MIMO with USRP

Analog and Digital Communications Final Project

Shane Kelly, Franton Lin, William Lu, & Byron Wasti

Spring, 2017

1 Introduction

With the increasing prevalence of wireless communications and a thirst for faster, more reliable data transmission, conventional wireless communications that involve a single transmission antenna and single receiving antenna are reaching fundamental limits. Multipath wave propagation and its effects, like fading, cut-out, and picket fencing, cause traditional wireless communication schemes to experience reductions in data rates and increases in error rates.

Multiple Input Multiple Output (MIMO) technology, which involves the use of two or more antennas and the transmission of multiple signals, mitigates the troubles caused by multipath propagation, and sometimes takes advantage of multipath propagation to reduce errors and increase data rates.

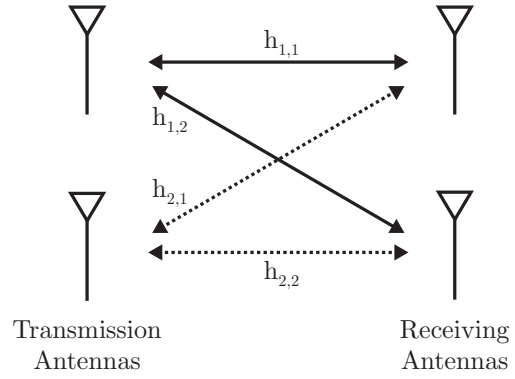

Figure 1: Example of a 2x2 spatial multiplexing MIMO system.

A specific MIMO transmission technique called spatial multiplexing is used to increase data rates. In this technique, two separate signals are sent on the same carrier frequency at the same time, but are transmitted from separate antennas. By transmitting from two different antennas, the "space dimension" can be used to separate the two signals. The "space dimension" is the environment between the transmission and receiving antennas, and spatial multiplexing takes advantage of the multiple paths possible between each transmission antenna and each receiving antenna to increase data rates.

The steps of implementation of spatial multiplexing are roughly as follows:

1. Estimate the channel response in order to generate a matrix of channel coefficients, **H**.
2. To remove the linear combination of multiple data streams, take advantage of the unitary matrices that result from finding the singular value decomposition of **H**.

3. Modify the data streams with information from the SVD of $\mathbf{H}$ ad transmit data.

4. Apply information from the SVD of $\mathbf{H}$ to the received data.

5. Compensate for channel effects like frequency and phase offset.

2 Spatial Multiplexing Theory

In an open-loop MIMO system with N_t transmission antennas and N_r receiver antennas, the relationship between the sent data and received data can be described by

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n},$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_{N_t}]^T$ is the $N_t \times 1$ vector of transmitted symbols while $\mathbf{x}$ and $\mathbf{n}$ are $N_r \times 1$ vectors of received symbols. $\mathbf{H}$ is the $N_r \times N_t$ matrix of channel coefficients. For our system, where we have 2 transmission antennas and 2 receiver antennas (2×2 MIMO), the relationship between the sent data and received data is described by

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}.$$

Because we have a high signal-to-noise ratio, we can approximate $\mathbf{n} = 0$. Our system then looks like this:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{1}$$

In short, our received data becomes the linear combination of our original sent data along with channel effects. In order to recover our original sent data, we need to "undo" this linear combination.

A clever way of doing this involves finding the singular value decomposition (SVD) of the $m \times n$ channel coefficient matrix, $\mathbf{H}$. This is

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{H}},$$

where $\mathbf{U}$ is a $m \times m$ unitary matrix, $\mathbf{\Sigma}$ is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal, and $\mathbf{V}^{\mathbf{H}}$ is the Hermitian transpose of a $n \times n$ unitary matrix. Because $\mathbf{U}$ and $\mathbf{V}^{\mathbf{H}}$ are both unitary matrices, we can multiply them by their Hermitian transpose to turn them into identity matrices, thereby eliminating them. We're then left with $\mathbf{\Sigma}$, and as a diagonal matrix, $\mathbf{\Sigma}$ allows us to "undo" the linear combination of our original sent data along with channel effects.

