MECHANICAL AND AEROSPACE ENGINEERING
UNIVERSITY OF CALIFORNIA, LOS ANGELES

Linear Dynamic Systems Course Project

Jiahui Lu

UID:204945099

A COURSE PROJECT REPORT

For

Linear Dynamic Systems (MAE 270A)

Instructure: Prof. Robert M'Closkey

2017.12.7

Content

1 TIME-DOMAIN MODEL IDENTIFICATION VIA HANKEL MATRIX ANALYSIS

The modeling and analysis of a two-input/two-system that produced this input-output data with the sampling frequency of 40 Hz for four channels is given. The state dimension is not known, though, with the objective being the determination of a state-space discrete-time system that can replicate the observed data. Note that we seek an asymptotically stable model because the impulse response data decays to zero. Also note that when using data sets from physical systems that rank is n because inevitably there is some noise in the measurement data and furthermore the system may be nonlinear even though it can be well-approximated with a linear model. Thus, it appears that we need an n-dimensional system to model the data, but a lower rank approximation almost always produces a “better" model. For example, asymptotic stability of the identified model is not explicitly enforced and using high model dimensions can actually produce unstable models. The inputs and system responses are given in Figure 1.

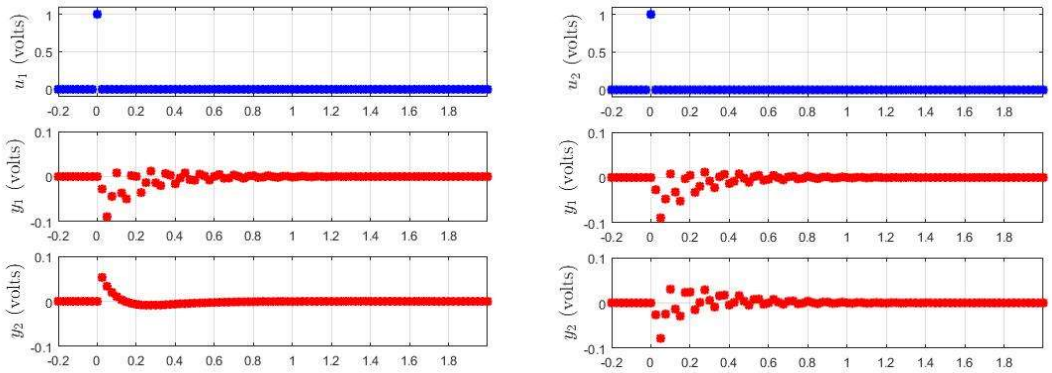

Figure 1: Actual impulse response of a two-input/two-output system

Task #1: Model identification

1.1 Singular Values of Hankel Matrix

Construct H_{100} and graph the singular values. This can be shown in Figure 2. In this plot we have also included singular values for H_{20}, H_{40}, and, H_{80} for comparison. Only the first 40 singular values are shown.

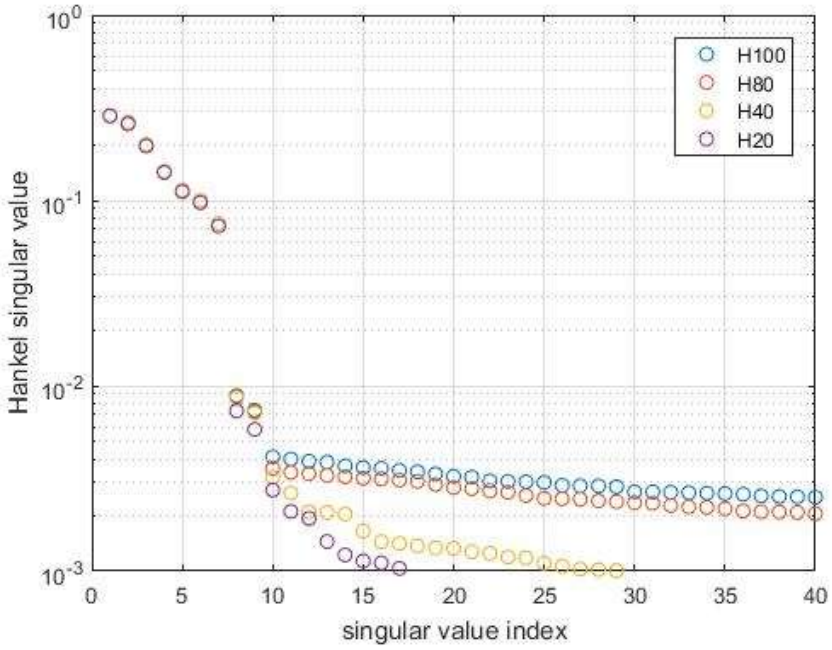

Figure 2: Hankel Singular Value

This plot states that a good initial choice for the state dimension is $n_s = 7$. The maximum absolute eigenvalue of the A matrix is in Table 1, which are all less than one suggesting asymptotically stables for all.

Table 1 Maximum Absolute Value of the Eigen Values for Distinct Dimension

n_s	*Max Norm of Eigenvalues*
6	*0.9526*
7	*0.9144*
10	*0.9141*
20	*0.9975*

1.2 Impulse Response Simulation

The impulse response (in time domain) for each model with different state dimension are show in Figure 3, and it is compared it to the measurement data. In the meanwhile, A, B, C matrices have already been computed for each state dimension of 6, 7, 10 and 20.

