M-PSK

Gallegos Ruiz Diana Abigail

Table of Contents

Introducción 1
4-PSK 2
8-PSK 7
Conclusiones 15

Introducción

Codificación M-aria

M-ario (eme ario) es un término derivado de la palabra binario. M sólo es un dígito que representa la cantidad de condiciones o combinaciones posibles para determinada cantidad de variables

binarias. Las dos técnicas de modulación digital que se han descrito hasta ahora (FSK binaria y

BPSK) son sistemas binarios; codifican bits individuales y sólo hay dos condiciones posibles de

salida. La FSK produce 1 lógico o frecuencia de marca, o un 0 lógico o frecuencia de espacio,

y la BPSK produce una fase de 1 lógico o una fase de 0 lógico. Los sistemas FSK y BPSK son

M-arios en los que M 2.

Muchas veces conviene, en la modulación digital, codificar a un nivel mayor que el binario (que a veces se dice más allá del binario, o más alto que el binario). Por ejemplo, un sistema PSK (PSK manipulación por desplazamiento de fase) con cuatro fases de salida posibles

es un sistema M-ario en el que M 4. Si hay ocho fases posibles de salida, M 8, etcétera. La

cantidad de condiciones de salida se calcula con la ecuación:

$$N = \log_2 M$$

Donde:

N cantidad de bits codificados

M cantidad de condiciones posibles de salida con N bit

4-PSK

Gallegos Ruiz Diana Abigail

```
clc
close all
clear all


% -------------- SECUENCIA DE BITS - ----------
num = randperm(255,15);
numb=cellstr(dec2bin(num));
for i=1:15 %Concatenar secuencia de bits
    if i==1 || i==2
        vec=strcat(numb(1),numb(2));
    else
        vec=strcat(vec,numb(i));
    end
end

vec=char(vec);

%-------GRÁFICA DE LA SECUENCIA GENERADA  -------------------

n=length(vec)*40;

%Valor de a
a=1/sqrt(2);

p1=sin(2*pi*n);
p2=cos(2*pi*n);
FI=[];
FQ=[];

for i=1:length(vec)
    fdn(i)=str2num(vec(i));
end
```

```
for i=1:2:length(vec)
        switch vec(1,i:i+1)
            case '11'
                FI=[FI, ones(1,40)*a];
                FQ=[FQ, ones(1,40)*a];
            case '01'
                FI=[FI ,ones(1,40)*a*-1];
                FQ=[FQ , ones(1,40)*a];
            case '00'
                FI=[FI ,ones(1,40)*a*-1];
                FQ=[FQ,ones(1,40)*a*-1];
            case '10'
                FI=[FI ,ones(1,40)*a];
                FQ=[FQ,ones(1,40)*a*-1];
        end
 end

figure(1)
tiledlayout(5,1)
nexttile
plot(fdn)
ylim([-0.1,1.2])
title('Señal digital f_n(n)')
nexttile
plot(fdn)
xlim([0,8])
title(strcat('Fragmento de secuencia: ',vec(1,1:8)))

nexttile
plot(FI)
xlim([0,200])
ylim([-1.2,1.2])
title('f_I(n)')

nexttile
plot(FQ)
xlim([0,200])
ylim([-1.2,1.2])
title('f_Q(n)')

FDN=fft(fdn);
nexttile;
plot(abs(FDN));
title('F_n(\omega)')
```

```
%------------------ MODULACIÓN Y RUIDO ------------------------------------
n=1:length(vec)*20;
FIm= sin(2*pi*n/13).*FI;
FQm=cos(2*pi*n/13).*FQ;
especI=fft(FIm);
especQ=fft(FQm);
```

