Capítulo 3. Conversión Analógica Digital

1. Objetivos

- Convertir una señal analógica a una señal digital.
- Observar algunos aspectos importantes de esta conversión como el muestreo, la cuantización, el código binario utilizado y la reconversión a señal analógica.

2. Introducción

Muchas señales de interés como la voz, biológicas o sísmicas son de esencia analógica, sin embargo, a veces es necesario poderlas convertir a señales digitales para procesarlas o almacenarlas. El tema general de la compresión de datos, de el cual la conversión analógica-digital es un caso especial se puede dividir en dos ramas principales:

- Cuantización, en el cual la fuente analógica es cuantizada en un número finito de niveles. En este proceso ocurre una distorsión inevitable, con lo cual hay pérdida de información. Ésta pérdida de información no puede ser recuperada.
- Comprensión sin pérdidas, en la cual un dato digital (normalmente resultado de una cuantización) es comprimido, representándolo con el menor número de bits posibles. La secuencia de datos originales pueden ser perfectamente recuperadas a partir de la secuencia comprimida.

3. Medida de la Información

3.1. Información y Entropía

La Teoría de la Información es la disciplina que se encarga del estudio y cuantificación de los procesos que se realizan sobre la información. La cantidad de información que nos proporciona cierto dato es menor cuanto más esperamos ese dato. La idea de la probabilidad de ocurrencia de cierto evento que nos da información puede modelarse por una variable aleatoria y su conjunto de mensajes y probabilidades asociadas. Al recibir un mensaje de esa variable aleatoria (fuente de información), se obtiene una cantidad de información, que depende sólo de la probabilidad de emisión de ese mensaje. La cantidad de información de una función creciente e inversa a la probabilidad, es decir, un mensaje con menor probabilidad proporciona mayor cantidad de información. Claude Shannon propuso una medida para la información. Si se supone una fuente X de información de n mensajes X_i, cada uno con probabilidad $P(x_i)$, la información al recibir un mensaje esta dado por la ecuación:

$$I(X = X_i) = -\log(P(X_i)) \tag{3.3.1}$$

La medida de la información o entropía de la fuente X esta dada por la esperanza matemática de la información de cada símbolo:

$$H(X) = -\sum_{i=1}^{n} p_i * I(X = X_i) = -\sum_{i=1}^{n} p_i * \log(p_i) \tag{3.3.2}$$

Si utilizamos logaritmo en base 2, la medida estará en bits. Si utilizamos neperianos, obtendremos nats. Pero lo que importa es que esta medida nos da una cota superior de la cota de compresión. No se puede inventar ninguna codificación que consiga una longitud en bits media por símbolo emitido menor que la entropía de la fuente sobre la que se realiza la codificación. Esto es, sin embargo, una cota teórica. Los compresores actuales no llegan a esta cota pero se quedan muy cerca.

3.2. Compresión sin pérdidas

La comprensión sin pérdidas es el término general utilizado para todos los esquemas que reducen el número de bits requeridos para la representación de una fuente, y que permite su perfecta recuperación.

Existen tipos de compresores que utilizan las propiedades estadísticas de la fuente de información para mejorar la codificación (el conjunto de símbolos de salida asociados a cada mensaje emitido por la fuente) de los mensajes de la fuente. Estos son los compresores estadísticos.

Este tipo de compresores parten de:

- Una Fuente de Información de n mensajes
- Las probabilidades de aparición de cada mensaje de la fuente (que pueden ser extraídas a priori de forma experimental o pueden ser dadas y fijas.
- Un alfabeto de salida que consta de una serie de símbolos (por ejemplo , el alfabeto binario consta de los símbolos 0 y 1),

Su objetivo es asignar una codificación para los mensajes de la fuente. Una codificación es una función que a cada mensaje de la fuente asigna una cadena de símbolos del alfabeto de salida. Esta codificación debe ser tal que explote la redundancia en la información dada por la fuente para producir compresión.

3.2.1. Codificación de Huffman

En la codificación de Huffman asignamos las palabras de código más largas a las salidas menos probables de la fuente y palabras de código más cortas a las más probables. Para hacer esto, comenzamos combinando las dos salidas menos probables de la fuente para generar una nueva salida combinada cuya probabilidad sea la suma de las probabilidades correspondientes. Se repite el proceso hasta que se deja solamente una salida combinada. De esta manera generamos un árbol. Comenzando por la raíz del árbol y asignado ceros y unos a las ramas que emergen del mismo nodo, generamos el código. El algoritmo será el siguiente:

1. **Ordenamos los símbolos de la fuente en orden decreciente de probabilidades.**
2. **Combinamos las dos salidas menos probables en una única salida cuya probabilidad es las suma de las correspondientes probabilidades.**
3. **Si el número de salidas que queda es 2, vamos al paso siguiente; en otro caso, volvemos al paso 1.**
4. **Asignamos arbitrariamente un 0 y un 1 a las dos salidas que nos queden.**
5. **Si una salida es el resultado de la combinación de dos salidas en un paso anterior, añadimos a la palabra de código correspondiente un 0 y un 1 para obtener las palabras de código de las salidas precedentes y repetimos el paso 5. Si ninguna salida está precedida por ninguna otra, entonces paramos.**

Figura 3.1. Código de Huffman

Ejercicio 3.1.

Enunciado

a) *Realice una función en MATLAB que calcule la probabilidad de que aparezcan diferentes caracteres en una secuencia x.*

```
%  uso  = [prob,chars]=lectura(x)
%
% donde    x      =  secuencia de entrada
%          [prob,chars]  =  chars es una cadena que guarda los caracteres que
%                           aparecen en la secuencia x y prob guarda la
frecuencia
%                           con la que aparece
```

b) *Escriba una función que, dado un vector de probabilidad, calcule la entropía.*

c) *Escriba una función que, genere el código de Huffman y la longitud media del código.*

d) *Genere un script en MATLAB que dada una secuencia, calcule la entropía de la fuente, la longitud media de las palabras código y su código correspondiente.*

Resultado

a)

```
function [prob,chars]=lectura(x)
long=length(x);
chars(1)=x(1);
busca=findstr(x,chars(1));
prob(1)=(length(busca))/long;
j=2;
for i=2:long
    caracter=x(i);
    busca=findstr(chars,caracter);
    if(length(busca)==0)
       busca2=findstr(x,caracter);
       chars(j)=caracter;
       prob(j)=(length(busca2))/long;
       j=j+1;
    end
end
```

b)

```
function h=entropia(p)

h=sum(-p.*log2(p));
```

c)

```
function [h,l]=huffman(p);
n=length(p);
q=p;
m=zeros(n-1,n);
for i=1:n-1
  [q,l]=sort(q);
  m(i,:)=[l(1:n-i+1),zeros(1,i-1)];
  q=[q(1)+q(2),q(3:n),1];
end
for i=1:n-1
  c(i,:)=blanks(n*n);
end
c(n-1,n)='0';
c(n-1,2*n)='1';
for i=2:n-1
  c(n-i,1:n-1)=c(n-i+1,n*(find(m(n-i+1,:)==1))...
  -(n-2):n*(find(m(n-i+1,:)==1)));
  c(n-i,n)='0';
  c(n-i,n+1:2*n-1)=c(n-i,1:n-1);
  c(n-i,2*n)='1';
  for j=1:i-1
    c(n-i,(j+1)*n+1:(j+2)*n)=c(n-i+1,...
    n*(find(m(n-i+1,:)==j+1)-1)+1:n*find(m(n-i+1,:)==j+1));
  end
end
for i=1:n
  h(i,1:n)=c(1,n*(find(m(1,:)==i)-1)+1:find(m(1,:)==i)*n);
  l1(i)=length(find(abs(h(i,:))~=32));
end
l=sum(p.*l1);
```