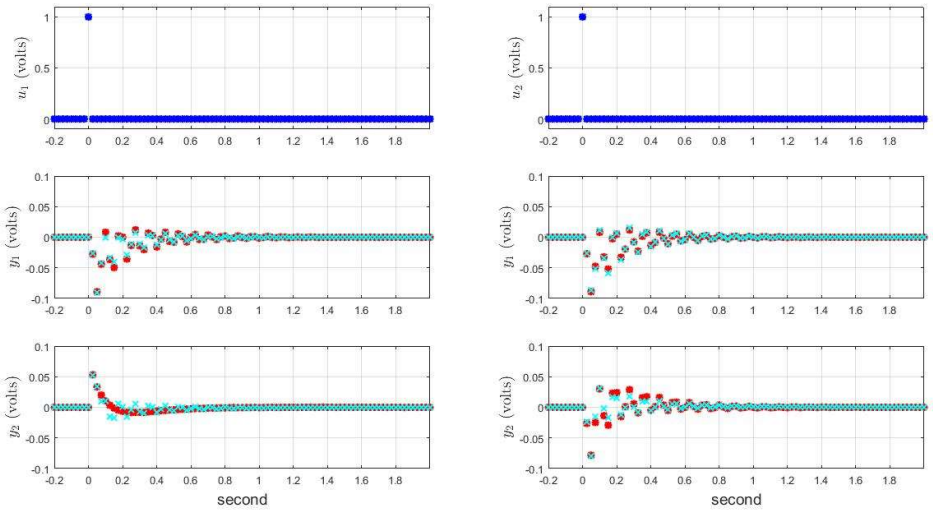

Figure 3 (a): Simulated vs. Actual System Response for Dimension of 6

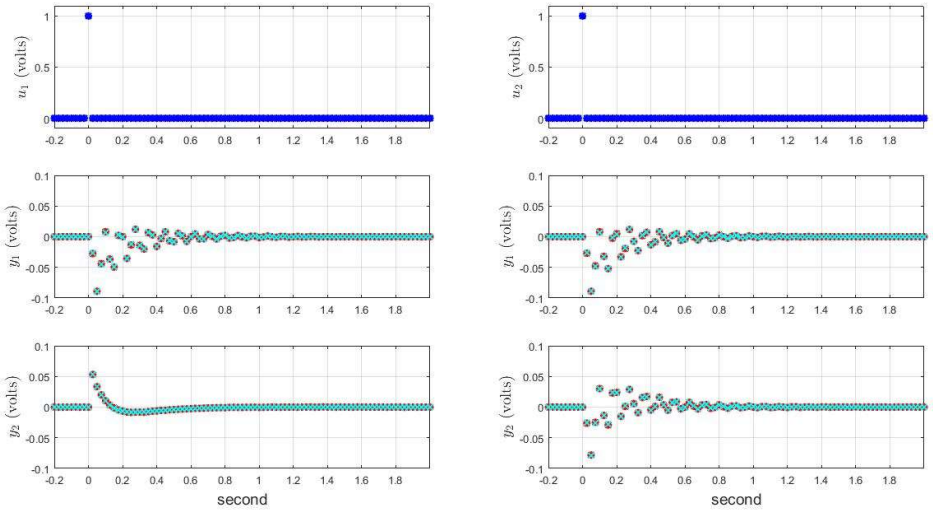

Figure 3 (b): Simulated vs. Actual System Response for Dimension of 7

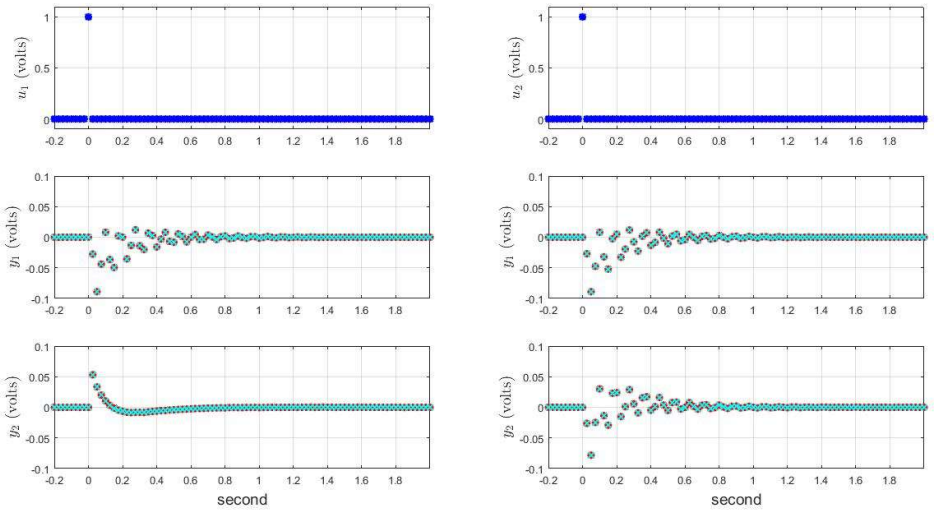

Figure 3 (c): Simulated vs. Actual System Response for Dimension of 10

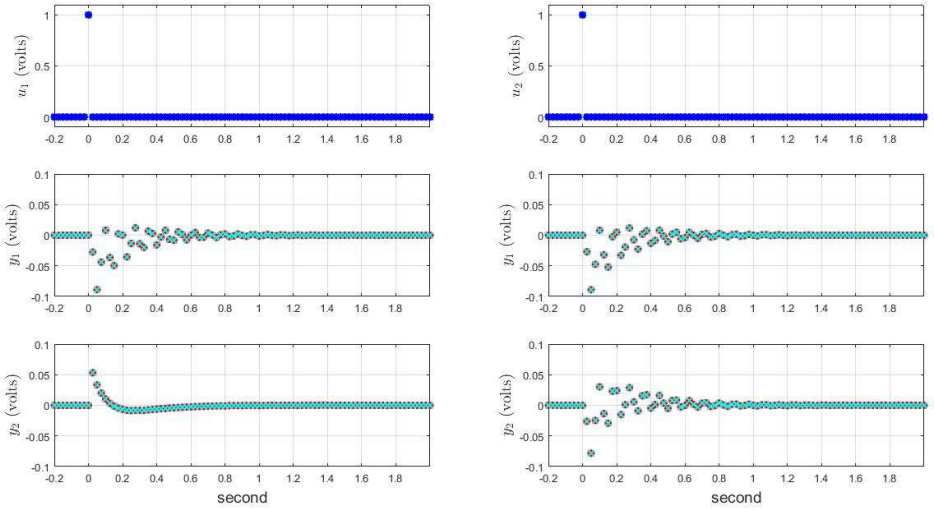

Figure 3 (d): Simulated Vs. Actual System Response for Dimension of 20

As shown above, the dimension with 6 is not able to reproduce perfect impulse response as the actual system, which is inferior. The remaining models are essentially creating responses that are indistinguishable from the impulse response of the actual system.

1.3 and 1.4 Frequency Response

Another way to compare the model to the data is to compare the model frequency response to the empirical frequency response obtained from the pulse response data. The model frequency response is given by

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

Where t_s = 1/40 s is the sampling period and the frequency w is in the interval of [0, w_{nyq}], where w_{nyq} is the Nyquist frequency, which equals to half of the sampling frequency which is w_{nyq} = 20 Hz (125.7 rad/s). Note that when evaluating the formula, it ought to be in units of radians per second. At each frequency, the frequency response is a 2x2 matrix with complex-valued elements. There are a total of eight figures because each scalar input-output channel has a magnitude and phase plot (all magnitudes of the (1,1) channel will be compared in a single figure, all phases of the (1,1) channel will be compared in a single figure, and so forth).

The pulse response data can be used to directly estimate the frequency response without the need for a parametric model. The the Fast Fourier Transform (FFT) with Matlab command "fft" is used. Estimate the empirical frequency response magnitude and phase and overlay these results on the model plots from the previous part. Figure 4 gives the corresponding output together and it can be observed that three of the models match up very well with the empirical frequency response results.

Three out of four channels have resonant frequencies at around 70 rad/s or so and the model with dimension n_s = 6 fails to some extent as for frequency response of the actual data.

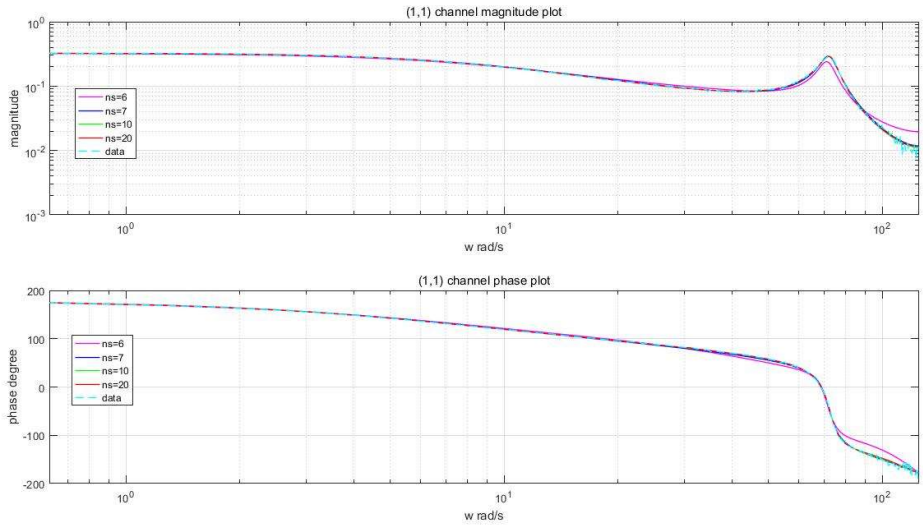

Figure 4 (a): Channel (1,1) Bode plot for Actual System Along with Four Distinct Dimensions

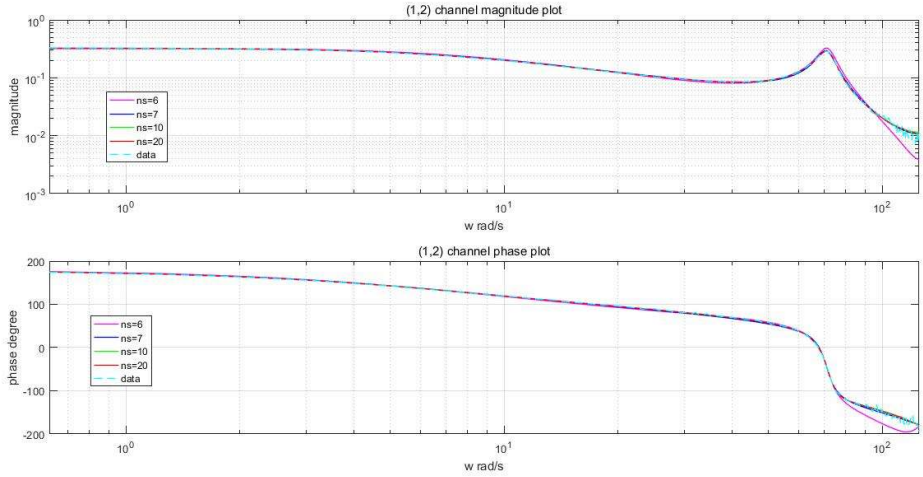

Figure 4 (b): Channel (1,2) Bode plot for Actual System Along with Four Distinct Dimensions

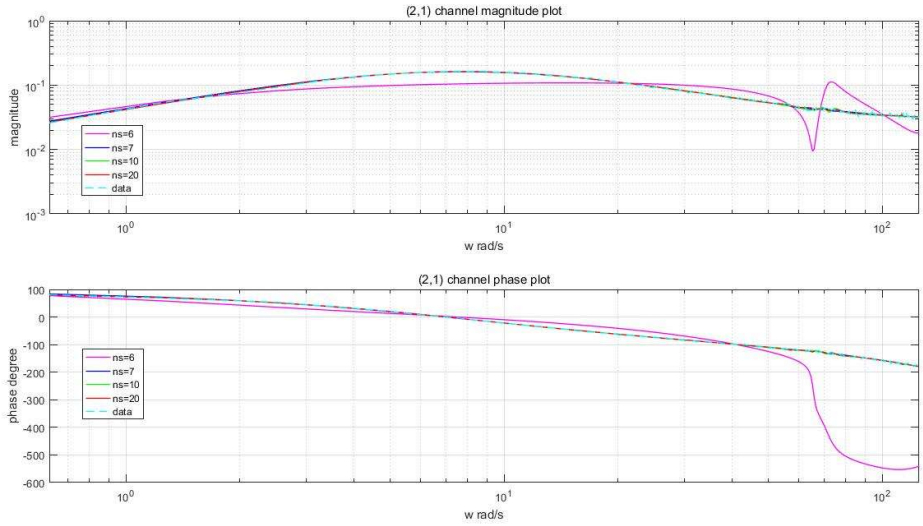

Figure 4 (b): Channel (2,1) Bode plot for Actual System Along with Four Distinct Dimensions

