CE2: Robust Control of an Active Suspension System

Advanced Control Systems

Martin Vuichard
Thammisetty Devakumar

Professor : Karimi Alireza

Wednesday, 27 February 2019

Introduction

This exercise studies an active suspension system. This structure models the shock of a vehicle. To develop a controller for this structure we will use H2 and H_∞ methods and Data driven methods.

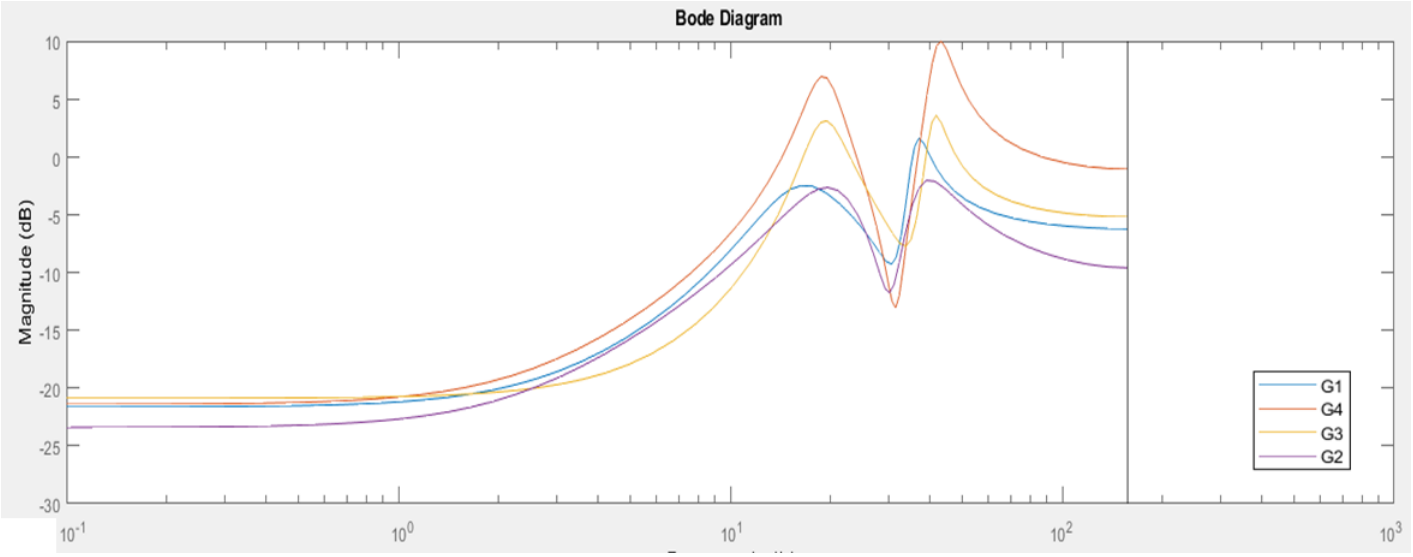
2.1 Multiplicative uncertainty

To be able to store the 4 models in a cell array, we must index them from 1 to 4 because the index G{0} is not accepted by Matlab. In our code the models are: G0 = G{1}, G1 = G{2}, G2 = G{3} and G3 = G{4}.
Some comments on our Matlab code :

