4TK4

Project 1

TIANZE ZHANG

4002813J

Part 1:

$$h(n)=\begin{cases} a[1-b(n)], & n=1.2.3 \\ 0 & \text{elsewhere} \end{cases}$$

$b(n)=-\cos\left(\frac{2\pi}{L}(n-2)\right)$ where $w=3.1$ $a=0.5$

$\therefore h(1)=0.28$ $h(2)=1$ $h(3)=0.28$

To calculate Wopt. we first randomly choose $a(n)$.

Say $a(1)$.

In order to calculate $a(1)$ with 11-tap equalizer

we need $x(1), x(2), x(3) \cdots x(11)$.

with $x(1)=a(0)h(1)+a(-1)h(2)+a(-2)h(3)+u(1)$

$x(2)=a(1)h(1)+a(0)h(2)+a(-1)h(3)+u(2)$

$x(n)=a(n-1)h(1)+a(n-2)h(2)+a(n-3)h(3)+u$

$\Phi_{xx}=$

	A	B	C	D	E	F	G	H	I	J	K
1	E[x(1)x(1)]	E[x(1)x(2)]	E[x(1)x(3)]	E[x(1)x(4)]	E[x(1)x(5)]	E[x(1)x(6)]	E[x(1)x(7)]	E[x(1)x(8)]	E[x(1)x(9)]	E[x(1)x(10)]	E[x(1)x(11)]
2	E[x(1)x(2)]	E[x(2)x(2)]	E[x(2)x(3)]	E[x(1)x(4)]	E[x(2)x(5)]	E[x(2)x(6)]	E[x(2)x(7)]	E[x(2)x(8)]	E[x(2)x(9)]	E[x(2)x(10)]	E[x(2)x(11)]
3	E[x(1)x(3)]	E[x(2)x(3)]	E[x(3)x(3)]	E[x(2)x(4)]	E[x(3)x(5)]	E[x(3)x(6)]	E[x(3)x(7)]	E[x(3)x(8)]	E[x(3)x(9)]	E[x(3)x(10)]	E[x(3)x(11)]
4	E[x(1)x(4)]	E[x(2)x(4)]	E[x(3)x(4)]	E[x(4)x(4)]	E[x(4)x(5)]	E[x(4)x(6)]	E[x(4)x(7)]	E[x(4)x(8)]	E[x(4)x(9)]	E[x(4)x(10)]	E[x(4)x(11)]
5	E[x(1)x(5)]	E[x(2)x(5)]	E[x(3)x(5)]	E[x(4)x(5)]	E[x(5)x(5)]	E[x(5)x(6)]	E[x(5)x(7)]	E[x(5)x(8)]	E[x(5)x(9)]	E[x(5)x(10)]	E[x(5)x(11)]
6	E[x(1)x(6)]	E[x(2)x(6)]	E[x(3)x(6)]	E[x(4)x(6)]	E[x(5)x(6)]	E[x(6)x(6)]	E[x(6)x(7)]	E[x(6)x(8)]	E[x(6)x(9)]	E[x(6)x(10)]	E[x(6)x(11)]
7	E[x(1)x(7)]	E[x(2)x(7)]	E[x(3)x(7)]	E[x(4)x(7)]	E[x(5)x(7)]	E[x(6)x(7)]	E[x(7)x(7)]	E[x(7)x(8)]	E[x(7)x(9)]	E[x(7)x(10)]	E[x(7)x(11)]
8	E[x(1)x(8)]	E[x(2)x(8)]	E[x(3)x(8)]	E[x(4)x(8)]	E[x(5)x(8)]	E[x(6)x(8)]	E[x(7)x(8)]	E[x(8)x(8)]	E[x(8)x(9)]	E[x(8)x(10)]	E[x(8)x(11)]
9	E[x(1)x(9)]	E[x(2)x(9)]	E[x(3)x(9)]	E[x(4)x(9)]	E[x(5)x(9)]	E[x(6)x(9)]	E[x(7)x(9)]	E[x(8)x(9)]	E[x(9)x(9)]	E[x(9)x(10)]	E[x(9)x(11)]
10	E[x(1)x(10)]	E[x(2)x(10)]	E[x(3)x(10)]	E[x(4)x(10)]	E[x(5)x(10)]	E[x(6)x(10)]	E[x(7)x(10)]	E[x(8)x(10)]	E[x(9)x(10)]	E[x(10)x(10	E[x(10)x(11)]
11	E[x(1)x(11)]	E[x(2)x(11)]	E[x(3)x(11)]	E[x(4)x(11)]	E[x(5)x(11)]	E[x(6)x(11)]	E[x(7)x(11)]	E[x(8)x(11)]	E[x(9)x(11)]	E[x(10)x(11	E[x(11)x(11)]

