ECE 462 Design Project

Joshua Wentzel
Caleb Laws

March 13, 2021

Contents

1 Problem Statement

The goal of this project was to create a 8-QAM digital communication system over an AWGN channel on Matlab. The signal constellation diagram is shown by Figure 1. This communication system was then placed through an increasingly noisy channel to test the Bit Error Rate (BER) and Symbol Error Rate (SER).

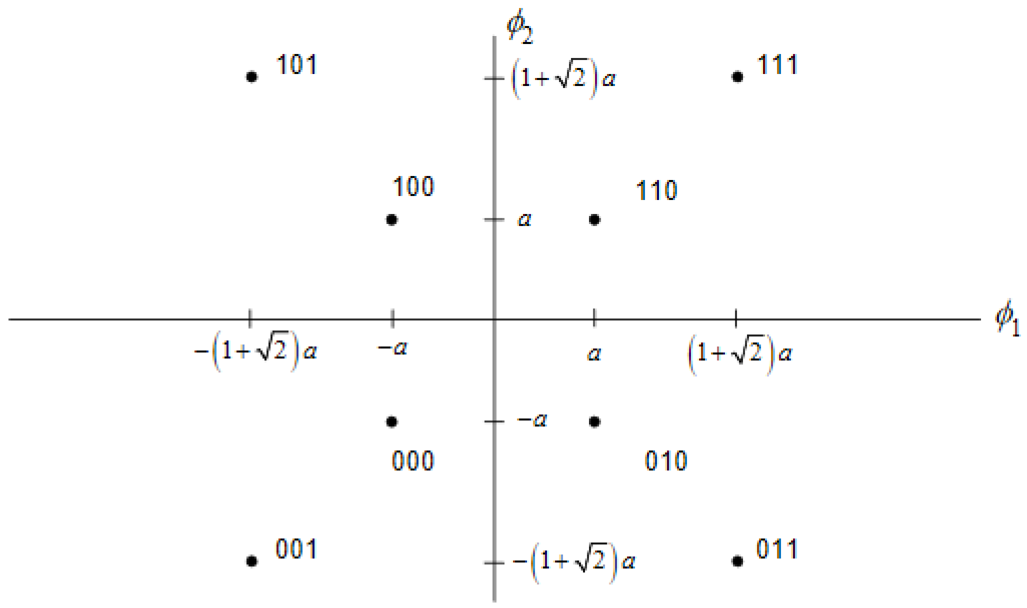

Figure 1: The symbol constellation diagram.

2 Assumptions

- The noise across the channel is Additive White Gaussian (AWG) and modeled by: $S_w(f) = N_0/2, \forall f$.
- The simulation SNR will start at 0 and increase by 2 up until 12 dB.
- There is equal probability that the transmitted will send each of the 8 symbols.
- Gaussian RNGs are used to generate uncorrelated, zero-mean noise components (,)IQnn each of which has a variance 20N= (because this is a baseband implementation).

3 Analysis

3.1 General Approach

This project tasked simulating 8-QAM digital communication in the presence of varying levels of noise consists of several distinct components. Each component belongs to one of two larger blocks. These blocks are depicted in Figure 2.

The "Generate Received Sequence" block is responsible for generating the 3-bit symbol sequence(S_i), mapping it to the constellation and multiplying appropriately to ensure proper SNR (finding (a_i, b_i)), adding appropriate noises (n_{Ii} and n_{Qi}), and sending the information to "Receiver Analysis" block.

The "Receiver Analysis" block is responsible for handling the data sent by the "Generate Received Sequence" block, determining detected symbols, evaluating transmission performance, and calculating error rates.

Three crucial questions must be answered before we can create the simulation: How do we find a_i and b_i? How do we find n_{Ii} and n_{Qi}? Once we have $r1_i$ and $r2_i$, how do we determine the detected symbol, D_i?

