Engineering Assignment Coversheet

THE UNIVERSITY OF MELBOURNE | MELBOURNE SCHOOL OF ENGINEERING

Please note that you:

- Must keep a full copy of your submission for this assignment
- Must staple this assignment
- Must NOT use binders or plastic folders except for large assignments

Student Number(s)

872301

755846

Group Code (if applicable):

5

Assignment Title:	Matlab Workshop 3
Subject Number:	ELEN90054
Subject Name:	Probability and random models
Student Name:	Yue Chang Hanqiu Zhang
Lecturer/Tutor:	Pasha Tolmachev
Due Date:	27/04/2018

For Late Assignments Only

Has an extension been granted? Yes / No (circle)

A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

Collusion

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorised collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct.

Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism: http://academichonesty.unimelb.edu.au/

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

STUDENT DECLARATION

Please sign below to indicate that you understand the following statements:

I declare that:

- This assignment is my own original work, except where I have appropriately cited the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

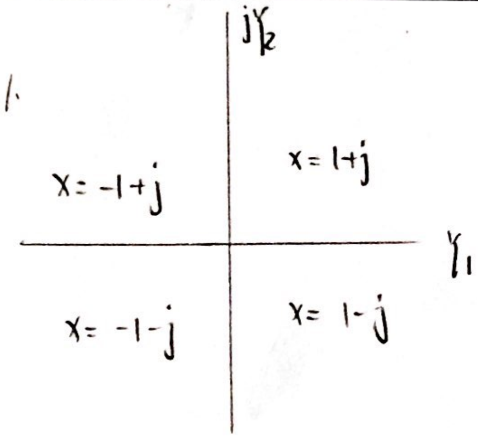
For the purposes of assessment, I give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).

Student signature Yue Chang Hanqiu Zhang Date 26/04/2018

PRM workshop 3

1.



2. let $v = R^2$

$R^2 \sim \exp(\lambda)$

$\therefore v \sim \exp(\lambda)$

$F_V(v) = 1 - e^{-\lambda v} \quad (v \geq 0)$

$F_R(r) = p(R \leq r) = p(\sqrt{V} \leq r) = p(V \leq r^2) = F_V(r^2) = 1 - e^{-\lambda r^2} \quad (r \geq 0)$

$$\therefore F_R(r) = \begin{cases} 1 - e^{-\lambda r^2} & (r \geq 0) \\ 0 & (r < 0) \end{cases}$$

$$f_R(r) = \frac{dF_R(r)}{dr} = \begin{cases} 2r\lambda e^{-\lambda r^2} & (r \geq 0) \\ 0 & (r < 0) \end{cases}$$

$\because R, \Theta$ are independent $\Theta \sim \text{Uniform}(0, 2\pi)$ $f_\Theta(\theta) = \begin{cases} \frac{1}{2\pi} & 0 \leq \theta \leq 2\pi \\ 0 & \text{elsewhere} \end{cases}$

$$f_{R\Theta}(r, \theta) = f_R(r) \cdot f_\Theta(\theta) = \begin{cases} \frac{r\lambda}{\pi} e^{-\lambda r^2} & r \geq 0 \text{ and } 0 \leq \theta \leq 2\pi \\ 0 & \text{elsewhere} \end{cases}$$

3. R, Θ independent

$E[N_1] = E[R\cos\theta] = E[R] \cdot E[\cos\theta]$ $\quad E[\cos\theta] = \int_0^{2\pi} \frac{1}{2\pi} \cos\theta \, d\theta = 0$ $\therefore \boxed{E[N_1] = 0}$

$E[N_1^2] = E[R^2 \cos^2\theta] = E[R^2] \cdot E[\cos^2\theta]$

$R^2 \sim \exp(\lambda)$ $\therefore E[R^2] = \frac{1}{\lambda}$

$E[\cos^2\theta] = E[\frac{1}{2}(1 + \cos 2\theta)] = E[\frac{1}{2}\cos 2\theta] + \frac{1}{2} = \int_0^{2\pi} \frac{1}{2\pi} \cdot \frac{1}{2}\cos 2\theta \, d\theta + \frac{1}{2} = 0 + \frac{1}{2} = \frac{1}{2}$

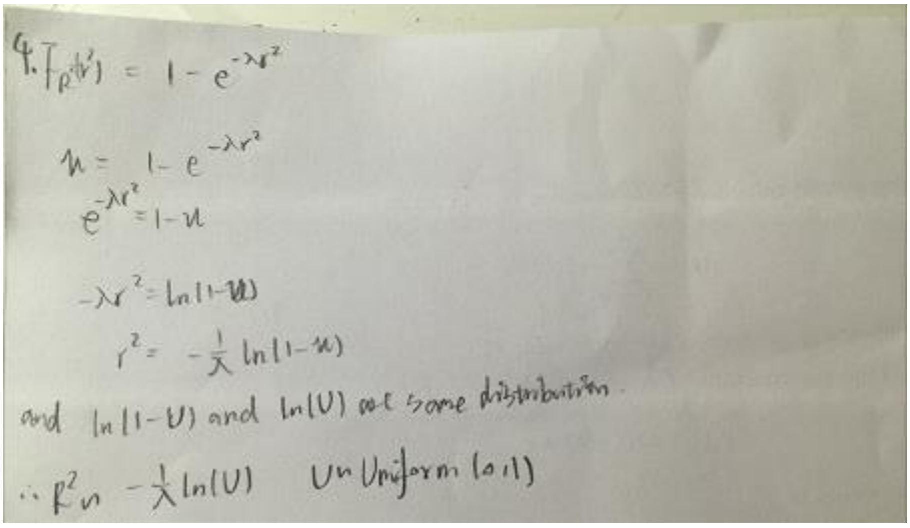
$\boxed{E[N_1^2] = \frac{1}{2\lambda}}$

$E[N_1 N_2] = E[R\cos\theta R\sin\theta] = E[R^2 \sin\theta\cos\theta] = E[R^2] \cdot E[\sin\theta \cdot \cos\theta] = \frac{1}{\lambda} E[\sin\theta\cos\theta]$

