**Lab 12**

**Non-Orthogonal Multiple Access (NOMA)**

Objectives:

1. How superposition coding works?
2. How successive interference cancellation (SIC) works?
3. Simulation of NOMA in AWGN channel considering 2 users
4. Simulation of NOMA in AWGN channel considering 3 users

**A) How superposition coding works?**

Non-Orthogonal Multiple Access (NOMA) is a candidate multiple access scheme for 5G and beyond. The fact that NOMA allows multiple users to transmit and receive simultaneously using the same frequency may appear intriguing. The two key operations that make NOMA possible are superposition coding which must be done at the transmitter side and successive interference cancellation (SIC) at the receiver side. In this part we will investigate superposition coding.

Let us say two users User 1 and User 2 are going to communicate simultaneously using the same frequency. Let denote User 1’s data and denote User 2’s data. For simplicity, let us assume that each user has just 4 bits of data to send. This assumption is far from reality but is sufficient to understand the basic working of NOMA.

Let = 1010 and = 0110. A graphical view of and is shown below:

A graph of a function

Description automatically generated with medium confidence

and must undergo digital modulation before transmission. Let’s use BPSK for simplicity. BPSK maps 0’s to -1’s and 1’s to +1’s. After BPSK modulation, is mapped to +1 -1 +1 -1 and is mapped to -1 +1 +1 -1 as shown below:

A graph of a function

Description automatically generated with medium confidence

NOMA requires superposition coding at the transmitter side. Superposition coding is a fancy term for power domain multiplexing. To superpose means to add. So basically, we’re going to add and together. But before doing that, we are going to multiply them with different power levels. Then we will add them together.

From the previous figure we can see that both and have a peak amplitude of . That means, they both have unit power or, 1 watt (remember that power=amplitude^2).

Let’s give power weights as =0.75 to user 1 and =0.25 to user 2. A rule to follow here is that and must sum up to 1.

First, let’s scale and with and respectively. Why square root? Because and represent the power scaling factors. To make them represent amplitude, we take the square root. After scaling, the signals look as shown below. Note that the amplitude of is scaled to and the amplitude of is scaled to . So, our amplitude scaled versions of the data are, and .

A graph of a graph of a function

Description automatically generated with medium confidence

Now it is time to add both the scaled signals together. The resulting signal is called superposition coded signal and it’s denoted by, . Adding them both together we get .

A graphical representation of x is shown below. Note that each corresponding term of and from the previous figure are added together to get the graph of x shown below:

A graph with red lines

Description automatically generated

This signal is the superposition coded NOMA signal that is actually transmitted into the channel. So, that’s how superposition coding is done. Note that is a linear combination of and . In the next section, we shall see how we can recover and from using SIC.

The above figures can be reproduced by using the following MATLAB code:

clc; clear all; close all;

x1 = [1 0 1 0];

x2 = [0 1 1 0];

xmod1 = 2\*x1-1;

xmod2 = 2\*x2-1;

a1 = 0.75; a2 = 0.25;

x = sqrt(a1)\*xmod1 + sqrt(a2)\*xmod2;

%Plot figures

ay = -2:0.2:2;

ax = ones(1,length(ay));

figure;

subplot(2,1,1)

stairs([x1,x1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Data of user 1 (x\_1)')

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([x2,x2(end)],'m','linewidth',2);

ylim([-2 2])

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Data of user 2 (x\_2)')

figure;

subplot(2,1,1)

stairs([xmod1,xmod1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Data of user 1 (x\_1)')

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([xmod2,xmod2(end)],'m','linewidth',2);

ylim([-2 2])

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Data of user 2 (x\_2)');

t1 = sqrt(a1)\*xmod1;

t2 = sqrt(a2)\*xmod2;

figure;

subplot(2,1,1)

stairs([t1,t1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Scaled data of user 1 ($$\sqrt{a\_1}x\_1$$)','Interpreter','latex','FontSize',13)

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([t2,t2(end)],'m','linewidth',2);

ylim([-2 2])

title('Scaled data of user 2 ($$\sqrt{a\_2}x\_2$$)','Interpreter','latex','FontSize',13)

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

figure;

stairs([x,x(end)],'r','linewidth',2);

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Superposition coded signal')

**B) How successive interference cancellation (SIC) works?**

In this section, we will see how SIC is carried out to decode the superposition coded signal at the receiver side.

