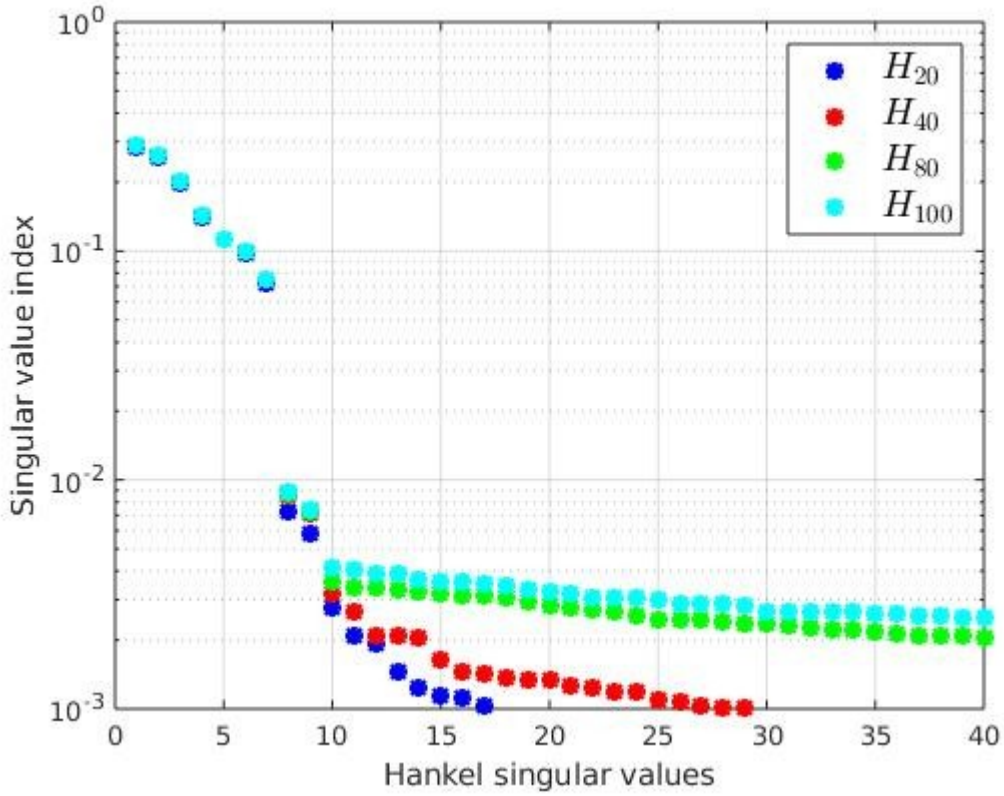
MAE 270A Lopez, Alonzo
Fall 2019 UID: 905352196

270A Final Project

Task #1: Model Identification

From the impulse response data, we construct the hankel matrices $H_{20}, H_{40}, H_{60}, H_{100}, \tilde{H}_{100}$ (constructhankels.m). Their singular values are calculated and plotted below using the svd command in matlab.



Then the SVD is computed for H_{100} , resulting in the U, S, and V matrices (computemodels.m). From the result of the SVD, we obtain A by the following process:

1. Define n_s as the selected model order, in our case $n_s = 6, 7, 10, 20$
2. For a value of n_s , define $O = U_1, C = \Sigma_{n_s} V_1$, where U_1 consists of the first n_s columns of U , and V_1 consists of the first n_s columns of V
3. Calculate $A = O_n^{\dagger} \tilde{H}_{100} C_n^{\dagger}$ (computemodels.m)
4. Repeat steps 2 and 3 for each value of n_s (computemodels.m)
5. Confirm $\lambda_{max} < 1$ to ensure that the model is asymptotically stable for all of the produced models
    1. For $n_s = 6$, $\lambda_{max} = 0.9526$
    2. For $n_s = 7$, $\lambda_{max} = 0.9144$
    3. For $n_s = 10$, $\lambda_{max} = 0.9141$

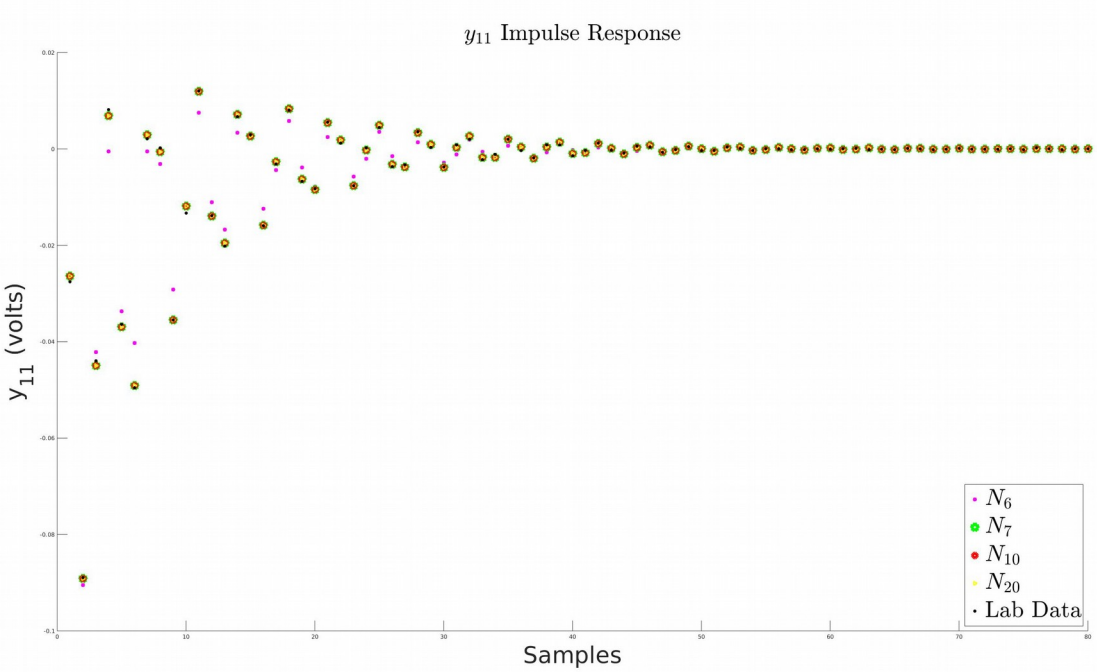
4. For $n_s = 20$, $\lambda_{max} = 0.9975$

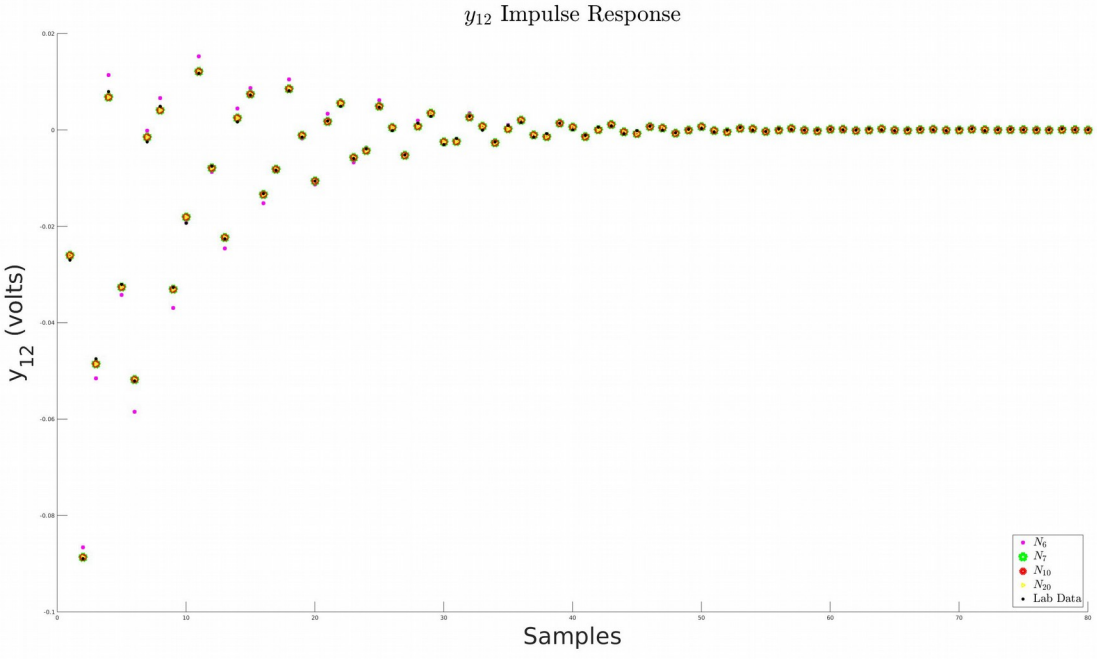
From these models we can simulate the impulse response and use that to compare with the empirical impulse response data. First, obtain *B* and *C* from the O_n (observability) and C_n (controllability) matrices. The O_n and C_n matrices are obtained for each value of n_s in computemodels.m. The below relationships allow us to obtain B and C from O_n and C_n . Note, $B \in R^{n_s x q}, C \in R^{m x n_s}$, where $m = q = 2$.

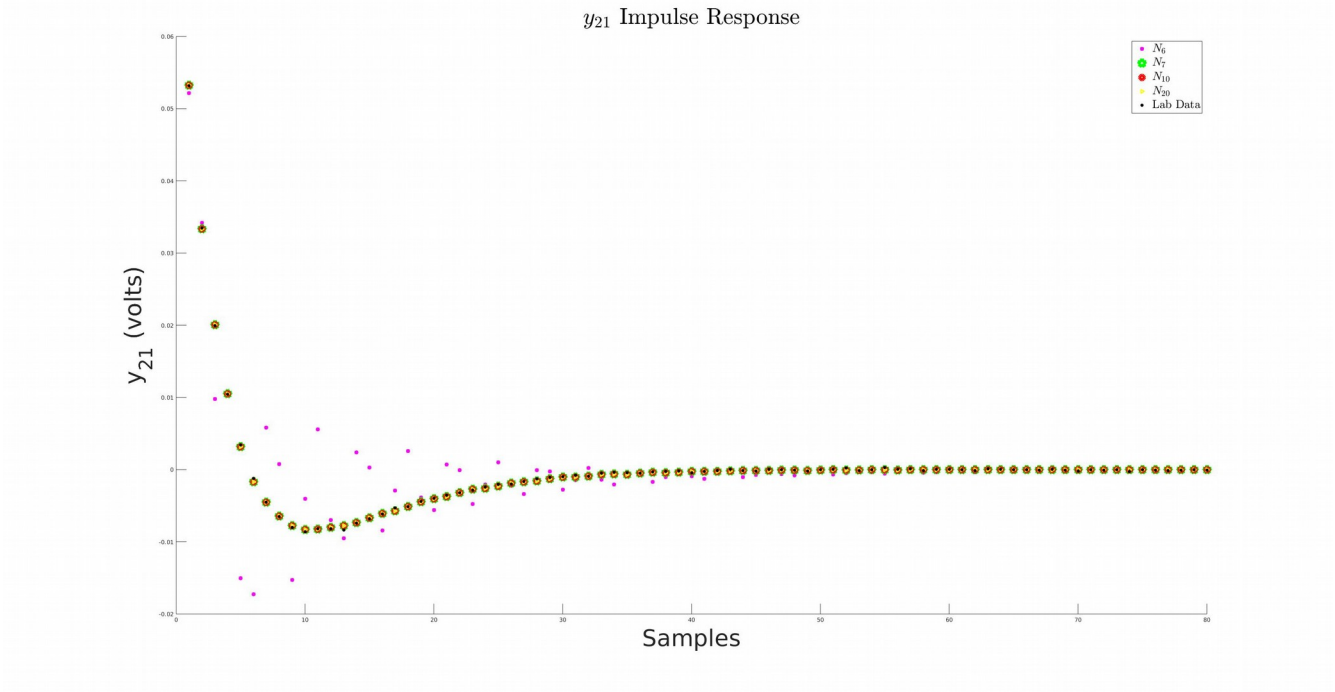
$$\mathcal{O}_n = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \in \mathbf{R}^{mn \times n_s}, \quad \mathcal{C}_n = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \in \mathbf{R}^{n_s \times nq}.$$

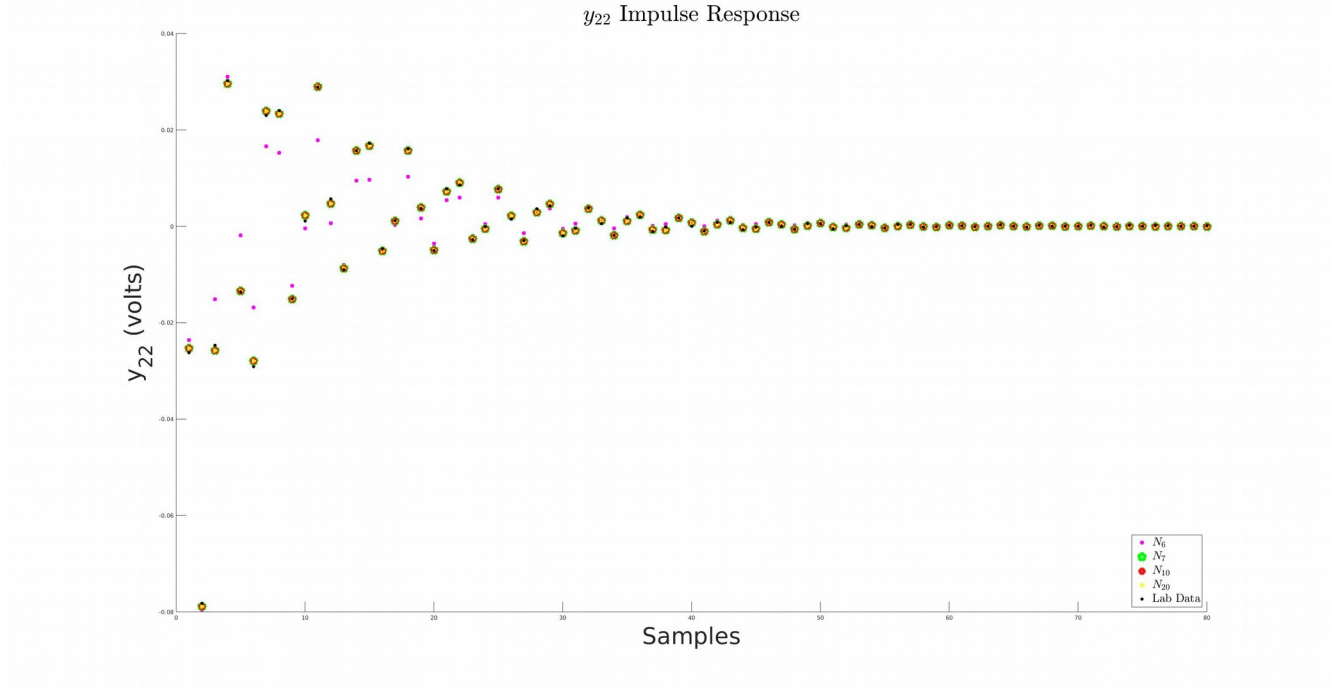
Now, in script sim_impulse_response.m, plot the experimentally obtained impulse response, then calculate C and B for each model order and plot the model's impulse response using

