6 Sistema de Comunicación Digital Lineal

Una vez que hemos estudiado en la Práctica 3 el Test de Hipótesis, en la Práctica 4 el transmisor lineal y en la Práctica 5 el demodulador óptimo en un canal AWGN, agruparemos todo lo anterior diseñando un sistema de Comunicación Digital Lineal, evaluando sus prestaciones. El aspecto clave de esta evaluación consiste en:

1. Calcular de forma experimental la probabilidad de error de bit y de símbolos de los sistemas que estudiemos.

2. Estudiar cómo esas probabilidades se modifican en función de la energía de las señales que se transmiten y de la modulación que se utilice.

3. Comparar los resultados experimentales con los teóricos.

6.1 Sistema de comunicación digital binario

La simulación de un sistema de comunicación binario se muestra en la Figura 6.1. Los bloques Tx y Dm representan, respectivamente, el transmisor y el demodulador óptimo.

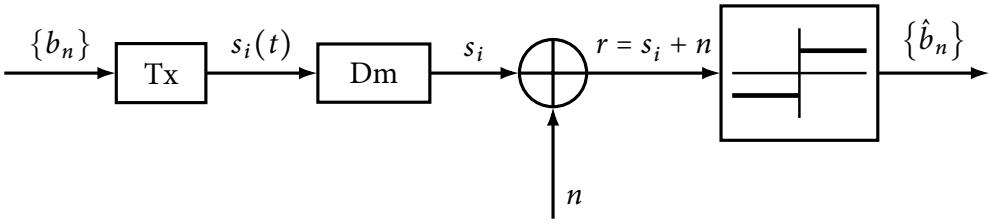

Figura 6.1 Simulación de un sistema de Comunicación Digital Binario.

En el siguiente código se desarrolla en MATLAB/Octave el esquema anterior, utilizando la función `transmisorpam` para generar una señal binaria antipodal, paso de baja en el código mostrado. De manera resumida:

1. Se genera la señal de información digital.
2. Se representa la constelación de las señales transmitidas.
3. Se representa el valor práctico de la probabilidad de error para los distintos valores de la varianza del ruido y se compara con el valor teórico.

Código 6.1 Sistema de comunicación digital binario.

```
% Dado un conjunto de parámetros, se genera la secuencia de información digital,
      se representa la constelación de las señales transmitidas, se obtiene la
     probabilidad de error teórica y la experimental y se representan ambas.
     Llama a transmisorpam para generar la secuencia.

% Nb = Nº de bits a transmitir
% Bn = Secuencia de dígitos binarios
% Eb = Energía media transmitida en Julios
% L = Nº de puntos que vamos a utilizar para transmitir un bit

clear all
close all

Nb=2^16;
Eb=9;
M=2;
L=4;

% Definición del pulso básico.
p=ones(1,L);

Bn=randi(2,1,Nb)-1;

% Obtiene la secuencia transmitida y el pulso normalizado
[Xn,Bn,An,phi,alfabeto]=transmisorpam(Bn,Eb,M,p,L);

% Actualiza el valor de Nb
Nb=length(Bn);

% Respuesta impulsiva del filtro adaptado
hr = fliplr(phi);

% Salida del filtro adaptado
yn = conv(hr,Xn);

% El resultado de muestrear la señal anterior
sn=yn(L:L:Nb*L);

% La secuencia a la salida en el caso de utilizar un demodulador de correlación
sn=phi*reshape(Xn,L,Nb);
```

```
% Obtiene valores únicos
sx=unique(sn);
xmax=max(sx)+max(sx)/10+0.5;
xmin=-(abs(min(sx))+abs(min(sx))/10)-0.5;

% Pinta los ejes
figure(1)
plot([xmin,xmax],[0,0],'b-');
hold on

% Representa la constelación
plot(sx,zeros(1,length(sx)), 'r*')
grid on

% Distancia mínima de la constelación
dmin = abs(sx(1)-sx(2));

% Definamos la relación señal ruido en dB
SNRdb = 0:1:20;

% Inicialización
correctos=zeros(1,Nb);
errorsimulado=zeros(size(SNRdb));

for ii=1:length(SNRdb)
    SNR = 10^(SNRdb(ii)/10); % relación señal ruido en unidades naturales
    varzn = Eb/(2*SNR);

    % El ruido a la entrada del detector
    zn = sqrt(varzn)*randn(size(sn));

    % El vector de observación a la entrada del detector
    rn = sn + zn;

    % El umbral está situado en sx(1)+sx(2))/2
    correctos = rn>(sx(1)+sx(2))/2==Bn;
    conterror = Nb-sum(correctos);
    errorsimulado(ii)=conterror/Nb;

end

% La representación del error
SNRdb2 = 0:0.1:20;
errorteorico=zeros(size(SNRdb2));

for ii=1:length(SNRdb2),
   SNR = 10^(SNRdb2(ii)/10);% relación señal ruido en unidades naturales

   % Error teórico de la constelación antipodal
   errorteorico(ii)=Qfunct(sqrt(2*SNR));

end

% Representación de la probabilidad de error
figure(2)
semilogy(SNRdb,errorsimulado,'r*');
```

```
hold on
semilogy(SNRdb2,errorteorico,'k-');
axis([1 16 10^(-6) 1])

grid on
xlabel('Relación señal/Ruido E_b/N_0 (dB)'), ...
   ylabel('Probabilidad de error de bit (P_b)')

% Crea la leyenda
legend({'Simulado','Teórico'},'Location','Best');
```

