# Principles of Wireless Communications Lab 2b: MIMO

Mark Goldwater and Anusha Datar

March 4, 2021

## 1 System Overview

### 1.1 Mathematical System Model

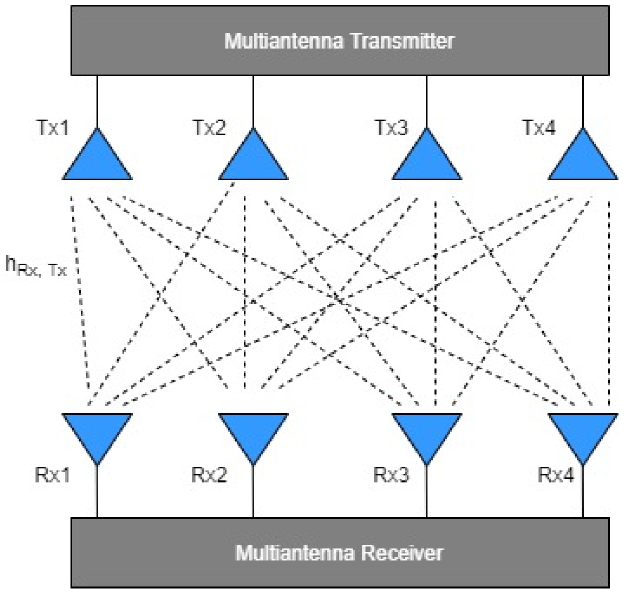


**Figure 1:** Diagram of our 4X4 MIMO system where four unique messages are sent by $Tx_{1-4}$ simultaneously.

Similarly to Lab 2A, we are implementing a MIMO communications system; however, in this lab it is a 4X4 MIMO system (four transmit antennas and four receive antennas), and we generate our own transmit data as well as simulate the channel instead of using prerecorded data. Once again, we are assuming a flat-fading channel model where the bandwidth of the transmitted signal is significantly less than the coherence bandwidth of the channel (where the channel's frequency response can be assumed to be flat). As a result, we can approximate the "channel" between each transmitter and receiver pair as a single complex coefficient (denoted by $h_{Rx,Tx}$).

With this in mind, we can represent the received signal at each of the receivers at sample $k$ (assuming flat-fading channel and carrier synchronization) with the following four equations:

$$y_1[k] = h_{11}x_1[k] + h_{12}x_2[k] + h_{13}x_3[k] + h_{14}x_4[k] + n_1[k] \tag{1}$$

$$y_2[k] = h_{21}x_1[k] + h_{22}x_2[k] + h_{23}x_3[k] + h_{24}x_4[k] + n_2[k] \tag{2}$$

$$y_3[k] = h_{31}x_1[k] + h_{32}x_2[k] + h_{33}x_3[k] + h_{34}x_4[k] + n_3[k] \tag{3}$$

$$y_4[k] = h_{41}x_1[k] + h_{42}x_2[k] + h_{43}x_3[k] + h_{44}x_4[k] + n_4[k] \tag{4}$$

We can then convert this system of four equations into a single matrix equation to work more efficiently with all streams of data at all times $k_1, k_2, k_3, ..., k_n$. These matrices are written below.

$$\boldsymbol{y} = \begin{bmatrix} y_1[k_1] & y_1[k_2] & \dots & y_1[k_n] \\ y_2[k_1] & y_2[k_2] & \dots & y_2[k_n] \\ y_3[k_1] & y_3[k_2] & \dots & y_3[k_n] \\ y_4[k_1] & y_4[k_2] & \dots & y_4[k_n] \end{bmatrix} \tag{5}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1[k_1] & x_1[k_2] & \dots & x_1[k_n] \\ x_2[k_1] & x_2[k_2] & \dots & x_2[k_n] \\ x_3[k_1] & x_3[k_2] & \dots & x_3[k_n] \\ x_4[k_1] & x_4[k_2] & \dots & x_4[k_n] \end{bmatrix} \tag{6}$$

$$\boldsymbol{n} = \begin{bmatrix} n_1[k_1] & n_1[k_2] & \dots & n_1[k_n] \\ n_2[k_1] & n_2[k_2] & \dots & n_2[k_n] \\ n_3[k_1] & n_3[k_2] & \dots & n_3[k_n] \\ n_4[k_1] & n_4[k_2] & \dots & n_4[k_n] \end{bmatrix} \tag{7}$$

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \tag{8}$$

With this new matrix representation of the data and channels, we can combine Equations 1, 2, 3, and 4 into the following single matrix equation.

