SIMULACIÓN Y ANÁLISIS DE LA INTERFERENCIA ICI EN OFDM SOBRE UN ENLACE PUNTO A PUNTO

ANEXOS

Gerson Orozco Morales
Jonathan Rosero Delgado

Universidad del Cauca
Facultad de Ingeniería Electrónica y de Telecomunicaciones
Departamento de Telecomunicaciones
Grupo Nuevas Tecnologías en Telecomunicaciones GNTT
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Noviembre de 2010

SIMULACIÓN Y ANÁLISIS DE LA INTERFERENCIA ICI EN OFDM SOBRE UN ENLACE PUNTO A PUNTO

ANEXOS

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Gerson Orozco Morales
Jonathan Rosero Delgado

Director
Mag. Harold Armando Romo Romero

Universidad del Cauca
Facultad de Ingeniería Electrónica y de Telecomunicaciones
Departamento de Telecomunicaciones
Grupo Nuevas Tecnologías en Telecomunicaciones GNTT
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Noviembre de 2010

TABLA DE CONTENIDO ANEXOS

LISTA DE FIGURAS

LISTA DE TABLAS

ANEXO A
CÓDIGOS UTILIZADOS PARA SIMULACIÓN EN MATLAB

Los códigos que se encuentran en el anexo A fueron los utilizados para simular los fenómenos de el CFO, STO y SCO en un sistema OFDM para un enlace punto a punto, los parámetros utilizados se encuentran detallados en el capitulo 4.

A.1. Código para Simular CFO

```
%% BER en sistema OFDM con ICI.
%El siguiente código simula el desempeño (BER) de un sistema OFDM
%que utiliza modulaciones BPSK, QPSK, 16-QAM sobre un canal Rician,
Rayleigh o AWGN cuando se ve
%afectado por la interferencia ICI que provoca el offset de frecuencia
portadora,
% a través de los siguientes pasos:
%   TRANSMISION
%   1-Crear los datos a enviar de forma aleatoria
%   2-mapearlos según el diagrama de constelación
%   3-adicionar codificación Gray
%   4-convertir los datos de serie a paralelo
%   5-introducir muestras de guarda
%   6-aplicar la IFFT sobre los datos mapeados
%   7-adicionar el prefijo cíclico
%   8-introducir el término que simula el offset de frecuencia
%
%   CANAL
%   9-realizar la convolución entre los símbolos que se crearon en
%   transmisión y el canal (cuando se trabaja con canales
multitrayecto)
%   10-adicionar el ruido AWGN
%
%   RECEPCION
%   11-conversion paralelo serie
%   12-quitar el prefijo cíclico
%   13-realizar la FFT
%   14-hacer ecualización en frecuencia
%   15-quitar muestras de guarda
%   16-demodular
%   17-demapeo
%   18-calcaular  la BER
%% VARIABLES Y CONSTANTES DEL SISTEMA

clc;
clear all;
close all;
%% variables del sistema------
NFFT=64;                                    %tamaño de la FFT/IFFT
Nsp=64;                                     %numero de subportadoras
Nspd=52;                                    %numero de subportadoras
de datos
Nspg=12;                                    %numero de subportadoras
de guarda
Npc=16;                                     %numero de muestras que
conforman el prefijo cíclico
```

```
Ns=64;                                          %numero de muestras que
conforman un símbolo OFDM
modulaciones ={'BPSK','QPSK','16-QAM'};         %permite escribir el tipo
de modulacion en el titulo de las graficas
MOD=1;                                          %(1=BPSK, 2=QPSK, 3=16-
QAM)
tcanal={'RICIAN','RAYLEIGH','AWGN'};            %%permite escribir el tipo
de canal en el titulo de las graficas
TipoCanal=1;                                    %(1=RICIAN, 2=RAYLEIGH,
3=AWGN)
Ts=8e-7;Fs=1/Ts;                                %periodo/frecuencia de
muestreo
Modulacion='psk';                               %tipo de modulación (PSK,
QAM)
M=2;                                            %índice de modulación
numerodesimbolos = 10000;                        %cantidad de símbolos a
transmitir
factorK=5;                                      %factor K para el canal
RICE
alfa=[0 0.1 0.2 0.3];%offset de frecuencia, para simular el efecto de
corrimiento de espectro
%-----------------------------------------------------------------
if strcmp(Modulacion,'qam')== 1                 %Dependiendo del valor de
la variable Modulacion crea los objetos
    Obj_modulador=modem.qammod(M);              %Obj_modulador y
Obj_demodulador que requieren las funciones de
    Obj_demodulador=modem.qamdemod(M);          %modulación y demodulación
para PSK y QAM.
else
    Obj_modulador=modem.pskmod(M);
    Obj_demodulador=modem.pskdemod(M);
end
%% relación señal a ruido
EbNo=0:5:30;
SNR=EbNo
+10*log10(Nspd/(NFFT+1))+10*log10(NFFT/(NFFT+Npc))+10*log10(log2(M));
%% transmisión
%Fuente: Genera bits de forma aleatoria
datosTx=randint(1,Nspd*numerodesimbolos);
%Mapeo: mapea los bits segun el tipo de modulación con el código Gray
mapeo=bi2de(reshape(datosTx,log2(M),numerodesimbolos*Nspd/log2(M))','l
eft-msb');
mapeo_Gray = bin2gray(mapeo,Modulacion,M);
senal=modulate(Obj_modulador,mapeo_Gray);
%serie-paralelo: convertir de serie a paralelo para aplicar la IFFT
SerieParaleloTx=reshape(senal,Nspd,numerodesimbolos/log2(M));
%Subportadoras de guarda: inserta subportadoras de guarda para evitar
interferencia
insertar_guardas=[zeros(Nspg/2,numerodesimbolos/log2(M));SerieParalelo
Tx;zeros(Nspg/2,numerodesimbolos/log2(M))];
%transformada inversa de FOURIER: realiza la modulación en múltiples
subportadoras
SimboloOFDM=ifft(insertar_guardas,NFFT);
%Prefijo cíclico: para reducir la ISI
SimboloOFDM_PC=[SimboloOFDM(Ns-Npc+1:Ns,:);SimboloOFDM];
%efecto offset de frecuencia para los diferentes valores
for k=1:numerodesimbolos/log2(M)

SimboloOFDM_TX(1:Ns+Npc,k)=SimboloOFDM_PC(1:Ns+Npc,k).*exp(2*j*pi*(alf
a(1)/Ns)*(0:Ns+Npc-1)).';
```

```
SimboloOFDM_TX2(1:Ns+Npc,k)=SimboloOFDM_PC(1:Ns+Npc,k).*exp(2*j*pi*(al
fa(2)/Ns)*(0:Ns+Npc-1)).';

SimboloOFDM_TX3(1:Ns+Npc,k)=SimboloOFDM_PC(1:Ns+Npc,k).*exp(2*j*pi*(al
fa(3)/Ns)*(0:Ns+Npc-1)).';

SimboloOFDM_TX4(1:Ns+Npc,k)=SimboloOFDM_PC(1:Ns+Npc,k).*exp(2*j*pi*(al
fa(4)/Ns)*(0:Ns+Npc-1)).';
end
% ----------------------------
%% canal
switch TipoCanal

    case 1
        num_trayectos=3;% numero de trayectos canal
