Principles of Wireless Communications Lab 3(a): OFDM

Mark Goldwater and Anusha Datar

March 23, 2021

1 Introduction

Orthogonal frequency-division multiplexing (OFDM) is a type of digital transmission that encodes data on multiple carrier frequencies. In other common modulation schemes such as binary phase-shift keying (BPSK), quadrature amplitude modulation (QAM), and $N \times M$ multiple input multiple output (MIMO) one assumes a flat-fading channel model. A flat-fading channel model can be assumed when the coherence bandwidth of the channel (where the channel can be assumed to be essentially flat) is significantly larger than the bandwidth of the transmitted signal, and allows one to represent the channel with the multiplication of a single complex coefficient. Figure 1 visually illustrates the comparison between a clean and messy channel.

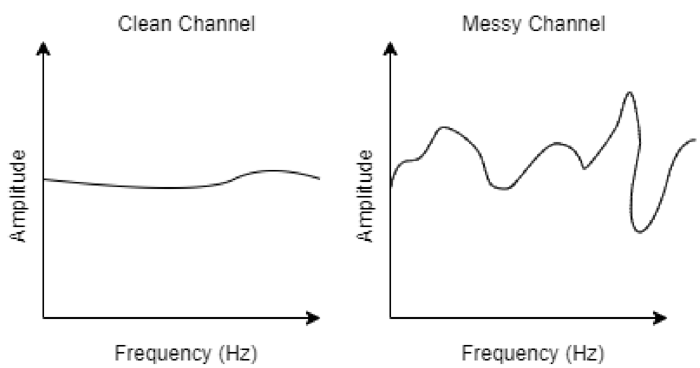

Figure 1: Comparison of a clean channel to a messy channel.

The frequency response of the channel shown in the above left diagram would cooperate nicely with a flat-fading model. Because the amplitude of the channel is essentially flat, its effects can be well-approximated with a single complex coefficient. However, the channel shown on the right in Figure 1 is significantly more "messy" in that it is not flat for any meaningfully long duration of time. As a result it is difficult to accurately approximate this channel using a single complex coefficient with the flat-fading channel model.

This issue is what OFDM tries to address by splitting up the channel frequency band into many smaller (effectively flat) bands for which the flat-fading model applies. The exact details of how this is implemented are described in subsequent sections of the report. This lab is based on achieving three distinct milestones:

1. Simulate OFDM assuming timing synchronization
2. Simulate OFDM, and perform timing synchronization
3. Imlement OFDM and timing synchronization on USRP B210 radios

1.1 Software Overview

Our simulation uses a single a main function - **simulate_with_synchronized_clocks.m**. As illustrated in Figure 2, this function calls modular helpers to estimate the channel matrix, encode the data, send data through the channel, correct for lag, and compute the error. All project code and associated documentation is available at `https://github.com/anushadatar/ofdm-implementation`. Each of the bolded functions in Figure 2 has its own file in this repository. Figure 2 below shows the flow of operations for channel estimation and data transmission.

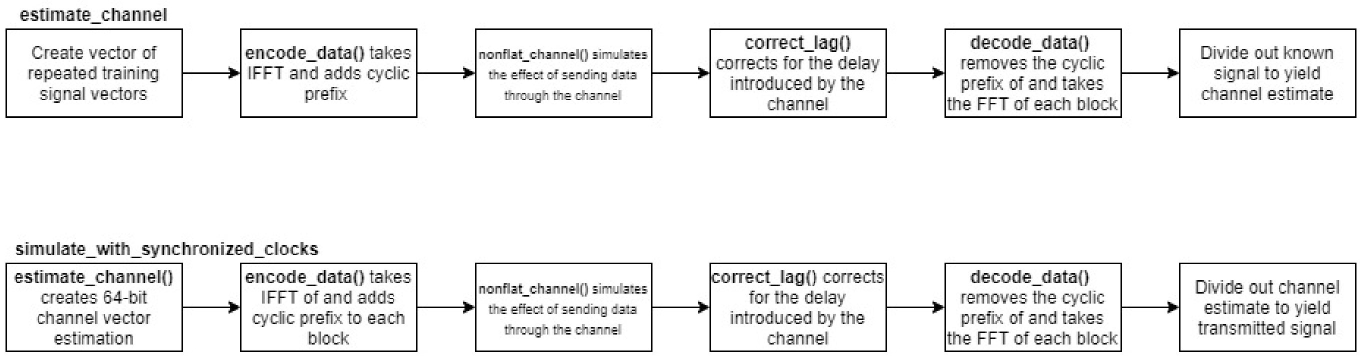

Figure 2: Software flowchart.