If we start with Equation (1) and we know what the channel coefficients, $\mathbf{H}$, look like, we can eliminate $\mathbf{V}^{\mathbf{H}}$ by multiplying our original sent data with $\mathbf{V}$.

$$\mathbf{y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{H}}\mathbf{V}\mathbf{x}$$
$$\mathbf{y} = \mathbf{U}\mathbf{\Sigma}\mathbf{x}$$

If we multiply the received data, $\mathbf{y}$, by $\mathbf{U^H}$, we can then eliminate $\mathbf{U}$.

$$\begin{aligned}\mathbf{U^H y} &= \mathbf{U^H U \Sigma x} \\ \mathbf{y} &= \mathbf{\Sigma x} \end{aligned} \tag{2}$$

Because we're simply left with Equation (2), we can separate out x_1 and x_2 from $\mathbf{x}$. All that is left to do is remove the channel effects encoded in $\mathbf{\Sigma}$.

3 Estimating Channels

For the above operation to work, we need to first estimate the channel coefficients, $\mathbf{H}$ so that we can take the SVD and find $\mathbf{V}$ and $\mathbf{U}$.

The cross-correlation of an output signal, y, and an input signal, x, can be defined as the auto-correlation of the input signal, R_{xx}, convolved with the impulse response of the channel, $h(\tau)$, as follows:

$$R_{yx}(\tau) = R_{xx} * h(\tau).$$

If the input signal, x, is white noise, then

$$R_{xx} = \delta(\tau),$$

so

$$\begin{aligned} R_{yx} &= \delta(\tau) * h(\tau) \\ &= h(\tau). \end{aligned}$$

This means that to estimate the channel from a transmitter to a receiver, we simply send a white noise signal and convolve the received signal by the inputted white noise.

4 Coarse Adjustment for Frequency Offset and Phase Offset

Once we estimate the channel and channel coefficients, calculate the SVD, and apply the theory from Section 2 on the preceding page, we need to adjust for the frequency offset and phase offset of the channel. We will eventually use a phase locked loop (PLL) to accurately correct for the frequency and phase offset of the channel, but a coarse adjustment of frequency and phase offset need to occur first.

The received signal, $y[m]$, can be defined as the input, $x[m]$, multiplied by the channel, h, multiplied by some complex exponential with unknown frequency and phase offset phase offset, $e^{2\pi f_\Delta m + \theta_\Delta}$. With the addition of some noise, $n[m]$, the received signal is

$$y[m] = he^{j2\pi f_\Delta m + \theta_\Delta} x[m] + n[m].$$

If the data transmission scheme is BPSK, then

$$x[m] = \pm 1.$$

The signal to noise ratio can be assumed to be quite high, which makes noise approximately zero as follows

$$n[m] = 0.$$

Plugging in these values for $x[m]$ and $n[m]$ leaves us with

$$y[m] = he^{j2\pi f_\Delta m + \theta_\Delta}(\pm 1) + 0,$$

which, when squared becomes

$$y^2[m] = h^2 e^{j2\pi 2 f_\Delta m + \theta_\Delta}.$$

We can now move the θ_Δ term into h and denote it as $\bar{h}$. Using this substitution makes

$$y^2[m] = \bar{h}^2 e^{j2\pi 2 f_\Delta m}.$$

Now, if you take the discrete time Fourier transform of $y^2[m]$, you can extract $\bar{h}^2$ and $2f_\Delta$.

$$\text{DTFT of } \left\{y^2[m]\right\} = \bar{h}^2 \delta(\Omega - 2f_\Delta).$$

A visual examination of the discrete time Fourier transform will reveal a large spike at $2f_\Delta$ with an amplitude of $\bar{h}^2$.

5 Phase Locked Loop (PLL) for QPSK

After roughly adjusting for the frequency and phase offset of the channel, we can use a PLL to find the remaining frequency and phase offset. To begin, we know that our discrete signal, $x[n]$ can be split into real and imaginary components.

$$x[n] = x_r[n] + x_i[n] \tag{3}$$

We know that some frequency and phase offset is applied, so we can add that to our mathematical model. If the offset is represented by $\phi = f_\Delta n + \theta_\Delta$, then the discrete signal can be represented by

$$x[n]e^{j\phi} \tag{4}$$

After using Euler's relation, we can combine Equation (3) and Equation (4).

$$
\begin{aligned}
w[n] &= (x_r[n] + jx_i[n])\,(cos\phi + jsin\phi) \\
w[n] &= x_r[n]cos\phi - x_i[n]sin\phi + jx_i[n]cos\phi + jx_r[n]sin\phi \\
\text{Imag}\{w[n]\} &= x_i[n]cos\phi + x_r[n]sin\phi \\
\text{Real}\{w[n]\} &= x_r[n]cos\phi - x_i[n]sin\phi
\end{aligned}
$$