        %Trayectos: un trayecto directo y dos indirectos
        TrayectoDirecto=1*ones(1,numerodesimbolos/log2(M));
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rice
        canal = [TrayectoDirecto;TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
equalizacion en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
        rt1(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX(1:Ns+Npc,jj));
        rt2(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX2(1:Ns+Npc,jj));
        rt3(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX3(1:Ns+Npc,jj));
        rt4(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX4(1:Ns+Npc,jj));
        end
        ParaleloSerie=reshape(rt1,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieA=reshape(rt2,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieB=reshape(rt3,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieC=reshape(rt4,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1,factorK);

    case 2
        num_trayectos=2;
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rayleigh
        canal = [TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
ecualización en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
```

```
        rt1(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX(1:Ns+Npc,jj));
        rt2(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX2(1:Ns+Npc,jj));
        rt3(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX3(1:Ns+Npc,jj));
        rt4(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_TX4(1:Ns+Npc,jj));
        end
        ParaleloSerie=reshape(rt1,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieA=reshape(rt2,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieB=reshape(rt3,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieC=reshape(rt4,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1);
    case 3
        num_trayectos=1;
        hF=1;
        ParaleloSerie =
reshape(SimboloOFDM_TX,1,(Ns+Npc)*numerodesimbolos/log2(M));
        ParaleloSerieA =
reshape(SimboloOFDM_TX2,1,(Ns+Npc)*numerodesimbolos/log2(M));
        ParaleloSerieB =
reshape(SimboloOFDM_TX3,1,(Ns+Npc)*numerodesimbolos/log2(M));
        ParaleloSerieC =
reshape(SimboloOFDM_TX4,1,(Ns+Npc)*numerodesimbolos/log2(M));
        BERtheory = berawgn(EbNo,Modulacion,M,'nondiff');
end
%%
for i = 1:length(SNR)
%Ruido: adicionar ruido AWGN a la señal, yt=h(t)*S(t)+ n(t) si el
canal
%Rice o Rayleigh y yt=S(t)+ n(t) si es awgn
%% awgn
%---se adiciona ruido awgn
yt=awgn(ParaleloSerie,SNR(i),'measured');
yt1=awgn(ParaleloSerieA,SNR(i),'measured');
yt2=awgn(ParaleloSerieB,SNR(i),'measured');
yt3=awgn(ParaleloSerieC,SNR(i),'measured');
%% recepción
%---se pasan los datos de  serie a paralelo en el receptor para
remover el prefijo cíclico y calcular la FFT
SerieParalelo2=reshape(yt,Ns+Npc+num_trayectos-
1,numerodesimbolos/log2(M));
SerieParalelo3=reshape(yt1,Ns+Npc+num_trayectos-
1,numerodesimbolos/log2(M));
SerieParalelo4=reshape(yt2,Ns+Npc+num_trayectos-
1,numerodesimbolos/log2(M));
SerieParalelo5=reshape(yt3,Ns+Npc+num_trayectos-
1,numerodesimbolos/log2(M));
%---se remueve el prefijo cíclico
SerieParalelo2(1:Npc,:)=[];
SerieParalelo3(1:Npc,:)=[];
SerieParalelo4(1:Npc,:)=[];
SerieParalelo5(1:Npc,:)=[];
%---se realiza la FFT
SimboloOFDM2=fft(SerieParalelo2,NFFT);
SimboloOFDM3=fft(SerieParalelo3,NFFT);
```

```
SimboloOFDM4=fft(SerieParalelo4,NFFT);
SimboloOFDM5=fft(SerieParalelo5,NFFT);
%---ecualización en frecuencia
Simbolo3=SimboloOFDM2./hF;
Simbolo4=SimboloOFDM3./hF;
Simbolo5=SimboloOFDM4./hF;
Simbolo6=SimboloOFDM5./hF;
%---remover muestras de guarda
remover_guarda=Simbolo3((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guarda1=Simbolo4((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guarda2=Simbolo5((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guarda3=Simbolo6((Nspg/2)+1:Nsp-(Nspg/2),:);
%---se pasan los datos de paralelo a serie para demodularlos
ParaleloSerie2=reshape(remover_guarda,1,(Nspd)*numerodesimbolos/log2(M
));
ParaleloSerie3=reshape(remover_guarda1,1,(Nspd)*numerodesimbolos/log2(
M));
ParaleloSerie4=reshape(remover_guarda2,1,(Nspd)*numerodesimbolos/log2(
M));
ParaleloSerie5=reshape(remover_guarda3,1,(Nspd)*numerodesimbolos/log2(
M));
%---demodulación de los datos
datosRx= demodulate(Obj_demodulador,ParaleloSerie2);
datosRx2= demodulate(Obj_demodulador,ParaleloSerie3);
datosRx3= demodulate(Obj_demodulador,ParaleloSerie4);
datosRx4= demodulate(Obj_demodulador,ParaleloSerie5);
%---demapeo
demapeo = gray2bin(datosRx,Modulacion,M);
datosRxA = de2bi(demapeo,'left-msb');
datosrx=reshape(datosRxA',1,Nspd*numerodesimbolos);

demapeo2 = gray2bin(datosRx2,Modulacion,M);
datosRxB = de2bi(demapeo2,'left-msb');
datosrx2=reshape(datosRxB',1,Nspd*numerodesimbolos);

demapeo3 = gray2bin(datosRx3,Modulacion,M);
datosRxC = de2bi(demapeo3,'left-msb');
datosrx3=reshape(datosRxC',1,Nspd*numerodesimbolos);

demapeo4 = gray2bin(datosRx4,Modulacion,M);
datosRxD = de2bi(demapeo4,'left-msb');
datosrx4=reshape(datosRxD',1,Nspd*numerodesimbolos);
%---cálculo de la BER
[errores BER1(i)]=biterr(datosTx,datosrx);
[errores BER2(i)]=biterr(datosTx,datosrx2);
[errores BER3(i)]=biterr(datosTx,datosrx3);
[errores BER4(i)]=biterr(datosTx,datosrx4);
end
%% grafica
close all; figure
%---grafica para canal AWGN
if TipoCanal==3

semilogy(EbNo,BER1,'b--',EbNo,BER2,'r-',EbNo,BER3,'m-',EbNo,BER4,'y-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
axis([0 14 1e-6 1e-0]);
```

```
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada alfa=0.0','BER simulada alfa=0.1','BER simulada
alfa=0.2','BER simulada alfa=0.3','BER teórica'});

else
%---grafica para canales Rician o Rayleigh
semilogy(EbNo,BER1,'b--',EbNo,BER2,'r-',EbNo,BER3,'m-',EbNo,BER4,'y-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada alfa=0.0','BER simulada alfa=0.1','BER simulada
alfa=0.2','BER simulada alfa=0.3','BER teórica'});
end
```