El ruido de $\sigma = 0.2$, se aprecia en el diagrama de constelación y se observa como tienen a acercarse a los cuadrantes.

```
 Y=(FIm+FQm)+randn(1,2400)*0.2;

 especY=fft(Y);

figure(2)
tiledlayout(3,1)
nexttile
plot(FIm)
title('f_I(n) modulada')
nexttile
plot(FIm)
xlim([0,200])
nexttile
plot(FQm)
plot(abs(especI));



figure(3)
tiledlayout(3,1)
nexttile
plot(FQm)
title('f_Q(n) modulada')
nexttile
plot(FQm)
xlim([0,200])
nexttile
plot(abs(especQ))



figure(4)
tiledlayout(3,1)
```

```
nexttile
plot(Y)
title('4 PSK')
nexttile
plot(Y)
xlim([0,200])
nexttile
plot(abs(especY))
```

En la demodulación se pone en práctica la detección de portadoras en fase y cuadratura

```
%------------------------ DEMODULACIÓN ----------------------------------
demFI= Y.*sin(2*pi*n/13);
demFQ=Y.*cos(2*pi*n/13);


%demFI=FIm.*sin(2*pi*n/13);
%demFQ=FQm.*cos(2*pi*n/13);


figure(5)
tiledlayout(2,1)
nexttile
plot(abs(fft(demFI)))
title('F_I(\omega)')
nexttile
plot(abs(fft(demFQ)))
title('F_Q(\omega)')


%------------------------------ FILTRADO ----------------------------------
wc=1/8; % Frecuencia de corte del filtro debería ser menor que la portadora
[N,D] = butter(1,wc,'low');
fIrec=filter(N,D,demFI);
fQrec=filter(N,D,demFQ);


figure(6)
tiledlayout(2,1)
nexttile
plot(fIrec)
title('F_I(t)')
nexttile
```

```
plot(fQrec)
title('F_Q(t)')

Frec=fIrec+fQrec;

fidiagrama=detectar(fIrec);
fqdiagrama=detectar(fQrec);

figure(7)
    plot(fidiagrama, fqdiagrama, '.', 'markersize', 20)
title('Diagrama de constelación');

vecrec=[];

prodI = convertir(fidiagrama);
prodQ = convertir(fqdiagrama);

for i=1:120
    vecrec=[vecrec prodI(i) prodQ(i)];
end

figure(8)
plot(vecrec);


%FUNCIÓN PARA DETECTAR 1 & 0
function detec = detectar(x)
k=length(x);
detec = zeros(1,k/40);
r=1;
while(r<120)
    for j=0:40:k-40
        avera=0;
        for i=1:40
          avera= x(j+i);
        end
        avera=avera/40;
        detec(r)=avera;
        r=r+1;
    end
end

end

%FUNCIÓN PARA CONVERTIR A BINARIO
```

```
function con = convertir(s)
a=1/sqrt(2);
con=[];
    for i=1:length(s)
            if s(i)==a
              con=[con,'1'];
            elseif s(i)==a*-1
              con=[con,'0'];
            else
              con=[con,'2'];
            end
    end
end
```

