7. APÉNDICE

A. VALORES ÓPTIMOS DE Φ Y U EN (3.38)

Si ***M = P-L***, el producto óptimo $\boldsymbol{U\Phi}$ puede encontrarse directamente usando un argumento directo del multiplicador de Lagrange. En casos más generales donde ***M ≤ P-L***, es más conveniente primero encontrar la óptima Φ para una opción dada ***U*** y después encontrar la matriz óptima ***U*** **.**

Para una matriz ***U*** fija, el problema de optimización en (3.38) puede ser reparametrizado en términos de $\Gamma = \Phi^2$ como

$$\min_{\Gamma} \ \mathrm{tr}(\Gamma^{-1}\boldsymbol{Z}) \ \text{ sujeto a } \mathrm{tr}(\Gamma) \le p_0 . \tag{7.1}$$

Aquí, como en (3.34), $\boldsymbol{Z} = \boldsymbol{Z}(\boldsymbol{U}) = \left(\begin{bmatrix} \boldsymbol{I}_M & \boldsymbol{0} \end{bmatrix} \boldsymbol{U}^H \boldsymbol{H}^H \boldsymbol{H}\boldsymbol{U} \begin{bmatrix} \boldsymbol{I}_M \\ \boldsymbol{0} \end{bmatrix} \right)^{-1}$.

La función Lagrangiana para este problema es

$$L(\Gamma.\mu) = \mathrm{tr}(\Gamma^{-1}\boldsymbol{Z}) + \mu(\mathrm{tr}(\Gamma^{-1}\boldsymbol{Z}) - p_0) \tag{7.2}$$

donde μ es el multiplicador de Lagrange. Igualando las derivadas de (7.2) con respecto a Γ y μ a cero, obtenemos las siguientes condiciones necesarias para la optimalidad:

$$\begin{aligned} -(\Gamma^{-1}\boldsymbol{Z}\Gamma^{-1})^{\boldsymbol{T}} + \mu \boldsymbol{I}_M &= 0 \\ \boldsymbol{tr}(\Gamma) &= p_0 \end{aligned} \tag{7.3}$$

Para satisfacer estas condiciones necesarias, se requiere que $\Gamma^2 = (1/\mu)\boldsymbol{Z}^{\boldsymbol{T}}$ y así

$$\Gamma = \frac{1}{\sqrt{\mu}} \left(\begin{bmatrix} \boldsymbol{I}_M & \boldsymbol{0} \end{bmatrix} \boldsymbol{U}^{\boldsymbol{H}}\boldsymbol{W}\Lambda^{-1}\boldsymbol{W}^H\boldsymbol{U} \begin{bmatrix} \boldsymbol{I}_M \\ \boldsymbol{0} \end{bmatrix} \right)^{-1/2} \tag{7.4}$$

donde $\boldsymbol{W}\Lambda\boldsymbol{W}^H$ es la descomposición en autovalores y vectores de $(\boldsymbol{H}^{\boldsymbol{H}}\boldsymbol{H})^{-1}$ como en (3.32), con los autovalores ordenados descendentemente. Sin embargo, como $\Gamma = \Phi^2$ y Φ es diagonal, la matriz óptima Γ debe ser diagonal y semidefinida positiva. Por lo tanto, como ***U*** es una matriz unitaria, el producto óptimo $\boldsymbol{U}^H\boldsymbol{W}$ debe ser una matriz de permutación denominada ***P***. Así

$\boldsymbol{U}_{\text{opt}} = \boldsymbol{WP}$. Haciendo la dependencia de $\boldsymbol{Z}$ con $\boldsymbol{U}$ explicita, el objetivo resultante en (7.1) es

$$\text{tr}(\Gamma_{\text{opt}}^{-1}\boldsymbol{Z}(\boldsymbol{WP})) = \sqrt{\mu}\sum_{i=1}^{M}\left[\boldsymbol{P}^{\boldsymbol{T}}\Lambda^{1/2}\boldsymbol{P}\right]_{ii} \tag{7.5}$$

La matriz óptima $\boldsymbol{P}$, coloca los $\boldsymbol{M}$ elementos más pequeños de $\Lambda^{1/2}$ en la esquina superior izquierda del producto matricial $\boldsymbol{P}^{\boldsymbol{T}}\Lambda^{1/2}\boldsymbol{P}$. Como los elementos de Λ están ordenados en orcen descendente, una $\boldsymbol{P}$ óptima es

$$\boldsymbol{P}_{\text{opt}} = \begin{bmatrix} 0 & \boldsymbol{I}_{(P\text{-}L\text{-}M)} \\ \boldsymbol{I}_{M} & 0 \end{bmatrix} \tag{7.6}$$

y así, una óptima $\boldsymbol{U}$ es $\boldsymbol{U}_{\text{opt}} = \boldsymbol{WP}_{\text{opt}} = [\boldsymbol{W}_{M}\boldsymbol{W}_{M}^{\perp}]$ como en (3.41). Por lo tanto, $\Gamma_{\text{opt}} = \Lambda_{M}^{1/2}/\sqrt{\mu}$ donde $\Lambda_{M} = \text{diag}(\lambda_{P\text{-}L\text{-}M+1.\ldots.}\lambda_{P\text{-}L})$. Escogiendo $\mu = \text{tr}(\Lambda_{M}^{1/2})/p_{0}$ tal que (7.3) se satisface, tenemos que

$$\Phi_{\text{opt}} = \Gamma_{\text{opt}}^{1/2} = \sqrt{\frac{p_{0}}{\text{tr}(\Lambda_{M}^{1/2})}}\Lambda_{M}^{1/4}. \tag{7.7}$$

Como $\boldsymbol{Z}(\boldsymbol{U})$ es semidefinida positiva por construcción, $\lambda_{i} \geq 0$ y entonces, los elementos diagonales de Φ son reales y no negativos como se requiere. Con todo tenemos (3.40).

B. DEMOSTRACIÓN DE LA PROPIEDAD 3

Como el producto matricial $\boldsymbol{GHF}$ es cuadrado y Hermético simétrico, véase propiedad P1, puede ser descompuesto como $\boldsymbol{GHF} = \boldsymbol{ABA}^{H}$, donde $\boldsymbol{B}$ es una matriz diagonal que contiene los autovalores de $\boldsymbol{GHF}$ y $\boldsymbol{A}$ es una matriz unitaria cuadrada de orden $\boldsymbol{M}$ cuyas columnas son los correspondientes autovectores de $\boldsymbol{GHF}$. Por lo tanto, $[\boldsymbol{GHF}]_{\boldsymbol{mm}} \geq \sum_{\boldsymbol{i}=1}^{\boldsymbol{M}}|a_{mi}|^{2}b_{i}$. Como $\boldsymbol{A}$ es unitaria, $\sum_{\boldsymbol{i}=1}^{\boldsymbol{M}}|a_{mi}|^{2} = 1$ y así, $min(b_{i}) \leq [\boldsymbol{GHF}]_{\boldsymbol{mm}} \leq max(b_{i})$, donde $min(b_{i})$ y $max(b_{i})$ denotan el mínimo y máximo elemento diagonal en $\boldsymbol{B}$, respectivamente; i.e., el mínimo y máximo autovalor del producto $\boldsymbol{GHF}$. Consecuentemente $0 \leq [\boldsymbol{GHF}]_{\boldsymbol{mm}} \leq 1$ si y solo si $0 \leq b_{i} \leq 1$, que es equivalente a tener $\boldsymbol{0} \preceq \boldsymbol{GHF} \preceq \boldsymbol{I}$, donde , para matrices Hermíticas simétricas $\boldsymbol{X}$ e $\boldsymbol{Y}$, $\boldsymbol{X} \preceq \boldsymbol{Y}$ denota que la matriz resultante de $\boldsymbol{Y}$ - $\boldsymbol{X}$ es semidefinida positiva.

Para probar que $\boldsymbol{GHF} \preceq \boldsymbol{I}$, debemos recordar que de la propiedad P1, teníamos que $\boldsymbol{GHF} = \boldsymbol{F}^H\boldsymbol{H}^H(\boldsymbol{R}_{vv} + \boldsymbol{HFF}^H\boldsymbol{H}^H)^{-1}\boldsymbol{HF}$. Así

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{F}^H\boldsymbol{H}^H \\ \boldsymbol{HF} & \boldsymbol{R}_{vv} + \boldsymbol{HFF}^H\boldsymbol{H}^H \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{HF} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{F}^H\boldsymbol{H}^H \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{vv} \end{bmatrix} \succeq \boldsymbol{0}, \tag{7.8}$$

entonces, aplicando el teorema complemento de Schur, [9,p.472], $\boldsymbol{I} - \boldsymbol{F}^H\boldsymbol{H}^H(\boldsymbol{R}_{vv} + \boldsymbol{HFF}^H\boldsymbol{H}^H)^{-1}\boldsymbol{HF} \succeq \boldsymbol{0}$, con lo cual se deduce que $\boldsymbol{GHF} \preceq \boldsymbol{I}$

Por otra parte, de la propiedad P2, como la suma de dos matrices semidefinidas positivas es también semidefinida positiva, tenemos que

$$\boldsymbol{GHF} = (\boldsymbol{GHF})(\boldsymbol{GHF})^H + \sigma^2\boldsymbol{GR}_{vv}\boldsymbol{G}^H \succeq \boldsymbol{0}. \tag{7.9}$$

Como resultado podemos concluir que $0 \le [\boldsymbol{GHF}]_{mm} \le 1 \;\; \forall m \in [1, M]$.

C. LA EXPRESIÓN MSE PARA ECUALIZACIÓN MMSE

Con la suposición que hemos venido asumiendo a lo largo de este proyecto, $\mathrm{E}\{\boldsymbol{ss}^{\mathrm{H}}\} = \boldsymbol{I}$ y $\mathrm{E}\{\boldsymbol{vv}^{\mathrm{H}}\} = \boldsymbol{R}_{vv}$, el error cuadrático medio de los símbolos ecualizados puede escribirse como

$$\begin{aligned} \mathrm{MSE} &= \mathrm{tr}\left((\boldsymbol{GHF} - \boldsymbol{I})(\boldsymbol{GHF} - \boldsymbol{I})^{\boldsymbol{H}}\right) + \mathrm{tr}\left(\boldsymbol{GR}_{vv}\boldsymbol{G}^{\boldsymbol{H}}\right) \\ &= \mathrm{tr}\left((\boldsymbol{GHF})(\boldsymbol{GHF})^{\boldsymbol{H}} + \boldsymbol{GR}_{vv}\boldsymbol{G}^{\boldsymbol{H}}\right) - \mathrm{tr}\left(\boldsymbol{GHF} + (\boldsymbol{GHF})^{\boldsymbol{H}} + \boldsymbol{I}\right) \end{aligned} \tag{7.10}$$

Usando las propiedades P1 y P2, la expresión anterior puede simplificarse como sigue:

$$\begin{aligned} \mathrm{MSE} &= \mathrm{tr}(\boldsymbol{GHF}) - \mathrm{tr}(\boldsymbol{GHF}) - \mathrm{tr}\left((\boldsymbol{GHF})^{\boldsymbol{H}}\right) + \mathrm{tr}(\boldsymbol{I}) \\ &= M - \mathrm{tr}(\Gamma) \end{aligned} \tag{7.11}$$

donde $\boldsymbol{GHF}^{\boldsymbol{H}} = \boldsymbol{U}\Gamma\boldsymbol{U}^{\boldsymbol{H}}$ de acuerdo a (4.20).

D. PRECODIFICADORES MBER PARA SEÑALIZACIÓN QAM CUADRADA DE ORDEN SUPERIOR

Usando la notación seguida en la Sección que precede a este apéndice, tal y como se indicó en la quinta de sus observaciones, para minimizar la cota inferior de la BER en el caso de constelaciones QAM cuadradas de orden superior ($k > 4$) es necesario resolver el siguiente problema de optimización,

$$\min_{\boldsymbol{\Phi},\boldsymbol{V}} \quad \mathrm{tr}\left(-\boldsymbol{\Gamma}\right) \tag{7.12.a}$$

$$\text{Sujeto a } \mathrm{tr}\left(\boldsymbol{\Phi}^2\right) \le p_0 \tag{7.12.b}$$

$$\left[\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{U}^{\boldsymbol{H}}\right]_{\boldsymbol{mm}} \ge \boldsymbol{c}_2 \tag{7.12.c}$$

donde $\boldsymbol{c}_2$ se definió en (4.42). El lema 1 que se desarrollo en el caso ZF, establece que existe una matriz unitaria $\boldsymbol{U}$ que satisface la restricción (7.12.c) si y solo si $\mathrm{tr}\left(\boldsymbol{\Gamma}\right) \ge M\boldsymbol{c}_2$, en ese caso la elección $\boldsymbol{U} = \boldsymbol{D}_{\boldsymbol{M}}$ es suficiente. Por tanto una solución al problema formulado en (7.12) puede ser obtenida resolviendo primero (4.30). Si al resolver, se cumple $\mathrm{tr}\left(\boldsymbol{\Gamma}\right) \ge M\boldsymbol{c}_2$ entonces la opción $\boldsymbol{U} = \boldsymbol{D}_{\boldsymbol{M}}$ satisface (7.12.c) y además tal $\boldsymbol{U}$ consigue alcanzar la cota inferior minimizada de la BER, por lo que el precodificador MBER diseñado para 4-QAM, resulta ser un precodificador para mínima probabilidad de error válido para señalización *k*-aria QAM cuadrada, en dichas condiciones.