A.2. Código para Simular STO

```
%% BER en sistema OFDM con ICI.
%El siguiente código simula el desempeño (BER) de un sistema OFDM
%que utiliza modulaciones BPSK, QPSK, 16-QAM sobre un canal Rician,
Rayleigh o AWGN cuando se ve
%afectado por la interferencia ICI que provoca el offset de tiempo de
símbolo,
%a través de los siguientes pasos:
%   TRANSMISION
%   1-Crear los datos a enviar de forma aleatoria
%   2-mapearlos según el diagrama de constelación
%   3-adicionar codificación Gray
%   4-convertir los datos de serie a paralelo
%   5-introducir muestras de guarda
%   6-aplicar la IFFT sobre los datos mapeados
%   7-adicionar el prefijo cíclico
%
%
%   CANAL
%   8-realizar la convolución entre los símbolos que se crearon en
%   transmisión y el canal (cuando se trabaja con canales
multitrayecto)
%   9-adicionar el ruido AWGN
%
%   RECEPCION
%   10-conversion paralelo serie
%   11-quitar el prefijo cíclico
%   12-se realiza corrimiento de símbolos para simular STO
%   13-realizar la FFT
%   14-hacer ecualización en frecuencia
%   15-quitar muestras de guarda
%   16-demodular
%   17-demapeo
%   18-calcaular  la BER
%% VARIABLES Y CONSTANTES DEL SISTEMA
clc;
clear all;
close all;
```

```
NFFT=64;                                        %tamaño de la FFT/IFFT
Nsp=64;                                         %numero de subportadoras
Nspd=52;                                        %numero de subportadoras
de datos
Nspg=12;                                        %numero de subportadoras
de guarda
Npc=16;                                         %numero de muestras que
conforman el prefijo cíclico
Ns=64;                                          %numero de muestras que
conforman un símbolo OFDM
modulaciones ={'BPSK','QPSK','16-QAM'};         %permite escribir el tipo
de modulacion en el titulo de las graficas
MOD=1;                                          %(1=BPSK, 2=QPSK, 3=16-
QAM)
tcanal={'RICIAN','RAYLEIGH','AWGN'};            %%permite escribir el tipo
de canal en el titulo de las graficas
TipoCanal=1;                                    %(1=RICIAN, 2=RAYLEIGH,
3=AWGN)
Ts=8e-7;Fs=1/Ts;                                %periodo/frecuencia de
muestreo
Modulacion='psk';                               %tipo de modulación (PSK,
QAM)
M=2;                                            %índice de modulación
numerodesimbolos = 10000;                        %cantidad de símbolos a
transmitir
factorK=5;                                      %factor K para el canal
RICE
total=Nspd+Npc+Nspg;
tao=[1 5 10]; %corrimiento de símbolos, puede tomar valores (-16,-15,-
14...16)
%-----------------------------------------------------
if strcmp(Modulacion,'qam')== 1                 %Dependiendo del valor de
la variable Modulacion crea los objetos
    Obj_modulador=modem.qammod(M);              %Obj_modulador y
Obj_demodulador que requieren las funciones de
    Obj_demodulador=modem.qamdemod(M);          %modulación y demodulación
para PSK y QAM.
else
    Obj_modulador=modem.pskmod(M);
    Obj_demodulador=modem.pskdemod(M);
end
%--------------------
%% relación señal a ruido
EbNo=0:5:30;
SNR=EbNo
+10*log10(Nspd/(NFFT+1))+10*log10(NFFT/(NFFT+Npc))+10*log10(log2(M));
%% transmisión-----------------------------------
%Fuente: Genera bits de forma aleatoria
datosTx=randint(1,Nspd*numerodesimbolos);
%Mapeo: mapea los bits segun el tipo de modulación con el código Gray
mapeo=bi2de(reshape(datosTx,log2(M),numerodesimbolos*Nspd/log2(M))','l
eft-msb');
mapeo_Gray = bin2gray(mapeo,Modulacion,M);
senal=modulate(Obj_modulador,mapeo_Gray);
%serie-paralelo: convertir de serie a paralelo para aplicar la IFFT
SerieParaleloTx=reshape(senal,Nspd,numerodesimbolos/log2(M));
%Subportadoras de guarda: inserta subportadoras de guarda para evitar
interferencia
insertar_guardas=[zeros(Nspg/2,numerodesimbolos/log2(M));SerieParalelo
Tx;zeros(Nspg/2,numerodesimbolos/log2(M))];
```

```
%-------------------------------------------------
for k=1:numerodesimbolos/log2(M)

OFFSET(1:64,k)=insertar_guardas(1:64,k).*exp(2*j*pi*(tao(1))/64*(0:63)
)';

OFFSET1(1:64,k)=insertar_guardas(1:64,k).*exp(2*j*pi*(tao(2))/64*(0:63
))';

OFFSET2(1:64,k)=insertar_guardas(1:64,k).*exp(2*j*pi*(tao(3))/64*(0:63
))';
end
%se realiza la IFFT
SimboloOFDM=ifft(OFFSET,NFFT);
SimboloOFDM1A=ifft(OFFSET1,NFFT);
SimboloOFDM2B=ifft(OFFSET2,NFFT);
%se inserta el prefijo cíclico
SimboloOFDM_PC=[SimboloOFDM(Ns-Npc+1:Ns,:);SimboloOFDM];
SimboloOFDM_PC1=[SimboloOFDM1A(Ns-Npc+1:Ns,:);SimboloOFDM1A];
SimboloOFDM_PC2=[SimboloOFDM2B(Ns-Npc+1:Ns,:);SimboloOFDM2B];

%% canal
switch TipoCanal

    case 1
        num_trayectos=3;% numero de trayectos canal
        %Trayectos: un trayecto directo y dos indirectos
        TrayectoDirecto=1*ones(1,numerodesimbolos/log2(M));
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rice
        canal = [TrayectoDirecto;TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
ecualización en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
        rt1(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC(1:Ns+Npc,jj));
        rt2(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC1(1:Ns+Npc,jj));
        rt3(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC2(1:Ns+Npc,jj));
        end
        ParaleloSerie=reshape(rt1,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieA=reshape(rt2,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieB=reshape(rt3,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1,factorK);

    case 2
        num_trayectos=2;
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rayleigh
```

```
        canal = [TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
ecualización en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
        rt1(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC(1:Ns+Npc,jj));
        rt2(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC1(1:Ns+Npc,jj));
        rt3(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC2(1:Ns+Npc,jj));
        end
        ParaleloSerie=reshape(rt1,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieA=reshape(rt2,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        ParaleloSerieB=reshape(rt3,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1);
    case 3
        num_trayectos=1;
        hF=1;
        ParaleloSerie =
reshape(SimboloOFDM_TX,1,(Ns+Npc)*numerodesimbolos/log2(M));
        ParaleloSerieA =
reshape(SimboloOFDM_TX2,1,(Ns+Npc)*numerodesimbolos/log2(M));
        ParaleloSerieB =
reshape(SimboloOFDM_TX3,1,(Ns+Npc)*numerodesimbolos/log2(M));
        BERtheory = berawgn(EbNo,Modulacion,M,'nondiff');
end
%%
for i = 1:length(SNR)
%Ruido: adicionar ruido AWGN a la señal, yt=h(t)*S(t)+ n(t) si el
canal
%Rice o Rayleigh y yt=S(t)+ n(t) si es awgn
    %% awgn
%---se adiciona ruido awgn
yt=awgn(ParaleloSerie,SNR(i),'measured');
yt1=awgn(ParaleloSerieA,SNR(i),'measured');
yt2=awgn(ParaleloSerieB,SNR(i),'measured');

SerieParalelo2=reshape(yt,Ns+Npc+num_trayectos-
1,(numerodesimbolos/log2(M)));
SerieParalelo2(total+1:total+num_trayectos-1,:)=[];
ParaleloSerie2=reshape(SerieParalelo2,1,total*(numerodesimbolos/log2(M
)));

SerieParalelo3=reshape(yt1,Ns+Npc+num_trayectos-
1,(numerodesimbolos/log2(M)));
SerieParalelo3(total+1:total+num_trayectos-1,:)=[];
ParaleloSerie3=reshape(SerieParalelo3,1,total*(numerodesimbolos/log2(M
)));

SerieParalelo4=reshape(yt2,Ns+Npc+num_trayectos-
1,(numerodesimbolos/log2(M)));
SerieParalelo4(total+1:total+num_trayectos-1,:)=[];
ParaleloSerie4=reshape(SerieParalelo4,1,total*(numerodesimbolos/log2(M
)));
%---se realiza corrimiento de símbolos
   if tao(1)>0
```

```
       B=padarray(ParaleloSerie2,[0 tao(1)],0,'post');

         for l=1:numerodesimbolos/log2(M)
          SerieParalelo2(1:64,l)=B(Npc+1+tao(1)+(total*(l-
1)):(total*l)+tao(1));
         end
   else
         for l=1:numerodesimbolos/log2(M)
         SerieParalelo2(1:64,l)=ParaleloSerie2(Npc+1+tao(1)+(total*(l-
1)):(total*l)+tao(1));

         end
   end
   if tao(2)>0
       C=padarray(ParaleloSerie3,[0 tao(2)],0,'post');

         for l=1:numerodesimbolos/log2(M)
          SerieParalelo3(1:64,l)=C(Npc+1+tao(2)+(total*(l-
1)):(total*l)+tao(2));
         end
   else
         for l=1:numerodesimbolos/log2(M)
         SerieParalelo3(1:64,l)=ParaleloSerie3(Npc+1+tao(2)+(total*(l-
1)):(total*l)+tao(2));
         end
   end
    if tao(3)>0
       D=padarray(ParaleloSerie4,[0 tao(3)],0,'post');

         for l=1:numerodesimbolos/log2(M)
          SerieParalelo4(1:64,l)=D(Npc+1+tao(3)+(total*(l-
1)):(total*l)+tao(3));
         end
    else
         for l=1:numerodesimbolos/log2(M)
         SerieParalelo4(1:64,l)=ParaleloSerie4(Npc+1+tao(3)+(total*(l-
1)):(total*l)+tao(3));
         end
    end
%---se realiza la FFT
SimboloOFDM2=fft(SerieParalelo2,NFFT);
SimboloOFDM3A=fft(SerieParalelo3,NFFT);
SimboloOFDM3B=fft(SerieParalelo4,NFFT);
%---ecualización en frecuencia
SImbolo3=SimboloOFDM2./hF;
SImbolo4=SimboloOFDM3A./hF;
SImbolo5=SimboloOFDM3B./hF;
%---remover muestras de guarda
remover_guarda=SImbolo3((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guardaA=SImbolo4((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guardaB=SImbolo5((Nspg/2)+1:Nsp-(Nspg/2),:);
%---se pasan los datos de paralelo a serie para demodularlos
ParaleloSerie2=reshape(remover_guarda,1,Nspd*numerodesimbolos/log2(M))
;
ParaleloSerie2A=reshape(remover_guardaA,1,Nspd*numerodesimbolos/log2(M
));
ParaleloSerie2B=reshape(remover_guardaB,1,Nspd*numerodesimbolos/log2(M
));
%---demodulación
datosRx= demodulate(Obj_demodulador,ParaleloSerie2);
datosRx1= demodulate(Obj_demodulador,ParaleloSerie2A);
```

```
datosRx2= demodulate(Obj_demodulador,ParaleloSerie2B);
%---demapeo de los datos
demapeo = gray2bin(datosRx,Modulacion,M);
datosRxA = de2bi(demapeo,'left-msb');
datosrx=reshape(datosRx',1,Nspd*numerodesimbolos);

demapeo2 = gray2bin(datosRx1,Modulacion,M);
datosRxB = de2bi(demapeo2,'left-msb');
datosrx2=reshape(datosRx1',1,Nspd*numerodesimbolos);

demapeo3 = gray2bin(datosRx2,Modulacion,M);
datosRxC = de2bi(demapeo3,'left-msb');
datosrx3=reshape(datosRx2',1,Nspd*numerodesimbolos);
%---cálculo de la BER
[errores BER1(i)]=biterr(datosTx,datosrx);
[errores BER2(i)]=biterr(datosTx,datosrx2);
[errores BER3(i)]=biterr(datosTx,datosrx3);
%-------------------------------------------------
end
%% grafica
close all; figure
%---grafica para canal AWGN
if TipoCanal==3

semilogy(EbNo,BER1,'b-',EbNo,BER2,'r-',EbNo,BER3,'m-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
axis([0 14 1e-6 1e-0]);
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada tao=1','BER simulada tao=5','BER simulada
tao=10','BER teórica'});

else
%---grafica para canales Rician o Rayleigh
semilogy(EbNo,BER1,'b-',EbNo,BER2,'r-',EbNo,BER3,'m-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada tao=1','BER simulada tao=5','BER simulada
tao=10','BER teórica'});
end
```

