Principles of Wireless Communications Lab 2a: MIMO

Mark Goldwater and Anusha Datar

February 25, 2021

1 System Overview

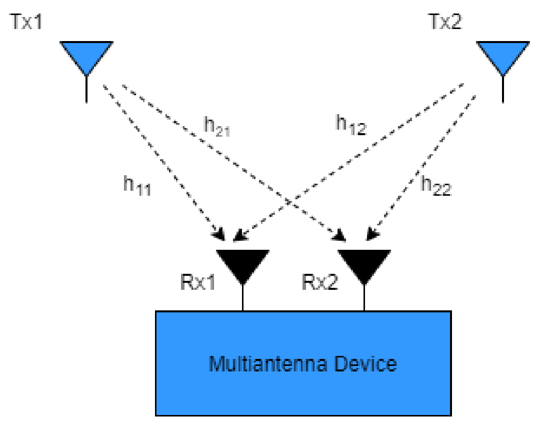


Figure 1: Here we show a diagram of our MIMO communications system model used to extract the messages sent from both Tx_1 and Tx_2.

In this lab, we are decoding two signals simultaneously sent from two separate transmitter and received by two antennas on the same receiver. We are once again assuming a flat-fading channel model where we assume that because the bandwidth of the transmitted signal is much less than the coherence bandwidth of the channel (where the channel's frequency response can be assumed to be flat). We can approximate the channel as a single complex scalar. However, because we are decoding a MIMO system, we have effectively four channels because the signal from each transmitter is received by both receivers as shown above in Figure 1. The format for the indexing of these scalars is $h_{Rx,Tx}$.

With this in mind, we can represent the received signal at Rx_1 and Rx_2 at sample k (assuming flat-fading channel and carrier synchronization) with the following two equations:

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

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

We can then convert these equations into a matrix form to work efficiently with all streams of data at all sampling times k_1, k_2, ..., k_n. These matrices are defined as follows.

$$\boldsymbol{y} = \begin{bmatrix} y_1[k_1] & y_1[k_2]\dots & y_1[k_n] \\ y_2[k_1] & y_2[k_2]\dots & y_2[k_n] \end{bmatrix} \tag{3}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1[k_1] & x_1[k_2]\dots & x_1[k_n] \\ x_2[k_1] & x_2[k_2]\dots & x_2[k_n] \end{bmatrix} \tag{4}$$

$$\boldsymbol{n} = \begin{bmatrix} n_1[k_1] & n_1[k_2]\dots & n_1[k_n] \\ n_2[k_1] & n_2[k_2]\dots & n_2[k_n] \end{bmatrix} \tag{5}$$

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

With this new matrix representation of the data and channel, we can combine Equations 1 and 2 into the following matrix equation.

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n} \tag{7}$$

This representation of our system will help us to untangle the distinct messages sent (x_1 by Tx_1 and x_2 by Tx_2) using both a *zero-forcing receiver* and an *MMSE receiver* which are described in subsequent sections.

1.1 Software Overview

We organized our software into two functions - zero_forcing_receiver and mmse_receiver. As illustrated in Figure 2, each of these functions call modular functions to correct for time delay, interpret the training signal, compute the values in and assemble the weight vectors, decode the signal, and compute the error. All project code and associated documentation is available at `https://github.com/anushadatar/simulated-mimo`. Each of the bolded functions in Figure 2 has its own file in this repository.