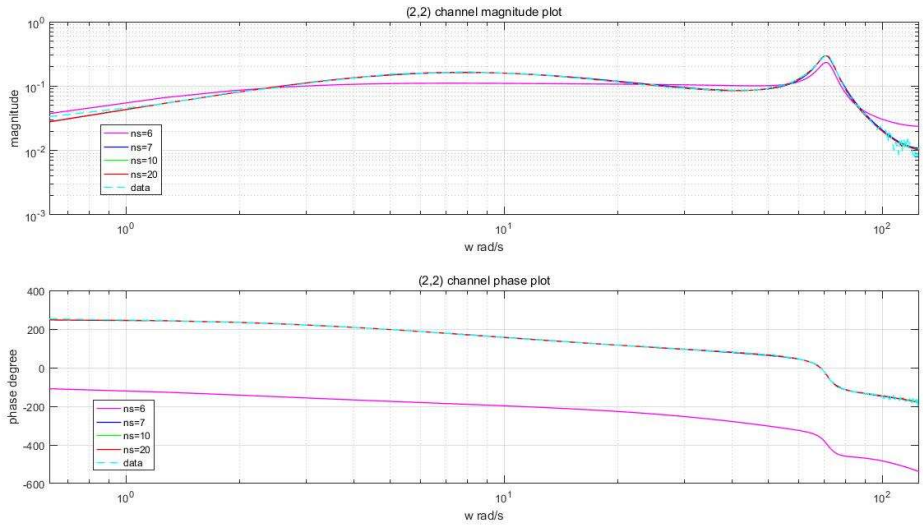

Figure 4 (d): Channel (2,2) Bode plot for Actual System Along with Four Distinct Dimensions

Task 2: Transmission Zeros of The MIMO Model and Zeros of Each Channel

In this part, the transmission zeros of the model will be calculated based on the results of matrix obtained from previous section. As is shown before, the models with state dimension of n_s = 7, 8 provide a rather accurate model of the actual system, it can be discerned from both time and frequency responses.

For calculating the zeros of the system, it can be computed from a generalized eigenvalue problem, which has the relationship:

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

If the system has a transmission zero, there exist non-zero vectors $c \in C^7$ and $w \in C^2$ such that the output remains zero for all samples.

2.1 Transmission Zeros of the Model

It can be shown that there are five finite transmission zeros of the n_s = 7 model, with the remaining four zero given as ‘’inf’’, which can be ignored. The valid zeros are given and sorted from large to small in the Table 2.

Table 2: Transmission zeros of the model *(ns = 7)*

z_p	*Transmission Zeros(ns = 7)*
z_1	*-2.6310*
z_2	*0.9988*
z_3	*-0.6078 + 0.6740i*
z_4	*-0.6078 - 0.6740i*
z_5	*-0.3241*

z_1 is called an unstable zero because the input it generates is unbounded as $|z_1|>1$. By contrast, the rest all other zeros have magnitude being less than one, which are asymptotically stable poles because the input they generate decays to zero.

2.2 Poles and Zeros Map Plotting

This section will plot both of the eigenvalues and zeros on the complex plane. A circle of radius one is drawn and model is asymptotically stable because the eigenvalues lie within the unit circle. It is denoted that eigenvalues with an "x" and the transmission zeros with a "o". Table 3 shows the eigenvalues.

Table 3: *Eigen Values (ns = 7)*

λ_p	*Eigen Values(ns = 7)*
λ_1	*-0.214 + 0.889i*
λ_2	*-0.214 - 0.889i*
λ_3	*-0.188 + 0.892i*
λ_4	*-0.188 - 0.892i*
λ_5	*0.77*
λ_6	*0.826*
λ_7	*0.869*

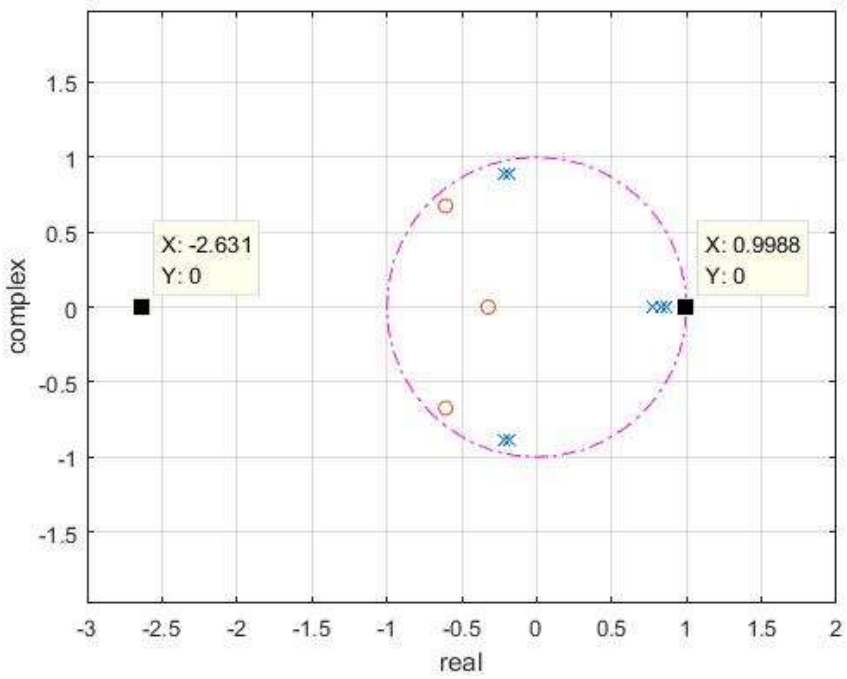

Figure 5: Zero-Poles Map for Dimension 7 Model

2.3 Discrete-time and Continuous-time Converting

A discrete-time eigenvalue can be converted into its equivalent continuous-time eigenvalues according to relationship:

$$\lambda_d = e^{\lambda_c t_s}$$

Discrete-time eigenvalue cannot be the only one since:

$$\lambda_d = e^{\lambda c t_s}$$

$$\lambda_c' = \lambda_c + 2\pi \frac{k}{t_s} j, k = \pm 1, \pm 2, \pm 3...$$

$$e^{\lambda c' t_s} = e^{(\lambda_c + 2\pi \frac{k}{t_s} j) t_s} = e^{(\lambda_c t_s + 2\pi k j)} = e^{\lambda c t_s} = \lambda_d$$

The one with smallest value for the imaginary part is chosen (this choice is justified as long as there is no aliasing of resonant frequencies during the data collection). Table 4 gives all the computed continuous-time eigenvalues.

Table 4 Continuous-Time Eigen Values

λ_p	*Eigen Values(ns = 7)*
λ_1	*-3.7023 +71.1407i*
λ_2	*-3.7023 -71.1407i*
λ_3	*- 3.5797 +72.2809i*
λ_4	*-3.5797 -72.2809i*
λ_5	*-10.4546*
λ_6	*-7.6464*
λ_7	*-5.6165*

Based on our results there are two damped oscillators in model at frequencies of 71.14 Hz and 72.28 Hz, which are consistent with the frequency response plot, which can be seen from the magnitude of frequency response with a peak at frequencies around 70 Hz. It can be deemed to be a feature for second-order-system with damp. As for the reason to be only one peak is that due to zero/poles cancellation and we can see that a pair of zeros are very close to one another pair of oscillatory eigenvalues.

2.4 Separate Single-input/single-output System

Now we treat each channel as a separate single-input/single-output system and graph the eigenvalues and transmission zeros calculated.

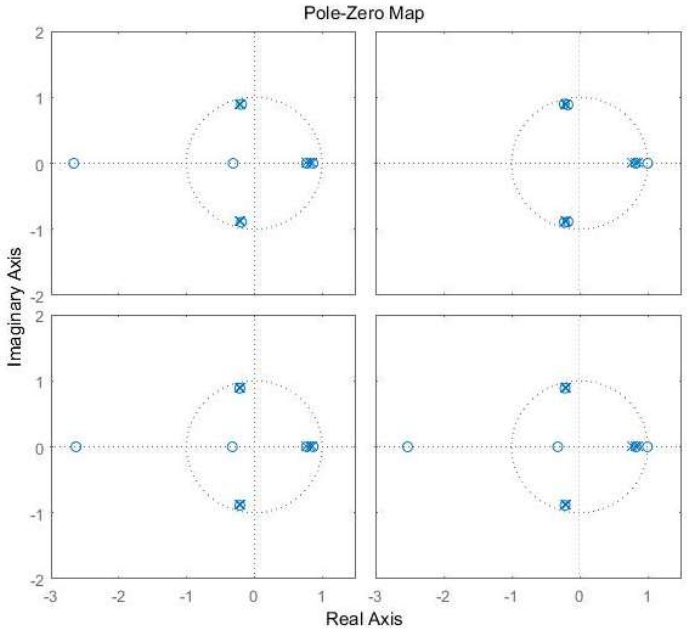

Figure 6: Zero-Poles Map (Each Channel) for Dimension 7 Model

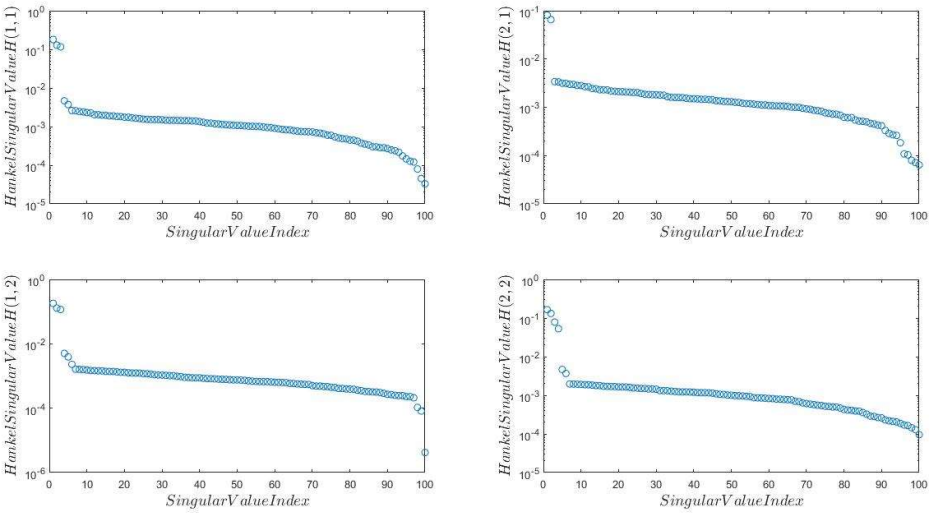

Figure 7: Hankel Singular Value for Each Channel

It can be observed that each channel, considered in isolation, can actually be described with a lower order model whose dimension is consistent with the near pole-zero cancellations that you observe in each channel.