d)

```
x='laboratorio de transmision de datos. leticia gestoso
barbero';
[prob,chars]=lectura(x);
e=entropia(prob);
[h,l]=huffman(prob);
[M,N]=size(h);
for i=1:length(prob)
    fprintf('Caracter = %c, probabilidad = %5f
\n',chars(i),prob(i));
    fprintf('Palabra codigo =');
    for j=1:N
    fprintf('%c',h(i,j));
    end
    fprintf('\n');
    %fprintf('Palabra de código asociada = %c%c%c%c',h(i,:));
end
fprintf('\n Entropia = %d',e);
fprintf('\n Longitud Media del código = %5f \n',l);
```

```
Caracter = l, probabilidad = 0.033333
Palabra codigo =           10000
Caracter = a, probabilidad = 0.100000
Palabra codigo =             010
Caracter = b, probabilidad = 0.050000
Palabra codigo =            0010
Caracter = o, probabilidad = 0.133333
Palabra codigo =             101
Caracter = r, probabilidad = 0.083333
Palabra codigo =            1110
Caracter = t, probabilidad = 0.083333
Palabra codigo =            1111
Caracter = i, probabilidad = 0.083333
Palabra codigo =            1100
Caracter =  , probabilidad = 0.116667
Palabra codigo =             011
Caracter = d, probabilidad = 0.050000
Palabra codigo =            0011
Caracter = e, probabilidad = 0.083333
Palabra codigo =            1101
Caracter = n, probabilidad = 0.033333
Palabra codigo =           10001
Caracter = s, probabilidad = 0.083333
Palabra codigo =             000
Caracter = m, probabilidad = 0.016667
Palabra codigo =          100110
Caracter = ., probabilidad = 0.016667
Palabra codigo =          100111
Caracter = c, probabilidad = 0.016667
Palabra codigo =          100100
Caracter = g, probabilidad = 0.016667
Palabra codigo =          100101

 Entropia = 3.728237e+000
 Longitud Media del código = 3.766667
```

4. Cuantización

Una fuente analógica emite una señal mensaje *x(t)* que podemos interpretar como una función muestra de un proceso aleatorio *X(t)* Cuando *X(t)* es un proceso aleatorio estacionario, limitado en banda, el teorema de muestreo nos permite representarlo mediante sus muestras $X(nT_s)$ tomadas a la frecuencia de Nyquist; esto es, mediante una secuencia de variables aleatorias: un proceso aleatorio discreto.

En general, después del muestreo las muestras tomarán valores en un rango continuo, que deberán aproximarse en el bloque de codificación de fuentes a un conjunto numerable y finito de valores, mediante un proceso que denominamos cuantización. Si el L el número de estos valores o niveles, mediante un simple esquema de codificación los representaremos a través de una secuencia de R dígitos binarios, con $R = \lfloor \log_2 L \rfloor + 1$, si L no es una potencia de 2 ó $\log_2 L$ en caso contrario. Este proceso origina una comprensión de los datos, pero también introduce una distorsión o pérdida de la fidelidad de la señal reproducida.

Figura 3.2. Cuantización

En general, los esquemas de cuantización pueden ser clasificados en cuantización escalar, cuantizan individualmente cada muestra (salida de la fuente) y cuantización vectorial, se cuantizan bloques de muestras, este último esquema no será estudiado aquí.

4.1. Cuantización Escalar

Un cuantizador escalar divide el rango de la variable aleatoria de entrada en un conjunto de L intervalos denominados *regiones de decisión* R_i, i=1, 2, ..., L. Siempre que la señal de entrada pertenezca a una región R_i se le asigna un *nivel de representación* o *nivel de cuantización*, $\hat{x}_i$

$$x \in R_i \Leftrightarrow Q(x) = \hat{x}_i \qquad (3.4.1)$$

Obviamente, una cuantización de este tipo introduce un error cuadrático medio $(x - \hat{x}_i)^2$. El error cuadrático medio está dado por:

$$D = \sum_{i=1}^{N} \int_{R_i} (x - \hat{x}_i)^2 f_X(x) dx \qquad (3.4.2)$$

Donde $f_X(x)$ denota la función de densidad de probabilidad de la variable aleatoria de la fuente. La razón señal a ruido de cuantización se define de la siguiente manera:

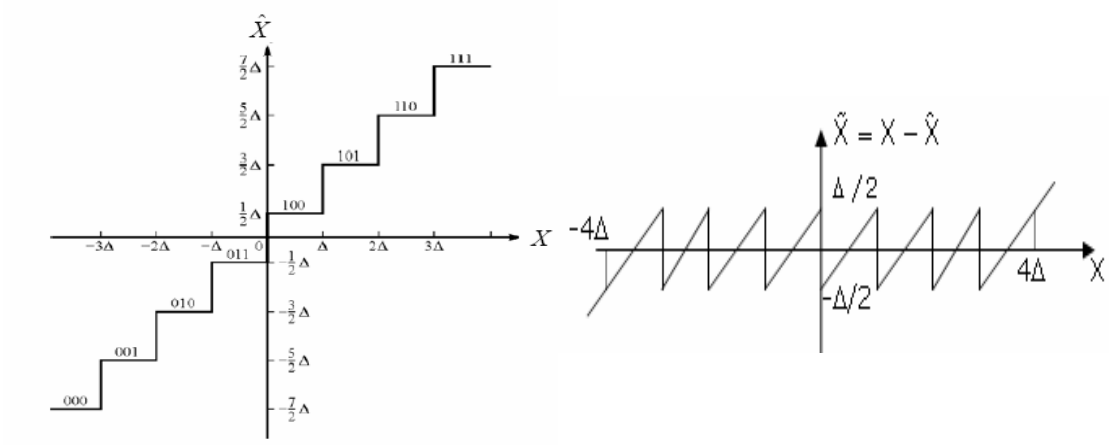
$$SQNR = 10\log_{10}\frac{E[X^2]}{D} \qquad (3.4.3)$$

4.1.1. Cuantización Uniforme

En una cuantización uniforme todas las regiones de cuantización excepto la primera (R_1) y la última (R_L) se toman de igual longitud, Δ, que denominamos *paso de cuantización.*

Existen dos tipos de cuantizadores uniformes:

- MId-rise
  - Numero de Niveles de representación (L) par
  - El cero no es un nivel de representación



- Mid-tread
  - Número de Niveles de representación (L) impar
  - El cero es un nivel de representación

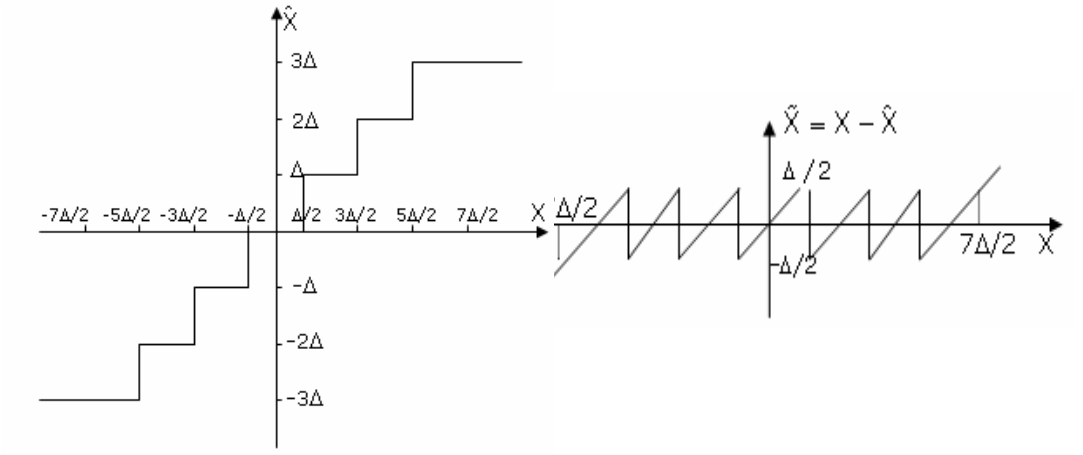


Figura 3.4. Cuantizador Mid-tread y Correspondiente Error de Cuantización

Ejercicio 3.2.

Enunciado

a) Realizar una función en Matlab 'cuantiza.m' que cuantice una señal y cuyo uso es el siguiente.