- **Line 9 to 13:** The code given in question 1 is used to generate the 4 models in cell array G
- **Line 17:** "I" generates a circular permutation of the numbers
- **Line 18:** G_{stack} stores all models except the one that is considered as nominal during the iteration
- **Line 25:** One draws in bode diagram the 4 models
- **Line 29 to 34:** We show the chosen multiplicative uncertainty W_2 in the console

```
%% Name: Devakumar Thammisetty, Martin Vuichard
% Course: Advanced control systems – Spring 2019 EPFL
% Lab2 – CE2
clc
clear all
load('ASdata.mat');

%% System identification
for i=1:4
    Zd = detrend(ASdata{i});
    G{i} = oe(Zd,[4 4 1]);
    Gf{i}= spa(Zd,100);
end

%% Nominal model – Multiplicative uncertainity
for j=1:4
    I = circshift(1:4, j-1);
    G_stack = stack(1, G{I(2)}, G{I(3)}, G{I(4)});
    Gf_stack = frd(G_stack, logspace(-1, 2, 60));

    orderW2 = 4;
    [Gu, Info] = ucover(Gf_stack, G{I(1)}, orderW2, 'InputMult');

    bode(Info.W1); hold('on');
end
legend('G1', 'G4', 'G3', 'G2');

G_stack = stack(1, G{1}, G{3}, G{4});
Gf_stack = frd(G_stack, logspace(-1, 2, 60));
[Gu, Info] = ucover(Gf_stack, G{2}, orderW2, 'InputMult');
W2 = tf(Info.W1)
```

Thanks to our code we obtain the following bode diagram :



Bode diagram for the 4 models

First, we see that the multiplicative uncertainty for the model "G2" (which corresponds to the model named G1 in the problem statement) in the Figure has the lowest magnitude. Hence, we choose G1 as the nominal model, since it give least multiplicative uncertainty

Finally the W_2 chosen for "G2" (G1 in the statement) as nominal model is given below

$$W_2 = \frac{0.3993z^4 - 1.117z^3 + 1.238z^2 - 0.6121z + 0.09655}{z^4 - 2.933z^3 + 3.661z^2 - 2.246z + 0.5848}$$

2.2 Model-based H_∞ control design

Selection of W1

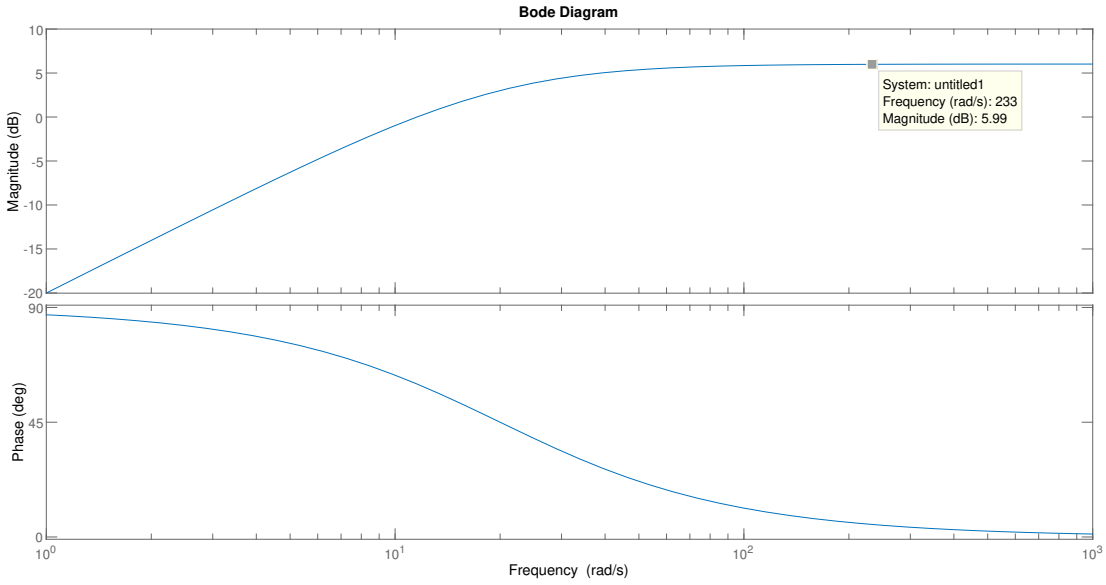
For meeting the given specifications for bandwidth and modulus margin, we know that the weighting filter will be of the form given below

$$W_1^{-1}(s) = \frac{s}{m(s+\omega_b)}$$

ω_b and m are given by the specifications : $\omega_b = 20rad/s$ and m = 0.5 then $W_1^{-1}(s) = \frac{2s}{s+20}$
Finally we get

$$W_1(s) = \frac{s+20}{2(s+0.00001)}$$

We use 0.00001 to avoid a pole at 0. Further, we discretise the continuous transfer function of W_1,using c2d command with tustin method. Finally, we plot the bode of W_1^{-1} to check if its less than 6dB at high frequencies. The bode of $W1^{-1}$ is given below which is less than 6dB at high frequencies, satisfies the specifications.

