Teoría de las telecomunicaciones
Laboratorio 1

Martín Noblía

Profesores:

Fabian Iakinchuk
Martín Castillo

Universidad Nacional de Quilmes

2015

1 Problema 1

Dada una fuente de datos en un archivo de texto nombrado:`texto.txt`. Escribir un algoritmo de manera tal que calcule la probabilidad de ocurrencia de cada símbolo existente en la fuente. Mostrar en un listado (sea en pantalla o en otro archivo `.txt` cada elemento de la fuente con su probabilidad calculada asociada. Calcular la entropía de la fuente.

1.1. resolución

Para resolver este problema utilizamos el lenguaje de programación Julia(`http://www.julialang.org`) el cual es *opensource*, dinámico y tiene velocidades de procesamiento relativas a C entre otras caracteristicas.

El script que realiza la tarea es el siguiente:

```
#= Script para el problema 1
desde consola:
julia problema1.jl
=#
# incluimos el modulo
include("teleco.jl")
# leemos el archivo
T = open(readall,"../Archivos/texto.txt");
# calculamos las frecuencias y probabilidades
d,p = Teleco.probs(T[1:end-1]);
# Mostramos los resultados
Teleco.show_source(d)
# Calculamos la entropia de la fuente
h = Teleco.entropy(p);
print("La fuente discreta texto.txt tiene una entropia: $h [bits/simbolo]")
```

Las definiciones de funciones estan agrupadas en el modulo `Teleco` que esta definido en el archivo `teleco.jl`. El cual mostramos a continuación:

```
# Modulo para el laboratorio1:Teoria de las telecomunicaciones
module Teleco


function probs(iter)
    # Funcion para calcular las probabilidades de ocurrencia de cada simbolo
    # en una fuente de informacion discreta sin memoria.
    # Input:
    # -----
    # iter: Cualquier fuente iterable(texto,vectores...etc)
    # Output:
    # ------
    # d: Diccionario con los elementos de la fuente y sus frecuencias
    # p: Vector de probabilidades

    d = Dict{eltype(iter), Int}()
    l = length(iter)

    for e in iter
        d[e] = get(d, e, 0) + 1
    end
```

```
    p = collect((values(d))) / l
    return d,p
end

function freqs_pr(iter)
    # Funcion para calcular
    pr = Collections.PriorityQueue()

    for e in iter
        pr["$e"] = get(pr, "$e", 0) + 1
    end

    return pr
end


function entropy(p)
   # Funcion para calcular la entropia de una
   # fuente discreta sin memoria
   # Input:
   # -----
   # p: Vector de probabilidades de la fuente
   # Output:
   # ------
   # h: Entropia de la fuente
    i = log2(1./p)
    aux = p.*i
    h = sum(aux)

    return h

end

function show_source(d)
    # Funcion para mostrar los elementos de una fuente
    # y su frecuencia de aparicion
    for (v,k) in zip(values(d),keys(d))
        println("elemento: $k --> frecuencia: $v")
    end
end

export probs, probs_pr,entropy, show_source


end
```

Cuya salida es:

```
elsuizo@debian:~/Telecomunicaciones/Laboratorios/Lab1/Programas/Julia$ ~/julia/./julia problema1.jl
elemento: á --> frecuencia: 4
elemento: l --> frecuencia: 6
elemento: m --> frecuencia: 2
elemento: r --> frecuencia: 5
```

```
elemento: f --> frecuencia: 1
elemento: o --> frecuencia: 3
elemento: t --> frecuencia: 2
elemento: a --> frecuencia: 14
elemento: E --> frecuencia: 1
elemento: v --> frecuencia: 2
elemento: n --> frecuencia: 5
elemento: e --> frecuencia: 13
elemento: w --> frecuencia: 2
elemento: h --> frecuencia: 3
elemento: d --> frecuencia: 3
elemento: c --> frecuencia: 2
elemento: ; --> frecuencia: 1
elemento: p --> frecuencia: 2
elemento: i --> frecuencia: 4
elemento: q --> frecuencia: 5
elemento: s --> frecuencia: 4
elemento: u --> frecuencia: 8
La fuente discreta texto.txt tiene una entropia: 4.066909301939965 [bits/simbolo]
```

2 Problema 2

Dado un archivo de texto nombrado “texto2.txt” el cual posee una longitud fija de 30 caracteres ASCII:

Escribir un algoritmo que codifique cada elemento de la fuente usando codificación Huffman. (Dicha codificación puede ser mostrada en pantalla o en un archivo de salida *.txt). El software creado debe calcular la entropía y la longitud media del código generado. Calcular en forma manual o con el mismo software la eficiencia de la compresión. Verifique si el código generado es óptimo.