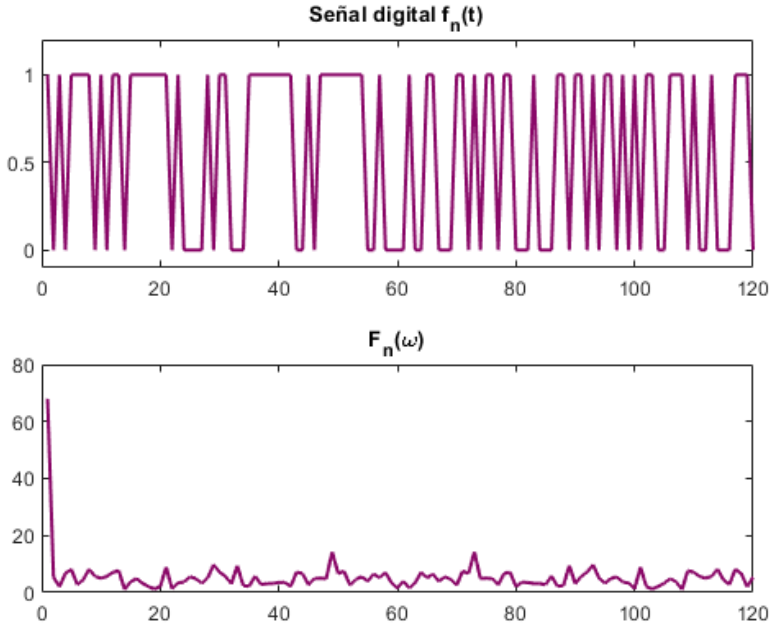
8-PSK

Para el 8-PSK se implementaron los dos convertidores analógicos digitales serie paralelo. Este programa funciona con el uso de funciones lógicas.

```
clc
close all
clear all


% -------------- SECUENCIA DE BITS - ----------
num = randperm(255,15);
numb=cellstr(dec2bin(num));
for i=1:15 %Concatenar secuencia de bits
    if i==1 || i==2
        vec=strcat(numb(1),numb(2));
    else
        vec=strcat(vec,numb(i));
    end
end

vec=char(vec);

%-------GRÁFICA DE LA SECUENCIA GENERADA  -------------------
```

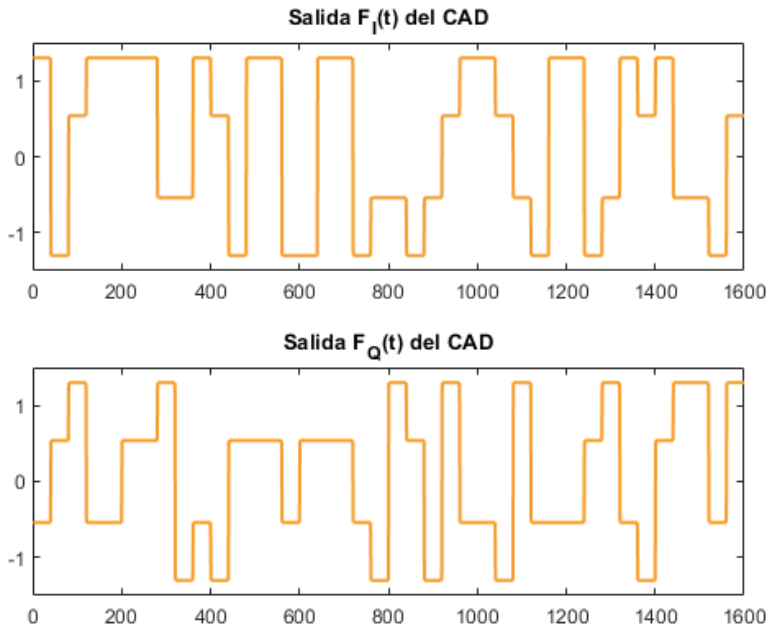
```
%Valor de a & b
a = 0.541;
b = 1.307;


FI=[];
FQ=[];

for i=1:length(vec)
    fdn(i)=str2num(vec(i));
end

%------------------- CONVERTIDOR ----------------------------------------
inde=1;
for i=1:3:length(fdn)
          I=fdn(1,i);
          Q=fdn(1,i+1);
          C=fdn(1,i+2);
          %PARA C
          FIver(inde)= I;
          Cver (inde)=C;
          switch num2str(([I C]))
             case '0  0'
                 FI=[FI, ones(1,40)*a*-1];
             case '0  1'
                 FI=[FI ,ones(1,40)*b*-1];
             case '1  0'
                 FI=[FI ,ones(1,40)*a];
             case '1  1'
                 FI=[FI ,ones(1,40)*b];
         end

          %PARA C NEGADA
           C=double(not(C));
%           FQver(inde)= [Q C];
          switch num2str(([Q C]))
             case '0  1'
                  FQ=[FQ, ones(1,40)*b*-1];
             case '0  0'
                  FQ=[FQ ,ones(1,40)*a*-1];
             case '1  1'
                  FQ=[FQ ,ones(1,40)*b];
             case '1  0'
                  FQ=[FQ , ones(1,40)*a];
          end
```

```
        inde=inde+1;
end




figure(1)
tiledlayout(2,1)
nexttile
plot(fdn,'Color','#8A0868','LineWidth',1.2)
ylim([-0.1,1.2])
title('Señal digital f_n(t)')
nexttile
plot(abs(fft(fdn)),'Color','#8A0868','LineWidth',1.2)
title('F_n(\omega)')
```