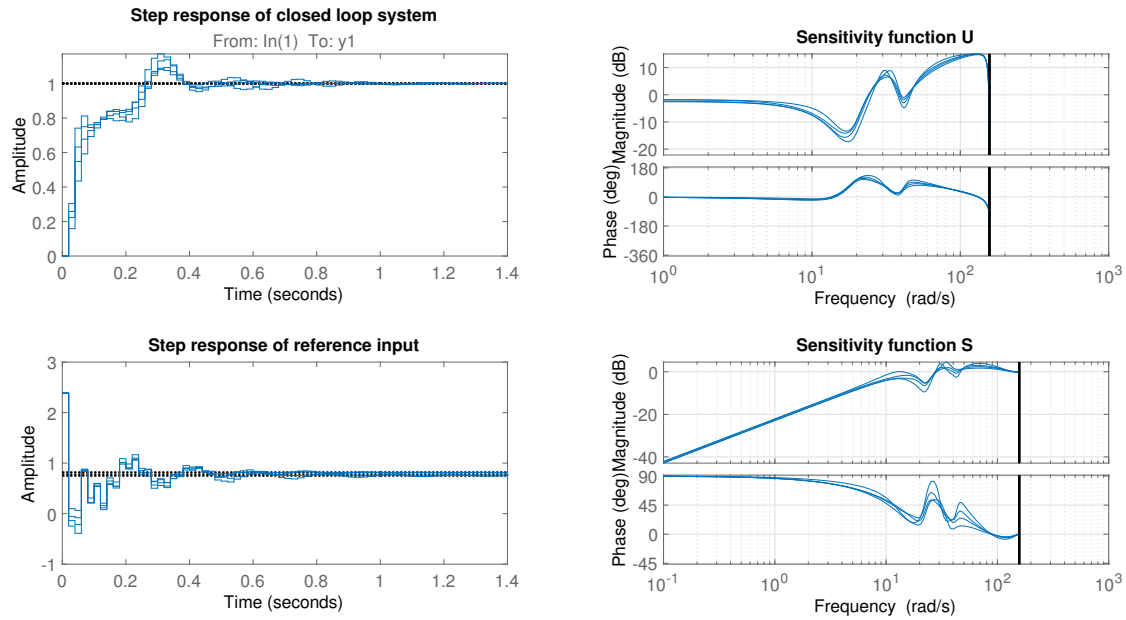


Bode of W_1^{-1}

```
%% Name: Devakumar Thammisetty, Martin Vuichard
% Course: Advanced control systems – Spring 2019 EPFL
% Lab2 – CE2
load('ASdata.mat');

%% System identification
for i=1:4
    Zd = detrend(ASdata{i});
    G{i} = oe(Zd,[4 4 1]);
    Gf{i}= spa(Zd,100);
end

%% Nominal model – Multiplicative uncertainity
col = ['r', 'g', 'b', 'k']
```

