The MIMO-OFDM Project

Almost done

Jack Langner
Department of Electrical Engineering
Cooper Union
Manhattan, NY
langner@cooper.edu

***Abstract*— For the final project in ECE-408: Wireless Communications of the SARS-COV-2 stricken Spring semester of 2020, the class was given a tall task. First, the project required the students to implement a Multiple Input Multiple Output (MIMO) channel with three fading conditions. Then, the students were to implement an Orthogonal Frequency Division Multiplexing (OFDM) waveform. Once both of these were completed, the goal was to combine the two concepts and thus mimic the back bone of most wireless standards today. Furthermore, there was the challenge of meeting a specific Bit Error Rate (BER), which would require additionally techniques to be applied for the desired throughput.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

SECRET it is not that in ECE-408: Wireless Communications an array of wireless standards have been discussed. This plays to the strengths of the instructor as he does this for a living and gives the students more exposure to the world of wireless communications. A topic of great importance in this realm is Multiple Input Multiple Output (MIMO) as it allows many users to communicate simultaneously, which is exceedingly useful as there are more devices attempting send and/or receive data. Additionally, as technology has advanced, it is possible to implement Orthogonal Frequency Division Multiplexing (OFDM) waveforms, allowing for fast and efficient use of the spectrum during communication. Given how widespread these concepts have become, it is only natural that the two would be implemented in tandem so that users can receive reliable service regardless of any other users that may be present. For this reason, MIMO OFDM has become the backbone of major standards, such as 802.11 (WiFi), and is continuing to be used in the next generation of wireless.

As a note, the original intent of this project was to increase the data rate through a flat fading MIMO channel with a target Bit Error Rate (BER). As I progressed through the project, I realized that it was difficult to achieve a a certain BER within a fading channel. So, my approach is now to highlight how changing the Forward Error Correction (FEC) and constellation size effect the BER because it does not matter if you transmit 54 Mbps if 50% of the information is wrong.

Identify applicable funding agency here. If none, delete this.

Following Professor Keene's outline of the project, section II covers the MIMO portion of the project, section III is concerned with OFDM and its performance in a flat fading channel. This is followed up by the heart of the project where MIMO and OFDM are combined in section IV where it will be shown how the two techniques work together. Finally, in section V there is a discussion of some of the choices I made and can be done differently.

II. 2× 2 MIMO

Earlier in the semester, the class was introduced to MIMO channels through the implementation of Alamouti codes. Here one form of Space-Time Block Codes (STBC) were explored as a way to combat the effects that a flat fading channel introduce. Using STBC, no knowledge of the channel needs to be assumed prior to data transmission and the receiver needs to be set up in such a way to be able to take advantage of the STBC. The only assumption is that the channel is slowly changing such that two successive samples on the same transmit path see the same channel coefficient. Another method of trying to undo the channel requires knowledge of the channel at the transmitter and/or the receiver. Perfect knowledge of the channel at all times is an unrealistic assumption, but estimating the channel periodically is not absurd. Through the use of preambles, prefixes, pilot tones, and so on it is not unreasonable to estimate the channel frequently, especially at the receiver.

In this section there will first be a brief overview of MIMO systems in section II-A. The following three sections, II-B, II-C, and II-D, three different channel filters are described to demonstrate how they can be effective in a MIMO environment. Finally, in section II-E results from simulations are given and discussed.

A. MIMO Overview

In a MIMO system the goal is to support multiple spatial streams, which can be thought of as users vying for data or as a way to beat the effects of a flat fading channel, which is known as diversity. In the general MIMO system the number of transmit antennas can be denoted by N_T, the number of receive antennas by N_R, and the total number of paths is $N_T N_R$. Fig. 1 shows how a 2 × 2 MIMO system would be set up, where at each time step both TX_1 and TX_2

are transmitting and RX_1 and RX_2 are receiving a linear combination of the transmitted data. To be more precise, the