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n} \tag{9}$$

## 1.2 Lab Requirements

With this new matrix representation of our 4X4 MIMO system, we will be able to take the following two approaches to mathematically separating the four unique messages simultaneously sent through the system:

1. Communicate BPSK data through the MIMO channel such that the transmitter cannot use any channel coefficients that we estimated.

2. Communicate BPSK data through the channel using appropriate transmite and receive processing to effectively parallelize the channels between the transmitter and receiver such that the transmitter knows the channel coefficients of the 4X4 channel.

Note that we are using a provided MATLAB function **MIMOChannel4x4.m** to simulate the 4x4 MIMO channel that takes in a matrix in which each row is one of the four unique message signals being transmitted.

## 1.3 Software Overview

We organized our software into two functions - **mmse_simulation.m** for the case where the transmitter does not know the channel parameters prior to sending the data across the channel and **SVD_simulation.m** for the case where the transmitter does know the channel parameters prior to sending the data across the channel. As illustrated in Figure 2, each of these functions call modular functions to estimate the channel matrix, send data through the channel, 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/4x4-simulated-mimo`. Each of the bolded functions in Figure 2 has its own file in this repository.

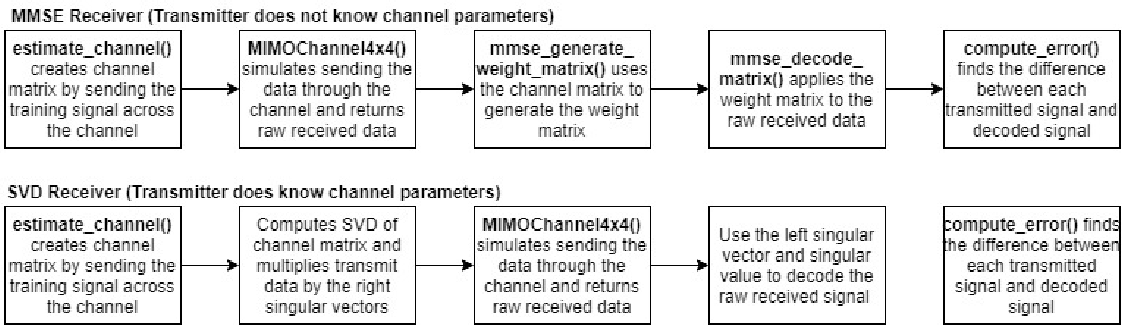

MMSE Receiver (Transmitter does not know channel parameters)
estimate_channel() creates channel matrix by sending the training signal across the channel
MIMOChannel4x4() simulates sending the data through the channel and returns raw received data
mmse_generate_weight_matrix() uses the channel matrix to generate the weight matrix
mmse_decode_matrix() applies the weight matrix to the raw received data
compute_error() finds the difference between each transmitted signal and decoded signal
SVD Receiver (Transmitter does know channel parameters)
estimate_channel() creates channel matrix by sending the training signal across the channel
Computes SVD of channel matrix and multiplies transmit data by the right singular vectors
MIMOChannel4x4() simulates sending the data through the channel and returns raw received data
Use the left singular vector and singular value to decode the raw received signal
compute_error() finds the difference between each transmitted signal and decoded signal

**Figure 2:** Software flowchart.

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

1. **[H, n0_var] = estimate_channel(x_train)** sends the training data through the channel and analyzes the resulting signal to determine the values in the channel matrix and the variance of the noise.

2. **Y = MIMOChannel4x4(X)** simulates sending the data through the channel. We received this function in the lab assignment.

3. **W = mmse_generate_weight_matrix(H, n0_var)** generates the weight matrix for the MMSE receiver by using the variance of the noise and the channel matrix.

