MAE270A Final Project

Helene Levy and Aileen Villalpando

December 11, 2020

1 Abstract

The goal of this project is to identify and verify the minimal state-space model for a two-input, two-output system. The initial information includes recorded pulse response data as well as white noise response data. We determined the physical components of the system with eigenvalue and transmission zero analysis. We verified the model using a norm analysis, which additionally provided insight on the robustness of the derived model. The minimal model has state dimension 7, and the physical system is composed of two oscillators, two low-pass filters, and one high-pass filter. We used a norm analysis of both the $\mathcal{H}_2$ norm and the $\mathcal{H}_\infty$ norm of to cross-validate the derived model and quantify its robustness.

2 Introduction

A multi-input/multi-output, discrete-time linear system with m outputs and q inputs has the form

$$\begin{aligned} \mathbf{x_{k+1}} &= A\mathbf{x_k} + B\mathbf{u_k} \\ \mathbf{y_k} &= C\mathbf{x_k} \end{aligned}$$

where the state dimension is n_s such that $A \in \mathbb{R}^{n_s \times n_s}$, $B \in \mathbb{R}^{n_s \times q}$ and $C \in \mathbb{R}^{m \times n_s}$.

The state-space model and state dimension of the system analyzed in this report are not known. It is our intent to (a) identify the state-space model, (b) identify the components of the system, (c) verify the model responds similarly to the system, (d) quantify the similarity of the model to the system, and (e) test the robustness of the model as a proxy for the system.

To begin, we knew that the system is a two-input, two-output system. The experimental data that was provided contains the outputs for two unit impulse responses, recorded at 40 Hz sampling frequency for approximately 10 seconds after the impulse. This data confirmed that there are two output channels that respond to inputs from two channels, and as such

we know $m = 2$ and $q = 2$. Given this information, we began our analysis.

3 Results

3.1 Task 1: Model Identification

3.1.1 A Preliminary Analysis of H_{100}

The discrete-time analog of the impulse response of a continuous-time system is described by these equations:

$$\begin{aligned} h_0 &= 0, \\ h_k &= CA^{k-1}B, \quad k = 1, 2, 3, ... \end{aligned}$$

These h_k elements can be organized into a Hankel matrix, H_n, which has the following antidiagonal construction:

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

The discretized pulse responses, h_ks, are $m \times q$ blocks. By definition, the elements of the Hankel matrix can be computed with the products of the system matrices A, B, and C.

These products are equivalently described as the output values, y_k, corresponding to a unit impulse u_k. In fact, the Hankel matrix derives its structure from its definition as the

product of the observability matrix $\mathcal{O}$, and controllability matrix $\mathcal{C}$,

$$\mathcal{O}_n = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \in \mathbb{R}^{mn \times n_s} \tag{2}$$

$$\mathcal{C}_n = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix} \in \mathbb{R}^{n_s \times nq} \tag{3}$$

which is the factorization of the general impulse response equation, $h_k = CA^{k-1}B$.

The data from this two-input, two-output system response yields 2×2 matrices where the elements of h_k blocks are the concatenated responses of the two output channels given by

$$h_k = \begin{bmatrix} \mathbf{y_1}(t_i) & \mathbf{y_2}(t_i) \end{bmatrix} t_i \in [0, t] \tag{4}$$

where y_1 represents the output of y in response to u_1 and y_2 represents the output of y in response to u_2 at some time t_i.

This Hankel matrix is useful for our purposes because it can be used to solve for the state-space model. In this particular case, H recovers the B and C matrices. The dimension, n, of H is 100. H_{100} for this two-input, two-output system is as follows

$$H_{100} = \begin{bmatrix} -0.0276 & -0.0270 & -0.0890 & -0.0889 & -0.0439 & -0.0475 & ... & -0.0002 \\ 0.0532 & -0.0262 & 0.0334 & -0.0783 & 0.0200 & -0.0247 & ... & 0.0001 \\ -0.0890 & -0.0889 & -0.0439 & -0.0475 & ... & ... & ... & -0.0001 \\ 0.0334 & -0.0783 & 0.0200 & -0.0247 & ... & ... & ... & 0.0001 \\ -0.0439 & -0.0475 & ... & ... & ... & ... & ... & \vdots \\ 0.0200 & -0.0247 & ... & ... & ... & ... & ... & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.0002 & 0.0001 & -0.0001 & 0.0001 & ... & ... & ... & ... \end{bmatrix} \in \mathbb{R}^{200 \times 200} \tag{5}$$

Although B and C can be extracted from H_{100}, $\tilde{H}_{100}$ is necessary to recover A, where

$$\tilde{H}_n = \begin{bmatrix} h_2 & h_3 & h_4 & ... & h_{n+1} \\ h_3 & h_4 & h_5 & ... & h_{n+2} \\ h_4 & h_5 & h_6 & ... & h_{n+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n+1} & h_{n+2} & h_{n+3} & ... & h_{2n} \end{bmatrix} = \mathcal{O}_n A \mathcal{C}_n \tag{6}$$

A is the result of the product of the left inverse of $\mathcal{O}^{\dagger}_{100}$, $\tilde{H}_{100}$, and ${\mathcal{C}^{\dagger}}_{100}$.

The first seven singular values of H_{100} make the greatest contributions to H_{100} as shown in Figure 1. After this point, the values decay rapidly. This demonstrates that a reasonable state space dimensionality is 7, which is the index of the last Hankel singular value that makes a significant contribution. State-space models were also calculated for $n_s \in \{6, 7, 8, 10, 20\}$. Prior to making any further analyses, the asymptotic stability of each model A is calculated below:

```
stab_checkA6 = 0.9526
stab_checkA7 = 0.9144
stab_checkA8 = 0.9137
stab_checkA10 = 0.9141
stab_checkA20 = 0.9975
```

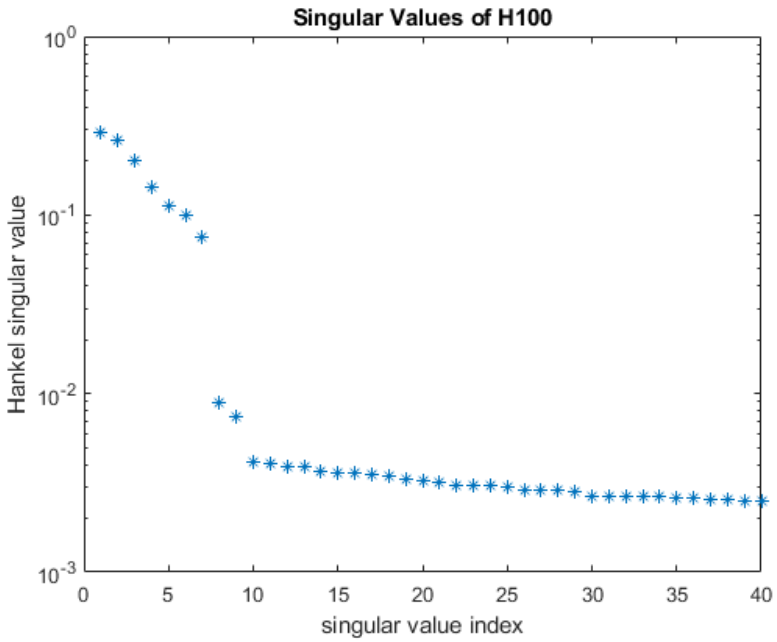


Figure 1: Singular Values of H_{100}

3.1.2 Visual Verification Method 1: Simulation of the Impulse Response with Models Derived from H_{100}

We simulated the impulse response for each model using the state-space definition, which we then compared to the measurement data. By visual analysis, the majority of the reproductions performed well, except for the $n_s = 6$ model (Figures 2 and 3). Models $n_s = \{7, 10, 40\}$ performed similarly (Figures 4, 5, 6, 7, 8, and 9).

