha tends to infinity.

1b

```
s = tf("s");
T = 0;
G_td = ((1-0.05*s)/(1+0.05*s))^(10) * (1/(1+s*T));
Gr = balancmr(G_td,'MaxError',0.1);
size(Gr)
```

```
State-space model with 1 outputs, 1 inputs, and 10 states.
```

```
%modelReducer(G_td)
```

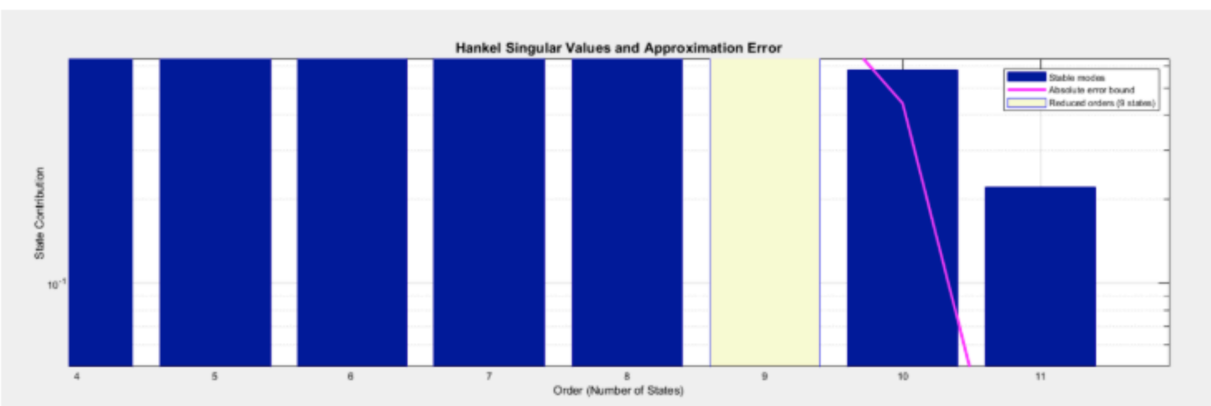


Smallest order model for T = 0 is 10.

```
s = tf("s");
T = 0.01;
G_td = ((1-0.05*s)/(1+0.05*s))^(10) * (1/(1+s*T));
Gr = balancmr(G_td,'MaxError',0.1);
size(Gr)
```

```
State-space model with 1 outputs, 1 inputs, and 11 states.
```

```
%modelReducer(G_td)
```

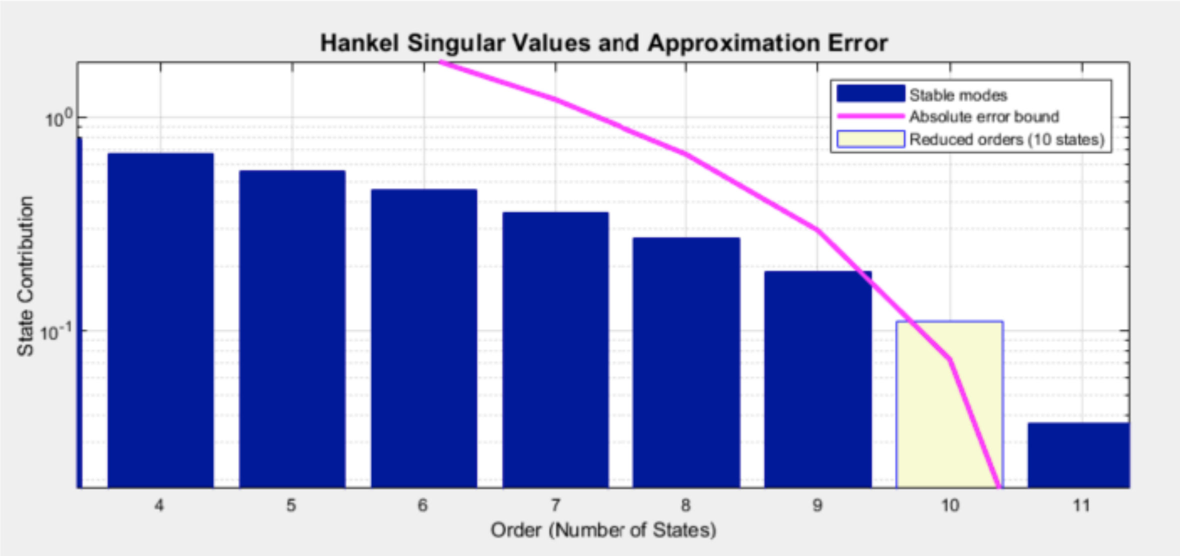


Smallest order model for T = 0.01 is 11.

```
s = tf("s");
T = 0.1;
G_td = ((1-0.05*s)/(1+0.05*s))^(10) * (1/(1+s*T));
Gr = balancmr(G_td,'MaxError',0.1);
size(Gr)
```

```
State-space model with 1 outputs, 1 inputs, and 10 states.
```

```
%modelReducer(G_td)
```

