Assignment 3, Digital communication 1TE747,VT24

Anton Blaho Mildton
Department of Electrial engineering
Uppsala University
`Anton.blahomildton@gmail.com`

Gustaf Löfdahl
Department of Electrial engineering
Uppsala University
`Gustaf.lofdahl@hotmail.com`

UPPSALA
UNIVERSITET

1 Introduction

In digital communication systems is the efficient and accurate transmission of data crucial and relies heavily on sophisticated encoding and decoding techniques. One aspect of this process involves the mapping of discrete bits to constellation symbols, which serves as the foundation for various modulation schemes employed in modern communication systems. Also, the ability to detect transmitted symbols amidst noise is paramount for ensuring reliable communication channels.

The objective of this assignment is to explore the mapping process from bits to constellation symbols. Furthermore, the techniques of Maximum A Posteriori (MAP) detection, a powerful method employed to decipher transmitted symbols even in the presence of noise.

2 Assignment 3 answers

The model created in this assignment serves as a simplified baseband representation of a passband transmission system. It encompasses three main components:

$$s[k] = s_1 + js_2 \tag{1}$$

One, the generation of symbols within a predefined constellation, each symbol represented by a complex number.

$$r[k] = s[k] + n[k] \tag{2}$$

Two, the assumption of a channel with unit gain, introducing complex-valued white Gaussian noise to the transmitted symbols.

$$r[k] = s[k] + n[k] = r_1[k] + jr_2[k] \tag{3}$$

Three, the representation of the received symbol as a complex number composed of the transmitted symbol and the added noise.

2.1 Subtask 1

The first task was to write a function creating the AWGN-channel with the complex noise. The goal is to introduce the noise with specified variance to an vector of complex symbols.

$$noisySignal = channel2(transmittedSignal, noisePower) \tag{4}$$

The input of the function is first $transmittedSignal$, a complex vector of symbols to be sent. The second is $noisePower$, it is divided into the noise variance and its square root is applied to both the real and imaginary part of the complex white Gaussian noise.

The function was validated by sending a vector where all symbols are zeros, thereby ensuring the result is only the added complex Gaussian white noise. Since the noise is scaled by the given $noisePower$ will the power of the resulting vector equal to the specified power when working correctly.

```matlab
function TestChannel2()
    % Example zero transmitted signal
      transmittedSignal = zeros(1,1000);
    % Example noise power
      noisePower = 1;
    % Apply noise to transmitted signal
      noisySignal = channel2(transmittedSignal, noisePower);
    % Calculate the power of the noisy signal
      noisyPower = mean(abs(noisySignal).^2);
      disp(['AWGN Noisy signal power: ', num2str(noisyPower)]);
end
```

Listing 1: MATLAB code

2.2 Subtask 2

The second task was to use a vector with 1000 of the symbol for $[01]$ in QPSK, with average power equal to one and in the complex plane. The vector is then sent through the $Channel2$, created in the previous task with the noise variances $[0.01, 0.1, 1]$.

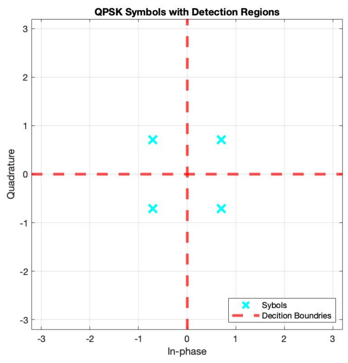

Figure 1:

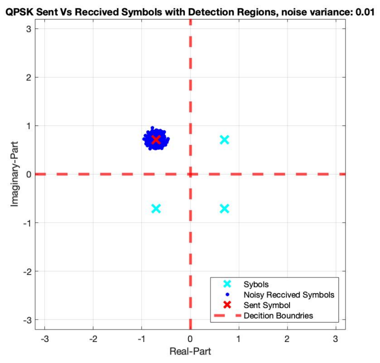

Figure 2:

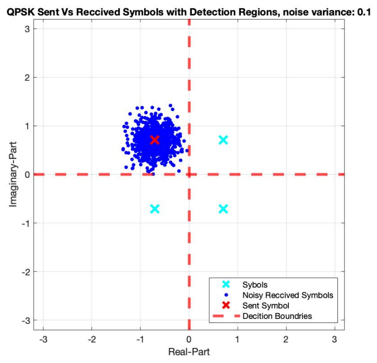

Figure 3:

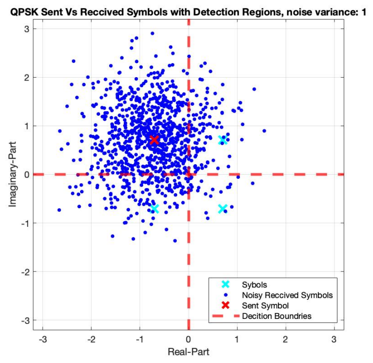

Figure 4:

Figure 5: QPSK Symbols, Detection Regions and Received Symbols

The noise added to each symbol is Gaussian. When a large number of symbols are transmitted through this channel, the noise contributions from each symbol add up, resulting in a Gaussian distribution of the received symbols around the transmitted symbol's constellation point. The statistical distribution of the received symbols will be of a complex Gaussian distribution, with the mean centered around the transmitted symbol and variance determined by the noise power. As can be observed in the figures of received symbols. In following table the empirical symbol error rate is presented.

Table 1: Empirical Symbol Error Rates (SER)

Variance/Symbols	1000	100000	10000000
0.01	0	0	0
0.1	0.002	0.0014	0.0015
1	0.281	0.2906	0.2923

As noise variance increases, Symbol Error Rate (SER) typically rises due to increased symbol distortion. Additionally, with more samples, the accuracy of the signal-to-noise ratio improves, aiding in better symbol detection statistics dependent on noise variance.

2.3 Subtask 3

The third task consist of constructing a function mapping a bit vector to the symbols of 16QAM, thus resulting in a symbol vector with the length one fourth of the original bit vector.

$$s = \text{maponto16QAM}(c) \tag{5}$$

The 16QAM symbols consist of 16 equally spaced points in a grid pattern on the complex plane, where the real axis relates to the amplitude and the imaginary axis relates to phase shift of the generated signal. The constellation is scaled so that the average symbol energy is equal to one. The symbols are placed in a Gray coding pattern ensuring an error to the closest vertical or horizontal symbol always result in one single bit error.

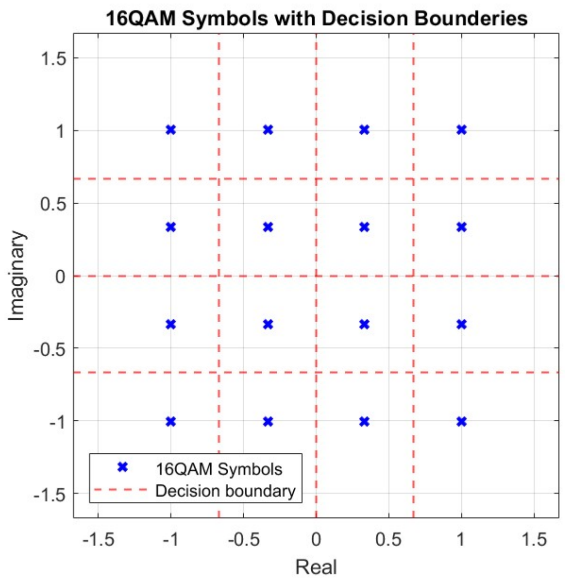

Figure 6: 16QAM Symbols and Detection Regions

The function defines the symbols and apply a preset scaling to achieve the average symbol energy of one. Then four bits are used to create the first symbol by converting the binary value of the four bits to an decimal number used as index to choose correct symbol. The loop continues until all bits are mapped to the correct symbol and the symbol vector is returned.

```matlab
function s = maponto16QAM(c)
Es = 1;
% Create 16QAM Symbols
P = 4*Es/(sqrt(2)+2*sqrt(10)+sqrt(18));
Symbols = [-3+3i,-3+1i,-3-3i,-3-1i, ...
           -1+3i,-1+1i,-1-3i,-1-1i, ...
            3+3i, 3+1i, 3-3i, 3-1i, ...
            1+3i, 1+1i, 1-3i, 1-1i]*P;

% Map bits to 16QAM symbols
s = (1:length(c)/4);
for i = 1:length(c)/4
    a = c(4*(i-1)+1:4*(i-1)+4);
    s(i) = Symbols(bin2dec(char(a + '0'))+1);
end
end
```