2.1. resolución

3 Problema 3

Dado dos archivos de texto llamados Castellano.txt e ingles.txt, los cuales están en diferente lenguaje, comprimir los mismos en formato ZIP y:

- a- Realizar una tabla donde se pueda visualizar el cociente entre: (tamaño nuevo / tamaño original) x100.
- b- Realizar una tabla con la probabilidad de ocurrencia de cada uno de los caracteres para cada texto.
- c- Analizar los resultados de a) y b). Sacar conclusiones.

3.1. resolución

Para la resolver este problema utilizamos un script en el lenguaje Julia, cuya salida de ejecución genera alguna de las respuestas. A continuación los códigos y las salidas:

Script:

```
# Script para la resolucion del problema 3

# incluimos el modulo
include("teleco.jl")

# leemos los bytes de la fuente castellano
s1 = open(readbytes,"../Archivos/castellano.txt");
# leemos los bytes de la fuente ingles
s2 = open(readbytes,"../Archivos/ingles.txt");
# leemos los bytes de la fuente castellano zipeada
s1_zip = open(readbytes,"../Archivos/castellano.txt.zip");
# leemos los bytes de la fuente ingles zipeada
s2_zip = open(readbytes,"../Archivos/ingles.txt.zip");

# calculamos las longitudes de los archivos(bytes)
l1 = length(s1);
l2 = length(s2);
l1_zip = length(s1_zip);
l2_zip = length(s2_zip);

# calculamos los ratios
r1 = (l1_zip / l1) * 100;
r2 = (l2_zip / l2) * 100;

# mostramos los resultados
print("La eficiencia de compresión para el caso de la fuente en castellano es: $r1\n")

print("La eficiencia de compresión para el caso de la fuente en ingles es: $r2\n")

# Leemos los archivos como texto plano
Tc = open(readall,"../Archivos/castellano.txt");
Ti = open(readall,"../Archivos/ingles.txt");

# Filtramos el texto para quitar los "\n" y los espacios " "
Tc = filter!(x->(x!='\n')&&(x!=' '),collect(Tc));
```

```
Ti = filter!(x->(x!='\n')&&(x!=' '),collect(Ti));

# calculamos las frecuencias y probabilidades
dc,pc = Teleco.probs(Tc);
d_i,p_i= Teleco.probs(Ti);
# Mostramos los resultados
println("Las frecuencias para el texto en castellano son:\n\n")
Teleco.show_source(dc)
println("Las frecuencias para el texto en ingles son:\n\n")
Teleco.show_source(d_i)
hc = Teleco.entropy(pc);
hi = Teleco.entropy(p_i);

print("La fuente discreta castellano.txt tiene una entropia: $hc [bits/simbolo]\n")
print("La fuente discreta ingles.txt tiene una entropia: $hi [bits/simbolo]\n")
```