4. **[x1, x2, x3, x4] = mmse_decode_matrix(received_data, weight_matrix)** applies the computed weight vectors to the received signal and output recovered signals for the MMSE receiver.

5. **[err, bits_x, bits_y] = compute_error(rx_data, tx_data)** 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**.

## 1.4 Data Preparation

To generate the training signal, $x_{train}$, we used MATLAB to create a vector of random BPSK data points (represented by $\pm 1$) and then up-sampled and convolved each data point by a pulse 40 units in width.

To generate the data signal, $x_{data}$, we used MATLAB to generate a 4 column matrix of 1024 random BPSK data points (represented by $\pm 1$) and then up-sampled and convolved each data point by a pulse 40 units in width.

We store both of these signals within a single file called **4x4MIMO.mat** in the data directory of our GitHub repository.

## 2 Decoding Methods

### 2.1 MMSE

We chose to use an MMSE receiver approach for the case where the receive and transmitter do not have a shared understanding of the channel characterization. This approach generates a weight matrix associated with the channel characterization the receiver generates (by analyzing the effect of the channel on a known training signal) and adds a regularization term. The MMSE approach solves the following optimization problem.

$$\hat{x}(\boldsymbol{y}) = \text{argmin}_{\boldsymbol{x}} ||\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}||^2 + \lambda ||x||^2 \tag{10}$$

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. We compute this matrix all at once, and then we can multiply each row with the received signal $y$ to yield the decoded signal associated with the row (for example, we can extract $x_1$ by multiplying the first row by $y$, $x_2$ by multiplying the second row by $y$, etc.).

In our implementation of the MMSE receiver, we computed the value of our regularization term ($\lambda$) by estimating the overall signal variance by computing the variance of a section of our training signal where neither transmitter is sending data (which theoretically only contains noise) received on one of the antennas.

### 2.2 SVD

In order to achieve the second goal of the lab in which **the transmitter has knowledge of the 4x4 MIMO channel estimation**, we will take advantage of the singular value decomposition (SVD).

#### 2.2.1 SVD Overview

Given an $m \times n$ matrix $X$, we can decompose it into the product of a series of three matrices as follows.

$$X = U\Sigma V^T = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ & & & \end{bmatrix} \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_n \\ | & | & & | \end{bmatrix}^T \tag{11}$$

Note that all the values in $\Sigma$ that are not along the diagonal of singular values are zero. If $X$ is a matrix where each column is a multidimensional vector that represents one "piece of data" (i.e. flattened representation of an image) each column in $U$ are our eigen-data vectors and provide an orthogonal basis on which to represent each of the data columns of $X$. Each of the singular values $\sigma_i$ in $\Sigma$ denote the significance of each $u_i$ and $v_i$. Note that it is always true that $\sigma_1 \geq \sigma_2 ... \geq \sigma_m$.

The significance of $V$ is a bit more difficult to explain. When each data column of $X$ represents a data point in time, each column in $V$, $v_i$, represents an eigen-timeseries; however, in the cases where each column of $X$ is an image, each column of $V^T$, $v_i^T$, weighs the mixtures of $u_i$ to reconstruct column $x_i$.

Another important feature of the SVD is that the matrices $U$ and $V$ are unitary, meaning that the following two equations hold.

$$UU^T = U^TU = I \tag{12}$$

$$VV^T = V^TV = I \tag{13}$$

The above will be useful when we are applying the SVD to our 4x4 MIMO commnications system. Lastly, it is important to note that the SVD is guaranteed to exist and is unique to a given data matrix.

