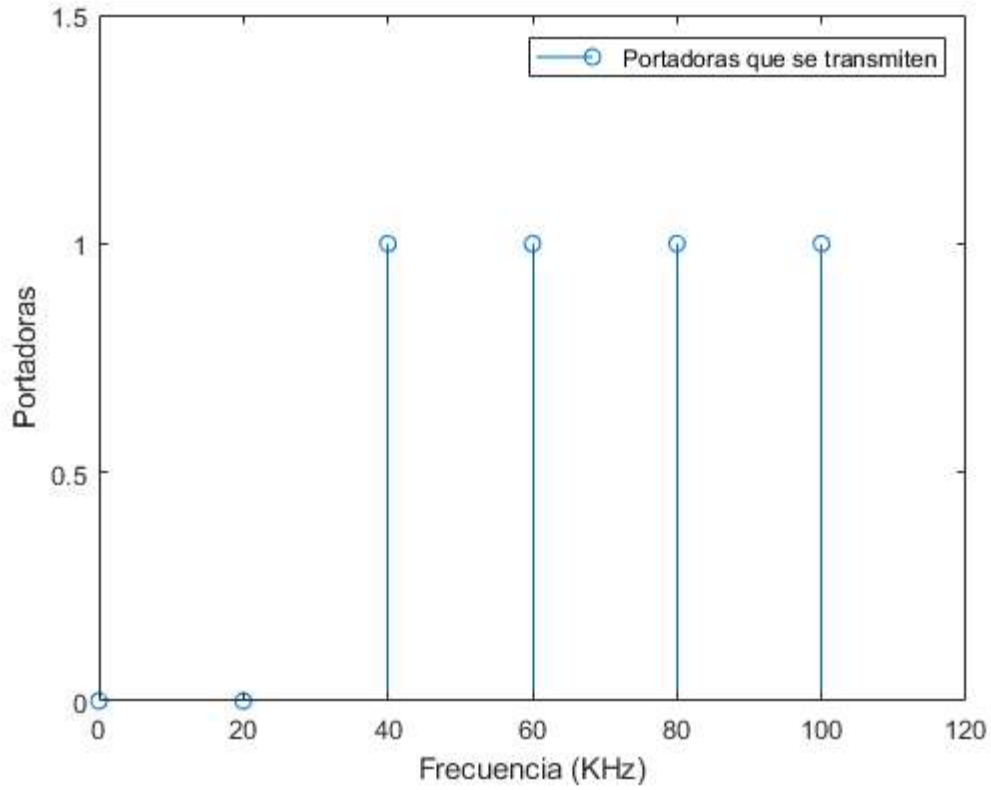
Contents

Definición de parámetros de transmisión

```
Representar gráficamente sobre un eje de frecuencias las portadoras que se transmiten
Empleando como número de muestras NFFT= 32, calcular los parámetros siguientes:
  - Frecuencia de muestreo fs
  - Periodo de IFFT, Tifft
```

```
   close all;  clear; format compact
%
%  Carga de datos iniciales
   datos_iniciales
%
   figure
   stem(F, Portadoras)
   xlabel('Frecuencia (KHz)')
   ylabel('Portadoras')
   ylim([0 1.5]); xlim([0  F(end)+F(2)])
   legend('Portadoras que se transmiten')
%
   disp(newline);
   disp(['Frecuencia de muestreo,        fs    = ', num2str(fs)])
   disp(['Periodo de muestreo,           Tifft = ', num2str(Tifft)])
%
```

```
Frecuencia de muestreo,        fs    = 640000
Periodo de muestreo,           Tifft = 5e-05
```



Sin degradación por error de frecuencia

```
    Nofdm = 200;
%
    m_ary = 16;
    [BER_16QAM_fft_de_MATLAB] = QAM_fft_de_MATLAB(Nofdm, m_ary)

    m_ary = 64;
    [BER_64QAM_fft_de_MATLAB] = QAM_fft_de_MATLAB(Nofdm, m_ary)

    m_ary = 256;
    [BER_256QAM_fft_de_MATLAB] = QAM_fft_de_MATLAB(Nofdm, m_ary)
%
```