Below are the declarations for and a short description of each of the functions used in the diagram above. Note that the **estimate_channel** function calls many of the same helper functions as **simulate_with_synchronized_clocks**.

1. **[H_k] = estimate_channel(x_train, block_size, prefix_size)** calculates a 64-bit estimate of the channel based using a transmission vector.

2. **[x_cyclic] = encode_data(x_data, block_size, prefix_size)** takes the IFFT of and adds the cyclic prefix to each block.

3. **[y_time] = nonflat_channel(x_cyclic)** simulates the effect of sending data through the channel.

4. **[y_time] = correct_lag((x_cyclic, y_time)** corrects for the delay introduced by the channel.

5. **[y_decoded] = decode_data(y_time, number_of_blocks, block_size, prefix_size)** removes the cyclic prefix of and takes the FFT of each block.

6. **[err] = compute_error(rx_data, tx_data)** computes the percent error between the transmitted and decoded signals.

1.2 Hardware Setup

Will be done in section c.

2 Mathematical Background

2.1 Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT)

2.1.1 Transformation Background

The Fourier transform is a mathematical transformation that decomposes functions of time into functions that depend on frequency, and does so by projecting the function onto different complex exponentials. For example, consider the function $f(t)$ in the time domain, that is projected onto a complex exponential that is oscillating at a frequency of 50 Hz. To accomplish this, one must use the inner product of $f(t)$ with the specified complex exponential (in the form $e^{-j2\pi ft}$) as follows.

$$< f, e^{-j2\pi(50)t} >= \int_{-\infty}^{\infty} f(t)e^{-j2\pi(50)t}dt \tag{1}$$

Note that the output of the above equation will be a complex number. However, in order to complete the full continuous time Fourier transform (CTFT), one must project $f(t)$ onto complex exponentials of every frequency. To accomplish this one can take the inner product of functions above and parameterize it by frequency by substituting an f for 50. This produces the following equation

$$F(f) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ft}dt \tag{2}$$

which is the continuous time Fourier transform of $f(t)$ and is denoted by $F(f)$ as it is now a function of frequency.

However, to practically apply the Fourier transform in this lab and in general, it needs to be discretized so that computers (whose digital hardware is fundamentally discrete) are able to use it to process data. In undergraduate courses, it is common to learn about the discrete-time Fourier transform (DTFT) which transforms the discrete-time signal $x[m]$ of length L as follows.

$$X(e^{j\Omega}) = \sum_{l=0}^{L-1} x[l]e^{-j\Omega l} \tag{3}$$

Note that taking the L-point DTFT is permitted due to the assumption that $x[m] = 0$ for $m < 0$ and $m \geq L-1$. In the above equation, the input parameter specifies a complex exponential that is **continuous in frequency and discrete in time**, hence the name discrete-time Fourier transform. Another important note is that because time is discrete in the DTFT, a sum, which is effectively the discrete version of an integral, is used.

Finally, to fully discretize the CTFT, one can can sample the DTFT to produce the discrete Fourier transform which is shown below and can be calculated using the $fft()$ function in MATLAB.

$$X_k = \sum_{l=0}^{L-1} x[l]e^{-2\pi j\frac{lk}{L}} \tag{4}$$

The inverse discrete Fourier transform (IDFT) is also fairly simple to calculate as one can switch the positions of $x[m]$ and X_k, flip the sign on the complex exponential, and divide the summation by the length of the signal as shown below.

$$x[m] = \frac{1}{L}\sum_{k=0}^{L-1} X_k e^{2\pi j\frac{mk}{L}} \tag{5}$$

In the IDFT, the frequency domain signal is essentially projected onto complex exponentials with frequencies of opposite signs (which means that they are rotating the opposite direction of the complex plane) and normalized by the length of the signal. The DFT and IDFT are crucial to the OFDM modulation scheme. In addition to being discrete in frequency (which is necessary given that one signals in the frequency domain when using OFDM), these transformations exhibit properties useful for OFDM which are explained in the subsequent section.

