

Anexo: Ficheros Matlab

A continuación se muestra el código de las funciones empleadas en este proyecto.

generagraficas

```
1 function generagraficas(M,tipomodulacion,pulso)
2 % F Javier Abad Peñalba
3 % Función que llama a las funciones que simulan los distintos
   sistemas y
4 % genera las gráficas de la BER frente a Eb/N0
5 % M orden de la modulación
6 % tipomodulación: modulación empleada
7 % pulso: pulso conformador utilizado
8
9 clc
10 close all
11
12 %% Parametros
13 EbN0dB=-5:1:10;      % Valores de EbNo
14
15 % Parametros códigos Hamming
16 n1=[7 15 31];
17 k1=[4 11 26];
18
19 % Parametros Cod Conv
20 codeRate1=1/2; constlen1=7; codegen1=[171 133]; tble1=32;
21 codeRate2=1/2; constlen2=9; codegen2=[561 753]; tble2=45;
22 codeRate3=1/2; constlen3=5; codegen3=[23 35];  tble3=16;
23 %% Llamada a funciones que calculan las BER
24
25 % Primero simulamos el canal sin codificador
26 BERnocod=bernocodcalaux(M,tipomodulacion,pulso);
27
28 % El siguiente paso simulamos los códigos Hamming
```

```
29 BERH74=berhammingcalcaux(M,tipomodulacion,pulso,n1(1),k1(1));
30 BERH1511=berhammingcalcaux(M,tipomodulacion,pulso,n1(2),k1(2));
31 BERH3126=berhammingcalcaux(M,tipomodulacion,pulso,n1(3),k1(3));
32
33 % Por último los convolucionales
34 BERConv1=bercodecalc2aux(M,tipomodulacion,pulso,codeRate1,
    constlen1,codegen1, tble1);
35 BERConv2=bercodecalc2aux(M,tipomodulacion,pulso,codeRate2,
    constlen2,codegen2, tble2);
36 BERConv3=bercodecalc2aux(M,tipomodulacion,pulso,codeRate3,
    constlen3,codegen3, tble3);
37 %% Gráficas
38 % Dibujamos Todas las curvas en una gráfica para comparar cuando
    conviene
39 % usar codificación y que tipo
40
41 %Gráfica Ber en funcion de Eb/no
42 figure(1)
43 plotH=semilogy(EbN0dB,(BERnocod),'b.-');
44 set(plotH,'LineWidth',1.5);
45 hold on
46 plotH=semilogy(EbN0dB,(BERH74),'mx-');
47 set(plotH,'LineWidth',1.5);
48 hold on
49 plotH=semilogy(EbN0dB,(BERH1511),'gx-');
50 set(plotH,'LineWidth',1.5);
51 hold on
52 plotH=semilogy(EbN0dB,(BERH3126),'rx-');
53 set(plotH,'LineWidth',1.5);
54 hold on
55 plotH=semilogy(EbN0dB,(BERConv1),'cx-');
56 set(plotH,'LineWidth',1.5);
57 hold on
58 plotH=semilogy(EbN0dB,(BERConv2),'kx-');
59 set(plotH,'LineWidth',1.5);
60 plotH=semilogy(EbN0dB,(BERConv3),'yx-');
61 set(plotH,'LineWidth',1.5);
62 hold on
63 xlabel('Eb/No, dB');
64 ylabel('Bit Error Rate');
65 title('Curva BER ');
66 grid on
67 legend('Sin Codificación','Hamming(7,4)','Hamming(15 11)','Hamming
    (31 26)','Conv1','Conv2','Conv3');
68 disp('BER simulada sin codificación')
69
70 % Imprimimos por pantalla los valores de la BER
71 BERnocod %para observar los valores de la BER simulada
72 disp('BER simulada codificación Hamming (7 4)')
73 BERH74
74 disp('BER simulada codificación Hamming (15 11)')
```

```

75 BERH1511
76 disp('BER simulada codificación Hamming (31 26)')
77 BERH3126
78 disp('BER simulada codificación Conv1')
79 BERConv1
80 disp('BER simulada codificación Conv2')
81 BERConv2
82 disp('BER simulada codificación Conv3')
83 BERConv3