As a quick recap, we had two users with data and respectively. We first did BPSK modulation to both and . Then we obtained the superposition coded signal as where, and are the power weights given to and respectively, such that, . We took an example of and . After BPSK modulation and choosing and , we obtained the superposition coded signal , as shown below:

A graph with red lines

Description automatically generated

Before decoding, let us take a look at SIC algorithm. SIC is an iterative algorithm where data is decoded in the order of decreasing power levels. That is, data corresponding to the user who is given the highest power is decoded first, then the data of the user who is given the next highest power is decoded. This process repeats till we have decoded all user’s data. For our simple case of two user NOMA system, the steps involved in SIC are described below:

**Step 1**: Directly decode x to get the signal that is weighted with high power. For example, if is given more weight, (i.e., ), direct decoding of gives .

**Step 2**: Multiply the signal decoded in step 1 by its corresponding weight and subtract it from x. For example, if is decoded in the previous step, subtract from . This would give us, .

**Step 3**: Decode the signal obtained in step 2 to get the other signal which was multiplexed with low power. For example, decoding of obtained from the previous step would yield us .

Now, let’s apply SIC to our example step by step. Here, we’ve made a perfect SIC assumption. Recall from the previous section that we chose values for power weights as and . This means that we have given more power weight than . So, following step 1, when we do direct decoding of , we would get .

What do we mean by direct decoding? In our case, we have used BPSK modulation at the transmitter side. So, what we mean by direct decoding is that we are going to apply BPSK demodulation directly to .

BPSK demodulation is basically, a simple thresholding. Let’s set the threshold to zero. For each symbol, if the amplitude exceeds zero, we are going to decode it as 1, and 0 otherwise.

A graph with red lines and black text

Description automatically generated

This is a plot of , and the threshold is denoted as a solid black horizontal line. Let’s decode symbol by symbol. We observe that the first and third symbols lie above the threshold of zero. So, we make the decision that the first and third transmitted bits are ones. The second and fourth symbols lie below the threshold. So, we make the decision that the second and fourth transmitted bits are zeros. Thus, the decoded sequence in order is 1010. Which is the same as .

In other words, we arrived at by directly performing BPSK demodulation on , totally ignoring the fact that had a component of too. This was possible because, we allocated a higher power weight to the component of . Thus, the presence of the other component of in can be safely ignored. In other words, we treat as interference and ignore it.

We’ve completed step 1 of SIC process. What is the intuition behind the next two steps SIC? We know . The values of and are known because they are design choices. Further, we obtained by following step 1. So, if we subtract from , we will be left with . From , it is pretty much easy to obtain .

Let’s move on to step 2. We have to multiply the component by its corresponding power weight and subtract it from . We have to be careful now. We have obtained to be 1010. But is present in its BPSK modulated form. In other words, does not contain as 1010. contains as 1 -1 1 -1, which is the BPSK modulated version of 1010. So, we have to subtract this BPSK modulated version of component from . After subtraction, the graph would look like this.

A graph of a number of symbols

Description automatically generated

Notice that this looks similar to the graph which shows scaled data of user 2 in the previous section.

We are done with step 2. Moving on to step 3. After subtraction, we simply must demodulate the resulting signal using BPSK demodulation rule as before. Following BPSK rule, we see that the first and fourth symbol would be demodulated as zeros and the other symbols would be demodulated as ones. Thus, the demodulated signal, put in order is 0110, which is same as .

This and the previous sections are written just to prove that NOMA indeed works as intended. We have successfully multiplexed two separate data in the power domain and recovered it. We have considered a perfectly ideal case by not including channel noise effects. Our goal here is just to understand how NOMA works. We will see the performance of NOMA in the presence of AWGN in the following sections.

MATLAB Code:

clc; clear all; close all;

x1 = [1 0 1 0];

x2 = [0 1 1 0];

xmod1 = 2\*x1-1;

xmod2 = 2\*x2-1;

a1 = 0.75; a2 = 0.25;

x = sqrt(a1)\*xmod1 + sqrt(a2)\*xmod2;

xdec1 = ones(1,length(x1));

xdec1(x<0)=-1;

xrem = x - sqrt(a1)\*xdec1;

xdec2 = zeros(1,length(x1));

xdec1(x<0)=0;

xdec2(xrem>0)=1;

%Plot figures

ay = -2:0.2:2;

ax = ones(1,length(ay));

figure;

subplot(2,1,1)

stairs([x1,x1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Data of user 1 (x\_1)')

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([x2,x2(end)],'m','linewidth',2);

ylim([-2 2])

