

ASSIGNMENT 5 REPORT

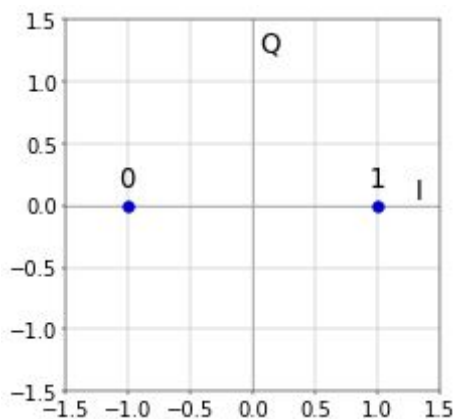
-JAYANT DUNEJA
2018102003

QUESTION-1:

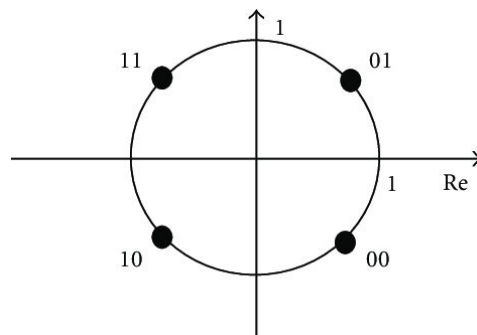
We generated the random bit by using the `randsrc()` function in Matlab. Both the probabilities were defined to be 0.5. This function returns a single bit and not a vector.

QUESTION-2:

1. **bpskmap():** This maps the given bits to -1,1. If the value of the bit is 0, then the value of the symbol is -1 and if the value of the bit is 1 then the value of the symbol is 1. I have written the given program using if statements nested inside a for loop.



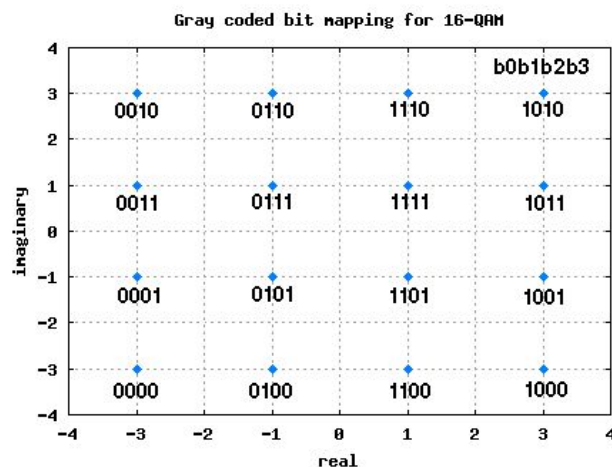
2. **qpskmap():** This function maps a sequence of 2 bits to 4 complex numbers. This was also implemented using the same technique, if statements nested in a for loop. This function takes its input as 2 vectors and the output is a single vector which contains the complex symbols.



3.fourpammap(): This function maps a sequence of 2 bits to 4 real values. This function takes 2 vectors as input and returns 1 output vector. The technique used is the same as described earlier.

Bit Sequence	Value
00	-3
01	-1
11	1
10	3

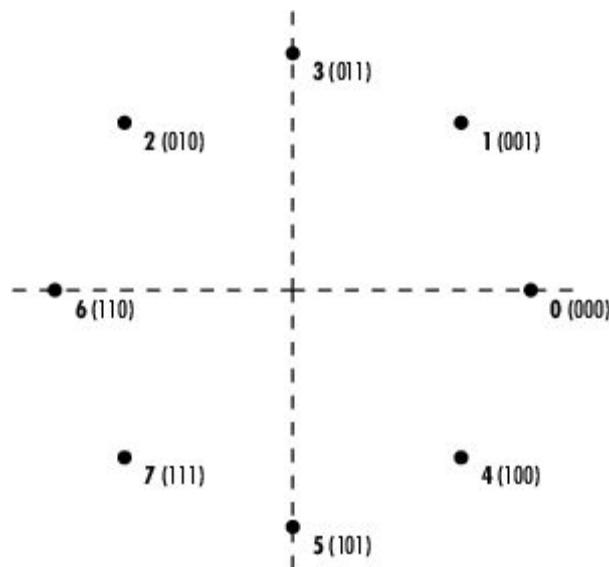
4.sixteenqammap(): This function maps a sequence of 4 bits to 16 complex numbers. This was also implemented using the same technique, if statements nested in a for loop.