A.3. Código para Simular SCO

```
%% BER en sistema OFDM con ICI.
%El siguiente código simula el desempeño (BER) de un sistema OFDM
%que utiliza modulaciones BPSK, QPSK, 16-QAM sobre un canal Rician,
Rayleigh o AWGN cuando se ve
%afectado por la interferencia ICI que provoca el offset de reloj de
muestreo,
%a través de los siguientes pasos:
%   TRANSMISION
%   1-Crear los datos a enviar de forma aleatoria
%   2-mapearlos según el diagrama de constelación
%   3-adicionar codificación Gray
%   4-convertir los datos de serie a paralelo
%   5-introducir muestras de guarda
%   6-aplicar la IFFT sobre los datos mapeados
%   7-adicionar el prefijo cíclico
%
%
%   CANAL
%   8-realizar la convolución entre los símbolos que se crearon en
%   transmisión y el canal (cuando se trabaja con canales
multitrayecto)
%   9-adicionar el ruido AWGN
%
%   RECEPCION
%   10-conversion paralelo serie
%   11-efecto del Sampling Clock Offset
%   12-quitar el prefijo cíclico
%   13-realizar la FFT
%   14-hacer ecualización en frecuencia
%   15-quitar muestras de guarda
%   16-demodular
%   17-demapeo
%   18-calcaular  la BER
%% VARIABLES Y CONSTANTES DEL SISTEMA
clc;
clear all;
close all;
NFFT=64;                                        %tamaño de la FFT/IFFT
Nsp=64;                                         %numero de subportadoras
Nspd=52;                                        %numero de subportadoras
de datos
Nspg=12;                                        %numero de subportadoras
de guarda
Npc=16;                                         %numero de muestras que
conforman el prefijo cíclico
Ns=64;                                          %numero de muestras que
conforman un símbolo OFDM
modulaciones ={'BPSK','QPSK','16-QAM'};         %permite escribir el tipo
de modulacion en el titulo de las graficas
MOD=1;                                          %(1=BPSK, 2=QPSK, 3=16-
QAM)
tcanal={'RICIAN','RAYLEIGH','AWGN'};            %%permite escribir el tipo
de canal en el titulo de las graficas
TipoCanal=1;                                    %(1=RICIAN, 2=RAYLEIGH,
3=AWGN)
Ts=8e-7;Fs=1/Ts;                                %periodo/frecuencia de
muestreo
Modulacion='psk';                               %tipo de modulación (PSK,
QAM)
```

```
M=2;                                                %índice de modulación
numerodesimbolos = 10000;                           %cantidad de símbolos a
transmitir
factorK=5;                                          %factor K para el canal
RICE
beta=[0.005 0.01 0.015];                            %offset de muestreo (0,
+/-0.005, +/-0.01, +/-0.015)


%-----------------------------------------------------------------
if strcmp(Modulacion,'qam')== 1                     %Dependiendo del valor de
la variable Modulación crea los objetos
    Obj_modulador=modem.qammod(M);                  %Obj_modulador y
Obj_demodulador que requieren las funciones de
    Obj_demodulador=modem.qamdemod(M);              %modulación y demodulación
para PSK y QAM.
else
    Obj_modulador=modem.pskmod(M);
    Obj_demodulador=modem.pskdemod(M);
end
%% ------------------------------------------------------------------
EbNo=0:3:30;
EsNo=EbNo
+10*log10(Nspd/(NFFT+1))+10*log10(NFFT/(NFFT+Npc))+10*log10(log2(M));
%% transmisión-------------------------------------
%Fuente: Genera bits de forma aleatoria
datosTx=randint(1,Nspd*numerodesimbolos);
%Mapeo: mapea los bits segun el tipo de modulación con el código Gray
mapeo=bi2de(reshape(datosTx,log2(M),numerodesimbolos*Nspd/log2(M))','l
eft-msb');
mapeo_Gray = bin2gray(mapeo,Modulacion,M);
senal=modulate(Obj_modulador,mapeo_Gray);
%serie-paralelo: convertir de serie a paralelo para aplicar la IFFT
SerieParaleloTx=reshape(senal,Nspd,numerodesimbolos/log2(M));
%Subportadoras de guarda: inserta subportadoras de guarda para evitar
interferencia
insertar_guardas=[zeros(Nspg/2,numerodesimbolos/log2(M));SerieParalelo
Tx;zeros(Nspg/2,numerodesimbolos/log2(M))];
%transformada inversa de FOURIER: realiza la modulación en múltiples
subportadoras
SimboloOFDM=ifft(insertar_guardas,NFFT);
%Prefijo cíclico: para reducir la ISI
SimboloOFDM_PC=[SimboloOFDM(Ns-Npc+1:Ns,:);SimboloOFDM];
SimboloOFDM_TX=SimboloOFDM_PC;

%% canal-------------------------------------------------------------

switch TipoCanal

    case 1
        num_trayectos=3;% numero de trayectos canal
        %Trayectos: un trayecto directo y dos indirectos
        TrayectoDirecto=1*ones(1,numerodesimbolos/log2(M));
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rice
        canal = [TrayectoDirecto;TrayectoA;TrayectoB];
```

```
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
ecualización en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
        rt(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC(1:Ns+Npc,jj));
        end
        Rt=reshape(rt,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1,factorK);

    case 2
        num_trayectos=2;
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        %Canal: tipo de canal Rayleigh
        canal = [TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
ecualización en RX
        %Convolución: efecto del multitrayecto en la señal
[r(t)=h(t)*S(t)]
        for jj = 1:numerodesimbolos/log2(M)
        rt(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC(1:Ns+Npc,jj));
        end
        Rt=reshape(rt,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
        BERtheory = berfading(EbNo,Modulacion,M,1);
    case 3
        num_trayectos=1;
        hF=1;
        Rt=SimboloOFDM_TX;
        BERtheory = berawgn(EbNo,Modulacion,M,'nondiff');
end
%%
for i = 1:length(EsNo)
%Ruido: adicionar ruido AWGN a la señal, yt=h(t)*S(t)+ n(t) si el
canal
%Rice o Rayleigh y yt=S(t)+ n(t) si es awgn
yt=awgn(Rt,EsNo(i),'measured');
%% recepción
Yt=reshape(yt,Ns+Npc+(num_trayectos-1),numerodesimbolos/log2(M));
    %SCO: se muestrea la señal a intervalos de TsRx=Ts(1+beta)seg para
simular
    %el efecto del Sampling Clock Offset.
    if beta(1)==0
        SimboloRx=Yt;
    else
        TsRx=Ts*(1+beta(1));
        [a,b] = rat(TsRx/Ts,0.0001);
        SimboloRx=resample(Yt,a,b,10);
    end

     if beta(2)==0
        SimboloRx1=Yt;
    else
        TsRx1=Ts*(1+beta(2));
        [c,d] = rat(TsRx1/Ts,0.0001);
```

```
        SimboloRx1=resample(Yt,c,d,10);
     end
    if beta(3)==0
        SimboloRx2=Yt;
    else
        TsRx2=Ts*(1+beta(3));
        [f,g] = rat(TsRx2/Ts,0.0001);
        SimboloRx2=resample(Yt,f,g,10);
    end

%---Retirar el prefijo ciclico
SimboloRx(1:Npc,:)=[];
SimboloRx1(1:Npc,:)=[];
SimboloRx2(1:Npc,:)=[];
%---Transformada directa de Fourier: demodula la información de las
subportadoras
SimboloOFDMRx=fft(SimboloRx,NFFT);
SimboloOFDMRx1=fft(SimboloRx1,NFFT);
SimboloOFDMRx2=fft(SimboloRx2,NFFT);
%---Ecualización el frecuencia
SimboloOFDMRX=SimboloOFDMRx./hF;
SimboloOFDMRX1=SimboloOFDMRx1./hF;
SimboloOFDMRX2=SimboloOFDMRx2./hF;
%---retirar portadoras de guarda
remover_guardas=SimboloOFDMRX((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guardas1=SimboloOFDMRX1((Nspg/2)+1:Nsp-(Nspg/2),:);
remover_guardas2=SimboloOFDMRX2((Nspg/2)+1:Nsp-(Nspg/2),:);
%---Paralelo-Serie: transformar de paralelo a serie
ParaleloSerieRx=reshape(remover_guardas,1,(Nspd)*numerodesimbolos/log2
(M));
ParaleloSerieRx1=reshape(remover_guardas1,1,(Nspd)*numerodesimbolos/lo
g2(M));
ParaleloSerieRx2=reshape(remover_guardas2,1,(Nspd)*numerodesimbolos/lo
g2(M));
%---demodular datos
datos= demodulate(Obj_demodulador,ParaleloSerieRx);
datos1= demodulate(Obj_demodulador,ParaleloSerieRx1);
datos2= demodulate(Obj_demodulador,ParaleloSerieRx2);
%---demapeo: demapear la información para obtenerla del mismo tipo de
la fuente
demapeo_Gray=gray2bin(datos,Modulacion,M);
demapeo=de2bi(demapeo_Gray,'left-msb');
datosRx=reshape(demapeo',1,Nspd*numerodesimbolos);

demapeo_Gray1=gray2bin(datos1,Modulacion,M);
demapeo1=de2bi(demapeo_Gray1,'left-msb');
datosRx1=reshape(demapeo1',1,Nspd*numerodesimbolos);

demapeo_Gray2=gray2bin(datos2,Modulacion,M);
demapeo2=de2bi(demapeo_Gray2,'left-msb');
datosRx2=reshape(demapeo2',1,Nspd*numerodesimbolos);
%---calcular la tasa de errores
[errores BER(i)]=biterr(datosTx,datosRx);
[errores BER1(i)]=biterr(datosTx,datosRx1);
[errores BER2(i)]=biterr(datosTx,datosRx2);

end
%% GRAFICAS
close all; figure
if TipoCanal==3
%---grafica para canal AWGN
```

```
semilogy(EbNo,BER,'b-',EbNo,BER1,'r-',EbNo,BER2,'m-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
axis([0 14 1e-6 1e-0]);
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada beta=0.005','BER simulada beta=0.01','BER
simulada beta=0.015','BER teórica'});

else
%---grafica para canales Rician o Rayleigh
semilogy(EbNo,BER,'b-',EbNo,BER1,'r-',EbNo,BER2,'m-
',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
xlabel('Eb/No [dB]');
ylabel('BER');
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL ',
tcanal{TipoCanal}]);
legend({'BER simulada beta=0.005','BER simulada beta=0.01','BER
simulada beta=0.015','BER teórica'});
end
```