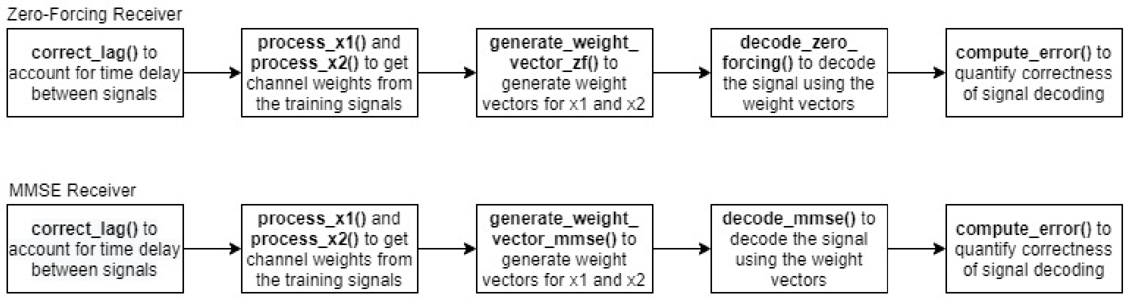

Zero-Forcing Receiver
correct_lag() to account for time delay between signals
process_x1() and process_x2() to get channel weights from the training signals
generate_weight_vector_zf() to generate weight vectors for x1 and x2
decode_zero_forcing() to decode the signal using the weight vectors
compute_error() to quantify correctness of signal decoding
MMSE Receiver
correct_lag() to account for time delay between signals
process_x1() and process_x2() to get channel weights from the training signals
generate_weight_vector_mmse() to generate weight vectors for x1 and x2
decode_mmse() to decode the signal using the weight vectors
compute_error() to quantify correctness of signal decoding

Figure 2: Software flowchart.

Below are the declarations and a short description of each of the functions used in the above diagram.

1. **[y1, y2] = correct_lag(x1, x2, y1, y2)** - The correct_lag function uses the cross correlation to account for the received signal lagging behind the vector of transmitted data so that we can retrieve training signals and data using the indices in the transmission steps detailed in Section 2.

2. **[h11, h21] = process_x1(x1, y1, y2, pulseWidth)** and **[h12, h22] = process_x2(x1, y1, y2, pulseWidth)** - These functions exploit our flat-fading channel assumption to calculate the elements of our channel matrix $\boldsymbol{H}$.

3. **w = generate_weight_vector_zf(H, one_hot)** and **[w1, w2] = generate_weight_vector_mmse(H)** - These functions use the zero-forcing receiver and MMSE receiver methods to extract x_1 and x_2 as described below.

4. **[x1_decoded, x2_decoded] = decode_zero_forcing(y, x1_w, x2_w)** and **[x1_decoded, x2_decoded] = decode_mmse(y, x1_w, x2_w)** apply the computed weight vectors to the received signal and output recovered signals.

5. **[err, bits_x, bits_y] = compute_error(decoded_signal, original_signal)** - These functions compare the decoded signal and original signal to determine the percent error **err**. It also outputs the downsampled data bits **bits_x** and **bits_y**.

2 Decoding the Received Signals

The data collection for this lab was carried out according to the following steps.

1. 5000 zero samples from Tx_1 and Tx_2.
2. Tx_1 sends 128 pseudo random bits, encoded using BPSK and 40 sample long rectangular pulses, Tx_2 sends zeros.
3. 5000 zero samples from both Tx_1 and Tx_2.
4. Tx_2 sends 128 pseudo random bits, encoded using BPSK and 40 sample long rectangular pulses, Tx_1 sends zeros.
5. 5000 zero samples from both Tx_1 and Tx_2.
6. Tx_1 and Tx_2 send 1024 data bits (they are randomly generated here), using BPSK and rectangular pulses that are 40 samples each.

Note that in steps 2 and 4, one transmitter is sending 128 pseudo random bits while the other is sending zeros. These two signals will be used as "training signals" to extract the approximation of the flat-fading channel coefficients. With these coefficients, we will be able to extract both x_1 and x_2 despite both transmitters sending them simultaneously in step 6 above.

2.1 Channel Estimation

Before we can perform any post processing on the received data, it is crucial that we account for the lag in the received signal vector relative to the transmitted signal vector. In order to do this, we can take the cross correlation of the known training signal with the data captured on each receiver. Based on the lag where the maximum correlation is identified we can shift each received vector back and append zeros to the end so that it maintains the same length.