%y=cuantiza(x,levelmin,levelmax,L)
%donde:
%x = señal a cuantizar
%levelmax y levelmin = niveles del cuantizador (levelmax= $\hat{x}_L$) (levelmin= $\hat{x}_1$)
%L = niveles del cuantizador

b) Realizar un script en matlab llamado 'ejercicio3_2b.m' que utilice la función diseñada en el apartado anterior y que dibuje lo siguiente:

- *Figura 3.2. (relación entre niveles de decisión y niveles de representación; y error de cuantización) para un cuantizador midrise con L=8, levelmax=3.5 y levelmin=-3.5.*
- *Figura 3.3. (relación entre niveles de decisión y niveles de representación ; y error de cuantización) para un cuantizador midtread con L=9, levelmax=4 y levelmin=-4.*

*c) Realice un script en matlab llamado 'ejercicio3_1c.m' que represente 2 figuras en cada una de ellas se representarán 2 señales, la primera: x(t)=4*cos(2*pi*t) muestreada cada 0,05 sg y x(t) cuantizada (la primera cuantizada con un cuantizador midrise y la segunda con un cuantizador midtread) usando los parámetros dados en el apartado b).*

Resultado

a)

```
function y=cuantiza(x,level_min,level_max,L)
%paso de cuantizacion
qx=(level_max-level_min)/(L-1);
%niveles de representacion
levelxr=zeros(1,L);
for j=0:L-1
    levelxr(1,j+1)=level_min+(j*qx);
end
%niveles de decision
levelx=zeros(1,(L+1));
level_miny=0-((L/2)*qx);
for j=0:L
    levelx(1,j+1)=level_miny+(j*qx);
end
%cuantizamos
for i=1:length(x)
    for j=1:L
        if(x(i)>=levelx(j) & x(i)<levelx(j+1))
            y(i)=levelxr(j);
        elseif(x(i)>=levelx(L))
            y(i)=levelxr(L);
        elseif(x(i)<=levelx(1))
            y(i)=levelxr(1);
        end
    end
end
```

b)

```
t=[-5:0.001:5];
x=t;
y=cuantiza(x,-3.5,3.5,8);
figure(1);
subplot(2,1,1);
plot(t,y);
xlabel('x');
ylabel('Q(x)');
title('Cuantizador Uniforme Midrise con L=8');
grid on;
error=x-y;
subplot(2,1,2);
plot(x,error);
xlabel('x');
ylabel('error');
title('Error de cuantizacion');
grid on;
y2=cuantiza(x,-4,4,9);
figure(2);
subplot(2,1,1);
plot(t,y2);
xlabel('x');
ylabel('Q(x)');
title('Cuantizador Uniforme Midtread con L=9');
grid on;
error=x-y2;
subplot(2,1,2);
plot(x,error);
xlabel('x');
ylabel('error');
title('Error de cuantizacion');
grid on;
```

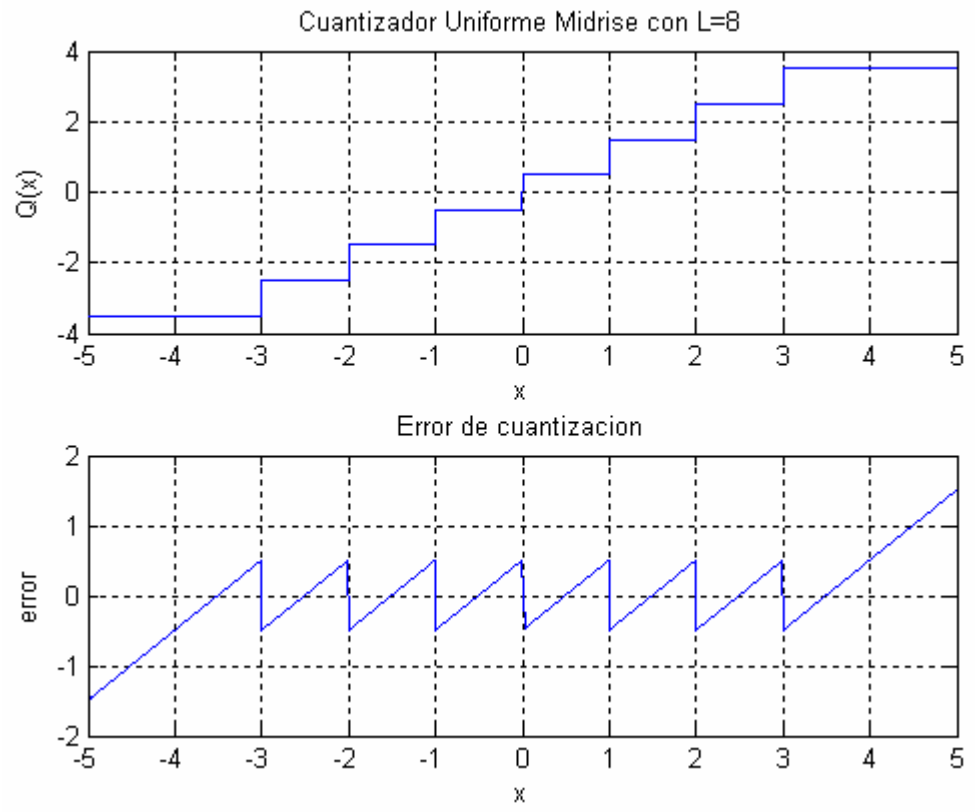


Figura 3.5. Cuantizador Uniforme Mid-rise

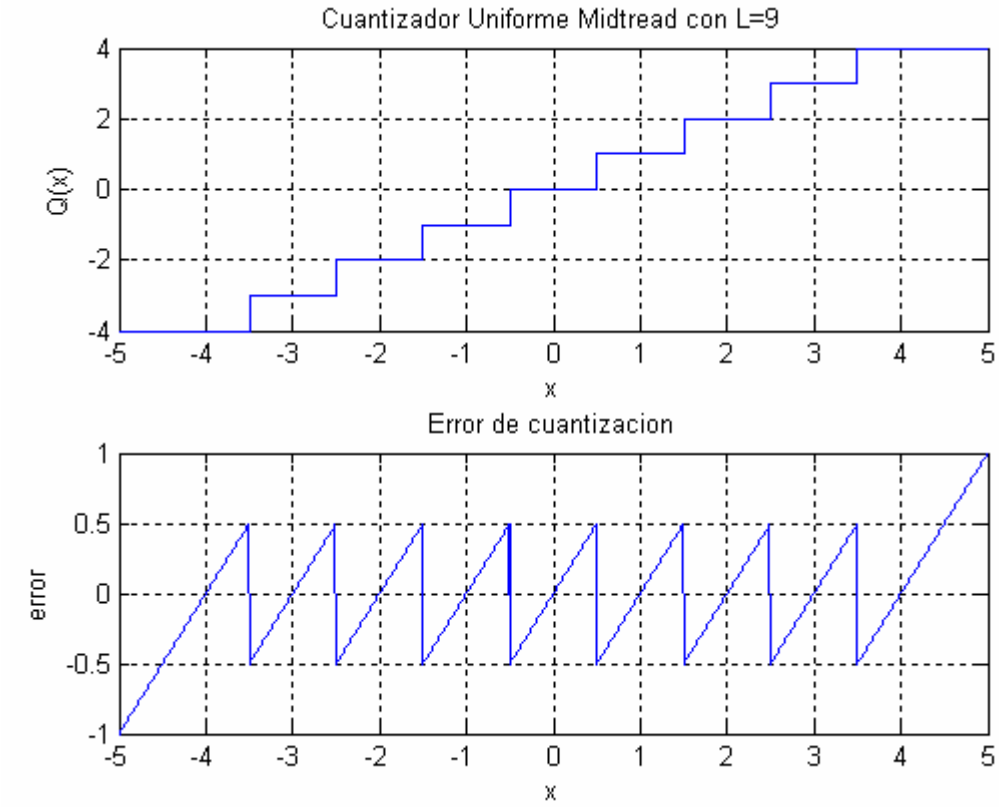


Figura 3.6. Cuantizador Uniforme Mid-tread

c

```
%Señal muestreada
t=[0:0.05:2];
x=4*cos(2*pi*t);

%Señal cuantizada
xr1=cuantiza(x,-3.5,3.5,8);
figure(1);
plot(t,x);
hold on;
plot(t,xr1,'rx');
hold off;
xlabel('t');
ylabel('x(t)');
title('senoide x(t)=4*cos(2*pi*t) cuantizada con L=8');
xr2=cuantiza(x,-4,4,9);
figure(1);
plot(t,x);
hold on;
plot(t,xr2,'rx');
hold off;
xlabel('t');
ylabel('x(t)');
title('senoide x(t)=4*cos(2*pi*t) cuantizada con L=9');
```