En la Figura 6.7 se muestran la constelación generada por el anterior código y en la Figura 6.3 las probabilidades de error teórico y experimental.

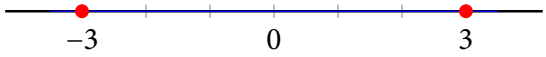

Figura 6.2 Constelación binaria antipodal.

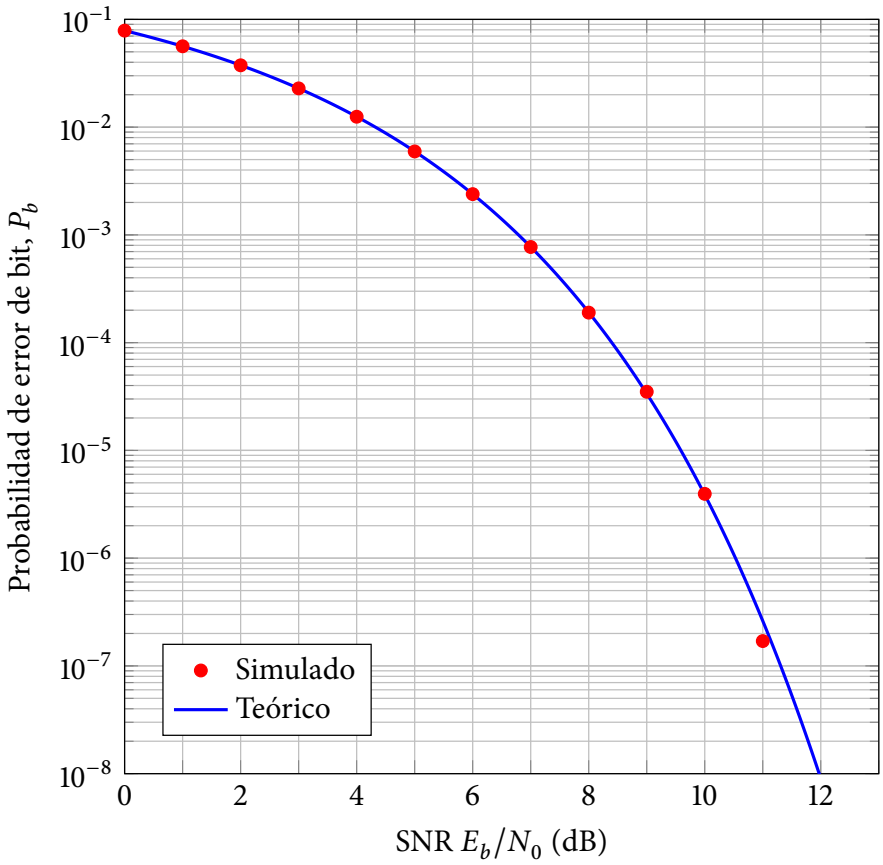

Figura 6.3 Probabilidad de error de un sistema binario antipodal.

En lugar de detenernos en el Código 6.1 para explicar algunos de sus características, vamos a continuar con un sistema algo más complejo y completo, un sistema M-PAM que engloba al anterior como un caso particular.