Listing 2: MATLAB code

2.4 Subtask 4

The fourth task consist of constructing another function mapping a bit vector to the symbols of BPSK, resulting in a symbol vector with the length one fourth of the original bit vector.

$$s = \text{mapontoBPSK}(c) \tag{6}$$

The BPSK symbol are either an one or a negative one, corresponding to two different phases 180 degrees apart of the carrier frequency of the signal. The decision boundary between the two symbols is placed along zero on the real axis.

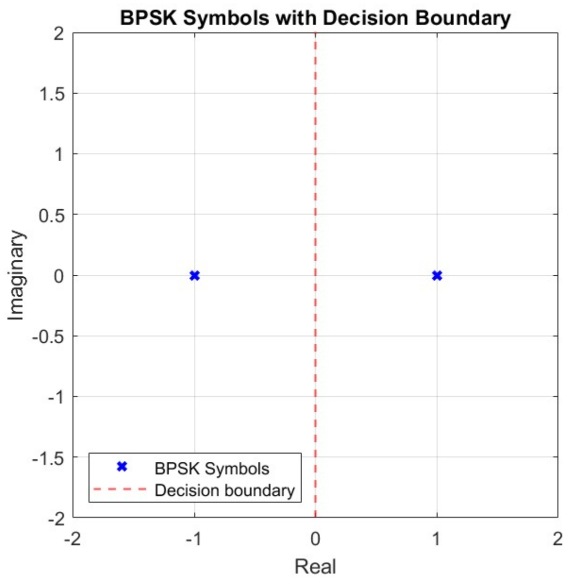

Figure 7: BPSK Symbols and Decision Boundary

The function simply converts the binary vector to and vector of ones and negative ones by multiplying the bit by two and subtracting one resulting in either an one or an negative one.

```matlab
function s = mapontoBPSK(c)
% Convert bits to symbols, BPSK mapping: 0 -> -1, 1 -> 1
s = 2*c - 1;
end
```

Listing 3: MATLAB code

2.5 Subtask 5

The fifth task was to construct a function detecting the symbols of 16QAM, converting the symbols back to bits using maximum likelihood detection.

$$c = \text{detect16QAM(r)} \tag{7}$$

The received symbols are Gaussian distributed around the originally sent symbol after passing through the complex AWGN channel. The spread of distribution depend on the variance of the noise in the channel.

