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 todos los problemas 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.

Las definiciones de funciones estan agrupadas en el modulo `Teleco` que esta definido en el archivo `teleco.jl`.

Cuya salida es:

```
julia> include("problema1.jl")
WARNING: replacing module Teleco
elemento: p --> frecuencia: 2 --> probabilidad: 0.021739130434782608
elemento: h --> frecuencia: 3 --> probabilidad: 0.03260869565217391
elemento: m --> frecuencia: 2 --> probabilidad: 0.021739130434782608
elemento: ; --> frecuencia: 1 --> probabilidad: 0.010869565217391304
elemento: v --> frecuencia: 2 --> probabilidad: 0.021739130434782608
elemento: q --> frecuencia: 5 --> probabilidad: 0.05434782608695652
elemento: t --> frecuencia: 2 --> probabilidad: 0.021739130434782608
elemento: e --> frecuencia: 13 --> probabilidad: 0.14130434782608695
elemento: n --> frecuencia: 5 --> probabilidad: 0.05434782608695652
elemento: d --> frecuencia: 3 --> probabilidad: 0.03260869565217391
elemento: l --> frecuencia: 6 --> probabilidad: 0.06521739130434782
elemento: f --> frecuencia: 1 --> probabilidad: 0.010869565217391304
elemento: w --> frecuencia: 2 --> probabilidad: 0.021739130434782608
elemento: i --> frecuencia: 4 --> probabilidad: 0.043478260869565216
elemento: E --> frecuencia: 1 --> probabilidad: 0.010869565217391304
elemento: á --> frecuencia: 4 --> probabilidad: 0.043478260869565216
elemento: a --> frecuencia: 14 --> probabilidad: 0.15217391304347827
elemento: r --> frecuencia: 5 --> probabilidad: 0.05434782608695652
elemento: u --> frecuencia: 8 --> probabilidad: 0.08695652173913043
elemento: o --> frecuencia: 3 --> probabilidad: 0.03260869565217391
elemento: s --> frecuencia: 4 --> probabilidad: 0.043478260869565216
elemento: c --> frecuencia: 2 --> probabilidad: 0.021739130434782608
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

Nuevamente las funciones que implementan los algoritmos para la resolucion de este problema estan en el archivo `teleco.jl`. El script que organiza la resolucion del problema es el siguiente:

Cuya salida es:

```
julia> include("problema2.jl")
WARNING: replacing module Teleco
Fuente y sus probabilidades
elemento: ú --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: i --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: e --> frecuencia: 4 --> probabilidad: 0.13333333333333333
elemento: l --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: u --> frecuencia: 2 --> probabilidad: 0.06666666666666667
elemento: a --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: j --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: o --> frecuencia: 4 --> probabilidad: 0.13333333333333333
elemento: . --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: m --> frecuencia: 3 --> probabilidad: 0.1
elemento: s --> frecuencia: 2 --> probabilidad: 0.06666666666666667
elemento: T --> frecuencia: 1 --> probabilidad: 0.03333333333333333
elemento: t --> frecuencia: 3 --> probabilidad: 0.1
elemento: r --> frecuencia: 5 --> probabilidad: 0.16666666666666666
Fuente y sus codigos de Huffmann
elemento: ú --> codigo: 1110
elemento: i --> codigo: 01001
elemento: e --> codigo: 000
elemento: l --> codigo: 01000
elemento: u --> codigo: 0101
elemento: j --> codigo: 00100
elemento: a --> codigo: 00101
elemento: o --> codigo: 100
elemento: m --> codigo: 101
elemento: . --> codigo: 00111
elemento: s --> codigo: 1111
elemento: T --> codigo: 00110
elemento: t --> codigo: 110
elemento: r --> codigo: 011
La fuente discreta texto2.txt tiene una entropia: 3.53624341298306 [bits/simbolo]

El codigo de Huffmann de la fuente discreta texto2.txt tiene una longitud promedio: 3.6999999999999997 [bi
La eficiencia del codigo Huffman para la fuente discreta texto2.txt es: 95.57414629683947
Es el codigo generado optimo? : true
```

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 las respuestas a y b. A continuación la salida del script `problema3.jl`

Salida del script:

