```
%%%%%%%%%%%%%%%%%%%%%%
function Codifica_Lena
%%%%%%%%%%%%%%%%%%%%%%
% Codifica a Lena (de 512x512 pixels con 8 bits por pixel) a una tasa
% reducida de [8 5 4 1 0.5 0.25 0.125 0.0625] bits por pixel usando
% codificación subbanda mediante filtros de espejo en cuadratura con
% reconstrucción perfecta
%
% Marco A. Alzate, Universidad Distrital, mayo de 2017
%
    [h0,h1,f0,f1] = DisenaFiltros;       % Diseña el banco de filtros
    x = double(imread('lena512.bmp'));   % Lee la imagen
    x = normal(double(x));               % Convierte a formato double
    [xa1{1},xa1{2},xa1{3},xa1{4}] = Analisis(x,h0,h1); % Calcula 4 bandas
    k = 0;                               % y descompone cada una de ellas
    for i=1:4                            % en cuatro bandas
        [xa2{k+1},xa2{k+2},xa2{k+3},xa2{k+4}] = Analisis(xa1{i},h0,h1);
        k = k+4;
    end
    % Despliega las dieciseis bandas resultantes junto con la imagen original
    y = [normal(xa2{1})  normal(xa2{2})   normal(xa2{5})   normal(xa2{6}); ...
         normal(xa2{3})  normal(xa2{4})   normal(xa2{7})   normal(xa2{8}); ...
         normal(xa2{9})  normal(xa2{10})  normal(xa2{13})  normal(xa2{14}); ...
         normal(xa2{11}) normal(xa2{12})  normal(xa2{15})  normal(xa2{16})];
    cm = [(0:255)' (0:255)' (0:255)']/255; % Mapa de colores (256 niveles de gris
    subplot(221); image(x); colormap(cm); title('original')
    subplot(222); image(y); colormap(cm); title('16 bandas')
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Sintesis con reconstruccion perfecta (sin cuantificación) %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    k = 0;                               % Reconstruye cada una de las cuatro
    for i=1:4                            % bandas originales
        xa1{i} = Sintesis(xa2{k+1},xa2{k+2},xa2{k+3},xa2{k+4},f0,f1);
        k=k+4;
    end                                  % Y con ella reconstruye la señal entera
    y = Sintesis(xa1{1},xa1{2},xa1{3},xa1{4},f0,f1);
    subplot(223); image(y); colormap(cm); title('sin cuantización')
    rms = sqrt(sum(sum((x - y).^2))/prod(size(x)))/2.55;
    display(['         Sin cuantización el rms del error es ' num2str(rms,'%3.15f') '%'])
    for i=1:16                           % Calcula la energá de cada banda
        E(i) = sum(sum(xa2{i}.^2));
    end
    subplot(224); bar(10*log10(E)); title('Energia de cada banda, en dB')
    plt=1;
    for NB = 16*[8 5 4 1 0.5 0.25 0.125 0.0625]  % Numero total de bits por asignar
        e = E;                                   % entre las bandas
        nb = zeros(16,1);
        for b = 1:NB
            [me,i] = max(e);
```

```
            nb(i) = nb(i)+1;                        % Un bit más para esta banda
            e(i) = e(i)/2;
        end
        for i=1:16                                  % Cuantifica cada banda con el
            if nb(i)==0                             % número asignado de bits
                xa2{i} = zeros(size(xa2{i}));
            else
                delta = (max(max(xa2{i})) - min(min(xa2{i})))/2^nb(i);
                xa2{i} = delta*floor(0.5 + xa2{i}/delta);
            end
        end                                         % Y reconstruye la señal con las
        k = 0;                                      % bandas cunatizadas
        for i=1:4
            xa1{i} = Sintesis(xa2{k+1},xa2{k+2},xa2{k+3},xa2{k+4},f0,f1);
            k=k+4;
        end
        y = Sintesis(xa1{1},xa1{2},xa1{3},xa1{4},f0,f1); % Señal con NB/16 bpp
        rms = sqrt(sum(sum((x - y).^2))/prod(size(x)))/2.55;
        display(['Cuantizando a ' num2str(NB/16,'%1.4f') ' bpp, el rms del error es ' num2str(rms,'%3.15f') '%'])
        if plt==1
            figure
        end
        subplot(2,2,plt); image(y); colormap(cm); title([num2str(NB/16) 'bpp'])
        plt = plt+1;
        if plt>4
            plt=1;
        end
    end
return

%%%%%%%%%%%%%%%%%%%%%%
function x = normal(x)
%%%%%%%%%%%%%%%%%%%%%%
% Normaliza la amplitud de x entre 0 y 255 para recorrer todos los niveles
% de gris
    x = floor(255*(x - min(min(x)))/(max(max(x)) - min(min(x))) + 0.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x00,x01,x10,x11] = Analisis(x,h0,h1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Descompone la imagen x en cuatro bandas LL, LH, HL y HH
    lf = length(h0);                                % Número de taps en el filtro
    sx = size(x);                                   % Tamano de la imagen
    ultimo  = sx+lf-1;                              % Longitud de la señal filtrada
    y   = extiendefilas(x,lf-1);                    % Extiende las filas según el filtro
    z   = conv2(y,h0(:)','valid');                  % Muestras válidas de la convolución bidimensional
    x00 = Filtra_Submuestrea(z,h0,lf-1,ultimo);     % Componente LL
    x01 = Filtra_Submuestrea(z,h1,lf-1,ultimo);     % Componente LH
    z   = conv2(y,h1(:)','valid');                  % Muestras válidas de la convolución bidimensional
```

```
    x10 = Filtra_Submuestrea(z,h0,lf-1,ultimo);  % Componente HL
    x11 = Filtra_Submuestrea(z,h1,lf-1,ultimo);  % Componente HH

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = Filtra_Submuestrea(x,f,n,ultimo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    y = x(:,2:2:ultimo(2));                     % Submuestrea las columnas
    y = extiendecolumnas(y,n);                  % Extiende las columnas segun el filtro
    y = conv2(y',f(:)','valid'); y = y';        % Muestras válidas de la convolución bidimensional
    y = y(2:2:ultimo(1),:);                     % Submuestrea las filas

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = extiendefilas(x,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    y = fliplr(x(:,1:n));                        % Extensión circular en espejo
    y = [y x];
    y = [y fliplr(x(:,(end-n+1):end))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = extiendecolumnas(x,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    y = flipud(x(1:n,:));                       % Extensión circular en espejo
    y = [y; x];
    y = [y; flipud(x((end-n+1):end,:))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x = Sintesis(x00,x01,x10,x11,f0,f1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reconstruye una imagen a partir de cuatro bandas
    x = Supermuestrea_Filtra(x00,f0,f0)+ ... % Componente LL
        Supermuestrea_Filtra(x01,f1,f0)+ ... % Componente LH
        Supermuestrea_Filtra(x10,f0,f1)+ ... % Componente HL
        Supermuestrea_Filtra(x11,f1,f1);     % Componente HH

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = Supermuestrea_Filtra(x,fa,fb)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Devuelve la porción central de la interpolación de la matriz x
% (supermuestrea y filtra) usando fa para las filas y fb para
% las columnas.