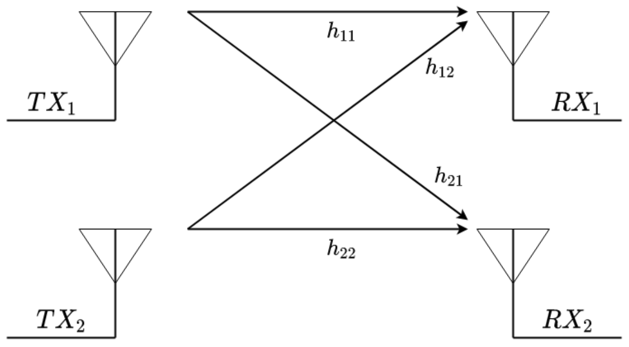


Fig. 1. 2 × 2 MIMO Sytem

received signals are modeled as

$$\begin{aligned} y_1 &= h_{11}x_1 + h_{12}x_2 + n_1 \\ y_2 &= h_{21}x_1 + h_{22}x_2 + n_2 \end{aligned}$$

or equivalently using matrix notation

$$\begin{aligned} y &= Hx + n \\ H &= \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \end{aligned}$$

where n represents a vector of independent Gaussian noise and H is used to represent the channel matrix which consists of 4 samples from a Rayleigh distribution. A MIMO channel is only as good as the way in which it handles H, so for this reason different approaches assume H varies slowly in time. In these experiments, H changes at every sample time which is realistic, however the methods used here to combat the ever changing H require complete knowledge of the channel either at the transmitter and/or the receiver at all times.

B. Pre-Coding

The first method to be discussed which handles H is referred to as pre-coding. In this scheme the Singular Value Decomposition (SVD) is computed for each H to give the following

$$H = U\Sigma V^H$$

As a note, H is $N_R \times N_T$, U is $N_R \times N_R$, Σ is $N_R \times N_T$ with the non-negative singular values of H on the main diagonal, V is $N_T \times N_T$, and $(\cdot)^H$ represents the Hermitian transpose of the matrix. Furthermore, both U and V are unitary, so multiplying by the Hermitian transpose results in the identity matrix of the appropriate size. Using this information, if x is multiplied on the right by V and y is multiplied on the right by U^H, the received vector can be simplified as follows

$$\begin{aligned} \tilde{x} &= Vx \\ \tilde{y} &= U^H y \\ &= U^H(H\tilde{x} + n) \\ &= U^H(U\Sigma V^H V x + n) \\ &= \Sigma x + \tilde{n} \end{aligned}$$

Since U is unitary, the power in n and $\tilde{n}$ will be same. Additionally, since Σ is a diagonal matrix, it would seem to make sense to multiply by Σ^{-1} to perfectly recover x, however some of the elements of Σ are less than 1, so then multiplying by Σ^{-1} will act to increase the noise power. Therefore, this is as good as pre-coding can using its knowledge of the channel at both the transmitter and receiver.

C. Zero Forcing

The second method that is used to combat the channel is called Zero Forcing. Using this scheme the pseudoinverse, or Moore-Penrose inverse, of the channel matrix is applied to the received signal vector. The zero forcing filter is defined by

$$\begin{aligned} W_{ZF} &= H^\dagger = (H^H H)^{-1} H^H \\ \tilde{y} &= W_{ZF} y = W_{ZF}(Hx + n) \\ &\approx x + W_{ZF} n \end{aligned}$$

Here the use of $\approx$ on the last line comes from the fact that the Moore-Penrose inverse is a minimum norm solution to $y = Hx$, so by applying $H^\dagger$, x may not be recovered exactly.

D. Minimum Mean Square Error

The third (and final) approach that is used to mitigate the effect of the channel is called the Minimum Mean Square Error (MMSE). The MMSE filter has a similar feel as the zero forcing approach. However, MMSE includes the variance of the noise within the calculation of the filter to try to account for the issues arising from the additive noise. The filter's calculation and application are as follows

$$\begin{aligned} W_{MMSE} &= (H^H H + \sigma_n^2 I)^{-1} H^H \\ \tilde{y} &= W_{MMSE}(y) \\ &= W_{MMSE}(Hx + n) \end{aligned}$$

The two additions are σ_n^2 and I which represent the variance of the noise at the n^{th} receiver and the identity matrix with dimensions $N_T \times N_T$, respectively. MMSE appears to offer an improvement over Zero Forcing as the noise power (variance) is accounted for in the matrix inversion.

E. MIMO Results

A new interpretation has come to light where channel models are represented as multipath components. So in the simulations, there are three channel models that include Rayleigh fading. These models are a channel with no multipath components, one multipath component $(1 + 0.1z^{-1})$, and a channel with two multipath components $(1+0.1z^{-1}+0.2z^{-2})$.

