THE CHINESE UNIVERSITY OF HONG KONG

Real-Time Wireless Acoustic Communication System

SIN Cheuk Wing

香港中文大學電子工程學系
DEPARTMENT OF ELECTRONIC ENGINEERING

Real-Time Wireless Acoustic Communication System

Author: SIN Cheuk Wing

Student ID: 1155147705

Supervisor: Prof. W. K. Ma

Associate Examiner: Prof. C. R. Huang

A project report presented to the
Chinese University of Hong Kong
In partial fulfillment of the
Degree of Bachelor of Engineering

Department of Electronic Engineering
The Chinese University of Hong Kong

April 2022

Abstract

Nowadays, technology is developing rapidly. Almost everyone has their own mobile phone or computer and surf the internet every day. Communication are being essential on our life. Therefore, this project aims to helps to understand more in communication. Back to the start, the objective of this project is to build the acoustic communication system.

In this project, MATLAB is used on programming and data analysis. In hardware part, microphone is used as the input, the receive part of the system, as well as the speaker are used as the output, the transmit part of the system. Some communication technics are used, including Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), Quadrature Amplitude Modulation (QAM) and Zero-forcing.

Regarding the project result, it is successfully to simulation those technics. ASK, PSK, FSK, signal instructive and destructive interference are successfully using the acoustic system. But due to COVID-19, Zero-forcing and QAM are just simulate fully software.

Acknowledgements

I would like to give my warmest thanks to my supervisor Prof W.K.Ma. He provided a lot of idea and notes for my project. I am thankful for his support and glad to choose this topic.

I would also like to acknowledge to Gavin. His guidance and advice helped me a lot through my work. He made my job possible.

Table of Content

1. Introduction

This final year project is to build an acoustic communication system. It is a lab-based project. In this project, MATLAB is used in programming for the purpose of component control and data analysis. Moreover, microphone is used as system input and speaker is used as system output. As well as audio interface is used to connect microphone speaker with the computer.

In semester 1, 1 microphone and 1 speaker are used to simulate a single input single output, SISO system. Three basic communication schemes are used, including Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK). In simulation, the system converts photo to signal and send the signal from microphone to speaker.

In semester 2, there are 2 phases of the work. First, 2 microphones and 1 speaker are used to simulate the signal interference. In simulation, same signal is sent by 2 speaker and see the result of its interference. Then, 2 microphones and 2 speakers are used to simulate the zero-forcing beamforming in multiple input multiple output, MIMO system.

This report begins with a detailed background on the project. Then, methodology and implementation will be presented. As well as the simulation results. Lastly, there will be discussion and conclusion on this project.

2. Background

2.1 Signal Basic

The basic signal is normally sine wave or cosine wave. The component includes the amplitude A_c, the carrier frequency f_c and the phase Φ.

$$s(t) = A_c \cos(2\pi f_c t + \Phi)$$

In propagation, there will be some time delay φ, attenuation α and some noise η.

$$y(t) = \alpha A_c \cos[2\pi f_c(t) + \emptyset + \varphi] + \eta(t)$$

Since this project is to build the acoustic communication system, we can let the speed of the signal is equal to the speed of sound, which is 343m/s. As the project setup are pre-set, we know the distance for propagation. Therefore, we can use the velocity formula to calculate the time needed.

The velocity formula is as shown below:

$$v = \frac{d}{\Delta t} \qquad i.e.\ \Delta t = \frac{d}{v}$$

After we have calculated the time needed, we can easily find the phase shifted due to propagation by the formula shown below:

$$\varphi = 2\pi f_c \Delta t$$

2.2 Communication modulation schemes [1]

Communication modulation is the process that convert data into signal. This project is focus on digital modulation, which converts binary data 0 and 1 to the carrier wave.

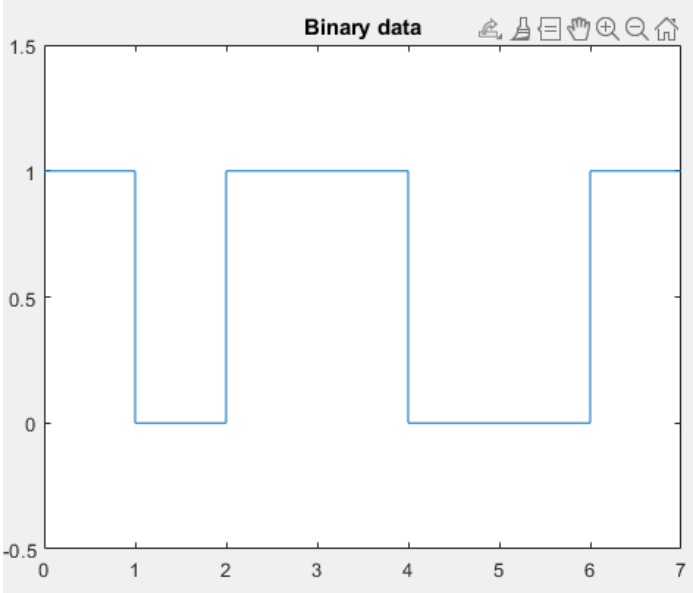

Figure 1: Binary data

Figure 1 shows the example of the binary data. It has 7 bits, they are [1,0,1,1,0,0,1]. Digital modulation includes single carrier modulation, Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK) and multicarrier modulation, Quadrature Amplitude Modulation (QAM).

In single carrier modulation, they have the same receiver model in demodulation, a two-path correlation receiver.

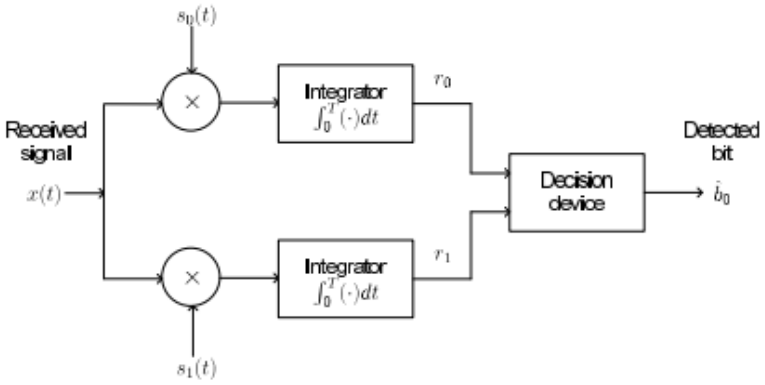

Figure 2: The two-path correlation receiver [1]

Figure 2 shows the block diagram of the two-path correlation receiver. $x(t)$ is the received signal, which $x(t) = s(t) * \eta(t)$, $s(t)$ the transmitted signal and $\eta(t)$ is the noise. In ideal case, $\eta(t)$ is equal to 0. $s_0(t)$ and $s_1(t)$ are the carrier signal for bit 0 and 1 respectively. T is the time interval. In this project T is normally equals 44100 because the frequency of the speaker is 44100Hz, it means there are 44100 array elements in signal for 1second. r_0 and r_1 is the value after integration.

The mathematical expression of r_0 and r_1 are as below:

$$r_0 = \int_0^T x(t) * s_0(t)\, dt$$

$$r_1 = \int_0^T x(t) * s_1(t)\, dt$$

The decision is made by comparing the r_0 and r_1.

The decision rule is as follow:

$$b = 0, r_1 - r_0 < \lambda$$

$$b = 1, r_1 - r_0 \geq \lambda$$

Where λ is threshold value, it will be changed when using different scheme.

2.2.1 Amplitude Shift Keying (ASK)

ASK is the modulation that switches the carrier wave on and off to represent bit 1 and 0 respectively.

The mathematical expression of ASK carrier signal is as follow:

For bit 0: $s_0(t) = 0$

For bit 1: $s_1(t) = A_c \cos(2\pi f_c t)$

The example of ASK is as shown below:

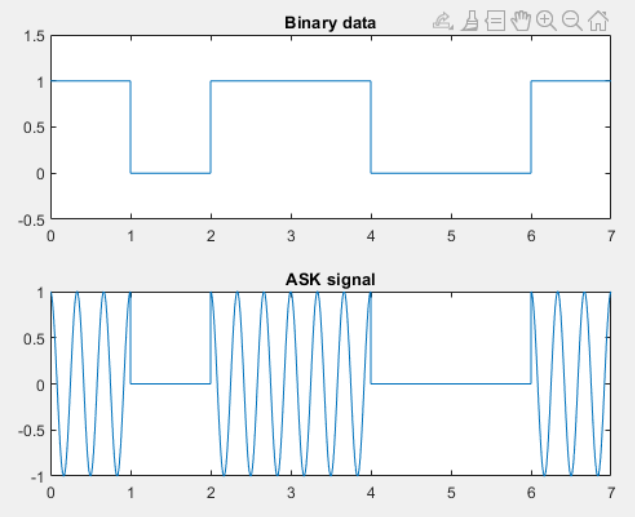

Figure 3: ASK signal example

In Figure 3, A_c is 1 and f_c is 3, it shows the ASK modulated signal with respect to figure 1 bit data. When the bit is 0, the modulated signal is just simply equals to 0, which is off state; When the bit is 1, the modulated signal becomes a wave, which is on state.

Regarding the demodulation of ASK signal, let begin with the threshold value λ. We can determine it by predicting the result of $r_1 - r_0$ on receiving bit 0 and 1 respectively.

Assume bit 0 is received,

$r_0 = \int_0^T 0 * 0 \, dt = 0$

$r_1 = \int_0^T 0 * A_c \cos(2\pi f_c t) \, dt = 0$

$r_1 - r_0 = 0$

Assume bit 1 is received,

$r_0 = \int_0^T A_c \cos(2\pi f_c t) * 0 \; dt = 0$

$r_1 = \int_0^T A_c \cos(2\pi f_c t) * A_c \cos(2\pi f_c t) \; dt$

$r_1 = \int_0^T \frac{A_c^2}{2} [1 + \cos(4\pi f_c t) \; dt \cong \frac{A_c^2 * T}{2}$

$r_1 - r_0 = \frac{A_c^2 * T}{2}$

Since the integration of cosine is equal to sine and its average is equal to 0, it is negligible compared to the first term and we can just simply ignore it.