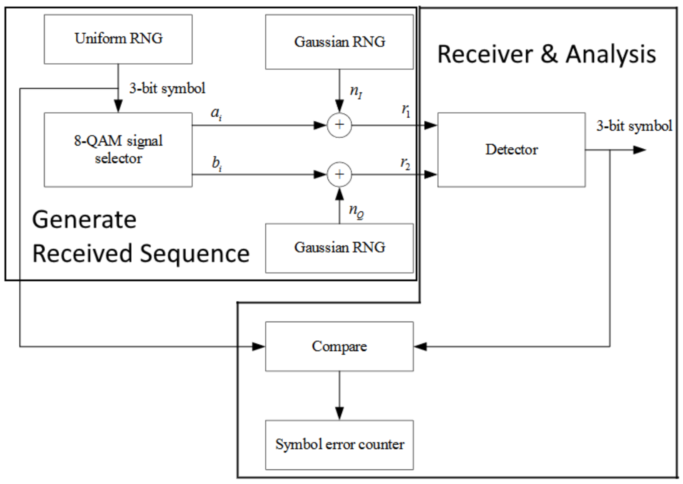

Figure 2: Block Diagram

3.2 Finding Signal Points and Noise

As it happens, the first two questions are very closely linked and thus will be addressed together.

3.2.1 Background

A key component of the "Generate Received Sequence" block is the 8-QAM signal selector. This component transforms the ith 3 bit symbol S_i to the corresponding signal points (a_i, b_i). For example, the 3 bit symbol '110' would be mapped to the signal point coordinates (a, a).

The question of what exactly a is follows shortly and is critical to implementing the "Generate Received Sequence" block.

3.2.2 Signal Energy

According to pages 419 and 424, we impose the condition that $d_{min} = 2\sqrt{\mathscr{E}_s}$ where d_{min} is the minimum distance between signal points. Referring to Figure 1 and using simple geometry we determine $d_{min} = 2a$ and that each neighbor of one of the inner four signal points is separated by exactly this distance. Thus we have the relationship

$$d_{min} = 2a = 2\sqrt{\mathscr{E}_s}.$$

Thus,

$$a = \sqrt{\mathscr{E}_s}.$$

Indeed a is no random constant, it is the square root of the signal energy and it scales the signal points accordingly.

3.2.3 Average Symbol Energy

We attempt to further understand the role of a by using the assumption that all signal points are equally probably and the equation

$$\mathscr{E}_{av} = \frac{\mathscr{E}_s}{M} \sum_{m=1}^{M} (a_{mc}^2 + b_{ms}^2).$$

Where $\mathscr{E}_{av}$ is the average energy contained in one symbol, $M = 8$ is the number of signal points (symbols), and (a_{mc}, b_{ms}) are the normalized coordinates (scaled by a) of the signal points given by Figure 1. We find

$$\mathscr{E}_{av} = \mathscr{E}_s(4 + 2\sqrt{2}) = a^2(4 + 2\sqrt{2}).$$

We now have related the average symbol energy to our constant $a = \sqrt{\mathscr{E}_s}$.

3.2.4 SNR and Noise

Next we attempt to fit SNR and noise into the picture. We start by noting that $\mathscr{E}_{av}/N_0$ is the average SNR/symbol. Thus we have the following:

$$\text{SNR/symbol} = \mathscr{E}_{av}/N_0$$

We are more concerned with the average SNR/bit, $\mathscr{E}_{bav}$. With $\mathscr{E}_{av} = k \times \mathscr{E}_{bav}$ where $M = 2^k$. Thus

$$\mathscr{E}_{av} = 3 \times \mathscr{E}_{bav}$$

and we have

$$\text{SNR/symbol} = 3 \times \text{SNR/bit}$$

We now refer to SNR/bit as "SNR" where we are careful to note that SNR is expressed linearly.

