TAREA #8

Alexis Luna
Neidys Vargas

1) En la etapa de RECEPCIÓN, agregaremos el ruido tipo AWGN de la siguiente manera de acuerdo con la Fig. 2, asumiendo que la señal recibida (que ya pasó por el canal de comunicaciones) tiene potencia de 1 Watt.

a) Asegúrese que la potencia de la señal recibida (Rx_Signal) es de 1 Watt.

b) Primeramente utilizaremos PNoise = 1; % Indicando que el AWGN tendrá la misma potencia que la señal recibida y por lo tanto el valor SNR_dB = 10*log10(P_Rx_Signal / PNoise) = 0 dB. Posteriormente cambiaremos este valor de PNoise para tener valores de SNR_dB en el rango de 0:3:30 dB

c) Generaremos el AWGN con potencia de valor igual a PNoise de la siguiente forma:

Noise = sqrt(PNoise) * randn(1,numel(Rx_Signal);

d) Verifique la potencia del Ruido con:

var(Noise) % Debe ser igual al valor de PNoise considerado

e) Calcule la relación señal a ruido para ese valor de PNoise con:

SNR_dB = 10*log10(P_Rx_Signal / PNoise)

f) Añada el ruido a la señal recibida de potencia unitaria de la siguiente forma

Rx_Signal_AWGN = Rx_Signal + Noise ;

2) Posteriormente, la señal ruidosa (Rx_Signal_AWGN) pasa a través del Match Filter. Verifique la PSD

```
load lena512.mat
%lenarec=lena512(252:379,318:445); 128x128
%lenarec=lena512(252:338,318:404);%87x87
lenarec=lena512(252:296,318:362); %45x45
figure(1);
imshow(uint8(lenarec))
title("Lena's eye");
b = de2bi(lenarec,8,'left-msb'); % Convert the samples from uint8 to binary
b = b';
bits = b(:);

bits2send = bits';
Fs= 96000;
mp= 10;
%No_bits= 8712;
Baud_rate= Fs/mp   %Symbols per second
Bit_rate=Baud_rate % bits/s
Ts=1/Fs;
% The bit rate is Rb= Rs= Fs / mp, because 1 bit= 1 symbol and  every symbol has mp
% samples per bit


n= 0:(mp-1);
w = pi/mp;
hs = sin(w*n);

pnrz=ones(1,mp);

s1=bits;
s1(s1==0)=-1;
s=zeros(1,numel(s1)*mp);
s(1:mp:end)=s1; %Impulse train
stem(s(1:mp*16))

xPNRZ_hs=conv(hs,s); %Pulse half sine
xPNRZ=conv(pnrz,s); %Pulse rectangular



figure();
pow = sum(xPNRZ_hs.^2)/numel(xPNRZ_hs);
pow_Deseada = 1;
xPNRZ_hs = sqrt(pow_Deseada/pow)*xPNRZ_hs;
```

TAREA #8

Alexis Luna

Neidys Vargas

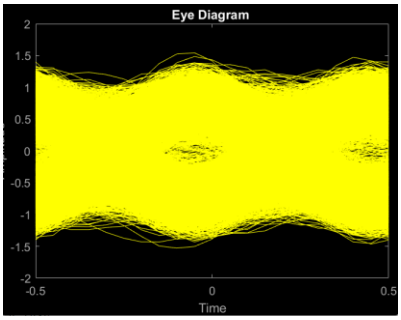
```
%eyediagram(xPNRZ_hs,2*mp);
figure();
%plot(xPNRZ(1:mp*16))
pow = sum(xPNRZ.^2)/numel(xPNRZ);
pow_Deseada = 1;
xPNRZ = sqrt(pow_Deseada/pow)*xPNRZ;
%eyediagram(xPNRZ,2*mp);


orden=60;     % Orden del Filtro
f=   [0 0.6 0.6 1];  % Vector de Frecuencias
m= [1 1 0 0]; % Vector de Magnitudes
f1 = fir2(orden,f,m);  % Coeficientes del Filtro usando FIR2( )
%fvtool(f1);

xPNRZ_filtrado_f1 = conv(xPNRZ_hs,f1);
%eyediagram(xPNRZ_filtrado_f1,2*mp);

xPNRZ_filtrado_f2 = conv(xPNRZ,f1);
%eyediagram(xPNRZ_filtrado_f2,2*mp);


pow1 = sum(xPNRZ_filtrado_f1.^2)/numel(xPNRZ_filtrado_f1);
pow_Deseada = 1;
xPNRZ_filtrado_f1 = sqrt(pow_Deseada/pow1)*xPNRZ_filtrado_f1;

pow2 = sum(xPNRZ_filtrado_f2.^2)/numel(xPNRZ_filtrado_f2);
pow_Deseada = 1;
xPNRZ_filtrado_f2 = sqrt(pow_Deseada/pow2)*xPNRZ_filtrado_f2;


%PNoise = N02 * sum(h.*h) * Ts % Observe que PNoise es un escalar
PNoise = 1;
PNoise = PNoise*mp;

Noise = sqrt(PNoise) * randn(1,numel(xPNRZ_filtrado_f1));
var(Noise) % Debe ser igual al valor de PNoise considerado
SNR_dB = 10*log10(1 / PNoise);

Noise2 = sqrt(PNoise) * randn(1,numel(xPNRZ_filtrado_f2));
var(Noise2) % Debe ser igual al valor de PNoise considerado
SNR_dB_2 = 10*log10(1 / PNoise);


Rx_Signal_AWGN_hs = xPNRZ_filtrado_f1 + Noise ;
Rx_Signal_AWGN_rec = xPNRZ_filtrado_f2 + Noise2 ;

%ejercicio 2
Match_filter1 = fliplr(hs);
Match_filter2 = fliplr(pnrz);

Const11 =conv(Rx_Signal_AWGN_hs,Match_filter1)/mp;
Const22 =conv(Rx_Signal_AWGN_rec,Match_filter2)/mp;
```

