Principles of Wireless Communications Lab 2A

Kristtya Guerra

February 23, 2021

Abstract

I demodulated transmitted and received data from a MIMO system using Matlab. The data was demodulated using the zero-forcing receiver methodology and the minimum-mean-square-error receiver methodology. I was nearly able to successfully retrieve the original transmitted signals with an error rate of 0% when using MMSE equalization and 0% when using zero-forcing to retrieve the original transmitted signal from transmitter 1.

1 Introduction

Multiple-input and multiple-output (MIMO) is a wireless technology setup that allows for the user to have multiple transmitters and receivers to exchange more data at the same time. In this lab, there were two transmitters and a single receiver with the layout as shown in figure 1.

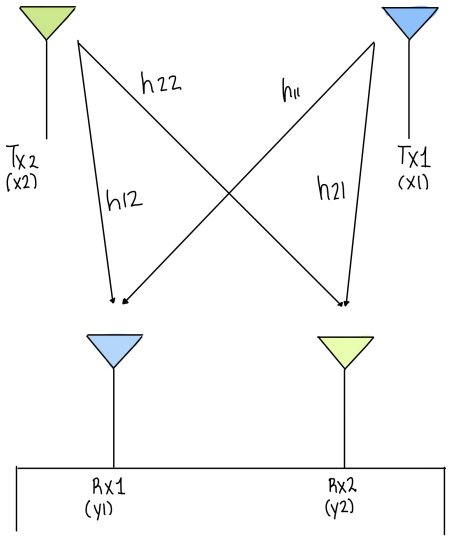

Figure 1: Multi-antenna receiver with two transmitters

Transmitter 1's intended receiver is Rx_1 while transmitter 2's intended receiver is Rx_2. The arrows show how the transmitters are sending data to both receivers. The data being processed for this lab was provided by the professor. The following is the sequence of the two transmitters:

1. **5000 zero samples** from transmitter 1 and 2
2. Transmitter 1 sends **128 pseudo random bits**, encoded using BPSK and **40 sample long rectangular pulses**, transmitter 2 sends zeros during this
3. **5000 zero samples** are sent from both transmitters
4. Transmitter 2 sends **128 pseudo random bits**, encoded using BPSK and **40 sample long rectangular pulses**, transmitter 2 sends zeros
5. **5000 zero samples** from both transmitters
6. Transmitters 1 and 2 send **1024 data bits**, which are randomly generated in this lab, using BPSK rectangular pulses that are **40 samples each**

In a real wireless system, the first 128 random bits would be known by the receiver while the 1024 data bits would be unknown data. This allows for data to be synchronized and the receiver to identify certain key points during transmission.

Although these systems allow for more data to be transmitted at the same time, there is the issue of interference occurring at the receiving end between the two signals being transmitted by Tx_1, Tx_2, and noise from the environment. As shown in figure 1, receiver 1 is still getting a signal from Tx_2 even though that recipient only wants data from Tx_1.

This relationship between the received signal and the transmitted signal can be expressed with the following equations:

$$y_1[k] = h_{11}x_1[k] + h_{12}x_2[k] + n_1[k] \tag{1}$$

$$y_2[k] = h_{21}x_1[k] + h_{22}x_2[k] + n_2[k] \tag{2}$$

With equation 1 representing the received signal from Rx_1 and equation 2 representing the received signal from Rx_2. h_{11}, h_{12}, and h_{21} represent the individual transfer functions between each interaction between the 4 transmitters and receivers. Since the channels are flat fading, the transfer functions can be characterized by complex numbers rather than a formula. By multiplying a transfer function with a desired input signal, you get a response to running the signal through that system, also known as an impulse response. The n components represent the individual noise that both signals experience.

Noisy signals can be cleaned using filters, but it becomes more complicated when dealing with interference caused by other signals holding data that may be at the same frequency. In that case, the environment between each transmitter-receiver relationship (h_{xy}) can be calculated and used to clean up the signals and remove interfering data and noise.