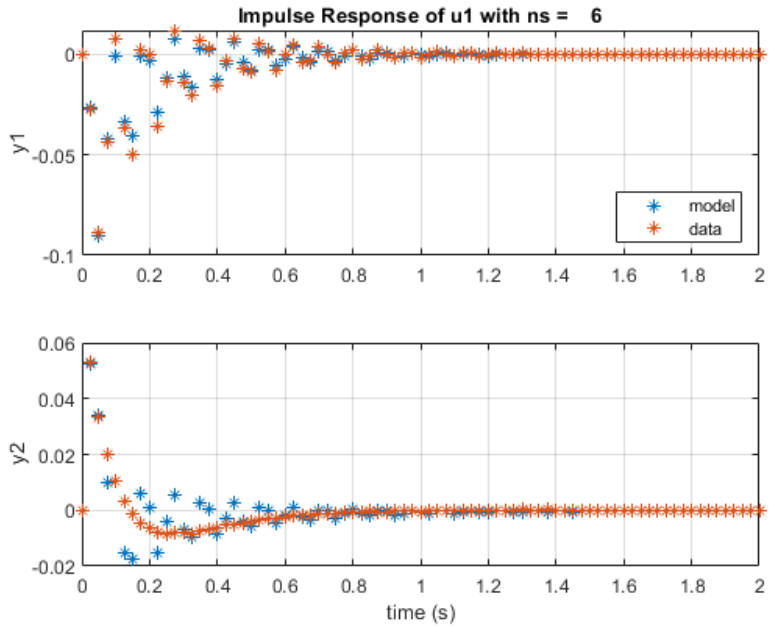


Figure 2: Impulse Response of u_1 to $n_s = 6$ Model Reconstruction

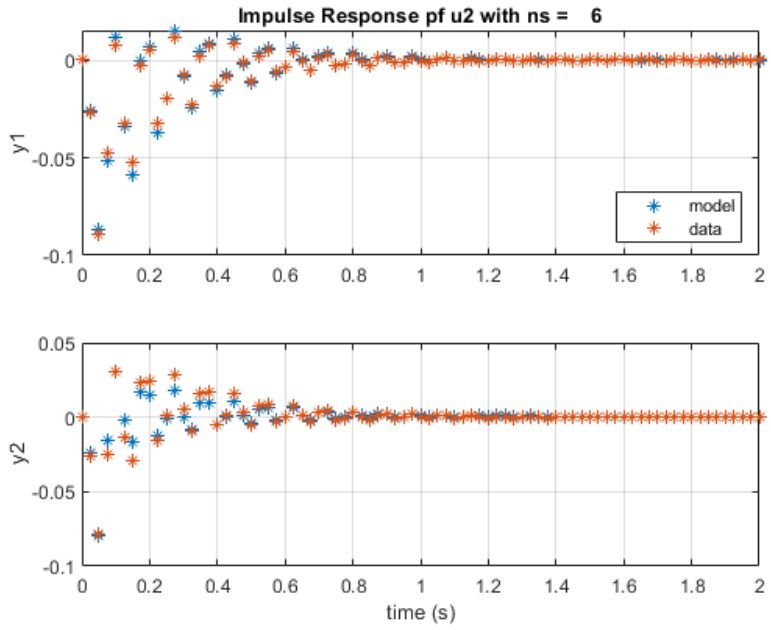


Figure 3: Impulse Response of u_2 to $n_s = 6$ Model Reconstruction

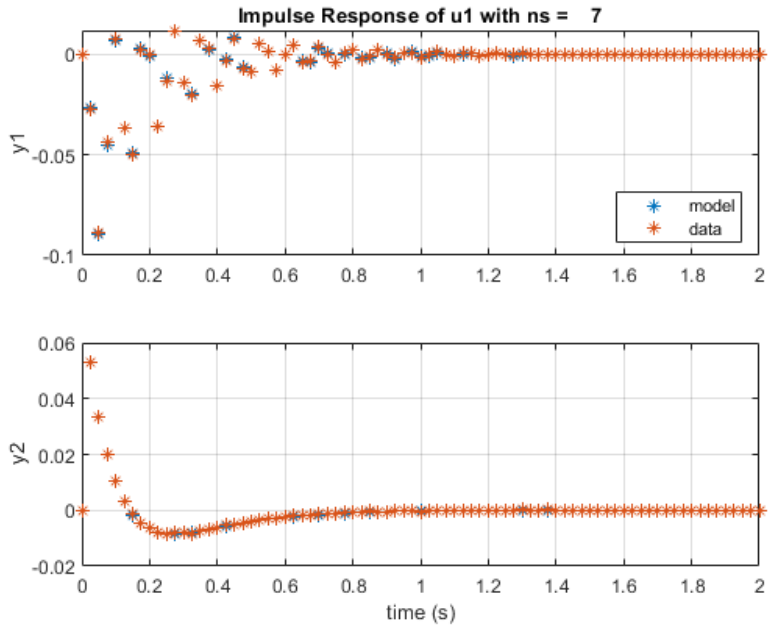


Figure 4: Impulse Response of u_1 to $n_s = 7$ Model Reconstruction

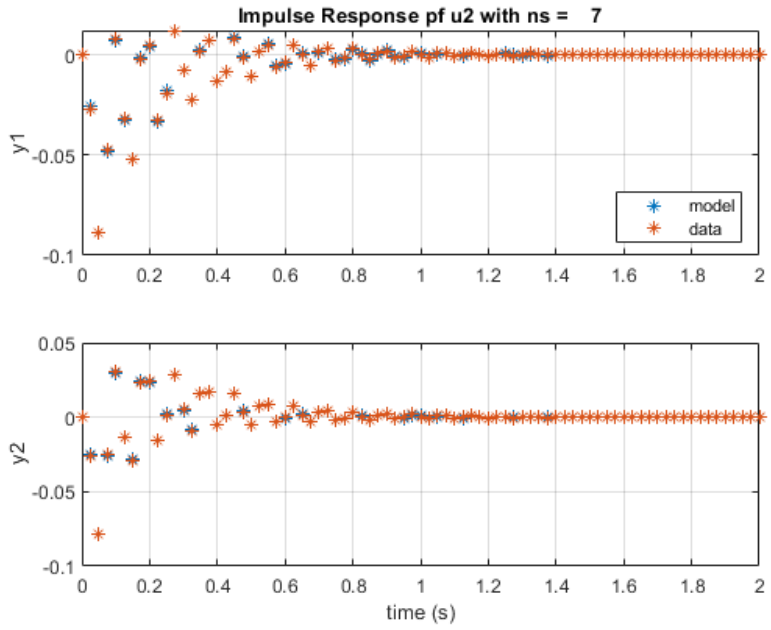


Figure 5: Impulse Response of u_2 to $n_s = 7$ Model Reconstruction

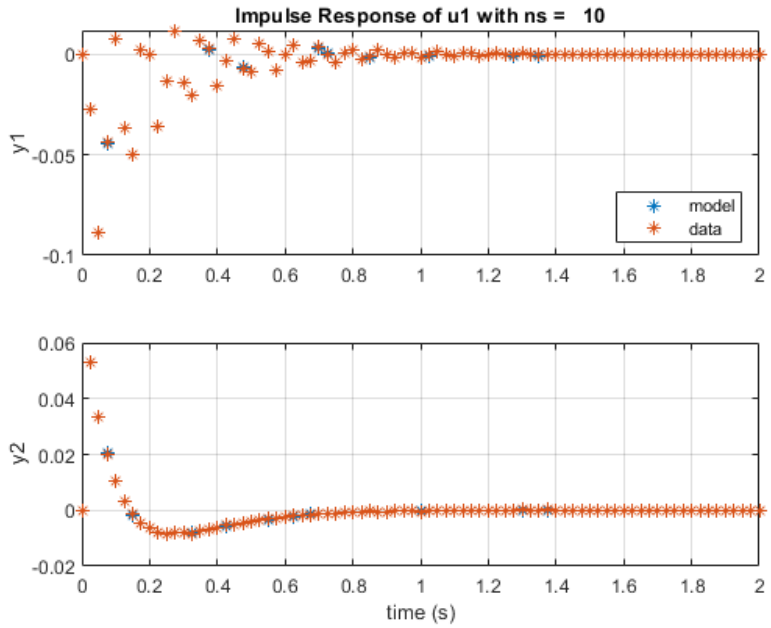


Figure 6: Impulse Response of u_1 to $n_s = 10$ Model Reconstruction

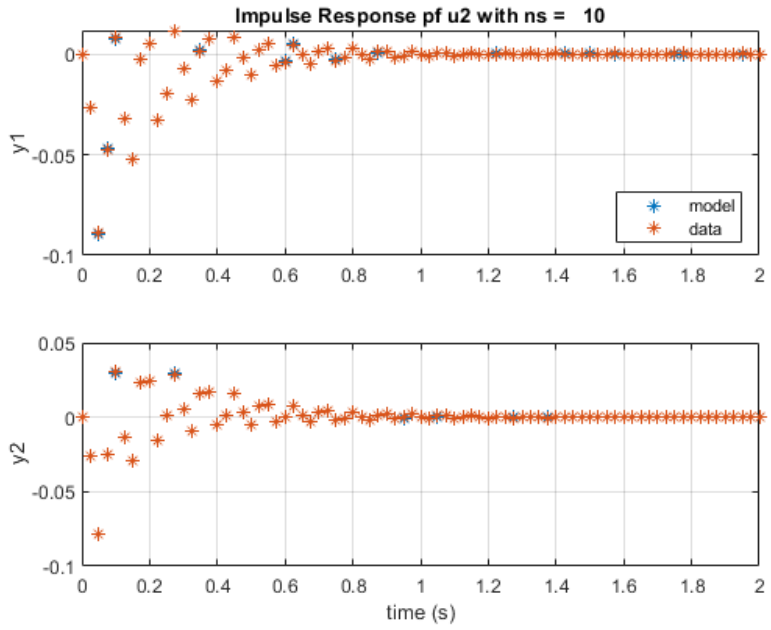


Figure 7: Impulse Response of u_2 to $n_s = 10$ Model Reconstruction

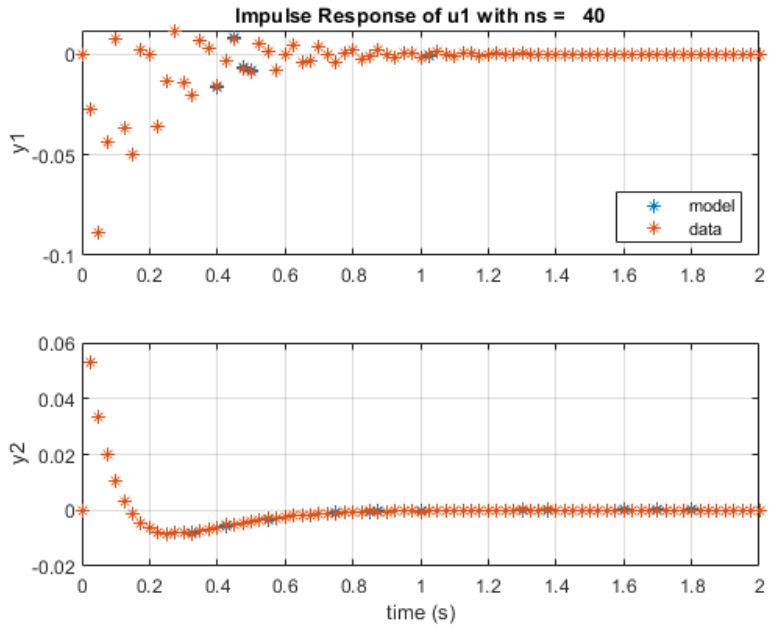


Figure 8: Impulse Response of u_1 to $n_s = 40$ Model Reconstruction

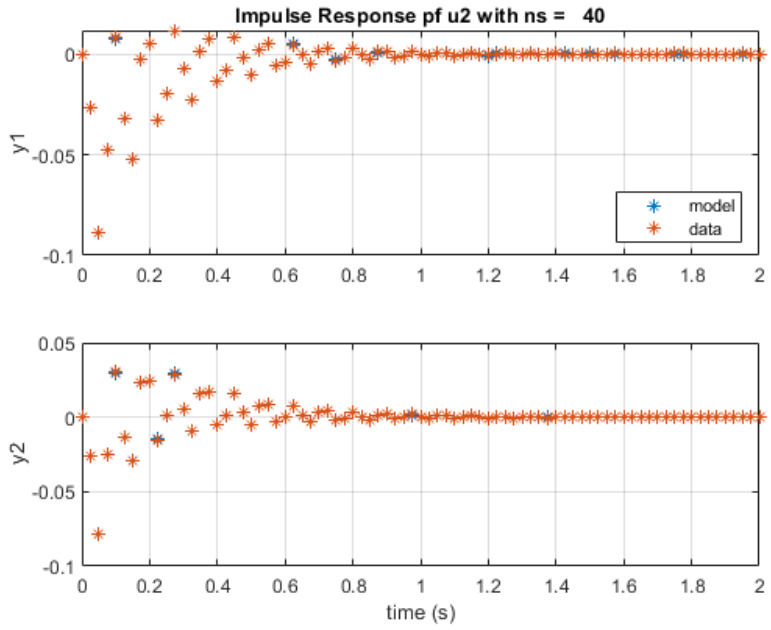


Figure 9: Impulse Response of u_2 to $n_s = 40$ Model Reconstruction

3.1.3 Visual Verification Method 2: Simulation of the Frequency Response with Models Derived from H_{100}

The performance of the model can also be simulated with the frequency response, defined as

$$C(e^{jwt_s}I - A)^{-1}B + D, t_s = \frac{1}{40} \tag{7}$$

where D is a matrix of zeros. These results are plotted as magnitude and phase plots (Figures 10, 11, 12, and 13). In each of these plots, the performance of the $n_s = 6$ model deviates most significantly from the empirical data and all other models (Figures 10,11, 12,13).