```
julia> include("problema3.jl")
WARNING: replacing module Teleco
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: m --> frecuencia: 125 --> probabilidad: 0.02962085308056872
elemento: h --> frecuencia: 45 --> probabilidad: 0.01066350710900474
elemento: E --> frecuencia: 10 --> probabilidad: 0.002369668246445498
elemento: t --> frecuencia: 172 --> probabilidad: 0.04075829383886256
elemento: C --> frecuencia: 2 --> probabilidad: 0.0004739336492890954
elemento: s --> frecuencia: 249 --> probabilidad: 0.05900473933649289
elemento: d --> frecuencia: 194 --> probabilidad: 0.04597156398104266
elemento: , --> frecuencia: 62 --> probabilidad: 0.014691943127962086
elemento: b --> frecuencia: 64 --> probabilidad: 0.015165876777251185
elemento: M --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: T --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: i --> frecuencia: 198 --> probabilidad: 0.04691943127962085
elemento: P --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: V --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: ¡ --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: ? --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: v --> frecuencia: 60 --> probabilidad: 0.014218009478672985
elemento: S --> frecuencia: 4 --> probabilidad: 0.0009478672985781991
elemento: r --> frecuencia: 257 --> probabilidad: 0.060900473933649286
elemento: l --> frecuencia: 225 --> probabilidad: 0.0533175355450237
elemento: z --> frecuencia: 19 --> probabilidad: 0.004502369668246445
elemento: y --> frecuencia: 56 --> probabilidad: 0.013270142180094787
elemento: q --> frecuencia: 46 --> probabilidad: 0.010900473933649289
elemento: U --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: D --> frecuencia: 5 --> probabilidad: 0.001184834123222749
elemento: á --> frecuencia: 26 --> probabilidad: 0.006161137440758294
elemento: N --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: H --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: j --> frecuencia: 15 --> probabilidad: 0.0035545023696682463
```

```
elemento: - --> frecuencia: 6 --> probabilidad: 0.0014218009478672985
elemento: ¿ --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: : --> frecuencia: 5 --> probabilidad: 0.001184834123222749
elemento: k --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: B --> frecuencia: 3 --> probabilidad: 0.0007109004739336493
elemento: ; --> frecuencia: 6 --> probabilidad: 0.0014218009478672985
elemento: I --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: é --> frecuencia: 28 --> probabilidad: 0.006635071090047393
elemento: A --> frecuencia: 6 --> probabilidad: 0.0014218009478672985
elemento: g --> frecuencia: 55 --> probabilidad: 0.013033175355450236
elemento: ú --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: p --> frecuencia: 102 --> probabilidad: 0.024170616113744076
elemento: e --> frecuencia: 521 --> probabilidad: 0.12345971563981042
elemento: x --> frecuencia: 4 --> probabilidad: 0.0009478672985781991
elemento: u --> frecuencia: 203 --> probabilidad: 0.0481042654028436
elemento: . --> frecuencia: 33 --> probabilidad: 0.007819905213270141
elemento: c --> frecuencia: 152 --> probabilidad: 0.03601895734597156
elemento: ó --> frecuencia: 30 --> probabilidad: 0.0071090047393364926
elemento: L --> frecuencia: 2 --> probabilidad: 0.0004739336492890954
elemento: f --> frecuencia: 26 --> probabilidad: 0.006161137440758294
elemento: o --> frecuencia: 322 --> probabilidad: 0.07630331753554502
elemento: ñ --> frecuencia: 11 --> probabilidad: 0.0026066350710900474
elemento: í --> frecuencia: 19 --> probabilidad: 0.004502369668246445
elemento: ! --> frecuencia: 1 --> probabilidad: 0.00023696682464454977
elemento: a --> frecuencia: 529 --> probabilidad: 0.12535545023696681
elemento: Y --> frecuencia: 6 --> probabilidad: 0.0014218009478672985
elemento: n --> frecuencia: 293 --> probabilidad: 0.06943127962085308
Las frecuencias para el texto en ingles son:


