1: 20 points

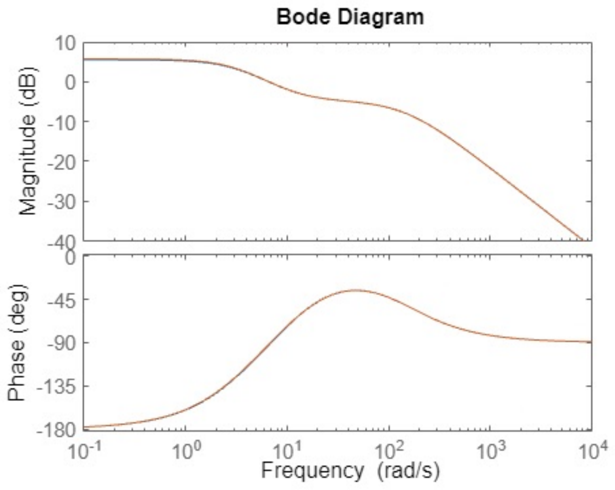
Write your own function to solve the simplified H_∞ optimal control problem discussed in class (and Zhou 14.2) that takes as inputs the generalized plant P, the number of control inputs, and the number of measurements (the same info that *hinfsyn* uses). To accomplish this you will need to (1) Partition the generalized plant to extract B_1, B_2, etc.; (2) Iterate on γ while checking the necessary conditions for controller existence (2 AREs and a spectral radius condition); and (3) For the final γ, construct the controller. Test the performance on the following SISO generalized plant - compare your results with *hinfsyn*. NOTE: The matrices below represent the state space realization of the generalized plant P.

$$A = \begin{bmatrix} -0.2 & 2 & 2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -10 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 1 & 0 \\ 40 & 20 & -20 \\ 60 & 30 & -30 \\ -2 & -1 & -3 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 3 & 0 & 0 & -3 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 & 0.9487 \\ 0 & 0 & -0.3162 \\ -0.4472 & 0.8944 & 0 \end{bmatrix}$$

```
%Problem 1
A = [-0.2 2 2 0;
    0 -1 0 0;
    0 0 -2 0;
    0 0 0 -10];
B = [2 1 0;
    40 20 -20;
    60 30 -30;
    -2 -1 -3];
C = [1 0 0 -1;
    3 0 0 -3;
    0 1 1 0];
D = [0 0 0.9487;
    0 0 -0.3162;
    -0.4472 0.8944 0];
P = ss(A, B, C, D);
K_custom = custom_hinf(P, 1, 1);
K_actual = hinfsyn(P,1,1);
bode(K_custom, K_actual)
```

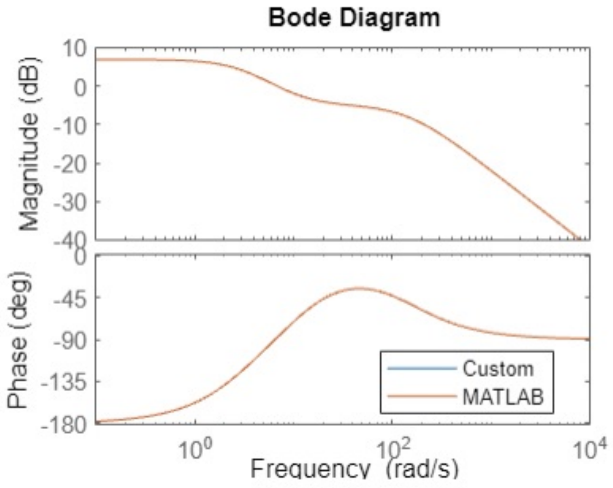
```
function K = custom_hinf(G, meas, cont)
    A = G.A;
    B = G.B;
    C = G.C;
    D = G.D;
    B_1 = B(:, 1:size(B,2) - cont);
    B_2 = B(:, size(B,2) - cont + 1:end);
    C_1 = C(1:size(C,1) - meas, :);
    C_2 = C(size(C,1) - meas+1:end, :);
    u_l = 1e5;
    l_l = max([1e-5, norm(D)]);
    g_new = mean([u_l, l_l]);
    g_try = u_l;
    while(abs(g_new - g_try) > .0001)
       g_try = g_new;
       try
           X = are(A,B_2*B_2'-B_1*B_1'/g_try^2,C_1'*C_1);
           Y = are(A',C_2'*C_2-C_1'*C_1/g_try^2,B_1*B_1');
           rho_svd = abs(eig(X * Y));
           rho = max(rho_svd) - g_try^2;
           if (rho < 0)
               u_l = g_try;
           else
               l_l = g_try;
           end
       catch
           l_l = g_try;
       end
        g_new = 1/2 * (l_l + u_l);
    end
    F = -B_2' * X;
    L = -Y * C_2';
    Z = inv(eye(size(X)) - g_try^(-2) * Y * X);
    A_new = A + g_try^(-2) * B_1 * B_1' * X + B_2 * F + Z * L * C_2;
    K = ss(A_new, -Z * L, F, 0);
end
```