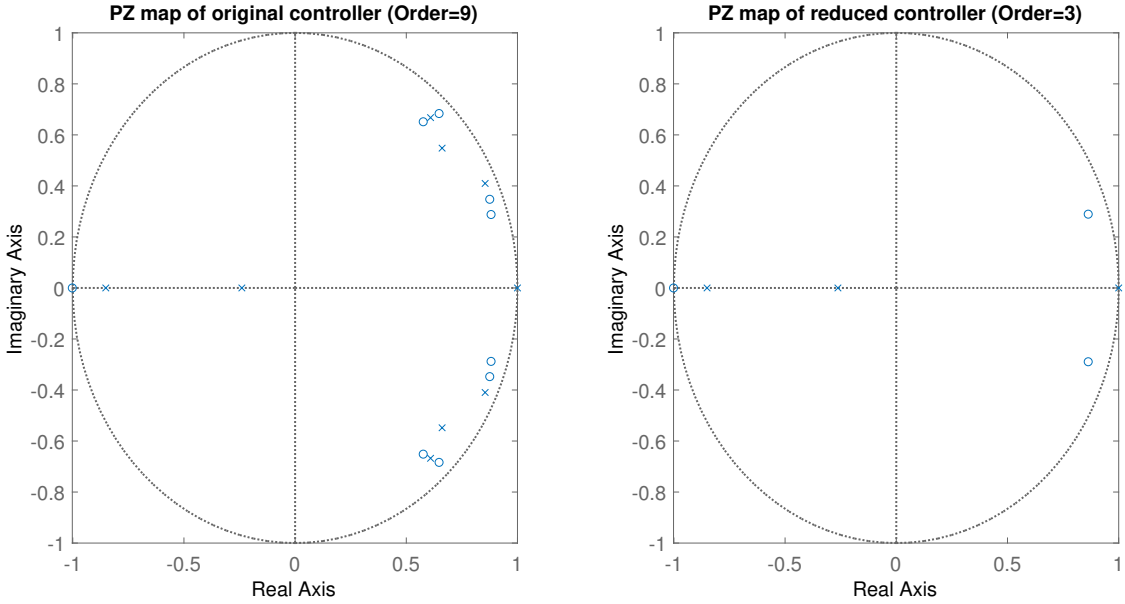
```
for j=1:4
    I = circshift(1:4, j-1);
    G_stack = stack(1, G{I(2)}, G{I(3)}, G{I(4)});
    %Gf_stack = stack(1, Gf{2}, Gf{3}, Gf{4});
    Gf_stack = frd(G_stack, logspace(-1, 2, 60));

    orderW2 = 4;
    [Gu, Info] = ucover(Gf_stack, G{I(1)}, orderW2, 'InputMult');

    bode(Info.W1, col(j)); hold('on');
    W2est{j} = Info.W1;
end
legend('G1 multiplicative', 'G4', 'G3', 'G2');

W1s = tf([1 20], [2 0.00002]);
n = 2;
W2 = W2est{n};
W1d = c2d(W1s, 0.02, 'tustin');
W3 = 1/10;
K = mixsyn(G{n}, W1d, W3, W2);

G_stack = stack(1, G{1}, G{3}, G{4}, G{2});
T = feedback(G_stack*K, 1);
U = feedback(K, G_stack);
S = feedback(1, G_stack*K);

step(T);
step(U);
bode(U);
bode(S);

pzmap(K);
Kr = reduce(K, 3);
pzmap(Kr);
Tk = feedback(G_stack*Kr, 1);
Uk = feedback(Kr, G_stack);
Sk = feedback(1, G_stack*Kr);

step(Tk,'r');
step(Uk, 'r');
bode(Uk, 'r');
bode(Sk, 'r');
```

H_∞ Controller is designed using mixed sensitivity approach. W3 filter is derived to meet the specifications viz. step of reference signal < 3V and magnitude of input sensitivity function less than 20dB at high frequencies. The step response of reference signal and closed loop system, bode of input sensitivity function U and sensitivity function S are shown together in figure below. We observe that the controller designed using mixed sensitivity approach meets the specifications.