Si por el contrario $\mathrm{tr}\left(\boldsymbol{\Gamma}\right) < M\boldsymbol{c}_2$ (y $\mathrm{tr}\left(\boldsymbol{\Gamma}\right) > M\boldsymbol{c}_1$) entonces la expresión para la BER en (4.40) no es convexa y no puede obtenerse un precodificador MBER con los métodos empleados en secciones previas.

Usando (4.32) y (4.33), el test $\mathrm{tr}\left(\boldsymbol{\Gamma}\right) \ge M\boldsymbol{c}_2$ puede convertirse en una cota inferior en la SNR de bloque que puede ser obtenida sin necesidad de resolver (4.30).

E. CÓDIGO MATLAB

❖ Ejemplo1_ZP.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros, ecualización Cero-Forzado%
%  y detección por umbral, sobre un canal con buena respuesta frecuencial %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all               % Eliminamos todas las variables existentes
close all               % Cerramos todas las ventanas

% Respuesta impulsiva del canal
h=[0.3038+0.2554i 0.5056+0.5587i 0.2855+ 0.0035i 0.2834+0.1843i 0.2793+0.0305i];
L=length(h)-1;          % Longitud Respuesta impulsiva del canal
SNR=0:1:18;             % Rango para la Relación Señal-Ruido
M=32;                   % Tamaño de Bloque inicial de símbolos
Po=1;                   % Restricción de potencia de bloque a transmitir
NumSymb=2*M*1e4;        % Número de símbolos a transmitir
P=36;                   % P>=M+L Tamaño final de bloque a transmitir

figure(1)               % Respuesta Impulsiva del Canal
x=0:M-1;
stem(x,abs(fft(h,M)))
axis([-1 32 0 2])
Xlabel('Índice del subcanal, i');
YLabel('|H(exp(j2pi*i/(P-L))|');

% Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm'); % B es Lambda optima M*M
[Y,C]=eigs(inv(Hzp'*Hzp),P-L-M,'lm');
PhiOpt=sqrt(Po/trace(B^0.5))*(B^0.25);
UOpt=[X Y];             % X es Wm
Dm=DFTMtx(M);           % Construimos la matriz DFT normalizada
FOpt=X*PhiOpt*Dm;       % Construimos F óptima
FOptZP=FOpt;

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

figure(2)               % BER Precodificadores ZP

% Dibujamos la BER para MBER-ZP y la Pe Teórica
[BER,Pe]=BerZP(FOpt,Hzp,B,M,P,NumSymb,SymbSec,SNR,1);
semilogy(SNR,BER,'-kd',SNR,Pe);
axis([0 SNR(length(SNR)) 1e-5 1])
grid
Xlabel('SNR(dB)');
YLabel('BER(ZP)');
hold on

% Construimos F ZP-OFDM
Fzp_ofdm=sqrt(Po/M)*Dm'*eye([M M]);

% Dibujamos la BER para ZP-OFDM
```

```
BER=BerZP(Fzp_ofdm,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-r+');
clear BER

% Construimos F MMSE-ZP-I
Fzp_mmse_zf_i=X*PhiOpt;

% Dibujamos la BER para MMSE-ZP-I
BER=BerZP(Fzp_mmse_zf_i,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-c*');
clear BER

% Construimos F ZP-MSNR
Fzp_msnr=sqrt(Po/trace(B))*X*(B^0.5);

% Dibujamos la BER para ZP-MSNR
BER=BerZP(Fzp_msnr,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-m<');
clear BER
hold off
legend('ZP-MBER','Pe-Teorica','ZP-OFDM','ZP-MMSE-ZF-I','ZP-MSNR',0);
SNRc=10*log10(3/(M*P)*abs(trace(B^0.5))^2);

% Dibujamos los elementos diagonales de GG' para todos los precodificadores
figure(3)
subplot(4,1,1); PlotDiag(Hzp,FOpt,'d','ZP-MBER')
subplot(4,1,2); PlotDiag(Hzp,Fzp_ofdm,'o','ZP-OFDM')
subplot(4,1,3); PlotDiag(Hzp,Fzp_mmse_zf_i,'<','ZP-MMSE-ZF-I')
subplot(4,1,4); PlotDiag(Hzp,Fzp_msnr,'*','ZP-MSNR')
Xlabel('Índice de los elementos diagonales de GG^H')
```

Published with MATLAB® 7.0

❖ BerZP.m

```
function [BER,Pe]=BerZP(F,H,B,M,P,NumSymb,SymbMtx,Ro,PeT)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para un rango de SNR, en un sistema de comunicaciones con
%       precodificador, transmisión por bloques con relleno de ceros (ZP),
%       ecualización zero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (ZP)
%       H: Matriz del Canal (ZP)
%       B: Matriz Lambda Óptima de autovalores.
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR
%       PeT: Bandera para indicar si se calcula la BER Teórica


% Matriz de ecualización
G=pinv(H*F);
clear k;
```

```
% Recorremos una tabla para los valores SNR.
for k=1:length(Ro)

        NoBlock=1/(10^(0.1*Ro(k)));         %Ruido de Bloque
        Var=NoBlock/P;                      %Varianza de Ruido

        % Bloques de M Símbolos a bloques de P (ZP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos
        SymbRx=SymbTx+Noise;

        % Secuencia de símbolos ecualizados
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));

        % Calculamos la BER
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M
                % Si parte real e imaginaria incorrecta, símbolo recibido con
2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido con
1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end
        BER(k)=bitsqam_KO/(2*NumSymb);

        %Probabilidad de error de bit mínima teórica (CP)
        if PeT==1
           Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace(B^0.5))^2)));
        end
end
```

Published with MATLAB® 7.0

❖ DFTMtx.m

```
function dft=DFTMtx(N)

% DESCRIPCIÓN:
%       Esta función calcula la matriz DFT normalizada de orden N
%
% ENTRADAS:
%       N: Orden de la Matriz
```

```
% Construimos la matriz DFT normalizada
Dn=zeros([N N]);
for k=1:N
    for n=1:N
        Dn(k,n)=1/sqrt(N)*exp(-i*2*pi*(k-1)*(n-1)/N);
    end
end
dft=Dn;
```

Published with MATLAB® 7.0

❖ BerZPT.m

```
function Pe=BerZPT(B,M,P,Ro)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit teórica para
%       un rango de SNR, en un sistema de comunicaciones con precodificador,
%       transmisión por bloques con relleno de ceros (ZP), ecualización
%       cero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       B: Matriz Lambda óptima de autovalores (ZP)
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       Ro: Rango para la SNR

% Recorremos una tabla para los valores SNR.
for k=1:length(Ro)
        NoBlock=1/(10^(0.1*Ro(k)));
        Var=NoBlock/P;

% Probabilidad de error de bit mínima teórica (ZP)
        Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace(B^0.5))^2)));
end
```

Published with MATLAB® 7.0

❖ PlotDiag.m

```
function diagelements=PlotDiag(Mtx1,Mtx2,Marker,Leyenda)

% DESCRIPCIÓN:
%       Función que dibuja los elementos diagonales de la matriz de
%       ecualización
%
% ENTRADAS:
%       Mtx1: Matriz del canal
%       Mtx2: Matriz del precodificador
%       Marker: Marca especial que representa los valores en la figura
%       Leyenda: Parámetro que etiqueta la figura obtenida

% Cálculo de la matriz de ecualización
Matrix=pinv(Mtx1*Mtx2);

% Representación de los elementos diagonales de la matriz de ecualización
```

```
diagelements=stem(1:length(diag(Matrix*Matrix')),(abs(diag(Matrix*Matrix')))
.',Marker);
Ylabel(Leyenda);
```

Published with MATLAB® 7.0

❖ BerCP.m

```
function [BER,Pe]=BerCP(F,H,B,M,P,L,NumSymb,SymbMtx,Ro,PeT)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para un rango de SNR, en un sistema de comunicaciones con
%       precodificador, transmisión por bloques con código cíclico (CP),
%       ecualización cero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (CP)
%       H: Matriz del Canal (CP)
%       B: Matriz Lambda Óptima de autovalores.
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       L: Longitud de la respuesta impulsiva del canal
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR
%       PeT: Bandera para indicar si se calcula la BER Teórica


% Matriz de ecualización
G=pinv(H*F);

clear k;
% Recorremos una tabla para los valores de SNR.
for k=1:length(Ro)

        NoBlock=1/(10^(0.1*Ro(k)));  % Ruido de Bloque
        Var=NoBlock/P;               % Varianza de Ruido

        % Bloques de M Símbolos a bloques de P (CP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos (CP)
        Rcp=[zeros([P-L L]) eye([P-L P-L])];
        SymbRx=SymbTx+(Rcp*Noise.').';

        % Secuencia de símbolos ecualizados
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M
                % Si parte real e imaginaria incorrecta, símbolo recibido con
2 bits erróneos
```

```
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido con
1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end
        BER(k)=bitsqam_KO/(2*NumSymb);

        % Probabilidad de error de bit mínima teórica (CP)
        if PeT==1
         Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace((inv(B'*B))^0.5))^2)));
        end

end
```

Published with MATLAB® 7.0

❖ BerCPM.m

```
function [BER,Pe]=BerCPM(F,H,B,M,P,L,NumSymb,SymbMtx,Ro,PeT)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para un rango de SNR por BIT, en un sistema de comunicaciones
%       con precodificador, transmisión por bloques con código cíclico (CP),
%       ecualización cero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (CP)
%       H: Matriz del Canal (CP)
%       B: Matriz Lambda Óptima de autovalores.
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       L: Longitud de la respuesta impulsiva del canal
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR por BIT
%       PeT: Bandera para indicar si se calcula la BER Teórica


% Matriz de ecualización
G=pinv(H*F);
clear k;

% Recorremos una tabla para los valores SNR por BIT
for k=1:length(Ro)

        NoBlock=1/(10^(0.1*Ro(k)));  % Ruido de Bloque
        Var=NoBlock/M;               % Varianza de Ruido por BIT

        % Bloques de M Símbolos a bloques de P (CP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
```

```
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos (CP)
        Rcp=[zeros([P-L L]) eye([P-L P-L])];
        SymbRx=SymbTx+(Rcp*Noise.').';

        % Secuencia de símbolos ecualizados
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M

                % Si parte real e imaginaria incorrecta, símbolo recibido con
2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido con
1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end

        % La BER se calcula como un simple cociente
        BER(k)=bitsqam_KO/(2*NumSymb);

        % Probabilidad de error de bit mínima teórica (CP)
        if PeT==1
         Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace((inv(B'*B))^0.5))^2)));
        end


end
```

Published with MATLAB® 7.0

❖ BerCPT.m

```
function Pe=BerCPT(B,M,P,Ro)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit teórica para
%       un rango de SNR, en un sistema de comunicaciones con precodificador,
%       transmisión por bloques con código cíclico (CP), ecualización
%       cero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       B: Matriz Lambda óptima de autovalores (CP)
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       Ro: Rango para la SNR

% Recorremos una tabla para los valores SNR.
```

```
for k=1:length(Ro)
        NoBlock=1/(10^(0.1*Ro(k)));
        Var=NoBlock/P;

        % Probabilidad de error de bit mínima teórica (CP)
        Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace((inv(B'*B))^0.5))^2)));
end
```

Published with MATLAB® 7.0

❖ BerZPM.m

```
function [BER,Pe]=BerZPM(F,H,B,M,P,NumSymb,SymbMtx,Ro,PeT)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para un rango de SNR por BIT, en un sistema de comunicaciones
%       con precodificador, transmisión por bloques con relleno de ceros(ZP),
%       ecualización cero forzado y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (ZP)
%       H: Matriz del Canal (ZP)
%       B: Matriz Lambda Óptima de autovalores.
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR por BIT
%       PeT: Bandera para indicar si se calcula la BER Teórica


% Matriz de ecualización
G=pinv(H*F);
clear k;

% Recorremos una tabla para los valores SNR por BIT
for k=1:length(Ro)

        NoBlock=1/(10^(0.1*Ro(k))); %Ruido de Bloque
        Var=NoBlock/M;              %Varianza de Ruido por BIT

        % Bloques de M Símbolos a bloques de P (ZP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos (ZP)
        SymbRx=SymbTx+Noise;
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M
```

```
                % Si parte real e imaginaria incorrecta, símbolo recibido con
2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido con
1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end

        % La BER se calcula como un simple cociente
        BER(k)=bitsqam_KO/(2*NumSymb);

        % Probabilidad de error de bit mínima teórica (ZP)
        if PeT==1
         Pe(k)=1/2*erfc(sqrt(M/(2*Var*real(trace(B^0.5))^2)));
        end
end
```

Published with MATLAB® 7.0

❖ BerDMT.m

```
function ber=BerDMT(Qwf,AWF,P,L,M,Hcp,NumSymb,Var)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental
%       para una determinada SNR, en un sistema de comunicaciones con
%       precodificador Water-Filling DMT, transmisión por bloques con
%       código cíclico, ecualización cero forzado y detección por umbral de
%       símbolos QAM.
%
% ENTRADAS:
%       Qwf: Matriz de columnas correspondientes a elementos no nulos de AWF
%       AWF: Matriz de Water-Filling de asignación de energía.
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       L: Longitud de la respuesta impulsiva del canal
%       Hcp: Matriz característica del canal (CP)
%       NumSymb: Número de símbolos que se trasmiten
%       Var: Varianza del ruido correspondiente a una SNR determinada

