

MAE M270A Project

Ganesha Durbha

Jared Jonas

December 18, 2020

Table of Contents

1	Summary	2
2	Results	2
2.1	Task 1	2
2.2	Task 2	6
2.3	Task 3	10
2.4	Task 4	12
2.5	Task 5	13
2.6	Task 6	13
3	Conclusions	14
4	Appendix	15

List of Figures

1	Singular Values of H_{100}	2
2	Impulse response of $n_s = 6$ system	3
3	Impulse response of $n_s = 7$ system	3
4	Impulse response of $n_s = 10$ system	4
5	Impulse response of $n_s = 20$ system	4
6	Frequency response of (1,1) channel	5
7	Frequency response of (1,2) channel	5
8	Frequency response of (2,1) channel	6
9	Frequency response of (2,2) channel	6
10	Poles and zeros of $n_s = 7$ system	7
11	Pole zero plots for each channel of $n_s = 7$ system	8
12	Hankel singular values for each channel	9
13	Pole zero plots for $n_s = 8$ system	10
14	Poles of the $n_s = 7$ system and their names	11
15	Block diagram of the system	11
16	The auto-correlation $R_{\mathbf{uu}}[k]$ of system noise	12
17	The cross-correlation $R_{\mathbf{yu}}[k]$ of system noise and experimental impulse response	13
18	Frequency response singular values and \mathcal{H}_∞ norm	14

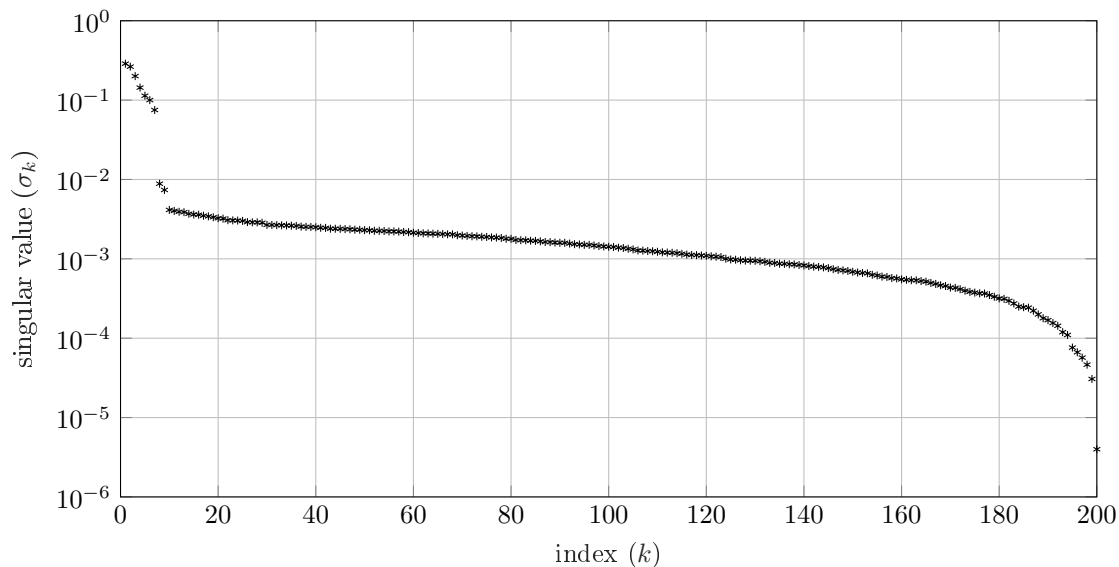
1. Summary

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

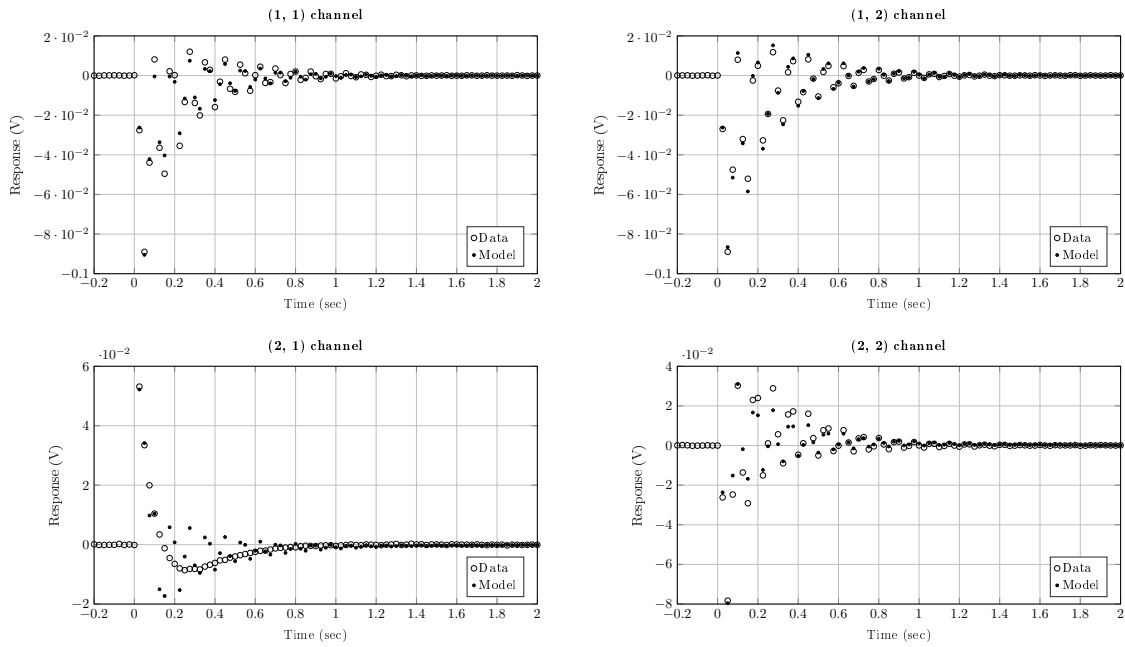
2.1. Task 1

Figure 1: Singular Values of H_{100}

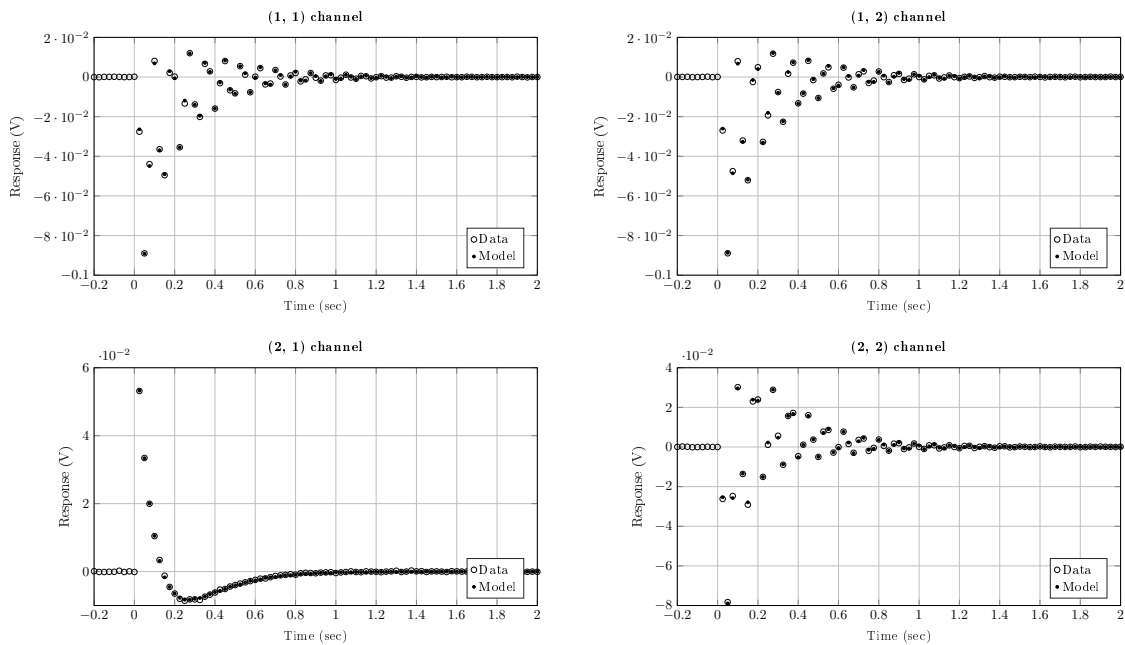


The singular values of the Hankel matrix H_{100} , arranged in decreasing order, show a sharp dip after the first seven values. This indicates that the frequency response of a system generated from the rank 7 approximation of the Hankel matrix shall closely resemble the empirical frequency response obtained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

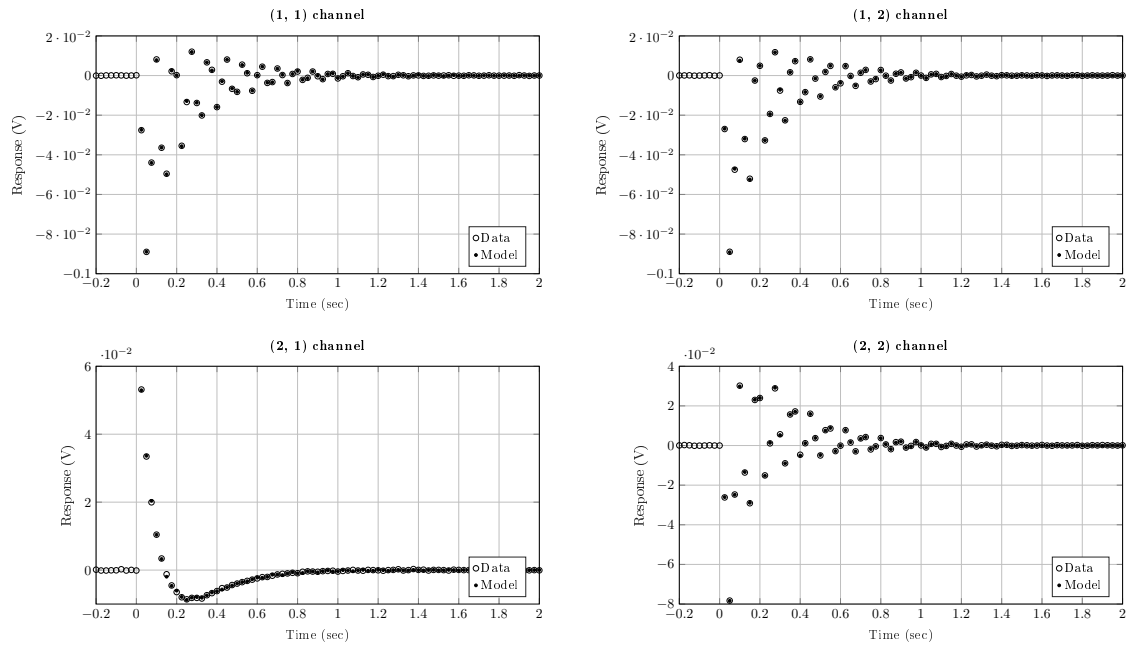
Models were generated with state dimensions 6, 7, 10, and 20. The magnitude of the dominant eigenvalue for each system was 0.953, 0.914, 0.914, and 0.998, respectively. Since all eigenvalues were confined within the unit circle, asymptotic stability criterion for discrete systems was satisfied. Models 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

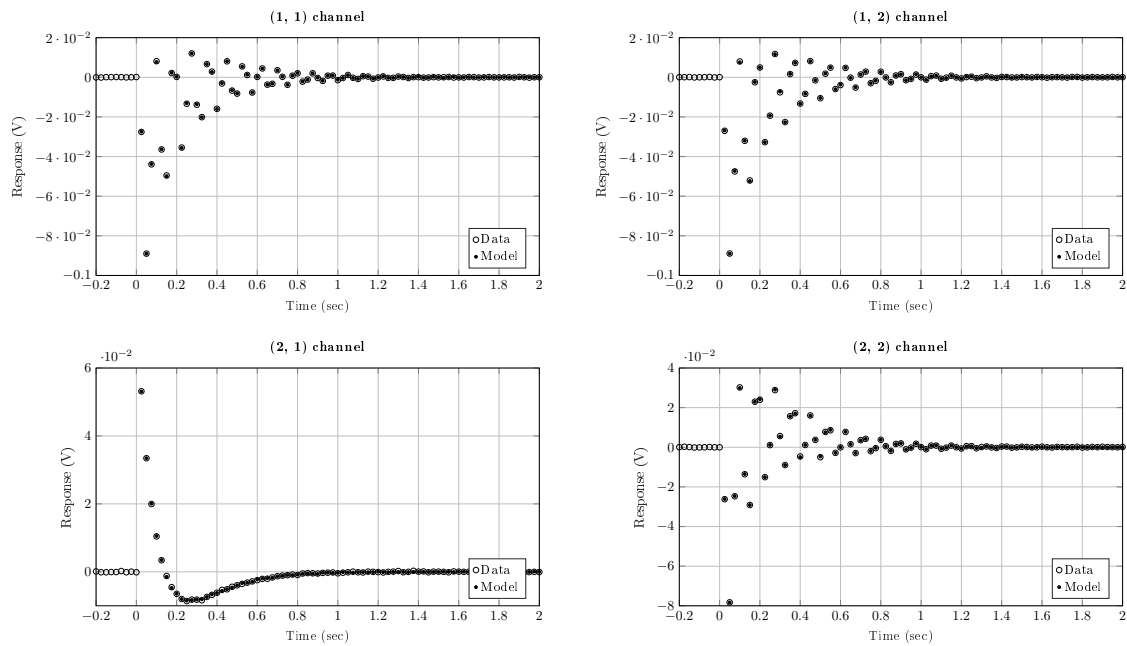
The deviation between the $n_s = 6$ model's impulse response and the data measurements must be noticed. This implies that significant information was lost with a rank 6 approximation.

