Aristotle University of Thessaloniki
Department of Electrical and Computer Engineering
Telecommunications Division

ΟΜΑΔΑ 11
Γιώργος Αβατάγγελος: 10536
Βαγγέλης Κάστας: 10358

Communication Systems II

Constellation based multiple access

1 Task1: Literature Review

In this section, we present a literature review of the task that we were assigned. The main substance of the research is to compare the performance analysis of different constellations based on multiple access. However, the performance of a communication system can be described with many ways, we used the Symbol Error Rate(SER) to compare the two systems. The findings and the results of this report is a product of a semestrial research.

The System 1 focuses on the average energy of the constellation and the distance between the symbols. We're given two M-PAM constellations with average energy a and (1-a) respectively. Considering analytical expressions, we simulated the System in MATLAB in order to find the value of a that accomplishes user fairness.

On the other hand, the System 2 is based on the rotation of the constellation. More specifically, the variable in this system is the angle 'theta' that rotates the constellation and the decision boundaries. Computing the optimal value of theta that provides user fairness involves mathematical methods and trigonometric equations that were used in the simulation.

Overall, the goal of these simulations to compare the results with other research articles and scientific reports, understanding the significance of the task and using algorithms and various methods to complete it, has been achieved.

2 Task 2: For System 1, by considering M = 4 and a = 0.25, develop a simulation in MATLAB to plot the SEP vs. SNR for both UE1 and UE2, where SNR = Et/No. Verify the simulation results through the SEP expressions for M-PAM provided in [2].

-Initialize the variables:

```
clc, clear, close all

M = 4;
N = 10000; %Number of symbols
a = 0.25;  %Average Energy of S1
Eg1 = sqrt(a/5);
Eg2 = sqrt((1-a)/5);
snr_dB = -10:0.5:20;
snr = 10.^(snr_dB/10);
num1_Err = zeros(1, length(snr));
num2_Err = zeros(1, length(snr));
```

-Generate the symbols:

```
pam1 = [-3*Eg1, -1*Eg1, 1*Eg1, 3*Eg1]; %4-PAM1
pam2 = [-3*Eg2, -1*Eg2, 1*Eg2, 3*Eg2]; %4-PAM2
s = zeros(4, 4);
for i = 1:length(pam1)
    for j = 1:length(pam2)
        s(i, j) = pam1(i) + 1i*pam2(j);
    end
end
qam = reshape(s, 1, 16); %16-QAM
symbols_sent = randsrc(1,N,qam);
Et = norm(var(symbols_sent) - (mean(symbols_sent))^2);
s_real = real(symbols_sent);
s_imag = imag(symbols_sent);
```

-Add Complex Gaussian Noise and compute the SEP

```
for i = 1:length(snr)

    std1 = sqrt(Et/(2*snr(i)));      %Standard Deviation 1
    std2 = sqrt(0.5*Et/(2*snr(i)));  %Standard Deviation 2
    cgnoise1 = std1*(randn(1,N) + 1i*randn(1,N));
    cgnoise2 = std2*(randn(1,N) + 1i*randn(1,N));
    r1 = symbols_sent + cgnoise1;
    r2 = symbols_sent + cgnoise2;
    y1 = real(r1);
    y2 = imag(r2);

    %Perioxes Apofasis
    reg1 = zeros(1,N);
    reg1(find(y1 < (-2*Eg1))) = -3*Eg1;
    reg1(find(y1 >= (2*Eg1))) = 3*Eg1;
    reg1(find(y1 >= (-2*Eg1) & y1 < 0)) = -1*Eg1;
    reg1(find(y1 >= 0 & y1 < (2*Eg1))) = 1*Eg1;

    reg2 = zeros(1,N);
    reg2(find(y2 < (-2*Eg2))) = -3*Eg2;
    reg2(find(y2 >= (2*Eg2))) = 3*Eg2;
    reg2(find(y2 >= (-2*Eg2) & y2 < 0)) = -1*Eg2;
    reg2(find(y2 >= 0 & y2 < (2*Eg2))) = 1*Eg2;

    num1_Err(i) = nnz(s_real - reg1);  %number of errors for user 1
    num2_Err(i) = nnz(s_imag - reg2);  %number of errors for user 1

end

SEP1 = num1_Err/N;
SEP2 = num2_Err/N;
```

