Chenyang Wang
UID: 105425757
Prof. M'Closkey

MAE 270A Course Project Report

1. Time-domain Model Identification via Hankel Matrix Analysis

In this part, we are going to identify a real system via Hankel Matrix Analysis. First, let's consider a multi-input-multi-output, discrete-time linear system. An important assumption of the system is that the system is asymptotically stable. And the system can be described as follow:

$$\begin{aligned}\mathbf{x}_{k+1} &= A\mathbf{x}_k + B\mathbf{u}_k\\ \mathbf{y}_k &= C\mathbf{x}_k\end{aligned}$$

In order to identify the system, we need to first construct the Hankel Matrix, which is defined as follow:

$$H_n = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n-1} \end{bmatrix} \in \mathbf{R}^{mn \times nq}.$$

Note that each 'h' in the H-matrix is constructed by organizing the system responses from each channel into a 2-by-2 matrix. By doing so, we first construct H_{100}, compute the singular values, and graph the singular values. Although we are not required to graph the singular values for H_{20}, H_{40}, H_{80}, for completeness, I computed them and included in the following graph:

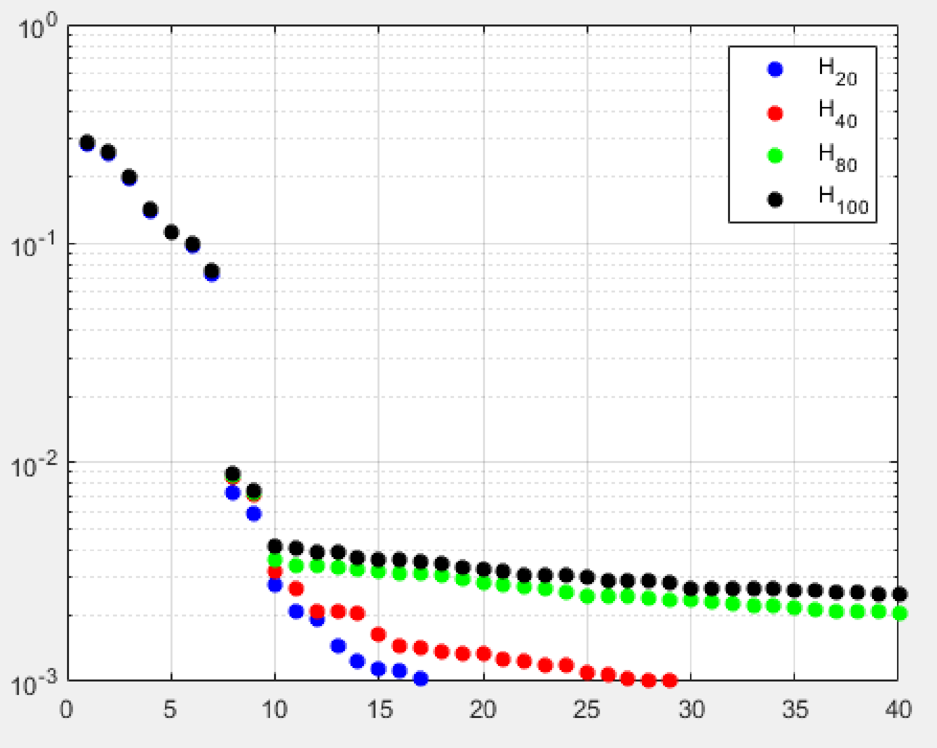


Figure 1. Singular Values of Hankel Matrix with different dimensions.

Next, we compute the models from H_{100} with different state dimensions: ns = 6, 7, 10, 20. To compute the model, one Hankel matrix is not enough, in fact we need to construct another Hankel matrix start with h_2, the definition of the second Hankel matrix is shown below:

$$\tilde{H}_n = \begin{bmatrix} h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ h_4 & h_5 & h_6 & \cdots & h_{n+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n+1} & h_{n+2} & h_{n+3} & \cdots & h_{2n} \end{bmatrix} = \mathcal{O}_n A \mathcal{C}_n.$$

By taking the SVD of the first Hankel Matrix, we get:

$$H_n = U\Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_{n_s} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1 \Sigma_{n_s} V_1^T$$

And I define my O$_n$ and C$_n$ to be:

$$\mathcal{O}_n = U_1 \Sigma_{n_s} \text{ and } \mathcal{C}_n = V_1^T$$

Once we get O$_n$ and C$_n$, the 'A' matrix of the system can then by calculated as follow:

$$A = \mathcal{O}_n^{\dagger} \tilde{H}_n \mathcal{C}_n^{\dagger}.$$

Once we get the 'A' matrix, we can then confirm that the models are asymptotically stable by taking the maximum eigenvalue of A. The results are: ns = 6, 0.9526; ns = 7, 0.9144; ns = 10, 0.9141; ns = 20, 0.9975. Thus, we confirm that all the models are asymptotically stable.

Next, we need to simulate the impulse response of each model versus the measurement data. The graphs are shown below:

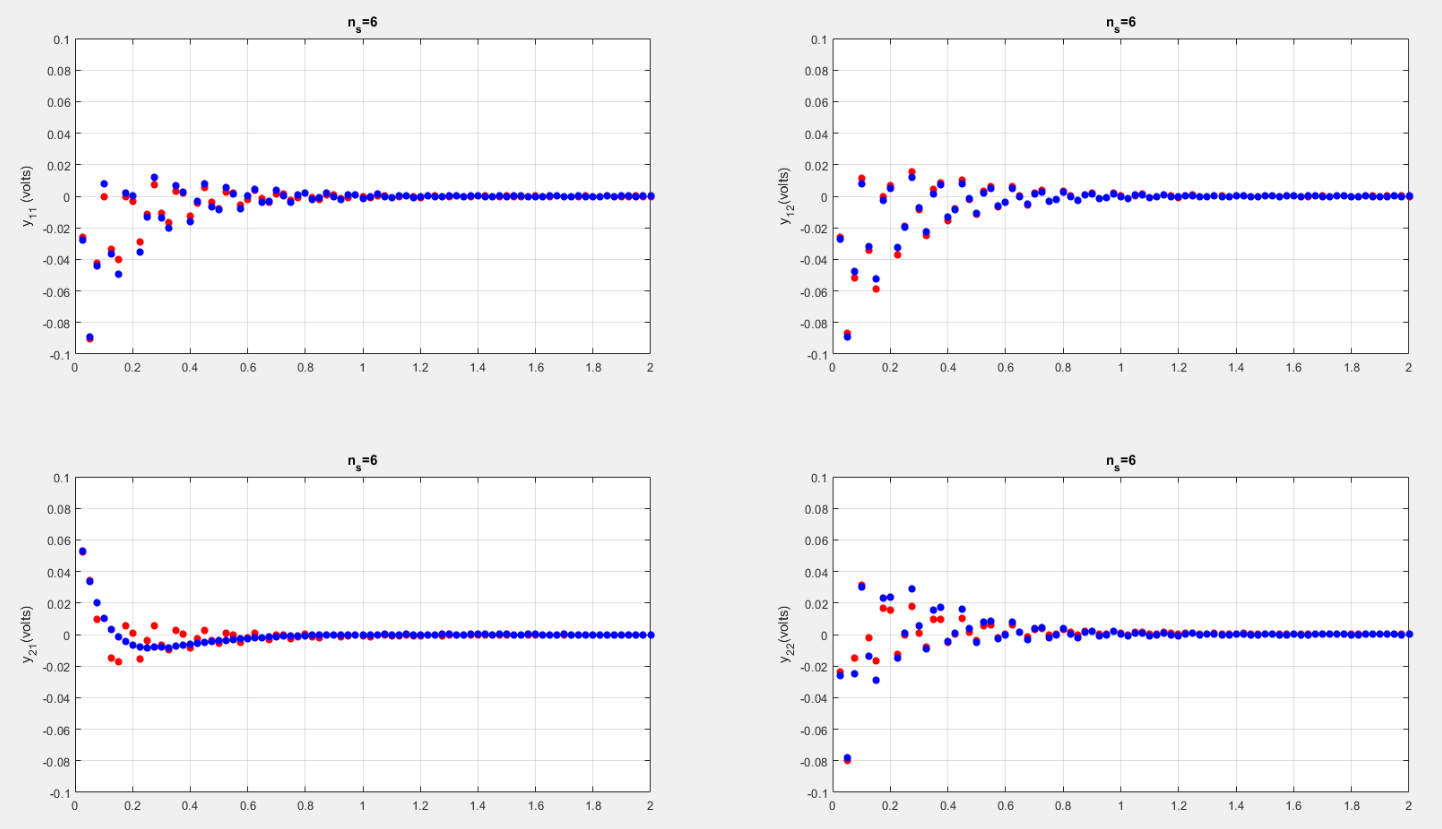


Figure 2. ns = 6 model

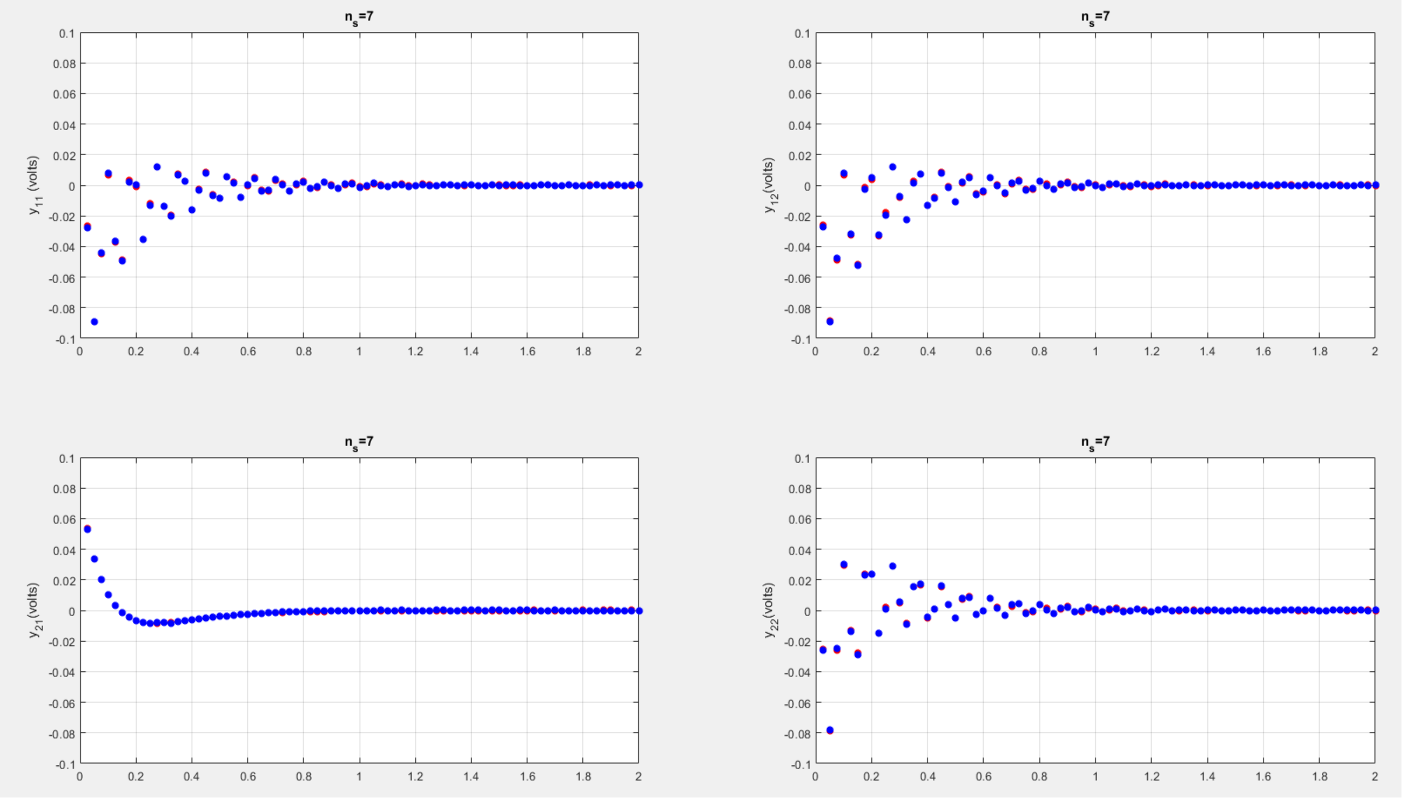


Figure 3. ns = 7 model

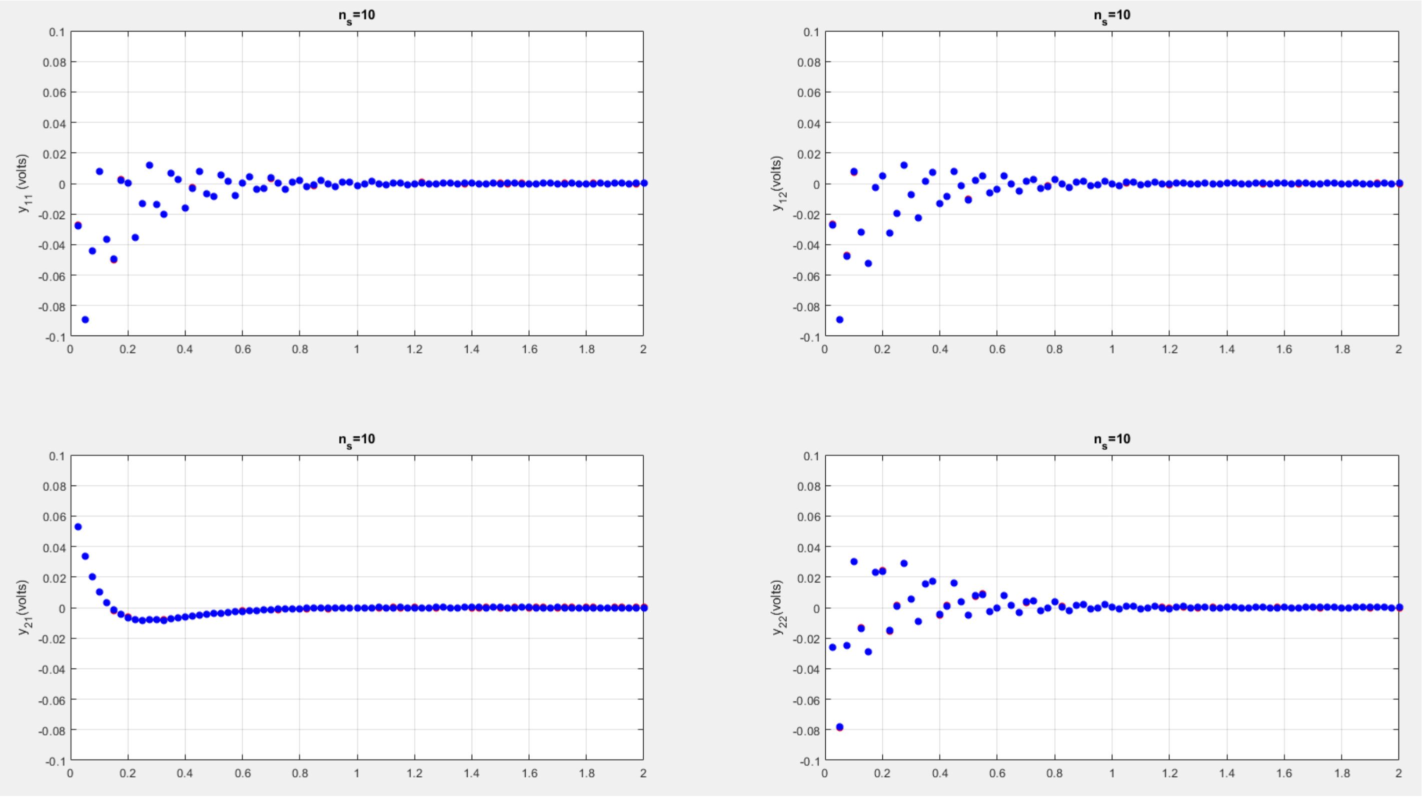


Figure 4. ns = 10 model

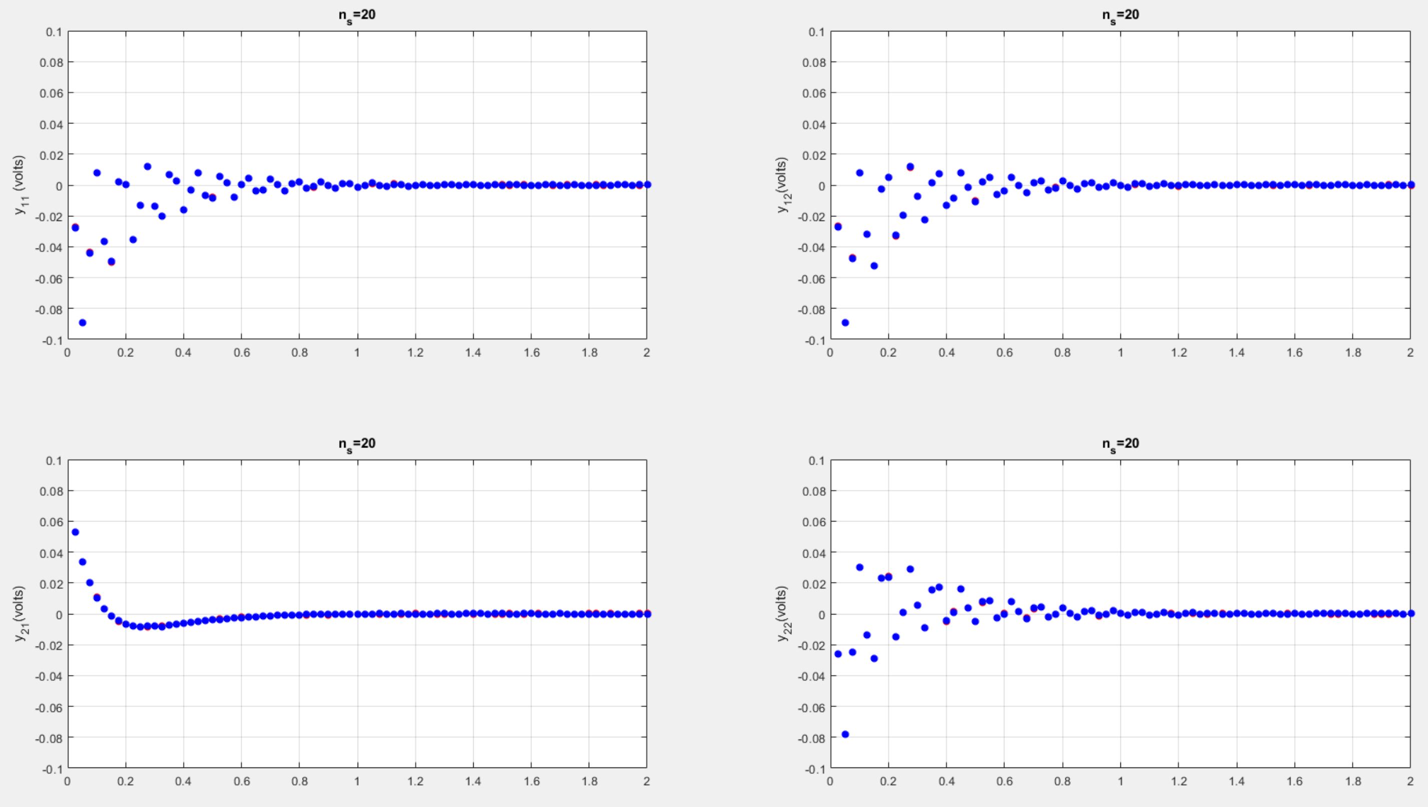


Figure 5. ns = 20 model.