from the matrix we can see that

$E[x(1)x(1)]$ & $E[x(2)x(2)]$ & $E[x(3)x(3)]$ & $E[x(n)x(n)]$ is all the same value as also.

$E[x(n)x(n+1)]$ Same value

$E[x(n)x(n+2)]$

⋮

$E[x(n)x(n+10)]$ Same value

$\therefore E[x^2(5)] = E[(a(4)h(1) + a(3)h(2) + a(2)h(3))^2]$

$E[x(n)x(n)]$: eg $E[x(1)x(1)] = h^2(1) + h^2(2) + h^2(3) + E[v(n)] = 1.1568 + 0.001$
$= 1.1578$

$E[x(n)x(n+1)]$: eg $E[x(1)x(2)] = h(1)\cdot h(2) + h_2 \cdot h_3 + E[v(n)] = 0.760 + 0.001$
$= 0.761$

$E[x(n)x(n+2)]$: eg $E[x(1)x(3)] = h(1)\cdot h(3) + E[v(n)] = 0.0784 + 0.001$
$= 0.0794$

Auto-crollation

$\phi_{xx} =$ =

1.1578	0.5610	0.0794	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.5610	1.1578	0.5610	0.0794	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0794	0.5610	1.1578	0.5610	0.0794	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0794	0.5610	1.1578	0.5610	0.0794	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0794	0.5610	1.1578	0.5610	0.0794	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0794	0.5610	1.1578	0.5610	0.0794	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0794	0.5610	1.1578	0.5610	0.0794	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0794	0.5610	1.1578	0.5610	0.0794	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0794	0.5610	1.1578	0.5610	0.0794
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0794	0.5610	1.1578	0.5610
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0794	0.5610	1.1578

Cross - Correlation

$$\underline{\Phi}_{a(1)} = \begin{bmatrix} E[x(1)\,a(1)] \\ E[x(2)\,a(1)] \\ E[x(3)\,a(1)] \\ \vdots \\ E[x(11)\,a(1)] \end{bmatrix} = \begin{bmatrix} a^2(1)\,h(1) \\ h(2) \\ h(3) \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} h(1) \\ h(2) \\ h(3) \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0.28 \\ 1 \\ 0.28 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

11 · 1

eg:

$$E[x(5)\,a(4)] = E[(a(4)\,h(1) + a(3)\,h(2) + a(2)\,h(3) + a(5))\,a(4)]$$

we looking for square like $a^2(4) = 1$. otherwise $= 0$.

$$\therefore E[x(5)\,a(4)] = a^2(4)\,h(1) = h(1)$$

$$E[x(6)\,a(4)] = h(2)$$

$$E[x(7)\,a(4)] = h(3)$$

$$W_{op} = \underline{\Phi}_{xx}^{-1} \cdot \underline{\Phi}_{ax}$$

$\underline{\Phi}_{xx}^{-1}$: auto; $\underline{\Phi}_{ax}$: cross

$$\epsilon^2_{min} = W_{op}^T\,\phi_{ax}\,W_{op} - 2\,W_{op}^T\,\phi_{ax} + E[a^2(n)]$$

