ANEXO

A.1 Aleatorizador/desaleatorizador

```
function [datos_a] = aleatorizador(d_entrada)

x=0;
y=0;
z=0;
l=length(d_entrada);
for n = 1:l,
a=xor(y,z);
datos_a(n)=xor(d_entrada(n),a);
z=y;
y=x;
x=datos_a(n);
end

function [ds] = desaleatorizador(s)

x=0;
y=0;
z=0;
l=length(s);
for n = 1:l,
a=xor(y,z);
ds(n)=xor(s(n),a);
z=y;
y=x;
x=s(n);
end
```

A.2 Entrelazador/desentrelazador

```
function [fi] = interleaver(f,n_max_sim,N,n_mod_tipo)

y=n_max_sim;   %El nº de filas es n_max_sim
p=N*n_max_sim*n_mod_tipo;    %La profundidad es el número total de bits
x=p/y;    % x es el número de columnas
lf=length(f);
a=1;
na=lf/p;
for k = 0:na-1,
% Escritura por filas
for i = 1:x,
for j = 1:y,
M(j,i)=f(a);
a=a+1;
```

```
end
end
% Lectura por columnas
b=1;
for j = 1:y,
for i = 1:x,
fi(k*p+b)=M(j,i);
b=b+1;
end
end
end
%
```

```
function [sd] = deinterleaver(isd,n_max_sim,N,n_cod_tipo)

y=n_max_sim;
p=n_max_sim*N*n_mod_tipo;
x=p/y;
lisd=length(isd);
a=1;
na=lisd/p;
for k = 0:na-1,
% Escritura
for j = 1:y,
for i = 1:x,
RM(j,i)=isd(a);
a=a+1;
end
end
% Lectura
b=1;
for i = 1:x,
for j = 1:y,
sd(k*p+b)=RM(j,i);
b=b+1;
end
end
end
%
```

B.1. Funciones correspondientes al sistema básico

```
function[n_ber]=sist_basico(n_EbNo_dB,n_max_sim,n_cod_tipo,v_canal)
```

```
%TRANSMISOR
n_ldM=n_cod_tipo;
v_bits=randint(n_max_sim*n_ldM,1);
[v_tx, n_ldM]= mapeador(v_bits,n_cod_tipo);
v_tx_secuencia=v_tx(:);
%CANAL
n_sigma_sqr=10^(-n_EbNo_dB/10)/n_ldM;
v_noise=sqrt(n_sigma_sqr/2)*(randn(size(v_tx_secuencia))+j*randn(size(v_tx_secuenc
ia)));
%Aplico el canal
v_ch=filter(v_canal,1,v_tx_secuencia)+v_noise;
%RECEPTOR
v_rx=v_ch(:);
v_softbits_rx=demapeador(v_rx,n_cod_tipo);
v_bits_rx=v_softbits_rx>0;
%Calculo la ber para esta S/N
n_ber=mean(v_bits_rx(:)~=v_bits(:));
```

```
function [v_tx,n_ldM]=mapeador(v_bits,n_cod_tipo)

v_tx=[];
n_ldM=n_cod_tipo;
load ENC2.mat
load ENC4.mat
load ENC16.mat
load ENC64.mat
switch n_cod_tipo

   %MAPEADOR BPSK
   case 1
      for i=1:1:length(v_bits)
         switch v_bits(i)
            case {0}
               v_tx=[v_tx s2(1)];
            case {1}
               v_tx=[v_tx s2(2)];
         end
      end

   %MAPEADOR QPSK
   case 2
      for i=1:2:length(v_bits)
      %Voy cogiendo los bits de 2 en 2
      ent(1:2)=v_bits(i:i+1);
         switch ent(1)
            case {0}
               switch ent(2)
```

```
                case {0}
                v_tx=[v_tx s4(1)];
                case {1}
                v_tx=[v_tx s4(2)];
            end
        case {1}
            switch ent(2)
                case {0}
                v_tx=[v_tx s4(3)];
                case {1}
                v_tx=[v_tx s4(4)];
            end
    end
end

%MAPEADOR 16-QAM
case 4
    for i=1:4:length(v_bits)
        %Voy cogiendo los bits de 4 en 4
    ent(1:4)=v_bits(i:i+3);
    switch ent(1)
        case {0}  %0
            switch ent(2)
                case {0}  %00
                    switch ent(3)
                        case {0}  %000
                            switch ent(4)
                                case {0}   %0000
                                  v_tx=[v_tx s16(1)];
                                case {1}   %0001
                                    v_tx=[v_tx s16(2)];
                            end
                        case {1}
                            switch ent(4)
                                case {0}  %0010
                                    v_tx=[v_tx s16(3)];
                                case {1}  %0011
                                    v_tx=[v_tx s16(4)];
                            end
                    end
                case {1}  %01
                    switch ent(3)
                        case {0}   %010
                            switch ent(4)
                                case {0} %0100
                                  v_tx=[v_tx s16(5)];
                                case {1}  %0101