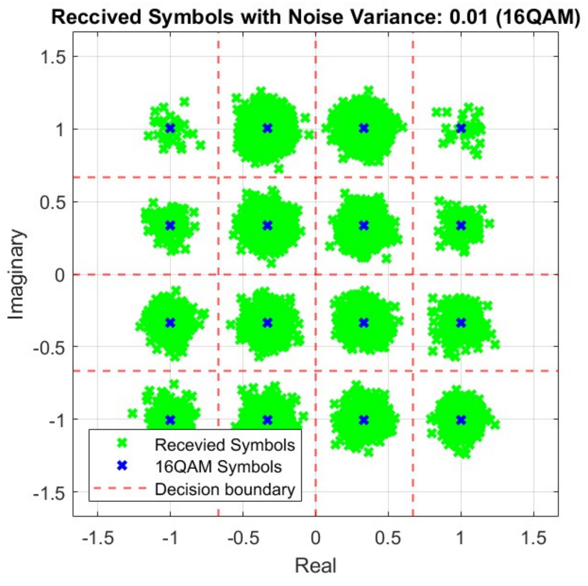

Figure 8: 16QAM Symbols, Decision Boundary and Received Symbols

The function determine the correct symbol by calculating the distance to every 16QAM symbol in the constellation and chooses the symbol closest to the received symbol. The index of the constellation symbol chosen is converted to a four bit binary number and added to the resulting vector returned after all symbols are processed.

```matlab
function c = detect16QAM(r)
Es = 1;
% Define 16-QAM constellation points
P = 4*Es/(sqrt(2)+2*sqrt(10)+sqrt(18));
Symbols = [-3+3i,-3+1i,-3-3i,-3-1i, ...
           -1+3i,-1+1i,-1-3i,-1-1i, ...
            3+3i, 3+1i, 3-3i, 3-1i, ...
            1+3i, 1+1i, 1-3i, 1-1i]*P;

% Perform detection
c = [];
for i = 1:length(r)
    % Compute Euclidean distances
    distances = abs(r(i) - Symbols).^2;
    [~, index] = min(distances); % Find index of the minimum distance

    % Map index to binary sequence
    c = [c, dec2bin(index-1,4) - '0'];
end
```

Listing 4: MATLAB code

2.6 Subtask 6

The sixth task was to construct the function detecting the symbols of BPSK, converting the symbols back to bits using maximum likelihood detection.

$$c = \text{detectBPSK}(r) \tag{8}$$

The received symbols are Gaussian distributed around the originally sent symbol after passing through the complex AWGN channel. The spread of distribution depend on the variance of the noise in the channel.

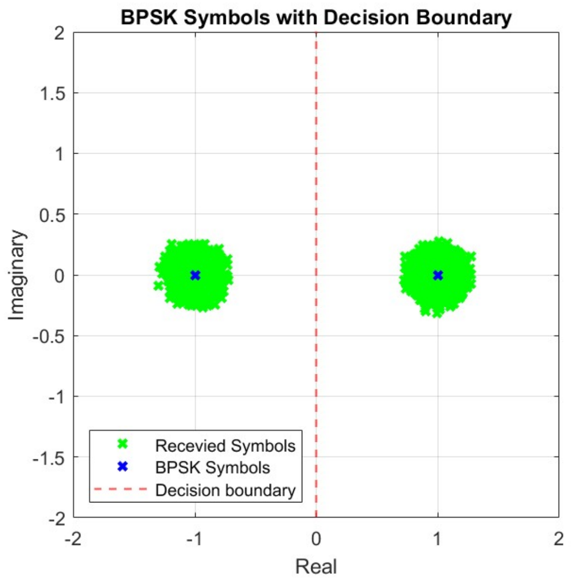

Figure 9: BPSK Symbols, Decision Boundary and Received Symbols

The function determine the correct symbol by determining if the recevied symbol has a positive or negative real part. If the real value is positive is the bit decided to be a one and if it is negative is the bit a zero. All bits are added to the resulting vector returned after all symbols are processed.

```matlab
function c = detectBPSK(r)
% Decision threshold for BPSK demodulation
threshold = 0;

% Perform detection
% If received symbol is greater than threshold, output 1; otherwise,
    output 0
c = (1:length(r));
for i = 1:length(r)
    if (r(i) > threshold)
        c(i) = 1;
    else
        c(i) = 0;
    end
end
end
```

Listing 5: MATLAB code

2.7 Subtask 7

The seventh task was to implement the functions from previous tasks to produce bit error rate (BER) curves for 16QAM and BPSK for multiple signal to noise ratios (SNR)s.

$$\gamma(s) = \frac{E_s}{N_0} \tag{9}$$

The noise variance was taken from a vector with 20 logarithmically spaced points from 0.01 to 1. For each value of noise variance was the signal noise ratio calculated and converted to decibels for plotting the BER curve. The BER were empirically found by sending a bit vector through the mapping function, then the channel ans finally the detection function. Then the original and resulting bit vectors were compared and the number of errors were divided by the total number of bits to find the error rate.