$E[a^2(n)] \to 1$

Part A 2. $\underline{w}(i+1) = \underline{w}(i) - \frac{1}{2}\mu \nabla \varepsilon^2 \big|_{w=w(i)}$

choose by yourself Step_size

$\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{11 \times 1}$

$\nabla \varepsilon^2 = -2[\underline{\Phi}_{xx}\underline{w} - \underline{\phi}_{dx}]$ Matlab using iteration

our task is to calculate $\varepsilon^2 - \varepsilon^2_{min}$

```
wop =

   -0.3038
    1.1766
   -0.3572
    0.1071
   -0.0317
    0.0093
   -0.0027
    0.0007
   -0.0002
    0.0001
   -0.0000
miu =

    0.0750

After 81.0 iterations, the minimum error is smaller than 0.01.
current excessive mean square error = 0.0098
Excess Mean Square Error = Mean Sqaure Error - Min = 0.0013
miu =

    0.0075

After 814.0 iterations, the minimum error is smaller than 0.01.
current excessive mean square error = 0.0100
Excess Mean Square Error = Mean Sqaure Error - Min = 0.0015
miu =

    0.0250

After 244.0 iterations, the minimum error is smaller than 0.01.
current excessive mean square error = 0.0099
Excess Mean Square Error = Mean Sqaure Error - Min = 0.0014
```

As we can see from matlab result, with $\mu = 0.075$, we need 81 iterations

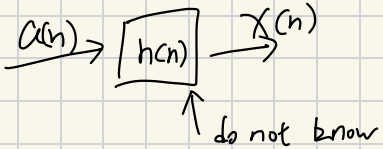
As we can see, the smaller the u, the more iteration we need to get a good enough W, which make sense, since u is part of the step size $\frac{1}{2}\mu\nabla\varepsilon^2$

Matlab code for part A

```
clc;
clear;

A = 1.1578;
B = 0.561;
C = 0.0794;
R1 = [A B C 0 0 0 0 0 0 0 0];
R2 = [B A B C 0 0 0 0 0 0 0];
R3 = [C B A B C 0 0 0 0 0 0];
R4 = [0 C B A B C 0 0 0 0 0];
R5 = [0 0 C B A B C 0 0 0 0];
R6 = [0 0 0 C B A B C 0 0 0];
R7 = [0 0 0 0 C B A B C 0 0];
R8 = [0 0 0 0 0 C B A B C 0];
R9 = [0 0 0 0 0 0 C B A B C];
R10 = [0 0 0 0 0 0 0 C B A B];
R11 = [0 0 0 0 0 0 0 0 C B A];

% Create a 2D matrix by stacking these rows
matrix_11x11 = [R1; R2; R3; R4; R5; R6; R7; R8; R9; R10; R11];

% Display the 11x11 matrix in a visually appealing format
fprintf('%8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n', matrix_11x11.');

auto_correlation = [R1; R2; R3; R4; R5; R6; R7; R8; R9; R10; R11];
cross_correlation = transpose([0.28 1 0.28 0 0 0 0 0 0 0 0]);
wopt = inv(auto_correlation) * cross_correlation
Error_mean_square = transpose(wopt) * auto_correlation * wopt - 2 * transpose(wopt) * cross_correlation + 1;

w = transpose([1 1 1 1 1 1 1 1 1 1 1]);
miu = 0.025
size = 1:1:2000;
threshold = 0.01; % Threshold variable
excessive_means_quare_error = zeros(1, 200); % Initialize the array

for i = size
    w = w - 0.5 * miu * 2 * ((auto_correlation) * w - cross_correlation);
    error_w = transpose(w) * auto_correlation * w - 2 * transpose(w) * cross_correlation + 1;
    excessive_means_quare_error(i) = error_w - Error_mean_square;

    if excessive_means_quare_error(i) <= threshold
        fprintf('After %.1f iterations, the minimum error is smaller than %.2f.\n', i, threshold);
        fprintf('current excessive mean square error = %.4f\n', excessive_means_quare_error(i));
        fprintf('Excess Mean Square Error = Mean Sqaure Error - Min = %.4f\n', excessive_means_quare_error(i) - Error_mean_square );

        break; % Exit the loop when the condition is met
    end
end

plot(1:i, excessive_means_quare_error(1:i)); % Plot only the data up to the stopping point
```

Part B.