These channels will be denoted 0, 1, and 2 respectively. Additionally, the Quadature Amplitude Modulation (QAM) order is chosen from $\{2, 4, 16\}$, meaning there are 1,2, or 4 bits per symbol, respectively. The following tables list the BER obtain using the different filters.

NF		Modulation Order		
		2	4	16
Channel	0	0.2530	0.3348	0.3981
	1	0.2528	0.3352	0.3989
	2	0.2539	0.3398	0.4006

Fig. 2. No Filter

PC		Modulation Order		
		2	4	16
Channel	0	0.0132	0.0184	0.1986
	1	0.0346	0.0364	0.2119
	2	0.0724	0.0733	0.2446

Fig. 3. Pre Coding

ZF		Modulation Order		
		2	4	16
Channel	0	0.0180	0.0250	0.0524
	1	0.0455	0.0496	0.1006
	2	0.1017	0.1045	0.1951

Fig. 4. Zero Forcing

MMSE		Modulation Order		
		2	4	16
Channel	0	0.0110	0.0159	0.0892
	1	0.0290	0.0293	0.1090
	2	0.0837	0.0768	0.1775

Fig. 5. Minimum Mean Square Error

In the above NF stands for no filter. We see that in all three cases, the filter applied helps lower BER, so the additional work is worth the effort. Additionally, as the constellation size increases, the BER increase, therefore a trade off between BER and data rate has been observed.

III. OFDM

Up to this point in the semester, data has been transmitted serial. This means that a single carrier frequency is used to upconvert a single symbol. From a learning persepective this is easy to understand and implement, but in the real world this would limit the data rate too much. Furthermore, using this serial transmission scheme, the data is very susceptible to be corrupted by the fading of the channel or the additive noise. These are some of the reasons that Orthogonal Frequency Division Multiplexing (OFDM) has become popular as it allows for transmission on multiple subcarriers. This spreading over frequency also allows the signal to be shorter than the coherence bandwidth of the channel, meaning that flat fading can be safely assumed. For this project, the OFDM waveform being implemented was supposed to be based on the 802.11 standard waveform, but there was an issue understanding that implementation, so it was decided that recreating MATLAB's comm.OFDMModulator object was sufficient.

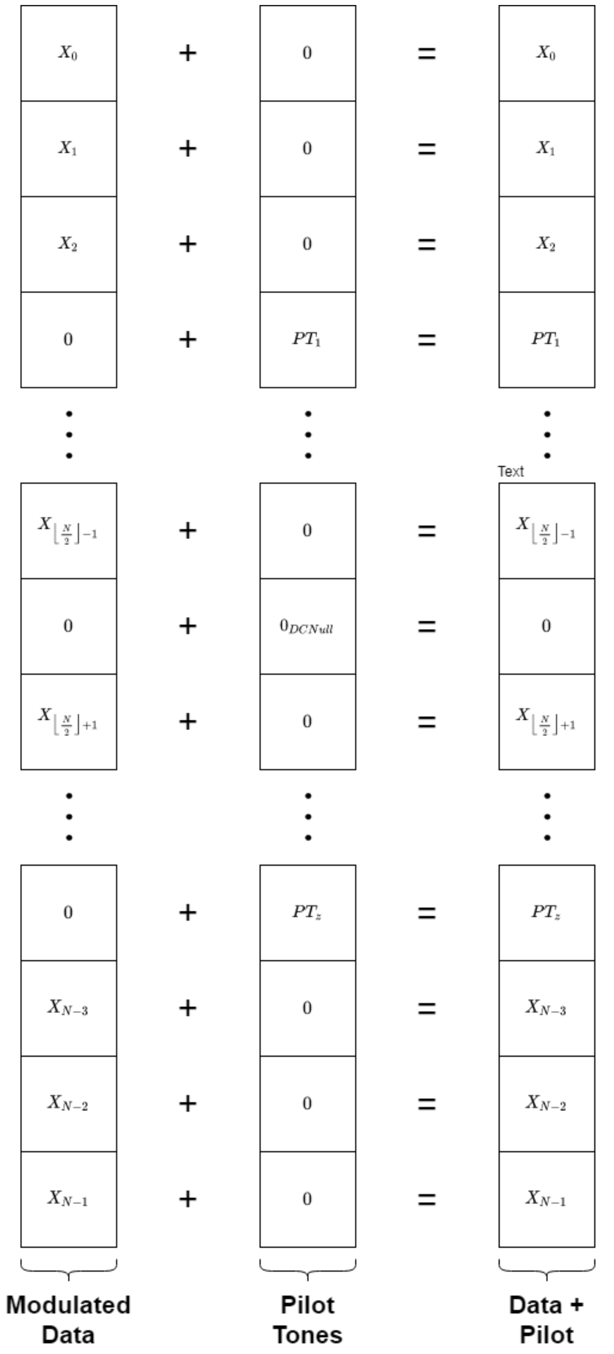
The process to OFDM modulate described here follows the 802.11 standard as it builds in an interleaver and forward error correction, which help to improve system performance. In terms of the project at hand though, these details are not the concern but are included to assist in in the OFDM BER performance. First, the amount of data that is going to be transmitted needs to be determined along with the error correction rate and constellation size. This allows the total number of bits, and therefore total number OFDM symbols, to be determined. Once this is done, a convolutional coder is used to encode the data, the data is broken up into frames, and each frame is interleaved. The interleaved data is then mapped onto the constellation being used, note that constellation refers to the order of Quadature Amplitude Modulation (QAM) that has been selected. After the data is modulated according the QAM constellation, the pilot tones and DC null are inserted to each frame. Fig. 6, which follows, shows how one can think of inserting the pilots and DC null into the data.