elemento: h --> frecuencia: 224 --> probabilidad: 0.0577319587628866
elemento: ’ --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: v --> frecuencia: 26 --> probabilidad: 0.006701030927835051
elemento: P --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
elemento: S --> frecuencia: 3 --> probabilidad: 0.0007731958762886598
elemento: d --> frecuencia: 179 --> probabilidad: 0.046134020618556704
elemento: M --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
elemento: m --> frecuencia: 102 --> probabilidad: 0.026288659793814433
elemento: c --> frecuencia: 82 --> probabilidad: 0.021134020618556702
elemento: E --> frecuencia: 3 --> probabilidad: 0.0007731958762886598
elemento: ? --> frecuencia: 3 --> probabilidad: 0.0007731958762886598
elemento: b --> frecuencia: 33 --> probabilidad: 0.008505154639175257
elemento: J --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
elemento: e --> frecuencia: 424 --> probabilidad: 0.10927835051546392
elemento: I --> frecuencia: 22 --> probabilidad: 0.005670103092783505
elemento: j --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: o --> frecuencia: 320 --> probabilidad: 0.08247422680412371
elemento: C --> frecuencia: 5 --> probabilidad: 0.001288659793814433
elemento: T --> frecuencia: 9 --> probabilidad: 0.002319587628865794
elemento: r --> frecuencia: 189 --> probabilidad: 0.04871134020618557
elemento: A --> frecuencia: 13 --> probabilidad: 0.0033505154639175256
elemento: w --> frecuencia: 99 --> probabilidad: 0.025515463917525773
elemento: k --> frecuencia: 32 --> probabilidad: 0.008247422680412371
elemento: F --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
```