TAREA #8
Alexis Luna
Neidys Vargas

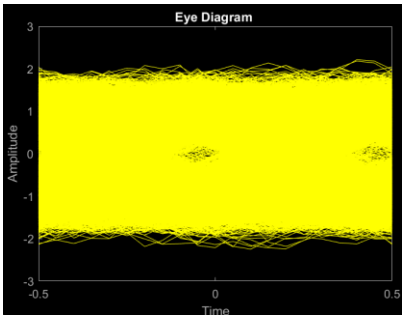
3) De acuerdo con lo que aprendimos del Diagrama de Ojo, la salida del Match Filter (MF) se muestrea en el
momento en que el Ojo está más abierto. Recuerde el significado de la Autocorrelación usando el MF, el cual nos dice que el punto (muestra) ideal donde la señal es máxima, corresponde a la energía del pulso transmitido.

a) Obtenga y analice el diagrama de ojo utilizando eyediagram ( )

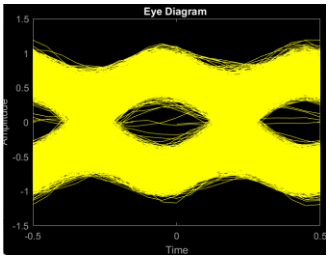
Half sine eye diagram PNOISE =1



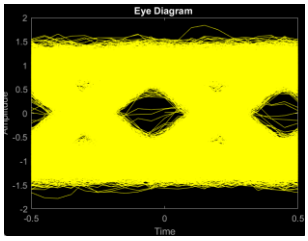
Rectangular eye diagram PNOISE =1



Half sine eye diagram PNOISE = 0.03125*mp



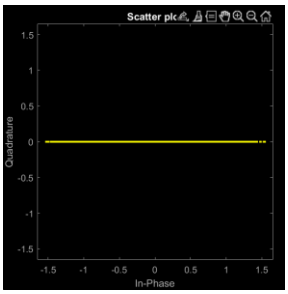
Rectangular eye diagram PNOISE =0.03125*mp



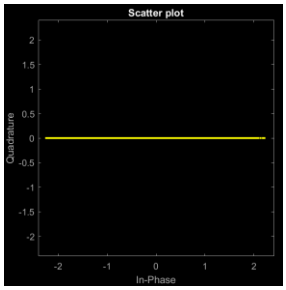
TAREA #8
Alexis Luna
Neidys Vargas

b) Obtenga y analice la constelación recibida utilizando scatterplot ( )

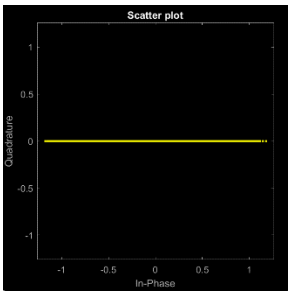
Half sine scatterplot PNOISE = 1



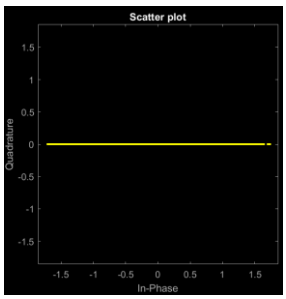
Rectangular scatterplot PNOISE = 1



Half sine scatterplot PNOISE = 0.03125*mp



Rectangular scatterplot PNOISE = 1



Al cambiar la potencia del ruido blanco podemos observar que depende de la señal, el diagrama de ojo y la constelación son un poco mas aceptables y por ende, podemos ver su característica. La señal senoidal, es mas apreciable cuando la potencia de ruido es menor.