### 2.2.2 Applying the SVD to MIMO Communications

Referring back to our the matrix equation that we are using to model our 4x4 MIMO system (Equation 9), we can substitute the SVD of $H$ for $H$ itself to obtain the following.

$$\begin{aligned} y &= Hx + n \\ &= U\Sigma V^{\dagger}x + n \end{aligned} \tag{14}$$

At this point let's assume that our system noise is negligible such that we can say $\boldsymbol{n} = 0$ which gives us the following.

$$y = U\Sigma V^{\dagger}x \tag{15}$$

Now, our goal is to solve for our input $x$ in terms of our output $y$. To do this let's pre-multiply both sides of the equation with $U^T$ giving us the equation below.

$$U^{\dagger}y = U^{\dagger}U\Sigma V^{\dagger}x \tag{16}$$

Next, if we recall that $U$ is unitary, we can apply Equation 12 to get the following.

$$\begin{aligned} U^{\dagger}y &= U^{\dagger}U\Sigma V^{\dagger}x \\ &= I\Sigma V^{\dagger}x \\ &= \Sigma V^{\dagger}x \end{aligned} \tag{17}$$

However, at this point we have $U^{\dagger}y = \Sigma V^{\dagger}x$ which is still a bit messy, so let's define a new step in the transmission process that is called **precoding** where we define $\tilde{x} = Vx$. We apply this transformation to the data before sending it across the channel. Note that this requires the transmitter to know the estimate of the channel matrix $H$ because it uses the matrix $V$ from the SVD. If we substitute $\tilde{x}$ into Equation 17, we get the following.

$$\begin{aligned} U^{T}y &= \Sigma V^{\dagger}\tilde{x} \\ &= \Sigma V^{\dagger}Vx \\ &= \Sigma Ix \\ &= \Sigma x \\ &= \begin{bmatrix} \sigma_1(x_1[k_1] & x_1[k_2] & \dots & x_1[k_n]) \\ \sigma_2(x_2[k_1] & x_2[k_2] & \dots & x_2[k_n]) \\ \sigma_3(x_3[k_1] & x_3[k_2] & \dots & x_3[k_n]) \\ \sigma_4(x_4[k_1] & x_4[k_2] & \dots & x_4[k_n]) \end{bmatrix} \end{aligned} \tag{18}$$

Note that between lines two and three in the derivation of Equation 18 above, we exploit the fact that $V$ is unitary by applying the identity $V^{\dagger}V = I$. Now, this expression is a lot simpler after we applied the **precoding** step. We can now use this expression to extract an estimate of each transmitted message ($x_{1-4}$) through the following steps.

1. Assuming that the transmit side has an estimate of the channel matrix $H$, calculate the SVD of $H$.
2. **Precode** the matrix of messages to transmit ($x$), using the $V$ matrix from the SVD, such that $\tilde{x} = Vx$.
3. Transmit the message signals in $\tilde{x}$ imultaneously.
4. Calculate $U^{T}y$ to obtain a matrix with the transmitted messages in each row ($x_i$) scaled by its respective singular value ($\sigma_i$).
5. Divide each row of the matrix calculated in the previous step by its respective singular value to obtain an estimate of the transmitted message.

Additionally, if we say $\tilde{y} = U^{\dagger}y$ we can rewrite our MIMO system as a series of parallel Gaussian channels (note that we take n = 0 in our processing).

$$\tilde{y}_1 = \sigma_1 x_1 + n_1 \tag{19}$$

$$\tilde{y}_2 = \sigma_2 x_2 + n_2 \tag{20}$$

$$\tilde{y}_3 = \sigma_3 x_3 + n_3 \tag{21}$$

$$\tilde{y}_4 = \sigma_4 x_4 + n_4 \tag{22}$$

This is to say that if the input is expressed in terms of a coordinate system defined by the columns of $V$ as we do by transmitting $\tilde{x} = Vx$ rather than $x$ (**precoding** step) and the output is expressed in terms of a coordinate system defined by the columns of $U$, then the relationship between the input and output is shown above and is drastically more simple.

## 2.3 Comparison of Methods

While both methods return a complete signal with no errors, both approaches have certain advantages and disadvantages. While the MMSE approach is more robust to changing conditions, it is a high-overhead process on the receiver side. The costs of these operations can be expensive, especially in cases with many antennas and many transmissions. Meanwhile, the SVD operation is much computationally simple and eliminates the overhead associated with sending and receiving the training signal, but it also requires that the channel characterization remain relatively constant.