$$h_k = CA^{(k-1)}B, k = 1, 2, 3\ldots$$


y_{11} Impulse Response
y_{11} (volts)
Samples
N_6
N_7
N_{10}
N_{20}
Lab Data


y_{12} Impulse Response
y_{12} (volts)
Samples
N_6
N_7
N_{10}
N_{20}
Lab Data

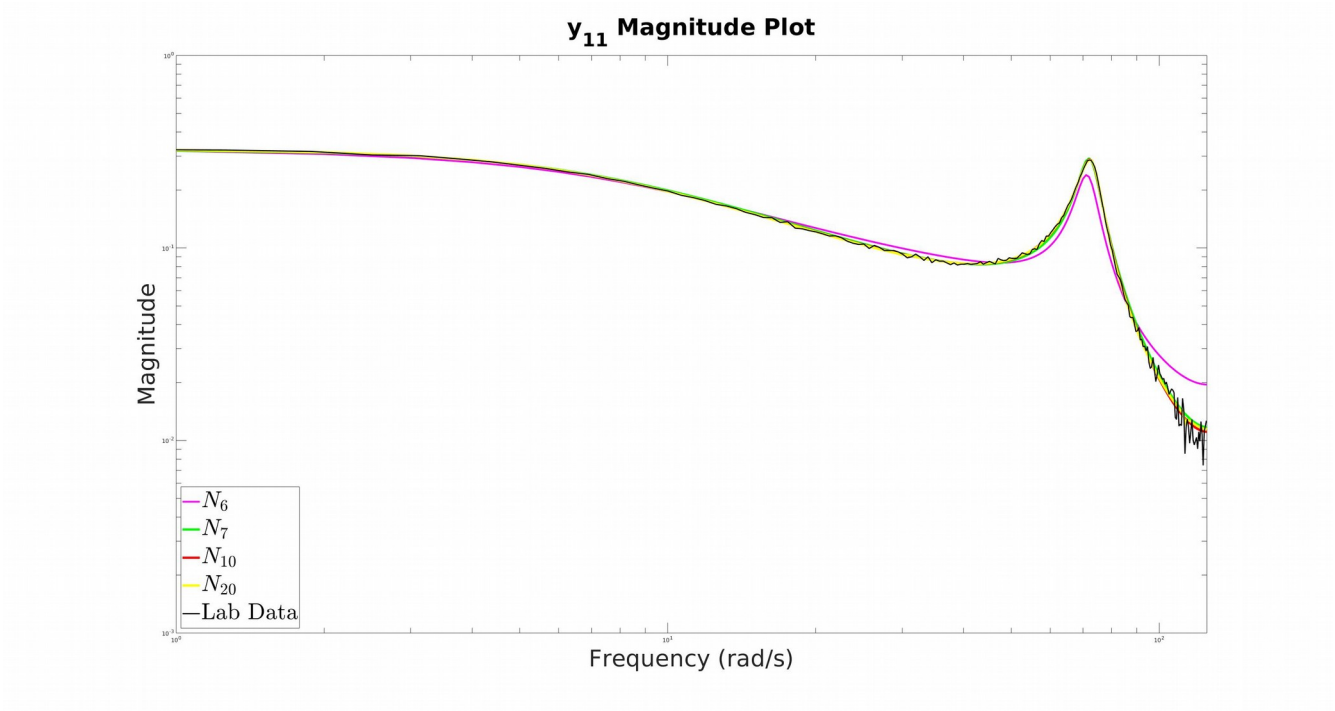

y_{21} Impulse Response
N_6
N_7
N_{10}
N_{20}
Lab Data
y_{21} (volts)
Samples

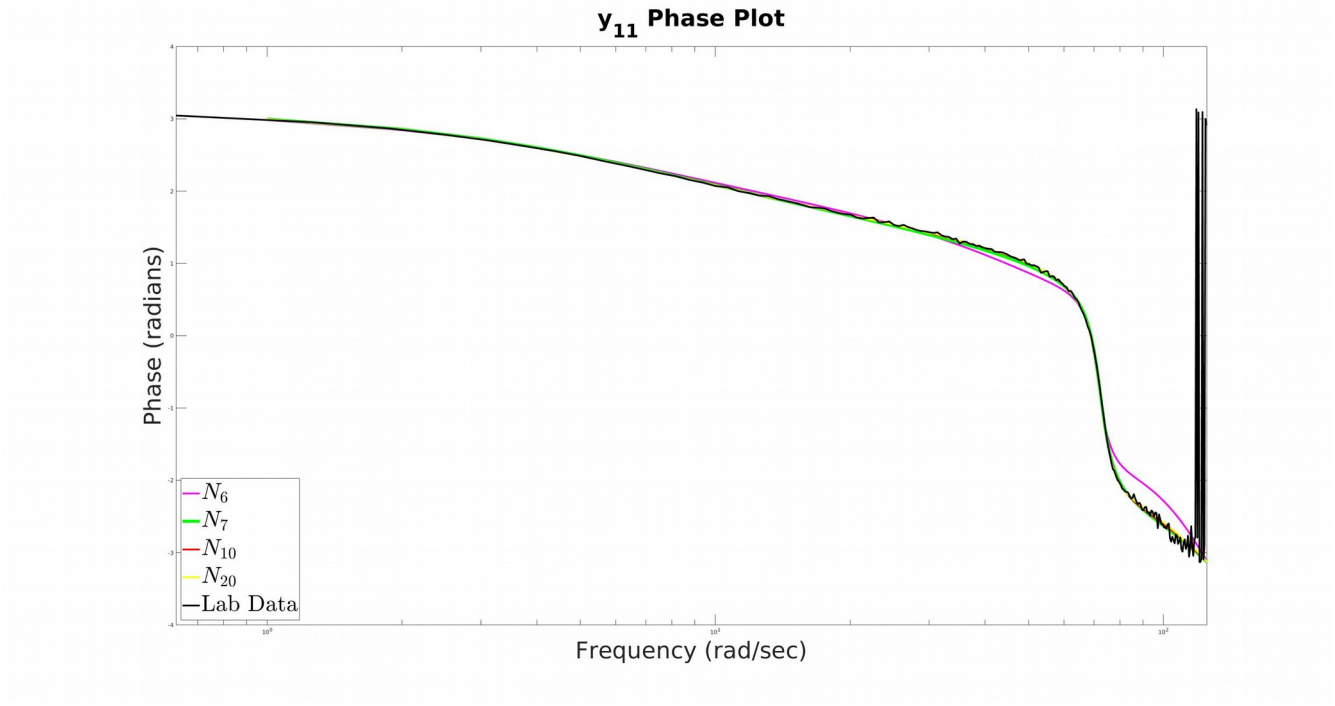

y_{22} Impulse Response
N_6
N_7
N_{10}
N_{20}
Lab Data
y_{22} (volts)
Samples

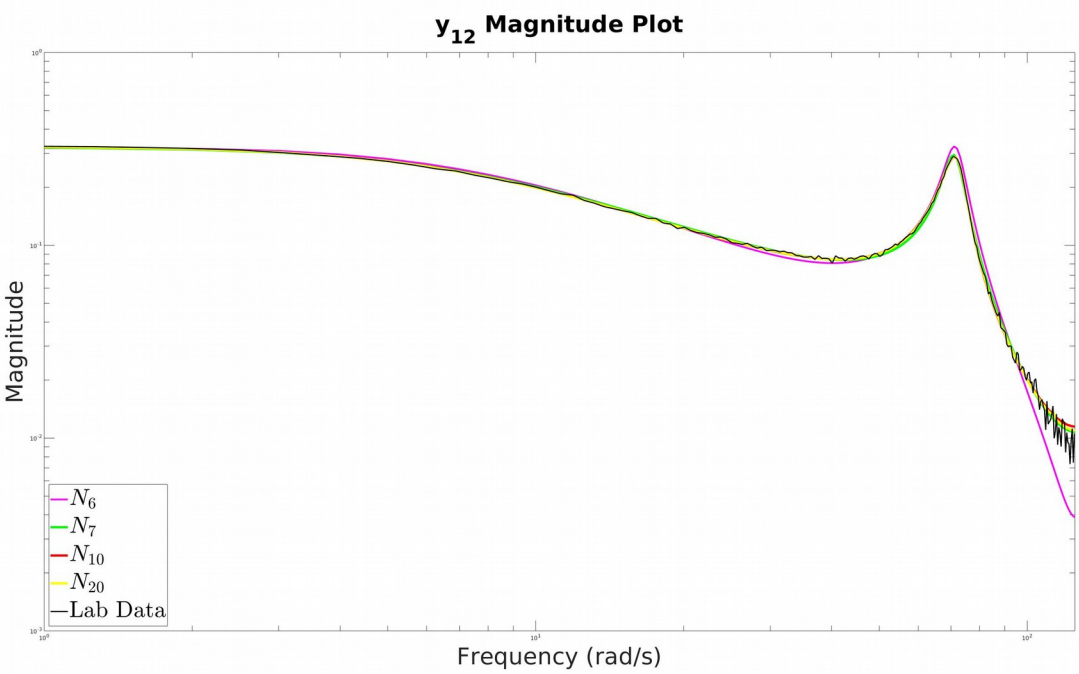
In addition to comparing the impulse response of the models and the empirical data, we compare the frequency response of the models with the frequency response obtained from the pulse response data. We calculate the frequency response of the models by

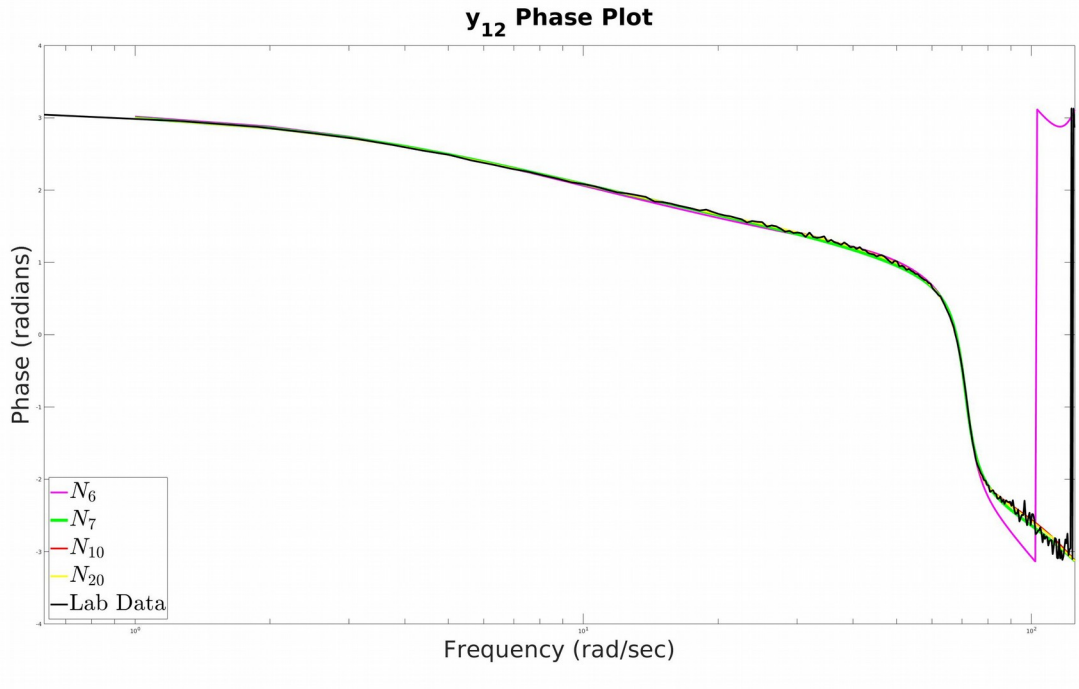
$$C\left(e^{jwt_s}I-A\right)^{-1}B+D$$

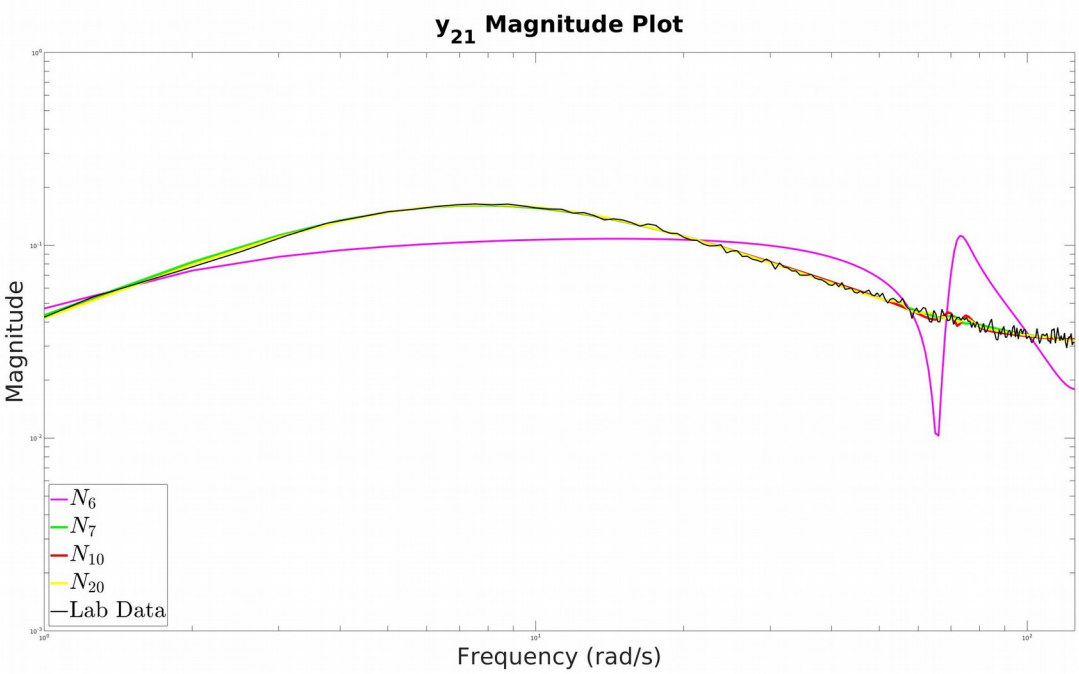
for a range of frequency values from 0 to 20 Hz, where 20 Hz is the Nyquist frequency w_{nyq}. In the formula above, the frequency w is evaluated in units of rad/s. The empirical data's frequency response is obtained through the fft code in freqresponse.m. The comparison is viewed through the following eight plots of frequency response magnitude and phase for $y_{11,}\, y_{12,}\, y_{21,} y_{22,}$ all plotted in freqresponse.m.