The methods used in this lab take advantage of the following relationship between the transmitted data (x) and the received data (y):

$$\hat{x} = w^{\dagger} y \tag{3}$$

where w, the weight matrix, is the inverse of the conjugate transpose of the matrix H multiplied by the matrix $\binom{1}{0}$. Both methodologies take different approaches to calculating weight matrix.

1.1 Zero Forcing (ZF)

Zero forcing focuses on minimizing the peak distortion of the equalized channel. Since we are working with matrices, there needs to be some manipulation of the H matrix, but the overall the Zero-Force method is basically just dividing the output signal, y, by the transfer function, H.

The weight vector used in zero forcing can be expressed as such:

$$\begin{aligned} w &= (H^{\dagger})^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} h_{11}^* h_{21}^* \\ h_{12}^* h_{22}^* \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \frac{1}{h_{11}^* h_{22}^* - h_{12}^* h_{21}^*} \begin{pmatrix} h_{22}^* - h_{21}^* \\ -h_{12}^* h_{11}^* \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \frac{1}{h_{11}^* h_{22}^* - h_{12}^* h_{21}^*} \begin{pmatrix} h_{22}^* \\ -h_{12}^* \end{pmatrix} \end{aligned} \tag{4}$$

The reason why there is an additional matrix $\binom{1}{0}$ is because based on the relationship shown in equations 1 and 2, only data from the first terms, $h_{11}x_1[k]$ from equation 1 and $h_{21}x_1[k]$ from equation 2 are important for decoding information that is important to receiver 1 when trying to decode signal x_1 from transmitter 1. If the user wants to decode x_2 from transmitter 2, then the binomial would be $\binom{0}{1}$, which to transmitter 1 would be interference.

1.2 Minimum Mean Squared Error (MMSE)

Minimum Mean Square Error is a linear equalizer that reduces ISI while also taking into account of noise. This means that we are dividing the received signal by the transfer function in addition to a noise characterization. The weight vector can be described with the equation:

$$w = H^H (HH^H + \lambda \mathbf{I})^{-1} \tag{5}$$

Note the similarities to the zero force weight vector shown in equation 5.There is now the inclusion of $\lambda\mathbf{I}$. I deduced lambda, λ, to be the ratio of the variance of noise to variance of the original signal, which ended up just being the variance of noise, σ_n. The $\mathbf{I}$ stands for the identity matrix. In my case, I used a 2x2 identity matrix, $\binom{10}{01}$.

2 Implementation

The equalization process was performed using Matlab, where the raw data was extracted and prepared for the demodulation process. The first problem was the data was that the received data was misaligned. There was a minor delay in the rx data. In order to account for this and "sync" the received and transmitted data, I made use of the known 128 pseudo bits.

In order to determine the delay of the signal, I found the cross correlation between the known 128 pseudo bits from the transmitted signal and the full received signal. The result of this is the cross correlation between the two signals- which range from values 0 to 1. I then found the index of the highest peak in the cross correlation data, which in turn gives me the amount of lag. In the received data, there was a lag of 20 bits between the received and transmitted data.

With the lag, I was able to cut off the first 20 data points from the received data arrays and padded the end with 20 zeros in order to make the transmitted and received data have the same length.

I then plotted out the transmitted and received data as presented in Figure 2 with vertical lines added at important points on the graphs.

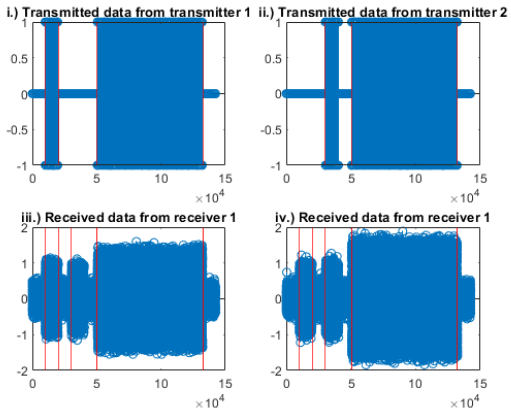

Figure 2: Transmitted Data from tx1 (i), transmitted data from tx2 (ii), received data from rx1 (iii), received data from rx2 (iv). The red lines indicated start and end times for data transmission