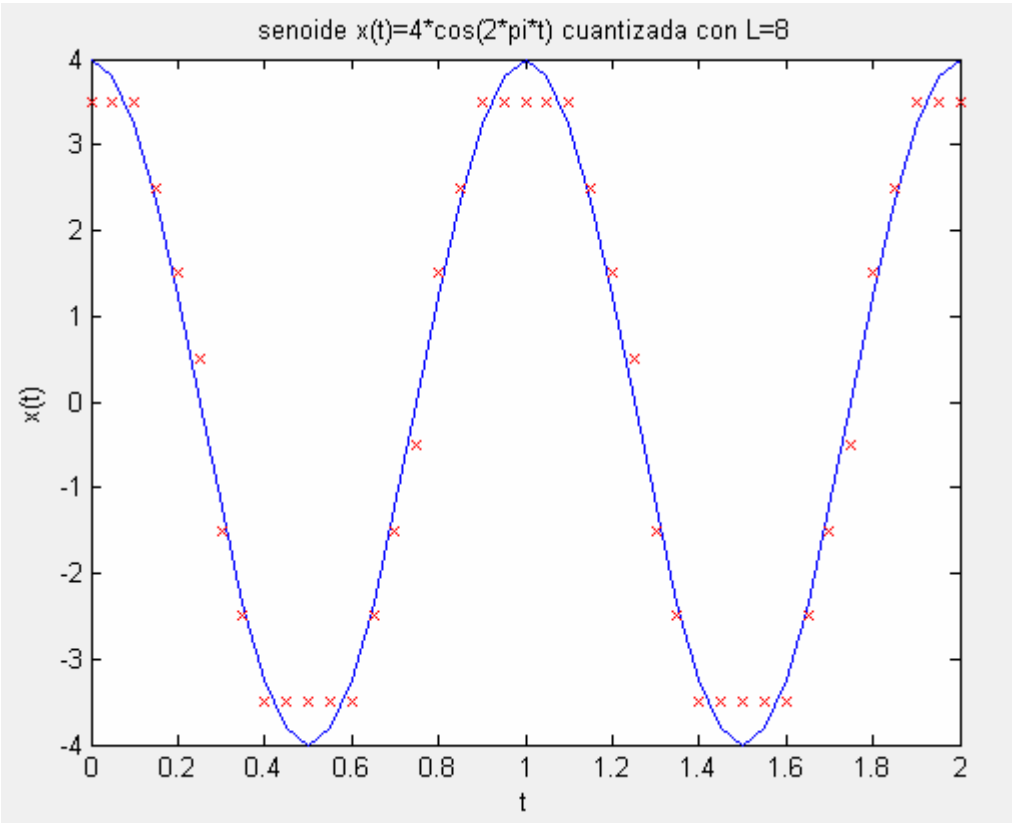


Figura 3.7. Coseno cuantizado con L=8

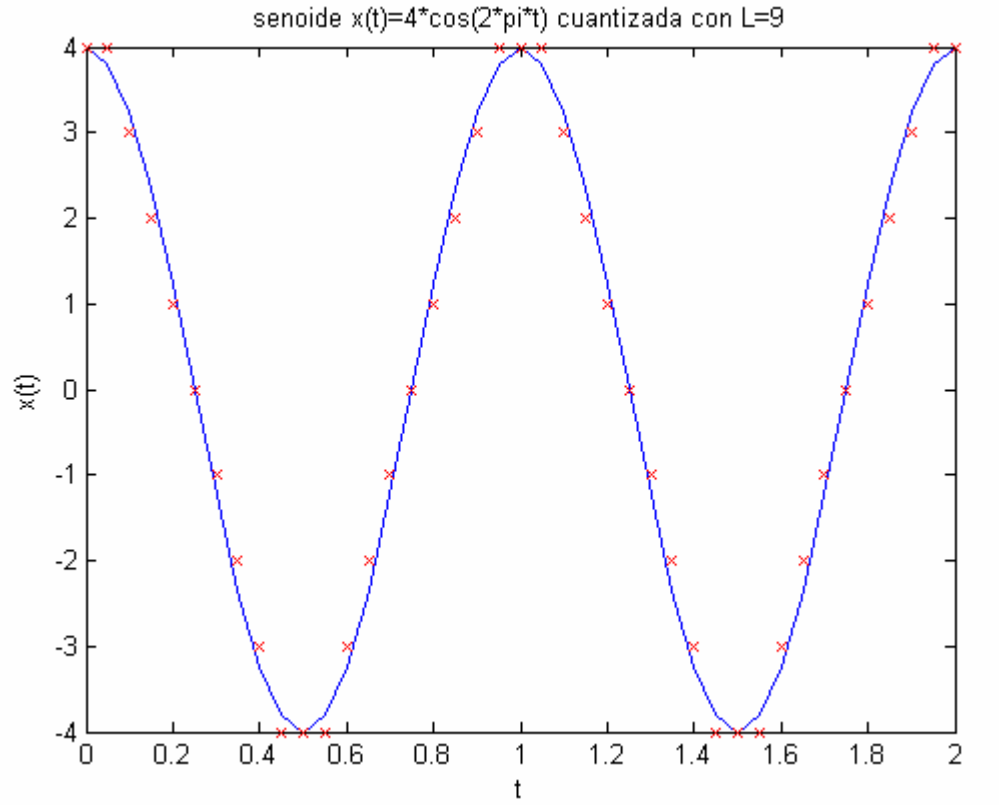


Figura 3.8. Coseno cuantizado con L=9

Ejercicio 3_3

Enunciado

Descargar el fichero 'fountainbw.tif'. La imagen fountainbw.tif es una imagen con escala de grises. Investigaremos que pasa cuando cuantizamos con menor número de bits/píxel. Cargar el fichero en Matlab y visualizarlo usando la siguiente secuencia de comandos.

```
>> y = imread('fountainbw.tif');
>> imshow(y)
```

La matriz imagen se inicializa al tipo uint8, así que necesitaremos convertir la matriz imagen a tipo double antes de la ejecución de cualquier cálculo. Para ello usa el comando z = double (y).

Usa la función anterior para realizar un script en MATLAB 'imagen_q.m' que use la función del ejercicio anterior (cuantiza.m) y que realice la cuantización de la imagen con L=256, 128, 32, 16 y 4.

Resultado

```
y=imread('fountainbw.tif');
z=double(y);
[M,N]=size(z);
L=[256 128 64 32 16 4];
level_max=max(reshape(z',1,M*N));
for j=1:6
    for i=1:1:M
        yc(i,:)=cuantiza(z(i,:),-level_max,level_max,L(j));
    end
    yr=uint8(yc);
    figure(j);
    title(['L=',num2str(L)]);
    imshow(yr);
    pause;
    yr=zeros(M,N);
end
```

Figura 3.9 (a)

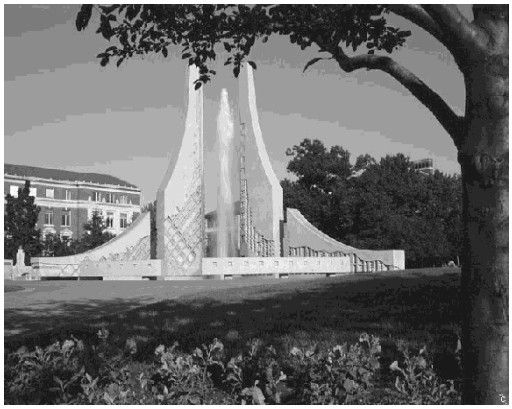


Figura 3.9 (b)

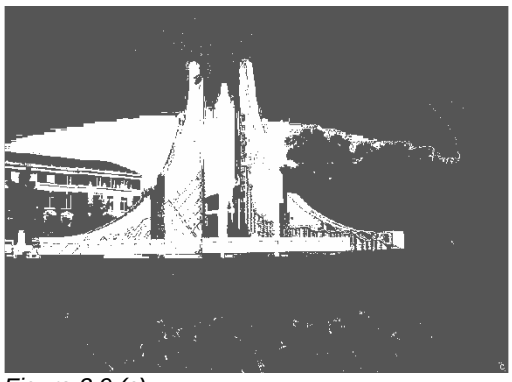


Figura 3.9 (c)

Figura 3.9. Imagen Cuantizada con L=256 (a), L=16 (b), L=4 (c)

De los resultados obtenidos podemos deducir que éste método no es el utlizado para cuantizar una imagen. Se suele utilizar métodos de cuantización vectorial para obtener resultados satisfactorios.