6.2 Sistema de comunicación digital M-PAM

La diferencia básica entre el sistema binario que acabamos de ver y el caso M-PAM la encontramos en la asignación secuencia de bit → símbolo → niveles de amplitud que se realiza en la función `transmisorpam` para $M > 2$ y la necesidad por tanto de realizar una asignación inversa en el receptor. Además, el detector no puede ser un detector de umbral como el que acabamos de ver, puesto que existen varios umbrales para distinguir entre los M símbolos transmitidos.

El sistema simulado se muestra en la Figura 6.4 en el que hemos querido destacar la existencia de la secuencia de niveles de amplitud que transmitimos $\{A_n\}$ y la secuencia que detectamos $\{\hat{A}_n\}$. En el receptor tenemos que tener en cuenta cómo se había hecho la asignación anterior para proceder a realizar la asignación inversa niveles de amplitud detectados → secuencia de bits detectados. Comparando las secuencias de bits y símbolos transmitidos y detectados podremos calcular la probabilidad de error experimental de bit y símbolo del sistema M-PAM.

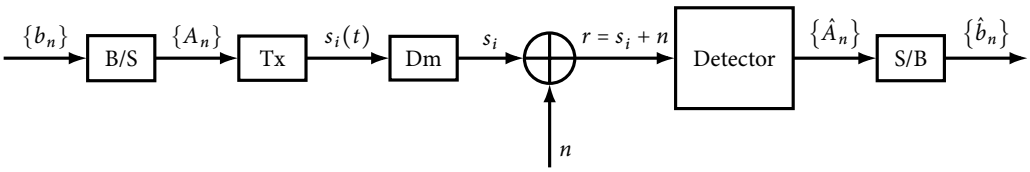

Figura 6.4 Sistema de comunicación digital M-PAM en banda base simulado.

Para realizar la simulación de este sistema definimos una función en MATLAB/Octave que detecta el nivel de amplitud/símbolo transmitido, a partir del vector de observación y otra función que partiendo de este nivel, obtiene la secuencia de bits correspondiente. La primera de estas funciones, que hemos llamado `detecta` se define en el siguiente código.

Código 6.2 Función detecta.

```
function [An] = detecta(rn, alfabeto)
% rn      = una secuencia de símbolos más ruido
% alfabeto = tabla con los niveles de amplitud/símbolos
%
% Genera:
% An = una secuencia de símbolos pertenecientes al alfabeto de acuerdo con
% una regla de distancia euclidiana mínima (mínima distancia)

% Longitud de la secuencia
N=length(rn);

% Inicializa
An=zeros(1,N);

for i=1:N
    [~, ind]=min(abs(rn(i)-alfabeto));
    An(i)=alfabeto(ind);
end
```

En el siguiente código presentamos una versión diferente de esta función sin utilizar el bucle `for`. Puede comprobarse con facilidad que esta función puede llegar a ser 20 veces más rápida que la anterior.

Código 6.3 Función detecta Sin el Bucle For.

```
function [An] = detectaSBF(rn,alfabeto)
% function [An] = detectaSBF(rn,alfabeto)
%
% rn      = una secuencia de símbolos más ruido
% alfabeto = tabla con los niveles de amplitud/símbolos
%
% Genera:
% An = una secuencia de símbolos pertenecientes al alfabeto de acuerdo con
% una regla de distancia euclidiana mínima (mínima distancia)

% Genera una repetición del alfabeto
alfabeto_repetido=repmat(alfabeto',1,length(rn));

% Genera una repetición de la secuencia del vector de observación
rn_repetido=repmat(rn,length(alfabeto),1);

% Obtiene el índice respecto al alfabeto
[~,ind]=min(abs(rn_repetido-alfabeto_repetido));

% Genera la secuencia de niveles detectados
An=alfabeto(ind);
```

Una vez encontrada la secuencia de símbolos, es necesario obtener los bits correspondientes, acción que hemos realizado con la llamada a una función denominada `simbolobit`. Una implementación de esta función se muestra en el código siguiente:

Código 6.4 Función simbolobit.

```
function [Bn] = simbolobit(An,alfabeto)
% function [Bn] = simbolobit(An,alfabeto)
% An      = secuencia de símbolos pertenecientes al alfabeto
% alfabeto = tabla con los símbolos utilizados en la transmisión
% Bn      = una secuencia de bit, considerando que los símbolos se habían
% generado siguiendo una codificación de Gray

% ¿Cuántos bits hay en cada símbolo?
k=log2(length(alfabeto));

if k>1
    distancia = abs(alfabeto(1)-alfabeto(2));
    indices  = round((An-alfabeto(1))/distancia);
    Bn       = reshape(de2gray(indices,k)',1,k*length(An));
else
    Bn=((An/max(alfabeto))+1)/2;
end
```

En esta función se hace una llamada a la función `de2gray` que se ha escrito adaptando una versión de Mathworks. Su código es el siguiente:

Código 6.5 Función de2gray.

```
function [g] = de2gray(d,n )

% Convierte un número decimal en un vector binario de longitud n
% Versión adaptada de una función de Mathworks

% Comprobaciones
mensaje='Sólo sirve para números enteros positivos';
d = d(:);
len_d = length(d);
if min(d) < 0
    error(mensaje);
elseif ~isempty(find(d==inf,1))
  error('This functions can not take Inf as the input.');
elseif find(d ~= floor(d))
  warning(mensaje);
end;

% assign the length
if nargin < 2;
    tmp = max(d);
    b1 = [];
    while tmp > 0
        b1 = [b1 rem(tmp, 2)];
        tmp = floor(tmp/2);
    end;
    n = length(b1);
end;

% initial value
b = zeros(len_d, n);

% parameter assignment
for i = 1 : len_d
    j = 1;
    tmp = d(i);
    while (j <= n) && (tmp > 0)
        b(i, j) = rem(tmp, 2);
        tmp = floor(tmp/2);
        j = j + 1;
    end;
end;

% los bits en el orden que utilizamos
b=fliplr(b);

g(:,1) = b(:,1);
for i = 2:size(b,2),
    g(:,i) = xor( b(:,i-1), b(:,i) );
end
```

Para mostrar las prestaciones del sistema PAM diseñado, en lo referente a cómo se modifica la probabilidad de error de símbolo y la probabilidad de error de bit equivalente cuando cambia la relación señal/ruido, podemos diseñar un sencillo script que muestre

estos valores. Además, nos conviene observar cómo evolucionan a medida que M se modifica. Las siguientes líneas de código cumplen con este objetivo y el resultado se muestra en las figuras 6.5 y 6.6.

Código 6.6 Sistema de comunicación digital M-PAM.

```
clear all
close all

L=4;
Nb=1e6;
Eb=9;

% definamos el pulso básico
p=ones(1,L);

codigocolor = [0 0 0;
               0  1  0;
               0  0  1;
               1  0  0;
               1  0  1;
               0  1  1];
jj=0;

for M=[2,4,8,16,32]
    jj=jj+1;
    Bn=randi(2,1,Nb)-1;
    [Xn,Bn,An,phi,alfabeto]=transmisorpam(Bn,Eb,M,p,L);

    Nb=length(Bn);  %Número de bits transmitidos
    Ns=length(An);   %Número de símbolos transmitidos

    % Demodulación mediante correlador
    sn=phi*reshape(Xn,L,Ns);

    % Definamos la relación señal ruido en dB y otros elementos
    SNRdb = 0:1:20;
    errorsimulados=zeros(size(SNRdb));
    errorsimuladob=zeros(size(SNRdb));

    for ii=1:length(SNRdb)
        SNR = 10^(SNRdb(ii)/10);
        varzn = Eb/(2*SNR);

        % El ruido a la entrada del detector
        zn = sqrt(varzn)*randn(size(sn));

        % El vector de observación a la entrada del detector
        rn = sn + zn;

        % Los símbolos detectados
        Andetectado = detectaSBF(rn, alfabeto);

        % Los bits correspondientes
        Bndetectado = simbolobit(Andetectado, alfabeto);
```

```
        % Contemos los errores
        conterrors = Ns-sum(Andetectado==An);
        errorsimulados(ii)=conterrors/Ns;

        conterrorb = Nb-sum(Bndetectado==Bn);
        errorsimuladob(ii)=conterrorb/Nb;

    end
    color=codigocolor(jj,1:end);

    % Representación de la probabilidad de error de símbolo
    figure(1)
    semilogy(SNRdb,errorsimulados,'r*');
    hold on

    % Para la representación del error teórico
    SNRdb2 = 0:0.01:20;
    errorteorico=zeros(size(SNRdb2));

    for ii=1:length(SNRdb2),
        SNR = 10^(SNRdb2(ii)/10);% relación señal ruido
        errorteorico(ii)=(2*(M-1)/M)*...
            Qfunct(sqrt((6*log2(M)/(M^2-1))*SNR));...
        echo off;
    end

    semilogy(SNRdb2,errorteorico,'-k','Color',color, 'LineWidth', 1);
    axis([0,20,1e-7,1])
    xlabel('Relación Señal/Ruido E_b/N_0 (dB)');
    ylabel('Probabilidad de error de símbolo (P_M)');
    grid on

    % Representación de la probabilidad de error de bit
    figure(2)
    semilogy(SNRdb,errorsimuladob,'r*');
    hold on

    % Para la representación del error teórico
    SNRdb2 = 0:0.01:20;
    errorteorico=zeros(size(SNRdb2));

    for ii=1:length(SNRdb2),
        SNR = 10^(SNRdb2(ii)/10);% relación señal ruido
        errorteorico(ii)=(2*(M-1)/(M*log2(M)))*...
            Qfunct(sqrt((6*log2(M)/(M^2-1))*SNR));...
        echo off;
    end

    semilogy(SNRdb2,errorteorico,'-k','Color',color, 'LineWidth', 1);
    axis([0,20,1e-7,1])
    xlabel('Relación Señal/Ruido E_b/N_0 (dB)');
    ylabel('Probabilidad de error de bit equivalente (P_b)');
    grid on
end
```