$E[\sin\theta\cos\theta] = E[\frac{1}{2}\sin 2\theta] = \int_0^{2\pi} \frac{1}{2}\sin 2\theta \frac{1}{2\pi} d\theta = 0$

$\therefore \boxed{E[N_1 N_2] = 0}$

Q4



4. $F_{R^2}(r^2) = 1 - e^{-\lambda r^2}$

$u = 1 - e^{-\lambda r^2}$

$e^{-\lambda r^2} = 1 - u$

$-\lambda r^2 = \ln(1-u)$

$r^2 = -\frac{1}{\lambda}\ln(1-u)$

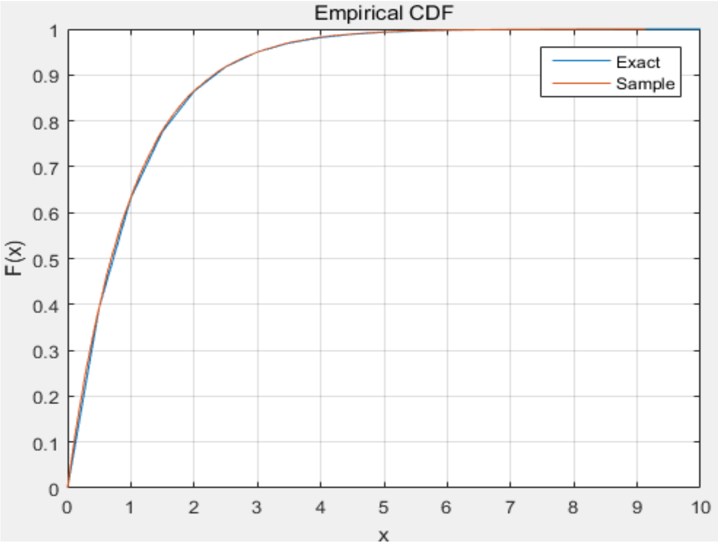
and $\ln(1-U)$ and $\ln(U)$ are same distribution.

$\therefore R^2 \sim -\frac{1}{\lambda}\ln(U)$ $\quad U \sim$ Uniform $(0,1)$

Code:

```
function[y1] = getexp(lamda)
u=rand;
y1=-(1/lamda)*log(u); %use uniform distribution for exp distribution
function q4
m=30000;
n = 10;
%plot the exact cdf by directly plot the cdf formula from Q2
x1 = 0:0.5:n;
figure(1);
plot(x1,1-exp(-x1));
hold on
%plot the sample cdf by using 'getexp' to create exponential random
r2 = zeros(1,m);
for i = 1:m
    r2(i) = getexp(1);
end
cdfplot(r2);
legend('Exact','Sample')
```

result



Q5

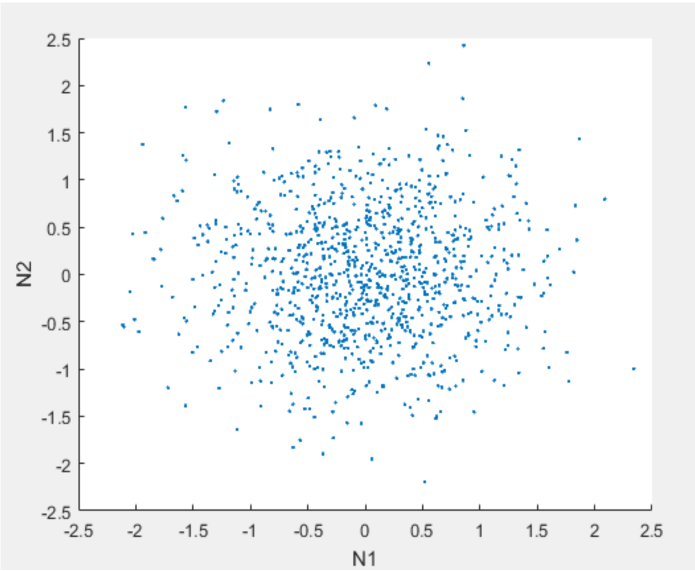
<u>Code:</u>

```
function[r,n1,n2] = cartesian(lamda)
r2 = getexp(lamda); %generate random variable r^2, theta and
theta = rand*2*pi;  %compute its cartesian coordinates N1,N2
r = sqrt(r2);
n1 = r*cos(theta);
n2 = r*sin(theta);
m = 30000;
lamda = 1;
R = zeros(1,m);
N1 = zeros(1,m);
N2 = zeros(1,m);
count1 = 0;
count2 = 0;
count3 = 0;
for i = 1:m
    [R(i),N1(i),N2(i)] = cartesian(lamda); %use function 'cartesian'
to compute R,N1,N2
    if N1(i)<1
       count1 = count1 + 1; %count the point that x1 is decoded
correctly
    end
    if N2(i)>-1
       count2 = count2 + 1; %count the point that x2 is decoded
correctly
    end
    if N1(i)<1 && N2(i)>-1
       count3 = count3 + 1; %count the point that both x1 and x2 are
```

