陳志偉 B06901126 電機五

Electrical Engineering Lab（topics on Communication System）
Lab6 Report

1. The figure showed the implement of 3 type of modulation, and the figure 2 showed the corresponding output.

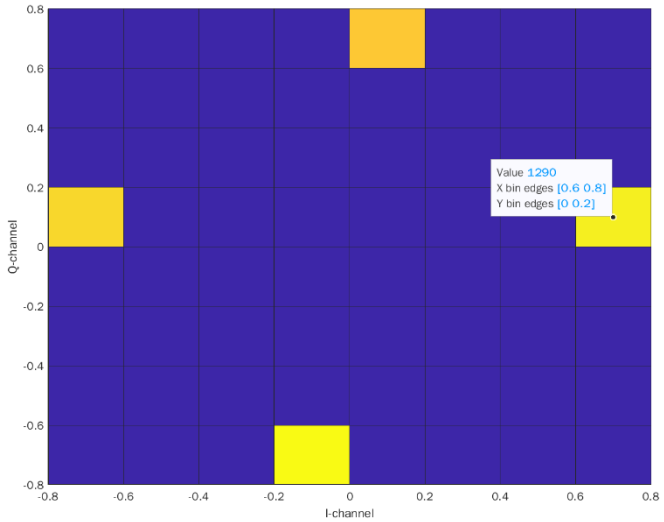
```
sym_seq = symbol_mapper([0 0 1 0], 4, 2, 'PAM')
sym_seq = symbol_mapper([0 0 1 0], 4, 2*sin(pi/4), 'PSK')
sym_seq = symbol_mapper([0 0 1 0], 4, 2, 'QAM')
```

Figure 1: Calling the implement function of problem 1

```
sym_seq =

     3    -3


sym_seq =

   1.0000 + 0.0000i  -0.0000 - 1.0000i


sym_seq =

   1.0000 - 1.0000i  -1.0000 - 1.0000i
```

Figure 2: The output of the sample calling of figure1

2. a)



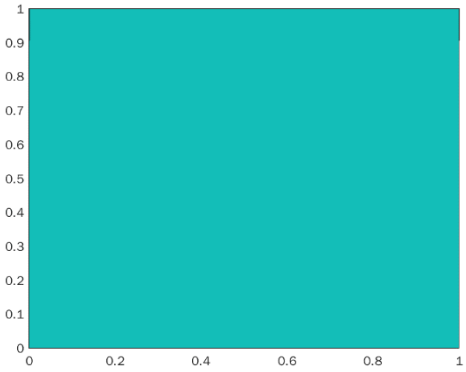


Figure 3 a): E_b/N_0 = 0 dB

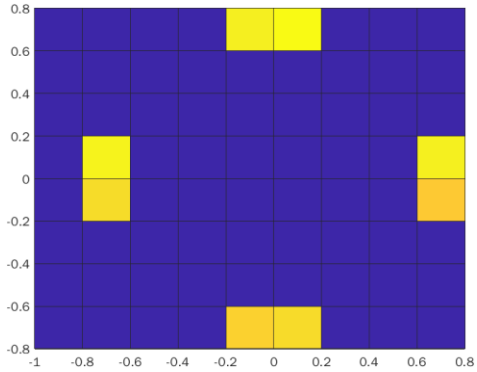


Figure 3 b): E_b/N_0 = 10 dB

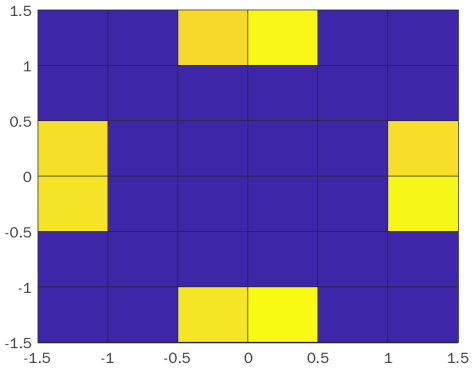


Figure 3 a): E_b/N_0 = 20 dB

Figure 3: The histogram of QPSK

b) From the Figure 4 the E_b/N_0 = 0 dB have almost 2% of SER and the rest of other 2 have 0% SER.

```
SER =

    0.0770         0         0
```

Figure 4: The SER from the received symbol sequence in problem 2a

3. a)