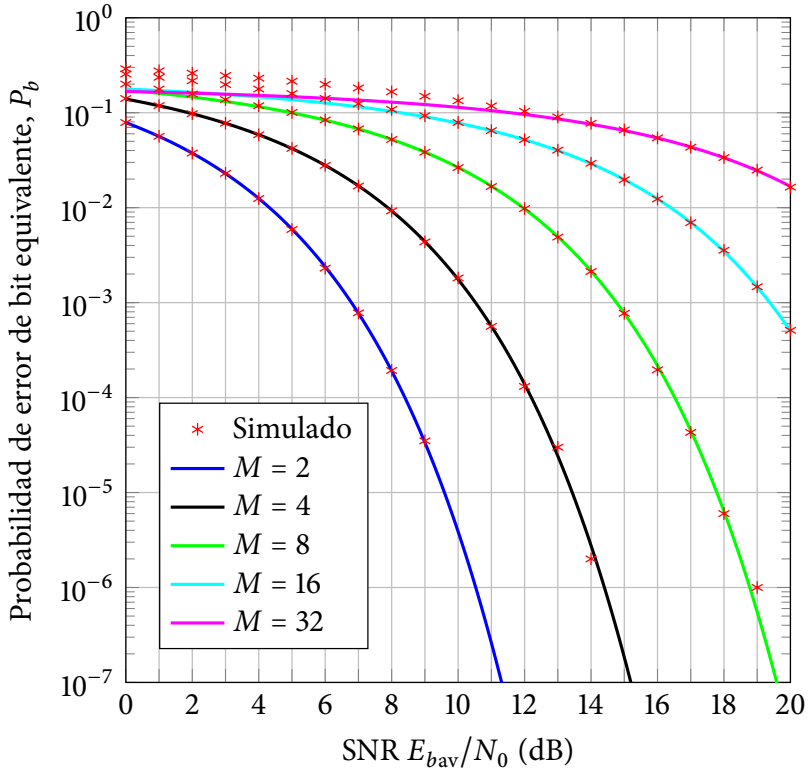

Figura 6.5 Probabilidad de error de bit equivalente M-PAM.

En el Código 6.6 se puede utilizar para la detección la función `detecta`, (Código 6.2) o `detectaSBF` (Código 6.3). Es fácil observar que en estas funciones se realiza una comparación de cada uno de los valores del vector observación `rn` con el alfabeto que estamos transmitiendo y se elige como símbolo transmitido aquel que se encuentre a una menor distancia del vector de observación; esto es, se realiza una detección de distancia mínima que en el caso AWGN se corresponde con un detector óptimo.