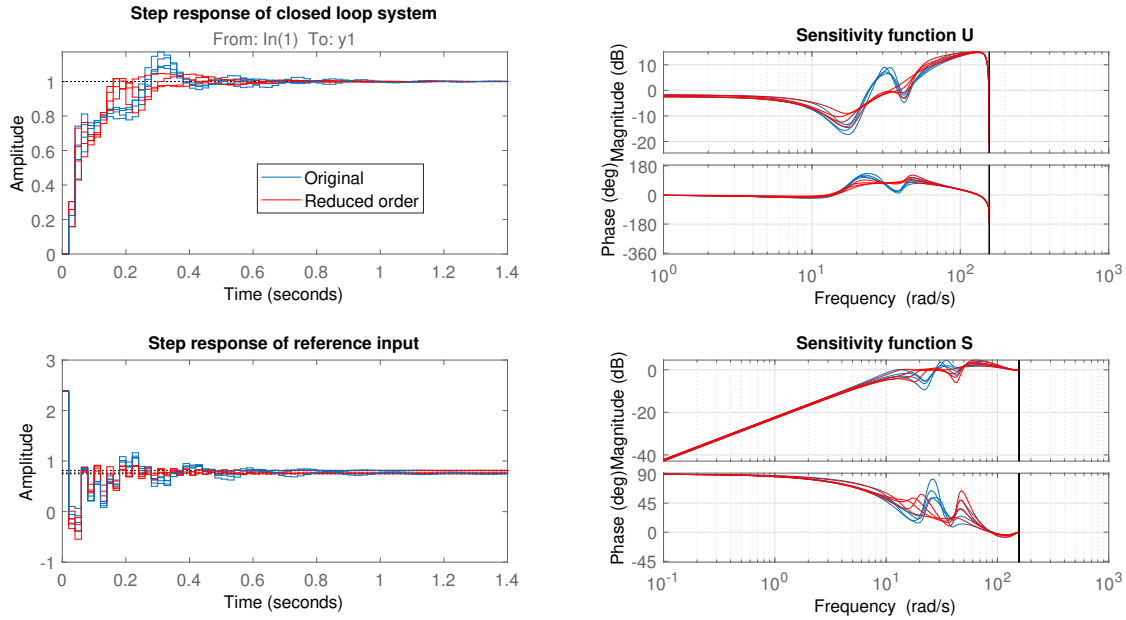
System response with mixed sensitivity based H_∞ controller

However, the order of the controller(K) is found to be 9. We looked at the pole zero map of the controller to see if there are any possible pole zero cancellations. We found that, there are no possible pole zero cancellations when the reduced order of the controller is 3, shown in figure below.



Pole zero map of controller: order reduction

Subsequently, to verify the system performance, we evaluated the step response of reference signal, step response of the closed loop system and bode plot of the sensitivity functions with the reduced order controller, results are shown below. Results of the reduced order controller are shown in red. We see that the reduced order controller performance is similar to the original controller and meets the design specifications.



System response with mixed sensitivity based H_∞ controller with reduced order

2.3 Model-Based H_2 Controller Design

The state-space equations of the closed-loop system are :

$$\begin{aligned} \dot{x(t)} &= A \cdot x(t) + B \cdot u(t) \\ y_1(t) &= C \cdot x(t) \\ y_2(t) &= -K \cdot x(t) \\ u(t) &= v(t) - K \cdot x(t) \end{aligned}$$

Therefore

$$\ddot{x(t)} = (AB)(t) + B(t)$$

One seeks a convex optimization problem using LMI :

$$y^*(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} C \\ -K \end{bmatrix} \cdot x(t) = C^* \cdot x(t)$$

Further, the optimization problem can be formulated as shown below step by step.

$$\min\left(tr\left(\begin{bmatrix} CL \\ -KL \end{bmatrix} \cdot \begin{bmatrix} C^T & -K^T \end{bmatrix}\right)\right)$$
$$A \cdot L + L \cdot A^T - B \cdot Y - Y^T \cdot B^T + B \cdot B^T \leq 0$$
$$with Y = KL$$

$$\min\left(tr\left(\left[\begin{array}{c|c} CLC^T & -CLK^T \\ \hline -KLC & KLK^T \end{array}\right]\right)\right)$$
$$A \cdot L + L \cdot A^T - B \cdot Y - Y^T \cdot B^T + B \cdot B^T \leq 0$$

$$\min\left(C \cdot L \cdot C^T + K \cdot L \cdot K^T\right)$$
$$A \cdot L + L \cdot A^T - B \cdot Y - Y^T \cdot B^T + B \cdot B^T \leq 0$$

