*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.

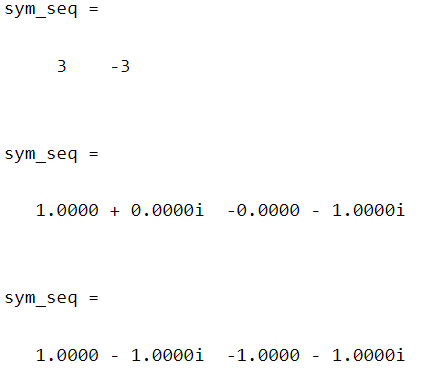


Figure 2: The output of the sample calling of figure1

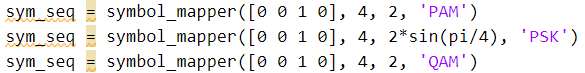


Figure 1: Calling the implement function of problem 1

1. a)

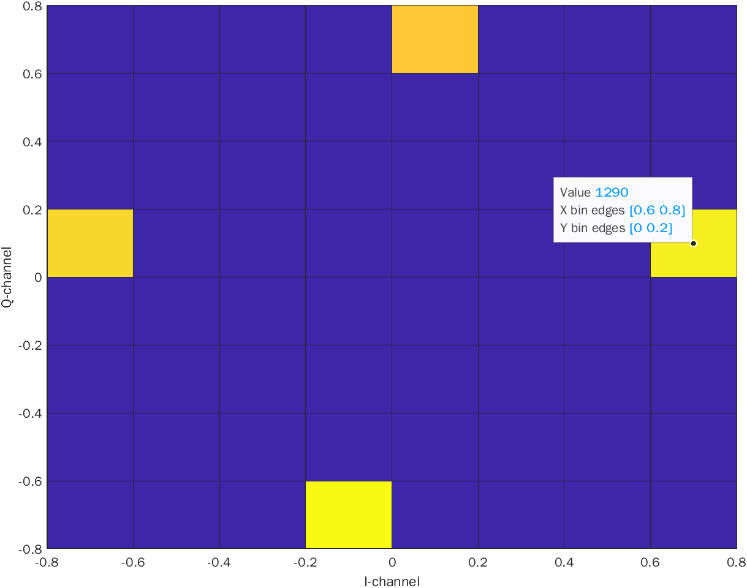


Figure 3: The histogram of QPSK

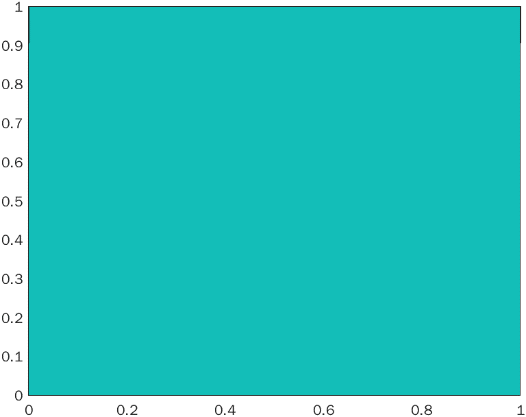


Figure 3 a): Eb/N0 = 0 dB