-Plot the SEP vs SNR for the 2 users

```
%PLOT
figure
SEP1_theory = (M-1)/M*erfc(sqrt(3/(M^2-1)/2*snr));
SEP2_theory = (M-1)/M*erfc(sqrt(3/(M^2-1)/2*snr*(1-a)/a));
semilogy(snr_dB, SEP1_theory, 'r', snr_dB, SEP2_theory, 'b')
hold on
grid on
semilogy(snr_dB, SEP1, 'y', snr_dB, SEP2, 'g');
xlabel('SNR(db)');
ylabel('SEP');
legend('SEP1theory', 'SEP2theory', 'SEP1', 'SEP2');
title('SEP vs SNR(dB)');
```

The simulation is below and you can see that the Symbol Error Probability for our system almost overlapping the theoretical graph.

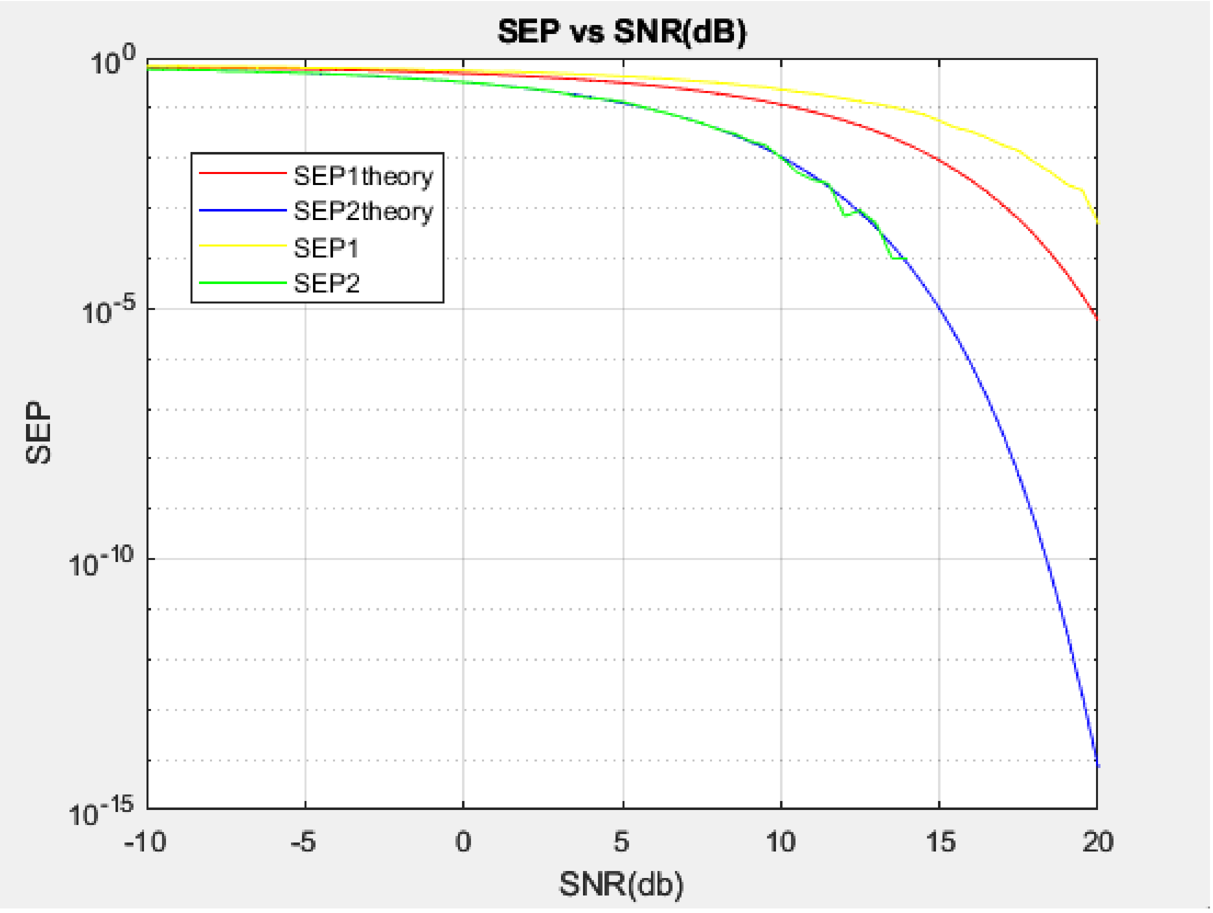

Figure 1: SEP

3 Task 3: Also, find at every SNR the value of α such that the maximum SEP between UE1 and UE2 is minimized.

*The solution below, that we suggest, is with arrays. However, we know that it's not the optimal because of the delay and the inaccuracy of the code(because of the delay we had to decrease the number of values of a so that the code run at a normal speed), but for some technical reasons we couldn't use the optimization toolbox to solve the task with optimization.

-Define the parameters:

```
clc, clear, close all

M = 4; % number of symbols in each constellation
N = 1000; % number of symbols to transmit
snr_dB = -10:0.5:20;
snr = 10.^(snr_dB/10);
a_values = linspace(0.001, 0.999, 100); % range of 'a' values to search
max_SEP = zeros(length(snr_dB), length(a_values));
a_opt = zeros(1, length(snr_dB));
```

-Find the optimal value of a:

```
for i = 1:length(snr_dB)
    for j = 1:length(a_values)
        a = a_values(j);

```

```
        Es1 = a;
        Es2 = (1-a);
        Et = Es1 + Es2;
        Eg1 = sqrt(a/5);
        Eg2 = sqrt((1-a)/5);

        s1 = randi([0 M-1], 1, N); % symbols for UE1