2.1.2 Properties

The DFT has several properties that are especially useful in the context of implementing OFDM systems.

1. **Linearity**
   In the case that $y[m] = \alpha p[m] + \beta q[m]$, the DFT of $y[m]$ is related to the DFTs of $p[m]$ and $q[m]$ by:

$$Y_k = \alpha P_k + \beta Q_k \tag{6}$$

2. **Periodicity**
   The L-point DFT of a signal is periodic with period L. In other words, If X_k is the L-point DFT of $x[m]$, then:
   $$X_{k+L} = X_k \tag{7}$$

3. **Circular convolution in time becomes multiplication in frequency**
   Circular convolution of two discrete signals with period L corresponds to a multiplication of those signals' DFTs. More specifically, one can define the L-point periodic extension of $x[m]$ as follows:
   $$\widetilde{x}[m] = x[m\%L] \tag{8}$$
   If X_k and H_k are the DFTs are $x[m]$ and $h[m]$ (assuming that both DT signals are zero when $m < 0$ and $m \geq L-1$), then if $y[m] = \widetilde{x} * h[m]$ and Y_k is the L-point DFT of $y[m]$ is
   $$Y_k = H_k X_k \tag{9}$$
   This implies that in the case of OFDM, one can model the channel as a discrete signal and then divide out its value to find X_k if the channel performs circular convolution on the transmitted signal. How this is carried out is discussed later.

2.2 Channel Estimation

2.2.1 Continuous-Time Introduction

Just like with other modulation schemes used in communications systems, one needs to perform some sort of channel estimation. To estimate the channel for OFDM, a similar approach to MIMO channel estimation is used where a training signal (which is known by both the transmitter and receiver) is sent over the channel. However, OFDM is used in cases when the channel is "messy" as previously discussed, so it is necessary to calculate a higher-resolution estimate of the channel rather than a single complex coefficient to use under the assumption of the flat-fading channel model. In the implementation of OFDM for this lab, the "messy" channel is split into 64 small slivers which are constant enough in magnitude to apply the flat-fading channel model.

First, one can assume that the data to be transmitted over the OFDM channel (± 1 for the BPSK data streams in this lab) are signaling in the frequency domain. An example of "1" encoded as a pulse in the frequency domain is shown below in Figure 3 where f_Δ is the spacing between sub-carriers which target different locally flat regions of the overall non-flat channel.

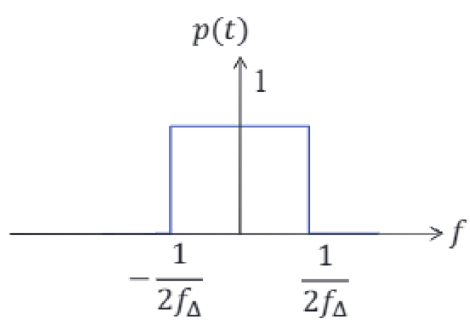

Figure 3: One bit of data signaled in the frequency domain [1].

Assuming the training signal to be sent is x_{train}, k is the discrete-time index, l denotes which sub-carrier is being used, p is a pulse function, and a_{kl} is the bit encoded at time k on sub-carrier l, the training signal can be represented in continuous time with the following equation.

$$x_{\text{train}}(t) = \sum_{k=-\infty}^{\infty} \sum_{l=0}^{63} a_{kl} p(t-kT) e^{j2\pi l t f_\Delta} \tag{10}$$

Note that this setup requires sending data over the channel in a 64-bit block. The next step is where the magic of OFDM happens, is why one encodes data used in OFDM in the frequency domain, and is the core of what inspires its name. To prepare the encoded data to send over the channel one must take its IDFT when implemented digitally to transform it into the time domain. For the sake of this explanation, consider the CTFT of the 64 sub-carriers at the zeroth symbol (where $k=0$ in Equation 10). First note that Equation 10 when $k=0$ is as follows

$$x_{\text{train},0}(t) = \sum_{l=0}^{63} a_{0l} p(t) e^{j2\pi l t f_\Delta} = p(t) \sum_{l=0}^{63} a_{0l} e^{j2\pi l t f_\Delta} \tag{11}$$