SymbKO=0;

% Construimos el precodificador Water-Filling DMT
Awf_=Qwf.'*AWF*Qwf;
Fwf_dmt=DFTMtx(P-L)'*Qwf*Awf_;

% Matriz de ecualización
G=pinv(Hcp*Fwf_dmt);

% Matriz de Símbolos QAM generados en bloques (Filas) de M
SymbMtx=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

% Bloques de M símbolos a bloques de P
```

```
SymbTx=(Hcp*Fwf_dmt*SymbMtx.').';

% Ruido de varianza Var
Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

% Secuencia de símbolos recibidos (CP)
Rcp=[zeros([P-L L]) eye([P-L P-L])];
SymbRx=SymbTx+(Rcp*Noise.').';

% Secuencia de símbolos ecualizados
SymbSecRx=(G*SymbRx.').';

% Secuencia de símbolos estimados en bloques de M
SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M

                % Si parte real e imaginaria incorrecta, símbolo recibido con
2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido con
1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                   bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end

% La BER se calcula como un simple cociente
ber=bitsqam_KO/(2*NumSymb);
```

Published with MATLAB® 7.0

❖ BERDropZP.m

```
function ber=BERDropZP(Hzp,FOpt_,NumSymb,M_,P,Var)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental
%       para una determinada SNR dentro del rango de descarte de subcanales,
%       en un sistema de comunicaciones con precodificador, transmisión por
%       bloques con relleno de ceros (ZP), esquema de descarte de subcanales
%       con mala respuesta en frecuencia, ecualización cero forzado y
%       detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       FOpt_: Matriz del Precodificador de dimensiones acordes al descarte
%              de subcanales (ZP)
%       Hzp: Matriz del Canal (ZP)
%       NumSymb: Número de símbolos que se trasmiten
%       M_: Número de subcanales tras el descarte, tamaño de bloque inicial
%       P: Tamaño de Bloque Final a transmitir tras el relleno de ceros.
%       Var: Varianza del ruido correspondiente a una SNR determinada

SymbKO=0;
```

```
% Matriz de ecualización
G=pinv(Hzp*FOpt_);

% Matriz de Símbolos QAM generados en bloques (Filas) de M_
SymbMtx=(1/sqrt(2))*(sign(randn(NumSymb/M_,M_))+i*sign(randn(NumSymb/M_,M_)));

% Bloques de M_ símbolos a bloques de P
SymbTx=(Hzp*FOpt_*SymbMtx.').';

% Ruido de varianza Var
Noise=sqrt(Var/2)*(randn(NumSymb/M_,P)+i*(randn(NumSymb/M_,P)));

% Secuencia de símbolos recibidos (ZP)
SymbRx=SymbTx+Noise;

% Secuencia de símbolos ecualizados
SymbSecRx=(G*SymbRx.').';

% Secuencia de símbolos estimados en bloques de M_
SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
bitsqam_KO=0;
for s=1:NumSymb/M_;
    for t=1:M_

        % Si parte real e imaginaria incorrecta, símbolo recibido con 2 bits
erróneos
        if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
           bitsqam_KO=bitsqam_KO + 2;

        % Si parte real o imaginaria incorrecta, símbolo recibido con 1 bit
erróneo
        elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
           bitsqam_KO=bitsqam_KO + 1;
        end
    end
end

% La BER se calcula como un simple cociente
ber=bitsqam_KO/(2*NumSymb);
```

Published with MATLAB® 7.0

❖ BERDropCP.m

```
function ber=BERDropCP(Hcp,FOpt_,NumSymb,M_,P,L,Var)
% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental
%       para una determinada SNR dentro del rango de descarte de subcanales,
%       en un sistema de comunicaciones con precodificador, transmisión por
%       bloques con código cíclico (CP), esquema de descarte de subcanales
%       con mala respuesta en frecuencia, ecualización cero forzado y
%       detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       Hcp: Matriz del Canal (CP)
%       FOpt_: Matriz del Precodificador de dimensiones acordes al descarte
%              de subcanales (CP)
```

```
%       NumSymb: Número de símbolos que se trasmiten
%       M_: Número de subcanales tras el descarte, tamaño de bloque inicial
%       P: Tamaño de Bloque Final a transmitir tras el relleno de ceros
%       L: Longitud de la respuesta impulsiva del canal
%       Var: Varianza del ruido correspondiente a una SNR determinada


% Matriz de ecualización
G=pinv(Hcp*FOpt_);

% Matriz de Símbolos QAM generados en bloques (Filas) de M_
SymbMtx=(1/sqrt(2))*(sign(randn(NumSymb/M_,M_))+i*sign(randn(NumSymb/M_,M_)));

% Bloques de M_ símbolos a bloques de P
SymbTx=(Hcp*FOpt_*SymbMtx.').';

% Ruido de varianza Var
Noise=sqrt(Var/2)*(randn(NumSymb/M_,P)+i*(randn(NumSymb/M_,P)));

% Secuencia de símbolos recibidos (CP)
Rcp=[zeros([P-L L]) eye([P-L P-L])];
SymbRx=SymbTx+(Rcp*Noise.').';

% Secuencia de símbolos ecualizados
SymbSecRx=(G*SymbRx.').';

% Secuencia de símbolos estimados en bloques de M_
SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
bitsqam_KO=0;
for s=1:NumSymb/M_;
    for t=1:M_

        % Si parte real e imaginaria incorrecta, símbolo recibido con bits
erróneos
        if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
           bitsqam_KO=bitsqam_KO + 2;

        % Si parte real o imaginaria incorrecta, símbolo recibido con 1 bit
erróneo
        elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
           bitsqam_KO=bitsqam_KO + 1;
        end
    end
end

% La BER se calcula como un simple cociente
ber=bitsqam_KO/(2*NumSymb);
```

Published with MATLAB® 7.0

❖ BERNoDropZP.m

```
function ber=BERNoDropZP(F,H,M,P,NumSymb,SymbMtx,Ro)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental
%       para un rango de SNR superior a la umbral de descarte de subcanales,
%       en un sistema de comunicaciones con precodificador, transmisión por
```

```
%       bloques con relleno de ceros (ZP), esquema de descarte de subcanales
%       con mala respuesta en frecuencia, ecualización cero forzado y
%       detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (ZP)
%       H: Matriz del Canal (ZP)
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR superior a la SNR umbral de descarte de
%           subcanales

% Matriz de ecualización
G=pinv(H*F);
clear k;

% Recorremos una tabla para los valores de SNR del rango > SNR Umbral
for k=1:length(Ro)

        No=1/(10^(0.1*Ro(k)));  %Ruido de Bloque
        Var=No/P;               %Varianza de Ruido

        % Bloques de M Símbolos a bloques de P (ZP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos (ZP)
        SymbRx=SymbTx+Noise;

        % Secuencia de símbolos ecualizados
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M

                % Si parte real e imaginaria incorrecta, símbolo recibido
                % con 2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 2;

                % Si parte real o imaginaria incorrecta, símbolo recibido
                % con 1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end

        % La BER se calcula como un simple cociente
        ber(k)=bitsqam_KO/(2*NumSymb);

end
```

Published with MATLAB® 7.0

❖ BERNoDropCP.m

```
function ber=BERNoDropCP(F,H,M,P,L,NumSymb,SymbMtx,Ro)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental
%       para un rango de SNR superior a la umbral de descarte de subcanales,
%       en un sistema de comunicaciones con precodificador, transmisión por
%       bloques con código cíclico (CP), esquema de descarte de subcanales
%       con mala respuesta en frecuencia, ecualización cero forzado y
%       detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       F: Matriz del Precodificador (CP)
%       H: Matriz del Canal (CP)
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       L: Longitud de la respuesta impulsiva del canal
%       NumSymb: Número de símbolos que se trasmiten
%       SymbMtx: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       Ro: Rango para la SNR superior a la SNR umbral de descarte de
%           subcanales

% Matriz de ecualización
G=pinv(H*F);
clear k;

for k=1:length(Ro)

        No=1/(10^(0.1*Ro(k)));  %Ruido de Bloque
        Var=No/P;               %Varianza de Ruido

        % Bloques de M Símbolos a bloques de P (CP)
        SymbTx=(H*F*SymbMtx.').';

        % Ruido de varianza Var
        Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

        % Secuencia de símbolos recibidos (CP)
        Rcp=[zeros([P-L L]) eye([P-L P-L])];
        SymbRx=SymbTx+(Rcp*Noise.').';

        % Secuencia de símbolos ecualizados
        SymbSecRx=(G*SymbRx.').';

        % Secuencia de símbolos estimados en bloques de M
        SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
        bitsqam_KO=0;
        for s=1:NumSymb/M;
            for t=1:M

                % Si parte real e imaginaria incorrecta, símbolo recibido
                % con 2 bits erróneos
                if real(SymbEst(s,t))*real(SymbMtx(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 2;
```

```
                % Si parte real o imaginaria incorrecta, símbolo recibido
                % con 1 bit erróneo
                elseif real(SymbEst(s,t))*real(SymbMtx(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbMtx(s,t))<0
                    bitsqam_KO=bitsqam_KO + 1;
                end
            end
        end

        % La BER se calcula como un simple cociente
        ber(k)=bitsqam_KO/(2*NumSymb);
 end
```

Published with MATLAB® 7.0

❖ MYFUN.m

```
function [Funcion,AWF]= myfun(x,Po,Var,AH)

% DESCRIPCIÓN:
%       Esta es la función objetivo que se emplea en el proceso de
%       recursivo para la construcción de la matriz Water-Filling DMT
%       de asignación de energía
%
% ENTRADAS:
%       x:  Constante que se debe ajustar para cumplir la restricción de
%           potencia transmitida
%       Po: Restricción de potencia transmitida
%       Var:Varianza del ruido correspondiente a una SNR determinada
%       AH: Matriz diagonal con la FFT de P-L puntos de la respuesta
%           impulsiva del canal


% Construcción de la matriz Water-Filling DMT de asignación de energía
Mtx=AH'*AH;
Awf=zeros([length(Mtx),length(Mtx)]);
for k=1:length(AH)
    if x> Var/Mtx(k,k)
        Awf(k,k)=sqrt(x-(Var/Mtx(k,k)));
    else
        Awf(k,k)=0;
    end
end

% Función a igualar a cero en el proceso iterarivo.
Funcion = trace(Awf*Awf')-Po;
AWF=Awf;
```

Published with MATLAB® 7.0

❖ SNRBit.m

```
function [BERM_,SNRM_,MDrop_]=SNRBit(MDrop,SNR,Ber);


% DESCRIPCIÓN:
%       Esta función obtiene la curva discontínua y multievaluada de la
```

```
%       probabilidad de error de bit cuando se emplea el esquema de descarte
%       de subcanales con baja ganancia para un rango de SNR por debajo del
%       umbral de donde se empiezan a suprimir dichos subcanales. Se
%       seleccionan los valores de SNR y BER cuando acontece un cambio en
%       el número de subcanales a emplear
%
%
% ENTRADAS:
%       MDrop: Vector con el número de subcanales empleados correspondiente
%              al Vector de SNR por debajo de la umbral de descarte
%       SNR:   Rango de valores de SNR por debajo de la SNR umbral
%       Ber:   Probabilidad de error de bit correspondiente al vector de SNR

    u=1;
    v=0;
    while u < length(MDrop)
        % Asignación de valores iniciales
        v=v+1;
        MDrop_(v) = MDrop(u);
        SNRM_(v)=SNR(u);
        BERM_(v)=Ber(u);

        % Mientras no haya cambio en el número de subcanales
        while MDrop(u+1)==MDrop_(v) & u < length(MDrop)-1
            u=u+1;
        end
        v=v+1;