        s2 = randi([0 M-1], 1, N); % symbols for UE2

        x = Eg1*pammod(s1, M) + Eg2*pammod(s2, M).*1j;

        % Calculate noise standard deviation
        sigma_1 = sqrt(Et/(2*snr(i)));
        sigma_2 = sqrt(0.5*Et/(2*snr(i)));

        % Generate complex Gaussian noise
        noise1 = sigma_1*(randn(1,N) + 1i*randn(1,N));
        noise2 = sigma_2*(randn(1,N) + 1i*randn(1,N));

        % Add noise to the signals and demodulate
        y1 = x + noise1; % received signal at UE1
        y2 = x + noise2; % received signal at UE2
        r1 = real(y1); % demodulated symbols at UE1
        r2 = imag(y2); % demodulated symbols at UE2

        % Maximum Likelihood Detection
        reg1 = zeros(1,N);
        reg1(find(r1 < (-2*Eg1))) = -3*Eg1;
        reg1(find(r1 >= (2*Eg1))) = 3*Eg1;
        reg1(find(r1 >= (-2*Eg1) & r1 < 0)) = -1*Eg1;
        reg1(find(r1 >= 0 & r1 < (2*Eg1))) = 1*Eg1;
        reg2 = zeros(1,N);
        reg2(find(r2 < (-2*Eg2))) = -3*Eg2;
        reg2(find(r2 >= (2*Eg2))) = 3*Eg2;
        reg2(find(r2 >= (-2*Eg2) & r2 < 0)) = -1*Eg2;
        reg2(find(r2 >= 0 & r2 < (2*Eg2))) = 1*Eg2;

        % Calculate the Symbol Error Probabilities (SEP) for UE1 and UE2
        SEP1 = nnz(reg1 - real(x))/N;
        SEP2 = nnz(reg2 - imag(x))/N;

        % Store the maximum SEP between UE1 and UE2
        max_SEP(i,j) = max(SEP1, SEP2);

    end
    l = find(max_SEP(i,:) == min(max_SEP(i,:)), 1, 'first');
    a_opt(i) = a_values(l);
end
```

-Plot 'optimal a' vs SNR:

```
% Plot the optimal 'a' value vs. SNR
figure;
plot(snr_dB, a_opt);
xlabel('SNR (dB)');
ylabel('Optimal a');
title('Optimal a vs. SNR');
```

We notice that the optimal value of a is not accurate as the values of a are limited.