Salida del script:

```
elsuizo@debian:~/Telecomunicaciones/Laboratorios/Lab1/Programas/Julia$ ~/julia/./julia problema3.jl
La eficiencia de compresión para el caso de la fuente en castellano es: 47.286527514231494
La eficiencia de compresión para el caso de la fuente en ingles es: 46.04361370716511
Las frecuencias para el texto en castellano son:


elemento: r --> frecuencia: 257
elemento: D --> frecuencia: 5
elemento: T --> frecuencia: 3
elemento: v --> frecuencia: 60
elemento: Y --> frecuencia: 6
elemento: y --> frecuencia: 56
elemento: d --> frecuencia: 194
elemento: c --> frecuencia: 152
elemento: I --> frecuencia: 1
elemento: g --> frecuencia: 55
elemento: p --> frecuencia: 102
elemento: i --> frecuencia: 198
elemento: q --> frecuencia: 46
elemento: S --> frecuencia: 4
elemento: : --> frecuencia: 5
elemento: á --> frecuencia: 26
elemento: f --> frecuencia: 26
elemento: ¿ --> frecuencia: 3
elemento: b --> frecuencia: 64
elemento: M --> frecuencia: 1
elemento: C --> frecuencia: 2
elemento: , --> frecuencia: 62
elemento: E --> frecuencia: 10
elemento: - --> frecuencia: 6
elemento: z --> frecuencia: 19
elemento: L --> frecuencia: 2
elemento: é --> frecuencia: 28
elemento: ú --> frecuencia: 1
elemento: s --> frecuencia: 249
elemento: t --> frecuencia: 172
```

```
elemento: a --> frecuencia: 529
elemento: P --> frecuencia: 3
elemento: x --> frecuencia: 4
elemento: ? --> frecuencia: 3
elemento: e --> frecuencia: 521
elemento: h --> frecuencia: 45
elemento: ; --> frecuencia: 6
elemento: A --> frecuencia: 6
elemento: í --> frecuencia: 19
elemento: V --> frecuencia: 1
elemento: H --> frecuencia: 1
elemento: l --> frecuencia: 225
elemento: m --> frecuencia: 125
elemento: ¡ --> frecuencia: 1
elemento: o --> frecuencia: 322
elemento: ó --> frecuencia: 30
elemento: ñ --> frecuencia: 11
elemento: U --> frecuencia: 1
elemento: ! --> frecuencia: 1
elemento: B --> frecuencia: 3
elemento: n --> frecuencia: 293
elemento: N --> frecuencia: 3
elemento: k --> frecuencia: 1
elemento: j --> frecuencia: 15
elemento: . --> frecuencia: 33
elemento: u --> frecuencia: 203
Las frecuencias para el texto en ingles son:


elemento: ’ --> frecuencia: 2
elemento: r --> frecuencia: 189
elemento: D --> frecuencia: 2
elemento: T --> frecuencia: 9
elemento: v --> frecuencia: 26
elemento: Y --> frecuencia: 3
elemento: w --> frecuencia: 99
elemento: y --> frecuencia: 81
elemento: d --> frecuencia: 179
elemento: c --> frecuencia: 82
elemento: I --> frecuencia: 22
elemento: g --> frecuencia: 96
elemento: p --> frecuencia: 64
elemento: i --> frecuencia: 229
elemento: q --> frecuencia: 2
elemento: S --> frecuencia: 3
elemento: f --> frecuencia: 79
elemento: M --> frecuencia: 1
elemento: b --> frecuencia: 33
elemento: C --> frecuencia: 5
elemento: , --> frecuencia: 75
elemento: E --> frecuencia: 3
elemento: W --> frecuencia: 7
elemento: - --> frecuencia: 1
elemento: L --> frecuencia: 2
```

```
elemento: s --> frecuencia: 221
elemento: t --> frecuencia: 303
elemento: a --> frecuencia: 326
elemento: P --> frecuencia: 1
elemento: x --> frecuencia: 2
elemento: ? --> frecuencia: 3
elemento: e --> frecuencia: 424
elemento: h --> frecuencia: 224
elemento: A --> frecuencia: 13
elemento: J --> frecuencia: 1
elemento: H --> frecuencia: 7
elemento: G --> frecuencia: 2
elemento: l --> frecuencia: 178
elemento: m --> frecuencia: 102
elemento: o --> frecuencia: 320
elemento: ! --> frecuencia: 4
elemento: B --> frecuencia: 1
elemento: n --> frecuencia: 261
elemento: F --> frecuencia: 2
elemento: N --> frecuencia: 1
elemento: k --> frecuencia: 32
elemento: j --> frecuencia: 2
elemento: . --> frecuencia: 39
elemento: " --> frecuencia: 12
elemento: u --> frecuencia: 105
La fuente discreta castellano.txt tiene una entropia: 4.410238843123111 [bits/simbolo]
La fuente discreta ingles.txt tiene una entropia: 4.430201945105413 [bits/simbolo]
```

4 Problema 4

Se propone simular una cuantización uniforme de 4, 3 y 2 bits para una señal sinusoidal de amplitud 1 y frecuencia 1 Hz.