This function takes its input as 4 vectors and the output is a single vector which contains the complex symbols.

5.eightpskmap(): This function maps a sequence of 3 bits to 8 complex numbers. This was also implemented using the same technique, if statements nested in a for loop. This function takes its input as 3 vectors and the output is a single vector which contains the complex symbols. The values taken are of the form

$$\exp(j \cdot 2 \cdot \pi \cdot i / 8), i = 0, 1, \dots, 7.$$



QUESTION-3:

- In this question firstly we define the Square Root Nyquist Pulse as our Transmitter and Receiver filter using the `rcosdesign()` inbuilt function in Matlab.
- Then 12000 random bits were generated and then these bits were mapped to 12000 symbols based on the qpsk mapping.
- These symbols were upsampled using the code snippet which was present in the textbook. Below is a paragraph which explains what upsampling is doing:

The symbols come in every T seconds, while the samples of the transmit filter are spaced by T/m . For example, the n th symbol contributes $b[n]p(t - nT)$ to the transmit filter output, and the $(n + 1)$ st symbol contributes $b[n + 1]p(t - (n + 1)T)$. Since $p(t - nT)$ and $p(t - (n + 1)T)$ are offset by T , they must be offset by m samples when sampling at a rate of m/T . Thus, if the symbols are input to a transmit filter whose discrete time impulse response is expressed at sampling rate m/T , then successive symbols at the input to the filter must be spaced by m samples. That is, in order to get the output as a convolution of the symbols with the transmit filter expressed at rate

m/T . We must insert $m - 1$ zeros between successive symbols to convert them to a sampling rate of m/T .

- After this the upsampled symbols are passed through the input transmitter filter and then passed through the receiver filter using the `conv()` function in Matlab.

QUESTION-4:

- In this question, the exact procedure of the previous question is followed and the only difference is that we have added Noise in this question.
- The noise being added is AWGN gaussian noise with mean=0 and standard deviation=sigma(sigma is calculated in the next question).
- The noise added is complex and both the real and imaginary parts are gaussian random variables.
- We have used the `Normrnd()` inbuilt function of Matlab to generate the noise.
- These random variables were added to the signal generated after the transmitter filter and then the sum of the noise and the original symbols were passed through the receiver filter.
- I have attached a Code snippet of the process I just explained.

```

symbols_upsampled1(1:m:nsymbols_upsampled1)-symbols1,%insert symbols with spacing m
|
transmit_filter = rcosdesign(a,nsymbols_upsampled1,m,'sqrt'); %Generating square root

%PASSING ORIGINAL SIGNAL THROUGH THE TRANSMITTER FILTER
tx_output=conv(symbols_upsampled1,transmit_filter);

%Gaussian noise generation
symbols= normrnd(0,sigma,size(tx_output)) + 1i*normrnd(0,sigma,size(tx_output));

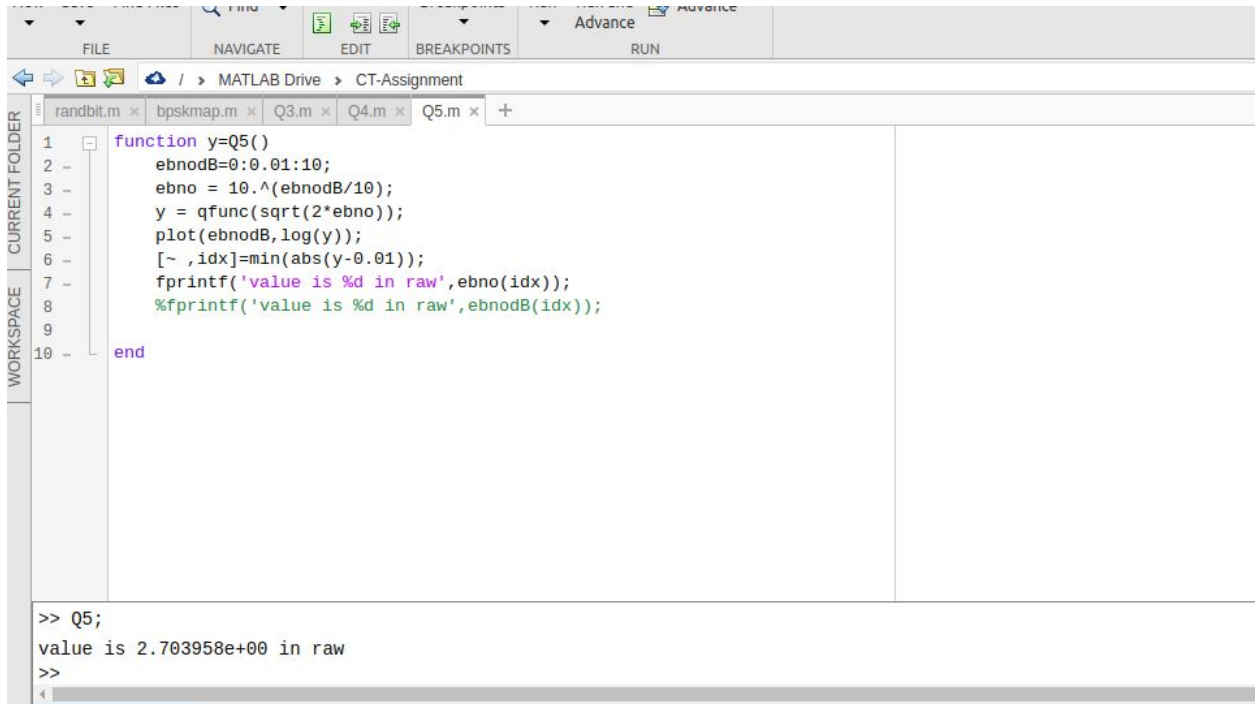
temp1=symbols + tx_output;
y=conv(temp1,transmit_filter);