Therefore, the threshold value can be set within 0 and $\frac{A_c^2 * T}{2}$. In actual case, there will be noise in communication, so threshold value should not be set to exactly 0 or $\frac{A_c^2 * T}{2}$. It is better to be set to the middle point of that range, which is $\frac{A_c^2 * T}{4}$.

2.2.2 Phase Shift Keying (PSK)

PSK is the modulation uses 0° and 180° phase shift in carrier signal to represent bit 1 and 0 respectively.

The mathematical expression of PSK transmitted signal is as follow:

For bit 0: $s_0(t) = A_c \cos(2\pi f_c t + \pi) = -A_c \cos(2\pi f_c t)$

For bit 1: $s_1(t) = A_c \cos(2\pi f_c t + 0) = +A_c \cos(2\pi f_c t)$

The example of PSK signal is as shown below:

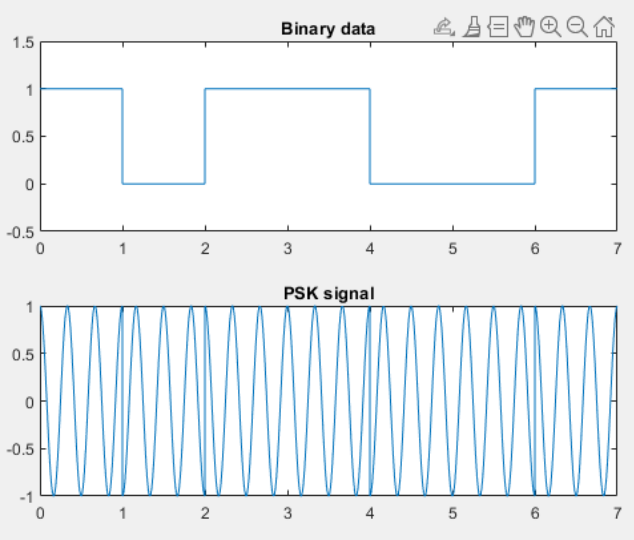

Figure 4: PSK signal example

In Figure 4, A_c is 1 and f_c is 3, it shows the PSK modulated signal with respect to figure 1 bit data. We can see the modulated bit 0 and bit 1 is similar but with different phase.

Regarding the demodulation of PSK signal, let begin with the threshold value λ. It is similar to the calculation on ASK demodulation.

Assume bit 0 is received,

$$r_0 = \int_0^T -A_c \cos(2\pi f_c t) * -A_c \cos(2\pi f_c t)\ dt$$

$$r_0 = \int_0^T A_c^2 \cos^2(2\pi f_c t)\, dt \cong \frac{A_c^2 * T}{2}$$

$$r_1 = \int_0^T -A_c \cos(2\pi f_c t) * A_c \cos(2\pi f_c t)\, dt$$

$$r_1 = \int_0^T -A_c^2 \cos^2(2\pi f_c t)\, dt \cong -\frac{A_c^2 * T}{2}$$

$$r_1 - r_0 = -A_c^2 * T$$

Assume bit 1 is received,

$$r_0 = \int_0^T A_c \cos(2\pi f_c t) * -A_c \cos(2\pi f_c t)\ \ dt$$

$$r_0 = \int_0^T -A_c^2 \cos^2(2\pi f_c t)\, dt \cong -\frac{A_c^2 * T}{2}$$

$$r_1 = \int_0^T A_c \cos(2\pi f_c t) * A_c \cos(2\pi f_c t)\, dt$$

$$r_1 = \int_0^T A_c^2 \cos^2(2\pi f_c t)\, dt \cong \frac{A_c^2 * T}{2}$$

$$r_1 - r_0 = A_c^2 * T$$

Therefore, the threshold value can be set within $-A_c^2 * T$ and$A_c^2 * T$. The threshold value should be set at the middle point which is 0 to avoid the noise effect during communication.

2.2.3 Frequency Shift Keying (FSK)

FSK is the modulation uses 2 different carrier frequency f_0 and f_1 in carrier signal to represent bit 1 and 0.

The mathematical expression of FSK transmitted signal is as follow:

For bit 0: $s_0(t) = A_c \cos(2\pi f_0 t)$

For bit 1: $s_1(t) = A_c \cos(2\pi f_1 t)$

The example of FSK signal is as shown below:

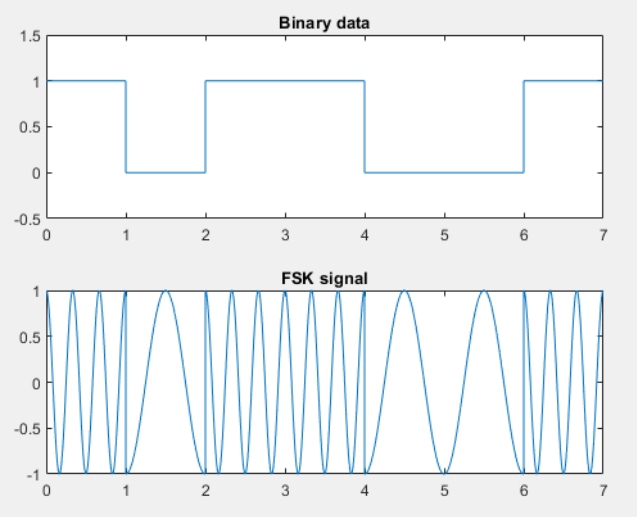

Figure 5: FSK signal example

Regarding the demodulation of FSK signal, let begin with the threshold value λ. It is similar to the calculation on PSK demodulation.

Assume bit 0 is received,

$$r_0 = \int_0^T A_c \cos(2\pi f_0 t) * A_c \cos(2\pi f_0 t)\ dt$$

By the trigonometric identities:

$$\cos(a)\cos(b) = \frac{1}{2}[\cos(a+b) + \cos(a-b)]$$

$$r_0 = \int_0^T A_c^2 \cos^2(2\pi f_0 t)\, dt \cong \frac{A_c^2 * T}{2}$$

$$r_1 = \int_0^T A_c \cos(2\pi f_0 t) * A_c \cos(2\pi f_1 t)\, dt$$

$r_1 = \int_0^T A_c^2 \left[\cos(2\pi(f_0+f_1)t) - \cos(2\pi(f_0-f_1)\,t)\right]dt \cong 0$

$r_1 - r_0 = -\frac{A_c^2 * T}{2}$

Assume bit 1 is received,

$r_0 = \int_0^T A_c \cos(2\pi f_1 t) * A_c \cos(2\pi f_0 t)\, dt$

$r_0 = \int_0^T A_c^2 \left[\cos(2\pi(f_0+f_1)t) - \cos(2\pi(f_0-f_1)\,t)\right]dt \cong 0$

$r_1 = \int_0^T A_c \cos(2\pi f_1 t) * A_c \cos(2\pi f_1 t)\ \ dt$

$r_1 = \int_0^T A_c^2 \cos^2(2\pi f_1 t)\, dt \cong \frac{A_c^2 * T}{2}$

$r_1 - r_0 = \frac{A_c^2 * T}{2}$

Therefore, the threshold value can be set within $-\frac{A_c^2 * T}{2}$ and$\frac{A_c^2 * T}{2}$. Similar to FSK, the threshold value should be set at the middle point which is 0 to avoid the noise effect during communication.

2.2.4 Quadrature Amplitude Modulation (QAM)

QAM is a modulation that use two carrier waves to transmit two set of data. Two carrier waves are in 90º out of phase.

By the trigonometric identities:

$\cos\left(x+\frac{\pi}{2}\right) = -\sin(x)$ and $\sin\left(x+\frac{\pi}{2}\right) = \cos(x)$

The 90º out of phase is equal to one carrier wave is cosine, the another is sine wave.

The two set of data are as known as in phase component and quadrature component, or we can say two set of data is stored in real part and imaginary part.

The mathematical expression of QAM transmitted signal is as follow:

$x(t) = Re\{s(t) * e^{j2\pi f_c t}\}$, where $s(t) = m_1(t) + jm_2(t)$

By Euler's formula

$e^{jx} = \cos(x) + jsin(x)$

$x(t) = Re\{[m_1(t) + jm_2(t)] * [\cos(2\pi f_c t) + jsin(2\pi f_c t)]\}$

$x(t) = m_1(t)\cos(2\pi f_c t) - m_2(t)sin(2\pi f_c t)$

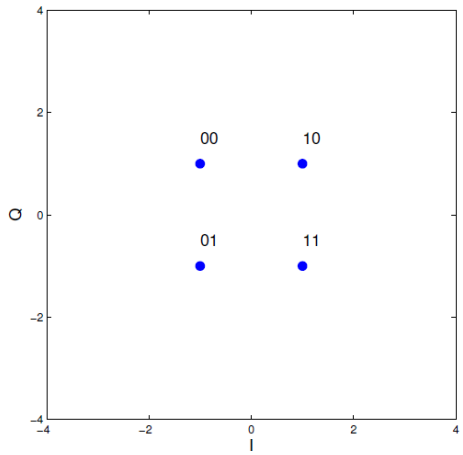

Figure 6: 4-ary QAM Constellation [1]

In this project, 4-ary QAM is used. Figure 6 shows the 4-ary QAM constellation, which $A = \{\pm 1, \pm j\}$. It means there are only 1bit, 0 and 1 in the real part and the imaginary

part. Therefore, we can use ASK, PSK or FSK in each part in QAM, and normally ASK is used.