$$\mathscr{E}_{av}/N_0 = \frac{3 \times \mathscr{E}_{bav}}{N_0}$$
$$\mathscr{E}_{av}/N_0 = 3 \times \text{SNR}$$
$$\mathscr{E}_{av} = 3 \times \text{SNR} \times N_0$$

$$\mathscr{E}_{av} = 3 \times \text{SNR} \times N_0$$
$$a^2(4+2\sqrt{2}) = 3 \times \text{SNR} \times N_0$$
$$a^2 = \frac{3 \times N_0}{(4+2\sqrt{2})}$$
$$a = \sqrt{\frac{3 \times N_0}{(4+2\sqrt{2})}}$$

Recall,

$$\text{SNR} = 10^{(\text{SNR}_{\text{dB}}/10)}$$

Substituting this we have:

$$a = \sqrt{\frac{3 \times 10^{(\text{SNR}_{\text{dB}}/10)} \times N_0}{(4+2\sqrt{2})}}$$

Let $N_0 = 1$ for convenience. We have:

$$a = \sqrt{\frac{3 \times 10^{(\text{SNR}_{\text{dB}}/10)}}{(4+2\sqrt{2})}}$$

With a found we can determine (a_i, b_i). Also, with $N_0 = 1$ we know that the noise has variance equal to 1. Thus we now have the answers to the first of our two questions.

3.3 Symbol Detection

Finally, we answer the question how do we determine the detected symbol, D_im from the received signal, $r1_i$ and $r2_i$. The answer is we calculate the distance between the received signal and each of the signal points. The minimum distance corresponds to the closest signal point and thus the most likely transmitted symbol. This can be visualized using a Voronoi diagram (Figure 3)in which the decision space is partitioned into regions closest to each of the signal points. Thus all critical questions have been answered and we can proceed to implementation.

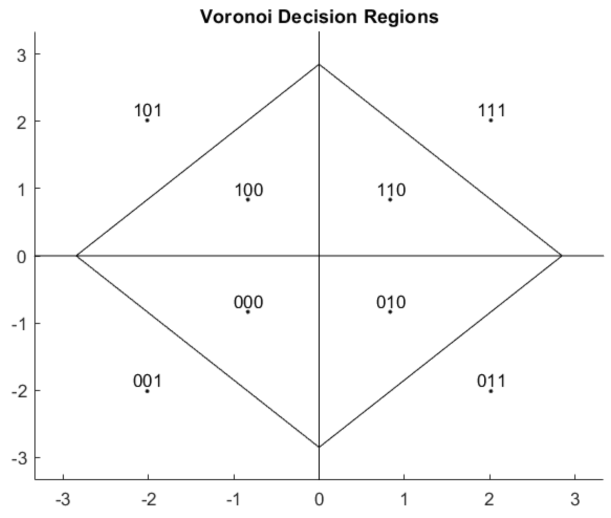

Figure 3: Voronoi Decision Regions

3.4 Additional Note: Gray Encoding

Gray Coding is ensuring that each symbol on the constellation diagram is only one bit away from it's nearest symbol. This minimizes the number of bits that can be demodulated with error. Referring back to the original symbol constellation diagram (Figure 1) we can prove graphically that each of the nearest symbols are all only one bit away from each other. This by definition is designing the model to be gray coded.

Additionally, we can use the simulation results to confirm that the error seen is what would be expected from gray coding. Unlike non gray coding, due to the low probability of receiving more then one incorrect bit in a symbol we expect to see a equal number of mistranslated bits and symbols which can also be written as $P_b = P_s/3$. As you can see in Table 1 in the Simulation Results section, the Bit to Symbol Error ratio is very close to 1 approaching it as the SNR increases. Again, this is what we would expect from a gray coded system.

4 Simulation Setup

4.1 Decentralized Design

The simulation is divided into three MATLAB programs, each with their own specific purpose. Because the generation of the received sequence and the actual receiving of the sequence are separated it is often possible to adjust one without needing run them both. This significantly hastened simulation development.

4.2 Definitions

The definitions.m program is responsible for storing several constants and calculated constants used by one or both programs necessary for the simulation. It is here that we store which SNRs will be tested, how many symbols will be transmitted for each SNR, the power_mul constant, and the constellation array.