2: 10 points

Write your own function to solve the simplified H_2 optimal control problem (same assumptions as Problem 1) with the same inputs as Problem 1. You will already have the partitioning down, and will now just need to solve 2 AREs and construct the controller. Test the performance on the same example as the H_∞ case and compare against the results from *h2syn*.

```
%Problem 2
K_custom = custom_h2(P, 1, 1);
K_actual = h2syn(P,1,1);
bode(K_custom, K_actual)
legend('Custom', 'MATLAB', Location='best')
```



```
function K = custom_h2(G, meas, cont)
    A = G.A;
    B = G.B;
    C = G.C;
    D = G.D;

    B_1 = B(:, 1:size(B, 2) - cont);
    B_2 = B(:, size(B, 2) - cont + 1:end);
    C_1 = C(1:size(C, 1) - meas, :);
    C_2 = C(size(C, 1) - meas + 1:end, :);

    X = are(A, B_2 * B_2', C_1' * C_1);
    Y = are(A', C_2' * C_2, B_1 * B_1');

    F = -B_2' * X;
    L = -Y * C_2';
    K = ss(A + B_2 * F + L* C_2, -L, F, 0);
end
```

3: 30 points

Design Problem: H_∞ Loop Shaping

Control of flexible structures can be modeled by control of series mass spring damper systems (will see a 2nd example of this in the lab where we track references on the 3rd mass). The model *MSD.slsx* provides a triple mass spring damper system, with the center mass position the output, a force on the first mass the control input, and a disturbance input acting on the third mass. One state space model for this system is given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -20 & -0.2 & 20 & 0.02 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 20 & 0.2 & -120 & -0.12 & 40 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 100 & 0.1 & -44 & -0.44 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0.2 & 0 & -0.2 & 0 & 0 \end{bmatrix}^T$$

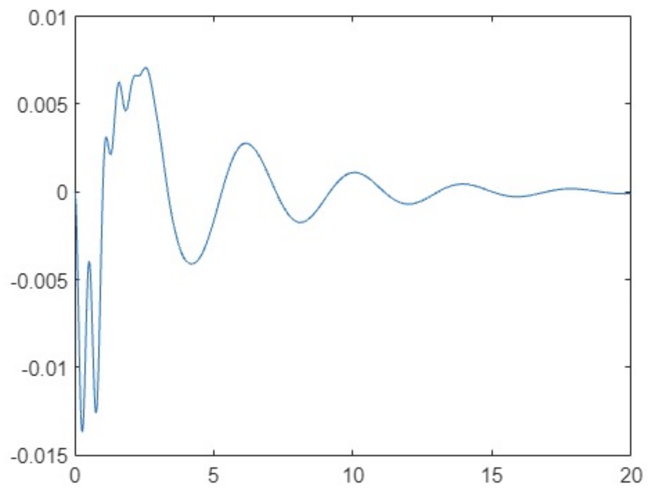
$$C = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

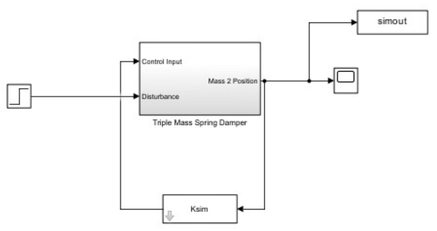
$$D = 0.$$

For this system, do the following:

(a) Design an LQG controller for the model that you found, with weights of your choice such that the position of mass 2 is highly penalized relative to the other states and noise inputs (HINT: check the C matrix to see which states represent mass 2). Simulate the controller performance with the disturbance a unit step input.

```
%Problem 3a
A = [0 1 0 0 0 0;-20 -0.2 20 0.02 0 0;0 0 0 1 0 0;20 0.2 -120 -0.12 40 0.4;0 0 0 0 0 1;0 0 100 0.1 -44 -0.44];
B = [0 0.2 0 -0.2 0 0]';
C = [1 0 1 0 0 0];
D = 0;
G = ss(A,B,C,D);

QXU = blkdiag(diag([1e2 1e-6 1e2 1e-6 1e-6 1e-6]), 1e-3);
QWV = blkdiag(1e-1*eye(6), 1e-3);

K_lqg = lqg(G, QXU, QWV);

m1 = 5;
k1 = 100;
c1 = 1;

m2 = 1;
k2 = 100;
c2 = 0.1;

m3 = 10;
k3 = 40;
c3 = 0.4;

K_sim = K_lqg;
sim('MSD_15a');
plot(simout.Time, simout.Data)
```