Thus we have three conditions (with γ_1 and γ_2, two scalar to optimize) :
Condition 1 : $C \cdot L \cdot C^T < \gamma_1$
Condition 2: $K \cdot L \cdot K^T < \gamma_2 \Leftrightarrow$Y $\cdot K^T < \gamma_2 \Leftrightarrow$Y $\cdot (L^{-1})^T \cdot Y^T < \gamma_2$ because L is symmetrical so L^{-1} is also symmetrical and therefore $(L^{-1})^T = L^{-1}$

$\Leftrightarrow 0 < \gamma_2 \cdot I - Y \cdot L^{-1} \cdot Y^T$

by the Scur lemma one gets : $\begin{bmatrix} \gamma_2 & Y \\ Y^T & L \end{bmatrix} > 0$

Condition 3 : $A \cdot L + L \cdot A^T - B \cdot Y - Y^T \cdot B^T + B \cdot B^T \leq 0$

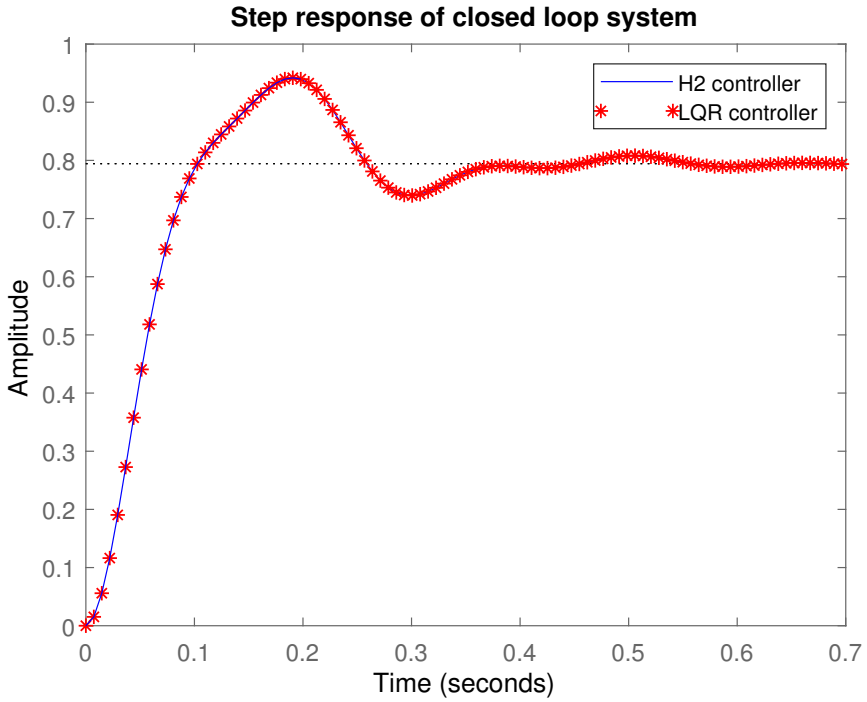
In the following one finds the code Matlab to use YALMIP :