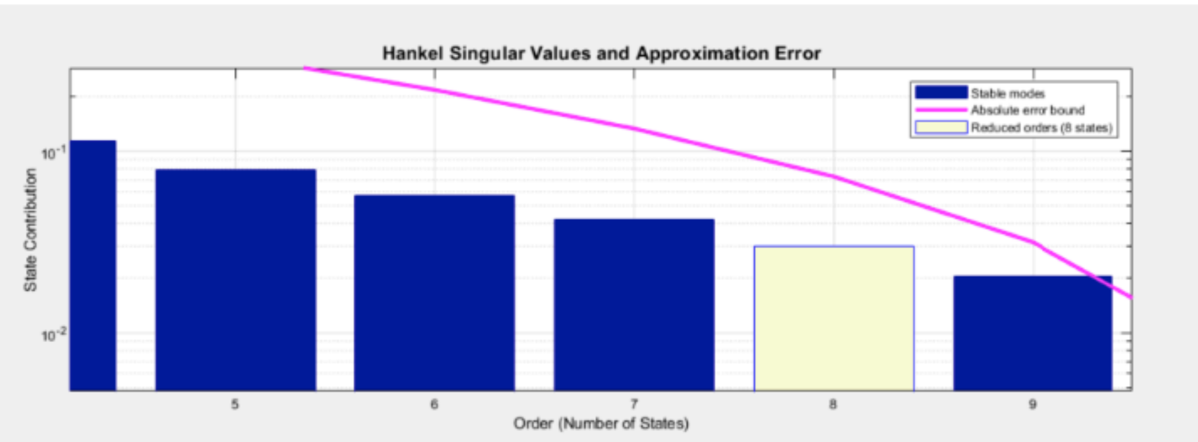


Smallest order model for T = 0.1 is 10.

```
s = tf("s");
T = 1;
G_td = ((1-0.05*s)/(1+0.05*s))^(10) * (1/(1+s*T));
Gr = balancmr(G_td,'MaxError',0.1);
size(Gr)
```

```
State-space model with 1 outputs, 1 inputs, and 8 states.
```

```
%modelReducer(G_td)
```

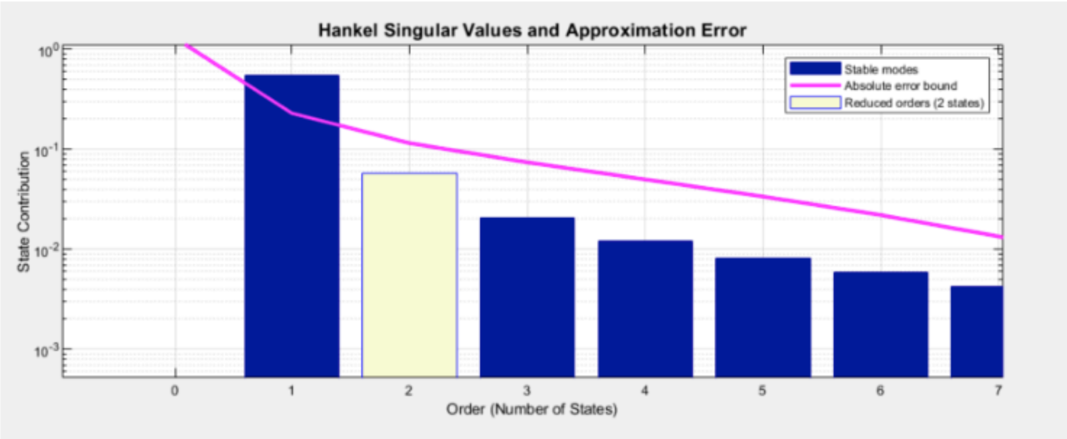


Smallest order model for T = 1 is 8.

```
s = tf("s");
T = 10;
G_td = ((1-0.05*s)/(1+0.05*s))^(10) * (1/(1+s*T));
Gr = balancmr(G_td,'MaxError',0.1);
size(Gr)
```

```
State-space model with 1 outputs, 1 inputs, and 3 states.
```

```
%modelReducer(G_td)
```



Smallest order model for T = 10 is 3.