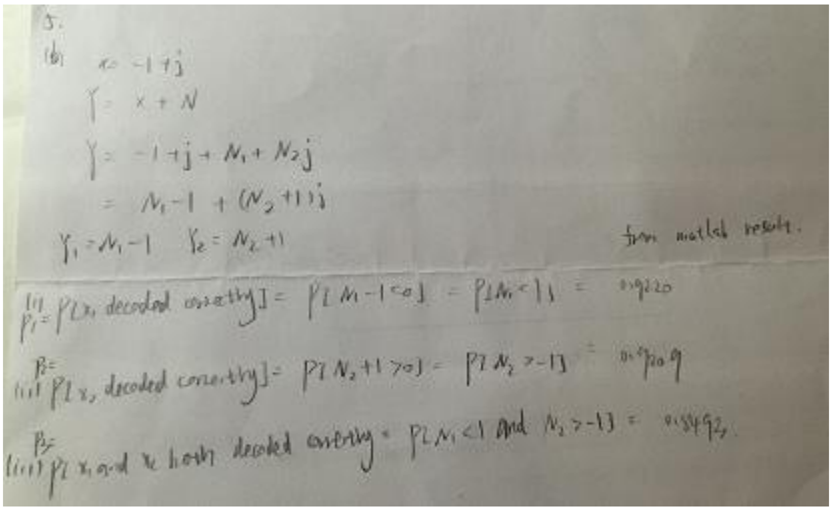
```
decoded correctly
    end
end
figure(2);
scatter(N1(1:1000), N2(1:1000),'.'); %scatter plot for first 1000
samples
xlabel('N1');
ylabel('N2');
p1 = count1/m; %probability for x1 decoded correctly
p2 = count2/m; %probability for x2 decoded correctly
p3 = count3/m; %probability for both x1 and x2 decoded correctly
disp(p1);
disp(p2);
disp(p3);
```

result:

a)



b)



5.

(b) $x = -1 + j$

$Y = x + N$

$Y = -1 + j + N_1 + N_2 j$

$= N_1 - 1 + (N_2 + 1)j$

$Y_1 = N_1 - 1 \quad Y_2 = N_2 + 1$

from matlab results.

(i) $p_1 = P[x_1$ decoded correctly$] = P[N_1 - 1 < 0] = P[N_1 < 1] = 0.9220$

(ii) $p_2 = P[x_2$ decoded correctly$] = P[N_2 + 1 > 0] = P[N_2 > -1] = 0.9209$

(iii) $p_3 = P[x_1$ and x_2 both decoded correctly$] = P[N_1 < 1$ and $N_2 > -1] = 0.8492$

```
0.9220

0.9209

0.8493
```

Q6

b. N_1 and N_2 are independent

Because from matlab result, $P\{N_1<1\} = 0.9220$ $P\{N_2>-1\} = 0.9209$ $P\{N_1<1 \text{ and } N_2>-1\} = 0.8493$

$P\{N_1<1 \text{ and } N_2>-1\} = P\{N_1<1\} \times P\{N_2>-1\}$ N_1 and N_2 independent.

Q7

Code:

```
p1 = normcdf(1,0,sqrt(0.5));
p2 = 1-normcdf(-1,0,sqrt(0.5));
p3 = p1*p2;
disp(p1);
disp(p2);
disp(p3);
```

```
>> q7
    0.9214

    0.9214

    0.8489
```

7. $N_1, N_2 \sim N(0, 0.5)$ N_1, N_2 independent

$p_1 = P\{X_1 \text{ decoded correctly}\} = P\{N_1<1\} = \text{normcdf}(1, 0, \sqrt{0.5}) = 0.9214$

$p_2 = P\{X_2 \text{ decoded correctly}\} = P\{N_2>-1\} = 1-\text{normcdf}(-1, 0, \sqrt{0.5}) = 0.9214$

$p_3 = P\{X_1 \text{ and } X_2 \text{ both decoded correctly}\} = p_1 p_2 = 0.8489$

which is close to Q5 b answer

Q8

Code:

```
function q8
m = 10000;
lamda = 1;
count = 0;
X1 = zeros(1,m);
X2 = zeros(1,m);
N1 = zeros(1,m);
```

```
N2 = zeros(1,m);
Y11 = [];
Y12 = [];
Y21 = [];
Y22 = [];
Y31 = [];
Y32 = [];
Y41 = [];
Y42 = [];
W11 = [];
W12 = [];
W21 = [];
W22 = [];
W31 = [];
W32 = [];
W41 = [];
W42 = [];
for i = 1:m
    [~,N1(i),N2(i)] = cartesian(lamda);
    r = rand; %generate 4 types of x with equal probability
    if(r < 0.25)
        X1(i) = 1;
        X2(i) = 1;
    end
    if(r > 0.25 && r < 0.5)
        X1(i) = 1;
        X2(i) = -1;
    end
    if(r > 0.5 && r < 0.75)
        X1(i) = -1;
        X2(i) = -1;
    end
    if(r > 0.75)
        X1(i) = -1;
        X2(i) = 1;
    end
    if X1(i)>0 && X2(i)>0 %for x in I quadrant
        y11 = N1(i)+X1(i); %compute y by adding noise
        y12 = N2(i)+X2(i);
        Y11 = [Y11 y11];
        Y12 = [Y12 y12];
        if y11<0||y12<0 %for wrongly decoded case as the decision rule
            W11 = [W11 y11];
            W12 = [W12 y12];
```