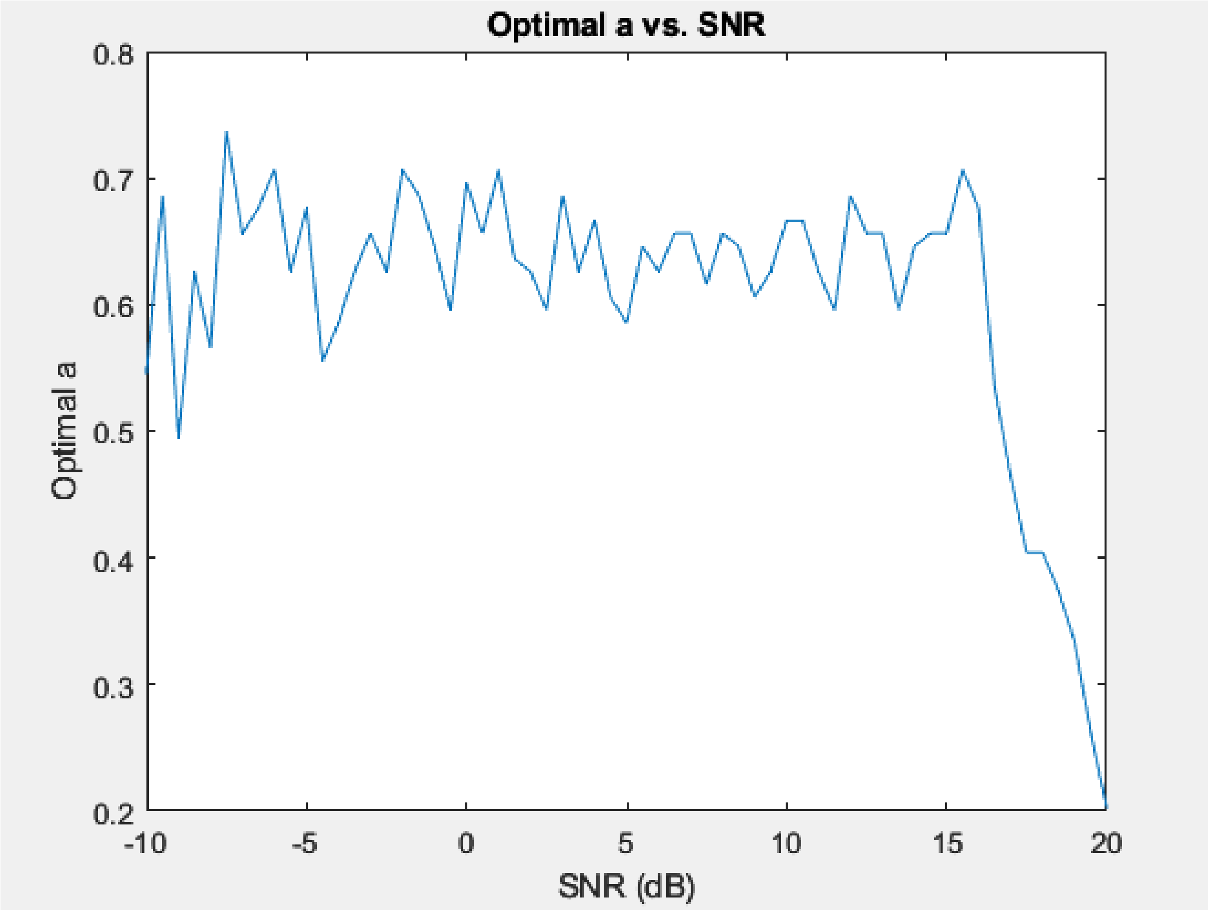

Figure 2: Optimal a

4 Task 4: Consider a 4-QAM constellation for System 2. Also, find at every SNR the value of ϑ such that the that the maximum SEP between UE1 and UE2 is minimized (i.e. the value of ϑ that achieves user fairness) and plot SEP of UE1 and UE2 vs. SNR, where SNR = Et/No. Compare the performance of System 1 and System 2.