```

bernocodcalaux

```

1  function BER=bernocodcalaux(M,tipomodulacion,pulso)
2
3  % F Javier Abad Peñalba
4  % simulación sistema sin codificador de canal
5  %
6  % BER: Devuelve en un vector el valor de la BER para los distintos
   valores
7  % de Eb/N0
8
9  % M:orden de la modulación
10 % tipomodulación: modulación empleada
11 % pulso: pulso conformador utilizado
12
13 clc;
14 close all;
15 %% Parámetros
16 EbNo = -5:1:10;
17 linEbNo = 10.^(EbNo(:).*0.1);
18 k = log2(M);
19
20 numSymb = 5e6; % Bits a transmitir
21 numPlot = 20;
22 Nsamp = 4;      % Tasa de muestreo
23 %% Generación de bits
24 x = randi([0 1], numSymb, 1);
25 % stem(0:numPlot-1, x(1:numPlot),'bx');
26 % axis([ 0 numPlot -0.2 1.2]); xlabel('Time'); ylabel('Amplitud')
   ;
27 % title('Símbolos Binarios antes de la Codificación Convolutiva')
   );
28 % legend off
29 %% Modulate
30
31 % Debemos elegir el tipo de modulación a emplear, para ello nos
   fijamos en
32 % el parámetro tipomodulacion
33
34 switch tipomodulacion
35     case 'psk'

```

```

36     % si es psk
37     hMod = modem.pskmod('M', M, 'PhaseOffset', pi/4, ...
38     'SymbolOrder', 'Gray', 'InputType', 'Bit');
39     msg_tx = modulate(hMod, x);
40     % Ahora dependiendo del pulso conformador
41     if pulso=='rectangulo'
42         msg_tx = rectpulse(msg_tx, Nsamp);
43     elseif pulso=='cosenoalza'
44         N = 40;                                % orrden del filtro
45         rolloff = 0.25;                        % Rolloff
46         filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
47         Raised Cosine', ...
48         'N,Beta', N, rolloff);                % Definición
49         rrcFilter = design(filtDef);           % Diseño
50         rrcFilter.Numerator = rrcFilter.Numerator * sqrt(Nsamp
51         );
52         % Normaliza el filtro
53         % Dibuja la respuesta impulsiva.
54         %fvtool(rrcFilter, 'impulse')
55         yUp = upsample(msg_tx, Nsamp);
56         msg_tx= filter(rrcFilter, yUp);
57
58         % Creaun diagrama de ojo.
59         % ploteye(msg_tx(N/2:2000), Nsamp);
60
61         %hScatter = commscope.ScatterPlot;
62         %hScatter = commscope.ScatterPlot;
63         %hScatter.Constellation = hMod.Constellation;
64         %hScatter.SamplesPerSymbol = Nsamp;
65         %hScatter.PlotSettings.Constellation = 'on';
66     end
67     % si la modulación es qam
68     case 'qam'
69         hMod= modem.qammod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER
70         ', 'Gray', ...
71         'INPUTTYPE', 'Bit');
72         msg_tx = modulate(hMod, x);
73         if pulso=='rectangulo'
74             msg_tx = rectpulse(msg_tx, Nsamp);
75         elseif pulso=='cosenoalza'
76             N = 40;                                % orden del filtro
77             rolloff = 0.25;                        % Rolloff
78             filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
79             Raised Cosine', ...
80             'N,Beta', N, rolloff);                % Definición
81             rrcFilter = design(filtDef);           % Diseño
82             rrcFilter.Numerator = rrcFilter.Numerator * sqrt(Nsamp
83             );
84             % Normaliza el filtro
85             % Dibuja la respuesta impulsiva.
86             %fvtool(rrcFilter, 'impulse')