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Data of user 2 (x\_2)')

figure;

subplot(2,1,1)

stairs([xmod1,xmod1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Data of user 1 (x\_1)')

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([xmod2,xmod2(end)],'m','linewidth',2);

ylim([-2 2])

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Data of user 2 (x\_2)');

t1 = sqrt(a1)\*xmod1;

t2 = sqrt(a2)\*xmod2;

figure;

subplot(2,1,1)

stairs([t1,t1(end)],'linewidth',2);

ylim([-2 2])

grid on; hold on;

title('Scaled data of user 1 ($$\sqrt{a\_1}x\_1$$)','Interpreter','latex','FontSize',13)

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

subplot(2,1,2)

stairs([t2,t2(end)],'m','linewidth',2);

ylim([-2 2])

title('Scaled data of user 2 ($$\sqrt{a\_2}x\_2$$)','Interpreter','latex','FontSize',13)

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

figure;

stairs([x,x(end)],'r','linewidth',2);

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Superposition coded signal')

plot(1:5,zeros(1,5),'k','linewidth',1.5)

figure;

stairs([xrem,xrem(end)],'r','linewidth',2);

grid on; hold on;

for u = 1:3

plot(ax\*(u+1),ay,':k','linewidth',2);

end

title('Superposition coded signal')

plot(1:5,zeros(1,5),'k','linewidth',1.5)

**C) Simulation of NOMA in AWGN channel considering 2 users**

In the previous two sections, we took a toy example to understand the concept of NOMA and to prove that NOMA indeed works as expected.

In this section, we will see how to simulate a simple two user NOMA system using MATLAB. We will be plotting the performance of NOMA in additive white gaussian noise (AWGN) channel. The AWGN assumption here is made to keep things simple so that we can pay more attention to the actual skeleton of NOMA implementation.

We are considering downlink transmission from base station (BS) to the two users. Our network will look like:

A diagram of a network model

Description automatically generated

We have a BS at the centre. It is transmitting simultaneously to two users, User 1 and User 2, using the same frequency carrier by applying NOMA. We are assuming an AWGN channel, and we are going to use QPSK modulation for both users.

Let , denote the respective distances from the BS such that, . Based on their distances, User 1 is the weakest/farthest user and User 2 is the strongest/nearest user to the BS. Let , denote the corresponding channel coefficients such that, . The channels are ordered this way because , where is the path loss exponent.

Let , denote the respective power allocation coefficients. According to the principles of NOMA, the weakest user must be allocated the most power and the strongest user must be allocated the least power. Therefore, the power allocation coefficients must be ordered as .

Implementation steps:

1. First, let’s set our simulation parameters:

Distances are set as , . The power allocation coefficients are set as , . Set path loss exponent as .

1. Generate a random binary data for User 1 and User 2 denoted as and . For random binary data generation, we can use the inbuilt MATLAB function called randi().

randi([0 1],1,N) will generate a binary sequence of length N.

1. Do QPSK modulation to the above generated data.
2. Do superposition coding. That is, compute
3. Add gaussian noise. After addition of noise, we have the received signal, for User 1 and, for User 2.
4. The decoding order should respect the following scheme:

A diagram of a process flow

Description automatically generated

1. For User 1, perform direct QPSK demodulation of to get .
2. For User 2, first perform direct QPSK demodulation of to get .
3. Remodulate into a QPSK signal to obtain
4. Multiply the estimate obtained in the previous step by and subtract it from . That is, do
5. Perform direct QPSK demodulation of to obtain
6. Compare the decoded bits with the transmitted bits to estimate BER using the inbuilt biterr() function of MATLAB.
7. Calculate the data rate of each user according to:

The MATLAB code is the following:

clc;

clear all; close all;

N = 5\*10^5; % Total number of samples to be transmitted

Pt = 30; % Transmit power (dBm)

pt = (10^-3)\*db2pow(Pt); % Transmit power (linear scale)

BW = 10^6; % Bandwidth = 1 MHz

No = -174 + 10\*log10(BW); % Noise power (dBm)

no = (10^-3)\*db2pow(No); % Noise power (linear scale)

% Note: The standard deviation is sigma and the variance is sigma^2 which is equal to noise power

d1 = 800; d2 = 300; % Distances from the Base Station

a1 = 0.75; a2 = 0.25; % Power allocation coefficients

eta = 2; %Path loss exponent in free space environment

% Generate channel gains considering the path loss

h1 = sqrt(d1^-eta);

h2 = sqrt(d2^-eta);