From the graphs, we can clearly see that the ns = 6 model is inferior, and all the other models are essentially the same.

Next, we compare the model to the data by comparing the model frequency response to the empirical frequency response we obtained from the pulse response data. Recall that the model frequency response can be obtained as follow:

$$C(e^{j\omega t_s}I - A)^{-1}B + D,$$

In particular, we plot the magnitude and phase of the frequency response of each model that we identified in the previous part. The graphs are shown below:

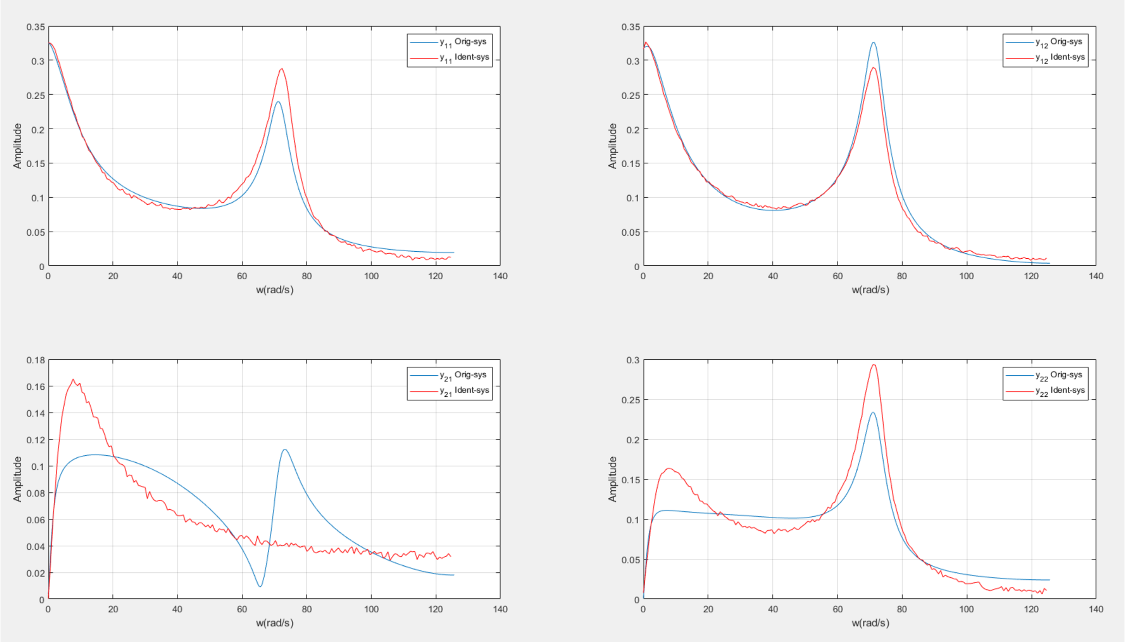


Figure 6. ns = 6, magnitude

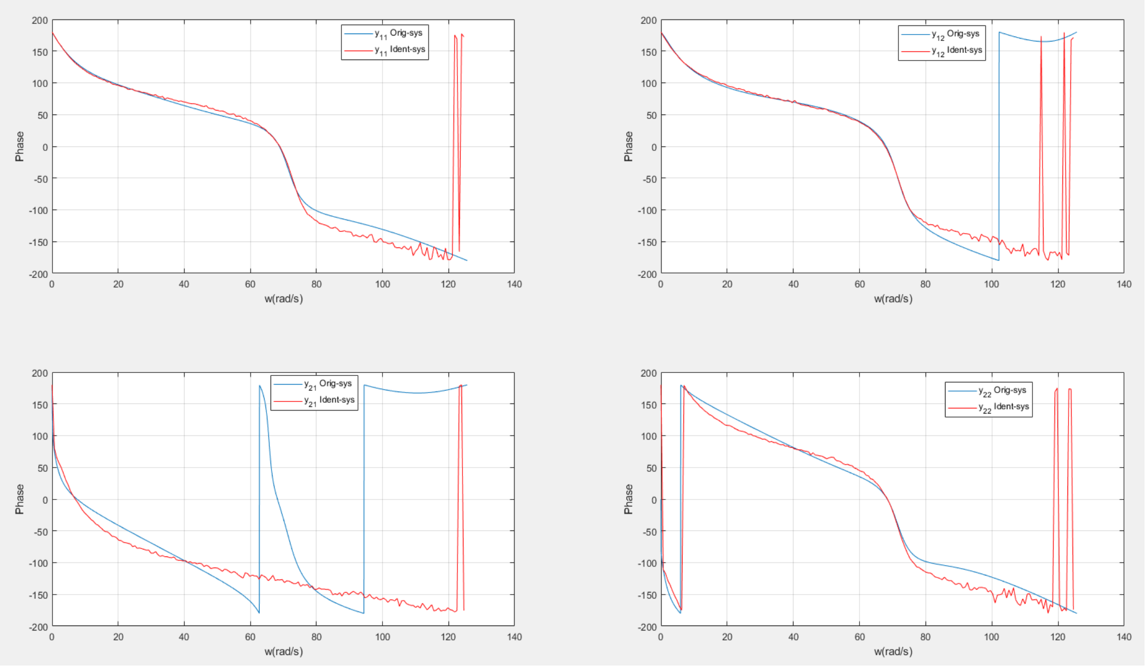


Figure 7. ns = 6, phase

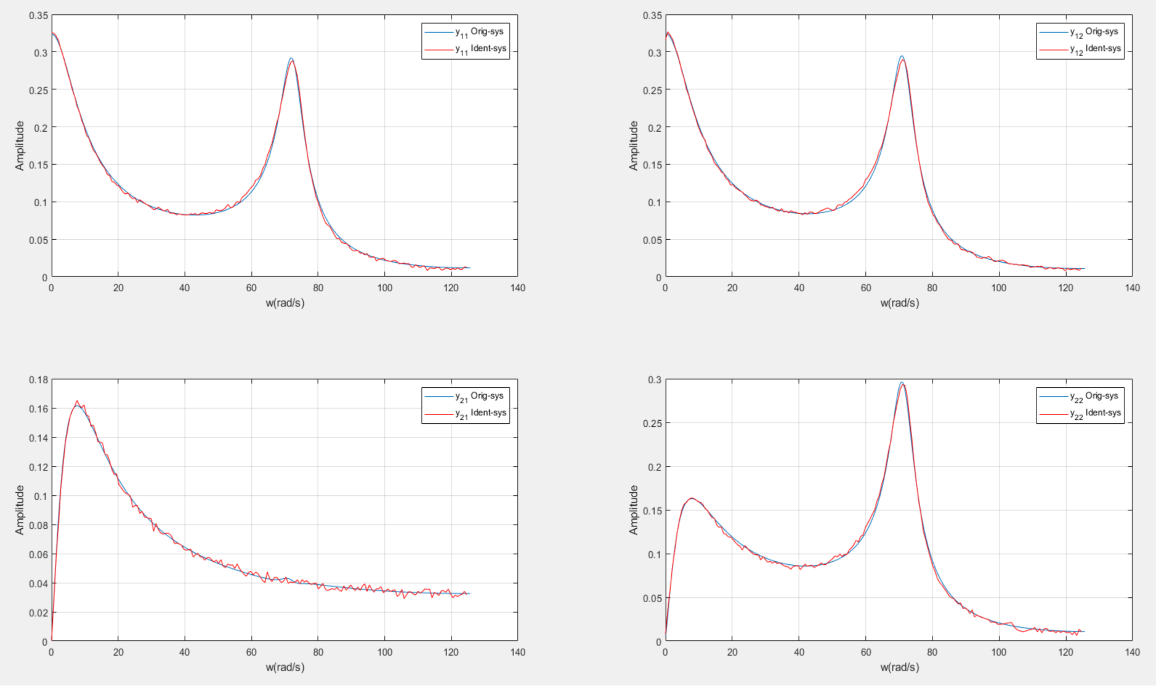


Figure 8. ns = 7, magnitude

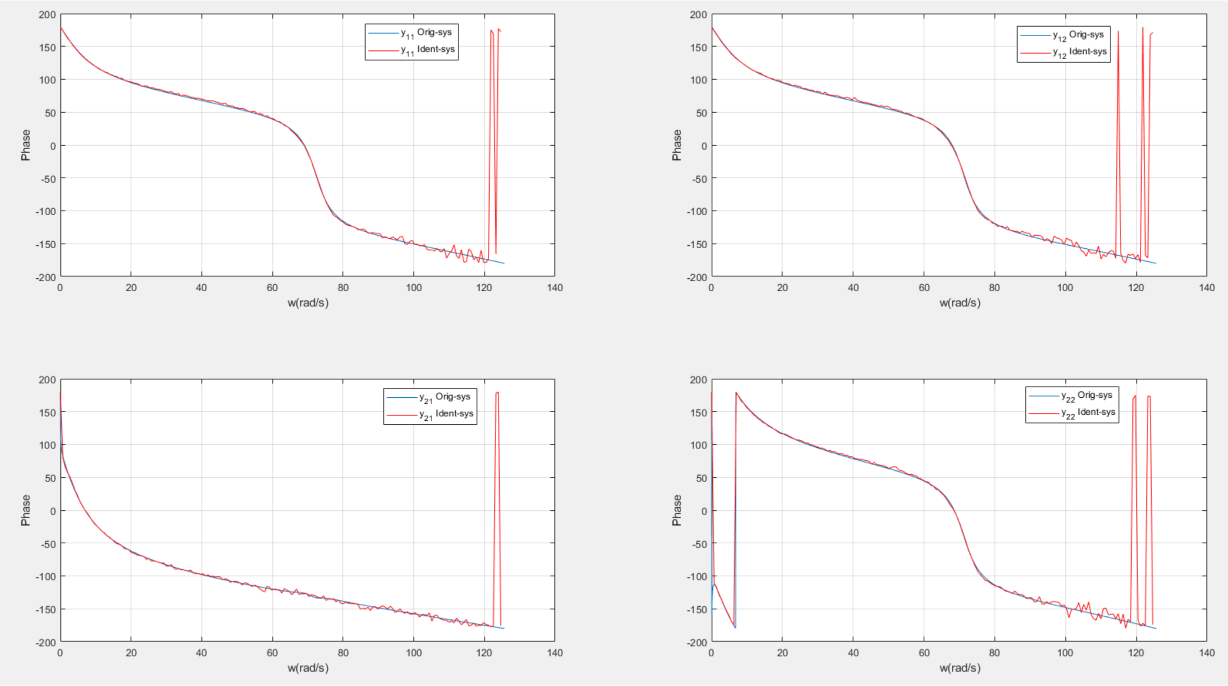


Figure 9. ns = 7, phase

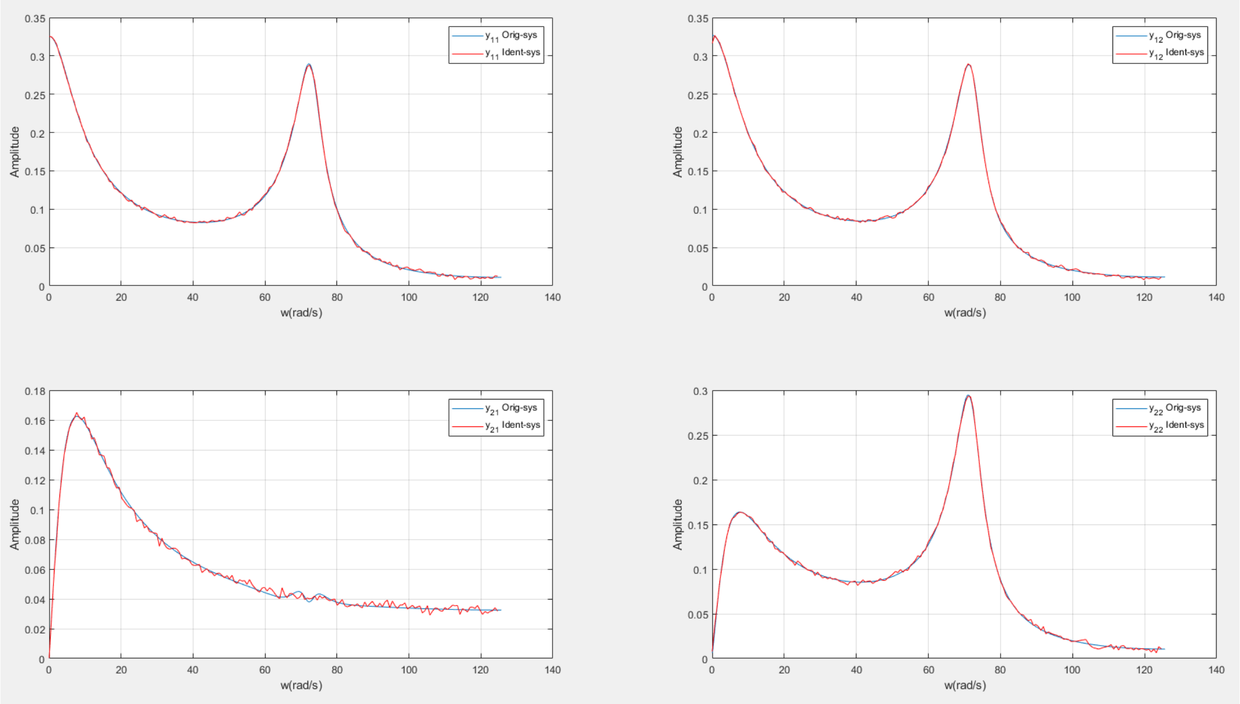


Figure 10. ns = 10, magnitude

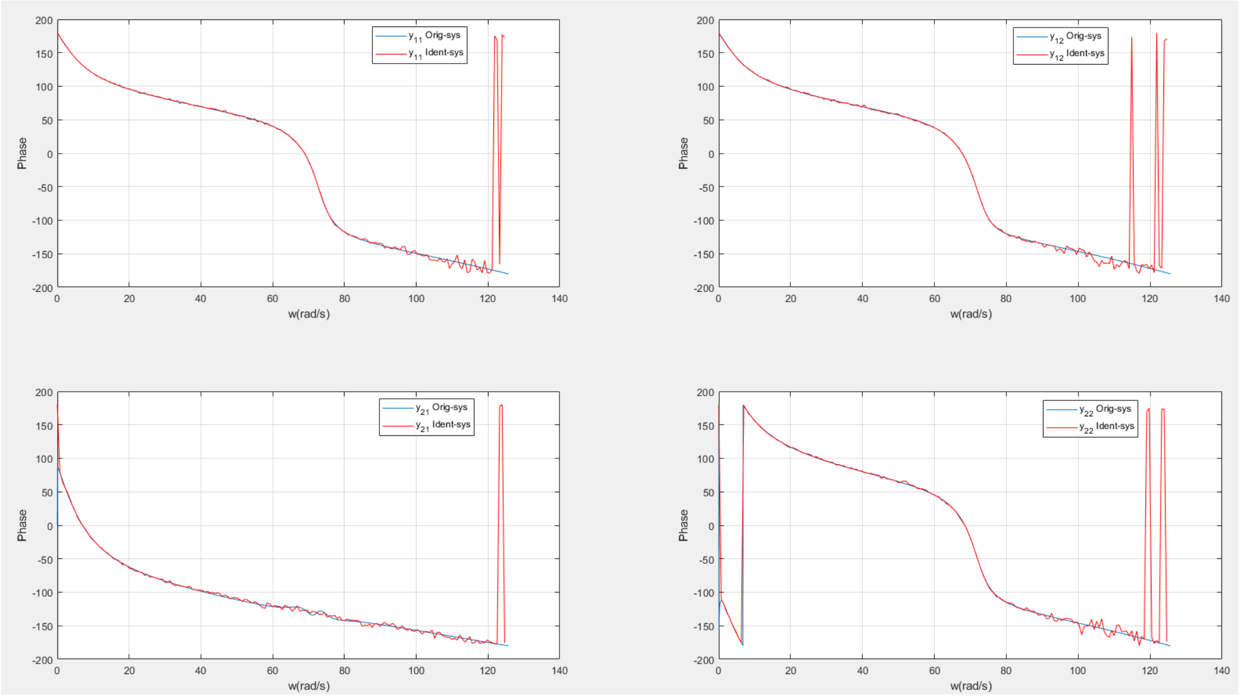


Figure 11. ns = 10, phase

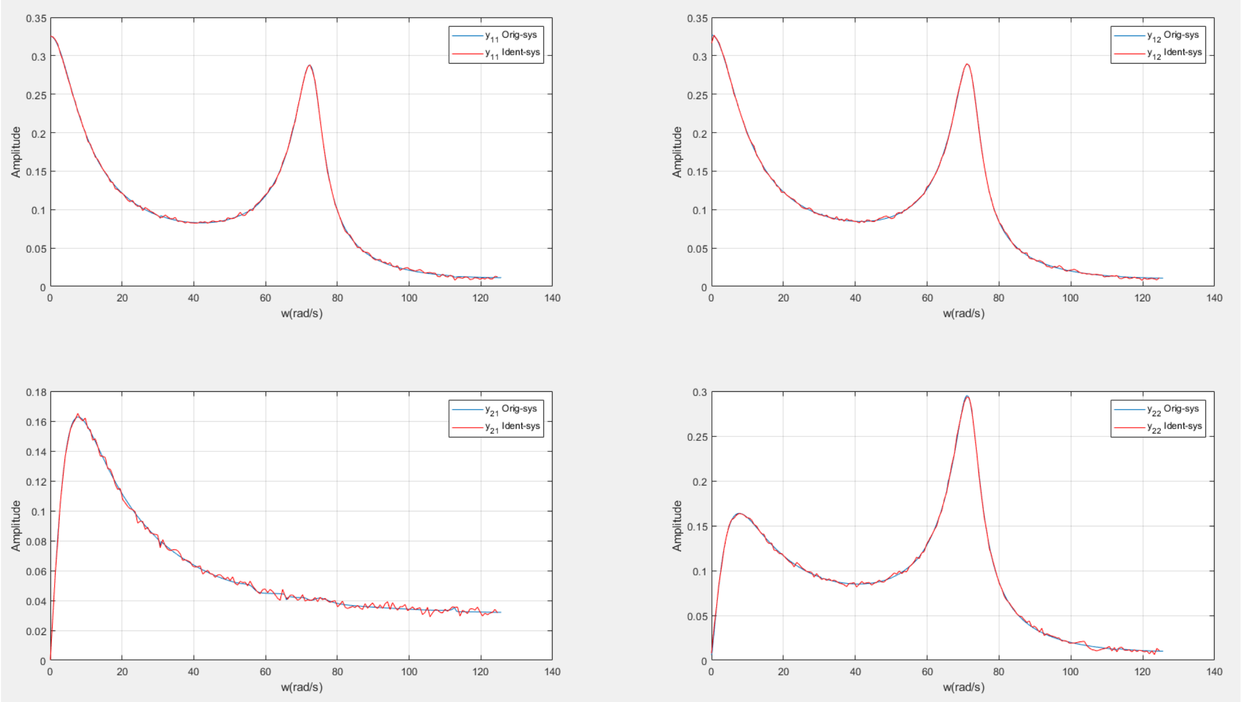


Figure 12. ns = 20, magnitude

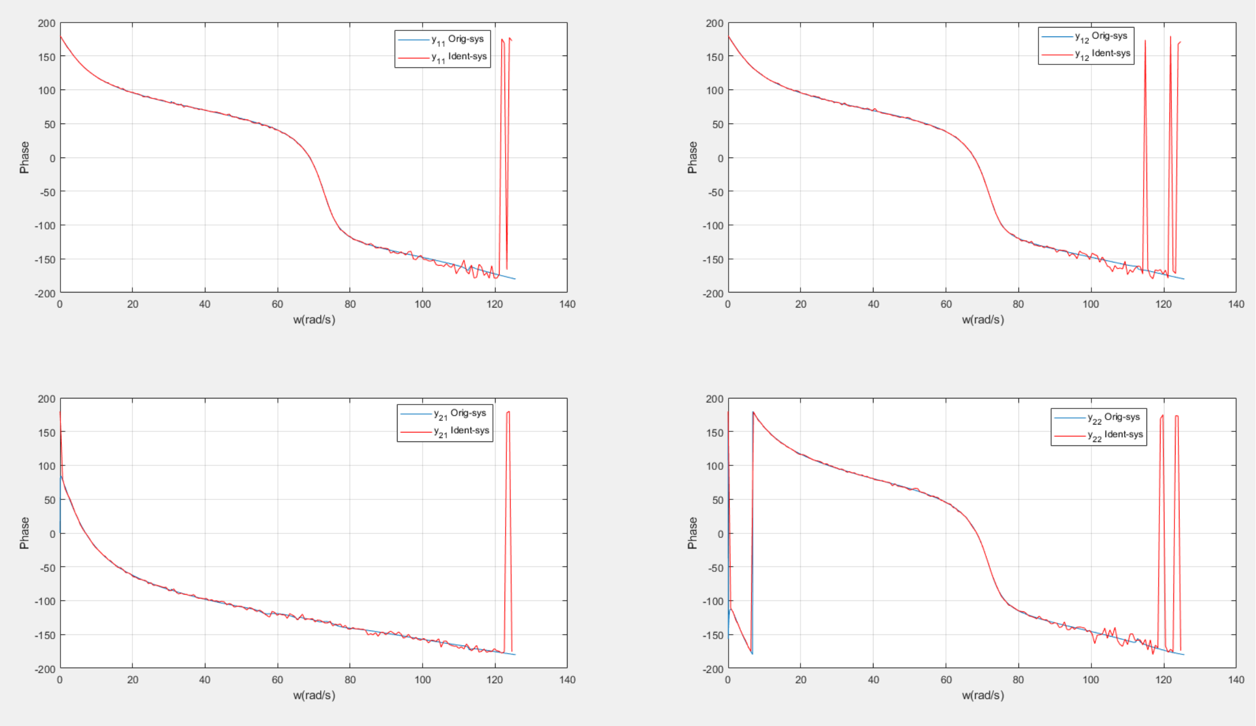
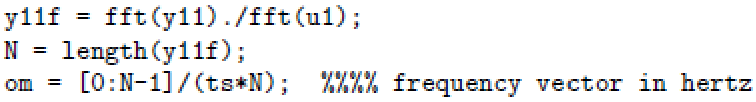


Figure 13. ns = 20, phase

As before, we can see from the graphs that the ns = 6 model is inferior, and the remaining models are essentially the same. Note that we can directly estimate the frequency response without the model by using the pulse response. The frequency response associated with each channel can be computed by using the following code:



```
y11f = fft(y11)./fft(u1);
N = length(y11f);
om = [0:N-1]/(ts*N);  %%%% frequency vector in hertz
```

In particular, we will overlay these results on top of the model plots from the previous part, and the graphs are shown below:

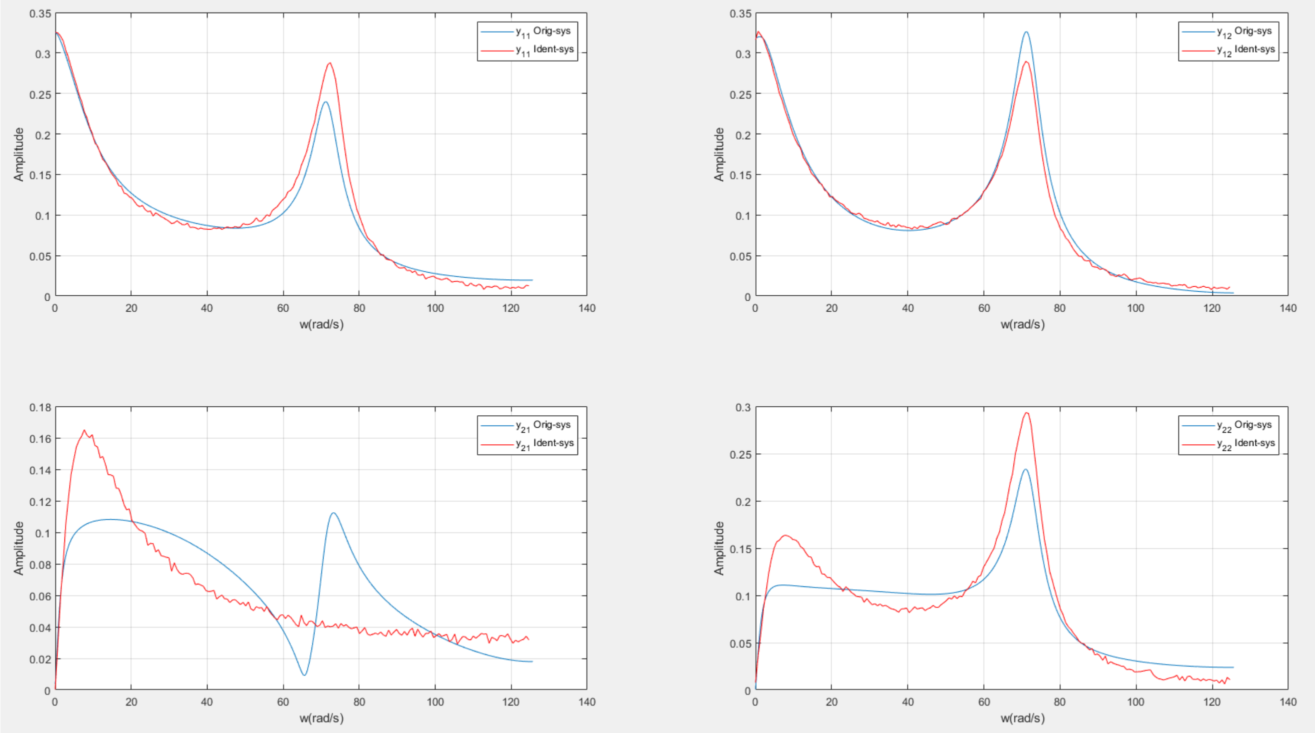


Figure 14. ns = 6, magnitude

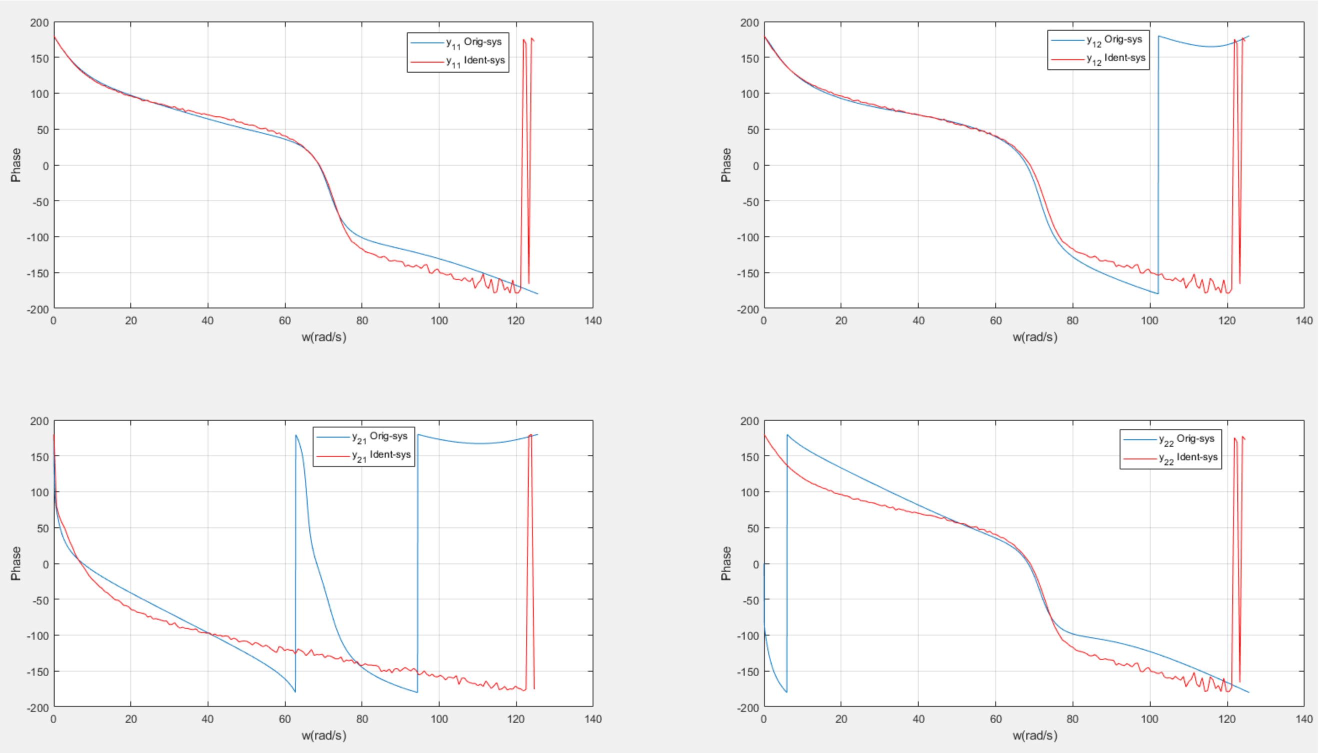


Figure 15. ns = 6, phase

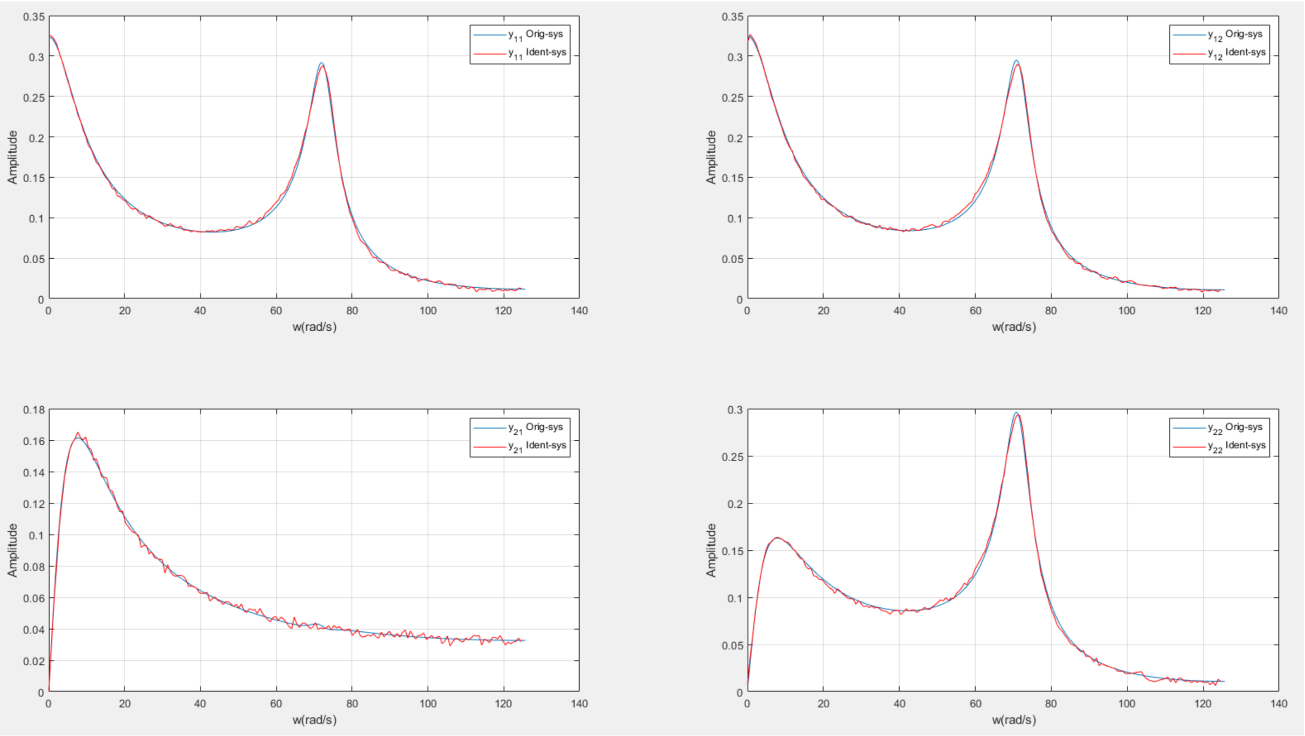


Figure 16. ns = 7, magnitude

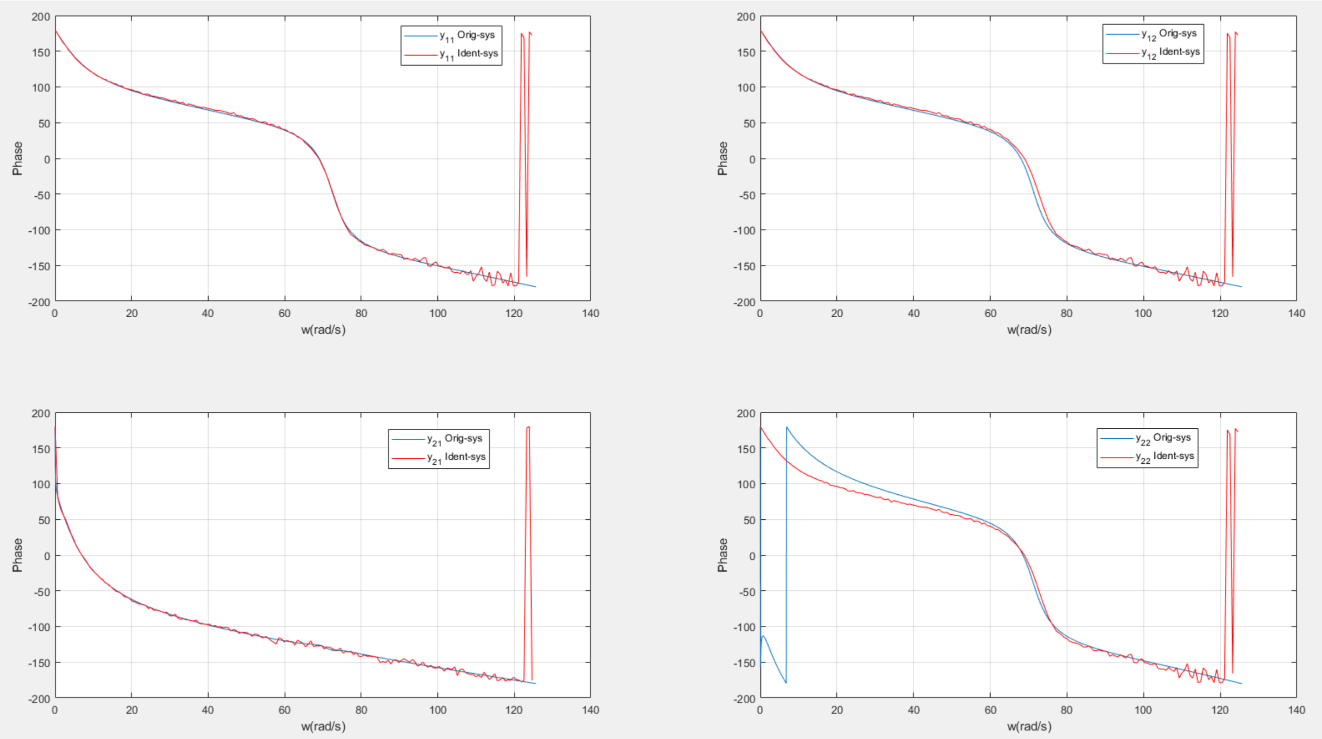


Figure 17. ns = 7, phase

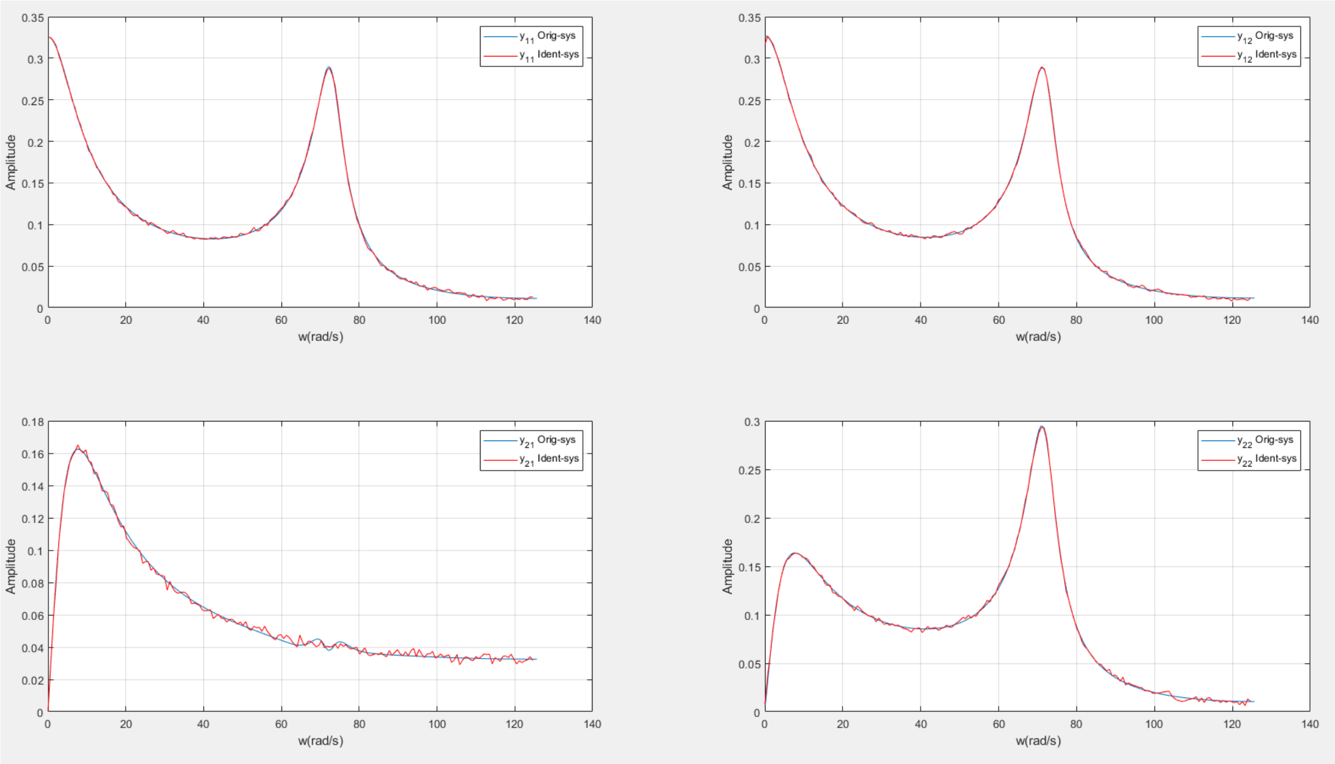


Figure 18. ns = 10, magnitude

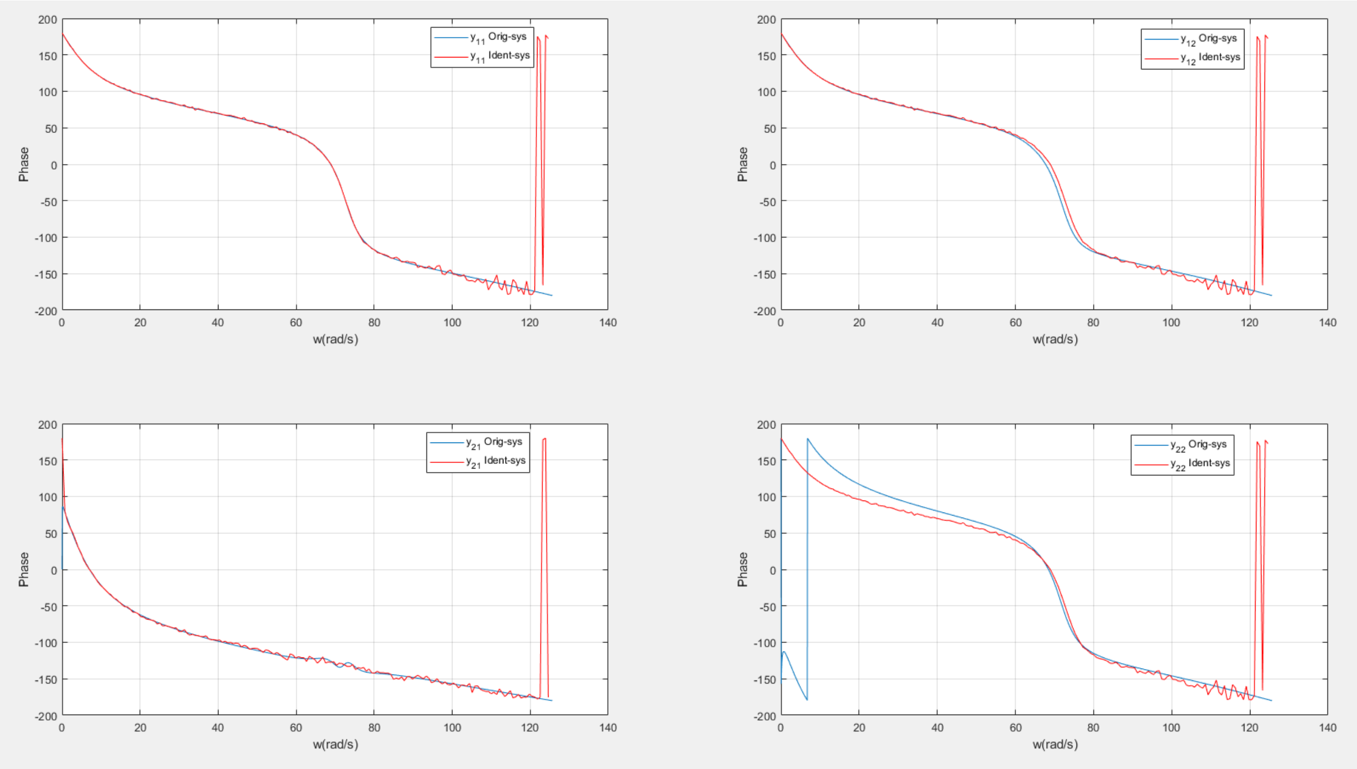


Figure 19. ns = 10, phase

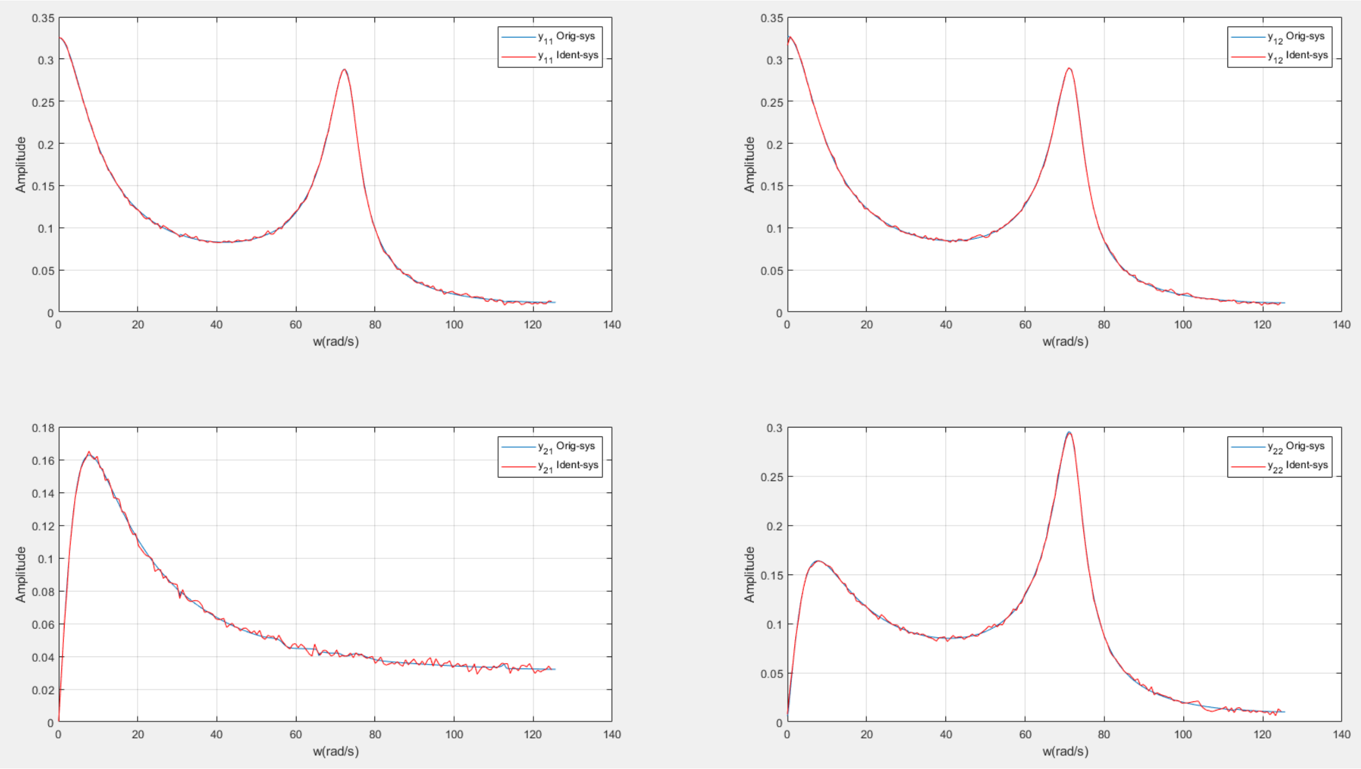


Figure 20. ns = 20, magnitude

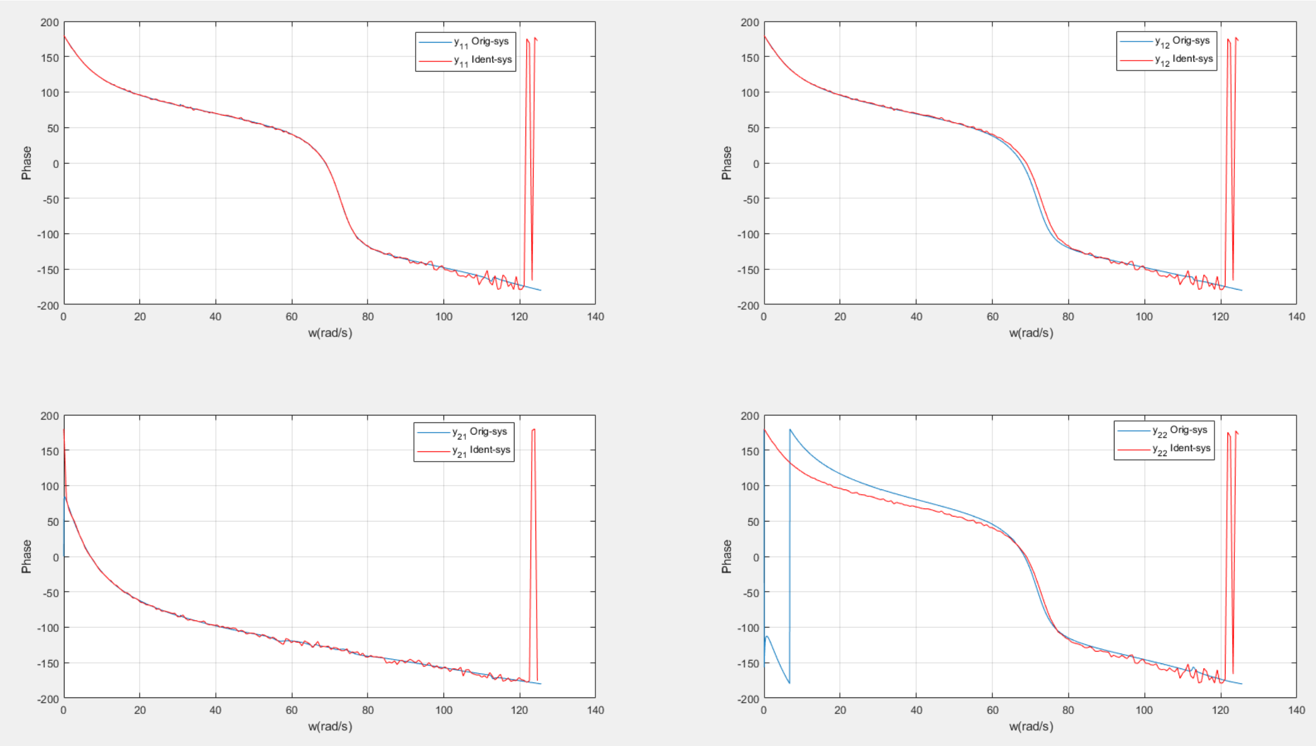


Figure 21. ns = 20, phase

One thing we can conclude is that we cannot use ns = 6 model as our identified model, instead the minimal state is ns = 7, and ns = 10 or 20 are essentially the same as the ns = 7 model, thus we will use ns = 7 model in the following analysis. The MATLAB codes for this part are shown below:

```
clc
close all
clear all
load u_rand.mat
load u1_impulse.mat
load u2_impulse.mat

y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs

y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;

%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));

%%% rescale IO data so that impulse input has magnitude 1
y11 = y11/max(u1);
y12 = y12/max(u2);
```

```matlab
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;
%%
%Hankel
co=['b*';'r*';'g*';'k*'];
N=[20 40 80 100];
for j=1:4
    n=N(j);
    row=[];
    Hankel=[];
    for k=0:n-1
        for i=1:n
            row=[row,[y11(mi+i+k) y12(mi+i+k);y21(mi+i+k) y22(mi+i+k)]];
            if i==n
                Hankel=[Hankel;row];
                row=[];
            end
        end
    end
    s=sort(svd(Hankel),'descend');
    figure(3),
    semilogy(s,co(j,:),'LineWidth',2)
    grid on;
    hold on;
    xlim([0 40]);
    ylim([1e-3 1]);
end
legend('H_2_0','H_4_0','H_8_0','H_1_0_0');
%% %Hankel2
row=[];
Hankel2=[];
n=100;
for k=0:n-1
    for i=1:n
        row=[row,[y11(mi+i+k+1) y12(mi+i+k+1);y21(mi+i+k+1) y22(mi+i+k+1)]];
        if i==n
           Hankel2=[Hankel2;row];
           row=[];
        end
    end
end
%%  set dimension
% ns = 6;
% ns = 7;
% ns = 10;
 ns = 20;
[u,s,v]=svd(Hankel);
u1p = u(:,1:ns);
v1p = v(:,1:ns);
s1p = s(1:ns,1:ns);
On = u1p*s1p;
Cn = v1p';

A = pinv(On)*Hankel2*pinv(Cn);
disp('max(abs(eig(A)))');
max(abs(eig(A)))
B = Cn(:,1:2);
C = On(1:2,:);
D = zeros(2,2);
sys1 = ss(A,B,C,D,'Ts',ts);

[yp,tp] = impulse(sys1,2);
yp = yp*ts;
y11p = yp(:,1,1);
y21p = yp(:,2,1);
y12p = yp(:,1,2);
y22p = yp(:,2,2);
tt=['n_s=',num2str(ns)];

figure(4),
subplot(2,2,1)
plot(tp(2:end),y11p(2:end),'r*',t(mi+1:end),y11(mi+1:end),'b*','LineWidth',2)
grid on;
ylabel('y_1_1 (volts)');
axis([0 2 -0.1 0.1]);
title(tt);

subplot(2,2,3)
plot(tp(2:end),y21p(2:end),'r*',t(mi+1:end),y21(mi+1:end),'b*','LineWidth',2)
grid on;
ylabel('y_2_1(volts)');
axis([0 2 -0.1 0.1]);
title(tt);

subplot(2,2,2)
plot(tp(2:end),y12p(2:end),'r*',t(mi+1:end),y12(mi+1:end),'b*','LineWidth',2)
grid on;
ylabel('y_1_2(volts)');
axis([0 2 -0.1 0.1]);
title(tt);

subplot(2,2,4)
plot(tp(2:end),y22p(2:end),'r*',t(mi+1:end),y22(mi+1:end),'b*','LineWidth',2)
```

```
grid on;
ylabel('y_2_2(volts)');
axis([0 2 -0.1 0.1]);
title(tt);

%%  model frequency response
w0=[0:0.01:20]*2*pi;
I=eye(size(A));
MG(1:2,1:2,1:length(w0))=0;
AG(1:2,1:2,1:length(w0))=0;
for k=1:length(w0)
    w=w0(k);
    G0=C*inv(exp(1i.*ts.*w)*I-A)*B+D;
    MG(:,:,k)=abs(G0);