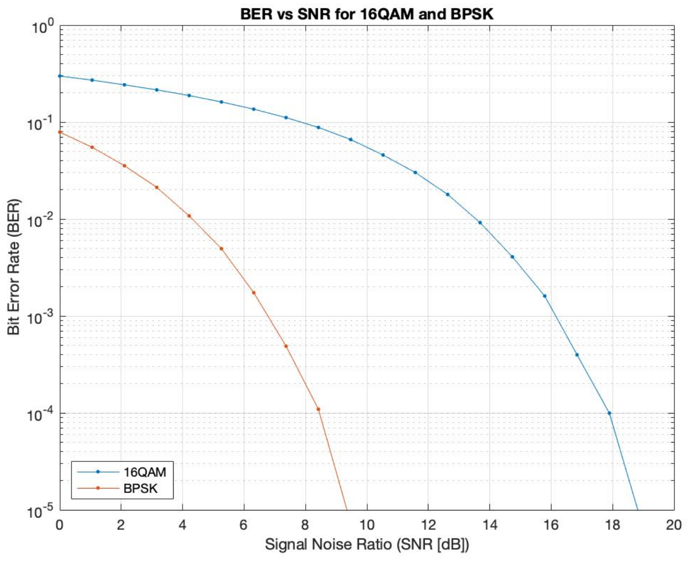

Figure 10: BPSK Symbols, Decision Boundary and Received Symbols

```
Es = 1;
No = logspace(-2,0,20);
EsNo = Es./No;
EsNo_dB = 10*log10(EsNo);

BER16QAM = (1:length(No));
%BER for all signal noise ratios
for i = 1:length(No)

    % 16QAM Symbols and AWGN channel
    s = maponto16QAM(c);
    r = channel2(s,No(i));
    b = detect16QAM(r);

    % Claculate and store BER for plotting
    BER = sum(c ~= b)/length(c);
    BER16QAM(i) = BER;
end
```

Listing 6: MATLAB code

2.8 Subtask 8

The eighth and final task was to provide the analytical expression for the BER of 16QAM and BPSK modulation. Then calculate the theoretical values and plot the BER curves to the empirical curves.

The theoretical BER for 16QAM and BPSK are given by equation 10 and 11.

$$\text{16QAM: } P_b \approx \frac{4}{log_2 M} Q\left(\sqrt{\frac{3\gamma_b log_2 M}{M-1}}\right) \tag{10}$$

$$\text{BPSK: } P_b \approx Q\left(\sqrt{2\gamma_b}\right) \tag{11}$$

When empirical and theoretical BER curves are plotted are the BPSK simulated and close to identical as the theoretical while the 16QAM does differ slightly. The empirical BER values for 16QAM are lower than the theoretical yet follows the same general shape of the curve.

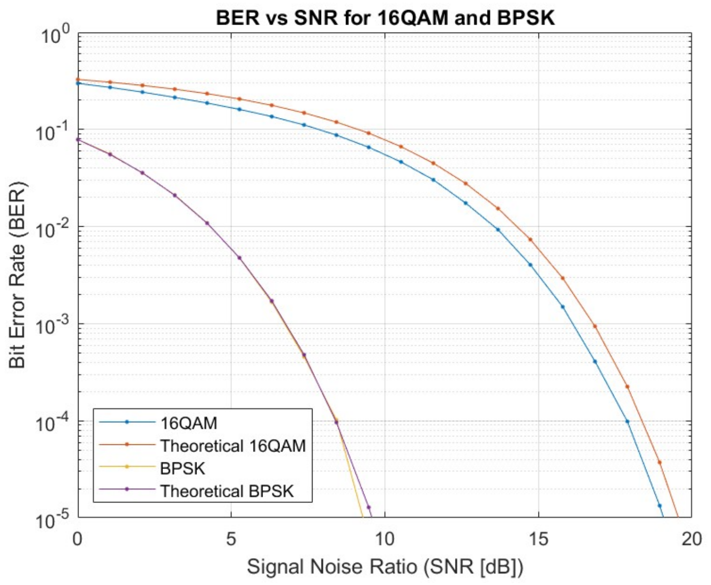

Figure 11: Empiriacal and Theoretical BER for 16QAM and BPSK

References

[1] *Wireless Communication, Andrea Goldsmith,* Available at: http://fa.ee.sut.ac.ir/Downloads/AcademicStaff/1/Courses/7/Andrea%20Goldsmith-Wireless%20Communications-Cambridge%20University%20Press%20%282005%29.pdf Accessed: March 1, 2024.

