Máster Universitario en Ingeniería de Telecomunicación

Equipos y Subsistemas de Comunicaciones

Práctica de Modulaciones Ópticas Avanzadas

Bloque III: Componentes y subsistemas ópticos

Daniel Pastor Abellán

0. Introducción

En esta práctica se abordan las técnicas de modulación ópticas de tipo coherente con diferentes codificaciones BPSK, QPSK, 16QAM, etc. Se imprentará mediante Matlab® un Transmisor óptico basado en doble modulador Mach-Zehnder (MZ) así como un Receptor óptico basado en un acoplador 2x4 MMI (Multi-Mode Interference coupler) y detección diferencial.

Mediante las líneas básicas de código que implementan las funcionalidades de los distintos componentes ópticos el alumno recorrerá todo el proceso de formación de las señales eléctricas, la conformación de pulsos ópticos, la modulación en fase y cuadratura, la transmisión por fibra y finalmente la detección coherente.

Mediante la visualización del diagrama IQ, el diagrama de ojos y el espectro óptico de las señales podrán verificar de forma práctica las características de las distintas modulaciones.

1. Implementación del transmisor

En este primer bloque de la práctica se implementará un transmisor óptico capaz de generar señales ópticas con modulaciones avanzadas en fase y cuadratura (IQ). Partiremos del bloque genérico compuesto por dos moduladores Dual-Drive como el mostrado en la figura 1.

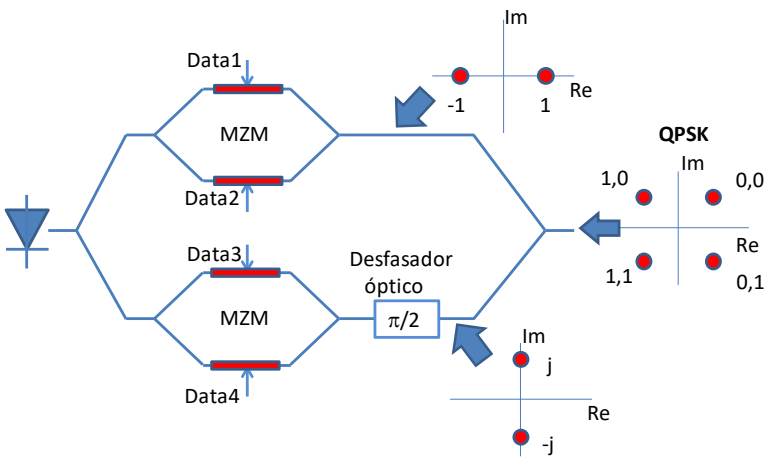

Figura 1.- Estructura del transmisor

Las señales eléctricas de entrada V_1 a V_4 (correspondientes a Data1 a Data4 en la figura) aplicadas a los respectivos puertos de RF modifican las fases ópticas en los brazas de los moduladores MZ (Mach-Zehnder), y podemos utilizar las expresiones teóricas vistas en clase para implementar su funcionamiento en matlab

$$E_s = E_e \exp\left[-j\left(\Delta\phi_1 + \Delta\phi_2\right)/2\right]\cos\left[\frac{\Delta\phi_1 - \Delta\phi_2}{2}\right] \quad (1)$$

Donde $\Delta\phi_i = \frac{\pi V_i}{V_\pi}$ siendo la V_π la tensión que produce un cambio de π radianes en la fase de uno de los brazos del MZM. Las señales ópticas se describirán en la simulación como envolventes complejas (módulo y fase) prescindiendo de la portadora óptica.