TAREA #8
Alexis Luna
Neidys Vargas

3. Muestree la señal a la salida del MF de acuerdo con el código de línea utilizado y considerando los retardos correspondientes

```
delay_signal = orden/2 + numel(pnrz)/2; %calculate the signal delay,
channel delay + match filter delay
delay_signal_hs = orden/2 + numel(hs)/2; %calculate the signal delay,
channel delay + match filter delay
start_recovery_count = delay_signal + mp/2;
start_recovery_count_hs = delay_signal_hs + mp/2;

PNRZ_recovery_rec = Const22(start_recovery_count:mp:end);
PNRZ_recovery_hs = Const11(start_recovery_count:mp:end);

PNRZ_recovery_rec = PNRZ_recovery_rec(1:numel(bits2send));
PNRZ_recovery_hs = PNRZ_recovery_hs(1:numel(bits2send));

bits_RX_PNRZ_1 = zeros(1,numel(bits2send));
bits_RX_PNRZ_2 = zeros(1,numel(bits2send));

xPNRZ_f_m1 = PNRZ_recovery_hs;
xPNRZ_f_m2 = PNRZ_recovery_rec;
```

4) Convierta las muestras de los símbolos a Bits recibidos

5) Calcule el BER para ese valor de SNR_dB

7) Repita el ejercicio a partir del punto (1) cambiando el valor de PNoise de tal manera que el valor de SNR_dB varíe de 0: 3: 30 dB. Recuerde que la potencia de la señal recibida se mantiene en 1 Watt.

8) Al final del ejercicio haga una gráfica que muestre el BER para cada valor de SNR_dB utilizado.

9) Realice un análisis de los resultados de los experimentos y comente sus conclusiones.

```
PNRZ_recovery_rec = PNRZ_recovery_rec(1:numel(bits2send));
PNRZ_recovery_hs = PNRZ_recovery_hs(1:numel(bits2send));

bits_RX_PNRZ_1 = zeros(1,numel(bits2send));
bits_RX_PNRZ_2 = zeros(1,numel(bits2send));

xPNRZ_f_m1 = PNRZ_recovery_hs;
xPNRZ_f_m2 = PNRZ_recovery_rec;

bits_RX_PNRZ_1(xPNRZ_f_m1 > 0) = 1;    %Umbral +
bits_RX_PNRZ_2(xPNRZ_f_m2 > 0) = 1;    %Umbral +

xPNRZ_BER_1_hs = (sum(xor(bits2send,bits_RX_PNRZ_1))/numel(bits2send)) *
100
xPNRZ_BER_2_rec = (sum(xor(bits2send,bits_RX_PNRZ_2))/numel(bits2send)) *
100
```

DB	PNOISE	BER HS	BER REC
0	1*mp	15.8025	15.8704
3	0.5*mp	7.9136	8.2778
6	0.25*mp	2.2284	2.5062
9	0.125*mp	0.2716	0.2222
12	0.0625*mp	0	0
15	0.03125*mp	0	0