For small ϕ,

$$
\begin{aligned}
\text{Imag}\{w[n]\} &\approx x_i[n] \\
\text{Real}\{w[n]\} &\approx x_r[n].
\end{aligned}
$$

Using the approximations for small ϕ,

$$\text{Sign}\{\text{Imag}\{w[n]\}\}\,\text{Real}\{w[n]\} \approx x_r[n]x_i[n]cos\phi - x_i^2[n]sin\phi \tag{5}$$

$$\text{Sign}\{\text{Real}\{w[n]\}\}\,\text{Imag}\{w[n]\} \approx x_r[n]x_i[n]cos\phi + x_r^2[n]sin\phi. \tag{6}$$

Equation (5) − Equation (6):

$$
\begin{aligned}
\text{Eq(5)} - \text{Eq(6)} &= -2x_i^2[n]sin\phi \\
&= -2sin\phi
\end{aligned}
$$

To correct for the frequency and phase offset:

$$
\begin{aligned}
\frac{1}{2}\left[\text{Eq(5)} - \text{Eq(6)}\right] &= sin\phi \\
&\approx \phi
\end{aligned}
$$

6 Implementation

We decided to implement a 2×2 MIMO system using two B210 USRPs. There were a number of factors that made implementation rather tricky. We initially got perfect data transmission by syncing the frequency of both USRPs using an external clock.

However, when moving away from the external clock, we ran into many issues when we tried to correct for the frequency offset of the two USRPs as well as the frequency drift that occurs during transmission.

This was largely a problem when approximating the channel, as oftentimes the frequency drift made the channel approximation largely incorrect. This meant our MIMO implementation did not send data that could be decoupled, leading to a roughly 50% error rate for both data streams.

By using a raised cosine for the pulses for our noise, we were able to minimize the frequency drift when doing the FFT frequency correction. The data we sent is shown in Figure 2, with the received data shown in Figure 3.

To extract our channel approximations, cross-correlated the received noise with the noise we sent for each of 4 different transmitter-receiver interactions. The height of the peak for the cross-correlation is approximately the channel's impulse response. These peaks can be seen in Figure 4.

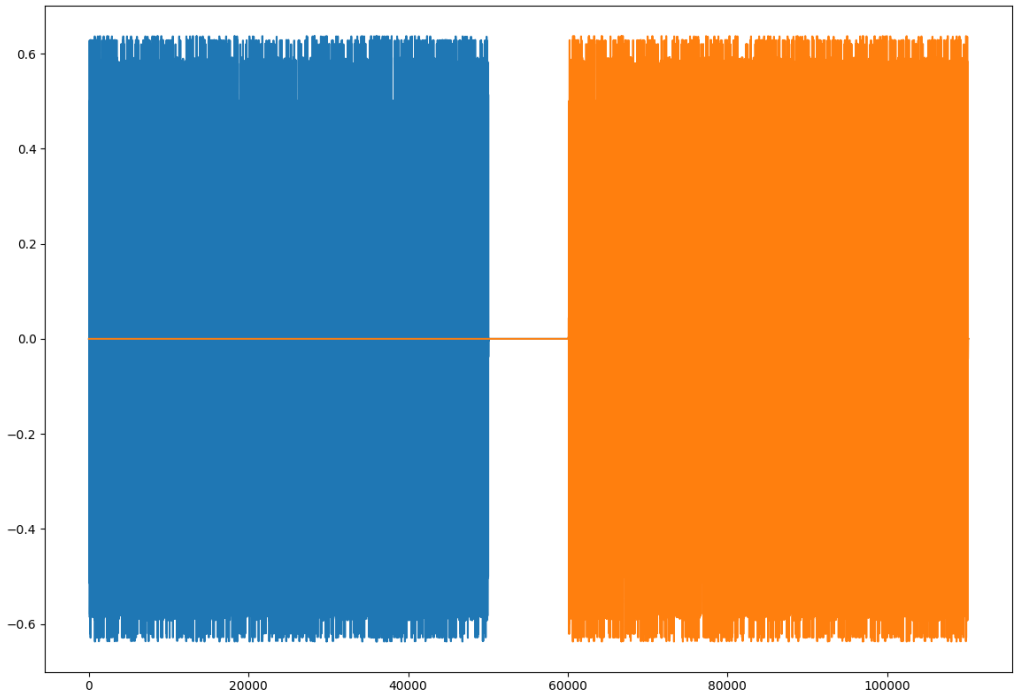

Figure 2: Noise sent for channel approximation. Blue shows noise sent by Transmitter 1, and orange shows noise sent by Transmitter 2.