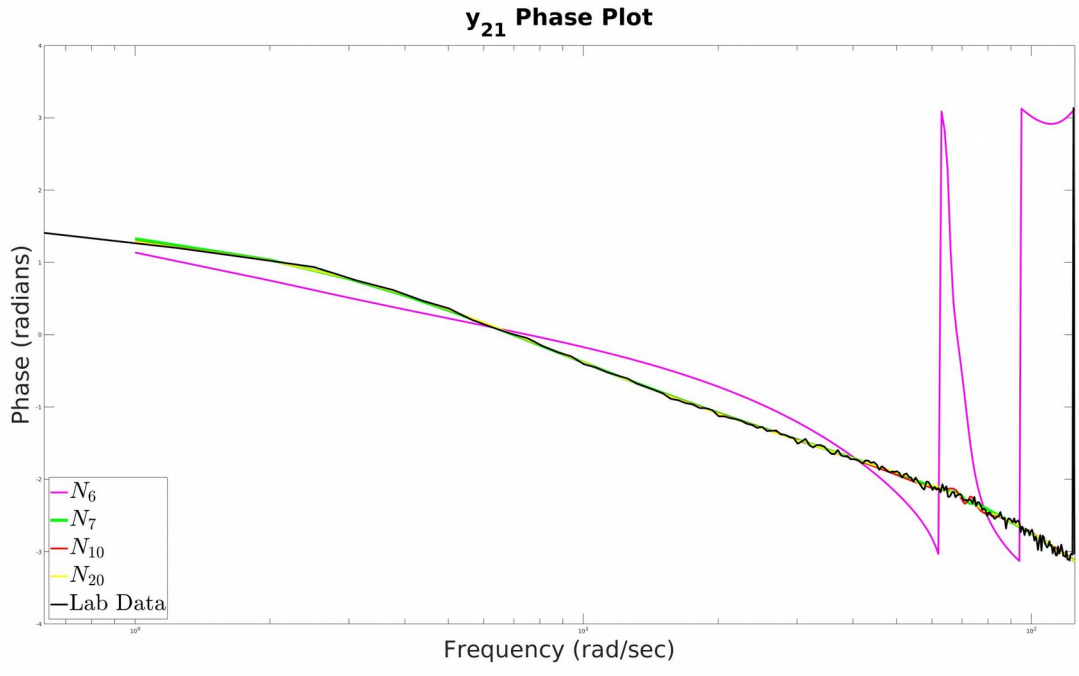

y_{11} Magnitude Plot
Magnitude
Frequency (rad/s)
N_6
N_7
N_{10}
N_{20}
Lab Data

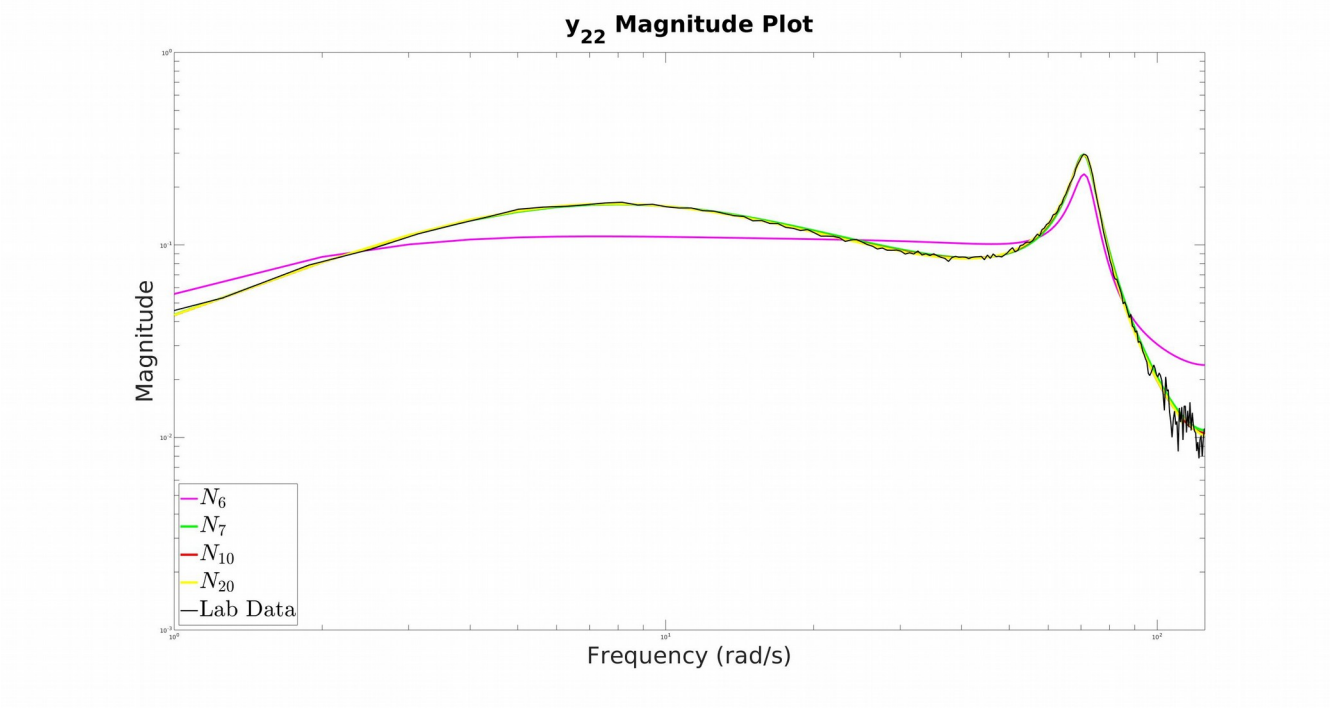

y_{11} Phase Plot
Phase (radians)
Frequency (rad/sec)
N_6
N_7
N_{10}
N_{20}
Lab Data

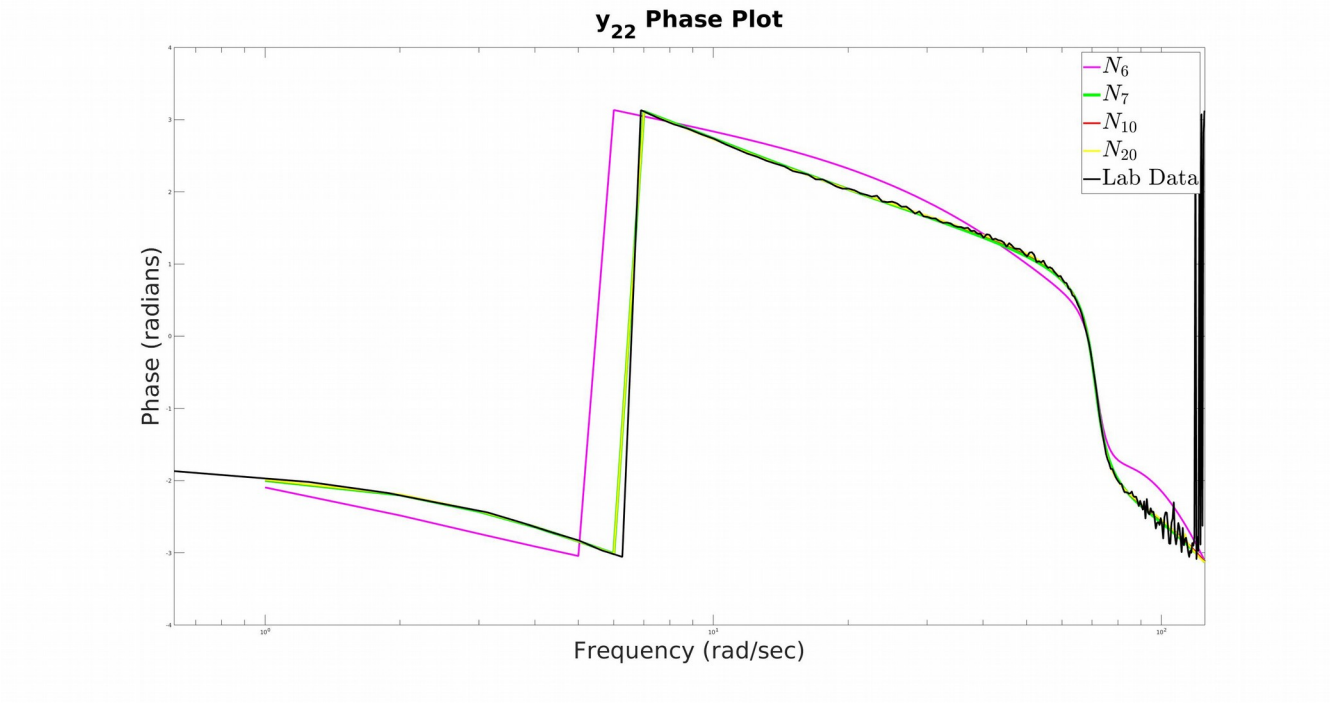

y_{12} Magnitude Plot
Magnitude
Frequency (rad/s)
N_6
N_7
N_{10}
N_{20}
Lab Data


y_{12} Phase Plot
Phase (radians)
Frequency (rad/sec)
N_6
N_7
N_{10}
N_{20}
Lab Data


y_{21} Magnitude Plot
Magnitude
Frequency (rad/s)
N_6
N_7
N_{10}
N_{20}
Lab Data


y_{21} Phase Plot
Phase (radians)
Frequency (rad/sec)
N_6
N_7
N_{10}
N_{20}
Lab Data





Up to this point, we have constructed four different models of different selected orders $n_s = 6, 7, 10, 20$ through the SVD and Hankel matrix reconstruction and compared both their impulse responses and frequency responses to the empirical data to conclude that $n_s = 7$ is a minimal realization for our model. The model $n_s = 6$ does not reproduce the impulse or frequency response

quite well, and $n_s = 10,20$ are both higher order approximations that do not recreate the model significantly better than $n_s = 7$.

Task #2: Transmission zeros of the MIMO model and zeros of each channel

For the model $n_s = 7$, we define

$$S(\alpha) = \begin{bmatrix} \alpha I - A & -B \\ C & D \end{bmatrix} \in C^{9x9}$$

Normally $rank\, S(\alpha) = 9$, but for some transmission zeros z, $rank\, S(z) < 9$. At this z, there exist a $c \in C^7$ and $w \in C^2$ vector such that

$$S(z)\begin{bmatrix} c \\ w \end{bmatrix} = \begin{bmatrix} zI - A & -B \\ C & D \end{bmatrix}\begin{bmatrix} c \\ w \end{bmatrix} = 0$$

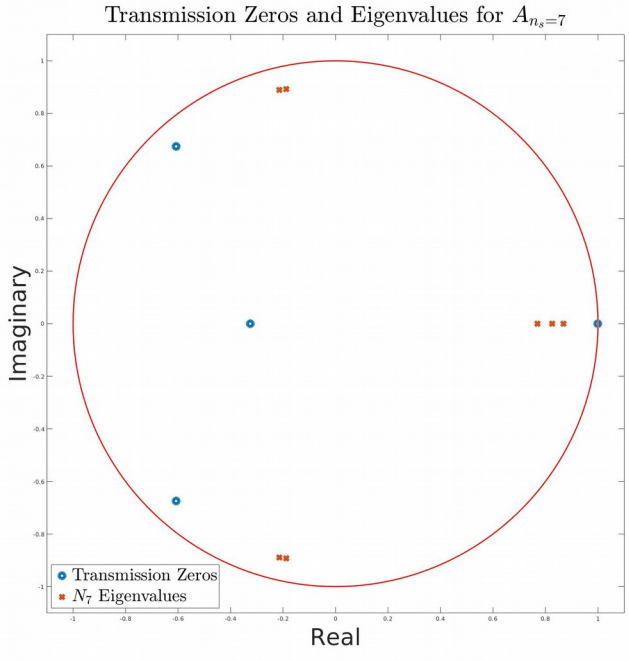
Solving the generalized eigenvalue problem of the form

$$Mv = \lambda K v$$

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

gives the finite transmission zeros listed below:

$$z_1 = -2.6310 + 0.0000\,i$$
$$z_2 = 0.9988 + 0.0000\,i$$
$$z_3 = -0.6078 + 0.6740\,i$$
$$z_4 = -0.6078 - 0.6740\,i$$
$$z_5 = -0.3241 + 0.0000\,i$$



The eigenvalues and transmission zeros are graphed in transmissionzeros.m. Note that the model $n_s = 7$ is asymptotically stable, and all the eigenvalues lie within the unit circle.

Converting the discrete time eigenvalues of A_7 into the continuous time eigenvalues by the relationship:

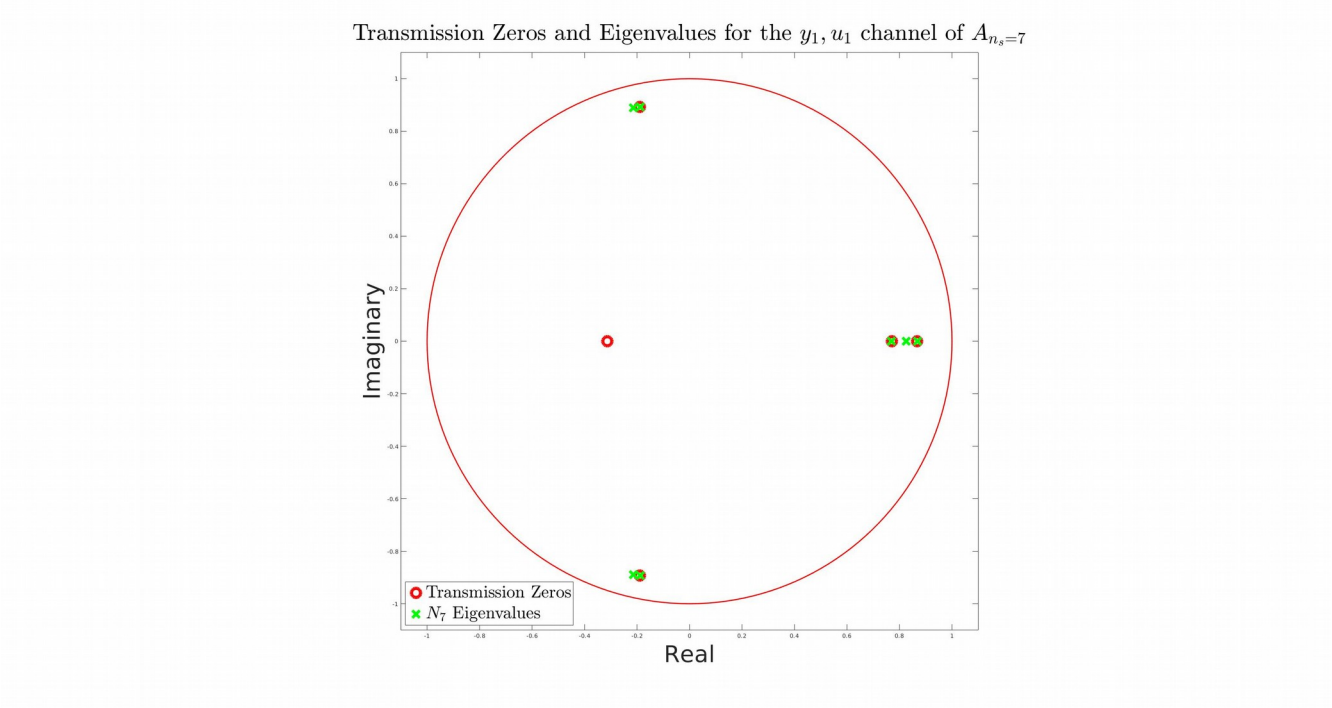
$$\lambda_d = e^{\lambda_c t_s}$$
$$\lambda_c = \frac{\ln(\lambda_d)}{t_s}$$

We obtain the below continuous time eigenvalues representing the poles of the transfer function of A_7.

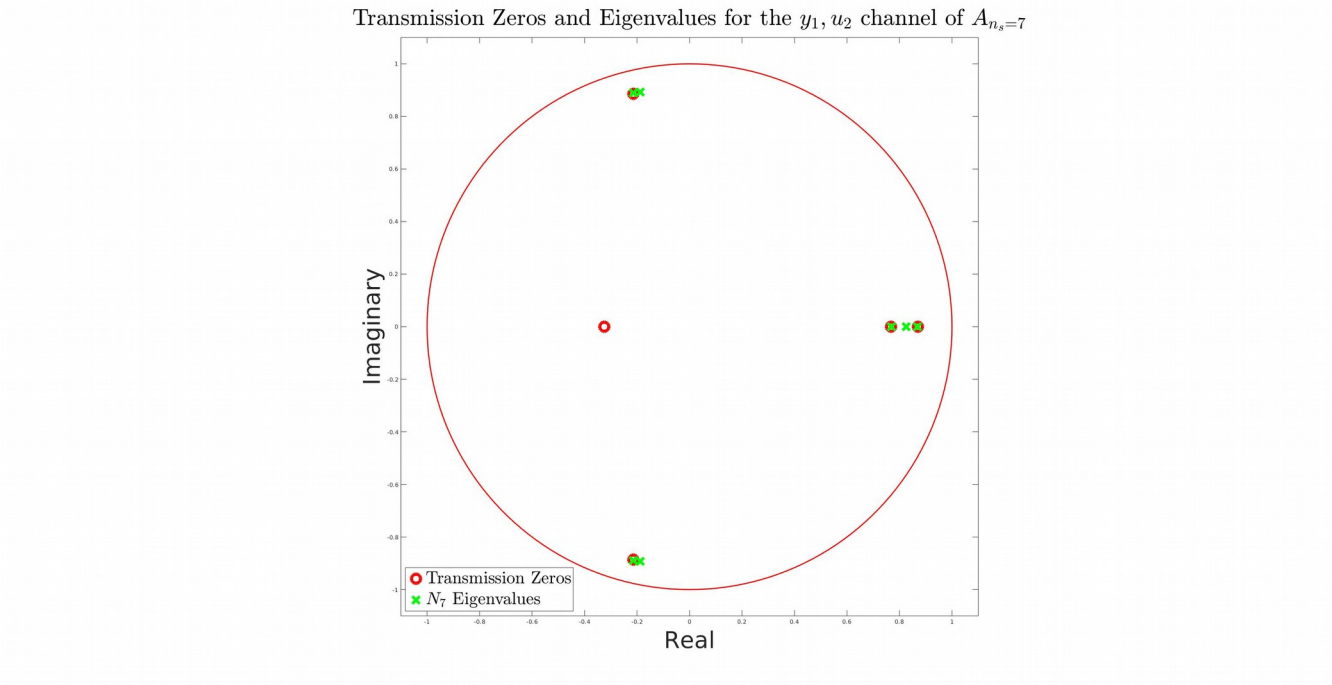
$$-3.6842 \pm 71.1394\,i$$
$$-3.5800 \pm 72.2737\,i$$
$$-10.4604 + 0.0000\,i$$
$$-7.6568 + 0.0000\,i$$
$$-5.6364 + 0.0000\,i$$

