MAE M270A Project

Ganesha Durbha
Jared Jonas

December 18, 2020

Table of Contents

List of Figures

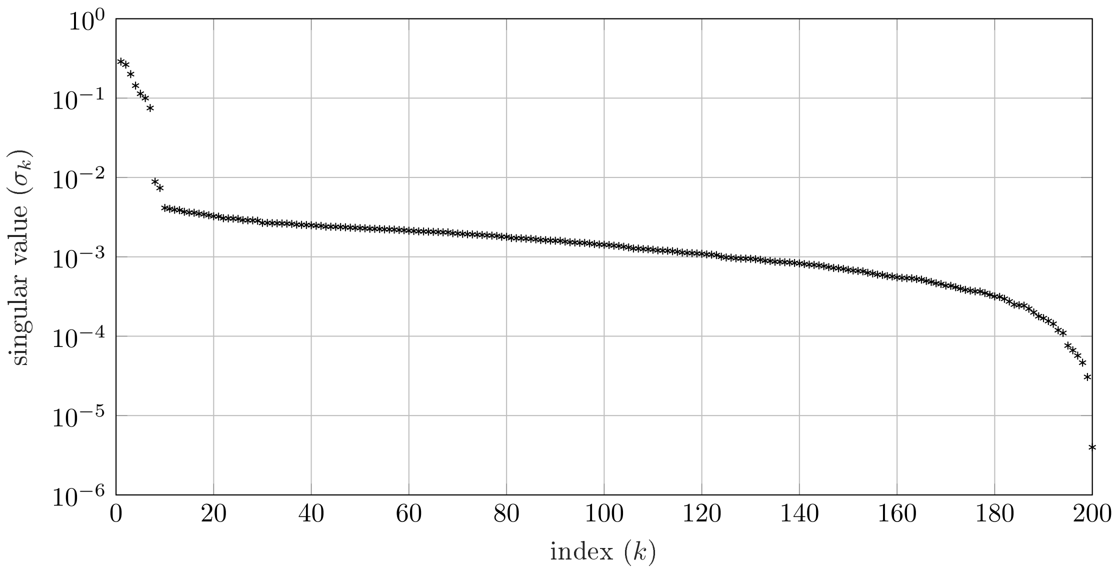
1. Summary

The project aims to identify and analyze a 2-input/2-output discrete-time model based on measurements of the system impulse response and with white-noise input to the system. The analysis methodically begins by selecting among several rank approximations of the Hankel matrix. The generated system's impulse response was compared against the data. The chosen model was verified by studying the pole-zero plots and the frequency response of each input-output channel in the system. The analysis resulted in creation of a block diagram of the system with components appropriately connected in each input-output channel.

2. Results

2.1. Task 1

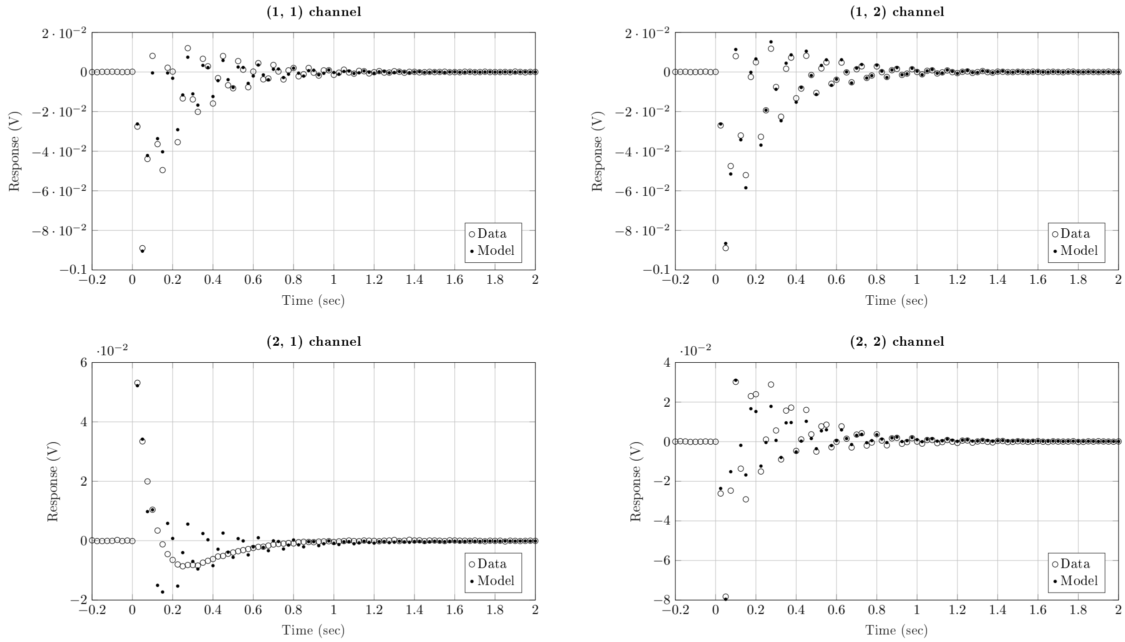
Figure 1: Singular Values of H_{100}



The singular values of the Hankel matrix H_{100}, arranged in decreasing order, show a sharp dip after the first seven values. This indicates that the frequency response of a system generated from the rank 7 approximation of the Hankel matrix shall closely resemble the empirical frequency response obtained from the measured data.

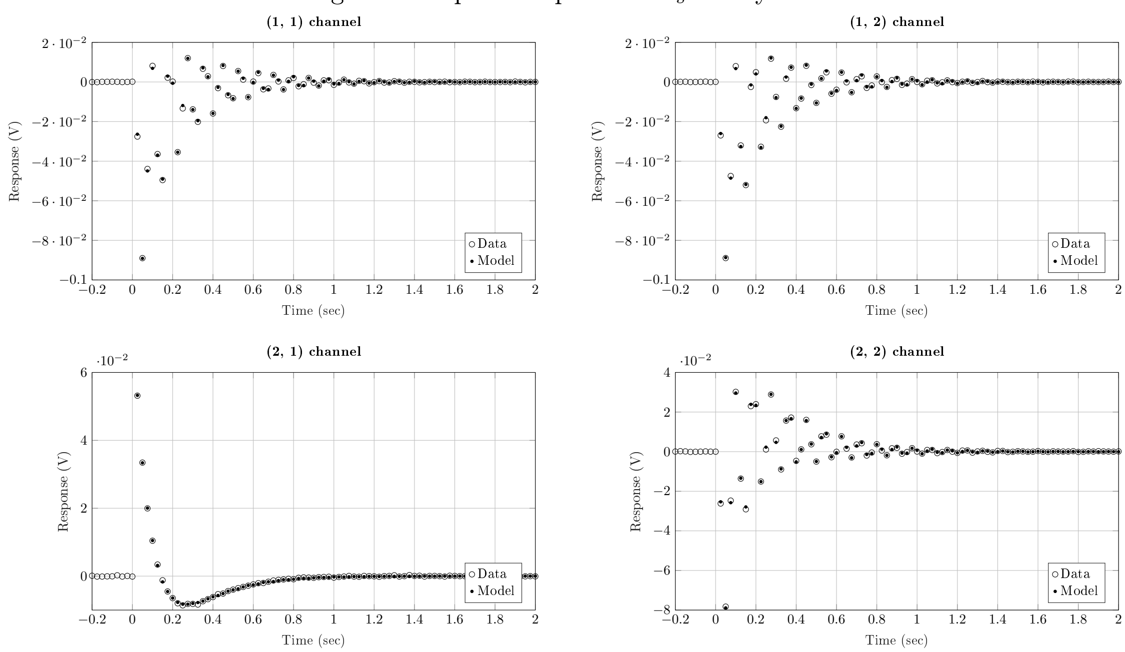
Models were generated with state dimensions 6, 7, 10, and 20. The magnitude of the dominant eigenvalue for each system was 0.953, 0.914, 0.914, and 0.998, respectively. Since all eigenvalues were confined within the unit circle, asymptotic stability criterion for discrete systems was satisfied. Models with the listed state dimensions were generated, and the theoretical impulse response was graphed for each channel alongside the pulse response data obtained empirically.

Figure 2: Impulse response of $n_s = 6$ system



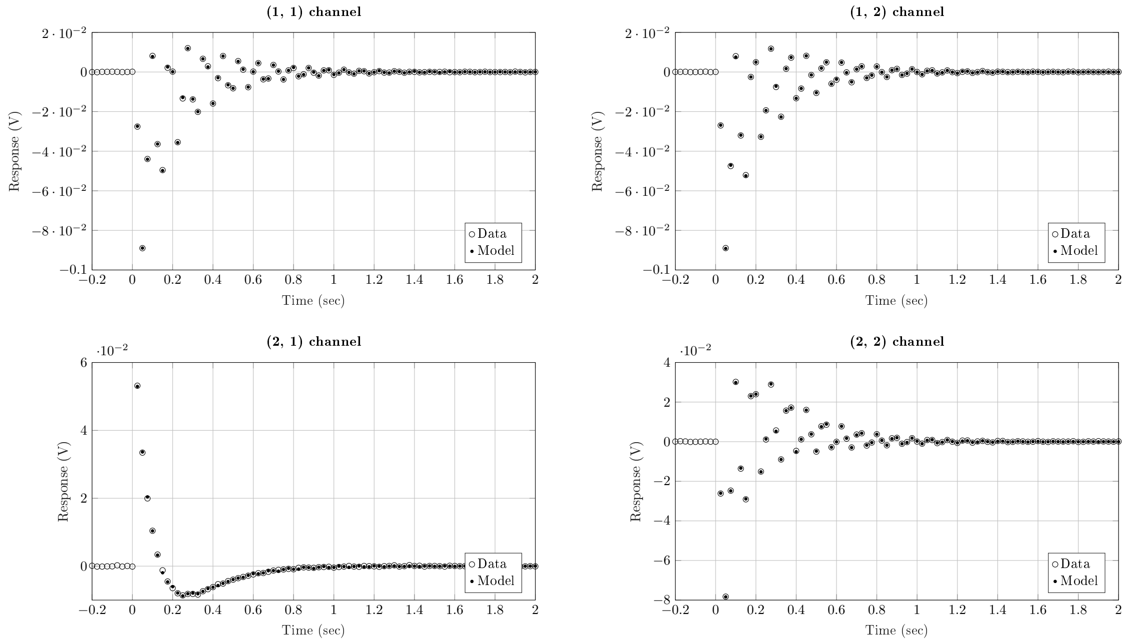
The deviation between the $n_s = 6$ model's impulse response and the data measurements must be noticed. This implies that significant information was lost with a rank 6 approximation.