With my data, I created my H matrix, which represents the 4 transfer functions for the different transmitter-receiver combinations. They are oriented in the same was as equations 1 and 2. The H matrix took the form as shown below:

$$H = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{6}$$

The transfer functions were calculated by dividing the received signals by transmitted signals such that $h_{11} = \frac{y_1}{x_1}$, $h_{12} = \frac{y_1}{x_2}$, $h_{21} = \frac{y_2}{x_1}$, and $h_{22} = \frac{y_2}{x_2}$.

After all of this preparation was complete, I was able to start demodulating.

The demodulating was straightforward - I recreated the weight formulas shown in equation 4 and equation 5 for zero forcing and MMSE respectively, and then multiplied the conjugate transpose of w with signals y_1 and y_2 depending on which signal that I want to demodulate.

Once the data was filtered, I retrieved the data bits. Knowing the location of the data bits in the received data, I was able to isolate the data bits. I made 4 separate arrays: an array for the original data bits from x_1 and x_2, and the recovered data using zero-forcing and MMSE. Each array held the pulses for the 1024 data bits, making the array 40960 bits long. In order to retrieve only bits rather than the series of pulses, I created associated arrays that took samples every 40 bits, starting at 20 points after the first data pulse is transmitted, allowing for the retrieval of all 1024 data bits from the transmitted signal and the received signal.

In order to calculate error, I compared the retrieved data with the original, transmitted data. To do so, I first subtracted the original data array with element-wise, normalized bits of retrieved data. This meant that if a data point was positive in the retrieved signal, it would be normalized and converted into a positive 1, and if it was negative it would be converted to a negative 1. I then subtracted the original data array by the retrieved data array. Afterwards, I took the sum of all of the elements in the array produced from subtracted the data and divided it by the amount of elements in the arrays, 1024. This produced a fraction, which I multiplied by 100, giving me the percent of error that occurred during the data retrieval.

3 Results

3.1 Zero-Forcing Results

Below are the results from Zero-forcing equalizing:

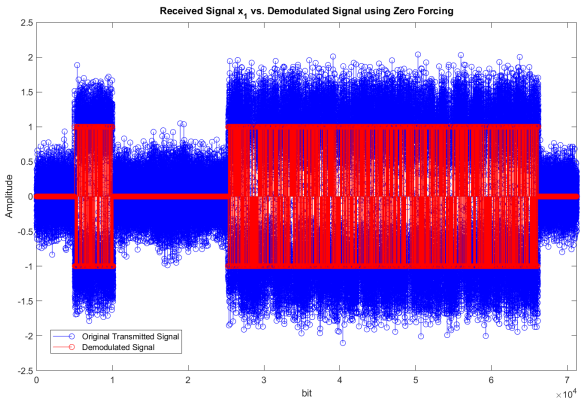

Figure 3: The retrieved signal, x_1, using a zero-forcing receiver on signal y_1

As shown in figure 3, the data was successfully demodulated while also minimizing the effects of the interfering signal x_2 from transmitter 2. I also attempted to retrieve data from transmitter 2 through the data received from signal 1. As shown in figure 4, a lot of the signal ended up becoming minimized. This is to be expected since x_2 was not intended to be sent to receiver 1 so the demodulated signal should not have many traces of the data.

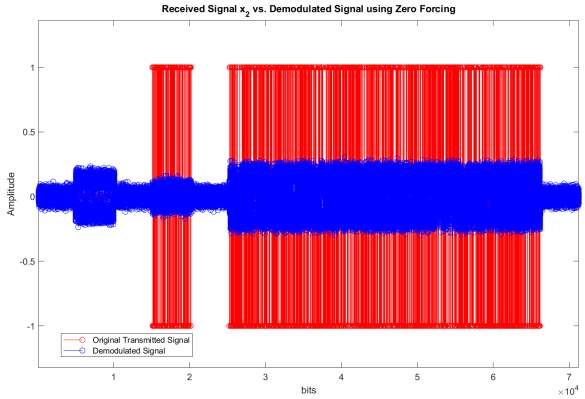

Figure 4: The retrieved signal, x_2, using a zero-forcing receiver on signal y_1

I then attempted to retrieve x_2 from receiver 2. As shown in figure 5, I was able to successfully amplify the signal x_2, however, the error rate was pretty high, which I will discuss in the error subsection.