```

```

82         yUp = upsample(msg_tx, Nsamp);
83         msg_tx= filter(rrcFilter, yUp);
84
85         % Crea un diagrama de ojo.
86         %ploteye(msg_tx(N/2:2000), Nsamp);
87     end
88     otherwise
89         warning('Warning 1', 'Tipo de modulación no valida');
90         exit; % Fuerza la terminación del programa
91
92 end
93
94 disp(['Tipo de modulacion: ' tipomodulacion ' con ' num2str(M) '
95     simbolos']);
96
97 %% Canal
98 % Enviamos la señal a través de un canal AWGN. El término,
99 % -10*log10(Nsamp), es necesario para escalar la potencia de ruido
100     con le
101 % muestreo. El término -10*log10(1/codeRate), escala la potencia
102     de ruido
103 % de acuerdo a la tasa de codificación
104
105 BER=zeros(1,16);
106 indice=1;
107 EbNo = -5:1:10;
108 linEbNo = 10.^(EbNo(:).*0.1);
109 for EbNo=-5:1:10
110
111     snr = EbNo -10*log10(Nsamp)+10*log10(k);% pasamos la señal por el
112         canal
113
114     msg_rx = awgn(msg_tx,snr,'measured');
115     %numModPlot = numEncPlot * Nsamp / k;
116     %tMod = (0:numModPlot-1) / Nsamp * k;
117     % plot(tMod, real(msg_tx(1:length(tMod))), 'c-', ...
118         % tMod, imag(msg_tx(1:length(tMod))), 'm-');
119     % axis([ min(tMod) max(tMod) -1.5 1.5]); xlabel('Tiempo'); ylabel
120         ('Amplitud');
121     % title('Símbolos Codificados tras modulación');
122     % legend('Fase', 'Cuadratura');
123
124 %% Señal Recibida
125 %% Demodulación
126 if tipomodulacion == 'psk'
127     hDemod = modem.pskdemod('M', M, 'PhaseOffset', pi/4, ...
128         'SymbolOrder', 'Gray', 'OutputType', 'Bit');
129     if pulso=='rectangulo'
130         msg_rx_int = intdump(msg_rx, Nsamp);
131     elseif pulso=='cosenoalza'

```

```

128     yRx = filter(rrcFilter, msg_rx);
129     yRx = downsample(yRx, Nsamp);
130     delay = N/(Nsamp*2); %Retardo de grupo
131     msg_rx_int = yRx(2*delay+1:end-2*delay);
132     update(hScatter, yRx)
133     title('Received Signal');
134 end
135 msg_demod = demodulate(hDemod, msg_rx_int); % demodulamos
136 elseif tipomodulacion == 'qam'
137     hDemod=modem.qamdemod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER',
        'Gray','OUTPUTTYPE', 'Bit', 'DECISIONTYPE', 'HARD DECISION
        ');
138     if pulso=='rectangulo'
139         msg_rx_int = intdump(msg_rx, Nsamp);
140     elseif pulso=='cosenoalza'
141         yRx = filter(rrcFilter, msg_rx);
142         yRx = downsample(yRx, Nsamp);
143         delay = N/(Nsamp*2); % Retardo de grupo
144         msg_rx_int = yRx(2*delay+1:end-2*delay);
145         %update(hScatter, yRx)
146         %title('Received Signal');
147     end
148     msg_demod = demodulate(hDemod, msg_rx_int);
149 end
150 % stem(tEnc, code(1:numEncPlot),'rx'); hold on;
151 % stem(tEnc, msg_demod(1:numEncPlot),'bo'); hold off;
152 % axis([0 numPlot -0.2 1.2]);
153 % xlabel('Tiempo'); ylabel('Amplitud'); title('Símbolos
    Demodulados' );
154
155
156
157 %% BER Computation
158 % Con biterr comparamos el mensaje original y el demodulado sin
    decodificar
159 % y codificado. Biterr calcula la Tasa de error de bir (BER)
160
161
162 [nCodErrs BERcoded] = biterr(x(1:end-(length(x)-length(
    msg_demod))), msg_demod(1:end));
163
164
165 BER(indice)=BERcoded;
166 indice=indice+1;
167 end

```

berhammingcalcaux

```

1 function BER=berhammingcalcaux(M,tipomodulacion,pulso,n1,k1)
2 % F Javier Abad Peñalba
3 % simulación con codificación de canal con