4.1.2. Cuantización **No Uniforme**

La solución son los llamados companders (de compresión-expansión). La idea es evitar el problema de un cuantizador no uniforme usando el siguiente rodeo.

- En primer lugar aplicamos una función no lineal c(x) a la variable original x.
- A continuación cuantizamos uniformemente la nueva variable c(x).
- Aplicando la función inversa c(x) a las muestras cuantizadas obtendremos las muestras correspondientes a un muestreo no uniforme de x.

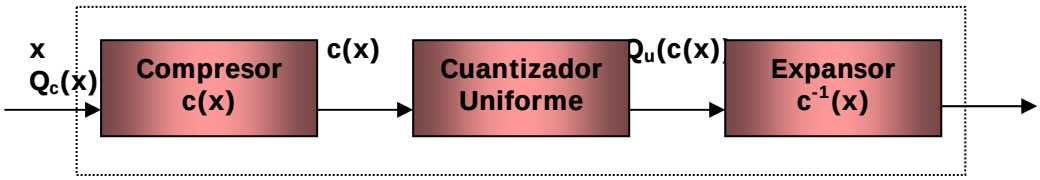


Figura 3.10. Cuantizador No uniforme

La figura adyacente lo explica gráficamente. Dada la no linealidad de c(x) intervalos equiespaciados (Q uniforme sobre el eje OY se traducen en saltos diferentes sobre el eje OX (Q no uniforme). El resultado global es equivalente a una cuantización no uniforme sobre la x original.

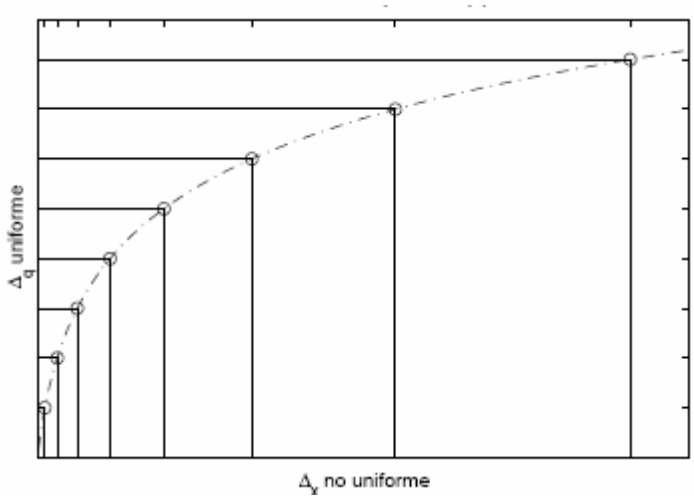


Figura 3.11. Función de compresión c(x)

Como la función principal de c(x) es comprimir valores de x en unas zonas y expandirlos en otras, a este método se le denomina compresión-expansión, o compander. Típicamente c(x) transforma el rango inicial de la variable x en el [-1,1] que posteriormente se cuantifica uniformemente.

5. Modulación por Codificación de Pulsos (PCM)

5.1. Introducción

Un sistema PCM es realmente un esquema de codificación de fuentes. En realidad hay que entenderlo como una codificación de pulsos modulados en amplitud, pulsos que surgen o bien del muestreo de una señal (proceso aleatorio) analógica limitada en banda o bien de la salida de una fuente discreta. Es decir, es una técnica de codificación de fuente y no una técnica de modulación digital.

Un transmisor PCM consta básicamente de tres secciones: un muestreador, un cuantizador y un codificador, tal y como se muestra la siguiente figura.

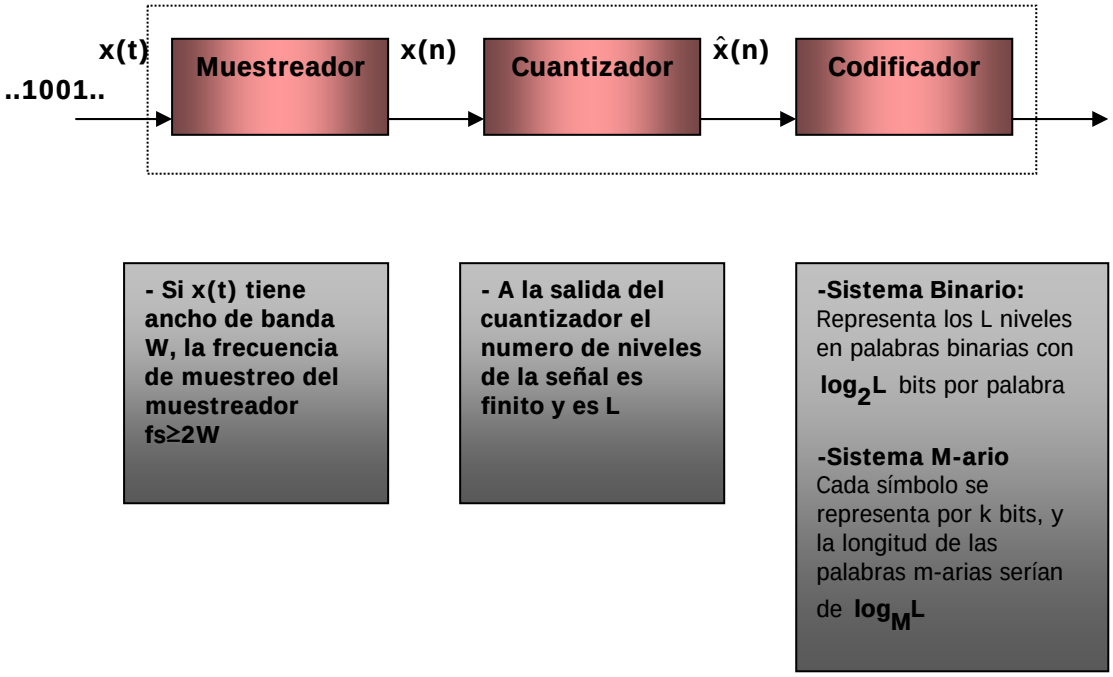


Figura 3.12. Sistema PCM

Una vez realizada la codificación necesitamos convertir la secuencia binaria de bits en formas de ondas eléctricas para poder transmitirlas por el canal de comunicación. Para ello el sistema PCM utiliza codificadores de línea.

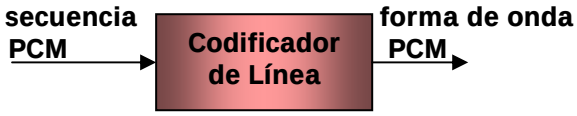


Figura 3.13. Codificador de Línea

5.2. PCM Uniforme

En un sistema PCM Uniforme el intervalo *[-Xmax,Xmax]* de longitud 2Xmax se divide en L subintervalos iguales cada uno de longitud:

$$\Delta = \frac{2X_{\max}}{L} \tag{3.5.1}$$

Si N es lo suficientemente grande, la función de densidad de la señal de entrada en cada subintervalo se puede suponer uniforme, resultando una distorsión:

$$D = E[X^2] = \frac{\Delta^2}{12} \tag{3.5.2}$$

Si se cumple $L = 2^R$, donde *R* son los bits requeridos para la representación de cada nivel. Esto implica que si el ancho de banda de la señal analógica es *W* y si muestreamos a la frecuencia de Nyquist, el ancho de banda requerido para la transmisión de la señal PCM es de al menos *RW*(en la práctica, *1.5*R*W*). La distorsión está dada por:

$$D = \frac{\Delta^2}{12} = \frac{X_{\max}{}^2}{3L^2} = \frac{X^2_{\max}}{3*4^R} \tag{3.5.3}$$

La relación señal a ruido:

$$(SNR)_0 = \frac{E[X^2]}{\Delta^2/12} = \frac{E[X^2]}{\frac{X^2_{\max}}{3*4^R}} = \frac{3*4^R*E[X^2]}{X^2_{\max}} \tag{3.5.4}$$

En decibelios:

$$SNR|_{dB} \approx 4.8 + 6R + P[X^2]|_{dB} \tag{3.5.5}$$

Después de la cuantización, los niveles cuantizados son codificados usando R bits para cada nivel de cuantización. El esquema de codificación normalmente empleado es el Código Binario Natural (CBN), en el cual el nivel más bajo se corresponde con una secuencia de todo ceros y el nivel mas alto con una secuencia todo unos.