2a

```
A = [-5 1 2; 1 -9 1; -1 -10 -3];
B = [0 1 0]';
C = [1 0 0; 0 0 1];
D = [0;0];
G = ss(A,B,C,D);
h2norm = h2norm_lmi_yalmip(G)
```

```
     4              0.115743
***                      new lower bound:      0.103352
     5              0.113392
***                      new lower bound:      0.107265
     6              0.112832
***                      new lower bound:      0.109341
     7              0.112359
***                      new lower bound:      0.110516
     8              0.112016
***                      new lower bound:      0.111166
     9              0.111908
***                      new lower bound:      0.111773
    10              0.111908
***                      new lower bound:      0.111860

 Result:  feasible solution of required accuracy
          best objective value:     0.111908
          guaranteed absolute accuracy:  4.84e-05
          f-radius saturation:  0.000% of R =  1.00e+09
h2norm = 0.3345
```

2b

```
A = [0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0;-2 2;0 1];
C = [1 0 0;0 0 1];
D = [0 0;0 0];
stabilizable = lmiisstabilizable_yalmip(ss(A,B,C,D))
```

```
 Solver for LMI feasibility problems L(x) < R(x)
    This solver minimizes  t  subject to  L(x) < R(x) + t*I
    The best value of t should be negative for feasibility

 Iteration   :    Best value of t so far

     1                         0.024819
     2                        -0.019360

 Result:  best value of t:    -0.019360
          f-radius saturation:  0.000% of R =  1.00e+09

################################################################
You are using LMILAB. Please don't use LMILAB with YALMIP
https://yalmip.github.io/solver/lmilab/

Install a better SDP solver
https://yalmip.github.io/allsolvers/

To get rid of this message, edit calllmilab.m
(but don't expect support when things do not work,
YALMIP + LMILAB => No support)
################################################################
stabilizable = 1
```

```
function stabilizable = lmiisstabilizable_yalmip(sys)
    [A,B,C,D] = ssdata(sys);
    gamma=sdpvar(1); % symbolic decision variables
    P = sdpvar(size(A,1));
%     define the constraints.
%  NOTE:  YALMIP does not actually handle strict inequalities.  We need to
%  SLIGHTLY modify the problem to get a proper solution
    eps = 1e-10;
    F = [P >= eps*eye(size(P)), A*P+P*A'<=gamma*B*B', gamma>=eps];
%   Solve the minimization problem.  For feasibility, we do not need to
%   provide an objective function.
    optimize(F);
    %Check to see if constraints were violated; a negative number implies
    %that a constraint was not satisfied.
    isStable = check(F);
    if(min(isStable) < 0)
        stabilizable = 0;
    else
        stabilizable = 1;
    end
end
```

```
detectable = lmiisdetectable_yalmip(ss(A,B,C,D))
```

```
 Solver for LMI feasibility problems L(x) < R(x)
    This solver minimizes  t  subject to  L(x) < R(x) + t*I
    The best value of t should be negative for feasibility

 Iteration   :    Best value of t so far

     1                        -0.607322

 Result:  best value of t:    -0.607322
          f-radius saturation:  0.000% of R =  1.00e+09

################################################################
You are using LMILAB. Please don't use LMILAB with YALMIP
https://yalmip.github.io/solver/lmilab/

Install a better SDP solver
https://yalmip.github.io/allsolvers/

To get rid of this message, edit calllmilab.m
(but don't expect support when things do not work,
YALMIP + LMILAB => No support)
################################################################
detectable = 1
```

```
function detectable = lmiisdetectable_yalmip(sys)
    [A,B,C,D] = ssdata(sys);
    w = sdpvar(size(B,2),size(A,1),'full'); % symbolic decision variables
    P = sdpvar(size(A,1));
%     define the constraints.
%  NOTE:  YALMIP does not actually handle strict inequalities.  We need to
%  SLIGHTLY modify the problem to get a proper solution
    eps = 1e-10;
    F = [P >= eps*eye(size(P)), A'*P+P*A+w'*C+C'*w<=eps];
%   Solve the minimization problem.  For feasibility, we do not need to
%   provide an objective function.
    optimize(F);
    %Check to see if constraints were violated; a negative number implies
    %that a constraint was not satisfied.
    isStable = check(F);
    if(min(isStable) < 0)
        detectable = 0;
    else
        detectable = 1;
    end
end
```