Una vez detectado el símbolo, en el Código 6.6 hacemos uso de la función `simbolobit`, definida en el Código 6.4 para obtener la secuencia de bit correspondiente al mismo. Debemos tener en cuenta que esta asignación símbolo → bit no es arbitraria puesto que en la asignación bit → símbolo habíamos utilizado una codificación de Gray. Por lo tanto, hemos escrito una función, llamada `de2gray` que se utiliza en la asignación descrita en el Código 6.4.

Destacar por último la estrecha relación entre los resultados de la simulación y los teóricos, como se pone de manifiesto en las gráficas de las figuras 6.5 y 6.6, para las cuales los valores teóricos vienen dados por:

$$P_M = \frac{2(M-1)}{M} Q\left(\sqrt{\frac{6\log_2 M}{M^2-1}\frac{E_{bav}}{N_0}}\right) \tag{6.1}$$

$$P_b = \frac{P_M}{\log_2 M} \tag{6.2}$$

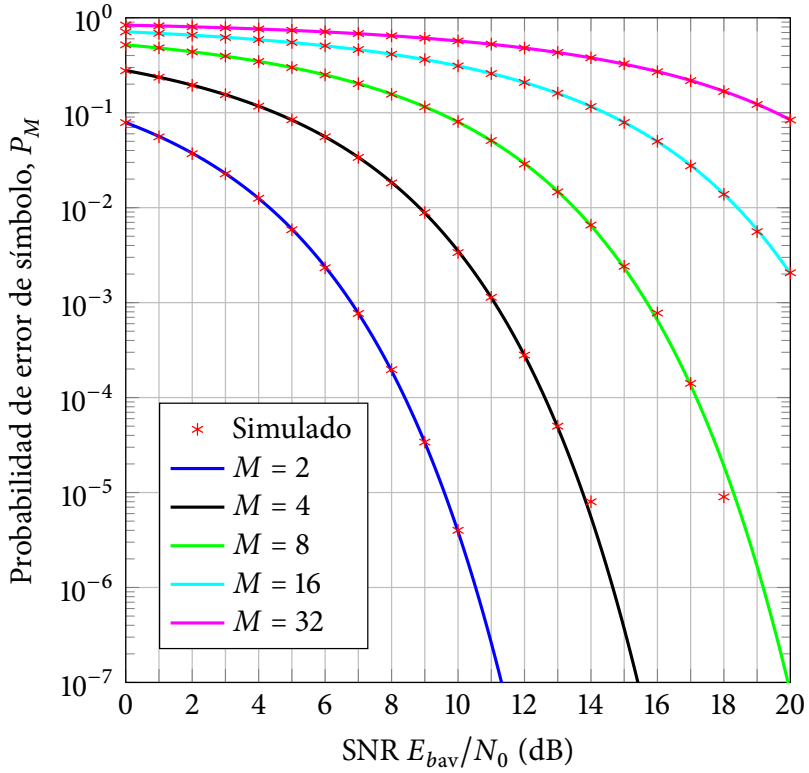

Figura 6.6 Probabilidad de error de símbolo M-PAM.

Ejercicios propuestos

6.1 Utilizando el Código 6.1, diseñar un sistema transmisor binario paso de banda en la que el pulso básico utilizado venga dado por:

```
p=cos(4*pi*(0:L-1)/L);
```

6.2 Utilizando la función `transmisorbinario` diseñada en el Ejercicio 4.3 para generar la señal transmitida en un sistema binario arbitrario, realizar la simulación de un sistema de comunicación digital binario arbitrario en banda base. En concreto:

a) Definir un demodulador de correlación.

b) Definir un demodulador mediante un filtro adaptado a la señal diferencia.

c) Representar la constelación de las señales transmitidas.

d) Representar la constelación de las señales a la entrada del detector.

e) Encontrar la distancia mínima de la constelación.

f) Encontrar la probabilidad de error de esta modulación binaria arbitraria en función de la energía promedio transmitida y comparar con la modulación binaria antipodal.

Para los pulsos básicos se pueden utilizar, por ejemplo, $s_1(t)$ un pulso rectangular de duración T_b y $s_2(t)$ un pulso rectangular distinto de cero únicamente entre 0 y $T_b/2$. Suponer que la energía por bit transmitida es la misma.

6.3 Diseñar una función en la que, dadas dos señales arbitrarias, con energías E_{b1} y E_{b2}, podamos obtener:

a) La constelación del sistema.

b) La distancia mínima de la constelación.

c) El coeficiente de correlación entre las señales.

d) La probabilidad de error de bit en función de la energía promedio transmitida.

e) La probabilidad de error de bit de un sistema antipodal con la misma energía promedio.