Ejercicio 3_4

a) Realice una función en MATLAB pcm_u.m que modifique la función cuantiza.m, para que devuelva dos parámetros más: cod (secuencia codificada) y snr(relación señal a ruido en decibelios).

%[snr,cod,y]=pcm_u(x,levelmin,levelmax,L)
% función que realiza la codificación PCM uniforme de una %
% secuencia
%donde:
%
%x = señal a cuantizar
%levelmax y levelmin = niveles del cuantizador (levelmax= $\hat{x}_L$) (levelmin= $\hat{x}_1$)
%L = niveles del cuantizador
%y = secuencia cuantizada
%cod = secuencia codificada
%snr = relación señal a ruido a la salida del cuantificador

```
function [snr,cod,y]=pcm_u(x,level_min,level_max,L)

%Numero de bits necesitados para codificar cada nivel de
%cuantizacion
R=ceil(log2(L));

%paso de cuantizacion
qx=(level_max-level_min)/(L-1);
cod1=0*R;

%niveles de representacion
levelxr=zeros(1,L);
codea=zeros(length(x),1);
for j=0:L-1
    levelxr(1,j+1)=level_min+(j*qx);
    code(j+1,1)=cod1+(j*1);
end
codea=de2bi(code,R,'left-msb');

%niveles de decision
levelx=zeros(1,(L+1));
level_miny=0-((L/2)*qx);
for j=0:L
    levelx(1,j+1)=level_miny+(j*qx);
end

%cuantizamos
for i=1:length(x)
    for j=1:L
        if(x(i)>=levelx(j) & x(i)<levelx(j+1))
            y(i)=levelxr(j);
            cod(i,:)=codea(j,:);
        elseif(x(i)>=levelx(L))
            y(i)=levelxr(L);
            cod(i,:)=codea(L,:);
        elseif(x(i)<=levelx(1))
            y(i)=levelxr(1);
            cod(i,:)=codea(1,:)
        end
    end
end
snr=20*log10(norm(x)/norm(x-y));
```

b) Realizar un script en MATLAB 'ejercicio3_4b.m' que genere una señal senosoidal de amplitud 1 y frecuencia 1. Usando el esquema pcm (pcm_u.m) cuantizar la señal con 8 niveles y con 16. Dibujar la señal original con la cuantizada en los mismos ejes. Compara el resultado de las SNR's en los dos casos.

```
paso=0.05;
Tfinal=10;
t=[0:paso:Tfinal];
x=sin(t);

[snr16,cod16,y16]=pcm_u(x,-0.95,0.95,16);
[snr8,cod8,y8]=pcm_u(x,-0.9,0.9,8);


plot(t,y8,t,x);
title(['PCM Uniforme con L=8. SNR = ' num2str(snr8)]);
legend('señal cuantizada','señal original')
axis([0 10.01 -1.01 1.01])
grid on;

pause %Pulsa una tecla para ver resultado con L=16

plot(t,y16,t,x);
title(['PCM Uniforme con L=16. SNR = ' num2str(snr16)]);
legend('señal cuantizada','señal original')
axis([0 10.01 -1.01 1.01]);
grid on;
```