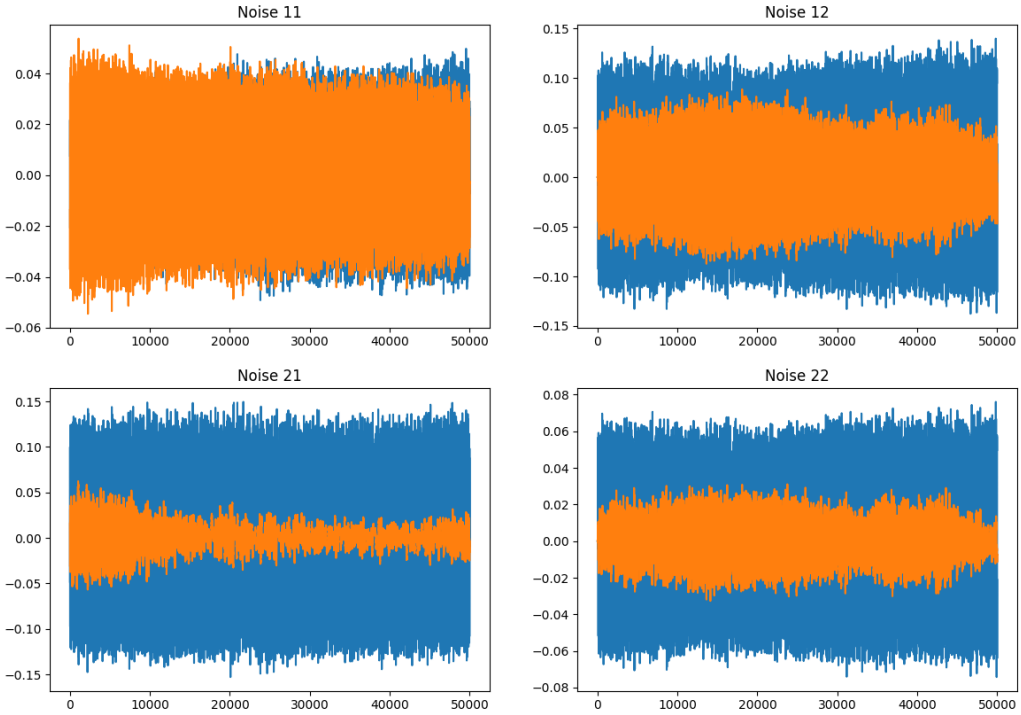

Figure 3: Noise received after correcting for frequency offset using the FFT technique. Note that there is slight drift over time.