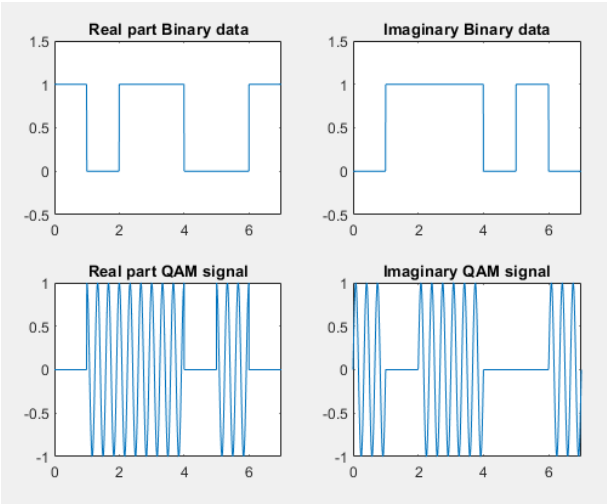

Figure 7: Separated QAM signal example

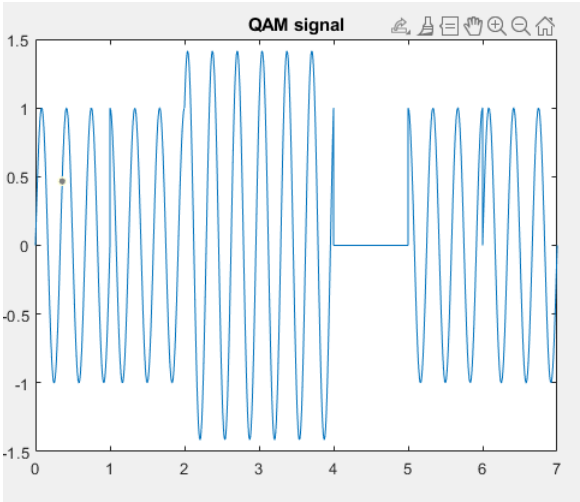

Figure 8: QAM signal example

Figure 7 shows the QAM signal in real part and imaginary part separately, it has the same modulation with ASK but with two carrier signal, sine wave and cosine wave.

Figure 8 shows the QAM signal which combines real part and imaginary part.

Regarding the demodulation of QAM signal, we can use 2 one-path correlation receivers to demodulate the QAM signal.

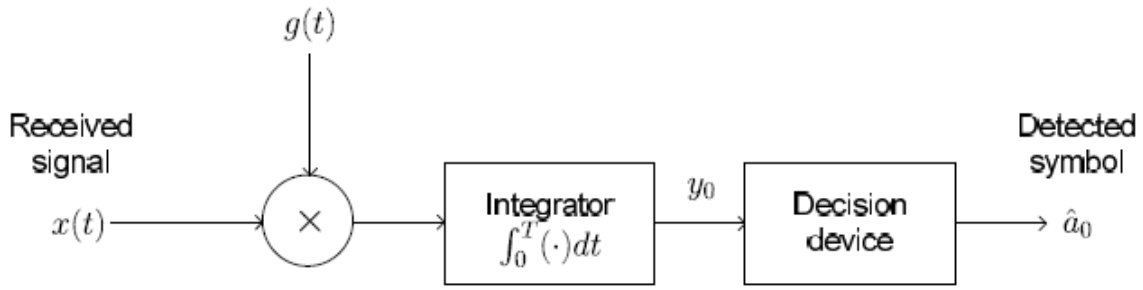

Figure 9: A one-path correlation receiver

Figure 9 shows the one-path correlation receiver. $x(t)$ is the received signal, which $x(t) = s(t) * \eta(t)$, s(t) is the transmitted signal and $\eta(t)$ is the noise. Since there are two carrier waves carrying two signal, we need to demodulation twice, one for real and

another for imaginary.

For the real part, $g(t) = 2\cos(2\pi f_c t)$

$y_0 = \int_0^T [m_1(t)\cos(2\pi f_c t) - m_2(t)sin(2\pi f_c t)] * 2\cos(2\pi f_c t)\, dt$

$y_0 = \int_0^T [2m_1(t)\cos^2(2\pi f_c t) - 2m_2(t)\cos(2\pi f_c t)\, sin(2\pi f_c t)]\, dt$

By the trigonometric identities:

$\cos(a)\cos(b) = \frac{1}{2}[\cos(a+b) + \cos(a-b)]$

$y_0 = \int_0^T [m_1(t) + m_1(t)\cos(4s\pi f_c t) - 2m_2(t)\cos(2\pi f_c t)\, sin(2\pi f_c t)]\, dt$

Similar to the calculation on ASK, after integration, the term $m_1(t)\cos(4s\pi f_c t)$ and $2m_2(t)\cos(2\pi f_c t)\, sin(2\pi f_c t)$ is negligible compared to the $m_1(t)$, we can simply ignore it. Thus, $y_0 \cong m_1(t) * T$. Lastly, we can find $m_1(t)$.

For the imaginary part, $g(t) = 2\sin(2\pi f_c t)$

$y_0 = \int_0^T [m_1(t)\cos(2\pi f_c t) - m_2(t)sin(2\pi f_c t)] * 2\sin(2\pi f_c t)\, dt$

$y_0 = \int_0^T [2m_1(t)\sin(2\pi f_c t)\cos(2\pi f_c t) - 2m_2(t)\sin^2(2\pi f_c t)] * dt$

Thus, $y_0 \cong m_2(t) * T$. Lastly, we can find $m_2(t)$.

2.3 Correlation [2]

Correlation means the relationship between two signals. If we want to find the similarity between two signals, we can calculate the correlation.

The mathematical expression of Correlation is as follow:

$R(\tau) = \int_{-\infty}^{\infty} x(t) * y^*(t-\tau)dt$

The correlation can also be computed by resorting to graphical technique. When two signal are overlapping, we can add them together for every point of the signal. One thing to notice is that the correlation will become largest when two signal are overlapping in phase if two signal are the same. By this identity, we can use the same signal and move it from left to right and calculate the correlation to check the time delay of the signal.

2.4 Signal interference

Signal interference is a phenomenon when there two signal wave or more. When two signal received at the same time, their effect add together.

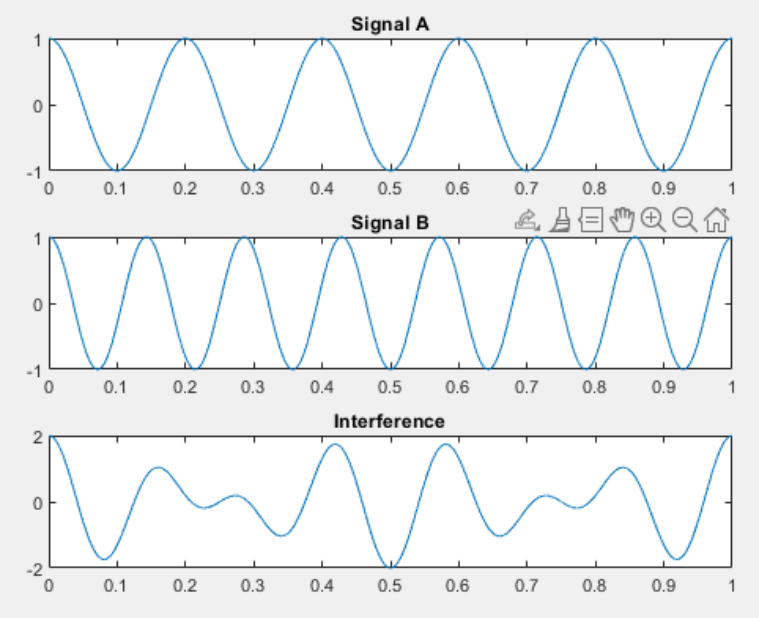

Figure 10: Signal interference

Figure 10 shows the example of signal interference. The interference of signal A and B are the sum of the value of every time domain. Since figure 10 signal A and B are randomly generated, the received signal becomes the new signal with new amplitude and frequency. If signal A and B are the in phase or with 180° phase different, constructive and destructive interference will be occurred.

2.4.1 Constructive interference

If two signals are in phase, the received signal will be 2 times the original signal.

Assume two signals are exactly the same: $s_A = s_B = cos(2\pi f_c t)$ and f_c = 5Hz

The received signal will be doubled: $s_{received} = 2cos(2\pi f_c t)$

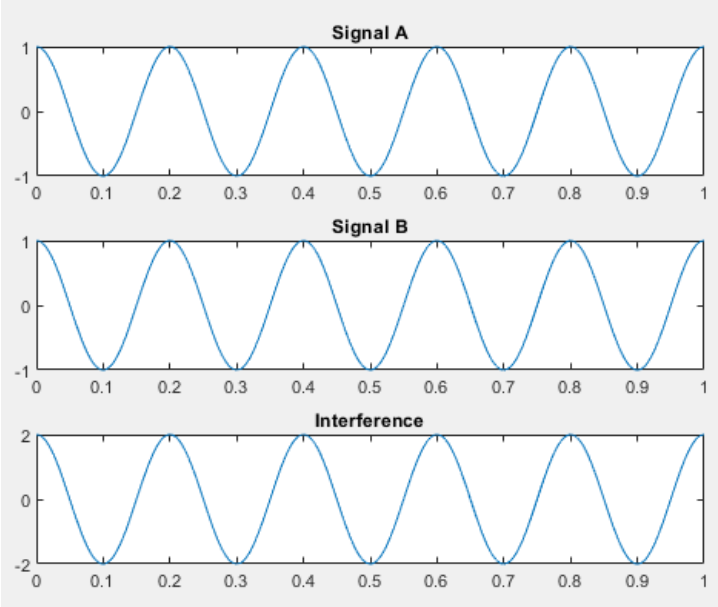

Figure 11: Constructive interference

Figure 11 shows the example of constructive interference. Signal A and signal B are in phase, the constructive interference occurred. We can see the received signal's frequency and phase remain unchanged, but with doubled amplitude. Therefore, we can say the received signal is the same signal with the transmit but its clearer. It means that the signal-to-noise ratio is increased. As a result, we can use the constructive interference make the transmitted signal clearer.

2.4.2 Destructive interference

If two signal are 180° out of phase, the signal will be cancelled by another signal.