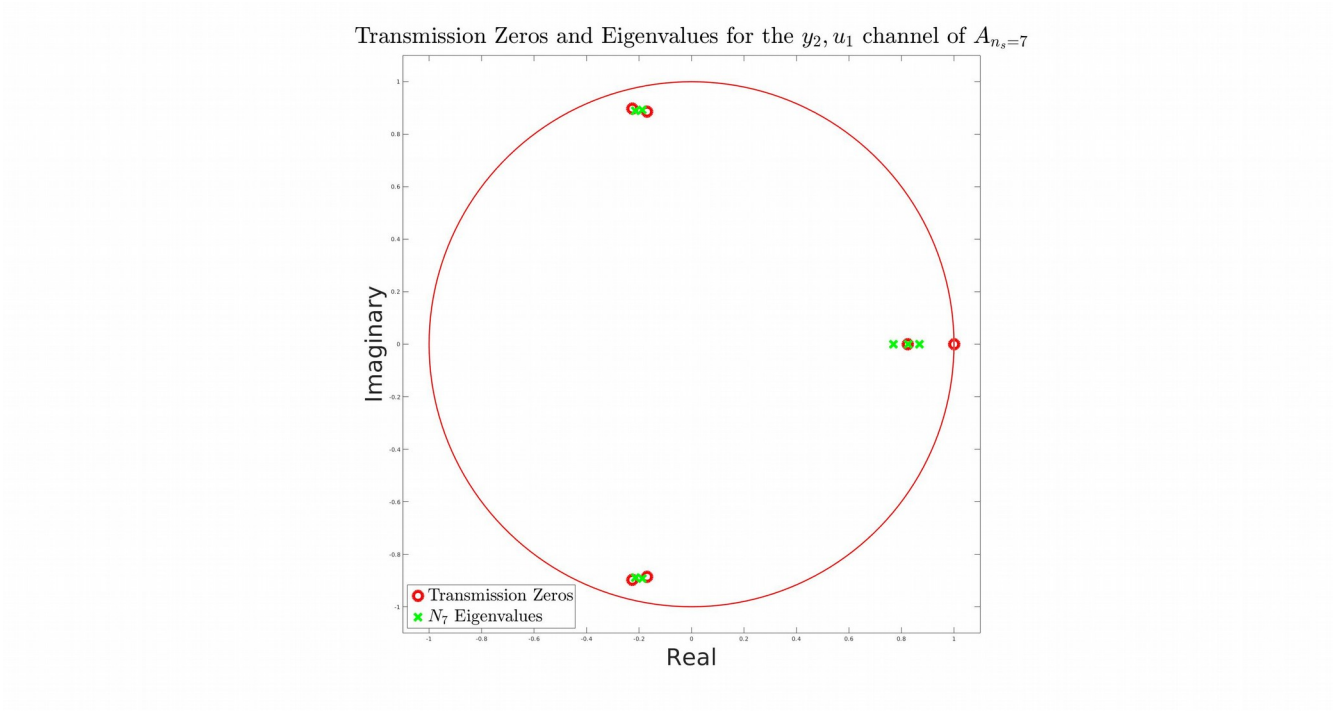
These eigenvalues are the roots of the characteristic polynomial of A_7, hence they are the poles of the transfer function describing this system. These eigenvalues show two damped oscillators with natural frequencies of approximately 71.1394 rad/s and 72.2737 rad/s. These natural frequencies agree with some of the magnitude plots for the A_7 model, which show under-damped complex conjugate poles around these frequencies across multiple channels. However, we must compare the transmission zeros and eigenvalue plots with the frequency response plots for each channel in order to confirm consistency between the two, specifically that near pole-zero cancellations or lack thereof and the frequency responses agree for each channel.

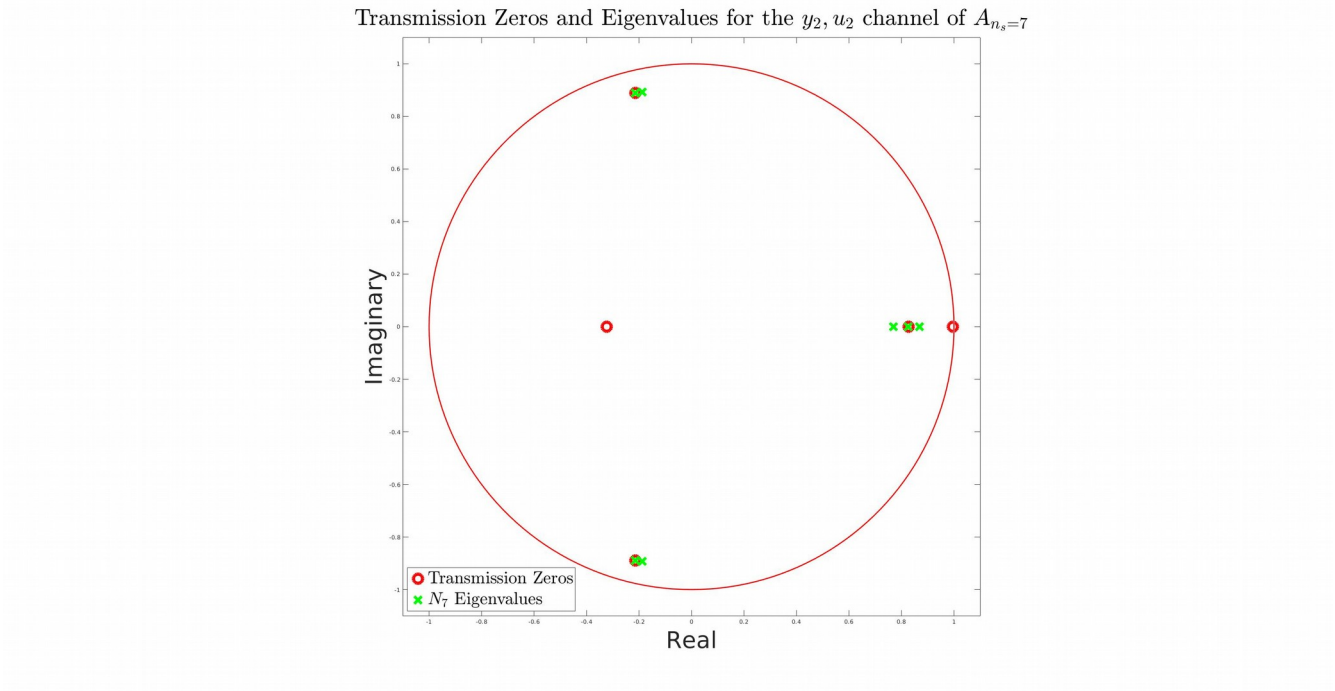
Looking at the channels individually by picking the appropriate column of B and row of C, and solving the generalized eigenvalue problem with these individual columns and rows of B and C, we can find the transmission zeros that apply to that channel. The eigenvalues are the same across all channels because those come from the A_7 model. By plotting the eigenvalues and poles together for each channel, we can observe any pole-zero cancellations specific to that channel and come up with a lower-order approximation for that channel's model. The above is executed in transmissionzeros.m for all channels $y_{11,}\, y_{12,}\, y_{21,} y_{22}$.



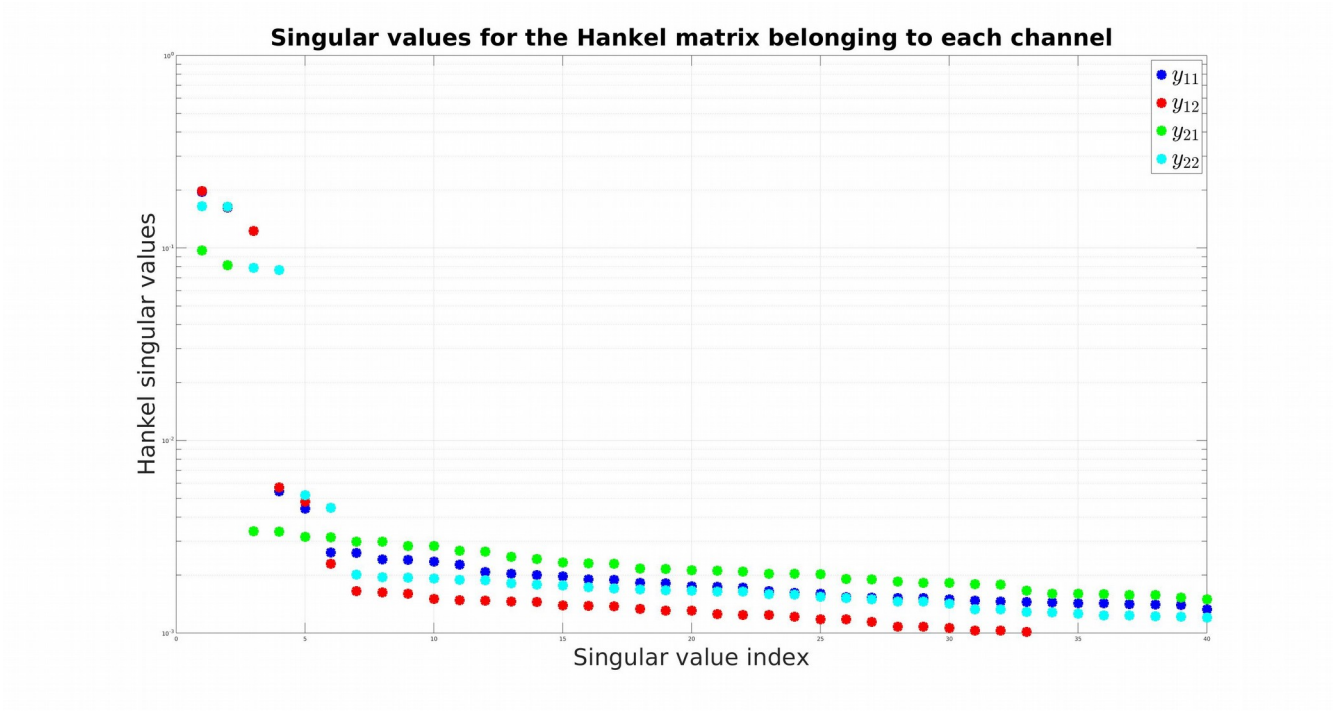
For the y_{11} channel, all of the poles are cancelled through pole-zero cancellation with the transmission zeros of this channel except for the complex conjugate pole pair at $w = 72.2737\, rad/s$ and the pole in the RHP. This is consistent with the frequency response plot for y_{11}, where we see a resonant peak at $w = 72.2737\, rad/s$ corresponding to the complex conjugate pole pair. For the $y_{12,}\, y_{21,}\, y_{22}$ channels we observe a similar agreement between the pole-zero plots and the frequency response where the frequency response curve matches with the poles and zeros that do not get cancelled through pole-zero cancellation





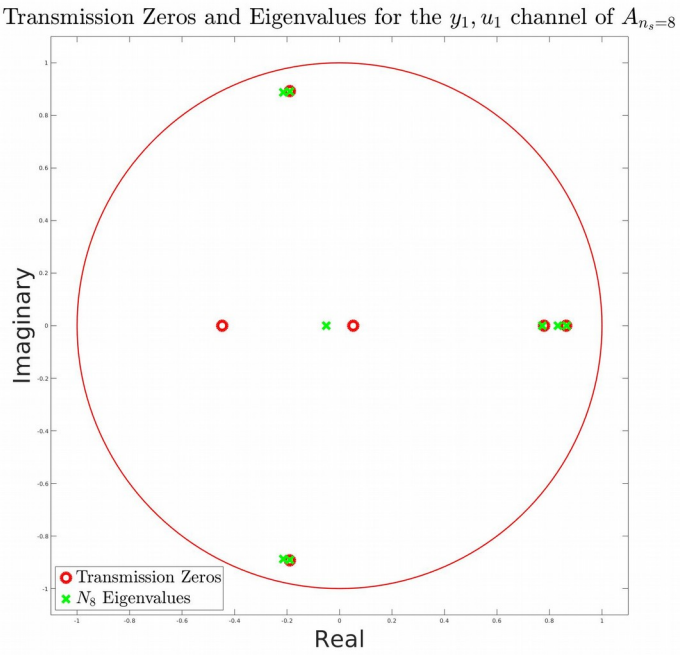


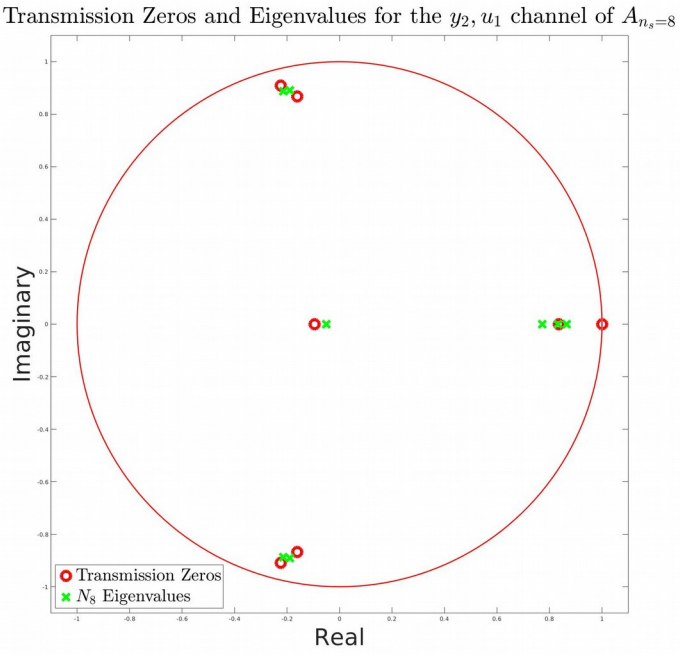
The singular values for each channel's Hankel matrix are plotted below.



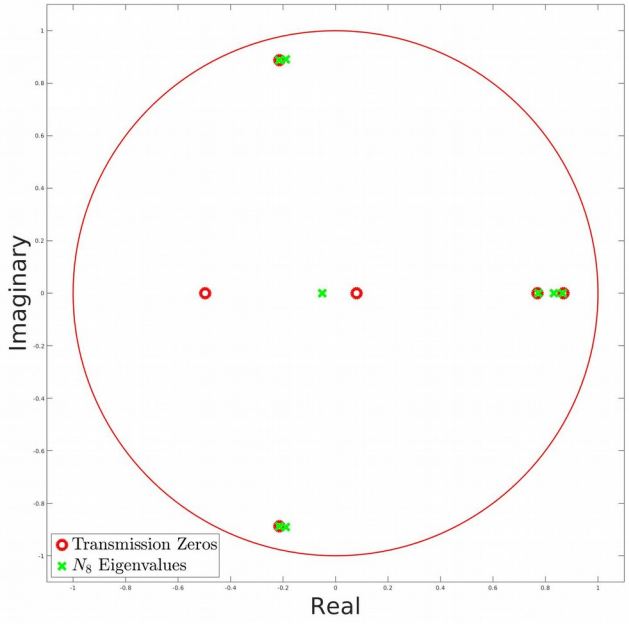
Each channel has fewer singular values on its own corresponding to the dynamics it is missing compared to the system as a whole. For both y_{11} and y_{12}, three of the model poles survive pole-zero cancellation by the transmission zeros of the channel, so we see three singular values at a higher tier magnitude above the rest for each of these channels. For y_{21}, we see two poles survive pole-zero cancellation, and for y_{22} we see four, so for each model we see two and four singular values, respectively, at a higher tier of magnitude above the rest of the singular values for that channel.

Below, see the pole-zero plots for each channel of $n_s = 8$. Notice that the pole-zero plot of each channel of the $n_s = 8$ model is nearly identical to that of the same channel in the $n_s = 7$ model with the exception of the introduction of a new pole accompanied by a zero in close proximity. This means that the extra state dimension the $n_s = 8$ model has over the $n_s = 7$ model does not have a great effect on the input-output properties of the modeled system.