In Part B. we do not know $h(n)$



Still start with an inital w.

$$w^{(i+1)} = w^{(i)} - \frac{1}{2}\mu \nabla \varepsilon^2 \Big|_{w=w^{(i)}}$$

we cannot use this formula b/c to use this we need know the channel (h(n))

where $\nabla \varepsilon^2 = -2[\phi_{xx} w - \phi_{dx}]$

So we need to use:

$$\nabla \varepsilon^2 = \begin{bmatrix} E[e(1)x(2)] \\ E[e(1)x(3)] \\ E[e(1)x(4)] \\ \vdots \\ E[e(1)x(12)] \end{bmatrix}$$

x(1) x(2) x(3)

a(1) a(2) a(3)

e(1) we first caput the product of

e(1) with the output x(1) x(2) x(3)

and after that we pump the sequence.

x(2) x(3) x(4)

a(2) a(3) a(4)

e(2)

right now. we can calculate the average.

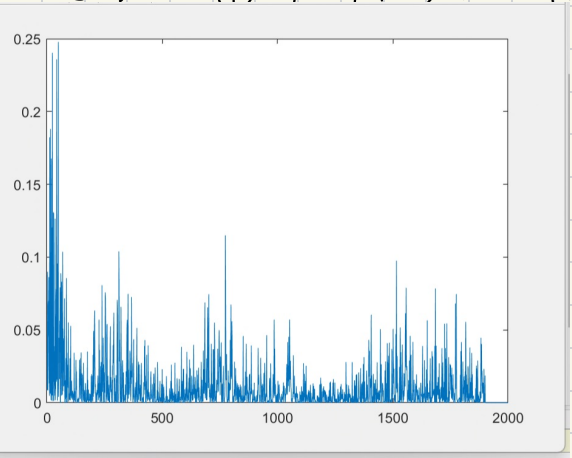
$$\frac{e(1)x(2) + e(2)x(2)}{2}$$

So for our Question

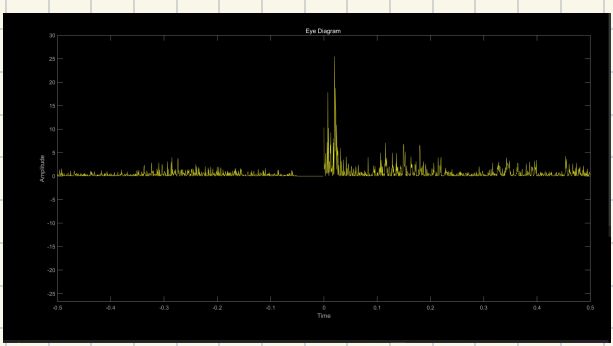
we need || average $E[\underline{e(n)x(n)} \cdots e]$
||

$$\nabla \xi^2 = 2 \begin{bmatrix} \underline{e(1)x(2) + e(2)x(3) + e(3)x(4) + \cdots e(11)x(12)} \\ 11 \\ \underline{e(1)x(3) + e(2)x(4) + e(3)x(5) + \cdots e(11)x(13)} \\ 11 \\ \vdots \\ \underline{e(1)x(12) + e(2)x(13) + e(3)x(14) + \cdots e(11)x(22)} \\ 11 \end{bmatrix}$$