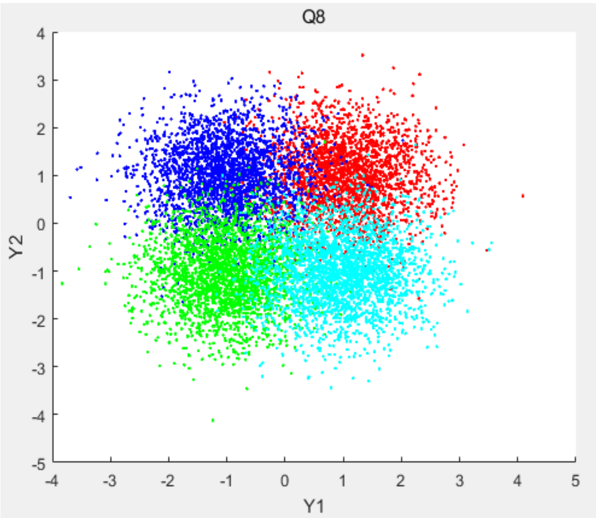
```
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)>0 %for x in II quadrant
        y21 = N1(i)+X1(i); %compute y by adding noise
        y22 = N2(i)+X2(i);
        Y21 = [Y21 y21];
        Y22 = [Y22 y22];
        if y21>0||y22<0 %for wrongly decoded case as the decision rule
            W21 = [W21 y21];
            W22 = [W22 y22];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)<0 %for x in III quadrant
        y31 = N1(i)+X1(i); %compute y by adding noise
        y32 = N2(i)+X2(i);
        Y31 = [Y31 y31];
        Y32 = [Y32 y32];
        if y31>0||y32>0 %for wrongly decoded case as the decision rule
            W31 = [W31 y31];
            W32 = [W32 y32];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)>0 && X2(i)<0 %for x in IV quadrant
        y41 = N1(i)+X1(i); %compute y by adding noise
        y42 = N2(i)+X2(i);
        Y41 = [Y41 y41];
        Y42 = [Y42 y42];
        if y41<0||y42>0 %for wrongly decoded case as the decision rule
            W41 = [W41 y41];
            W42 = [W42 y42];
        else
            count = count+1; %count the correctly decoded points
        end
    end
end
figure(3);
hold on
%plot y for 4 quadrant in different color(I-red,II-blue,III-green,IV-
```

```
cyan)
scatter(Y11, Y12,'.','r');
scatter(Y21, Y22,'.','b');
scatter(Y31, Y32,'.','g');
scatter(Y41, Y42,'.','c');
xlabel('Y1');
ylabel('Y2');
title('Q8');
%plot wrongly decoded points in different colors as before
figure(4);
title('Q8 wrongly decoded')
hold on
scatter(W11, W12,'.','r');
scatter(W21, W22,'.','b');
scatter(W31, W32,'.','g');
scatter(W41, W42,'.','c');
xlabel('Y1');
ylabel('Y2');
%compute wrongly decoded probability(1-correct)
p=1-(count/m);
disp(p)
```

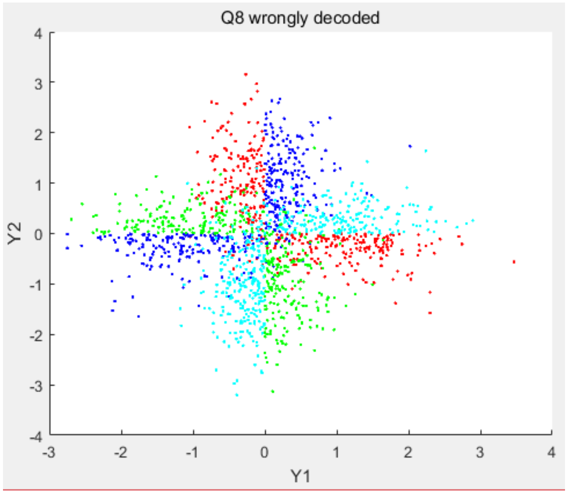
result

a)

lamda=1



b)



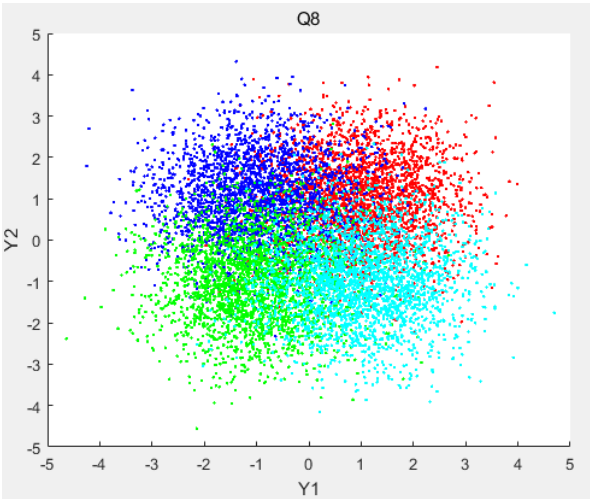
c)

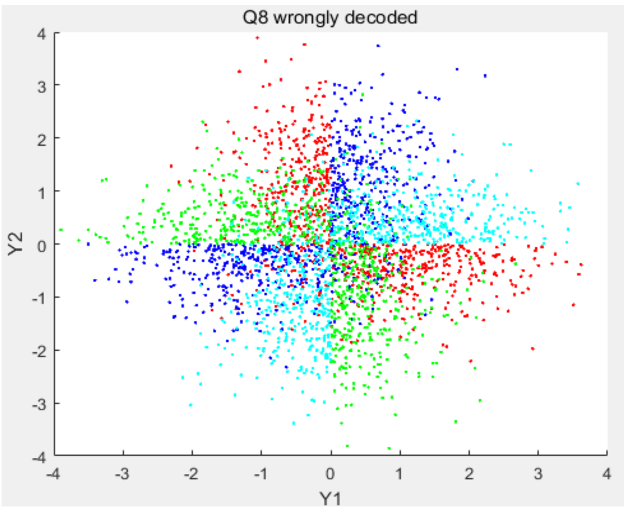
p(error)=0.1527

```
>> q8
     0.1527
```

d)