    AG(:,:,k)=angle(G0)*180/pi;
end

figure(5);
subplot(2,2,1)
plot(w0,reshape(MG(1,1,1:end),1,length(w0)));
grid on;
y11pp=y11(mi:end);
L=length(y11pp);
y11f=fft(y11pp,L);
y11f(1)=y11f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,abs(y11f(1:end/2)),'r');
legend('y_1_1 Orig-sys','y_1_1 Ident-sys');

subplot(2,2,3)
plot(w0,reshape(MG(2,1,1:end),1,length(w0)))
grid on;
y21pp=y21(mi:end);
L=length(y21pp);
y21f=fft(y21pp,L);
y21f(1)=y21f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,abs(y21f(1:end/2)),'r');
legend('y_2_1 Orig-sys','y_2_1 Ident-sys');

subplot(2,2,2)
plot(w0,reshape(MG(1,2,1:end),1,length(w0)))
grid on;
```

```
y12pp=y12(mi:end);
L=length(y12pp);
y12f=fft(y12pp,L);
y12f(1)=y12f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,abs(y12f(1:end/2)),'r');
legend('y_1_2 Orig-sys','y_1_2 Ident-sys');

subplot(2,2,4)
plot(w0,reshape(MG(2,2,1:end),1,length(w0)))
grid on;
y22pp=y22(mi:end);
L=length(y22pp);
y22f=fft(y22pp,L);
y22f(1)=y22f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,abs(y22f(1:end/2)),'r');
legend('y_2_2 Orig-sys','y_2_2 Ident-sys');
%%
figure(6);
subplot(2,2,1)
plot(w0,reshape(AG(1,1,1:end),1,length(w0)))
grid on;
y11pp=y11(mi:end);
L=length(y11pp);
y11f=fft(y11pp,L);
y11f(1)=y11f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,angle(y11f(1:end/2))*180/pi,'r');
legend('y_1_1 Orig-sys','y_1_1 Ident-sys');

subplot(2,2,3)
plot(w0,reshape(AG(2,1,1:end),1,length(w0)))
grid on;
y21pp=y21(mi:end);
L=length(y21pp);
y21f=fft(y21pp,L);
y21f(1)=y21f(1);
f=1/ts/L*(0:(L-1));
hold on;
```

```
ylabel('Phase');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,angle(y21f(1:end/2))*180/pi,'r');
legend('y_2_1 Orig-sys','y_2_1 Ident-sys');

subplot(2,2,2)
plot(w0,reshape(AG(1,2,1:end),1,length(w0)))
grid on;
y12pp=y12(mi:end);
L=length(y12pp);
y12f=fft(y12pp,L);
y12f(1)=y12f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
plot(f(1:end/2)*2*pi,angle(y12f(1:end/2))*180/pi,'r');
legend('y_1_2 Orig-sys','y_1_2 Ident-sys');

subplot(2,2,4)
plot(w0,reshape(AG(2,2,1:end),1,length(w0)))
grid on;
y22pp=y22(mi:end);
L=length(y22pp);
y22f=fft(y22pp,L);
y22f(1)=y22f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Phase');
xlabel('w(rad/s)');
plot(f(1:end/2)*2*pi,angle(y22f(1:end/2))*180/pi,'r');
legend('y_2_2 Orig-sys','y_2_2 Ident-sys');
%% 4
y11pp=y11(mi:end);
y21pp=y21(mi:end);
y12pp=y12(mi:end);
y22pp=y22(mi:end);
u1pp=u1(mi:end);
u2pp=u2(mi:end);
y11f = fft(y11pp)./fft(u1pp);
y21f = fft(y21pp)./fft(u1pp);
y12f = fft(y12pp)./fft(u2pp);
y22f = fft(y22pp)./fft(u2pp);
N = length(y11f);
om = [0:N-1]/(ts*N); %%%% frequency vector in hertz
MGp11=abs(y11f);
AGp11=angle(y11f)*180/pi;
MGp21=abs(y21f);
AGp21=angle(y21f)*180/pi;
```

```
MGp12=abs(y12f);
AGp12=angle(y12f)*180/pi;
MGp22=abs(y22f);
AGp22=angle(y22f)*180/pi;

figure(7);
subplot(2,2,1)
plot(w0,reshape(MG(1,1,1:end),1,length(w0)));
grid on;
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),MGp11(1:end/2),'r');
legend('y_1_1 Orig-sys','y_1_1 Ident-sys');

subplot(2,2,3)
plot(w0,reshape(MG(2,1,1:end),1,length(w0)))
grid on;
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),MGp21(1:end/2),'r');
legend('y_2_1 Orig-sys','y_2_1 Ident-sys');

subplot(2,2,2)
plot(w0,reshape(MG(1,2,1:end),1,length(w0)))
grid on;
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),MGp12(1:end/2),'r');
legend('y_1_2 Orig-sys','y_1_2 Ident-sys');

subplot(2,2,4)