6.4 Dado un sistema binario ortogonal igualmente probable, si llamamos **r** al vector de observación, con los vectores señal dados por:

$$\mathbf{s}_1 = \begin{bmatrix} \sqrt{E_b} \\ 0 \end{bmatrix}, \quad \mathbf{s}_2 = \begin{bmatrix} 0 \\ \sqrt{E_b} \end{bmatrix}$$

a) Demostrar que la regla de detección óptima puede expresarse en la forma:

$$\hat{X}(y_{\text{obs}}) = x_i \Leftrightarrow r_i > r_j, \quad i,j = 1,2$$

b) Basándonos en la regla anterior, diseñar un receptor óptimo.

c) Comprobar que la probabilidad de error teórica y experimental coinciden.

6.5 Dada la analogía existente entre un transmisor M-PAM y un transmisor M-PSK, diseñar un detector para un sistema M-PSK en un canal AWGN, obteniendo la probabilidad de error de símbolos y de bit.

6.6 Comparar las prestaciones de un sistema M-PAM y un sistema M-PSK para diversos valores de M.

Ejercicio Práctica 6 (Básico)

Basándonos en el sistema de comunicación PAM banda base diseñado en la Sección 6.2, en este Ejercicio Básico queremos diseñar un sistema de comunicación PAM paso de banda M-ario en un canal AWGN y comprobar sus prestaciones para diversos valores de M y relación señal/ruido. En concreto:

1. Diseñar un transmisor paso de banda mediante la función `transmisorpam`, cuando:
   - $N_b = 10^6$.
   - $M = 8$.
   - $E_b = 8$ Julios.
   - $L = 32$.
   - El pulso conformador se muestra en la Figura 6.7.
   - Frecuencia portadora $\omega_c = 8\pi/T$.
   - $R_b = 10^4$ bps.
2. Representar la señal discreta transmitida para los primeros 10 símbolos generados.
3. Representar la señal continua transmitida para los primeros 10 símbolos generados.
4. Determinar la energía de la señal transmitida. ¿Coincide la energía media por bit transmitida con la elegida en el diseño? Razonar la posible discrepancia.
5. Demodular la señal transmitida cuando se utiliza un demodulador que utiliza un único correlador.
6. Representar la respuesta impulsiva discreta de un filtro adaptado que pudiera utilizarse como parte de un demodulador.
7. Determinar la secuencia de niveles de amplitud a la entrada del detector.
8. Determinar la varianza del ruido presente a la entrada del detector cuando la relación $E_{b\text{av}}/N_0 = 16$ dB. ¿Cuáles serían los umbrales de detección MAP?

9. La probabilidad de error de símbolo P_M y la probabilidad de error de bit equivalente P_b pueden obtenerse a partir de las ecuaciones (6.1) y (6.2). Representarlas frente a $E_{b\text{av}}/N_0$ medido en decibelios cuando $E_{b\text{av}}/N_0$ varía entre 0 y 20 dB.

10. Utilizando una secuencia de $N_b = 10^6$ bits transmitidos, comprobar experimentalmente que los valores de P_b obtenidos concuerdan con los teóricos cuando la relación $E_{b\text{av}}/N_0$ varía entre 0 y 20 dB.

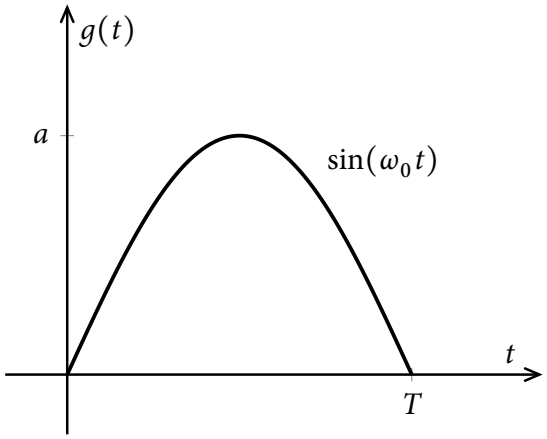

Figura 6.7

Ejercicio Práctica 6

Basándonos en el receptor PAM diseñado, en este ejercicio queremos diseñar un sistema 32-QAM rectangular en banda base y obtener experimentalmente la probabilidad de error de bit equivalente de la misma. Debido a su posible complejidad, vamos a escribir un guión de cómo podríamos realizar este diseño.