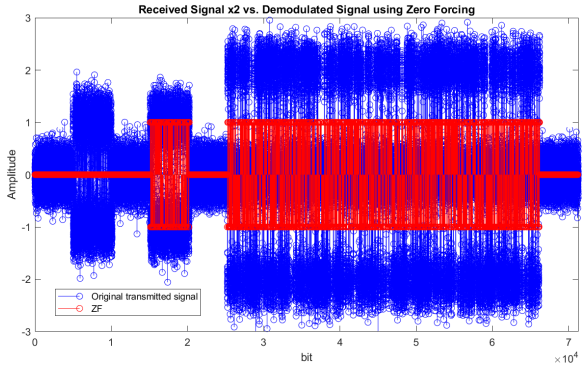

Figure 5: The retrieved signal, x_1, using a zero-forcing receiver on signal y_1

3.2 MMSE Results

Below are the results from MMSE equalizing:

I first attempted to retrieve signal x_1 from data received by receiver 1. As shown in figure 6, I was able to successfully retrieve the signal by minimizing the affects of interference signals and noise.

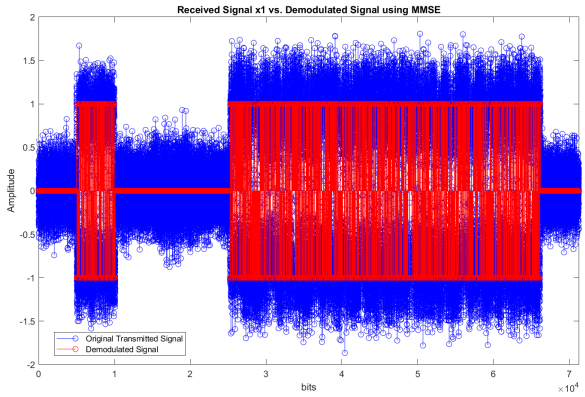

Figure 6: The retrieved signal, x_1, using a MMSE receiver on signal y_1

I then attempted to retrieve the signal x_2 from data received by receiver 2. As shown in figure 7, I was able to successfully amplify the signal, x_2, however the noise also was amplified. I am unsure why this has occured. It is similar to the results from retrieving signal x_2 from receiver 2 using zero-forcing only with MMSE there was 0 error in retrieving the 1024 data bits.

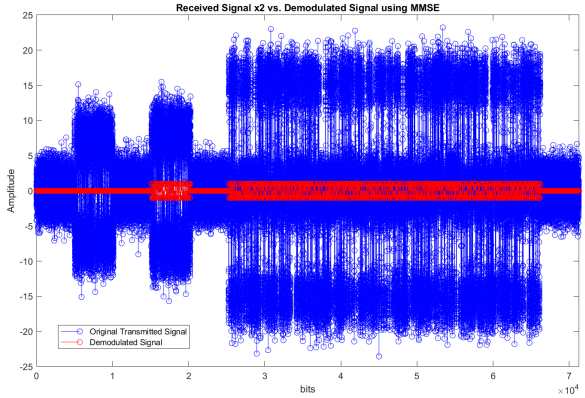

Figure 7: The retrieved signal, x_2, using a MMSE receiver on signal y_1

3.3 Error

Although the plots for the retrieved data looked messy, upon calculating error I found that I was able to retrieve x_1 using zero-forcing and and MMSE, and x_2 using MMSE with zero error. The percentage of error when attempting to retrieve signal x_2 from receiver 2 proved to be unsuccessful, producing an error of 47.65625 %. This is somewhat visible in figure 8. It is visible that the retrieval using an MMSE receiver reduced noise much more than retrieving data using zero-forcing.