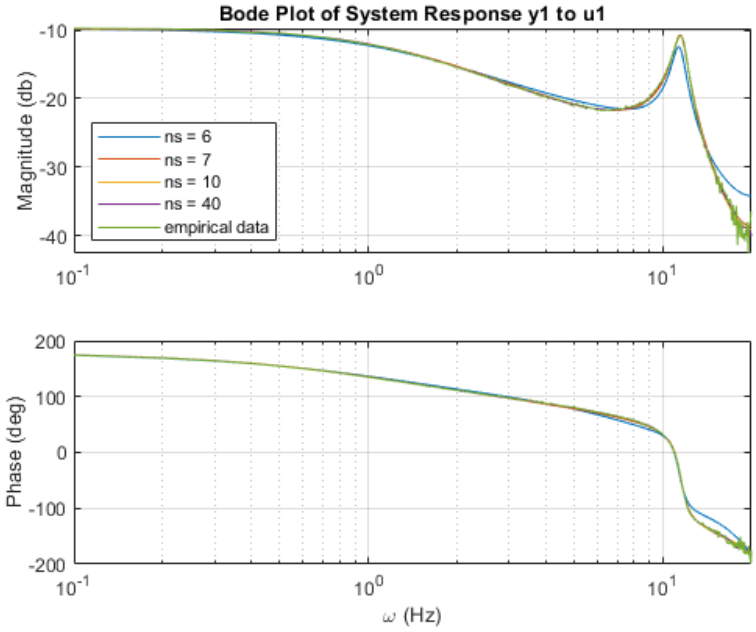


Figure 10: Frequency Response of the (1,1) Channel

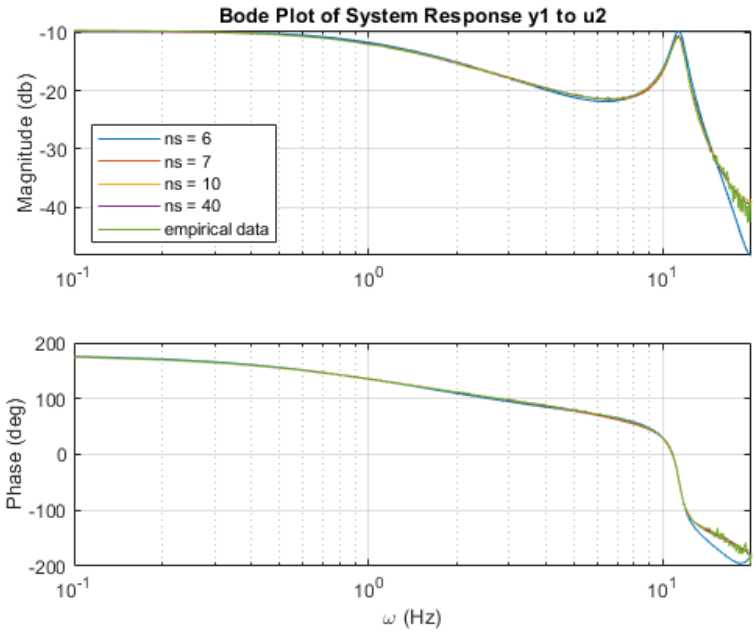


Figure 11: Frequency Response of the (1,2) Channel

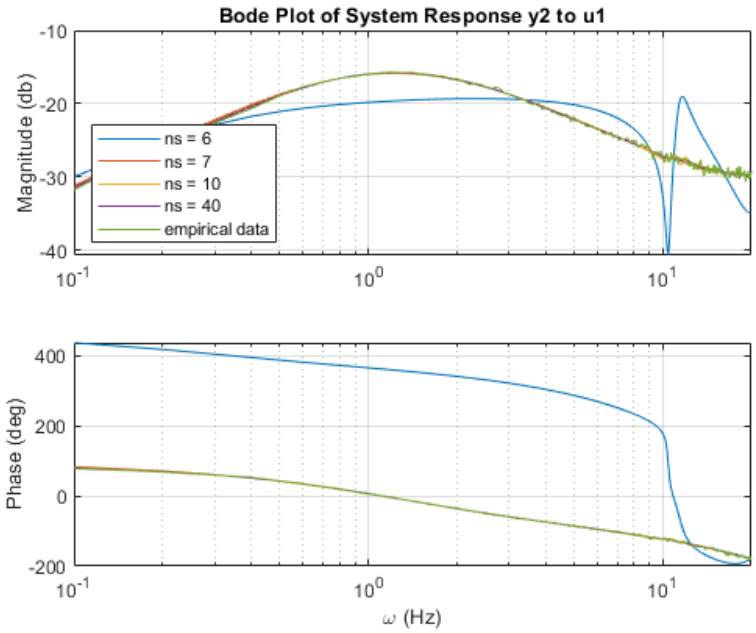


Figure 12: Frequency Response of the (2,1) Channel

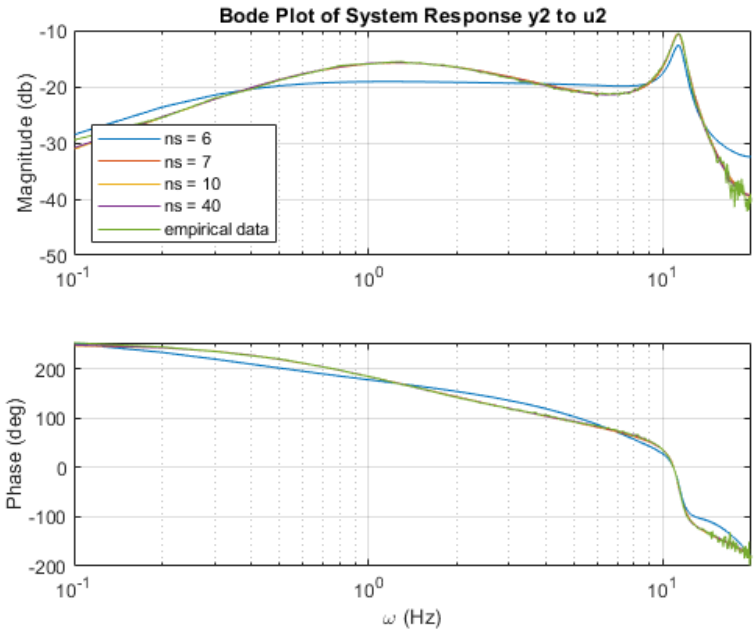


Figure 13: Frequency Response of the (2,2) Channel

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

The eigenvalues and transmission zeros were plotted for the $n_s = 7$ model, which is the model of lowest state that adequately replicates the impulse response and frequency response of the data. We used the location of the eigenvalues and zeros to determine the physical components of the system.

The zeros of the system were determined by the generalized eigenvalue problem:

$$\begin{bmatrix} A & B \\ -C & -D \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = z \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} \tag{8}$$

Looking at the poles in Figure 14, one can see the two-input, two-output system consists of two oscillators, and three low-pass filters. The discrete eigenvalues all lie within the unit circle, indicating asymptotic stability.

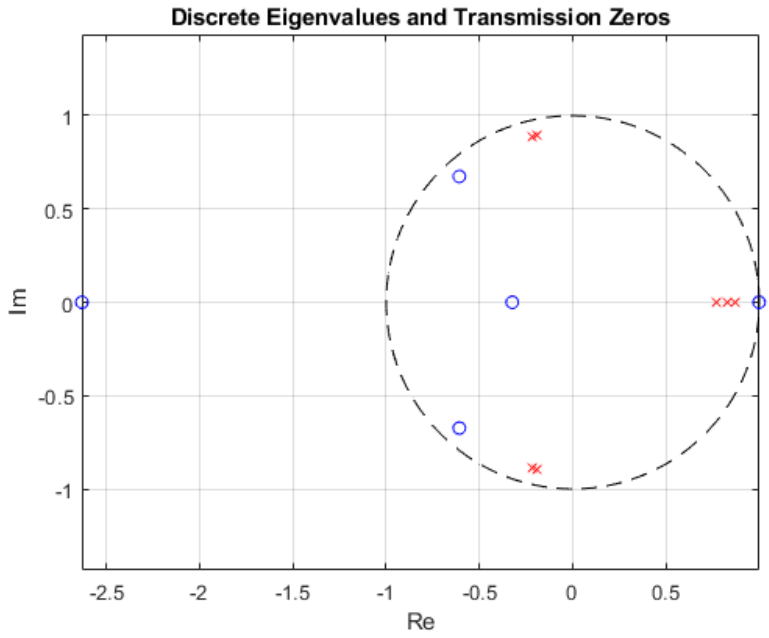


Figure 14: MIMO Discrete Eigenvalues and Transmission Zeros $n_s = 7$

There are five *finite* transmission zeros. The first transmission zero with largest magnitude, z_1, is unstable because the input it generates is unbounded ($|z_1| > 1$). The zeros following after the first are asymptotically stable because the inputs they generate decay to zero.

```
transmission zeros of ns = 7:
  -2.6310 + 0.0000i
   0.9988 + 0.0000i
  -0.6078 + 0.6740i
  -0.6078 - 0.6740i
  -0.3241 + 0.0000i
```

The discrete eigenvalues of the $n_s = 7$ system were converted to continuous time eigenvalues. The continuous time eigenvalues all have negative real parts, again indicating asymptotic stability of the system (See Figure 15).

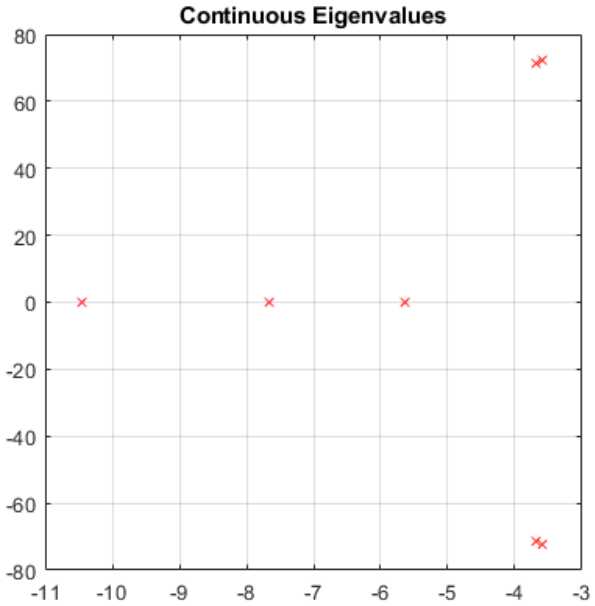


Figure 15: MIMO Continuous Eigenvalues $n_s = 7$

The imaginary part of the continuous time eigenvalues, λ_c, were chosen to be the smallest value because the imaginary part is not uniquely defined. There are two oscillators because there are two complex pairs of eigenvalues. The natural frequencies of the oscillators were found to be 11.322 and 11.503 Hz by analyzing the imaginary part. These frequencies are approximately the frequency at which there is an amplitude peak in the bode plots of Section 3.1.3. This is reasonable because the peak frequency is approximately equal to the natural frequency of the system for small ζ, i.e. little to no damping.

Peak frequency equation:

$$\omega_p = \omega_n \sqrt{1 - 2\zeta^2}$$

```
From the set of lam_c we see there are two damped oscillators:
lam_c =
  -3.6842 +71.1394i
  -3.6842 -71.1394i
```

```
-3.5800 +72.2737i
-3.5800 -72.2737i
```

3.2.1 $n_s = 7$ Discrete Eigenvalues and Transmission Zeros per Channel

In order to determine which components of the physical system are linked to which input or output, the eigenvalues and transmission zeros are plotted for each channel of the $n_s = 7$ model. The poles corresponding to components of the physical system will be cancelled by a transmission zero if the channel does not contain that component.

(1,1) Channel

The (1,1) Channel, corresponding to u_1 input and y_1 output, consists of one low-pass filter and one oscillator (See Figure 16). This single channel can then be represented by a three state model (poles from oscillator pair and one low-pass filter).

The deduction that this channel contains an oscillator also matches the result of the channel's frequency response plot, Figure 10, as it contains a peak indicating resonance. The bode plot also reflects the shape of a low-pass filter as the low frequencies are passed and the magnitude begins to decrease afterwards.

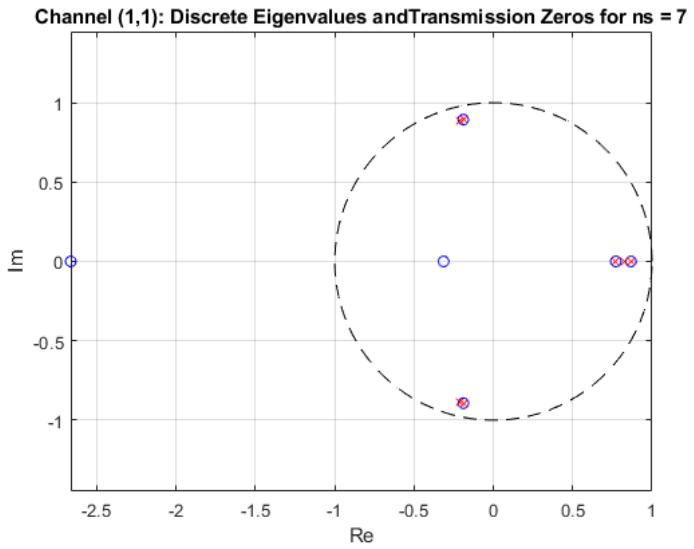


Figure 16: Channel(1,1) Eigenvalues and Zeros for $n_s = 7$

```
3 state model: one oscillator and one LP filter in channel (1,1)
Singular Values for Channel (1,1):
    0.1955
    0.1621
    0.1224
```

(1,2) Channel

The (1,2) Channel, corresponding to u_2 input and y_1 output, consists of one low-pass filter and one oscillator (See Figure 17). This single channel can then be represented by just a three state model (poles from oscillator pair and one low-pass filter).

Similar to Channel (1,1), the frequency response plot of Channel (1,2), Figure 11, also shows characteristics of having both a low-pass filter and resonant component (letting low frequencies pass and having a peak amplitude).

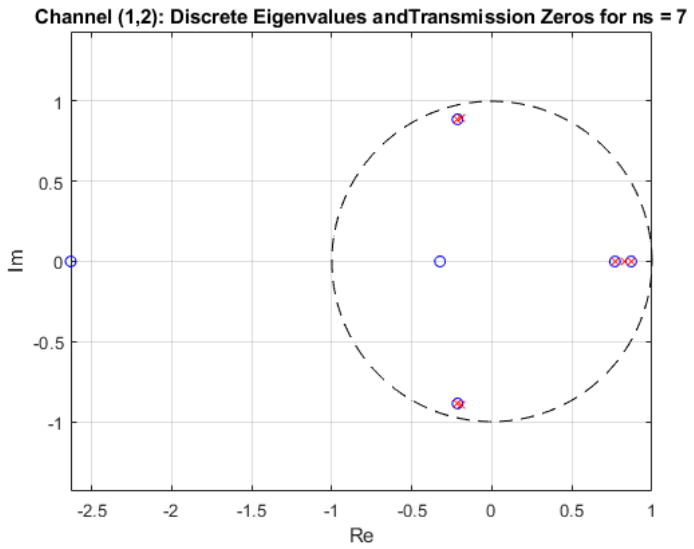


Figure 17: Channel(1,2) Eigenvalues and Zeros for $n_s = 7$

```
3 state model: one oscillator and one LP filter in channel (1,2)
Singular Values for Channel (1,2):
    0.1972
    0.1627
    0.1227
```

(2,1) Channel

The (2,1) Channel, corresponding to u_1 input and y_2 output, consists of one low-pass filter and one high-pass filter (See Figure 18). The high-pass filter comes the fact that an uncanceled transmission zero is close to two poles. This single channel can then be represented by just a two state model (two filters, one low-pass and one high-pass).