        % Asignación de valores al haber un cambio en el número de subcanales
        MDrop_(v) = MDrop(u);
        SNRM_(v)=SNR(u);
        BERM_(v)=Ber(u);
        u=u+1;
    end
```

Published with MATLAB® 7.0

❖ Ejemplo3_ZP.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros, ecualización Cero-Forzado%
%  y detección por umbral sobre 500 realizaciones de un canal aleatorio   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



clear all               % Eliminamos todas las variables existentes
close all               % Cerramos todas las ventanas

SNR=0:1:25;             % Rango para la Relación Señal-Ruido
M=16;                   % Tamaño de Bloque inicial de símbolos
Po=1;                   % Restricción de potencia de bloque a transmitir
L=4;                    % Longitud Respuesta impulsiva del canal
P=20;                   % P>=M+L Tamaño final de bloque a transmitir
NumSymb=M*1e2;          % Numeros de simbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
```

```
for w=1:500 % 500 realizaciones
    cont=w
    % Generamos los coeficentes de la respuesta impulsiva
    taps=randn(1,L+1)+i*randn(1,L+1);
    h=taps/sqrt(sum(abs(taps).^2));

    % Construimos la matriz toeplitz ZP con la respuesta del canal
    Hzp=zeros([P P-L]);
    c=[h zeros([1 P-L-1])];
    r=[h(1) zeros([1 P-L-1])];
    Hzp=toeplitz(c,r);
    [X,B]=eigs(inv(Hzp'*Hzp),M,'sm');      % B es Lambda optima M*M
    [Y,C]=eigs(inv(Hzp'*Hzp),P-L-M,'lm');
    PhiOpt=sqrt(Po/trace(B^0.5))*(B^0.25);
    UOpt=[X Y];              % X es Wm
    Dm=DFTMtx(M);            % Construimos la matriz DFT normalizada
    FOpt=X*PhiOpt*Dm;        % Construimos F óptima
    FOptZP=FOpt;

    % Matriz con vectores fila de BER´s teóricas para cada canal
    Pe(w,:)=BerZPT(B,M,P,SNR);

    % Construimos F ZP-OFDM
    Fzp_ofdm=sqrt(Po/M)*Dm'*eye([M M]);

    % Matriz con vectores fila de BER's experimentales para cada canal
    BER1(w,:)=BerZP(Fzp_ofdm,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);

    % Construimos F MMSE-ZP-I
    Fzp_mmse_zf_i=X*PhiOpt;
    % Matriz con vectores fila de BER's experimentales para cada canal
    BER2(w,:)=BerZP(Fzp_mmse_zf_i,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);

    % Construimos F ZP-MSNR
    Fzp_msnr=sqrt(Po/trace(B))*X*(B^0.5);
    % Matriz con vectores fila de BER's experimentales para cada canal
    BER3(w,:)=BerZP(Fzp_msnr,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);

    clear h Fzp_ofdm Fzp_mmse_zf_i Fzp_msnr Hzp X Y B PhiOpt UOpt
    clear Dm FOpt FOptZP c r taps
end

% Evaluamos la BER media, teórica y de cada precodificador, para todos los
% canales aleatorios generados.
Pe=1/w*sum(Pe(:,:));
BER1=1/w*sum(BER1(:,:));
BER2=1/w*sum(BER2(:,:));
BER3=1/w*sum(BER3(:,:));

% Dibujamos la BER para los precodificadores ZP y la BER Teórica
semilogy(SNR,Pe,'-+',SNR,BER1,'-*',SNR,BER2,'->',SNR,BER3,'-<');
axis([0 SNR(length(SNR)) 1e-4 1])
grid
Xlabel('SNR(dB)');
YLabel('BER(ZP)');
legend('ZP-MBER','ZP-OFDM','ZP-MMSE-ZF-I','ZP-MSNR',0);
```

Published with MATLAB® 7.0

❖ Ejemplo1_CP.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con código cíclico, ecualización Cero-Forzado  %
%  y detección por umbral, sobre un canal con buena respuesta frecuencial %
%  y opcionalmente esquema de descarte de subcanales de mala respuesta    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

%Respuesta impulsiva del canal
h=[0.3038+0.2554i 0.5056+0.5587i 0.2855+ 0.0035i 0.2834+0.1843i
0.2793+0.0305i];
L=length(h)-1;          % Longitud Respuesta impulsiva del canal
SNR=0:1:18;             % Rango para la Relación Señal-Ruido
M=32;                   % Tamaño de Bloque inicial de símbolos
Po=1;                   % Restricción de potencia de bloque a transmitir
NumSymb=2*M*1e4;        % Numeros de simbolos a transmitir
P=36;                   % P>=M+L Tamaño final de bloque a transmitir

figure(1) % Respuesta Impulsiva del Canal
x=0:M-1;
stem(x,abs(fft(h,M)))
axis([-1 32 0 2])
Xlabel('Índice del subcanal, i');
YLabel('|H(exp(j2pi*i/(P-L))|');

% Construimos la matriz CP con la respuesta del canal
f=fft(h,P-L);
AH=diag(f.');
[X,C]=eigs(AH,P-L,'lm');
[Z,index]=sort(diag(C));
B=diag(Z);
AH_=[zeros([M P-L-M]),eye([M M])]*B*[zeros([P-L-M M]);eye([M M])];
I = eye([length(index),length(index)]);
D= I(index,:);
Pm=(D*inv(X)).'* [zeros([P-L-M M]);eye([M M])];
Hcp=DFTMtx(P-L)'*AH*DFTMtx(P-L);

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

% Construimos F MBER-CP óptima
FOpt=sqrt(Po/(trace((AH_'*AH_)^(-0.5))))*DFTMtx(P-L)'*Pm*((AH_'*AH_)^(-
0.25))*DFTMtx(M);
FOptCP=FOpt;


figure(2)               %BER Precodificadores CP

% Dibujamos la BER para MBER-CP y la BER Teorica
[BER,Pe]=BerCP(FOpt,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,1);
 semilogy(SNR,BER,'-kd',SNR,Pe);
hold on
clear BER Pe
```

```
% Construimos F CP-OFDM
Fcp_ofdm=sqrt(Po/M)*DFTMtx(P-L)'*eye([M M]);

% Dibujamos la BER para CP-OFDM
BER=BerCP(Fcp_ofdm,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-r+');
clear BER

% Construimos F MMSE-CP-I
Fcp_mmse_zf_i=DFTMtx(P-L)'*Pm*sqrt(Po/(trace((AH_'*AH_)^(-
0.5))))*((AH_'*AH_)^(-0.25));

% Dibujamos la BER para MMSE-ZF-I
BER=BerCP(Fcp_mmse_zf_i,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-c*');
clear BER
legend('CP-MBER','P_e Teorica','CP-OFDM','CP-MMSE-ZF-I',0);
axis([0 SNR(length(SNR)) 1e-5 1])
Xlabel('SNR(dB)');
YLabel('BER(CP)');
grid
hold off

% Construimos el Precodificador DMT Waterfilling
u=1;
Paso1=0.04;

% Recorremos una tabla para los valores SNR.
for k=1:length(SNR)
        No=1/(10^(0.1*SNR(k)));
        x0=100;
        Var=No/P;
        options = optimset('Display','off'); % Desactiva el Display
        x = fsolve(@myfun,x0,options,Po,Var,AH);
        [Result,AWF]=myfun(x,Po,Var,AH);
        indices=find(AWF);
        Qwf=zeros(size(AWF));
        Qwf(indices)=1;
        if  length(u)~=0
            Aux=diag(diag(Qwf)-1);
            [u,v]=find(Aux);
            kdrop=k;
            SNRd=SNR(kdrop); % SNR umbral de descarte de subcanales
        else

            % Probabilidad de error para el rango de SNR sin descarte
            BERNoDrop(k-kdrop)=BerDMT(Qwf,AWF,P,L,M,Hcp,NumSymb,Var);
            if k==length(SNR)
               Fwf_dmt=DFTMtx(P-L)'*Qwf*Qwf.'*AWF*Qwf; % F DMT Waterfilling
            end
        end
end
SNRDropDMT=-6:Paso1:SNRd; % Rango de SNR en el que hay descarte de subcanales
for k=1:length(SNRDropDMT)
        No=1/(10^(0.1*SNRDropDMT(k)));
        x0=100;
        Var=No/P;
        options = optimset('Display','off');
        x = fsolve(@myfun,x0,options,Po,Var,AH);
        [Result,AWF]=myfun(x,Po,Var,AH);
```

```
        indices=find(AWF);
        Qwf=zeros(size(AWF));
        Qwf(indices)=1;
        Aux=diag(diag(Qwf)-1);
        [u,v]=find(Aux);
        Qwf(:,u) = [];
        M_=M-length(u);
        MDMTDrop(k)=M_; % Número de subcanales que se emplean para cada SNR
        NumSymb=2*M_*1e4;
        % Probabilidad de error para el rango de SNR con descarte
        BERDrop(k)=BerDMT(Qwf,AWF,P,L,M_,Hcp,NumSymb,Var);
 end
BerDMTotal=[BERDrop, BERNoDrop];
SNRNoDropDMT=SNR(kdrop+1):1:SNR(length(SNR));
SNRDMTotal=[SNRDropDMT,SNRNoDropDMT];
BERNoDropDMT=BERNoDrop;
MDMT=[MDMTDrop, M*ones([1 length(SNR(kdrop+1):1:SNR(length(SNR)))])];

% Dibujamos los elementos diagonales de GG' para todos los precodificadores
figure(3)
subplot(4,1,1); PlotDiag(Hcp,FOpt,'d','CP-MBER')
subplot(4,1,2); PlotDiag(Hcp,Fcp_ofdm,'o','CP-OFDM')
subplot(4,1,3); PlotDiag(Hcp,Fcp_mmse_zf_i,'<','CP-MMSE-ZF-I')
subplot(4,1,4); PlotDiag(Hcp,Fwf_dmt,'*','Waterfilling DMT')
Xlabel('Índice de los elementos diagonales de GG^H')

% BER Precodificadores ZP CP y DMT Waterfilling con subcanales descartados
figure(4)
semilogy (SNRDMTotal,BerDMTotal);
hold on

% Dibujamos la BER para MBER-CP
NumSymb=2*M*1e4;
BER=BerCP(FOpt,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'-kx');
clear BER BERDrop BERNoDrop u No k SymbSec

% Cálculo del umbral por debajo del cual hay descarte de subcanales en el
% esquema MBER-CP
SNRc=10*log10(3/(M*P)*abs(trace((AH_'*AH_)^-0.5))^2);
Paso2=0.08;
SNRDrop=-8:Paso2:SNRc;

% Cálculo de la BER para SNR por debajo de la umbral de descarte
for k=1:length(SNRDrop)
        No=1/(10^(0.1*SNRDrop(k)));
        ro=Po/No;
        Var=No/P;
        M_=M;
        SNRc_=SNRc;

        while ro < 10^(SNRc_/10)
            M_=M_-1;
            AH__=[zeros([M_ P-L-M_]),eye([M_ M_])]*B*[zeros([P-L-M_
M_]);eye([M_ M_])];
            SNRc_=10*log10(3/(M_*P)*abs(trace((AH__'*AH__)^-0.5))^2);
        end

        MCPDrop(k)=M_; % Número de subcanales empleados para cada SNR
        Pm_=(D*inv(X)).'* [zeros([P-L-M_ M_]);eye([M_ M_])];
```

```
        FOpt_=sqrt(Po/(trace((AH__'*AH__)^(-0.5))))*DFTMtx(P-
L)'*Pm_*((AH__'*AH__)^(-0.25))*DFTMtx(M_);
        NumSymb=2*M_*1e4;
        BERDrop(k)=BERDropCP(Hcp,FOpt_,NumSymb,M_,P,L,Var);
end

SNRNoDrop=[SNRc:1:SNR(length(SNR)),SNR(length(SNR))];
NumSymb=2*M*1e4;
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

% Cálculo de la BER para SNR por encima de la umbral de descarte
BERNoDrop=BERNoDropCP(FOpt,Hcp,M,P,L,NumSymb,SymbSec,SNRNoDrop);
BERCPNoDrop=BERNoDrop;
SNRNoDropCP=SNRNoDrop;
SNRDropCP=SNRDrop;
BerCPDropTotal=[BERDrop, BERNoDrop];
SNRCPDropTotal=[SNRDrop, SNRNoDrop];
MCP=[MCPDrop,M*ones([1 length(SNRNoDrop)])];

% Dibujamos la BER para MBER-CP con esquema de descarte de subcanales
semilogy (SNRCPDropTotal,BerCPDropTotal,'-g');
clear X Y B C FOpt FOpt_ BERDrop BERNoDrop SNRDrop SNRNoDrop SNRc SNRc_ Pm_
AH__

% Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm');%B es la matriz lambda óptima de autovalores
[Y,C]=eigs(inv(Hzp'*Hzp),P-L-M,'lm');
PhiOpt=sqrt(Po/trace(B^0.5))*(B^0.25);
UOpt=[X Y];             % X es Wm
Dm=DFTMtx(M);           % Construimos la matriz DFT normalizada
FOpt=X*PhiOpt*Dm;       % Construimos F óptima MBER-ZP
FOptZP=FOpt;
BER=BerZP(FOpt,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);

% Dibujamos la BER para MBER-ZP
semilogy(SNR,BER,'-md')
clear BER X Y C

% Cálculo del umbral por debajo del cual hay descarte de subcanales en el
% esquema MBER-ZP
Paso3=0.08;
SNRc=10*log10(3/(M*P)*abs(trace(B^0.5))^2);
SNRDrop=-8:Paso3:SNRc;

% Cálculo de la BER para SNR por debajo de la umbral de descarte
% Recorremos una tabla para los valores de SNR<SNRc
for k=1:length(SNRDrop)
        No=1/(10^(0.1*SNRDrop(k)));
        ro=Po/No;
        Var=No/P;
        M_=M;
        SNRc_=SNRc;
        while ro < 10^(SNRc_/10)
            M_=M_-1;
            if M_==31
              [X_,B_]=eigs(inv(Hzp'*Hzp),M_,'sr');%B_ es Lambda_ optima M_*M_
              s=M_:-1:1;
```

```
                X_=X_(:,s);
                B_=diag(sort(diag(B_),'descend'));
            else
            [X_,B_]=eigs(inv(Hzp'*Hzp),M_,'sm');
            end
            SNRc_=10*log10(3/(M_*P)*abs(trace(B_^0.5))^2);
        end
        MZPDrop(k)=M_;    % Número de subcanales empleados para cada SNR
        [Y_,C_]=eigs(inv(Hzp'*Hzp),P-L-M_,'lm');
        PhiOpt_=sqrt(Po/trace(B_^0.5))*(B_^0.25);
        UOpt_=[X_ Y_];    % X_ es Wm_
        Dm_=DFTMtx(M_);
        FOpt_=X_*PhiOpt_*Dm_;
        NumSymb=2*M_*1e4;
        BERDrop(k)=BERDropZP(Hzp,FOpt_,NumSymb,M_,P,Var);
end