4.3 Received Signal Generation

The received_signal_generator.m program is responsible for generating the symbol sequence, mapping it to the constellation, multiplying by the necessary amount to ensure proper SNR, adding appropriate noise, and saving the information for use by the receiver program.

4.4 Receiving

The receiver.m program is responsible for loading the data saved by the received signal generator, determining detected symbols, evaluating transmission performance, and calculating error rates.

5 Simulation Results

Symbol and Bit Error Rate

dB SNR	Symbol Error Rate	Bit Error Rate	Bit to Symbol Error Ratio
0dB	0.477260	0.197243	1.239848
2dB	0.384040	0.146497	1.144386
4dB	0.282490	0.100883	1.071365
6dB	0.181840	0.062537	1.031731
8dB	0.093590	0.031680	1.015493
10dB	0.035580	0.011927	1.005621
12dB	0.008120	0.002707	1.000000

Table 1: Analysis of gray coding.

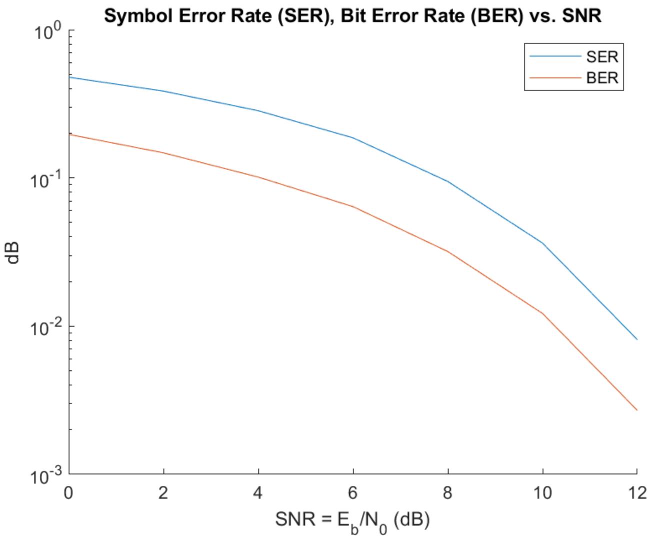

Figure 4: SER and BER comparison

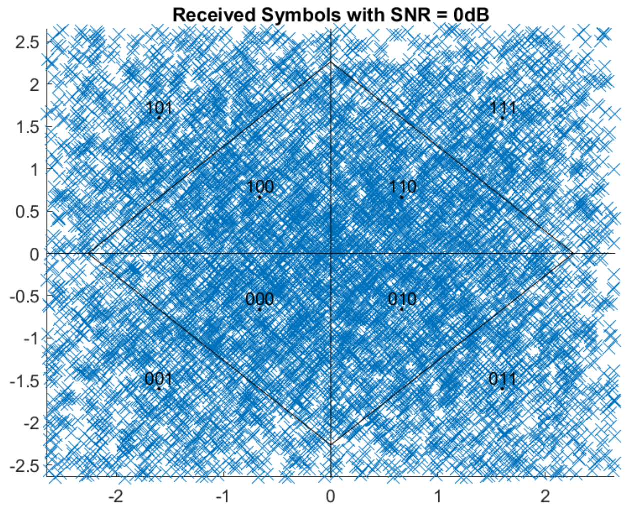