- **Line 36:** d2c is used to get G1.
- **Line 33 to 36:**The matrix A, B, C and D are then recovered.
- **Line 47:** This is condition 3
- **Line 48:** This is condition 2
- **Line 49:** This is condition 1
- **Line 51:** One optimize the variables
- **Line 53:** We recover the controller K
- **Line 54:** The closed system transfer function is recovered with controller K
- **Line 55:** As requested in the last question one checks the consistency of the results with the command lqr
- **Line 56:** The closed loop system transfer function when controller K is generated by the lqr command

```
%% Name: Devakumar Thammisetty, Martin Vuichard
% Course: Advanced control systems - Spring 2019 EPFL
% Lab2 - CE2
clc
clear all
load('ASdata.mat');

%% System identification
for i=1:4
    Zd = detrend(ASdata{i});
    G{i} = oe(Zd,[4 4 1]);
    Gf{i}= spa(Zd,100);
end

%% Nominal model - Multiplicative uncertainity
```

```
col = ['r', 'g', 'b', 'k'];
for j=1:4
    I = circshift(4:-1:1, j);
    G_stack = stack(1, G{I(2)}, G{I(3)}, G{I(4)});
    orderW2 = 4;
    [Gu, Info] = ucover(Gf_stack, G{I(1)}, orderW2, 'InputMult');

    bode(Info.W1, col(j)); hold('on');
    W2est{j} = Info.W1;
end
legend('G1 multiplicative', 'G2', 'G3', 'G4');

%% 2.3

Gc = d2c(G{2});

ssR = ss(Gc)
A = ssR.A
B = ssR.B
C = ssR.C
D = ssR.D
gamma1=sdpvar(1,1);
gamma2=sdpvar(1,1);
Y = sdpvar(1,4); %
L=sdpvar(4,4,'symmetric');
lmi=[A*L+L*A'-B*Y-Y'*B'+B*B']<=0;
lmi=[lmi, [gamma2 Y;Y' L]>=0];
lmi=[lmi,C*L*C'<=gamma1];
options=sdpsettings('solver','lmilab');
optimize(lmi,gamma1 + gamma2, options);

% Estimation of H2 controller gain
Kh2 = value(Y)*inv(value(L));
G_cl_H2 = ss(A-B*Kh2, B, C, D);

% LQR gain estimation
K_lqr = lqr(A, B, C'*C, eye(1));
G_cl_lqr = ss(A-B*K_lqr, B, C, D);

step(G_cl_H2, 'b');
step(G_cl_lqr, 'r*');
```

The values of the two controllers are very close (the difference is of the order of 10^{-4} in the Figure below). Moreover, we can see on the graph that the responses of the two systems are almost equal which demonstrates the consistency of the results and that the controller K developed by the YALMIP method is a good model. Step response of the closed loop system with both LQR and H2 controllers is shown below.



Step response of the closed loop system with H2 and LQR controllers

Following data shows the values of MATLAB LQR controller and H2 controller.

```
>> Kh2

Kh2 =

    1.7454    0.6242    3.0256    1.3354

>> K_lqr

K_lqr =

    1.7454    0.6243    3.0253    1.3354
```

Comparison of H2 controller values with MATLAB LQR command

2.4 Data-driven control design

The same specifications as in model based H_∞ are used for the design. Hence, we use same weighting filters W1 and W3. Further, we choose integrator with a small gain as initial controller so that the optimization algorithm can improve teh controller with number iterations.In addition to the model based H_∞ constrains used in 2.2, here we aim to minimise the 2-norm of the tracking error. Hence, we expect a better performance using this method for the same specifications.