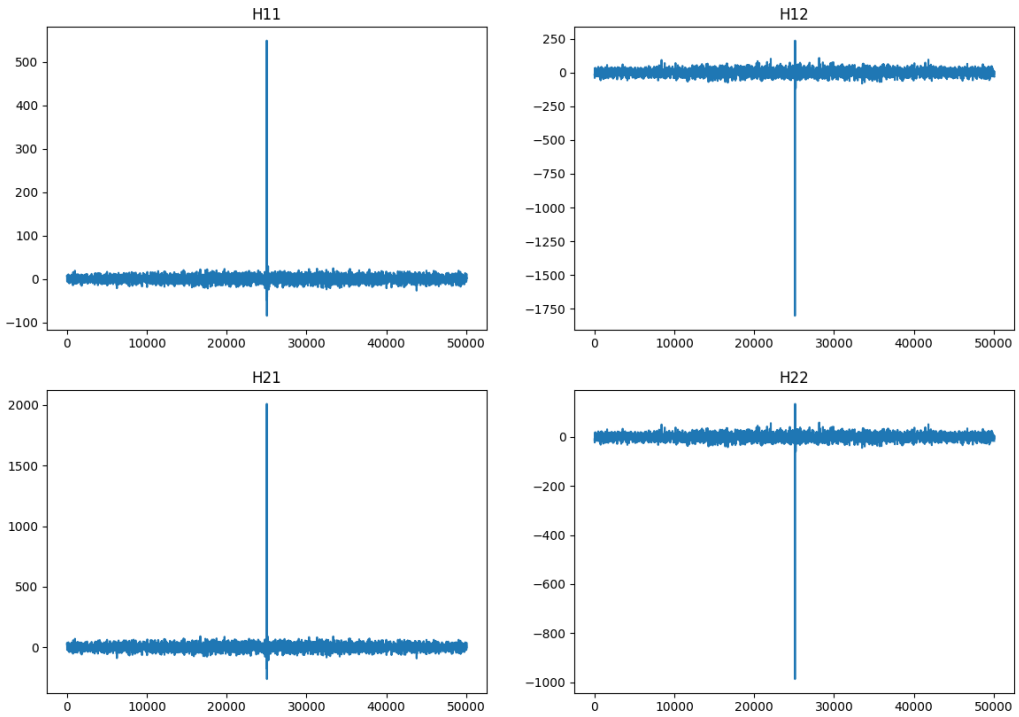

Figure 4: Approximating the channel response from the height of the peak of the cross-correlated received signal with the noise sent.

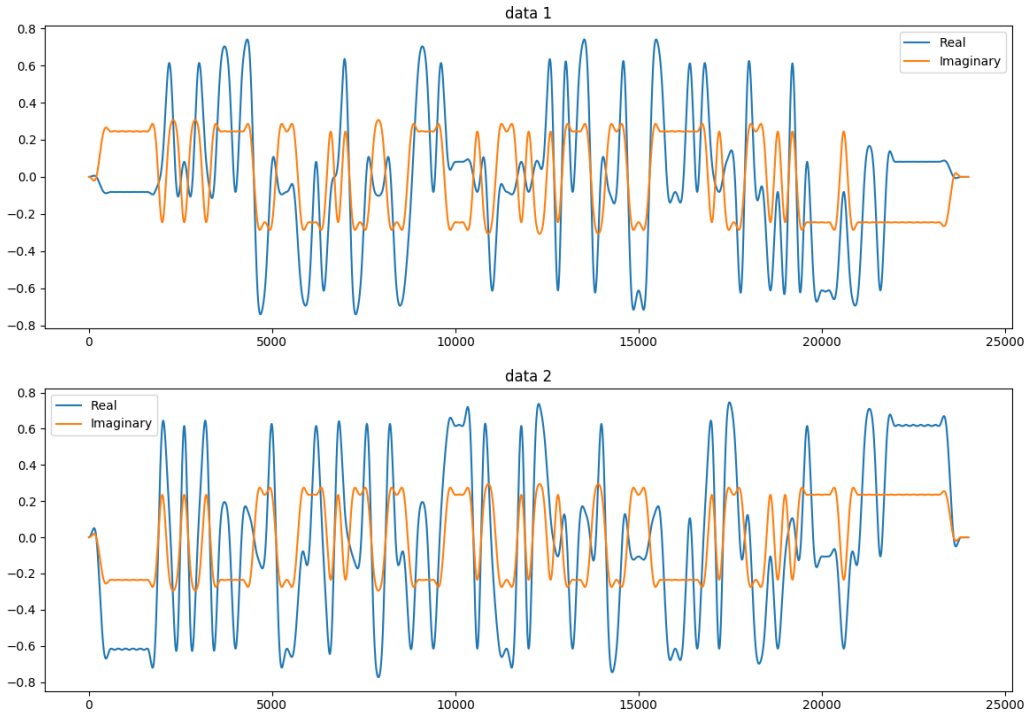

Figure 5: Data sent after applying the $\mathbf{V}$ matrix. Note that the original data was all real values, and that the $\mathbf{V}$ matrix has added an imaginary component to allow for MIMO.

After extracting our values for the channel approximation, we took our SVD and applied the $\mathbf{V}$ matrix to our data-to-be-sent. The data sent can be seen in Figure 5.

$$\mathbf{H} = \begin{bmatrix} 548.77 - 618.91j & -1801.35 - 854.614j \\ 2008.41 - 157.767j & -987.159 - 235.436j \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} -0.61442173 + 0.29947892j & 0.72933966 + 0.02935951j \\ -0.71140736 + 0.16339487j & -0.65912145 - 0.18099883j \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 2853.86206055 & 1266.26013184 \end{bmatrix}$$

$$\mathbf{V^H} = \begin{bmatrix} -0.69278103 + 0.j & 0.53073865 + 0.48823246j \\ -0.72114801 + 0.j & -0.50986159 - 0.46902743j \end{bmatrix}$$

Finally, we managed to receive data that was (somewhat) reasonable. The corrected received data is shown in figure 6 on the next page. For our first data stream, we had a miserable percent error of 39.66%. However, our second data stream had perfect recovery of data. This most likely means that our $\mathbf{H}$ matrix was skewed to favor one of the data-transmission channels over the other.