A Appendix

```
close all
clearvars
clc

load("Bitstream4bit.mat");
c = estimatedBitStream;

%TestChannel2()
%QPSK()
%Test16QAM(c)
TestBPSK(c)
%BERplot(c)

%Functions
function noisySignal = channel2(transmittedSignal,noisePower)
% Calculate the variance
variance = noisePower/2;

% Noise for real and imaginary parts
noise_real = sqrt(variance) * randn(size(transmittedSignal));
noise_imaginary = sqrt(variance) * randn(size(transmittedSignal));

% Combine to form complex noise
noise = noise_real + 1i * noise_imaginary;

% Add noise to transmitted signal
noisySignal = transmittedSignal + noise;
end

function QPSK()
% Number of symbols
numSymbols = 4;

% Phase angles for symbols
phase_angles = [pi/4, 3*pi/4, 5*pi/4, 7*pi/4];

% Create complex symbols
symbols = (1:4);
for i = 1:numSymbols
    symbols(i) = exp(1i * phase_angles(i));
end

% Verify average symbol power
average_power = mean(abs(symbols).^2);
disp(['QPSK Average symbol power: ', num2str(average_power)]);

% Plot the symbols and detection regions in the complex plane
figure;
plot(real(symbols), imag(symbols), 'cx', 'MarkerSize', 10,'LineWidth',
     3);
xlabel('In-phase');
ylabel('Quadrature');
title('QPSK Symbols with Detection Regions');

xlim([-3.2, 3.2]);
```

```
ylim([-3.2, 3.2]);
axis square;
grid on;
hold on;

% Detection regions and plot
xline(0, 'r--', 'LineWidth', 3);
yline(0, 'r--', 'LineWidth', 3);
legend('Sybols', 'Decision Boundries', 'Location', 'southeast')

hold off;

% Verify for different noise power
SignalSymbols = symbols(2) * ones(1, 1000);
noisePower2 = [0.01, 0.1, 1];

for i = 1:length(noisePower2)
    ReccivedSymbols = channel2(SignalSymbols,noisePower2(i));
    figure;
    plot(real(symbols), imag(symbols), 'cx', 'MarkerSize', 10, '
   LineWidth', 3);
    hold on;
    plot(real(ReccivedSymbols), imag(ReccivedSymbols), 'b.', '
   MarkerSize', 10, 'LineWidth', 2);
    plot(real(SignalSymbols), imag(SignalSymbols), 'rx', 'MarkerSize',
    10, 'LineWidth', 3);
    xlabel('Real-Part');
    ylabel('Imaginary-Part');

    title(['QPSK Sent Vs Reccived Symbols with Detection Regions,
   noise variance: ', num2str(noisePower2(i))]);
    xline(0, 'r--', 'LineWidth', 3);
    yline(0, 'r--', 'LineWidth', 3);
    xlim([-3.2, 3.2]);
    ylim([-3.2, 3.2]);
    legend('Sybols', 'Noisy Reccived Symbols', 'Sent Symbol', '
   Decision Boundries', 'Location', 'southeast')
    axis square;
    grid on;
    hold off;

    %SER
    ErrorSymbol = sum(angle(ReccivedSymbols) > pi | angle(
   ReccivedSymbols) < pi/2);
    SER = ErrorSymbol/length(ReccivedSymbols);
    disp(['QPSK Noise varriance: (', num2str(noisePower2(i)),'),
   Empirical SER: ', num2str(SER)]);%, ', Empirical BER: ' num2str(
   SER)]);

end
end

function s = maponto16QAM(c)
Es = 1;
% Create 16QAM Symbols
P = 4*Es/(sqrt(2)+2*sqrt(10)+sqrt(18));