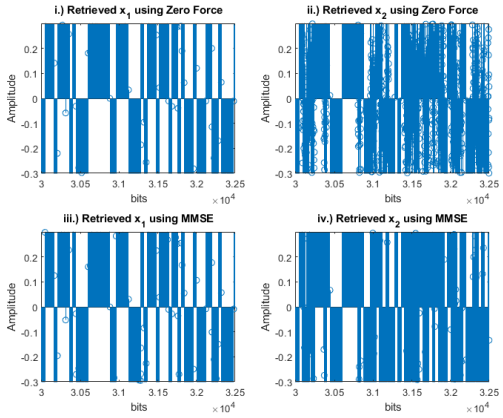

Figure 8: Zoomed plots of the retrieval of the signals: x_1 using zero forcing (i), signal x_2 using zero forcing (ii), x_1 using MMSE (iii), and x_2 using MMSE.

4 Future Work + response to if there were more transmitters and antennas

If I had additional time to work on this lab, I would try to determine why the of error for retrieving x_2 from signal was so high. It could be due to the noise amplification becoming too large that data retrieval became more difficult as compared to the x_2 retrieval using MMSE.

What if there were more antennas and transmitters In the case that there are more antennas and transmitters, the relationship between y and x shown in equation 1 can be generalized as:

$$y_A[k] = h_{11}x_1[k] + h_{12}x_2[k] + ... + h_{AB}x_B[k] + n_1[k] \tag{7}$$

where there are some A number of receivers and B number of transmitters. A is the index for the receivers and B is the index for the transmitters.

The same methodologies can be used, however, the matrices will be much larger. The main issue would be dealing with matrix multiplication and ensuring orientations are correct, but decoding is still possible, though MMSE would be a better approach than ZF as it does not amplify noise as much.

5 Reflection

After going through different sources, including scholarly papers, websites, engineering blogs, and forums, I was able to understand zero force equalizing and minimum mean square error. I understand the concept of minimum mean square error, however, there were so many different versions of the same formula for w that I did get slightly confused on what the noise component actually was.

I also am curious on why I was unable to retrieve the signal x_2 properly using both methods. When using zero-forcing, the error rate was high to a sense that it didn't make much sense since I was able to filter x_1. I am also unsure why x_2 using MMSE was amplified so much.

My lab can be viewed at:
https://github.com/Kristtiya/PrinciplesofWirelessCommunications

6 References

1. Notes, "What is MIMO Wireless Technology," Electronics Notes. [Online]. Available: https://www.electronics-notes.com/articles/antennas-propagation/mimo/what-is-mimo-multiple-input-multiple-output-wireless-technology.php. [Accessed: 23-Feb-2021].

2. https://cioffi-group.stanford.edu/doc/book/chap3.pdf

3. S. Bhagwatker, B. P. Patil, "Performance of MMSE channel equalization for MIMO OFDM system," presented at the ICCUBEA.[PDF]. Available: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf

4. J. Eilert, Di Wu and Dake Liu, "Implementation of a programmable linear MMSE detector for MIMO-OFDM," 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 2008, pp. 5396-5399, doi: 10.1109/ICASSP.2008.4518880.

5. EE359 staff, "MIMO detection algorithms," [Online]. Available: https://web.stanford.edu/class/ee359/previous/pdfs/mimo-detection-algorithms.pdf?fbclid=IwAR0pj_KZsP6fyDpdziKEkl1L6ASuW42BOZzgQJeDJVK-pcjcGNzv9asEv-I

6. Stanford University. (2008). Estimation. [Powerpoint]. Available: https://stanford.edu/class/ee363/lectu

7. M. Tuchler, A. C. Singer and R. Koetter, "Minimum mean squared error equalization using a priori information," in IEEE Transactions on Signal Processing, vol. 50, no. 3, pp. 673-683, March 2002. doi: 10.1109/78.984761