Then two null pads are added at both ends of every frame, this is done so that the signal does not spread too much in frequency and also brings the frame up to the appropriate size for the IFFT. As shown in Fig. 7 each frame gets the two null pads which helps to limit spreading in the frequency domain. Here there are a total of J OFDM frames.

At this point, each frame is reordered according to the *ifftshift* function in MATLAB and then an IFFT with a specific number of points is computed. After the IFFT, a cyclic prefix is added, in which the last G samples are copied to the front of the frame.

The cyclic prefix is also helpful as it can be used for timing offset. When a burst is initially received, by calculating the correlation of the beginning and end of the frame, the start of a symbol can be found. Finally, a window function is applied to the domain samples and then the frames are lined up so that the last sample from a frame lines up with the first sample in the next frame and the two samples are added together. This is done so that the symbols appear to be continuous to the transmit antenna. The process of lining up the frames is shown in Fig. 9 and how the result is a continuous stream of OFDM symbols.

The above describes a general process. The actual implementation consisted of 48 data subcarriers, 4 pilot tones, and the DC null, so 53 points of the IFFT are accounted for immediately. Then 2 pads consisting of only 0's of length 6 and 5, respectively, are added to the left and right of each set of 53 frequency domain symbols. This brings the total



X_0 + 0 = X_0
X_1 + 0 = X_1
X_2 + 0 = X_2
0 + PT_1 = PT_1
Text
$X_{\lfloor \frac{N}{2} \rfloor -1}$ + 0 = $X_{\lfloor \frac{N}{2} \rfloor -1}$
0 + 0_{DCNull} = 0
$X_{\lfloor \frac{N}{2} \rfloor +1}$ + 0 = $X_{\lfloor \frac{N}{2} \rfloor +1}$
0 + PT_z = PT_z
X_{N-3} + 0 = X_{N-3}
X_{N-2} + 0 = X_{N-2}
X_{N-1} + 0 = X_{N-1}
Modulated Data
Pilot Tones
Data + Pilot

Fig. 6. Inserting Pilots

number of frequency domain symbols per frame up to 64 and then a 64 point IFFT is used bring the data into the time domain. Afterwards, the last 16 samples are added in front of the frame to act as the cyclic prefix, so each frame consists of 80 time domain samples. The window function is derived from $\sin^2(t)$ with samples at $x \in \{\pi/4, \pi/2, 3\pi/4\}$, so the values of the function are 0.5 and 1. After the appropriate overlapping and addition of time domain samples is done, a single stream of time domain samples is obtained and it will have length $N_{sym} \cdot (80-1)+1$ where N_{sym} is the total number