The frequency response plot of this channel, Figure 12, reflects these physical components as it did not contain a sharp amplitude peak, indicating there is no resonant component. Instead, the magnitude of the plot forms a hill shape which corresponds to the bode diagram of a high-pass, low-pass filter combination.

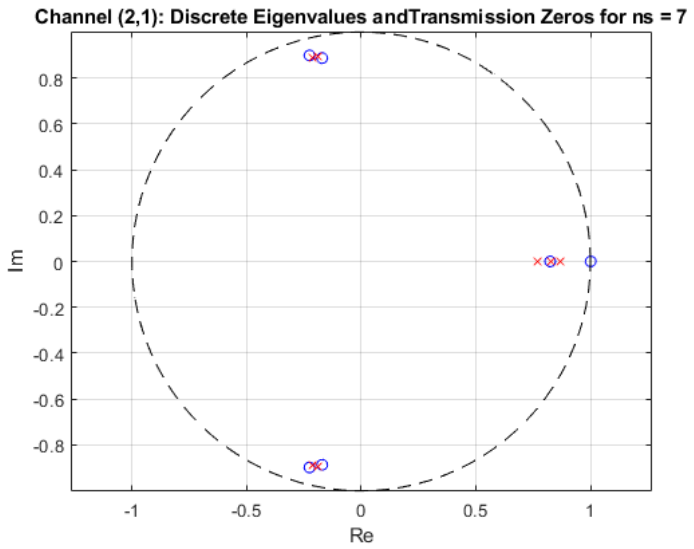


Figure 18: Channel(2,1) Eigenvalues and Zeros for $n_s = 7$

```
2 state model: one LP and one HP filter in channel (2,1)
Singular Values for Channel (2,1):
    0.0971
    0.0811
```

(2,2) Channel

The (2,2) Channel, corresponding to u_2 input and y_2 output, consists of an oscillator, one low-pass filter and one high-pass filter (See Figure 19). The high-pass filter comes from the fact an uncanceled transmission zero is close to two channel poles. This single channel can then be represented by just a four state model (poles from oscillator pair and two filters).

The frequency response plot of Channel (2,2), Figure 13, reflects these physical components as it has the hill shaped magnitude of a high-pass and low-pass filter combination, as well as a peak amplitude indicative of a resonant component (oscillator).

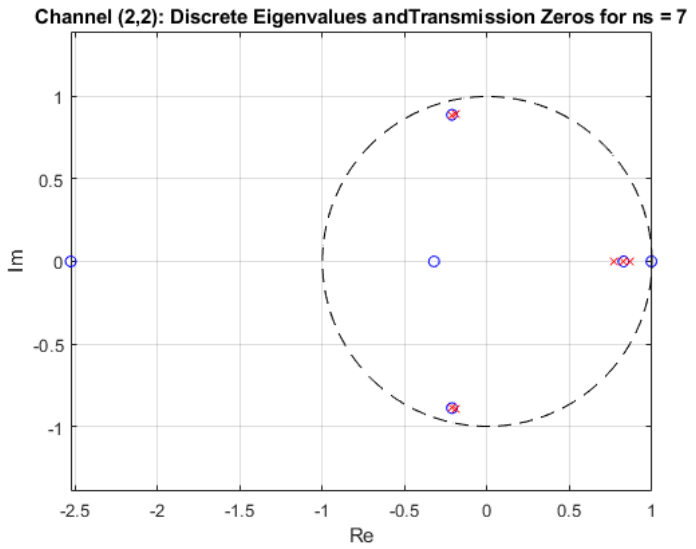
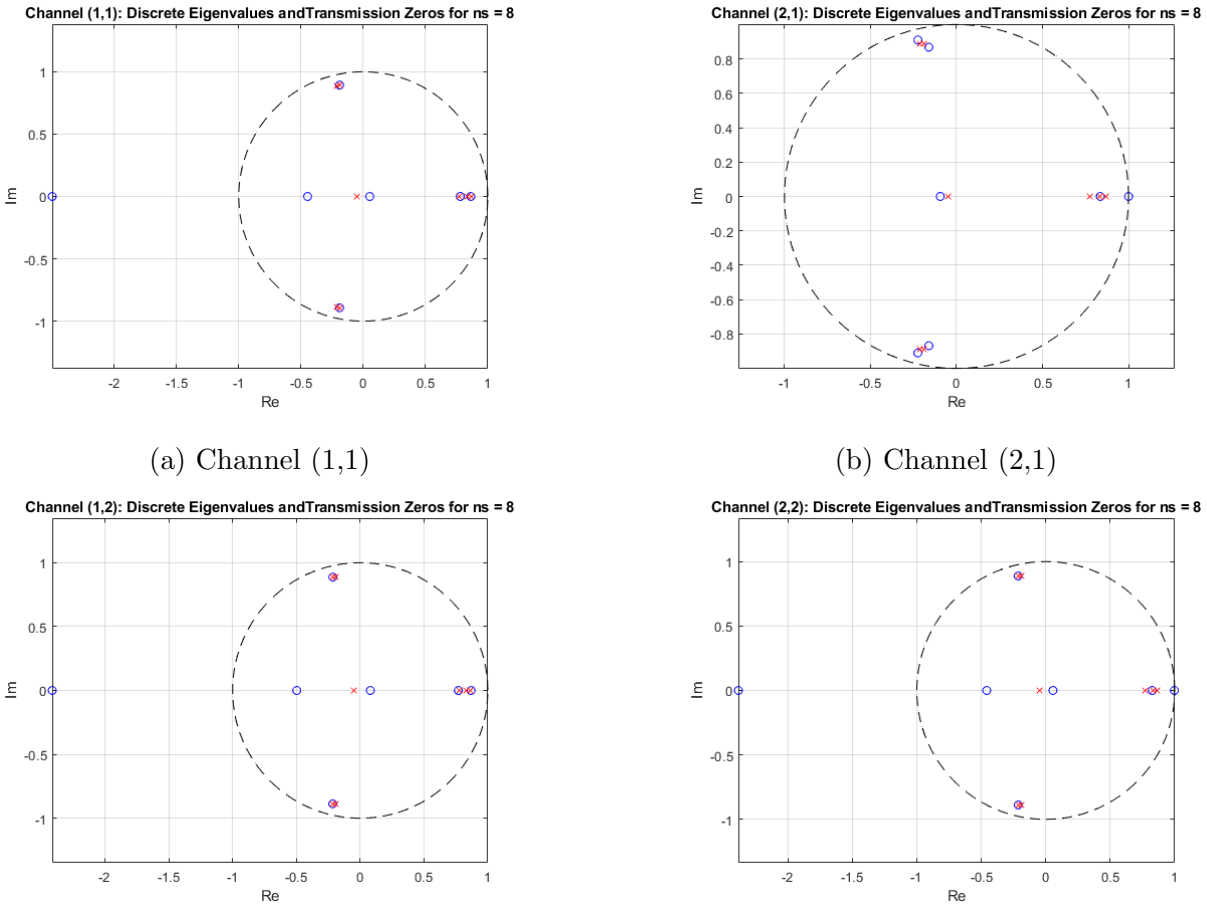


Figure 19: Channel(2,2) Eigenvalues and Zeros for $n_s = 7$

```
4 state model: one oscillator and one LP filter, one HP filter in channel (2,2)
Singular Values for Channel (2,2):
    0.1649
    0.1638
    0.0787
    0.0767
```

3.2.2 $n_s = 8$ Discrete Eigenvalues and Transmission Zeros per Channel

The eigenvalues and transmission zeros of each channel of the $n_s = 8$ are also plotted to demonstrate that the extra pole ends up being cancelled by a transmission zero and each channel contains the same physical components. From Figures 20a - 20d, the extra eigenvalue at $\sim (0, 0)$ due to the added state dimension is always cancelled by a transmission zero.



(a) Channel (1,1)

(b) Channel (2,1)

(c) Channel (1,2)

(d) Channel (2,2)

Figure 20: Eigenvalues and Zeros for $n_s = 8$

3.3 Task 3: Block diagram from analysis of individual channels

The topology of the system can be determined after analyzing the behavior of the eigenvalues and transmission zeros. As previously mentioned, there are two oscillators, **OSC1** and **OSC2**, marked by the appearance of the complex conjugate pairs located in the left-half plane (Figure 20). The three poles on the real axis in the right-half plane are low-pass filters, **LP1**, **LP2**, and **LP3**. The last component of the system is a high-pass filter, **HP**, observed in channels (2,1) and (2,2) with the appearance of a zero at $s = 1$. It is important to note

that the high-pass filter is effectively the result of the association between the zero and one of the neighboring poles.

Each channel exhibits a unique topology, however, it cannot be determined which specific pole is associated with the zero at $s = 1$. As a result, the system diagram can exist in two forms:

1. Inputs in the (1,1) channel pass through **OSC1** and **LP2**.
2. Inputs in the (1,2) channel pass through **OSC2** and **LP2**.
3. Inputs in the (2,1) channel pass through **LP1** and **HP**, a result of the combination of the zero at $s = 1$ and **LP3**.
4. Inputs in the (2,2) channel pass through **OSC2**, **LP2**, and **HP**, a result the zero at $s = 1$ and **LP3**.

Alternatively,

1. Inputs in the (1,1) channel pass through **OSC1** and **LP2**.
2. Inputs in the (1,2) channel pass through **OSC2** and **LP2**.
3. Inputs in the (2,1) channel pass through **LP3** and **HP**, which is the result of the combination of the zero at $s = 1$ and **LP1**.
4. Inputs in the (2,2) channel pass through **OSC2**, **LP3**, and **HP**, which is the result of the zero at $s = 1$ and **LP1**.

The following figure describes the inputs visually (Figure 21).

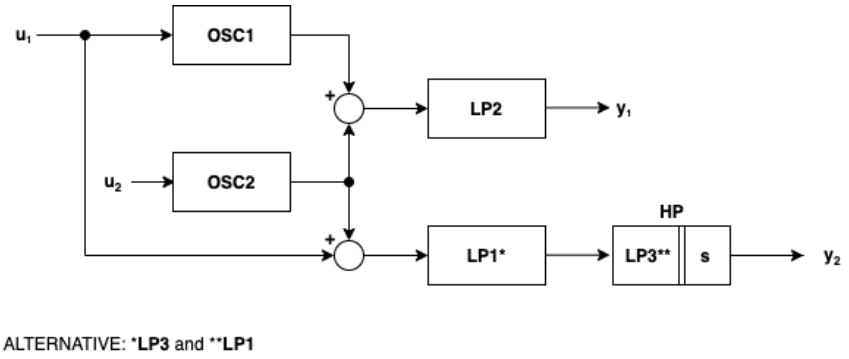


Figure 21: Autocorrelation of Inputs

Overall, the system contains two oscillators, two low-pass filters, and one high-pass filter. However, because the association of the zero at $s = 1$ to its respected pole cannot be definitively determined from the information provided, we state that there are three potential low-pass filters, with the understanding that one of the poles is affiliated with the zero at $s = 1$. In some instances, the effects of one of the low-pass filters (**LP2**) is effectively cancelled with the appearance of a transmission zero at that location; as a result, that pole is not present in the transfer function for channels (2,1) and (2,2) (Figure 21). The behavior of the high-pass filter was observed in the (2,1) and (2,2) channels with a rounded and decreasing magnitude curve towards higher frequencies in the magnitude portion of the Bode plots, which is characteristic of high-pass filters interacting with low-pass filters (Figures 12 and 13).

3.4 Task 4: Impulse response identification from white noise inputs

The system was tested using white noise input in order to construct the impulse response. A pulse input like used in Section 3.1 is not always ideal to determine an impulse response due to disturbance and measurement noise. The ideas of auto-correlation and cross-correlation are utilized to create the impulse response using white noise.

$$\text{Auto-correlation of } \mathbf{u} : R_{uu}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{u}_{k+q}\mathbf{u}_q^T \ \in \mathbb{R}^{n\times n} \tag{9}$$

$$\text{Auto-correlation of } \mathbf{y} : R_{yy}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{y}_{k+q}\mathbf{y}_q^T \ \in \mathbb{R}^{m\times m} \tag{10}$$

$$\text{Cross-correlation } : R_{yu}[k] = \lim_{p\to\infty} \frac{1}{2p} \sum_{q=-p}^{p} \mathbf{y}_{k+q}\mathbf{u}_q^T \ \in \mathbb{R}^{m\times n} \tag{11}$$

where k is the "lag index" which corresponds to time $t_s k$ where t_s is the sample period for the data.