```

```

4 % empleando un codificador Hamming que le pasamos como parámetro
   en
5 % la llamada a la función. Además también elegimos el tipo de
   modulación
6 % que queremos, QAM o PSK y el nº de símbolos de la constelación M
7 %
8
9 % BER: Devuelve en un vector el valor de la BER para los distintos
   valores
10 % de Eb/N0
11
12 % M:orden de la modulación
13 % tipomodulación: modulación empleada
14 % pulso: pulso conformador utilizado
15 % n1 k1 parámetros del código
16
17 clc;
18 close all;
19
20 %% Parámetros
21 EbNo = -5:1:10;
22 linEbNo = 10.^(EbNo(:).*0.1);
23 k = log2(M);
24 codeRate=k1/n1;
25 numSymb = 5e6; % Bits a transmitir
26 numPlot = 20;
27 Nsamp = 4; % Tasa de muestreo
28 %% Generación de bits
29 x = randi( [0 1], numSymb, 1);
30 % stem(0:numPlot-1, x(1:numPlot),'bx');
31 % axis([ 0 numPlot -0.2 1.2]); xlabel('Time'); ylabel('Amplitud')
   ;
32 % title('Símbolos Binarios antes de la Codificación ');
33 % legend off
34
35 %% Encoder
36 % Definimos un código de Hamming y lo usamos
37 % para codificar los datos binarios.
38 % La figura muestrra los símbolos codificados,
39
40 code = encode(x,n1,k1,'hamming');
41 numEncPlot = numPlot / codeRate;
42 tEnc = (0:numEncPlot-1) * codeRate;
43 % stem(tEnc, code(1:length(tEnc)),'rx');
44 % axis([min(tEnc) max(tEnc) -0.2 1.2]); xlabel('Tiempo'); ylabel
   ('Amplitud');
45 % title('Símbolos Binarios después de la codificación ');
46
47 %% Modulate
48
49 % Debemos elegir el tipo de modulación a emplear, para ello nos

```

```
    fijamos en
50 % el parámetro tipomodulacion
51
52 switch tipomodulacion
53     case 'psk'
54         % si es psk
55         hMod = modem.pskmod('M', M, 'PhaseOffset', pi/4, ...
56             'SymbolOrder', 'Gray', 'InputType', 'Bit');
57         msg_tx = modulate(hMod, code);
58         % Ahora dependiendo del pulso conformador
59         if pulso=='rectangulo'
60             msg_tx = rectpulse(msg_tx, Nsamp);
61         elseif pulso=='cosenoalza'
62             N = 40;                                % Orden del filtro
63             rolloff = 0.25;                          % Rolloff
64             filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
65                 Raised Cosine', ...
66                 'N,Beta', N, rolloff);               % Definicion
67             rrcFilter = design(filtDef);              % Diseño del filtro
68                                                     % Normaliza el filtro
69             % Dibuja la respuesta impulsiva.
70             % fvtool(rrcFilter, 'impulse')
71             yUp = upsample(msg_tx, Nsamp);           % Upsample the
72                                                     signal
73             msg_tx= filter(rrcFilter, yUp);           % Apply square root
74                                                     raised cosine filter
75
76             % Crea un diagrama de ojo
77             %
78             %hScatter = commscope.ScatterPlot;
79             %hScatter = commscope.ScatterPlot;
80             %hScatter.Constellation = hMod.Constellation;
81             %hScatter.SamplesPerSymbol = Nsamp;
82             %hScatter.PlotSettings.Constellation = 'on';
83         end
84         % si la modulación es qam
85     case 'qam'
86         hMod= modem.qammod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER
87             ', 'Gray', ...
88             'INPUTTYPE', 'Bit');
89         msg_tx = modulate(hMod, code);
90         % Ahora dependiendo del pulso conformador
91         if pulso=='rectangulo'
92             msg_tx = rectpulse(msg_tx, Nsamp);
93         elseif pulso=='cosenoalza'
94             N = 40;                                % Orden del filtro
95             rolloff = 0.25;                          % Rolloff
```

```

95         filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
           Raised Cosine', ...
96         'N,Beta', N, rolloff);           % definición
97         rrcFilter = design(filtDef);      % Diseño
98         rrcFilter.Numerator = rrcFilter.Numerator * sqrt(Nsamp
           );
99
           % Normaliza el filtro
100        % dibuja la respuesta impulsiva
101        %fvtool(rrcFilter, 'impulse')
102        yUp = upsample(msg_tx, Nsamp);     % Sube la señal
103        msg_tx= filter(rrcFilter, yUp);    % Aplica el coseno
           alzado
104
105        % Create eye diagram for part of filtered signal.
106        %ploteye(msg_tx(N/2:2000), Nsamp);
107
108        %hScatter = commscope.ScatterPlot;
109        %hScatter = commscope.ScatterPlot;
110        %hScatter.Constellation = hMod.Constellation;
111        %hScatter.SamplesPerSymbol = Nsamp;
112        %hScatter.PlotSettings.Constellation = 'on';
113    end
114
115    otherwise
116        warning('Warning 1', 'Tipo de modulación no valida');
117        exit; % Fuerza la terminación del programa
118
119 end
120
121 disp(['Tipo de modulacion: ' tipomodulacion ' con ' num2str(M) '
       simbolos']);
122
123
124 %% Canal
125 % Envíamos la señal a través de un canal AWGN. El término,
126 %  $-10 \cdot \log_{10}(N_{\text{samp}})$ , es necesario para escalar la potencia de ruido
       con le
127 % muestreo. El término  $-10 \cdot \log_{10}(1/\text{codeRate})$ , escala la potencia
       de ruido
128 % de acuerdo a la tasa de codificación
129
130 BER=zeros(1,16);
131 indice=1;
132 EbNo = -5:1:10;
133 linEbNo = 10.^(EbNo(:).*0.1);
134 for EbNo=-5:1:10
135
136     snr = EbNo -10*log10(Nsamp)+10*log10(k)+10*log10(codeRate);
137
138     msg_rx = awgn(msg_tx,snr,'measured'); % pasamos la señal por el
       canal