```
elemento: i --> frecuencia: 229 --> probabilidad: 0.05902061855670103
elemento: q --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: s --> frecuencia: 221 --> probabilidad: 0.05695876288659794
elemento: ! --> frecuencia: 4 --> probabilidad: 0.0010309278350515464
elemento: W --> frecuencia: 7 --> probabilidad: 0.0018041237113402063
elemento: x --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: t --> frecuencia: 303 --> probabilidad: 0.07809278350515464
elemento: n --> frecuencia: 261 --> probabilidad: 0.0672680412371134
elemento: H --> frecuencia: 7 --> probabilidad: 0.0018041237113402063
elemento: D --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: a --> frecuencia: 326 --> probabilidad: 0.08402061855670104
elemento: " --> frecuencia: 12 --> probabilidad: 0.003092783505154639
elemento: - --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
elemento: , --> frecuencia: 75 --> probabilidad: 0.019329896907216496
elemento: u --> frecuencia: 105 --> probabilidad: 0.027061855670103094
elemento: f --> frecuencia: 79 --> probabilidad: 0.020360824742268042
elemento: N --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
elemento: g --> frecuencia: 96 --> probabilidad: 0.024742268041237112
elemento: l --> frecuencia: 178 --> probabilidad: 0.04587628865979381
elemento: y --> frecuencia: 81 --> probabilidad: 0.020876288659793813