If the input signal u is "zero mean, unit variance white noise" then

$$R_{uu}[k] = \begin{cases} I & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{12}$$

The auto-correlation of the white-noise input is plotted versus lag time to show that the two input channels are uncorrelated for all lag factors k, and with a variance of approximately 4 (see Figure 22 and the following MATLAB results).

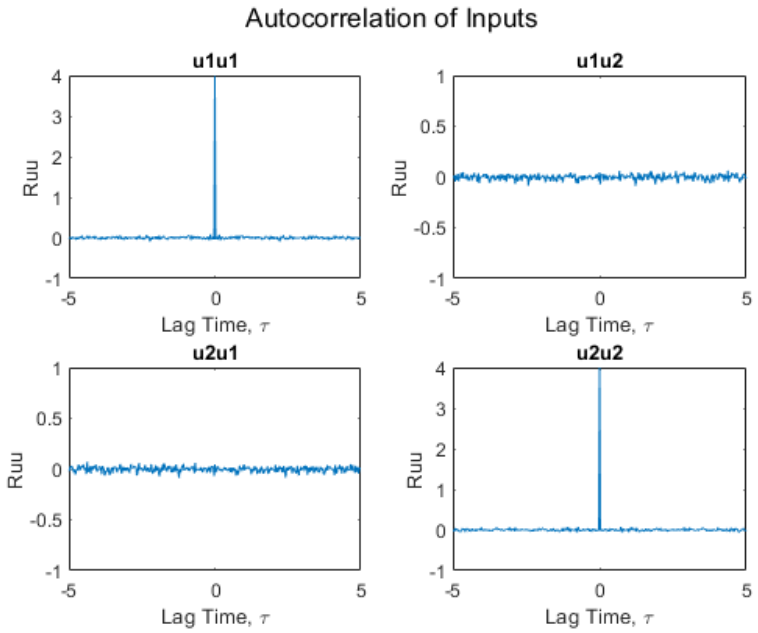


Figure 22: Autocorrelation of Inputs

```
Mean of each input channel: mean(u1) = -0.001, mean(u2) = 0.001
Ruu[0]=
    3.9878    0.0465
    0.0465    4.0219
```

A property of white noise input is that the cross correlation with output is equal to the system's impulse response. This is because R_{uu} is of the form in Equation (12) but in our case instead of unit variance, variance of 4. Plugging this result into Equation (13), gives

the impulse response.

$$\begin{aligned} R_{yu}[k] &= \lim_{p\to\infty}\frac{1}{2p}\sum_{q=-p}^{p}\mathbf{y}_{k+q}\mathbf{u}_q^T \\ &= \lim_{p\to\infty}\frac{1}{2p}\sum_{q=-p}^{p}\mathbf{y}_{q}\mathbf{u}_{q-k}^T \\ &= \lim_{p\to\infty}\frac{1}{2p}\sum_{q=-p}^{p}\left(\sum_{q=-\infty}^{\infty}h_l\mathbf{u}_{q-l}\right)\mathbf{u}_{q-k}^T \\ &= \sum_{q=-\infty}^{\infty}h_l\left(\lim_{p\to\infty}\frac{1}{2p}\sum_{q=-p}^{p}\mathbf{u}_{q-l}\mathbf{u}_{q-k}^T\right) \\ &= \sum_{q=-\infty}^{\infty}h_l R_{uu}[k-l] \end{aligned} \tag{13}$$

The cross correlation of each white noise input/output channel normalized by $\sigma = 2$ was compared with the original impulse response data used to test the system in Task 1, Section 3.1. Figures 23 and 24 demonstrate the impulse response derived from the cross correlation of white noise input data is very similar to the original data from a pulse input.

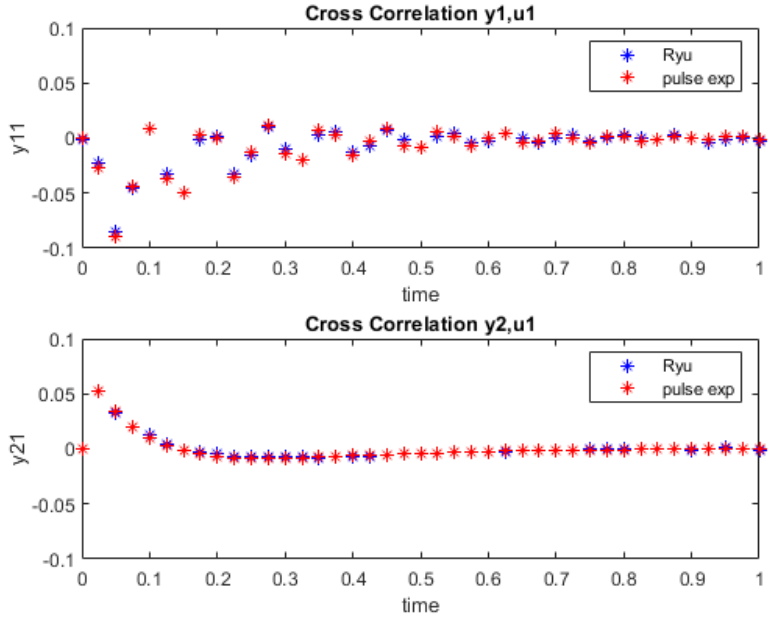


Figure 23: Impulse Response from Cross Correlation for u_1

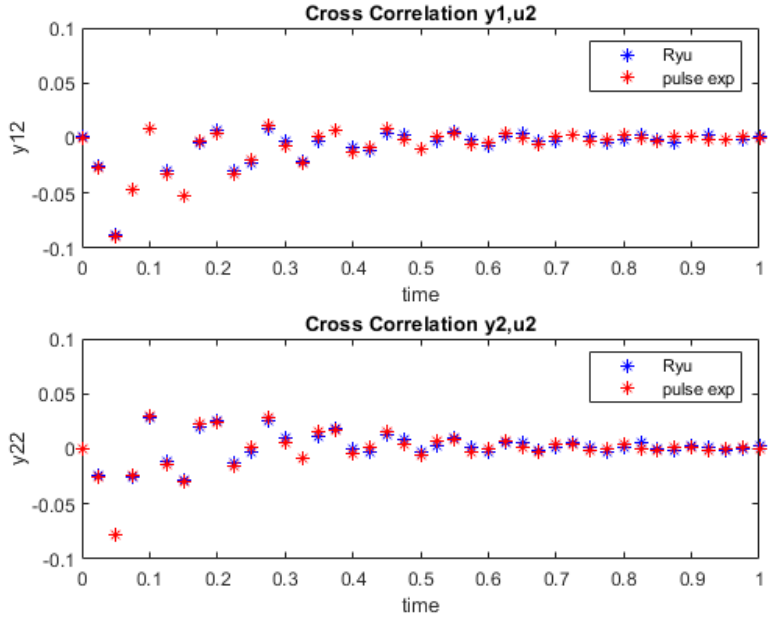


Figure 24: Impulse Response from Cross Correlation for u_2

3.5 Task 5: $\mathcal{H}_2$ norm analysis of identified model

Previously, we showed that the auto-correlation of u, R_{uu}, was not of unit variance (code output on page 24). This suggests each input and output data can be scaled by its standard deviation (in this case, 2). The RMS value of the scaled output data y then becomes

$$||y||^2_{\text{RMS}} = \text{tr}R_{yy}[0] \tag{14}$$

and yields

```
RMS_y =

    0.2239
```

This measurement is, informally stated, a method of quantifying the size of the signal produced by noise and is a valuable feature of the system properties.

This property is also described in part by the $\mathcal{H}_2$ norm of a system, denoted $||P||_{\mathcal{H}_2}$ where P is a discrete-time system for our purposes. In particular, the square root of the integral

of the square of the Frobenius norm of the system's impulse response, described below:

$$||P||_{\mathcal{H}_2} = \sum_{k=0}^{\infty} ||h_k||_F^2, h_k = \text{discrete-time pulse response} \tag{15}$$

Further,

$$||P||_{\mathcal{H}_2} = \sum_{k=1}^{\infty} \text{tr}(h_k^T h_k) = \sum_{k=1}^{\infty} \text{tr}(B^T (A^T)^k C^T C A^k B) \tag{16}$$

and also,

$$||P||_{\mathcal{H}_2} = \sum_{k=1}^{\infty} \text{tr}(h_k^T h_k) = \sum_{k=1}^{\infty} \text{tr}(C A^k B B^T (A^T)^k C^T) \tag{17}$$

We demonstrated that $||P||_{\mathcal{H}_2}$ of the derived 7-state model from our previous Hankel matrix analysis in Section 3.1 outputs a very similar value using Equation 16 as well as with 17 correspondingly below:

```
Using Equation 9 in the project description:
P_H2_eq9 =
    0.2189
```

and

```
Using Equation 10 in the project description:
P_H2_eq10 =
    0.2189
```

Lastly, when approximating $||P||_{\mathcal{H}_2}$ using experimental pulse response data without the model (Equation 15), we find that

```
Using Equation 8 in the project description:
P_H2_eq8 =
    0.2297
```

(Although the results of Equations 15, 16, 17, and RMS calculations (Equation 14 are almost identical and the $\mathcal{H}_\in$ norm is a norm, it is important to note that $||v||_{\text{RMS}}^2$ is not a norm.)

These results demonstrate that the "size" of a noisy signal is small and the system will exhibit some impact from noise in the variance of its output signals.

3.6 Task 6: $\mathcal{H}_\infty$ norm analysis of identified model

The $\mathcal{H}_\infty$ norm is calculated in order to perform a robustness analysis of our system. If an asymptotically stable, unknown system has an $\mathcal{H}_\infty$ norm less than 1 then at those frequencies the true model is approximately the nominal model.

3.6.1 $\mathcal{H}_\infty$ Calculation Theory

The $\mathcal{H}_\infty$ norm is defined as the maximum singular value of the system's frequency response function, $P(j\omega) = C(j\omega I - A)^{-1}B + D$, shown in Equation (18).

$$||P||^2_{\mathcal{H}_\infty} = \sup_\omega \bar{\sigma}^2(P(j\omega)) = \sup_\omega \lambda_{max}(P(j\omega)^* P(j\omega)) \tag{18}$$

By connecting the the two systems $P(j\omega)$ and its hermitian $P(j\omega)^*$ the following state-space realization can be determined.

$$\begin{aligned} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{z}} \end{bmatrix} &= \begin{bmatrix} A & 0 \\ C^*C & -A^* \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} + \begin{bmatrix} B \\ C^*D \end{bmatrix} \mathbf{u} \\ \bar{\mathbf{y}} &= \begin{bmatrix} D^*C & -B^* \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} + D^*D\mathbf{u} \end{aligned} \tag{19}$$

To simplify calculations, the parameter γ is introduced. The system from $\mathbf{u}$ to $\bar{\mathbf{y}}$ is denoted L. Each output channel of $\bar{y}$ is multiplied by $\frac{1}{\gamma^2}$ For a fixed γ, at some frequency ω_0 the largest eigenvalue associated with the frequency response of $\frac{1}{\gamma^2}L$ equals 1 for some eigenvector $\mathbf{v}$.

$$\begin{aligned}
\left\| \frac{1}{\gamma^2} P \right\|_{\mathcal{H}_\infty}^2 &= \frac{1}{\gamma^2} \sup_\omega \bar{\sigma}(P(j\omega))^2 \\
&= \frac{1}{\gamma^2} \sup_\omega \lambda_{max} L(j\omega) \\
&= \frac{1}{\gamma^2} L(j\omega_0) \\
&= 1 \\
\implies ||P||_{\mathcal{H}_\infty} &= \gamma
\end{aligned} \tag{20}$$

By manipulating Equation (19) by setting $\bar{\mathbf{y}} = \gamma^2 \mathbf{u}$, a closed loop A matrix is derived and presented in Equation (21).

$$A_{clp}(\gamma) = \begin{bmatrix} A + BD_\gamma^{-1} D^* C & -BD_\gamma^{-1} B^* \\ C^*C + C^* D D_\gamma^{-1} D^* C & -A^* - C^* D D_\gamma^{-1} B^* \end{bmatrix} \tag{21}$$

Instead of searching over the frequency for the largest singular value of $P(j\omega)$, a range of γ was searched to determine whether A_{clp} has imaginary eigenvalues. The relationship between the $\mathcal{H}_\infty$ and A_{clp} is summarized below:

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

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

3.6.2 $\mathcal{H}_\infty$ norm and Frequency Response Plot Comparison

Using the discrete-time $n_s = 7$ model derived in Section 3.1, the $\mathcal{H}_\infty$ norm and the frequency at which this maximum singular value occurs are calculated with methods described in Section 3.6.1. These results are compared to the maximum singular value of the actual frequency response data. The results are as follows:

```
Hinf norm = 0.468 = -6.587 db
where max singular value achieves largest value, w = 11.285 Hz
```

```
From data, max sigma = 0.468 = -6.592 db
where max singular value achieves largest value, w = 11.471 Hz
```

The calculated $\mathcal{H}_\infty$ norm is less than one, meaning the derived model is robust, and the location of the max singular value is around the value for the cut-off frequency.