```

AMAND WINDOW

QUESTION-5:

- In this question we compute the ideal error probability for BPSK which is given by $Q(\sqrt{2 \cdot E_b/N_0})$.
- We have used the inbuilt `qfunc` to generate this value.
- Next we had to determine the value of E_b/N_0 for which the value of probability was 0.01.
- As we were not sure whether the probability would take this given value, I minimised `abs(prob - 0.01)` using the `min()` function in Matlab and I got the value from there. A graph was plotted between `log(prob)` and E_b/N_0 and the graph would be attached in the folder. Attached below is the code snippet and the output I got from this program.



The image shows a MATLAB IDE window with the following components:

- Top Toolbar:** FILE, NAVIGATE, EDIT, BREAKPOINTS, RUN, and a dropdown menu with 'Advance' and 'Advance'.
- Current Folder:** MATLAB Drive > CT-Assignment
- Workspace:** randbit.m, bpskmap.m, Q3.m, Q4.m, Q5.m
- Code Editor:**

```
1 function y=Q5()
2     ebnodB=0:0.01:10;
3     ebno = 10.^(ebnodB/10);
4     y = qfunc(sqrt(2*ebno));
5     plot(ebnodB,log(y));
6     [~,idx]=min(abs(y-0.01));
7     fprintf('value is %d in row',ebno(idx));
8     %fprintf('value is %d in row',ebnodB(idx));
9
10 end
```
- Command Window:**

```
>> Q5;
value is 2.703958e+00 in row
>>
```

QUESTION-6:

- In this question we perform downsampling of the symbols we got after the receiver and the symbols we got before the receiver.
- We use the following formula for downsampling the symbols after the receiver(a in the given code snippet) and symbols before the receiver(b in the code snippet) respectively.
- Also in this question we calculated the value of sigma by using the fact that for BPSK $E_b=1$. Hence $\sigma=\sqrt{\frac{1}{2}*ebno}$ where ebno is the parameter I am passing in the functions.

```

3
4 - nymbols=12000;
5 - a=zeros(1,nymbols);
6 - m=4; %upsampling factor
7 - for i=1:nymbols
8 -     a(i)=out(nsym_up*m +1 + (m*(i-1))); % a(i) are the
9 - end
10 - b=zeros(1,nymbols);
11 - for i=1:nymbols
12 -     b(i)=in(nsym_up*m/2 +1 + (m*(i-1))); %b(i) are the
13 - end
14 - scatter(real(a),imag(a));
15 %scatter(real(b),imag(b));
16
17 - end

```

QUESTION-7:

- In this question we made the decisions to decode the received symbols and then we calculated the error probability and compared that prob to the ideal value.
- We also calculated the error prob for the symbols before the receiver filter and that was significantly larger than the other prob, and the result was as we had predicted.
- The reason for this is that the ISI in the decision statistics.