                                    v_tx=[v_tx s16(6)];
                            end
                        case {1}  %011
```

```
                switch ent(4)
                   case {0}  %0110
                      v_tx=[v_tx s16(7)];
                   case {1}   %0111
                      v_tx=[v_tx s16(8)];
                end %del switch ent(4)
             end %del switch ent(3)
          end  %del switch ent(2)
       case {1}  %1
          switch ent(2)
             case {0}  %10
                switch ent(3)
                   case {0}  %100
                      switch ent(4)
                         case {0}   %1000
                           v_tx=[v_tx s16(9)];
                         case {1}   %1001
                            v_tx=[v_tx s16(10)];
                      end
                   case {1}
                      switch ent(4)
                         case {0}  %1010
                            v_tx=[v_tx s16(11)];
                         case {1}  %1011
                            v_tx=[v_tx s16(12)];
                      end
                end
             case {1}  %11
                switch ent(3)
                   case {0}   %110
                      switch ent(4)
                         case {0} %1100
                           v_tx=[v_tx s16(13)];
                         case {1}  %1101
                            v_tx=[v_tx s16(14)];
                      end
                   case {1}  %111
                      switch ent(4)
                         case {0}  %1110
                            v_tx=[v_tx s16(15)];
                         case {1}   %1111
                            v_tx=[v_tx s16(16)];
                      end %del switch ent(4)
                end %del switch ent(3)
          end  %del switch ent(2)
       end  %del switch ent(1)
       end %del for de mod 16 QAM

       case 6
       %MAPEADOR 64 QAM
```

```
for i=1:6:length(v_bits)
%Voy cogiendo los bits de 6 en 6
ent(1:6)=v_bits(i:i+5);
switch ent(1)
   case {0}  %0
      switch ent(2)
         case {0}  %00
            switch ent(3)
               case {0}  %000
                  switch ent(4)
                     case {0}   %0000
                     switch ent(5)
                        case {0}  %00000
                           switch ent(6)
                              case {0} %000000
                                 v_tx=[v_tx s64(1)];
                              case {1}  %000001
                                 v_tx=[v_tx s64(2)];
                           end  %fin del swi6
                        case {1}  %00001
                           switch ent(6) %00001
                              case {0}  %000010
                                 v_tx=[v_tx s64(3)];
                              case {1}  %000011
                                 v_tx=[v_tx s64(4)];
                           end %fin del swi6
                     end  %fin del swi5
                     case{1}  %0001
                        switch ent(5)
                           case {0} %00010
                              switch ent(6)
                                 case {0} %000100
                                    v_tx=[v_tx s64(5)];
                                 case {1}  %000101
                                    v_tx=[v_tx s64(6)];
                              end  % fin del swi6
                           case{1}   %00011
                              switch ent(6)
                                 case {0}  %000110
                                    v_tx=[v_tx s64(7)];
                                 case {1}    %000111
                                    v_tx=[v_tx s64(8)];
                              end  %fin del swi6
                        end  %fin swi5
                  end %fin del swi4
               case {1} %001
                  switch ent(4)
                     case {0}  %0010
                        switch ent(5)