Symbols = [-3+3i,-3+1i,-3-3i,-3-1i, ...
           -1+3i,-1+1i,-1-3i,-1-1i, ...
            3+3i, 3+1i, 3-3i, 3-1i, ...
            1+3i, 1+1i, 1-3i, 1-1i]*P;

% Map bits to 16QAM symbols
s = (1:length(c)/4);
for i = 1:length(c)/4
    a = c(4*(i-1)+1:4*(i-1)+4);
```

```matlab
    s(i) = Symbols(bin2dec(char(a + '0'))+1);
end
end

function s = mapontoBPSK(c)
% Convert bits to symbols, BPSK mapping: 0 -> -1, 1 -> 1
s = 2*c - 1;
end

function c = detect16QAM(r)
Es = 1;
% Define 16-QAM constellation points
P = 4*Es/(sqrt(2)+2*sqrt(10)+sqrt(18));
Symbols = [-3+3i,-3+1i,-3-3i,-3-1i, ...
           -1+3i,-1+1i,-1-3i,-1-1i, ...
            3+3i, 3+1i, 3-3i, 3-1i, ...
            1+3i, 1+1i, 1-3i, 1-1i]*P;

% Perform detection
c = [];
for i = 1:length(r)
    % Compute Euclidean distances
    distances = abs(r(i) - Symbols).^2;
    [~, index] = min(distances); % Find index of the minimum distance

    % Map index to binary sequence
    c = [c, dec2bin(index-1,4) - '0'];
end

end

function c = detectBPSK(r)
% Decision threshold for BPSK demodulation
threshold = 0;

% Perform detection
% If received symbol is greater than threshold, output 1; otherwise,
    output 0
c = (1:length(r));
for i = 1:length(r)
    if (r(i) > threshold)
        c(i) = 1;
    else
        c(i) = 0;
    end
end
end

function TestChannel2()
% Example zero transmitted signal
transmittedSignal = zeros(1,1000);

% Example noise power
noisePower = 1;

% Apply noise to transmitted signal
noisySignal = channel2(transmittedSignal, noisePower);

% Calculate the power of the noisy signal
noisyPower = mean(abs(noisySignal).^2);
disp(['AWGN Noisy signal power: ', num2str(noisyPower)]);
end

function Test16QAM(c)
Es = 1;
```

```
s = maponto16QAM(c);
r = channel2(s,0.01);
b = detect16QAM(r);

% Define 16-QAM constellation points
P = 4*Es/(sqrt(2)+2*sqrt(10)+sqrt(18));

Symbols = [-3+3i,-3+1i,-3-3i,-3-1i, ...
           -1+3i,-1+1i,-1-3i,-1-1i, ...
            3+3i, 3+1i, 3-3i, 3-1i, ...
            1+3i, 1+1i, 1-3i, 1-1i]*P;

% Verify Average Symbol Energy
Es = 0;
for i = 1:length(Symbols)
    Es = Es + abs(Symbols(i));
end
Es = Es/numel(Symbols);
disp(['16QAM average symbol energy: ', num2str(Es)]);

%SER
DeterminedSymbols = maponto16QAM(b);
ErrorSymbol = sum(DeterminedSymbols ~= s);
BER = ErrorSymbol/length(s);

%BER
ErrorBit = sum(c ~= b);
SER = ErrorBit/length(c);

disp(['16QAM Empirical SER: ', num2str(SER), ', Empirical BER: '
    num2str(BER)]);

% Plot Symbols
figure;
plot(real(Symbols), imag(Symbols), 'bx','LineWidth', 2);
xlabel('Real');
ylabel('Imaginary');
title('16QAM Symbols with Decision Bounderies');
xlim([-5*P,5*P]);
ylim([-5*P,5*P]);
xline(0, 'r--', 'LineWidth', 1);
xline(2*P, 'r--', 'LineWidth', 1);
xline(-2*P, 'r--', 'LineWidth', 1);
yline(0, 'r--', 'LineWidth', 1);
yline(2*P, 'r--', 'LineWidth', 1);