2.5 Dimension Eight Zeros-Poles Map

The eigenvalue and zero of dimension 8 is plotted with each channel. Clearly, this figures are quite similar besides there is an additional pole/zero cancellation in each channel.

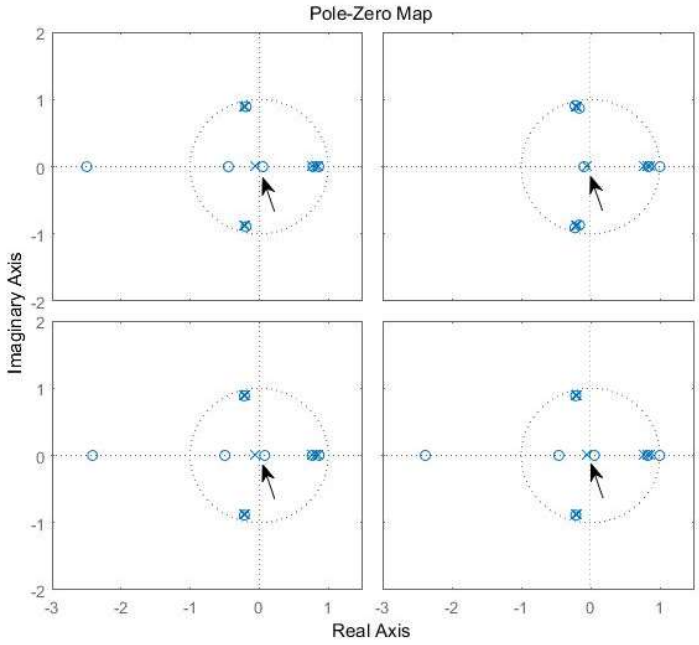

Figure 8: Zero-Poles Map (Each Channel) for Dimension 8 Model

It states that this model is over-parameterized in some sense because its pulse response and frequency response are essentially identical to those associated with former model. Also, this creates essentially a pole-zero cancellation in each channel so that the \extra" state dimension in the $n_s = 8$ case does not have much impact on the input-output properties of the system compared with the lower dimension one.

Task #3: Block Diagram from Analysis of Individual Channels

From the analysis of the two input-two output system, there appear to be two oscillators in the system. Furthermore, there are three poles on the positive real axis (near 0.8). These poles correspond to low-pass filters. The poles of each oscillator are distinct (despite the fact that they are quite close) so give each complex conjugate pair of oscillator poles a distinct label like “OSC1” and “OSC2”. Also label the three distinct real poles as “LP1”, “LP2” and “LP3”.

a block diagram showing how OSC1, OSC2, LP1, LP2 and LP3 are connected (from left to right naming this with ascending number) to form the two input-two output system is given as below:

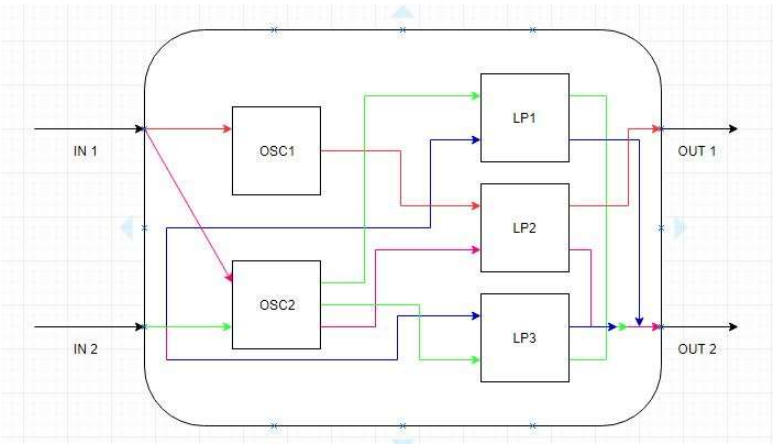

Figure 9: Block Diagram

This map is not unique because these poles cannot be determined respectively for which will stand for these distinct labels.

Finally, some channels have a zero at s = 1 this zero can be combined with one of the low-pass poles to create a high-pass filter. One of the low-pass poles that are not canceled can be paired with this zero to make the high-pass filter (Like LP1 or LP3). It will affect the high-pass filter evident in the frequency response graphs by looking at the figures of y21 and y22. It can be discerned that having zeros and poles constructing high pass filter, the shape of corresponding is the shaping of having high pass filter first then having low pass filter (one possible order). Therefore, they go under the both filters’ influence.

2 CORRELATION FUNCTIONS

Task #4: Impulse Response Identification from White Noise Inputs

A pulse input to a system is not always ideal in order to determine its impulse response because disturbances and measurement noise also contribute to the observed outputs, and because of limitations on how large the input magnitude can be, it may not be possible to increase the pulse height to a level where the system's response is dominated by the effect of the pulse input. Thus, it is advantageous to use inputs which are persistent since it is possible to transfer more energy to the system when limits on the input magnitude are present, as is always the case in any real test environment. A persistent input which is quite useful in this regard is “white noise”. Therefore, in this part, the study of inputs with regarding to the white noises to the system will be conducted as well as study the system with a new method to see the response.

4.1 The Mean of Each Input Sequence

The mean of each input sequence can be obtain by showing the given equations are satisfied:

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

$$R_{uu}[k] = I \leftarrow k = 0$$

$$R_{uu}[k] = 0 \leftarrow k \neq 0$$

This would show that the input sequences are zero mean and unit variance.

We can get the means of input:

$$u_1 = -0.00097$$
$$u_2 = 0.0013$$

Which is about to be zero.

4.2 The Estimates of the Four Entries R_{uu}

The estimates of the four entries of R_{uu} for indices $k \in [-200,\ 200]$ corresponding to the lag factor $\tau \in [-5,\ 5]$ can be graphed as following:

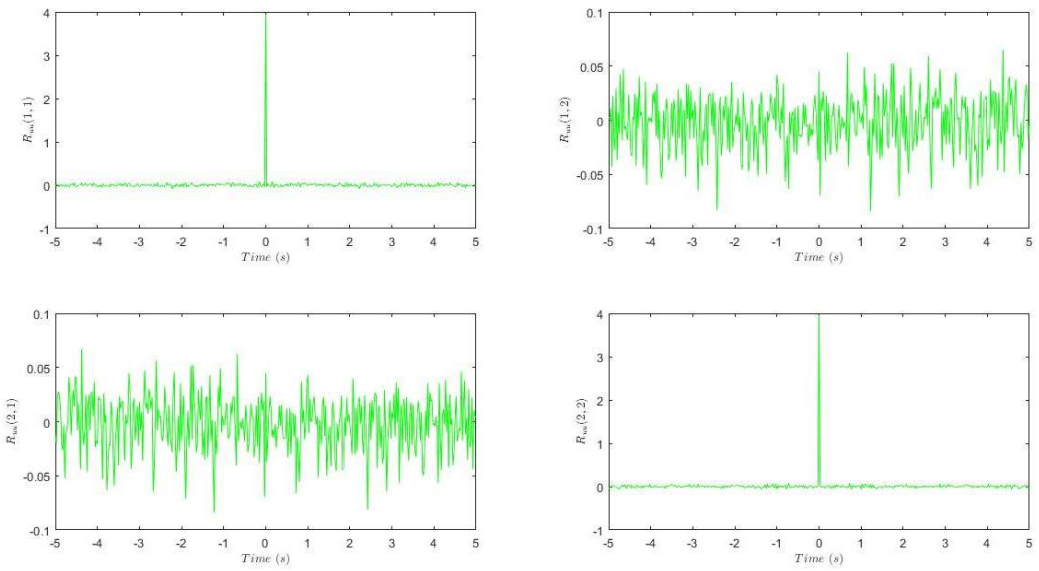

Figure 10: The Estimates of the Four Entries R_{uu}

4.3 Auto-correlation of Input with No Lag

As for auto-correlation of u vector input with no lag, computing $R_{uu}[0]$:

$$R_{uu}[0] = \begin{bmatrix} 3.9203 & 0.0443 \\ 0.0443 & 3.9544 \end{bmatrix}$$

It can be approximately seen as:

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

we can get that it is zero mean. however, the variance is approximately 4.