Figure 5: 0dB SNR

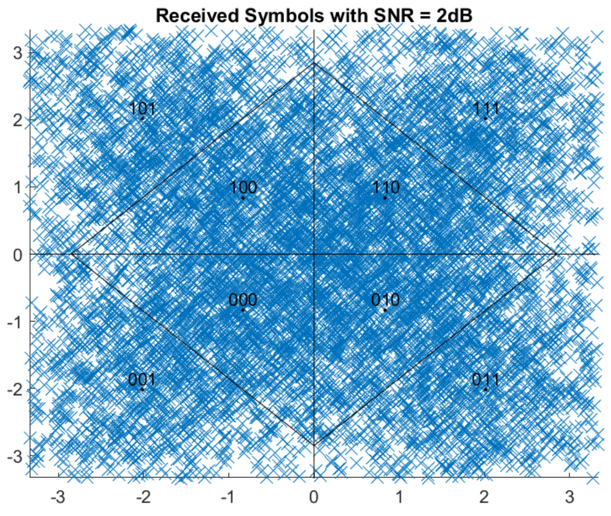

Figure 6: 2dB SNR

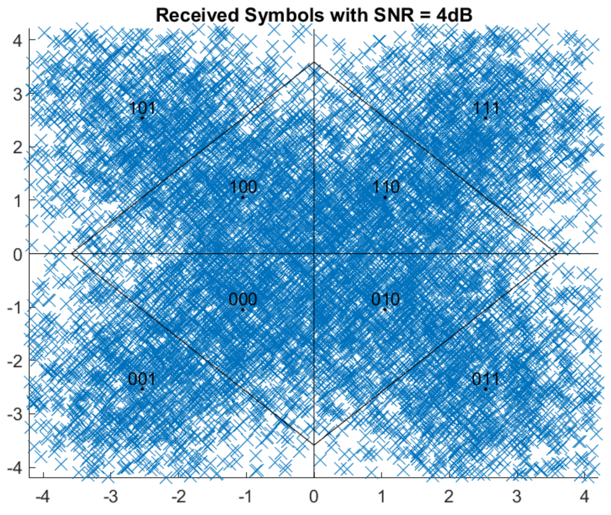

Figure 7: 4dB SNR

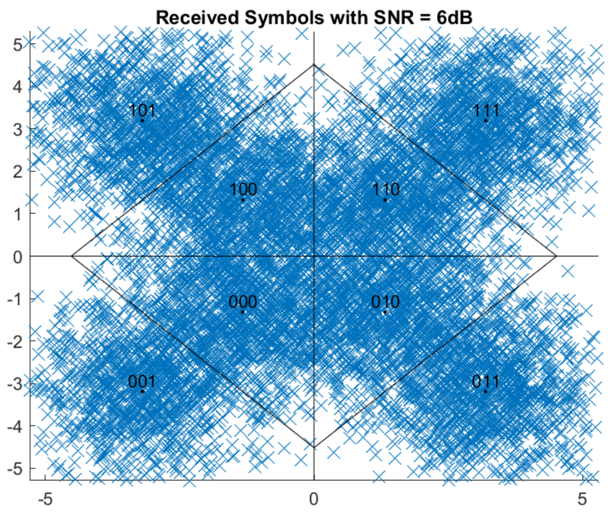

Figure 8: 6dB SNR

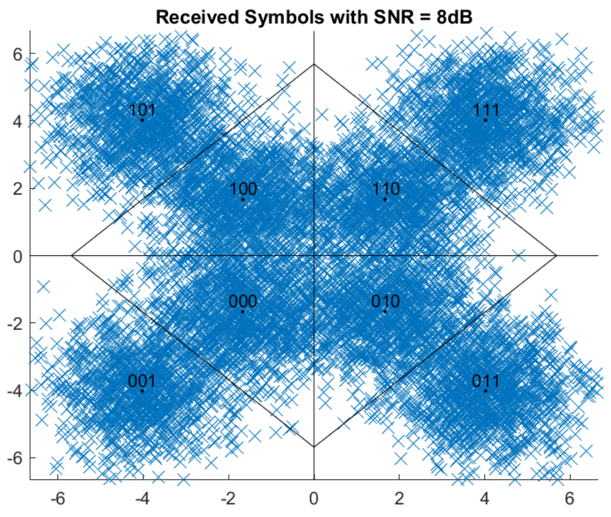

Figure 9: 8dB SNR

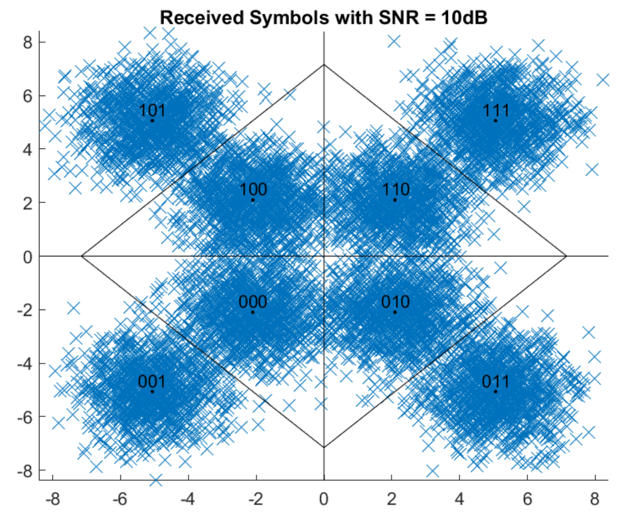

Figure 10: 10dB SNR

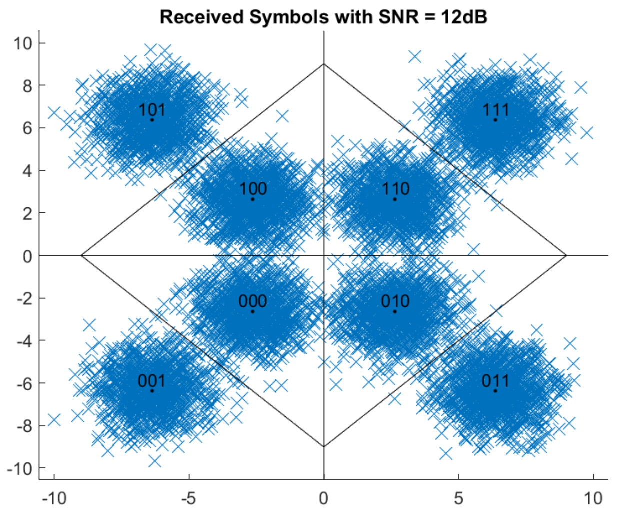

Figure 11: 12dB SNR

6 Discussion of Results

As expected the increase of SNR has visibly improved the performance of the system. Referencing the table of Symbol Error rates our system will be able to transmit a 3 Bit symbol on a 6dB SNR channel with an 98.968269% certainty. This is quite reliable for a fairly high noise system. With a SNR of 12dB we see that our system is able to transmit symbols with a success rate of 99.188%.

The validity of the reliability of this system on top of the increase performance in transmission rate that QAM has compared to other transmission methods such as AM or FM makes it an appropriate choice for high bandwidth systems. QAM is currently being used in the 802.11 WiFi standard and high resolution digital video transmission so the applications of QAM will be prevalent for the foreseeable future.

7 Appendix

7.1 Definitions.m

```
% Create Definitions file

SNRs_tested_dB = [0 2 4 6 8 10 12];

total_experiments = length(SNRs_tested_dB);

power_mul = 3/(4+2*sqrt(2));  % used for finding Es = 'a'


symbols_per_experiment = 10^5;
plotted_symbols_per_experiment = symbols_per_experiment/10;

bits_per_symbol = 3;
symbols = 8;

% % figure drawing constants
% width=1280; % default 550
% height=200;
% height_modifier = 1.2;

mul = 1 + sqrt(2);

encoding_constellation = [-1 -1;-mul -mul;1 -1;mul -mul;-1 1;-mul mul;1 1; mul mul];
global encoding_constellation_x;
global encoding_constellation_y;
encoding_constellation_x = [-1,-mul,1,mul,-1,-mul,1,mul];
encoding_constellation_y = [-1,-mul,-1,-mul,1,mul,1,mul];

save('Definitions.mat')
```