% Cálculo de la BER para SNR por encima de la umbral de descarte
SNRNoDrop=[SNRc:1:SNR(length(SNR)),SNR(length(SNR))];
NumSymb=2*M*1e4;
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
BERNoDrop=BERNoDropZP(FOpt,Hzp,M,P,NumSymb,SymbSec,SNRNoDrop);
BerZPDropTotal=[BERDrop, BERNoDrop];
SNRZPDropTotal=[SNRDrop, SNRNoDrop];
MZP=[MZPDrop,M*ones([1 length(SNRNoDrop)])];

% Dibujamos la BER para MBER-CP con esquema de descarte de subcanales
semilogy (SNRZPDropTotal,BerZPDropTotal,'-r');
legend('DMT-Waterfilling','CP-MBER','CP-MBER-DROP','ZP-MBER','ZP-MBER-
DROP',0);
axis([0 18 1e-5 1])
Xlabel('SNR(dB)');
YLabel('BER');
grid
hold off
clear X_ Y_ UOpt_ FOpt_ SNRc_ B_ X_

% Tamaños de bloque para el sistema con descarte de subcanales para los
% distintos precodificadores en el rango de SNR de estudio
figure(5)
plot(SNRDMTotal,MDMT,SNRCPDropTotal,MCP,SNRZPDropTotal,MZP )
Xlabel('SNR(dB)');
YLabel('Tamaño De Bloque');
legend('DMT-Waterfilling','CP-MBER-DROP','ZP-MBER-DROP',0);
axis([0 18 13 32])
grid

figure(6) %Dibujamos la BER frente a la SNR por Bit

% Dibujamos la BER para el esquema MBER-CP frente a la SNR por bit
BER=BerCPM(FOptCP,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'b')
hold on
clear BER

% Dibujamos la BER para el esquema MBER-ZP frente a la SNR por bit
BER=BerZPM(FOptZP,Hzp,B,M,P,NumSymb,SymbSec,SNR,0);
semilogy(SNR,BER,'c')
clear BER

% Dibujamos la BER para el esquema MBER-ZP con descarte de subcanales
```

```
% frente a la SNR por bit
[BERM_,SNRM_,MDrop_]=SNRBit(MZPDrop,SNRDrop,BerZPDropTotal);
semilogy
([SNRM_+10*log10(P./MDrop_),SNRNoDrop+10*log10(P/M)],[BERM_,BERNoDrop],'m');
clear BERM_ SNRM_ MDrop_

% Dibujamos la BER para el esquema MBER-CP con descarte de subcanales
% frente a la SNR por bit
[BERM_,SNRM_,MDrop_]=SNRBit(MCPDrop,SNRDropCP,BerCPDropTotal);
semilogy
([SNRM_+10*log10(P./MDrop_),SNRNoDropCP+10*log10(P/M)],[BERM_,BERCPNoDrop],'g
');
axis([0 18 1e-5 1])
clear BERM_ SNRM_ MDrop_

% Dibujamos la BER para el esquema DMT-WF con descarte de subcanales
% frente a la SNR por bit
[BERM_,SNRM_,MDrop_]=SNRBit(MDMTDrop,SNRDropDMT,BerDMTotal);
semilogy
([SNRM_+10*log10(P./MDrop_),SNRNoDropDMT+10*log10(P/M)],[BERM_,BERNoDropDMT],
'r');
clear BERM_ SNRM_ MDrop_

% Leyenda
legend('CP-MBER','ZP-MBER','ZP-MBER-DROP','CP-MBER-DROP','DMT-
Waterfilling',0);
Xlabel('SNR por Bit(dB)');
YLabel('BER(CP Y ZP)');
axis([0 18 1e-5 1])
grid
```

Published with MATLAB® 7.0

❖ Ejemplo3_CP.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con código cíclico, ecualización Cero-Forzado  %
%  y detección por umbral sobre 500 realizaciones de un canal aleatorio   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

SNR=0:1:25;        % Rango para la Relación Señal-Ruido
M=16;              % Tamaño de Bloque inicial de símbolos
Po=1;              % Restricción de potencia de bloque a transmitir
L=4;               % Longitud Respuesta impulsiva del canal
P=20;              % P>=M+L Tamaño final de bloque a transmitir
NumSymb=M*1e2;     %Numeros de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

for w=1:500  % 500 realizaciones
    cont=w

    % Generamos los coeficentes aleatorios de la respuesta impulsiva
    taps=randn(1,L+1)+i*randn(1,L+1);
```

```
    h=taps/sqrt(sum(abs(taps).^2)); % Normalizamos a 1

    % Construimos la matriz CP con la respuesta del canal
    f=fft(h,P-L);
    AH=diag(f.');
    [X,C]=eigs(AH,P-L,'lm');
    [Z,index]=sort(diag(C));
    B=diag(Z);
    AH_=[zeros([M P-L-M]),eye([M M])]*B*[zeros([P-L-M M]);eye([M M])];
    I = eye([length(index),length(index)]);
    D= I(index,:);
    Pm=(D*inv(X)).'* [zeros([P-L-M M]);eye([M M])];
    Hcp=DFTMtx(P-L)'*AH*DFTMtx(P-L);

    % Matriz con vectores fila de BER´s teóricas para cada canal
    Pe(w,:)=BerCPT(B,M,P,SNR);

    %Construimos F CP-OFDM
    Fcp_ofdm=sqrt(Po/M)*DFTMtx(P-L)'*eye([M M]);

    % Matriz con vectores fila de BER's experimentales para cada canal
    BER1(w,:)=BerCP(Fcp_ofdm,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);

    %Construimos F MMSE-CP-I
    Fcp_mmse_zf_i=DFTMtx(P-L)'*Pm*sqrt(Po/(trace((AH_'*AH_)^(-
0.5))))*((AH_'*AH_)^(-0.25));

    % Matriz con vectores fila de BER's experimentales para cada canal
    BER2(w,:)=BerCP(Fcp_mmse_zf_i,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,0);


    clear h Fcp_ofdm Fcp_mmse_zf_i Hcp X Y B Z index AH f I D Pm h taps
end

% Evaluamos la BER media, teórica y de cada precodificador, para todos los
% canales aleatorios generados.
Pe=1/w*sum(Pe(:,:));
BER1=1/w*sum(BER1(:,:));
BER2=1/w*sum(BER2(:,:));

%Dibujamos la BER para los precodificadores CP y la BER Teórica
semilogy(SNR,Pe,'-+',SNR,BER1,'-*',SNR,BER2,'->');
axis([0 SNR(length(SNR)) 1e-4 1])
grid
Xlabel('SNR(dB)');
YLabel('BER(CP)');
legend('CP-MBER','CP-OFDM','CP-MMSE-ZF-I',0);
```

Published with MATLAB® 7.0

❖ Ejemplo3_Drop.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques ZP, CP y DMT-WF, ecualización Cero-Forzado     %
%  y detección por umbral sobre 500 realizaciones de un canal aleatorio   %
%  con esquema de descarte de subcanales                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear all       % Eliminamos todas las variables existentes
close all       % Cerramos todas las ventanas

SNR=0:1:25;     % Rango para la Relación Señal-Ruido
M=16;           % Tamaño de Bloque inicial de símbolos
Po=1;           % Restricción de potencia de bloque a transmitir
L=4;            % Longitud Respuesta impulsiva del canal
P=20;           % P>=M+L Tamaño final de bloque a transmitir
NumSymb=M*1e2;  %Numeros de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
cp=0;

for w=1:500     % 500 realizaciones
    cont=w

    % Generamos los coeficentes aleatorios de la respuesta impulsiva
    taps=randn(1,L+1)+i*randn(1,L+1);
    h=taps/sqrt(sum(abs(taps).^2));   % Normalizamos a 1

    % Construimos la matriz CP con la respuesta del canal
    f=fft(h,P-L);
    AH=diag(f.');
    [X,C]=eigs(AH,P-L,'lm');
    [Z,index]=sort(diag(C));
    B=diag(Z);
    AH_=[zeros([M P-L-M]),eye([M M])]*B*[zeros([P-L-M M]);eye([M M])];
    I = eye([length(index),length(index)]);
    D= I(index,:);
    Pm=(D*inv(X)).'* [zeros([P-L-M M]);eye([M M])];
    Hcp=DFTMtx(P-L)'*AH*DFTMtx(P-L);

    % Construimos F MBER-CP óptima
    FOpt=sqrt(Po/(trace((AH_'*AH_)^(-0.5))))*DFTMtx(P-L)'*Pm*((AH_'*AH_)^(-
0.25))*DFTMtx(M);
    FOptCP=FOpt;

    % Construimos el Precodificador DMT Waterfilling
    u=1;
    Paso1=1;

    % Recorremos una tabla para los valores SNR.
    for k=1:length(SNR)
            No=1/(10^(0.1*SNR(k)));
            x0=100;
            Var=No/P;
            options = optimset('Display','off'); % Desactiva el Display
            x = fsolve(@myfun,x0,options,Po,Var,AH);
            [Result,AWF]=myfun(x,Po,Var,AH);
            indices=find(AWF);
            Qwf=zeros(size(AWF));
            Qwf(indices)=1;
            if  length(u)~=0
                Aux=diag(diag(Qwf)-1);
                [u,v]=find(Aux);
                kdrop=k;
                SNRd=SNR(kdrop); % SNR umbral de descarte de subcanales
            else
```

```
                % Probabilidad de error para el rango de SNR sin descarte
                BERNoDrop(k-kdrop)=BerDMT(Qwf,AWF,P,L,M,Hcp,NumSymb,Var);

                if k==length(SNR)

                   % F DMT Waterfilling
                   Fwf_dmt=DFTMtx(P-L)'*Qwf*Qwf.'*AWF*Qwf;

                end
            end
    end

    % Rango de SNR en el que hay descarte de subcanales
    SNRDropDMT=0:Paso1:SNRd;
    for k=1:length(SNRDropDMT)
            No=1/(10^(0.1*SNRDropDMT(k)));
            x0=100;
            Var=No/P;
            options = optimset('Display','off');
            x = fsolve(@myfun,x0,options,Po,Var,AH);
            [Result,AWF]=myfun(x,Po,Var,AH);
            indices=find(AWF);
            Qwf=zeros(size(AWF));
            Qwf(indices)=1;
            Aux=diag(diag(Qwf)-1);
            [u,v]=find(Aux);
            Qwf(:,u) = [];
            M_=M-length(u);
            MDMTDrop(k)=M_;% Número de subcanales que se emplean para cada
SNR
            NumSymb=M_*1e2;

            % Probabilidad de error para el rango de SNR con descarte
            BERDrop(k)=BerDMT(Qwf,AWF,P,L,M_,Hcp,NumSymb,Var);
    end
    if length(BERDrop)==length(SNR)
        BERNoDrop=[];
    end

    % Matriz con vectores fila de BER's experimentales para cada canal
    BerDMTotal(w,:)=[BERDrop, BERNoDrop];
    SNRNoDropDMT=SNRd+1:1:SNR(length(SNR));
    SNRDMTotal(w,:)=[SNRDropDMT,SNRNoDropDMT];
    BERNoDropDMT=BERNoDrop;

    clear  BERDrop BERNoDrop u No k %SymbSec

    % Cálculo del umbral por debajo del cual hay descarte de subcanales en el
    % esquema MBER-CP
    SNRc=10*log10(3/(M*P)*abs(trace((AH_'*AH_)^-0.5))^2);
    Paso2=1;
    % Cálculo de la BER para SNR por debajo de la umbral de descarte
    if SNRc<SNR(length(SNR))
        SNRDrop=0:Paso2:SNRc;
        cp=cp+1;

        % Recorremos una tabla para los valores de SNR
        for k=1:length(SNRDrop)
                No=1/(10^(0.1*SNRDrop(k)));
                ro=Po/No;
```

```
                Var=No/P;
                M_=M;
                SNRc_=SNRc;

                while ro < 10^(SNRc_/10)
                    M_=M_-1;
                    AH__=[zeros([M_ P-L-M_]),eye([M_ M_])]*B*[zeros([P-L-M_
M_]);eye([M_ M_])];
                    SNRc_=10*log10(3/(M_*P)*abs(trace((AH__'*AH__)^-0.5))^2);
                end

                MCPDrop(k)=M_;% Número de subcanales empleados para cada SNR
                Pm_=(D*inv(X)).'* [zeros([P-L-M_ M_]);eye([M_ M_])];
                FOpt_=sqrt(Po/(trace((AH__'*AH__)^(-0.5))))*DFTMtx(P-
L)'*Pm_*((AH__'*AH__)^(-0.25))*DFTMtx(M_);
                NumSymb=M_*1e2;
                % Probabilidad de error para el rango de SNR con descarte
                BERDrop(k)=BERDropCP(Hcp,FOpt_,NumSymb,M_,P,L,Var);
        end

        SNRNoDrop=[SNRc:1:SNR(length(SNR)),SNR(length(SNR))];
        NumSymb=M*1e2;

SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
        BERNoDrop=BERNoDropCP(FOpt,Hcp,M,P,L,NumSymb,SymbSec,SNRNoDrop);
        BERCPNoDrop=BERNoDrop;
        SNRNoDropCP=SNRNoDrop;
        SNRDropCP=SNRDrop;

        % Matriz con vectores fila de BER's experimentales para cada canal
        BerCPDropTotal(w,:)=[BERDrop, BERNoDrop];
        SNRCPDropTotal(w,:)=[SNRDrop, SNRNoDrop];

    end

    clear X Y B C FOpt FOpt_ BERDrop BERNoDrop SNRDrop SNRNoDrop SNRc SNRc_
Pm_ AH__

    % Cálculo del umbral por debajo del cual hay descarte de subcanales en el
    % esquema MBER-ZP