4.4 Cross-Correlation Impulse Response

Suppose u and y are the input-output sequences associated with an asymptotically stable linear time invariant system, then, u and y are related. The cross-correlation between input u and output y can be denoted as R_{yu}. This result demonstrates that a \virtual" test can be conducted: if input u produces output y, then input R_{uu} produces output R_{yu}.

So here to estimate $R_{yu}[k]$ for $\tau \in [-0.2,\ 2]$ second and then graph the first column of R_{yu} normalized by the variance of the first channel of u as well as graph the second column of R_{yu} normalized by the variance of the second channel of u, which can be shown below:

The results are very close because the system are somehow related for the distinct input conditions. It is verified in the plots that the impulse response obtain from cross-correlation and actual system's response are very close.

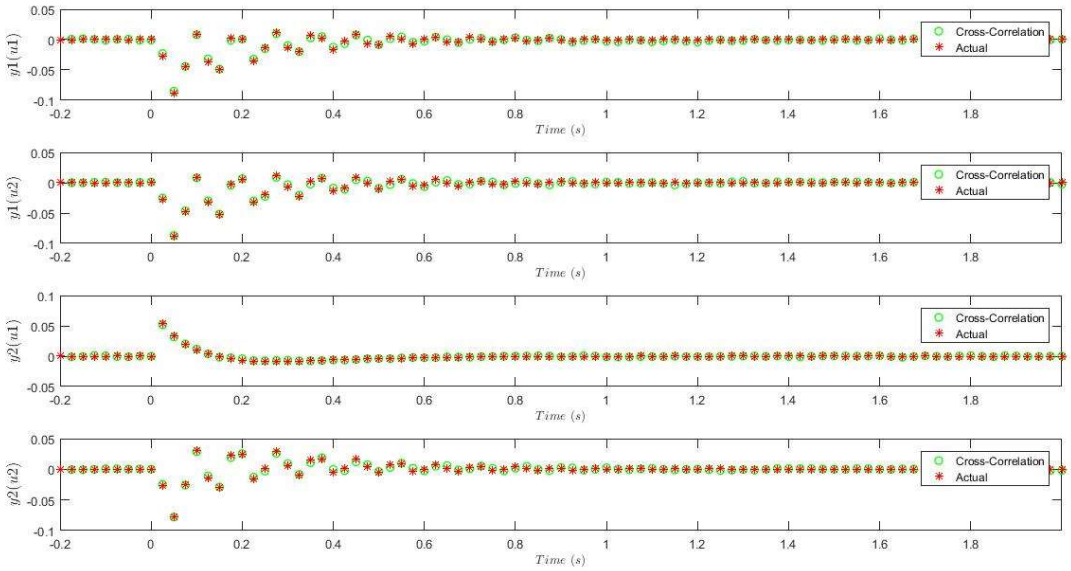

Figure 11: The Impulse Response Obtained from Cross-Correlation and Actual System's Response

3 SYSTEM NORM

3.1 H_2 Norm

Just norms were defined for vectors and matrices, it is possible to define norms for systems and in this case, the input "vectors" are actually drawn from some class of signals. Some of the system norms are induced norms, and others are not. One major difference between signal and system norms compared to norms on finite dimensional vector spaces is that on finite dimensional vector spaces (including matrices) all norms are equivalent in the sense that a vector that is small in one norm will also be small in another norm. This does not generalize to norms on in finite dimensional vector spaces, i.e. linear spaces of signals and systems.

Back to this part, the H_2 norm of this discrete system will be calculated, which will be obtained by a variety of methods. These ways will end up giving the same values. One thing to be noted is that H_2 norm for certain closed-loop input-output channels is minimized so as to be a good objective for controller design.

Task 5: H_2 Norm Analysis of Identified Model

5.1 Root Mean Square Value

Since the input sequence did not have the unit variance, in that case, it is needed to scale input-output data by the factor of 2 since the variance is approximately 4.

It can be given by the RMS value

$$\|v\|_{RMS}^2 = \lim_{p\to\infty} \frac{1}{2p} \sum_{k=-p}^{p} \|v\|_2^2$$

To make the connection between the RMS value of output y and the H_2 norm

$$\|y\|_{RMS}^2 = \lim_{p\to\infty} \frac{1}{2p} \sum_{k=-p}^{p} tr(y_k {y_k}^T) = tr\left(\lim_{p\to\infty} \frac{1}{2p} \sum_{k=-p}^{p} tr(y_k {y_k}^T) \right) = trR_{yy}[0]$$

The Root Mean Square value is calculated for the scaled output data y

$$\|y\|_{RMS} = 0.2237$$

Which is equal to the H_2.

2. Using State-Space Form

Based on the 7-state model derived from the Hankel matrix analysis, this discrete-time system is causal, asymptotically stable, and has the state-space realization $x_{k+1} = Ax_k + Bu_k, y_k = Cx_k$,

$$\|P\|_{H_2}^2 = tr(B^T G_o(\infty)B)$$

$$\|P\|_{H_2}^2 = tr(CG_C(\infty)C^T)$$

So in this case,

$$\|P\|_{H_2}^2 = tr(B^T G_o(\infty)B) = 0.2297$$

$$\|P\|_{H_2}^2 = tr(CG_C(\infty)C^T) = 0.2297$$

3. Using the Experimental Pulse Response Data Only (No Model).

The experimental pulse response with data only and without model can give a modified expression

$$\|P\|_{H_2}^2 = \sum_{k=0}^{\infty} \|h_k\|_F^2, h_k = \textit{discrete-time pulse response}$$

It can be calculated with the input of the actual pulse response matrix and Frobenius norm option “fro” input

$$\|P\|_{H_2}^2 = 0.2298$$

A quick summary is that these values are quite close to each other. The reason why the first one with RMS is a bit smaller is that we do the scaling work, which renders the normalized the input-output data by a factor of 2. That is an error. Anyway, these distinct methods yield a good and constant calculation.

3.2 H_∞ Norm

The H_2 norm of a system is of great importance in control systems design because a typical objective is to design the controller so that the H2 norm of certain closed-loop input-output channels in minimized (say, from some force disturbances that are modeled as broadband white noise inputs to the motion of a vibration-sensitive instrument; in this case it is very natural to consider the H_2 norm of those IO channels as something to be minimized with feedback). The H_2 norm is not an induced norm, though, and does not satisfy the submultiplicative property.

This property makes the H2 norm unsuitable for addressing another critical aspect of control systems design, namely, the robustness of the controller to perturbations of the plant dynamics. The H_∞ norm of an asymptotically stable linear system P is denoted as

$$\|P\|_{H\infty} = \sup \bar{\sigma}(P(j\omega))$$

In this final part, the H_∞ of the 7-state model obtained is calculated for H_∞ with the continuous-time linear dynamic system. As for discrete-time linear dynamic system, the 'equivalent' continuous-time system matrices from those of discrete-time is obtained. Note that the continuous-time system has no direct physical meaning - it is merely constructed so that it's frequency response function at any given has a unique match with the physical discrete-time system's frequency response.

Task #6: H_∞ Norm Analysis of Identified Model

6.1 Continuous-time System Case

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 is given. It will return the H1 norm of the system within the specified tolerance, and the approximate frequency at which the maximum gain is achieved.

Based on the algorithm explained in the project, the closed loop matrix $A_{clp}(\gamma)$ is first calculated at each increment step of and then the eigenvalues of this $A_{clp}(\gamma)$ are calculated. If there exists γ such that:

$\|P\|_{H\infty} \geq \gamma \Leftrightarrow$ there exists a purely imaginary eigenvalue(s) of $A_{clp}(\gamma)$

or $\|P\|_{H\infty} < \gamma \Leftrightarrow$ there exists no purely imaginary eigenvalue(s) of $A_{clp}(\gamma)$

Thus, instead of searching over frequency for the largest singular value of $p(j\omega)$, we search over γ and determine whether $A_{clp}(\gamma)$ has imaginary eigenvalues. A bisection procedure on γ can compute $\|P\|_{H\infty}$ to arbitrary accuracy.

6.2 Discrete-time System Case

A Matlab function that accepts as inputs the state-space matrices of a discrete-time system, upper and lower limits for the γ search, and a tolerance as well as the sample period t_s is given. It will return the H_∞ norm of the system within the specified tolerance, and the approximate frequency at which the maximum gain is achieved.

The frequency response associated with the discrete-time system

$$x_{k+1} = Ax_k + Bu_k, y_k = Cx_k + Du_k$$

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

and the H_∞ norm of the system is defined as $\|P\|_{H\infty} = \max_{\omega \in [0, \omega_{nyq}]} \bar{\sigma}(P(e^{j\omega t_s}))$. Note that the continuous-time system has no direct physical meaning - it is merely constructed so that it's frequency response function at any given has a unique match with the physical discrete-time system's frequency response.

6.3 Outcome of H_∞ Norm

Compute the H_∞ norm of your identified discrete-time model using the Matlab functions

$$\|P\|_{H\infty} = 0.4698$$

The frequency at which it is achieved $\omega_0 = 71.2213$

6.4 Discrete-time Frequency Response

The discrete-time frequency response of the identified model on a frequency grid in the interval $[0, \omega_{nyq}]$ as well as the singular values of the frequency response at each frequency is given in this section. Because the frequency response matrix of the system is a 2x2 matrix, there will be two singular values obtained at each frequency domain. Then it overlay on top of the singular values, the singular values computed from the empirical frequency response data in the graph.