7.2 Generate_Experiments.m

```
% used for generating the data

clc;
clear all;
close all;
SHOW_CHARTS = 1;

% load definitions
Definitions
load('Definitions.mat');

for idx = 1:total_experiments
    SNR_db = SNRs_tested_dB(idx);
    fprintf("SNR (dB) = %ddB\n",SNR_db);
    SNR = 10^(SNR_db/10);
    fprintf("SNR (linear) = %f\n",SNR);
    scale_factor = sqrt(SNR*power_mul); % Es = 'a'
    fprintf("Sqrt(Es) = %f\n",scale_factor);
    AverageSymbolPower = (4+2*sqrt(2))*scale_factor*scale_factor;
    fprintf("AverageSymbolPower = %f\n",AverageSymbolPower);
    AveragePowerPerBit = AverageSymbolPower/3;
    fprintf("AveragePowerPerBit = %f\n",AveragePowerPerBit);
    fprintf("\n");

    [s,n1,n2] = RandStream.create('mlfg6331_64','NumStreams',3);

    symbol_sequence = randi(s,[0 7],symbols_per_experiment,1);
    noise_1 = randn(n1,symbols_per_experiment,1);
    noise_2 = randn(n2,symbols_per_experiment,1);
    a_i = zeros(symbols_per_experiment,1);
    b_i = zeros(symbols_per_experiment,1);
    for i = 1:length(symbol_sequence)
        con_output = encoding_constellation(symbol_sequence(i)+1,:);
        %con_output = [1 1];
        a_i(i) = con_output(1);
        b_i(i) = con_output(2);
%         a_i(i) = 1;
%         b_i(i) = 1;
%         symbol_sequence(i) = 6;
    end
    a_i = a_i*(scale_factor);
    b_i = b_i*(scale_factor);

    received_a_i = a_i + noise_1;
    received_b_i = b_i + noise_2;

    channel_data = [symbol_sequence,noise_1,noise_2,a_i,b_i,received_a_i,received_b_i];
    filenameOUT = strcat('channel_data',num2str(idx),'.mat');
    save( filenameOUT, 'channel_data' );

    constant_data = [idx SNR_db SNR scale_factor];
    filenameOUT = strcat('constant_data',num2str(idx),'.mat');
    save( filenameOUT, 'constant_data' );
end
```