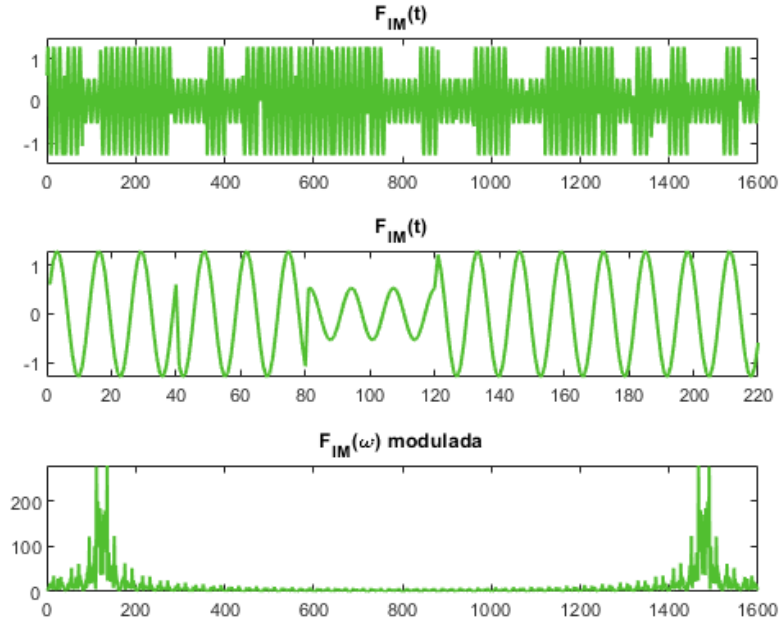

```
figure(2)
tiledlayout(2,1)
nexttile
plot(FI,'LineWidth',1.3,'Color','#f59e2c')
ylim([-b-0.2,b+0.2])
```

```
title(' Salida F_I(t) del CAD ')
nexttile
plot(FQ,'LineWidth',1.3,'Color','#f59e2c')
ylim([-b-0.2,b+0.2])
title(' Salida F_Q(t) del CAD ')
```