### 2.3.1 Signal to Noise Ratio

To calculate the signal-to-noise ratio (SNR), we used the following equation:

$$\text{SNR} = 10\log_{10}\left(\left(\frac{\text{rms(signal)}}{\text{rms(noise)}}\right)^2\right) = 20\log_{10}\left(\frac{\text{rms(signal)}}{\text{rms(noise)}}\right) \tag{23}$$

Note that "signal" in the above equation is technically "signal + noise"; however, we assume that the power of the signal is much greater than the power of the noise such that $\text{Power}_{\text{signal}} + \text{Power}_{\text{noise}} \approx \text{Power}_{\text{signal}}$. Also note the the signal and noise were effectively "measured" in front of each of the receiver antennas.

| Signal | SVD Method SNR (dB) | MMSE Method SNR (dB) |
|---|---|---|
| $\text{Rx}_1$ | 108.12 | 107.99 |
| $\text{Rx}_2$ | 110.41 | 110.44 |
| $\text{Rx}_3$ | 104.85 | 104.81 |
| $\text{Rx}_4$ | 107.48 | 107.62 |
| Average | 107.72 | 107.72 |

**Table 1:** SNR at each of the receiver antennas of our 4x4 MIMO communications system.

Our SNRs are very similar for both methods; this makes sense because we used the same channel, which is simulated to have an extremely low noise level, and we recover each BPSK bit to a value of $\pm 1$ with no errors.

## 3 Results

All of the data associated with this lab assignment is simulated, and the behavior of the channel is deterministic.

## 3.1 MMSE Results

The figure below depicts the first ten pulses of the initial transmitted data from antenna one in the case of the MMSE approach. Because the transmitter has no knowledge of the channel characteristics, it simply sends the raw pulses across the channel.

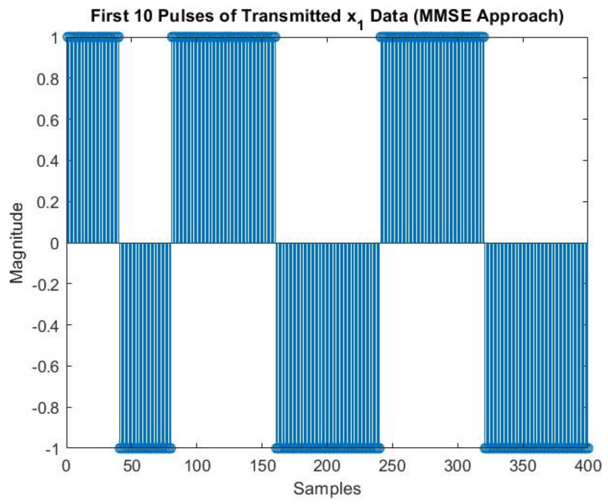


**Figure 3:** First 10 Pulses of MMSE Transmit Data from Antenna 1.

The following figure shows the first ten pulses of the raw received data received by antenna 1 in the MMSE case - the signal has been distorted and includes components of all four signals.

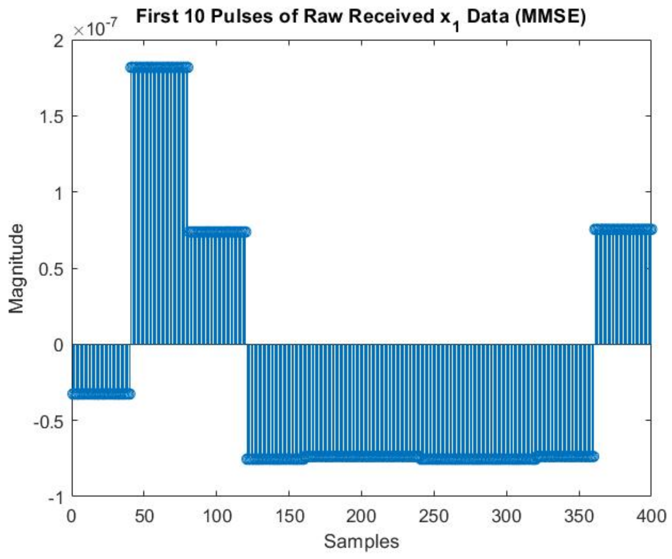
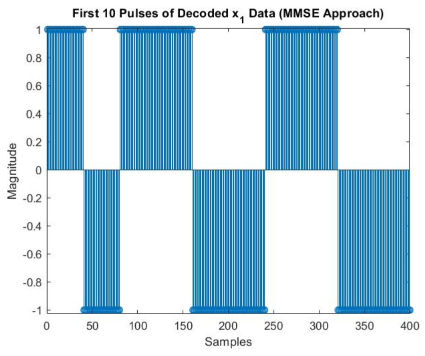
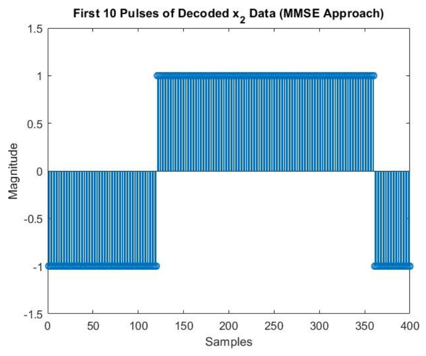