The image shows a MATLAB code editor with the following code snippet:

```

1 function y=Q7(ebno) %ebno is the value we calculated in Q5
2
3 [reciever_out,reciever_in,original]=Q6(ebno);
4 nsymbols=12000;
5 for i=1:length(reciever_out)
6     if(real(reciever_out(i)) > 0)
7         reciever_out(i)=1;
8     else
9         reciever_out(i)=-1;
10    end
11 end
12 check1=reciever_out-original;
13 val1=nnz(check1);
14 ans1=val1/nsymbols;
15 fprintf('error percentage after reciver_filter %d \n',ans1);
16
17
18 for i=1:length(reciever_in)
19     if(abs(real(reciever_in(i))-1) >= abs(real(reciever_in(i))+1)) % Mapping the values by using Map rule for

```

The command window shows the following output:

```

error percentage after reciver_filter 1.075000e-02
error percentage before reciver_filter 9.700000e-02
>> |

```

- Above is a code snippet and answer for the values of the prob we got.
- We can also observe the decision rule that I have used for this question in the code snippet. I have used ML decoding rule.

QUESTION-8:

- This question was very similar to Question 5, the only difference being that we have used a different expression for calculating the error prob.
- The expression this time was: $\text{prob} = Q(\sqrt{4 \cdot \text{ebno}/5})$
- We used the same technique as Question 5 to find the value of E_b/N_0 .
- Given below is the code snippet and the result we obtained.

The image shows a MATLAB IDE window with the following components:

- Toolbar:** Includes buttons for 'New', 'Save', 'Find Files', 'Find', 'Breakpoints', 'Run', 'Run and Advance', and 'Advance'.
- File Explorer:** Shows the current directory as 'MATLAB Drive'.
- Editor:** Contains the following MATLAB code:

```
1 function y=Q8() %I will only be calculating the value of ideal value
2 %of eb/no in this code and will create the random
3 %signal in the next Question only
4 ebnoB=0:0.01:10;
5 ebno = 10.^(ebnoB/10); |
6 y = qfunc(sqrt(4*ebno/5));
7 plot(ebnoB,log(y));
8 [~,idx]=min(abs(y-0.01));
9 fprintf('value is %d',ebno(idx));
10 %fprintf('value is %d',ebnoB(idx));
11
12
13
14 end
```
- Command Window:** Shows the execution output:

```
>> Q8;
value is 6.760830e+00
>>
```

QUESTION-9:

- For this question I have combined the codes of the questions 4,6,7 and made the following changes:
 - ❖ Nysmbols are now 6000
 - ❖ The mapping function has changed and now fourpammap() was called for the mapping
 - ❖ The way in which the value of sigma was calculated was changed. In this code $\sigma = \sqrt{(5/(4 \cdot ebno))}$; We used this formula because we got $E_b = 5/2$ for the 4-Pam constellation.
 - ❖ Decision rule also changed and the below code snippet shows the decision rule and the value of prob we got.

```

ew Save Find Files Find Breakpoints Run Run and Advance Advance
FILE NAVIGATE EDIT BREAKPOINTS RUN
/ > MATLAB Drive > CT-Assignment
randbit.m x bpskmap.m x Q3.m x Q4.m x Q5.m x Q12.m x Q10.m x Q6.m x Q7.m x Q8.m x Q9.m x +
42 %RULE.
43 reciever_out=a;
44 for i=1:nsymbols
45     if(real(reciever_out(i))>=2 )
46         reciever_out(i)=3;
47     elseif(real(reciever_out(i))<2 && real(reciever_out(i)) >=0 )
48         reciever_out(i)=1;
49     elseif((real(reciever_out(i))<0 && real(reciever_out(i)) >= -2 ))
50         reciever_out(i)=-1;
51     else
52         reciever_out(i)=-3;
53     end
54 end
55 original=symbols1;
56 check1=reciever_out-original;
57 val1=nnz(check1);
58 ans1=val1/nsymbols;
59 fprintf('error percentage after reciver_filter %d \n',ans1);
60
COMMAND WINDOW
>> Q9(6.7608);
error percentage after reciver_filter 1.700000e-02
>>

```

QUESTION -9 : (QPSK)