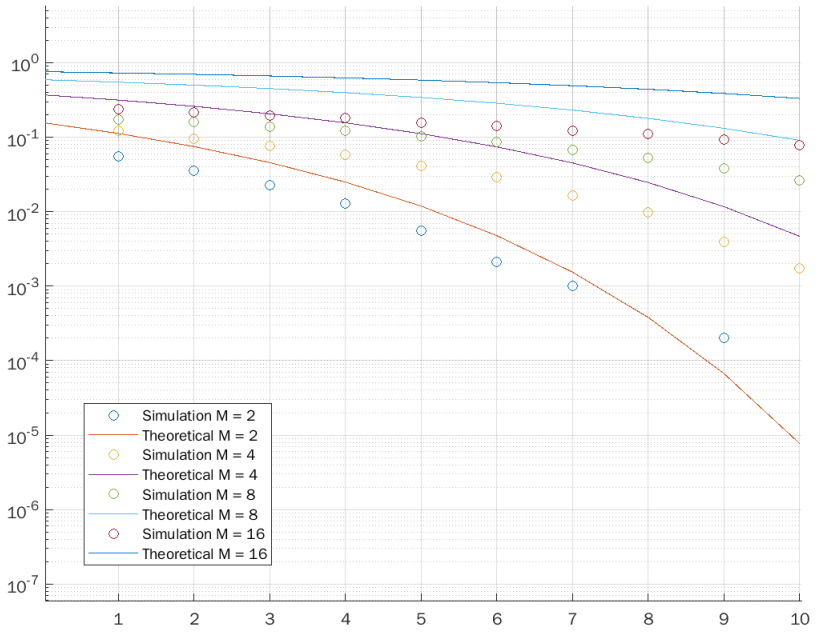


Figure 5: The simulation of PAM

b)

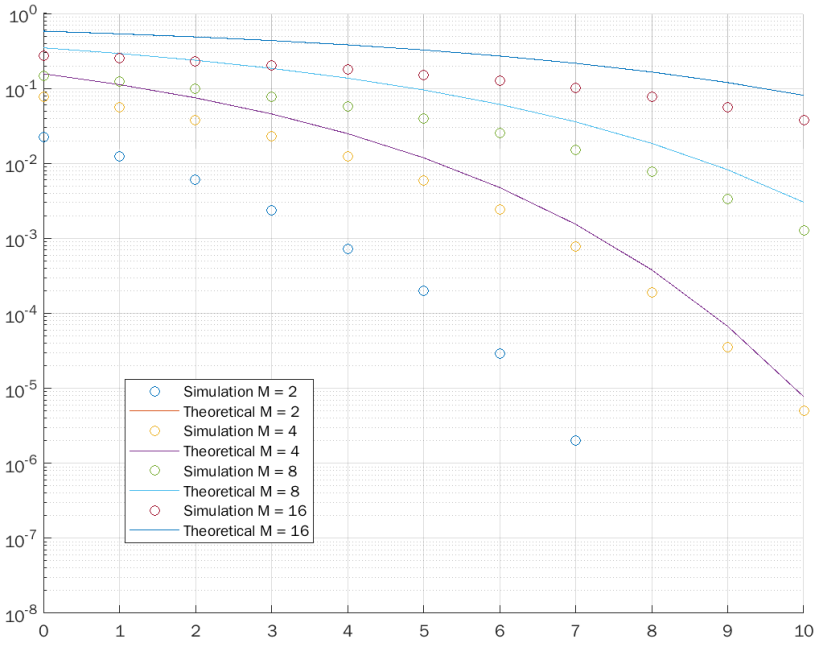


Figure 6: The simulation result of PSK

c)