% Generate noise samples for the two users

n1 = sqrt(no)\*(randn(N/2,1) + 1i\*randn(N/2,1))/sqrt(2);

n2 = sqrt(no)\*(randn(N/2,1) + 1i\*randn(N/2,1))/sqrt(2);

% Generate random binary message data for the two users

x1 = randi([0 1],N,1);

x2 = randi([0 1],N,1);

%Create QPSKModulator and QPSKDemodulator objects

QPSKmod = comm.QPSKModulator('BitInput',true);

QPSKdemod = comm.QPSKDemodulator('BitOutput',true);

%Perform QPSK modulation

xmod1 = step(QPSKmod, x1);

xmod2 = step(QPSKmod, x2);

% Scaling the 2 signals by multplying with the allocated powers

xmod1 = sqrt(a1)\*xmod1;

xmod2 = sqrt(a2)\*xmod2;

% Do super-position coding

x = xmod1 + xmod2;

% Plot the 2 signals and the superimposed ones

figure

scatter(real(xmod1), imag(xmod1), 'b', 'filled');

hold on; box on

scatter(real(xmod2), imag(xmod2), 'MarkerEdgeColor', [0.4660 0.6740 0.1880], 'MarkerFaceColor', [0.4660 0.6740 0.1880]);

legend('User 1', 'User 2', 'Location','best')

title('All Transmitted signals without superposition coding')

figure

scatter(real(x), imag(x), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

legend('Superimposed Signal', 'Location','best')

title('Transmitted Superimposed Signal')

% Received signals

y1 = sqrt(pt)\*x.\*h1 + n1; % At user 1

y2 = sqrt(pt)\*x.\*h2 + n2; % At user 2

% Plot the superimposed signal received by each user

figure

scatter(real(y1), imag(y1));

hold on; box on

scatter(real(y2), imag(y2));

legend('Superimposed Signal received by User 1', 'Superimposed Signal received by User 2');

title('Received Superimposed Signals')

% Perform equalization

eq1 = y1./h1;

eq2 = y2./h2;

% Decode at user 1 (Direct decoding)

dec1 = step(QPSKdemod, eq1);

figure

scatter(real(eq1), imag(eq1), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 1')

% Decode at user 2

dec12 = step(QPSKdemod, eq2); %Direct demodulation to get U1's data

dec12\_remod = step(QPSKmod, dec12); %Remodulation of U1's data

dec12\_remod = sqrt(a1\*pt)\*dec12\_remod; % re-scale by multiplying with the allocated power

rem2 = eq2 - dec12\_remod; %SIC to remove U1's data

dec2 = step(QPSKdemod, rem2); %Direct demodulation of remaining signal

figure

scatter(real(eq1), imag(eq1), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 2')

figure

scatter(real(dec12\_remod), imag(dec12\_remod), 'b', 'linewidth', 1.2);

hold on

scatter(real(xmod1), imag(xmod1), 'bx', 'linewidth', 1);

box on

title('Remodulation of U1 signal by User 2')

legend('U1 signal recovered by User 2', 'Original transmitted Signal of User 1', 'Location', 'best')

figure

scatter(real(rem2), imag(rem2), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 2 after removing signal 1')

% BER calculation

ber1 = biterr(dec1, x1)/N;

ber2 = biterr(dec2, x2)/N;

% Data-rate calculation

g1 = (abs(h1)).^2;

g2 = (abs(h2)).^2;

R1 = log2(1 + ( (a1\*pt\*g1)/(a2\*pt\*g1+no) ) );

R2 = log2(1 + ( (a2\*pt\*g2)/(no) ) );

After running the code, you should have the following figures:



The above figure shows the signals transmitted by the two users before superposition coding.

The following figure shows the superposition (or superimposed) signal:



We can see that combining (superimposing) two QPSK signals will lead to 16 constellations.

At the receiver side, the superimposed signal received by each user is the following:



After performing equalization to supress the noise effect from the received signals, the received superimposed signals by the two users are the shown below:





Since User 1 is the far user, it will directly decode the received superimposed signal to get its own data treating the other signal as interference.

User 2 will decode first User 1’s signal and remove it from the received superimposed signal (i.e., interference cancellation). The following figure shows how User 2 successfully recovered the signal transmitted by User 1:



And after cancelling the estimated signal of User 2 from the superimposed signal, we can see in the following figure that User 2 was able to get its own signal which is identical to the transmitted one.