First, note that the summation in Equation 11 closely resembles the IDFT. Thus what is transmitted is the IDFT of our discrete pulses. However, to discover how the channel "sees" the data, one must take the Fourier transform of the transmitted data. After taking the CTFT of Equation 11, the following (as shown in 4) is obtained. This equation includes sinc functions lined up as shown in Figure 11 below.

$$X_{\text{train},0}(f) = \left(\frac{sin(\pi f \frac{1}{f_\Delta})}{\frac{\pi f}{f_\Delta}} \right) * \left(\sum_{l=0}^{63} a_{0l} \delta(f - l f_\Delta) \right) \tag{12}$$

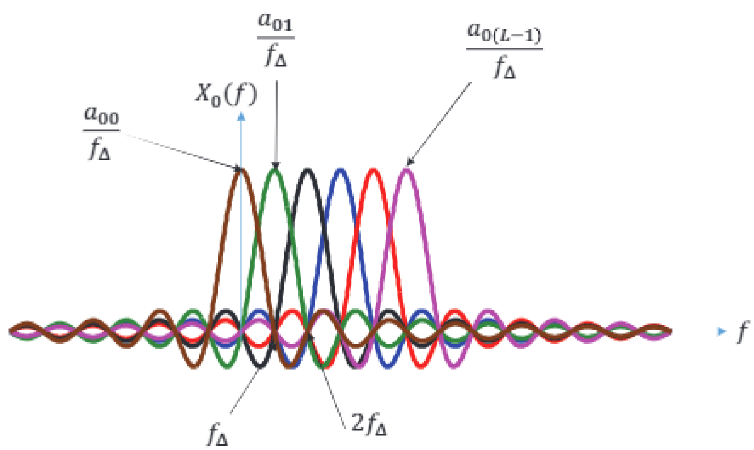

Figure 4: The first symbol of our OFDM signal prepared to be sent across the channel [1].

Note that $L = 63$ in the Figure 4 above. The main peak of each sinc function aligns with the zeros of all the others. This is where the "orthogonal" comes from in the name OFDM. If f_Δ is small enough, than each of the sinc sub-carriers can be considered a narrow-band signal (because the main lobe of each is where the majority of the power is of each sinc function).

At this point it may seem logical to send the above signal over the channel and use the fact that the receiver also knows x_{train} to solve for the channel coefficient that is applied to each sub-carrier. However, when implementing OFDM digitally, because signaling occurs in the frequency domain, the DFT and IDFT are used which required a few extra steps to be taken to account for the circular convolution discussed above. In other words, one must account for the fact that, when using the DFT/IDFT, multiplication in the frequency domain is not convolution in the time domain, but rather circular convolution.

2.2.2 Discrete Time Implementation

Starting from the beginning, this time in discrete time, the steps to carrying out OFDM are as follows:

1. Create a 64-bit training vector to transmit across the channel. Each bit here translates to one subcarrier in an OFDM symbol.

$$X_{k,\text{train}} = \begin{bmatrix} X_0 & X_1 & X_2 & \dots & X_{63} \end{bmatrix} \tag{13}$$

2. The training data vector created above is signaling in the frequency. One must take the 64-point IDFT of the x_{train} to transform it into the time domain so that orthogonal sinc functions can divide the frequencies of the channel into small flat bands so that one can derive a channel coefficient for each of the narrow-band sub-carriers.

$$x_{k,\text{train}}[m] = IDFT\{X_{k,\text{train}}\} = \begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_{63} \end{bmatrix} \tag{14}$$

3. Here is where the discrete implementation differs greatly, for **circular convolution** in the time domain is multiplication in the frequency domain. Mathematically, to perform a circular convolution in discrete time, one takes the 64-bit vector and line them side-by-side infinitely which is expressed as follows

$$\bar{x}_{k,\text{train}}[m] = x_{k,\text{train}}[m\%64] \tag{15}$$

Given the periodic extension of our time-domain 64-bit block in Equation 15, the equation $y[m] = \bar{x}[m] * h[m]$ describes the periodic extension being transmitted through a channel via circular convolution. Given that Y_k, X_k, and H_k are the DFTs of their respective signals, the following equation is also true.

$$Y_k = H_k X_{k,\text{train}} \tag{16}$$

However, the caveat here is that when sending an infinite signal over the channel, it will take an infinite amount of time to send. Thus, the transmission rate is effectively $0\frac{\text{bits}}{\text{second}}$.