1. Generar la señal QAM mediante la función `transmisorqam`. En concreto, los parámetros de generación podrían ser:
   - **a)** $M_1 = 4$, $M_2 = 8$.
   - **b)** $E_{b\text{av}} = 4$ J.
   - **c)** $N_b = 10^6$.
   - **d)** $p_1(t)$ y $p_2(t)$ mostrados en la Figura 6.8.
   - **e)** $L = 4$.

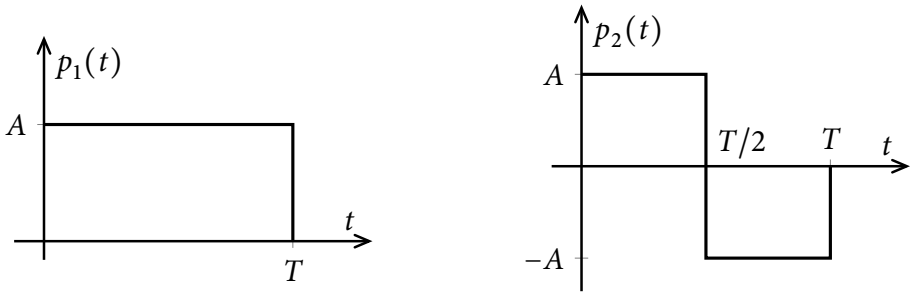

Figura 6.8

2. El receptor estará formado por dos receptores PAM diseñados mediante correladores.
3. Si llamamos $\{\hat{A}_{nI}\}$ a la secuencia de símbolos detectados en la componente en fase y $\{\hat{A}_{nQ}\}$ a los detectados en la componente en cuadratura, hacer uso de la función `simbolobit` para obtener las correspondientes secuencias de bits de las componentes en fase y cuadratura, $\{\hat{B}_{nI}\}$ y $\{\hat{B}_{nQ}\}$, respectivamente.
4. La secuencia de bit detectada debe obtenerse intercalando las secuencias de bit anteriores. Esta operación puede realizarse con facilidad mediante el uso de la función

que hemos denominado une, que realiza la operación simétrica a la función split. Una posible realización de esta función une se muestra en el siguiente código.

Código 6.7 Función une.

```
function [Bndetectado]=une(BndetectadoI,BndetectadoQ,M1,M2)
% [Bndetectado]=une(BndetectadoI,BndetectadoQ,M1,M2)
% BndetectadoI = una secuencia de símbolos binarios correspondientes a los
      bits en posiciones múltiplos de k1
% BndetectadoQ = una secuencia de símbolos binarios correspondientes a los
      bits en posiciones múltiplos de k2
% M1 = nº de símbolos de la componente en fase
% M2 = nº de símbolos de la componente en cuadratura

% Devuelve
% Bndetectado = Los bits entremezclados

k1=log2(M1);
k2=log2(M2);
N=length(BndetectadoI)/k1;
C1=reshape(BndetectadoI',k1,N)';
C2=reshape(BndetectadoQ',k2,N)';
C=[C1,C2];
Bndetectado = reshape(C',N*(k1+k2),1)';
```

5. La probabilidad de error de bit equivalente del sistema QAM se obtendrá comparando la secuencia de dígitos binarios transmitidos y los detectados. A efectos de comparación, sabemos que esta probabilidad de error viene dada por:

$$P_b = \frac{N_e}{\log_2 M_1 + \log_2 M_2} Q\left[\sqrt{\frac{6\left(\log_2 M_1 + \log_2 M_2\right)}{M_1^2 + M_2^2 - 2} \frac{E_{b\text{av}}}{N_0}}\right] \tag{6.3}$$

siendo N_e el número medio de puntos que comparten frontera, que en el caso de una constelación rectangular QAM viene dado por:

$$N_e = 4\left(1 - \frac{1}{2M_1} - \frac{1}{2M_2}\right) \tag{6.4}$$

6. Repetir todos los puntos anteriores utilizando únicamente la función transmisorpam para generar la señal QAM.

7. Repetir todos los puntos anteriores utilizando la función transmisorqamVC diseñada en el Ejercicio 4.8.