Bits y Símbolos:

- Como señal de entrada emplearemos una secuencia digital Pseudoaleatoria (PRBS: Pseudo Random Bit Sequence) de datos generada por una función específica de Matlab® que nos facilita el trabajo.
- Dicha secuencia de bits debe ser dividida en grupos de bits que corresponderán con los "símbolos" a transmitir. Por ejemplo para una codificación QPSK tenemos 2 bits por símbolo, dado que hay 4 posibles símbolos. En una codificación 16QAM tenemos 16 símbolos distintos por lo que cada símbolo corresponderá a grupos de 4 bits.
- Finalmente debemos hacer corresponder a cada símbolo de la codificación empleada su valor IQ, es decir sus valores en el plano complejo IQ (fase y cuadratura).

  Este trabajo lo realiza el siguiente trozo de código. Que podéis copiar directamente del texto. Como ejemplo tenéis ya implementada una codificación OOK (On-Off Keying) y una BPSK (Binary Phase Shift Keying).

C1

```
close all
clear all

% OOK ejemplo
M=1;  %%% bits por símbolo
V=[1 0];
V=V/max(real(V));

%BPSK ejemplo
% M=1;
% V=[-1 1];
% V=V/max(real(V));

%%%%% parametros básicos de la simulación
N_simbols=512;
N_bits=N_simbols*M;
data=idinput(N_bits,'PRBS',[0 1],[0 1])';

puntos_por_simbolo=100;
B=10e9;
Tb=1/B;
ts=Tb/puntos_por_simbolo;

puntos=puntos_por_simbolo* N_simbols;
t=(1:1:puntos)*ts;

RZ=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% niveles de I / Q %%%%%%%%%%%%%%%%%%%

signal=zeros(size(t));

for n=1:1:N_simbols

simbol_index=bin2dec(num2str(data(M*(n-1)+1:1:M*(n-1)+M)));
valor_IQ=V(simbol_index+1);

       ----------PARTE A IMPLEMENTAR

vx(n,:)=[vI1 vI2 vQ1 vQ2];

end
```

<u>Cálculo de los valores de tensión V_1-V_4:</u>

Dentro del bucle “for” en el C1, debemos también calcular los valores de tención que hay que aplicar a los cuatro puertos de los dos moduladores MZ. Lógicamente estos valores de tensión dependerán del valor V_π para la tecnología empleada en los moduladores (nosotros tomaremos un valor de V_π=5 volts).

En este punto nos tendremos que fijar en la ecuación (1) que describe el comportamiento, tanto en fase como en amplitud de la respuesta del modulador MZ. Para escoger los niveles de tensión apropiados para cada símbolo IQ debemos tener presente que:

a) El "chirp" generado en la modulación sea cero por lo tanto emplearemos siempre una estrategia PUSH-PULL

b) Nos fijaremos también en que existen tanto valores positivos como negativos de los valores I o Q de cada símbolo.

Conformación de las señales eléctricas $V_1(t)$ a $V_4(t)$

El proceso de construcción de las señales eléctricas se puede llevar a cabo mediante una concatenación de vectores de datos que contienen la "forma" de un símbolo. Por simplicidad tomaremos un valor constante a lo largo del periodo de símbolo para las señales de tensión aplicadas. Posteriormente a la concatenación empleamos la función "conv" de Matlab® para suavizar los flancos de subida y bajada de forma que la señal se aproxime a la que generaría un circuito driver real con una limitación de ancho de banda real.

En este caso podéis emplear el siguiente código directamente copiando.

```
%%%%%%%% construir las señales electicas
e_signal_I1=[];
e_signal_I2=[];
e_signal_Q1=[];
e_signal_Q2=[];

unos=ones(size(1:1:puntos_por_simbolo));
for n=1:1:N_simbols
e_signal_I1=[e_signal_I1 unos*vx(n,1)];
e_signal_I2=[e_signal_I2 unos*vx(n,2)];
e_signal_Q1=[e_signal_Q1 unos*vx(n,3)];
e_signal_Q2=[e_signal_Q2 unos*vx(n,4)];
end
vent=10;
e_signal_I1=conv(e_signal_I1,1+cos(pi*[-
vent:1:vent]/vent),'same');
e_signal_I1=e_signal_I1/sum(1+cos(pi*[-vent:1:vent]/vent));

e_signal_I2=conv(e_signal_I2,1+cos(pi*[-
vent:1:vent]/vent),'same');
e_signal_I2=e_signal_I2/sum(1+cos(pi*[-vent:1:vent]/vent));

e_signal_Q1=conv(e_signal_Q1,1+cos(pi*[-
vent:1:vent]/vent),'same');
e_signal_Q1=e_signal_Q1/sum(1+cos(pi*[-vent:1:vent]/vent));

e_signal_Q2=conv(e_signal_Q2,1+cos(pi*[-
vent:1:vent]/vent),'same');
e_signal_Q2=e_signal_Q2/sum(1+cos(pi*[-vent:1:vent]/vent));
```