Figure 3: Impulse response of $n_s = 7$ system



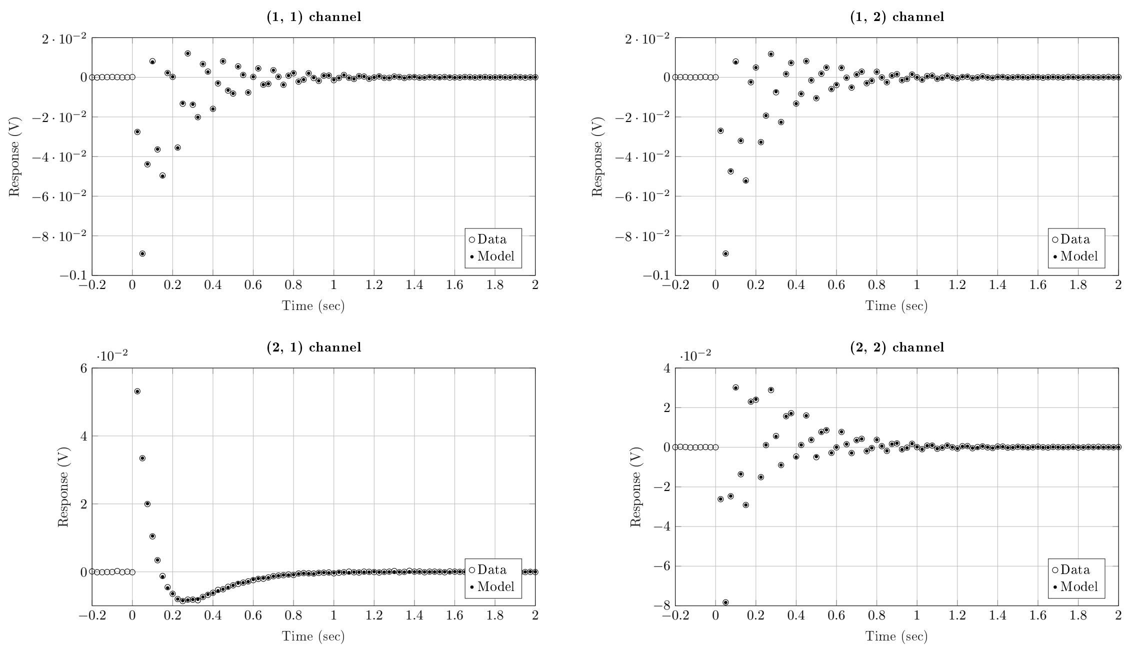
The $n_s = 7$ model's impulse response matched well with the measurements.

Figure 4: Impulse response of $n_s = 10$ system



The $n_s = 10$ model's impulse response matched well with the measurements.

Figure 5: Impulse response of $n_s = 20$ system



The $n_s = 20$ model's impulse response matched best with the measurements. Now, the frequency response from each model was generated, and a bode plot was created for each channel.

Figure 6: Frequency response of $(1, 1)$ channel

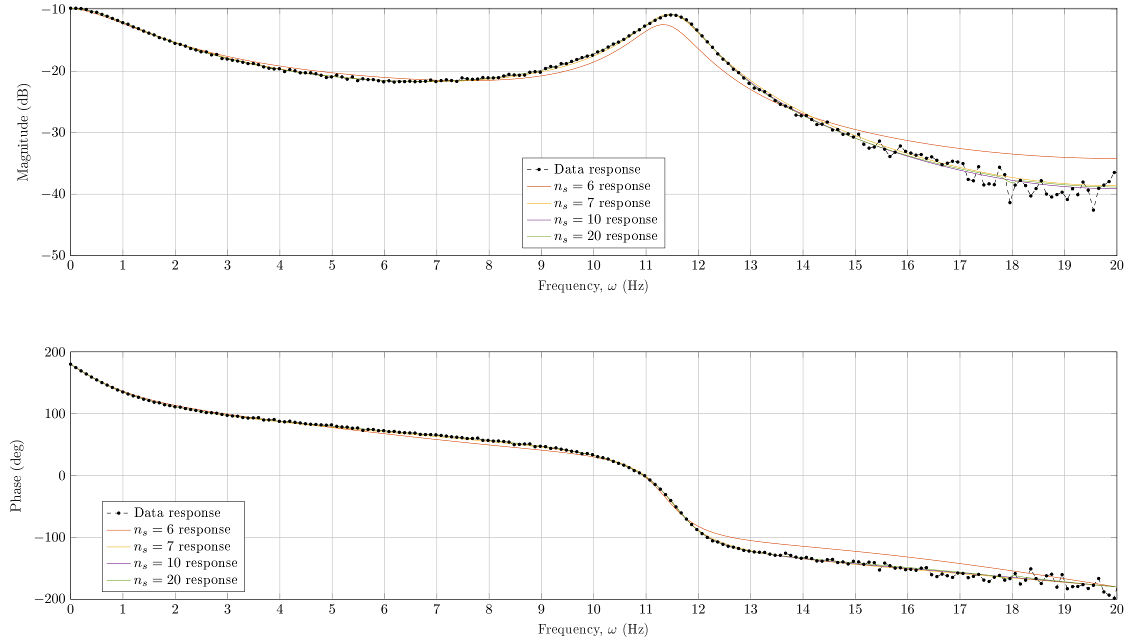


Figure 7: Frequency response of $(1, 2)$ channel

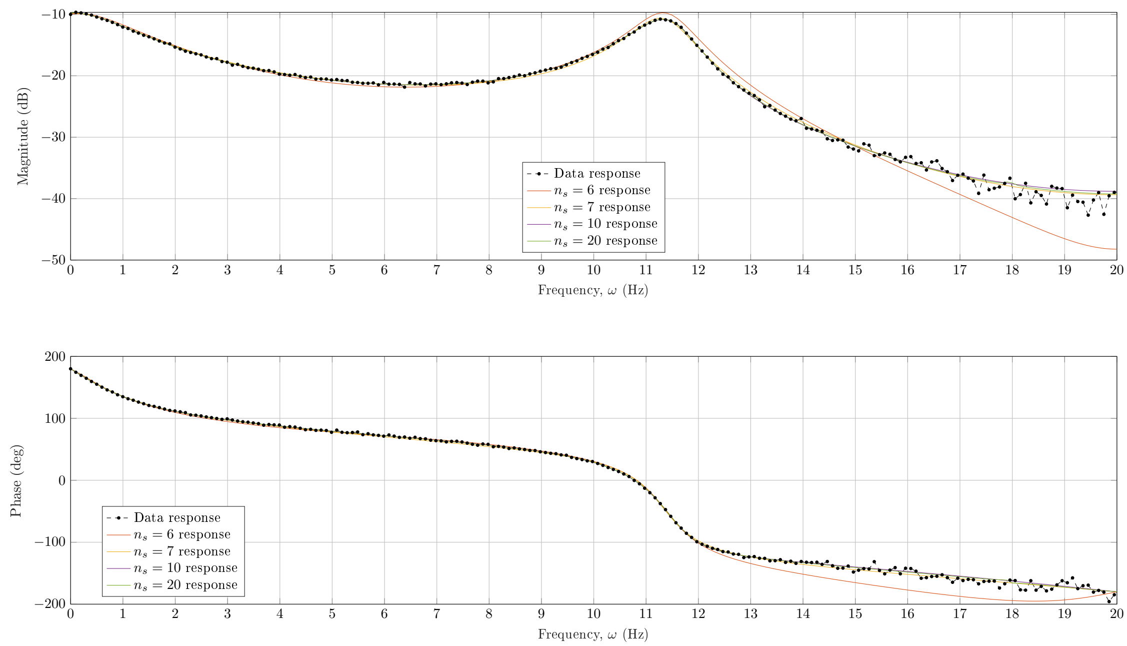


Figure 8: Frequency response of $(2, 1)$ channel

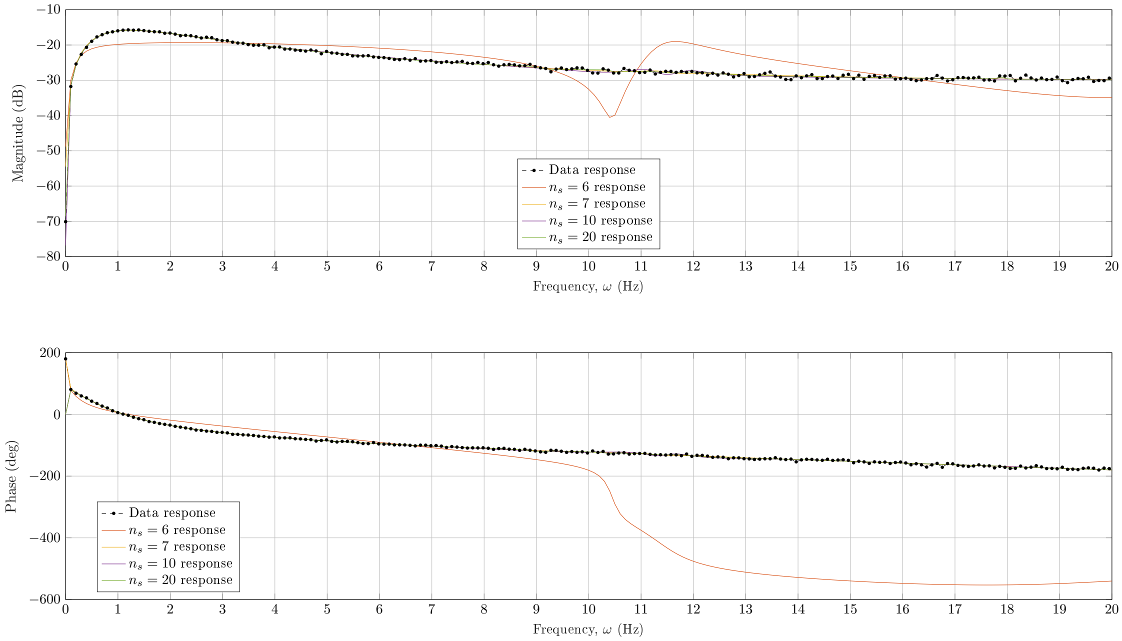
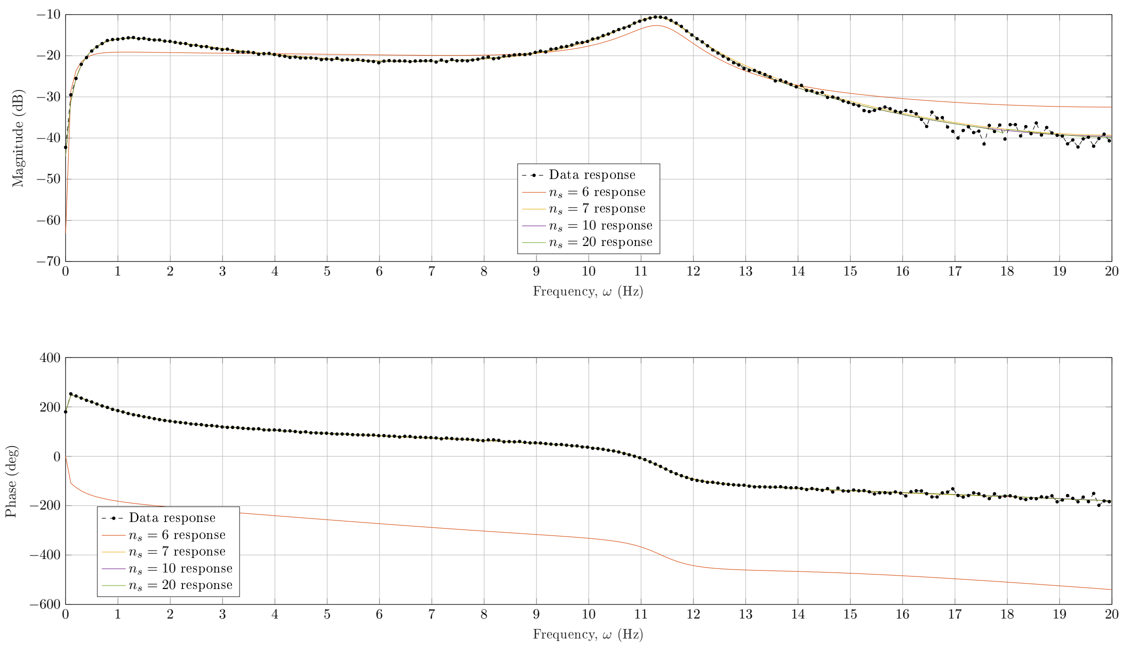