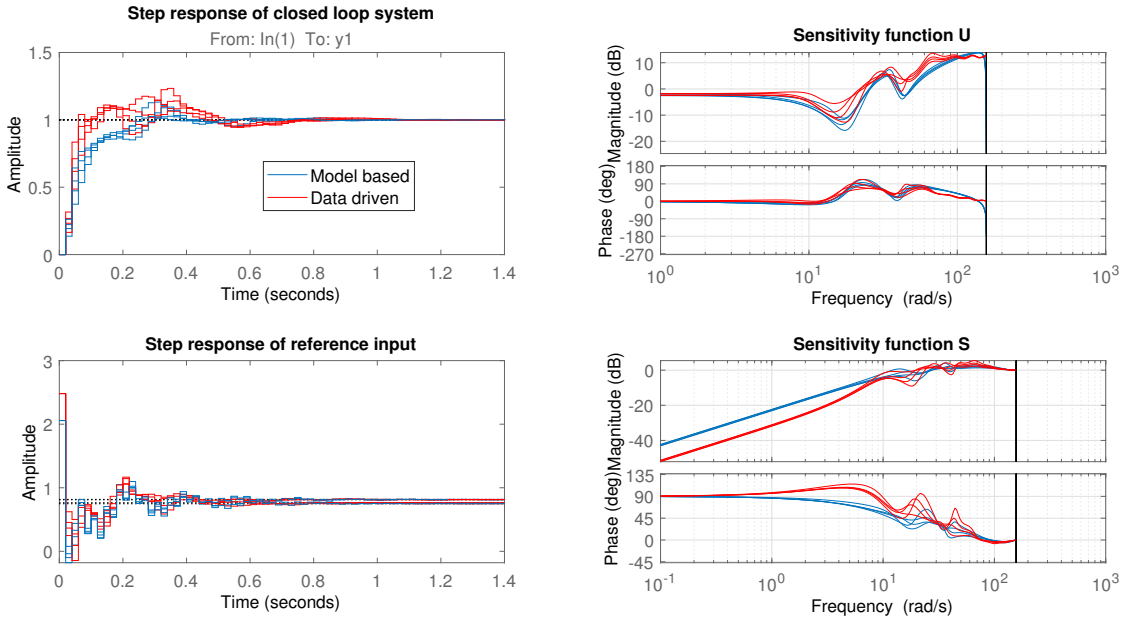
We found that the data-driven controller with same W3 filter as in model based controller give slightly higher input signal at the start (>3v). Hence, we updated the filter W3 in data based controller to keep the input singal below 3v. The results are shown below.

We use 'cinfW1' and 'cinfW3' in datadriven.m as constrains to have infinity norm less than 1. Further, we revise the W3 filter from model based controller from 1/10 to 1/5, to have input signal less than 3v.

The MATLAB code used for the data driven controller design is given below. We use mosek for solving the optimization problem.

```
%% Name: Devakumar Thammisetty, Martin Vuichard
% Course: Advanced control systems – Spring 2019 EPFL
% Lab2 – CE2
clc
clear all
load('ASdata.mat');

%% System identification
for i=1:4
    Zd = detrend(ASdata{i});
    G{i} = oe(Zd,[4 4 1]);
    Gf{i}= spa(Zd,100);
end

%% 2.4
W1s = tf([1 20], [2 0.00002]);
W1d = c2d(W1s, 0.02, 'tustin');
W3 = 1/5;
Ts = 0.02;
w = logspace(-2, log10(pi/Ts), 500);
z = tf('z', 0.02);
Kinit = 0.001*(1/(z^10*(z-1)));
G_stack = stack(1, G{3}, G{2}, G{4}, G{1});
P = data_driven('G', G_stack, 'W', w', 'cinfW1', W1d, 'cinfW3', W3, 'Kinit',
    Kinit, 'order', 4, 'Ts', Ts, 'tol', 0.0001, 'maxiter', 50);
[Kdd, obj] = solve(P);

% Data driven results
Tk = feedback(G_stack*Kdd, 1);
Uk = feedback(Kdd, G_stack);
Sk = feedback(1, G_stack*Kdd);

step(Tk,'r');
step(Uk, 'r');
bode(Uk, 'r');
bode(Sk, 'r');
```

The comparison of the data driven controller with the Model based controllers designed in 2.2 is given below.



Comparison of Data driven controller performance with model based controller

We observe that the data driven controller gives better tracking as seen from the first sub plot. Further, all the design specifications are met. In summary, data driven controller for this problem gave better performance when compared to model based controller interms of tracking, input reference signal amplitude.

Conclusion

In this exercise, we identified a nominal model from various test data. Further, we estimated various filters for meeting the controller design specifications. Subsequently, we designed controllers using different methods such as model based H2 and H_∞ methods and Data driven methods. The results of various controllers are compared. It is observed that, the controller design using various methods meets the design specifications. However, the data based controller gave the best tracking performance in addition to meeting the controller performance specifications.