                           case {0} %00100
```

```
                        switch ent(6)
                           case {0}  %001000
                              v_tx=[v_tx s64(9)];
                           case {1}  %001001
                              v_tx=[v_tx s64(10)];
                        end  %fin del swi6
                     case{1}   %00101
                        switch ent(6)
                           case {0}  %001010
                              v_tx=[v_tx s64(11)];
                           case {1}  %001011
                              v_tx=[v_tx s64(12)];
                        end %fin del swi6
                  end  %fin del swi5
               case {1}  %0011
                  switch ent(5)
                     case {0} %00110
                        switch ent(6)
                           case {0}  %001100
                              v_tx=[v_tx s64(13)];
                           case {1}  %001101ç
                              v_tx=[v_tx s64(14)];
                        end  %fin del swi6
                     case{1}   %00111
                        switch ent(6)
                           case {0}  %001110
                              v_tx=[v_tx s64(15)];
                           case {1}  %001111
                              v_tx=[v_tx s64(16)];
                        end %fin del swi6
                  end  %fin del swi5
            end %fin del swi4
      end %fin del swi3
   case {1}  %01
      switch ent(3)
         case {0}  %010
            switch ent(4)
               case{0} %0100
                  switch ent(5)
                     case {0} %01000
                        switch ent(6)
                           case {0}  %010000
                              v_tx=[v_tx s64(17)];
                           case {1}   %010001
                              v_tx=[v_tx s64(18)];
                        end  %fin del swi6
                     case {1}  %01001
                        switch ent(6)
                           case {0}  %010010
                              v_tx=[v_tx s64(19)];
```

```
                    case {1}  %010011
                        v_tx=[v_tx s64(20)];
                end  %fin del swi6
            end  %fin de swi5
        case {1}  %0101
            switch ent(5)
                case {0}  %01010
                    switch ent(6)
                        case {0} %010100
                            v_tx=[v_tx s64(21)];
                        case {1}  %010101
                            v_tx=[v_tx s64(22)];
                    end %fin del swi6
                case {1}  %01011
                    switch ent(6)
                        case {0}  %010110
                            v_tx=[v_tx s64(23)];
                        case {1}  %010111
                            v_tx=[v_tx s64(24)];
                    end  %fin del swi6
            end  %fin del sw5
    end %fin de swi4
case {1}   %011
    switch ent(4)
        case {0}  %0110
            switch ent(5)
                case {0}  %01100
                    switch ent(6)
                        case {0} %011000
                            v_tx=[v_tx s64(25)];
                        case {1}  %011001
                            v_tx=[v_tx s64(26)];
                    end  %fin swi6
                case {1}   %01101
                    switch ent(6)
                        case {0} %011010
                            v_tx=[v_tx s64(27)];
                        case {1}  %011011
                            v_tx=[v_tx s64(28)];
                    end %fin swi6
            end %fin de swi5
        case {1}  %0111
            switch ent(5)
                case {0}  %01110
                    switch ent(6)
                        case {0} %011100
                            v_tx=[v_tx s64(29)];
                        case {1}  %011101
                            v_tx=[v_tx s64(30)];