Figure 9: Frequency response of $(2, 2)$ channel

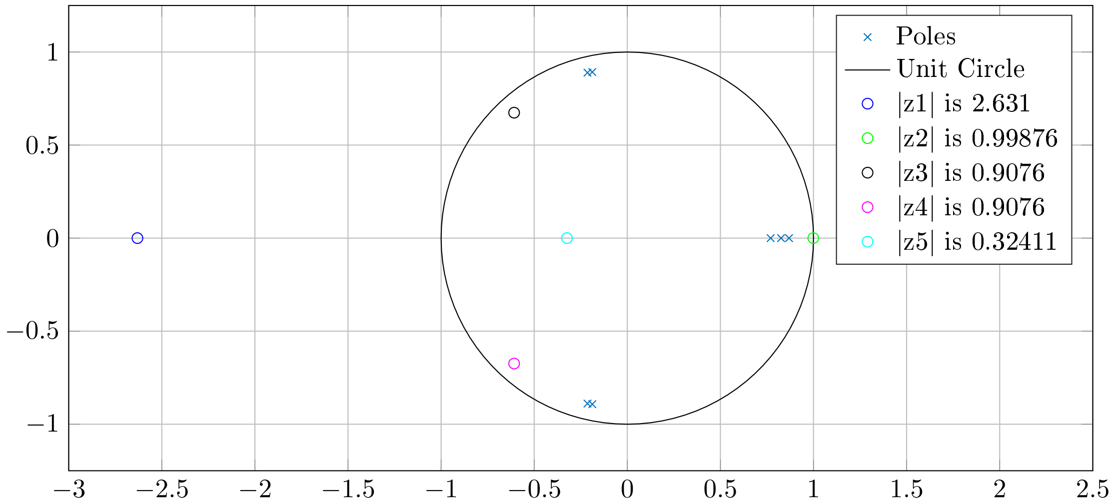


In all frequency response plots, the $n_s = 6$ model produced poor reproductions. This validates the unsuitability of a 6 state model. The faithful reproduction by the $n_s = 7$ model ensured its selection for the rest of the tasks, as it is the system with smallest state dimension that accomplishes this.

2.2. Task 2

In order to better understand the input-output properties of the system, the poles and zeros of the $n_s = 7$ system were investigated.

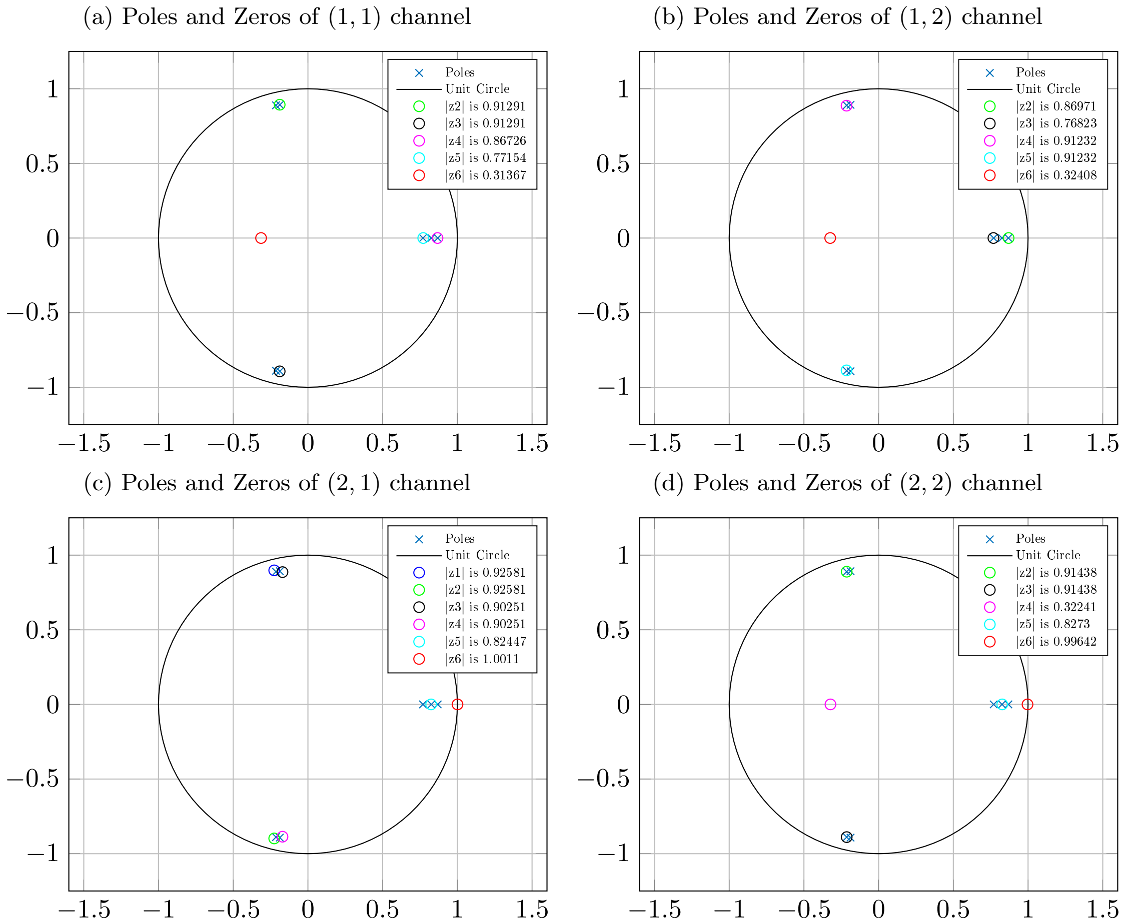
Figure 10: Poles and zeros of $n_s = 7$ system



The five finite transmission zeros and their magnitudes are displayed in the figure above. In discrete-time systems, the asymptotically stable zeros are confined within the unit circle. In the $n_s = 7$ model, there was one unstable zero, denoted by $|z_1|$. Due to asymptotic stability of the model, all 7 eigenvalues are confined within the unit circle.

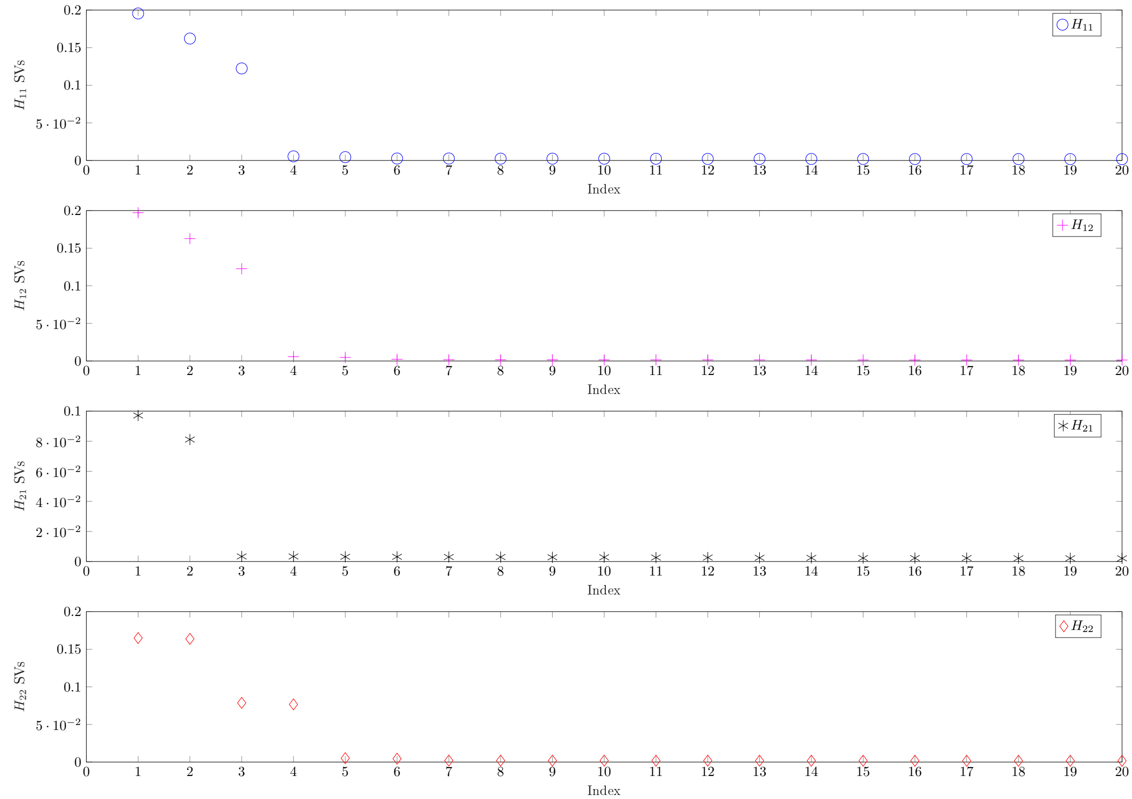
The equivalent continuous-time eigenvalues are $-3.6842 + 71.1394i$, $-3.6842 - 71.1394i$, $-3.5800 + 72.2737i$, $-3.5800 - 72.2737i$, -10.4604, and -7.6568. The eigenvalues with a non-zero imaginary part are associated with oscillation. A negative real part in that case indicates damping. The complex conjugate pair $-3.6842 + 71.1394i, -3.6842 - 71.1394i$ is associated with a damped oscillator with natural frequency $71.139\text{rad}/\sec$, i.e. 11.322Hz. The complex conjugate pair $-3.5800 + 72.2737i$ and $-3.5800 - 72.2737i$ is associated with a damped oscillator with natural frequency $72.2737\text{rad}/\sec$, i.e. 11.5027Hz. These frequencies manifest themselves in the magnitude graphs in the $(1,1)$, $(1,2)$, and $(2,2)$ channels at the frequency listed. There is also a corresponding drop in the phase graph. Now, the individual channels will be investigated for pole-zero cancellations.

Figure 11: Pole zero plots for each channel of $n_s = 7$ system



Upon seeing the pole-zero plots for each channel of the $n_s = 7$ system, a few things can be noticed. The (1,1) channel has four pole-zero cancellations and can be described by a 3 state model. The (1,2) channel has four pole-zero cancellations and can be described by a 3 state model. The (2,1) channel has five pole-zero cancellations and can be described by a 2 state model. The (2,2) channel has three pole-zero cancellations and can be described by a 4 state model. For the $(1,1)$, $(1,2)$, and $(2,2)$ channels, there is one uncancelled damped-oscillator pair, which corresponds to a resonance in the frequency response graphs as stated before. The $(2,1)$ frequency response graph has no obvious peak, which is supported by the fact that both pairs of complex poles are cancelled in that channel. The Hankel singular values were found and displayed below to support these claims.