```

```

139 numModPlot = numEncPlot * Nsamp / k;
140 tMod = (0:numModPlot-1) / Nsamp * k;
141 ');
142
143 %% Señal Recibida
144 %% Demodulación
145 if tipomodulacion == 'psk'
146     hDemod = modem.pskdemod('M', M, 'PhaseOffset', pi/4, ...
147         'SymbolOrder', 'Gray', 'OutputType', 'Bit');
148     if pulso=='rectangulo'
149         msg_rx_int = intdump(msg_rx, Nsamp);
150     elseif pulso=='cosenoalza'
151         yRx = filter(rrcFilter, msg_rx); % aplicamos el pulso
152         yRx = downsample(yRx, Nsamp);
153         delay = N/(Nsamp*2); % Retardo de grupo
154         msg_rx_int = yRx(2*delay+1:end-2*delay);
155         %update(hScatter, yRx)
156         %title('Received Signal');
157     end
158     msg_demod = demodulate(hDemod, msg_rx_int); %demodulamos
159 elseif tipomodulacion == 'qam'
160     hDemod=modem.qamdemod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER',
161         'Gray','OUTPUTTYPE', 'Bit', 'DECISIONTYPE', 'HARD DECISION
162         ');
163     if pulso=='rectangulo'
164         msg_rx_int = intdump(msg_rx, Nsamp);
165     elseif pulso=='cosenoalza'
166         yRx = filter(rrcFilter, msg_rx);
167         yRx = downsample(yRx, Nsamp);
168         delay = N/(Nsamp*2); % Retardo de grupo
169         msg_rx_int = yRx(2*delay+1:end-2*delay);
170         %update(hScatter, yRx)
171         %title('Received Signal');
172     end
173     msg_demod = demodulate(hDemod, msg_rx_int);
174 end
175 % stem(tEnc, code(1:numEncPlot),'rx'); hold on;
176 % stem(tEnc, msg_demod(1:numEncPlot),'bo'); hold off;
177 % axis([0 numPlot -0.2 1.2]);
178 % xlabel('Tiempo'); ylabel('Amplitud'); title('Símbolos
179     Demodulados' );
180
181 %% Decodificador
182 % Empleamos vitdec para decodificar la señal demodulada. empleamos
183     hard
184 % decision
185
186 msg_dec = decode(msg_demod,n1,k1,'hamming');
187 % stem(0:numPlot-1, x(1:numPlot), 'rx'); hold on;
188 % stem(0:numPlot-1, msg_dec(1+tblen:numPlot+tblen), 'bo'); hold
189     off;