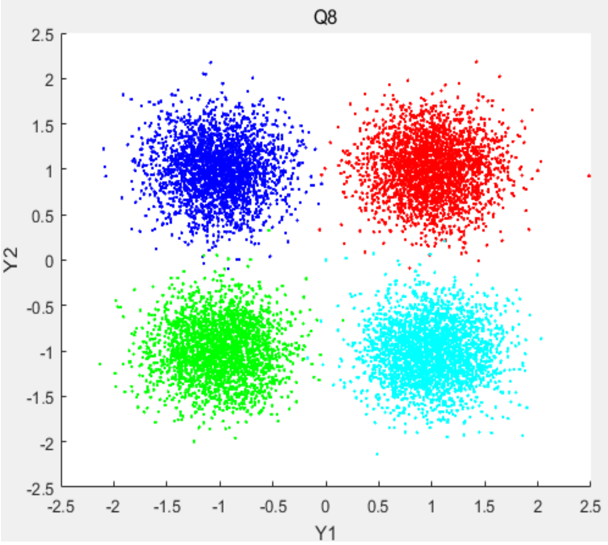
<u>lamda=0.5</u>

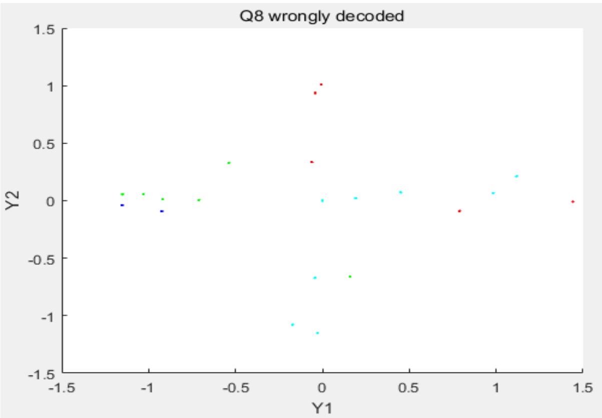




```
>> q8
     0.2874
```

Lamda=5






```
>> q8
    0.0020
```

From the results we can see that with bigger lamda, the 4 types of received Y are more separate in 4 quadrant and intersection between clouds is smaller, the wrongly decoded points also become less and the decoder error probability become smaller.

The reason is that for exp distribution variance is 1/(lamda)^2, which means that the bigger lamda is the smaller variance is. With smaller variance the radius of output Y clouds will be smaller because they stays closer to the mean, the cloud will be more constraint in its quadrant so that the decision rule is determine the input by see which quadrant the output is will have less errors, and the error probability and the number of wrongly decoded points will be smaller.

Q9

Code:

Q9_calculate.m

```
k = linspace(-2,0,1000);%find new decision rule of k
p = (3/4)* (normcdf(1-k,0,sqrt(0.5))).^2 +
(1/12)*normcdf(1+k,0,sqrt(0.5)).^2 + 2 *(1/12)* normcdf(1-
k,0,sqrt(0.5)) .* normcdf(1+k,0,sqrt(0.5));
[pmax index] = max(p);
fprintf('k=%f,p(correct max)=%f,p(error min)=%f\n',k(index),pmax,1 -
pmax);
```

q9_new.m

```
function q9_new
m = 10000;
lamda = 1;
count = 0;
X1 = zeros(1,m);
X2 = zeros(1,m);
N1 = zeros(1,m);
N2 = zeros(1,m);
Y11 = [];
Y12 = [];
Y21 = [];
Y22 = [];
Y31 = [];
Y32 = [];
Y41 = [];
Y42 = [];
W11 = [];
W12 = [];
W21 = [];
W22 = [];
W31 = [];
W32 = [];
W41 = [];
W42 = [];
for i = 1:m
    [~,N1(i),N2(i)] = cartesian(lamda);
    r = rand;%generate 4 types of x with different probability as
required
    if(r < 0.75)
       X1(i) = 1;
       X2(i) = 1;
    end
    if(r > 0.75 && r < 0.833)
       X1(i) = 1;
       X2(i) = -1;
```

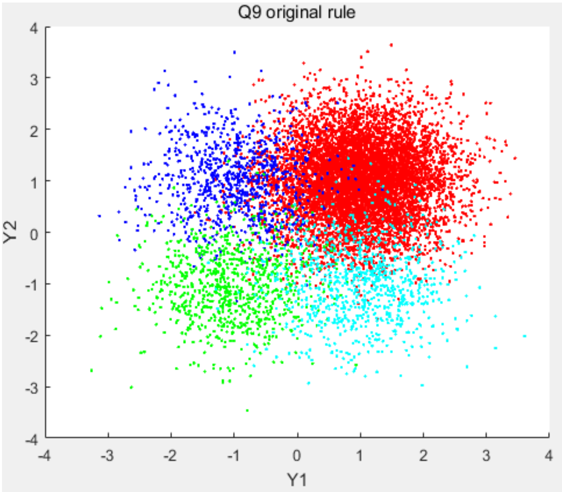
```
    end
    if(r > 0.833 && r < 0.916)
        X1(i) = -1;
        X2(i) = -1;
    end
    if(r > 0.916)
        X1(i) = -1;
        X2(i) = 1;
    end
    if X1(i)>0 && X2(i)>0 %for x in I quadrant
        y11 = N1(i)+X1(i); %compute y by adding noise
        y12 = N2(i)+X2(i);
        Y11 = [Y11 y11];
        Y12 = [Y12 y12];
        if y11<-0.4224||y12<-0.4224 %change as new decision rule as
computed in q9_calculate
            W11 = [W11 y11];
            W12 = [W12 y12];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)>0 %for x in II quadrant
        y21 = N1(i)+X1(i); %compute y by adding noise
        y22 = N2(i)+X2(i);
        Y21 = [Y21 y21];
        Y22 = [Y22 y22];
        if y21>-0.4224||y22<-0.4224 %for wrongly decoded case as the
decision rule
            W21 = [W21 y21];
            W22 = [W22 y22];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)<0 %for x in III quadrant
        y31 = N1(i)+X1(i); %compute y by adding noise
        y32 = N2(i)+X2(i);
        Y31 = [Y31 y31];
        Y32 = [Y32 y32];
        if y31>-0.4224||y32>-0.4224 %for wrongly decoded case as the
decision rule
            W31 = [W31 y31];
            W32 = [W32 y32];
```