Assume two signals are exactly the same:

$s_A = cos(2\pi f_c t)$ and $s_B = cos(2\pi f_c t + \pi)$, f_c = 5Hz

The received signal will be doubled: $s_{received} = 0$

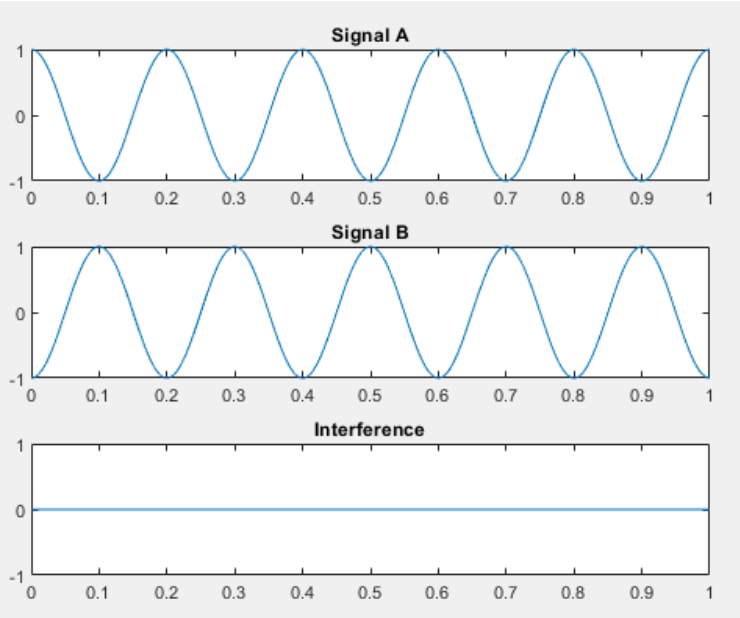

Figure 12: Destructive interference

Figure 12 shows the example of destructive interference. Signal A and signal B are 180° out of phase. It shows that when signal A achieve +ve peak, signal B is -ve peak; when signal A achieve -ve peak, signal B is +ve peak. We can say $s_A = -s_B$. Therefore, when they meet each other, they will be cancelled out by another. We can use destructive interference to cancel out the useless or unwanted signal.

2.5 Zero-forcing beamforming

Zero-forcing beamforming is a method used in multiple input multiple output, MIMO system.

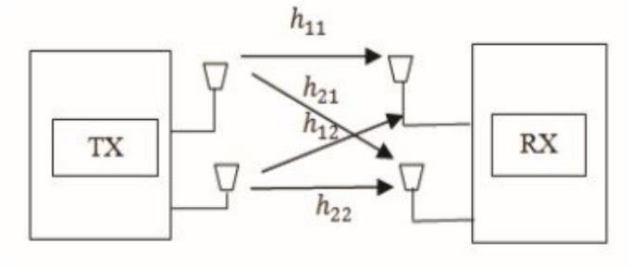

Figure 13: 2x2 MIMO model [3]

Figure 13 shows the MIMO system with 2 input and 2 output. h means the channel with includes the attenuation and the phase shifted due to the time used.

The mathematical express of h are as shown below:

$$h = \alpha e^{j\varphi} \text{ and } H = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

As figure 13 shows, Rx1 and Rx2 will receive both Tx1 and Tx2 with the corresponding h. The aim of zero-forcing beamforming is to cancel out the unwanted signal by nulling the other interference. For example, Rx1 will only receive Tx1; Rx2 will only receive Tx2

To perform zero-forcing, we need to first find out 4 channels h first. It can be done by transmit data one by one. First, Tx1 transmit a basic signal. Then, record it on Rx0 and Rx1 and then we can calculate the h_{11} and h_{21} by comparing the received signal and the ideal signal. After that, Tx2 transmit a basic signal. Similarly, we can calculate h_{12} and h_{22}. After finding all channel, the MIMO can perform the zero-forcing. Before transmitting signal, pre-coding is needed. The modulation scheme is similar to QAM for each signal. Let s_1 and s_2 be the signal sent in Tx1 and Tx2, and they are

modulated by QAM. They are then needed to do zero-forcing precoding by multiple the invert matrix of H. i.e., $x(t) = H^{-1}s(t)$, where $x(t) = [x_1\ x_2]^\top\ and\ s(t) = [s_1\ s_2]^\top$. Tx1 will send x_1 and Tx2 will send x_2. After received the signal, the surrounding environment will multiply H while signal transmitting. i.e. the received signal $y(t) = Hx(t)$, where $y(t) = [y_1\ y_2]^\top$.

By the matrix identity, $HH^{-1} = I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

y(t) then become $y(t) = HH^{-1}s(t) = s(t)$. So, we get $y_1 = s_1$ and $y_2 = s_2$. As the result, it has cancelled out all other interference. Which Rx1 is only received s_1 and Tx2 is only received s_2. At the end of the receiver part, we can simply use QAM demodulation scheme to get the binary data.

3. Methodology & Implementation

3.1 Simulation set-up

3.1.1 Software

1. MATLAB

MATLAB is a programing and numeric computing platform. It has been widely used in the field of engineering and science. It provides the function on data analysis, graphics, programming, app building, web deployment, parallel computing, etc.

In this project, MATLAB has been used on signal processing, the signals are analyzed by MATLAB to simulate the communication system. MATLAB has also been used in programming. The hardware is controlled by the script on MATLAB.

3.1.2 Hardware

1. Microphone

Figure 14: Microphone

Figure 14 shows the microphone used in the project. It is used as the input of the communication system which receive the sound. Two microphones are used to simulate multiple input.

2. Speaker

Figure 15: Speaker

Figure 15 shows the speaker used in the project. It is used as the output of the communication system which transmit sound. Two speakers will be used to simulate multiple output. The speaker's frequency is 44100Hz. It means that it needs 44100 array elements in MATLAB coding. If the bit rate is equal to 1, the length of the modulated signal on every bit should equal to 44100.

3. Audio interface

Figure 16: Audio interface

Figure 16 shows the audio interface used in the project. It is used to connect speaker and microphone with the computer in order to script the speaker and microphone.

4. Setup

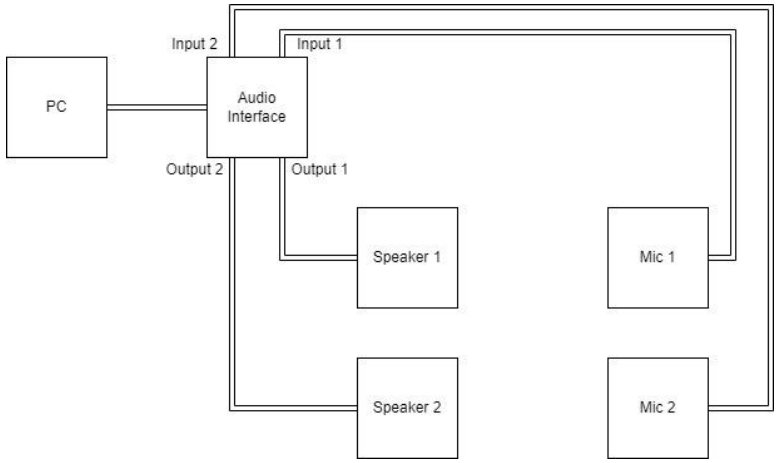

Figure 17: System block diagram

Figure 17 is the system block diagram; it shows the connection between each hardware. 2 speakers and 2 microphones have been placing in 4 conner of the rectangle but the length and width can be modulated.

3.2 Procedure

	Description	Date and Time Completed
1	ASK, PSK, FSK simulation	Semester 1
2	Data research on signal interference	January, 2022
3	Constructive and destructive simulation	February, 2022
4	Data research on Zero-forcing	Marth, 2022
5	Zero-forcing simulation	April, 2022

The table above shows the work procedure and its completion time. In semester 1, ASK, FSK and PSK simulation had been worked out, including transmitting random bit and photo. By changing the carrier frequency, bit rate, amplitude, we can see the different effect on communication.

In this semester, signal interference and zero-forcing have been mainly working on.

Regarding signal interference, 2x1 communication system have been used on simulation. 1 microphone and 2 speakers are used. 2 speakers output the same signal and observe the effect on received signal. By theoretical calculation and trial and error, the phase shift is changed to perform different interference while transmitting. Then, the zero-forcing simulation have been worked out after some data research. This simulation unlike the previous simulation, it is simulated fully by software, since I cannot use the DSP-Lab due to the COVID-19.

4. Simulation results

4.1.1 ASK, PSK, FSK simulation

To simulate ASK, FSK and PSK modulation, two program have been worked out. A SISO system is used, which use only 1 microphone and 1 speaker in simulation. The first program is first randomly generate a set of binary data by the MATLAB function randi(). 2 bits [1,0] are added in the beginning of the data. These 2 bits are used to the correlation in receiver part in order to find the time delay duo to propagation. Then, the binary data is modulated by the corresponding scheme. After modulation, the signal is sent from the speaker to microphone. Then, the program will demodulate the received signal using corresponding scheme. As the property, the received signal will be started in the 2times of the array, which means the 44100 elements of 1st time receiving will be all zero. Therefore, we need to cancel it out before analyzing the signal.

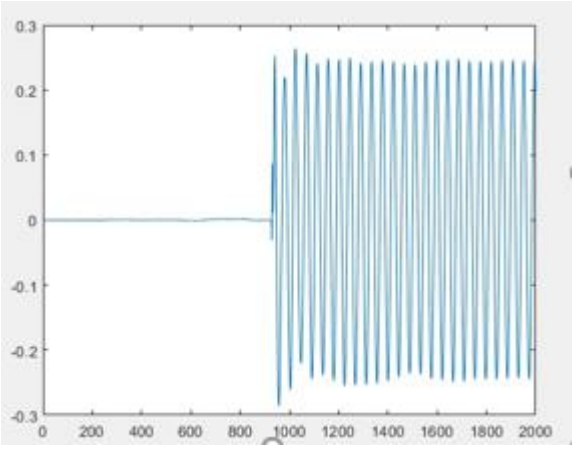

Figure 18: The beginning part of the received signal

Figure 18 shows the beginning part of the received signal. Besides the property, there are some useless data due to the propagation delay. Since the first 2 bits must be [1,0], we can calculate the correlation of bit 1 carrier signal with fist 88200 elements. The point when the correlation is largest is the time delay and we can delete all information

in front of the delay. After canceled out the delay and unwanted elements, the program will normalize the signal to the amplitude, normally it is equal to 1. In order to normalize the signal, the signal is divided by its maximum value. Then the program will start demodulating the signal using the corresponding scheme. The threshold value λ for PSK and FSK is set as 0 and for ASK is set as $\frac{44100}{4*b}$, where b means the bit sent per second. Lastly, demodulated bit will be plotted with the original bit to check whether it is correct or not.

Figure 19: Bit Checking

Figure 19 shows the checking example, we can see there are green for bit 2, which means the bit is different from the original. Otherwise, the red circle will cover the green one if it is the same

After the first program to try the modulation scheme, the similar program have been used to transmit photo.