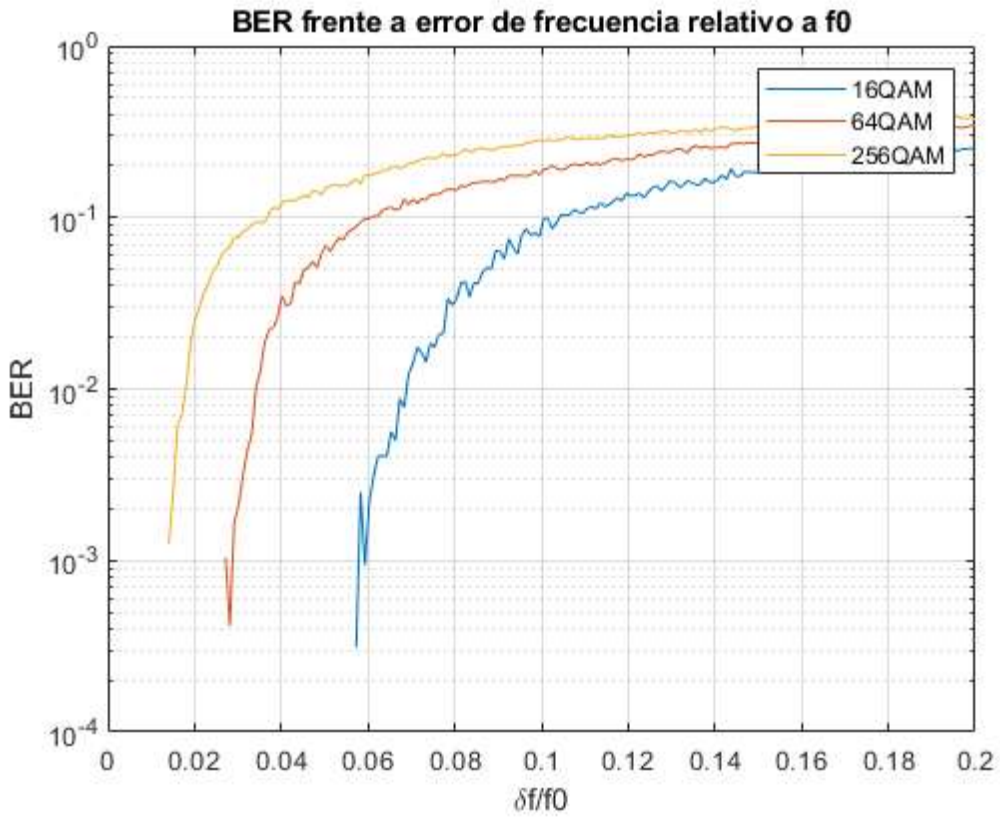
```
BER_16QAM_fft_de_MATLAB =
     0
BER_64QAM_fft_de_MATLAB =
     0
BER_256QAM_fft_de_MATLAB =
     0
```

Con degradación por error de frecuencia

Representación de BER en función del error relativo de frecuencia

```
    deltaf_f0 = linspace(0, 0.2, 200);
    Nofdm = 200;
%
    m_ary = 16;
    BER_16QAM = zeros(1, length(deltaf_f0));
```

```
for i=1:length(deltaf_f0)
     [BER_16QAM(i)] = QAM(Nofdm, m_ary, deltaf_f0(i));
end
m_ary = 64;
BER_64QAM = zeros(1, length(deltaf_f0));
for i=1:length(deltaf_f0)
     [BER_64QAM(i)] = QAM(Nofdm, m_ary, deltaf_f0(i));
end
m_ary = 256;
BER_256QAM = zeros(1, length(deltaf_f0));
for i=1:length(deltaf_f0)
     [BER_256QAM(i)] = QAM(Nofdm, m_ary, deltaf_f0(i));
end
figure
semilogy(deltaf_f0,BER_16QAM); hold on; grid on;
semilogy(deltaf_f0,BER_64QAM);
semilogy(deltaf_f0,BER_256QAM); hold off;
xlabel('\deltaf/f0')
ylabel('BER')
title('BER frente a error de frecuencia relativo a f0')
legend('16QAM', '64QAM', '256QAM')
```



"OFDM_imperfecciones_main"

"datos_iniciales"

```
% Definición de parámetros de transmisión
%
NFFT   = 32;
%
nportadoras      = 4;                    % número de portadoras consecutivas con energía
Portadoras = [0  0  ones(1,nportadoras)];
if (2*length(Portadoras) + 1) >= NFFT
```

```
    mensaje = '     NFFT muy pequeño para el número de portadoras';
    error(mensaje)
end
F          = 20*[0:length(Portadoras)-1];
kmin  = min(find(Portadoras~=0));
kmax  = max(find(Portadoras~=0));
%

deltaf = F(2)*10^3 ;
f0     = deltaf;
Tg     = 0;
Tifft  = 1/deltaf ;
Ts     = Tifft/NFFT ;
fs     = 1/Ts ;
Ncp    = 0 ;
Tcp    = Ncp*Ts ;
%
```