- The code for this was the same as Question 9 and only the mapping function and decision rule were changed.
- This time the qpskmap() function was called.

```

ew Save Find Files Find Breakpoints Run Run and Advance Advance
FILE NAVIGATE EDIT BREAKPOINTS RUN
/ > MATLAB Drive > CT-Assignment
randbit.m x bpskmap.m x Q3.m x Q4.m x Q5.m x Q12.m x Q10.m x Q6.m x Q7.m x Q8.m x Q9.m x Q9qpsk.m x +
44 distance=zeros(1,4);
45 for i=1:nsymbols
46     distance(1)=reciever_out(i)-(1+1i);
47     distance(2)=reciever_out(i)-(-1+1i);
48     distance(3)=reciever_out(i)-(-1-1i);
49     distance(4)=reciever_out(i)-(1-1i);
50     distance=abs(distance);
51     [~,idx]=min(distance);
52     if(idx==1)
53         reciever_out(i)=1+1i;
54     elseif(idx==2)
55         reciever_out(i)= -1+1i;
56     elseif(idx==3)
57         reciever_out(i)=-1-1i;
58     elseif(idx==4)
59         reciever_out(i)=1-1i;
60     end
61 end
62
COMMAND WINDOW
>> Q9qpsk(6.7608);
error percentage after reciver_filter 1.733333e-02
>>

```

- Above is the code snippet of the decision rule and the value for the probability we got.

QUESTION-10:

- This code is similar to Question 9 with the following changes:
 - ❖ Nysmbols are now 3000
 - ❖ The mapping function has changed and now sixteenqammap() was called for the mapping
 - ❖ The way in which the value of sigma was calculated was changed. In this code $\sigma = \sqrt{(15/(16 \cdot E_b))}$; We used this formula because we got $E_b = 15/8$ for the 16-Qam constellation.
 - ❖ Decision rule also changed and the below code snippet shows the decision rule and the value of prob we got.

```

CURRENT FOLDER
WORKSPACE
50 - distance(1)=reciever_out(i)-(-3-3i);
51 - distance(2)=reciever_out(i)-(-3-1i);
52 - distance(3)=reciever_out(i)-(-3+1i);
53 - distance(4)=reciever_out(i)-(-3+3i);
54
55 - distance(5)=reciever_out(i)-(-1+3i);
56 - distance(6)=reciever_out(i)-(-1+1i);
57 - distance(7)=reciever_out(i)-(-1-1i);
58 - distance(8)=reciever_out(i)-(-1-3i);
59
60 - distance(9)=reciever_out(i)-(1-3i);
61 - distance(10)=reciever_out(i)-(1-1i);
62 - distance(11)=reciever_out(i)-(1+1i);
63 - distance(12)=reciever_out(i)-(1+3i);
64
65 - distance(13)=reciever_out(i)-(3+3i);
66 - distance(14)=reciever_out(i)-(3+1i);
67 - distance(15)=reciever_out(i)-(3-1i);
68 - distance(16)=reciever_out(i)-(3-3i);

COMMAND WINDOW
>> Q10(6.7608);
error percentage after reciver_filter 1.266667e-02
>>

```

FILE NAVIGATE EDIT BREAKPOINTS RUN

/ > MATLAB Drive > CT-Assignment

randbit.m × bpskmap.m × Q3.m × Q4.m × Q5.m × Q12.m × Q10.m × Q6.m × Q7.m × Q8.m × Q9.m × Q9qpsk.m × +

```

71 - [~,idx]=min(distance);
72 - if(idx==1)
73 -     reciever_out(i)= -3-3i;
74 - elseif(idx==2)
75 -     reciever_out(i)= -3-1i;
76 - elseif(idx==3)
77 -     reciever_out(i)= -3+1i;
78 - elseif(idx==4)
79 -     reciever_out(i)= -3+3i;
80 -
81 -     elseif(idx==5)
82 -         reciever_out(i)= -1+3i;
83 -     elseif(idx==6)
84 -         reciever_out(i)= -1+1i;
85 -     elseif(idx==7)
86 -         reciever_out(i)= -1-1i;
87 -     elseif(idx==8)
88 -         reciever_out(i)= -1-3i;
89 -

```

COMMAND WINDOW

```

>> Q10(6.7608);
error percentage after reciver_filter 1.266667e-02
>>