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

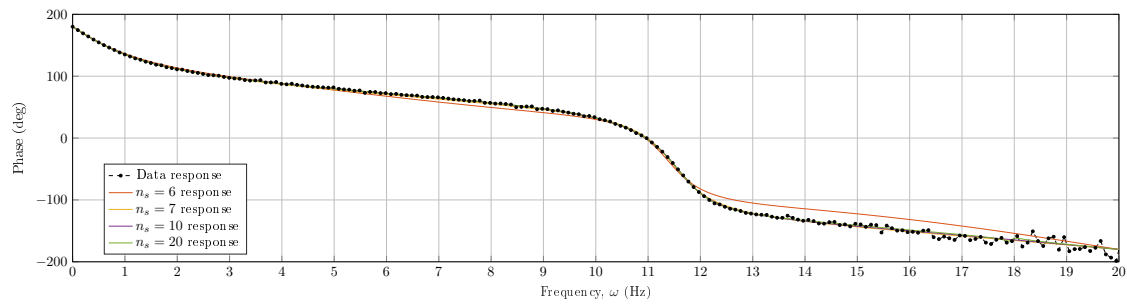
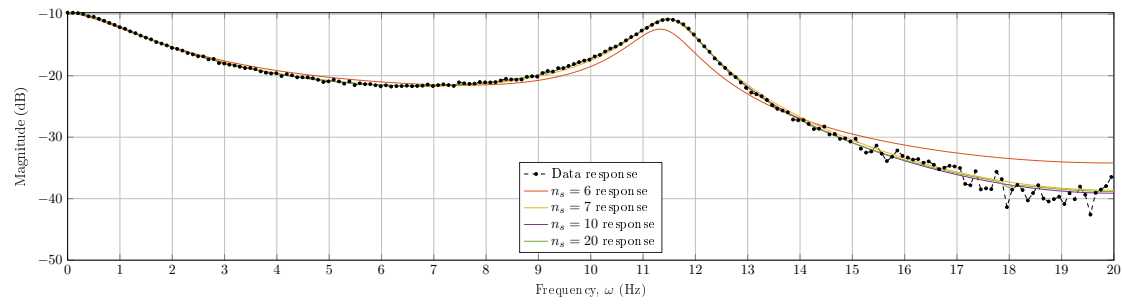


Figure 7: Frequency response of (1,2) channel

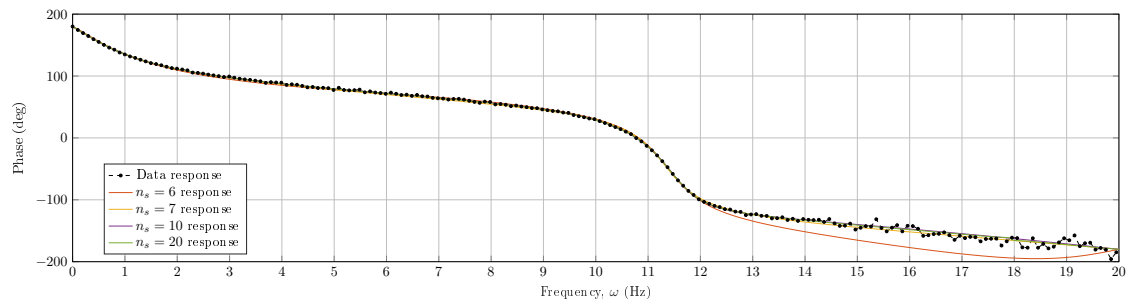
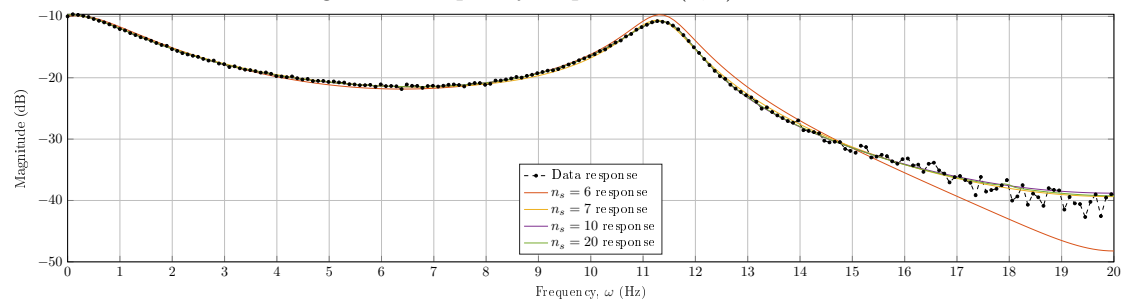


Figure 8: Frequency response of (2, 1) channel

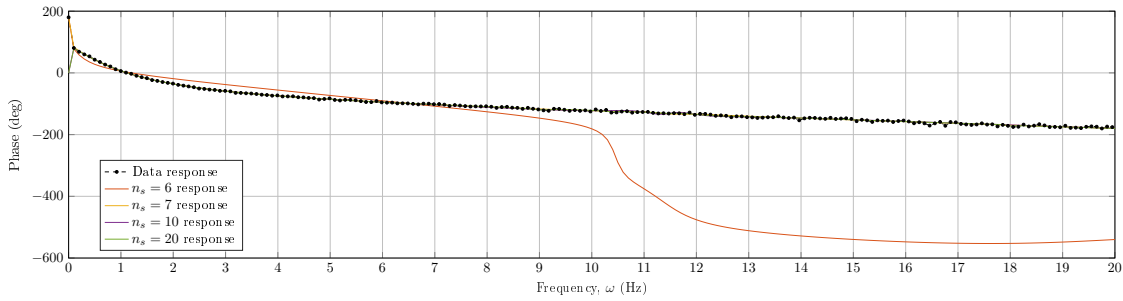
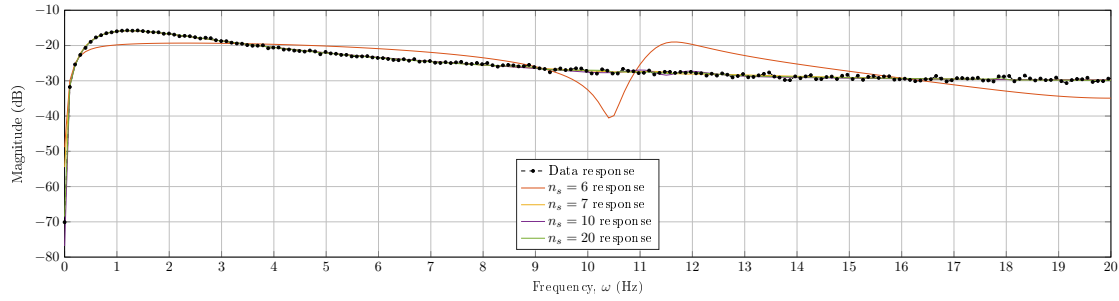
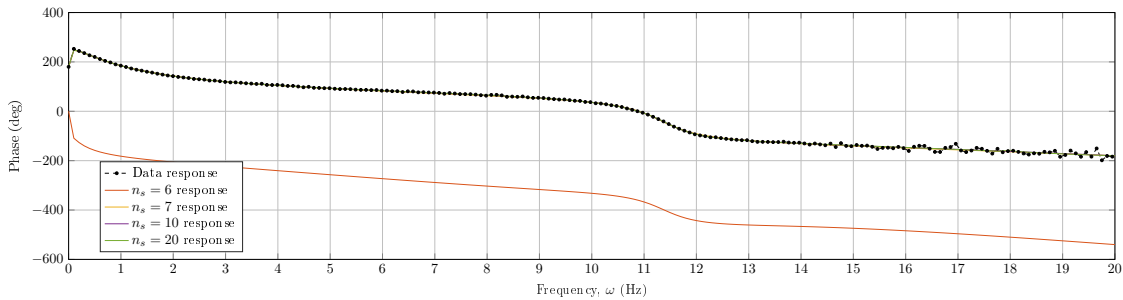
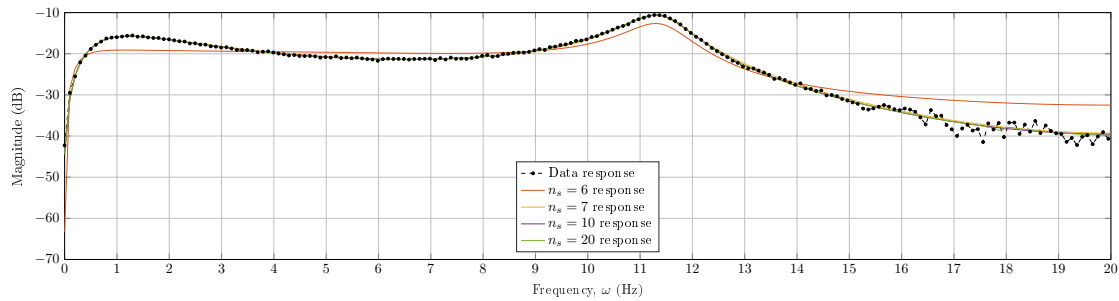


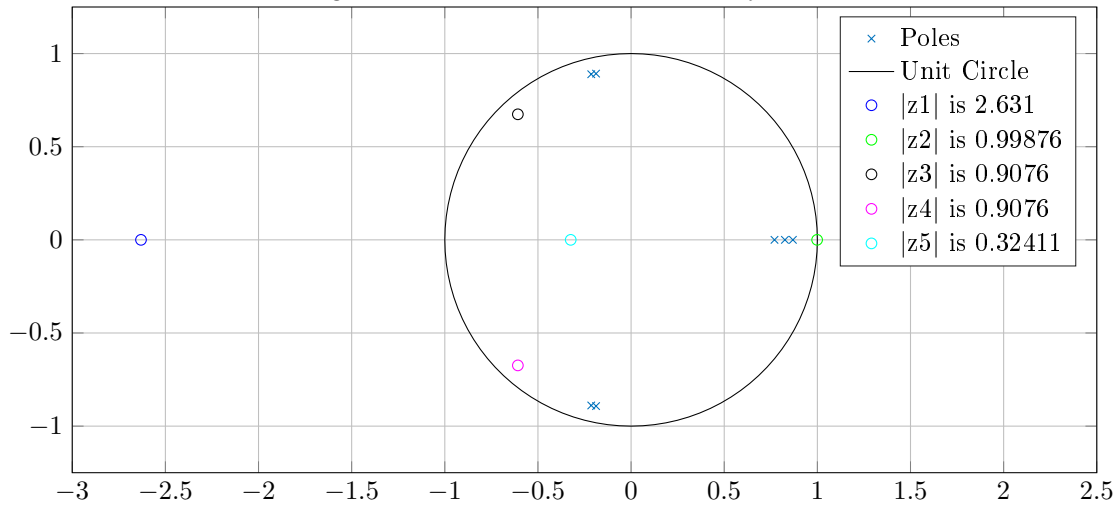
Figure 9: Frequency response of (2, 2) channel



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.