**Exercise**

**D) Simulation of NOMA in AWGN channel considering 3 users with QPSK Modulation**

In the previous section, we implemented a two user NOMA. Can we multiplex more than two users in a single carrier in NOMA? Of course. In this section, we are going to multiplex three users, each following QPSK modulation in a single frequency carrier.

In the system model, let use consider a wireless network consisting of three NOMA users, numbered U1, U2, and U3. Let , and denote their respective distances from the base station (BS) such that, . Based on their distances, U1 is the weakest/farthest user and U3 is the strongest/nearest user to the BS. Let , , and denote their corresponding channel coefficients such that, . The channels are ordered this way because .

Let , , and denote their respective power allocation coefficients. According to the principles of NOMA, the weakest user must be allocated the most power and the strongest user must be allocated the least power. Therefore, the power allocation coefficients must be ordered as .

Signal Model

Let , , and denote the QPSK modulated messages that the BS needs to send to U1, U2, and U3 respectively. Then, the superposition coded signal transmitted by the BS is given by:

The received signal at the user is given by:

Where denotes AWGN at receiver of Ui.

**SIC decoding procedure**

A diagram of a process flow

Description automatically generated

At User 1:

Since U1 is allocated the highest power, he will perform direct decoding from , treating the signals of U2 and U3 as interference. Thus, the achievable rate of U1 is,

From the above equation, we make one important observation. Since is present at the denominator, now we want to satisfy . Only then, U1’s power will dominate in the transmit signal, and in the received signal, .

At User 2

Next let’s write the rate equation for U2. Since and, , U2 must perform successive interference cancellation to remove U1’s data and treat U3 as interference. After removing U1’s data by SIC, the achievable rate for U2 is,

Since is present in the interference term at the denominator, we want to satisfy .

At User 3

Finally, U3 () has to perform SIC two times to remove both U1 and U2 data from . Since the term dominates in , it must be removed first. After that, the term must be removed. The achievable rate is,

Use the following simulation parameters:

Distances are set as , , . The power allocation coefficients are set as , , . These values satisfy the conditions and . Set path loss exponent as .

We should follow the same logic done in Part C.

**Try to follow the steps below and fill in the missing parts:**

1. Start with the following transmission parameters:

clc;

clear all; close all;

N = 5\*10^5; % Total number of samples to be transmitted

Pt = 30; % Transmit power (dBm)

pt = (10^-3)\*db2pow(Pt); % Transmit power (linear scale)

BW = 10^6; % Bandwidth = 1 MHz

No = -174 + 10\*log10(BW); % Noise power (dBm)

no = (10^-3)\*db2pow(No); % Noise power (linear scale)

d1 = 500; d2 = 200; d3 = 70; % Distances from the Base Station

a1 = 0.8; a2 = 0.15; a3 = 0.05; % Power allocation coefficients

eta = 2; %Path loss exponent in free space environment

1. Generate Channel gains considering the path loss

% Generate channel gains considering the path loss

h1 =

h2 =

h3 =

1. Generate noise samples for the 3 users

%Generate noise samples for the three users

n1 =

n2 =

n3 =

1. Generate random binary data for the 3 users

%Generate random binary message data for the three users

x1 =

x2 =

x3 =

1. Create QPSKModulator and QPSKDemodulator Objects

%Create QPSKModulator and QPSKDemodulator objects

QPSKmod =

QPSKdemod =

1. Perform QPSK modulation

%Perform QPSK modulation

xmod1 =

xmod2 =

xmod3 =

1. Scaling the 3 signals by multiplying with the allocated power values

% Scaling the 3 signals by multplying with the allocated powers

xmod1 =

xmod2 =

xmod3 =

1. Do superposition coding

% Do super-position coding

x =

1. Plot the 3 signals and the superimposed one

% Plot the 3 signals and the superimposed ones

figure

scatter(real(xmod1), imag(xmod1), 'b', 'filled');

hold on; box on

scatter(real(xmod2), imag(xmod2), 'MarkerEdgeColor', [0.4660 0.6740 0.1880], 'MarkerFaceColor', [0.4660 0.6740 0.1880]);

scatter(real(xmod3), imag(xmod3), 'r', 'filled');

legend('User 1', 'User 2', 'User 3', 'Location','best')

title('All Transmitted signals without superposition coding')

figure

scatter(real(x), imag(x), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

legend('Superimposed Signal', 'Location','best')

title('Transmitted Superimposed Signal')

1. Received signals by each user

% Received signals at user1, at user2 and at user3

y1 =

y2 =

y3 =

1. Plot superimposed signal received by each user

% Plot the superimposed signal received by each user

figure

scatter(real(y1), imag(y1));

hold on; box on

scatter(real(y2), imag(y2));

scatter(real(y3), imag(y3));

legend('Superimposed Signal received by User 1', 'Superimposed Signal received by User 2',...