elemento: L --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: Y --> frecuencia: 3 --> probabilidad: 0.0007731958762886598
elemento: p --> frecuencia: 64 --> probabilidad: 0.016494845360824743
elemento: G --> frecuencia: 2 --> probabilidad: 0.0005154639175257732
elemento: . --> frecuencia: 39 --> probabilidad: 0.010051546391752578
elemento: B --> frecuencia: 1 --> probabilidad: 0.0002577319587628866
La fuente discreta castellano.txt tiene una entropia: 4.410238843123112 [bits/simbolo]
La fuente discreta ingles.txt tiene una entropia: 4.430201945105413 [bits/simbolo]
```

Vemos que el texto en castellano tiene mejor eficiencia, esto se debe a que el algoritmo que implementa el formato ZIP es de diccionario dinamico y dada la estructura repetitiva que tienen los textos en castellano(por ejemplo el texto tiene muchos *que*) hace que su compresion sea mayor.

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:

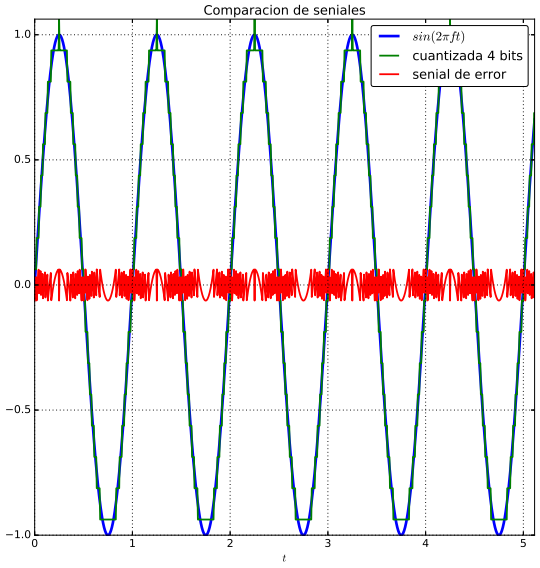

Figura 1: Cuantización 4 bits

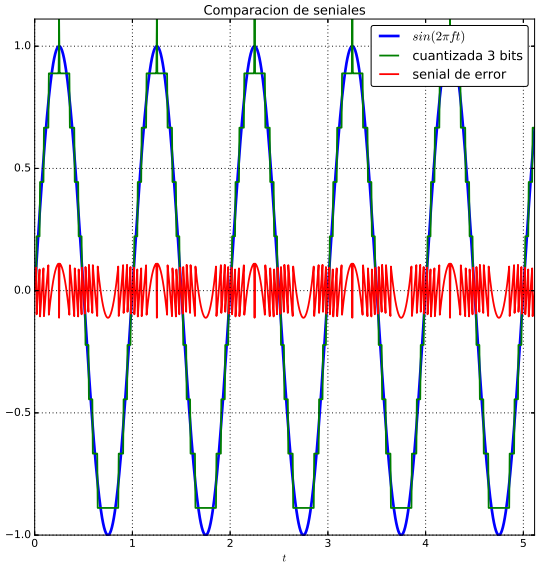

Figura 2: Cuantización 3 bits

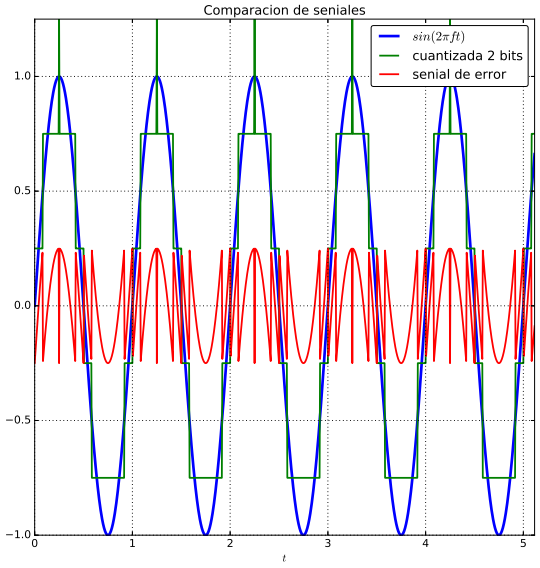

Figura 3: Cuantización 2 bits

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.

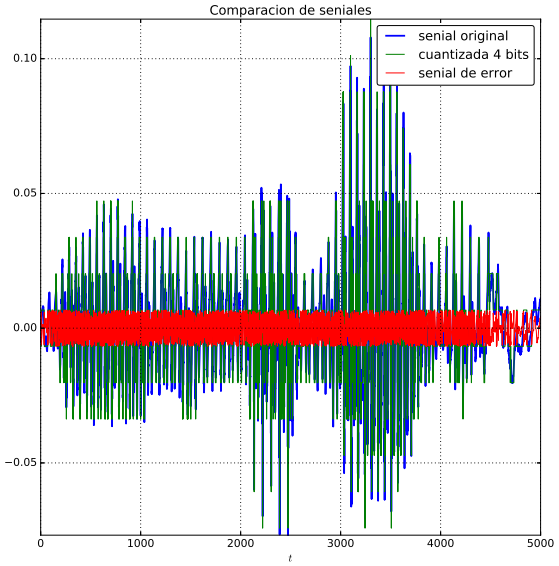

Figura 4: Cuantización 4 bits

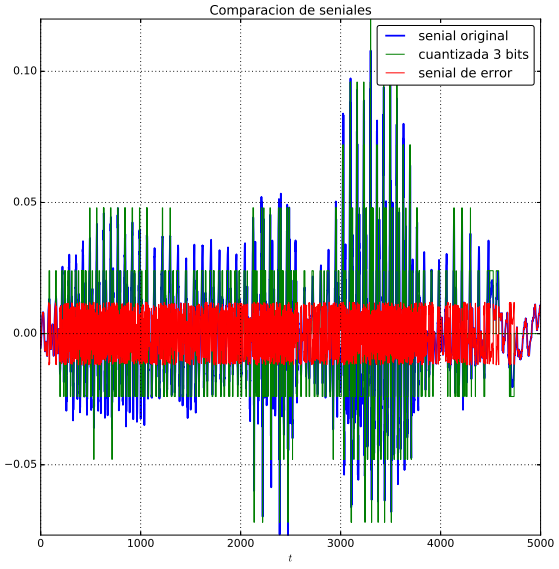

Figura 5: Cuantización 3 bits

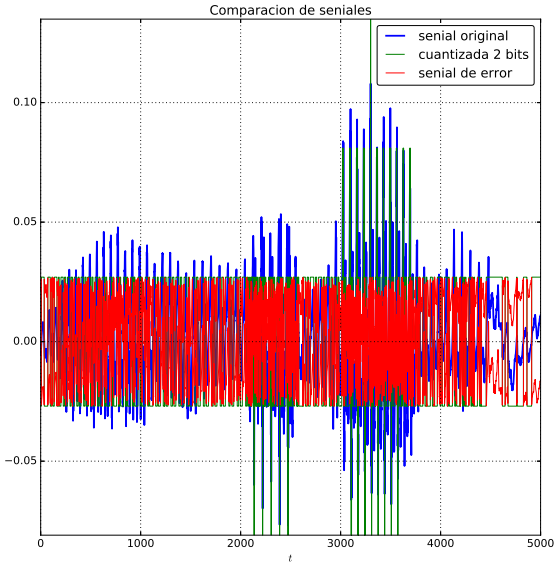

Figura 6: Cuantización 2 bits

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:

```
julia> include("problema5.jl")
WARNING: replacing module Teleco2
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.

6.1. resolución

Nuevamente se realizó un script(`problema6.jl`) que realiza la lectura-escritura de los archivos `wav` y además comparamos gráficamente todas las seniales. A continuación las salidas del mencionado script.

```
julia> include("problema6.jl")
La relación senial-ruido para 2 bits es: 4.994832156111841 [Db]
La relación senial-ruido para 3 bits es: 9.01438241013517 [Db]
La relación senial-ruido para 4 bits es: 26.059268713048507 [Db]
```

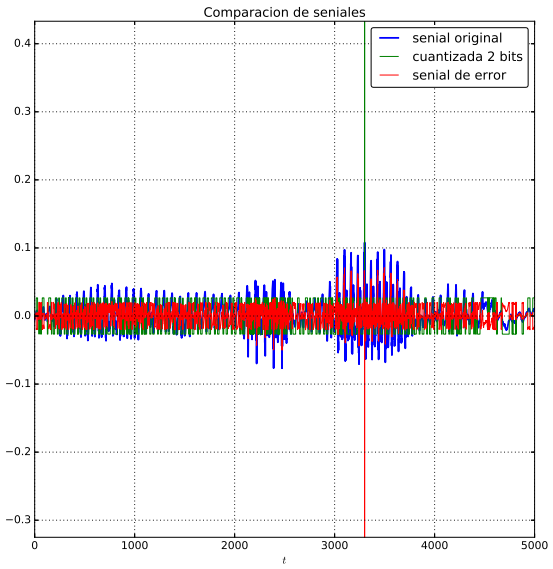

Figura 7: cuantización 2 bits con la ley-μ

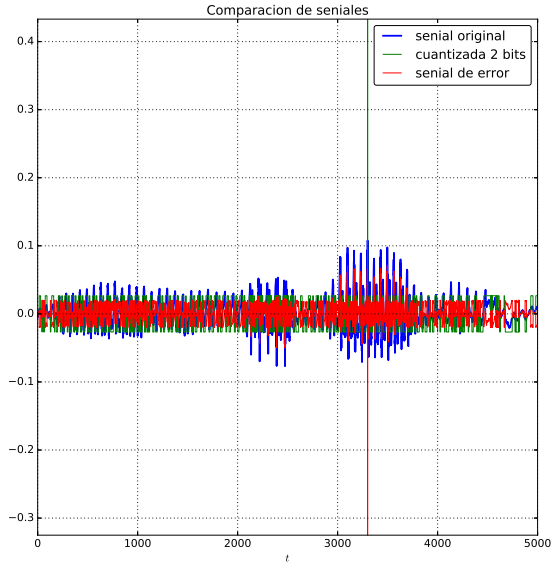

Figura 8: cuantización 3 bits con la ley-μ

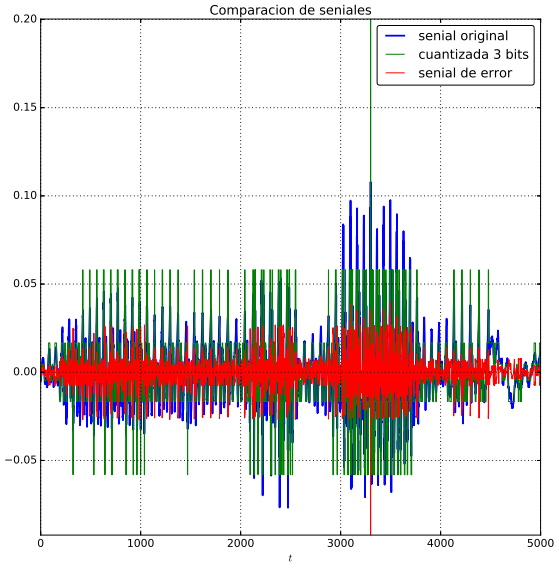

Figura 9: cuantización 4 bits con la ley-μ

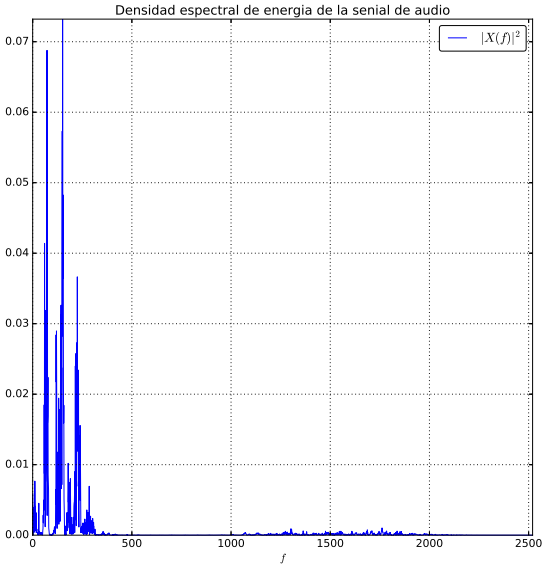

Figura 10: densidad espectral energia original

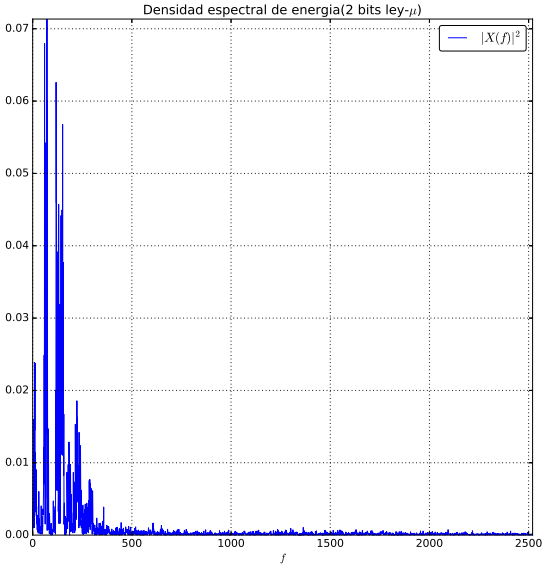

Figura 11: densidad espectral energia 2 bits

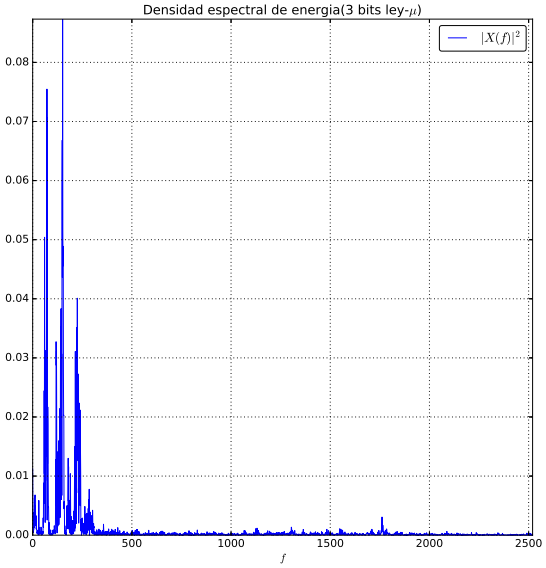

Figura 12: densidad espectral energia 3 bits

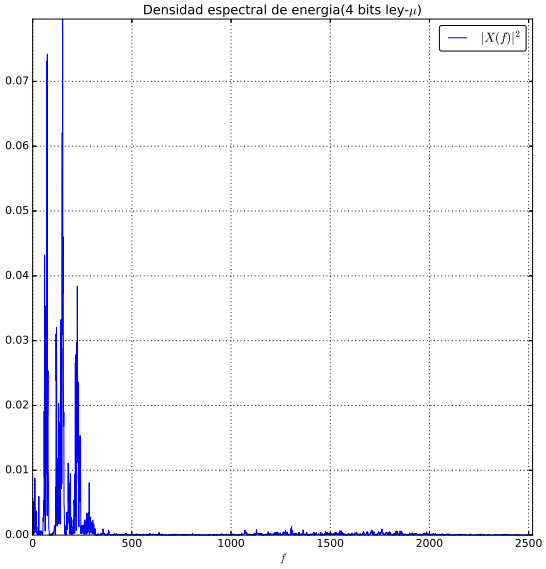