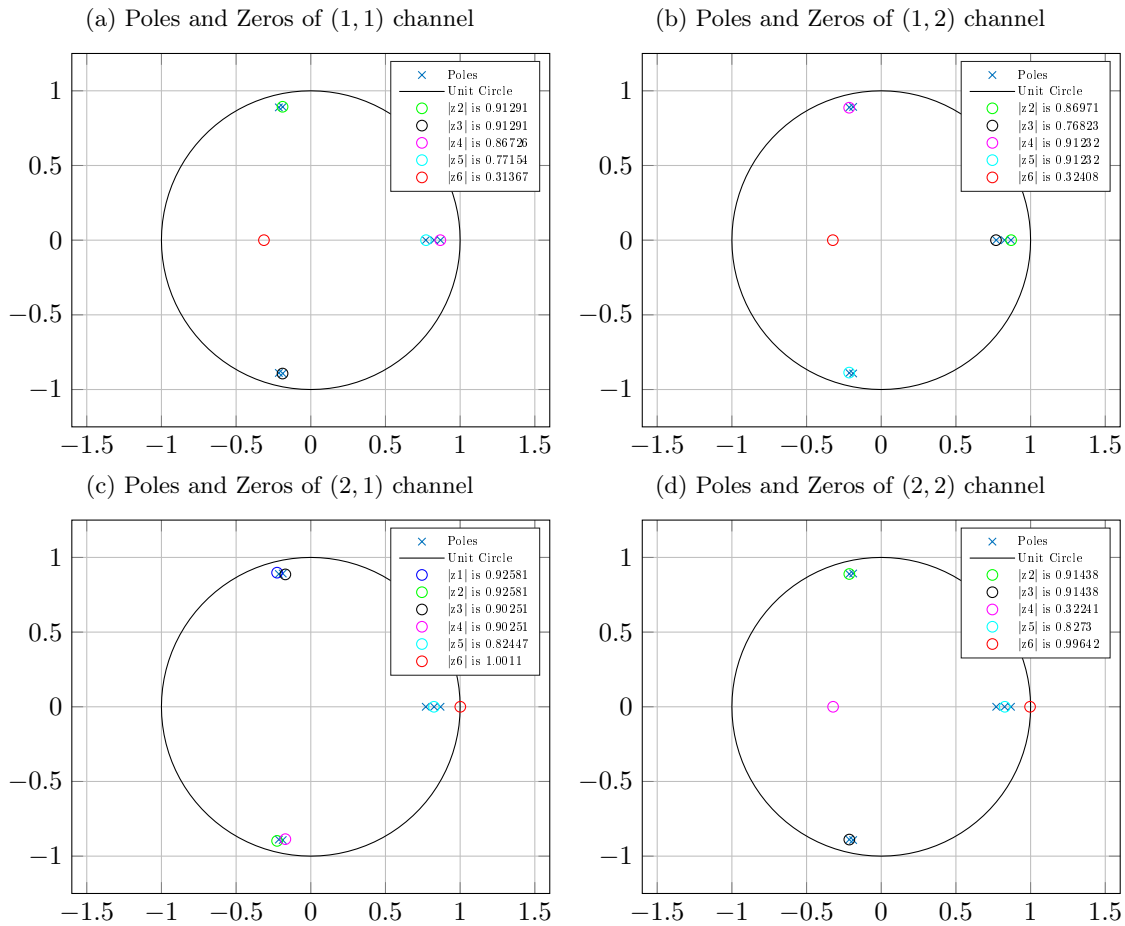
2.2. Task 2

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

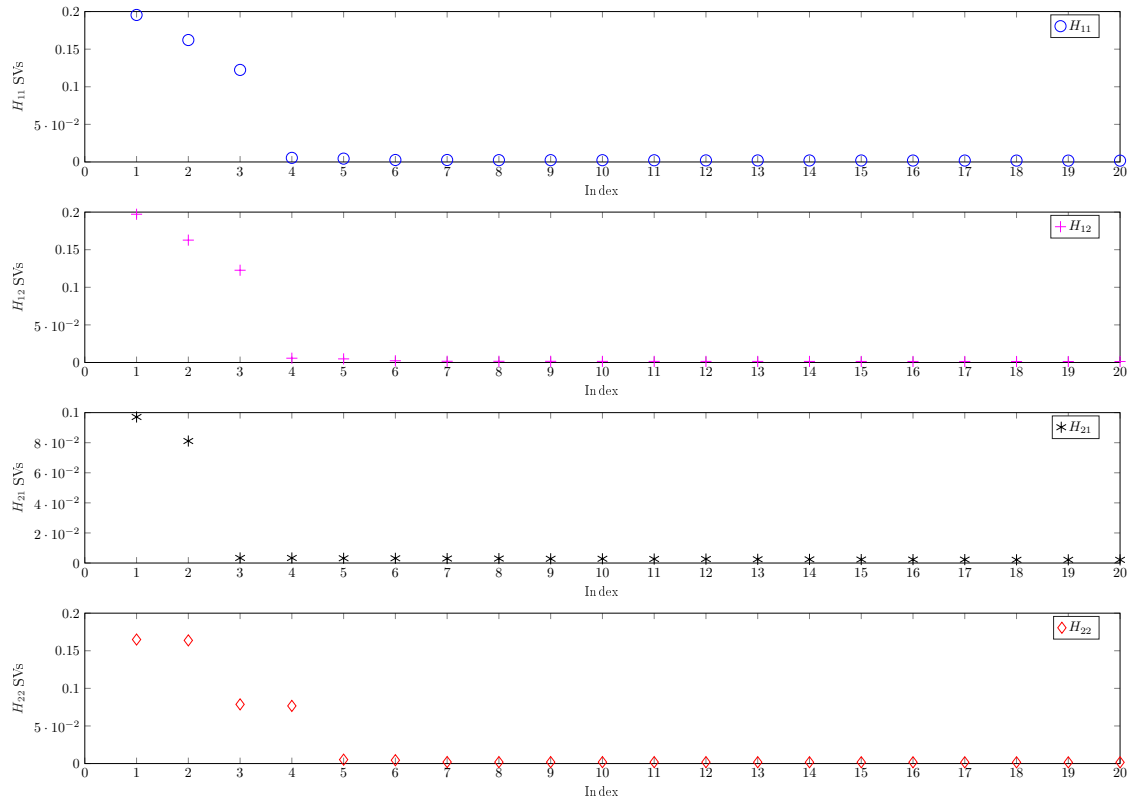
The five finite transmission zeros and their magnitudes are displayed in the figure above. In discrete-time systems, the asymptotically stable zeros are confined within the unit circle. In the $n_s = 7$ model, there was one unstable zero, denoted by $|z_1|$. Due to asymptotic stability of the model, 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 71.139rad/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 72.2737rad/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.

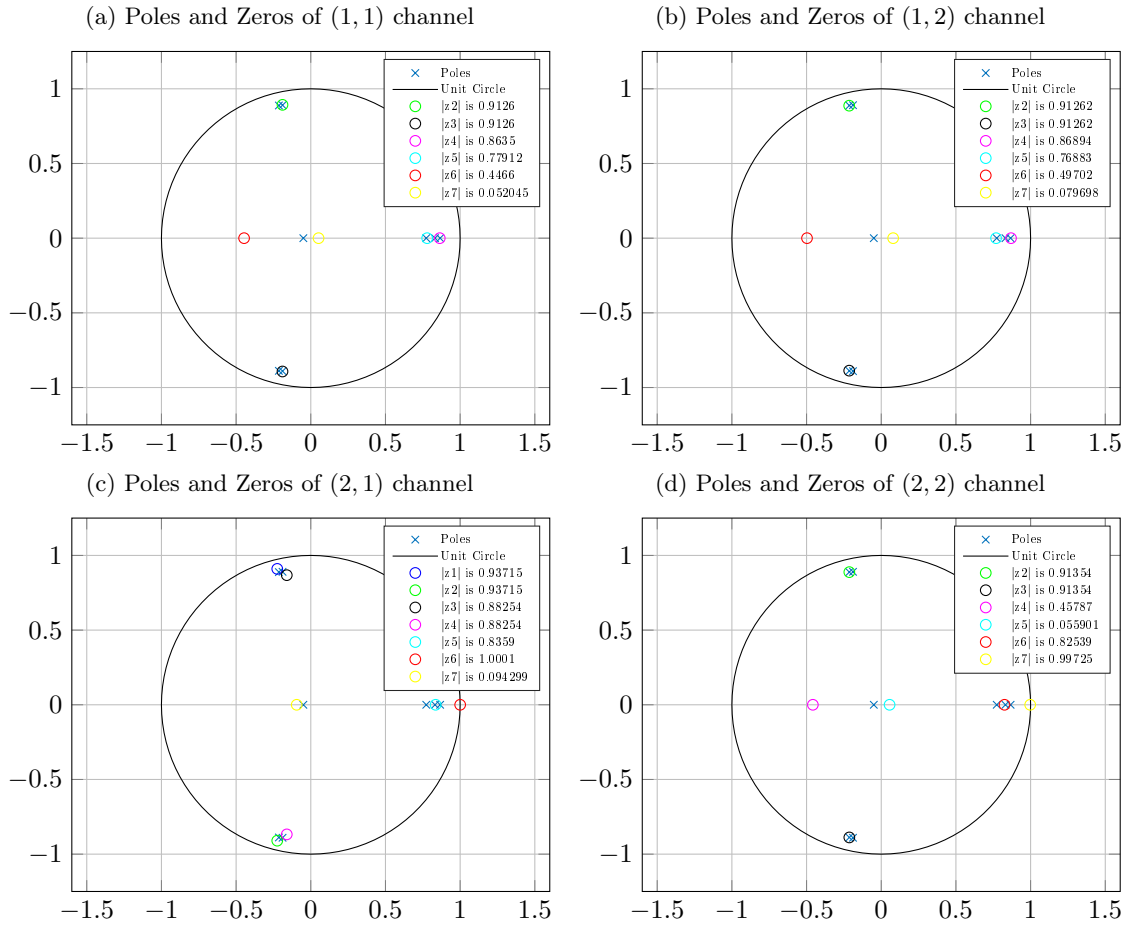
Figure 11: Pole zero plots for each channel of $n_s = 7$ system

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 uncanceled 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



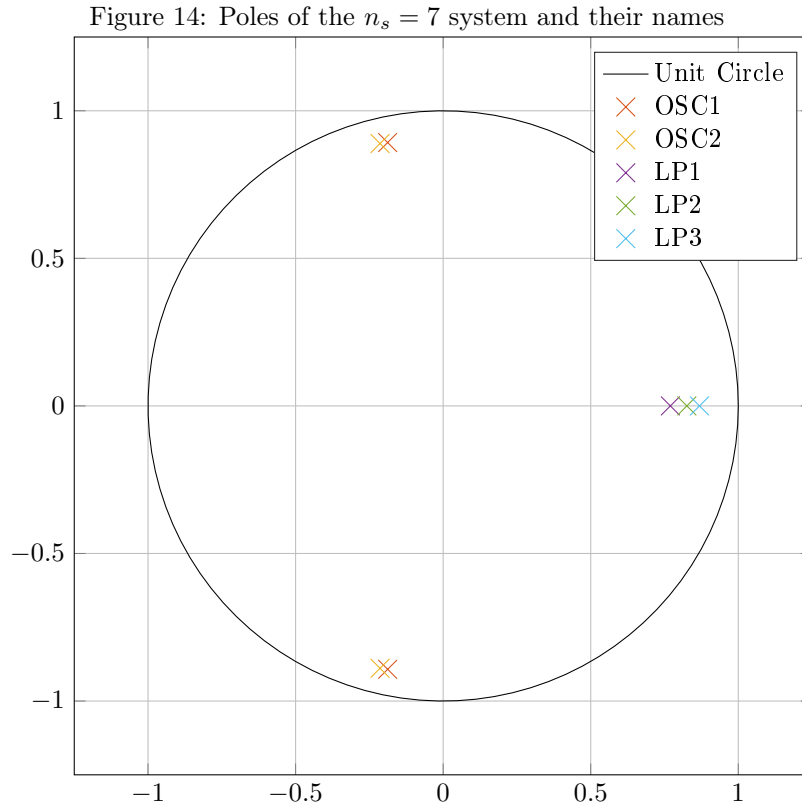
The discussion regarding the number of states in each input-output channel is reflected in the figures shown above. The Hankel singular values for the (1,1) and (1,2) channel show a dip after the first three values, confirming the three poles that were not cancelled in the pole-zero plots of the (1,1), 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

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.