The way to solve this problem is to append just enough of the end of the signal to the front of the data to "trick" the channel into performing circular convolution. Thus, one must define $\bar{x}_{k,\text{train}}$ not to be an infinite periodic extension, but a periodic extension that adds the last 16 samples of a 64-bit block (known as the cyclic prefix) to its front before sending. This makes the transmitted data 80 samples.

$$x_{k,\text{train}} = \begin{bmatrix} x_{48} & x_{49} & \dots & x_{63} & x_0 & x_1 & x_2 & \dots & x_{63} \end{bmatrix} \tag{17}$$

Additionally, the OFDM standard assumes that the impulse response of the channel is 16 samples long, meaning that if one appends the last 16 samples of a 64-bit block to its beginning, one can to line up many blocks with a cyclic prefix to send in one large block. The reason for this is that there will be a point, during the convolution with the channel, when the impulse response of the channel is completely within the cyclic prefix. As a result, there will be no cross-pollination between OFDM data blocks. So as a final step, 100 of these cyclically prefixed training signals are lined up to send at once so that the final channel estimation can be an average of the estimation calculated from each of the known "blocks". However, note that, the rest of the steps here are explained as if there is only one block for simplicity. In the Figure 5 below, there are two 64-bit OFDM blocks with 16-bit cyclic prefixes (80 samples each in total).

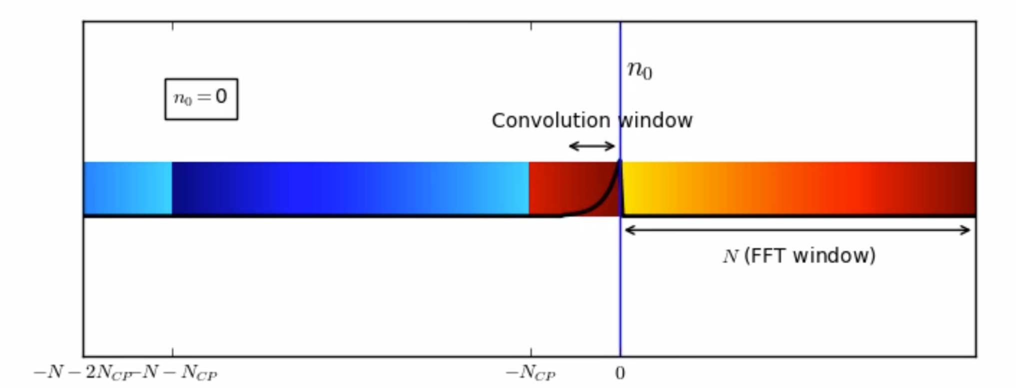

Figure 5: Two 64-bit OFDM blocks with cyclic prefixes (80 samples each in total) being convolved with the channel [2].

Note that, at the depicted part of the convolution, the channel impulse response is completely within the cyclic prefix. As a result, the OFDM data block from the first 80-sample chunk will not affect the next because of the cyclic prefix "barrier". This is a nice additional benefit.

4. Next, one must remove the cyclic prefix from the received data, $y[m]$, which entails removing the first 16 bits of the received signal to obtain the following

$$y[m] = \begin{bmatrix} y_0 & y_1 & y_2 & \dots & y_{63} \end{bmatrix} \tag{18}$$

5. After removing the cyclic prefix, one must compute the DFT of the received signal because bits were encoded in the frequency domain.

$$Y_k = DFT\{y[m]\} \tag{19}$$

6. At this point in the channel estimation process, $X_{k,\text{train}}$ and Y_k are known by the receiver. Thus one can use the relationship $Y_k = H_k X_{k,\text{train}}$ to solve for H_k as follows.

$$H_k = \frac{Y_k}{X_{k,\text{train}}} \tag{20}$$

Note that H_k is a 64-bit vector which contains coefficients to equalize the portion of the channel by which each sub-carrier is processed. Also note that when sending a long string of 100 "blocks" each must separately be divided by H_k and then averaged. A channel estimation is now obtained!