The first step in decoding the received signals is to calculate a sufficient approximation of the channel between each transmitter and receiver present in the system. Recalling that we are assuming that our system can be modeled using a flat-fading channel estimation, estimating the elements of our channel matrix, $\boldsymbol{H}$, is fairly simple. To estimate $h_{Rx,Tx}$ we can element-wise divide the reception of the previously mentioned and known training signal by the training signal that was sent. We then take the mean of this element-wise division to try and account for the noise in the channel as shown in the below equations.

$$h_{11} = \text{mean}(y_{1,\text{train}} \oslash x_{1,\text{train}}) \tag{8}$$

$$h_{21} = \text{mean}(y_{2,\text{train}} \oslash x_{1,\text{train}}) \tag{9}$$

$$h_{12} = \text{mean}(y_{1,\text{train}} \oslash x_{2,\text{train}}) \tag{10}$$

$$h_{22} = \text{mean}(y_{2,\text{train}} \oslash x_{2,\text{train}}) \tag{11}$$

With these coefficients (which are real numbers in our case because the data was sent using BPSK) and our time-aligned signals, we are now able to construct our $\boldsymbol{H}$ matrix in Equation 6 and untangle x_1 and x_2.

2.2 Zero-Forcing Receiver

2.2.1 Derivation

For the sake of this explanation, let us suppose that we are trying to extract x_1 using a zero-forcing receiver. In order to do this, we will design a weight vector which we multiply by Equation 27 as shown below

$$\hat{x}_1 = w^\dagger \boldsymbol{y} \tag{12}$$

where the † symbol denotes the conjugate transpose. Simplifying the model even further and claiming that our noise is negligible ($\boldsymbol{n} = 0$), we can substitute the fact that $\boldsymbol{y} = \boldsymbol{Hx}$ and calculate using the above equation.

$$\begin{aligned}
\hat{x}_1 &= w^\dagger \boldsymbol{Hx} \\
&= \begin{bmatrix} w_1^* & w_2^* \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= \begin{bmatrix} w_1^* h_{11} + w_2^* h_{21} & w_1^* h_{12} + w_2^* h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= (w_1^* h_{11} + w_2^* h_{21}) x_1 + (w_1^* h_{12} + w_2^* h_{22}) x_2
\end{aligned} \tag{13}$$

In order to ensure that $\hat{x}_1 = x_1$ we need the following to be true so that the contributions of x_2 in both y_1 and y_2 are negated.

$$w_1^* h_{11} + w_2^* h_{21} = 1 \tag{14}$$

$$w_1^* h_{12} + w_2^* h_{22} = 0 \tag{15}$$

We can express this as the following matrix equation and solve for w.

$$\begin{aligned}
\begin{bmatrix} w_1^* & w_2^* \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}^\dagger \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
\boldsymbol{H}^\dagger w &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
w &= (\boldsymbol{H}^\dagger)^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned} \tag{16}$$

If we want to extract x_2 instead, we can simply flip the zero and one in the one-hot vector used in the above calculation and apply it to y in a similar way to Equation 12. Thus, we can apply our zero-forcing receiver weight vectors as follows.

$$\hat{x}_1 = \left((\boldsymbol{H}^\dagger)^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^\dagger \boldsymbol{y} = w_1^\dagger \boldsymbol{y} \tag{17}$$

$$\hat{x}_2 = \left((\boldsymbol{H}^\dagger)^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)^\dagger \boldsymbol{y} = w_2^\dagger \boldsymbol{y} \tag{18}$$

2.2.2 Why Does It Work?

To gain a better intuition as to why this works (and why it is called a zero-forcing receiver) let us fully expand the calculation of w for the case where we are trying to obtain $\hat{x}_1$.

$$\begin{aligned} w &= (\boldsymbol{H}^{\dagger})^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} h_{11}^{*} & h_{12}^{*} \\ h_{21}^{*} & h_{22}^{*} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{h_{11}^{*}h_{22}^{*} - h_{12}^{*}h_{21}^{*}} \begin{bmatrix} h_{22}^{*} \\ -h_{12}^{*} \end{bmatrix} \end{aligned} \tag{19}$$