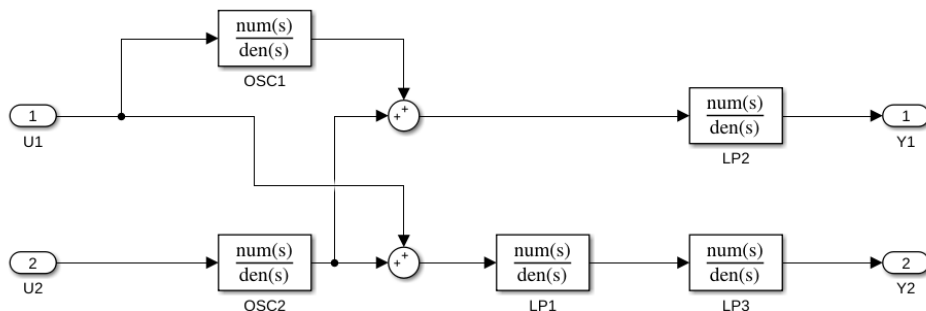
2.3. Task 3

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:



Starting with channel (1,1), it has one uncanceled oscillator in the pole-zero plot and one uncanceled 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 uncanceled oscillator, and one uncanceled 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 uncanceled. Therefore, one of those low pass filters is actually a high pass filter, although it cannot be determined which.

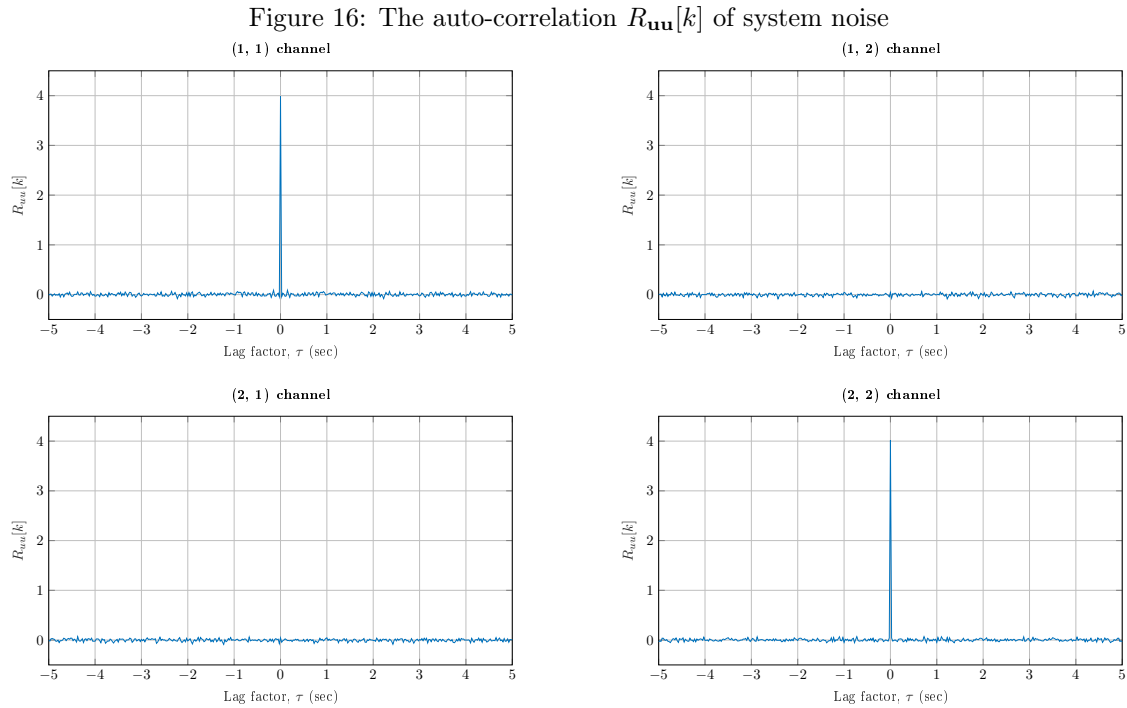
2.4. Task 4

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:

$$\mu(u_1) = -0.0009V$$

$$\mu(u_2) = 0.0013V$$

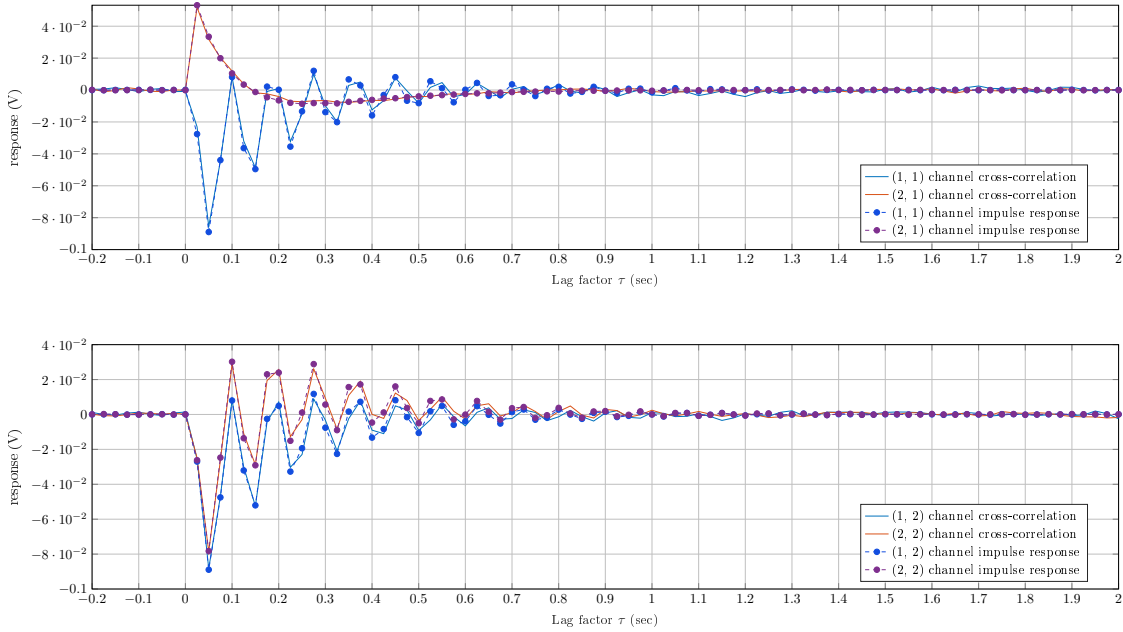
The auto-correlation of the input channels, $R_{\mathbf{uu}}[k]$, was found in order to determine the degree to which the inputs are correlated. Each channel of this auto-correlation was graphed for $\tau = [-5, 5]\text{sec}$



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 $R_{yu}[k]$ 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:

$$\begin{aligned} \|\mathbf{y}\|_{\text{RMS}} &= 0.2227 \\ \|P\|_{\mathcal{H}_2} \text{ (from } G_O) &= 0.2297 \\ \|P\|_{\mathcal{H}_2} \text{ (from } G_C) &= 0.2297 \\ \|P\|_{\mathcal{H}_2} \text{ (from pulse response)} &= 0.2298 \end{aligned}$$

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

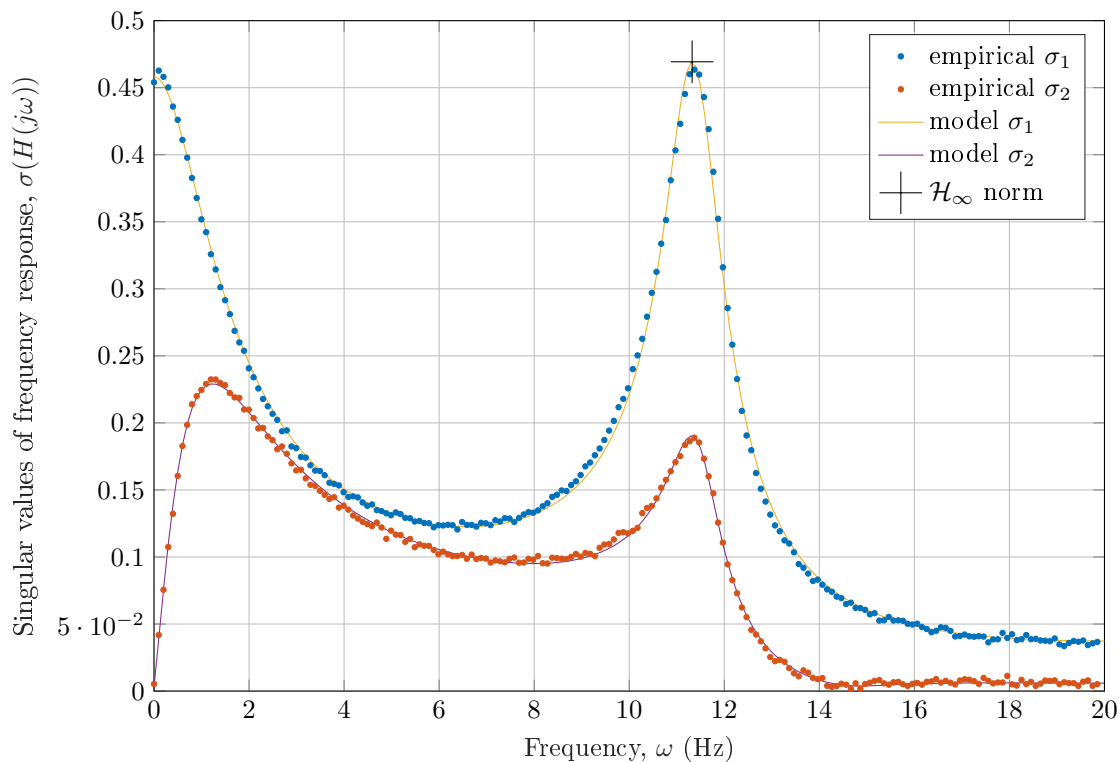
system, the norm and its corresponding frequency are found to be

$$\|P\|_{\mathcal{H}_\infty} = 0.4693$$

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

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}_∞ norm



A marker was placed at the frequency and magnitude corresponding to the \mathcal{H}_∞ 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

1	Task 1 code	15
2	Task 2 code	20
3	Task 3 code	26
4	Task 4 code	26
5	Task 5 code	28
6	Task 6 code	28
7	DiscreteModel file (used for tasks 3-6)	29
8	Hinf_cont function (used for task 6)	30
9	Hinf_dis function (used for task 6)	30