*The solution below, that we suggest, is with arrays. However, we know that it's not the optimal because of the delay and the inaccuracy of the code(because of the delay we had to decrease the number of values of theta so that the code run at a normal speed), but for some technical reasons we couldn't use the optimization toolbox to solve the task with optimization.

-Initialize the parameters

```
clc, clear, close all


N = 1000; %number of symbols sent
M = 2; %number of symbols in each constellation
a = 0.25; % average constellation energy of 2-PAM
theta = 0.01*pi:pi/100:pi; %angle of the 4-QAM constellation
```

```

Eg1 = sqrt(3*a);
Eg2 = sqrt(3*(1-a));
Et = 1; %average constellation energy for the 4-QAM

% SNR
SNR_dB = -10:0.5:20;
SNR = 10.^(SNR_dB/10);


max_SEP = zeros(length(SNR), length(theta));
qam = zeros(M, M);
theta_opt = zeros(1, length(SNR));
```

-Find the optimal value of theta for every SNR:

```
for i = 1:length(SNR)
    SEP1 = zeros(1, length(theta)); %matrix for SEP for UE1
    SEP2 = zeros(1, length(theta)); %matrix for SEP for UE2
    for j = 1:length(theta)

                                  % Generate the 4-QAM constellation
        pam1 = pammod([0, M-1], M, theta(j));
        pam2 = pammod([0, M-1], M, theta(j));
        PAM1 = Eg1 * pam1;
        PAM2 = Eg2 * pam2;
        for k = 1:M
            for l = 1:M
                qam(k,l) = PAM1(k) + 1i*PAM2(l);
            end
        end
        QAM = reshape(qam, 1, 4);
        real_q = real(QAM);
        imag_q = imag(QAM);

        % The recieved signal before the noise
        x = randsrc(1, N, PAM1) + 1i * randsrc(1, N, PAM2);

        % Calculate noise standard deviation
        sigma_1 = sqrt(Et/(2*SNR(i)));
        sigma_2 = sqrt(0.5*Et/(2*SNR(k)));

        % Generate complex Gaussian noise
        noise_1 = sigma_1*(randn(1,N) + 1i*randn(1,N));
        noise_2 = sigma_2*(randn(1,N) + 1i*randn(1,N));

        % Add noise to the signals and demodulate
        r1 = x + noise_1; %recieved signal UE1
        r2 = x + noise_2; %recieved signal at UE2
        y1 = real(r1); %demodulated symbols at UE1
        y2 = imag(r2); %demodulated symbols at UE2

        % Maximum Likelihood Detection
        d1 = norm(abs(real_q(4) - real_q(2))); %Euclidean Distance
        d2 = norm(abs(real_q(1) - real_q(2))); %Euclidean Distance
        reg1 = zeros(1, length(theta));
        if (real_q(4)>0 & real_q(2)>0)
            reg1(find(y1>real_q(4)+d1/2)) = real_q(2);
            reg1(find(y1>0  &  y1<real_q(4)+d1/2)) = real_q(4);
            reg1(find(y1<-(real_q(4)+d1/2))) = real_q(3);
            reg1(find(y1<0  &  y1>-(real_q(4)+d1/2))) = real_q(1);