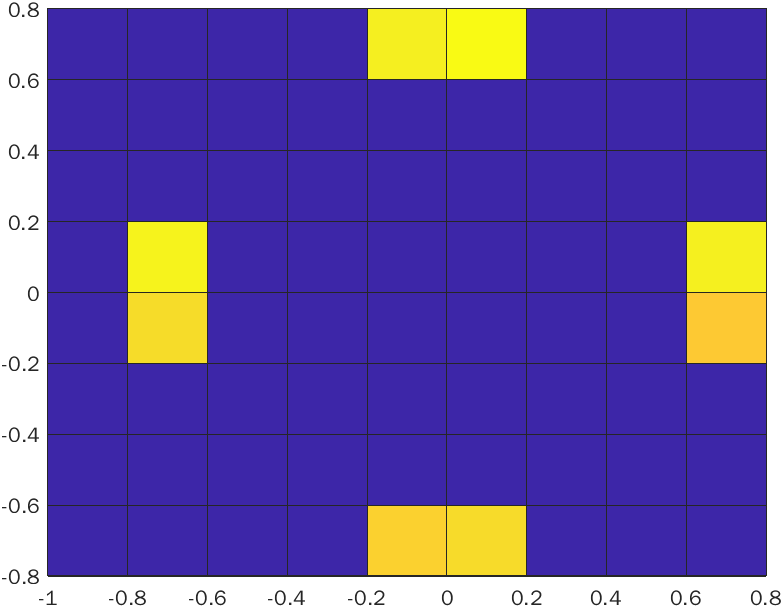


Figure 3 b): Eb/N0 = 10 dB

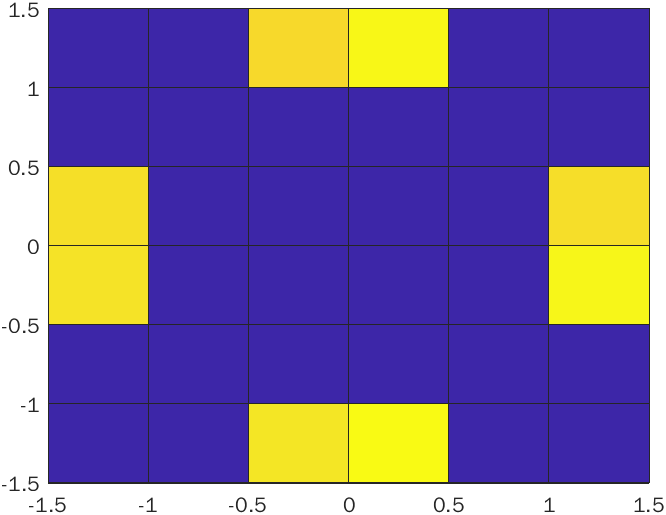
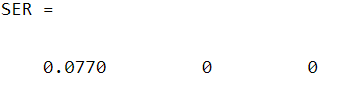


Figure 3 a): Eb/N0 = 20 dB

b) From the Figure 4 the Eb/N0 = 0 dB have almost 2% of SER and the rest of other 2 have 0% SER.

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



1. a)

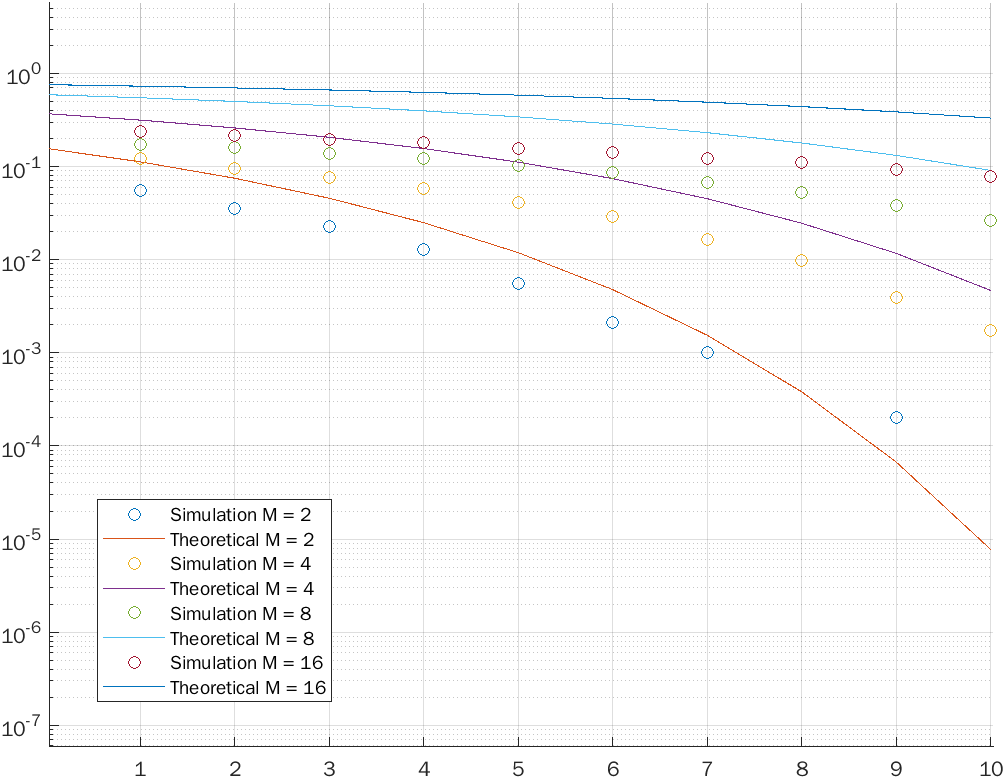


Figure 5: The simulation of PAM

b)