Figura 13: densidad espectral energia 4 bits

Vemos que se mejora la relacion senial-ruido como esperabamos ya que se trata de una senial de voz y la ley de compresion μ esta pensada para estas seniales.

7 Problema 7

Dada los siguientes pares de señales, añada ruido y pruebe el comportamiento del filtro adaptado para la deteccion de las mismas.

$$S_{1a}(t) = \begin{cases} A & : 0 < t < \frac{T}{2} \\ 0 & : \frac{T}{2} < t < T \end{cases}$$

$$S_{1b}(t) = \begin{cases} 0 & : 0 < T < \frac{T}{4} \\ A & : \frac{T}{4} < t < \frac{3T}{4} \\ 0 & : \frac{3T}{4} < t < T \end{cases}$$

$$S_{2a}(t) = \begin{cases} A & : 0 < t < \frac{T}{2} \\ 0 & : \frac{T}{2} < t < T \end{cases}$$

$$S_{2b}(t) = \begin{cases} 0 & : 0 < t < \frac{T}{2} \\ A & : \frac{T}{2} < t < T \end{cases}$$

$$S_{3a}(t) = \begin{cases} A & : 0 < t < \frac{T}{2} \\ 0 & : \frac{T}{2} < T < T \end{cases}$$

$$S_{3b}(t) = \begin{cases} -A & : 0 < t < \frac{T}{2} \\ 0 & : \frac{T}{2} < t < T \end{cases}$$

7.1. resolucion

Nuevamente se realizó un script(`problema7.jl`) que realiza y además comparamos gráficamente todas las seniales. A continuación las salidas del mencionado script.