2c

```
A = [-5 1 0;0 1 1;1 1 1];
B1 = [0 0;0 1;1 0];
B2 = [0.05 0 0.03]';
C = [1 0 0;0 2 1];
D = [1 3;1 0];
[K, ~] = h2sf(A, B1, B2, C, D);
```

```
    23              6.149579e-03
    24              6.149579e-03
    25              6.149579e-03
    26              6.149579e-03
    27              6.149579e-03
    28              6.149579e-03
***                new lower bound: 6.148980e-03

 Result:  feasible solution of required accuracy
          best objective value: 6.149579e-03
          guaranteed absolute accuracy:  5.99e-07
          f-radius saturation:  3.392% of R =  1.00e+09

################################################################
You are using LMILAB. Please don't use LMILAB with YALMIP
https://yalmip.github.io/solver/lmilab/

Install a better SDP solver
https://yalmip.github.io/allsolvers/
```

```
K
```

```
 K = 2x3
     -0.3606   -4.8947   -4.3199
     -0.2517    0.8280    0.8501
```

```
function [K,gamma] = h2sf(A, B1, B2, C, D)
    z = sdpvar(size(C*A, 1));
    w = sdpvar(size(B1,2),size(A,1),'full'); % symbolic decision variables
    X = sdpvar(size(A,1));
    gamma = sdpvar(1);
%     define the constraints.
%  NOTE:  YALMIP does not actually handle strict inequalities.  We need to
%  SLIGHTLY modify the problem to get a proper solution
    eps = 1e-10;
    con1 = A*X+B1*w+(A*X+B1*w)'+B2*B2';
    con2 = [-z C*X+D*w; (C*X+D*w)' -X];
    F = [con1 <= eps, con2<=eps, trace(z)<=gamma];
%   Solve the minimization problem.  For feasibility, we do not need to
%   provide an objective function.
    optimize(F,gamma);
    gamma = sqrt(value(gamma));
    K = value(w) * inv(value(X));
end
```

3. H_∞ Controller

```
s = tf("s");
G = (1/(75*s+1))*[87.8 -86.4;108.2 -109.6];
WI = (10*s+10)/(s+100)*eye(2);
Wp = (0.5*s+10)/(s+0.1);
Wp = (Wp/(s/1000+1))*eye(2);

systemnames = 'G WI Wp';
inputvar = '[du{2};r{2};u{2}]';
outputvar = '[WI;Wp;r-G]';  %For h2hinfsyn, we need the uncertainties (H_inf outputs) at the top
input_to_G = '[u+du]';
input_to_Wp = '[r-G]';
input_to_WI = '[u]';
cleanupsysic = 'yes';  %This drops all the useless variables from workspace
P = sysic;
```

1) H_∞ Controller synthesis

```
[Kinf,CL,GAM] = hinfsyn(P,2,2);%This is for a sanity check
Kinf
```

Mu controller

```
delta1 = ultidyn('delta1',[1,1]);
delta2 = ultidyn('delta2',[1,1]);
Ghat = G*(eye(2)+[delta1 0;0 delta2]*WI); %input multiplicative structured uncertainty
systemnames = 'Ghat Wp';
inputvar = '[r{2};u{2}]';
outputvar = '[Wp;Ghat;r-Ghat]';  %For h2hinfsyn, we need the uncertainties (H_inf outputs) at the top
input_to_Ghat = '[u]';
input_to_Wp = '[r-Ghat]';
cleanupsysic = 'yes';  %This drops all the useless variables from workspace
Pmu = sysic;
[Kmu,CLP,mu_mu] = musyn(Pmu,2,2);
```

```
D-K ITERATION SUMMARY:
-----------------------------------------------------------------
                       Robust performance              Fit order
-----------------------------------------------------------------
  Iter         K Step       Peak MU       D Fit             D
    1           10.84        10.84        10.94            20
    2           7.997        7.997            8            12
    3           6.091        6.091        6.104            12
    4           5.231        5.231         5.24            12
    5           4.839        4.839        4.844            12
    6           4.632        4.632        4.647            12
    7            4.51        4.511        4.531            12
    8           4.434        4.434        4.457            16
    9            4.38         4.38        4.404            16
   10            4.34        4.342        4.365            16

Best achieved robust performance: 4.34
```

```
Kmu
```