The singular values of the model's frequency response were then plotted alongside the empirical singular values. From Figure 25, one can see both plots were very similar to each other. The $\mathcal{H}_\infty$ norm computed from the model also locates the magnitude and corresponding frequency where the maximum singular value achieves its largest value.

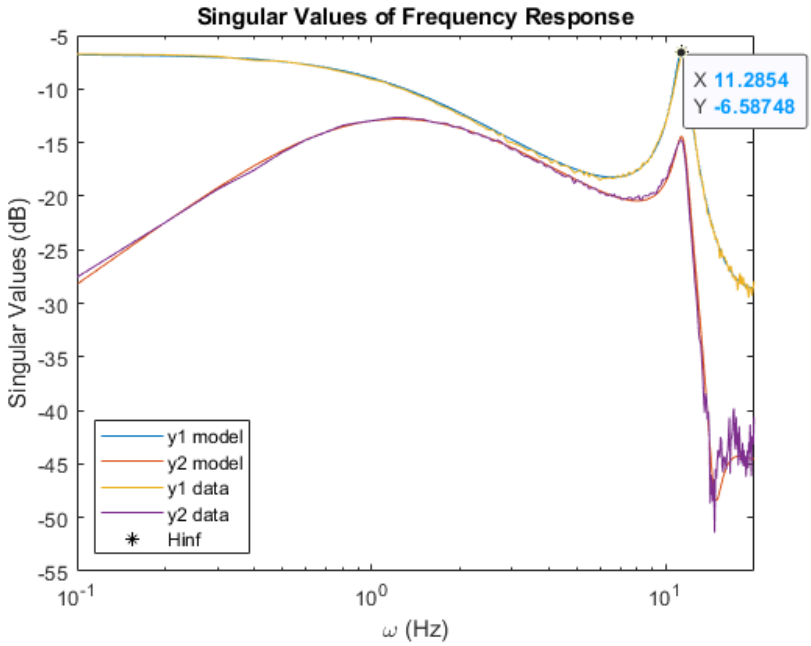


Figure 25: Singular Values of Frequency Response

4 Conclusion

To conclude, this project provides an in-depth study of model identification and verification of a multiple input and output system. We determined the minimal model, {A,B,C,D}

(where D is a zero matrix), of the system using a Hankel matrix construction and manipulation of pulse response data. Second, we determined the physical components of the system with eigenvalue and transmission zero analysis. The identified components are two oscillators, two low-pass filters and one high-pass filter. Additionally, we validated the recreated pulse response of the our minimal model representation by using white-noise input and comparing the result to the collected data in Section 3.1. Finally, we quantitatively verified the minimal model representation with an H_2 norm analysis and a H_∞ norm analysis. We conclude that the optimal model for the system has a state dimension 7.