- a- Comparar la señal error y su relación con el número de bits de cuantizador para 4 bits ($q = 2/16$), 3 bits ($q = 2/8$) y 2 bits ($q = 2/8$)
- b- Sacar conclusiones.

4.1. resolución

El modulo en donde guardamos las funciones es `teleco2.jl` que utilizamos posteriormente en el script *problema4.jl*, donde comparamos la señal original $x(t) = Asin(2\pi ft)$ con sus versiones quantizadas uniformemente (con los distintos niveles propuestos)

A continuación su salida gráfica:

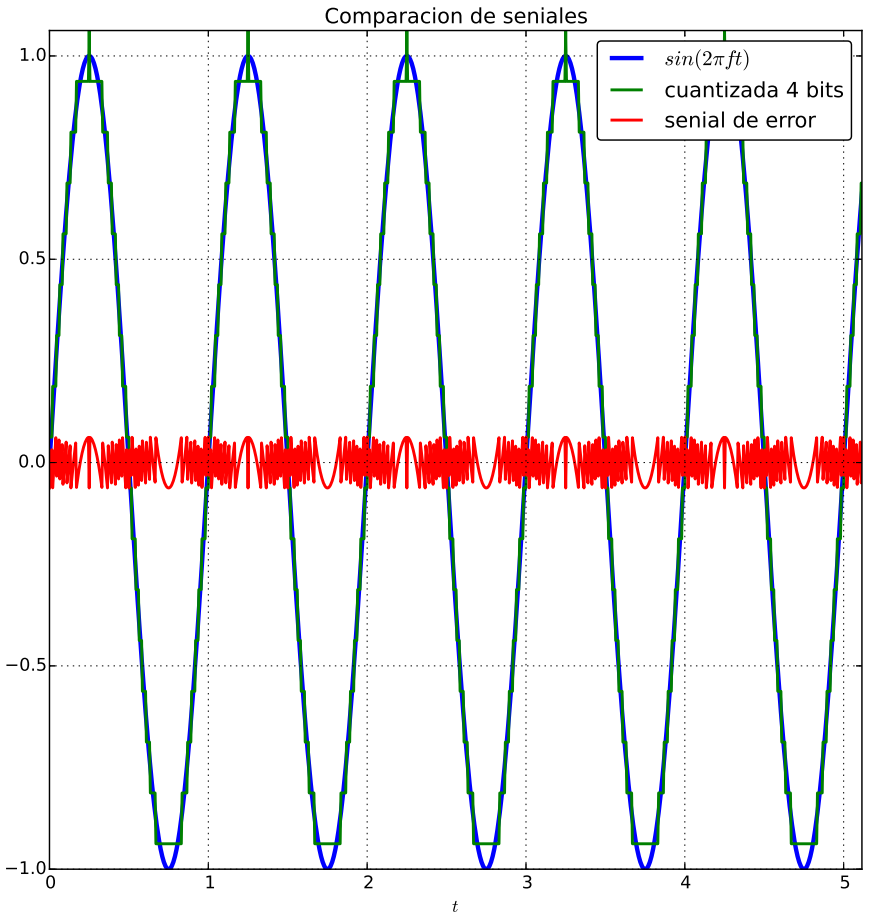

Figura 1: Cuantización 4 bits

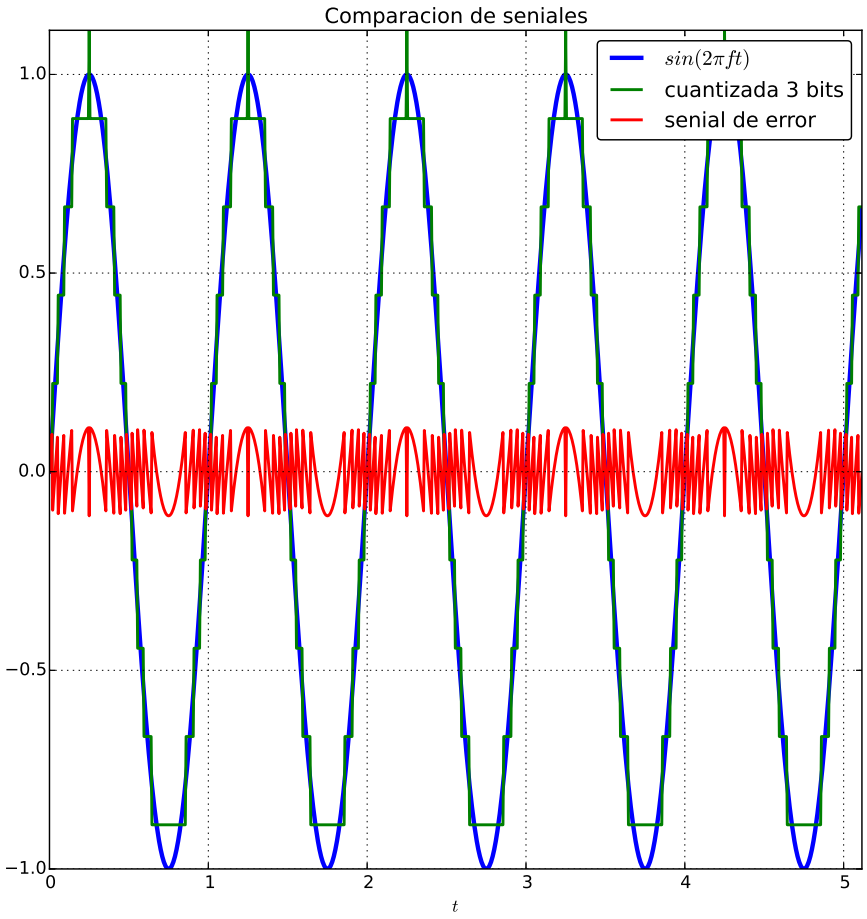

Figura 2: Cuantización 3 bits

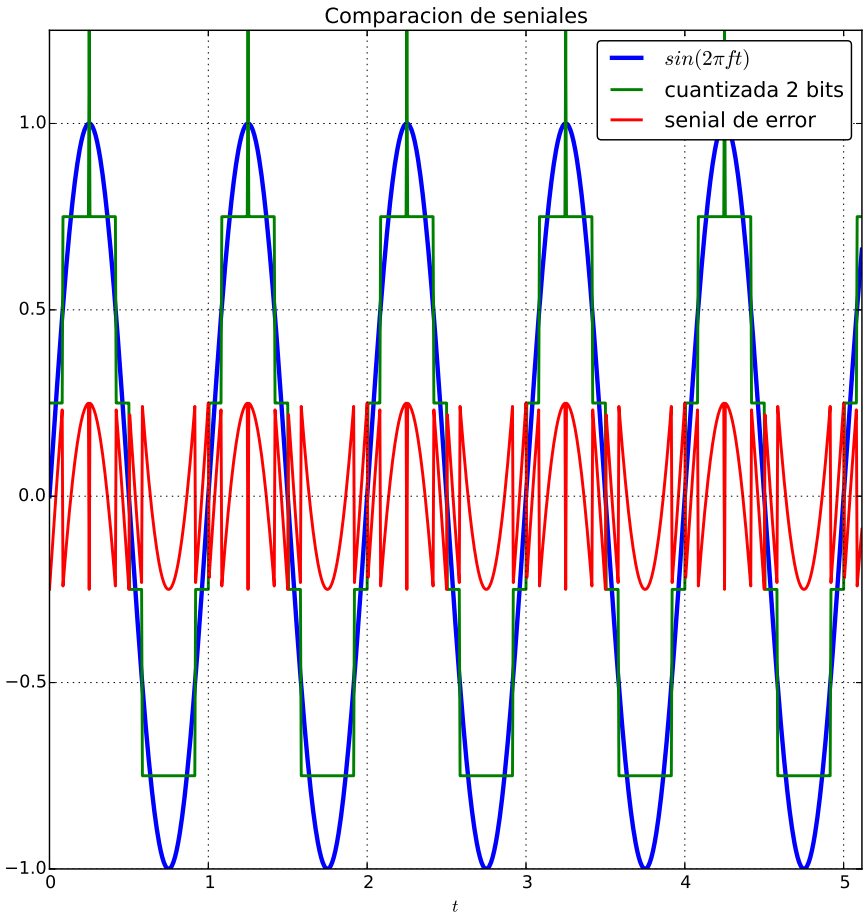

Figura 3: Cuantización 2 bits

Como se puede apreciar la señal de error $e(t) = x - x_{q_i}$ tiene amplitud más grande al cuantizar a la señal original con menos niveles.

5 Problema 5

Se propone ahora simular los efectos de la cuantización uniforme sobre una señal de voz (tomada de algún archivo de audio de pocos segundos de duración muestreada con n bits de manera que su calidad sea media). A partir de una señal de voz original se pide cuantizar la señal con menos bits que la original

- a- Medir la SNR para cada uno de los casos anteriores y reproducir la señal cuantizada en la tarjeta de audio. Comentar los resultados.

5.1. resolución

Nuevamente se realizó un script(`problema5.jl`) que realiza la lectura-escritura de los archivos `wav` y además comparamos gráficamente todas las seniales. A continuación las salidas del mencionado script.