ANEXO B
EL PREFIJO CÍCLICO EN LA REDUCCIÓN DE LA INTERFERENCIA ISI

Una de las más importantes características de la tecnología OFDM es su robustez ante la interferencia ISI, esto se debe a la mayor duración del símbolo y al uso del prefijo cíclico. La duración del prefijo cíclico se selecciona mayor al máximo retardo de canal para que los efectos de interferencia producidos por los canales multitrayecto afecten a las muestras que componen el CP y no a la carga útil del símbolo. Además de la reducción de la interferencia algunos sistemas utilizan el prefijo también para tareas de sincronización pero originalmente fue creado para contrarrestar la ISI.
Las siguientes son algunas pruebas realizadas para verificar como el uso del prefijo cíclico puede ayudar a mantener la tasa de errores con valores cercanos a los teóricos. Los parámetros de simulación utilizados se muestran en a tabla B.1.

PARAMETROS DE SIMULACIÓN			
Tipo de canal	Rice con K=5		
Trayectos	**A**	**B**	**C**
Retardos	0	1.92 E-5 seg.	3.84 E-5 seg
Potencias	1	0.14	0.06
Duración prefijo cíclico	3.07 E-4 seg		

Tabla B.1. Parámetros para simular la BER

En la figura B.1 se compara la BER teórica de color verde y la BER simulada de color azul para una modulación BPSK, cuando la duración del PC es mayor al máximo retardo de canal.

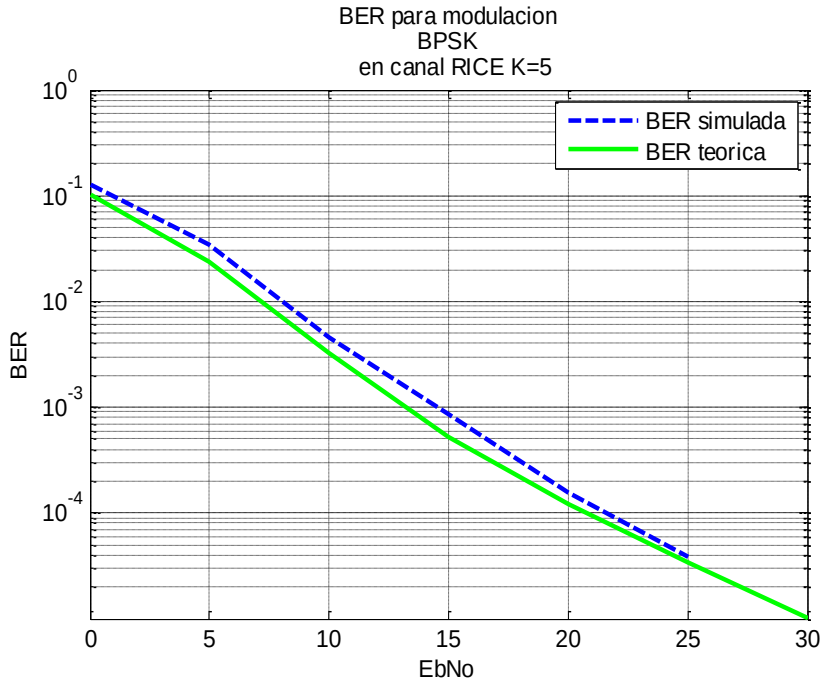

Fig. B.1. BER cuando el máximo exceso de retardo es menor a la duración del prefijo cíclico.

A medida que se incrementa el máximo retardo de canal, observamos que no se tiene efecto sobre la tasa de errores siempre que este sea menor a la duración del prefijo cíclico, como se observa en la figura B.2 obtenida utilizando los siguientes parámetros:

- Retardo de la trayectoria B: 1.5 E-4 seg.
- Retardo de la trayectoria C: 1.7 E-4 seg.
- Máximo exceso de retardo: 1.7 E-4 seg.