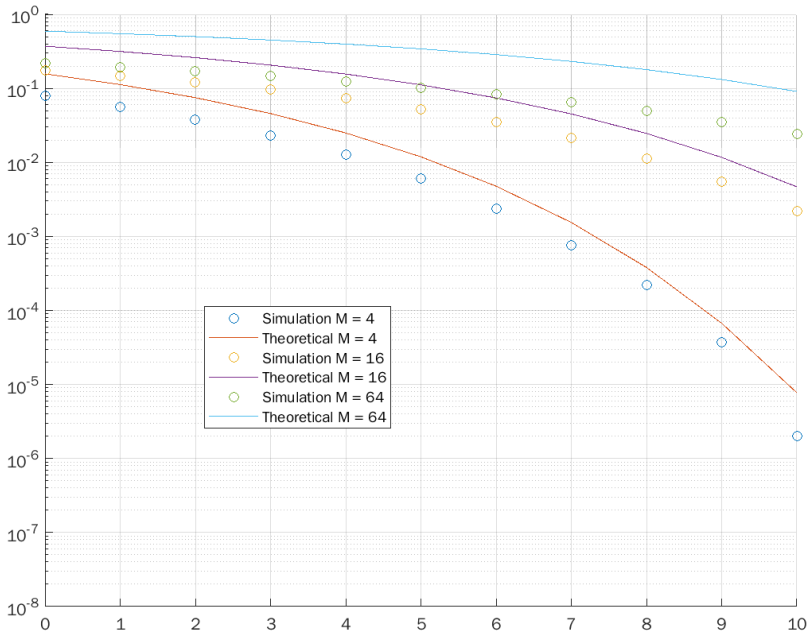


Figure 7: The simulation result of QAM

d) From the result of 3a) to 3c), the larger of E_b/N_o the smaller of SER. The simulation results are not as expected as the result of theoretical, but the gradient of 2 results is similarly.