```

New Save Find Files Find Breakpoints Run Run and Advance Advance

FILE NAVIGATE EDIT BREAKPOINTS RUN

/ > MATLAB Drive > CT-Assignment

randbit.m × bpskmap.m × Q3.m × Q4.m × Q5.m × Q12.m × Q10.m × Q6.m × Q7.m × Q8.m × Q9.m × Q9qpsk.m × +

```

90 - elseif(idx==9)
91 -     reciever_out(i)= 1-3i;
92 - elseif(idx==10)
93 -     reciever_out(i)= 1-1i;
94 - elseif(idx==11)
95 -     reciever_out(i)= 1+1i;
96 - elseif(idx==12)
97 -     reciever_out(i)= 1+3i;
98 -
99 -     elseif(idx==13)
100 -         reciever_out(i)= 3+3i;
101 -     elseif(idx==14)
102 -         reciever_out(i)= 3+1i;
103 -     elseif(idx==15)
104 -         reciever_out(i)= 3-1i;
105 -     elseif(idx==16)
106 -         reciever_out(i)= 3-3i;
107 -
108 -

```

COMMAND WINDOW

```

>> Q10(6.7608);
error percentage after reciver_filter 1.266667e-02
>>

```

QUESTION-11:

- This code is similar to Question 9 with the following changes:

- ❖ Nysmbols are now 4000
- ❖ The mapping function has changed and now EightpskMapping() was called for the mapping
- ❖ The way in which the value of sigma was calculated was changed. In this code $\sigma = \sigma = \sqrt{(1/(6 \cdot E_b N_0))}$;

We used this formula because we got $E_b = 1/3$ for the 8-psk constellation.

- ❖ Decision rule also changed and the below code snippet shows the decision rule and the value of prob we got.

The image shows a MATLAB IDE window with the following code in the editor:

```

48 distance=zeros(1,8);
49 for i=1:nsymbols
50     distance(1)=reciever_out(i)-exp(0*1i*pi/4);
51     distance(2)=reciever_out(i)-exp(1*1i*pi/4);
52     distance(3)=reciever_out(i)-exp(3*1i*pi/4);
53     distance(4)=reciever_out(i)-exp(2*1i*pi/4);
54     distance(5)=reciever_out(i)-exp(7*1i*pi/4);
55     distance(6)=reciever_out(i)-exp(6*1i*pi/4);
56     distance(7)=reciever_out(i)-exp(4*1i*pi/4);
57     distance(8)=reciever_out(i)-exp(5*1i*pi/4);
58     distance=abs(distance);
59     [~,idx]=min(distance);
60     if(idx==1)
61         reciever_out(i)= exp(0*1i*pi/4);
62     elseif(idx==2)
63         reciever_out(i)= exp(1*1i*pi/4);
64     elseif(idx==3)
65         reciever_out(i)= exp(3*1i*pi/4);
66     elseif(idx==4)

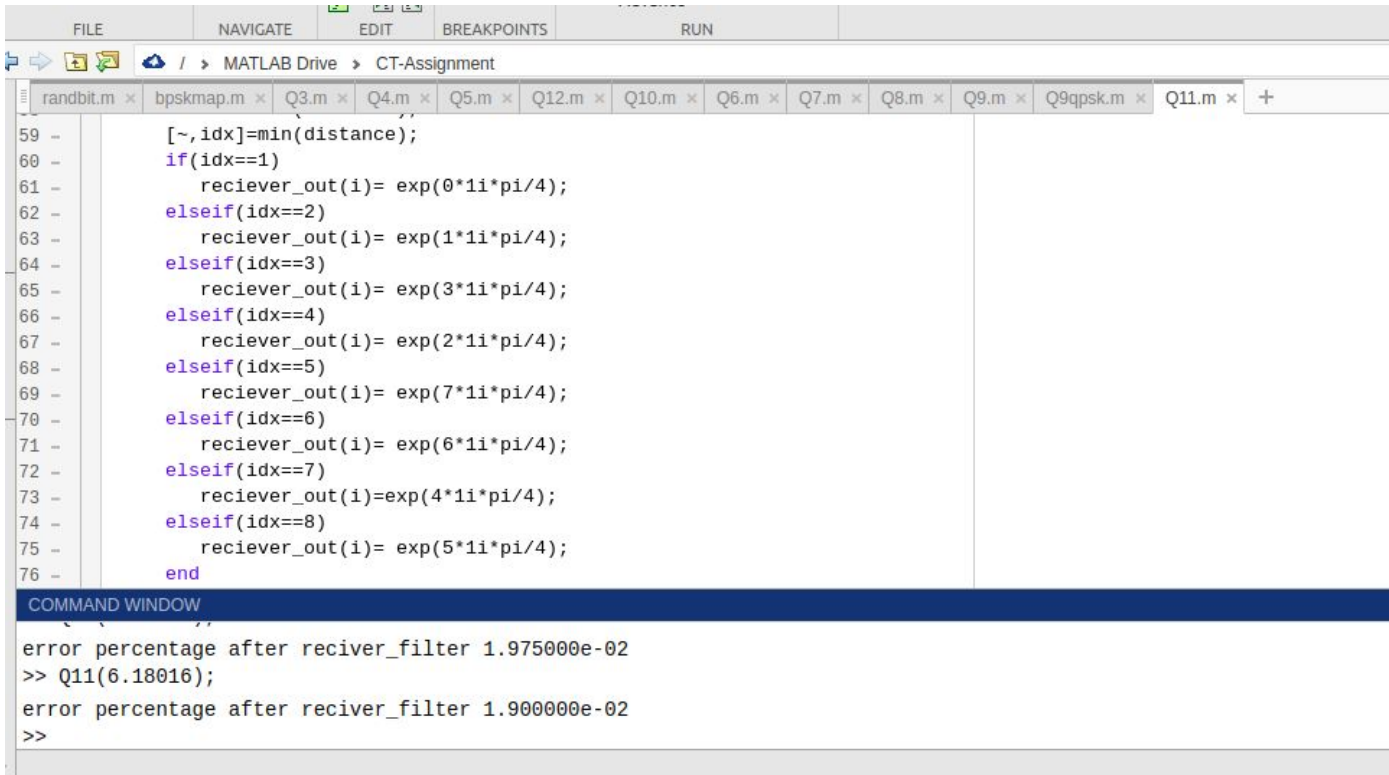
```