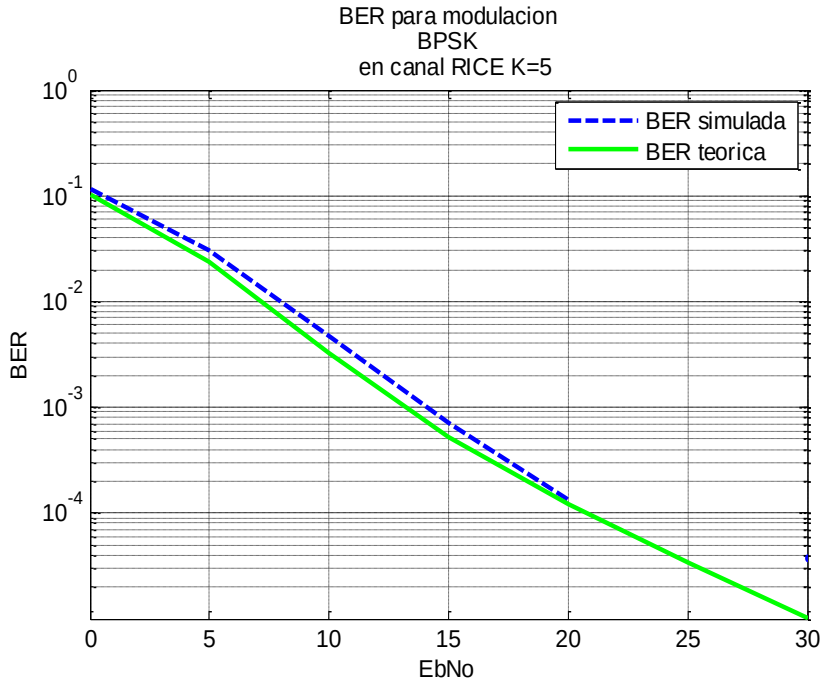

Fig. B.2. BER para máximo retardo de canal con duración igual a la mitad de la duración del prefijo cíclico.

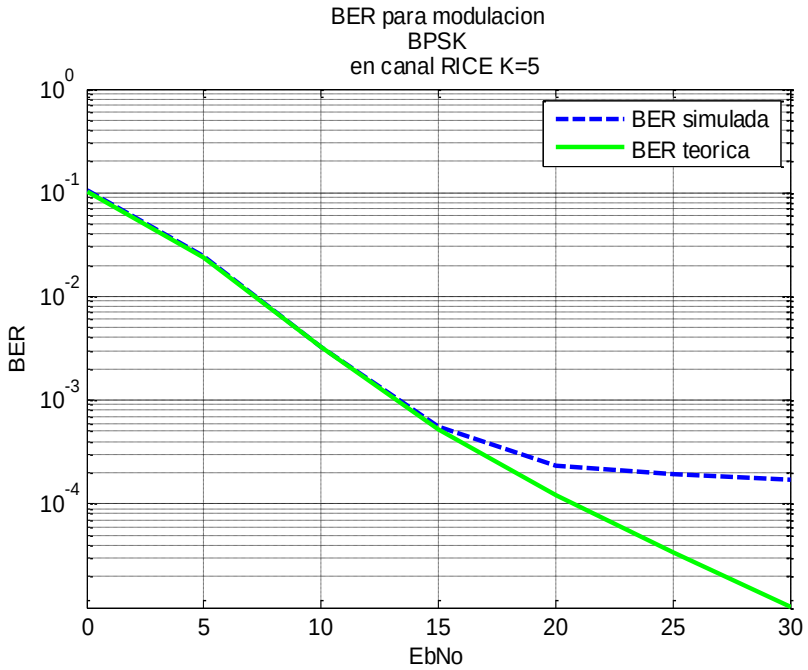

Fig. B.3. BER cuando el máximo exceso de retardo es mayor a la duración del prefijo cíclico.

Al tener un prefijo cíclico cuya duración sea un poco menor al máximo retardo de canal se tiene que la probabilidad de error se incrementa, como se observa en la figura B.3 a través de los parámetros:

- Retardo de la trayectoria B: 3.46 E-4 seg.
- Retardo de la trayectoria C: 3.65 E-4 seg.
- Máximo exceso de retardo: 3.65 E-4 seg.

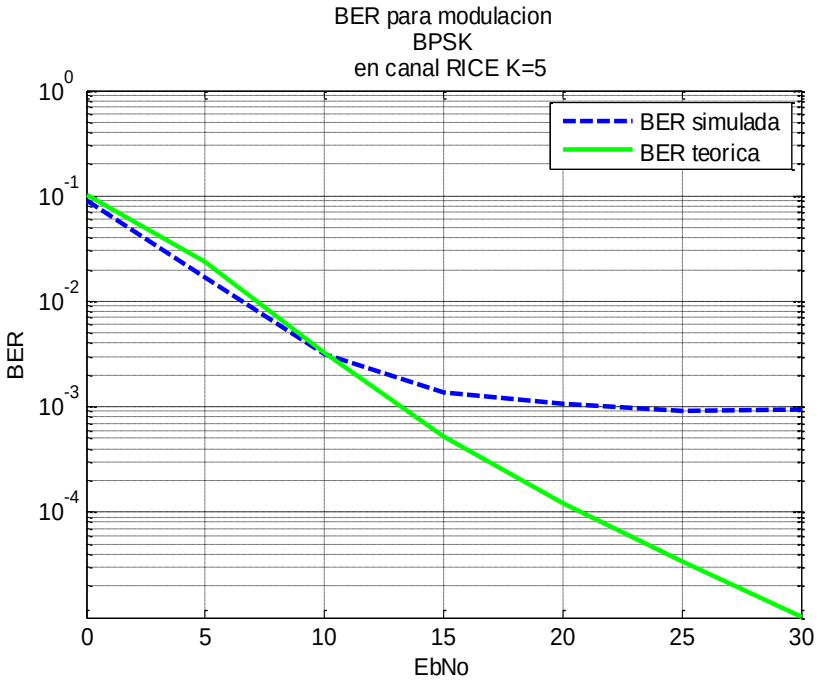

Fig. B.4. BER cuando el máximo exceso de retardo es mayor a la duración del prefijo cíclico.

A medida que se incrementa el máximo retardo de canal por encima de la duración del prefijo cíclico, la probabilidad de error se incrementa aun más, como se observa en la figura B.4, obtenida con los siguientes parámetros:

- Retardo de la trayectoria B: 6.92 E-4 seg.
- Retardo de la trayectoria C: 7.1 E-4 seg.
- Máximo exceso de retardo: 7.1 E-4 seg.