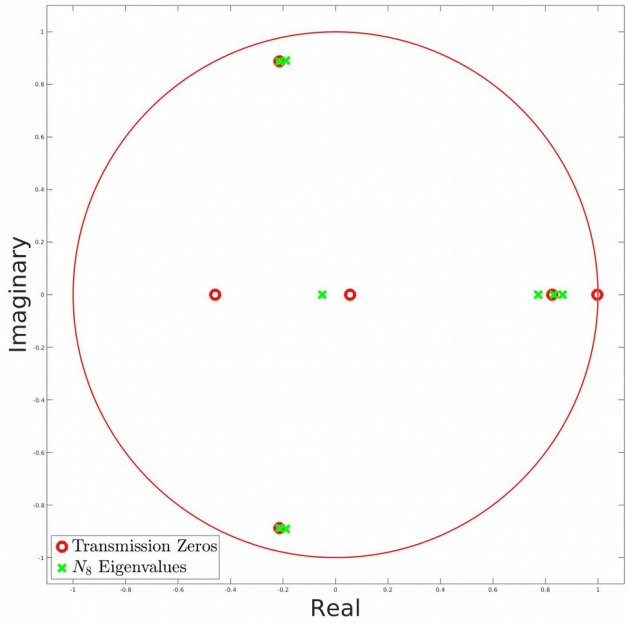




Transmission Zeros and Eigenvalues for the y_1, u_2 channel of $A_{n_s=8}$



Transmission Zeros and Eigenvalues for the y_2, u_2 channel of $A_{n_s=8}$



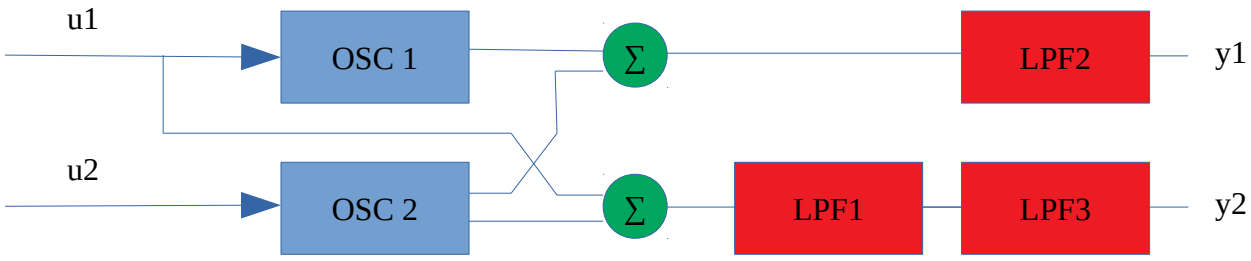
Task #3: Block diagram from analysis of individual channels

By analyzing the two input-two output system, we identify two oscillators, OSC1 at 72.2737 rad/s and OSC2 at 71.1394 rad/s. Additionally, we identify three distinct real poles on the positive real axis, LP1 at 0.7699, LP2 at 0.8258, and LP3 at 0.8686.

From pole-zero cancellations, it is clear that:

- y_{11} is affected by OSC 1 and LP2.
- y_{12} is affected by OSC 2 and LP2.
- y_{21} is affected by neither oscillator, but it is affected by LP1 and LP3.
- y_{22} is affected by OSC 2, LP1, and LP3.

Thus the following block diagram is constructed.



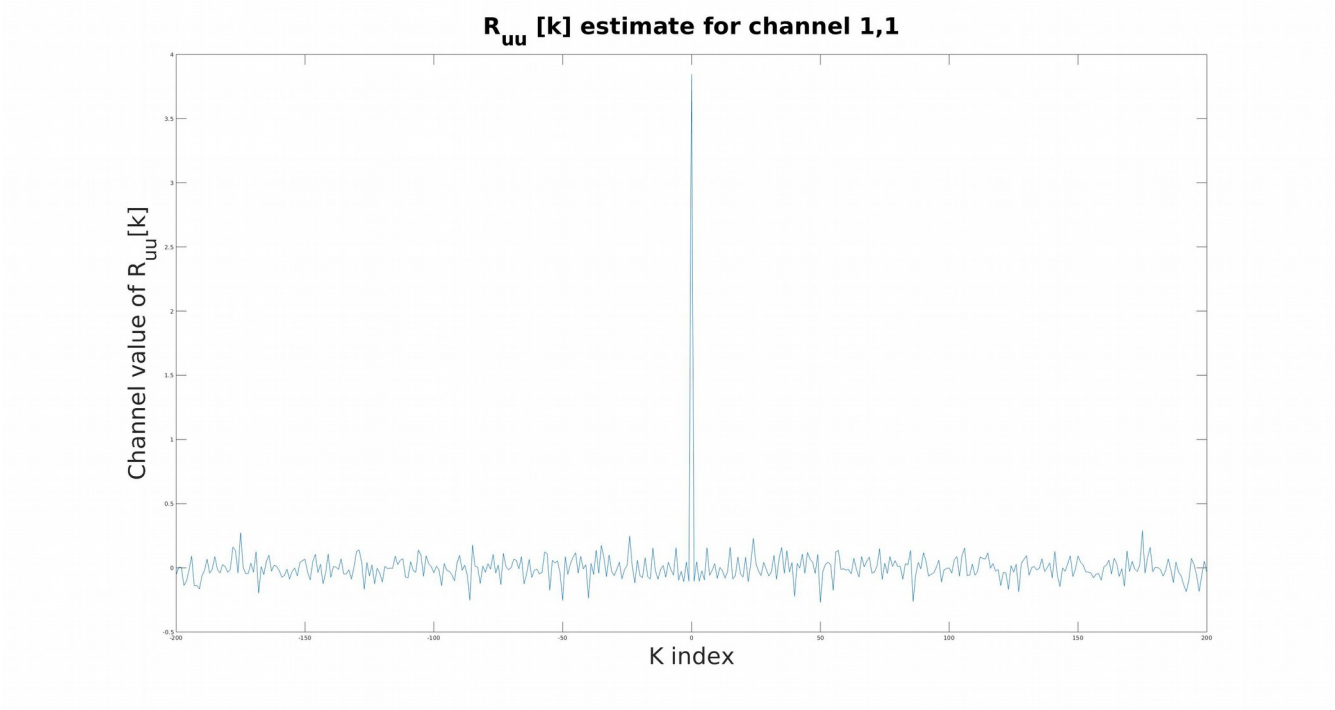
Task #4: Impulse Response identification from white noise inputs

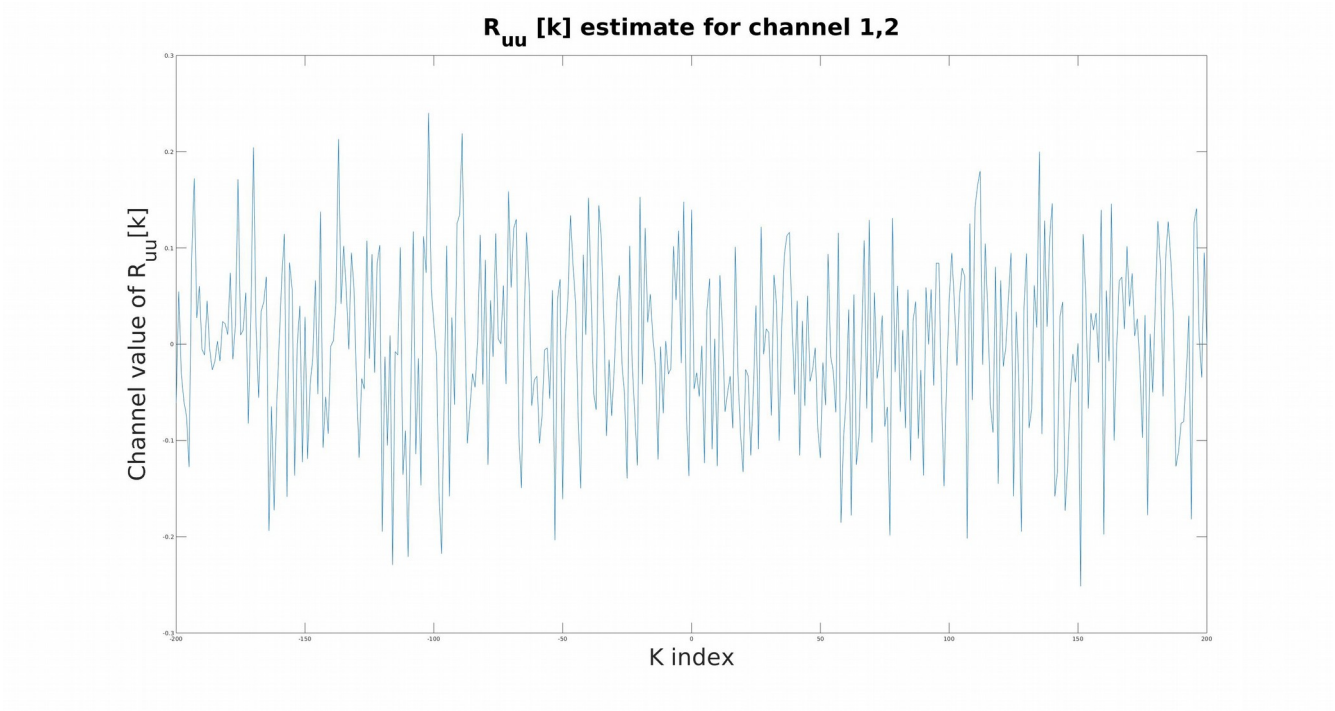
We can also recreate the impulse response from white noise input, which is useful in real experimental settings where there are limits on the impulse magnitude an experimental system can create or handle. In order to use white noise input, we must first verify that our input sequence is approximately zero mean. In whitenoise.m script, we take the mean of each white noise input signal u_1 and u_2.

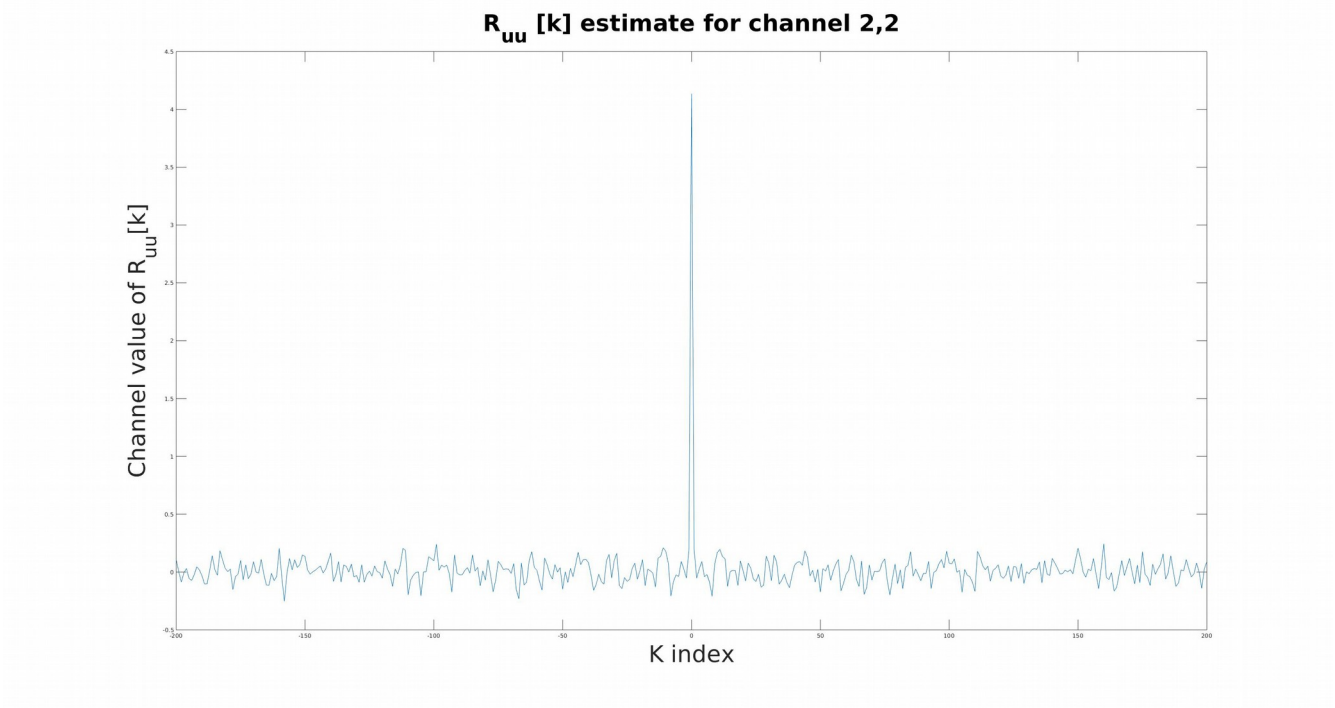
$$\bar{u_1} = -0.00096619$$

$$\bar{u_2} = 0.0013$$

We can visualize $R_{uu}[k]$, the auto-correlation of **u** across channels 1 and 2, below for $k \in [-200, 200]$, which corresponds to $\tau \in [-5,5]$ seconds