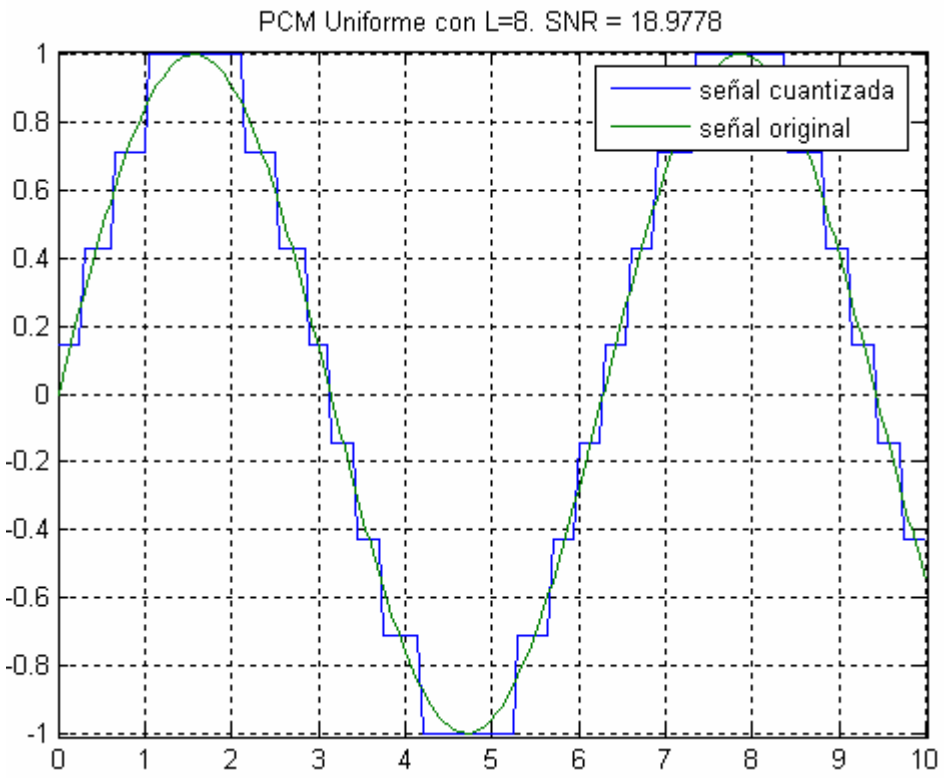


Figura 3.14. Coseno cuantizado siguiendo un esquema PCM Uniforme con L=8

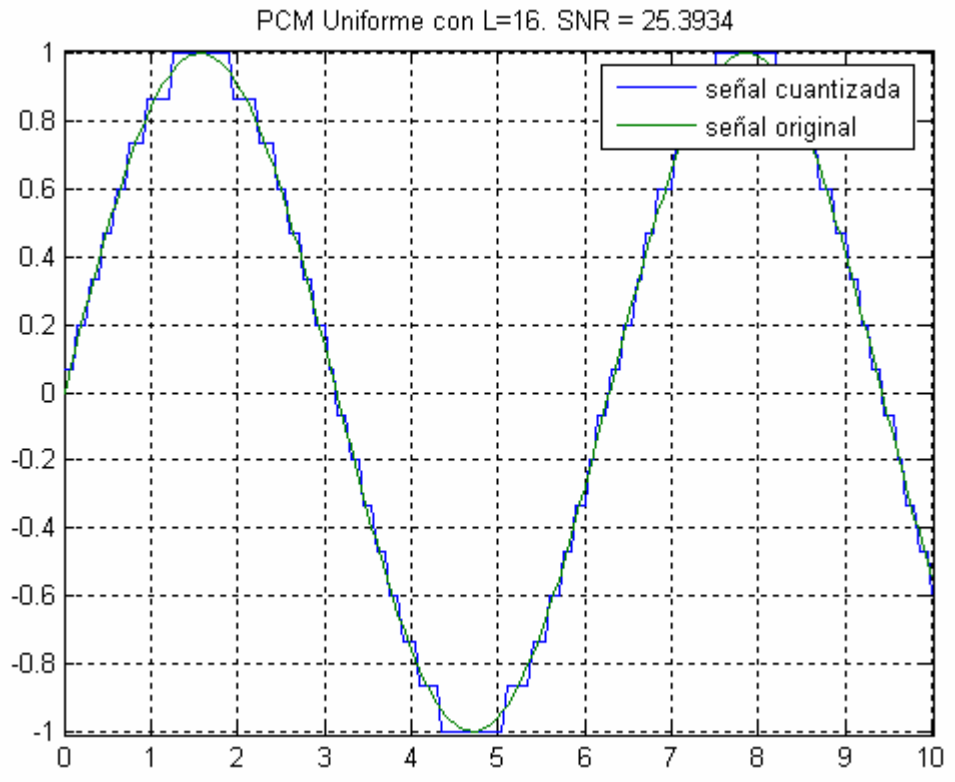


Figura 3.15. Coseno cuantizado siguiendo un esquema PCM Uniforme con L=16

c) Realizar un script en Matlab 'ejercicio3_4c.m' que dibuje en una misma 2 gráficas, la primera será la señal cuantizada anterior y la segunda la señal codificada para L=16

```
paso=0.05;
Tfinal=3;
t=[0:paso:Tfinal];
x=sin(t);
[snr16,cod16,y16]=pcm_u(x,-0.95,0.95,16);
[M,N]=size(cod16);
paso2=paso/N;
t2=[0:paso2:(Tfinal+((N-1)*paso2))];
y_pcm=(reshape(cod16',1,M*N));

subplot(2,1,1);
plot(t,y16,t,x);
title('Senoide cuantizada con L=16');
axis([0 3.01 -1.01 1.01]);
grid on;

subplot(2,1,2);
plot(t2,y_pcm);
title('Senoide codificada con L=16');
axis([0 3.01 -0.01 1.01]);
grid on;
```

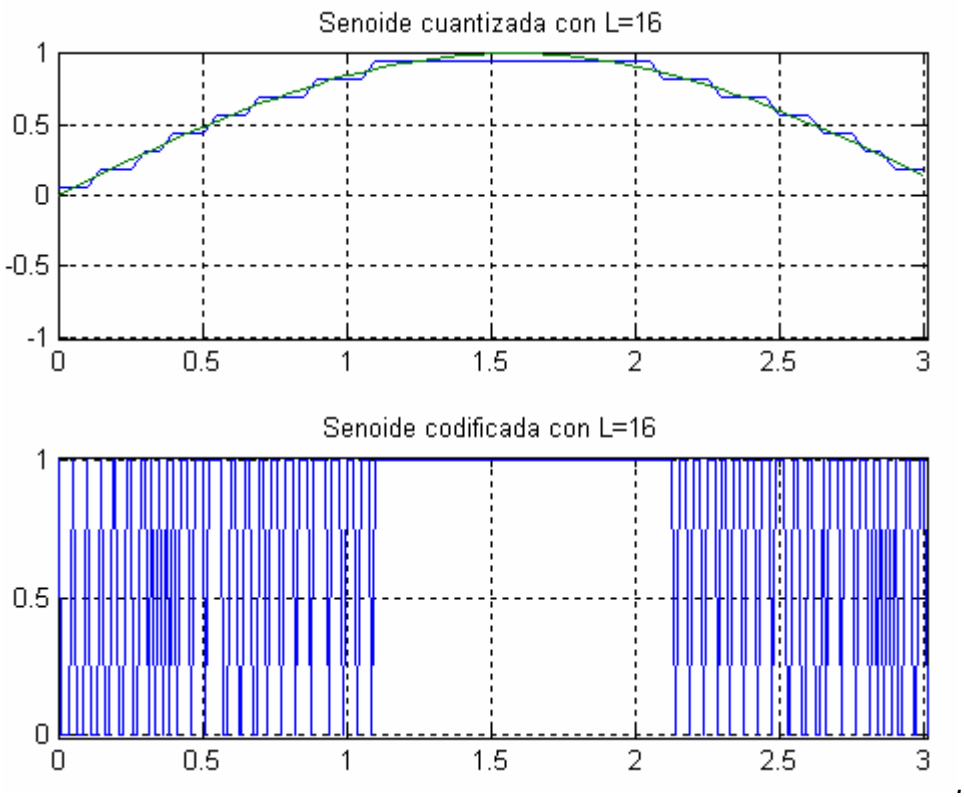


Figura 3.16. Cuantización y Codificación siguiendo un esquema PCM Uniforme con L=8

5.3. PCM No Uniforme

Los primeros sistemas PCM se implementaron con funciones logarítmicas suaves de compresión. Hoy en día, la mayoría de los sistemas PCM usan pedazos de aproximaciones lineales a la característica de compresión logarítmica. En América del norte se utiliza la siguiente característica de compresión llamada ley μ.

$$c(x) = \operatorname{sgn}(x)\frac{\log\left(1+\mu\left|x/x_{max}\right|\right)}{\log(1+\mu)} \quad 0 \le \frac{x}{x_{máx}} \le 1 \qquad (3.5.6)$$

Donde μ es el parámetro que controla la cantidad de compresión y en la ley estándar tiene un valor habitual de μ =255.

Otra característica de compresión, usada principalmente en Europa, es la ley A definida como:

$$\frac{c(x)}{x_{máx}} = \begin{cases} \dfrac{\dfrac{A|x|}{x_{máx}}}{1+\ln A}\,sg(x) & 0 \le \dfrac{|x|}{x_{máx}} \le \dfrac{1}{A} \\ \dfrac{1+\ln\left(\dfrac{A|x|}{x_{máx}}\right)}{1+\ln A}\,sg(x) & \dfrac{1}{A} \le \dfrac{|x|}{x_{máx}} \le 1 \end{cases} \qquad (3.5.7)$$

Ejercicio 3_5

Enunciado

a) Realice dos funciones en matlab 'leymu.m' e 'invmu.m' que implemente la característica de compresión de la ley μ y su inversa, su uso será el siguiente:

```
% [c,xmax]  = leymu(x,mu)
%        realiza la función de compresión siguiendo la ley mu
%         c    =   característica de compresión
%        x     =   señal original a cuantizar
%       mu    =  parámetro de compresión
%    xmax    =  maximo de x

%   x  = inv_mu(c,mu,xmax)
%        realiza la función de expansión  siguiendo la ley mu
```

```
function [c,xmax]=leymu(x,mu)

xmax=max(x);
c=sign(x).*(log(1+(mu*abs(x/xmax)))/log(1+mu))
```

```
function x=invmu(c,mu,xmax)
K=log(1+mu);
x=exp(abs(c)*log(1+mu))-1.0;
x=x/mu;
x=xmax*x;
x=x.*sign(c);
```