Next, if we take Equation 7, ignoring noise, and decompose the channel matrix into the flat-fading coefficients relevant to each message $\boldsymbol{h}_1 = \begin{bmatrix} h_{11} \\ \boldsymbol{h}_{21} \end{bmatrix}$ and $h_2 = \begin{bmatrix} h_{12} \\ h_{22} \end{bmatrix}$, we can write the following system equation.

$$\boldsymbol{y} = x_1\boldsymbol{h}_1 + x_2\boldsymbol{h}_2 \tag{20}$$

Noticing that $\boldsymbol{h}_2$ is orthogonal to w, the result of operating on $\boldsymbol{y}$ with w will force the term containing x_2 to zero because $w^{\dagger}(x_2\boldsymbol{h}_2) = 0$ because of this relationship. This is why this type of receiver is called "zero-forcing."

2.3 MMSE Receiver [1]

The zero-forcing receiver can amplify noise in cases where the ratio between the maximum singular value and the minimum singular value of the matrix $\boldsymbol{H}$ is very large. To account for this difference, we can add a regularization term $\lambda||x||^2$ and calculate a new weight vectors which are applied to the received data in the same way. We essentially wish to solve the following optimization problem.

$$\hat{x}(\boldsymbol{y}) = \text{argmin}_{\boldsymbol{x}}||\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}||^2 + \lambda||x||^2 \tag{21}$$

To find the values in the weight vector after we apply this regularization factor, we solve the optimization problem and yield our individual messages in each row of the resulting matrix. Note that in our code, however, we apply each weight vector individually as opposed to all simultaneously to the received data.

$$\hat{x}(\boldsymbol{y}) = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \boldsymbol{y} = (\boldsymbol{H}^H + \lambda \boldsymbol{I})^{-1}\boldsymbol{H}^H\boldsymbol{y} \tag{22}$$

Similar to in the case of the zero forcing receiver, we can use separate portions of the computed weight vector to extract the value of each signal here because we calculated all the weight vectors at once (as opposed to calculating each individually in our explanation of the zero-forcing receiver). Again, in the context of our implementation, we can extract x_1 by taking the first row of the weight vector and multiplying it by $\boldsymbol{y}$. To extract the value of x_2, we can take the second row of the weight vector and multiply it by $\boldsymbol{y}$.

In our implementation of the MMSE receiver, we computed the value of our regularization term by estimating the overall signal variance by computing the variance of the first 5000 data points (which theoretically only contain noise because both transmitters are silent) received on one of the antennas.

2.4 Multiple Antennas

In a system with additional transmitters (with associated antennas) and additional receiving antennas (on the same receiving device), the core principle of both of these approaches would remain the same, but the dimensions of the matrices characterizing the channel would scale accordingly. If we have a system with m transmitters and m receivers, we have to account for the channel between every pair of transmitters and receivers with the following $\boldsymbol{H}$ matrix.

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2m} \\ h_{31} & h_{32} & h_{33} & \dots & h_{3m} \\ \vdots & \vdots & \vdots & \ddots & \\ h_{m1} & h_{m2} & h_{m3} & & h_{mm} \end{bmatrix} \tag{23}$$

We would then also have equations like Equations 14 and 26 for each message we are sending to extract a desired data stream. The case for where x_1 is desired is shown below.

$$w_1^* h_{11} + w_2^* h_{21} + ... + w_m^* h_{m1} = 1 \tag{24}$$

$$w_1^* h_{12} + w_2^* h_{22} + ... + w_m^* h_{m2} = 0 \tag{25}$$

$$\vdots$$

$$w_1^* h_{1m} + w_2^* h_{2m} + ... + w_m^* h_{mm} = 0 \tag{26}$$

This would also make our weight vector size $m \times 1$. Our data vectors $(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{n})$ would also be size $m \times 1$ as we would need to add rows up to y_m, x_m, and n_m as shown below.