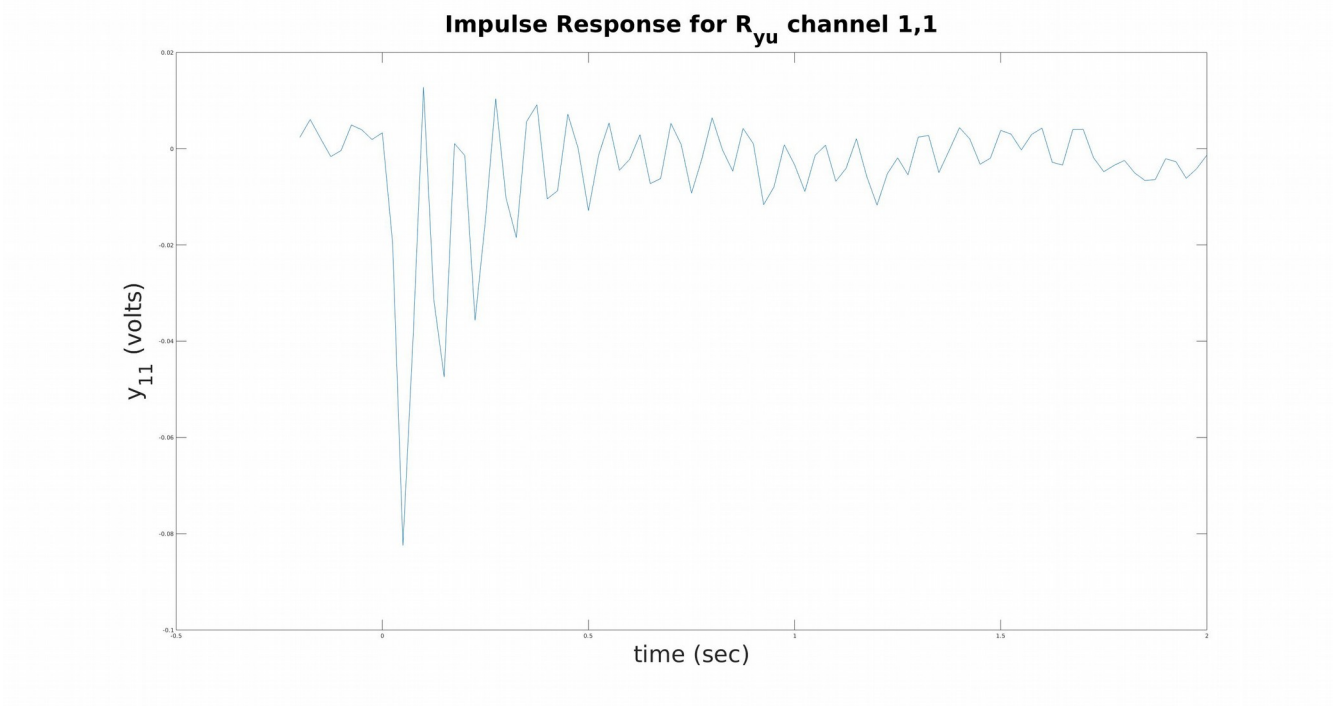


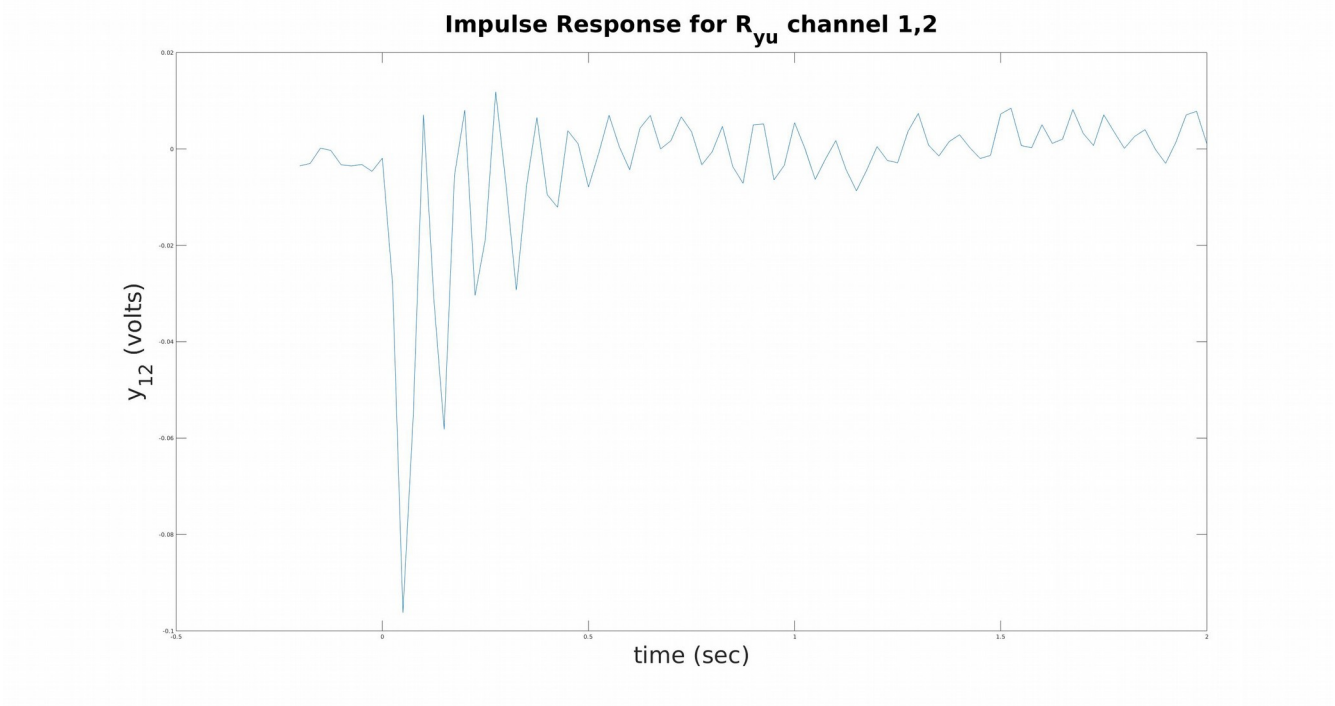
Note that

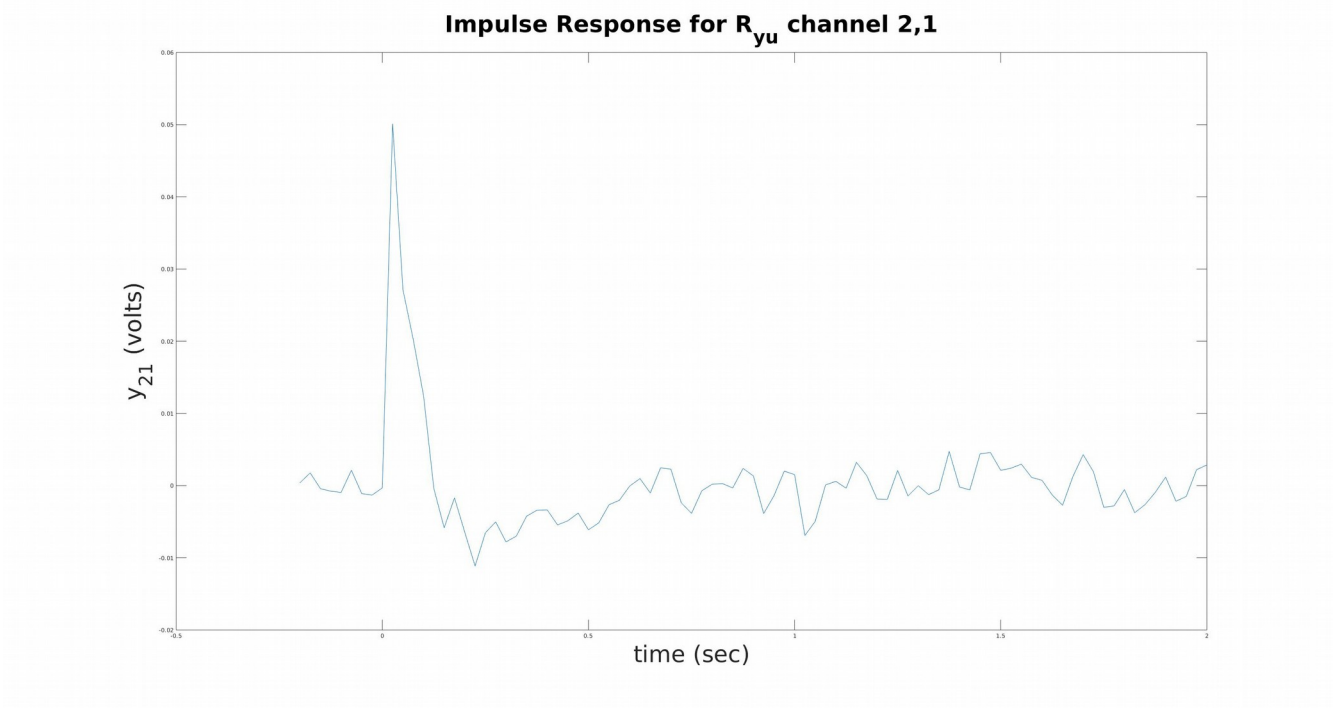
$$R_{uu}[0]=\begin{pmatrix}3.8460 & 0.1397\\ 0.1397 & 4.1354\end{pmatrix}\simeq\begin{pmatrix}4 & 0\\ 0 & 4\end{pmatrix}$$

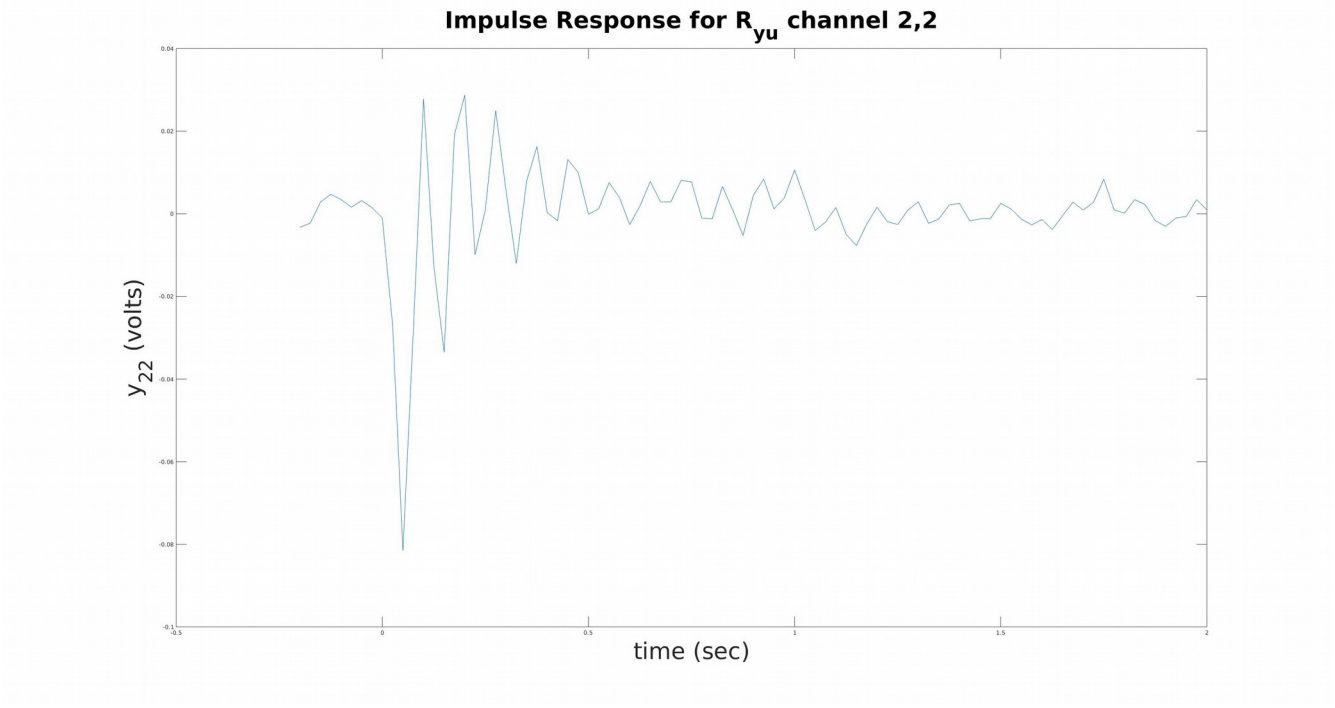
Therefore, the variance of each input channel is 4, not 1.

Finally, we can estimate $R_{yu}[k]$ for $\tau \in [-0.2, 2]$ seconds and observe the impulse response of each channel. This impulse response simulated from the cross-correlation $R_{yu}[k]$ matches the empirical impulse response.






Impulse Response for R_{yu} channel 2,1
y_{21} (volts)
time (sec)


Impulse Response for R_{yu} channel 2,2
y_{22} (volts)
time (sec)

Task # 5: H_2 norm analysis of identified model

We can use the H_2 norm to compare the white noise input-output data **y** with the $n_s = 7$ model and with the experimental pulse response data. The RMS value of the scaled output data y from the white noise input is

$$\|y\|_{RMS}^2 = 0.2068$$

The H_2 norm of P calculated using the observability Gramian (9), where P is the 7-state model derived from Hankel matrix analysis, is

$$\|P\|_{H_2} = 0.2189$$

And using the controllability Gramian (10), the H_2 norm of P is calculated to be

$$\|P\|_{H_2} = 0.2189$$

Finally, the H_2 norm of the experimental pulse response data is

$$\|P\|_{H_2} = 0.2091$$

In the case of the white noise input-output, the H_2 norm gives us the RMS measurement of the output **y**, and we can use it to compare against the H_2 norm calculations of the model obtained from Hankel matrix analysis as well as the experimental data to confirm that these different methods of analyzing the system are in agreement.

Task 6: H_∞ norm analysis of identified model

While the H_2 norm is useful from an analysis and controller design perspective, it fails to meet the submultiplicative property and is therefore unsuitable for addressing the robustness of the controller to perturbations of the plant dynamics. For this, we use the H_∞ norm instead. Using the H_∞ norm we can define a "nominal" plant model and a permissible deviation from that nominal model, quantifying these deviations using the norm. Additionally, we can analyze the robustness of a controller to perturbations in plant dynamics using the H_∞ norm.

To compute the H_∞ norm of the discrete-time model and the frequency at which it is achieved, use the script calchinfnorm.m. This returns

$$\|P\|_{H_\infty} = 0.4693, \omega = 71.1508\, rad/s$$

This norm matches with our frequency response calculated from the empirical data and the models, which shows a resonant peak for most of the channels at that frequency.

Appendix

constructhankels.m

```
startindex = 42
y11rel = y11(startindex:end);
y12rel = y12(startindex:end);
y21rel = y21(startindex:end);
y22rel = y22(startindex:end);
h20 = constructmat(20, y11rel, y12rel, y21rel, y22rel);
h40 = constructmat(40, y11rel, y12rel, y21rel, y22rel);
h80 = constructmat(80, y11rel, y12rel, y21rel, y22rel);
h100 = constructmat(100, y11rel, y12rel, y21rel, y22rel);
h100tilde = constructmattilde(100, y11rel, y12rel, y21rel, y22rel);

function hankel = constructmat(n, o11, o12, o21, o22)
   hankel = [];
   for i = 1:1:n
      r1 = [];
      r2 = [];
      for j = i:1:i+n-1
         r1 = [r1,o11(j),o12(j)];
         r2 = [r2,o21(j),o22(j)];
      end
      hankel = [hankel;r1;r2];
   end
end

function hankel = constructmattilde(n, o11, o12, o21, o22)
   hankel = [];
   for i = 2:1:n+1
      r1 = [];
      r2 = [];
      for j = i:1:i+n-1
         r1 = [r1,o11(j),o12(j)];
         r2 = [r2,o21(j),o22(j)];
      end
      hankel = [hankel;r1;r2];
   end
end
```

singularvals_1.m

```
s20 = svd(h20);
s40 = svd(h40);
s80 = svd(h80);
s100 = svd(h100);

%%% plot the singular values calculated above
figure(1);
semilogy(s20,'b*','LineWidth',2);
hold on
semilogy(s40,'r*','LineWidth',2);
semilogy(s80,'g*','LineWidth',2);
semilogy(s100,'c*','LineWidth',2);
hold off
ylabel('Singular value index');
xlabel('Hankel singular values');
legend({'$H_{20}$','$H_{40}$','$H_{80}$','$H_{100}$'},'FontSize',14,'Interpreter','Latex');
grid on;
ylim([10^-3 1]);
xlim([0 40]);
```

computemodels.m

```
[U, S, V] = svd(h100);

%%% compute a model for A from model order n_s=6
ns_6 = 6;
U1_6 = U(:,1:ns_6);
V1_6 = V(:,1:ns_6);
S_6 = S(1:ns_6,1:ns_6);
Obs_6 = U1_6;
Con_6 = S_6*V1_6';
Obs_6leftinv = inv(Obs_6'*Obs_6)*Obs_6';
Con_6rightinv = Con_6'*inv(Con_6*Con_6');
A_6 = Obs_6leftinv * h100tilde * Con_6rightinv;
max(abs(eig(A_6)))

%%% compute a model for A from model order n_s=7
ns_7 = 7;
U1_7 = U(:,1:ns_7);
V1_7 = V(:,1:ns_7);
S_7 = S(1:ns_7,1:ns_7);
Obs_7 = U1_7;
Con_7 = S_7*V1_7';
Obs_7leftinv = inv(Obs_7'*Obs_7)*Obs_7';
Con_7rightinv = Con_7'*inv(Con_7*Con_7');
A_7 = Obs_7leftinv * h100tilde * Con_7rightinv;
max(abs(eig(A_7)))

%%% compute a model for A from model order n_s=8
ns_8 = 8;
U1_8 = U(:,1:ns_8);
V1_8 = V(:,1:ns_8);
S_8 = S(1:ns_8,1:ns_8);
Obs_8 = U1_8;
Con_8 = S_8*V1_8';
Obs_8leftinv = inv(Obs_8'*Obs_8)*Obs_8';
Con_8rightinv = Con_8'*inv(Con_8*Con_8');
A_8 = Obs_8leftinv * h100tilde * Con_8rightinv;
max(abs(eig(A_8)))


%%% compute a model for A from model order n_s=10
ns_10 = 10;
U1_10 = U(:,1:ns_10);
V1_10 = V(:,1:ns_10);
S_10 = S(1:ns_10,1:ns_10);
Obs_10 = U1_10;
Con_10 = S_10*V1_10';
Obs_10leftinv = inv(Obs_10'*Obs_10)*Obs_10';
Con_10rightinv = Con_10'*inv(Con_10*Con_10');
```

```
A_10 = Obs_10leftinv * h100tilde * Con_10rightinv;
max(abs(eig(A_10)))


%%% compute a model for A from model order n_s=20
ns_20 = 20;
U1_20 = U(:,1:ns_20);
V1_20 = V(:,1:ns_20);
S_20 = S(1:ns_20,1:ns_20);
Obs_20 = U1_20;
Con_20 = S_20*V1_20';
Obs_20leftinv = inv(Obs_20'*Obs_20)*Obs_20';
Con_20rightinv = Con_20'*inv(Con_20*Con_20');
A_20 = Obs_20leftinv * h100tilde * Con_20rightinv;
max(abs(eig(A_20)))
```