C2

Conformación de señales ópticas:

Una vez tenemos calculados los valores de tensión debemos asignar una "forma" a los pulsos transmitidos que transportan la información de "símbolo". Esto es lo que se conoce también como "*pulse carving*". Una posible forma para los pulsos es la RZ_33, es decir "con retorno a cero" y con un 33% de ciclo de trabajo frente al periodo de símbolo. La figura 2 muestra la forma de implementar este tipo de pulsos a nivel práctico. Se emplea un modulador MZ adicional al que se le aplica una señal de RF de frecuencia mitad de la tasa binaria deseada B/2, con una excursión de tensión de pico a pico de V_π y centrado en un máximo de la función de transferencia.

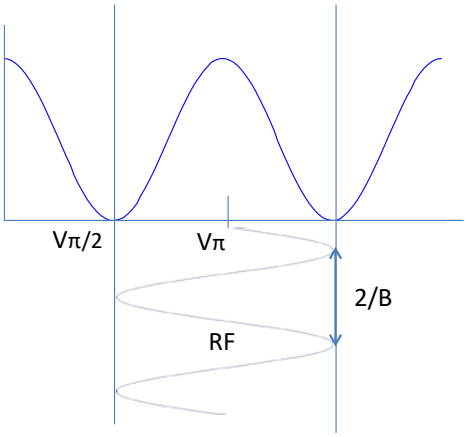

Figura 2.- Conformación de pulsos RZ33

En el simulador podemos tomar la expresión cerrada que describe exactamente el proceso de modulación descrito en la figura 2, esto es:

`%%%%% conformación de los pulsos ópticos` `carver=sin((pi/2)*(1+sin(pi*(t-Tb/2)/Tb)));`	C3

También podríamos transmitir con una codificación NRZ por lo que desactivaríamos las líneas de código del *pulse carving* en el punto adecuado del programa.

EVALUACIÓN DE RESULTADOS

Trabajaremos con una secuencia PRBS de longitud potencia de 2, por ejemplo 512 bits (aunque esto se puede aumentar en función de la velocidad de ejecución). Si bien es más que suficiente para esta práctica.

Para visualizar las señales emplearemos la función (plot (t, ….)) genérica de Matlab® con los argumentos adecuados, o las siguientes funciones ya implementadas que son:

- *IQ_rep: Muestrea la señal en el centro del símbolo, y representa el Diagrama I/Q de las señales aplicadas.*
- *scope_rep: Trocea la señal en los intervalos de símbolo y representa todas las trazas construyendo así el Diagrama de Ojos*
- *osa_rep: Representa el espectro óptico de la señal*

R1) Resultados del apartado 1

Para la modulaciones siguientes: OOK, BPSK, QPSK, 16 QAM, 64QAM

I. ***Obtenga las señales $V_1(t)$ a $V_4(t)$, (zoom temporal de 5-10 símbolos).***
II. ***Obtenga las señales ópticas de salida del transmisor P(t) en "potencia o módulo" (zoom temporal de 5-10 símbolos)***
III. ***Obtenga el diagrama de ojos de la señal***
IV. ***El diagrama IQ***
V. ***El espectro óptico***

Comente en cada caso las características de las diferentes modulaciones y las diferencias entre ellas que considere relevantes, tanto a nivel de señal en el tiempo, espectral, diagrama de ojos o IQ. Pruebe la codificación RZ33 o NRZ y observe las diferencias en el diagrama de ojos y en el espectro.