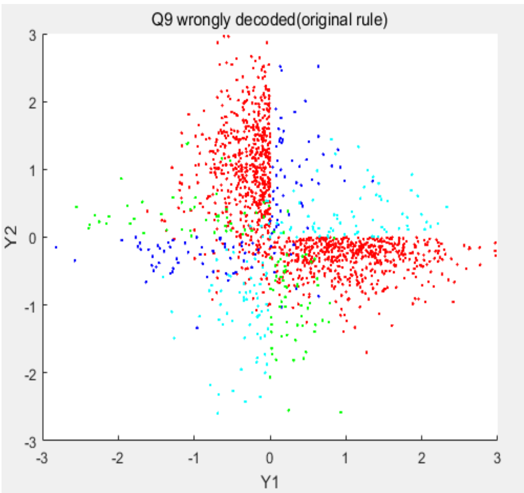
```
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)>0 && X2(i)<0 %for x in IV quadrant
        y41 = N1(i)+X1(i); %compute y by adding noise
        y42 = N2(i)+X2(i);
        Y41 = [Y41 y41];
        Y42 = [Y42 y42];
        if y41<-0.4224||y42>-0.4224 %for wrongly decoded case as the
decision rule
            W41 = [W41 y41];
            W42 = [W42 y42];
        else
            count = count+1; %count the correctly decoded points
        end
    end
end
figure(11);
hold on
%plot y for 4 quadrant in different color(I-red,II-blue,III-green,IV-
cyan)
scatter(Y11, Y12,'.','r');
scatter(Y21, Y22,'.','b');
scatter(Y31, Y32,'.','g');
scatter(Y41, Y42,'.','c');
xlabel('Y1');
ylabel('Y2');
title('Q9 new rule')
%plot wrongly decoded points in different colors as before
figure(12);
title('Q9 wrongly decoded(new rule)')
hold on
scatter(W11, W12,'.','r');
scatter(W21, W22,'.','b');
scatter(W31, W32,'.','g');
scatter(W41, W42,'.','c');
xlabel('Y1');
ylabel('Y2');
%compute wrongly decoded probability(1-correct)
p=1-(count/m);
disp(p)
```

result:

By changing the probability of input X, we can see the error probability is now 0.1521, which is pretty close to the error probability in Q8. That is because in Q8 we increase the input in I quadrant, correspondingly, the errors in I quadrant increase. So that the error probability is close to Q8.

```
>> q9_original
    0.1521
```





To make error probability smaller, intuitively from the scatter plot below, we can see that there is a bigger cloud of points in I quadrant which means that we need to make a bigger space for I quadrant to hold correctly decoded points, that is to move the horizontal axis downward and move the vertical axis to left. So K must be negative. Here we do some analysis and calculation to determine new decision rule.

Q9.

Assume decision rule is: decodes x_1 as 1 if the channel output $y_1 > k$, and as -1 otherwise; similarly for x_2 and y_2.

Here we try to determine the value of k when error propobility is smallest.

① For $x = 1+j$ $(x_1 = 1, x_2 = 1)$, we compute $P_1 = P$(both x_1 and x_2 decoded correctly)

$$\begin{cases} y_1 > k \\ \text{and} \\ y_2 > k \end{cases} \Rightarrow \begin{cases} x_1 + N_1 > k \\ \text{and} \\ x_2 + N_2 > k \end{cases} \Rightarrow \begin{cases} N_1 > k - x_1 \\ \text{and} \\ N_2 > k - x_2 \end{cases} \Rightarrow \begin{cases} N_1 > k-1 \\ \text{and} \\ N_2 > k-1 \end{cases}$$

$\lambda = 1$ $N_1, N_2 \sim N(0, 0.5)$ and N_1, N_2 independent.

$P_1 = P(\text{both } x_1 \text{ and } x_2 \text{ correct}) = P(N_1 > k-1) \cdot P(N_2 > k-1) = [\text{normcdf}[(1-k), 0, \sqrt{0.5}]]^2$

② For $x = -1+j$ $(x_1 = -1, x_2 = 1)$ compute $P_2 = P$(both x_1 x_2 decoded correctly)

$$\begin{cases} y_1 < k \\ \text{and} \\ y_2 > k \end{cases} \Rightarrow \begin{cases} x_1 + N_1 < k \\ \\ x_2 + N_2 > k \end{cases} \Rightarrow \begin{cases} N_1 < k - x_1 \\ \\ N_2 > k - x_2 \end{cases} \Rightarrow \begin{cases} N_1 < k+1 \\ \\ N_2 > k-1 \end{cases}$$

$P_2 = P(N_1 < k+1) \cdot P(N_2 > k-1) = \text{normcdf}(1-k, 0, 0.5)\,\text{normcdf}(k+1, 0, \sqrt{0.5})$

③ For $x = -1-j$ $(x_1 = -1, x_2 = -1)$ compute $P_3 = P$(both x_1 x_2 decoded correctly)

$$\begin{cases} y_1 < k \\ \\ y_2 < k \end{cases} \Rightarrow \begin{cases} x_1 + N_1 < k \\ \\ x_2 + N_2 < k \end{cases} \Rightarrow \begin{cases} N_1 < k+1 \\ \\ N_2 < k+1 \end{cases}$$

$P_3 = [\text{normcdf}(k+1, 0, \sqrt{0.5})]^2$

④ For $x = 1-j$ $(x_1 = 1, x_2 = -1)$ compute $P_4 = P$(both x_1 x_2 decoded correctly)

$$\begin{cases} y_1 > k \\ y_2 < k \end{cases} \Rightarrow \begin{cases} x_1 + N_1 > k \\ x_2 + N_2 < k \end{cases} \Rightarrow \begin{cases} N_1 > k-1 \\ N_2 < k+1 \end{cases}$$

$P_4 = \text{normcdf}(1-k, 0, \sqrt{0.5}) \cdot \text{normcdf}(k+1, 0, \sqrt{0.5})$

KOKUYO

P(both x_1 and x_2 correctly decoded under 4 cases) $= \frac{3}{4}P_1 + \frac{1}{12}P_2 + \frac{1}{12}P_3 + \frac{1}{12}P_4$