sim_impulse_response.m

```
%%% Simulate impulse response for ns=6
C_6 = Obs_6(1:2,:);
B_6 = Con_6(:,1:2);
y11_6 = [];
y12_6 = [];
y21_6 = [];
y22_6 = [];

for k = 1:1:80
   hk = C_6*A_6^(k-1)*B_6
   y11_6 = [y11_6, hk(1,1)];
   y12_6 = [y12_6, hk(1,2)];
   y21_6 = [y21_6, hk(2,1)];
   y22_6 = [y22_6, hk(2,2)];
end


%%% Simulate impulse response for ns=7
C_7 = Obs_7(1:2,:);
B_7 = Con_7(:,1:2);
y11_7 = [];
y12_7 = [];
y21_7 = [];
y22_7 = [];

for k = 1:1:80
   hk = C_7*A_7^(k-1)*B_7
   y11_7 = [y11_7, hk(1,1)];
   y12_7 = [y12_7, hk(1,2)];
   y21_7 = [y21_7, hk(2,1)];
   y22_7 = [y22_7, hk(2,2)];
end

%%% Simulate impulse response for ns=10
C_10 = Obs_10(1:2,:);
B_10 = Con_10(:,1:2);
y11_10 = [];
y12_10 = [];
y21_10 = [];
y22_10 = [];

for k = 1:1:80
   hk = C_10*A_10^(k-1)*B_10
   y11_10 = [y11_10, hk(1,1)];
   y12_10 = [y12_10, hk(1,2)];
   y21_10 = [y21_10, hk(2,1)];
   y22_10 = [y22_10, hk(2,2)];
end
```

```
%%% Simulate impulse response for ns=20
C_20 = Obs_20(1:2,:);
B_20 = Con_20(:,1:2);
y11_20 = [];
y12_20 = [];
y21_20 = [];
y22_20 = [];

for k = 1:1:80
   hk = C_20*A_20^(k-1)*B_20
   y11_20 = [y11_20, hk(1,1)];
   y12_20 = [y12_20, hk(1,2)];
   y21_20 = [y21_20, hk(2,1)];
   y22_20 = [y22_20, hk(2,2)];
end

%%% plot impulse response for y11 across all models
figure(1);
hold on
plot(y11_6,'m*','LineWidth',2,'MarkerSize',6);
plot(y11_7,'gp','MarkerSize',15,'LineWidth',15);
plot(y11_10,'ro','LineWidth',8,'MarkerSize',8);
plot(y11_20,'y>','LineWidth',2,'MarkerSize',6);
plot(y11rel, 'k*','LineWidth',2,'MarkerSize',4);
hold off
xlim([0 80]);
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southeast');
ylabel('y_{11} (volts)','FontSize',40);
xlabel('Samples','FontSize',40);
title('$y_{11}$ Impulse Response' ,'FontSize',40,'Interpreter','Latex');


%%% plot impulse response for y12 across all models
figure(2);
hold on
plot(y12_6,'m*','LineWidth',2,'MarkerSize',6);
plot(y12_7,'gp','MarkerSize',15,'LineWidth',15);
plot(y12_10,'ro','LineWidth',8,'MarkerSize',8);
plot(y12_20,'y>','LineWidth',2,'MarkerSize',6);
plot(y12rel, 'k*','LineWidth',2,'MarkerSize',4);
hold off
xlim([0 80]);
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',20,'Interpreter','Latex',
'Location', 'southeast');
ylabel('y_{12} (volts)','FontSize',40);
xlabel('Samples','FontSize',40);
title('$y_{12}$ Impulse Response','FontSize',40, 'Interpreter','Latex');
```

```
%%% plot impulse response for y21 across all models
figure(3);
hold on
plot(y21_6,'m*','LineWidth',2,'MarkerSize',6);
plot(y21_7,'gp','MarkerSize',15,'LineWidth',15);
plot(y21_10,'ro','LineWidth',8,'MarkerSize',8);
plot(y21_20,'y>','LineWidth',2,'MarkerSize',6);
plot(y21rel, 'k*','LineWidth',2,'MarkerSize',4);
hold off
xlim([0 80]);
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',20,'Interpreter','Latex');
ylabel('y_{21} (volts)','FontSize',40);
xlabel('Samples','FontSize',40);
title('$y_{21}$ Impulse Response','FontSize',40, 'Interpreter','Latex');

%%% plot impulse response for y22 across all models
figure(4);
hold on
plot(y22_6,'m*','LineWidth',2,'MarkerSize',6);
plot(y22_7,'gp','MarkerSize',15,'LineWidth',15);
plot(y22_10,'ro','LineWidth',8,'MarkerSize',8);
plot(y22_20,'y>','LineWidth',2,'MarkerSize',6);
plot(y22rel, 'k*','LineWidth',2,'MarkerSize',4);
hold off
xlim([0 80]);
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',20,'Interpreter','Latex',
'Location', 'southeast');
ylabel('y_{22} (volts)','FontSize',40);
xlabel('Samples','FontSize',40);
title('$y_{22}$ Impulse Response','FontSize',40, 'Interpreter','Latex');
```

freqresponse.m

```
wnyq = 20.0*2*pi;
w = [0:1:wnyq];
ts = 1/40;

% calculate the freq response for ns=6
yf6 = [];
for i=1:1:length(w)
   hk = C_6/(exp(1j*w(i)*ts)*eye(length(A_6))-A_6)*B_6;
   yf6 = [yf6; hk(1,1), hk(1,2), hk(2,1), hk(2,2)];
end

% calculate the freq response for ns=7
yf7 = [];
for i=1:1:length(w)
   hk = C_7/(exp(1j*w(i)*ts)*eye(length(A_7))-A_7)*B_7;
   yf7 = [yf7; hk(1,1), hk(1,2), hk(2,1), hk(2,2)];
end

% calculate the freq response for ns=10
yf10 = [];
for i=1:1:length(w)
   hk = C_10/(exp(1j*w(i)*ts)*eye(length(A_10))-A_10)*B_10;
   yf10 = [yf10; hk(1,1), hk(1,2), hk(2,1), hk(2,2)];
end

% calculate the freq response for ns=20
yf20 = [];
for i=1:1:length(w)
   hk = C_20/(exp(1j*w(i)*ts)*eye(length(A_20))-A_20)*B_20;
   yf20 = [yf20; hk(1,1), hk(1,2), hk(2,1), hk(2,2)];
end

% calculate the freq response for the lab data
y11f = fft(y11)./fft(u1);
om11 = [0:length(y11f)-1]*2*pi/(ts*length(y11f)); %%%% frequency vector in rad/s
y12f = fft(y12)./fft(u1);
om12 = [0:length(y12f)-1]*2*pi/(ts*length(y12f)); %%%% frequency vector in rad/s
y21f = fft(y21)./fft(u2);
om21 = [0:length(y21f)-1]*2*pi/(ts*length(y21f)); %%%% frequency vector in rad/s
y22f = fft(y22)./fft(u2);
om22 = [0:length(y22f)-1]*2*pi/(ts*length(y22f)); %%%% frequency vector in rad/s

labelfontsize = 40;

% magnitude plot for all y_11
index = 1;
figure(index);
loglog(w, abs(yf6(:,index)),'m','LineWidth',3);
```

```
hold on
loglog(w, abs(yf7(:,index)),'g','LineWidth',4);
loglog(w, abs(yf10(:,index)),'r','LineWidth',4);
loglog(w, abs(yf20(:,index)),'y','LineWidth',4);
loglog(om11(1,1:floor(length(om11)/2)),abs(y11f(1,1:floor(length(y11f)/2))), 'k', 'LineWidth',2);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
xlim([10^0,125]);
ylim([10^-3 10^0]);
ylabel('Magnitude', 'FontSize',labelfontsize);
xlabel('Frequency (rad/s)', 'FontSize',labelfontsize);
title('y_{11} Magnitude Plot','FontSize',40);

% magnitude plot for all y_12
index=2;
figure(index);
loglog(w, abs(yf6(:,index)),'m','LineWidth',3);
hold on
loglog(w, abs(yf7(:,index)),'g','LineWidth',4);
loglog(w, abs(yf10(:,index)),'r','LineWidth',4);
loglog(w, abs(yf20(:,index)),'y','LineWidth',4);
loglog(om12(1,1:floor(length(om12)/2)),abs(y12f(1,1:floor(length(y12f)/2))), 'k', 'LineWidth',2);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
xlim([10^0,125]);
ylim([10^-3 10^0]);
ylabel('Magnitude', 'FontSize',labelfontsize);
xlabel('Frequency (rad/s)', 'FontSize',labelfontsize);
title('y_{12} Magnitude Plot','FontSize',40);


% magnitude plot for all y_21
index=3;
figure(index);
loglog(w, abs(yf6(:,index)),'m','LineWidth',3);
hold on
loglog(w, abs(yf7(:,index)),'g','LineWidth',4);
loglog(w, abs(yf10(:,index)),'r','LineWidth',4);
loglog(w, abs(yf20(:,index)),'y','LineWidth',4);
loglog(om21(1,1:floor(length(om21)/2)),abs(y21f(1,1:floor(length(y21f)/2))), 'k', 'LineWidth',2);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
xlim([10^0,125]);
ylim([10^-3 10^0]);
ylabel('Magnitude', 'FontSize',labelfontsize);
xlabel('Frequency (rad/s)', 'FontSize',labelfontsize);
```

```
title('y_{21} Magnitude Plot','FontSize',40);


% magnitude plot for all y_22
index=4;
figure(index);
loglog(w, abs(yf6(:,index)),'m','LineWidth',3);
hold on
loglog(w, abs(yf7(:,index)),'g','LineWidth',4);
loglog(w, abs(yf10(:,index)),'r','LineWidth',4);
loglog(w, abs(yf20(:,index)),'y','LineWidth',4);
loglog(om22(1,1:floor(length(om22)/2)),abs(y22f(1,1:floor(length(y22f)/2))), 'k', 'LineWidth',2);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
xlim([10^0,125]);
ylim([10^-3 10^0]);
ylabel('Magnitude', 'FontSize',labelfontsize);
xlabel('Frequency (rad/s)', 'FontSize',labelfontsize);
title('y_{22} Magnitude Plot','FontSize',40);


% phase plot for all y_11
index = 1;
figure(5);
semilogx(w, angle(yf6(:,index)), 'm','LineWidth',3);
hold on
semilogx(w, angle(yf7(:,index)), 'g','LineWidth',5);
semilogx(w, angle(yf10(:,index)), 'r','LineWidth',3);
semilogx(w, angle(yf20(:,index)), 'y','LineWidth',3);
semilogx(om11(1,1:floor(length(om11)/2)),angle(y11f(1,1:floor(length(y11f)/2))), 'k','LineWidth',3);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
title('y_{11} Phase Plot','FontSize',40);
ylabel('Phase (radians)', 'FontSize', labelfontsize);
xlabel('Frequency (rad/sec)', 'FontSize', labelfontsize);

% phase plot for all y_12
index = 2;
figure(6);
semilogx(w, angle(yf6(:,index)), 'm','LineWidth',3);
hold on
semilogx(w, angle(yf7(:,index)), 'g','LineWidth',5);
semilogx(w, angle(yf10(:,index)), 'r','LineWidth',3);
semilogx(w, angle(yf20(:,index)), 'y','LineWidth',3);
semilogx(om12(1,1:floor(length(om12)/2)),angle(y12f(1,1:floor(length(y12f)/2))), 'k','LineWidth',3);
hold off
```

```
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
title('y_{12} Phase Plot','FontSize',40);
ylabel('Phase (radians)', 'FontSize', labelfontsize);
xlabel('Frequency (rad/sec)', 'FontSize', labelfontsize);

% phase plot for all y_21
index = 3;
figure(7);
semilogx(w, angle(yf6(:,index)), 'm','LineWidth',3);
hold on
semilogx(w, angle(yf7(:,index)), 'g','LineWidth',5);
semilogx(w, angle(yf10(:,index)), 'r','LineWidth',3);
semilogx(w, angle(yf20(:,index)), 'y','LineWidth',3);
semilogx(om21(1,1:floor(length(om21)/2)),angle(y21f(1,1:floor(length(y21f)/2))), 'k','LineWidth',3);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'southwest');
title('y_{21} Phase Plot','FontSize',40);
ylabel('Phase (radians)', 'FontSize', labelfontsize);
xlabel('Frequency (rad/sec)', 'FontSize', labelfontsize);

% phase plot for all y_22
index = 4;
figure(8);
semilogx(w, angle(yf6(:,index)), 'm','LineWidth',3);
hold on
semilogx(w, angle(yf7(:,index)), 'g','LineWidth',5);
semilogx(w, angle(yf10(:,index)), 'r','LineWidth',3);
semilogx(w, angle(yf20(:,index)), 'y','LineWidth',3);
semilogx(om22(1,1:floor(length(om22)/2)),angle(y22f(1,1:floor(length(y22f)/2))), 'k','LineWidth',3);
hold off
legend({'$N_{6}$','$N_{7}$', '$N_{10}$', '$N_{20}$', 'Lab Data'},'FontSize',40,'Interpreter','Latex',
'Location', 'northeast');
title('y_{22} Phase Plot','FontSize',40);
ylabel('Phase (radians)', 'FontSize', labelfontsize);
xlabel('Frequency (rad/sec)', 'FontSize', labelfontsize);
```