2.3 Data Transmission

Generating a channel estimate allows for sending a data vector (i.e. a vector of data not known by the receiver) across the channel. To do so, the transmitter divides the data vector into 64-bit blocks. Then, it leverages the same strategy used in channel estimation of taking the IFFT of each block and then inserting a cyclic prefix containing the last 16 bits of each block to the beginning of the data block. This step facilitates the circular convolution of the block. The receiver then can remove the cyclic prefix (i.e. the first sixteen bits) from each block of the received data and compute the DFT to yield Y_k, which is equal to the product of the channel and the transmit data. To recover the original signal, the receiver divides out the channel estimate for each block to perform channel equalization.

$$X_{k,\text{data}} = \frac{Y_k}{H_k} \tag{21}$$

The diagram in Figure 6 below illustrates this process for a single block. A data transmission may consist of many side-by-side blocks, but the overall process should be conducted on each individual block (and not on the entire vector).

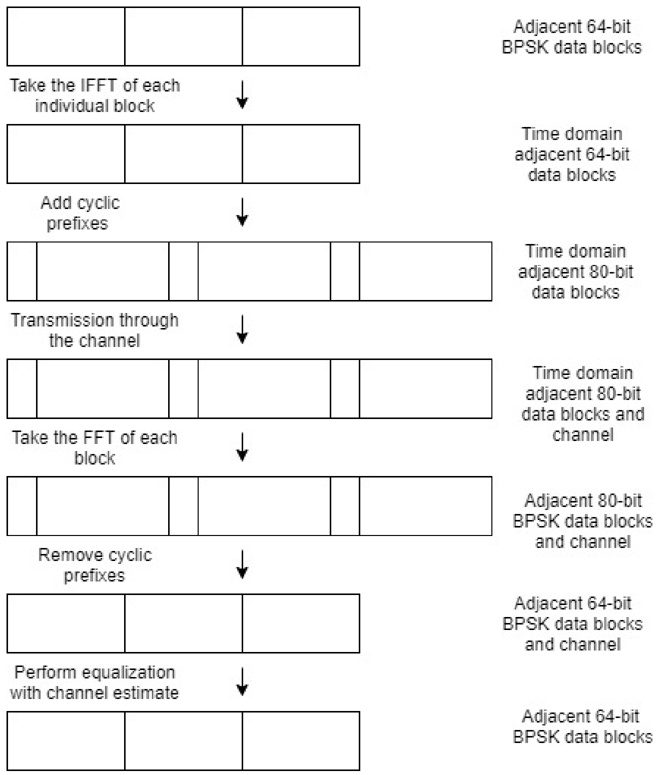

Figure 6: Data Transmission Diagram.

2.4 Timing Synchronization

Will be done in section b.

2.5 Hardware Implementation

Will be done in section c.

3 Results

3.1 Simulation - Assuming Time Synchronization

In Figure 7 below is a comparison of the actual frequency response of the channel (left) with the estimated frequency response.

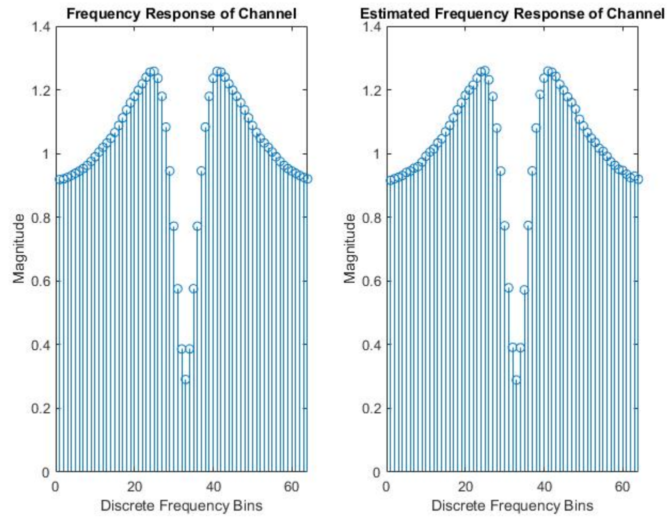

Figure 7: Comparison of the frequency response of the channel to the estimated frequency response of the channel.