ANEXO C
EL PREFIJO CÍCLICO VS SNR

C.1. Código en Matlab Utilizado para la Simulación

```
%% VARIABLES Y CONSTANTES DEL SISTEMA
clc;
clear all;
close all;
NFFT=64;                                    %tamaño de la FFT/IFFT
Nsp=64;                                     %numero de subportadoras
Nspd=52;                                    %numero de subportadoras
de datos
Nspg=12;                                    %numero de subportadoras
de guarda
Npc=16;                                     %numero de muestras que
conforman el prefijo cíclico
Ns=64;                                      %numero de muestras que
conforman un símbolo OFDM
modulaciones ={'BPSK','QPSK','16-QAM'};     %permite escribir el tipo
de modulacion en el titulo de las graficas
MOD=1;                                      %(1=BPSK, 2=QPSK, 3=16-
QAM)
Ts=8e-7;Fs=1/Ts;                            %periodo/frecuencia de
muestreo
Modulacion='psk';                           %tipo de modulación (PSK,
QAM)
M=2;                                        %índice de modulación
numerodesimbolos = 1000;                    %cantidad de símbolos a
transmitir
factorK=5;                                  %factor K para el canal
RICE
%-------------------------------------------------------------------
if strcmp(Modulacion,'qam')== 1 %Dependiendo del valor de la variable
Modulacion crea los objetos
    Obj_modulador=modem.qammod(M);%Obj_modulador y Obj_demodulador que
requieren las funciones de
    Obj_demodulador=modem.qamdemod(M);%modulacion y demodulacion para
PSK y QAM.
else
    Obj_modulador=modem.pskmod(M);
    Obj_demodulador=modem.pskdemod(M);
end
%% relacion señal a ruido
EbNo=0:3:30;
SNR=EbNo
+10*log10(Nspd/(NFFT+1))+10*log10(NFFT/(NFFT+Npc))+10*log10(log2(M));
%% transmision
% se generan los datos a transmitir
datosTx=randint(1,Nspd*numerodesimbolos);
%Mapeo: mapea los bits segun el tipo de modulacion con el codigo Gray
qamdata=bi2de(reshape(datosTx,log2(M),numerodesimbolos*Nspd/log2(M))',
'left-msb');
maping = bin2gray(qamdata,Modulacion,M);
Mapeo = modulate(Obj_modulador,maping);
%serie-paralelo: convertir de serie a paralelo para aplicar la IFFT
SerieParaleloTX = reshape(Mapeo,Nspd,numerodesimbolos/log2(M));
%Subportadoras de guarda: inserta subportadoras de guarda para evitar
interferencia
insertar_guardas=[zeros(Nspg/2,numerodesimbolos/log2(M));SerieParalelo
TX;zeros(Nspg/2,numerodesimbolos/log2(M))];
```

```
%transformada inversa de FOURIER: realiza la modulacion en multiples
subportadoras
SimboloOFDM=ifft(insertar_guardas,NFFT);
%se inserta el prefijo ciclico
SimboloOFDM_PC=[SimboloOFDM(Ns-Npc+1:Ns,:);SimboloOFDM];
% ----------------------------
%% canal multitrayecto Rician
 num_trayectos=3;% numero de trayectos canal
        %Trayectos: un trayecto directo y dos indirectos
        TrayectoDirecto=1*ones(1,numerodesimbolos/log2(M));
        TrayectoA=(sqrt(0.07))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        TrayectoB=(sqrt(0.03))*(randn(1,numerodesimbolos/log2(M)) +
j*randn(1,numerodesimbolos/log2(M)));
        canal = [TrayectoDirecto;TrayectoA;TrayectoB];
        hF = fft(canal,NFFT);%respuesta en frecuencia, para hacer
equalizacion en RX
   for jj = 1:numerodesimbolos/log2(M)
       %se realiza la convolucion entre el canal y la señal
       desvanecimiento(1:Ns+Npc+num_trayectos-1,jj) =
conv(canal(1:num_trayectos,jj),SimboloOFDM_PC(1:Ns+Npc,jj));
   end
%% --------------
ParaleloSerie=reshape(desvanecimiento,1,(Ns+Npc+num_trayectos-
1)*numerodesimbolos/log2(M));
for i = 1:length(SNR)

%% awgn
%---se adiciona ruido awgn
yt=awgn(ParaleloSerie,SNR(i),'measured');
%% recepsion
SerieParalelo2=reshape(yt,Ns+Npc+num_trayectos-
1,numerodesimbolos/log2(M));
%---se remueve el prefijo cíclico
SerieParalelo2(1:Npc,:)=[];
%---se realiza la FFT
SimboloOFDM2=fft(SerieParalelo2,NFFT);
%---equalizacion en frecuencia
Simbolo3=SimboloOFDM2./hF;
%---remover muestras de guarda
remo_guarda=Simbolo3((Nspg/2)+1:Nsp-(Nspg/2),:);
%---se pasan los datos de paralelo a serie para demodularlos
ParaleloSerie2=reshape(remo_guarda,1,(Nspd)*numerodesimbolos/log2(M));
%---demodulación
datosRx= demodulate(Obj_demodulador,ParaleloSerie2);%demodulación de
los datos
%---demapeo
demaping = gray2bin(datosRx,Modulacion,M);
datosRxA = de2bi(demaping,'left-msb');
datosrx=reshape(datosRxA',1,Nspd*numerodesimbolos);
%---calculo de la BER
[errores BER(i)]=biterr(datosTx,datosrx);

end
%% grafica
BERtheory = berfading(EbNo,Modulacion,M,1,5);% calcula la BER teorica
con canal Rician y factor K de 5
close all; figure
semilogy(EbNo,BER,'b--',EbNo,BERtheory,'g-','LineWidth',2);
hold on;
grid;
```

```
xlabel('Eb/No [dB]');
ylabel('BER');
axis([0 30 1e-5 1e-0]);
title(['BER CON MODULACIÓN ',modulaciones{MOD},' Y CANAL MULTITRAYECTO
RICIAN']);
legend({'BER con PC=16','BER teorica'});
```

C.2. Resultados

El anexo C tiene como finalidad mostrar la influencia que tiene el prefijo cíclico en la relación señal al ruido, y por ende en el aumento o la reducción de la tasa de error de bit, con este fin se realizaron simulaciones con diferentes valores de prefijo cíclico en un sistema OFDM libre de interferencia ICI y en un canal multitrayecto tipo Rician con factor k de 5.

En las figuras de este anexo, la línea punteada de color azul representa el tamaño del prefijo cíclico, mientras que la línea de color verde representa la grafica teórica de un canal multitrayecto anteriormente mencionado y con modulación QPSK.

En la figura C.1. Se observa que la probabilidad de error aumenta, esto se debe a que esta simulación se realizo sin prefijo cíclico, por lo que el máximo exceso de retardo del canal es mayor lo que produce interferencia ISI.

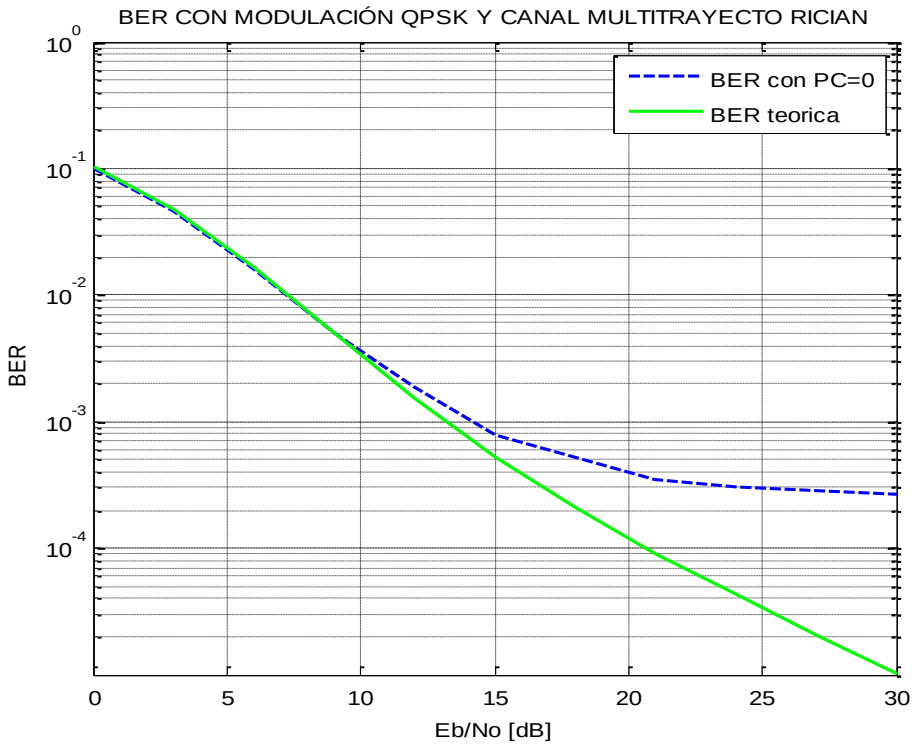

Fig. C.1. BER en un sistema OFDM libre de interferencia ICI y sin el uso del prefijo cíclico.

En las figuras C.2 y C.3 se muestran las simulaciones realizadas con prefijo cíclico de 4 y 8 muestras respectivamente, como se puede observar la probabilidad de error es similar a la teórica, siendo mas exacta la simulación que se muestra en la figura C.2.

La figura C.4 se simulo con un valor de prefijo cíclico de 16 muestras, se observa que sigue la tendencia de la línea teórica, pero con una probabilidad de error un poco

mayor, esto se debe a la reducción de la relación señal al ruido como se miró en la ecuación 4.1.

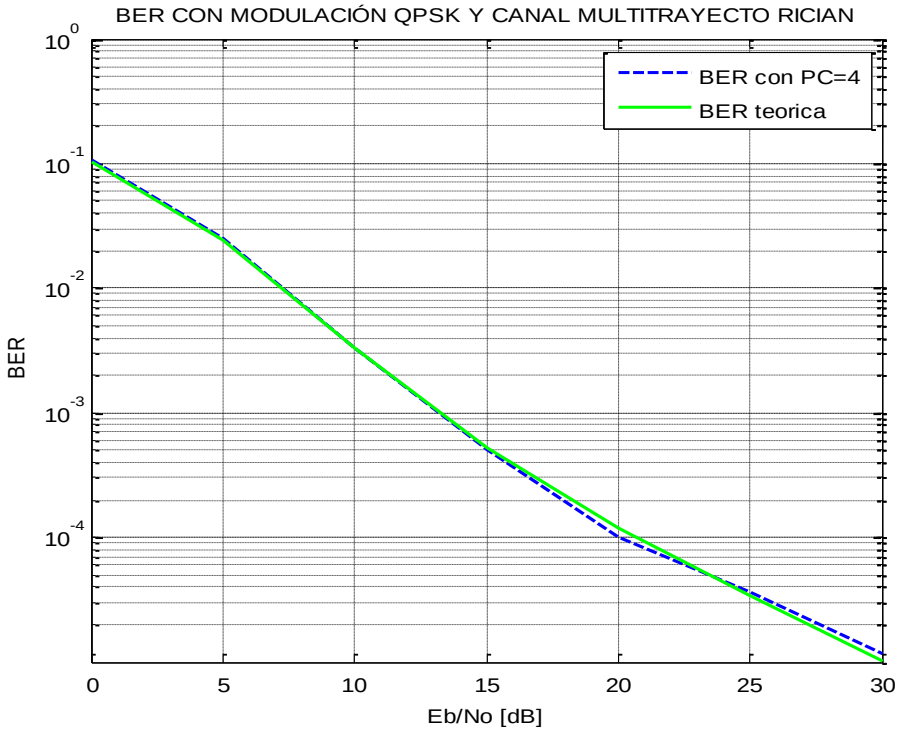

Fig. C.2. BER en un sistema OFDM libre de interferencia ICI y con prefijo cíclico de 4 muestras.