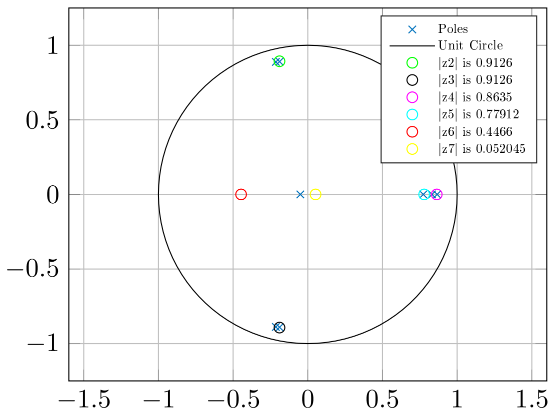
Figure 12: Hankel singular values for each channel



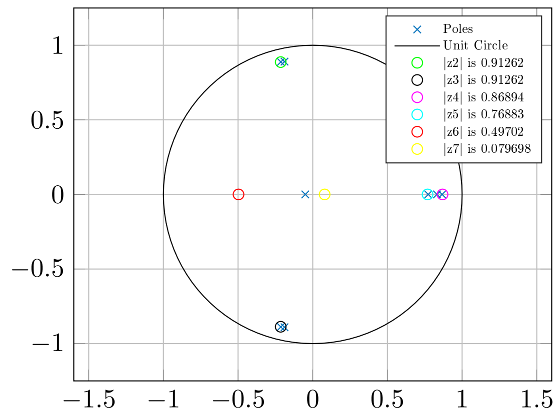
The discussion regarding the number of states in each input-output channel is reflected in the figures shown above. The Hankel singular values for the (1,1) and (1,2) channel show a dip after the first three values, confirming the three poles that were not cancelled in the pole-zero plots of the (1,1), and (1,2) channels. The Hankel singular values for the (2,1) channel drop after two values, confirming the channel's description by a 2 dimensional model. The Hankel singular values for the (2,2) channel drop after the first four values, and there exist four poles that were not cancelled in the pole-zero plot of the (2,2) channel.

Figure 13: Pole zero plots for $n_s = 8$ system

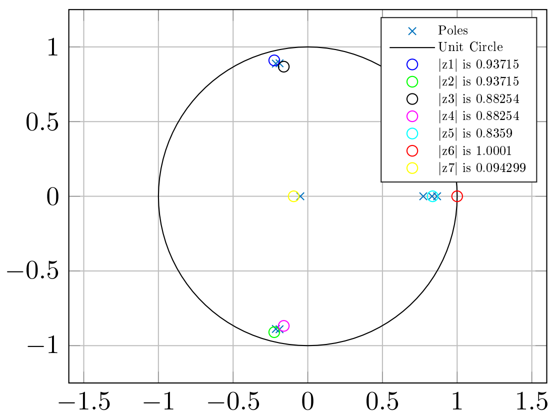
(a) Poles and Zeros of $(1, 1)$ channel



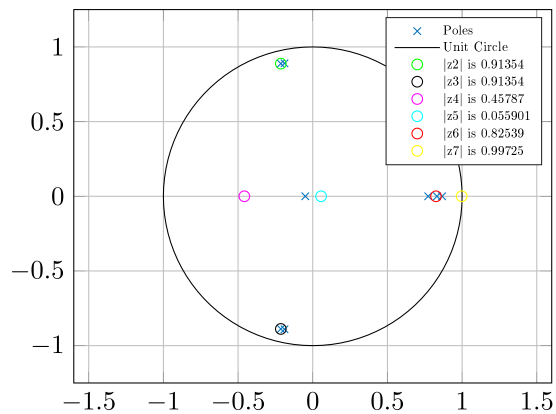
(b) Poles and Zeros of $(1, 2)$ channel



(c) Poles and Zeros of $(2, 1)$ channel



(d) Poles and Zeros of $(2, 2)$ channel

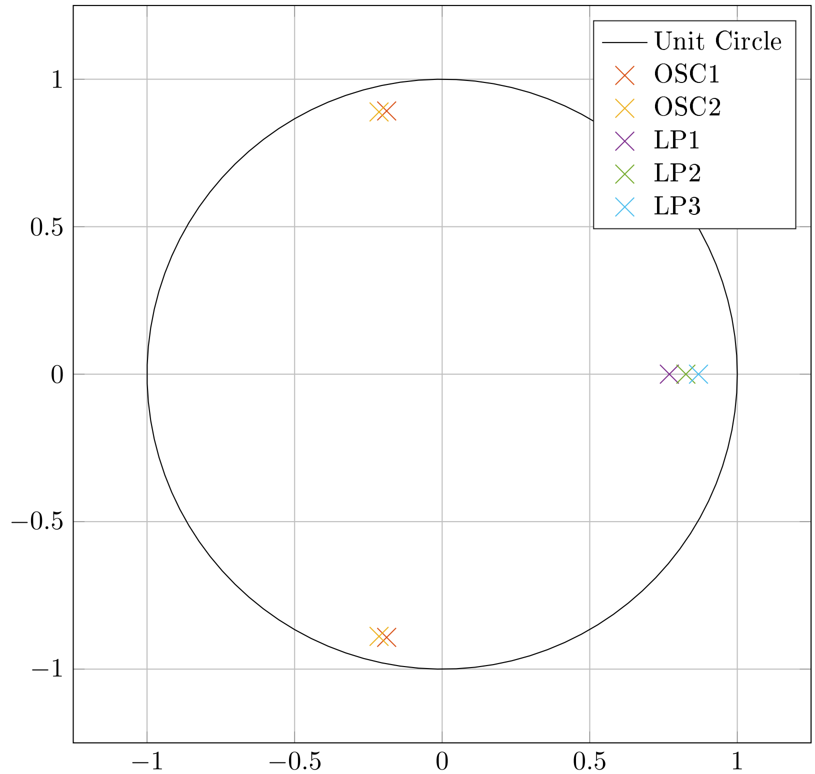


On generating a model with $n_s = 8$, for every channel, a new pole-zero cancellation was observed near the origin in comparison with the $n_s = 7$ model. Thus, the $n_s = 8$ model does not significantly modify the input-output response. This provides more evidence that the $n_s = 7$ system has the ideal number of states.

2.3. Task 3

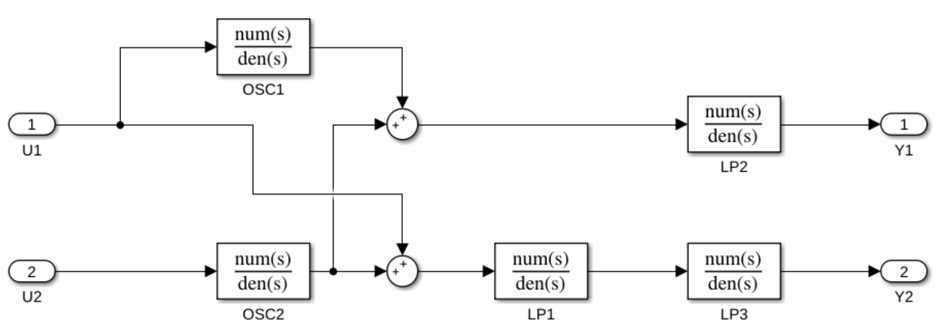
To create a block diagram representation of the system, each individual channel was considered first. From the previous two tasks, the “blocks“ each input/output pair was determined based off of the channel’s pole-zero plots, frequency response, and Hankel singular values. Labels were assigned to the poles representing each functional block as shown:

Figure 14: Poles of the $n_s = 7$ system and their names



Starting with channel $(1, 1)$, it has one uncancelled oscillator in the pole-zero plot and one uncancelled real pole. This means input 1 passes through OSC1 and LP2 on its way to output 1. With channel $(1, 2)$, it has a different uncancelled oscillator, and one uncancelled real pole. This means input 2 passes through OSC2 and LP2 on its way to output 1. For channel $(2, 1)$, there are no oscillators, and there are two low pass filters. So input 1 passes through LP1 and LP3 then to output 2. Finally, channel $(2, 2)$ contains one oscillator pair and two real poles. So input 2 passes through OSC2, LP1, and LP3 before going to output 2. These four channels can be combined into one block diagram as shown below.

Figure 15: Block diagram of the system



However, this block diagram is not unique. One thing to consider is that there is a zero on the real axis at 1, which occurs when LP1 and LP3 are uncancelled. Therefore, one of those low pass filters is actually a high pass filter, although it cannot be determined which.

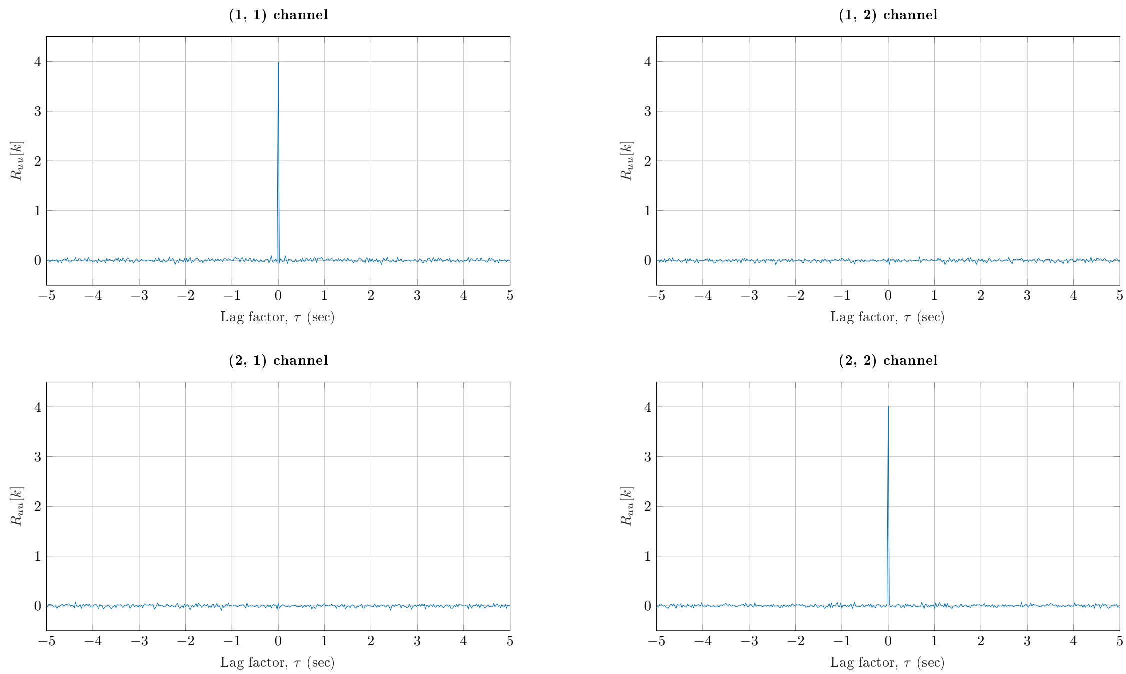
2.4. Task 4

This task involves the analysis of the propagation of noise through the system. The data from ten minutes of noise as an input to the system was collected, and the mean of the input channel is as shown:

$$\mu(u_1) = -0.0009\text{V}$$
$$\mu(u_2) = 0.0013\text{V}$$

The auto-correlation of the input channels, $R_{\mathbf{uu}}[k]$, was found in order to determine the degree to which the inputs are correlated. Each channel of this auto-correlation was graphed for $\tau = [-5, 5]$sec