plot(w0,reshape(MG(2,2,1:end),1,length(w0)))
grid on;
hold on;
ylabel('Amplitude');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),MGp22(1:end/2),'r');
legend('y_2_2 Orig-sys','y_2_2 Ident-sys');
%%
figure(8);
subplot(2,2,1)
plot(w0,reshape(AG(1,1,1:end),1,length(w0)))
grid on;
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
```

```
plot(2*pi*om(1:end/2),AGp11(1:end/2),'r');
legend('y_1_1 Orig-sys','y_1_1 Ident-sys');

subplot(2,2,3)
plot(w0,reshape(AG(2,1,1:end),1,length(w0)))
grid on;
y21pp=y21(mi:end);
L=length(y21pp);
y21f=fft(y21pp,L);
y21f(1)=y21f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),AGp21(1:end/2),'r');
legend('y_2_1 Orig-sys','y_2_1 Ident-sys');

subplot(2,2,2)
plot(w0,reshape(AG(1,2,1:end),1,length(w0)))
grid on;
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),AGp11(1:end/2),'r');
legend('y_1_2 Orig-sys','y_1_2 Ident-sys');

subplot(2,2,4)
plot(w0,reshape(AG(2,2,1:end),1,length(w0)))
grid on;
y22pp=y22(mi:end);
L=length(y22pp);
y22f=fft(y22pp,L);
y22f(1)=y22f(1);
f=1/ts/L*(0:(L-1));
hold on;
ylabel('Phase');
xlabel('w(rad/s)')
plot(2*pi*om(1:end/2),AGp11(1:end/2),'r');
legend('y_2_2 Orig-sys','y_2_2 Ident-sys');
```

2. Transmission Zeros of the MIMO model and zeros of each channel

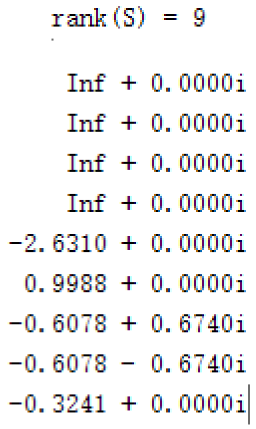
In this part, we will first show that the rank of this matrix is 9:

$$S(\alpha) = \begin{bmatrix} \alpha I - A & -B \\ C & D \end{bmatrix} \in \mathbf{C}^{9\times 9}$$

And then compute the zeros by considering this generalized eigenvalue problem:

$$\begin{bmatrix} A & B \\ -C & -D \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = z \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix}.$$

The results from MATLAB output are shown below:



```
rank(S) = 9

    Inf + 0.0000i
    Inf + 0.0000i
    Inf + 0.0000i
    Inf + 0.0000i
-2.6310 + 0.0000i
 0.9988 + 0.0000i
-0.6078 + 0.6740i
-0.6078 - 0.6740i
-0.3241 + 0.0000i
```

We can see from the results that the rank is indeed 9, and there are indeed five finite transmission zeros and four zeros given as ‘Inf’.

Next, we graph the e-vals and transmission zeros of the ns = 7 model in the complex plane. Note that we denote the e-vals with an ‘x’ and the transmission zeros with a ‘o’. The graph is shown below:

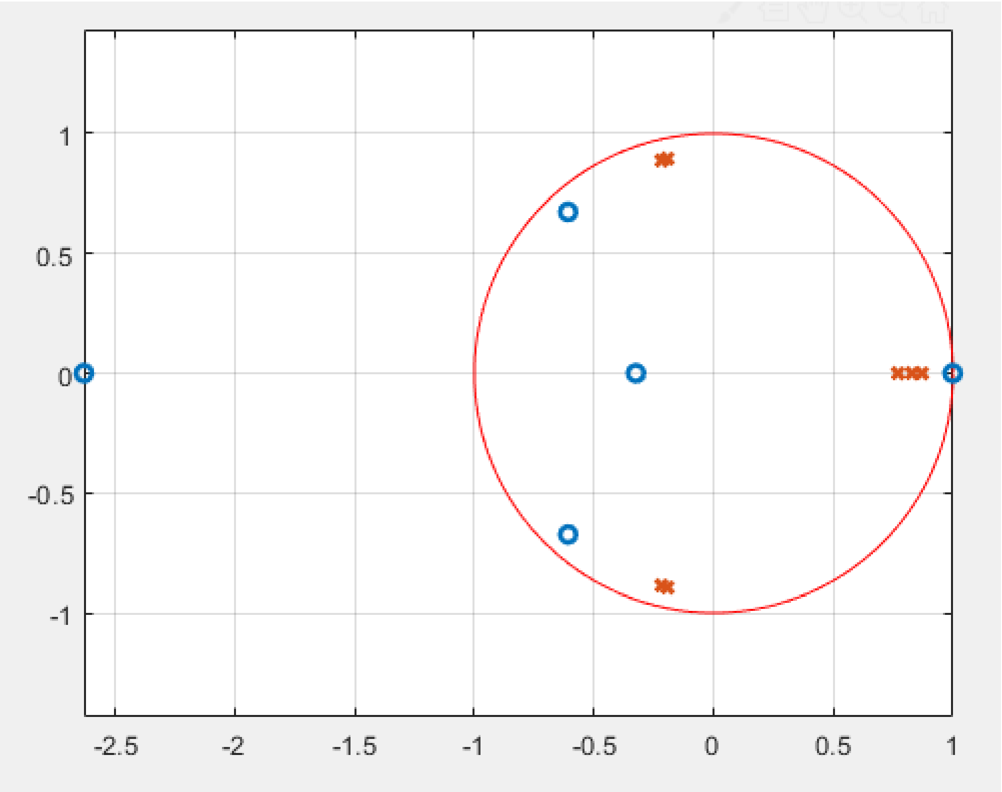
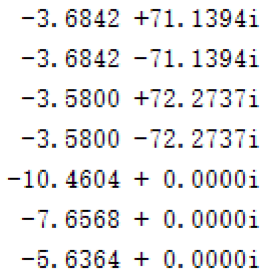


Figure 22. Eigenvalues and Transmission Zeros of the ns = 7 model

Next, a discrete time eigenvalue can be converted into its ‘equivalent’ continuous time eigenvalues by the following equation: $\lambda_d = e^{\lambda cts}$, and the MATLAB output is shown below:



```
 -3.6842 +71.1394i
 -3.6842 -71.1394i
 -3.5800 +72.2737i
 -3.5800 -72.2737i
-10.4604 + 0.0000i
 -7.6568 + 0.0000i
 -5.6364 + 0.0000i
```

From this result, we can see that there are two oscillators in this model, and their approximate natural frequencies are 71.1394 and 72.2737. Recall the frequency response graph from task 1, given these two frequencies, we can see that there are peaks around these frequencies three of the four channels.

Now, for the ns = 7 model, we will graph the eigenvalues and transmission zeros for each individual channel, and also calculate the Hankel singular values for each channel. The graphs are shown below:

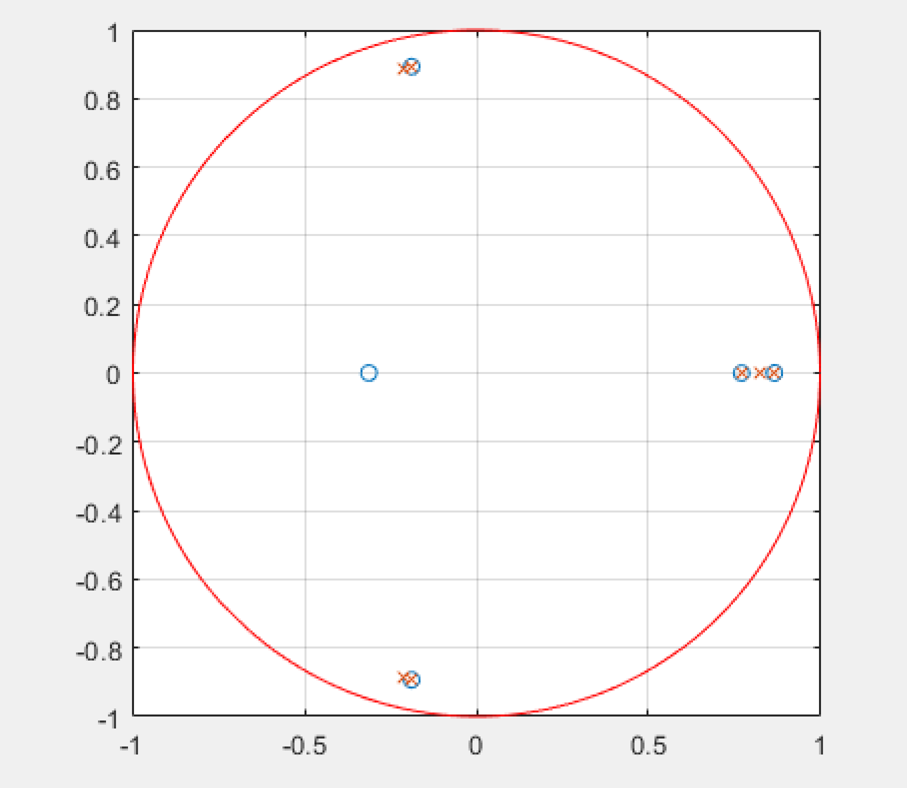


Figure 23. Eigenvalues and Transmission Zeros of the 11 channel.