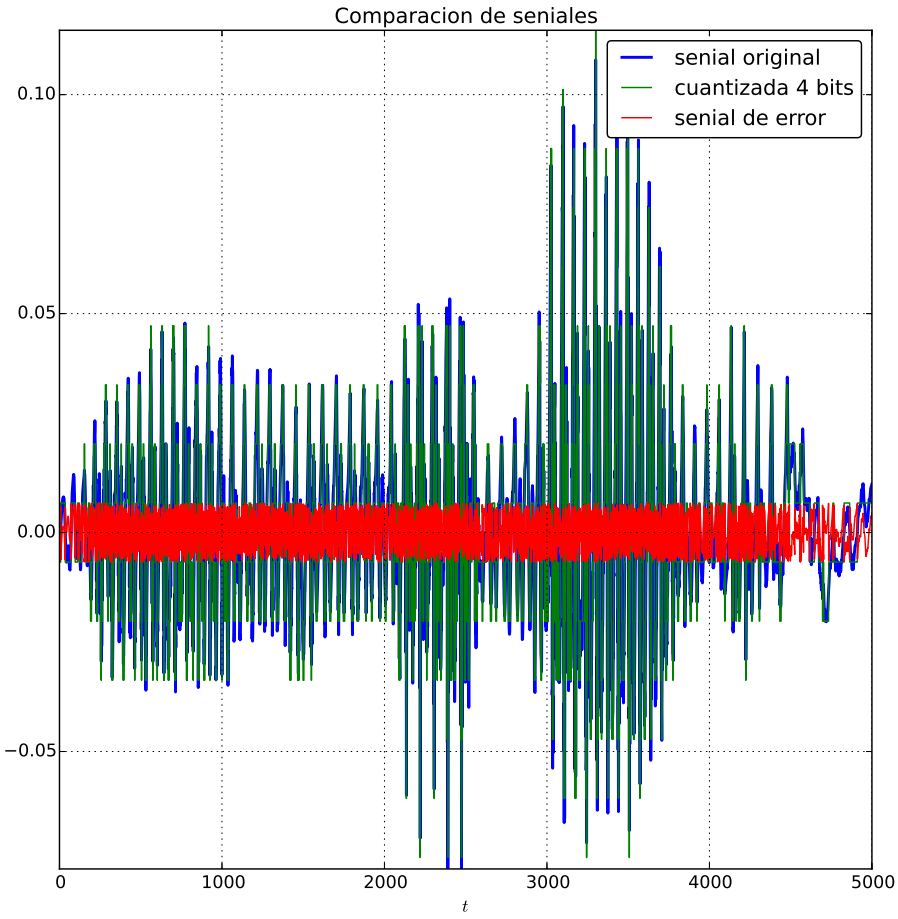

Figura 4: Cuantización 4 bits

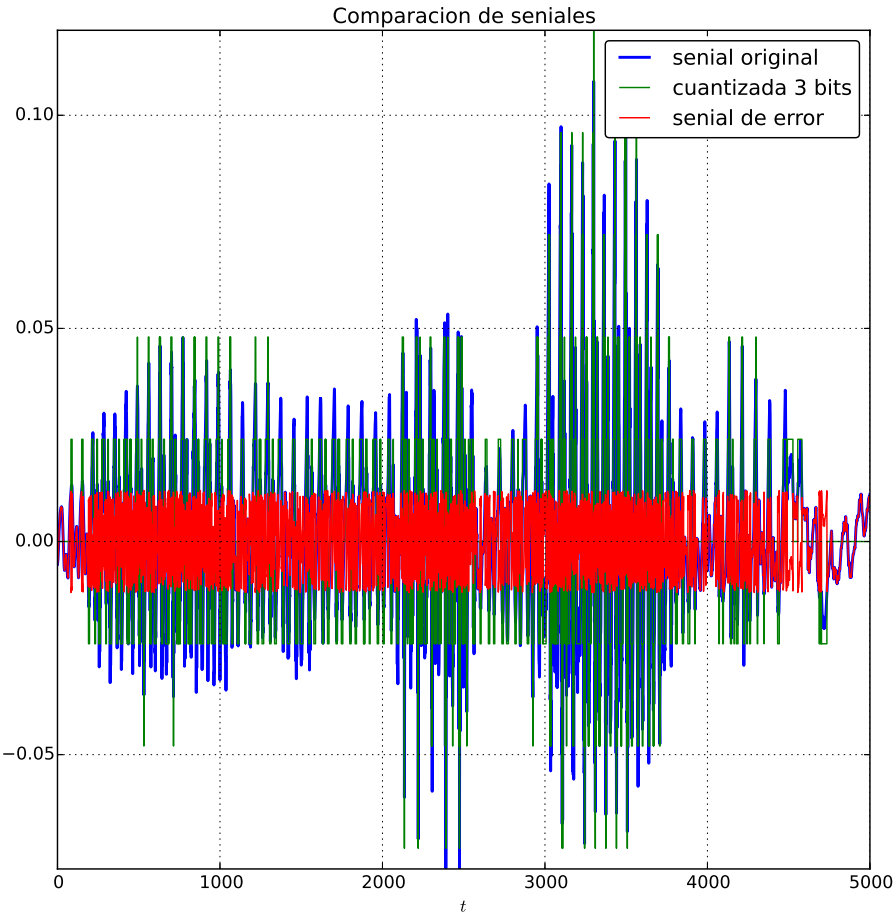

Figura 5: Cuantización 3 bits

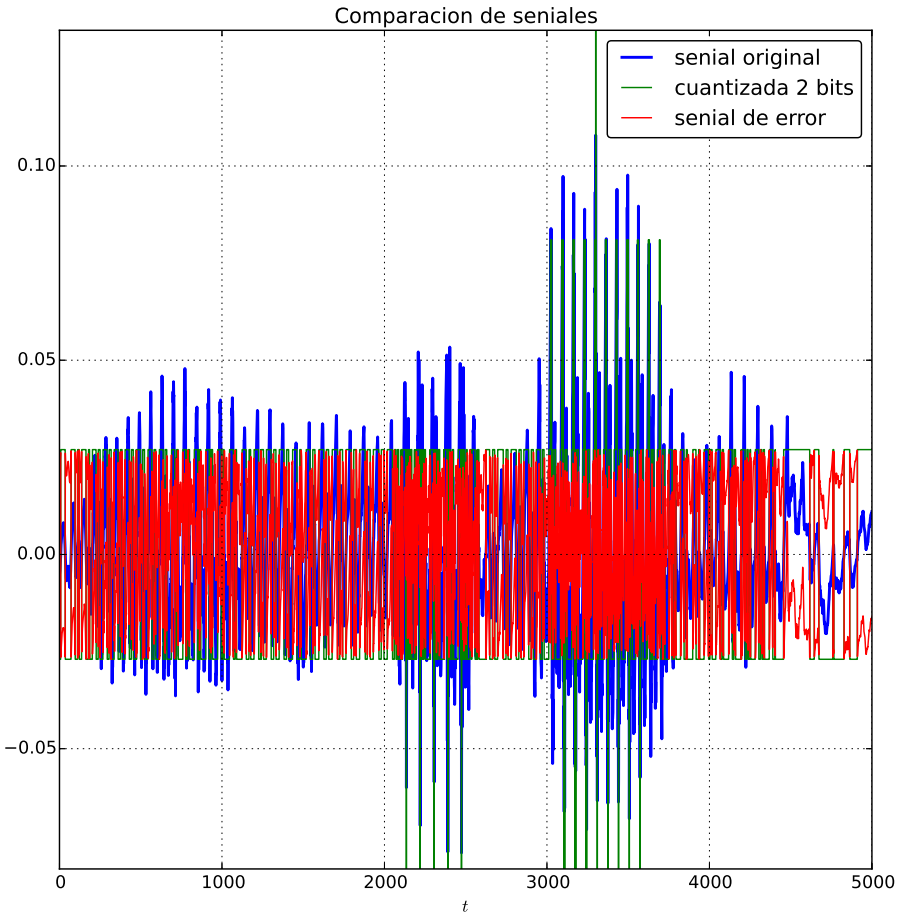

Figura 6: Cuantización 2 bits

La relación senial-ruido para una senial cuantizada uniformemente se puede expresar:

$$\left(\frac{S}{N}\right)_q = \frac{L^2 q^2/4}{q^2/12} = 3L^2 \tag{1}$$

Donde L son los números de niveles de cuantización.

Por ello para 2 bits de resolución tenemos(en $[Db]$):

```
julia> 10*log10(12*1^2/2/(2/4)^2)
13.80211241711606
```

para 3 bits

```
julia>10*log10(12*1^2/2/(2/8)^2)
19.822712330395685
```

para 4 bits

```
julia>10*log10(12*1^2/2/(2/16)^2)
25.84331224367531
```

A la salida del script obtemos:

```
elsuizo@debian:~/Telecomunicaciones/Laboratorios/Lab1/Programas/Julia$ ~/julia/./julia problema5.jl
```

```
INFO: Loading help data...
La relación senial-ruido para 2 bits es: 4.7383394267141[Db]