Although this error-rate is pretty horrible (especially considering our low data-rate), it does demonstrate MIMO communication that is marginally better than just guessing.

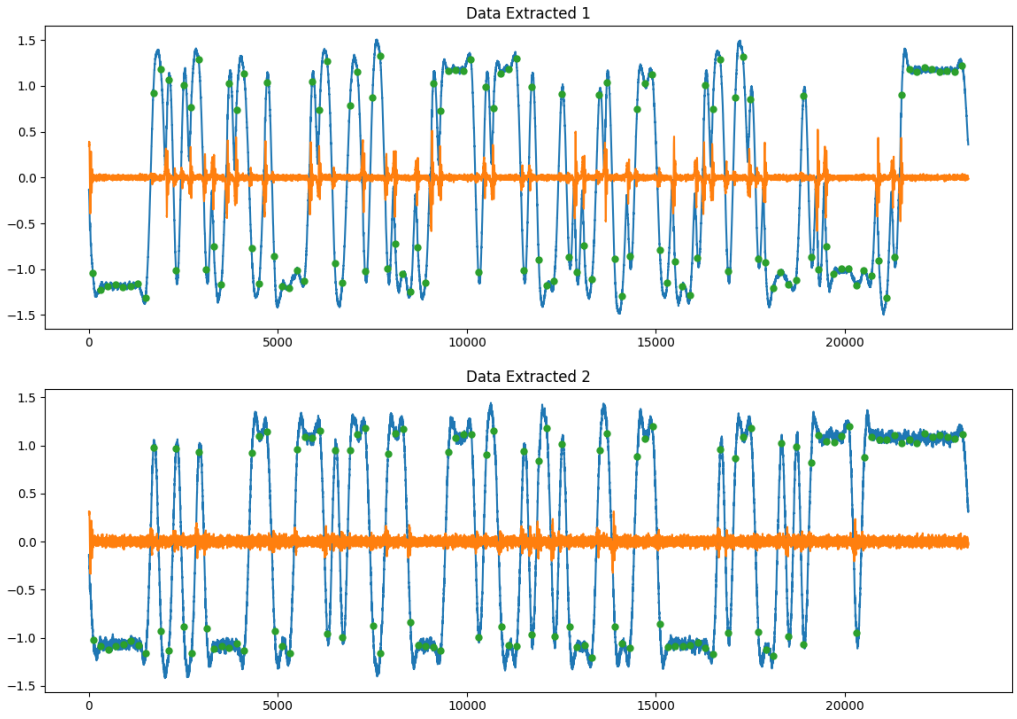

Figure 6: Here is the final extracted data. The error rate for the top data-set is 39.66%, while the error rate for the bottom data-set is a nice 00.00%. Note the blue is real data, the orange is imaginary data and the green dots show where we sample the data.

7 Discussion of What Went Wrong

There were a number of flaws in our final implementation that made it not function properly. The main issue we continually faced was correcting for the frequency offset and drift between the two USRPs.

To correct the frequency offset during channel estimation, we attempted to use both the FFT and PLL methods. The FFT method for correcting for frequency offsets functioned for the most part, but lead to a rather large frequency drift (especially when working with noise that had square-pulses). We attempted to use a PLL described in Section 5 on page 4 which locked onto only one of the received signals, but had the error correct all of them. The goal of this was to correct for the frequency drift, but to avoid removing the phase-difference between the received signals.

However, this technique did not fix the frequency drift for the two received signals that came second. We then tried using two PLLs, which locked onto one of each of the pairs of received signals, while applying the error to their respective pair. This brought its own issues, the most important being that it eliminated the phase offset between the two received signals which had been PLL'ed, which meant that the MIMO algorithm would not longer function properly.

Switching to a raised-cosine for our noise pulse seemed to mitigate the frequency drift, which we believe is in part to working better with the FFT method for frequency offset correction. However, this did not fix the issue completely. Without accurate channel estimates, sending and correctly decoding data is difficult.

We believe if we had additional time, and slightly better testing conditions where the position of objects could remain more constant, we would be able to develop a functional MIMO implementation. Since we were able to get perfect data transmission when the USRPs were running off an external clock, we know that the math (and most of the implementation) is functional.

A Code

For the code we used to implement this project, please go to our GitHub repository: `https://github.com/williamalu/mimo_usrp`