    %Construimos la matriz toeplitz ZP con la respuesta del canal
    Hzp=zeros([P P-L]);
    c=[h zeros([1 P-L-1])];
    r=[h(1) zeros([1 P-L-1])];
    Hzp=toeplitz(c,r);
    [X,B]=eigs(inv(Hzp'*Hzp),M,'sm'); %B es la matriz Lambda óptima M*M
    [Y,C]=eigs(inv(Hzp'*Hzp),P-L-M,'lm');
    PhiOpt=sqrt(Po/trace(B^0.5))*(B^0.25);
    UOpt=[X Y];                 %X es Wm
    Dm=DFTMtx(M);               %Construimos la matriz DFT normalizada
    FOpt=X*PhiOpt*Dm;           %Construimos F optima
    FOptZP=FOpt;

    clear X Y C

    Paso3=1;
    SNRc=10*log10(3/(M*P)*abs(trace(B^0.5))^2); %Umbral de descarte
    SNRDrop=0:Paso3:SNRc;           %Rango para la SNR de descarte

    % Recorremos una tabla para los valores de SNR<SNRc
```

```
    for k=1:length(SNRDrop)
            No=1/(10^(0.1*SNRDrop(k)));
            ro=Po/No;
            Var=No/P;
            M_=M;
            SNRc_=SNRc;
            while ro < 10^(SNRc_/10)
                M_=M_-1;
                if M_==31
                  [X_,B_]=eigs(inv(Hzp'*Hzp),M_,'sr'); %B_ es Lambda_ optima
M_*M_
                  s=M_:-1:1;
                  X_=X_(:,s);
                  B_=diag(sort(diag(B_),'descend'));
                else
                [X_,B_]=eigs(inv(Hzp'*Hzp),M_,'sm');
                end
                SNRc_=10*log10(3/(M_*P)*abs(trace(B_^0.5))^2);
            end
            MZPDrop(k)=M_; % Número de subcanales empleados para cada SNR
            [Y_,C_]=eigs(inv(Hzp'*Hzp),P-L-M_,'lm');
            PhiOpt_=sqrt(Po/trace(B_^0.5))*(B_^0.25);
            UOpt_=[X_ Y_]; %X_ es Wm_
            Dm_=DFTMtx(M_);
            FOpt_=X_*PhiOpt_*Dm_;
            NumSymb=M_*1e2;

            % Probabilidad de error para el rango de SNR con descarte
            BERDrop(k)=BERDropZP(Hzp,FOpt_,NumSymb,M_,P,Var);
    end

    SNRNoDrop=[SNRc:1:SNR(length(SNR)),SNR(length(SNR))];
    NumSymb=M*1e2;

SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
    BERNoDrop=BERNoDropZP(FOpt,Hzp,M,P,NumSymb,SymbSec,SNRNoDrop);

    % Matriz con vectores fila de BER's experimentales para cada canal
    BerZPDropTotal(w,:)=[BERDrop, BERNoDrop];
    SNRZPDropTotal(w,:)=[SNRDrop, SNRNoDrop];
    MZP=[MZPDrop,M*ones([1 length(SNRNoDrop)])];

    clear X_ Y_ UOpt_ FOpt_ SNRc_ B_ X_
    clear BERDrop BERNoDrop SNRDrop SNRNoDrop
    clear h Fcp_ofdm Fcp_mmse_zf_i Hcp X Y B Z index AH f I D Pm h taps
end


 % Evaluamos la BER media y teórica de cada sistema con precodificador
 % específico y esquema de descarte de subcanales, para todos los
 % canales aleatorios generados.

BER1=1/w*sum(BerDMTotal(:,:));
BER2=1/cp*sum(BerCPDropTotal(:,:));
BER3=1/w*sum(BerZPDropTotal(:,:));
semilogy(1/w*sum(SNRDMTotal(:,:)),BER1,'-
*',1/cp*sum(SNRCPDropTotal(:,:)),BER2,'-
>',1/w*sum(SNRZPDropTotal(:,:)),BER3,'-o');

axis([0 SNR(length(SNR)) 1e-4 1])
grid
```

```
Xlabel('SNR(dB)');
YLabel('BER');
legend('Waterfilling DMT','CP-MBER-DROP','ZP-MBER-DROP',0);
```

Published with MATLAB® 7.0

❖ MMSE.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros o código cíclico,         %
%  ecualización por Mínimo Error Cuadrático Medio y detección por umbral, %
%  sobre un canal con respuesta buena frecuencial                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

%Respuesta impulsiva del canal
h=[0.1710 + 0.1292i 0.1245 - 1.1361i -1.5995 + 0.7320i 1];
K=sum(abs(h).^2);
h=h/sqrt(K);            %Normalizamos
L=length(h)-1;          %Longitud de la respuesta impulsiva

SNR=[10.5,11:1:20];     %Rango para la Relación Señal-Ruido de bloque
M=32;                   %Tamaño de Bloque inicial de símbolos
Po=1;                   %Restricción de potencia de bloque a transmitir
NumSymb=3*M*1e4;        %Número de símbolos a transmitir
P=35;                   %P>=M+L Tamaño final de bloque a transmitir


%Respuesta Impulsiva del Canal
figure(1)
x=0:M-1;
stem(x,abs(fft(h,M)))
axis([-1 32 0 2])
Xlabel('Índice del subcanal, i');
YLabel('|H(exp(j2pi*i/(P-L))|');

%Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

%Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm');   %B es la matriz Lambda de autovalores
Dm=DFTMtx(M);                       %Construimos la matriz DFT normalizada

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*sqrt(B)-Var.*B);

    % Precodificadores ZP
    F=X*Phi*Dm;                         % MBER-ZP para ecualización MMSE
    F_zpmmsei=X*Phi*eye([M M]);         % Precodificador MMSE-I-ZP
```

```
    F_zptdma=sqrt(Po/M)*eye([M M]);     % Precodificador TDMA-ZP
    F_zpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([M M]);% Precodificador OFDM-ZP

    % BER para los distintos precodificadores ZP y ecualización MMSE
    [ZP_MBER(k),PeT(k)]=BER_MMSE_ZP(Hzp,F,NumSymb,SymbSec,M,P,Var);
    ZP_MMSEI_BER(k)=BER_MMSE_ZP(Hzp,F_zpmmsei,NumSymb,SymbSec,M,P,Var);
    ZP_TDMA_BER(k)=BER_MMSE_ZP(Hzp,F_zptdma,NumSymb,SymbSec,M,P,Var);
    ZP_OFDM_BER(k)=BER_MMSE_ZP(Hzp,F_zpofdm,NumSymb,SymbSec,M,P,Var);
end

% Dibujamos la BER Teórica y para el sistema con los distintos
% precodificadores ZP que emplea ecualización MMSE
figure(2)
semilogy (SNR,ZP_MBER,'-d', SNR,PeT,'-o', SNR,ZP_MMSEI_BER,'->',
SNR,ZP_TDMA_BER,'-*', SNR,ZP_OFDM_BER,'-<');
axis([11 20 1e-5 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('ZP-MBER','MBER-ZP Teórica','ZP-MMSE-I','ZP-TDMA','ZP-OFDM',0);

%Construimos la matriz CP con la respuesta del canal
f=fft(h,P-L);
A=diag(f.');
Hcp=DFTMtx(P-L)'*A*DFTMtx(P-L);
[Z,index]=sort(diag(inv(A'*A)),'descend');
I = eye([length(index),length(index)]);
PER= I(index,:);
B=diag(Z);
D=DFTMtx(P-L);

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*(B^0.5)-Var.*B);

    % Precodificadores ZP
    F=D'*PER.'*Phi*D;             %Construimos MBER-CP para ecualización MMSE
    F_cpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([P-L P-L]);%Precodificador OFDM-CP
    F_dmt=D'*PER.'*Phi*eye([P-L P-L]);               % Precodificador DMt-CP

    % BER para los distintos precodificadores CP y ecualización MMSE
    [CP_MBER(k),PeT(k)]=BER_MMSE_CP(Hcp,F,NumSymb,SymbSec,M,P,L,Var);
    CP_OFDM_BER(k)=BER_MMSE_CP(Hcp,F_cpofdm,NumSymb,SymbSec,M,P,L,Var);
    MMSE_DMT_BER(k)=BER_MMSE_CP(Hcp,F_dmt,NumSymb,SymbSec,M,P,L,Var);
end


% Dibujamos la BER Teórica y para el sistema con los distintos
% precodificadores CP que emplea ecualización MMSE
figure(3)
semilogy (SNR,CP_MBER,'-d', SNR,PeT,'-o',SNR,CP_OFDM_BER,'-
>',SNR,MMSE_DMT_BER,'-*')
axis([11 20 1e-5 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('CP-MBER','MBER-CP Teórica','CP-OFDM','MMSE DMT')
```

```
% Cálculo del umbral por debajo del cual hay descarte de subcanales en el
% esquema MBER-ZP
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm'); %B es Lambda
SNRc=10*log10(1/P*max(4/M*((trace(B^0.5))^2)-
trace(B),trace(B^0.5)*sqrt(B(1,1))-trace(B)));
SNRDrop=[0:abs(SNRc)/24:SNRc];

% Cálculo de la BER para SNR por debajo de la umbral de descarte
for k=1:length(SNRDrop)
        No=1/(10^(0.1*SNRDrop(k)));
        ro=Po/No;
        M_=M;
        SNRc_=SNRc;
        X_=X;
        B_=B;

        while ro <= 10^(SNRc_/10)
            M_=M_-1;
            B_(1,:)=[];
            B_(:,1)=[];
            SNRc_=10*log10(1/P*max(4/M_*((trace(B_^0.5))^2)-
trace(B_),trace(B_^0.5)*sqrt(B_(1,1))-trace(B_)));;
            X_(:,1)=[];
        end

        MZPDrop(k)=M_; % Número de subcanales empleados para cada SNR
        Var=No/P;
        Phi_=sqrt((Po+Var*trace(B_))/trace(B_^0.5).*sqrt(B_)-Var.*B_);
        D_=DFTMtx(M_);

        F_=X_*Phi_*D_;
        NumSymb=2*M_*1e4;

SymbSec_=(1/sqrt(2))*(sign(randn(NumSymb/M_,M_))+i*sign(randn(NumSymb/M_,M_))
);

        % Probabilidad de error para el rango de SNR con descarte
        BERDrop(k)=BER_MMSE_ZP(Hzp,F_,NumSymb,SymbSec_,M_,P,Var);
end

% Dibujamos la BER para el esquema MBER-ZP con descarte de subcanales y
% ecualización MMSE
figure(4)
semilogy ([SNRDrop,SNR],[BERDrop,ZP_MBER],'-r');
axis([0 25 1e-5 1])
grid
hold on


% Cálculo del umbral por debajo del cual hay descarte de subcanales en el
% esquema MBER-CP
B=diag(Z);
X=D'*PER.';
SNRc=10*log10(1/P*max(4/M*((trace(B^0.5))^2)-
trace(B),trace(B^0.5)*sqrt(B(1,1))-trace(B)));
SNRDropCP=0:abs(SNRc)/25:SNRc;

% Cálculo de la BER para SNR por debajo de la umbral de descarte
for k=1:length(SNRDropCP)
        No=1/(10^(0.1*SNRDropCP(k)));
        ro=Po/No;
```

```
        M_=M;
        SNRc_=SNRc;
        X_=X;
        B_=B;
        while ro <= 10^(SNRc_/10)
            M_=M_-1;
            B_(1,:)=[];
            B_(:,1)=[];
            SNRc_=10*log10(1/P*max(4/M_*((trace(B_^0.5))^2)-
trace(B_),trace(B_^0.5)*sqrt(B_(1,1))-trace(B_)));;
            X_(:,1)=[];
        end

        MCPDrop(k)=M_; % Número de subcanales empleados para cada SNR
        Var=No/P;
        Phi_=sqrt((Po+Var*trace(B_))/trace(B_^0.5).*sqrt(B_)-Var.*B_);
        D_=DFTMtx(M_);

        F_=X_*Phi_*D_;
        NumSymb=2*M_*1e4;

SymbSec_=(1/sqrt(2))*(sign(randn(NumSymb/M_,M_))+i*sign(randn(NumSymb/M_,M_))
);

        % Probabilidad de error para el rango de SNR con descarte
        BERDropCP(k)=BER_MMSE_CP(Hcp,F_,NumSymb,SymbSec_,M_,P,L,Var);
end

% Dibujamos la BER para el esquema MBER-CP con descarte de subcanales y
% ecualización MMSE
CP_MBER(:,1)=[];
SNR(:,1)=[];
semilogy ([SNRDropCP,SNR],[BERDropCP,CP_MBER],'-b');

% Construimos el Precodificador DMT Waterfilling
SNR=0:1:25;
NumSymb=2*M*1e4;
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));
u=1;

% Recorremos una tabla para los valores SNR
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    x0=100;
    Var=No/P;
    options = optimset('Display','off'); % Desactiva el Display
    x = fsolve(@myfun,x0,options,Po,Var,A);
    [Result,AWF]=myfun(x,Po,Var,A);
    indices=find(AWF);
    Qwf=zeros(size(AWF));
    Qwf(indices)=1;
    if  length(u)~=0
            Aux=diag(diag(Qwf)-1);
            [u,v]=find(Aux);
            kdrop=k;
            SNRd=SNR(kdrop);%SNR umbral de descarte de subcanales en WF-DMT

    else
    % F DMT Waterfilling
    Fwf_dmt=DFTMtx(P-L)'*Qwf*Qwf.'*AWF*Qwf;
    % Probabilidad de error para el rango de SNR sin descarte
```

```
    WF_DMT_BERNoDrop(k-
kdrop)=BER_MMSE_CP(Hcp,Fwf_dmt,NumSymb,SymbSec,M,P,L,Var);
    end

end

% Rango de SNR en el que hay descarte de subcanales
SNRDropDMT=0:0.3:SNRd;
for k=1:length(SNRDropDMT)
        No=1/(10^(0.1*SNRDropDMT(k)));
        x0=100;
        Var=No/P;
        options = optimset('Display','off'); % Desactiva el Display
        x = fsolve(@myfun,x0,options,Po,Var,A);
        [Result,AWF]=myfun(x,Po,Var,A);
        indices=find(AWF);
        Qwf=zeros(size(AWF));
        Qwf(indices)=1;
        Aux=diag(diag(Qwf)-1);
        [u,v]=find(Aux);
        Qwf(:,u) = [];
        M_=M-length(u);
        MDMTDrop(k)=M_; % Número de subcanales que se emplean para cada SNR
        NumSymb=2*M_*1e4;

SymbSec_=(1/sqrt(2))*(sign(randn(NumSymb/M_,M_))+i*sign(randn(NumSymb/M_,M_))
);

        % F DMT Waterfilling
        Fwf_dmt_=DFTMtx(P-L)'*Qwf*Qwf.'*AWF*Qwf;

        % Probabilidad de error para el rango de SNR con descarte

WF_DMT_BERDrop(k)=BER_MMSE_CP(Hcp,Fwf_dmt_,NumSymb,SymbSec_,M_,P,L,Var);
 end
BerDMTotal=[WF_DMT_BERDrop,WF_DMT_BERNoDrop];
SNRNoDropDMT=SNR(kdrop+1):1:SNR(length(SNR));
SNRDMTotal=[SNRDropDMT,SNRNoDropDMT];

% Dibujamos la BER para el esquema WF-DMT con descarte de subcanales y
% ecualización MMSE
semilogy(SNRDMTotal,BerDMTotal,'-g')
Xlabel('SNR(dB)');
YLabel('BER');
legend('ZP-MBER-DROP','CP-MBER-DROP','Water-filing DMT',0);
```