Listing 1: Task 1 code

```

1  %***** TASK 1 *****
2  clear
3  close all
4
5  load u1_impulse.mat
6  y11 = u1_impulse.Y(3).Data;
7  y21 = u1_impulse.Y(4).Data;
8  u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
9  [m,mi] = max(u1>0); %%% find index where pulse occurs
10
11 load u2_impulse.mat
12 y12 = u2_impulse.Y(3).Data;
13 y22 = u2_impulse.Y(4).Data;
14 u2 = u2_impulse.Y(2).Data;
15
16 %%% remove any offsets in output data using data prior to pulse application
17 y11 = y11 - mean(y11([1:mi-1]));
18 y12 = y12 - mean(y12([1:mi-1]));
19 y21 = y21 - mean(y21([1:mi-1]));
20 y22 = y22 - mean(y22([1:mi-1]));
21
22 %%% rescale IO data so that impulse input has magnitude 1
23 y11 = y11/max(u1);
24 y12 = y12/max(u2);
25 y21 = y21/max(u1);
26 y22 = y22/max(u2);
27 u1 = u1/max(u1);
28 u2 = u2/max(u2);
29
30 %***** TASK 1, Question 1 and 2 *****
31
32 ts = 1/40; %%% sample period
33 N = length(y11); %%% length of data sets
34 t = [0:N-1]*ts - 1;
35
36 m = 2; %number of output channels
37 q = 2; %number of input channels
38 % n s.t size(H) = mn X qn
39 n = 100;
40 %n = (401 - 41)/2 ; %n is half the number of samples beginning from t = 1
41
42 ind_off = 41;
43 H = zeros(m*n, q*n);
44 Htil = zeros(m*n, q*n);
45
46 %To create H and H tilda matrices
47 r = 1;
48 c = 1;
49 for k = 1 : n
50     ind = ind_off + k + [0:n-1];
51
52     H(r,[c:q:c*q*(n-1)]) = y11(ind);
53     H(r+1,[c:q:c*q*(n-1)]) = y21(ind);
54     H(r,[c+1:q:c+1*q*(n-1)]) = y12(ind);
55     H(r+1,[c+1:q:c+1*q*(n-1)]) = y22(ind);
56
57     Htil(r,[c:q:c*q*(n-1)]) = y11(ind+1);

```



```

58     Htil(r+1,[c:q:c+q*(n-1)]) = y21(ind+1);
59     Htil(r,[c+1:q:c+1+q*(n-1)]) = y12(ind+1);
60     Htil(r+1,[c+1:q:c+1+q*(n-1)]) = y22(ind+1);
61
62     r = r + m;
63 end
64
65 s_vals = svd(H);
66 figure(1);
67 plot(s_vals,'b.','MarkerSize',10)
68 xlabel('singular value index','FontSize',12);
69 ylabel('H_{100} singular values','FontSize',12);
70 grid on
71
72 %when ns = 6
73 ns = 6;
74 [U, S, V] = svd(H);
75 U1 = U([1:m*n],[1:ns]);
76 S1 = S([1:ns],[1:ns]);
77 V1 = V([1:n*q],[1:ns]);
78
79 On6 = U1 * sqrt(S1);
80 Cn6 = sqrt(S1) * V1';
81
82 On6inv = inv(sqrt(S1)) * U1';
83 Cn6inv = V1 * inv(sqrt(S1));
84
85 C6 = On6([1:m],:);
86 B6 = Cn6(:,[1:q]);
87 A6 = On6inv * Htil * Cn6inv;
88
89 mev6 = max(abs(eig(A6)));
90 fprintf('When ns = 6, dominant eval is: %f \n',mev6);
91
92 %Plotting impulse response
93 h = zeros(m,q,n);
94 x = B6;
95 for k = 1:n
96     h(:, :, k) = C6 * x;
97     x = A6 * x;
98 end
99
100 tsim = [1 : n] * ts;
101 figure()
102 subplot(211)
103 plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
104 xlabel('Time')
105 ylabel('y_{11}')
106 legend('Model6', 'Data')
107 grid on
108 axis([-0.2 2 -0.1 0.1])
109
110 subplot(212)
111 plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
112 xlabel('Time')
113 ylabel('y_{21}')
114 legend('Model6', 'Data')
115 grid on
116 axis([-0.2 2 -0.1 0.1])
117
118 figure()
119 subplot(211)
120 plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
121 xlabel('Time')
122 ylabel('y_{12}')
123 legend('Model6', 'Data')
124 grid on
125 axis([-0.2 2 -0.1 0.1])
126
127 subplot(212)
128 plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
129 xlabel('Time')
130 ylabel('y_{22}')
131 legend('Model6', 'Data')
132 grid on
133 axis([-0.2 2 -0.1 0.1])
134
135 %when ns = 7
136 ns = 7;
137 [U, S, V] = svd(H);
138 U1 = U([1:m*n],[1:ns]);
139 S1 = S([1:ns],[1:ns]);
140 V1 = V([1:n*q],[1:ns]);
141
142 On7 = U1 * sqrt(S1);
143 Cn7 = sqrt(S1) * V1';

```

```

144
145 On7inv = inv(sqrt(S1)) * U1';
146 Cn7inv = V1 * inv(sqrt(S1));
147
148 C7 = On7([1:m],:);
149 B7 = Cn7(:,[1:q]);
150 A7 = On7inv * Htil * Cn7inv;
151
152 mev7 = max(abs(eig(A7)));
153 fprintf('When ns = 7, dominant eval is: %f \n',mev7);
154
155 %Plotting impulse response for model7
156 h = zeros(m,q,n);
157 x = B7;
158 for k = 1:n
159     h(:,k) = C7 * x;
160     x = A7 * x;
161 end
162
163 tsim = [1 : n] * ts;
164 figure()
165 subplot(211)
166 plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
167 xlabel('Time')
168 ylabel('y_{11}')
169 legend('Model7', 'Data')
170 grid on
171 axis([-0.2 2 -0.1 0.1])
172
173 subplot(212)
174 plot(tsim,squeeze(h(2,1,:)),'b*', t, y21, 'g*')
175 xlabel('Time')
176 ylabel('y_{21}')
177 legend('Model7', 'Data')
178 grid on
179 axis([-0.2 2 -0.1 0.1])
180
181 figure()
182 subplot(211)
183 plot(tsim,squeeze(h(1,2,:)),'b*', t, y12, 'g*')
184 xlabel('Time')
185 ylabel('y_{12}')
186 legend('Model7', 'Data')
187 grid on
188 axis([-0.2 2 -0.1 0.1])
189
190 subplot(212)
191 plot(tsim,squeeze(h(2,2,:)),'b*', t, y22, 'g*')
192 xlabel('Time')
193 ylabel('y_{22}')
194 legend('Model7', 'Data')
195 grid on
196 axis([-0.2 2 -0.1 0.1])
197
198 %when ns = 10
199 ns = 10;
200 [U, S, V] = svd(H);
201 U1 = U([1:m*n],[1:ns]);
202 S1 = S([1:ns],[1:ns]);
203 V1 = V([1:n*q],[1:ns]);
204
205 On10 = U1 * sqrt(S1);
206 Cn10 = sqrt(S1) * V1';
207
208 On10inv = inv(sqrt(S1)) * U1';
209 Cn10inv = V1 * inv(sqrt(S1));
210
211 C10 = On10([1:m],:);
212 B10 = Cn10(:,[1:q]);
213 A10 = On10inv * Htil * Cn10inv;
214
215 mev10 = max(abs(eig(A10)));
216 fprintf('When ns = 10, dominant eval is: %f \n',mev10);
217
218 h = zeros(m,q,n);
219 x = B10;
220 for k = 1:n
221     h(:,k) = C10 * x;
222     x = A10 * x;
223 end
224
225 tsim = [1 : n] * ts;
226 figure()
227 subplot(211)
228 plot(tsim,squeeze(h(1,1,:)),'b*', t, y11, 'g*')
229 xlabel('Time')

```

```

230 ylabel('y_{11}')
231 legend('Model10', 'Data')
232 grid on
233 axis([-0.2 2 -0.1 0.1])
234
235 subplot(212)
236 plot(tsim,squeeze(h(2,1,:)), 'b*', t, y21, 'g*')
237 xlabel('Time')
238 ylabel('y_{21}')
239 legend('Model10', 'Data')
240 grid on
241 axis([-0.2 2 -0.1 0.1])
242
243 figure()
244 subplot(211)
245 plot(tsim,squeeze(h(1,2,:)), 'b*', t, y12, 'g*')
246 xlabel('Time')
247 ylabel('y_{12}')
248 legend('Model10', 'Data')
249 grid on
250 axis([-0.2 2 -0.1 0.1])
251
252 subplot(212)
253 plot(tsim,squeeze(h(2,2,:)), 'b*', t, y22, 'g*')
254 xlabel('Time')
255 ylabel('y_{22}')
256 legend('Model10', 'Data')
257 grid on
258 axis([-0.2 2 -0.1 0.1])
259
260 %when ns = 20
261 ns = 20;
262 [U, S, V] = svd(H);
263 U1 = U([1:m*n],[1:ns]);
264 S1 = S([1:ns],[1:ns]);
265 V1 = V([1:n*q],[1:ns]);
266
267 On20 = U1 * sqrt(S1);
268 Cn20 = sqrt(S1) * V1';
269
270 On20inv = inv(sqrt(S1)) * U1';
271 Cn20inv = V1 * inv(sqrt(S1));
272
273 C20 = On20([1:m],:);
274 B20 = Cn20(:, [1:q]);
275 A20 = On20inv * Htil * Cn20inv;
276
277 mev20 = max(abs(eig(A20)));
278 fprintf('When ns = 20, dominant eval is: %f \n', mev20);
279
280 h = zeros(m,q,n);
281 x = B20;
282 for k = 1:n
283     h(:, :, k) = C20 * x;
284     x = A20 * x;
285 end
286
287 tsim = [1 : n] * ts;
288 figure()
289 subplot(211)
290 plot(tsim,squeeze(h(1,1,:)), 'b*', t, y11, 'g*')
291 xlabel('Time')
292 ylabel('y_{11}')
293 legend('Model20', 'Data')
294 grid on
295 axis([-0.2 2 -0.1 0.1])
296
297 subplot(212)
298 plot(tsim,squeeze(h(2,1,:)), 'b*', t, y21, 'g*')
299 xlabel('Time')
300 ylabel('y_{21}')
301 legend('Model20', 'Data')
302 grid on
303 axis([-0.2 2 -0.1 0.1])
304
305 figure()
306 subplot(211)
307 plot(tsim,squeeze(h(1,2,:)), 'b*', t, y12, 'g*')
308 xlabel('Time')
309 ylabel('y_{12}')
310 legend('Model20', 'Data')
311 grid on
312 axis([-0.2 2 -0.1 0.1])
313
314 subplot(212)
315 plot(tsim,squeeze(h(2,2,:)), 'b*', t, y22, 'g*')

```

```

316 xlabel('Time')
317 ylabel('y_{22}')
318 legend('Model20','Data')
319 grid on
320 axis([-0.2 2 -0.1 0.1])
321
322 fprintf('Since all dominant evals have a magnitude less than 1, all models are asymptotically stable');
323
324 %***** TASK 1, Question 3 and 4 *****
325 %Computing Frequency response
326 wnyq = 20;
327 w = [0:wnyq/100:wnyq];%w in Hertz
328 Hf6 = zeros(m,q,length(w));
329 Hf7 = zeros(m,q,length(w));
330 Hf10 = zeros(m,q,length(w));
331 Hf20 = zeros(m,q,length(w));
332
333 for k = 1:length(w)
334     Hf6(:,k) = C6 * inv(exp(1j*2*pi*w(k)*ts)*eye(6) - A6) * B6;
335     Hf7(:,k) = C7 * inv(exp(1j*2*pi*w(k)*ts)*eye(7) - A7) * B7;
336     Hf10(:,k) = C10 * inv(exp(1j*2*pi*w(k)*ts)*eye(10) - A10) * B10;
337     Hf20(:,k) = C20 * inv(exp(1j*2*pi*w(k)*ts)*eye(20) - A20) * B20;
338 end
339
340 %Computing emperical frequency response
341 y11f = fft(y11)./fft(u1);
342 N = length(y11f);
343 om = [0:N-1]/(ts*N); %%% frequency vector in hertz
344 y21f = fft(y21)./fft(u1);
345 y12f = fft(y12)./fft(u2);
346 y22f = fft(y22)./fft(u2);
347
348 %Plotting abs(Freq response of channel (1,1))
349 figure()
350 plot(w, abs(squeeze(Hf6(1,1,:))), 'b-')
351 hold on
352 plot(w, abs(squeeze(Hf7(1,1,:))), 'r-')
353 plot(w, abs(squeeze(Hf10(1,1,:))), 'g-')
354 plot(w, abs(squeeze(Hf20(1,1,:))), 'm-')
355 plot(om,abs(y11f), 'k--')
356 ylabel('|H(w)| (1,1)')
357 xlabel('Frequency (Hz)')
358 xlim([0 20])
359 legend('Model6','Model7','Model10','Model20','Data')
360 title('Freq Response magnitude of (1,1) channel')
361 grid on
362 hold off
363
364 %Plotting abs(Freq response) of channel (2,1)
365 figure()
366 plot(w, abs(squeeze(Hf6(2,1,:))), 'b-')
367 hold on
368 plot(w, abs(squeeze(Hf7(2,1,:))), 'r-')
369 plot(w, abs(squeeze(Hf10(2,1,:))), 'g-')
370 plot(w, abs(squeeze(Hf20(2,1,:))), 'm-')
371 plot(om,abs(y21f), 'k--')
372 xlabel('Frequency (Hz)')
373 xlim([0 20])
374 ylabel('|H(w)| (2,1)')
375 legend('Model6','Model7','Model10','Model20','Data')
376 title('Freq Response magnitude of (2,1) channel')
377 grid on
378 hold off
379
380 %Plotting abs(Freq response) of channel (1,2)
381 figure()
382 plot(w, abs(squeeze(Hf6(1,2,:))), 'b-')
383 hold on
384 plot(w, abs(squeeze(Hf7(1,2,:))), 'r-')
385 plot(w, abs(squeeze(Hf10(1,2,:))), 'g-')
386 plot(w, abs(squeeze(Hf20(1,2,:))), 'm-')
387 plot(om,abs(y12f), 'k--')
388 xlabel('Frequency (Hz)')
389 xlim([0 20])
390 ylabel('|H(w)| (1,2)')
391 legend('Model6','Model7','Model10','Model20','Data')
392 title('Freq Response magnitude of (1,2) channel')
393 grid on
394 hold off
395
396 %Plotting abs(Freq response) of channel (2,2)
397 figure()
398 plot(w, abs(squeeze(Hf6(2,2,:))), 'b-')
399 hold on
400 plot(w, abs(squeeze(Hf7(2,2,:))), 'r-')
401 plot(w, abs(squeeze(Hf10(2,2,:))), 'g-')