'Superimposed Signal received by User 3');

title('Received Superimposed Signals')

1. Perform equalization

% Perform equalization

eq1 =

eq2 =

eq3 =

1. Decode at user 1(Direct decoding)

% Decode at user 1 (Direct decoding)

dec1 =

1. Plot superimposed signal received by user1

figure

scatter(real(eq1), imag(eq1), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 1')

1. Decode at user 2

% Decode at user 2

dec12 = %Direct demodulation to get U1's data

dec12\_remod = %Remodulation of U1's data

dec12\_remod = %re-scale by multiplying with the allocated power

rem2 = %SIC to remove U1's data

dec2 = %Direct demodulation of remaining signal

1. Plot the superimposed signal received by user 2, the recovered signal of user 1 and the superimposed signal after removing user 1’s signal

figure

scatter(real(eq1), imag(eq1), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 2')

figure

scatter(real(dec12\_remod), imag(dec12\_remod), 'b', 'linewidth', 1.2);

hold on

scatter(real(xmod1), imag(xmod1), 'bx', 'linewidth', 1);

box on

title('Remodulation of U1 signal by User 2')

legend('U1 signal recovered by User 2', 'Original transmitted Signal of User 1', 'Location', 'best')

figure

scatter(real(rem2), imag(rem2), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 2 after removing signal 1')

1. Decode at user 3

% Decode at user 3

dec13 = %Direct demodulation to get U1's data

dec13\_remod = %Remodulation of U1's data

dec13\_remod = %re-scale by multiplying with the allocated power

rem31 = %SIC to remove U1's data

dec23 = %Direct demodulation of remaining signal to get U2's data

dec23\_remod = %Remodulation of U2's data

dec23\_remod = %re-scale by multiplying with the allocated power

rem3 = %SIC to remove U2's data

dec3 = %Demodulate remaining signal to get U3's data

1. Plot the superimposed signal before and after removing user 1’s signal and user 2’s signal. Also plot the recovered signal of user 1 and user 2

figure

scatter(real(eq3), imag(eq3), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 3')

figure

scatter(real(dec13\_remod), imag(dec13\_remod), 'b', 'linewidth', 1.2);

hold on

scatter(real(xmod1), imag(xmod1), 'bx', 'linewidth', 1);

box on

title('Remodulation of U1 signal by User 3')

legend('U1 signal recovered by User 3', 'Original transmitted Signal of User 1', 'Location','best')

figure

scatter(real(rem31), imag(rem31), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 3 after removing signal 1')

figure

scatter(real(dec23\_remod), imag(dec23\_remod), 'MarkerEdgeColor', [0.4660 0.6740 0.1880], 'linewidth', 1.2);

hold on

scatter(real(xmod2), imag(xmod2), 'x', 'MarkerEdgeColor', [0.4660 0.6740 0.1880], 'MarkerFaceColor', [0.4660 0.6740 0.1880], 'linewidth', 1);

box on

title('Remodulation of U2 signal by User 3')

legend('U2 signal recovered by User 3', 'Original transmitted Signal of User 2', 'Location', 'best')

figure

scatter(real(rem3), imag(rem3), 'r', 'linewidth', 1.2);

hold on

scatter(real(xmod3), imag(xmod3), 'rx', 'linewidth', 1);

box on

title('Superimposed Signal received by User 3 after removing signal 2')

legend('U3 signal recovered by User 3 after removing signal 1 and signal 2', 'Original transmitted Signal of User 3', 'Location', 'best')

figure

scatter(real(rem3), imag(rem3), 'MarkerEdgeColor', [0.8500 0.3250 0.0980], 'MarkerFaceColor', [0.8500 0.3250 0.0980]);

box on

title('Superimposed Signal received by User 3 after removing signal 1 and signal 2')

1. BER calculation

% BER calculation

ber1 =

ber2 =

ber3 =

1. Data-rate calculation

% Data-rate calculation

R1 =

R2 =

R3 =

**For the report upload only the script of Exercise D.**