The COMMAND WINDOW shows the following output:

```

error percentage after reciver_filter 1.975000e-02
>> Q11(6.18016);
error percentage after reciver_filter 1.900000e-02
>>

```



The image shows a MATLAB IDE window with a script editor and a command window. The script editor contains a loop that iterates over 11 samples (Q1 to Q11) and demodulates each sample based on its index. The command window shows the results of the demodulation, indicating that the error percentage after the receiver filter is very low, approximately 1.975e-02 for Q11 and 1.9e-02 for the other samples.

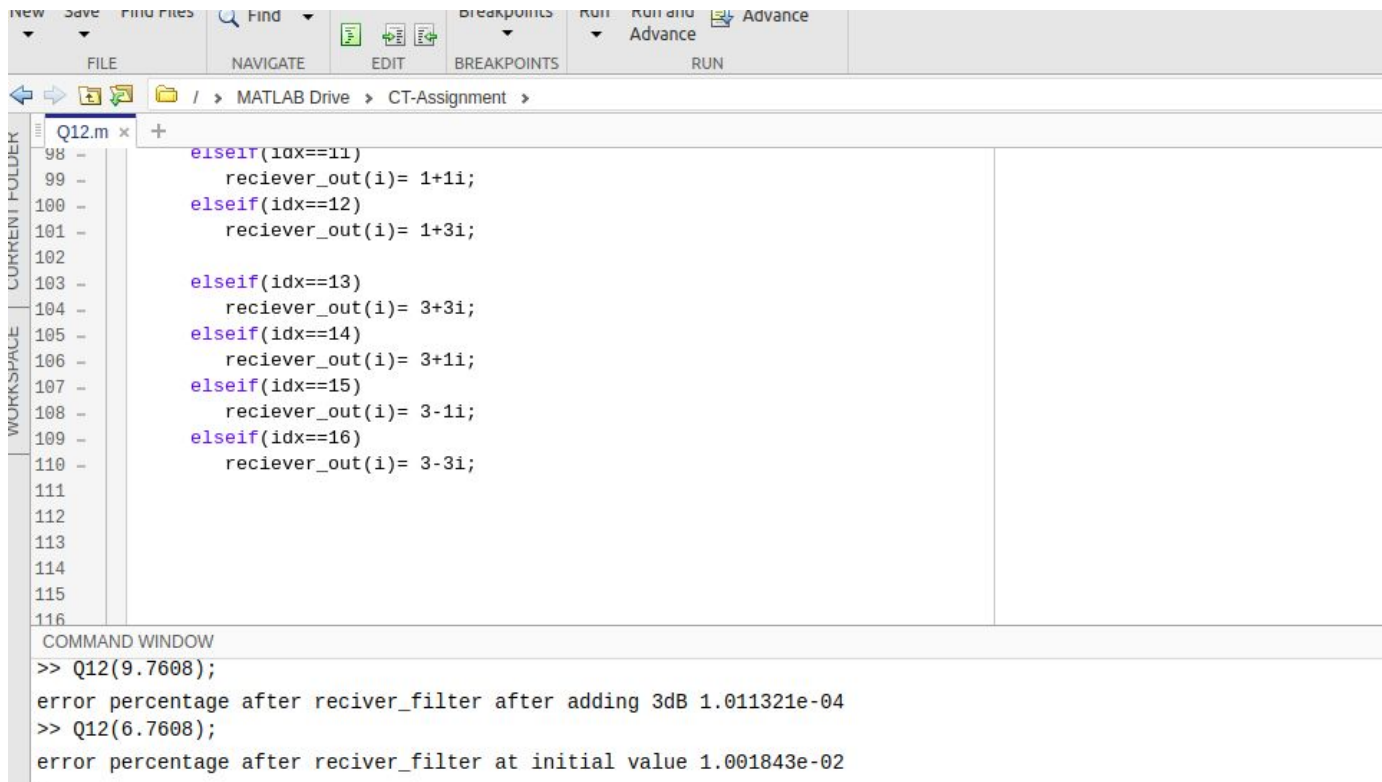
```
59 - [~,idx]=min(distance);
60 - if(idx==1)
61 -     reciever_out(i)= exp(0*1i*pi/4);
62 - elseif(idx==2)
63 -     reciever_out(i)= exp(1*1i*pi/4);
64 - elseif(idx==3)
65 -     reciever_out(i)= exp(3*1i*pi/4);
66 - elseif(idx==4)
67 -     reciever_out(i)= exp(2*1i*pi/4);
68 - elseif(idx==5)
69 -     reciever_out(i)= exp(7*1i*pi/4);
70 - elseif(idx==6)
71 -     reciever_out(i)= exp(6*1i*pi/4);
72 - elseif(idx==7)
73 -     reciever_out(i)=exp(4*1i*pi/4);
74 - elseif(idx==8)
75 -     reciever_out(i)= exp(5*1i*pi/4);
76 - end
```

COMMAND WINDOW

```
error percentage after reciver_filter 1.975000e-02
>> Q11(6.18016);
error percentage after reciver_filter 1.900000e-02
>>
```

QUESTION 12:

- In this question we have added the noise before we have convoluted with the transmitter filter.