2. Implementación del receptor

En este segundo apartado completaremos el proceso implementando un receptor óptico para codificaciones coherentes como el mostrado en la figura 3. Está basado en un componente óptico pasivo que proporciona la relación de desfase adecuada y emplea cuatro detectores en configuración diferencial.

Implemente el esquema de la figura 3 partiendo de la señal de datos *s(t)* generada en el apartado anterior hasta obtener las fotocorrientes de salida de cada par de detectores, que corresponderán con las señales I(t) y Q(t) en el dominio electrónico. Suponga un ancho de banda infinito para los detectores por simplicidad (aunque no supone pérdida de generalidad)

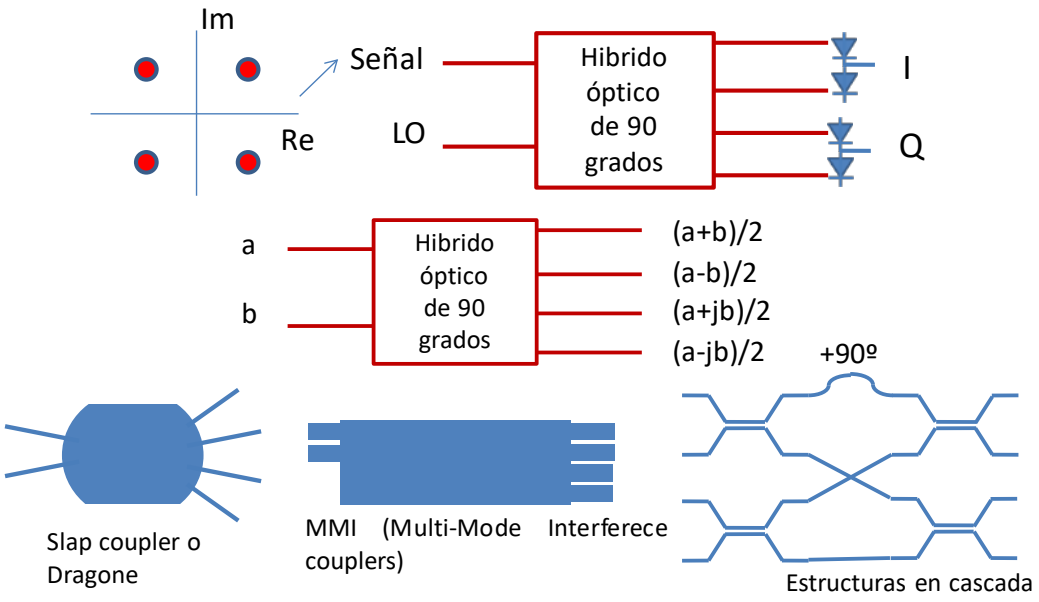

Figura 3.- Esquema del receptor coherente

R2) Resultados del apartado 2

Para la modulaciones siguientes: OOK, BPSK, QPSK, 16 QAM, (64QAM optativo)

I. ***Obtenga las señales $i_1(t)$ a $i_4(t)$ (corrientes detectadas), (zoom temporal de 5-10 símbolos).***
II. ***Obtenga el diagrama de ojos de la señal $i_I(t)$ y $i_Q(t)$***
III. ***El diagrama IQ***
IV. ***¿Observando el diagrama IQ, se consigue recuperar las señales transmitidas?, ¿Cuál es el efecto de la fase del Oscilador Local? (siguiente apartado)***

3. Algunas fuentes de degradación _____

a. Derivas de fase entre oscilador local y señal
b. Efectos de la dispersión cromática
c. Ruido en los detectores

3.a Derivas de fase entre oscilador local y señal

En general, las técnicas de detección coherente que emplean oscilador local (LO) requieren un ajuste preciso entre éste (LO) y la portadora óptica que transporta la señal modulada (s(t)). En el simulador implementado resulta inmediato introducir una diferencia de fase óptica arbitraria entre ambas señales.