        elseif real_q(4)<0 & real_q(2)>0
            reg1(find(y1 > min(real_q(1),real_q(2)) + d2/2)) =
            max(real_q(1),real_q(2));
```

```

            reg1(find(y1 < -(min(real_q(1),real_q(2)) + d2/2))) =
            -max(real_q(1),real_q(2));

            reg1(find(y1>0  &  y1<min(real_q(1),real_q(2)) + d2/2)) =
            min(real_q(1),real_q(2));

            reg1(find(y1<0  &  y1>-(min(real_q(1),real_q(2)) + d2/2)))
            = -min(real_q(1),real_q(2));
        else
            reg1(find(y1>min(real_q(1),real_q(3))+d1/2)) =
            max(real_q(1),real_q(3));

            reg1(find(y1>0  &  y1<min(real_q(1),real_q(3))+d1/2)) =
            min(real_q(1),real_q(3));

            reg1(find(y1<-(min(real_q(1),real_q(3))+d1/2))) =
            -max(real_q(1),real_q(3));

            reg1(find(y1<0  &  y1>-(min(real_q(1),real_q(3))+d1/2))) =
            -min(real_q(1),real_q(3));
        end
        d3 = norm(abs(imag_q(3) - imag_q(4))); %Euclidean Distance
        d4 = norm(abs(imag_q(1) - imag_q(3))); %Euclidean Distance
        reg2 = zeros(1, length(theta));
        if imag_q(3)>0
            reg2(find(y2>imag_q(3)+d3/2)) = imag_q(4);
            reg2(find(y2>0  &  y2<imag_q(3)+d3/2)) = imag_q(3);
            reg2(find(y2<-(real_q(3)+d3/2))) = imag_q(1);
            reg2(find(y2<0  &  y2>-(imag_q(3)+d3/2))) = imag_q(2);
        elseif imag_q(3)<0 & imag_q(4)>0
            reg2(find(y2 > min(imag_q(2),imag_q(4)) + d4/2)) =
            max(imag_q(2),imag_q(4));

            reg2(find(y2 < -(min(imag_q(2),imag_q(4)) + d4/2))) =
            -max(imag_q(2),imag_q(4));

            reg2(find(y2>0  &  y2<min(imag_q(2),imag_q(4)) + d4/2)) =
            min(imag_q(2),imag_q(4));

            reg2(find(y2<0  &  y2>-(min(imag_q(2),imag_q(4)) + d4/2)))
            = -min(imag_q(2),imag_q(4));
        else
            reg2(find(y2>min(imag_q(2),imag_q(1))+d3/2)) =
            max(imag_q(1),imag_q(2));

            reg2(find(y2>0  &  y2<min(imag_q(2),imag_q(1))+d3/2)) =
            min(imag_q(1),imag_q(2));

            reg2(find(y2<-(min(imag_q(1),imag_q(2))+d3/2))) =
            -max(imag_q(1),imag_q(2));

            reg2(find(y2<0  &  y2>-(min(imag_q(1),imag_q(2))+d3/2))) =
            -min(imag_q(1),imag_q(2));
        end

        % Calculate the Symbol Error Probabilities for UE1 and UE2
        SEP1(j) = nnz(reg1 - real(x))/N;
        SEP2(j) = nnz(reg2 - imag(x))/N;

        % Store the maximum SEP between UE1 and UE2
        max_SEP(i,j) = max(SEP1(j), SEP2(j));
    end
```

```
    [~,l] = find(max_SEP(i,:) == min(max_SEP(i,:)), 1, 'first');
    theta_opt(i) = theta(l);
end
```