It is clear that they are almost identical, so the method of using 100 64-bit training signals to calculate individual channel estimates which are averaged proved successful.

Figure 8 below is a comparison of the first ten bits of the transmitted BPSK signal with the first ten bits of the received BPSK signal.

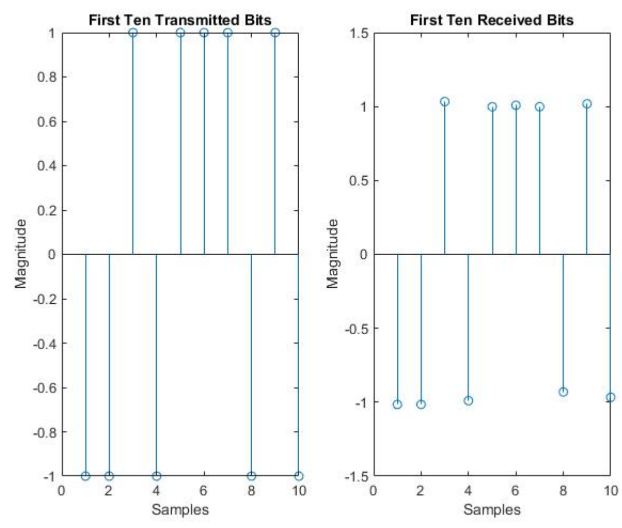

Figure 8: Comparison of the transmitted and received signals using OFDM modulation.

It is also clear in this comparison that the two are almost identical with the exception of small magnitude differences in the received data which is most likely due to noise introduced in the channel.

In the constellation plot of the received data in Figure 9 below, there is clear separation between the ± 1 BPSK values.

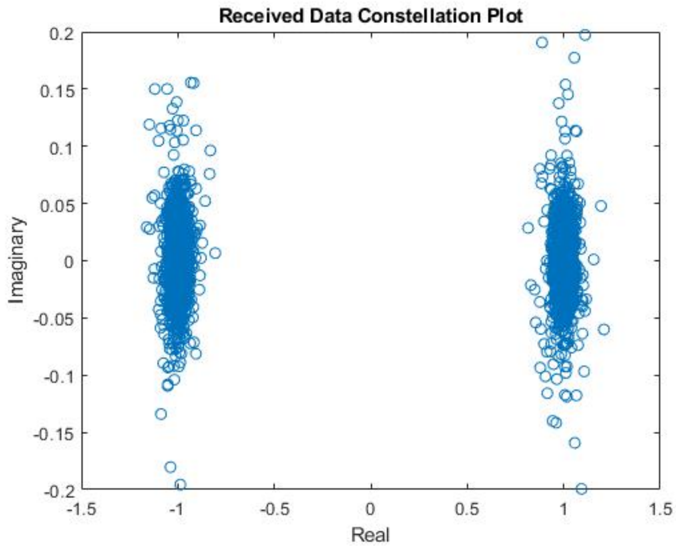

Figure 9: Constellation plot of received data.

Additionally, they variance among the data points along the real component of the signal is quite small and can also be observed in Figure 8, as the magnitudes of the shown bits from the received signal are close to ± 1. However, there is a nontrivial variance among the data points in the imaginary component which, in addition to the small variance in the real component, is due to noise introduced in the channel.

3.2 Simulation - Assuming Frequency Offset

Will be done in section b.

3.3 Hardware Implementation

Will be done in section c.

4 References

[1] Govindasamy S., “OFDM I: Intro and The Discrete Fourier Transform”.

[2] “The Cyclic Prefix in OFDM”, https://dspillustrations.com/pages/posts/misc/the-cyclic-prefix-cp-in-ofdm.html