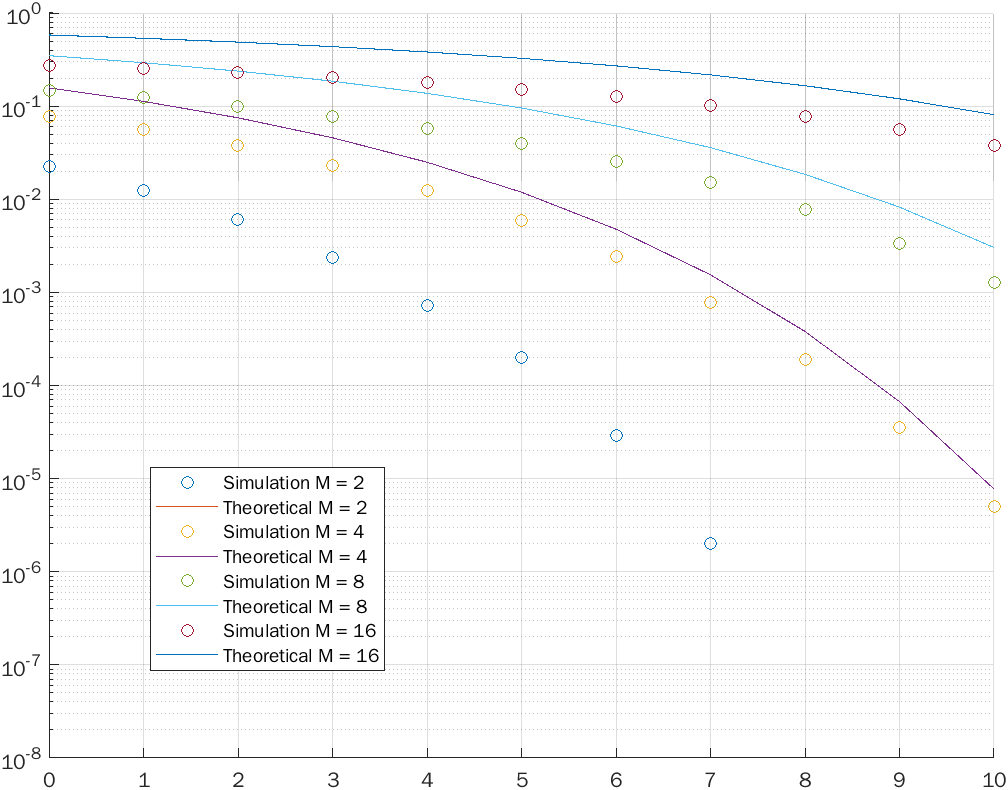


Figure 6: The simulation result of PSK

c)