**Figure 4:** First 10 Pulses of MMSE Raw Received Data from Antenna 1.

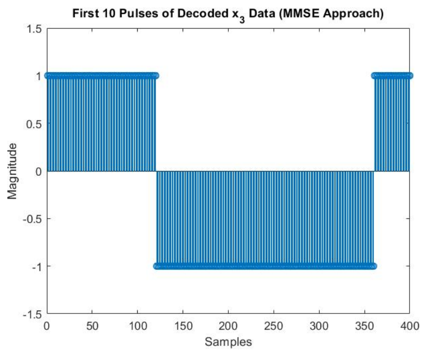
The following figures show the first ten pulses of each of the decoded signals in the MMSE case.



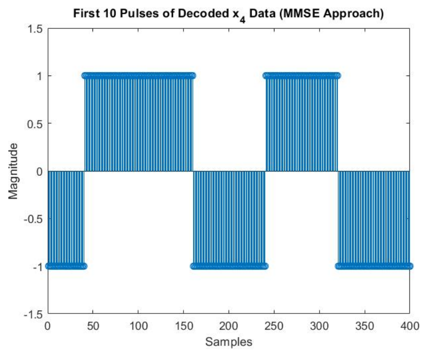
**(a)** First 10 Pulses of MMSE Decoded $x_1$ signal.



**(b)** First 10 Pulses of MMSE Decoded $x_2$ signal.



**(c)** First 10 Pulses of MMSE Decoded $x_3$ signal.



**(d)** First 10 Pulses of MMSE Decoded $x_4$ signal.

**Figure 5:** First 10 pulses of each of the signals decoded with the MMSE method.

## 3.2 SVD Results

The figure below depicts the first ten pulses of the initial transmitted data from antenna one in the case of the SVD approach. Because the transmitter has knowledge of the channel characteristics, it modifies the transmit data accordingly to send $\tilde{x} = Vx$.

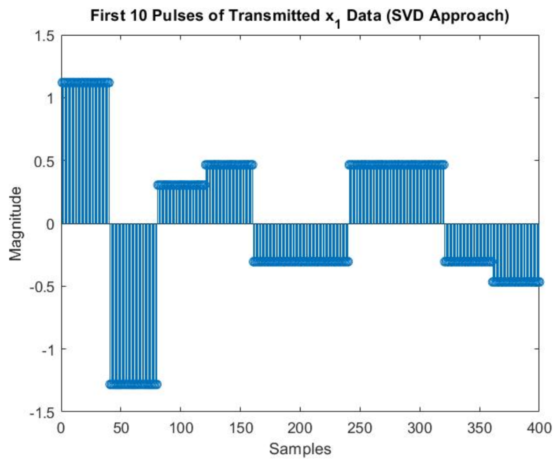


**Figure 6:** First 10 Pulses of SVD Transmit Data from Antenna 1.

The following figure shows the first ten pulses of the raw received data received by antenna 1 in the

SVD case - the signal has been distorted and includes components of all four signals.