3) $\mathrm{H2}/H\infty$ Controller synthesis

```
[K2inf,CL,normz,info] = h2hinfsyn(prescale(ss(P)),2,2,2,[0,1],'HINFMAX',1);
K2inf
```

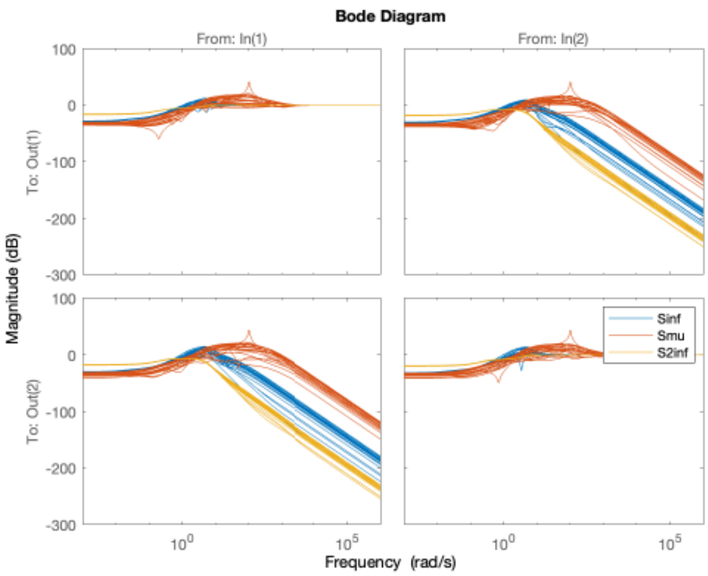
K2inf =

A =

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13
x1	−18.96	5.836	−0.1167	0.1204	0.06514	0.02329	−0.001168	0.0003553	0.007327	−0.02632	−0.03396	0.01921	−0.003615
x2	5.788	−31.96	−0.1372	−0.03984	−0.001209	−0.01292	0.008306	0.0003096	−0.001547	−0.08594	−0.1186	−0.008592	−0.01128
x3	−0.003888	−0.002563	−999.2	0.1845	−0.06666	0.147	4.137	17.89	−2.488	1.15	1.324	7.169	−20.17
x4	−0.002278	0.00856	0.1565	−999.5	0.263	0.2531	2.053	9.802	3.427	0.6797	0.8329	−21.08	−6.523
x5	−0.0004428	0.003581	0.1272	0.2246	−999.9	0.2169	−6.467	5.28	1.545	0.4386	0.5075	−9.195	−6.478
x6	−0.002125	0.004189	0.006813	0.1346	0.1087	−1000	7.537	3.029	0.9831	−0.01855	0.02242	−6.214	2.914
x7	−0.006217	0.04827	−4.353	−1.21	0.1349	−8.684	0.01522	0.1914	−5.406e−05	1.656	4.268	0.05683	0.2596
x8	0.008371	0.0006927	18.65	9.315	3.095	5.91	−0.2195	−0.8237	0.01335	−13.09	−14.63	−0.3384	−1.121
x9	0.1394	−0.03858	−3.498	−5.135	19.56	11.55	0.0712	−0.003861	−0.1315	0.8124	0.9825	0.1786	0.09186
x10	−0.002259	0.002047	0.7845	0.5185	0.07538	0.4438	−0.684	−12.38	0.6782	−995.7	5.192	−15.11	−59.31
x11	−0.003207	0.003284	0.7525	0.5708	0.0931	0.4706	−6.312	−12.75	0.1295	4.25	−995	−18.11	−53.07
x12	0.03772	−0.0135	7.149	−18.38	−8.269	−5.951	−0.1313	−0.3156	0.09638	−14.77	−19.08	−1.144	−1.923
x13	0.008183	0.01351	−11.6	−3.395	1.908	−6.989	−0.3426	−1.312	0.03539	−56.62	−66.9	−2.106	−7.276
x14	0.003576	0.09279	10.94	3.831	2.168	−0.3125	0.1239	0.3134	−0.006368	20.41	23.4	0.7584	2.704

Uncertain Sensitivity Plot

```
bodemag(inv(eye(2)+Ghat*Kinf),inv(eye(2)+Ghat*Kmu),inv(eye(2)+Ghat*K2inf));
legend("Sinf","Smu","S2inf");
```



Mu-synthesis controller has best performance