Figure 16: The auto-correlation $R_{\mathbf{uu}}[k]$ of system noise

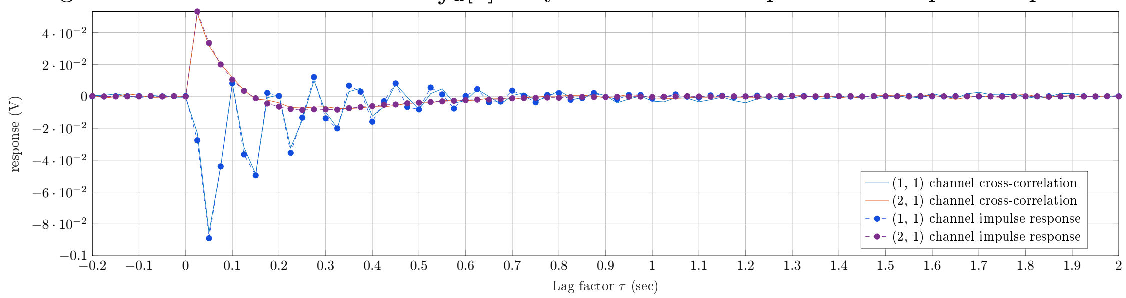


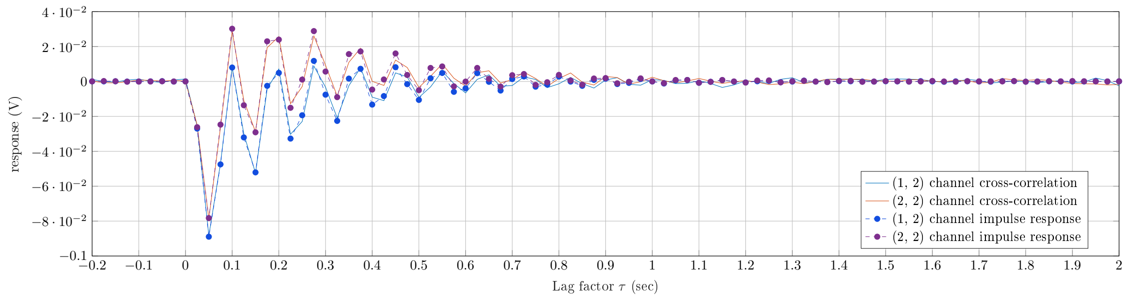
The $(1,1)$ and $(2,2)$ channels have a sharp peak at approximately $\tau = 0$. Each channel otherwise has a very small amount of noise. This is a good indication that both inputs are indeed zero mean, uncorrelated white noise. To find the variance, the values of $R_{\mathbf{uu}}[0]$ were calculated:

$$R_{\mathbf{uu}}[0] = \begin{bmatrix} 3.9854 & 0.0437 \\ 0.0437 & 4.0193 \end{bmatrix}$$

This shows that the variance of each input channel is not unit variance; rather, each channel has a variance of approximately 4. Because the inputs and outputs of a discrete time system are related via convolution, the cross correlation can be used to demonstrate how the system would behave with an impulse input. Therefore the cross-correlation, $R_{\mathbf{yu}}[k]$, of the system normalized by the variance found before was calculated and graphed below.

Figure 17: The cross-correlation $R_{\mathbf{yu}}[k]$ of system noise and experimental impulse response





This "virtual test" of the system matches well with the impulse response determined empirically. This shows that there are multiple sources that can be used to create an accurate model of a system. This could be useful if data found from an impulse response is dominated by noise.

2.5. Task 5

The $\mathcal{H}_2$ norm of the system was found using multiple methods in this section. First, it was calculated from the RMS value of the system output subjected to a white noise input. Then, it was found using the observability and controllability gramians of the $n = 7$ state space model. Finally, it was calculated from the empirical impulse response data. The result of these calculations are as shown:

$$\|\mathbf{y}\|_{\text{RMS}} = 0.2227$$
$$\|P\|_{\mathcal{H}_2} \text{ (from } G_{\mathcal{O}}) = 0.2297$$
$$\|P\|_{\mathcal{H}_2} \text{ (from } G_{\mathcal{C}}) = 0.2297$$
$$\|P\|_{\mathcal{H}_2} \text{ (from pulse response)} = 0.2298$$

The four values listed are close to each other, especially the last three, signalling that the norm was identified accurately. The first value, calculated from the noisy input, is a bit lower than the other three, which could possibly be caused by the fact that it was calculated from random noise, or possibly caused by the fact that the input-output data was recorded for a finite amount of time.

2.6. Task 6

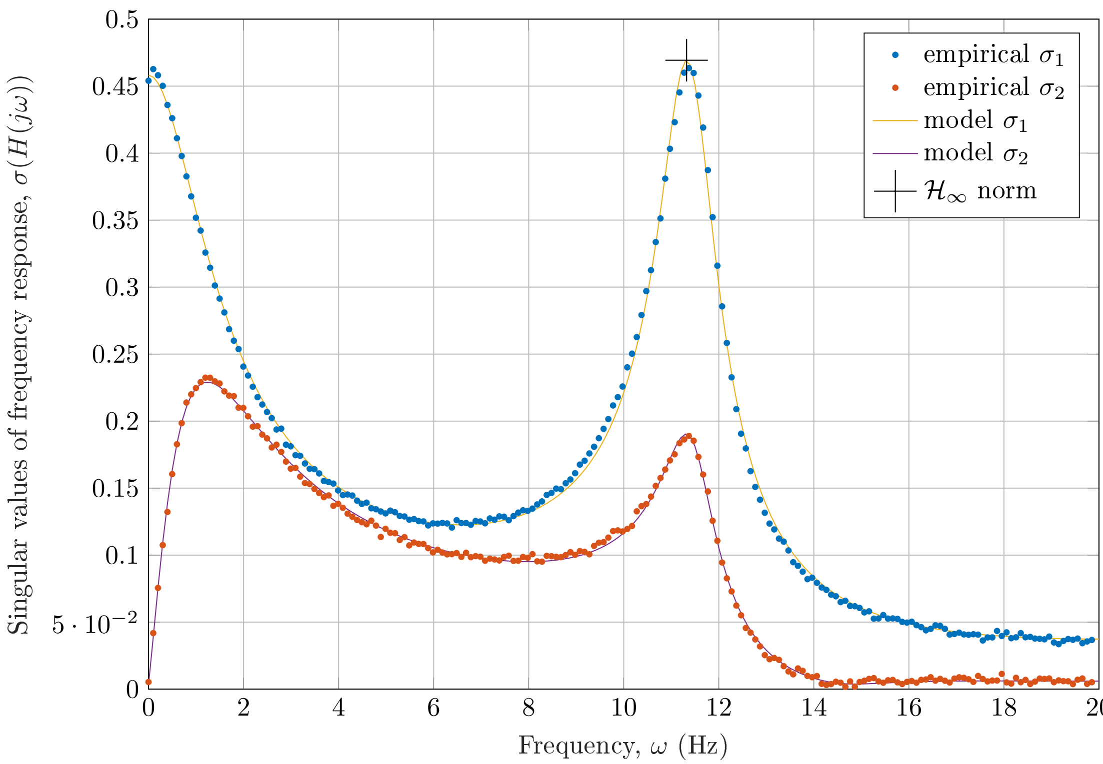
It is of great interest to calculate the $\mathcal{H}_\infty$ norm of our system; designing a controller to minimize this norm increases the robustness of the system. An algorithm to find the $\mathcal{H}_\infty$ norm was implemented for both discrete and continuous state-space systems and is included in the appendix. Upon calculating this norm on the $n = 7$

system, the norm and its corresponding frequency are found to be

$$\|P\|_{\mathcal{H}_\infty} = 0.4693$$
$$\omega_{\mathcal{H}_\infty} = 71.1442\text{rad}/\sec = 11.3229\text{Hz}$$

The singular values of the model-derived frequency response and empirical frequency response were calculated on the interval $[0, \omega_{\text{nyq}}]$Hz and subsequently plotted.

Figure 18: Frequency response singular values and $\mathcal{H}_\infty$ norm



A marker was placed at the frequency and magnitude corresponding to the $\mathcal{H}_\infty$ norm. This lines up exactly where it was expected to, the maximum singular value.

3. Conclusions

The main task of this project was to create an accurate model of a real-life system based off of empirical data. Based off the results of task 1, it was found that a state-space system with $n_s = 7$ matches both the frequency response and impulse response data well. Using this system, the pole-zero plots of each channel were investigated, which led to the identification of the "blocks" that each input-output pair passes through. This, along with the frequency response and Hankel singular values of each channel, led to the creation of a block diagram representation of the system. Although the block diagram representation could not be uniquely determined, the one listed in the figure should produce identical results to the system that produced the empirical data. As an alternative method to using the impulse response data, task 4 involved the analysis of the system when subjected to zero mean, uncorrelated white noise inputs. The cross-correlation of the system was generated, which matched the impulse response experimental data well. Finally, two system norms, $\|P\|_{\mathcal{H}_2}$ and $\|P\|_{\mathcal{H}_\infty}$, were found using multiple methods to provide more insight to the system.