transmissionzeros.m

```
% get transmission zeros of S(alpha)
M = [A_7, B_7; -1.*C_7, -1.*zeros(2)];
K = [eye(7), zeros(7, 2); zeros(2, 9)];
tzeros = eig(M, K);
[tvecs, tzerosDiag] = eig(M,K);
realz = real(tzeros);
imagz = imag(tzeros);

% get the discrete-time eigenvalues of the A_7 model
evals = eig(A_7);
reale = real(evals);
image = imag(evals);

% % plot the transmission zeros and evals
% font = 'Helvetica';
% plot(realz, imagz, 'o', 'LineWidth', 12, 'MarkerSize', 10);
% hold on
% plot(reale, image, 'x', 'LineWidth', 12, 'MarkerSize', 10);
% viscircles([0, 0], 1)
% hold off
% xlim([-1.1, 1.1]);
% ylim([-1.1, 1.1]);
% axis square
% legend({'Transmission Zeros', '$N_{7}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
% ylabel('Imaginary', 'FontSize',40, 'fontname', font);
% xlabel('Real', 'FontSize',40, 'fontname', font);
% title('Transmission Zeros and Eigenvalues for $A_{n_{s}=7}$','Interpreter','Latex','FontSize',40,
'fontname', font);

% calculate the continuous time evals of A_7
evalsc = [];
for i=1:length(evals)
    evalsc = [evalsc; log(evals(i))/ts];
end

% link transmission zeros to each channel
for i=2:6
   i;
   y_k = C_7*tvecs(1:7,i).*tzerosDiag(i,i)^100;
   tzerosDiag(i,i);
end

% calculate and plot the transmission zeros and eigenvalues for each channel
Kchan = [eye(7), zeros(7, 1); zeros(1, 8)]; % this K is the same for all channels, the right matrix in the
generalized eigenvalue problem
markersizetzeros = 16;
linewidthtzeros = 16;
```

```
% 11 channel corresponding to y1 and u1
M11 = [A_7, B_7(:,1); -1.*C_7(1,:), -1.*zeros(1)];
tz11 = eig(M11, Kchan);
[tvecs11, tz11Diag] = eig(M11,Kchan);
realz11 = real(tz11);
imagz11 = imag(tz11);

figure(2)
font = 'Helvetica';
plot(realz11, imagz11, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale, image, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{7}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{1}, u_{1}$ channel of
$A_{n_{s}=7}$','Interpreter','Latex','FontSize',40, 'fontname', font);


% 12 channel corresponding to y1 and u2
M12 = [A_7, B_7(:,2); -1.*C_7(1,:), -1.*zeros(1)];
tz12 = eig(M12, Kchan);
[tvecs12, tz12Diag] = eig(M12,Kchan);
realz12 = real(tz12);
imagz12 = imag(tz12);

figure(3)
font = 'Helvetica';
plot(realz12, imagz12, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale, image, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{7}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{1}, u_{2}$ channel of
$A_{n_{s}=7}$','Interpreter','Latex','FontSize',40, 'fontname', font);
```

```
% 21 channel corresponding to y2 and u1
M21 = [A_7, B_7(:,1); -1.*C_7(2,:), -1.*zeros(1)];
tz21 = eig(M21, Kchan);
[tvecs21, tz21Diag] = eig(M21,Kchan);
realz21 = real(tz21);
imagz21 = imag(tz21);

figure(4)
font = 'Helvetica';
plot(realz21, imagz21, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale, image, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{7}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{2}, u_{1}$ channel of
$A_{n_{s}=7}$','Interpreter','Latex','FontSize',40, 'fontname', font);

% 22 channel corresponding to y2 and u2
M22 = [A_7, B_7(:,2); -1.*C_7(2,:), -1.*zeros(1)];
tz22 = eig(M22, Kchan);
[tvecs22, tz22Diag] = eig(M22,Kchan);
realz22 = real(tz22);
imagz22 = imag(tz22);

figure(5)
font = 'Helvetica';
plot(realz22, imagz22, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale, image, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{7}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{2}, u_{2}$ channel of
$A_{n_{s}=7}$','Interpreter','Latex','FontSize',40, 'fontname', font);
```

```
% construct Hankel matrices for each channel and calculate the singular
% values
nhankel = 100;

% y11 hankel
y11hankel = [];
for i=1:nhankel
   thisrow = [];
   for k=i:i+nhankel-1
      thisrow = [thisrow, y11rel(k)];
   end
   y11hankel = [y11hankel; thisrow];
end
y11hankel_sv = svd(y11hankel);

% y12 hankel
y12hankel = [];
for i=1:nhankel
   thisrow = [];
   for k=i:i+nhankel-1
      thisrow = [thisrow, y12rel(k)];
   end
   y12hankel = [y12hankel; thisrow];
end
y12hankel_sv = svd(y12hankel);

% y21 hankel
y21hankel = [];
for i=1:nhankel
   thisrow = [];
   for k=i:i+nhankel-1
      thisrow = [thisrow, y21rel(k)];
   end
   y21hankel = [y21hankel; thisrow];
end
y21hankel_sv = svd(y21hankel);

% y22 hankel
y22hankel = [];
for i=1:nhankel
   thisrow = [];
   for k=i:i+nhankel-1
      thisrow = [thisrow, y22rel(k)];
   end
   y22hankel = [y22hankel; thisrow];
end
y22hankel_sv = svd(y22hankel);
```

```
% plot singular value index for each channel
% uncomment to display plot
markersize = 16;
linesize = 16;
figure(6);
semilogy(y11hankel_sv,'b*','LineWidth',linesize, 'MarkerSize', markersize);
hold on
semilogy(y12hankel_sv,'r*','LineWidth',linesize, 'MarkerSize', markersize);
semilogy(y21hankel_sv,'g*','LineWidth',linesize, 'MarkerSize', markersize);
semilogy(y22hankel_sv,'c*','LineWidth',linesize, 'MarkerSize', markersize);
hold off
ylabel('Hankel singular values', 'FontSize', 40, 'fontname', font);
xlabel('Singular value index', 'FontSize', 40, 'fontname', font);
legend({'$y_{11}$','$y_{12}$','$y_{21}$','$y_{22}$'},'FontSize',40,'Interpreter','Latex', 'fontname',
font);
title('Singular values for the Hankel matrix belonging to each channel', 'FontSize', 40, 'fontname', font);
grid on;
ylim([10^-3 1]);
xlim([0 40]);

%% ns=8
%%% Create the pole-zero plot for ns=8 as a whole

C_8 = Obs_8(1:2,:);
B_8 = Con_8(:,1:2);
M_8 = [A_8, B_8; -1.*C_8, -1.*zeros(2)];
K_8 = [eye(8), zeros(8, 2); zeros(2, 10)];
tzeros_8 = eig(M_8, K_8); % transmission zeros of the ns=8 model
[tvecs_8, tzerosDiag_8] = eig(M_8,K_8); %tzeros and their vectors
realz_8 = real(tzeros_8); % for plotting
imagz_8 = imag(tzeros_8); % for plotting

% get the discrete-time eigenvalues of the A_8 model
evals_8 = eig(A_8);
reale_8 = real(evals_8);
image_8 = imag(evals_8);

% plot the transmission zeros and evals of the ns=8 model
figure(7)
plot(realz_8, imagz_8, 'o', 'LineWidth', 12, 'MarkerSize', 10);
hold on
plot(reale_8, image_8, 'x', 'LineWidth', 12, 'MarkerSize', 10);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{8}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
```

```
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for $A_{n_{s}=8}$','Interpreter','Latex','FontSize',40,
'fontname', font);

% Calculate and create the pole-zero plot for each channel of ns=8
% these plots should all have an extra pole-zero cancellation so it is
% essentially the same as ns=7
Kchan_8 = [eye(8), zeros(8, 1); zeros(1, 9)]; % this K is the same for all channels, the right matrix in
the generalized eigenvalue problem
markersizetzeros = 16;
linewidthtzeros = 16;

% 11 channel corresponding to y1 and u1
M11_8 = [A_8, B_8(:,1); -1.*C_8(1,:), -1.*zeros(1)];
tz11_8 = eig(M11_8, Kchan_8);
[tvecs11_8, tz11Diag_8] = eig(M11_8,Kchan_8);
realz11_8 = real(tz11_8);
imagz11_8 = imag(tz11_8);


figure(8)
font = 'Helvetica';
plot(realz11_8, imagz11_8, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale_8, image_8, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{8}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{1}, u_{1}$ channel of
$A_{n_{s}=8}$','Interpreter','Latex','FontSize',40, 'fontname', font);


% 12 channel corresponding to y1 and u2
M12_8 = [A_8, B_8(:,2); -1.*C_8(1,:), -1.*zeros(1)];
tz12_8 = eig(M12_8, Kchan_8);
[tvecs12_8, tz12Diag_8] = eig(M12_8,Kchan_8);
realz12_8 = real(tz12_8);
imagz12_8 = imag(tz12_8);

figure(9)
font = 'Helvetica';
plot(realz12_8, imagz12_8, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
```

```
hold on
plot(reale_8, image_8, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{8}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{1}, u_{2}$ channel of
$A_{n_{s}=8}$','Interpreter','Latex','FontSize',40, 'fontname', font);

% 21 channel corresponding to y2 and u1
M21_8 = [A_8, B_8(:,1); -1.*C_8(2,:), -1.*zeros(1)];
tz21_8 = eig(M21_8, Kchan_8);
[tvecs21_8, tz21Diag_8] = eig(M21_8,Kchan_8);
realz21_8 = real(tz21_8);
imagz21_8 = imag(tz21_8);

figure(10)
font = 'Helvetica';
plot(realz21_8, imagz21_8, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
plot(reale_8, image_8, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{8}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{2}, u_{1}$ channel of
$A_{n_{s}=8}$','Interpreter','Latex','FontSize',40, 'fontname', font);

% 22 channel corresponding to y2 and u2
M22_8 = [A_8, B_8(:,2); -1.*C_8(2,:), -1.*zeros(1)];
tz22_8 = eig(M22_8, Kchan_8);
[tvecs22_8, tz22Diag_8] = eig(M22_8,Kchan_8);
realz22_8 = real(tz22_8);
imagz22_8 = imag(tz22_8);

figure(11)
font = 'Helvetica';
plot(realz22_8, imagz22_8, 'ro', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
hold on
```

```
plot(reale_8, image_8, 'gx', 'LineWidth', linewidthtzeros, 'MarkerSize', markersizetzeros);
viscircles([0, 0], 1)
hold off
xlim([-1.1, 1.1]);
ylim([-1.1, 1.1]);
axis square
legend({'Transmission Zeros', '$N_{8}$ Eigenvalues'},'FontSize',30,'Interpreter','Latex', 'Location',
'southwest', 'fontname', font);
ylabel('Imaginary', 'FontSize',40, 'fontname', font);
xlabel('Real', 'FontSize',40, 'fontname', font);
title('Transmission Zeros and Eigenvalues for the $y_{2}, u_{2}$ channel of
$A_{n_{s}=8}$','Interpreter','Latex','FontSize',40, 'fontname', font);
```

whitenoise.m

```
syms k
y1 = u_rand.Y(3).Data;
y2 = u_rand.Y(4).Data;
y = [y1;y2];
u1 = u_rand.Y(1).Data;
u2 = u_rand.Y(2).Data;
u = [u1; u2];
ut = u';
ts = 1/40;
N = length(y1);
t = [0:N-1]*ts - 1;

% part 1: verify that each input sequence is approx zero mean
u1mean = mean(u1); %   -9.6619e-04
u2mean = mean(u2); %     0.0013

% part 2: Determine and graph estimates of the four entries of R_uu for indices k
% plot the entries versus lag factor k
% kzero = length(u1)/2-1;
kzero = 12000;
p = 1000;
pvec = [-p+kzero, p+kzero];

% Ruu is a 3D array where each "page" is a 2x2 matrix of Ruu[k], and each
% page has index k and there is a page for all k in [-200, 200]. Examples:
% https://www.mathworks.com/help/matlab/math/multidimensional-arrays.html

Ruu = zeros(2,2,401);
for k=kzero-200:kzero+200
   for q = -p:p
      Ruu(:,:,k-kzero+201) = Ruu(:,:,k-kzero+201) + u(:,k+q)*ut(q+kzero,:);
   end
   Ruu(:,:,k-kzero+201) = Ruu(:,:,k-kzero+201) *1/(2*p);
end

kvec = [-200:200];
% figure(1)
% plot(kvec,squeeze(Ruu(1,1,:)));
% xlabel('K index', 'FontSize', 40);
% ylabel('Channel value of R_{uu}[k]', 'FontSize', 40);
% title('R_{uu} [k] estimate for channel 1,1', 'FontSize', 40)
% figure(2)
% plot(kvec,squeeze(Ruu(1,2,:)));
% xlabel('K index', 'FontSize', 40);
% ylabel('Channel value of R_{uu}[k]', 'FontSize', 40);
% title('R_{uu} [k] estimate for channel 1,2', 'FontSize', 40)
% figure(3)
% plot(kvec,squeeze(Ruu(2,1,:)));
```

```
% xlabel('K index', 'FontSize', 40);
% ylabel('Channel value of R_{uu}[k]', 'FontSize', 40);
% title('R_{uu} [k] estimate for channel 2,1', 'FontSize', 40)
% figure(4)
% plot(kvec,squeeze(Ruu(2,2,:)));
% xlabel('K index', 'FontSize', 40);
% ylabel('Channel value of R_{uu}[k]', 'FontSize', 40);
% title('R_{uu} [k] estimate for channel 2,2', 'FontSize', 40)

% Ruu[0] is approximately [[1,0],[0,1]]
Ruu(:,:,201)


% part 4: estimate R_yu[k]
ts = 1/40;
u1var = var(u1);
u2var = var(u2);
kyu = [-0.2/ts:2.0/ts];
lenkyu = length(kyu);
Ryu = zeros(2,2,lenkyu);
for k=kzero+kyu(1):kzero+kyu(end)
   for q = -p:p
      Ryu(:,:,k-kzero-kyu(1)+1) = Ryu(:,:,k-kzero-kyu(1)+1) + y(:,k+q)*ut(q+kzero,:);
   end
   Ryu(:,:,k-kzero-kyu(1)+1) = Ryu(:,:,k-kzero-kyu(1)+1) *1/(2*p);
end
yutvec = kyu.*ts;

% plot each column of Ryu normalized by the variance of the appropriate
% channel of u. These plots resemble the impulse response of the empirical
% data and the models generated previously
figure(5)
plot(yutvec,squeeze(Ryu(1,1,:))/u1var);
title('Impulse Response for R_{yu} channel 1,1', 'FontSize', 40);
xlabel('time (sec)', 'FontSize', 40)
ylabel('y_{11} (volts)', 'FontSize', 40)
figure(6)
plot(yutvec,squeeze(Ryu(1,2,:))/u2var);
title('Impulse Response for R_{yu} channel 1,2', 'FontSize', 40);
xlabel('time (sec)', 'FontSize', 40)
ylabel('y_{12} (volts)', 'FontSize', 40)
figure(7)
plot(yutvec,squeeze(Ryu(2,1,:))/u1var);
title('Impulse Response for R_{yu} channel 2,1', 'FontSize', 40);
xlabel('time (sec)', 'FontSize', 40)
ylabel('y_{21} (volts)', 'FontSize', 40)
figure(8)
plot(yutvec,squeeze(Ryu(2,2,:))/u2var);
title('Impulse Response for R_{yu} channel 2,2', 'FontSize', 40);
```

```
xlabel('time (sec)', 'FontSize', 40)
ylabel('y_{22} (volts)', 'FontSize', 40)
```

h2norm.m

```
% part 1: calculate the RMS value of the scaled output data y
Ryy0 = zeros(2,2);
yt = y';
k = kzero;
for q = -p:p
   Ryy0 = Ryy0 + y(:,k+q)*yt(k+q,:);
end
Ryy0 = Ryy0 *1/(2*p);
% rmsy_1 = (trace(Ryy0))^(1/2)
rmsy_1 = (trace(Ryy0))


% part 2: Use (9) and (10) to compute the H2 norm
gramo = zeros(7,7);
for k=1:1000
   gramo = gramo + A_7'^k*C_7'*C_7*A_7^k;
end
rmsy_9 = (trace(B_7'*gramo*B_7))^(1/2) % 0.2189

gramc = zeros(7,7);
for k=1:1000
   gramc = gramc + A_7^k*B_7*B_7'*A_7'^k;
end
rmsy_10 = (trace(C_7*gramc*C_7'))^(1/2) % 0.2189

% part 3: Use (8) experimental pulse response to compute the H2 norm
rmsy_8 = 0;
for k=42:400
   rmsy_8 = rmsy_8 + (norm([y11(k), y12(k); y21(k), y22(k)]))^(2);
end
rmsy_8 = rmsy_8^(1/2) % 0.2091
```

hinfnorm.m

```
% task 6

% part 1: Write a Matlab function that accepts as inputs the state-space matrices of a continuous-time system, upper and lower
% limits for the γ search, and a tolerance, and then returns the H ∞ norm of the system computed to within the specified
% tolerance, and the approximate frequency at which the maximum gain is achieved.

function [hinfnormd, freqcont] = hinfnorm(A, B, C, D, uplim, lolim, tol)
   % calculate the continuous time matrices from the state space matrices
   Ac = -1*inv(eye(length(A))+A)*(eye(length(A))-A)
   Bc = sqrt(2)*inv(eye(length(A))+A)*B
   Cc = sqrt(2)*C*inv(eye(length(A))+A)
   Dc = D - C*inv(eye(length(A))+A)*B
   % call computehinfnormcont
   [hinfnormd, freqcont] = computehinfnormcont(Ac, Bc, Cc, Dc, uplim, lolim, tol)
end

function [hinfnormc, freqcont] = computehinfnormcont(A, B, C, D, uplim, lolim, tol)
   % start with gamma between uplim and lolim
   g = lolim+(uplim-lolim)/2;
   epsilon = 0.0000000001; % checks zero real part of eig(Aclp)

   % apply the bisection procedure over gamma using Aclp(g)
   while uplim - lolim > tol
      % update D(g) and Aclp(g)
      Dg = g^2*eye(length(D)) - D'*D;
      Aclp = [A + B*inv(Dg)*D'*C, -1*B*inv(Dg)*B'; C'*C+C'*D*inv(Dg)*D'*C, -1*A'-C'*D*inv(Dg)*B'];

      % get the eigenvalues of Aclp
      evals = eig(Aclp);

      % check if any evals of Aclp are purely imaginary
      purelyimag = false;
      for i=1:length(evals)
         if abs(real(evals(i))) < epsilon & abs(imag(evals(i))) > epsilon
            real(evals(i));
            freqcont = abs(imag(evals(i)));
            purelyimag = true;
            disp('found purely imag eigval');
            break % breaks the for loop once it finds a purely imaginary eval
         end
      end

      if purelyimag
         % found a purelyimag eigval, increase gamma & lolim and search over upper half
         lolim = g;
```

```
            g = lolim + (uplim-lolim)/2;
            purelyimag = false; % reset purelyimag check
        else
            % didn't find a purelyimag eigval, decrease gamma & uplim and
            % search over lower half
            disp('reducing uplim');
            uplim = g;
            g = lolim + (uplim-lolim)/2;
        end
    end
    eig(Aclp)
    hinfnormc = g
end
```

calchinfnorm.m

```
[hinfnorm7, v] = hinfnorm(A_7, B_7, C_7, zeros(2,2), 10000, 0, 0.000000001)
discretefreq = log((1 + 1j*v)/(1 - 1j*v))/(1j*ts)
```