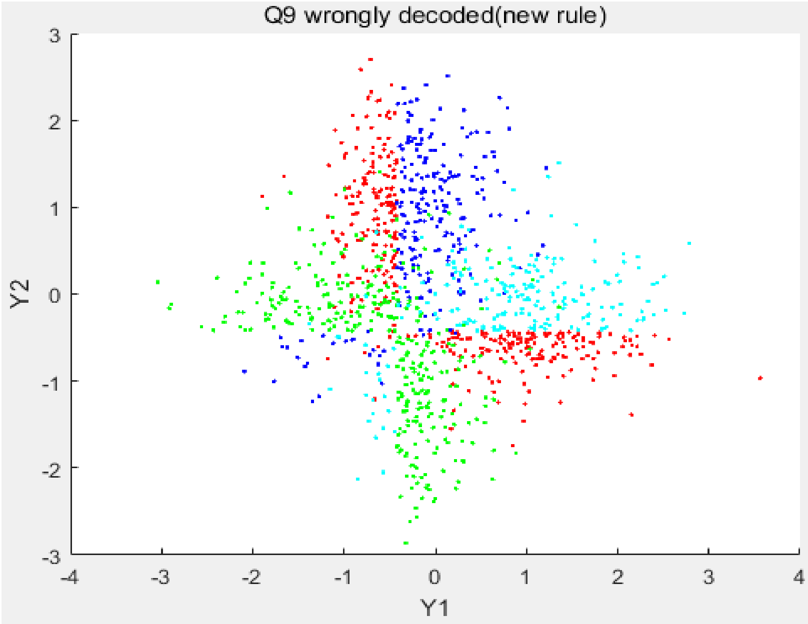
For error probability to be smallest, this probability need to be biggest.

Using Matlab to compute k when $\frac{3}{4}P_1 + \frac{1}{12}P_2 + \frac{1}{12}P_3 + \frac{1}{12}P_4$ is max.

```
>> q9_calculate
k=-0.422422,p(correct max)=0.898812,p(error min)=0.101188
```

New decision rule: decodes x1 as 1 if Y1>-0.4224 and as -1 otherwise; similarly for x2 and Y2. Change the decision rule of the original code, outcome now is smaller error probability and less wrongly decoded points in the plot :

```
>> q9_new
   0.1088
```



Q10

Code:

Q10 calculate.m

```
k = linspace(-2.0,0,1000);%find new decision rule of k
p = (1)* (normcdf(1-k,0,sqrt(0.5))).^2 +
(0)*normcdf(1+k,0,sqrt(0.5)).^2 + 2 *(0)* normcdf(1-k,0,sqrt(0.5)) .*
normcdf(1+k,0,sqrt(0.5));
[pmax index] = max(p);
fprintf('k=%f,p(correct max)=%f,p(error min)=%f\n',k(index),pmax,1 -
pmax);
```

Q10_new.m

```
m = 10000;
lamda = 1;
count = 0;
X1 = zeros(1,m);
X2 = zeros(1,m);
N1 = zeros(1,m);
```

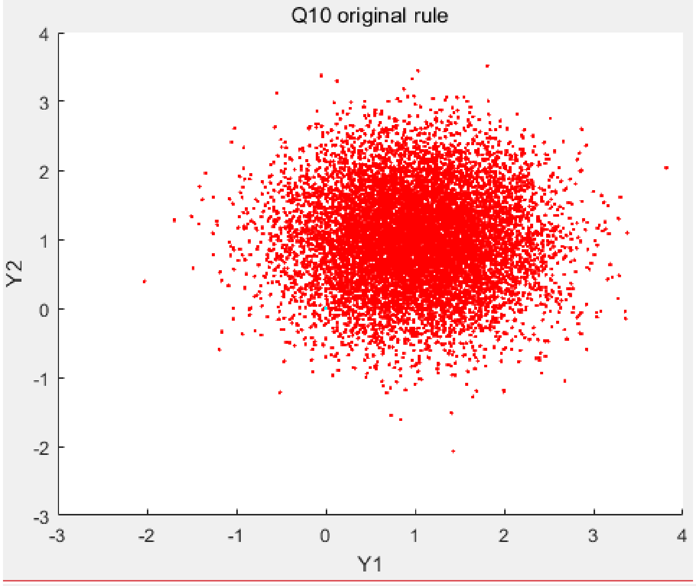
```
N2 = zeros(1,m);
Y11 = [];
Y12 = [];
Y21 = [];
Y22 = [];
Y31 = [];
Y32 = [];
Y41 = [];
Y42 = [];
W11 = [];
W12 = [];
W21 = [];
W22 = [];
W31 = [];
W32 = [];
W41 = [];
W42 = [];
for i = 1:m
    [~,N1(i),N2(i)] = cartesian(lamda);
    X1(i) = 1;
    X2(i) = 1;
    if X1(i)>0 && X2(i)>0 %for x in I quadrant
        y11 = N1(i)+X1(i); %compute y by adding noise
        y12 = N2(i)+X2(i);
        Y11 = [Y11 y11];
        Y12 = [Y12 y12];
        if y11<-3||y12<-3 %for wrongly decoded case as the decision
rule%change as new decision rule as computed in q9_calculate
            W11 = [W11 y11];
            W12 = [W12 y12];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)>0 %for x in II quadrant
        y21 = N1(i)+X1(i); %compute y by adding noise
        y22 = N2(i)+X2(i);
        Y21 = [Y21 y21];
        Y22 = [Y22 y22];
        if y21>-3||y22<-3 %for wrongly decoded case as the decision
rule
            W21 = [W21 y21];
            W22 = [W22 y22];
        else
```