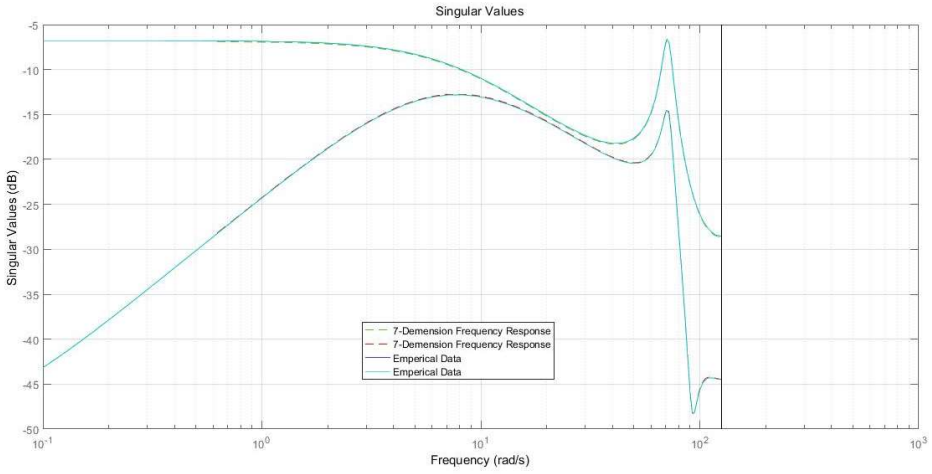

Figure 12: Singular Values of Frequency Response of Empirical and True System

It can be discerned that observe that the model singular values are very close to the empirical singular values and that the H1 norm computed from the model locates the magnitude and corresponding frequency where the maximum singular value achieves its largest value.