I. Inserte una diferencia de fase constante entre s(t) y LO de valor [π/10 , π/4] para el caso de una señal QPSK y comente el resultado en el diagrama IQ de las señales $i_I(t)$ y $i_Q(t)$ recuperadas.

3.b Efecto de la dispersión cromática

Evaluaremos en este punto el efecto de la dispersión cromática sobre la señal. Para ello aplicamos el modelo de la constante de propagación de la fibra como suma de una serie de Taylor donde aparecen los términos de dispersión de 1º y 2º orden β_2 y β_3 respectivamente (2). Supondremos una fibra sin pérdidas (dado que solo afecta como un factor de reducción), y emplearemos su modelo de filtro de fase equivalente $H(w) = \exp(j\beta(w)L)$

$$\beta(w) = \beta_0 + \beta_1(w - w_o) + \frac{1}{2}\beta_2(w - w_o)^2 + \frac{1}{6}\beta_3(w - w_o)^3 \qquad (2)$$

En nuestro simulador *(w-w_0)=dw* será la diferencia de pulsación respecto de la portadora óptica (centro del desarrollo de Tailor). El código en C3 puede servir de orientación para obtener el vector de "dw" a partir del tiempo de muestreo "ts". Además utilizaremos la FFT (Fast Fourier Transform) e IFFT para transformar la señal del dominio temporal al dominio frecuencia y viceversa. También podemos emplear la función "fftshift" con "fft" para obtener un espectro centrado alrededor de la pulsación dw=0 y poder emplear el vector dw tal y como está definido.

C3

```
%%%%%%%%  fibra
p=length(t);
dw=2*pi*((((-p/2)+1:1:p/2)/p)*(1/ts));
...
espectro_salida_fibra=fftshift(fft(salida)).*……
```

```
tras_fibra=ifft(fftshift(espectro_salida_fibra));
```

Obtenga:

I. El diagrama IQ y el diagrama de ojos tras el receptor cuando se intercala entre transmisor y receptor una fibra óptica estándar en 3ª ventana (1550nm) con D=17ps/nmkm (despreciamos S en este caso). Tome varias distancias de fibra: 10, 25, 50 km (hasta observar un deterioro del diagrama IQ) con una tasa de símbolo de 10Gbaud/s. Emplee las codificaciones QPSK y 16QAM.

II. Suponga un sistema óptico que emplea tramos de fibra compensadora de dispersión (DCF) de forma que podemos suponer eliminada la dispersión de 1º orden, quedando en este caso solo la dispersión residual de segundo orden. Suponiendo un valor S=0.08 ps/nm^{2}km, evalué el diagrama IQ y el diagrama de para una velocidad de símbolo B=40GS/s (16QAM) y L=[300-500] km, hasta observar un claro deterioro del diagrama IQ.

Fórmulas

$D=-\frac{2\pi c}{\lambda^2}\beta_2$	$S=\left(\frac{2\pi c}{\lambda^2}\right)^2\beta_3$

3.c Ruido en los detectores

El ruido presente en la fotocorriente lo podemos insertar en el simulador justo tras la detección como una señal aleatoria de media cero. La potencia de ruido (todos los ruidos sumados) la podemos parametrizar mediante una relación de señal a ruido total SNR. Una forma de implementarlo es la indicada en C4. Observe alguno de los diagramas IQ y diagramas de ojos de los apartados anteriores con diferentes valores de SNR.