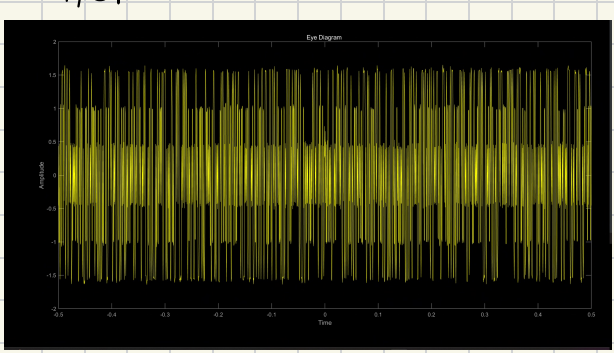
$e(1) = d(1) - [w_1 x(1) + w_2 x(2) + w_3 x(3) + \cdots w_{11} x(11)]$



before:



After



Part B:

```
wnew =

    0.8002
   -0.3727
    0.2035
   -0.2176
    0.0343
    0.0062
    0.1487
   -0.0047
   -0.1441
   -0.2267
   -0.1054
```

Part A

```
w =

   -0.2420
    1.0709
   -0.2480
    0.0251
    0.0154
   -0.0114
    0.0049
   -0.0038
    0.0062
   -0.0078
    0.0053
```

As we can see from the final iteration, w from Part A is generally smaller than w from Part B.

As we can see, the smaller the u, the more iteration we need to get a good enough W, which make sense, since u is part of the step size $\frac{1}{2} u \nabla \varepsilon^2$

Matlab Code:

```
clc;
clear;
%inputing the chanel values and random signals set.
h=[0.28 1 0.28];
x= rand(1,2000);

% implementing random generator output as requested.
for a = 1:1:2000
   if (x(a)>0.5)
      RG(a) = 1;
   elseif(x(a)<=0.5)
      RG(a) = -1;
   end
end

ho = conv(RG,h); %output signal
received = ho + normrnd(0,sqrt(0.001),[1,length(ho)]); %received signal factored in noise


for emsemble = 1:1:50
   wnew = rand(11,1);
   errsqur = transpose([0 0 0 0 0 0 0 0 0 0 0]);
   temp = [zeros(1,2000)];
   for loop = 1:1:1900
         err(loop) = received(loop+1:1:loop+11)*wnew - RG(loop);
         stor(loop) = err(loop)^2;
         errsqur = errsqur + err(loop)*transpose(received(loop+1:1:loop+11));
         wnew = wnew - 0.5*0.0075*2*(errsqur/loop);
   end
   temp(1:length(stor)) = temp(1:length(stor)) + stor;
end
MSE = temp/50;
wnew
size = 1:1:2000;
plot(size,MSE);

eyediagram(received,length(ho));
eyediagram(temp,length(ho));
```

Part c). for $w = 3.5$

$$b(1) = -\cos\left(\frac{2\pi}{3.5}\right) = 0.223$$

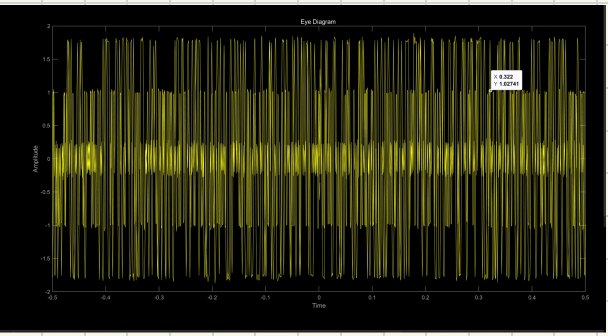
$$b(2) = -1$$

$$b(3) = 0.223$$

$$h(1) = 0.5\,(1 - 0.223) = 0.389$$

$$h(2) = 0.5\,(1 - (-1)) = 1$$

$$h(3) = 0.389$$




```
wop =

   -1.0550
    2.4574
   -1.6251
    0.7316
    0.2218
   -1.0979
    1.7347
   -2.0078
    1.8677
   -1.3605
    0.6401

After 756.0 iterations, the minimum error is smaller than 0.01.
current excessivemeansquareerror = 0.0095
Errormeansquare - excessivemeansquareerror = 0.4244
```

as we can see, with $\mu = 0.02$, we need 756 iteration to make the error smaller than 0.01.