Published with MATLAB® 7.0

❖ BER_MMSE_ZP.m

```
function [Ber,Pe]=BER_MMSE_ZP(H,F,NumSymb,SymbSec,M,P,Var)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para una determinada SNR, en un sistema de comunicaciones con
%       precodificador, transmisión por bloques con relleno de ceros (ZP),
%       ecualización MMSE y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       H: Matriz del Canal (ZP)
```

```
%       F: Matriz del Precodificador (ZP)
%       NumSymb: Número de símbolos que se trasmiten
%       SymbSec: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       Var: Varianza del Ruido para una SNR específica.

% Matriz de ecualización
G=F'*H'*inv(Var*eye([P P])+H*F*F'*H');

% Bloques de M Simbolos a bloques de P (ZP)
SymbTx=(H*F*SymbSec.').';

% Ruido de varianza Var
Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

% Secuencia de símbolos recibidos
SymbRx=SymbTx+Noise;

% Secuencia de símbolos ecualizados
SymbSecRx=(G*SymbRx.').';

% Secuencia de símbolos estimados en bloques de M
SymbEst=sign(real(SymbSecRx))+i*sign(imag(SymbSecRx));
bitsqam_KO=0;
for s=1:NumSymb/M;
    for t=1:M

        % Si parte real e imaginaria incorrecta, símbolo recibido con 2 bits
erróneos
        if real(SymbEst(s,t))*real(SymbSec(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbSec(s,t))<0
           bitsqam_KO=bitsqam_KO + 2;

        % Si parte real o imaginaria incorrecta, símbolo recibido con 1 bit
erróneo
        elseif real(SymbEst(s,t))*real(SymbSec(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbSec(s,t))<0
           bitsqam_KO=bitsqam_KO + 1;
        end
    end
end

%Probabilidad de error de bit experimental teórica para ecualización MMSE(CP)
Ber=bitsqam_KO/(2*NumSymb);
Q=diag(G*H*F);
R=diag(Q);

%Probabilidad de error de bit mínima teórica para ecualización MMSE(CP)
Pe=0.5/M*sum(erfc(real(1./(sqrt(diag(2*(inv(R)-eye([M M])))))))));
```

Published with MATLAB® 7.0

❖ BER_MMSE_CP.m

```
function [Ber,Pe]=BER_MMSE_CP(H,F,NumSymb,SymbSec,M,P,L,Var)

% DESCRIPCIÓN:
%       Esta función calcula la probabilidad de error de bit experimental y
%       teórica para una determinada SNR, en un sistema de comunicaciones con
```

```
%       precodificador, transmisión por bloques con código cíclico (CP),
%       ecualización MMSE y detección por umbral de símbolos QAM.
%
% ENTRADAS:
%       H: Matriz del Canal (ZP)
%       F: Matriz del Precodificador (ZP)
%       NumSymb: Número de símbolos que se trasmiten
%       SymbSec: Matriz de Símbolos QAM generados en bloques (Filas) de M
%       M: Tamaño de Bloque Inicial
%       P: Tamaño de Bloque Final a transmitir
%       L: Longitud de la respuesta impulsiva del canal
%       Var: Varianza del Ruido para una SNR específica.

% Matriz de ecualización
G=F'*H'*inv(Var*eye([P-L P-L])+H*F*F'*H');

% Bloques de M Simbolos a bloques de P (CP)
SymbTx=(H*F*SymbSec.').';

% Ruido de varianza Var
Noise=sqrt(Var/2)*(randn(NumSymb/M,P)+i*(randn(NumSymb/M,P)));

% Secuencia de símbolos recibidos
Rcp=[zeros([P-L L]), eye([P-L P-L])];
SymbRx=SymbTx+(Rcp*Noise.').';

% Secuencia de símbolos ecualizados
SymbSecRx=(G*SymbRx.').';

% Secuencia de símbolos estimados en bloques de M
SymbEst=sqrt(1/2)*(sign(real(SymbSecRx))+i*sign(imag(SymbSecRx)));
bitsqam_KO=0;
for s=1:NumSymb/M;
    for t=1:M

        % Si parte real e imaginaria incorrecta, símbolo recibido con 2 bits
erróneos
        if real(SymbEst(s,t))*real(SymbSec(s,t))<0 &
imag(SymbEst(s,t))*imag(SymbSec(s,t))<0
           bitsqam_KO=bitsqam_KO + 2;

        % Si parte real o imaginaria incorrecta, símbolo recibido con 1 bit
erróneo
        elseif real(SymbEst(s,t))*real(SymbSec(s,t))<0 |
imag(SymbEst(s,t))*imag(SymbSec(s,t))<0
           bitsqam_KO=bitsqam_KO + 1;
        end
    end
end

%Probabilidad de error de bit experimental teórica para ecualización MMSE(CP)
Ber=bitsqam_KO/(2*NumSymb);
Q=diag(G*H*F);
R=diag(Q);

%Probabilidad de error de bit mínima teórica para ecualización MMSE(CP)
Pe=0.5/M*sum(erfc(real(1./(sqrt(diag(2*(inv(R)-eye([M M])))))))));
```

Published with MATLAB® 7.0

❖ MMSE_ZP_Flat.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros (ZP), ecualización por    %
%  Mínimo Error Cuadrático Medio y detección por umbral, sobre 500        %
%  realizaciones de un canal selectivo en frecuencia con desvanecimiento  %
%  Rayleigh lento con perfil plano en retraso                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

L=4;          % Longitud de la respuesta impulsiva
M=16;         % Tamaño de Bloque inicial de símbolos
P=20;         % P>=M+L Tamaño final de bloque a transmitir
SNR=[0:1:30]; % Rango para la Relación Señal-Ruido de bloque
Po=1;         % Restricción de potencia de bloque a transmitir

NumSymb=2*M*1e2; % Número de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

for w=1:500  % 500 realizaciones del canal
cont=w

% Construimos la respuesta impulsiva del canal selectivo en frecuencia con
% desvanecimiento Rayleigh lento con perfil plano en retraso
h=sqrt(0.5/(L+1))*(randn(1,L+1)+i*randn(1,L+1));

% Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm');    %B es la matriz Lambda de autovalores
Dm=DFTMtx(M);                        %Construimos la matriz DFT normalizada

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*sqrt(B)-Var.*B);

    % Precodificadores ZP
    F=X*Phi*Dm;                    % MBER-ZP para ecualización MMSE
    F_zpmmsei=X*Phi*eye([M M]);    % Precodificador MMSE-I-ZP
    F_zptdma=sqrt(Po/M)*eye([M M]); % Precodificador TDMA-ZP
    F_zpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([M M]);% Precodificador OFDM-ZP

    % BER para los distintos precodificadores ZP y ecualización MMSE
    [ZP_MBER(w,k),PeT(w,k)]=BER_MMSE_ZP(Hzp,F,NumSymb,SymbSec,M,P,Var);
    ZP_MMSEI_BER(w,k)=BER_MMSE_ZP(Hzp,F_zpmmsei,NumSymb,SymbSec,M,P,Var);
```

```
    ZP_TDMA_BER(w,k)=BER_MMSE_ZP(Hzp,F_zptdma,NumSymb,SymbSec,M,P,Var);
    ZP_OFDM_BER(w,k)=BER_MMSE_ZP(Hzp,F_zpofdm,NumSymb,SymbSec,M,P,Var);
end
clear h Hzp c r X B Dm No Var Phi F F_zpmmsei F_zptdma F_zpofdm
end

% Dibujamos la BER media Teórica y para el sistema con los distintos
% precodificadores ZP que emplea ecualización MMSE, sobre todos los canales
% característicos planos en retraso con desvanecimiento Rayleigh generados
figure(2)
semilogy (SNR,1/w*sum(ZP_MBER(:,:)),'-d',SNR,1/w*sum(PeT(:,:)),'-
o',SNR,1/w*sum(ZP_MMSEI_BER(:,:)),'->',SNR,1/w*sum(ZP_TDMA_BER(:,:)),'-*',
SNR,1/w*sum(ZP_OFDM_BER(:,:)),'-<');
axis([0 30 1e-6 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('ZP-MBER','ZP-MBER Teórica','ZP-MMSE-I','ZP-TDMA','ZP-OFDM',0);
```

Published with MATLAB® 7.0

❖ MMSE_ZP_Delay.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros (ZP), ecualización por    %
%  Mínimo Error Cuadrático Medio y detección por umbral, sobre 500        %
%  realizaciones de un canal selectivo en frecuencia con desvanecimiento  %
%  Rayleigh lento con perfil decadente en retraso                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

L=4;          % Longitud de la respuesta impulsiva
M=16;         % Tamaño de Bloque inicial de símbolos
P=20;         % P>=M+L Tamaño final de bloque a transmitir
SNR=[0:1:30]; % Rango para la Relación Señal-Ruido de bloque
Po=1;         % Restricción de potencia de bloque a transmitir

NumSymb=2*M*1e2; % Numeros de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

Beta=(2^L)/(2^(L+1)-1); % Constante que se emplea en la respuesta impulsiva

for w=1:500  % 500 realizaciones del canal
cont=w

% Construimos la respuesta impulsiva del canal selectivo en frecuencia con
% desvanecimiento Rayleigh lento con perfil decadente en retraso
for l=1:L+1

    h(l)=sqrt(0.5*Beta*2^(-l))*(randn(1)+i*randn(1));

end

%Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
```

```
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm'); %B es la matriz Lambda de autovalores
Dm=DFTMtx(M);                     %Construimos la matriz DFT normalizada

for k=1:length(SNR) % Recorremos una tabla para los valores de SNR por
bloque.
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*sqrt(B)-Var.*B);

    % Precodificadores ZP
    F=X*Phi*Dm;                           % MBER-ZP para ecualización MMSE
    F_zpmmsei=X*Phi*eye([M M]);           % Precodificador MMSE-I-ZP
    F_zptdma=sqrt(Po/M)*eye([M M]);       % Precodificador TDMA-ZP
    F_zpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([M M]);% Precodificador OFDM-ZP

    % BER para los distintos precodificadores ZP y ecualización MMSE
    [ZP_MBER(w,k),PeT(w,k)]=BER_MMSE_ZP(Hzp,F,NumSymb,SymbSec,M,P,Var);
    ZP_MMSEI_BER(w,k)=BER_MMSE_ZP(Hzp,F_zpmmsei,NumSymb,SymbSec,M,P,Var);
    ZP_TDMA_BER(w,k)=BER_MMSE_ZP(Hzp,F_zptdma,NumSymb,SymbSec,M,P,Var);
    ZP_OFDM_BER(w,k)=BER_MMSE_ZP(Hzp,F_zpofdm,NumSymb,SymbSec,M,P,Var);
end
clear h Hzp c r X B Dm No Var Phi F F_zpmmsei F_zptdma F_zpofdm
end

% Dibujamos la BER media Teórica y para el sistema con los distintos
% precodificadores ZP que emplea ecualización MMSE, sobre todos los canales
% característicos decadentes en retraso con desvanecimiento Rayleigh
generados
figure(2)
semilogy (SNR,1/w*sum(ZP_MBER(:,:)),'-d',SNR,1/w*sum(PeT(:,:)),'-
o',SNR,1/w*sum(ZP_MMSEI_BER(:,:)),'->',SNR,1/w*sum(ZP_TDMA_BER(:,:)),'-*',
SNR,1/w*sum(ZP_OFDM_BER(:,:)),'-<');
axis([0 30 1e-4 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('ZP-MBER','ZP-MBER Teórica','ZP-MMSE-I','ZP-TDMA','ZP-OFDM',0);
```