Figure 20: Lenna

Figure 20 shows the photo Lenna, that will be used in simulation. The program will first convert the Lenna to the photo in gray. In data view, the data reduce the dimension from 3 to 1. Then the bit data is reshaped to 1 row and added [1,0] in the beginning. After that, the bit data is modulated by the corresponding scheme. The signal is then transmitted to microphone and received. The received signal is then demodulated and reshaped as it original size instead of a row. Lastly, the received photo will be print out in computer.

4.1.2 ASK, PSK, FSK simulation result

Refer to figure 18, the delay element is normally 700~1000 which is make sense in theoretical calculation. As well as the received photo are the same or similar to the original one.

Figure 21: photo under resolution level Q=2

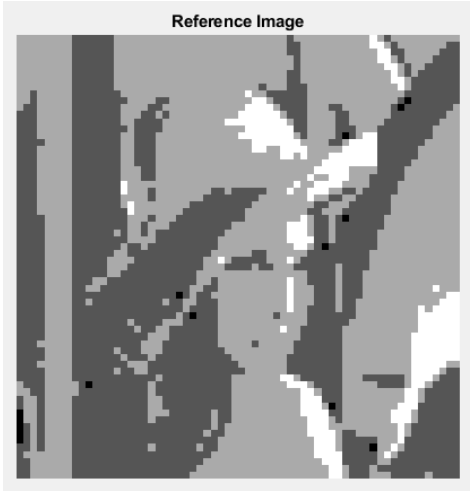

Figure 22: photo under resolution level Q=4

Figure 23: photo under resolution level Q=16

Figures 21 22 and 23 show the received photo under resolution level 2 4 and 16 respectively. It is obvious that the photo is clearer when using a higher resolution, but it will also cause the signal to become longer. To conclude, the simulation on ASK PSK and FSK are both successful.

4.2.1 Signal interference simulation

To check the effect on signal interference, a 2x1 SIMO system is simulated, which use 2 speakers and 1 microphone.

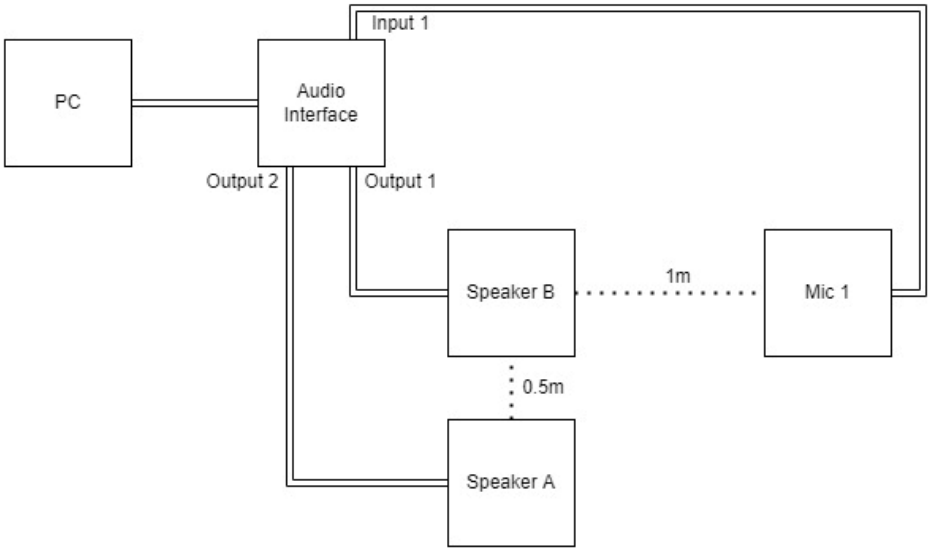

Figure 24: Set-up for testing signal interference

Figure 24 shows the simulate set-up for testing signal interference. The distance between speaker A and B is 0.5m and the distance between speaker B to microphone is 1m. By Pythagorean theorem, distance between speaker A and microphone is 1.12m. Since two signals need to be synchronized and they have different propagation distance, the signal needs to be pre-coded in order to synchronize two signals. Time shift is used in this simulation.

The calculation on time delay are as follow:

Time needed for speaker B's signal: $\Delta t = \frac{1}{343} s$

Time needed for speaker A's signal: $\Delta t = \frac{1.12}{343} s$

Since the propagation distance of speaker A is longer than speaker B, we need to delay

the signal in speaker B in other to synchronize them.

The calculation on delay elements is as follow:

$$delay\ element = 44100 * \left(\frac{1.12}{343} - \frac{1}{343}\right) = 15.42 \cong 15$$

As a result, 15 element is need in theoretical in order to make signal B arrive at the same time with signal A.

The program will first voice out separately and record the received signal for the reference, then 2 speakers will voice out together in order to perform signal interference. For simulating constructive interference, signal A and signal B are the same signal with the modulation scheme, ASK. Which the carrier signal is $\sin(2\pi f_c t)$. But signal B will have few delays element at the beginning part of the signal array. For simulating destructive interference, signal B's carrier signal become $\sin(2\pi f_c t + \pi)$ while signal A's remains the same.

4.2.2 Signal interference simulation result

There are 3 simulation result. The first simulation uses 15 delay element to synchronize two signal. The result are as follow:

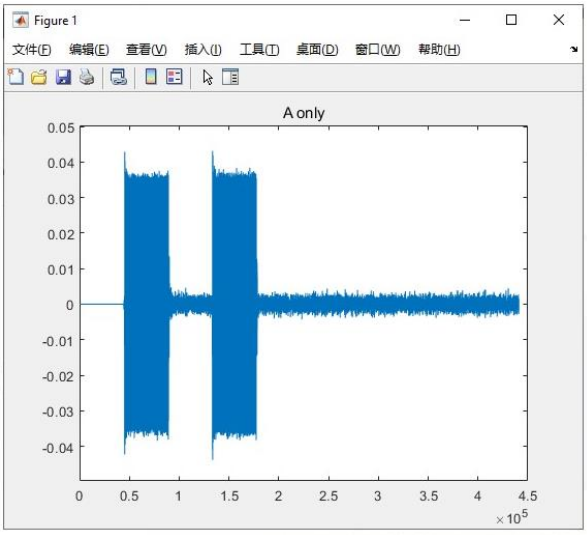

Figure 25: 1st result A

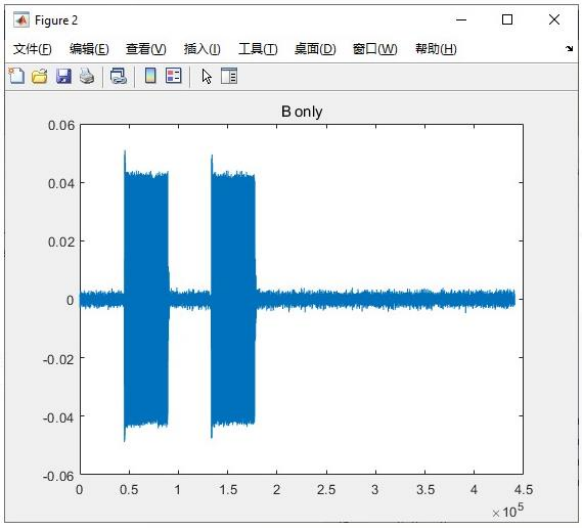

Figure 26: 1st result B

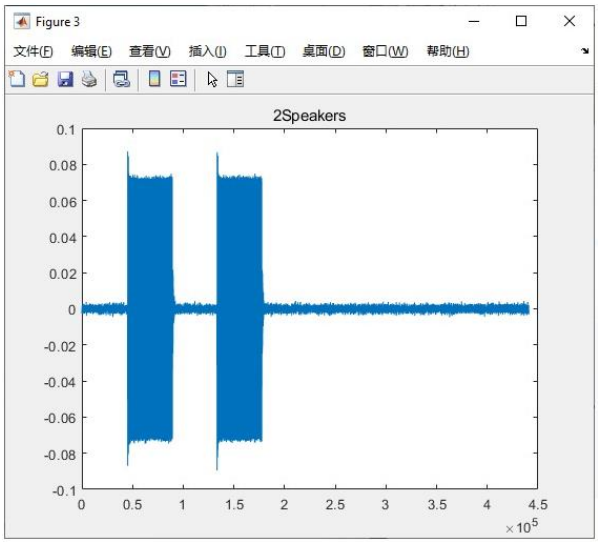

Figure 27: 1st result

Figure 25 shows the received signal form speaker A, Figure 26 shows the received signal form speaker B and Figure 27 shows the received signal with interference. From figure 25 and 26, we can see the amplitude of received signal A is around 0.037 and the amplitude of received signal B is around 0.04. The theoretical amplitude of constructive interference is 0.08. From figure 27, the amplitude of received signal is around 0.07. which means it is not in phase perfectly.

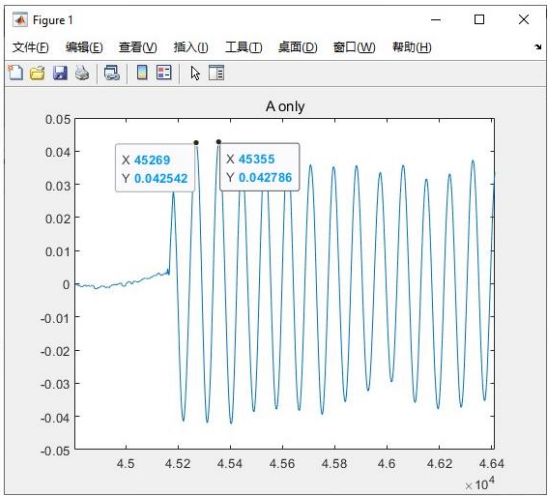

Figure 28: 1st result A (zoomed in)

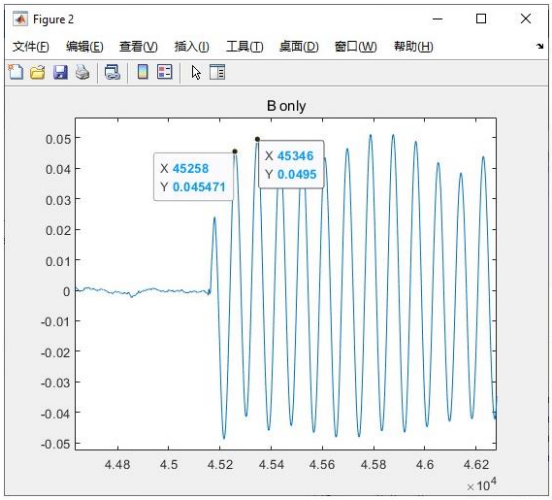

Figure 29: 1st result B (zoomed in)

Figures 28 and 29 shows the zoomed in graph on received signal A and signal B respectively. From figure 28, the second and third peak are in 45269 and 45355. From figure 29, the second and third pear are in 45258 and 45346. Thus, signal B are sill 10 elements faster than signal A. Therefore, the delay elements are added 10 more, 25 in total in the 2nd try.