yline(-2*P, 'r--', 'LineWidth', 1);
axis square;
grid on;
hold on;
box on;
legend('16QAM Symbols', 'Decision boundary', 'Location', 'southwest');

% Plot
figure;
hold on;
plot(real(r), imag(r), 'gx','LineWidth', 2);
plot(real(Symbols), imag(Symbols), 'bx','LineWidth', 2);
xlabel('Real');
ylabel('Imaginary');
title('Reccived Symbols with Noise Variance: 0.01 (16QAM)');
xlim([-5*P,5*P]);
ylim([-5*P,5*P]);
xline(0, 'r--', 'LineWidth', 1);
xline(2*P, 'r--', 'LineWidth', 1);
xline(-2*P, 'r--', 'LineWidth', 1);
```

```
yline(0, 'r--', 'LineWidth', 1);
yline(2*P, 'r--', 'LineWidth', 1);
yline(-2*P, 'r--', 'LineWidth', 1);
axis square;
box on;
grid on;
legend('Recevied Symbols', '16QAM Symbols', 'Decision boundary', '
    Location', 'southwest');

end

function TestBPSK(c)
s = mapontoBPSK(c);
r = channel2(s,0.01);

figure;
xlabel('Real');
ylabel('Imaginary');
title('BPSK Symbols with Decision Boundary');
xlim([-2,2]);
ylim([-2,2]);
axis square;
grid on;
hold on;
box on;
plot(real(r), imag(r), 'gx','LineWidth', 2);
plot([-1,1], [0,0], 'bx','LineWidth', 2);
xline(0, 'r--', 'LineWidth', 1);
legend('Recevied Symbols', 'BPSK Symbols', 'Decision boundary', '
    Location', 'southwest');

figure;
xlabel('Real');
ylabel('Imaginary');
title('BPSK Symbols with Decision Boundary');
xlim([-2,2]);
ylim([-2,2]);
axis square;
grid on;
hold on;
box on;
plot([-1,1], [0,0], 'bx','LineWidth', 2);
xline(0, 'r--', 'LineWidth', 1);
legend('BPSK Symbols', 'Decision boundary', 'Location', 'southwest');

end

function BERplot(c)
% Repeating vector for larger amount of samples
c = [c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c];

% Noise variance vector
No = logspace(-2,0,20);

%testing
BER16QAM = (1:length(No));
for i = 1:length(No)
    %
    s = maponto16QAM(c);
    r = channel2(s,No(i));
    b = detect16QAM(r);

    % Claculate and store BER for plotting
    BER = sum(c ~= b)/length(c);
    BER16QAM(i) = BER;
```

```
end

BERBPSK = (1:length(No));
for i = 1:length(No)
    %
    s = mapontoBPSK(c);
    r = channel2(s,No(i));
    b = detectBPSK(r);

    % Claculate and store BER for plotting
    BER = sum(c ~= b)/length(c);
    BERBPSK(i) = BER;
end

% Calculating theoretical BER
Es = 1;
Eb = Es/log2(16);
EsNo = Es./No;
EbNo = Eb./No;
EsNo_dB = 10*log10(EsNo);

BER16QAMTHEO = (4/log2(16))*qfunc(sqrt((3*(EbNo)*log2(16))/(16-1)));
BERBPSKTHEO = qfunc(sqrt(2*EsNo));

% Plotting BER
figure;
semilogy(EsNo_dB, BER16QAM, '.-');
hold on;
semilogy(EsNo_dB, BER16QAMTHEO, '.-');
semilogy(EsNo_dB, BERBPSK, '.-');
semilogy(EsNo_dB, BERBPSKTHEO, '.-');
title('BER vs SNR for 16QAM and BPSK');
xlabel('Signal Noise Ratio (SNR [dB])');
ylabel('Bit Error Rate (BER)');
ylim([10^-5,1]);
grid on;
legend('16QAM', 'Theoretical 16QAM', 'BPSK', 'Theoretical BPSK', '
    Location', 'southwest');


end
```
Listing 7: MATLAB code