-Plot 'optimal theta(rad)' vs SNR:

```
% Plot the optimal 'theta' value vs. SNR
figure;
plot(SNR_dB, theta_opt, '-o');
xlabel('SNR (dB)');
ylabel('Optimal theta');
title('Optimal theta vs. SNR');
```

Below is the plot between the optimal value of theta and SNR. It's worth mentioning that as the SNR increasing the value of theta tends towards pi.

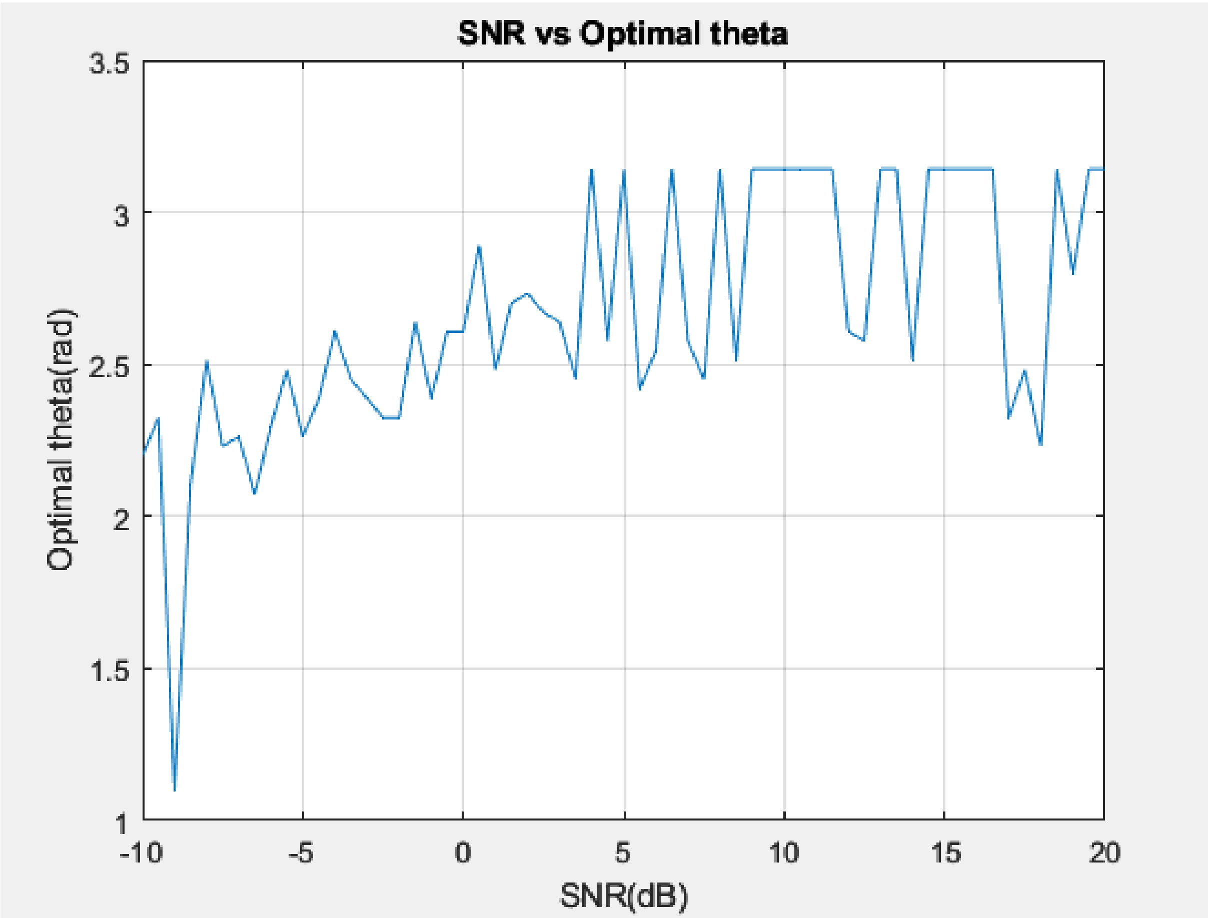

Figure 3: Optimal a

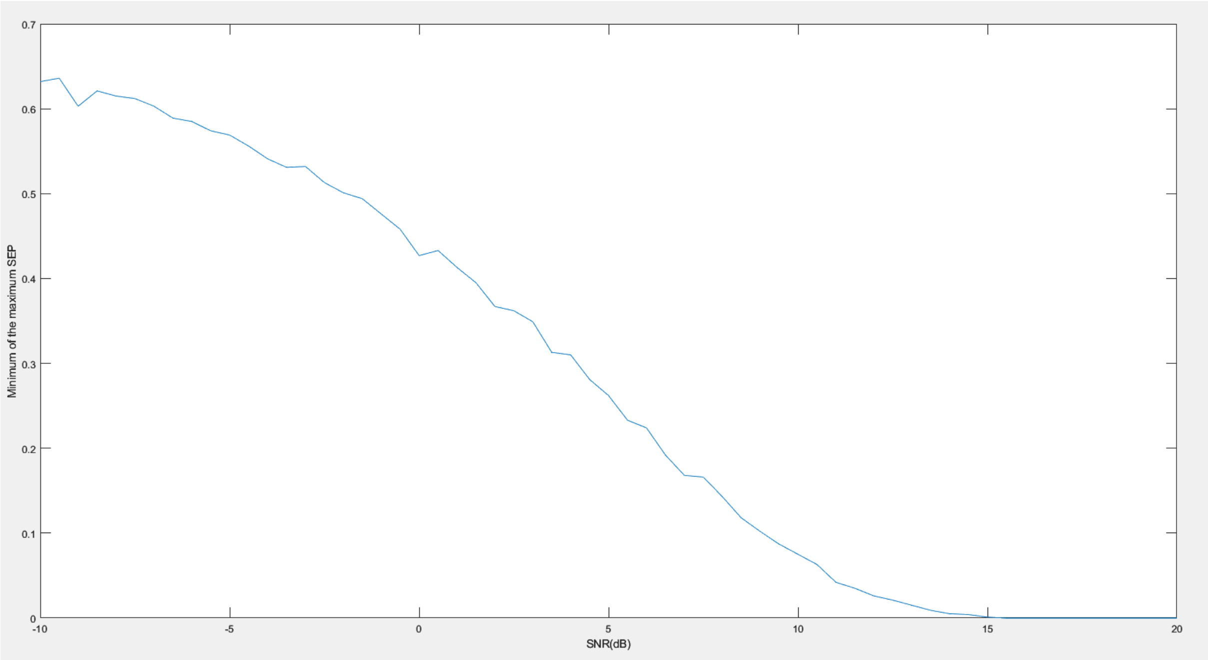

Figure 4: System 1