A Main Code

```
%MAE 270A Project
%Helene Levy Aileen Villalpando
clear; close all; clc;

%plotting commands %change to false if don't want to plot
%task 1 plots
plot_resp = true;
plot_fresp = true;

%task 2 plots
tzero_plot = true;

%task 4 plots
plot_auto = true;
plot_cross = true;

%task 6 plots
hinf_plot = true;


%% Loading and "massaging" initial data
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;
```

```
31
32 %%% remove any offsets in output data using data prior to pulse application
33 y11 = y11 - mean(y11([1:mi-1]));
34 y12 = y12 - mean(y12([1:mi-1]));
35 y21 = y21 - mean(y21([1:mi-1]));
36 y22 = y22 - mean(y22([1:mi-1]));
37
38 %%% rescale IO data so that impulse input has magnitude 1
39 y11 = y11/max(u1);
40 y12 = y12/max(u2);
41 y21 = y21/max(u1);
42 y22 = y22/max(u2);
43 u1 = u1/max(u1);
44 u2 = u2/max(u2);
45
46 ts = 1/40; %%%% sample period
47 N = length(y11); %%%% length of data sets
48 t = [0:N-1]*ts - 1;
49
50 %% Plotting Original Data
51 % figure
52 % subplot(311)
53 % plot(t,u1,'b*','LineWidth',2)
54 % ylabel('$u_1$ (volts)','FontSize',14,'Interpreter','Latex');
55 % grid on
56 % axis([-0.2 2 -0.1 1.1])
57 %
58 % subplot(312)
59 % plot(t,y11,'r*','LineWidth',2)
60 % ylabel('$y_1$ (volts)','FontSize',14,'Interpreter','Latex');
61 % grid on
62 % axis([-0.2 2 -0.1 0.1])
63 %
```

```
%
% 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
% 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])
%

%% Task 1
%joining matrices of data representing same input
```

```
97  y1 = [y11; y21];
98  y2 = [y12; y22];
99
100 %plotting example hankel svds in prompt
101 ex = [20 40 80];
102 figure
103 for k = 1:length(ex)
104     Hex = hankel_n(u1,y1,y2,ex(k));
105     svd_vec = svd(Hex);
106     j = 1:1:40;
107     plot(j,svd_vec(j),'*');
108     hold on;
109     set(gca,'YScale', 'log');
110     axis([0 40 10e-4 10e-1]);
111     hold on;
112 end
113 xlabel('singular value index');
114 ylabel('Hankel singular value');
115 legend("H20","H40","H80");
116
117 %% Task 1 (cont) Constructing Hankel Matrix and Plotting Singular Values
118 %constructing H100 matrix and plotting singular values
119 [H100,Htil] = hankel_n(u1,y1,y2,100);
120 svd_vec = svd(H100);
121
122 figure
123 k = 1:1:40;
124 plot(k,svd_vec(k),'*');
125 hold on;
126 set(gca,'YScale', 'log');
127 axis([0 40 10e-4 10e-1]);
128 xlabel('singular value index');
129 ylabel('Hankel singular value');
```

```
130 title('Singular Values of H100');
131
132 %% Simulating Impulse Response of Each State Model
133 %iterating through different state dims
134 state_dim = [6,7,10,40];
135 An ={0}; Bn = {0}; Cn = {0}; Dn = {0};
136 for i = 1:length(state_dim)
137     nmod = state_dim(i);
138
139     [An{i},Bn{i},Cn{i},Dn{i}] = model_generator(H100,Htil,nmod);
140
141     %simulating response to u1 = [1;0]
142     h1 = zeros(2,100);
143     x1 = Bn{i}*[1;0]; %value x at k = 1
144     for k = 1:100
145         h1(:,k) = Cn{i}*x1;
146         x1 = An{i}*x1;
147     end
148
149     %simulating response to u2 = [0;1]
150     h2 = zeros(2,100);
151     x2 = Bn{i}*[0;1]; %value x at k = 1
152     for k = 1:100
153         h2(:,k) = Cn{i}*x2;
154         x2 = An{i}*x2;
155     end
156     tsim = [1:100]*ts;
157
158     if(plot_resp)
159         %plotting each input/output response combo
160         figure
161         subplot(211)
162         plot(tsim,h1(1,:),'*',t,y11,'*');
```

```
163         grid on
164         xlim([0 2]);
165         ylabel('y1');
166         title(sprintf('Impulse Response of u1 with ns = %4d',nmod));
167         legend('model','data','Location','southeast');
168 
169         subplot(212)
170         plot(tsim,h1(2,:),'*',t,y21,'*');
171         grid on
172         xlim([0 2]);
173         ylabel('y2');
174         xlabel('time (s)');
175 
176         figure
177         subplot(211)
178         plot(tsim,h2(1,:),'*',t,y12,'*');
179         grid on
180         xlim([0 2]);
181         ylabel('y1');
182         title(sprintf('Impulse Response pf u2 with ns = %4d',nmod));
183         legend('model','data','Location','southeast');
184 
185         subplot(212)
186         plot(tsim,h2(2,:),'*',t,y22,'*');
187         grid on
188         xlim([0 2]);
189         ylabel('y2');
190         xlabel('time (s)');
191     end
192 end
193 fprintf('State dimension 6 has an inferior reproduction of the data\n');
194 
195 %% Task 1 (cont) Creating and Plotting Model Frequency Response
```

```
196 %plotting bode diagrams for different model order
197 N_w = 200;
198 w_span = linspace(0,20,N_w); %omega span in hertz
199 %iterating through different state dimensions
200 for k = 1:length(state_dim)
201     P = ss(An{k},Bn{k},Cn{k},Dn{k},ts);
202     [P11_mag,P11_ph] = bode(P(1,1),2*pi*w_span);
203     P11_mag = 20*log10(squeeze(P11_mag));
204     P11_ph = squeeze(P11_ph);
205
206     [P12_mag,P12_ph] = bode(P(1,2),2*pi*w_span);
207     P12_mag = 20*log10(squeeze(P12_mag));
208     P12_ph = squeeze(P12_ph);
209
210     [P21_mag,P21_ph] = bode(P(2,1),2*pi*w_span);
211     P21_mag = 20*log10(squeeze(P21_mag));
212     P21_ph = squeeze(P21_ph);
213
214     [P22_mag,P22_ph] = bode(P(2,2),2*pi*w_span);
215     P22_mag = 20*log10(squeeze(P22_mag));
216     P22_ph = squeeze(P22_ph);
217     if plot_fresp
218         %plotting model freq resp
219         figure (11)
220         subplot(211)
221         plot(w_span,P11_mag);
222         set(gca,'XScale', 'log');
223         ylabel('Magnitude (db)');
224         title('Bode Plot of System Response y1 to u1');
225         grid on; hold on;
226
227         subplot(212)
228         plot(w_span,P11_ph);
```

```
229         set(gca,'XScale', 'log');
230         ylabel('Phase (deg)');
231         xlabel('\omega (Hz)');
232         grid on; hold on;
233
234         figure (12)
235         subplot(211)
236         plot(w_span,P21_mag);
237         set(gca,'XScale', 'log');
238         ylabel('Magnitude (db)');
239         title('Bode Plot of System Response y2 to u1');
240         grid on; hold on;
241
242         subplot(212)
243         plot(w_span,P21_ph);
244         set(gca,'XScale', 'log');
245         ylabel('Phase (deg)');
246         xlabel('\omega (Hz)');
247         grid on; hold on;
248
249         figure (13)
250         subplot(211)
251         plot(w_span,P12_mag);
252         set(gca,'XScale', 'log');
253         ylabel('Magnitude (db)');
254         title('Bode Plot of System Response y1 to u2');
255         grid on; hold on;
256
257         subplot(212)
258         plot(w_span,P12_ph);
259         set(gca,'XScale', 'log');
260         ylabel('Phase (deg)');
261         xlabel('\omega (Hz)');
```

```
262         grid on; hold on;
263
264         figure (14)
265         subplot(211)
266         plot(w_span,P22_mag);
267         set(gca,'XScale', 'log');
268         ylabel('Magnitude (db)');
269         title('Bode Plot of System Response y2 to u2');
270         grid on; hold on;
271
272         subplot(212)
273         plot(w_span,P22_ph);
274         set(gca,'XScale', 'log');
275         ylabel('Phase (deg)');
276         xlabel('\omega (Hz)');
277         grid on; hold on;
278     end
279 end
280
281
282 %% Task 1 (cont) Plotting Empirical Frequency Response
283 if(plot_fresp)
284     figure (11);
285     subplot(211)
286     y11f = fft(y11)./fft(u1);
287     N = length(y11f);
288     om = [0:N-1]/(ts*N);
289     plot(om,20*log10(y11f));
290     set(gca,'XScale', 'log');
291     xlim([0 20]);
292     legend('ns = 6','ns = 7' ,'ns = 10' ,'ns = 40','empirical data',...
293         'Location','southwest');
294     subplot(212)
```

```
295     plot(om,unwrap(angle((y11f)))*180/pi);
296     set(gca,'XScale', 'log');
297     xlim([0 20]);
298
299     figure (12);
300     subplot(211)
301     y21f = fft(y21)./fft(u1);
302     N = length(y21f);
303     om = [0:N-1]/(ts*N);
304     plot(om,20*log10(y21f));
305     set(gca,'XScale', 'log');
306     xlim([0 20]);
307     legend('ns = 6','ns = 7' ,'ns = 10' ,'ns = 40','empirical data',...
308         'Location','southwest');
309     subplot(212)
310     plot(om,unwrap(angle((y21f)))*180/pi);
311     set(gca,'XScale', 'log');
312     xlim([0 20]);
313
314     figure (13);
315     subplot(211)
316     y12f = fft(y12)./fft(u2);
317     N = length(y12f);
318     om = [0:N-1]/(ts*N);
319     plot(om,20*log10(y12f));
320     set(gca,'XScale', 'log');
321     xlim([0 20]);
322     legend('ns = 6','ns = 7' ,'ns = 10' ,'ns = 40','empirical data',...
323         'Location','southwest');
324     subplot(212)
325     plot(om,unwrap(angle((y12f)))*180/pi);
326     set(gca,'XScale', 'log');
327     xlim([0 20]);
```

```
328
329     figure (14);
330     subplot(211)
331     y22f = fft(y22)./fft(u2);
332     N = length(y22f);
333     om = [0:N-1]/(ts*N);
334     plot(om,20*log10(y22f));
335     set(gca,'XScale', 'log');
336     xlim([0 20]);
337     legend('ns = 6','ns = 7' ,'ns = 10' ,'ns = 40','empirical data',...
338         'Location','southwest');
339     subplot(212)
340     plot(om,unwrap(angle((y22f)))*180/pi);
341     set(gca,'XScale', 'log');
342     xlim([0 20]);
343 end
344
345 %% Task 2: Transmission Zeros of MIMO Model
346 %2.1 finding transmission zeros of state model 7
347 ns = 7;
348 [A7,B7,C7,D7] = model_generator(H100,Htil,ns);
349 [z,lam] = tzero_lam_gen(A7,B7,C7,D7,tzero_plot);
350 title('Discrete Eigenvalues and Transmission Zeros');
351 fprintf('transmission zeros of ns = %d: \n',ns);
352 disp(z);
353
354 %2.3 continuous time eigenvalues
355 lamc = log(lam)./ts;
356 %minimizing the omega by 2pik/ts for ones with imaginary part
357 %frequency bounded by [-pi/ts; pi/ts] rad/s which they already are
358
359 if(tzero_plot)
360     figure;
```

```
361     plot(real(lamc),imag(lamc),'rx');
362     axis square ;grid on;
363     title('Continuous Eigenvalues');
364 end
365
366 fprintf('From the set of lam_c we see there are two damped oscillators:\n')
367 disp('lam_c =');disp(lamc(1:4));
368 fprintf('Their natural frequences are %4.3f and %4.3f Hz \n',...
369         imag(lamc(1))/(2*pi),imag(lamc(3))/(2*pi));
370 fprintf(['These frequencies look like the approximate cut-off ',...
371         'frequencies of the frequency responses\n\n']);
372
373 %% Task 2 (cont) Transmission Zeros for Each Channel
374 %creating b1 b2 and c1 c2
375 b7_1 = B7(:,1); b7_2 = B7(:,2);
376 c7_1 = C7(1,:); c7_2 = C7(2,:);
377
378 %Channel (1,1) transmission zeros and eigenvalues
379 [z11,lam11] = tzero_lam_gen(A7,b7_1,c7_1,0,tzero_plot);
380 title(sprintf(['Channel (1,1): Discrete Eigenvalues and',...
381     'Transmission Zeros for ns = %d'],ns));
382 fprintf(['3 state model: one oscillator and '...
383         'one LP filter in channel (1,1)\n']);
384
385 %determining channel's singular values
386 H_11 = hankel_1n(u1,y11,100,3);
387 sig = svd(H_11);
388 disp('Singular Values for Channel (1,1):'); disp(sig(1:3));
389
390 %Channel (2,1) y2,u1
391 [z21,lam21] = tzero_lam_gen(A7,b7_1,c7_2,0,tzero_plot);
392 title(sprintf(['Channel (2,1): Discrete Eigenvalues and',...
393     'Transmission Zeros for ns = %d'],ns));
```

```
394 fprintf('2 state model: one LP and one HP filter in channel (2,1)\n');
395
396 %determining channel's singular values
397 H_21 = hankel_1n(u1,y21,100,2);
398 sig = svd(H_21);
399 disp('Singular Values for Channel (2,1):'); disp(sig(1:2));
400
401 %Channel (1,2) y1,u2
402 [z12,lam12] = tzero_lam_gen(A7,b7_2,c7_1,0,tzero_plot);
403 title(sprintf(['Channel (1,2): Discrete Eigenvalues and',...
404     'Transmission Zeros for ns = %d'],ns));
405 fprintf(['3 state model: one oscillator and one '...
406         'LP filter in channel (1,2)\n']);
407
408 %determining channel's singular values
409 H_12 = hankel_1n(u2,y12,100,3);
410 sig = svd(H_12);
411 disp('Singular Values for Channel (1,2):'); disp(sig(1:3));
412
413 %Channel (2,2) y2,u2
414 [z22,lam22] = tzero_lam_gen(A7,b7_2,c7_2,0,tzero_plot);
415 title(sprintf(['Channel (2,2): Discrete Eigenvalues and',...
416     'Transmission Zeros for ns = %d'],ns));
417 fprintf(['4 state model: one oscillator and one LP filter, '...
418         'one HP filter in channel (2,2)\n']);
419
420 %determining channel's singular values
421 H_22 = hankel_1n(u2,y22,100,4);
422 sig = svd(H_22);
423 disp('Singular Values for Channel (2,2):'); disp(sig(1:4));
424
425 %% Task 2 (cont) Pole-Zero plot for nmod = 8
426 ns = 8;
```

```
[A8,B8,C8,D8] = model_generator(H100,Htil,ns);

%creating b1 b2 and c1 c2
b8_1 = B8(:,1); b8_2 = B8(:,2);
c8_1 = C8(1,:); c8_2 = C8(2,:);

%Channel (1,1) transmission zeros and eigenvalues
[z11,lam11] = tzero_lam_gen(A8,b8_1,c8_1,0,tzero_plot);
title(sprintf(['Channel (1,1): Discrete Eigenvalues and',...
    'Transmission Zeros for ns = %d'],ns));

%Channel (2,1) y2,u1
[z21,lam21] = tzero_lam_gen(A8,b8_1,c8_2,0,tzero_plot);
title(sprintf(['Channel (2,1): Discrete Eigenvalues and',...
    'Transmission Zeros for ns = %d'],ns));

%Channel (1,2) y1,u2
[z12,lam12] = tzero_lam_gen(A8,b8_2,c8_1,0,tzero_plot);
title(sprintf(['Channel (1,2): Discrete Eigenvalues and',...
    'Transmission Zeros for ns = %d'],ns));

%Channel (2,2) y2,u2
[z22,lam22] = tzero_lam_gen(A8,b8_2,c8_2,0,tzero_plot);
title(sprintf(['Channel (2,2): Discrete Eigenvalues and',...
    'Transmission Zeros for ns = %d'],ns));

%% Task 4: Impulse Response ID from White Noise Inputs
%parameters
load u_rand.mat
y1 = u_rand.Y(3).Data;
y2 = u_rand.Y(4).Data;
u1_rand = u_rand.Y(1).Data;
u2_rand = u_rand.Y(2).Data;
```

```
460 ts = 1/40;
461 N = length(y1);
462 t2 = [0:N-1]*ts - 1;
463
464 %remove any offsets from output data
465 y1 = y1 - mean(y1);
466 y2 = y2 - mean(y2);
467
468 u = [u1_rand;u2_rand];
469 y = [y1;y2];
470
471 fprintf('Mean of each input channel: mean(u1) = %4.3f, mean(u2) = ...
       %4.3f\n',...
472         mean(u1_rand), mean(u2_rand));
473
474 %% Task 4 (cont) Autocorrelation of u and plots
475
476 %preallocations
477 % want k[-200,200] so give a 100 index offset on either end of data
478 q_min = 300;
479 q_max = length(u1_rand)-300;
480 N = length(q_min:q_max)-1;
481
482 Ruu = zeros(2,401*2);
483 for i = 1:401
484     %k in [-200, 200]
485     k = i-201;
486
487     %preallocation
488     sum_Ruu = zeros(2,2);
489
490     for q = q_min: q_max
491         %summing each uk+q*uq^T over q set
```

```
492         sum_Ruu = 1/(N)*(u(:,k+q)*u(:,q)')+sum_Ruu;
493     end
494     Ruu(:,2*i-1:2*i) = sum_Ruu;
495 end
496 Tau = linspace(-5,5,length(Ruu)/2);
497 
498 %plotting autocorrelation graphs
499 if plot_auto
500     figure;
501     sgtitle('Autocorrelation of Inputs');
502     subplot(221);
503     %autocorr u1u1
504     plot(Tau,Ruu(1,1:2:end)); hold on;
505     axis([-5 5 -1 4]);
506     xlabel('Lag Time, \tau');
507     ylabel('Ruu');
508     title('u1u1');
509 
510     %autocorr u1u2
511     subplot(222);
512     plot(Tau,Ruu(1,2:2:end)); hold on;
513     axis([-5 5 -1 1]);
514     xlabel('Lag Time, \tau');
515     ylabel('Ruu');
516     title('u1u2');
517 
518     %autocorr u2u1
519     subplot(223);
520     plot(Tau,Ruu(2,1:2:end)); hold on;
521     axis([-5 5 -1 1]);
522     xlabel('Lag Time, \tau');
523     ylabel('Ruu');
524     title('u2u1');
```

```
525
526     %autocorr u2u2
527     subplot(224);
528     plot(Tau,Ruu(2,2:2:end)); hold on;
529     axis([-5 5 -1 4]);
530     xlabel('Lag Time, \tau');
531     ylabel('Ruu');
532     title('u2u2');
533 end
534 %variance of input matrix (Ruu[0]);
535 V = Ruu(:,401:402);
536 disp('Ruu[0]=');disp(V);
537 fprintf('This indicates a variance of ¬4 for each input channel\n');
538
539 %% Task 4 (cont) Cross correlation of u,y and plots
540
541 %preallocations
542 % want k[-200,200] so give a 100 index offset on either end of data
543 q_min = 300;
544 q_max = length(u1_rand)-300;
545 N = length(q_min:q_max)-1;
546
547 Ruu = zeros(2,401*2);
548 %lag time corresponds to T = k*ts -> k = Tbound/ts
549 kmin = -0.2/ts; kmax= 2/ts;
550 for i = 1:kmax-kmin+1
551     %k in [-200, 200]
552     %k is [-0.2/ts,2/ts]
553     k = i+(kmin-1);
554
555     %preallocation
556     sum_Ryu = zeros(2,2);
557
```

```
558     for q = q_min: q_max
559         %summing each uk+q*uq^T over q set
560         sum_Ryu = 1/(N)*(y(:,k+q)*u(:,q)')+sum_Ryu;
561     end
562     Ryu(:,2*i-1:2*i) = sum_Ryu;
563 end
564 %normalizing columns by variance of u1 data
565 Ryu(:,1:2:end) = Ryu(:,1:2:end)/(V(1,1));
566 
567 %normalizing columns by variance of u2 data
568 Ryu(:,2:2:end) = Ryu(:,2:2:end)/(V(2,2));
569 Tau = linspace(-0.2,2,length(Ryu)/2);
570 
571 %plotting cross correlation graphs
572 if plot_cross
573     figure;
574     subplot(211);
575     %crosscorr y1u1
576     plot(Tau,Ryu(1,1:2:end),'b*',t,y11,'r*');
577     axis([0 1 -0.1 0.1]);
578     xlabel('time');
579     ylabel('y11');
580     legend('Ryu','pulse exp');
581     title('Cross Correlation y1,u1');
582 
583     %crosscorr y2u1
584     subplot(212);
585     plot(Tau,Ryu(2,1:2:end),'b*',t,y21,'r*');
586     axis([0 1 -0.1 0.1]);
587     xlabel('time');
588     ylabel('y21');
589     legend('Ryu','pulse exp');
590     title('Cross Correlation y2,u1');
```

```
591
592
593     %crosscorr y1u2
594     figure;
595     subplot(211);
596     plot(Tau,Ryu(1,2:2:end),'b*',t,y12,'r*');
597     axis([0 1 -0.1 0.1]);
598     xlabel('time');
599     ylabel('y12');
600     legend('Ryu','pulse exp');
601     title('Cross Correlation y1,u2');
602
603     %crosscorr y2u2
604     subplot(212);
605     plot(Tau,Ryu(2,2:2:end),'b*',t,y22,'r*');
606     axis([0 1 -0.1 0.1]);
607     xlabel('time');
608     ylabel('y22');
609     legend('Ryu','pulse exp');
610     title('Cross Correlation y2,u2');
611 end
612
613 %% Task 4 (cont) Autocorrelation of y (scaled)
614 y_scale = y/2;
615 %preallocations
616 % want k[-200,200] so give a 100 index offset on either end of data
617 q_min = 300;
618 q_max = length(y1)-300;
619 N = length(q_min:q_max)-1;
620
621 Ryy = zeros(2,401*2);
622 for i = 1:401
623     %k in [-200, 200]
```

```
624     k = i-201;
625 
626     %preallocation
627     sum_Ryy = zeros(2,2);
628 
629     for q = q_min: q_max
630         %summing each uk+q*uq^T over q set
631         sum_Ryy = 1/(N)*(y_scale(:,k+q)*y_scale(:,q)')+sum_Ryy;
632     end
633     Ryy(:,2*i-1:2*i) = sum_Ryy;
634 end
635 Ryy_0 = Ryy(:,401:402);
636 
637 %% Task 5: H2 norm
638 %RMS value of output when u is zero mean unit variance noise
639 y_rms = sqrt(trace(Ryy_0));
640 fprintf('RMS value of scaled output y = %4.4f\n',y_rms);
641 
642 %||P||H2 from 7 state model
643 ns = 7;
644 [A7,B7,C7,D7] = model_generator(H100,Htil,ns);
645 
646 P_norm9 = 0;
647 for k = 1: length(u1)
648     P_norm9 = P_norm9 + trace(B7'*(A7')^k*(C7')*C7*(A7^k)*B7);
649 end
650 P_norm9 = sqrt(P_norm9);
651 fprintf('||P||H2 using State Space Model Eqn (9) = %4.4f\n',P_norm9);
652 
653 P_norm10 = 0;
654 for k = 1: length(u1)
655     P_norm10 = P_norm10 + trace(C7*(A7^k)*B7*(B7')*(A7')^k*C7');
656 end
```

```
657 P_norm10 = sqrt(P_norm10);
658 fprintf('||P||H2 using State Space Model Eqn (10) = %4.4f\n',P_norm10);
659
660 %||P||H2 from discrete time pulse response
661 y_imp = [y11 y12; y21 y22];
662 P_norm8 = 0;
663 for k = 1: length(u1)
664     P_norm8 = P_norm8 + trace(y_imp(:,2*k-1:2*k)'*y_imp(:,2*k-1:2*k));
665 end
666 P_norm8 = sqrt(P_norm8);
667 fprintf('||P||H2 using discrete time pulse response y Eqn (8) = ...
       %4.4f\n',...
668         P_norm8);
669
670 %% Task 6: H infinity norm
671
672 plot_6fresp = true;
673 %H infinity norm calculations
674 svd_vec = svd(H100); ns = 7;
675 %recommended bounds for gamma found online
676 gam = [svd_vec(1),2*sum(svd_vec)];
677 tol = 0.01;
678 [Hinf, wmax] = Hinf_norm_d(A7,B7,C7,D7,gam,tol,ts);
679 fprintf('\nHinf norm = %4.3f = %4.3f db\n',Hinf,20*log10(Hinf));
680 fprintf('where max singular value achieves largest value, w = %4.3f ...
       Hz\n', ...
681     wmax/(2*pi));
682
683 %frequency response plot using model
684 P = ss(A7,B7,C7,D7,ts);
685 [SV,W] = sigma(P,2*pi*w_span); hold on;
686 if hinf_plot
687     figure;
```

```
688     plot(W/(2*pi),20*log10(SV(1,:)));  hold on;
689     plot(W/(2*pi),20*log10(SV(2,:))); hold on;
690 end
691
692 %frequency response plot using pulse response data
693 y11f = fft(y11)./fft(u1);
694 y21f = fft(y21)./fft(u1);
695 y12f = fft(y11)./fft(u2);
696 y22f = fft(y22)./fft(u2);
697
698 %forming frequency model values for each corresponding frequency
699 N = length(y11f);
700 om = [0:N-1]/(ts*N);
701 sig = zeros(2,length(om));
702 max_sig = 0; max_w = 0;
703 for i = 1:length(om)
704     H = [y11f(i) y12f(i); y21f(i) y22f(i)];
705     %taking singular values of each H
706     sig(:,i) = svd(H);
707     if max(sig(:,i)) > max_sig
708         max_sig = max(sig(1,i));
709         max_w = om(i);
710     end
711 end
712
713 fprintf('\nFrom data, max sigma = %4.3f = %4.3f db\n',max_sig,...
714         20*log10(max_sig));
715 fprintf('where max singular value achieves largest value, w = %4.3f ...
       Hz\n', ...
716     max_w);
717
718 %plotting frequency vs. singular values of pulse response data
719 if hinf_plot
```

```
720     plot(om,20*log10(sig(1,:)));  hold on;
721     plot(om,20*log10(sig(2,:))); hold on;
722     plot(wmax/(2*pi),20*log10(Hinf),'k*'); hold on;
723     legend('y1 model','y2 model','y1 data','y2 ...
            data','Hinf','Location','southwest');
724     axis([0,20,-55, -5])
725     set(gca,'XScale', 'log');
726     xlabel('\omega (Hz)');
727     ylabel('Singular Values (dB)');
728     title('Singular Values of Frequency Response');
729 end
```