The corresponding Hankel singular values are: 0.1944, 0.1586, 0.1195, 0.0046, 0.0035, 0.0013, 0.0008, 0.0006, 0.0006, 0.0006, 0.0005, 0.0003, 0.0003, 0.0003, 0.0003, 0.0002, 0.0002, 0.0001, 0.0001, 0.0000.

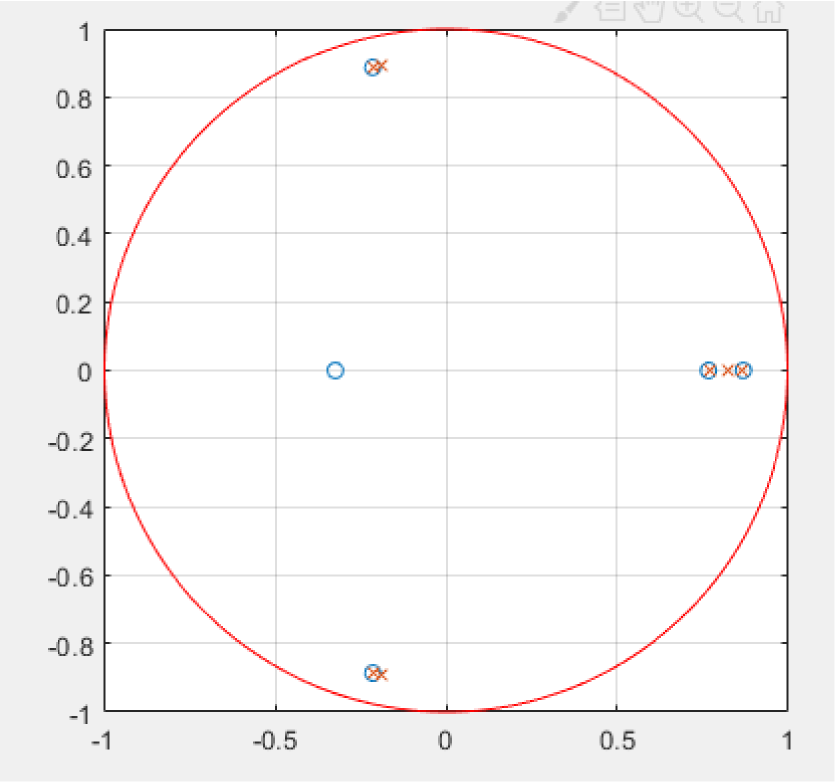


Figure 24. Eigenvalues and Transmission Zeros of the 12 channel.

The corresponding Hankel singular values are: 0.1961, 0.1593, 0.1195, 0.0046, 0.0037, 0.0012, 0.0008, 0.0004, 0.0003, 0.0003, 0.0003, 0.0003, 0.0003, 0.0002, 0.0002, 0.0002, 0.0002, 0.0002, 0.0001, 0.0001.

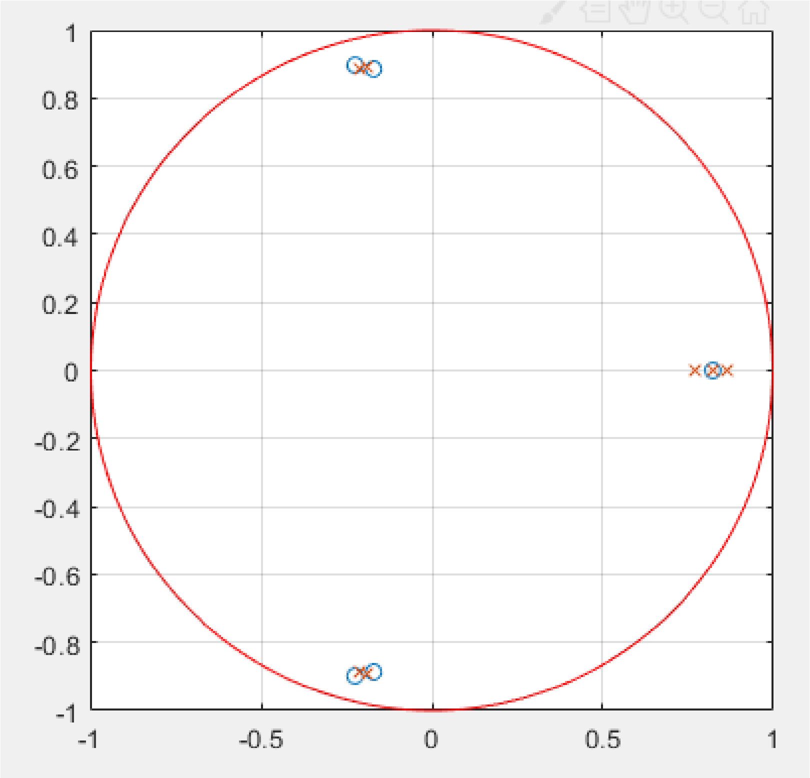


Figure 25. Eigenvalues and Transmission Zeros of the 21 channel.

The corresponding Hankel singular values are: 0.0961, 0.0796, 0.0013, 0.0011, 0.0011, 0.0010, 0.0010, 0.0009, 0.0007, 0.0007, 0.0005, 0.0005, 0.0005, 0.0004, 0.0004, 0.0003, 0.0003, 0.0002, 0.0002, 0.0000.

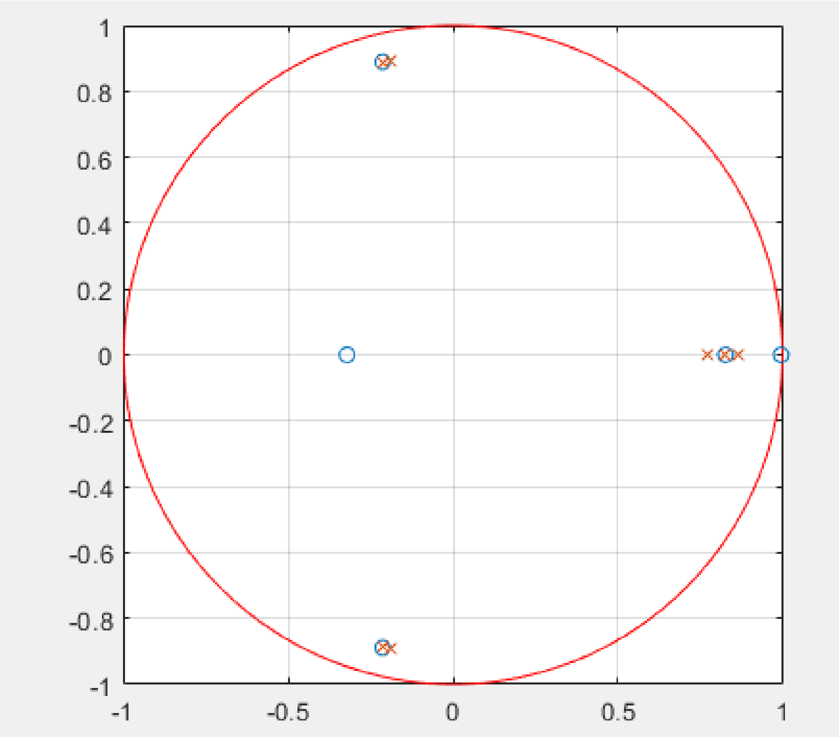


Figure 26. Eigenvalues and Transmission Zeros of the 22 channel.

The corresponding Hankel singular values are: 0.1614, 0.1601, 0.0769, 0.0752, 0.0043, 0.0035, 0.0011, 0.0010, 0.0009, 0.0007, 0.0006, 0.0005, 0.0005, 0.0004, 0.0004, 0.0004, 0.0002, 0.0002, 0.0002, 0.0000.

Looking at these graphs and recall the channel frequency response graphs from task 1, considering the pole-zero cancellation, we can see that for example, there is no oscillator for channel 21, and this also can be seen from the frequency response graph of channel 21, which has no peak. On the other hand, the other 3 channels have one oscillator, which also can be seen from the frequency response graphs, which have one peak at the frequency around 71.1394 or 72.2737.

Now, we will consider the ns = 8 model, and plot another set of pole-zero plot for each channel. The graphs are shown below:

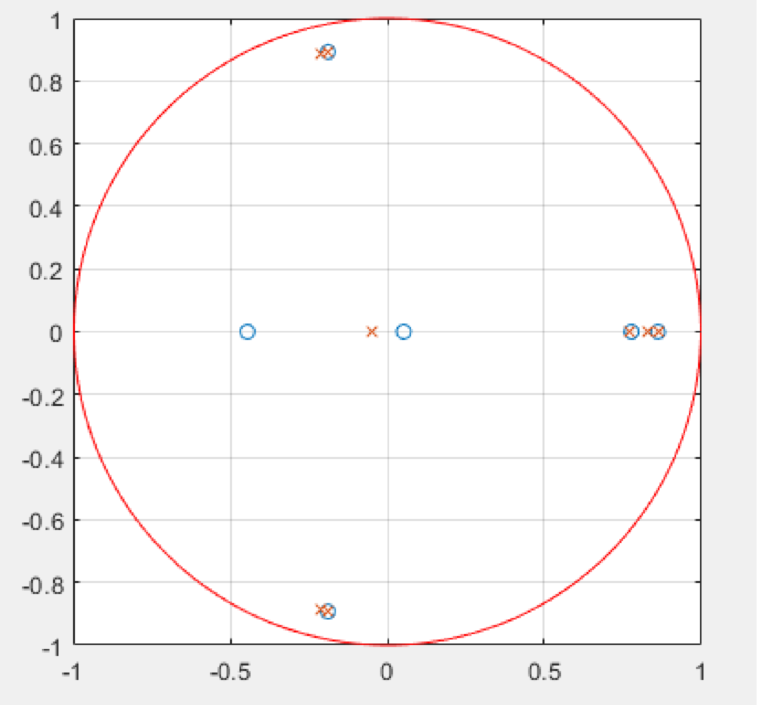


Figure 27. ns = 8 model, 11 channel.

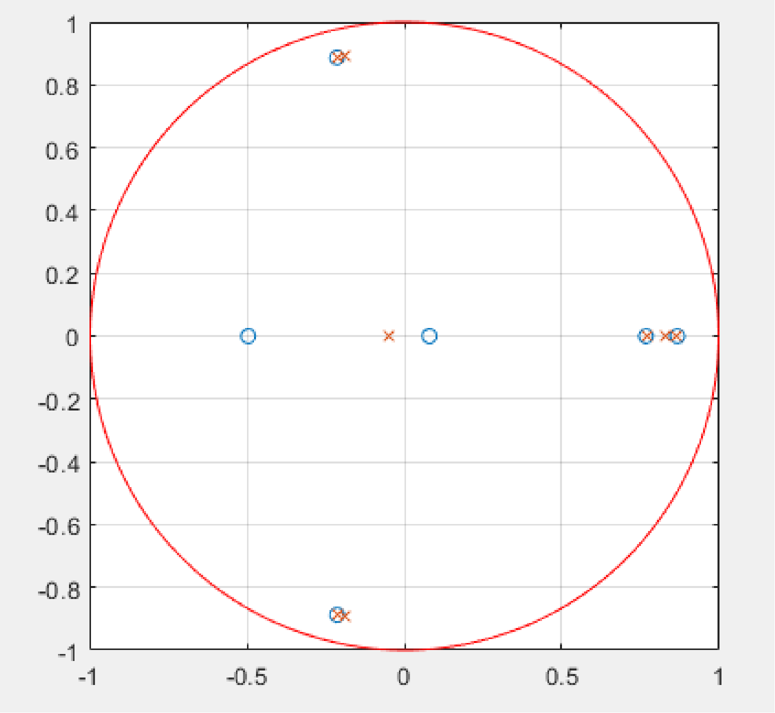


Figure 28. ns = 8 model, 12 channel.

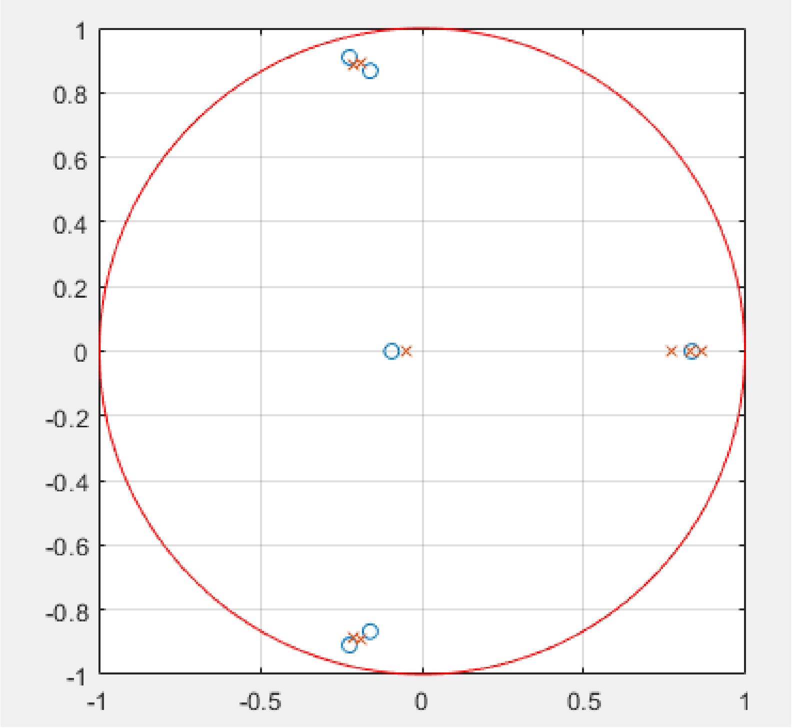


Figure 29. ns = 8 model, 21 channel.

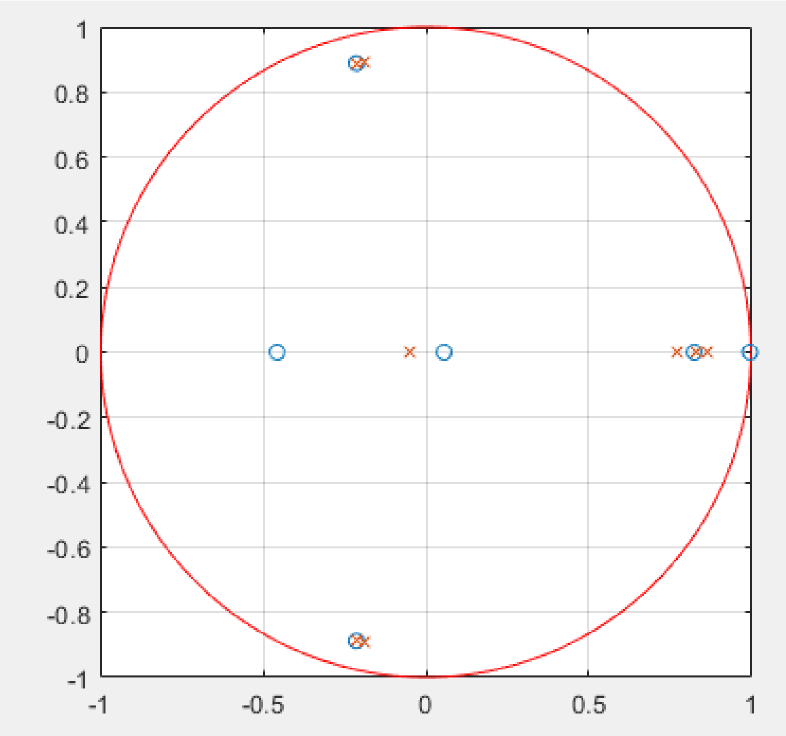


Figure 30. ns = 8 model, 22 channel.

We can see that when using the ns = 8 model, the added pole is accompanied by a zero in proximity, and in fact this occurs for all channels, thus we conclude that the ‘extra’ state dimension does not have too much impact on the I/O properties of the system when compared to the ns = 7 model.