"QAM_fft_de_MATLAB"

```
function [BER_QAM] = QAM_fft_de_MATLAB(Nofdm, m_ary)
%
% BER_QAM       Valor obtenido de BER
% Nofdm         Número de símbolos OFDM
%
%
% Definición de parámetros de transmisión
  datos_iniciales
%
% Secuencia de bits
  Txbits = secuencia(Nofdm, nportadoras, m_ary);
%
% Generación de símbolos QAM a partir de la secuencia de bits
  M = qammod(Txbits', m_ary, 'gray','InputType', 'bit').'; % Senyal modulada
%
% Generación del vector de muestras temporales a la salida del modulador
%
  X = zeros(NFFT, Nofdm);
  Y = X;
  y_ofdm  = X;
 %
  M2                 = reshape(M, nportadoras, Nofdm);
  X(kmin:kmax,:)     = M2(1:nportadoras,:);
  X(NFFT/2+2:NFFT,:) = flipud(conj(X(2:NFFT/2,:)));
  y_ofdm             = ifft(X,NFFT, 'symmetric');

  for i=1:Nofdm
        Y(:,i) = fft(y_ofdm(:,i), NFFT);
  end
  Y = Y(kmin:kmax,1:Nofdm);
  Y = reshape(Y,1,Nofdm*(1+kmax-kmin));
  Y = Y.';
%
% Demodulación de símbolo y cálculo de BER
  Rxbits  = qamdemod(Y, m_ary, 'gray', 'OutputType', 'bit').'; % Señal demodulada
  BER_QAM = sum(abs(Txbits-Rxbits))/length(Txbits);
%
end
```

"SECUENCIA"

```
%
  function [bits] = secuencia(Nofdm, n_portadoras, m_ary);
%
% Esta función genera una secuencia de bits correspondiente a Nofdm símbolos, cuando el nivel de
% modulación es m_ary. Los bits del primer símbolo son deterministas, y el resto aleatorios
%
% bits          secuencia de bits de salida
% Nofdm         número de símbolos ofdm
% n_portadoras  número de portadoras moduladas
% m_ary         nivel de modulación
%
  k     = log2(m_ary);
  n     = Nofdm*n_portadoras*k;
  bits  = zeros(1,n);
  bits  = [repmat([1 0], 1, n_portadoras*k/2)   round(rand(1, n - n_portadoras*k))];
%
end
```

"QAM"

```
function [BER_QAM] = QAM(Nofdm, m_ary, deltaf_f0)
%
% BER_QAM       Valor obtenido de BER
% Nofdm         Número de símbolos OFDM
% deltaf_f0     Error relativo de frecuecia en la demodulación
%
%
% Definición de parámetros de transmisión
  datos_iniciales
%
% Secuencia de bits
  Txbits = secuencia(Nofdm, nportadoras, m_ary);
%
% Generación de símbolos QAM a partir de la secuencia de bits
  M = qammod(Txbits', m_ary, 'gray','InputType', 'bit').'; % Senyal modulada
%
% Generación del vector de muestras temporales a la salida del modulador
%
  X = zeros(NFFT, Nofdm);
  Y = X;
  y_ofdm  = X;
 %
  M2                 = reshape(M, nportadoras, Nofdm);
  X(kmin:kmax,:)     = M2(1:nportadoras,:);
  X(NFFT/2+2:NFFT,:) = flipud(conj(X(2:NFFT/2,:)));
  y_ofdm             = ifft(X,NFFT, 'symmetric');

  for i=1:Nofdm
        Y(:,i) = fft_manual(y_ofdm(:,i), NFFT, deltaf_f0);
  end
  Y = Y(kmin:kmax,1:Nofdm);
  Y = reshape(Y,1,Nofdm*(1+kmax-kmin));
  Y = Y.';
%
```

```
% Demodulación de símbolo y cálculo de BER
  Rxbits  = qamdemod(Y, m_ary, 'gray', 'OutputType', 'bit').'; % Señal demodulada
  BER_QAM = sum(abs(Txbits-Rxbits))/length(Txbits);
%
end
```

"fft_manual"

```
function X = fft_manual(y, NFFT, deltaf_f0)
%
   datos_iniciales
   X = zeros(NFFT,1);
   n = (1:NFFT)' ;
   for k=kmin:kmax
        kf = k +  deltaf_f0;
        X(k) = y'*[exp(-j*2*pi/NFFT*(kf-1)*(n-1))] ;
   end
%
end
```

Published with MATLAB® R2020b