```

```

185 % axis([0 numPlot -0.2 1.2]); xlabel('Tiempo'); ylabel('Amplitud
    ');
186 % title('Símbolos Decodificados' );
187 %size(x);
188 %size(msg_dec);
189 %% BER Computation
190 % Con biterr comparamos el mensaje original y el demodulado sin
    decodificar
191 % y codificado. Biterr calcula la Tasa de error de bir (BER)
192
193
194         [nCodErrs BERcoded] = biterr(x(1:end), msg_dec(1:end-(
            length(msg_dec)-length(x))));
195
196 BER(indice)=BERcoded;
197 indice=indice+1;
198 end

```

bernocodcalaux

```

1  function BER=bernocodcalaux(M,tipomodulacion,pulso)
2
3  % F Javier Abad Peñalba
4  % simulación sistema sin codificador de canal
5  %
6  % BER: Devuelve en un vector el valor de la BER para los distintos
    valores
7  % de Eb/N0
8
9  % M:orden de la modulación
10 % tipomodulación: modulación empleada
11 % pulso: pulso conformador utilizado
12
13 clc;
14 close all;
15 %% Parámetros
16 EbNo = -5:1:10;
17 linEbNo = 10.^(EbNo(:).*0.1);
18 k = log2(M);
19
20 numSymb = 5e6; % Bits a transmitir
21 numPlot = 20;
22 Nsamp = 4;      % Tasa de muestreo
23 %% Generación de bits
24 x = randi( [0 1], numSymb, 1);
25 % stem(0:numPlot-1, x(1:numPlot),'bx');
26 % axis([ 0 numPlot -0.2 1.2]); xlabel('Time'); ylabel('Amplitud')
    ;
27 % title('Símbolos Binarios antes de la Codificación Convolutional'
    );
28 % legend off

```

```
29 %% Modulate
30
31 % Debemos elegir el tipo de modulación a emplear, para ello nos
    fijamos en
32 % el parámetro tipomodulacion
33
34 switch tipomodulacion
35     case 'psk'
36         % si es psk
37         hMod = modem.pskmod('M', M, 'PhaseOffset', pi/4, ...
38             'SymbolOrder', 'Gray', 'InputType', 'Bit');
39         msg_tx = modulate(hMod, x);
40         % Ahora dependiendo del pulso conformador
41         if pulso=='rectangulo'
42             msg_tx = rectpulse(msg_tx, Nsamp);
43         elseif pulso=='cosenoalza'
44             N = 40;                                % orrden del filtro
45             rolloff = 0.25;                          % Rolloff
46             filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
                Raised Cosine', ...
47                 'N,Beta', N, rolloff);               % Definición
48             rrcFilter = design(filtDef);              % Diseño
49             rrcFilter.Numerator = rrcFilter.Numerator * sqrt(Nsamp
                );
50
51             % Normaliza el filtro
52             % Dibuja la respuesta impulsiva.
53             %fvtool(rrcFilter, 'impulse')
54             yUp = upsample(msg_tx, Nsamp);
55             msg_tx= filter(rrcFilter, yUp);
56
57             % Creaun diagrama de ojo.
58             % ploteye(msg_tx(N/2:2000), Nsamp);
59
60             %hScatter = commscope.ScatterPlot;
61             %hScatter = commscope.ScatterPlot;
62             %hScatter.Constellation = hMod.Constellation;
63             %hScatter.SamplesPerSymbol = Nsamp;
64             %hScatter.PlotSettings.Constellation = 'on';
65         end
66         % si la modulación es qam
67         case 'qam'
68             hMod= modem.qammod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER
                ', 'Gray', ...
69                 'INPUTTYPE', 'Bit');
70             msg_tx = modulate(hMod, x);
71             if pulso=='rectangulo'
72                 msg_tx = rectpulse(msg_tx, Nsamp);
73             elseif pulso=='cosenoalza'
74                 N = 40;                                % orden del filtro
75                 rolloff = 0.25;                          % Rolloff
76                 filtDef = fdesign.pulseshaping(Nsamp, 'Square Root
```

```

        Raised Cosine', ...
76      'N,Beta', N, rolloff);          % Definición
77      rrcFilter = design(filtDef);    % Diseño
78      rrcFilter.Numerator = rrcFilter.Numerator * sqrt(Nsamp
        );
79      % Normaliza el filtro
80      % Dibuja la respuesta impulsiva.
81      %fvtool(rrcFilter, 'impulse')
82      yUp = upsample(msg_tx, Nsamp);
83      msg_tx= filter(rrcFilter, yUp);
84
85      % Crea un diagrama de ojo.
86      %ploteye(msg_tx(N/2:2000), Nsamp);
87      end
88      otherwise
89          warning('Warning 1', 'Tipo de modulación no valida');
90          exit; % Fuerza la terminación del programa
91
92      end
93
94      disp(['Tipo de modulacion: ' tipomodulacion ' con ' num2str(M) '
        simbolos']);
95
96
97      %% Canal
98      % Enviamos la señal a través de un canal AWGN. El término,
99      % -10*log10(Nsamp), es necesario para escalar la potencia de ruido
        con le
100     % muestreo. El término -10*log10(1/codeRate), escala la potencia
        de ruido
101     % de acuerdo a la tasa de codificación
102
103     BER=zeros(1,16);
104     indice=1;
105     EbNo = -5:1:10;
106     linEbNo = 10.^(EbNo(:).*0.1);
107     for EbNo=-5:1:10
108
109         snr = EbNo -10*log10(Nsamp)+10*log10(k);% pasamos la señal por el
            canal
110
111         msg_rx = awgn(msg_tx,snr,'measured');
112         %numModPlot = numEncPlot * Nsamp / k;
113         %tMod = (0:numModPlot-1) / Nsamp * k;
114         % plot(tMod, real(msg_tx(1:length(tMod))), 'c-', ...
115         %      tMod, imag(msg_tx(1:length(tMod))), 'm-');
116         % axis([ min(tMod) max(tMod) -1.5 1.5]); xlabel('Tiempo'); ylabel
            ('Amplitud');
117         % title('Símbolos Codificados tras modulación');
118         % legend('Fase', 'Cuadratura');
119