TAREA #8
Alexis Luna
Neidys Vargas
Codigo utilizado

```
load lena512.mat
%lenarec=lena512(252:379,318:445); 128x128
%lenarec=lena512(252:338,318:404);%87x87
lenarec=lena512(252:296,318:362); %45x45
figure(1);
imshow(uint8(lenarec))
title("Lena's eye");
b = de2bi(lenarec,8,'left-msb'); % Convert the samples from uint8 to binary
b = b';
bits = b(:);

bits2send = bits';
Fs= 96000;
mp= 10;
%No_bits= 8712;
Baud_rate= Fs/mp   %Symbols per second
Bit_rate=Baud_rate % bits/s
Ts=1/Fs;
% The bit rate is Rb= Rs= Fs / mp, because 1 bit= 1 symbol and  every symbol has
mp
% samples per bit


n= 0:(mp-1);
w = pi/mp;
hs = sin(w*n);

pnrz=ones(1,mp);

s1=bits;
s1(s1==0)=-1;
s=zeros(1,numel(s1)*mp);
s(1:mp:end)=s1; %Impulse train
stem(s(1:mp*16))

xPNRZ_hs=conv(hs,s); %Pulse half sine
xPNRZ=conv(pnrz,s); %Pulse rectangular



figure();
pow = sum(xPNRZ_hs.^2)/numel(xPNRZ_hs);
pow_Deseada = 1;
xPNRZ_hs = sqrt(pow_Deseada/pow)*xPNRZ_hs;
%eyediagram(xPNRZ_hs,2*mp);
figure();
%plot(xPNRZ(1:mp*16))
pow = sum(xPNRZ.^2)/numel(xPNRZ);
pow_Deseada = 1;
xPNRZ = sqrt(pow_Deseada/pow)*xPNRZ;
%eyediagram(xPNRZ,2*mp);



orden=60;     % Orden del Filtro
f=   [0 0.6 0.6 1];  % Vector de Frecuencias
m= [1 1 0 0]; % Vector de Magnitudes
f1 = fir2(orden,f,m);  % Coeficientes del Filtro usando FIR2( )
%fvtool(f1);

xPNRZ_filtrado_f1 = conv(xPNRZ_hs,f1);
%eyediagram(xPNRZ_filtrado_f1,2*mp);
```

TAREA #8
Alexis Luna
Neidys Vargas

```
xPNRZ_filtrado_f2 = conv(xPNRZ,f1);
%eyediagram(xPNRZ_filtrado_f2,2*mp);


pow1 = sum(xPNRZ_filtrado_f1.^2)/numel(xPNRZ_filtrado_f1);
pow_Deseada = 1;
xPNRZ_filtrado_f1 = sqrt(pow_Deseada/pow1)*xPNRZ_filtrado_f1;

pow2 = sum(xPNRZ_filtrado_f2.^2)/numel(xPNRZ_filtrado_f2);
pow_Deseada = 1;
xPNRZ_filtrado_f2 = sqrt(pow_Deseada/pow2)*xPNRZ_filtrado_f2;


%PNoise = N02 * sum(h.*h) * Ts % Observe que PNoise es un escalar
PNoise = 1;
%PNoise = PNoise*mp;

Noise = sqrt(PNoise) * randn(1,numel(xPNRZ_filtrado_f1));
var(Noise) % Debe ser igual al valor de PNoise considerado
SNR_dB = 10*log10(1 / PNoise);

Noise2 = sqrt(PNoise) * randn(1,numel(xPNRZ_filtrado_f2));
var(Noise2) % Debe ser igual al valor de PNoise considerado
SNR_dB_2 = 10*log10(1 / PNoise);


Rx_Signal_AWGN_hs = xPNRZ_filtrado_f1 + Noise ;
Rx_Signal_AWGN_rec = xPNRZ_filtrado_f2 + Noise2 ;

%ejercicio 2
Match_filter1 = fliplr(hs);
Match_filter2 = fliplr(pnrz);

Const11 =conv(Rx_Signal_AWGN_hs,Match_filter1)/mp;
Const22 =conv(Rx_Signal_AWGN_rec,Match_filter2)/mp;


delay_signal = orden/2 + numel(pnrz)/2; %calculate the signal delay, channel
delay + match filter delay
delay_signal_hs = orden/2 + numel(hs)/2; %calculate the signal delay, channel
delay + match filter delay
start_recovery_count = delay_signal + mp/2;
start_recovery_count_hs = delay_signal_hs + mp/2;

PNRZ_recovery_rec = Const22(start_recovery_count:mp:end);
PNRZ_recovery_hs = Const11(start_recovery_count:mp:end);

PNRZ_recovery_rec = PNRZ_recovery_rec(1:numel(bits2send));
PNRZ_recovery_hs = PNRZ_recovery_hs(1:numel(bits2send));

bits_RX_PNRZ_1 = zeros(1,numel(bits2send));
bits_RX_PNRZ_2 = zeros(1,numel(bits2send));

xPNRZ_f_m1 = PNRZ_recovery_hs;
xPNRZ_f_m2 = PNRZ_recovery_rec;

bits_RX_PNRZ_1(xPNRZ_f_m1 > 0) = 1;   %Umbral +
bits_RX_PNRZ_2(xPNRZ_f_m2 > 0) = 1;   %Umbral +

xPNRZ_BER_1_hs = (sum(xor(bits2send,bits_RX_PNRZ_1))/numel(bits2send)) * 100
xPNRZ_BER_2_rec = (sum(xor(bits2send,bits_RX_PNRZ_2))/numel(bits2send)) * 100

%bertool()
```