4. Appendix

List of Listings

Listing 1: Task 1 code

```
%****************  TASK 1   *********************
clear
close all

load u1_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

load u2_impulse.mat
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);

%****************  TASK 1, Question 1 and 2  *********************

ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;

m = 2; %number of output channels
q = 2; %number of input channels
% n s.t size(H) = mn X qn
n = 100;
%n = (401 - 41)/2 ;%n is half the number of samples beginning from t = 1

ind_off = 41;
H = zeros(m*n, q*n);
Htil = zeros(m*n, q*n);

%To create H and H tilda matrices
r = 1;
c = 1;
for k = 1 : n
    ind = ind_off + k + [0:n-1];

    H(r,[c:q:c+q*(n-1)]) = y11(ind);
    H(r+1,[c:q:c+q*(n-1)]) = y21(ind);
    H(r,[c+1:q:c+1+q*(n-1)]) = y12(ind);
    H(r+1,[c+1:q:c+1+q*(n-1)]) = y22(ind);

    Htil(r,[c:q:c+q*(n-1)]) = y11(ind+1);
```

```
    Htil(r+1,[c:q:c+q*(n-1)]) = y21(ind+1);
    Htil(r,[c+1:q:c+1+q*(n-1)]) = y12(ind+1);
    Htil(r+1,[c+1:q:c+1+q*(n-1)]) = y22(ind+1);

    r = r + m;
end

s_vals = svd(H);
figure(1);
plot(s_vals,'b.','MarkerSize',10)
xlabel('singular value index','FontSize',12);
ylabel('H_{100} singular values','FontSize',12);
grid on

%when ns = 6
ns = 6;
[U, S, V] = svd(H);
U1 = U([1:m*n],[1:ns]);
S1 = S([1:ns],[1:ns]);
V1 = V([1:n*q],[1:ns]);

On6 = U1 * sqrt(S1);
Cn6 = sqrt(S1) * V1';

On6inv = inv(sqrt(S1)) * U1';
Cn6inv = V1 * inv(sqrt(S1));

C6 = On6([1:m],:);
B6 = Cn6(:,[1:q]);
A6 = On6inv * Htil * Cn6inv;

mev6 = max(abs(eig(A6)));
fprintf('When ns = 6, dominant eval is: %f \n',mev6);

%Plotting impulse response
h = zeros(m,q,n);
x = B6;
for k = 1:n
    h(:,:,k) = C6 * x;
    x = A6 * x;
end

tsim = [1 : n] * ts;
figure()
subplot(211)
plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
xlabel('Time')
ylabel('y_{11}')
legend('Model6', 'Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
xlabel('Time')
ylabel('y_{21}')
legend('Model6','Data')
grid on
axis([-0.2 2 -0.1 0.1])

figure()
subplot(211)
plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
xlabel('Time')
ylabel('y_{12}')
legend('Model6','Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
xlabel('Time')
ylabel('y_{22}')
legend('Model6','Data')
grid on
axis([-0.2 2 -0.1 0.1])

%when ns = 7
ns = 7;
[U, S, V] = svd(H);
U1 = U([1:m*n],[1:ns]);
S1 = S([1:ns],[1:ns]);
V1 = V([1:n*q],[1:ns]);

On7 = U1 * sqrt(S1);
Cn7 = sqrt(S1) * V1';
```

```
On7inv = inv(sqrt(S1)) * U1';
Cn7inv = V1 * inv(sqrt(S1));

C7 = On7([1:m],:);
B7 = Cn7(:,[1:q]);
A7 = On7inv * Htil * Cn7inv;

mev7 = max(abs(eig(A7)));
fprintf('When ns = 7, dominant eval is: %f \n',mev7);

%Plotting impulse response for model7
h = zeros(m,q,n);
x = B7;
for k = 1:n
    h(:,:,k) = C7 * x;
    x = A7 * x;
end

tsim = [1 : n] * ts;
figure()
subplot(211)
plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
xlabel('Time')
ylabel('y_{11}')
legend('Model7', 'Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
xlabel('Time')
ylabel('y_{21}')
legend('Model7','Data')
grid on
axis([-0.2 2 -0.1 0.1])

figure()
subplot(211)
plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
xlabel('Time')
ylabel('y_{12}')
legend('Model7','Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
xlabel('Time')
ylabel('y_{22}')
legend('Model7','Data')
grid on
axis([-0.2 2 -0.1 0.1])

%when ns = 10
ns = 10;
[U, S, V] = svd(H);
U1 = U([1:m*n],[1:ns]);
S1 = S([1:ns],[1:ns]);
V1 = V([1:n*q],[1:ns]);

On10 = U1 * sqrt(S1);
Cn10 = sqrt(S1) * V1';

On10inv = inv(sqrt(S1)) * U1';
Cn10inv = V1 * inv(sqrt(S1));

C10 = On10([1:m],:);
B10 = Cn10(:,[1:q]);
A10 = On10inv * Htil * Cn10inv;

mev10 = max(abs(eig(A10)));
fprintf('When ns = 10, dominant eval is: %f \n',mev10);

h = zeros(m,q,n);
x = B10;
for k = 1:n
    h(:,:,k) = C10 * x;
    x = A10 * x;
end

tsim = [1 : n] * ts;
figure()
subplot(211)
plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
xlabel('Time')
```

```
ylabel('y_{11}')
legend('Model10', 'Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
xlabel('Time')
ylabel('y_{21}')
legend('Model10','Data')
grid on
axis([-0.2 2 -0.1 0.1])

figure()
subplot(211)
plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
xlabel('Time')
ylabel('y_{12}')
legend('Model10','Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
xlabel('Time')
ylabel('y_{22}')
legend('Model10','Data')
grid on
axis([-0.2 2 -0.1 0.1])

%when ns = 20
ns = 20;
[U, S, V] = svd(H);
U1 = U([1:m*n],[1:ns]);
S1 = S([1:ns],[1:ns]);
V1 = V([1:n*q],[1:ns]);

On20 = U1 * sqrt(S1);
Cn20 = sqrt(S1) * V1';

On20inv = inv(sqrt(S1)) * U1';
Cn20inv = V1 * inv(sqrt(S1));

C20 = On20([1:m],:);
B20 = Cn20(:,[1:q]);
A20 = On20inv * Htil * Cn20inv;

mev20 = max(abs(eig(A20)));
fprintf('When ns = 20, dominant eval is: %f \n',mev20);

h = zeros(m,q,n);
x = B20;
for k = 1:n
    h(:,:,k) = C20 * x;
    x = A20 * x;
end

tsim = [1 : n] * ts;
figure()
subplot(211)
plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
xlabel('Time')
ylabel('y_{11}')
legend('Model20', 'Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
xlabel('Time')
ylabel('y_{21}')
legend('Model20','Data')
grid on
axis([-0.2 2 -0.1 0.1])

figure()
subplot(211)
plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
xlabel('Time')
ylabel('y_{12}')
legend('Model20','Data')
grid on
axis([-0.2 2 -0.1 0.1])

subplot(212)
plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
```

```
xlabel('Time')
ylabel('y_{22}')
legend('Model20','Data')
grid on
axis([-0.2 2 -0.1 0.1])

fprintf('Since all dominant evals have a magnitude less than 1, all models are asymptotically stable');

%****************  TASK 1, Question 3 and 4  ********************
%Computing Frequency response
wnyq = 20;
w = [0:wnyq/100:wnyq];%w in Hertz
Hf6 = zeros(m,q,length(w));
Hf7 = zeros(m,q,length(w));
Hf10 = zeros(m,q,length(w));
Hf20 = zeros(m,q,length(w));

for k = 1:length(w)
    Hf6(:,:,k) = C6 * inv(exp(1j*2*pi*w(k)*ts)*eye(6) - A6) * B6;
    Hf7(:,:,k) = C7 * inv(exp(1j*2*pi*w(k)*ts)*eye(7) - A7) * B7;
    Hf10(:,:,k) = C10 * inv(exp(1j*2*pi*w(k)*ts)*eye(10) - A10) * B10;
    Hf20(:,:,k) = C20 * inv(exp(1j*2*pi*w(k)*ts)*eye(20) - A20) * B20;
end

%Computing emperical frequency response
y11f = fft(y11)./fft(u1);
N = length(y11f);
om = [0:N-1]/(ts*N); %%%% frequency vector in hertz
y21f = fft(y21)./fft(u1);
y12f = fft(y12)./fft(u2);
y22f = fft(y22)./fft(u2);

%Plotting abs(Freq response of channel (1,1))
figure()
plot(w, abs(squeeze(Hf6(1,1,:))), 'b-')
hold on
plot(w, abs(squeeze(Hf7(1,1,:))), 'r-')
plot(w, abs(squeeze(Hf10(1,1,:))), 'g-')
plot(w, abs(squeeze(Hf20(1,1,:))), 'm-')
plot(om,abs(y11f), 'k--')
ylabel('|H(w)| (1,1)')
xlabel('Frequency (Hz)')
xlim([0 20])
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response magnitude of (1,1) channel')
grid on
hold off

%Plotting abs(Freq response) of channel (2,1)
figure()
plot(w, abs(squeeze(Hf6(2,1,:))), 'b-')
hold on
plot(w, abs(squeeze(Hf7(2,1,:))), 'r-')
plot(w, abs(squeeze(Hf10(2,1,:))), 'g-')
plot(w, abs(squeeze(Hf20(2,1,:))), 'm-')
plot(om,abs(y21f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('|H(w)| (2,1)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response magnitude of (2,1) channel')
grid on
hold off

%Plotting abs(Freq response) of channel (1,2)
figure()
plot(w, abs(squeeze(Hf6(1,2,:))), 'b-')
hold on
plot(w, abs(squeeze(Hf7(1,2,:))), 'r-')
plot(w, abs(squeeze(Hf10(1,2,:))), 'g-')
plot(w, abs(squeeze(Hf20(1,2,:))), 'm-')
plot(om,abs(y12f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('|H(w)| (1,2)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response magnitude of (1,2) channel')
grid on
hold off

%Plotting abs(Freq response) of channel (2,2)
figure()
plot(w, abs(squeeze(Hf6(2,2,:))), 'b-')
hold on
plot(w, abs(squeeze(Hf7(2,2,:))), 'r-')
plot(w, abs(squeeze(Hf10(2,2,:))), 'g-')
```

```
plot(w, abs(squeeze(Hf20(2,2,:))), 'm-')
plot(om,abs(y22f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('|H(w)| (2,2)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response magnitude of (2,2) channel')
grid on
hold off

%Plotting angle(Freq response of channel (1,1))
figure()
plot(w, angle(squeeze(Hf6(1,1,:))), 'b-')
hold on
plot(w, angle(squeeze(Hf7(1,1,:))), 'r-')
plot(w, angle(squeeze(Hf10(1,1,:))), 'g-')
plot(w, angle(squeeze(Hf20(1,1,:))), 'm-')
plot(om,angle(y11f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('angle(H(w)) (1,1)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response angle of (1,1) channel')
grid on
hold off

%Plotting angle(Freq response) of channel (2,1)
figure()
plot(w, angle(squeeze(Hf6(2,1,:))), 'b-')
hold on
plot(w, angle(squeeze(Hf7(2,1,:))), 'r-')
plot(w, angle(squeeze(Hf10(2,1,:))), 'g-')
plot(w, angle(squeeze(Hf20(2,1,:))), 'm-')
plot(om,angle(y21f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('angle(H(w))(2,1)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response angle of (2,1) channel')
grid on
hold off

%Plotting angle(Freq response) of channel (1,2)
figure()
plot(w, angle(squeeze(Hf6(1,2,:))), 'b-')
hold on
plot(w, angle(squeeze(Hf7(1,2,:))), 'r-')
plot(w, angle(squeeze(Hf10(1,2,:))), 'g-')
plot(w, angle(squeeze(Hf20(1,2,:))), 'm-')
plot(om,angle(y12f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('angle(H(w)) (1,2)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response angle of (1,2) channel')
grid on
hold off

%Plotting angle(Freq response) of channel (2,2)
figure()
plot(w, angle(squeeze(Hf6(2,2,:))), 'b-')
hold on
plot(w, angle(squeeze(Hf7(2,2,:))), 'r-')
plot(w, angle(squeeze(Hf10(2,2,:))), 'g-')
plot(w, angle(squeeze(Hf20(2,2,:))), 'm-')
plot(om,angle(y22f), 'k--')
xlabel('Frequency (Hz)')
xlim([0 20])
ylabel('angle(H(w)) (2,2)')
legend('Model6','Model7','Model10','Model20','Data')
title('Freq Response phase of (2,2) channel')
grid on
hold off
```