                    end  %fin swi6
```

```
                        case {1}   %01111
                           switch ent(6)
                              case {0} %011110
                                 v_tx=[v_tx s64(31)];
                              case {1}  %011111
                                 v_tx=[v_tx s64(32)];
                           end %fin swi6
                     end %fin de swi5
                  end %fin swi4
               end %fin del switch 3
            end  %fin del swi2

         case {1}  %1
            switch ent(2)
               case {0}  %10
                  switch ent(3)
                     case {0}  %100
                        switch ent(4)
                           case {0}   %1000
                           switch ent(5)
                              case {0}  %10000
                                 switch ent(6)
                                    case {0} %100000
                                       v_tx=[v_tx s64(33)];
                                    case {1}  %100001
                                       v_tx=[v_tx s64(34)];
                                 end  %fin del swi6
                              case {1}  %00001
                                 switch ent(6) %00001
                                    case {0}  %100010
                                       v_tx=[v_tx s64(35)];
                                    case {1}  %100011
                                       v_tx=[v_tx s64(36)];
                                 end %fin del swi6
                           end  %fin del swi5
                           case{1}  %1001
                              switch ent(5)
                                 case {0} %10010
                                    switch ent(6)
                                       case {0} %100100
                                          v_tx=[v_tx s64(37)];
                                       case {1}  %000101
                                          v_tx=[v_tx s64(38)];
                                    end  % fin del swi6
                                 case{1}   %10011
                                    switch ent(6)
                                       case {0}  %100110
                                          v_tx=[v_tx s64(39)];
                                       case {1}    %100111
                                          v_tx=[v_tx s64(40)];
```

```
                    end  %fin del swi6
                end  %fin swi5
            end %fin del swi4
        case {1} %101
            switch ent(4)
                case {0}  %1010
                    switch ent(5)
                        case {0} %10100
                            switch ent(6)
                                case {0}  %101000
                                    v_tx=[v_tx s64(41)];
                                case {1}  %001001
                                    v_tx=[v_tx s64(42)];
                            end  %fin del swi6
                        case{1}   %00101
                            switch ent(6)
                                case {0}  %001010
                                    v_tx=[v_tx s64(43)];
                                case {1}  %001011
                                    v_tx=[v_tx s64(44)];
                            end %fin del swi6
                    end  %fin del swi5
                case {1}  %1011
                    switch ent(5)
                        case {0} %10110
                            switch ent(6)
                                case {0}  %101100
                                    v_tx=[v_tx s64(45)];
                                case {1}  %101101
                                    v_tx=[v_tx s64(46)];
                            end  %fin del swi6
                        case{1}   %10111
                            switch ent(6)
                                case {0}  %101110