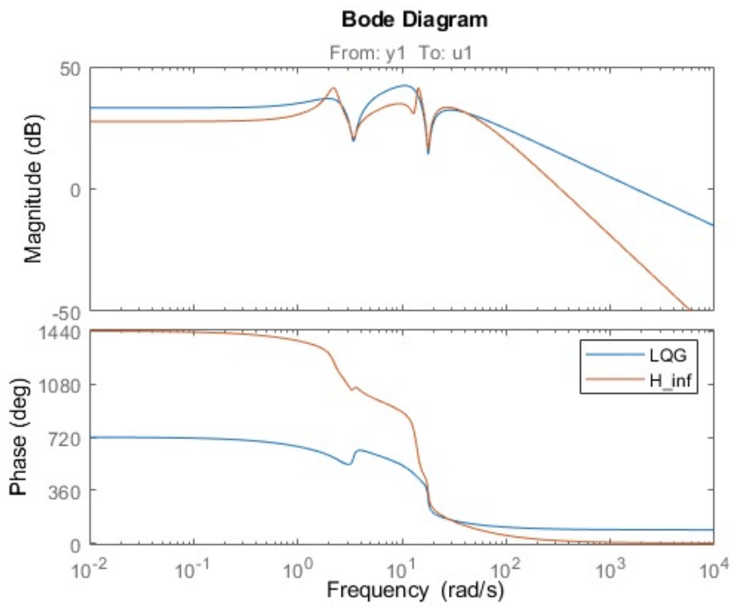




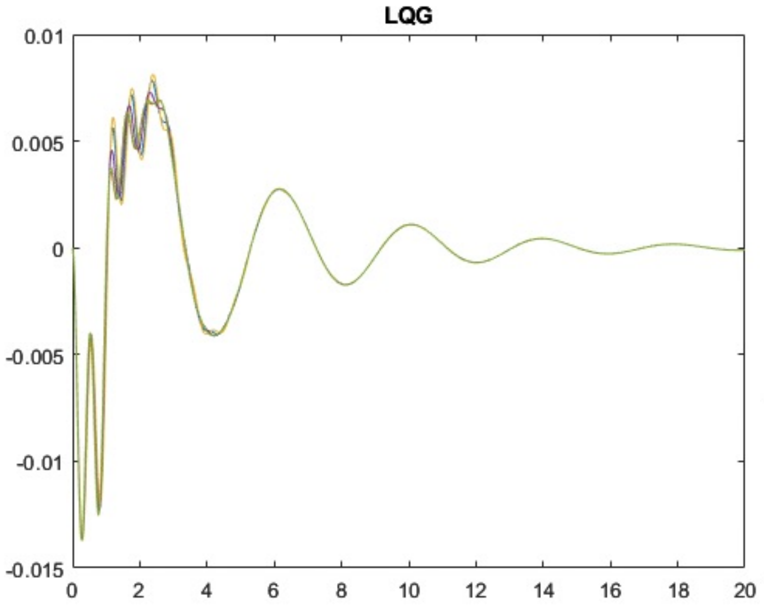
(b) Design an H_∞ loop shaping controller based on your LQG design from the last part. Simulate the peformance of the system as in the LQG design. What amount of normalized coprime factor uncertainty are you able to tolerate? If necessary, retune your LQG design to achieve $\gamma < 3$. Generate a Bode plot of the LQG controller and H_∞ loop shaping controller on the same graph.

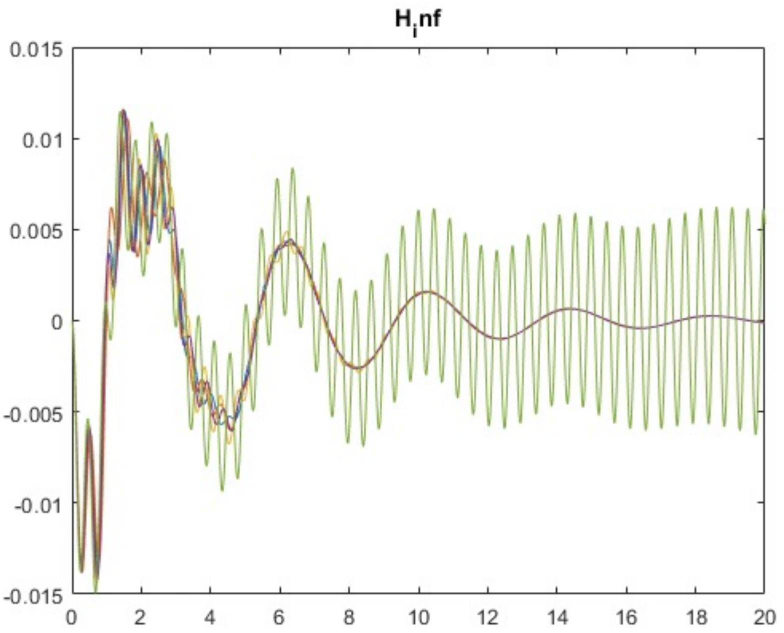
```
%Problem 3b
[K_inf, ~, GAM, ~] = ncfsyn(G, K_sim);
GAM
K_sim = K_inf;
sim('MSD_15a');
bode(K_lqg, K_inf)
legend('LQG','H_inf')
```