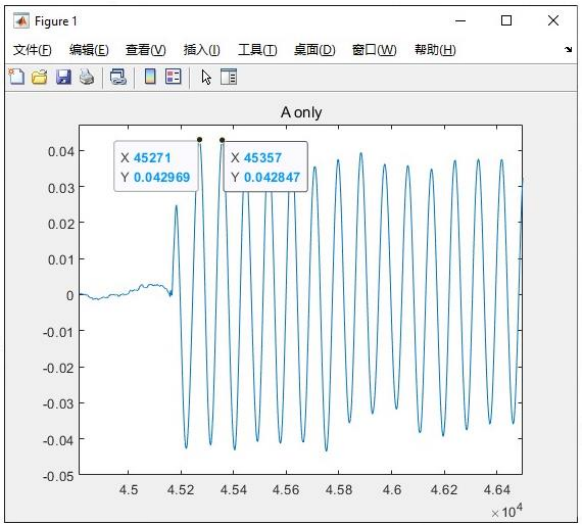

Figure 30: 2nd result A

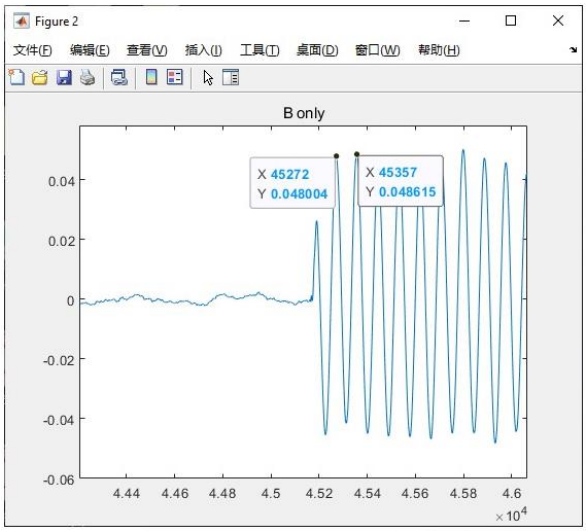

Figure 31: 2nd result B

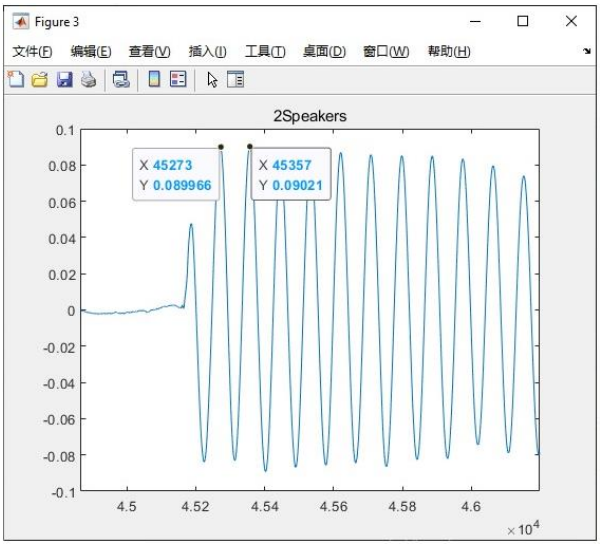

Figure 32: 2nd result

Figures 30 and 31 shows the zoomed in wave on received signal A and B respectively. Figure 32 shows the zoomed in on received signal with interference. From figure 30, the second and third peak are in 45271 and 45357 respectively as well as the amplitude is 0.42. From figure 31, the second and third peak are in 45272 and 45357 respectively as well as the amplitude is 0.48. It shows that they are more in phase than 1st simulation. From figure 32, the amplitude is 0.9. it is similar to the theoretical calculation. Therefore, in the same set up, 25 elements should be used in order to synchronize two signals. And constructive interference has been successfully simulated.

After simulated constructive interference, destructive interference has been simulated. As mentioned, signal A = $\sin(2\pi f_c t)$ and B = $\sin(2\pi f_c t + \pi)$.

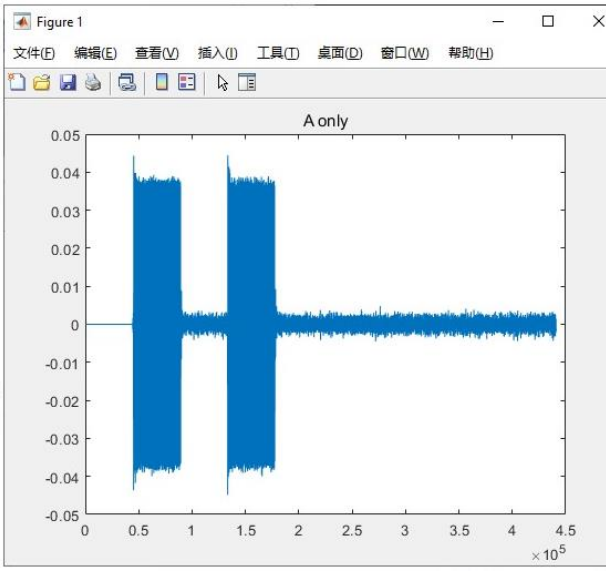

Figure 33: 3rd result A

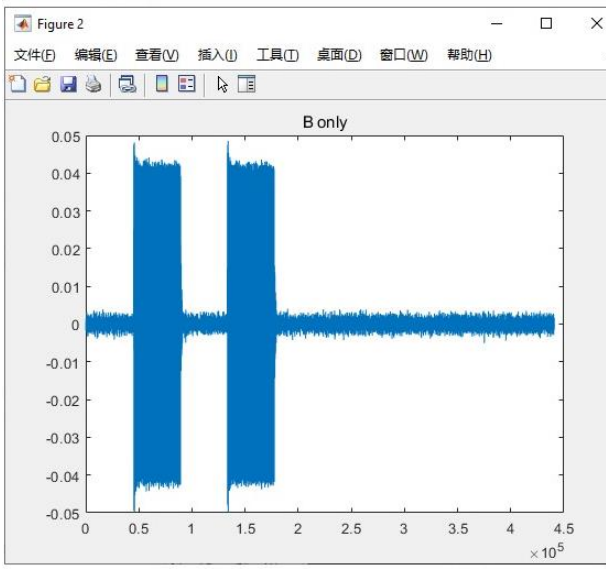

Figure 34: 3rd result B

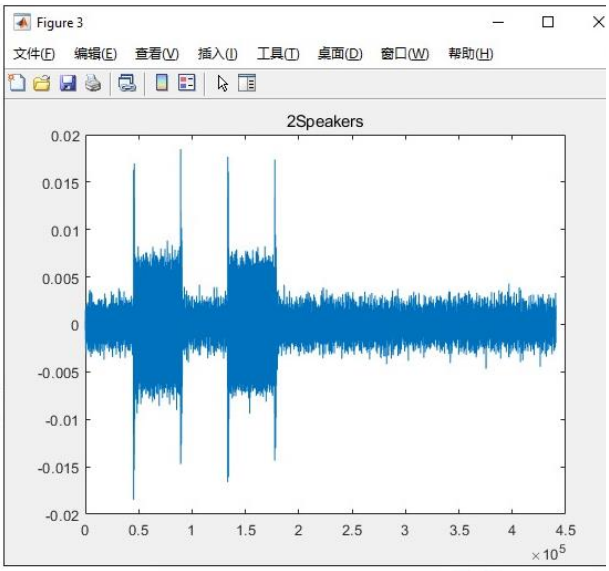

Figure 34: 3rd result

Figures 33 and 34 show the received signal of A and B respectively. Figure 34 shows the interference result. From figure 34, the signal amplitude is within 0.01. Although the theoretical should be all zero, there are noise in actual case, and it is hard to be exactly zero. The amplitude within 0.01 is small enough. Therefore, destructive interference has been successfully simulated.

4.3.1 Zero-forcing simulation

Zero-forcing are simulated fully software by a program. a 2x2 MIMO system are simulated. The signals are modulated using QAM and 1 stream carries 2 set of data. There are 4 set of message in total. They are generated by the function randi().

The program flow are as follows:

First, the program begin with the basic simulation setup as well as the H matrix. It set the amplitude to 1, carrier frequency is 1000 and number of bits is 10. Then, the program will generate the message signal randomly by randi(). And then modulate the bit to pulse signal. After that, the pulse M will multiply the inverse H matrix to become s1 and s2. x1 and x2 equals to real part of exp(j*2*pi*f_c*t) multiply by s1 and s2 respectively. For receiving part, y1 y2 equals to to attenuation times shifted x1 and x2. In demodulation part, for both y1 and y2, it will multiply by 2cos(2*pi*f_c*t) and -2sin(2*pi*f_c*t). By finding the mean of the solution in each interval, the bit is then demodulated.