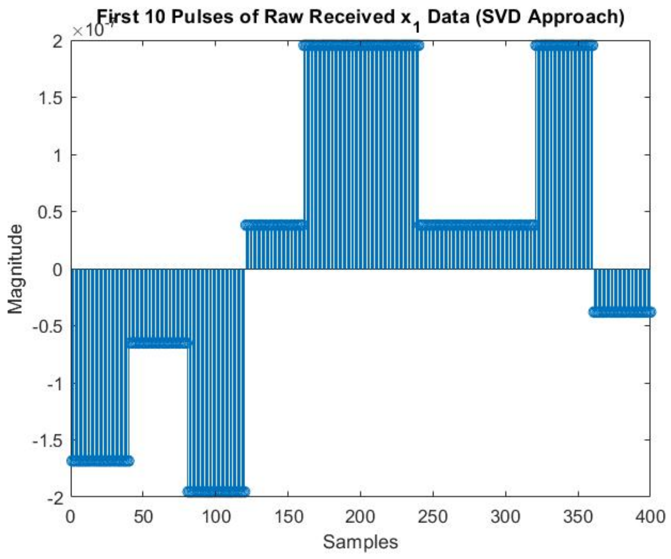
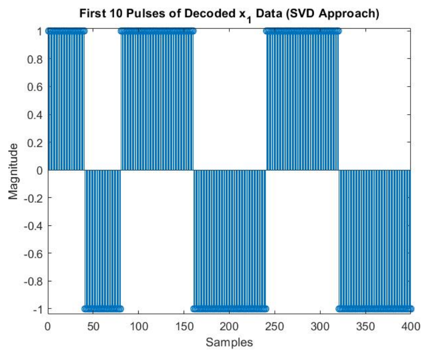
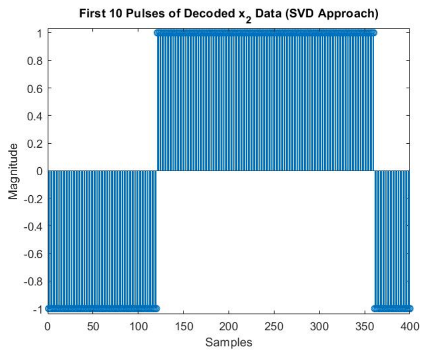


**Figure 7:** First 10 Pulses of SVD Raw Received Data from Antenna 1.

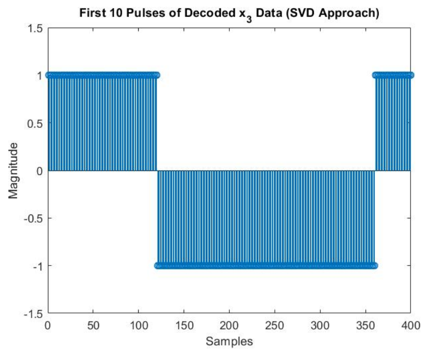
The following figures show the first ten pulses of each of the decoded signals in the MMSE case.



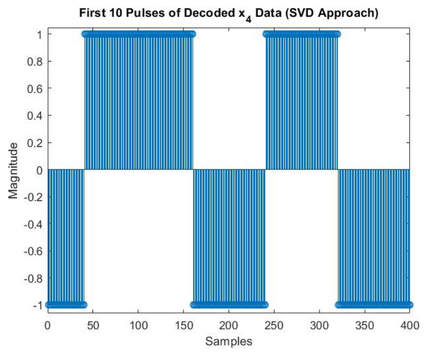
**(a)** First 10 Pulses of SVD Decoded $x_1$ signal.



**(b)** First 10 Pulses of SVD Decoded $x_2$ signal.



**(c)** First 10 Pulses of SVD Decoded $x_3$ signal.



**(d)** First 10 Pulses of SVD Decoded $x_4$ signal.

**Figure 8:** First 10 pulses of each of the signals decoded with the SVD method.

## 4 References

[1] Tse D. P. Viswanath. Fundamentals of Wireless Communication Chapter 7.

[2] Brunton S., Singular Value Decomposition (SVD): Mathematical Overview. https://www.youtube.com/watch?v=nbBvu