                                    v_tx=[v_tx s64(47)];
                                case {1}  %101111
                                    v_tx=[v_tx s64(48)];
                            end %fin del swi6
                    end  %fin del swi5
            end %fin del swi4
    end %fin del swi3
case {1}  %11
    switch ent(3)
        case {0}  %110
            switch ent(4)
                case{0} %1100
                    switch ent(5)
                        case {0} %11000
                            switch ent(6)
                                case {0}  %110000
```

```
                        v_tx=[v_tx s64(49)];
                    case {1}   %110001
                        v_tx=[v_tx s64(50)];
                end %fin del swi6
            case {1}  %11001
                switch ent(6)
                    case {0}  %110010
                        v_tx=[v_tx s64(51)];
                    case {1}  %110011
                        v_tx=[v_tx s64(52)];
                end   %fin del swi6
        end  %fin de swi5
    case {1}  %1101
        switch ent(5)
            case {0}  %11010
                switch ent(6)
                    case {0} %110100
                        v_tx=[v_tx s64(53)];
                    case {1}  %010101
                        v_tx=[v_tx s64(54)];
                end %fin del swi6
            case {1}  %11011
                switch ent(6)
                    case {0}  %110110
                        v_tx=[v_tx s64(55)];
                    case {1}  %110111
                        v_tx=[v_tx s64(56)];
                end  %fin del swi6
        end  %fin del sw5
    end %fin de swi4
case {1}   %111
    switch ent(4)
        case {0}  %1110
            switch ent(5)
                case {0}  %11100
                    switch ent(6)
                        case {0} %111000
                            v_tx=[v_tx s64(57)];
                        case {1}  %111001
                            v_tx=[v_tx s64(58)];
                    end  %fin swi6
                case {1}   %11101
                    switch ent(6)
                        case {0} %111010
                            v_tx=[v_tx s64(59)];
                        case {1}  %111011
                            v_tx=[v_tx s64(60)];
                    end %fin swi6
            end %fin de swi5
        case {1}  %1111
```

```
                                switch ent(5)
                                   case {0}  %11110
                                      switch ent(6)
                                         case {0} %111100
                                            v_tx=[v_tx s64(61)];
                                         case {1}  %111101
                                            v_tx=[v_tx s64(62)];
                                      end  %fin swi6
                                   case {1}   %11111
                                      switch ent(6)
                                         case {0} %111110
                                            v_tx=[v_tx s64(63)];
                                         case {1}  %111111
                                            v_tx=[v_tx s64(64)];
                                      end %fin swi6
                                end %fin de swi5
                             end %fin swi4
                          end %fin del switch 3
                       end  %fin del swi2
                    end  %fin del swi1
                    end  %fin del for
end %fin del tipo de cod



function [v_dem] = demapeador(ms,n_cod_tipo)

v_dem=[];
load ENC2.mat
load ENC4.mat
load ENC16.mat
load ENC64.mat

switch n_cod_tipo
   case 1
      %CODIFICADOR BPSK
      for i=1:1:length(ms)
         pc=ms(i);
         a(1)=abs(pc-s2(1));
         a(2)=abs(pc-s2(2));
         ma=min(a);
         switch ma
            case a(1)
               v_dem=[v_dem 0];
            case a(2)
               v_dem=[v_dem 1];