The MATLAB codes for this part are shown below:

```
clc
close all
clear all
load u_rand.mat
load u1_impulse.mat
load u2_impulse.mat

y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs

y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;

%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));

%%% rescale IO data so that impulse input has magnitude 1
```

```
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;

%%
%Hankel
row1 =[];
Hankeln=[];
n=100;
for k=0:n-1
   for i=1:n
       row1 =[row1,[y11(mi+i+k) y12(mi+i+k);y21(mi+i+k) y22(mi+i+k)]];
       if i==n
         Hankeln =[Hankeln;row1];
         row1 =[];
       end
   end
end
%% %Hankelp
row2 =[];
Hankelnp=[];
n=100;
for k=0:n-1
   for i=1:n
       row2 =[row2,[y11(mi+i+k+1) y12(mi+i+k+1);y21(mi+i+k+1) y22(mi+i+k+1)]];
       if i==n
         Hankelnp=[Hankelnp;row2];
         row2 =[];
       end
   end
end
[u,s,v]=svd(Hankeln);
%%  set dimension
% ns=7;
ns=8;
u1p=u(:,1:ns);
v1p=v(:,1:ns);
s1p=s(1:ns,1:ns);
On=u1p*s1p;
Cn=v1p';
A=pinv(On)*Hankelnp*pinv(Cn);
B=Cn(:,1:2);
C=On(1:2,:);
```

```
D=zeros(2,2);
sys1=ss(A,B,C,D,'Ts',ts);


%%  task2
a = 0.1;
I = eye(size(A));
S = [a*I-A -B;C D];
display(['rank(S) = ', num2str(rank(S))]);
AA=[A B;-C -D];
BB=[I zeros(size(B));zeros(size(C)) zeros(size(D))];
[V,D]=eig(AA,BB);
g_evals = sort(diag(D),'descend')
XD=real(g_evals(5:end));
YD=imag(g_evals(5:end));
abs_g_evals = abs(g_evals)
evals_of_A = eig(A);
display(evals_of_A);
xea=real(evals_of_A);
yea=imag(evals_of_A);
%%  %2
angle=0:0.01:2*pi;
R=1;
x=R*cos(angle);
y=R*sin(angle);
figure(1),
plot(x,y,'r');
axis equal
hold on;
grid on;
plot(XD,YD,'o','linewidth',2);
plot(xea,yea,'x','linewidth',2);

%% %3
evals_c=(1/ts)*log(evals_of_A);
display(evals_c);

%% %4
% AA=[A B(:,1);-C(1,:) -0]; %% 11
% AA=[A B(:,2);-C(1,:) -0]; %% 12
% AA=[A B(:,1);-C(2,:) -0]; %% 21
 AA=[A B(:,2);-C(2,:) -0]; %% 22
BB=[I zeros(size(B(:,1)));zeros(size(C(1,:))) 0];
[V,D]=eig(AA,BB);
evals_channel = sort(diag(D),'descend')
XD=real(evals_channel(4:end));
YD=imag(evals_channel(4:end));
abs_evals_channel = abs(evals_channel);
display(abs_evals_channel);
```

```
%% Hankel
row = [];
Hankel = [];
n = 20;
for k=0:n-1
    for i=1:n
        row = [row,y22(mi+i+k)];
        if i==n
            Hankel = [Hankel;row];
            row = [];
        end
    end
end
[u,s,v] = svd(Hankel);
s_diag = diag(s);
display(s_diag);
%%  %circle
angle=0:0.01:2*pi;
R=1;
x=R*cos(angle);
y=R*sin(angle);
figure(2),
plot(x,y,'r');
axis square
hold on;
grid on;
plot(XD,YD,'o');
plot(xea,yea,'x');
```

3. Block Diagram from Analysis of individual channels

From the previous parts, we know that there are two oscillators and three distinct real poles in the system. In this part, we will first define the two complex conjugate pair of poles as ‘OSC1’ and ‘OSC2’, and the three distinct real poles as ‘LP1’, ‘LP2’, and ‘LP3’. Labels on the pole-zero plot is shown below:

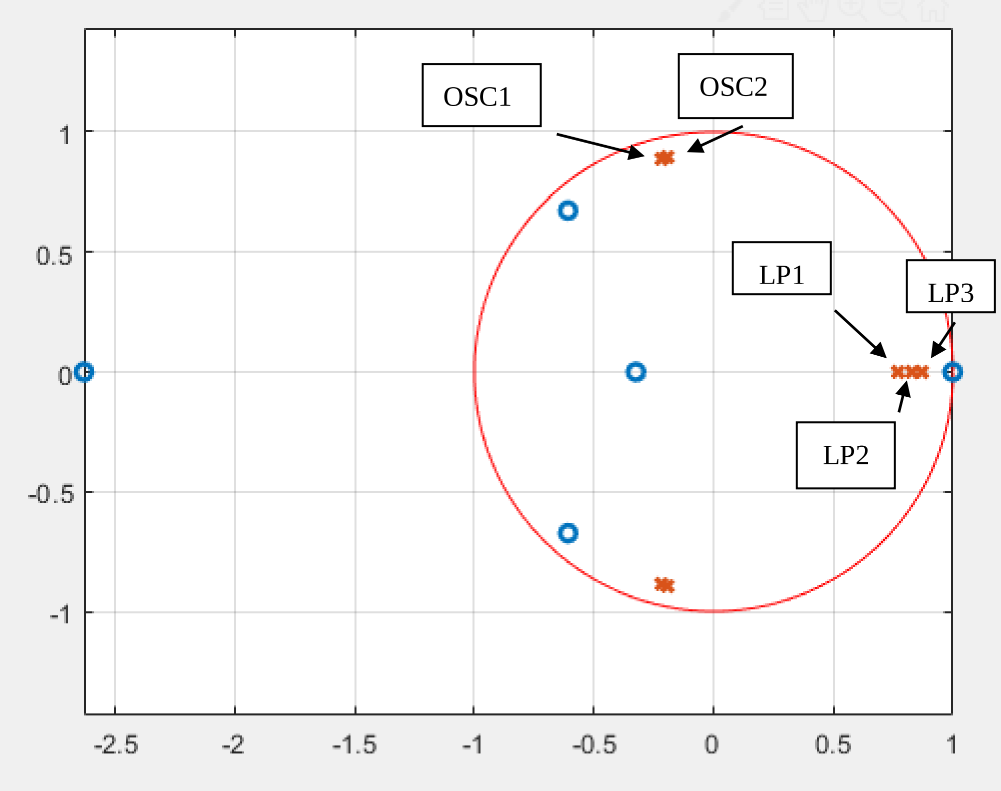


Figure 31. Labels of the OSCs and LPs

Given these labels, and based on the results we obtained from the previous part that which poles are not canceled by zeros in the individual channels, we can determine how these oscillators and low-pass filters are ‘connected’ for each channel:

u1 → OSC1 → LP2 → y11
u2 → OSC2 → LP2 → y12
u1 → LP1 → LP3 → y21
u2 → OSC2 → LP1 → LP3 → y22

Thus, the block diagram can then be drawn as follow:

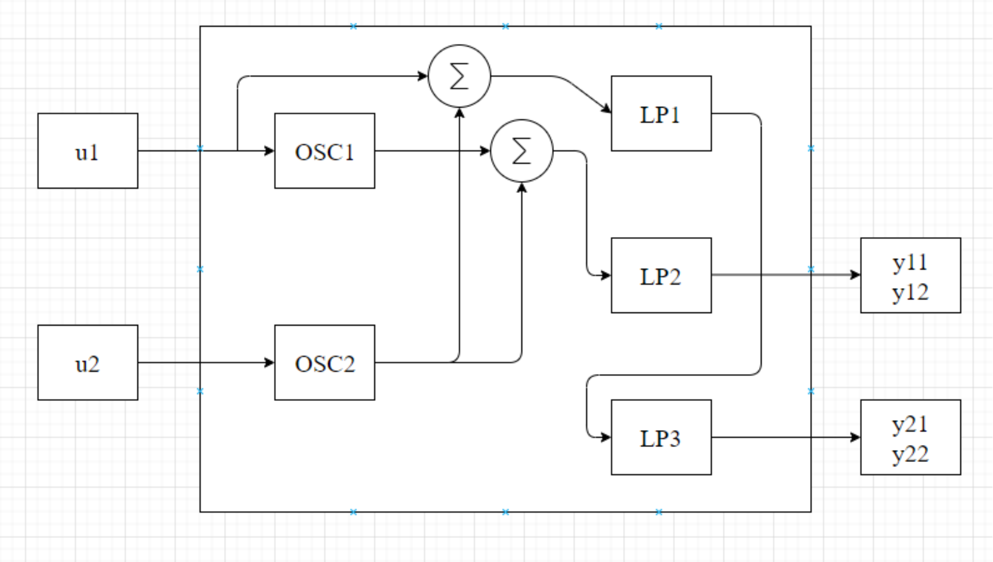
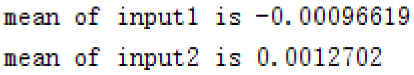


Figure 32. Block Diagram for the two-input-two-output system

The basic idea of this block diagram is that by using the sums, we can ensure that when only u1 is ‘on’, we get y11 and y21, on the other hand when only u2 is ‘on’, we get y12 and y22. However, one thing to be noticed is that we cannot uniquely determine the topology for this block diagram, i.e. we don’t know the sequence. For example, we cannot tell whether it is the oscillator first or the low-pass filter first. Furthermore, we only know that it is either LP1 or LP3 can be paired with the zero at s = 1 to make a high-pass filter, but we cannot determine which one.

4. Impulse Response Identification from White Noise Inputs

In this part, we first verify that each input sequence is approximately zero mean. From the MATLAB outputs, which is shown below, we see that they are indeed approximately zero mean.



```
mean of input1 is -0.00096619
mean of input2 is 0.0012702
```

And then let us consider this function:

$$R_{\mathbf{uu}}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{u}_{k+q}\mathbf{u}_q^T \in \mathbf{R}^{n\times n}$$

This is called the Auto-correlation of u. Now, we need to use this function to determine and graph the four entries of R_{uu} for indices k from -200 to 200. The graphs are shown below:

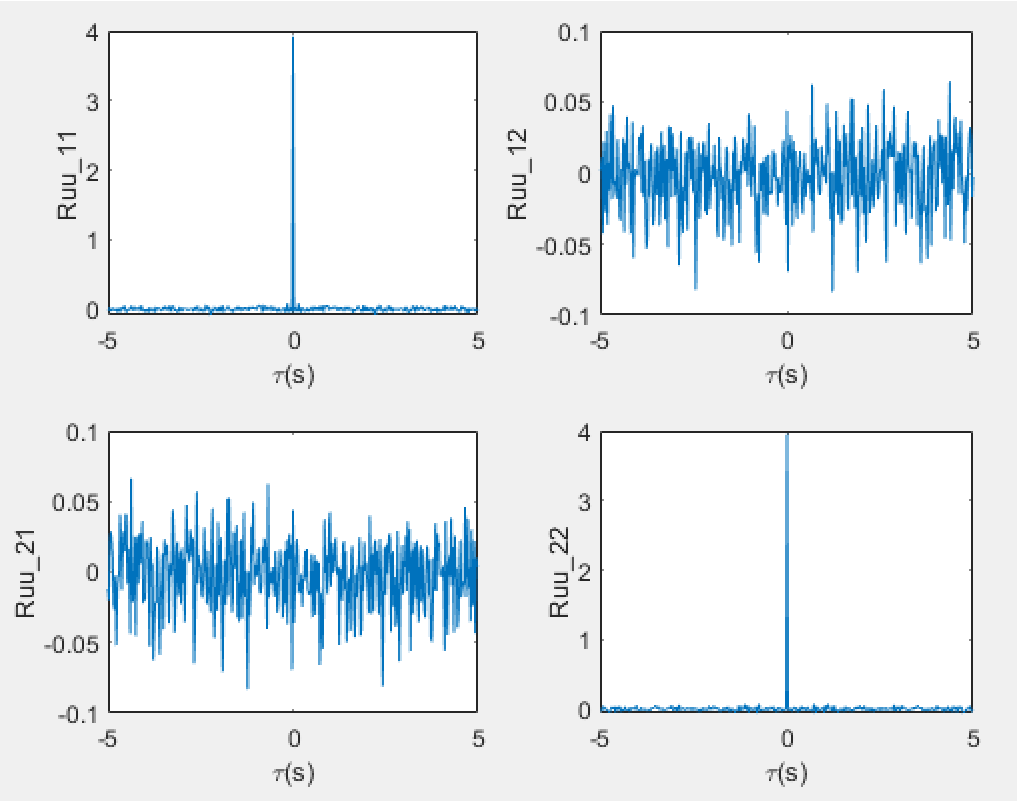
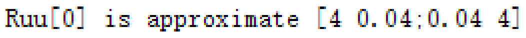


Figure 33. Values for each of the 4 entries in R_{uu}

And then, we need to show that:

$$R_{\mathbf{uu}}[0] \approx \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

This can be easily verified by using the formula presented above, and the MATLAB result is:



```
Ruu[0] is approximate [4 0.04;0.04 4]
```

Thus, we say that the variance of each input channel is 4, not 1.

Next, we need to consider this function:

$$R_{\mathbf{yu}}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{y}_{k+q}\mathbf{u}_q^T \in \mathbf{R}^{m\times n}.$$

This is called the cross-correlation. Now, we need to estimate $R_{yu}[k]$ for τ from -0.2 to 2 second, and then graph the first column of R_{yu} normalized by the variance of the first channel of u, and the second column of R_{yu} normalized by the variance of the second channel of u. The graphs are shown below:

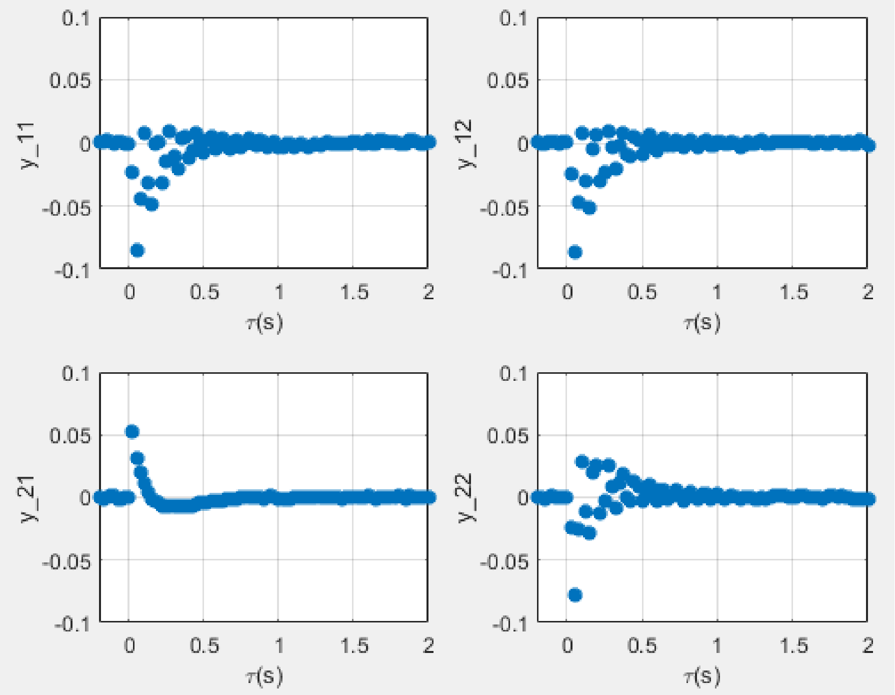


Figure 34. $R_{yu}[k]$ for each channel

Recall the experimental impulse response we obtained from the pulse response experiments, we can see that this result is very close to the experimental impulse response. The MATLAB codes for this part are shown below:

```
clc
close all
clear all
load u_rand.mat
y1 = u_rand.Y(3).Data;
y2 = u_rand.Y(4).Data;
u1 = u_rand.Y(1).Data;
u2 = u_rand.Y(2).Data;
ts = 1/40;
N = length(y1);
t = [0:N-1]*ts - 1;

%% 1
display(['mean of input1 is ', num2str(mean(u1))])
display(['mean of input2 is ', num2str(mean(u2))])

%% 2
u=[u1;u2];
Ruu=zeros(2);
```

```
Ruus=zeros([2 2 401]);
lag=200;
j=1;
for la=-200:200
    Ruu=zeros(2);
    Ruu_temp=zeros(2);
    for q=201:N-200
        Ruu_temp=u(:,la+q)*u(:,q)';
        Ruu=Ruu+Ruu_temp;
    end
    Ruu=Ruu/N;
    Ruus(:,:,j)=Ruu;
    j=j+1;
end
la=(-200:200)*ts;
figure(1),
subplot(221);
plot(la,reshape(Ruus(1,1,:),1,length(la)));
xlabel('\tau(s)');
ylabel('Ruu\_11');

subplot(222);
plot(la,reshape(Ruus(1,2,:),1,length(la)));
xlabel('\tau(s)');
ylabel('Ruu\_12');

subplot(223);
plot(la,reshape(Ruus(2,1,:),1,length(la)));
xlabel('\tau(s)');
ylabel('Ruu\_21');

subplot(224);
plot(la,reshape(Ruus(2,2,:),1,length(la)));
xlabel('\tau(s)');
ylabel('Ruu\_22');

%% 3
u=[u1;u2];
Ruu=zeros(2);
for q=1:N
    Ruu_temp=u(:,q)*u(:,q)';
    Ruu=Ruu+Ruu_temp;
end
Ruu=Ruu/N;
display(['Ruu[0] is approximate ', mat2str(Ruu,1)]);

%% 4
u=[u1;u2];
y=[y1;y2];
```

```
Ryu=zeros(2);
lag=0.2/ts;
Ryus=zeros([2 2 (0.2+2)/ts+1]);
j=1;
for la=-lag:lag*10
   Ryu=zeros(2);
   Ryu_temp=zeros(2);
   for q=lag+1:N-2/ts
      Ryu_temp=y(:,la+q)*u(:,q)';
      Ryu=Ryu+Ryu_temp;
   end
   Ryu=Ryu/N;
   Ryus(:,:,j)=Ryu;
   j=j+1;
end
la=(-lag:lag*10)*ts;
Ryusp1=Ryus(:,1,:)/var(u1);
Ryusp2=Ryus(:,2,:)/var(u2);
y1pp=reshape(Ryusp1,2,89);
y2pp=reshape(Ryusp2,2,89);

figure(2)
subplot(221)
plot(la,y1pp(1,:),'*','linewidth',2);
xlabel('\tau(s)');
ylabel('y\_11');
grid on
axis([-0.2 2 -0.1 0.1]);

subplot(223)
plot(la,y1pp(2,:),'*','linewidth',2);
grid on
axis([-0.2 2 -0.1 0.1]);
xlabel('\tau(s)');
ylabel('y\_21');

subplot(222)
plot(la,y2pp(1,:),'*','linewidth',2);
xlabel('\tau(s)');
ylabel('y\_12');
grid on
axis([-0.2 2 -0.1 0.1]);

subplot(224)
plot(la,y2pp(2,:),'*','linewidth',2);
grid on
axis([-0.2 2 -0.1 0.1]);
xlabel('\tau(s)');
ylabel('y\_22');
```

5. H_2 Norm Analysis of the Identified Model

In this part, we will use the ‘u_rand.mat’ data. First thing to notice is that since we have already verified that $R_{uu}[0]$ is approximately [4 0;0 4], this means that the input channels are not unit variance, so we scale the input and output data by a factor of 2 so that each input channel now has unit variance.