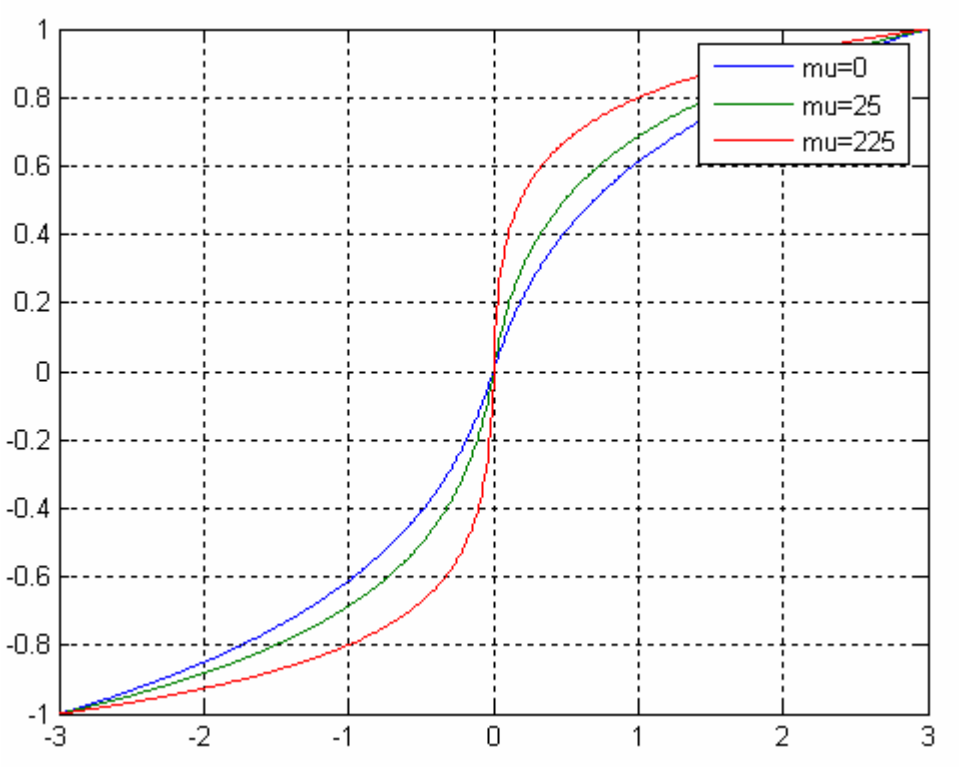
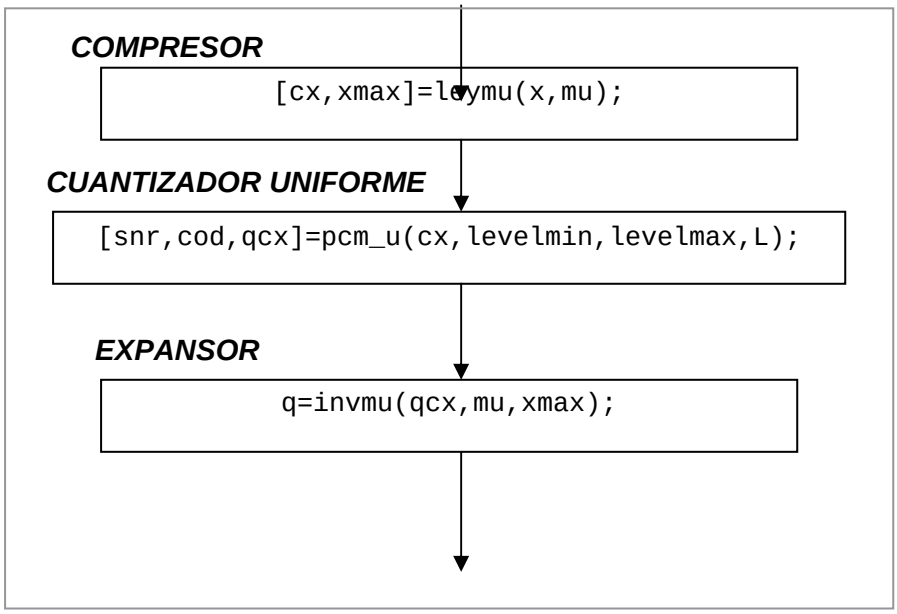


Figura 3.17 .Característica de compresión. Ley μ

b) Realice una funcion en matlab 'pcm_mu.m' que realice una codificación pcm no uniforme siguiendo la ley μ Sigue el siguiente esquema.

```
[snr_mu,cod_mu,q_mu]=pcm_mu(x,levelmin,levelmax,L,mu);
```



```
function [snr_mu,cod_mu,q_mu]=pcm_mu(x,levelmin,levelmax,L,mu)

[cx,xmax]=leymu(x,mu);
[snr,cod_mu,qcx]=pcm_u(cx,levelmin,levelmax,L);
q_mu=invmu(qcx,mu,xmax);
snr_mu=20*log(norm(x)/norm(x-q_mu));
```

c) Repita el apartado b) del ejercicio anterior utilizando la función pcm_mu para L=128 y L=256..

```
paso=0.05;
Tfinal=10;
t=[0:paso:Tfinal];
x=sin(t);
mu=255;

[snr128,cod128,y128]=pcm_mu(x,-1,1,128,255);
[snr256,cod256,y256]=pcm_mu(x,-1,1,256,255);

plot(t,y128,t,x);
title(['PCM No Uniforme con L=128 (mu=255). SNR = '
num2str(snr128)]);
legend('señal cuantizada','señal original')
axis([0 10.01 -1.01 1.01])
grid on;

pause %Pulsa una tecla para ver resultado con L=16

plot(t,y256,t,x);
title(['PCM No Uniforme con L=256 (mu=255). SNR = '
num2str(snr256)]);
legend('señal cuantizada','señal original');
axis([0 10.01 -1.01 1.01]);
grid on;
```

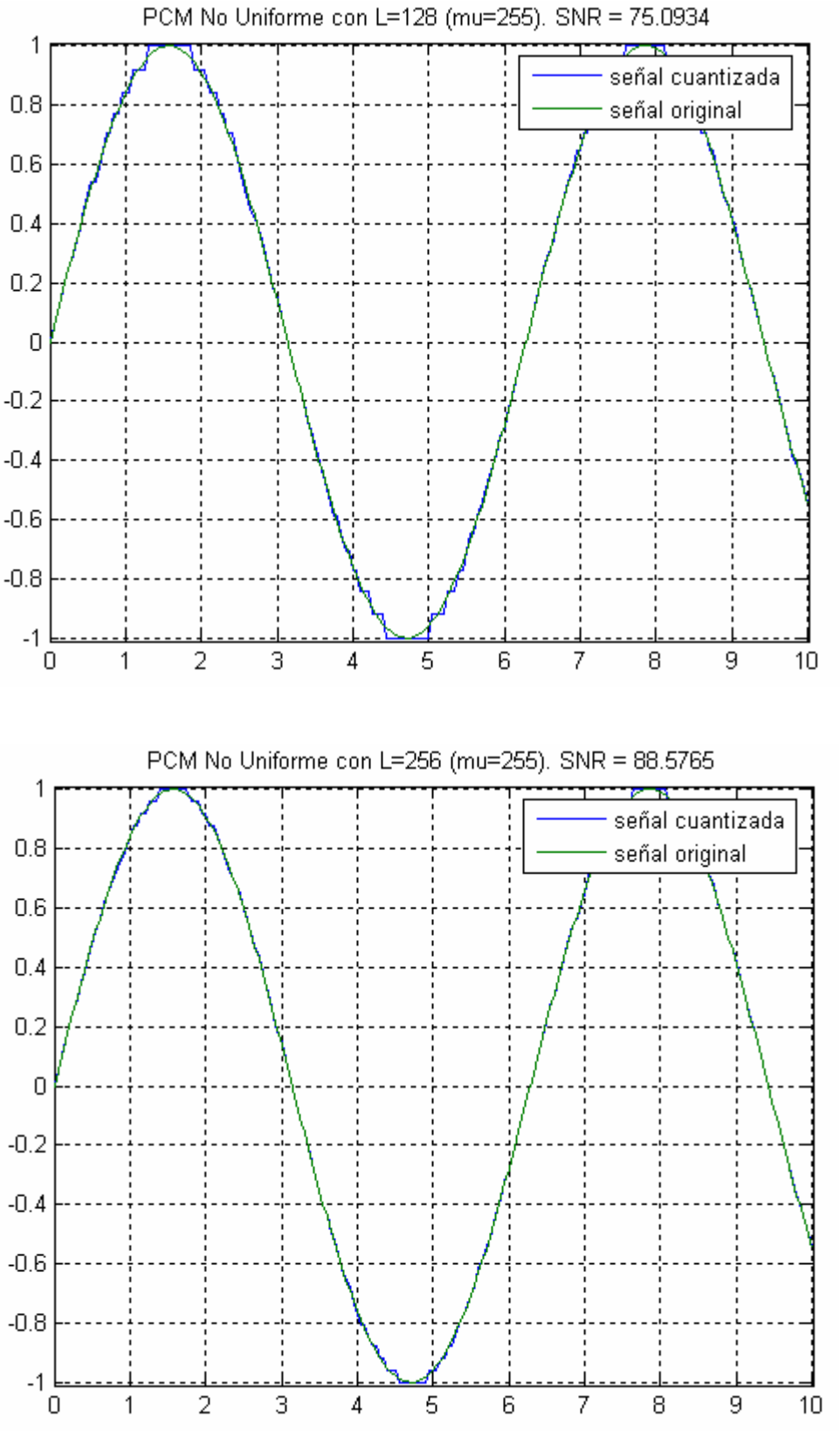


Figura 3.18. Senoide codificada con PCM no uniforme que sigue ley μ