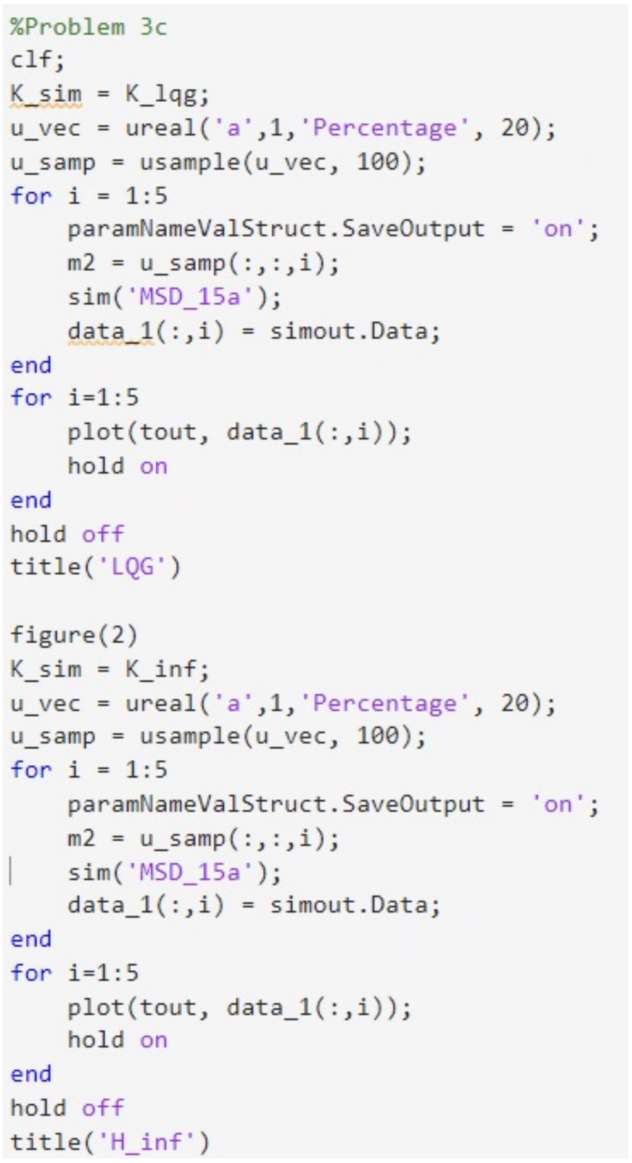
$\gamma = 1.64$



(c) Now check the robustness of your designs to parametric uncertainty in the mass m_2. Applying 20% uncertainty in this mass, compare the trajectories of the LQG vs. H_∞ loop shaping designs, keeping the disturbance a unit step. Using *usample*, simulate the performance for multiple values of m_2 (at least 5) on the same plot. Make one plot for the LQG controller and one for the H_∞ loop shaping controller.







```
%Problem 3c
clf;
K_sim = K_lqg;
u_vec = ureal('a',1,'Percentage', 20);
u_samp = usample(u_vec, 100);
for i = 1:5
    paramNameValStruct.SaveOutput = 'on';
    m2 = u_samp(:,:,i);
    sim('MSD_15a');
    data_1(:,i) = simout.Data;
end
for i=1:5
    plot(tout, data_1(:,i));
    hold on
end
hold off
title('LQG')

figure(2)
K_sim = K_inf;
u_vec = ureal('a',1,'Percentage', 20);
u_samp = usample(u_vec, 100);
for i = 1:5
    paramNameValStruct.SaveOutput = 'on';
    m2 = u_samp(:,:,i);
    sim('MSD_15a');
    data_1(:,i) = simout.Data;
end
for i=1:5
    plot(tout, data_1(:,i));
    hold on
end
hold off
title('H_inf')
```