Now, we first compute the RMS value of the scaled output data y. This can be computed by considering the equation shown below:

$$\|\mathbf{y}\|_{\text{RMS}}^2 = \lim_{p\to\infty} \frac{1}{2p} \sum_{k=-p}^{p} \text{tr}\left(\mathbf{y}_k \mathbf{y}_k^T\right) = \text{tr}\left(\lim_{p\to\infty} \frac{1}{2p} \sum_{k=-p}^{p} \mathbf{y}_k \mathbf{y}_k^T\right) = \text{tr}\, R_{\mathbf{yy}}[0].$$

This means that we need to first compute $R_{yy}[0]$, which is given by this equation:

$$\text{Auto-correlation of } \mathbf{y}: \ R_{\mathbf{yy}}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{y}_{k+q} \mathbf{y}_q^T \in \mathbf{R}^{m\times m},$$

And then by taking the square root of the trace of $R_{yy}[0]$, we will get the desired result. In fact, this is very easy to implement in MATLAB, and the result is **0.2237**.

Next, we will consider $\| P \|_{H2}$, where P is the 7-state model we have identified via the Hankel Matrix. This can be computed by two different but similar equations:

$$= \text{tr}\left(B^T G_o(\infty) B\right),$$

$$= \text{tr}\left(C G_c(\infty) C^T\right)$$

Notice that Go is the observability gramian and Gc is the controllability gramian. Both gramians can be calculated from the Lyapunov equations:

$$A^T G_o(\infty) A - G_o(\infty) = -C^T C$$
$$A G_c(\infty) A^T - G_c(\infty) = -B B^T.$$

And in fact, by using the built-in ‘lyap’ function in MATLAB, we can get Go and Gc easily. Then by taking the square root of the trace, we can get the desired result. When using Go, the computed result is ‘**0.22968**’, and when using Gc, the computed result is ‘**0.22968**’. Now, $\| P \|_{H2}$ can also be estimated only by using the experimental pulse response data. Consider this equation:

$$\|P\|_{\mathcal{H}_2}^2 = \sum_{k=1}^{\infty} \|h_k\|_F^2,$$
$$= \sum_{k=1}^{\infty} \text{tr}\left(h_k^T h_k\right)$$

Again, we first sum up the traces, and then by taking the square root of the sum, we can get the desired result. The computed result by using this equation is ‘**0.22983**’.

Thus, we can see that we get essentially the same values. The MATLAB codes for this part are shown below:

```
clc
close all
clear all
load u_rand.mat
load u1_impulse.mat
load u2_impulse.mat

y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs

y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;

%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));

%%% rescale IO data so that impulse input has magnitude 1
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;

%%
%Hankel
row1 = [];
Hankel1 = [];
n = 100;
for k = 0:n-1
    for i = 1:n
       row1 = [row1,[y11(mi+i+k) y12(mi+i+k);y21(mi+i+k) y22(mi+i+k)]];
       if i == n
          Hankel1 =[Hankel1;row1];
          row1 = [];
       end
    end
end

%% %Hankel2
```

```
row = [];
Hankel2 = [];
n = 100;
for k = 0:n-1
   for i = 1:n
      row = [row,[y11(mi+i+k+1) y12(mi+i+k+1);y21(mi+i+k+1) y22(mi+i+k+1)]];
      if i == n
         Hankel2 = [Hankel2;row];
         row = [];
      end
   end
end
[u,s,v] = svd(Hankel1);
%% compute system
ns = 7;
u1p = u(:,1:ns);
v1p = v(:,1:ns);
s1p = s(1:ns,1:ns);
On = u1p*s1p;
Cn = v1p';
A = pinv(On)*Hankel2*pinv(Cn);
B = Cn(:,1:2);
C = On(1:2,:);
D = zeros(2,2);
%% RMS of output y
y1 = (u_rand.Y(3).Data)/2;
y2 = (u_rand.Y(4).Data)/2;
u1 = u_rand.Y(1).Data;
u2 = u_rand.Y(2).Data;
ts = 1/40;
N = length(y1);
t = [0:N-1]*ts - 1;

y = [y1;y2];
Ryy = zeros(2);
for q=1:N
   Ryy_temp = y(:,q)*y(:,q)';
   Ryy = Ryy + Ryy_temp;
end
Ryy = Ryy/N;

display(num2str(sqrt(trace(Ryy))),'y_rms is approximately: ');

%% equation (9) and (10)
C_lyap = C'*C;
B_lyap = B*B';
Go = dlyap(A',A,C_lyap);
Gc = dlyap(A,A',B_lyap);
```

```
P_1 = sqrt(trace(B'*Go*B));
P_2 = sqrt(trace(C*Gc*C'));

display(num2str(P_1),'using (9): ');
display(num2str(P_2),'using (10): ');

%% equation (8)
trace_sum = 0;
for k = 1:401
    h_k = [y11(k),y12(k);y21(k),y22(k)];
    trace_temp = trace(h_k'*h_k);
    trace_sum = trace_sum + trace_temp;
end
P_3 = sqrt(trace_sum);
display(num2str(P_3),'using (8): ');
```

6. H∞ norm analysis of the identified model

First, we need to write a MATLAB function that takes in the state-space matrices of a continuous-time system, upper and lower bounds for the γ search, and a tolerance, and then returns the norm and the approximate frequency at which the maximum gain is achieved.

To write this function, we need to first construct some matrices: D_γ and $A_{clp}(\gamma)$, which are defined as follows:

$$\underbrace{(\gamma^2 I - D^* D)}_{D_\gamma}$$

$$\begin{aligned} A_{clp}(\gamma) &= \begin{bmatrix} A & 0 \\ C^*C & -A^* \end{bmatrix} + \begin{bmatrix} B \\ C^*D \end{bmatrix} \mathbf{u} \\ &= \begin{bmatrix} A & 0 \\ C^*C & -A^* \end{bmatrix} + \begin{bmatrix} B \\ C^*D \end{bmatrix} D_\gamma^{-1} \begin{bmatrix} D^*C & -B^* \end{bmatrix} \\ &= \begin{bmatrix} A + BD_\gamma^{-1}D^*C & -BD_\gamma^{-1}B^* \\ C^*C + C^*DD_\gamma^{-1}D^*C & -A^* - C^*DD_\gamma^{-1}B^* \end{bmatrix} \end{aligned}$$

And then we apply the bisection search algorithm to search for γ, and once we find γ, it will be the norm we are computing, and the frequency can be determined by maximum imagine part eigenvalues of $A_{clp}(\gamma)$. Detailed implementation of this algorithm can be easily seen from the MATLAB codes shown below:

```
function [Hinf,winf] = Inf_norm_continuous(A,B,C,D,gamma_l,gamma_u,tol)
    while(gamma_u - gamma_l) > tol
    gamma = (gamma_l + gamma_u)/2;
    D_gamma = (gamma^2)*eye(size(D'*D,1)) - D'*D;
    A1 = A+B*inv(D_gamma)*D'*C;
    D1 = D*inv(D_gamma)*D';
    A_clp = [A1,(B*inv(D_gamma)*B');-C'*(eye(size(D1))+D1)*C, -A1'];
    eig_test = eig(A_clp);
    if (min(abs(real(eig_test))) < 1e-10)
```

```
        gamma_l = gamma;
    else
        gamma_u = gamma;
    end
    Hinf = gamma;
    winf = max(imag(eig_test));
    end
end
```

Once we have implemented the function for continuous-time system, we can then modify the function to compute the discrete-time case. Consider the (A,B,C,D) matrices for the discrete-time case, these matrices can be ‘converted’ into the continuous-time case by considering the equations below:

$$
\begin{aligned}
A_c &= -(I+A)^{-1}(I-A) \\
B_c &= \sqrt{2}(I+A)^{-1}B \\
C_c &= \sqrt{2}C(I+A)^{-1} \\
D_c &= D - C(I+A)^{-1}B
\end{aligned}
$$

We can then use these ‘continuous-time case’ matrices as input matrices to the already implemented continuous-time function. The output norm is then the desired value. However, the frequency needs some special treatment. Let ‘*v*’ be the output frequency from the continuous-time function, then the desired discrete-time frequency ‘ω’ can be calculated by considering this equation:

$$e^{j\omega t_s} = \frac{1+j\nu}{1-j\nu}$$

The implementation of this discrete-time function can be easily done, and the MATLAB codes are shown below:

```
function [Hinf,winf3] = Inf_norm_discrete(A,B,C,D,wl,wu,tol,ts)
Ac = -(inv((eye(length(A))+A))) * (eye(length(A))-A);
Bc = sqrt(2) * (inv((eye(length(A))+A))) * B;
Cc = sqrt(2) * C * (inv((eye(length(A))+A)));
Dc = D - C * (inv((eye(length(A))+A))) * B;
[Hinf,winf1] = Inf_norm_continuous(Ac,Bc,Cc,Dc,wl,wu,tol);
winf2 = (log((1+(1i*winf1))/(1-(1i*winf1))))/(1i*ts);
winf3 = real(winf2);
end
```

Now, we apply this discrete-time function to actually compute the H_∞ norm and frequency of the identified system. The computed norm is **‘0.4693’**, and the computed frequency is **‘71.7152’**. And then, we compute the discrete-time frequency response in the interval [0, ω_{nyq}], and then plot the singular values of the frequency response on this frequency grid. Note that ω_{nyq} = 20Hz, so it is about 125 rad/s. And then, we plot the singular values computed from the empirical frequency response data. The graph is shown below:

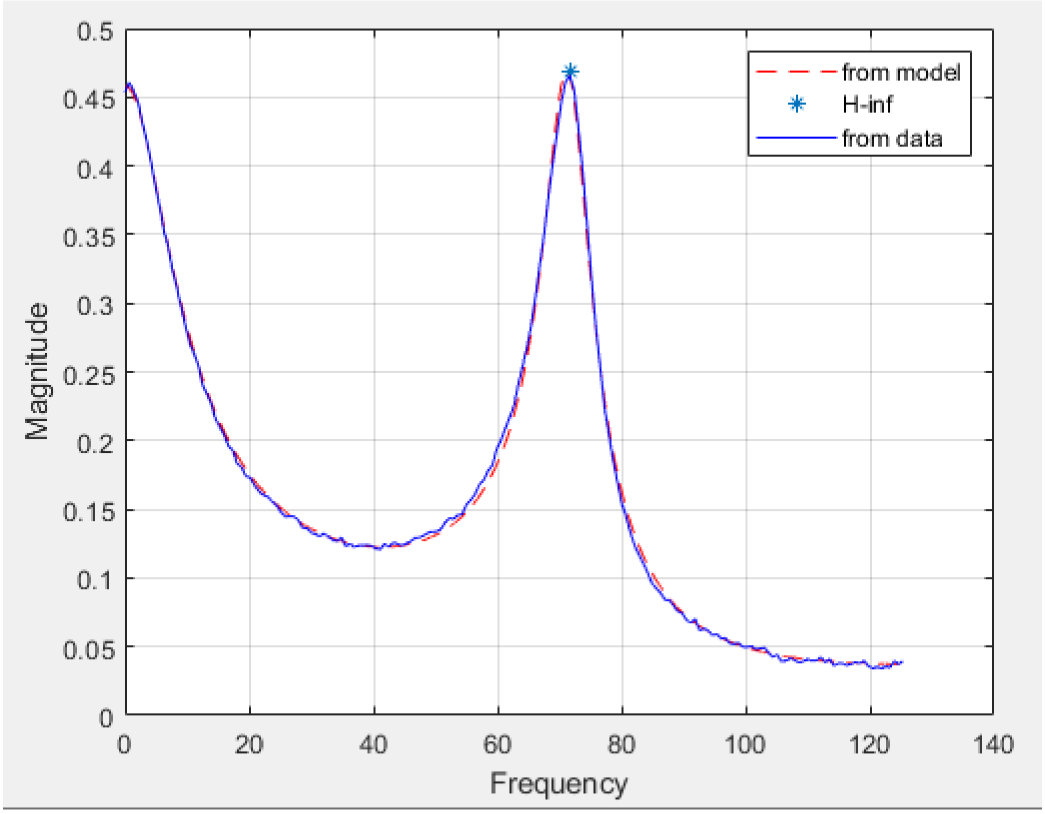


Figure 35. Singular values and H∞ norm

We can clearly see that two sets of singular values are very close to each other, and in fact the H∞ norm indicates the magnitude and corresponding frequency where the maximum singular value achieves its largest value. The MATLAB codes for this part are shown below:

```
clc
close all
clear all
%%task6
load u1_impulse.mat
load u2_impulse.mat

y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs

y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;

%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));

%%% rescale IO data so that impulse input has magnitude 1
```

```
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;

y11pp=y11(mi:end);
y21pp=y21(mi:end);
y12pp=y12(mi:end);
y22pp=y22(mi:end);
u1pp=u1(mi:end);
u2pp=u2(mi:end);
y11f = fft(y11pp)./fft(u1pp);
y21f = fft(y21pp)./fft(u1pp);
y12f = fft(y12pp)./fft(u2pp);
y22f = fft(y22pp)./fft(u2pp);
N = length(y11f);
om = [0:N-1]/(ts*N);
%%
%Hankel
row1 = [];
Hankel1 = [];
n = 100;
for k = 0:n-1
    for i = 1:n
       row1 = [row1,[y11(mi+i+k) y12(mi+i+k);y21(mi+i+k) y22(mi+i+k)]];
       if i == n
         Hankel1 =[Hankel1;row1];
         row1 = [];
       end
    end
end
%% %Hankel2
row = [];
Hankel2 = [];
n = 100;
for k = 0:n-1
    for i = 1:n
       row = [row,[y11(mi+i+k+1) y12(mi+i+k+1);y21(mi+i+k+1) y22(mi+i+k+1)]];
       if i == n
         Hankel2 = [Hankel2;row];
         row = [];
       end
    end
end
```

```
[u,s,v] = svd(Hankel1);
%%  compute system
ns = 7;
u1p = u(:,1:ns);
v1p = v(:,1:ns);
s1p = s(1:ns,1:ns);
On = u1p*s1p;
Cn = v1p';
A = pinv(On)*Hankel2*pinv(Cn);
B = Cn(:,1:2);
C = On(1:2,:);
D = zeros(2,2);
wl = 0;
wu = 125;
tol = 1e-15;
ts = 1/40;
%3
[Hinf,winf]=Inf_norm_discrete(A,B,C,D,wl,wu,tol,ts)
%4
u = 1;
for w = wl:0.01:wu
    Pw = C*inv((exp(1i*w*ts)*eye(length(A))-A))*B+D;
    S1(:,u) = svd(Pw);
    u = u+1;
end

for i = 1:1:181
    hk = [y11f(i) y12f(i);y21f(i) y22f(i)];
    S2(:,i) = svd(hk);
    i = i + 1;
end
w=wl:0.01:wu;
figure(1)
plot(w,S1(1,:),'--r');
grid on;
hold on;
plot(winf,Hinf,'*');
xlabel('Frequency');
ylabel('Magnitude');
hold on;
plot(2*pi*om(1:181),S2(1,:),'b');
legend('from model','H-inf','from data');
```