         end
      end

   case 2
```

```
    %CODIFICADOR QPSK
     for i=1:1:length(ms)
        pc=ms(i);
        a(1)=abs(pc-s4(1));
        a(2)=abs(pc-s4(2));
        a(3)=abs(pc-s4(3));
        a(4)=abs(pc-s4(4));
        ma=min(a);
        switch ma
           case a(1)
              v_dem=[v_dem c4(1,1:2)];
           case a(2)
              v_dem=[v_dem c4(2,1:2)];
           case a(3)
              v_dem=[v_dem c4(3,1:2)];
           case a(4)
              v_dem=[v_dem c4(4,1:2)];
        end
     end

case 4
    %CODIFICADOR 16-QAM
    for i=1:1:length(ms)
       pc=ms(i);
        a(1)=abs(pc-s16(1));
        a(2)=abs(pc-s16(2));
        a(3)=abs(pc-s16(3));
        a(4)=abs(pc-s16(4));
        a(5)=abs(pc-s16(5));
        a(6)=abs(pc-s16(6));
        a(7)=abs(pc-s16(7));
        a(8)=abs(pc-s16(8));
        a(9)=abs(pc-s16(9));
        a(10)=abs(pc-s16(10));
        a(11)=abs(pc-s16(11));
        a(12)=abs(pc-s16(12));
        a(13)=abs(pc-s16(13));
        a(14)=abs(pc-s16(14));
        a(15)=abs(pc-s16(15));
        a(16)=abs(pc-s16(16));
        ma=min(a);
        switch ma
           case a(1)
              v_dem=[v_dem c16(1,1:4)];
           case a(2)
              v_dem=[v_dem c16(2,1:4)];
           case a(3)
              v_dem=[v_dem c16(3,1:4)];
           case a(4)
              v_dem=[v_dem c16(4,1:4)];
```

```
            case a(5)
               v_dem=[v_dem c16(5,1:4)];
            case a(6)
               v_dem=[v_dem c16(6,1:4)];
            case a(7)
               v_dem=[v_dem c16(7,1:4)];
            case a(8)
               v_dem=[v_dem c16(8,1:4)];
            case a(9)
               v_dem=[v_dem c16(9,1:4)];
            case a(10)
               v_dem=[v_dem c16(10,1:4)];
            case a(11)
               v_dem=[v_dem c16(11,1:4)];
            case a(12)
               v_dem=[v_dem c16(12,1:4)];
            case a(13)
               v_dem=[v_dem c16(13,1:4)];
            case a(14)
               v_dem=[v_dem c16(14,1:4)];
            case a(15)
               v_dem=[v_dem c16(15,1:4)];
            case a(16)
               v_dem=[v_dem c16(16,1:4)];
         end %end del switch ma
      end  %end del for 16 qam


   case 6
   %CODIFICADOR 64-QAM
   for i=1:1:length(ms)
          pc=ms(i);
           a(1)=abs(pc-s64(1));
           a(2)=abs(pc-s64(2));
           a(3)=abs(pc-s64(3));
           a(4)=abs(pc-s64(4));
           a(5)=abs(pc-s64(5));
           a(6)=abs(pc-s64(6));
           a(7)=abs(pc-s64(7));
           a(8)=abs(pc-s64(8));
           a(9)=abs(pc-s64(9));
           a(10)=abs(pc-s64(10));
           a(11)=abs(pc-s64(11));
           a(12)=abs(pc-s64(12));
           a(13)=abs(pc-s64(13));
           a(14)=abs(pc-s64(14));
           a(15)=abs(pc-s64(15));
           a(16)=abs(pc-s64(16));
           a(17)=abs(pc-s64(17));
           a(18)=abs(pc-s64(18));
```

```
a(19)=abs(pc-s64(19));
a(20)=abs(pc-s64(20));
a(21)=abs(pc-s64(21));
a(22)=abs(pc-s64(22));
a(23)=abs(pc-s64(23));
a(24)=abs(pc-s64(24));
a(25)=abs(pc-s64(25));
a(26)=abs(pc-s64(26));
a(27)=abs(pc-s64(27));
a(28)=abs(pc-s64(28));
a(29)=abs(pc-s64(29));
a(30)=abs(pc-s64(30));
a(31)=abs(pc-s64(31));
a(32)=abs(pc-s64(32));
a(33)=abs(pc-s64(33));
a(34)=abs(pc-s64(34));
a(35)=abs(pc-s64(35));
a(36)=abs(pc-s64(36));
a(37)=abs(pc-s64(37));
a(38)=abs(pc-s64(38));
a(39)=abs(pc-s64(39));
a(40)=abs(pc-s64(40));
a(41)=abs(pc-s64(41));
a(42)=abs(pc-s64(42));
a(43)=abs(pc-s64(43));
a(44)=abs(pc-s64(44));
a(45)=abs(pc-s64(45));
a(46)=abs(pc-s64(46));
a(47)=abs(pc-s64(47));
a(48)=abs(pc-s64(48));
a(49)=abs(pc-s64(49));
a(50)=abs(pc-s64(50));
a(51)=abs(pc-s64(51));
a(52)=abs(pc-s64(52));
a(53)=abs(pc-s64(53));
a(54)=abs(pc-s64(54));
a(55)=abs(pc-s64(55));
a(56)=abs(pc-s64(56));
a(57)=abs(pc-s64(57));
a(58)=abs(pc-s64(58));
a(59)=abs(pc-s64(59));
a(60)=abs(pc-s64(60));
a(61)=abs(pc-s64(61));
a(62)=abs(pc-s64(62));