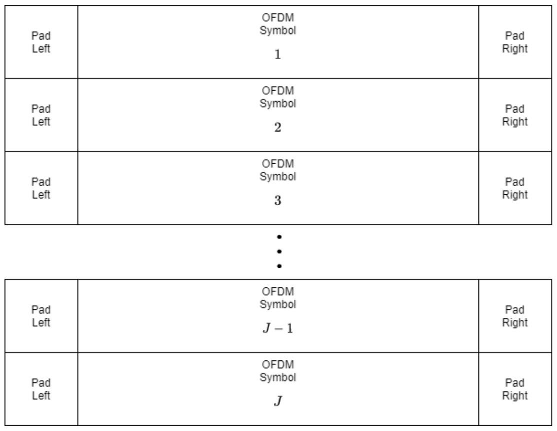


Fig. 7. Addition of Null Pads

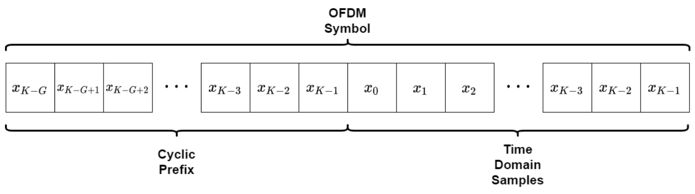


Fig. 8. Time Sample Cyclic Prefix

of OFDM symbols generated.

In order to better illustrate the window function and why it makes sense to overlap with this window function, one period of the function along with the overlap are shown.

As can be seen in Fig. 11 the minimum value where two consecutive windows overlap is 0.9992 which is very close to 1, so there is essentially no lose. Explicitly, this would correspond to a lose of 0.0035 dB.

OFDM results here.

IV. 2×2 MIMO OFDM

The third part of the project combines the previous two, meaning that a 2×2 MIMO system was implemented where the transmitted signals are OFDM modulated. This is meant to simulate the backbone of most modern wireless communication standards. Having implemented the MIMO system with filters and OFDM separately, combing the two ideas was straight forward. Using the structure of the MIMO functions, the OFDM modulation was inserted where the random data

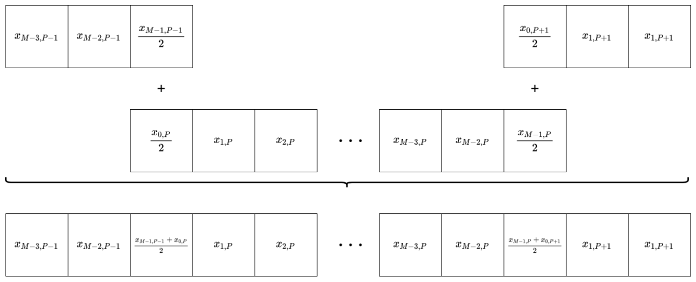


Fig. 9. Frame Overlap

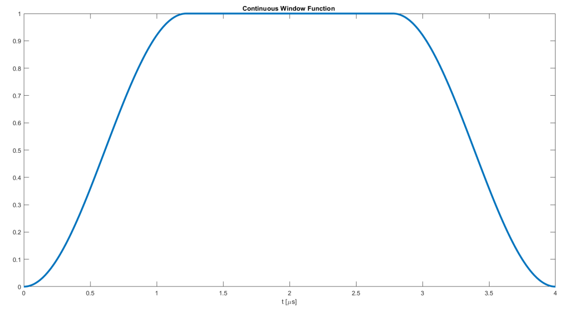


Fig. 10. Window Funtion, $\sin^2(t)$

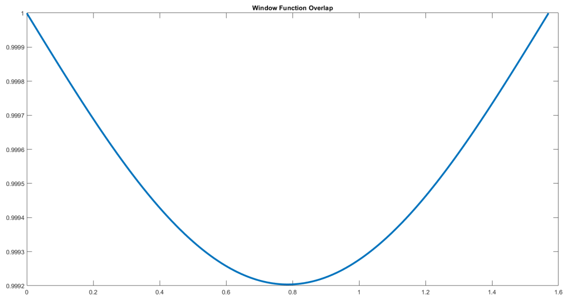


Fig. 11. Window Function Overlap

was previous generated, then the OFDM data streams were transmitted across the channel, and then demodulated on the receiver side. To determine how effective data transmission is in this setting, the BER for both transmitted data streams are reported as TX1 and TX2.