C4

```
%%%% ruido con SNR

SNR=25;
SNRlin=10.^(SNR/10);
i_media=mean([mean(i1) mean(i2) mean(i3) mean(i4)]);
desv_ruido=sqrt((i_media.^2)/SNRlin);

rt1=(rand(size(t))-0.5*ones(size(t)))*desv_ruido;
rt2=(rand(size(t))-0.5*ones(size(t)))*desv_ruido;
rt3=(rand(size(t))-0.5*ones(size(t)))*desv_ruido;
rt4=(rand(size(t))-0.5*ones(size(t)))*desv_ruido;

i1=i1+rt1;
i2=i2+rt2;
i3=i3+rt3;
i4=i4+rt4;
```

Funciones:

```
%%%%%%%%% Representa la constelación de la señal en el campo
%%%%%%%%% "signal", muestreando la señal
%%%%%%%%% los parámetros de entrada son:
%%%%% t: Vector de tiempo
%%%%% signal: Señal a representar que debe ser real
%%%%% Tb: Tiempo de símbolo
%%%%% puntos: Número de puntos por símbolo
%%%%% N_simbolos: número de simbolos que componen la señal
%%%%% muestreo: permite desplazar el punto de muestreo dentro
%%%%% del intervalo de símbolo, siendo 0.5 para mirad de simbolo
%%%%% y hasta 0 y 1 para ir hacia los extremos

function IQ_rep(t,signal,Tb,N_simbols,muestreo);

signal=signal+1j*1e-9;
t_muestras=Tb*(1:1:N_simbols)-(Tb*muestreo);
sampled=interp1(t,signal,t_muestras);
figure
plot(sampled,'r.')

a=axis;
amp=a(4)-a(3);
dif_t=a(2)-a(1);

amp=max(amp,dif_t);
a(1)=a(1)-amp*0.1;
a(2)=a(2)+amp*0.1;
a(3)=a(3)-amp*0.1;
a(4)=a(4)+amp*0.1;
axis(a);
grid on
```

```
%%%%%%%%% Representa el espectro óptico de la señal introducida
%%%%%%%%% en el campo "signal", los parámetros de entrada son:
%%%%% t: Vector de tiempo
%%%%% signal: Señal a representar que debe ser real
%%%%% Tb: Tiempo de símbolo

function osa_rep(t,signal,Tb);

ts=t(2)-t(1);
p=length(t);
rango=4*(1/Tb)/1e9;
frec=((((-p/2)+1:1:p/2)/p)*(1/ts))/1e9;

figure
Espectro_dB=20*log10(abs(fftshift((fft(signal)))));
Espectro_dB=Espectro_dB-max(Espectro_dB);
plot(frec,Espectro_dB);
```

```
a=axis;
a(1)=-rango;
a(2)=rango;
a(3)=-90;
a(4)=10;
axis(a);
```

```
%%%%%%%%% Representa el diagrama de ojos de la señal que se
%%%%%%%%% introduzca
%%%%%%%%% en el campo "signal", los parámetros de entrada son:
%%%%% t: Vector de tiempo
%%%%% signal: Señal a representar que debe ser real
%%%%% n_ojos: Número de ojos a representas (1 o 2 normalmente)
%%%%% puntos: Número de puntos por símbolo

function scope_rep(t,signal,n_ojos,puntos);

puntos=puntos*n_ojos;

ts=t(2)-t(1);
t_eje=(-(puntos/2)+1:1:puntos/2)*ts;
simbolos=(length(t)/puntos)*n_ojos;

figure

for n=1:1:(simbolos/n_ojos)

t_n=((n-1)*puntos+1:1:(n*puntos));
plot(t_eje,signal(t_n))
hold on
end

a=axis;
amp=(a(4)-a(3));
dif_t=a(2)-a(1);

a(1)=a(1)-dif_t*0.1;
a(2)=a(2)+dif_t*0.1;

a(3)=a(3)-amp*0.1;
a(4)=a(4)+amp*0.1;

axis(a);
```