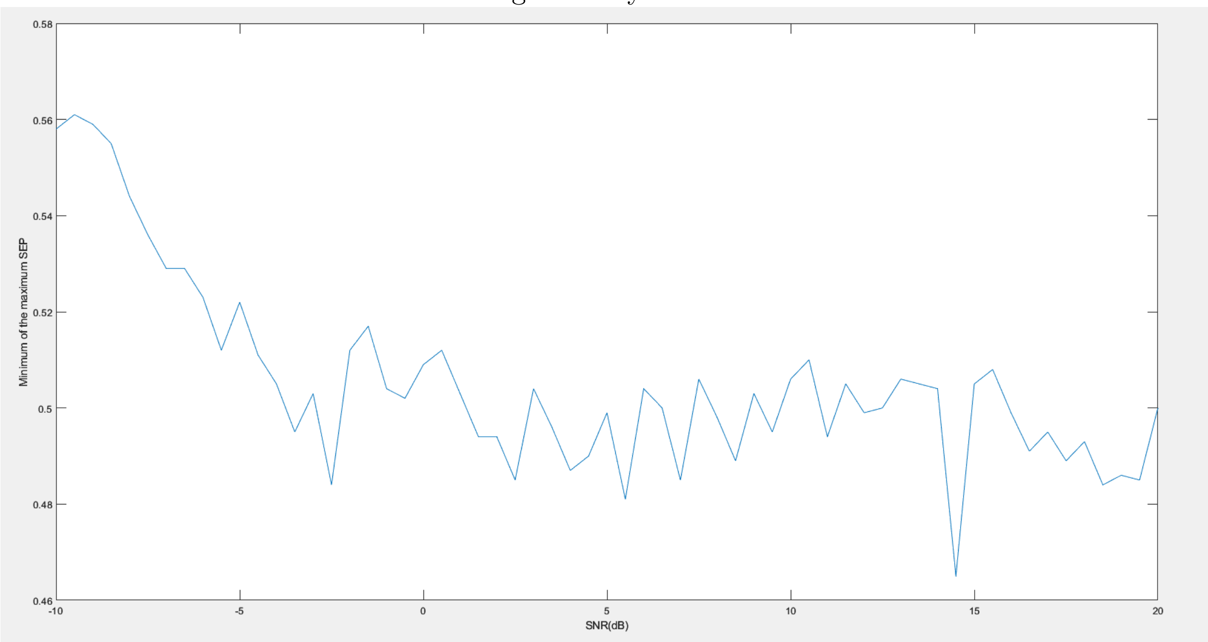

Figure 5: System 2

Printing and comparing the minimum of the maximum SEP between the 2 users for every SNR for each system, we come to a conclusion that the System 2 is more efficient for channels with high noise(i.e. SNR<=5) and that the System 1 transmit better when the SNR is big.

References

1)G. K. Karagiannidis and K. N. Pappi, Telecommunication Systems, 4th ed., Tziolas Publications, March 2017 (in Greek