- This helps in removing the noise and we see that the probability of error becomes very close to ideal after this.
- After this, we calculated the probability after adding 3dB to the earlier value of E_b/n_0 .
- The result was as expected and the probability of error dropped to approximately 0.0001.
- Also after making the scatter plots, we find out that the constellation comes closer when we run the code on the 3dB excess value.
- You can see the results in the below code snippet:



The image shows a MATLAB IDE window. The top menu bar includes 'New', 'Save', 'Find Files', 'Find', 'Breakpoints', 'Run', 'Run and Advance', and 'Advance'. Below the menu bar is a toolbar with icons for file operations, navigation, editing, breakpoints, and running. The current file is 'Q12.m' located at 'MATLAB Drive > CT-Assignment'. The script content is as follows:

```
98 - elseif(idx==11)
99 -     reciever_out(i)= 1+1i;
100 - elseif(idx==12)
101 -     reciever_out(i)= 1+3i;
102 -
103 - elseif(idx==13)
104 -     reciever_out(i)= 3+3i;
105 - elseif(idx==14)
106 -     reciever_out(i)= 3+1i;
107 - elseif(idx==15)
108 -     reciever_out(i)= 3-1i;
109 - elseif(idx==16)
110 -     reciever_out(i)= 3-3i;
111 -
112 -
113 -
114 -
115 -
116 -
```

The Command Window at the bottom shows the following commands and outputs:

```
>> Q12(9.7608);
error percentage after reciever_filter after adding 3dB 1.011321e-04
>> Q12(6.7608);
error percentage after reciever_filter at initial value 1.001843e-02
```

THE END