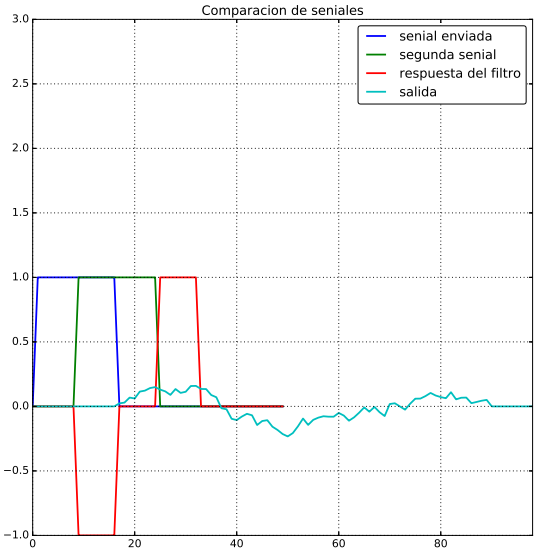

Figura 14: salidas del filtro adaptado para el conjuinto 1

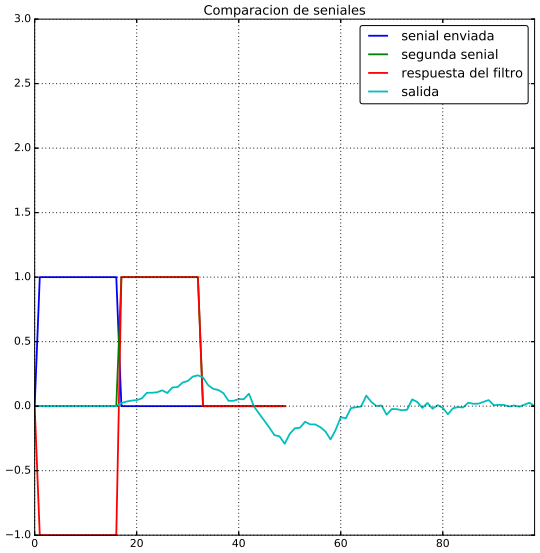

Figura 15: salidas del filtro adaptado para el conjuinto 2

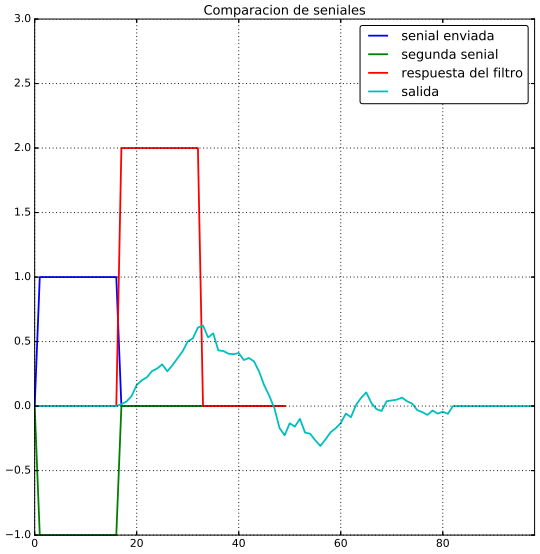

Figura 16: salidas del filtro adaptado para el conjuinto 3

Vemos que el conjunto 3 que es una senial bipolar se comporta mejor, como esperabamos ya que se puede transmitir con 3[db] menos de potencia a igual tasa de error de bit o probabilidad de error de bit.

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`