a(63)=abs(pc-s64(63));
a(64)=abs(pc-s64(64));
ma=min(a);
switch ma
   case a(1)
      v_dem=[v_dem c64(1,1:6)];
```

```
case a(2)
   v_dem=[v_dem c64(2,1:6)];
case a(3)
   v_dem=[v_dem c64(3,1:6)];
case a(4)
   v_dem=[v_dem c64(4,1:6)];
case a(5)
   v_dem=[v_dem c64(5,1:6)];
case a(6)
   v_dem=[v_dem c64(6,1:6)];
case a(7)
   v_dem=[v_dem c64(7,1:6)];
case a(8)
   v_dem=[v_dem c64(8,1:6)];
case a(9)
   v_dem=[v_dem c64(9,1:6)];
case a(10)
   v_dem=[v_dem c64(10,1:6)];
case a(11)
   v_dem=[v_dem c64(11,1:6)];
case a(12)
   v_dem=[v_dem c64(12,1:6)];
case a(13)
   v_dem=[v_dem c64(13,1:6)];
case a(14)
   v_dem=[v_dem c64(14,1:6)];
case a(15)
   v_dem=[v_dem c64(15,1:6)];
case a(16)
   v_dem=[v_dem c64(16,1:6)];
case a(17)
   v_dem=[v_dem c64(17,1:6)];
case a(18)
   v_dem=[v_dem c64(18,1:6)];
case a(19)
   v_dem=[v_dem c64(19,1:6)];
case a(20)
   v_dem=[v_dem c64(20,1:6)];
case a(21)
   v_dem=[v_dem c64(21,1:6)];
case a(22)
   v_dem=[v_dem c64(22,1:6)];
case a(23)
   v_dem=[v_dem c64(23,1:6)];
case a(24)
   v_dem=[v_dem c64(24,1:6)];
case a(25)
   v_dem=[v_dem c64(25,1:6)];
case a(26)
   v_dem=[v_dem c64(26,1:6)];
```

```
case a(27)
   v_dem=[v_dem c64(27,1:6)];
case a(28)
   v_dem=[v_dem c64(28,1:6)];
case a(29)
   v_dem=[v_dem c64(29,1:6)];
case a(30)
   v_dem=[v_dem c64(30,1:6)];
case a(31)
   v_dem=[v_dem c64(31,1:6)];
case a(32)
   v_dem=[v_dem c64(32,1:6)];
case a(33)
   v_dem=[v_dem c64(33,1:6)];
case a(34)
   v_dem=[v_dem c64(34,1:6)];
case a(35)
   v_dem=[v_dem c64(35,1:6)];
case a(36)
   v_dem=[v_dem c64(36,1:6)];
case a(37)
   v_dem=[v_dem c64(37,1:6)];
case a(38)
   v_dem=[v_dem c64(38,1:6)];
case a(39)
   v_dem=[v_dem c64(39,1:6)];
case a(40)
   v_dem=[v_dem c64(40,1:6)];
case a(41)
   v_dem=[v_dem c64(41,1:6)];
case a(42)
   v_dem=[v_dem c64(42,1:6)];
case a(43)
   v_dem=[v_dem c64(43,1:6)];
case a(44)
   v_dem=[v_dem c64(44,1:6)];
case a(45)
   v_dem=[v_dem c64(45,1:6)];
case a(46)
   v_dem=[v_dem c64(46,1:6)];
case a(47)
   v_dem=[v_dem c64(47,1:6)];
case a(48)
   v_dem=[v_dem c64(48,1:6)];
case a(49)
   v_dem=[v_dem c64(49,1:6)];
case a(50)
   v_dem=[v_dem c64(50,1:6)];
case a(51)
   v_dem=[v_dem c64(51,1:6)];
```

```
            case a(52)
               v_dem=[v_dem c64(52,1:6)];
            case a(53)
               v_dem=[v_dem c64(53,1:6)];
            case a(54)
               v_dem=[v_dem c64(54,1:6)];
            case a(55)
               v_dem=[v_dem c64(55,1:6)];
            case a(56)
               v_dem=[v_dem c64(56,1:6)];
            case a(57)
               v_dem=[v_dem c64(57,1:6)];
            case a(58)
               v_dem=[v_dem c64(58,1:6)];
            case a(59)
               v_dem=[v_dem c64(59,1:6)];
            case a(60)
               v_dem=[v_dem c64(60,1:6)];
            case a(61)
               v_dem=[v_dem c64(61,1:6)];
            case a(62)
               v_dem=[v_dem c64(62,1:6)];
            case a(63)
               v_dem=[v_dem c64(63,1:6)];
            case a(64)
               v_dem=[v_dem c64(64,1:6)];
         end %end del switch ma
   end %fin del for de 64 qam
end %end de switch cod
```

B.2. Funciones correspondientes al sistema multiportadora

```
function[n_ber]=sist_multiportadora(n_EbNo_dB,n_max_sim,v_canal)

%TRANSMISOR
v_bits=randint(n_max_sim*2,1);
[v_tx]= modulador(v_bits);
v_tx_secuencia=v_tx(:);
%CANAL
n_sigma_sqr=10^(-n_EbNo_dB/10)/2;
v_noise=sqrt(n_sigma_sqr/2)*(randn(size(v_tx_secuencia))+j*randn(size(v_tx_secuenc
ia)));
%Aplico el canal
v_ch=filter(v_canal,1,v_tx_secuencia)+v_noise;
%RECEPTOR
v_rx=v_ch(:);
v_softbits_rx=demodulador(v_rx);
```

```
v_bits_rx=v_softbits_rx>0;
%Calculo la ber para esta S/N
n_ber=mean(v_bits_rx(:)~=v_bits(:));


function [v_tx]=modulador(v_bits)

v_tx=[];
turno=1;
j=sqrt(-1);

%MAPEADOR QPSK
for i=1:2:length(v_bits)
   ent(1:2)=v_bits(i:i+1);
   %voy cogiendo los bits de dos en dos
   switch turno
      case 1   %Portadora fc=0