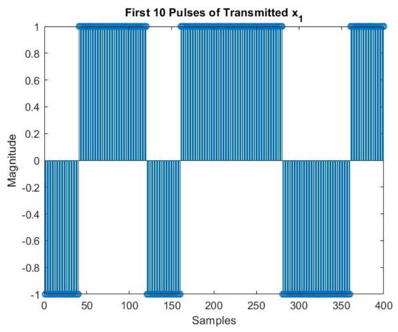
$$\boldsymbol{y} = \begin{bmatrix} y_1[k_1] & y_1[k_2] & \dots & y_1[k_n] \\ y_2[k_1] & y_2[k_2] & \dots & y_2[k_n] \\ \vdots & \vdots & \vdots & \vdots \\ y_m[k_1] & y_m[k_2] & \dots & y_m[k_n] \end{bmatrix} \tag{27}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1[k_1] & x_1[k_2] & \dots & x_1[k_n] \\ x_2[k_1] & x_2[k_2] & \dots & x_2[k_n] \\ \vdots & \vdots & \vdots & \vdots \\ x_m[k_1] & x_m[k_2] & \dots & x_m[k_n] \end{bmatrix} \tag{28}$$

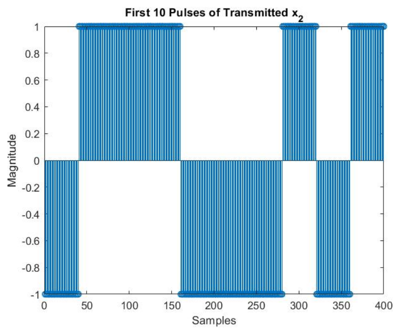
$$\boldsymbol{n} = \begin{bmatrix} n_1[k_1] & n_1[k_2] & \dots & n_1[k_n] \\ n_2[k_1] & n_2[k_2] & \dots & n_2[k_n] \\ \vdots & \vdots & \vdots & \vdots \\ n_m[k_1] & n_m[k_2] & \dots & n_m[k_n] \end{bmatrix} \tag{29}$$

3 Results

To compute the magnitude of the error associated with our communication system, we wrote a function that sampled the data bit from the center of each pulse in the transmitted data and corrected received data and determined the percent error or the isolated data, y_1 and y_2, relative to the transmitted data, x_1 and x_2. **Both our zero-forcing receiver and our MMSE receiver were able to achieve zero error when processing the data that was provided for this lab.**



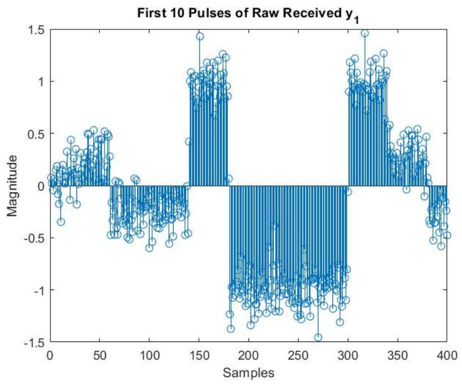
(a) First 10 pulses of x_1 to be transmitted.



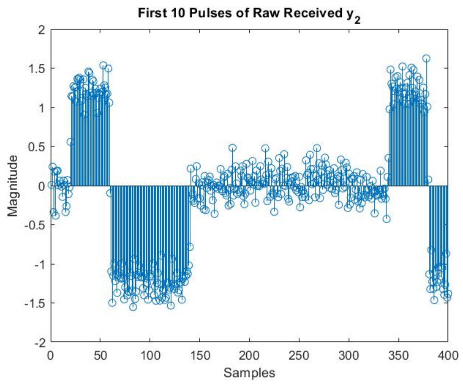
(b) First 10 pulses of x_2 to be transmitted.

Figure 3: The message signals x_1 and x_2 before they are transmitted. The modulation scheme used is BPSK with 40-sample pulses.

Above in Figure 3 are the first ten pulses of x_1 and x_2 which are to be transmitted simultaneously by Tx_1 and Tx_2 respectively. Although above we only show the first ten pulses for clarity, the full data contains 1,024 pseudo-randomly generated bits.