Appendix

Code

```
close all; clear; clc;
% sym_seq = symbol_mapper([0 0 0 1 1 1 10], 4, 2, 'PAM')
% sym_seq = symbol_mapper([0 0 1 0], 4, 2*sin(pi/4), 'PSK')
% sym_seq = symbol_mapper([0 0 1 0], 4, 2, 'QAM')

% M = 4;
% d = 1;
% x = randi([0 1],10^4,1);
% sym_qpsk = symbol_mapper(x, M, d, 'PSK');

%histogram2(real(sym_qpsk), imag(sym_qpsk),
'DisplayStyle','tile','ShowEmptyBins','on')
%xlabel('I-channel')
%ylabel('Q-channel')

% e_n = [0, 10, 20];
% SER = [];
%
% for i = 1:length(e_n)
%     N_o = 1/((10^(e_n(i)/10))*4*log2(M)*(sin(pi/M)^2));
%     sym_wgn = sym_qpsk;
%     wgn_real = 0+sqrt(N_o/2)*randn(length(sym_qpsk),1);
%     wgn_imag = 0+sqrt(N_o/2)*randn(length(sym_qpsk),1);
%     for j = 1:length(sym_qpsk)
%         sym_wgn(j) = sym_wgn(j) + complex(wgn_real(j), wgn_imag(j));
%     end
%
%     bin_seq = MD_symbol_demapper(sym_wgn, M, d, 'PSK');
%     error = 0;
%     for k = 1:length(bin_seq)
%         if bin_seq(k) ~= num2str(x(k))
%             error = error+1;
%         end
%     end
%     SER = [SER, error/length(bin_seq)];
% end
%
% display(SER)

name = 'QAM';
d = 2;
max_db = 10;
legendmat = {};
f = figure();
hold on

for i = 1:3
    M = 4^i;
    x = randi([0 1],i*10^6,1);
    sym_qpsk = symbol_mapper(x, M, d, name);
    SER = [];
    theoretical = [];
```

```
    for j = 0:max_db
%         N_o = (d^2 * (M^2-1))/ (10^(j/10) * 12 * log2(M)); % PAM
%         N_o = d^2/((10^(j/10))*4*log2(M)*(sin(pi/M))^2); % PSK
        N_o = (d^2)*(M-1)/( (10^(j/10)) * 6 * log2(M)); % QAM

        sym_wgn = sym_qpsk;
        wgn_real = sqrt(N_o/2)*randn(length(sym_qpsk),1);
        wgn_imag = sqrt(N_o/2)*randn(length(sym_qpsk),1);
        for k = 1:length(sym_qpsk)
            sym_wgn(k) = sym_wgn(k) + complex(wgn_real(k), wgn_imag(k));
        end
        bin_seq = MD_symbol_demapper(sym_wgn, M, d, name);

        bin_seq = str2num(reshape(bin_seq, [length(bin_seq), 1]));

        [num, ratio] = symerr(bin_seq, x);

        SER = [SER, ratio];
%         theo = sqrt((6*log2(M))*(10^(j/10))/(M^2 -1)); % PAM
%         theo = sqrt(2*log2(M)*(sin(pi/M))^2 * (10^(j/10))); % PSK
        theo = sqrt(3*log2(M)* (10^(j/10)) / (M - 1));
        theoretical = [theoretical, 2*qfunc(theo)];
    end
    display(SER);

    semilogy(0:max_db, SER, 'o');
    semilogy(0:max_db, theoretical);
    ylim([1E-8 1])
    l = sprintf('M = %d', M);
    set(gca, 'YScale', 'log');
    grid on
end
legend({'Simulation M = 4', 'Theoretical M = 4', 'Simulation M = 16',
'Theoretical M = 16', 'Simulation M = 64', 'Theoretical M = 64', 'Simulation
M = 16', 'Theoretical M = 16'});



function bin_seq = MD_symbol_demapper(sym_seq, M, d, name)
    if mod(log2(M), 1) ~= 0
        error 'Input M must be the power of 2'
    else
        bits_len = log2(M);
    end

    gray_code = dec2bin(0, bits_len);
    gray_code = generateGrayCode(gray_code, bits_len);
    gray_code = cellstr(reshape(gray_code, bits_len, [])');

            sym = generateSYM(M, d, name);

            bin_seq = [];
            for i = 1:length(sym_seq)
                eucli_dist = 10^3;
                hold_sym = gray_code{1};
                for j = 1:length(sym)
                    dist = norm(sym(j) - sym_seq(i))^2;
```

```
                    if dist < eucli_dist
                        eucli_dist = dist;
                        hold_sym = gray_code{j};
                    end
                end
                bin_seq = [bin_seq, hold_sym];
            end

        end

        function sym_seq = symbol_mapper(bin_seq, M, d, name)
            if mod(log2(M), 1) ~= 0
                error 'Input M must be the power of 2'
            else
                bits_len = log2(M);
            end

            if ~isa(bin_seq, 'char')
                bin_seq = sprintf('%d', bin_seq);
            end

            gray_code = dec2bin(0, bits_len);
            gray_code = generateGrayCode(gray_code, bits_len);
            gray_code = cellstr(reshape(gray_code, bits_len, [])');

            sym = generateSYM(M, d, name);
            sym_seq = [];

            while mod(length(bin_seq), bits_len) ~= 0
                bin_seq = [bin_seq, '0'];
            end

            bin_seq = cellstr(reshape(bin_seq, bits_len, [])');

            for i = 1:length(bin_seq)
                for j = 1:length(gray_code)
                    if strcmp(gray_code{j}, bin_seq{i})
                        sym_seq = [sym_seq, sym(j)];
                    end
                end
            end
        end

        function sym = generateSYM(M, d, name)
            sym = [];
            if name == "PAM"
                for m = 1:M
                    sym = [sym, (d/2)*M+d/2-d*m];
                end

            elseif name == "PSK"
                d = d/(2*sin(pi/(M)));
                for m = 1:M
                    s = complex(d*cos(2*pi*(m-1)/M), d*sin(2*pi*(m-1)/M));
                    sym = [sym, s];
                end

            elseif name == "QAM"
```

```
                row = floor(sqrt(M));
                col = M/row;
                for j = 1:col
                    ax = ((d/2)*col + d/2 - d*j);
                    for i = 1:row
                        ay = (d*i+1-d*row)*((-1)^(j+1));
                        s = complex(ax, ay);
                        sym = [sym, s];
                    end
                end
            end
        end

        function gray_code = generateGrayCode(start_bin, bits_len)
            for i = 1:2^bits_len-1
                pre_code = start_bin(end-(bits_len-1):end);
                if rem(i,2) == 0
                    for j=length(pre_code):-1:1
                        if pre_code(j) == '1'
                            if pre_code(j-1) == '0'
                                pre_code(j-1) = '1';
                                break
                            else
                                pre_code(j-1) = '0';
                                break
                            end
                        end
                    end
                elseif rem(i,2) ~= 0
                    if pre_code(end) == '0'
                        pre_code(end) = '1';
                    else
                        pre_code(end) = '0';
                    end
                end
                start_bin = [start_bin, pre_code];
            end

            gray_code = start_bin;
        end
```