No Filter			Modulation Order(Coding Rate)					
			2(1/4)	4(1/4)	2(1/3)	4(1/3)	2(1/2)	4(1/2)
TX1	Channel	0	0.3484	0.4158	0.3850	0.4211	0.4748	0.4853
		1	0.3543	0.4118	0.3820	0.4174	0.4743	0.4815
		2	0.3665	0.4180	0.3831	0.4246	0.4828	0.4918
TX2	Channel	0	0.3478	0.4138	0.3755	0.4141	0.4788	0.4854
		1	0.3592	0.4149	0.3877	0.4201	0.4804	0.4839
		2	0.3564	0.4201	0.3928	0.4268	0.4864	0.4821

Fig. 12. MIMO-OFDM No Filter

Pre Coding			Modulation Order(Coding Rate)					
			2(1/4)	4(1/4)	2(1/3)	4(1/3)	2(1/2)	4(1/2)
TX1	Channel	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
		1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
		2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TX2	Channel	0	0.0003	0.0026	0.0013	0.0322	0.0033	0.1303
		1	0.0000	0.0049	0.0015	0.0470	0.0037	0.1909
		2	0.0013	0.0287	0.0108	0.1109	0.0421	0.3368

Fig. 13. MIMO-OFDM Pre Coding

Again, we see the trade off in BER and Modulation order, and that using any of the three methods to combat the channel increases performance. New to the mix is the error correcting code which helps most notably on TX1 for Pre coding no

Zero Forcing			Modulation Order(Coding Rate)					
			2(1/4)	4(1/4)	2(1/3)	4(1/3)	2(1/2)	4(1/2)
TX1	Channel	0	0.2861	0.3753	0.3213	0.3884	0.4087	0.4618
		1	0.3248	0.3915	0.3504	0.4082	0.4368	0.4840
		2	0.4019	0.4497	0.4220	0.4619	0.4893	0.4993
TX2	Channel	0	0.2911	0.3758	0.3210	0.3887	0.4086	0.4605
		1	0.3228	0.3927	0.3546	0.4197	0.4349	0.4726
		2	0.4041	0.4457	0.4210	0.4614	0.4854	0.5011

Fig. 14. MIMO-OFDM Zero Forcing

MMSE			Modulation Order(Coding Rate)					
			2(1/4)	4(1/4)	2(1/3)	4(1/3)	2(1/2)	4(1/2)
TX1	Channel	0	0.0005	0.0263	0.0106	0.0898	0.0253	0.2480
		1	0.0033	0.0676	0.0252	0.1457	0.0860	0.3507
		2	0.0743	0.2676	0.1763	0.3260	0.3663	0.4752
TX2	Channel	0	0.0011	0.0273	0.0106	0.0930	0.0288	0.2512
		1	0.0029	0.0696	0.0254	0.1468	0.0809	0.3401
		2	0.0767	0.2703	0.1734	0.3211	0.3672	0.4776

Fig. 15. MIMO-OFDM MMSE

error were seen during the simulation on any channel, so that's pretty cool.

V. CONCESSIONS

Here I briefly describe some of the things I did implement, but ultimately decided to use a MATLAB function instead. This is mainly due for ease of integration. The first thing that comes to mind is not implementing the OFDM waveform as described in the 802.11 standard, there was a serious attempt but there seemed to be multiple definitions within the standard and when checking an example provided within 802.11 I still could not figure it out. This is why I opted to recreate the comm.OFDMModulator object as I could easily check that I was producing the same output as Mathworks' object and know that it was something that would realistically be used in an OFDM system. Also, I apologize for not addressing timing offset, I know how to do handle it and what to look for in the received signal, but the time simply disappeared and handing in something is better than nothing.

Referring back to section II-A the matrix representation of the received signal in the 2× 2 MIMO is more accurately given by

$$y(t+\Delta t) = H(t)x(t) + n(t)$$

which shows how the channel matrix, transmitted signal, and noise change at all times. Furthermore, the Δt represents the propagation delay and is another aspect of the MIMO system that needs to be accounted for in the design.

Also, only using Rayleigh fading for the three channels is not ideal, but this project was about implementing MIMO-OFDM and not figuring out how to generate complex random variables such that the magnitude follows a Rician distribution.

Wish I simulated for longer to generate tables, but I wanted to finish. I am sorry.

ACKNOWLEDGEMENTS

For the last time in this class I would like to thank Karol Wadalowski. Even though he did not help me with this project,

I feel it necessary to praise him behind his back because he is one of the nicest people I know and I hope him nothing but the best in the future.