```

```
120 %% Señal Recibida
121 %% Demodulación
122 if tipomodulacion == 'psk'
123     hDemod = modem.pskdemod('M', M, 'PhaseOffset', pi/4, ...
124     'SymbolOrder', 'Gray', 'OutputType', 'Bit');
125     if pulso=='rectangulo'
126         msg_rx_int = intdump(msg_rx, Nsamp);
127     elseif pulso=='cosenoalza'
128         yRx = filter(rrcFilter, msg_rx);
129         yRx = downsample(yRx, Nsamp);
130         delay = N/(Nsamp*2); %Retardo de grupo
131         msg_rx_int = yRx(2*delay+1:end-2*delay);
132         update(hScatter, yRx)
133         title('Received Signal');
134     end
135     msg_demod = demodulate(hDemod, msg_rx_int); % demodulamos
136 elseif tipomodulacion == 'qam'
137     hDemod=modem.qamdemod('M', M, 'PHASEOFFSET', 0, 'SYMBOLORDER',
138     'Gray','OUTPUTTYPE', 'Bit', 'DECISIONTYPE', 'HARD DECISION
139     ');
140     if pulso=='rectangulo'
141         msg_rx_int = intdump(msg_rx, Nsamp);
142     elseif pulso=='cosenoalza'
143         yRx = filter(rrcFilter, msg_rx);
144         yRx = downsample(yRx, Nsamp);
145         delay = N/(Nsamp*2); % Retardo de grupo
146         msg_rx_int = yRx(2*delay+1:end-2*delay);
147         %update(hScatter, yRx)
148         %title('Received Signal');
149     end
150     msg_demod = demodulate(hDemod, msg_rx_int);
151 end
152 % stem(tEnc, code(1:numEncPlot),'rx'); hold on;
153 % stem(tEnc, msg_demod(1:numEncPlot),'bo'); hold off;
154 % axis([0 numPlot -0.2 1.2]);
155 % xlabel('Tiempo'); ylabel('Amplitud'); title('Símbolos
156 Demodulados' );
157
158 %% BER Computation
159 % Con biterr comparamos el mensaje original y el demodulado sin
160 % decodificar
161 % y codificado. Biterr calcula la Tasa de error de bir (BER)
162
163 [nCodErrs BERcoded] = biterr(x(1:end-(length(x)-length(
164     msg_demod))), msg_demod(1:end));
165
166 BER(indice)=BERcoded;
```

```
166 indice=indice+1;  
167 end
```