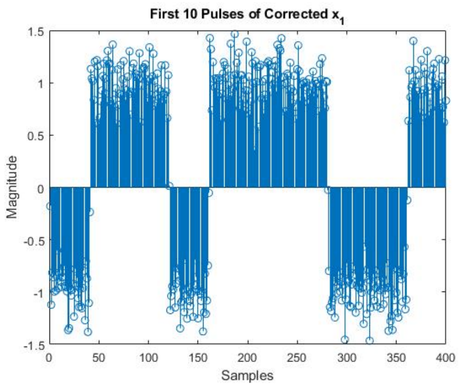
(a) First 10 pulses of raw received data at Rx_1.



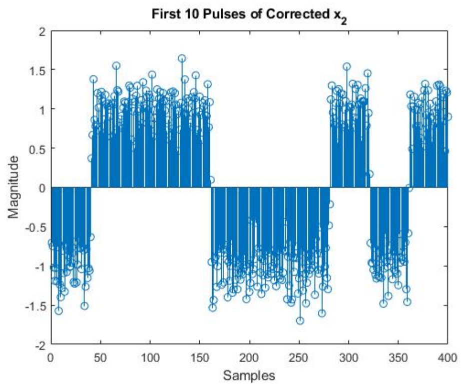
(b) First 10 pulses of raw received data at Rx_2.

Figure 4: The raw received data at Rx_1 and Rx_2 when x_1 and x_2 were sent simultaneously.

Above in Figure 4, we have the first ten pulses of raw unprocessed data captured simultaneously by Rx_1 and Rx_2. It is clear that while some pulses are visible, the data overall looks messy and the two receivers are capturing a different series of samples.



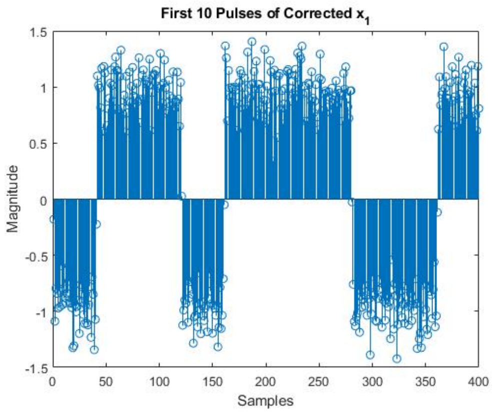
(a) First 10 pulses of x_1 recovered via a zero-forcing receiver.



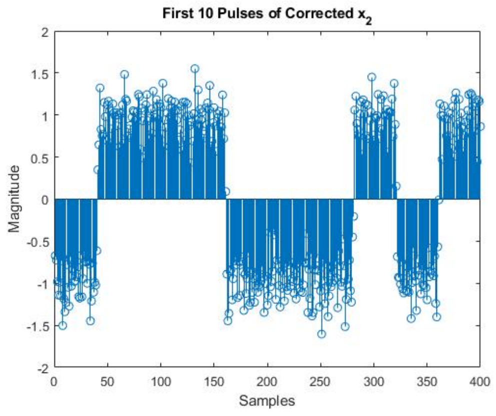
(b) First 10 pulses of x_2 recovered via a zero-forcing receiver.

Figure 5: Above we show x_1 and x_2 untangled using a zero-forcing receiver after being simultaneously transmitted by Tx_1 and Tx_2.

In Figure 5 above we see the first ten pulses of both x_1 and x_2 extracted using a zero-forcing receiver (described in Section 2.2). Compared to the plots in Figure 4, there are clear pulses which match those in the original transmitted signals shown in Figure 3. Although the magnitude slightly differs, the pulse sufficiently represent the original signals.



(a) First 10 pulses of x_1 recovered via an MMSE receiver.



(b) First 10 pulses of x_2 recovered via an MMSE receiver.

Figure 6: Above we show x_1 and x_2 untangled using an MMSE receiver after being simultaneously transmitted by Tx_1 and Tx_2.

Lastly, above in Figure 6, we see the first ten pulses of both x_1 and x_2 recovered using an MMSE receiver (described in Section 2.3). Once again, pulses are clearly discernible and match the original signals shown in Figure 3.

4 References

[1] EE359 staff, Stanford University. MIMO detection algorithms.
[2] Govindasamy S., Multiantenna Systems.