7.3 Receiver.m

```
% used for calculating error rates

clc;
clear all;
close all;
SHOW_CHARTS = 1;

% load definitions
Definitions
load('Definitions.mat');

%format long
SymbolErrorRates = zeros(total_experiments,1);
BitErrorRates = zeros(total_experiments,1);
TightSymbolBounds = zeros(total_experiments,1);

for idx = 1:total_experiments
    filenameIN1 = strcat('channel_data',num2str(idx),'.mat');
    filenameIN2 = strcat('constant_data',num2str(idx),'.mat');
    stuff1 = matfile(filenameIN1);
    stuff2 = matfile(filenameIN2);
    channel_data = stuff1.channel_data;
    constant_data = stuff2.constant_data;

    symbol_sequence = channel_data(:,1);
    noise_1 = channel_data(:,2);
    noise_2 = channel_data(:,3);
    a_i = channel_data(:,4);
    b_i = channel_data(:,5);
    received_a_i = channel_data(:,6);
    received_b_i = channel_data(:,7);

    %idx = constant_data(1);    not needed
```

```
SNR_db = constant_data(2);
SNR = constant_data(3);
scale_factor = constant_data(4);

detected_symbol_sequence = zeros(symbols_per_experiment,1);

total_symbol_errors = 0;
total_bit_errors = 0;

for s = 1:length(symbol_sequence)
    closest_symbol_index = 1;
    closest_distance = compute_distance_from_symbol(received_a_i(s),received_b_i(s),scal
    for i = 2:8
        distance = compute_distance_from_symbol(received_a_i(s),received_b_i(s),scale_fa
        if(distance < closest_distance)
            closest_distance = distance;
            closest_symbol_index = i;
        end
    end
    detected_symbol_sequence(s) = closest_symbol_index;
    detected_string = num2str(dec2bin(closest_symbol_index-1,3));
    real_string = num2str(dec2bin(symbol_sequence(s),3));
    if(strcmp(detected_string(1),real_string(1))~= 1)
        total_bit_errors = total_bit_errors+1;
    end
    if(strcmp(detected_string(2),real_string(2))~= 1)
        total_bit_errors = total_bit_errors+1;
    end
    if(strcmp(detected_string(3),real_string(3))~= 1)
        total_bit_errors = total_bit_errors+1;
    end

    if((closest_symbol_index-1) ~= symbol_sequence(s))
        total_symbol_errors = total_symbol_errors + 1;
    end

end
fprintf("Test %idB:\n",SNR_db);
fprintf("symbols_per_experiment = %f\n",length(symbol_sequence));
fprintf("total_symbol_errors = %f\n",total_symbol_errors);
fprintf("total_bit_errors = %f\n",total_bit_errors);
fprintf("verify that total_bit_errors is between one third and triple total_symbol_error
fprintf("%f <= %f <= %f\n",total_symbol_errors/3,total_bit_errors,total_symbol_errors*3)
EstimatedSymbolErrorRate = total_symbol_errors/length(symbol_sequence);
fprintf("EstimatedSymbolErrorRate = %f\n",EstimatedSymbolErrorRate);
EstimatedBitErrorRate = total_bit_errors/(length(symbol_sequence)*3);
```

```
    fprintf("EstimatedBitErrorRate = %f\n",EstimatedBitErrorRate);
    AvgBitErrorsPerSymbolError = (EstimatedBitErrorRate*3)/EstimatedSymbolErrorRate;
    fprintf("AvgBitErrorsPerSymbolError = %f\n",AvgBitErrorsPerSymbolError);

    fprintf("The greycode estimate assumes 1 bit error per symbol error\n");
    GreycodeEstimateOfBitErrorRate = EstimatedSymbolErrorRate/3;
    fprintf("GreycodeEstimateOfBitErrorRate = %f\n",GreycodeEstimateOfBitErrorRate);
    fprintf("As SNR increases, EstimatedBitErrorRate approaches GreycodeEstimateOfBitErrorRa

    TightSymbolBound = 1 - (1-2*qfunc(sqrt(SNR*(3/7))))^2;
    fprintf("TightSymbolBound = %f\n",TightSymbolBound);
    fprintf("\n");
    fprintf("\n");

    SymbolErrorRates(idx) = EstimatedSymbolErrorRate;
    BitErrorRates(idx) = EstimatedBitErrorRate;
    TightSymbolBounds(idx) = TightSymbolBound;

    figure(idx)
    clf('reset')
    hold on;
    scatter(received_a_i(1:plotted_symbols_per_experiment),received_b_i(1:plotted_symbols_pe
    voronoi(encoding_constellation_x*scale_factor,encoding_constellation_y*scale_factor,'k')
    for k = 1:8
        text((encoding_constellation_x(k)-0.2)*scale_factor,(encoding_constellation_y(sym(k)
    end
    hold off;
    msg = strcat("Received Symbols with SNR = ", num2str(SNR_db),"dB");
    %msg = strcat("Voronoi Decision Regions");
    title(msg);
    axis([-4 4 -4 4]*scale_factor)
    pathname = strcat('outputs/received_symbols_SNR_',num2str(SNR_db),'.png');
    saveas(gcf,pathname)
end

figure(total_experiments+1)
clf('reset')
set(gca, 'YScale', 'log')
hold on;
plot(SNRs_tested_dB, SymbolErrorRates)
plot(SNRs_tested_dB, BitErrorRates)
hold off;
legend("SER","BER");
title("Symbol Error Rate (SER), Bit Error Rate (BER) vs. SNR")
xlabel("SNR = E_b/N_0 (dB)");
ylabel("dB");
```

```
saveas(gcf,'outputs/SER-BER-Chart.png')


function distance = compute_distance_from_symbol(a_i,b_i,scale_factor,symbol_index)
    global encoding_constellation_x;
    global encoding_constellation_y;
    symbol_x = encoding_constellation_x(symbol_index)*scale_factor;
    symbol_y = encoding_constellation_y(symbol_index)*scale_factor;

    distance = sqrt((a_i-symbol_x)^2+(b_i-symbol_y)^2);
end
```