         fc=0;
         turno=2;
      case 2
         fc=pi/16;  %Portadora fc=45
         turno=3;
      case 3
         fc=pi/8;
         turno=4;
      case 4
         fc=pi/4;
         turno=1;
   end

   switch ent(1)
       case {0}
          switch ent(2)
              case {0}
                    v_tx=[v_tx cos(fc+pi/4)+j*sin(fc+pi/4)];
              case {1}
                    v_tx=[v_tx cos(fc+(3*pi/4))+j*sin(fc+(3*pi/4))];
           end
        case {1}
          switch ent(2)
               case {0}
                    v_tx=[v_tx cos(fc+(5*pi/4))+j*sin(fc+(5*pi/4))];
               case {1}
                    v_tx=[v_tx cos(fc+(7*pi/4))+j*sin(fc+(7*pi/4))];
           end
   end
end


function [v_dem] = demodulador(ms)
```

```
v_dem=[];
turno=1;
load ENC4.mat
j=sqrt(-1);

for i=1:1:length(ms)
   switch turno
      case 1
         fc=0;
         turno=2;
      case 2
         fc=pi/16;
         turno=3;
      case 3
         fc=pi/8;
         turno=4;
      case 4
         fc=pi/4;
         turno=1;
   end
   pc=ms(i);
   s1=cos(fc+pi/4)+j*sin(fc+pi/4);
   s2=cos(fc+(3*pi/4))+j*sin(fc+(3*pi/4));
   s3=cos(fc+(5*pi/4))+j*sin(fc+(5*pi/4));
   s4=cos(fc+(7*pi/4))+j*sin(fc+(7*pi/4));

   a(1)=abs(pc-s1);
   a(2)=abs(pc-s2);
   a(3)=abs(pc-s3);
   a(4)=abs(pc-s4);
   ma=min(a);

   switch ma
      case a(1)
         v_dem=[v_dem c4(1,1:2)];
      case a(2)
         v_dem=[v_dem c4(2,1:2)];
      case a(3)
         v_dem=[v_dem c4(3,1:2)];
      case a(4)
         v_dem=[v_dem c4(4,1:2)];
   end
end
```

B.3. Funciones correspondientes al sistema OFDM

```
function[n_ber]=sist_ofdm (n_EbNo_dB,n_u0,N,n_max_sim,n_cod_tipo,v_canal)
Ng=((N-N*n_u0)/n_u0);
Ng=ceil(Ng);
load ENC4.mat

%TRANSMISOR
n_ldM=n_cod_tipo;
v_bits=randint(n_max_sim*N*n_ldM,1);
% Aleatorizador
v_bits_alea=aleatorizador(v_bits);
%Interleaver
v_bits_inter=interleaver(v_bits_alea, n_max_sim,N, n_cod_tipo);
%Modulador
[v_tx, n_ldM]= mapeador (v_bits_inter,n_cod_tipo);
m_tx=reshape(v_tx,N,n_max_sim);
m_tx_ofdm=sqrt(N).*ifft(m_tx,N);
%Inserción del Prefijo Cíclico
m_tx_ofdm_guard=inser_pc(m_tx_ofdm,N,Ng,n_max_sim);
v_ofdm_seq=m_tx_ofdm_guard(:);

%CANAL
n_sigma_sqr=10^(-n_EbNo_dB/10)*(N+Ng)/N/n_ldM;
v_eq=fft(v_canal,N);
v_ruido=sqrt(n_sigma_sqr/2)*(randn(size(v_ofdm_seq))+j*randn(size(v_ofdm_seq)));
%Aplico el canal al primer símbolo
v_ofdm_chan(1:N+Ng)=filter(v_canal,1,v_ofdm_seq(1:N+Ng))+v_ruido(1:N+Ng);
%Aplico el canal a los símbolos restantes uno a uno
for i=2:1:n_max_sim
   v_ofdm_chan((((i-1)*(N+Ng))+1):(i*(N+Ng)))=filter(v_canal,1,v_ofdm_seq((((i-
1)*(N+Ng))+1):(i*(N+Ng))))+v_ruido((((i-1)*(N+Ng))+1):(i*(N+Ng)));
end

%RECEPTOR
m_rx_ofdm=reshape(v_ofdm_chan,N+Ng,n_max_sim);
%Extracción del Prefijo Cíclico
m_rx_ofdm=extrac_pc(m_rx_ofdm,N,Ng,n_max_sim);
m_rx=1/sqrt(N)*fft(m_rx_ofdm);
%Matriz de ecualizacion
m_eq=repmat(v_eq(:),1,size(m_rx,2));
m_rx=m_rx./m_eq;
v_rx=m_rx(:);
%Demapeador
v_bits_rx=demapeador(v_rx,n_cod_tipo);
%Deinterleaver
v_bits_rx_deinter=deinterleaver(v_bits_rx,n_max_sim,N,n_cod_tipo);
%Desaleatorizador
%v_bits_rx_dalea=desaleatorizador(v_bits_rx_deinter);
%Cálculo de la ber
n_ber=mean(v_bits_rx_deinter(:)~=v_bits(:));
```

```
function [m_tx_ofdm_guard]=inser_pc(m_tx_ofdm,N,Ng,n_max_sym)

%inserto el prefijo cíclico
m_tx_ofdm_guard(1:Ng,1:n_max_sym)=m_tx_ofdm(N-Ng+1:N,1:n_max_sym);
%y la matriz de transmisión
m_tx_ofdm_guard(Ng+1:N+Ng,1:n_max_sym)=m_tx_ofdm(1:N,1:n_max_sym);

function [r_tx_ofdm_sinpc]=extrac_pc(m_rx_ofdm,N,Ng,n_max_sym)

%extraigo el prefijo cíclico
r_tx_ofdm_sinpc(1:N,1:n_max_sym)=m_rx_ofdm(Ng+1:N+Ng,1:n_max_sym);
```

C.1. Función correspondiente a Zero Padding

```
function [m_tx_ofdm_guard]=inser_ceros(m_tx_ofdm,N,Ng,n_max_sym)

%inserto el intervalo de guarda todo-cero
m_tx_ofdm_guard(1:Ng,1:n_max_sym)=0;
%y la matriz de transmisión
m_tx_ofdm_guard(Ng+1:N+Ng,1:n_max_sym)=m_tx_ofdm(1:N,1:n_max_sym);
```