La relación senial-ruido para 3 bits es: 9.682605271481124[Db]
La relación senial-ruido para 4 bits es: 20.744776704544634[Db]
```

Como sabemos las seniales de voz estadisticamente tienen amplitudes bajas, osea que los niveles altos de voz son pocos probables, de alli que para esta senial de voz obtenemos resultados distintos de los teoricos que modelan a la probabilidad de $p(q) = 1/q$ además si tomamos como indicador de como se degrada la senial de acuerdo a la varianza del error producido por el redondeo, vimos que la varianza $\sigma^2 = \frac{q^2}{12}$ se corresponde con la potencia media de ruido a la cuantización. Por ello como podemos ver en la cuantización uniforme el ruido se mantiene constante ya que q es constante, pero como los niveles de amplitud no entonces la relación senial-ruido varía.

6 Problema 6

Modificando el diagrama utilizado en la simulación de la cuantización uniforme para señales de voz para implementar el cuantizador ley-μ.

- a- Medir la SNR para cada uno de los casos (cuantización de 4, 3 y 2 bits) y reproducir la señal cuantizada en la tarjeta de audio.
- b- Modificar el esquema y adicionar dos "SCOPE", uno antes de comprimir la señal con un filtro con ley-μ y otro posterior al filtro con ley-μ. Sacar conclusiones en base a lo visualizado.
- c- Comparar con los resultados del cuantizador uniforme y sacar conclusiones.
- d- Graficar la Densidad Espectral de la Señal (Utilizando Matlab) y Sacar Conclusiones:
  - A la entrada del Sistema.
  - A la salida del filtro ley-μ.
  - A la salida del Sistema.

Referencias

[1] Digital communications, fundamentals and applications. Bernard Sklar

[2] Julia: A fresh approach to numerical computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah(2014)`http://arxiv.org/abs/1411.1607`