Appendix

```
%----------------------------%
%Task One
%----------------------------%
%% Part 0: Plot system responses and input
clear
close all
load u1_impulse.mat
y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs
load u2_impulse.mat
y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;
%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));
%%% rescale IO data so that impulse input has magnitude 1
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;
% figure(1);
% subplot(311)
% plot(t,u1,'b*','LineWidth',2)
% ylabel('$u_1$ (volts)','FontSize',14,'Interpreter','Latex');
% grid on
% axis([-0.2 2 -0.1 1.1])
% subplot(312)
% plot(t,y11,'r*','LineWidth',2)
% ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
% grid on
% axis([-0.2 2 -0.1 0.1])
% subplot(313)
% plot(t,y21,'r*','LineWidth',2)
% ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
% grid on
% axis([-0.2 2 -0.1 0.1])
% figure(2);
% subplot(311)
% plot(t,u2,'b*','LineWidth',2)
% ylabel('$u_2$ (volts)','FontSize',14,'Interpreter','Latex');
% grid on
% axis([-0.2 2 -0.1 1.1])
% subplot(312)
% plot(t,y12,'r*','LineWidth',2)
% ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
% grid on
% axis([-0.2 2 -0.1 0.1])
% subplot(313)
% plot(t,y22,'r*','LineWidth',2)
% ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
```

```
% grid on
% axis([-0.2 2 -0.1 0.1])

%% Part 1: Model identification
H = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H(n,j) = y11(41+i);
        H(n+1,j) = y12(41+i);
        H(n,j+1) = y21(41+i);
        H(n+1,j+1) = y22(41+i);
        j = j+2;
    end
    n = n+2;
end

[U,D,V] = svd(H);
hsv100 = zeros(40,1);
for i = 1:40;
    hsv100(i) = (D(i,i));
end
i = 1:40;
% semilogy(i,hsv100','o');
hold on;
%% ns Problem
ns = 6;
for i = 1:ns
    Si(i,i) = hsv100(i);
end
U = U(:,1:ns);
V = V(:,1:ns);
O = U;
C = Si*V';

H_hat = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H_hat(n,j) = y11(42+i);
        H_hat(n+1,j) = y12(42+i);
        H_hat(n,j+1) = y21(42+i);
        H_hat(n+1,j+1) = y22(42+i);
        j = j+2;
    end
    n = n+2;
end

A = U'*H_hat*V/Si;
Aem = max(abs(eig(A)));
B = C(1:ns,1:2);
C = O(1:2,1:ns);

h = zeros(2,100);
    for k = 1:100
        a = C*A^(k-1)*B;
        h(1,2*k-1) = a(1,1);
        h(2,2*k-1) = a(2,1);
        h(1,2*k) = a(1,2);
        h(2,2*k) = a(2,2);
```

```
        qa(k+41)=a(1,1);
        qb(k+41)=a(2,1);
        qc(k+41)=a(1,2);
        qd(k+41)=a(2,2);
    end

%Plot Response
figure(1);
subplot(321)
plot(t,u1,'b*','LineWidth',2);
ylabel('$u_1$ (volts)','FontSize',14,'Interpreter','Latex');    grid on;
axis([-0.2 2 -0.1 1.1])
subplot(323)
plot(t,y11,'r*','LineWidth',2)
hold on

plot(t(1:141),qa,'cx','LineWidth',1.5);
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex'); grid on
axis([-0.2 2 -0.1 0.1])

subplot(325)
plot(t,y21,'r*','LineWidth',2)
hold on
plot(t(1:141),qc,'cx','LineWidth',1.5);
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
xlabel('second','FontSize',14)
axis([-0.2 2 -0.1 0.1])

subplot(322)
plot(t,u2,'b*','LineWidth',2)
ylabel('$u_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 1.1])
subplot(324)
plot(t,y12,'r*','LineWidth',2)
hold on
plot(t(1:141),qb,'cx','LineWidth',1.5);
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 0.1])
subplot(326)
plot(t,y22,'r*','LineWidth',2)
hold on
plot(t(1:141),qd,'cx','LineWidth',1.5);
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
xlabel('second','FontSize',14)
axis([-0.2 2 -0.1 0.1])
%% task3&4
clear i;

ts = 1/40;
I = eye(ns);
fr_s = [];

for w = 0:0.1*2*pi:20*2*pi
    fr_s = [fr_s (C*inv(exp(w*ts*i)*I-A)*B)];
end
```

```
y_f11 = [];
y_f12 = [];
for j=1:2:401
    y_f11 = [y_f11 fr_s(1,j)];
    y_f12 = [y_f12 fr_s(2,j)];
end

y_f21 = [];
y_f22 = [];
for j=2:2:402
    y_f21 = [y_f21 fr_s(1,j)];
    y_f22 = [y_f22 fr_s(2,j)];
end

y11f = fft(y11)./fft(u1);
N = length(y11f);
om1 = [0:N-1]/(ts*N);

y21f = fft(y21)./fft(u1);
N = length(y21f);
om2 = [0:N-1]/(ts*N);

y12f = fft(y12)./fft(u2);
N = length(y12f);
om3 = [0:N-1]/(ts*N);

y22f = fft(y22)./fft(u2);
N = length(y22f);
om4 = [0:N-1]/(ts*N);

%%
tf = om1*2*pi;
w = 0:0.1*2*pi:20*2*pi;

subplot(421)
loglog(w,abs(y_f11),'LineWidth',1.5);
grid on
hold on
loglog(tf,abs(y11f));xlim([0,125]);
grid on
hold on
title('y_{11} magnitude')

subplot(423)
loglog(w,abs(y_f12),'LineWidth',1.5);
grid on
hold on
loglog(tf,abs(y12f));xlim([0,125]);
grid on
hold on
title('y_{12} magnitude')

subplot(425)
loglog(w,abs(y_f21),'LineWidth',1.5);
grid on
hold on
loglog(tf,abs(y21f));xlim([0,125]);
grid on
hold on
title('y_{21} magnitude')
```

```
subplot(427)
loglog(w,abs(y_f22),'LineWidth',1.5);
grid on
hold on
loglog(tf,abs(y22f));xlim([0,125]);
grid on
hold on
title('y_{22} magnitude')

subplot(422)
semilogx(w,180/pi*phase(y_f11),'LineWidth',1.5);
grid on
hold on
semilogx(tf,180/pi*phase(y11f),'LineWidth',1.5);xlim([0,125]);
grid on
hold on
title('y_{11} Phase Angel')

subplot(424)
semilogx(w,180/pi*phase(y_f12),'LineWidth',1.5);
grid on
hold on
semilogx(tf,180/pi*phase(y12f),'LineWidth',1.5);xlim([0,125]);
grid on
hold on
title('y_{12} Phase Angel')

subplot(426)
semilogx(w,180/pi*phase(y_f21));
grid on
hold on
semilogx(tf,180/pi*phase(y21f),'LineWidth',1.5);xlim([0,125]);
grid on
hold on
title('y_{21} Phase Angel')

subplot(428)
semilogx(w,180/pi*phase(y_f22));
grid on
hold on
semilogx(tf,180/pi*phase(y22f),'LineWidth',1.5);xlim([0,125]);
grid on
hold on
title('y_{22} Phase Angel')


%----------------------------%
%Task Two
%----------------------------%
%% Part 0: Plot system responses and input
clear
close all
load u1_impulse.mat
y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs
load u2_impulse.mat
y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;
%%% remove any offsets in output data using data prior to pulse application
```

```
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));
%%% rescale IO data so that impulse input has magnitude 1
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;
figure(1);
subplot(311)
plot(t,u1,'b*','LineWidth',2)
ylabel('$u_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 1.1])
subplot(312)
plot(t,y11,'r*','LineWidth',2)
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 0.1])
subplot(313)
plot(t,y21,'r*','LineWidth',2)
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
xlabel('second','FontSize',14)
set(gca,'FontSize',14)
axis([-0.2 2 -0.1 0.1])
figure(2);
subplot(311)
plot(t,u2,'b*','LineWidth',2)
ylabel('$u_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 1.1])
subplot(312)
plot(t,y12,'r*','LineWidth',2)
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 0.1])
subplot(313)
plot(t,y22,'r*','LineWidth',2)
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
xlabel('second','FontSize',14)
set(gca,'FontSize',14)
axis([-0.2 2 -0.1 0.1])

%% Part 1: Model identification
H = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H(n,j) = y11(41+i);
        H(n+1,j) = y12(41+i);
        H(n,j+1) = y21(41+i);
        H(n+1,j+1) = y22(41+i);
        j = j+2;
    end
```

```
    n = n+2;
end

[U,D,V] = svd(H);
hsv100 = zeros(40,1);
for i = 1:40;
    hsv100(i) = (D(i,i));
end
i = 1:40;
scatter(i,hsv100');
hold on;
%% ns Problem
ns = 7;
for i = 1:ns
    Si(i,i) = hsv100(i);
end
U = U(:,1:ns);
V = V(:,1:ns);
O = U;
C = Si*V';

H_hat = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H_hat(n,j) = y11(42+i);
        H_hat(n+1,j) = y12(42+i);
        H_hat(n,j+1) = y21(42+i);
        H_hat(n+1,j+1) = y22(42+i);
        j = j+2;
    end
    n = n+2;
end

A = U'*H_hat*V/Si;
Aem = max(abs(eig(A)));
B = C(1:ns,1:2);
C = O(1:2,1:ns);

h = zeros(2,100);
    for k = 1:100
        a = C*A^(k-1)*B;
        h(1,2*k-1) = a(1,1);
        h(2,2*k-1) = a(2,1);
        h(1,2*k) = a(1,2);
        h(2,2*k) = a(2,2);

        qa(k+41)=a(1,1);
        qb(k+41)=a(2,1);
        qc(k+41)=a(1,2);
        qd(k+41)=a(2,2);
    end
% tzero(A,B,C,[]);
%%
D = 0;
iopzmap(idss(A,B,C,D,'Ts',1/40))
% hold on
% plot(real(poles),imag(poles),'ro');
% hold on;
% plot(real(zeros),imag(zeros),'rx');
% axis square;
```

```
% axis([-3 1.5 -2.5 2.5]);
% axis([-1.5 1.5 -3 1.5 ],'square')
% command line:
% 40*log(-0.214 + 0.889i);

% -0.214 + 0.889i
% -0.214 - 0.889i
% -0.188 + 0.892i
% -0.188 - 0.892i
% 0.77
% 0.826
% 0.869

% -2.6310
% 0.9988
% -0.6078 + 0.6740i
% -0.6078 - 0.6740i
% -0.3241
% D = 0;
%
% S2=[ A    B;
%     -C   -D];
% I = [eye(size(A))  zeros(7,2);
%      zeros(2,7) zeros(2,2)];
% [v2 d2] = eig(S2,I);
%
% plot(eig(A) ,'x');
% hold on
% plot(diag(d2),'o')
% xlim([-3 2])
%
% xlabel('real')
% ylabel('complex')
% axis equal
% grid on
%  alpha=0:pi/20:2*pi;    %]
%
%  R=1;                   %
%  x=R*cos(alpha);
%
%  y=R*sin(alpha);

%  plot(x,y,'m-.')

%% H for H1,2,3,4
% H_1_100 = zeros(1,100);
% H_1_n   = zeros(1,100);
% H_2_100 = zeros(1,100);
% H_2_n   = zeros(1,100);
% H_3_100 = zeros(1,100);
% H_3_n   = zeros(1,100);
% H_4_100 = zeros(1,100);
% H_4_n   = zeros(1,100);
%
% for k=1:100
%     H_1_100(k) = y11(42+k);
%     H_1_n(k)      = y11(141+k);
%     H_2_100(k) = y21(42+k);
%     H_2_n(k)      = y21(141+k);
%     H_3_100(k) = y12(42+k);
%     H_3_n(k)      = y12(141+k);
%     H_4_100(k) = y22(42+k);
```

```
%     H_4_n(k)      = y22(141+k);
% end
% H1 = hankel(H_1_100, H_1_n);
% H2 = hankel(H_2_100, H_2_n);
% H3 = hankel(H_3_100, H_3_n);
% H4 = hankel(H_4_100, H_4_n);
%
% [~,S1,~] = svd(H1);
% [~,S2,~] = svd(H2);
% [~,S3,~] = svd(H3);
% [~,S4,~] = svd(H4);
% hsv100 = zeros(100,1);