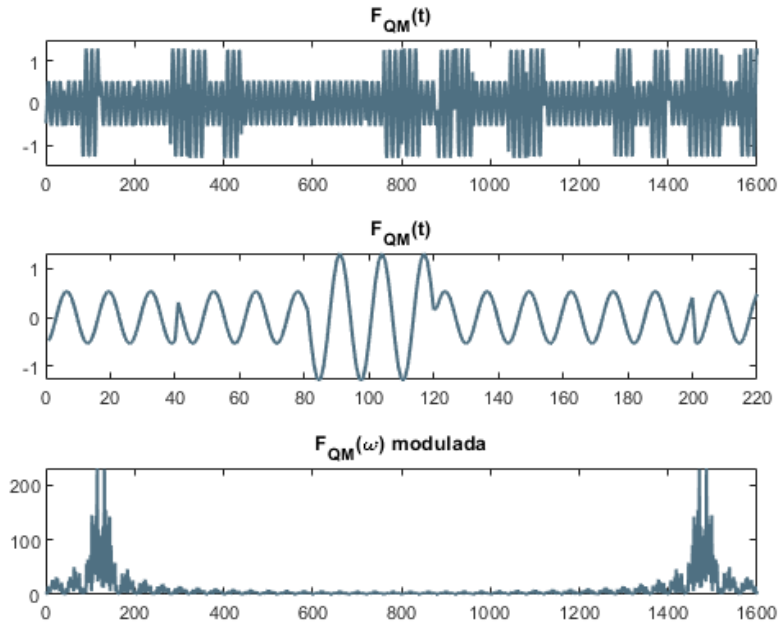

```
%-------------------------- MODULACIÓN -------------------------------------
n=1:length(FI);
p1=sin(2*pi*n/13);
p2=cos(2*pi*n/13);


fmI=FI.*p1;
fmQ=FQ.*p2;

%FI
figure(3)
tiledlayout(3,1);
nexttile
plot(fmI,'LineWidth',1.3,'Color','#51bf30')
ylim([-b-0.2,b+0.2])
```

```
title('F_{IM}(t)')
nexttile
plot(fmI,'LineWidth',1.3,'Color','#51bf30')
xlim([0,220])
title('F_{IM}(t)')
nexttile
plot(abs(fft(fmI)),'LineWidth',1.3,'Color','#51bf30')
title('F_{IM}(\omega) modulada')
```

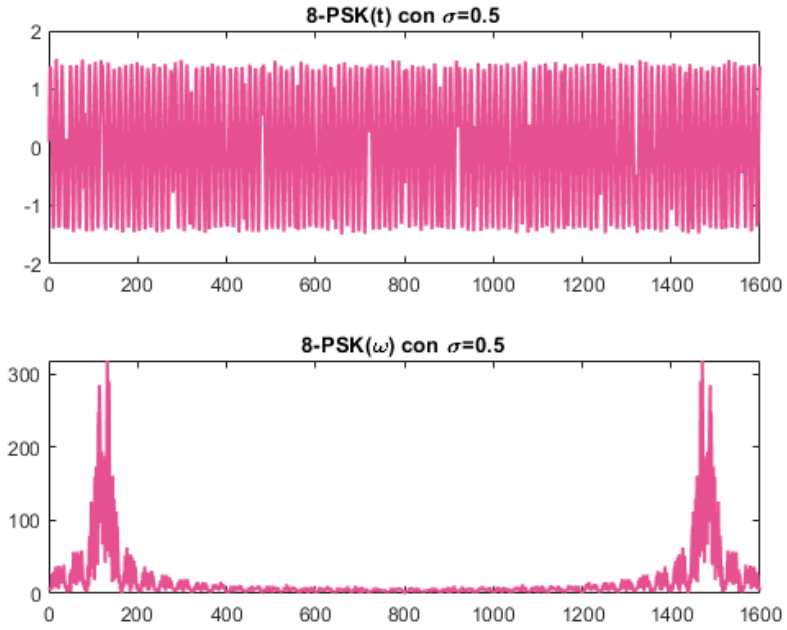



```
%FQ
figure(4)
tiledlayout(3,1);
nexttile
plot(fmQ,'LineWidth',1.3,'Color','#4f7082')
ylim([-b-0.2,b+0.2])
title('F_{QM}(t)')
nexttile
plot(fmQ,'LineWidth',1.3,'Color','#4f7082')
xlim([0,220])
title('F_{QM}(t)')
```

```
nexttile
plot(abs(fft(fmQ)),'LineWidth',1.3,'Color','#4f7082')
title('F_{QM}(\omega) modulada')
```