Listing 2: Task 2 code

```
%********************   TASK 2  ************************
clear
close all
clc

load u1_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
```

```
[~, mi] = max(u1>0); %%% find index where pulse occurs

load u2_impulse.mat
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);

%****************  TASK 2, Question 1 and 2  ********************

ts = 1/40; %%%% sample period

m = 2; %number of output channels
q = 2; %number of input channels
n = 100; % n s.t size(H) = mn X qn

ind_off = 41;
H = zeros(m*n, q*n);
Htil = zeros(m*n, q*n);

%To create H and H tilda matrices
r = 1;
c = 1;
for k = 1 : n
    ind = ind_off + k + (0:n-1);

    H(r,c:q:c+q*(n-1)) = y11(ind);
    H(r+1,c:q:c+q*(n-1)) = y21(ind);
    H(r,c+1:q:c+1+q*(n-1)) = y12(ind);
    H(r+1,c+1:q:c+1+q*(n-1)) = y22(ind);

    Htil(r,c:q:c+q*(n-1)) = y11(ind+1);
    Htil(r+1,c:q:c+q*(n-1)) = y21(ind+1);
    Htil(r,c+1:q:c+1+q*(n-1)) = y12(ind+1);
    Htil(r+1,c+1:q:c+1+q*(n-1)) = y22(ind+1);

    r = r + m;
end

%when ns = 7
ns = 7;
[U, S, V] = svd(H);
U1 = U(1:m*n, 1:ns);
S1 = S(1:ns, 1:ns);
V1 = V(1:n*q, 1:ns);

On7 = U1 * sqrt(S1);
Cn7 = sqrt(S1) * V1';

On7inv = inv(sqrt(S1)) * U1';
Cn7inv = V1 * inv(sqrt(S1));

C7 = On7([1:m],:);
B7 = Cn7(:,[1:q]);
A7 = On7inv * Htil * Cn7inv;

M = [A7 B7; -C7 zeros(2)];
N = [eye(7) zeros(7,2); zeros(2,7) zeros(2)];%disp(diag(D))%Resulted in D(1) and D(8) as inf
[~, D] = eig(M, N);


%2,3,4,5,6 elements of the diagonal are not inf and are in descending
%order. Elements 4 and 5 of the diagonal are equal
z1 = D(2,2); z2 = D(3,3); z3 = D(4,4); z4 = D(5,5); z5 = D(6,6);
disp('The finite transmission zeros are\n')
fprintf('The five finite transmission zeros are:\n')
fprintf('z1 is %f + %fi and |z1| is %f \n',real(z1),imag(z1),abs(z1))
fprintf('z2 is %f + %fi and |z2| is %f \n',real(z2),imag(z2),abs(z2))
fprintf('z3 is %f + %fi and |z3| is %f \n',real(z3),imag(z3),abs(z3))
fprintf('z4 is %f + %fi and |z4| is %f \n',real(z4),imag(z4),abs(z4))
fprintf('z5 is %f + %fi and |z5| is %f \n',real(z5),imag(z5),abs(z5))

eigA7 = eig(A7);
ut = 0:2*pi/100:2*pi;
figure()
```

```
plot(eigA7,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
title('Poles and Zeros')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-3 2.5 -1 1])
axis equal
grid on
hold off

%****************  TASK 2, Question 3  ********************
%Converting discrete evals to continuous evals
eigc_real = 1/ts * log(abs(eigA7));
eigc_im = 1/ts * angle(eigA7);
eigc = eigc_real + 1j * eigc_im;

fprintf('\n lambda_continuous \n');
disp(eigc);

disp('eigc([1,2,3,4]) have Re(.) < 0')
disp('There are two complex conjugate pairs with negative real parts (2 damped oscillators)')
fprintf('\n Oscillator 1 has a freq %f rad/sec = %f hertz\n',imag(eigc(1)),imag(eigc(1))/(2*pi));

%****************  TASK 2, Question 4  ********************

%For 1-1 channel the system is
B11 = B7(:,1);
C11 = C7(1,:);
M = [A7 B11; -C11 zeros(1)];
N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
[~, D] = eig(M,N);
z11 = diag(D);
z11 = z11([2,3,4,5,6,7]);
disp('Finite zeros for 1-1 channel:')
disp(z11)
lic = num2cell(z11);
[~, z2, z3, z4, z5, z6]=deal(lic{:});
figure()
plot(eigA7,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
title('Poles and Zeros 1-1 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off

%For 1-2 channel the system is
B12 = B7(:,2);
C12 = C7(1,:);
M = [A7 B12; -C12 zeros(1)];
N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
[~, D] = eig(M,N);
z12 = diag(D);
z12= z12([2,3,4,5,6,7]);
disp('Finite zeros for 1-2 channel:')
disp(z12)
lic = num2cell(z12);
[~, z2, z3, z4, z5, z6]=deal(lic{:});
figure()
plot(eigA7,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
title('Poles and Zeros 1-2 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
```

```
grid on
hold off

%For 2-1 channel
B21 = B7(:,1);
C21 = C7(2,:);
M = [A7 B21; -C21 zeros(1)];
N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
[~, D] = eig(M,N);
z21 = diag(D);
z21 = z21([2,3,4,5,6,7]);
disp('Finite zeros for 2-1 channel:')
disp(z21)
lic = num2cell(z21);
[z1, z2, z3, z4, z5, z6]=deal(lic{:});
figure()
plot(eigA7,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
title('Poles and Zeros 2-1 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off

%For 2-2 channel
B22 = B7(:,2);
C22 = C7(2,:);
M = [A7 B22; -C22 zeros(1)];
N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
[~, D] = eig(M,N);
z22 = diag(D);
z22 = z22([2,3,4,5,6,7]);
disp('Finite zeros for 2-2 channel:')
disp(z22)
lic = num2cell(z22);
[~, z2, z3, z4, z5, z6]=deal(lic{:});
figure()
plot(eigA7,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
title('Poles and Zeros 2-2 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off

%Hankel Matrix for 1-1 channel
m = 1; q =1;
H11 = zeros(m*n, q*n);
H12 = zeros(m*n, q*n);
H21 = zeros(m*n, q*n);
H22 = zeros(m*n, q*n);
r = 1;
c = 1;
for k = 1 : n
    ind = ind_off + k + [0:n-1];

    H11(r,:) = y11(ind);
    H12(r,:) = y12(ind);
    H21(r,:) = y21(ind);
    H22(r,:) = y22(ind);

    r = r + m;
end

figure()
subplot(411)
temp = svd(H11);
disp('The first 5 Hankel singular values for the 1-1 channel are')
disp(temp([1:5]));
```

```
plot(temp,'bo','DisplayName','H11');
ylabel('H11 sing vals');
xlabel('Index');xlim([0,20]);
legend
hold on

subplot(412)
temp = svd(H12);
disp('The first 5 Hankel singular values for the 1-2 channel are')
disp(temp([1:5]));
plot(temp,'m+','DisplayName','H12');
ylabel('H12 sing vals');
xlabel('Index');xlim([0,20]);
legend
hold on

subplot(413)
temp = svd(H21);
disp('The first 5 Hankel singular values for the 2-1 channel are')
disp(temp([1:5]));
plot(temp,'k*','DisplayName','H21');
ylabel('H21 sing vals');
xlabel('Index');
xlim([0,20]);
legend
hold on

subplot(414)
temp = svd(H22);
disp('The first 5 Hankel singular values for the 1-2 channel are')
disp(temp([1:5]));
plot(temp,'rd','DisplayName','H22');
ylabel('H22 sing vals');
xlabel('Index');
xlim([0,20]);
legend
hold on

%****************  TASK 2, Question 5  ********************

%Generating A,B, and C for ns = 8
m = 2; %number of output channels
q = 2; %number of input channels
ns = 8;
[U, S, V] = svd(H);
U1 = U([1:m*n],[1:ns]);
S1 = S([1:ns],[1:ns]);
V1 = V([1:n*q],[1:ns]);

On8 = U1 * sqrt(S1);
Cn8 = sqrt(S1) * V1';

On8inv = inv(sqrt(S1)) * U1';
Cn8inv = V1 * inv(sqrt(S1));

C8 = On8([1:m],:);
B8 = Cn8(:,[1:q]);
A8 = On8inv * Htil * Cn8inv;
eigA8 = eig(A8);

%For 1-1 channel the system is
B11 = B8(:,1);
C11 = C8(1,:);
M = [A8 B11; -C11 zeros(1)];
N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
[~, D] = eig(M,N);
z11 = diag(D);
z11 = z11([2,3,4,5,6,7,8]);
disp('Finite zeros for 1-1 channel:')
disp(z11)
lic = num2cell(z11);
[z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
figure()
plot(eigA8,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
%plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
title('ns = 8, Poles and Zeros 1-1 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
```

```
axis([-1 1 -1 1])
axis equal
grid on
hold off

%For 1-2 channel the system is
B12 = B8(:,2);
C12 = C8(1,:);
M = [A8 B12; -C12 zeros(1)];
N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
[V,D] = eig(M,N);
%disp(diag(D))%Resulted in D(1) and D(9) as inf
z12 = diag(D);
z12 = z12([2,3,4,5,6,7,8]);
disp('Finite zeros for 1-2 channel:')
disp(z12)
lic = num2cell(z12);
[z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
figure()
plot(eigA8,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
%plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
title('ns = 8, Poles and Zeros 1-2 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off

%For 2-1 channel the system is
B21 = B8(:,1);
C21 = C8(2,:);
M = [A8 B21; -C21 zeros(1)];
N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
[V,D] = eig(M,N);
%disp(diag(D))%Resulted in D(1) and D(9) as inf
z21 = diag(D);
z21 = z21([2,3,4,5,6,7,8]);
disp('Finite zeros for 2-1 channel:')
disp(z21)
lic = num2cell(z21);
[z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
figure()
plot(eigA8,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
title('ns = 8, Poles and Zeros 2-1 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off

%For 2-2 channel the system is
B22 = B8(:,2);
C22 = C8(2,:);
M = [A8 B22; -C22 zeros(1)];
N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
[V,D] = eig(M,N);
%disp(diag(D))%Resulted in D(1) and D(9) as inf
z22 = diag(D);
z22 = z22([2,3,4,5,6,7,8]);
disp('Finite zeros for 2-2 channel:')
disp(z22)
lic = num2cell(z22);
[z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
figure()
plot(eigA8,'x','DisplayName','Poles');
hold on
plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
```

```
%plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
title('ns = 8, Poles and Zeros 2-2 channel')
legend
set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
axis([-1 1 -1 1])
axis equal
grid on
hold off
```

Listing 3: Task 3 code

```
clear
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Calculate model and find poles
model = DiscreteModel();
H100 = model.find_hankel_matrix(100, 0);
H100_tilde = model.find_hankel_matrix(100, 1);
[A7, ~, ~] = model.compute_model(H100, H100_tilde, 7);
eig_A7 = eig(A7);

% Plot unit circle and poles
plot(exp(1j*linspace(0, 2*pi, 100)), 'k');
hold on;
plot(eig_A7(1:2), 'x', 'MarkerSize', 10);
plot(eig_A7(3:4), 'x', 'MarkerSize', 10);
plot(eig_A7(5), 0, 'x', 'MarkerSize', 10);
plot(eig_A7(6), 0, 'x', 'MarkerSize', 10);
plot(eig_A7(7), 0, 'x', 'MarkerSize', 10);
hold off;
grid on;
axis equal;
axis([-1.25 1.25 -1.25 1.25]);
legend(["Unit Circle" "OSC1" "OSC2" "LP1" "LP2" "LP3"]);
```