    sx = size(x);
    lf = length(fa);
    z  = zeros(2*sx(1)-1,sx(2)); z(1:2:end,:) = x;
    y  = conv2(z',fa(:)','full'); y = y';
    sy = size(y); z = zeros(sy(1),2*sy(2)-1); z(:,1:2:end) = y;
    y  = conv2(z,fb(:)','full');
    y = y(lf-1:2*sx,lf-1:2*sx);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [h0,h1,f0,f1] = DisenaFiltros
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [N, Fo, Ao, W] = firpmord([0.44 0.56],[1 0],[0.002 0.002],2); % Parametros de diseño del filtro
    L=ceil((N-2)/4); N=4*L+2;                                     % Ajusta a un orden adecuado
    h = firpm(N,Fo,Ao,W);                                         % Filtro Prototipo H(z)
    h(2:2:N)=zeros(size(h(2:2:N))); h(2*L+2)=1/2;                 % Asegura la media banda
    h = (h + fliplr(h))/2;                                        % Asegura fase cero
    H = abs(fft(h,16384)); d = 1.01*max(H(ceil(0.57*8192):8191)); % Encuentra el verdadero rizado
    N = 2*L+1;                                                    % Orden del filtro h0
    h(N+1)=0.5 + d;                                               % Asegura respuesta no negativa
    zh=roots(h);                                                  % Raices de h(1)*z^2N + ... + h(2N+1)
    z0=zh(find(abs(zh)<1));                                       % Ceros de fase mínima (dentro del circulo unitario)
    h0=1;                                                         % Expandimos el producto como polinomio de z^-1.
    for i=1:N, h0=conv(h0,[-z0(i)  1]); end                       % Prod(1-zk*z^-1)
    h0=real(h0);                                                  % Algunos componentes imaginarios del orden 10^-16
    h0=sqrt(2)*h0/sum(h0);                                        % Normaliza para ganancia sqrt(2) en dc
    f0=fliplr(h0);                                                % Pasabajos de síntesis
    h1=((-1).^(0:N)).*f0;                                         % Pasaaltos de análisis
    f1=((-1).^(1:(N+1))).*h0;                                     % Pasaaltos de síntesis
```

```
% Construye la función de escala y la wavelet de Daubechies a partir
% de los filtros QMF de reconstrucción perfecta de cuatro taps
% diseñados en clase
%
% Marco A. Alzate, Universidad Distrital, mayo de 2017
%
h0 = [(1-sqrt(3)); (3-sqrt(3)); (3+sqrt(3)); (1+sqrt(3))]/4; % Pasabajos de análisis
f0 = flipud(h0);                                            % Pasabajos de síntesis
f1 = ((-1).^(0:3)').*h0;                                    % Pasaaltos de síntesis
phi = f0;                                                   % Aproximación a la función de escala
psi = f1;                                                   % Aproximación a la wavelet
for i=1:20                                                  % 20 iteraciones
    disp([length(phi) length(psi)])                         % Longitud de las aproximaciones
    if i<17                                                 % Grafica las primeras 16 aproximaciones
        phi = [0; phi; 0]; psi = [0; psi; 0];               % Unidades de tiempo: Muestras...
        k = 3*(0:length(phi)-1)'/(length(phi)-1);           % ... de la primera aproximación
        subplot(4,4,i); plot(k,[phi psi]); axis tight
        phi = phi(2:end-1); psi = psi(2:end-1);
    end
    hu = zeros(2*length(phi),1);                            % Interpolación
    hu(1:2:end) = phi;
    phi = conv(f0,hu);                                      % Filtro
    hu = zeros(2*length(psi),1);                            % Interpolación
    hu(1:2:end) = psi;
    psi = conv(f0,hu);                                      % Filtro
end

figure                                                      % La wavelet de Daubechies
psi = [0; psi; 0];                                          % es un fractal
k = 3*(0:length(psi)-1)'/(length(psi)-1);
subplot(221); plot(k,psi); axis([0 3 -1.5 2])
subplot(222); plot(k,psi); axis([2.12 2.57 -0.05 0.25])
subplot(223); plot(k,psi); axis([2.46 2.57 -0.002 0.009])
subplot(224); plot(k,psi); axis([2.542 2.57 -0.00005 0.0003])
```