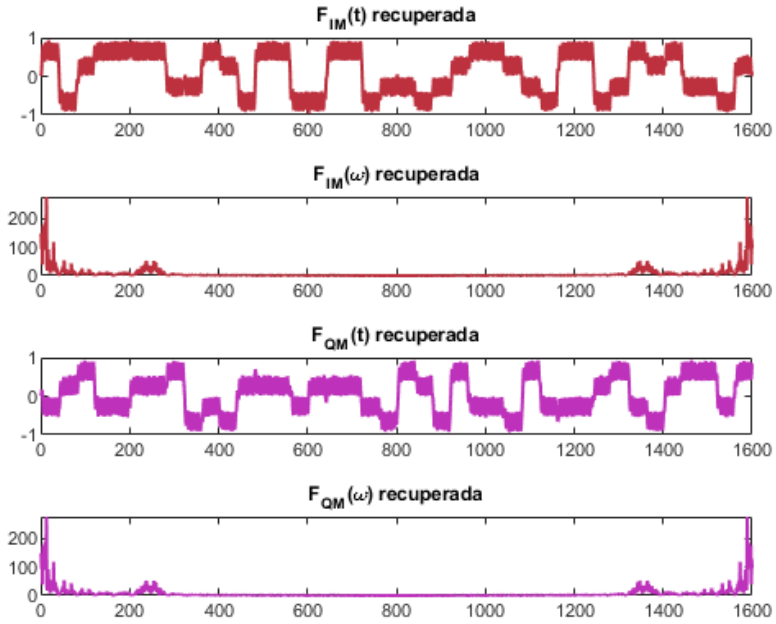



```
%-------------------------------- RUIDO ----------------------------------
s=fmQ+fmI;
r = 0.05*randn(1,length(s));
s = s+r;
figure(5)
tiledlayout(2,1);
nexttile
plot(s,'LineWidth',1.3,'Color','#e65091')
title('8-PSK(t) con \sigma=0.5')
nexttile
plot(abs(fft(s)),'LineWidth',1.3,'Color','#e65091')
title('8-PSK(\omega) con \sigma=0.5')
```




```
%-------------------- RECEPTOR--- ----------------------------------------

 % ---  DEMODULACIÓN   ---

demFI= s.*sin(2*pi*n/13);
demFQ=s.*cos(2*pi*n/13);

wc=1/8; % Frecuencia de corte del filtro debería ser menor que la portadora
[N,D] = butter(1,wc,'low');
fIrec=filter(N,D,demFI);
fQrec=filter(N,D,demFQ);

figure(6)
tiledlayout(4,1)
nexttile
plot(fIrec,'LineWidth',1.3,'Color','#bd313f')
title('F_{IM}(t) recuperada')
nexttile
plot(abs(fft(fIrec)),'LineWidth',1.3,'Color','#bd313f')
```

```
title('F_{IM}(\omega) recuperada')
nexttile
plot(fQrec,'LineWidth',1.3,'Color','#bd31bb' )
title('F_{QM}(t) recuperada')
nexttile
plot(abs(fft(fIrec)),'LineWidth',1.3,'Color','#bd31bb')
title('F_{QM}(\omega) recuperada')
```




```
%------------------------     DETECTOR  -----------------------
 fidiagrama=detectar(fIrec);
fqdiagrama=detectar(fQrec);

z=1
```

```
z = 1
```

```
for i=1:2:80

diagramaver2(z)=fidiagrama(i+1);
diagramaver1(z)=fidiagrama(i);
 z=z+1;
```

```
end

%[diagramaver1' FIver' Cver' diagramaver2']


%FUNCIÓN PARA DETECTAR 1 & 0
```

```
function detec = detectar(x)
k=length(x);
detec = zeros(1,(k/40));
r=1;
while(r<40)
    for j=0:20:k-20
        avera=0;
        for i=1:20
          avera= x(j+i);
        end
        avera=avera/40;
         if avera <= 0
             detec(r)= 0;
         else
             detec(r)= 1;
         end
        r=r+1;
    end
end
end
```

Conclusiones

Con el aumento de bits M, mayor es la complejidad de programarlo en MatLab, sin embargo en la práctica, se vuelven sistemas con mayor velocidad de transmisión y con tendecia a errores que pueden ser detectados con el uso de la Teoría de la Información.