4.3.2 Zero-forcing simulation result

Signal 1 is [1+1j,1+1j,1+1j,0+0j,0+0j,0+0j,0+1j,1+1j,0+0j,0+0j]

Signal 2 is [0+0j,0+1j,1+0j,1+0j,1+0j,0+0j0+1j,1+1j,1+1j,1+0j]

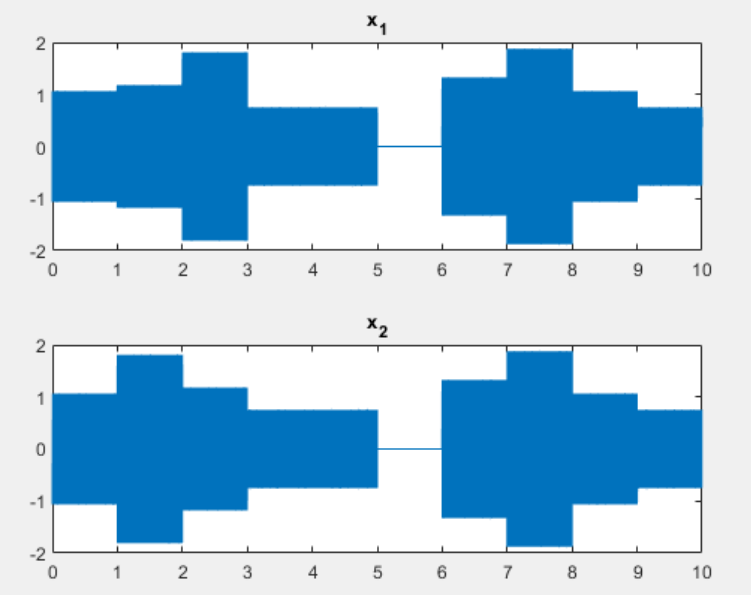

Figure 35: Zero-forcing transmitted signal

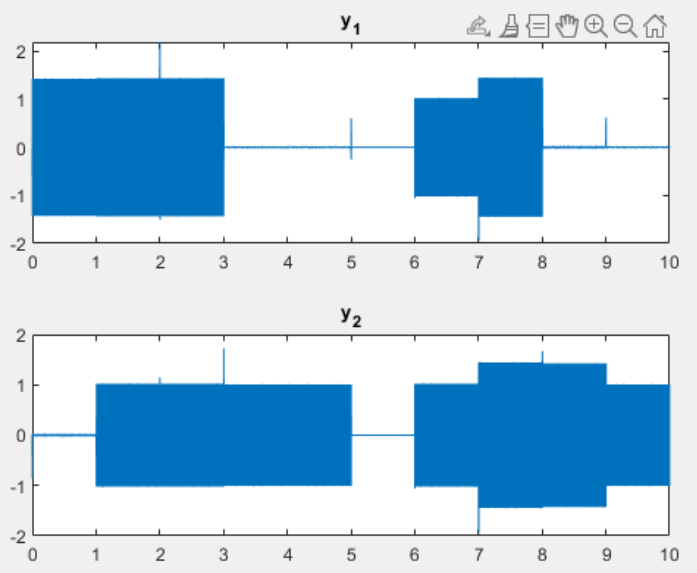

Figure 36: Zero-forcing received signal

Figure 35 shows the transmitted signal result. Figure 36 shows the zero-forcing received signal. Figure 36 is similar to the original QAM signal on m1 and m2 respectively.

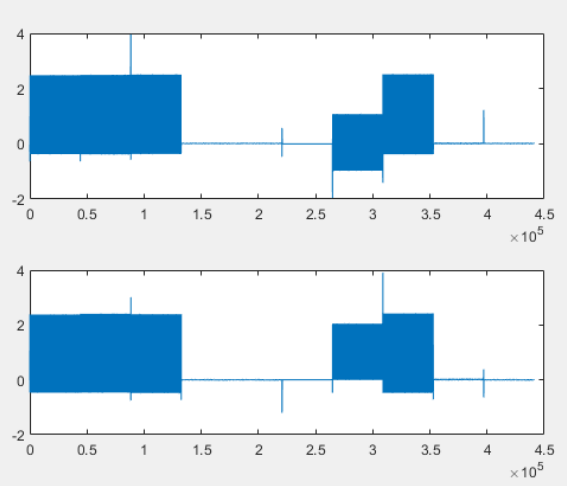

Figure 37: Demodulation on y1

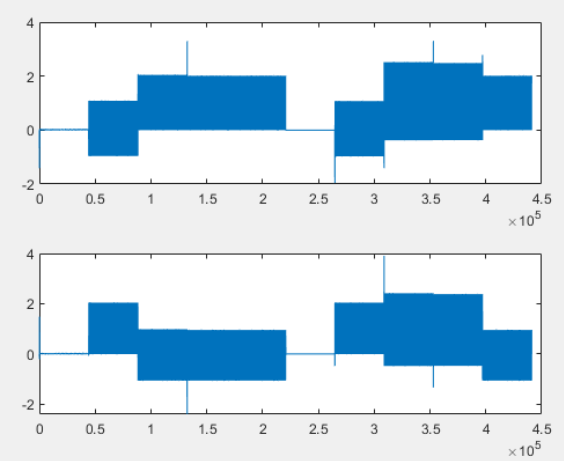

Figure 38: Demodulation on y2

Figure 37 and 38 show the demodulation on y1 and y 2 respectively, the top graph is multiplied by cosine and the below is multiplied by sine. We can see the average of each interval is equal to the 1 or 0, which is the representing bit. The top graph is represent the real part of the original message and the below is the imaginary part of the message. This is how the program determines the received bit.

5. Discussion

Regarding the ASK, PSK and FSK modulation, three modulation schemes have been simulated successfully. In an acoustic communication system, I will say ASK and FSK is better than PSK. In ASK, the modulated signal is either on or off, which we can clearly hear when the transmitted bit is 1 and when the transmitted bit is 0. We can determine the signal by ourselves as well. For FSK, although we may not able to determine the signal, we can hear different sound that responding bit 0 or 1. At least we can notice there are changes in transmitting bit 0 and 1 since it use different carrier frequency. However, we are hard to notice anything while using PSK. PSK signal are just +ve and -ve, which is nothing different when we are hearing. On the purpose on acoustic communication, we want everyone to feel more in communication using ears. Therefore, I will say PSK is the worst modulation scheme in acoustic.

Regarding the signal interference simulation, the theoretical value is different from the actual case. It may because there some other delay element. For example, although two speakers are same brand and model, their output voice are different, speaker B is always louder than speaker A. So, there may have some delay in two speakers even if they are programed to voice out at the same time. The calculation time on MATLAB programming may also cause the delay different. After trial and error, the signal is more in phase than using the theoretical value. At the simulation, I have used the time shift property to synchronize two signals. But in actual case, phase shift is more common to synchronize the signal. And channel element, which includes attenuation and phase shifted should be calculated by trial. For example, if 2 output and 2 input are used, they should output one by one to record the received signal. Then, attenuation can be determined by the amplitude of received signal divide by the transmitted signal amplitude, and the phase shift can use the same method I have done, which is use correlation to calculate the shift.

Regarding zero-forcing simulation, it is unfortunate that I cannot use the real set-up to simulate it. The MATLAB simulation seems to be correct because it just the theory.

6. Conclusion

To conclude, I have simulated the ASK PSK FSK in semester 1. A jpg photo has successfully transmitted and received using the 1x1 acoustic system. In semester 2, 2x1 acoustic system is used on testing the signal interference. The constructive and destructive interference are simulated successfully by shifting the time and changing the carrier signal. As well as zero-forcing have been simulated on MATLAB.

7. Reference

[1]: Wing Kin Ma, CUHK, ELEG2301 Handout 14: Digital Passband Transmission,

Link: https://www.ee.cuhk.edu.hk/~wkma/eleg2310/handouts/14-digital-passband.pdf

[2]: Sneha H.L, ALL ABOUT CIRCUITS, Understanding Correlation,

Link: https://www.allaboutcircuits.com/technical-articles/understanding-correlation/

[3]: Halberd Bastion Pty Ltd, 2x2 MIMO,

Link: https://halberdbastion.com/resources/wireless/mimo/2x2-mimo

8. Appendices

1. ASK-PSK-FSK program

```
close all; clear; clc;

%% Read Image "Lenna"
im = im2double(imread('Lenna.png'));
im = rgb2gray(im);
im = imresize(im,[64 64]);
figure();
imshow(im);
title('Orginal Image');

%% Setup
Q = 16;
A_c = 1;
f_c = 5000;
bitpersec = 100;

f_speaker = 44100;
t = linspace(0,1,f_speaker);

playRec = audioPlayerRecorder('Device', 'STUDIO-CAPTURE', 'PlayerChannelMapping', [1],
'RecorderChannelMapping', [1]);

error=0;
error_percent=0;

s_0 = 0;
s_1 = cos(2*pi*f_c*t);
s_1 = s_1(1:f_speaker/bitpersec)';
%% Image --> bit:
bit_im = im(:);
bit_im = bit_im./(max(abs(bit_im)));
bit_im = round(bit_im.*(Q-1));
bit_tx = de2bi(bit_im,'left-msb');
bit_tx = bit_tx';
bit_tx = bit_tx(:);
totalbit = length(bit_tx);
bit_tx = [1;1;bit_tx];
totalbit_tx = length(bit_tx);

s = zeros((ceil(totalbit_tx/bitpersec)+2)*f_speaker,1);
s_receive = zeros((ceil(totalbit_tx/bitpersec)+2)*f_speaker,1);
s_rrx = zeros((ceil(totalbit_tx/bitpersec)+2)*f_speaker,1);

%% ASK
for i = 1:1:totalbit_tx
   if bit_tx(i) == 1
      s((i-1)*f_speaker/bitpersec+1:i*f_speaker/bitpersec,1) = s_1;
   else
      s((i-1)*f_speaker/bitpersec+1:i*f_speaker/bitpersec) = s_0;
   end
end
```

```
%% Tx & Rx
for i = 1:f_speaker:length(s)
   output = s(i:i+f_speaker-1,1);
   input = playRec(output);
   s_receive(i:i+f_speaker-1,1)= input(:);
end

%% Correlation
s_receive=s_receive(f_speaker:end);
[corr,lags]=xcorr(s_1,s_receive(1:3*f_speaker/bitpersec));
delay = find(corr==max(corr));
delay = abs(lags(delay));
s_rrx = s_receive((delay+1):(ceil(totalbit_tx/bitpersec)*f_speaker+delay));

%% Normalize
s_rrx(:) = s_rrx./max(s_rrx((3/10)*f_speaker/bitpersec:(7/10)*f_speaker/bitpersec));

%% Bit Detection
for i=1:totalbit_tx
   x_1 = s_rrx((i-1)*f_speaker/bitpersec+1:(i)*f_speaker/bitpersec).*s_1;
   r_1 = sum(x_1);
   if (r_1)>=f_speaker/bitpersec/4
      bit_rx(i,1) = 1;
   else
      bit_rx(i,1)=0;
   end
end

figure();
plot(bit_tx,'o','Color','g');hold on
plot(bit_rx,'o','Color','r');hold off
xlim([0.5 totalbit+0.5]);
ylim([-0.5 1.5]);
title('Red=Received,Green=Original');
for i = 1:totalbit_tx
   if bit_tx(i) ~= bit_rx(i)
      error = error+1;
   end
end

error_percent = error/totalbit_tx;

%% bit --> Image
bit_ref = bit_tx(3:end);
bit_ref = reshape(bit_ref, log2(Q),[])';
im_ref = bi2de(bit_ref,'left-msb');
im_ref = im_ref ./((Q-1));
im_ref = reshape(im_ref,64,64);

bit_r = bit_rx(3:end);
bit_r = reshape(bit_r, log2(Q),[])';
im_rx = bi2de(bit_r,'left-msb');
im_rx = im_rx ./((Q-1));
im_rx = reshape(im_rx,64,64);

figure();
```

```
imshow(im_ref);
title('Reference Image');

figure();
imshow(im_rx);
title('Received Image');
```