Published with MATLAB® 7.0

❖ MMSE_CP_Flat.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con código cíclico (CP), ecualización por      %
%  Mínimo Error Cuadrático Medio y detección por umbral, sobre 500        %
%  realizaciones de un canal selectivo en frecuencia con desvanecimiento  %
%  Rayleigh lento con perfil plano en retraso                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

L=4;          % Longitud de la respuesta impulsiva
M=16;         % Tamaño de Bloque inicial de símbolos
P=20;         % P>=M+L Tamaño final de bloque a transmitir
```

```
SNR=[0:1:30]; % Rango para la Relación Señal-Ruido de bloque
Po=1;         % Restricción de potencia de bloque a transmitir

NumSymb=2*M*1e2; % Número de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

for w=1:500   % 500 realizaciones del canal
cont=w

% Construimos la respuesta impulsiva del canal selectivo en frecuencia con
% desvanecimiento Rayleigh lento con perfil plano en retraso
h=sqrt(0.5/(L+1))*(randn(1,L+1)+i*randn(1,L+1));

% Construimos la matriz CP con la respuesta del canal
f=fft(h,P-L);
A=diag(f.');
Hcp=DFTMtx(P-L)'*A*DFTMtx(P-L);
[Z,index]=sort(diag(inv(A'*A)),'descend');
I = eye([length(index),length(index)]);
PER= I(index,:);
B=diag(Z);
D=DFTMtx(P-L);

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*(B^0.5)-Var.*B);

    % Precodificadores CP
    F=D'*PER.'*Phi*D;                       % CP-MBER para ecualización MMSE
    F_cpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([P-L P-L]);% Precodificador CP-
OFDM
    F_dmt=D'*PER.'*Phi*eye([P-L P-L]);      % Precodificador CP-DMT
    F_cpi=sqrt(Po/M)*eye([P-L P-L]);        % Precodificador CP-I

    % BER para los distintos precodificadores CP y ecualización MMSE
    [CP_MBER(w,k),PeT(w,k)]=BER_MMSE_CP(Hcp,F,NumSymb,SymbSec,M,P,L,Var);
    CP_OFDM_BER(w,k)=BER_MMSE_CP(Hcp,F_cpofdm,NumSymb,SymbSec,M,P,L,Var);
    MMSE_DMT_BER(w,k)=BER_MMSE_CP(Hcp,F_dmt,NumSymb,SymbSec,M,P,L,Var);
    CP_I_BER(w,k)=BER_MMSE_CP(Hcp,F_cpi,NumSymb,SymbSec,M,P,L,Var);
end

clear h Hcp A Z D B PER index I No Var Phi F F_dmt F_cpi F_cpofdm
end
% Dibujamos la BER media Teórica y para el sistema con los distintos
% precodificadores CP que emplea ecualización MMSE, sobre todos los canales
% característicos planos en retraso con desvanecimiento Rayleigh generados
figure(2)
semilogy (SNR,1/w*sum(CP_MBER(:,:)),'-d',SNR,1/w*sum(PeT(:,:)),'-
o',SNR,1/w*sum(MMSE_DMT_BER(:,:)),'->',SNR,1/w*sum(CP_I_BER(:,:)),'-*',
SNR,1/w*sum(CP_OFDM_BER(:,:)),'-<');
axis([0 30 1e-6 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('CP-MBER','CP-MBER Teórica','CP-MMSE-I','CP-I','CP-OFDM',0);
```

Published with MATLAB® 7.0

❖ MMSE_CP_Delay.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con código cíclico (CP), ecualización por      %
%  Mínimo Error Cuadrático Medio y detección por umbral, sobre 500        %
%  realizaciones de un canal selectivo en frecuencia con desvanecimiento  %
%  Rayleigh lento con perfil decadente en retraso                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

L=4;          % Longitud de la respuesta impulsiva
M=16;         % Tamaño de Bloque inicial de símbolos
P=20;         % P>=M+L Tamaño final de bloque a transmitir
SNR=[0:1:39]; % Rango para la Relación Señal-Ruido de bloque
Po=1;         % Restricción de potencia de bloque a transmitir

NumSymb=2*M*1e2; % Número de símbolos a transmitir

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

Beta=(2^L)/(2^(L+1)-1);% Constante que se emplea en la respuesta impulsiva

for w=1:500
cont=w

% Construimos la respuesta impulsiva del canal selectivo en frecuencia con
% desvanecimiento Rayleigh lento con perfil decadente en retraso
for l=1:L+1

    h(l)=sqrt(0.5*Beta*2^(-l))*(randn(1)+i*randn(1));

end

% Construimos la matriz CP con la respuesta del canal
f=fft(h,P-L);
A=diag(f.');
Hcp=DFTMtx(P-L)'*A*DFTMtx(P-L);
[Z,index]=sort(diag(inv(A'*A)),'descend');
I = eye([length(index),length(index)]);
PER= I(index,:);
B=diag(Z);
D=DFTMtx(P-L);

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*(B^0.5)-Var.*B);

    % Precodificadores CP
```

```
    F=D'*PER.'*Phi*D;                    % CP-MBER para ecualización MMSE
    F_cpofdm=sqrt(Po/M)*(DFTMtx(P-L))'*eye([P-L P-L]);%Precodificador CP-OFDM
    F_dmt=D'*PER.'*Phi*eye([P-L P-L]);  % Precodificador CP-DMT
    F_cpi=sqrt(Po/M)*eye([P-L P-L]);    % Precodificador CP-I

    % BER para los distintos precodificadores CP y ecualización MMSE
    [CP_MBER(w,k),PeT(w,k)]=BER_MMSE_CP(Hcp,F,NumSymb,SymbSec,M,P,L,Var);
    CP_OFDM_BER(w,k)=BER_MMSE_CP(Hcp,F_cpofdm,NumSymb,SymbSec,M,P,L,Var);
    MMSE_DMT_BER(w,k)=BER_MMSE_CP(Hcp,F_dmt,NumSymb,SymbSec,M,P,L,Var);
    CP_I_BER(w,k)=BER_MMSE_CP(Hcp,F_cpi,NumSymb,SymbSec,M,P,L,Var);
end

clear h Hcp A Z D B PER index I No Var Phi F F_dmt F_cpi F_cpofdm
end

% Dibujamos la BER media Teórica y para el sistema con los distintos
% precodificadores CP que emplea ecualización MMSE, sobre todos los canales
% característicos decadentes en retraso con desvanecimiento Rayleigh
generados
figure(2)
semilogy (SNR,1/w*sum(CP_MBER(:,:)),'-d',SNR,1/w*sum(PeT(:,:)),'-
o',SNR,1/w*sum(MMSE_DMT_BER(:,:)),'->',SNR,1/w*sum(CP_I_BER(:,:)),'-*',
SNR,1/w*sum(CP_OFDM_BER(:,:)),'-<');
axis([0 39 1e-4 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('CP-MBER','CP-MBER Teórica','CP-MMSE-I','CP-I','CP-OFDM',0);
```

Published with MATLAB® 7.0

❖ ZFvsMMSE.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Simulación de un sistema de comunicaciones con Precodificadores de     %
%  transmisión por bloques con relleno de ceros o código cíclico,         %
%  sobre un canal con buena respuesta frecuencial y detección por umbral, %
%  comparando entre el empleo de ecualización Cero Forzado y ecualización %
%  por Mínimo Error Cuadrático Medio                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all     % Eliminamos todas las variables existentes
close all     % Cerramos todas las ventanas

% Respuesta impulsiva del canal
h=[0.1710 + 0.1292i 0.1245 - 1.1361i -1.5995 + 0.7320i 1];
K=sum(abs(h).^2);
h=h/sqrt(K);         % Normalizamos
L=length(h)-1;       % Longitud de la respuesta impulsiva

SNR=[10:1:20];       % Rango para la Relación Señal-Ruido de bloque
M=32;                % Tamaño de Bloque inicial de símbolos
Po=1;                % Restricción de potencia de bloque a transmitir
P=35;                % P>=M+L Tamaño final de bloque a transmitir

NumSymb=5*M*1e4;     % Número de simbolos a transmitir

%Magnitud de la Respuesta Impulsiva del Canal
figure(1)
```

```
x=0:M-1;
stem(x,abs(fft(h,M)))
axis([0 32 0 2])
Xlabel('Indice del subcanal, i');
YLabel('|H(exp(j2pi*i/(P-L))|');

% Generamos la secuencia de símbolos 4-QAM en bloques de M
SymbSec=(1/sqrt(2))*(sign(randn(NumSymb/M,M))+i*sign(randn(NumSymb/M,M)));

% Caso ZP
% Construimos la matriz toeplitz ZP con la respuesta del canal
Hzp=zeros([P P-L]);
c=[h zeros([1 P-L-1])];
r=[h(1) zeros([1 P-L-1])];
Hzp=toeplitz(c,r);
[X,B]=eigs(inv(Hzp'*Hzp),M,'sm');   % B es la matriz Lambda de autovalores
[Y,C]=eigs(inv(Hzp'*Hzp),P-L-M,'lm');
PhiOpt=sqrt(Po/trace(B^0.5))*(B^0.25);
UOpt=[X Y];       % X es Wm
Dm=DFTMtx(M);     % Construimos la matriz DFT normalizada
FOpt=X*PhiOpt*Dm; % Construimos el Precodificador MBER-ZP para ecualización
ZF
FOptZP=FOpt;


% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*sqrt(B)-Var.*B);

    % Construimos el Precodificador MBER-ZP para ecualización MMSE
    F=X*Phi*Dm;

    % BER para el precodificador MBER-ZP con ecualización MMSE
    [ZP_MBER(k),PeT(k)]=BER_MMSE_ZP(Hzp,F,NumSymb,SymbSec,M,P,Var);

end

%BER para el precodificador MBER-ZP con ecualización ZF
[BER,Pe]=BerZP(FOptZP,Hzp,B,M,P,NumSymb,SymbSec,SNR,1);

% Dibujamos las BER's Teóricas y experimentales obtenidas para los sistemas
% con precodificador MBER-ZP en los casos de emplear ecualización ZF y MMSE
figure(2)
semilogy (SNR,BER,'g',SNR,Pe,'r',SNR,ZP_MBER,'-d', SNR,PeT,'-mo');
axis([10 20 1e-5 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('MBER-ZP-ZF','MBER-ZP-ZF Teórica','MBER-ZP-MMSE','MBER-ZP-MMSE
Teórica',0)


% Caso CP
% Construimos la matriz CP con la respuesta del canal
f=fft(h,P-L);
A=diag(f.');
Hcp=DFTMtx(P-L)'*A*DFTMtx(P-L);
[Z,index]=sort(diag(inv(A'*A)),'descend');
I = eye([length(index),length(index)]);
```

```
PER= I(index,:);
B=diag(Z);
D=DFTMtx(P-L);

% Recorremos una tabla para los valores de SNR por bloque
for k=1:length(SNR)
    No=1/(10^(0.1*SNR(k)));
    Var=No/P;
    Phi=sqrt((Po+Var*trace(B))/trace(B^0.5).*(B^0.5)-Var.*B);

    %Construimos el precodificador MBER-CP para ecualización MMSE
    F=D'*PER.'*Phi*D;

    % BER para el precodificador MBER-CP con ecualización MMSE
    [CP_MBER_MMSE(k),PeT(k)]=BER_MMSE_CP(Hcp,F,NumSymb,SymbSec,M,P,L,Var);

end

AH=diag(f.');
[X,C]=eigs(AH,P-L,'lm');
[Z,index]=sort(diag(C));
B=diag(Z);
AH_=[zeros([M P-L-M]),eye([M M])]*B*[zeros([P-L-M M]);eye([M M])];
I = eye([length(index),length(index)]);
D= I(index,:);
Pm=(D*inv(X)).'* [zeros([P-L-M M]);eye([M M])];

% Construimos el Precodificador MBER-ZP para ecualización ZF
FOpt=sqrt(Po/(trace((AH_'*AH_)^(-0.5))))*DFTMtx(P-L)'*Pm*((AH_'*AH_)^(-
0.25))*DFTMtx(M);
FOptCP=FOpt;

%BER para el precodificador MBER-CP con ecualización ZF
[CP_MBER_ZF,Pe]=BerCP(FOptCP,Hcp,AH_,M,P,L,NumSymb,SymbSec,SNR,1);


% Dibujamos las BER's Teóricas y experimentales obtenidas para los sistemas
% con precodificador MBER-CP en los casos de emplear ecualización ZF y MMSE
figure(3)
semilogy (SNR,CP_MBER_MMSE,'-d', SNR,PeT,'-o',SNR,CP_MBER_ZF,'-d',SNR,Pe,'-
<')
axis([10 20 1e-5 1])
grid
Xlabel('SNR(dB)');
YLabel('BER');
legend('MBER-CP-MMSE','MBER-CP-MMSE Teórica','MBER-CP-ZF','MBER-CP-ZF
Teórica')
```

Published with MATLAB® 7.0