So with $w \uparrow$, we need more iteration to converge.

This make sense with bigger w, the channel $h(n)$ become smaller,

$$\underset{\Delta}{w}^{(i+1)} = w^{(i)} - \frac{1}{2}\mu \nabla \mathcal{E}^2 \Big|_{w = w^{(i)}}$$

$$\underset{\Delta}{\nabla}\mathcal{E}^2 = -2\left[\Phi_{xx} w - \phi_{dx}\right]$$

and we know cross-correlation & auto correlation is calculated from $h(n)$. with $h(n)$ decrease, the value of ϕ_{xx} decrease as well.

And when we do gradient decent, $\frac{1}{2}\mu \cdot \nabla \varepsilon^2$ is the step-size, as ε become smaller, the entire step size become smaller, thus we know more iteration to get a good W.

```
clc;
clear;
%change W to 3.5
h=[0 0.389 1 0.389];
x= rand(1,2000);

h1 = 0.389;
h2 = 1;
h3 = 0.389;
v = 0.001;
A = h1^2+h2^2+h3^3+v;
B = h1*h2+h2*h3+v;
C = h1*h3+v;

R1 = [A B C 0 0 0 0 0 0 0];
R2 = [B A B C 0 0 0 0 0 0];
R3 = [C B A B C 0 0 0 0 0];
R4 = [0 C B A B C 0 0 0 0];
R5 = [0 0 C B A B C 0 0 0];
R6 = [0 0 0 C B A B C 0 0];
R7 = [0 0 0 0 C B A B C 0];
R8 = [0 0 0 0 0 C B A B C];
R9 = [0 0 0 0 0 0 C B A B];
R10 = [0 0 0 0 0 0 0 C B A];
R11 = [0 0 0 0 0 0 0 0 C B A];

% Create a 2D matrix by stacking these rows
matrix_11x11 = [R1; R2; R3; R4; R5; R6; R7; R8; R9; R10; R11];

% Display the 11x11 matrix in a visually appealing format
fprintf('%8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n', matrix_11x11.');

auto_correlation = [R1; R2; R3; R4; R5; R6; R7; R8; R9; R10; R11];
cross_correlation = transpose([h1 h2 h3 0 0 0 0 0 0 0 0]);
wopt = inv(auto_correlation) * cross_correlation
Error_mean_square = transpose(wopt) * auto_correlation * wopt - 2 * transpose(wopt) * cross_correlation + 1;

w = transpose([1 1 1 1 1 1 1 1 1 1 1]);
miu = 0.075;
size = 1:1:2000;
threshold = 0.01; % Threshold variable
excessive_mean_square_error = zeros(1, 200); % Initialize the array

for i = size
    w = w - 0.5 * miu * 2 * ((auto_correlation) * w - cross_correlation);
    error_w = transpose(w) * auto_correlation * w - 2 * transpose(w) * cross_correlation + 1;
    excessive_mean_square_error(i) = error_w - Error_mean_square;

    if excessive_mean_square_error(i) <= threshold
        fprintf('After %.1f iterations, the minimum error is smaller than %.2f.\n', i, threshold);
        fprintf('current excessivemeansquareerror = %.4f\n', excessive_mean_square_error(i));
        fprintf('Errormeansquare - excessivemeansquareerror = %.4f\n', excessive_mean_square_error(i) - Error_mean_square );
        break; % Exit the loop when the condition is met
    end
end

for a = 1:1:2000
    if (x(a)>0.5)
        random_generator(a) = 1;
    elseif(x(a)<=0.5)
        random_generator(a) = -1;
    end
end

ho = conv(random_generator,h);
received = ho + normrnd(0,sqrt(0.001),[1,length(ho)]);

eyediagram(received,1000);
```