```

```

402 plot(w, abs(squeeze(Hf20(2,2,:))), 'm-')
403 plot(om,abs(y22f), 'k--')
404 xlabel('Frequency (Hz)')
405 xlim([0 20])
406 ylabel('|H(w)| (2,2)')
407 legend('Model6','Model7','Model10','Model20','Data')
408 title('Freq Response magnitude of (2,2) channel')
409 grid on
410 hold off
411
412 %Plotting angle(Freq response of channel (1,1))
413 figure()
414 plot(w, angle(squeeze(Hf6(1,1,:))), 'b-')
415 hold on
416 plot(w, angle(squeeze(Hf7(1,1,:))), 'r-')
417 plot(w, angle(squeeze(Hf10(1,1,:))), 'g-')
418 plot(w, angle(squeeze(Hf20(1,1,:))), 'm-')
419 plot(om,angle(y11f), 'k--')
420 xlabel('Frequency (Hz)')
421 xlim([0 20])
422 ylabel('angle(H(w)) (1,1)')
423 legend('Model6','Model7','Model10','Model20','Data')
424 title('Freq Response angle of (1,1) channel')
425 grid on
426 hold off
427
428 %Plotting angle(Freq response) of channel (2,1)
429 figure()
430 plot(w, angle(squeeze(Hf6(2,1,:))), 'b-')
431 hold on
432 plot(w, angle(squeeze(Hf7(2,1,:))), 'r-')
433 plot(w, angle(squeeze(Hf10(2,1,:))), 'g-')
434 plot(w, angle(squeeze(Hf20(2,1,:))), 'm-')
435 plot(om,angle(y21f), 'k--')
436 xlabel('Frequency (Hz)')
437 xlim([0 20])
438 ylabel('angle(H(w)) (2,1)')
439 legend('Model6','Model7','Model10','Model20','Data')
440 title('Freq Response angle of (2,1) channel')
441 grid on
442 hold off
443
444 %Plotting angle(Freq response) of channel (1,2)
445 figure()
446 plot(w, angle(squeeze(Hf6(1,2,:))), 'b-')
447 hold on
448 plot(w, angle(squeeze(Hf7(1,2,:))), 'r-')
449 plot(w, angle(squeeze(Hf10(1,2,:))), 'g-')
450 plot(w, angle(squeeze(Hf20(1,2,:))), 'm-')
451 plot(om,angle(y12f), 'k--')
452 xlabel('Frequency (Hz)')
453 xlim([0 20])
454 ylabel('angle(H(w)) (1,2)')
455 legend('Model6','Model7','Model10','Model20','Data')
456 title('Freq Response angle of (1,2) channel')
457 grid on
458 hold off
459
460 %Plotting angle(Freq response) of channel (2,2)
461 figure()
462 plot(w, angle(squeeze(Hf6(2,2,:))), 'b-')
463 hold on
464 plot(w, angle(squeeze(Hf7(2,2,:))), 'r-')
465 plot(w, angle(squeeze(Hf10(2,2,:))), 'g-')
466 plot(w, angle(squeeze(Hf20(2,2,:))), 'm-')
467 plot(om,angle(y22f), 'k--')
468 xlabel('Frequency (Hz)')
469 xlim([0 20])
470 ylabel('angle(H(w)) (2,2)')
471 legend('Model6','Model7','Model10','Model20','Data')
472 title('Freq Response phase of (2,2) channel')
473 grid on
474 hold off

```

Listing 2: Task 2 code

```

1 %***** TASK 2 *****
2 clear
3 close all
4 clc
5
6 load u1_impulse.mat
7 y11 = u1_impulse.Y(3).Data;
8 y21 = u1_impulse.Y(4).Data;
9 u1 = u1_impulse.Y(1).Data; %% note that the pulse magnitude is 5

```

```

10  [~, mi] = max(u1>0); %%% find index where pulse occurs
11
12  load u2_impulse.mat
13  y12 = u2_impulse.Y(3).Data;
14  y22 = u2_impulse.Y(4).Data;
15  u2 = u2_impulse.Y(2).Data;
16
17  %%% remove any offsets in output data using data prior to pulse application
18  y11 = y11 - mean(y11(1:mi-1));
19  y12 = y12 - mean(y12(1:mi-1));
20  y21 = y21 - mean(y21(1:mi-1));
21  y22 = y22 - mean(y22(1:mi-1));
22
23  %%% rescale IO data so that impulse input has magnitude 1
24  y11 = y11/max(u1);
25  y12 = y12/max(u2);
26  y21 = y21/max(u1);
27  y22 = y22/max(u2);
28
29  %***** TASK 2, Question 1 and 2 *****
30
31  ts = 1/40; %%% sample period
32
33  m = 2; %number of output channels
34  q = 2; %number of input channels
35  n = 100; % n s.t size(H) = mn X qn
36
37  ind_off = 41;
38  H = zeros(m*n, q*n);
39  Htil = zeros(m*n, q*n);
40
41  %To create H and H tilde matrices
42  r = 1;
43  c = 1;
44  for k = 1 : n
45      ind = ind_off + k + (0:n-1);
46
47      H(r,c:q:c*q*(n-1)) = y11(ind);
48      H(r+1,c:q:c*q*(n-1)) = y21(ind);
49      H(r,c+1:q:c+1+q*(n-1)) = y12(ind);
50      H(r+1,c+1:q:c+1+q*(n-1)) = y22(ind);
51
52      Htil(r,c:q:c*q*(n-1)) = y11(ind+1);
53      Htil(r+1,c:q:c*q*(n-1)) = y21(ind+1);
54      Htil(r,c+1:q:c+1+q*(n-1)) = y12(ind+1);
55      Htil(r+1,c+1:q:c+1+q*(n-1)) = y22(ind+1);
56
57      r = r + m;
58  end
59
60  %when ns = 7
61  ns = 7;
62  [U, S, V] = svd(H);
63  U1 = U(1:m*n, 1:ns);
64  S1 = S(1:ns, 1:ns);
65  V1 = V(1:n*q, 1:ns);
66
67  On7 = U1 * sqrt(S1);
68  Cn7 = sqrt(S1) * V1';
69
70  On7inv = inv(sqrt(S1)) * U1';
71  Cn7inv = V1 * inv(sqrt(S1));
72
73  C7 = On7([1:m],:);
74  B7 = Cn7(:,[1:q]);
75  A7 = On7inv * Htil * Cn7inv;
76
77  M = [A7 B7; -C7 zeros(2)];
78  N = [eye(7) zeros(7,2); zeros(2,7) zeros(2)]; %disp(diag(D))%Resulted in D(1) and D(8) as inf
79  [~, D] = eig(M, N);
80
81
82  %2,3,4,5,6 elements of the diagonal are not inf and are in descending
83  %order. Elements 4 and 5 of the diagonal are equal
84  z1 = D(2,2); z2 = D(3,3); z3 = D(4,4); z4 = D(5,5); z5 = D(6,6);
85  disp('The finite transmission zeros are\n')
86  fprintf('The five finite transmission zeros are:\n')
87  fprintf('z1 is %f + %fi and |z1| is %f \n',real(z1),imag(z1),abs(z1))
88  fprintf('z2 is %f + %fi and |z2| is %f \n',real(z2),imag(z2),abs(z2))
89  fprintf('z3 is %f + %fi and |z3| is %f \n',real(z3),imag(z3),abs(z3))
90  fprintf('z4 is %f + %fi and |z4| is %f \n',real(z4),imag(z4),abs(z4))
91  fprintf('z5 is %f + %fi and |z5| is %f \n',real(z5),imag(z5),abs(z5))
92
93  eigA7 = eig(A7);
94  ut = 0:2*pi/100:2*pi;
95  figure()

```

```

96 plot(eigA7,'x','DisplayName','Poles');
97 hold on
98 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
99 plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
100 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
101 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
102 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
103 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
104 title('Poles and Zeros')
105 legend
106 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
107 axis([-3 2.5 -1 1])
108 axis equal
109 grid on
110 hold off
111
112 %***** TASK 2, Question 3 *****
113 %Converting discrete evals to continuous evals
114 eigc_real = 1/ts * log(abs(eigA7));
115 eigc_im = 1/ts * angle(eigA7);
116 eigc = eigc_real + 1j * eigc_im;
117
118 fprintf('\n lambda_continuous \n');
119 disp(eigc);
120
121 disp('eigc([1,2,3,4]) have Re(.) < 0')
122 disp('There are two complex conjugate pairs with negative real parts (2 damped oscillators)')
123 fprintf('\n Oscillator 1 has a freq %f rad/sec = %f hertz\n',imag(eigc(1)),imag(eigc(1))/(2*pi));
124
125 %***** TASK 2, Question 4 *****
126
127 %For 1-1 channel the system is
128 B11 = B7(:,1);
129 C11 = C7(1,:);
130 M = [A7 B11; -C11 zeros(1)];
131 N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
132 [~, D] = eig(M,N);
133 z11 = diag(D);
134 z11 = z11([2,3,4,5,6,7]);
135 disp('Finite zeros for 1-1 channel:')
136 disp(z11)
137 lic = num2cell(z11);
138 [~, z2, z3, z4, z5, z6]=deal(lic{:});
139 figure()
140 plot(eigA7,'x','DisplayName','Poles');
141 hold on
142 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
143 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
144 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
145 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
146 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
147 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
148 title('Poles and Zeros 1-1 channel')
149 legend
150 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
151 axis([-1 1 -1 1])
152 axis equal
153 grid on
154 hold off
155
156 %For 1-2 channel the system is
157 B12 = B7(:,2);
158 C12 = C7(1,:);
159 M = [A7 B12; -C12 zeros(1)];
160 N = [eye(7) zeros(7,1);zeros(1,7) zeros(1)];
161 [~, D] = eig(M,N);
162 z12 = diag(D);
163 z12= z12([2,3,4,5,6,7]);
164 disp('Finite zeros for 1-2 channel:')
165 disp(z12)
166 lic = num2cell(z12);
167 [~, z2, z3, z4, z5, z6]=deal(lic{:});
168 figure()
169 plot(eigA7,'x','DisplayName','Poles');
170 hold on
171 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
172 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
173 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
174 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
175 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
176 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
177 title('Poles and Zeros 1-2 channel')
178 legend
179 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
180 axis([-1 1 -1 1])
181 axis equal