Listing 4: Task 4 code

```
close all
clear
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Load the random data
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;
u = [u1; u2];
y = [y1; y2];

ts = 1/40;
model = DiscreteModel();

% Task 4.1: Verify means are approx. 0
u1_mean = mean(u1)
u2_mean = mean(u2)
y1_mean = mean(y1)
y2_mean = mean(y2)

% Task 4.2: Graph estimates of Ruu for k in [-200, 200]
k = -200:200;
Ruu = zeros(4, length(k));

for i=1:length(k)
    % 5000 is trade-off between accuracy and speed
    Ruu(:, i) = reshape(find_correlation(u, u, k(i), length(u)*2), [4 1]);
end

figure(1);
subplot(2, 2, 1);
plot(k*ts, Ruu(1, :));
axis([-5 5 -0.5 4.5]);
xlabel("Lag factor, $\tau$ (sec)")
```

```
ylabel("$R_{uu}[k]$");
title("(1, 1) channel")
grid on

subplot(2, 2, 2);
plot(k*ts, Ruu(3, :));
axis([-5 5 -0.5 4.5]);
xlabel("Lag factor, $\tau$ (sec)")
ylabel("$R_{uu}[k]$");
title("(1, 2) channel")
grid on

subplot(2, 2, 3);
plot(k*ts, Ruu(2, :));
axis([-5 5 -0.5 4.5]);
xlabel("Lag factor, $\tau$ (sec)")
ylabel("$R_{uu}[k]$");
title("(2, 1) channel")
grid on

subplot(2, 2, 4);
plot(k*ts, Ruu(4, :));
axis([-5 5 -0.5 4.5]);
xlabel("Lag factor, $\tau$ (sec)")
ylabel("$R_{uu}[k]$");
title("(2, 2) channel")
grid on

sgtitle(["Autocorrelation of $u$, $R_{uu}$" "for $\tau\in[-5, 5]$"], ...
    'interpreter', 'latex');

% Task 4.3: Find Ruu(0)
Ruu0 = find_correlation(u, u, 0, length(u)*2)

% Task 1.4: Estimate Ryu for tau in [-0.2, 2]
k = -0.2/ts:2/ts;
Ryu = zeros(4, length(k));

for i=1:length(k)
    Ryu(:, i) = reshape(find_correlation(y, u, k(i), length(u)*2), [4 1]);
end

% Normalize columns
Ryu11 = Ryu(1, :)./Ruu0(1, 1);
Ryu12 = Ryu(3, :)./Ruu0(1, 1);
Ryu21 = Ryu(2, :)./Ruu0(2, 2);
Ryu22 = Ryu(4, :)./Ruu0(2, 2);

figure(2);
subplot(2, 1, 1);
plot(k*ts, [Ryu11; Ryu21]);
hold on
plot(model.t, model.y11, '.--');
plot(model.t, model.y21, '.--');
xlim([-0.2 2]);
xlabel("Lag factor $\tau$ (sec)");
ylabel("response (V)");
legend(["(1, 1) channel autocorrelation" ...
    "(2, 1) channel autocorrelation" "(1, 1) channel impulse response" ...
    "(2, 1) channel impulse response"], ...
    'location', 'southeast');
grid on

subplot(2, 1, 2);
plot(k*ts, [Ryu12; Ryu22]);
hold on
plot(model.t, model.y12, '.--');
plot(model.t, model.y22, '.--');
xlim([-0.2 2]);
xlabel("Lag factor $\tau$ (sec)");
ylabel("response (V)");
legend(["(1, 2) channel autocorrelation" ...
    "(2, 2) channel autocorrelation" "(1, 2) channel impulse response" ...
    "(2, 2) channel impulse response"], ...
    'location', 'southeast');
grid on

sgtitle(["Autocorrelation of $u$, $R_{yu}$ vs data impulse response" ...
    "for $\tau\in[-0.2, 2]$"]);

sqrt(trace(find_correlation(y./2, y./2, 0, length(u)*2)))

function Rab = find_correlation(a, b, k, p_est)
    Rab = zeros(size(a, 1), size(b, 1));
    n = size(a, 2);
    elems = 0;
```

```

    for q=-p_est:p_est
        % Can't allow indices outside of [1, length(a)]
        if (q >= 1 && k+q >= 1 && q <= n && k+q <= n)
            Rab = Rab + a(:, k+q)*b(:, q)';
            elems = elems + 1;
        end
    end
    Rab = Rab/elems;
end
```

Listing 5: Task 5 code

```
close all
clear

% Load the random data and normalize it
load u_rand.mat
y1 = u_rand.Y(3).Data/2;
y2 = u_rand.Y(4).Data/2;
y = [y1-mean(y1); y2-mean(y2)];

% Task 5.1: Find ||y||_RMS^2 of scaled output
Y_RMS = sqrt(sum(vecnorm(y).^2)/size(y, 2))

% Task 5.2: Compute ||P||_H2^2 for ns=7 system
% ns=7 model from task 1
model = DiscreteModel();
H100 = model.find_hankel_matrix(100, 0);
H100_tilde = model.find_hankel_matrix(100, 1);
[A7, B7, C7] = model.compute_model(H100, H100_tilde, 7);

H2_norm_1 = sqrt(trace(B7'*dlyap(A7', C7'*C7)*B7))
H2_norm_2 = sqrt(trace(C7*dlyap(A7, B7*B7')*C7'))

% Task 5.3: Compute ||P||_H2^2 from impulse response data
H2_norm_data = sqrt(sum(model.y11.^2) + sum(model.y12.^2) + ...
    sum(model.y21.^2) + sum(model.y22.^2))
```

Listing 6: Task 6 code

```
clear
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Compute frequency responses
model = DiscreteModel();
H100 = model.find_hankel_matrix(100, 0);
H100_tilde = model.find_hankel_matrix(100, 1);
[A7, B7, C7] = model.compute_model(H100, H100_tilde, 7);
omega = 0:0.1:(1/(2*model.ts));
F7 = model.compute_freq_resp(A7, B7, C7, omega);

y11f = fft(model.y11)./fft(model.u1);
y12f = fft(model.y12)./fft(model.u2);
y21f = fft(model.y21)./fft(model.u1);
y22f = fft(model.y22)./fft(model.u2);
om_data = (0:length(y11f)/2-1)/(model.ts*length(y11f));

% Find Hinf norm
[H_val, f_val] = Hinf_dis(A7, B7, C7, zeros(2), ...
    sum(svd(H100)), 0, 1e-6, model.ts);
H_val
f_val

% Now compute and graph singular values
SV_model = zeros(2, size(F7, 2));
for k=1:size(F7, 2)
    SV_model(:, k) = svd([F7(1, k) F7(3, k); F7(2, k) F7(4, k)]);
end

SV_data = zeros(2, length(om_data));
for k=1:length(om_data)
    SV_data(:, k) = svd([y11f(k) y12f(k); y21f(k) y22f(k)]);
end

plot(om_data, SV_data, '.', omega, SV_model, '-');
hold on
plot(f_val/(2*pi), H_val, 'k+', 'MarkerSize', 10);
grid on
xlabel("Frequency, $\omega$ (Hz)");
ylabel("Singular values of frequency response, $\sigma(H(j\omega))$")
```

```
legend(["empirical $\sigma_1$" "empirical $\sigma_2$" ...
    "model $\sigma_1$" "model $\sigma_2$" "$H_\infty$ norm"], ...
    'location', 'NorthEast');
hold off
```

Listing 7: DiscreteModel file (used for tasks 3-6)

```
classdef DiscreteModel
    properties
        y11
        y12
        y21
        y22
        u1
        u2
        N
        ts
        t
        t_imp
    end

    methods
        function model=DiscreteModel()
            % Load and clean data
            load u1_impulse.mat;
            load u2_impulse.mat;

            y11 = u1_impulse.Y(3).Data;
            y21 = u1_impulse.Y(4).Data;
            y12 = u2_impulse.Y(3).Data;
            y22 = u2_impulse.Y(4).Data;
            u1 = u1_impulse.Y(1).Data;
            u2 = u2_impulse.Y(2).Data;

            % Remove DC offset in data
            [~, mi1] = max(u1 > 0);
            [~, mi2] = max(u2 > 0);
            y11 = y11 - mean(y11(1:mi1 - 1));
            y12 = y12 - mean(y12(1:mi2 - 1));
            y21 = y21 - mean(y21(1:mi1 - 1));
            y22 = y22 - mean(y22(1:mi2 - 1));
            u1 = u1 - mean(u1(1:mi1 - 1));
            u2 = u2 - mean(u2(1:mi2 - 1));
            mu1 = max(u1);
            mu2 = max(u2);

            % rescale IO data so that impulse input has magnitude 1
            model.y11 = y11/mu1;
            model.y12 = y12/mu2;
            model.y21 = y21/mu1;
            model.y22 = y22/mu2;
            model.u1 = u1/mu1;
            model.u2 = u2/mu2;

            model.N = length(u1);
            model.ts = 1/40;
            model.t_imp = mi1;
            model.t = ((1:model.N) - model.t_imp)*model.ts;

        end

        function Hn=find_hankel_matrix(model, n, offset)
            Hn = zeros(2*n);
            for r=1:n
                for c=1:n
                    k = r+c-1+offset+model.t_imp;
                    Hn(2*r-1:2*r, 2*c-1:2*c) = ...
                        [model.y11(k) model.y12(k); ...
                        model.y21(k) model.y22(k)];
                end
            end
        end

        function [A, B, C]=compute_model(~, H, H_tilde, n)
            % Truncate H to use the first n singular values
            [U, S, V] = svd(H);
            Un = U(:, 1:n);
            Sn = S(1:n, 1:n);
            Vn = V(:, 1:n);

            % Find the observability and controllabiltiy matrices
            % and their inverses.
            Sn12 = diag(sqrt(diag(Sn)));
            Snn12 = diag(1./sqrt(diag(Sn)));
            On = Un*Sn12;
```

```
            On_inv = Snn12*Un';
            Cn = Sn12*Vn';
            Cn_inv = Vn*Snn12;

            % Calculate system matrices
            A = On_inv*H_tilde*Cn_inv;
            B = Cn(:, 1:2);
            C = On(1:2, :);
        end

        % Find the impulse response of the given system and channel
        function h=compute_imp_resp(~, A, B, C, n)
            h = zeros(4, n);
            Ap = eye(size(A));
            for k=1:n
                hk = C*Ap*B;
                h(:, k) = reshape(hk, [4 1]);
                Ap = Ap*A;
            end
        end

        % Find the frequency reponse of the given system and channel
        function F=compute_freq_resp(model, A, B, C, omega)
            TS = model.ts;
            F = zeros(4, length(omega));
            for i=1:length(omega)
                F_temp = C*inv(exp(2i*pi*omega(i)*TS).*eye(size(A))-A)*B;
                F(:, i) = reshape(F_temp, [4 1]);
            end
        end
    end
end
```

Listing 8: Hinf_cont function (used for task 6)

```
function [gam, freq] = Hinf_cont(A, B, C, D, up, lo, tol)

freq = -1;

while (up - lo)/lo > tol/2

    gam = (up + lo)/2;
    Dg = gam^2 * eye(size(D' * D, 1)) - D' * D;
    Aclp = [A + B *inv(Dg)*D'*C, -B*inv(Dg)*B'; C'*C+C'*D*inv(Dg)*D'*C, -A'-C'*D*inv(Dg)*B'];

    evals = eig(Aclp);
    t = 0;
    for eind = 1:length(evals)
        if abs(real(evals(eind))) < 1e-8
            freq = abs(imag(evals(eind)));
            t = 1;
        end
    end
    if t == 1
        lo = gam;
    else
        up = gam;
    end

end

end
```

Listing 9: Hinf_dis function (used for task 6)

```
function [Hinf, freq] = Hinf_dis(A, B, C, D, up, lo, tol, tsam)
%INPUTS: Function to compute H_inf for continuous time systems,
%        discrete time matrices - A, B, C, D
%        upper limit, lower limit, and tolerance for gamma
%RETURNS the H_inf norm and the discrete time frequency at which it ocuurs.
Ac = -inv(eye(size(A)) + A) * (eye(size(A)) - A);
Bc = sqrt(2) * inv(eye(size(A)) + A) * B;
Cc = sqrt(2) * C * inv(eye(size(A)) + A);
Dc = D - C * inv(eye(size(A)) + A) * B;

[Hinf, freq] = Hinf_cont(Ac, Bc, Cc, Dc, up, lo, tol);
freq = 1/tsam * angle((1 + 1i * freq)/ (1 - 1i * freq));

end
```