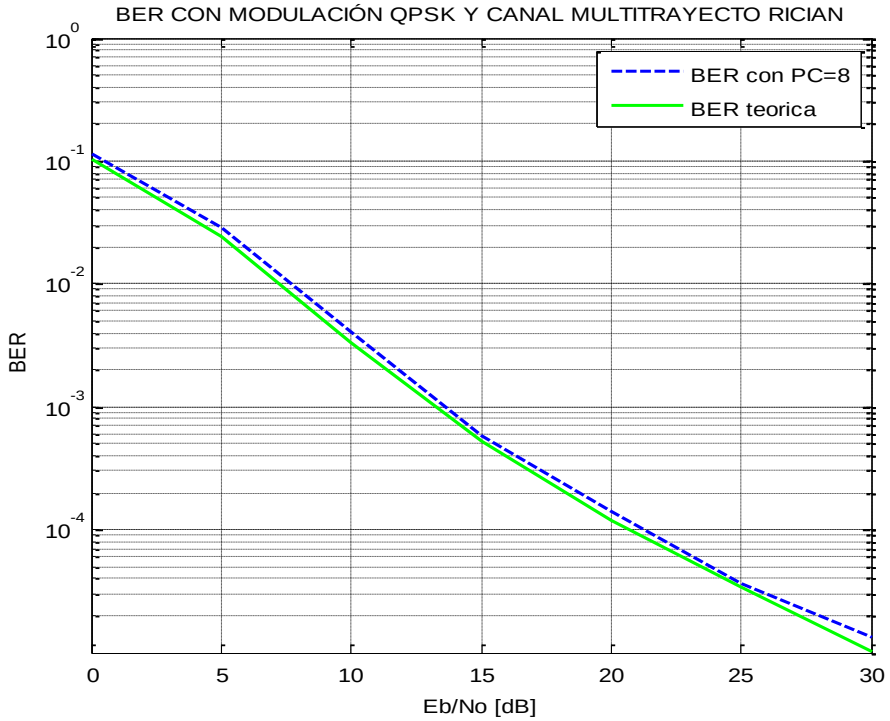

Fig. C.3. BER en un sistema OFDM libre de interferencia ICI y con prefijo cíclico de 8 muestras.

En la figura C.5 se puede observar un mayor alejamiento de la línea que simula el valor del prefijo cíclico de 32 muestras.

Con estas simulaciones se pudo concluir que un aumento en el valor de las muestras del prefijo cíclico provoca un aumento de la probabilidad de error, debido a la disminución de la relación señal al ruido del sistema OFDM.

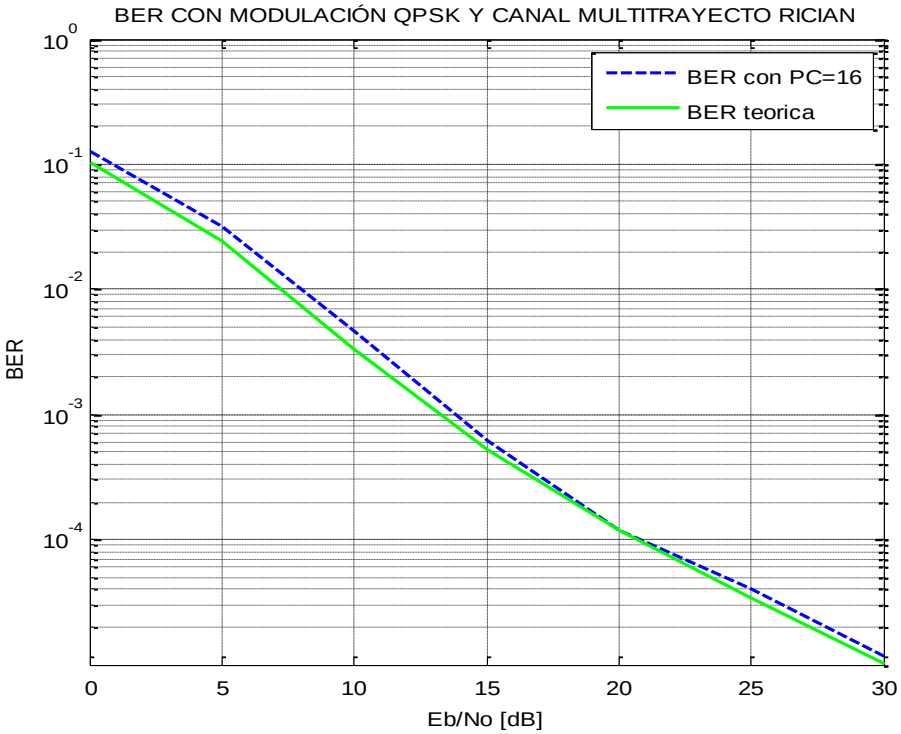

Fig. C.4. BER en un sistema OFDM libre de interferencia ICI y con prefijo cíclico de 16 muestras.

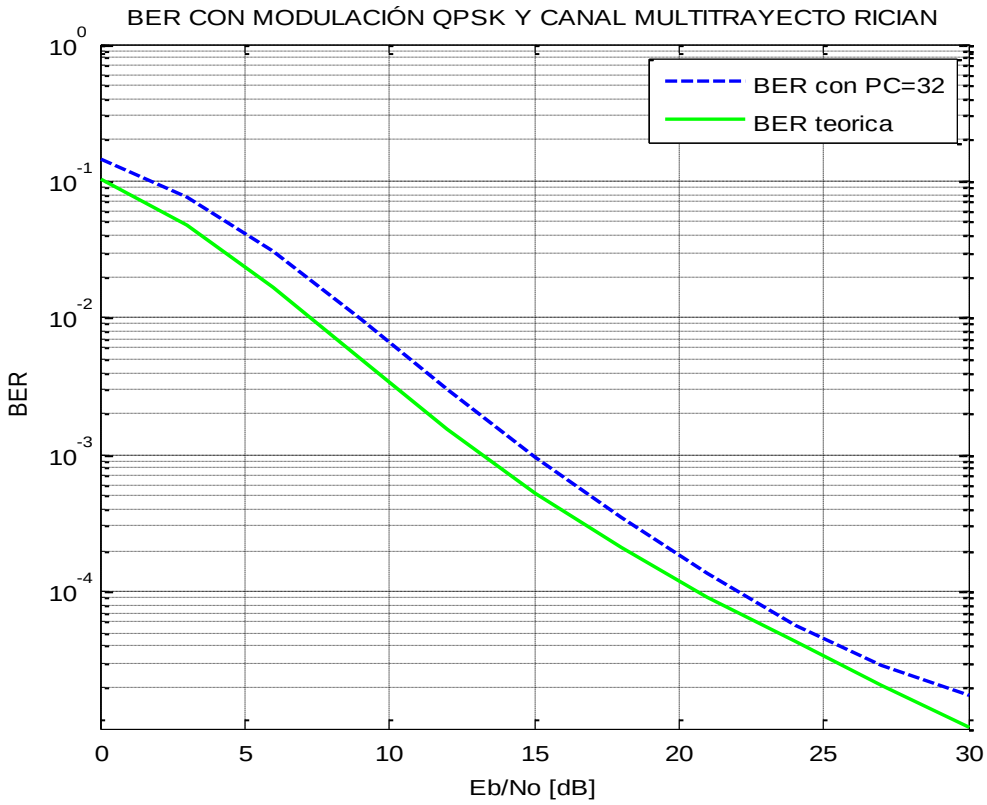

Fig. C.5. BER en un sistema OFDM libre de interferencia ICI y con prefijo cíclico de 32 muestras.