2. Signal interference program

```
close all; clear; clc;

bit_tx = [1,0,1,0];
A_c = 1;
f_c = 500;

bitpersec =1;
f_speaker = 44100;
Numdelay = 25;

t = linspace(0,1,f_speaker);
playRec = audioPlayerRecorder('Device', 'STUDIO-CAPTURE', 'PlayerChannelMapping', [1,2],
'RecorderChannelMapping', [1]);
s_A = zeros(ceil(10/bitpersec)*44100,1);
s_B = zeros(ceil(10/bitpersec)*44100,1);

s_0 = 0;
s_A_1 = sin(2*pi*f_c*t);
s_A_1 = s_A_1(1:f_speaker/bitpersec)';
s_B_1 = sin(2*pi*f_c*t);
s_B_1 = s_B_1(1:f_speaker/bitpersec)';

for i = 1:1:length(bit_tx) %1(Longer)
   if bit_tx(i) == 1
      s_A((i-1)*f_speaker/bitpersec+1:i*f_speaker/bitpersec,1) = s_A_1;
   else
      s_A((i-1)*f_speaker/bitpersec+1:i*f_speaker/bitpersec,1) = s_0;
   end
end

for i = 1:1:length(bit_tx) %2
   if bit_tx(i) == 1
      s_B((i-1)*f_speaker/bitpersec+1+Numdelay:i*f_speaker/bitpersec+Numdelay,1) = s_B_1;
   else
      s_B((i-1)*f_speaker/bitpersec+1+Numdelay:i*f_speaker/bitpersec+Numdelay,1) = s_0;
   end
end

for i = 1:f_speaker:length(s_A)
   A = s_A(i:i+f_speaker-1,1);
   B = zeros(f_speaker,1);
   audioToDevice = [B,A];
   input = playRec(audioToDevice);
   s_receive_A(i:i+f_speaker-1,1)= input(:);
end

figure();
plot(s_receive_A);
title('A only');

for i = 1:f_speaker:length(s_A)
   A = zeros(f_speaker,1);
   B = s_B(i:i+f_speaker-1,1);
   %%output(:,2) = s(i:i+f_speaker-1,1);
   audioToDevice = [B,A];
```

```
    input = playRec(audioToDevice);
    s_receive_B(i:i+f_speaker-1,1)= input(:);
end

figure();
plot(s_receive_B);
title('B only');

for i = 1:f_speaker:length(s_A)
    A = s_A(i:i+f_speaker-1,1);
    B = s_B(i:i+f_speaker-1,1);
    %%output(:,2) = s(i:i+f_speaker-1,1);
    audioToDevice = [B,A];
    input = playRec(audioToDevice);
    s_receive_AB(i:i+f_speaker-1,1)= input(:);
end

figure();
plot(s_receive_AB);
title('2Speakers');
```

3. Zero-forcing program

```
close all; clear; clc;

%% Setup
A_c = 1;
f_c = 1000;
f_speaker = 44100;
N = 10;                    %number of bit
t = linspace(0,N,N*f_speaker);

%% H matrix
att = 0.8;                 %attenuation
dis = [1,1.118;1.118,1];    %distance
det_t = dis/343;           %time delay
ang_diff = 2*pi*f_c*det_t;  %phase diff
H = att*exp(-1j*ang_diff);
H_inv = inv(H);

%% Message
% m_1 = randi([0 1],1,N) + 1j*randi([0 1],1,N);
% m_2 = randi([0 1],1,N) + 1j*randi([0 1],1,N);
m_1 = [1+1j,1+1j,1+1j,0+0j,0+0j,0+0j,0+1j,1+1j,0+0j,0+0j]; % 1010 and 0101
m_2 = [0+0j,0+1j,1+0j,1+0j,1+0j,0+0,0+1j,1+1j,1+1j,1+0j]; % 1100 and 0011

m_1t=zeros(1,N*f_speaker);
m_2t=zeros(1,N*f_speaker);

for i=1:N
   if real(m_1(i))==1
      m_1t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_1t(1,(i-1)*f_speaker+1 :i*f_speaker) + ones(1,f_speaker);
   elseif real(m_1(i))==0
      m_1t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_1t(1,(i-1)*f_speaker+1 :i*f_speaker) + zeros(1,f_speaker);
   end
   if imag(m_1(i))==1
      m_1t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_1t(1,(i-1)*f_speaker+1 :i*f_speaker) + j*ones(1,f_speaker);
   elseif imag(m_1(i))==0
      m_1t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_1t(1,(i-1)*f_speaker+1 :i*f_speaker) + j*zeros(1,f_speaker);
   end
   if real(m_2(i))==1
      m_2t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_2t(1,(i-1)*f_speaker+1 :i*f_speaker) + ones(1,f_speaker);
   elseif real(m_2(i))==0
      m_2t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_2t(1,(i-1)*f_speaker+1 :i*f_speaker) + zeros(1,f_speaker);
   end
   if imag(m_2(i))==1
      m_2t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_2t(1,(i-1)*f_speaker+1 :i*f_speaker) + j*ones(1,f_speaker);
   elseif imag(m_2(i))==0
      m_2t(1,(i-1)*f_speaker+1 :i*f_speaker)=m_2t(1,(i-1)*f_speaker+1 :i*f_speaker) + j*zeros(1,f_speaker);
   end
end

M = [m_1t;m_2t];

figure();
subplot(2,1,1);
plot(t,real(m_1t));
% ylim([-0.1 1.1]);
title("Real m_1(t)");
```

```
subplot(2,1,2);
plot(t,imag(m_1t));
% ylim([-0.1 1.1]);
title("Imag m_1(t)");

figure();
subplot(2,1,1);
plot(t,real(m_2t));
% ylim([-0.1 1.1]);
title("Real m_2(t)");
subplot(2,1,2);
plot(t,imag(m_2t));
% ylim([-0.1 1.1]);
title("Imag m_2(t)");

%% Transmit signal
S = H_inv*M;
s_1 = S(1,:);
s_2 = S(2,:);

x_1=real(s_1.*exp(j*2*pi*f_c*t));
x_2=real(s_2.*exp(j*2*pi*f_c*t));

figure();
subplot(2,1,1);
plot(t,x_1);
title("x_1");
subplot(2,1,2);
plot(t,x_2);
title("x_2");

%% Receive
det_t_op = round(1/343*44100);          %opposite
det_t_dia = round(1.118/343*44100);     %diagonally

x_1_1 = [zeros(1,det_t_op),x_1(1:end-det_t_op)];
x_1_2 = [zeros(1,det_t_dia),x_1(1:end-det_t_dia)];
x_2_1 = [zeros(1,det_t_dia),x_2(1:end-det_t_dia)];
x_2_2 = [zeros(1,det_t_op),x_2(1:end-det_t_op)];

y_1 = att.*x_1_1 + att.*x_2_1;
y_2 = att.*x_1_2 + att.*x_2_2;

figure();
subplot(2,1,1);
plot(t,y_1);
title("y_1");
subplot(2,1,2);
plot(t,y_2);
title("y_2");

%% Demodulation

y_1_dm1 = y_1.*(2.*cos(2*pi*f_c*t));
y_1_dm2 = y_1.*(-2.*sin(2*pi*f_c*t));
y_2_dm1 = y_2.*(2.*cos(2*pi*f_c*t));
y_2_dm2 = y_2.*(-2.*sin(2*pi*f_c*t));
for i=1:N
    y_1_real(i) = round(mean(y_1_dm1((i-1)*f_speaker+1:i*f_speaker)));
```

```
        y_1_imag(i) = round(mean(y_1_dm2((i-1)*f_speaker+1:i*f_speaker)));
        y_2_real(i) = round(mean(y_2_dm1((i-1)*f_speaker+1:i*f_speaker)));
        y_2_imag(i) = round(mean(y_2_dm2((i-1)*f_speaker+1:i*f_speaker)));
end


%% Check
figure();
subplot(2,1,1);
plot(real(m_1),'o','Color','r');hold on
plot(y_1_real,'o','Color','g');hold off
xlim([0.5 N+0.5]);
ylim([-0.5 1.5]);
title('Real m_1 checking');
subplot(2,1,2);
plot(imag(m_1),'o','Color','r');hold on
plot(y_1_imag,'o','Color','g');hold off
xlim([0.5 N+0.5]);
ylim([-0.5 1.5]);
title('Imag m_1 checking');

figure();
subplot(2,1,1);
plot(real(m_2),'o','Color','r');hold on
plot(y_2_real,'o','Color','g');hold off
xlim([0.5 N+0.5]);
ylim([-0.5 1.5]);
title('Real m_2 checking');
subplot(2,1,2);
plot(imag(m_2),'o','Color','r');hold on
plot(y_2_imag,'o','Color','g');hold off
xlim([0.5 N+0.5]);
ylim([-0.5 1.5]);
title('Imag m_2 checking');
```

4.