B Helper Functions

```
1  function [Ass,Bss,Css,Dss] = model_generator(H,Htil,nmod)
2  %generating a model of size nmod based on larger hankel matrices
3
4  %taking the svd of the H100 matrix and decreasing to new state dim
5  [U,S,V] = svd(H);
6  Un = U(:,1:nmod);
7  Sn = S(1:nmod,1:nmod);
8  Vn = V(:,1:nmod);
9
10 %creating a new hankel matrix based on lower state dim
11 Hnew = Un*Sn*(Vn');
12
13 %Observability and Controllability
14 O = Un*diag(sqrt(diag(Sn)));
15 C = diag(sqrt(diag(Sn)))*Vn';
16
```

```
%C matrix in state model
Css = O(1:2,:);

%B matrix in state model
Bss = C(:,1:2);

%A matrix creation in state model
Oleft = inv(diag(sqrt(diag(Sn))))*Un';
Cright = Vn*inv(diag(sqrt(diag(Sn))));
Ass = Oleft*Htil*Cright;

Dss = zeros(2,2);
end
```

```
function [H,Htil] = hankel_n(u,y1,y2,nr)
%finding Hankel Matrix for 2 input 2 output system
ind_off = find(u); %finds the index of the first non zero value

nc = nr;
H = zeros(2*nr,2*nc);
Htil = zeros(2*nr,2*nc);
for k = 1:nr
    %indexing based on offset when pulse hits
    ind = ind_off + k + [0:nc-1];

    %creating H based off response data with little h being 2x2 matrices
    %because 2 input 2 output system
    %   | h1 h2
    %   | h2 h3
    %(2k-1:2k) index needed in order to skip every 2 rows to make room for
    %2x2 matrix
    H(2*k-1:2*k,1:2:end)=y1(:,ind);
```

```
    H(2*k-1:2*k,2:2:end)=y2(:,ind);

    %creating Htilda which is just H but shifted over
    %   | h2 h3
    %   | h3 h4
    Htil(2*k-1:2*k,1:2:end)=y1(:,ind+1);
    Htil(2*k-1:2*k,2:2:end)=y2(:,ind+1);
end
end
```

```
function [Hnew,H] = hankel_1n(u,y,nr,nmod)
%finding Hankel Matrix for 1 input 1 output system
ind_off = find(u); %finds the index of the first non zero value

nc = nr;
H = zeros(nr,nc);
Htil = zeros(nr,nc);
for k = 1:nr
    %indexing based on offset when pulse hits
    ind = ind_off + k + [0:nc-1];

    H(k,:)=y(ind);
    Htil(k,:)=y(ind+1);
end

%taking the svd of the H100 matrix and decreasing to new state dim
[U,S,V] = svd(H);
Un = U(:,1:nmod);
Sn = S(1:nmod,1:nmod);
Vn = V(:,1:nmod);

%creating a new hankel matrix based on lower state dim
```

```
23 Hnew = Un*Sn*(Vn');
24 end
```

```
1  function [z,lam] = tzero_lam_gen(A,B,C,D,plot_bool)
2  %finding the transmission zeros and eigenvalues based on system matrices
3
4  ns = length(A); %state dimension
5  n = ns + size(B,2); %# columns of M matrix
6
7  %Constructing M matrix
8  %M = [A B; -C -D]
9  M = zeros(n,n);
10 M(1:ns,1:ns) = A;
11 M(1:ns,ns+1:end) = B;
12 M(ns+1:end,1:ns) = -C;
13 M(ns+1:end,ns+1:end) = -D;
14
15 %transmission zero calculation
16 z = eig(M,diag([ones(ns,1);zeros(n-ns,1)]));
17 z = sort(z,'descend'); %sorting from greatest magnitude to least
18 inf_check = isinf(z);
19 z_ind = find(inf_check); %finding all the indices of "inf" values
20 z = z(z_ind(end)+1:end); %removing inf values
21
22 lam = eig(A);
23
24 if (plot_bool)
25     figure;
26     %plotting transmission zeros and eigenvalues
27     plot(real(z),imag(z),'bo',real(lam),imag(lam),'rx');
28     hold on;
29
```

```
30      %plotting circle radius 1
31      th = linspace(0,2*pi); r = 1;
32      x_circ = r*cos(th);
33      y_circ = r*sin(th);
34      plot(x_circ,y_circ,'k--');
35      xlabel('Re'); ylabel('Im');
36      axis equal;grid on;
37  end
38  end
```

```
1   function [H_infn,nu] = Hinf_norm_c(A,B,C,D,gam,tol)
2   %finding the Hinf norm and wmax of a continuous time system
3   I = eye(size(D'*D));
4   gamL = gam(1);
5   gamU = gam(2);
6
7   gamH = gamU;
8   while (gamH - gamL)/gamL > tol
9       gamma = (gamL + gamH)/2;
10
11      %building Aclp
12      Dg = gamma^2*I-D'*D;
13      Aclp = [A+B*inv(Dg)*D'*C, -B*inv(Dg)*B';...
14              C'*C+C'*D*inv(Dg)*D'*C, -A'-C'*D*inv(Dg)*B'];
15      lam = eig(Aclp);
16
17      %if the minimum real part is close to zero (only imaginary part?)
18      if min(abs(real(lam))) < 1.0e-5
19          gamL = gamma;
20      else
21          gamH = gamma;
22      end
```

```
23 end
24 H_infn = gamma;
25
26 %finding purely imaginary eigenvalue -> continuous time freq
27 nu = 0;
28 for k = 1:length(lam)
29     %if almost no real part
30     if  abs(real(lam(k))) < 1.0e-5
31         nu = abs(imag(lam(k)));
32     end
33 end
34
35 end
```

```
1  function [H_infn,w_max] = Hinf_norm_d(A,B,C,D,gam,tol,ts)
2  %finding the Hinf norm and wmax of a discrete time system
3
4  %converting discrete time matrices to continuous
5  I = eye(size(A));
6  Ac = -inv(I+A)*(I-A);
7  Bc = sqrt(2)*inv(I+A)*B;
8  Cc = sqrt(2)*C*inv(I+A);
9  Dc = D - C*inv(I+A)*B;
10
11 [H_infn,nu] = Hinf_norm_c(Ac,Bc,Cc,Dc,gam,tol);
12
13 w_max = log((1 + 1j*nu)/(1 - 1j*nu))/(1j*ts);
14 w_max = real(w_max);
15 end
```