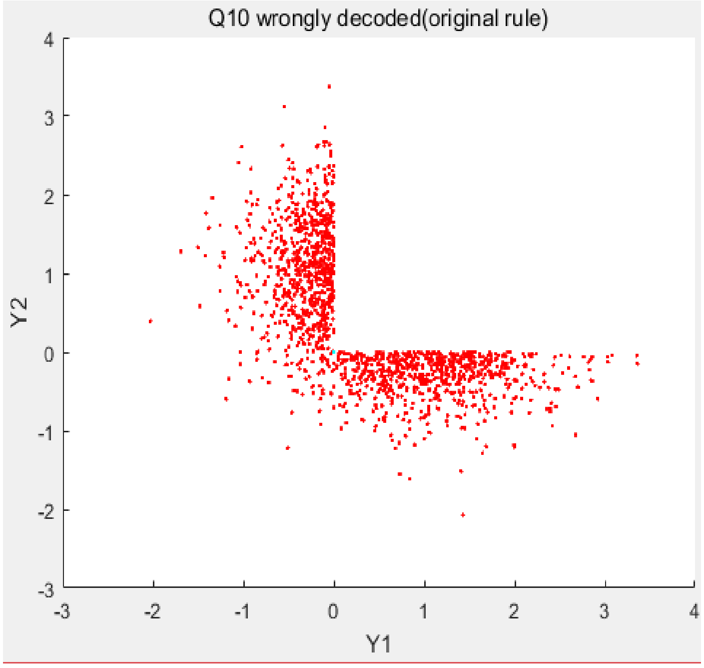
```
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)<0 && X2(i)<0 %for x in III quadrant
        y31 = N1(i)+X1(i); %compute y by adding noise
        y32 = N2(i)+X2(i);
        Y31 = [Y31 y31];
        Y32 = [Y32 y32];
        if y31>-3||y32>-3 %for wrongly decoded case as the decision
rule
            W31 = [W31 y31];
            W32 = [W32 y32];
        else
            count = count+1; %count the correctly decoded points
        end
    end
    if X1(i)>0 && X2(i)<0 %for x in IV quadrant
        y41 = N1(i)+X1(i); %compute y by adding noise
        y42 = N2(i)+X2(i);
        Y41 = [Y41 y41];
        Y42 = [Y42 y42];
        if y41<-3||y42>-3 %for wrongly decoded case as the decision
rule
            W41 = [W41 y41];
            W42 = [W42 y42];
        else
            count = count+1; %count the correctly decoded points
        end
    end
end
figure(15);
hold on
%plot y for 4 quadrant in different color(I-red,II-blue,III-green,IV-
cyan)
scatter(Y11, Y12,'.','r');
scatter(Y21, Y22,'.','b');
scatter(Y31, Y32,'.','g');
scatter(Y41, Y42,'.','c');
xlabel('Y1');
ylabel('Y2');
title('Q10 new rule')
%plot wrongly decoded points in different colors as before
figure(16);
title('Q10 wrongly decoded(new rule)')
```

```
hold on
scatter(W11, W12,'.','r');
scatter(W21, W22,'.','b');
scatter(W31, W32,'.','g');
scatter(W41, W42,'.','c');
xlabel('Y1');
ylabel('Y2');
%compute wrongly decoded probability(1-correct)
p=1-(count/m);
disp(p)
```

result:

when original decision rule is applied.





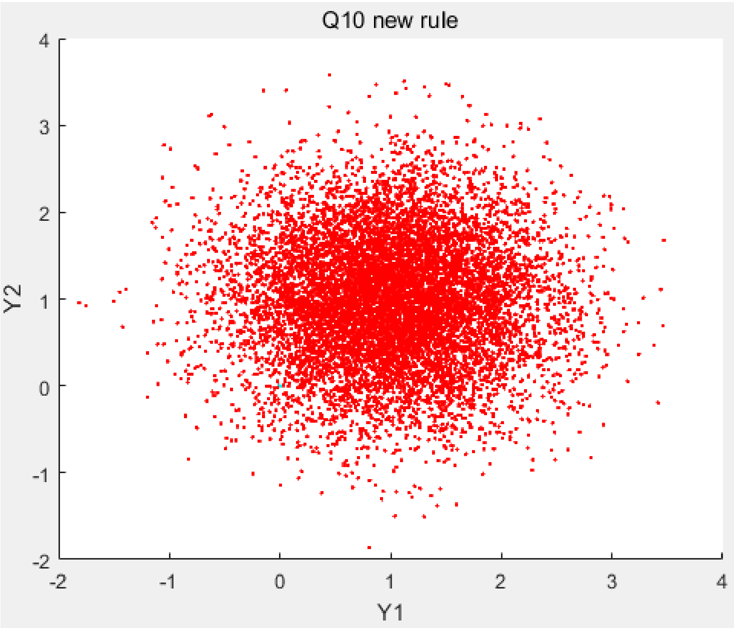
```
>> q10_original
   0.1513
```

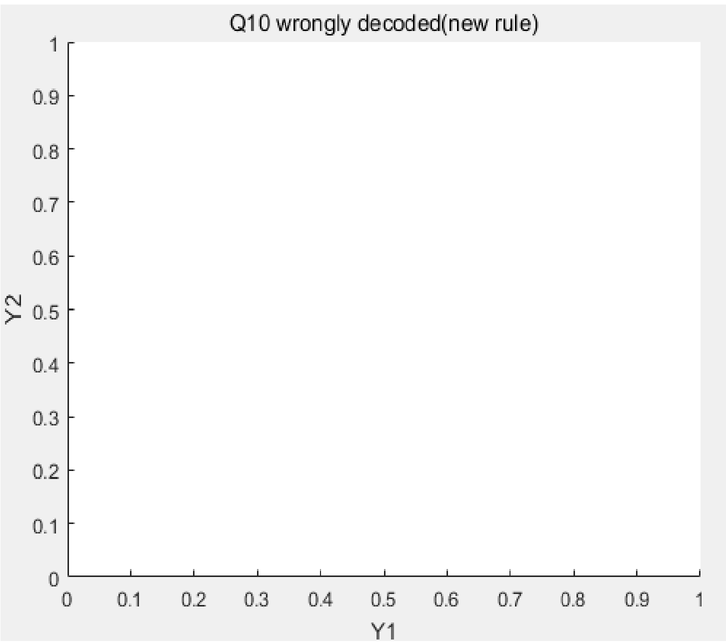
Similarly, as in Q9, now we just need to compute k when the correct probability (1*p1+0*p2+0*p3+0*p4) to be close to 1. Using Matlab we get when k is -3, the error probability is 0 . And intuitively, the more k approaches to negative infinity, the more error probability approaches to 0.

So the new decision rule for error probability to be small is: decodes x1 as 1 if Y1>-3 and as -1 otherwise; similarly for x2 and Y2.

```
>> q10_calculate
k=-3.000000,p(correct max)=1.000000,p(error min)=0.000000
```

After applying new decision rule, do the simulation again we get zero error probability. Results as below:





```
>> q10_original
    0.1513

>> q10_new
     0
```