% for i = 1:100;
%     hsv100_1(i) = (S1(i,i));
%     hsv100_2(i) = (S2(i,i));
%     hsv100_3(i) = (S3(i,i));
%     hsv100_4(i) = (S4(i,i));
% end
% i = 1:100;
% figure(7)
% subplot(221)
% semilogy(i,hsv100_1','o')
% xlabel('$Singular Value Index$','FontSize',14,'Interpreter','Latex');
% ylabel('$Hankel Singular Value H(1,1)$','FontSize',14,'Interpreter','Latex');
% subplot(222)
% semilogy(i,hsv100_2','o')
% xlabel('$Singular Value Index$','FontSize',14,'Interpreter','Latex');
% ylabel('$Hankel Singular Value H(2,1)$','FontSize',14,'Interpreter','Latex');
% subplot(223)
% semilogy(i,hsv100_3','o')
% xlabel('$Singular Value Index$','FontSize',14,'Interpreter','Latex');
% ylabel('$Hankel Singular Value H(1,2)$','FontSize',14,'Interpreter','Latex');
% subplot(224)
% semilogy(i,hsv100_4','o')
% xlabel('$Singular Value Index$','FontSize',14,'Interpreter','Latex');
% ylabel('$Hankel Singular Value H(2,2)$','FontSize',14,'Interpreter','Latex');
% hold on;


%----------------------------%
%Task Four
%----------------------------%
load u_rand.mat
y1 = u_rand.Y(3).Data;
y2 = u_rand.Y(4).Data;
u1 = u_rand.Y(1).Data;
u2 = u_rand.Y(2).Data;
ts = 1/40;
N = length(y1);
t = [0:N-1]*ts - 1;


p = 12000;
u = [u1; u2];
y = [y1; y2];
mean(u1)
mean(u2)

a = zeros(1,401);b = zeros(1,401);c = zeros(1,401);d = zeros(1,401);

for k = 1:401
    sm = zeros(2);
```

```
    for q = 202:2*p-202
          sm = sm + u(:,q+k-201)*(u(:,q))' ;
    end
    matr = 1/(2*p)*sm;
    a(k) = matr(1,1);b(k) = matr(1,2);c(k) = matr(2,1);d(k) = matr(2,2);
end
%%

figure(2222)
x = linspace(-5,5,401);
subplot(221)
plot (x,a,'g','LineWidth',0.5)
ylabel('$R_{uu}(1,1)$','FontSize',10,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
subplot(222)
plot (x,b,'g','LineWidth',0.5)
ylabel('$R_{uu}(1,2)$','FontSize',10,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
subplot(223)
plot (x,c,'g','LineWidth',0.5)
ylabel('$R_{uu}(2,1)$','FontSize',10,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
subplot(224)
plot (x,d,'g','LineWidth',0.5)
ylabel('$R_{uu}(2,2)$','FontSize',10,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');


%% problem 4
a = zeros(1,89);b = zeros(1,89);c = zeros(1,89);d = zeros(1,89);

for k = 1:89
    sm = zeros(2);
    for q = 89:2*p-89
          sm = sm + y(:,q+k-8)*(u(:,q))' ;
    end
    matr = 1/(2*p)*sm;

    a(k) = matr(1,1)/var(u1);b(k) = matr(1,2)/var(u2);c(k) =
matr(2,1)/var(u1);d(k) = matr(2,2)/var(u2);
end
%%

x = linspace(-0.2,2,89);
figure(4444)
subplot(411)
plot (x+0.025,a,'go',t,y11,'r*','LineWidth',1)
ylabel('$y1(u1)$','FontSize',12,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
h41 = legend ('Cross-Correlation','Actual');
set(h41,'FontSize',9);
xlim([-0.2 2])

subplot(413)
plot (x+0.025,c,'go',t,y21,'r*','LineWidth',1)
ylabel('$y2(u1)$','FontSize',12,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
h42 = legend ('Cross-Correlation','Actual');
set(h42,'FontSize',9);
xlim([-0.2 2])
```

```
subplot(412)
plot (x+0.025,b,'go',t,y12,'r*','LineWidth',1)
ylabel('$y1(u2)$','FontSize',12,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
h43 = legend ('Cross-Correlation','Actual');
set(h43,'FontSize',9);
xlim([-0.2 2])

subplot(414)
plot (x+0.025,d,'go',t,y22,'r*','LineWidth',1)
ylabel('$y2(u2)$','FontSize',12,'Interpreter','Latex');
xlabel('$Time$ $(s)$','FontSize',10,'Interpreter','Latex');
h44 = legend ('Cross-Correlation','Actual');
set(h44,'FontSize',9);
xlim([-0.2 2])


%-----------------------------%
%Task Five
%-----------------------------%
load u_rand.mat
y1 = u_rand.Y(3).Data/2;
y2 = u_rand.Y(4).Data/2;
u1 = u_rand.Y(1).Data/2;
u2 = u_rand.Y(2).Data/2;

ts = 1/40;
N = length(y1);
t = [0:N-1]*ts - 1;

p = 12000;
u = [u1; u2];
y = [y1; y2];

a = zeros(1,401);b = zeros(1,401);c = zeros(1,401);d = zeros(1,401);

%%Problem 1
Ruu = [0 0 ; 0 0];
for q = 1:24001
    Ruu = Ruu + y(:,q)*(y(:,q))';
end
y_rms = sqrt(diag((1/(2*p))*Ruu));
RMS = norm(y_rms)

%%problem 2
sys =ss(A,B,C,D,ts);
Gc = gram(sys,'c');
Go = gram(sys,'o');
PH2_1 = norm(sqrt(diag(B'*Go*B)))
PH2_2 = norm(sqrt(diag(C*Gc*C')))

% problem 3
 %experimental norm
 PH2_3  = 0;
for i = 1:401
    n = norm([y11(i) y12(i);y21(i) y22(i)],'fro');
    PH2_3 = PH2_3+n^2;
end
PH2_3   = sqrt(PH2_3);
PH2_3   = norm(PH2_3);
```

```
%----------------------------%
%Task Six
%----------------------------%
%% Part 0: Plot system responses and input
clear
close all
load u1_impulse.mat
y11 = u1_impulse.Y(3).Data;
y21 = u1_impulse.Y(4).Data;
u1 = u1_impulse.Y(1).Data; %%% note that the pulse magnitude is 5
[m,mi] = max(u1>0); %%% find index where pulse occurs
load u2_impulse.mat
y12 = u2_impulse.Y(3).Data;
y22 = u2_impulse.Y(4).Data;
u2 = u2_impulse.Y(2).Data;
%%% remove any offsets in output data using data prior to pulse application
y11 = y11 - mean(y11([1:mi-1]));
y12 = y12 - mean(y12([1:mi-1]));
y21 = y21 - mean(y21([1:mi-1]));
y22 = y22 - mean(y22([1:mi-1]));
%%% rescale IO data so that impulse input has magnitude 1
y11 = y11/max(u1);
y12 = y12/max(u2);
y21 = y21/max(u1);
y22 = y22/max(u2);
u1 = u1/max(u1);
u2 = u2/max(u2);
ts = 1/40; %%%% sample period
N = length(y11); %%%% length of data sets
t = [0:N-1]*ts - 1;
figure(1);
subplot(311)
plot(t,u1,'b*','LineWidth',2)
ylabel('$u_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 1.1])
subplot(312)
plot(t,y11,'r*','LineWidth',2)
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 0.1])
subplot(313)
plot(t,y21,'r*','LineWidth',2)
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
xlabel('second','FontSize',14)
set(gca,'FontSize',14)
axis([-0.2 2 -0.1 0.1])
figure(2);
subplot(311)
plot(t,u2,'b*','LineWidth',2)
ylabel('$u_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 1.1])
subplot(312)
plot(t,y12,'r*','LineWidth',2)
ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
axis([-0.2 2 -0.1 0.1])
subplot(313)
plot(t,y22,'r*','LineWidth',2)
ylabel('$y_2$ (volts)','FontSize',14,'Interpreter','Latex');
grid on
```

```
xlabel('second','FontSize',14)
set(gca,'FontSize',14)
axis([-0.2 2 -0.1 0.1])

%% Part 1: Model identification
H = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H(n,j) = y11(41+i);
        H(n+1,j) = y12(41+i);
        H(n,j+1) = y21(41+i);
        H(n+1,j+1) = y22(41+i);
        j = j+2;
    end
    n = n+2;
end

[U,D,V] = svd(H);
hsv100 = zeros(40,1);
for i = 1:40;
    hsv100(i) = (D(i,i));
end
i = 1:40;
scatter(i,hsv100');
hold on;
%% ns Problem
ns = 7;
for i = 1:ns
    Si(i,i) = hsv100(i);
end
U = U(:,1:ns);
V = V(:,1:ns);
O = U;
C = Si*V';

H_hat = zeros(200,200);
n = 1;
for k=1:100
    j = 1;
    for i = k:k+99
        H_hat(n,j) = y11(42+i);
        H_hat(n+1,j) = y12(42+i);
        H_hat(n,j+1) = y21(42+i);
        H_hat(n+1,j+1) = y22(42+i);
        j = j+2;
    end
    n = n+2;
end

A = U'*H_hat*V/Si;
Aem = max(abs(eig(A)));
B = C(1:ns,1:2);
C = O(1:2,1:ns);

h = zeros(2,100);
    for k = 1:100
        a = C*A^(k-1)*B;
        h(1,2*k-1) = a(1,1);
        h(2,2*k-1) = a(2,1);
        h(1,2*k) = a(1,2);
```

```
        h(2,2*k) = a(2,2);

        qa(k+41)=a(1,1);
        qb(k+41)=a(2,1);
        qc(k+41)=a(1,2);
        qd(k+41)=a(2,2);
    end

% Define frequency response function
%------function continues-Time------

% function out = hinf61(sys,ttol,~)
%
% H = 0;
% I = eye(length(D));
% ww = 0; %number of lo with good
% q = 0;
% N = 1;
% Nmax = 10000;
% zer = 1e-5;
% while N < Nmax
%     c = (ll+up)/2
%     Dl = (c)^2*I-D'*D;
%     a = A+B*inv(Dl)*D'*C;
%     b = -B*inv(Dl)*B';
%     c = C'*C+C'*D*inv(Dl)*D'*C;
%     d = -A'-C'*D*inv(Dl)*B';
%     Aclp = [a b; c d];
%     reale = real(eig(Aclp));
%     image = imag(eig(Aclp));
%
%     for ii = (1: length(real(eig(Aclp))))
%         if abs(reale(ii)) < zer && image(ii) > zer || (b-a)/2 < tolerance
%             display(gam)
%             return
%         end
%     end
%     N = N+1;
%
%     ll = c;
% end
% Ga = unique(H);
% length(Ga)
% P       =C*inv((1j*om*I)-A)*B+D;
% step    = 0.1 ; %step size

% for lo      = [ll:step:ul]
% Dl = lo^2*I-D'D;
% a  = A+B*inv(Dl)*D'*C;
% b  = -B*inv(Dl)*B';
% c  = C'*C+C'*D*inv(Dl)*D'*C;
% d  = -A'-C'*D*inv(Dl)*B';
%
% Aclp    = [a b; c d];
%
% end
% Ac = -inv(I + A)*(I-A)
% Bc = sqrt(2)*inv(I+A)*B
% Cc = sqrt(2)*inv(I+A)
% Dc = D - C*inv(I+A)*B
```

```
% narginchk(1,3);
% ni = nargin;
%
% if (ni<2 || isempty(ttol))
%     ttol = 1e-3;
% end
%
%
% % SYSTEM matrices
% [a,b,c,d] = unpck(sys);
% sys=ss(a,b,c,d);
%     if max(real(eig(a)))< 0 % stable:compute gain
%         [f,w] = getPeakGain(sys,ttol);
%         out = [f f*(1+ttol) w];
%     end
%


%------function Discrete-Time------

% function out = hinf62(sys,ttol,~)
%
% narginchk(1,3);
% ni = nargin;
%
% if (ni<2 || isempty(ttol))
%     ttol = 1e-3;
% end
%
%
% % SYSTEM matrices
% [a,b,c,d] = unpck(sys);
% sys=ss(a,b,c,d);
%     if max(real(eig(a)))< 0 % stable:compute gain
%         [f,w] = getPeakGain(sys,ttol);
%         out = [f f*(1+ttol) w];
%     end
%

%%

clear i;

ts = 1/40;
I = eye(ns);
fr_s = [];

for w = 0:0.1*2*pi:20*2*pi
    fr_s = [fr_s (C*inv(exp(w*ts*i)*I-A)*B)];
end

y_f11 = [];
y_f12 = [];
for j=1:2:401
    y_f11 = [y_f11 fr_s(1,j)];
    y_f12 = [y_f12 fr_s(2,j)];
end

y_f21 = [];
y_f22 = [];
for j=2:2:402
```

```
    y_f21 = [y_f21 fr_s(1,j)];
    y_f22 = [y_f22 fr_s(2,j)];
end

Hsv = zeros(2,201);
for n = 1:2:401
    Hsv(:,0.5*(n-1)+1) = svd(fr_s(:,n:n+1));
end
%%
w = 0:0.1*2*pi:20*2*pi;
semilogx(w,20*log10(Hsv(1,:)),w,20*log10(Hsv(2,:)))
% hold on
% sigma(ss(A,B,C,0,1/40))
```