```

```

182 grid on
183 hold off
184
185 %For 2-1 channel
186 B21 = B7(:,1);
187 C21 = C7(2,:);
188 M = [A7 B21; -C21 zeros(1)];
189 N = [eye(7) zeros(7,1); zeros(1,7) zeros(1)];
190 [~, D] = eig(M,N);
191 z21 = diag(D);
192 z21 = z21([2,3,4,5,6,7]);
193 disp('Finite zeros for 2-1 channel:')
194 disp(z21)
195 lic = num2cell(z21);
196 [z1, z2, z3, z4, z5, z6]=deal(lic{:});
197 figure()
198 plot(eigA7,'x','DisplayName','Poles');
199 hold on
200 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
201 plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
202 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
203 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
204 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
205 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
206 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
207 title('Poles and Zeros 2-1 channel')
208 legend
209 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
210 axis([-1 1 -1 1])
211 axis equal
212 grid on
213 hold off
214
215 %For 2-2 channel
216 B22 = B7(:,2);
217 C22 = C7(2,:);
218 M = [A7 B22; -C22 zeros(1)];
219 N = [eye(7) zeros(7,1); zeros(1,7) zeros(1)];
220 [~, D] = eig(M,N);
221 z22 = diag(D);
222 z22 = z22([2,3,4,5,6,7]);
223 disp('Finite zeros for 2-2 channel:')
224 disp(z22)
225 lic = num2cell(z22);
226 [z2, z3, z4, z5, z6]=deal(lic{:});
227 figure()
228 plot(eigA7,'x','DisplayName','Poles');
229 hold on
230 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
231 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
232 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
233 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
234 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
235 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
236 title('Poles and Zeros 2-2 channel')
237 legend
238 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
239 axis([-1 1 -1 1])
240 axis equal
241 grid on
242 hold off
243
244 %Hankel Matrix for 1-1 channel
245 m = 1; q = 1;
246 H11 = zeros(m*n, q*n);
247 H12 = zeros(m*n, q*n);
248 H21 = zeros(m*n, q*n);
249 H22 = zeros(m*n, q*n);
250 r = 1;
251 c = 1;
252 for k = 1 : n
253     ind = ind_off + k + [0:n-1];
254
255     H11(r,:) = y11(ind);
256     H12(r,:) = y12(ind);
257     H21(r,:) = y21(ind);
258     H22(r,:) = y22(ind);
259
260     r = r + m;
261 end
262
263 figure()
264 subplot(411)
265 temp = svd(H11);
266 disp('The first 5 Hankel singular values for the 1-1 channel are')
267 disp(temp([1:5]));

```



```

268 plot(temp,'bo','DisplayName','H11');
269 ylabel('H11 sing vals');
270 xlabel('Index');xlim([0,20]);
271 legend
272 hold on
273
274 subplot(412)
275 temp = svd(H12);
276 disp('The first 5 Hankel singular values for the 1-2 channel are')
277 disp(temp([1:5]));
278 plot(temp,'m+','DisplayName','H12');
279 ylabel('H12 sing vals');
280 xlabel('Index');xlim([0,20]);
281 legend
282 hold on
283
284 subplot(413)
285 temp = svd(H21);
286 disp('The first 5 Hankel singular values for the 2-1 channel are')
287 disp(temp([1:5]));
288 plot(temp,'k+','DisplayName','H21');
289 ylabel('H21 sing vals');
290 xlabel('Index');
291 xlim([0,20]);
292 legend
293 hold on
294
295 subplot(414)
296 temp = svd(H22);
297 disp('The first 5 Hankel singular values for the 1-2 channel are')
298 disp(temp([1:5]));
299 plot(temp,'rd','DisplayName','H22');
300 ylabel('H22 sing vals');
301 xlabel('Index');
302 xlim([0,20]);
303 legend
304 hold on
305
306 %***** TASK 2, Question 5 *****
307
308 %Generating A,B, and C for ns = 8
309 m = 2; %number of output channels
310 q = 2; %number of input channels
311 ns = 8;
312 [U, S, V] = svd(H);
313 U1 = U([1:m*n],[1:ns]);
314 S1 = S([1:ns],[1:ns]);
315 V1 = V([1:n*q],[1:ns]);
316
317 On8 = U1 * sqrt(S1);
318 Cn8 = sqrt(S1) * V1';
319
320 On8inv = inv(sqrt(S1)) * U1';
321 Cn8inv = V1 * inv(sqrt(S1));
322
323 C8 = On8([1:m],:);
324 B8 = Cn8(:,[1:q]);
325 A8 = On8inv * Htil * Cn8inv;
326 eigA8 = eig(A8);
327
328 %For 1-1 channel the system is
329 B11 = B8(:,1);
330 C11 = C8(1,:);
331 M = [A8 B11; -C11 zeros(1)];
332 N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
333 [~, D] = eig(M,N);
334 z11 = diag(D);
335 z11 = z11([2,3,4,5,6,7,8]);
336 disp('Finite zeros for 1-1 channel:')
337 disp(z11)
338 lic = num2cell(z11);
339 [z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
340 figure()
341 plot(eigA8,'x','DisplayName','Poles');
342 hold on
343 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
344 %plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
345 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
346 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
347 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
348 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
349 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
350 plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
351 title('ns = 8, Poles and Zeros 1-1 channel')
352 legend
353 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')

```

```

354 axis([-1 1 -1 1])
355 axis equal
356 grid on
357 hold off
358
359 %For 1-2 channel the system is
360 B12 = B8(:,2);
361 C12 = C8(1,:);
362 M = [A8 B12; -C12 zeros(1)];
363 N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
364 [V,D] = eig(M,N);
365 %disp(diag(D))%Resulted in D(1) and D(9) as inf
366 z12 = diag(D);
367 z12 = z12([2,3,4,5,6,7,8]);
368 disp('Finite zeros for 1-2 channel:')
369 disp(z12)
370 lic = num2cell(z12);
371 [z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
372 figure()
373 plot(eigA8,'x','DisplayName','Poles');
374 hold on
375 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
376 %plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
377 %plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
378 %plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
379 %plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
380 %plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
381 %plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
382 %plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
383 title('ns = 8, Poles and Zeros 1-2 channel')
384 legend
385 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
386 axis([-1 1 -1 1])
387 axis equal
388 grid on
389 hold off
390
391 %For 2-1 channel the system is
392 B21 = B8(:,1);
393 C21 = C8(2,:);
394 M = [A8 B21; -C21 zeros(1)];
395 N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
396 [V,D] = eig(M,N);
397 %disp(diag(D))%Resulted in D(1) and D(9) as inf
398 z21 = diag(D);
399 z21 = z21([2,3,4,5,6,7,8]);
400 disp('Finite zeros for 2-1 channel:')
401 disp(z21)
402 lic = num2cell(z21);
403 [z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
404 figure()
405 plot(eigA8,'x','DisplayName','Poles');
406 hold on
407 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')
408 %plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
409 %plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
410 %plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
411 %plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
412 %plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
413 %plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
414 %plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
415 title('ns = 8, Poles and Zeros 2-1 channel')
416 legend
417 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
418 axis([-1 1 -1 1])
419 axis equal
420 grid on
421 hold off
422
423 %For 2-2 channel the system is
424 B22 = B8(:,2);
425 C22 = C8(2,:);
426 M = [A8 B22; -C22 zeros(1)];
427 N = [eye(8) zeros(8,1);zeros(1,8) zeros(1)];
428 [V,D] = eig(M,N);
429 %disp(diag(D))%Resulted in D(1) and D(9) as inf
430 z22 = diag(D);
431 z22 = z22([2,3,4,5,6,7,8]);
432 disp('Finite zeros for 2-2 channel:')
433 disp(z22)
434 lic = num2cell(z22);
435 [z1 z2 z3 z4 z5 z6 z7]=deal(lic{:});
436 figure()
437 plot(eigA8,'x','DisplayName','Poles');
438 hold on
439 plot(cos(ut),sin(ut),'k-','DisplayName','Unit Circle')

```

```

440 %plot(real(z1),imag(z1),'bo','DisplayName',['|z1| is ',num2str(abs(z1))]);
441 plot(real(z2),imag(z2),'go','DisplayName',['|z2| is ',num2str(abs(z2))]);
442 plot(real(z3),imag(z3),'ko','DisplayName',['|z3| is ',num2str(abs(z3))]);
443 plot(real(z4),imag(z4),'mo','DisplayName',['|z4| is ',num2str(abs(z4))]);
444 plot(real(z5),imag(z5),'co','DisplayName',['|z5| is ',num2str(abs(z5))]);
445 plot(real(z6),imag(z6),'ro','DisplayName',['|z6| is ',num2str(abs(z6))]);
446 plot(real(z7),imag(z7),'yo','DisplayName',['|z7| is ',num2str(abs(z7))]);
447 title('ns = 8, Poles and Zeros 2-2 channel')
448 legend
449 set(gca, 'XAxisLocation', 'origin', 'YAxisLocation', 'origin')
450 axis([-1 1 -1 1])
451 axis equal
452 grid on
453 hold off

```

Listing 3: Task 3 code

```

1 clear
2 set(groot,'defaulttextinterpreter','latex');
3 set(groot, 'defaultAxesTickLabelInterpreter','latex');
4 set(groot, 'defaultLegendInterpreter','latex');
5
6 % Calculate model and find poles
7 model = DiscreteModel();
8 H100 = model.find_hankel_matrix(100, 0);
9 H100_tilde = model.find_hankel_matrix(100, 1);
10 [A7, ~, ~] = model.compute_model(H100, H100_tilde, 7);
11 eig_A7 = eig(A7);
12
13 % Plot unit circle and poles
14 plot(exp(1j*linspace(0, 2*pi, 100)), 'k');
15 hold on;
16 plot(eig_A7(1:2), 'x', 'MarkerSize', 10);
17 plot(eig_A7(3:4), 'x', 'MarkerSize', 10);
18 plot(eig_A7(5), 0, 'x', 'MarkerSize', 10);
19 plot(eig_A7(6), 0, 'x', 'MarkerSize', 10);
20 plot(eig_A7(7), 0, 'x', 'MarkerSize', 10);
21 hold off;
22 grid on;
23 axis equal;
24 axis([-1.25 1.25 -1.25 1.25]);
25 legend(["Unit Circle" "OSC1" "OSC2" "LP1" "LP2" "LP3"]);

```

Listing 4: Task 4 code

```

1 close all
2 clear
3 set(groot,'defaulttextinterpreter','latex');
4 set(groot, 'defaultAxesTickLabelInterpreter','latex');
5 set(groot, 'defaultLegendInterpreter','latex');
6
7 % Load the random data
8 load u_rand.mat
9 y1 = u_rand.Y(3).Data;
10 y2 = u_rand.Y(4).Data;
11 u1 = u_rand.Y(1).Data;
12 u2 = u_rand.Y(2).Data;
13 u = [u1; u2];
14 y = [y1; y2];
15
16 ts = 1/40;
17 model = DiscreteModel();
18
19 % Task 4.1: Verify means are approx. 0
20 u1_mean = mean(u1)
21 u2_mean = mean(u2)
22 y1_mean = mean(y1)
23 y2_mean = mean(y2)
24
25 % Task 4.2: Graph estimates of Ruu for k in [-200, 200]
26 k = -200:200;
27 Ruu = zeros(4, length(k));
28
29 for i=1:length(k)
30     % 5000 is trade-off between accuracy and speed
31     Ruu(:, i) = reshape(find_correlation(u, u, k(i), length(u)*2), [4 1]);
32 end
33
34 figure(1);
35 subplot(2, 2, 1);
36 plot(k*ts, Ruu(1, :));
37 axis([-5 5 -0.5 4.5]);
38 xlabel("Lag factor, $\tau$ (sec)")