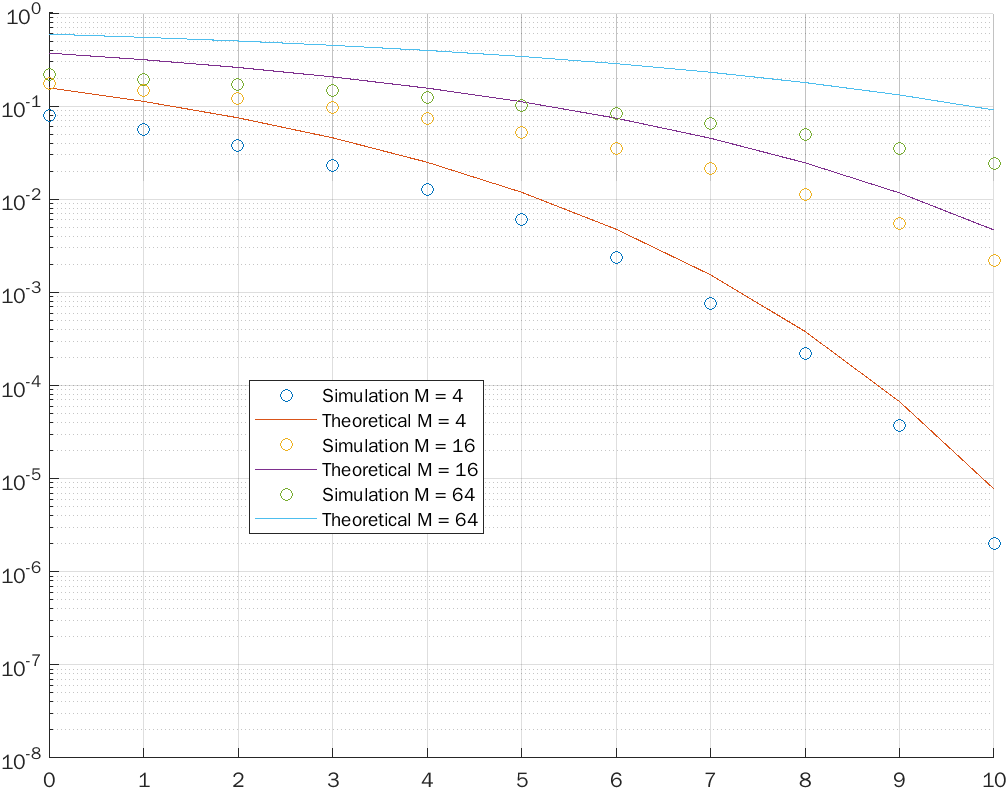


Figure 7: The simulation result of QAM

d) From the result of 3a) to 3c), the larger of Eb/No 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**

1. close all; clear; clc;
2. % sym\_seq = symbol\_mapper([0 0 0 1 1 1 10], 4, 2, 'PAM')
3. % sym\_seq = symbol\_mapper([0 0 1 0], 4, 2\*sin(pi/4), 'PSK')
4. % sym\_seq = symbol\_mapper([0 0 1 0], 4, 2, 'QAM')
5. % M = 4;
6. % d = 1;
7. % x = randi([0 1],10^4,1);
8. % sym\_qpsk = symbol\_mapper(x, M, d, 'PSK');
9. %histogram2(real(sym\_qpsk), imag(sym\_qpsk), 'DisplayStyle','tile','ShowEmptyBins','on')
10. %xlabel('I-channel')
11. %ylabel('Q-channel')
12. % e\_n = [0, 10, 20];
13. % SER = [];
14. %
15. % for i = 1:length(e\_n)
16. % N\_o = 1/((10^(e\_n(i)/10))\*4\*log2(M)\*(sin(pi/M)^2));
17. % sym\_wgn = sym\_qpsk;
18. % wgn\_real = 0+sqrt(N\_o/2)\*randn(length(sym\_qpsk),1);
19. % wgn\_imag = 0+sqrt(N\_o/2)\*randn(length(sym\_qpsk),1);
20. % for j = 1:length(sym\_qpsk)
21. % sym\_wgn(j) = sym\_wgn(j) + complex(wgn\_real(j), wgn\_imag(j));
22. % end
23. %
24. % bin\_seq = MD\_symbol\_demapper(sym\_wgn, M, d, 'PSK');
25. % error = 0;
26. % for k = 1:length(bin\_seq)
27. % if bin\_seq(k) ~= num2str(x(k))
28. % error = error+1;
29. % end
30. % end
31. % SER = [SER, error/length(bin\_seq)];
32. % end
33. %
34. % display(SER)
35. name = 'QAM';
36. d = 2;
37. max\_db = 10;
38. legendmat = {};
39. f = figure();
40. hold on
41. for i = 1:3
42. M = 4^i;
43. x = randi([0 1],i\*10^6,1);
44. sym\_qpsk = symbol\_mapper(x, M, d, name);
45. SER = [];
46. theoretical = [];
47. for j = 0:max\_db
48. % N\_o = (d^2 \* (M^2-1))/ (10^(j/10) \* 12 \* log2(M)); % PAM
49. % N\_o = d^2/((10^(j/10))\*4\*log2(M)\*(sin(pi/M))^2); % PSK
50. N\_o = (d^2)\*(M-1)/( (10^(j/10)) \* 6 \* log2(M)); % QAM
51. sym\_wgn = sym\_qpsk;
52. wgn\_real = sqrt(N\_o/2)\*randn(length(sym\_qpsk),1);
53. wgn\_imag = sqrt(N\_o/2)\*randn(length(sym\_qpsk),1);
54. for k = 1:length(sym\_qpsk)
55. sym\_wgn(k) = sym\_wgn(k) + complex(wgn\_real(k), wgn\_imag(k));
56. end
57. bin\_seq = MD\_symbol\_demapper(sym\_wgn, M, d, name);
58. bin\_seq = str2num(reshape(bin\_seq, [length(bin\_seq), 1]));
59. [num, ratio] = symerr(bin\_seq, x);
60. SER = [SER, ratio];
61. % theo = sqrt((6\*log2(M))\*(10^(j/10))/(M^2 -1)); % PAM
62. % theo = sqrt(2\*log2(M)\*(sin(pi/M))^2 \* (10^(j/10))); % PSK
63. theo = sqrt(3\*log2(M)\* (10^(j/10)) / (M - 1));
64. theoretical = [theoretical, 2\*qfunc(theo)];
65. end
66. display(SER);
67. semilogy(0:max\_db, SER, 'o');
68. semilogy(0:max\_db, theoretical);
69. ylim([1E-8 1])
70. l = sprintf('M = %d', M);
71. set(gca, 'YScale', 'log');
72. grid on
73. end
74. 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'});
75. function bin\_seq = MD\_symbol\_demapper(sym\_seq, M, d, name)
76. if mod(log2(M), 1) ~= 0
77. error 'Input M must be the power of 2'
78. else
79. bits\_len = log2(M);
80. end
81. gray\_code = dec2bin(0, bits\_len);
82. gray\_code = generateGrayCode(gray\_code, bits\_len);
83. gray\_code = cellstr(reshape(gray\_code, bits\_len, [])');
84. sym = generateSYM(M, d, name);
85. bin\_seq = [];
86. for i = 1:length(sym\_seq)
87. eucli\_dist = 10^3;
88. hold\_sym = gray\_code{1};
89. for j = 1:length(sym)
90. dist = norm(sym(j) - sym\_seq(i))^2;
91. if dist < eucli\_dist
92. eucli\_dist = dist;
93. hold\_sym = gray\_code{j};
94. end
95. end
96. bin\_seq = [bin\_seq, hold\_sym];
97. end
98. end
99. function sym\_seq = symbol\_mapper(bin\_seq, M, d, name)
100. if mod(log2(M), 1) ~= 0
101. error 'Input M must be the power of 2'
102. else
103. bits\_len = log2(M);
104. end
105. if ~isa(bin\_seq, 'char')
106. bin\_seq = sprintf('%d', bin\_seq);
107. end
108. gray\_code = dec2bin(0, bits\_len);
109. gray\_code = generateGrayCode(gray\_code, bits\_len);
110. gray\_code = cellstr(reshape(gray\_code, bits\_len, [])');
111. sym = generateSYM(M, d, name);
112. sym\_seq = [];
113. while mod(length(bin\_seq), bits\_len) ~= 0
114. bin\_seq = [bin\_seq, '0'];
115. end
116. bin\_seq = cellstr(reshape(bin\_seq, bits\_len, [])');
117. for i = 1:length(bin\_seq)
118. for j = 1:length(gray\_code)
119. if strcmp(gray\_code{j}, bin\_seq{i})
120. sym\_seq = [sym\_seq, sym(j)];
121. end
122. end
123. end
124. end
125. function sym = generateSYM(M, d, name)
126. sym = [];
127. if name == "PAM"
128. for m = 1:M
129. sym = [sym, (d/2)\*M+d/2-d\*m];
130. end
131. elseif name == "PSK"
132. d = d/(2\*sin(pi/(M)));
133. for m = 1:M
134. s = complex(d\*cos(2\*pi\*(m-1)/M), d\*sin(2\*pi\*(m-1)/M));
135. sym = [sym, s];
136. end
137. elseif name == "QAM"
138. row = floor(sqrt(M));
139. col = M/row;
140. for j = 1:col
141. ax = ((d/2)\*col + d/2 - d\*j);
142. for i = 1:row
143. ay = (d\*i+1-d\*row)\*((-1)^(j+1));
144. s = complex(ax, ay);
145. sym = [sym, s];
146. end
147. end
148. end
149. end
150. function gray\_code = generateGrayCode(start\_bin, bits\_len)
151. for i = 1:2^bits\_len-1
152. pre\_code = start\_bin(end-(bits\_len-1):end);
153. if rem(i,2) == 0
154. for j=length(pre\_code):-1:1
155. if pre\_code(j) == '1'
156. if pre\_code(j-1) == '0'
157. pre\_code(j-1) = '1';
158. break
159. else
160. pre\_code(j-1) = '0';
161. break
162. end
163. end
164. end
165. elseif rem(i,2) ~= 0
166. if pre\_code(end) == '0'
167. pre\_code(end) = '1';
168. else
169. pre\_code(end) = '0';
170. end
171. end
172. start\_bin = [start\_bin, pre\_code];
173. end
174. gray\_code = start\_bin;
175. end