```

```

39 ylabel("$R_{uu}[k]$");
40 title("(1, 1) channel")
41 grid on
42
43 subplot(2, 2, 2);
44 plot(k*ts, Ruu(3, :));
45 axis([-5 5 -0.5 4.5]);
46 xlabel("Lag factor, $\tau$ (sec)")
47 ylabel("$R_{uu}[k]$");
48 title("(1, 2) channel")
49 grid on
50
51 subplot(2, 2, 3);
52 plot(k*ts, Ruu(2, :));
53 axis([-5 5 -0.5 4.5]);
54 xlabel("Lag factor, $\tau$ (sec)")
55 ylabel("$R_{uu}[k]$");
56 title("(2, 1) channel")
57 grid on
58
59 subplot(2, 2, 4);
60 plot(k*ts, Ruu(4, :));
61 axis([-5 5 -0.5 4.5]);
62 xlabel("Lag factor, $\tau$ (sec)")
63 ylabel("$R_{uu}[k]$");
64 title("(2, 2) channel")
65 grid on
66
67 sgtitle(["Autocorrelation of $u$, $R_{uu}$" "for $\tau$ in [-5, 5]"], ...
68         'interpreter', 'latex');
69
70 % Task 4.3: Find Ruu(0)
71 Ruu0 = find_correlation(u, u, 0, length(u)*2)
72
73 % Task 1.4: Estimate Ryu for tau in [-0.2, 2]
74 k = -0.2/ts:2/ts;
75 Ryu = zeros(4, length(k));
76
77 for i=1:length(k)
78     Ryu(:, i) = reshape(find_correlation(y, u, k(i), length(u)*2), [4 1]);
79 end
80
81 % Normalize columns
82 Ryu11 = Ryu(1, :)./Ruu0(1, 1);
83 Ryu12 = Ryu(3, :)./Ruu0(1, 1);
84 Ryu21 = Ryu(2, :)./Ruu0(2, 2);
85 Ryu22 = Ryu(4, :)./Ruu0(2, 2);
86
87 figure(2);
88 subplot(2, 1, 1);
89 plot(k*ts, [Ryu11; Ryu21]);
90 hold on
91 plot(model.t, model.y11, 'r--');
92 plot(model.t, model.y21, 'r--');
93 xlim([-0.2 2]);
94 xlabel("Lag factor $\tau$ (sec)");
95 ylabel("response (V)");
96 legend(["(1, 1) channel autocorrelation" ...
97         "(2, 1) channel autocorrelation" "(1, 1) channel impulse response" ...
98         "(2, 1) channel impulse response"], ...
99         'location', 'southeast');
100 grid on
101
102 subplot(2, 1, 2);
103 plot(k*ts, [Ryu12; Ryu22]);
104 hold on
105 plot(model.t, model.y12, 'r--');
106 plot(model.t, model.y22, 'r--');
107 xlim([-0.2 2]);
108 xlabel("Lag factor $\tau$ (sec)");
109 ylabel("response (V)");
110 legend(["(1, 2) channel autocorrelation" ...
111         "(2, 2) channel autocorrelation" "(1, 2) channel impulse response" ...
112         "(2, 2) channel impulse response"], ...
113         'location', 'southeast');
114 grid on
115
116 sgtitle(["Autocorrelation of $u$, $R_{yu}$ vs data impulse response" ...
117         "for $\tau$ in [-0.2, 2]"]);
118
119 sqrt(trace(find_correlation(y./2, y./2, 0, length(u)*2)))
120
121 function Rab = find_correlation(a, b, k, p_est)
122     Rab = zeros(size(a, 1), size(b, 1));
123     n = size(a, 2);
124     elems = 0;

```

```

125
126     for q=-p_est:p_est
127         % Can't allow indices outside of [1, length(a)]
128         if (q >= 1 && k+q >= 1 && q <= n && k+q <= n)
129             Rab = Rab + a(:, k+q)*b(:, q)';
130             elems = elems + 1;
131         end
132     end
133     Rab = Rab/elems;
134 end

```

Listing 5: Task 5 code

```

1  close all
2  clear
3
4  % Load the random data and normalize it
5  load u_rand.mat
6  y1 = u_rand.Y(3).Data/2;
7  y2 = u_rand.Y(4).Data/2;
8  y = [y1-mean(y1); y2-mean(y2)];
9
10 % Task 5.1: Find ||y||_RMS^2 of scaled output
11 Y_RMS = sqrt(sum(vecnorm(y).^2)/size(y, 2))
12
13 % Task 5.2: Compute ||P||_H2^2 for ns=7 system
14 % ns=7 model from task 1
15 model = DiscreteModel();
16 H100 = model.find_hankel_matrix(100, 0);
17 H100_tilde = model.find_hankel_matrix(100, 1);
18 [A7, B7, C7] = model.compute_model(H100, H100_tilde, 7);
19
20 H2_norm_1 = sqrt(trace(B7'*dlyap(A7', C7'*C7)*B7))
21 H2_norm_2 = sqrt(trace(C7*dlyap(A7, B7*B7')*C7'))
22
23 % Task 5.3: Compute ||P||_H2^2 from impulse response data
24 H2_norm_data = sqrt(sum(model.y11.^2) + sum(model.y12.^2) + ...
25     sum(model.y21.^2) + sum(model.y22.^2))

```

Listing 6: Task 6 code

```

1  clear
2  set(groot, 'defaulttextinterpreter', 'latex');
3  set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
4  set(groot, 'defaultLegendInterpreter', 'latex');
5
6  % Compute frequency responses
7  model = DiscreteModel();
8  H100 = model.find_hankel_matrix(100, 0);
9  H100_tilde = model.find_hankel_matrix(100, 1);
10 [A7, B7, C7] = model.compute_model(H100, H100_tilde, 7);
11 omega = 0:0.1:(1/(2*model.ts));
12 F7 = model.compute_freq_resp(A7, B7, C7, omega);
13
14 y11f = fft(model.y11)./fft(model.u1);
15 y12f = fft(model.y12)./fft(model.u2);
16 y21f = fft(model.y21)./fft(model.u1);
17 y22f = fft(model.y22)./fft(model.u2);
18 om_data = (0:length(y11f)/2-1)/(model.ts*length(y11f));
19
20 % Find Hinf norm
21 [H_val, f_val] = Hinf_dis(A7, B7, C7, zeros(2), ...
22     sum(svd(H100)), 0, 1e-6, model.ts);
23 H_val
24 f_val
25
26 % Now compute and graph singular values
27 SV_model = zeros(2, size(F7, 2));
28 for k=1:size(F7, 2)
29     SV_model(:, k) = svd([F7(1, k) F7(3, k); F7(2, k) F7(4, k)]);
30 end
31
32 SV_data = zeros(2, length(om_data));
33 for k=1:length(om_data)
34     SV_data(:, k) = svd([y11f(k) y12f(k); y21f(k) y22f(k)]);
35 end
36
37 plot(om_data, SV_data, 'o', omega, SV_model, '-');
38 hold on
39 plot(f_val/(2*pi), H_val, 'k+', 'MarkerSize', 10);
40 grid on
41 xlabel("Frequency, $\omega$ (Hz)");
42 ylabel("Singular values of frequency response, $\sigma(H(j\omega))$")

```

```

43 legend(["empirical $\sigma_1$" "empirical $\sigma_2$" ...
44         "model $\sigma_1$" "model $\sigma_2$" "$H_{\infty}$ norm"], ...
45         'location', 'NorthEast');
46 hold off

```

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

```

1  classdef DiscreteModel
2      properties
3          y11
4          y12
5          y21
6          y22
7          u1
8          u2
9          N
10         ts
11         t
12         t_imp
13     end
14
15     methods
16         function model=DiscreteModel()
17             % Load and clean data
18             load u1_impulse.mat;
19             load u2_impulse.mat;
20
21             y11 = u1_impulse.Y(3).Data;
22             y21 = u1_impulse.Y(4).Data;
23             y12 = u2_impulse.Y(3).Data;
24             y22 = u2_impulse.Y(4).Data;
25             u1 = u1_impulse.Y(1).Data;
26             u2 = u2_impulse.Y(2).Data;
27
28             % Remove DC offset in data
29             [~, mi1] = max(u1 > 0);
30             [~, mi2] = max(u2 > 0);
31             y11 = y11 - mean(y11(1:mi1 - 1));
32             y12 = y12 - mean(y12(1:mi1 - 1));
33             y21 = y21 - mean(y21(1:mi1 - 1));
34             y22 = y22 - mean(y22(1:mi2 - 1));
35             u1 = u1 - mean(u1(1:mi1 - 1));
36             u2 = u2 - mean(u2(1:mi2 - 1));
37             mu1 = max(u1);
38             mu2 = max(u2);
39
40             % rescale IO data so that impulse input has magnitude 1
41             model.y11 = y11/mu1;
42             model.y12 = y12/mu2;
43             model.y21 = y21/mu1;
44             model.y22 = y22/mu2;
45             model.u1 = u1/mu1;
46             model.u2 = u2/mu2;
47
48             model.N = length(u1);
49             model.ts = 1/40;
50             model.t_imp = mi1;
51             model.t = ((1:model.N) - model.t_imp)*model.ts;
52
53         end
54
55         function Hn=find_hankel_matrix(model, n, offset)
56             Hn = zeros(2*n);
57             for r=1:n
58                 for c=1:n
59                     k = r+c-1+offset+model.t_imp;
60                     Hn(2*r-1:2*r, 2*c-1:2*c) = ...
61                         [model.y11(k) model.y12(k); ...
62                         model.y21(k) model.y22(k)];
63                 end
64             end
65         end
66
67         function [A, B, C]=compute_model(~, H, H_tilde, n)
68             % Truncate H to use the first n singular values
69             [U, S, V] = svd(H);
70             Un = U(:, 1:n);
71             Sn = S(1:n, 1:n);
72             Vn = V(:, 1:n);
73
74             % Find the observability and controllability matrices
75             % and their inverses.
76             Sn12 = diag(sqrt(diag(Sn)));
77             Snn12 = diag(1./sqrt(diag(Sn)));
78             On = Un*Sn12;

```

```

79         On_inv = Snn12*Un';
80         Cn = Sn12*Vn';
81         Cn_inv = Vn*Snn12;
82
83         % Calculate system matrices
84         A = On_inv*H_tilde*Cn_inv;
85         B = Cn(:, 1:2);
86         C = On(1:2, :);
87     end
88
89     % Find the impulse response of the given system and channel
90     function h=compute_imp_resp(~, A, B, C, n)
91         h = zeros(4, n);
92         Ap = eye(size(A));
93         for k=1:n
94             hk = C*Ap*B;
95             h(:, k) = reshape(hk, [4 1]);
96             Ap = Ap*A;
97         end
98     end
99
100    % Find the frequency reponse of the given system and channel
101    function F=compute_freq_resp(model, A, B, C, omega)
102        TS = model.ts;
103        F = zeros(4, length(omega));
104        for i=1:length(omega)
105            F_temp = C*inv(exp(2i*pi*omega(i)*TS).*eye(size(A))-A)*B;
106            F(:, i) = reshape(F_temp, [4 1]);
107        end
108    end
109 end
110 end

```

Listing 8: Hinf_cont function (used for task 6)

```

1  function [gam, freq] = Hinf_cont(A, B, C, D, up, lo, tol)
2
3  freq = -1;
4
5  while (up - lo)/lo > tol/2
6
7      gam = (up + lo)/2;
8      Dg = gam^2 * eye(size(D' * D, 1)) - D' * D;
9      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'];
10
11      evals = eig(Aclp);
12      t = 0;
13      for eind = 1:length(evals)
14          if abs(real(evals(eind))) < 1e-8
15              freq = abs(imag(evals(eind)));
16              t = 1;
17          end
18      end
19      if t == 1
20          lo = gam;
21      else
22          up = gam;
23      end
24
25  end
26
27  end

```

Listing 9: Hinf_dis function (used for task 6)

```

1  function [Hinf, freq] = Hinf_dis(A, B, C, D, up, lo, tol, tsam)
2  %INPUTS: Function to compute H_inf for continuous time systems,
3  %         discrete time matrices - A, B, C, D
4  %         upper limit, lower limit, and tolerance for gamma
5  %RETURNS the H_inf norm and the discrete time frequency at which it occurs.
6  Ac = -inv(eye(size(A)) + A) * (eye(size(A)) - A);
7  Bc = sqrt(2) * inv(eye(size(A)) + A) * B;
8  Cc = sqrt(2) * C * inv(eye(size(A)) + A);
9  Dc = D - C * inv(eye(size(A)) + A) * B;
10
11  [Hinf, freq] = Hinf_cont(Ac, Bc, Cc, Dc, up, lo, tol);
12  freq = 1/tsam * angle((1 + 1i * freq)/(1 - 1i * freq));
13
14  end

```