TRANSMISIÓN DE DATOS 2006/07

Hoja de Resultados I

(Prácticas 1, 2 y 3)

A continuación se detallarán los resultados esperados para varios ejemplos significativos correspondientes a las prácticas 1, 2 y 3 de transmisión de datos. El objetivo de esta especificación de resultados es proporcionar la información necesaria para que el alumno pueda comprobar la correcta implementación de las prácticas.

1 Práctica 1: Codificación Huffman

1.1 Programa huffman.

1.1.1 Ejemplo 1:

Entrada:

[.4 .3 .2 .1]

Salida:

codigo =

[0.4000] '0'

[0.3000] '10'

[0.2000] '111'

[0.1000] '110'

H = 1.8464

longitudMedia = 1.9000

1.1.2 Ejemplo 2:

Entrada:

[.05 .05 .1 .1 .15 .15 .20 .20]

Salida:

codigo =

[0.0500] '11101'

[0.0500] '11100'

[0.1000] '100'

[0.1000] '1111'

[0.1500] '110'

[0.1500] '101'

[0.2000] '01'

[0.2000] '00'

H = 2.8464

longitudMedia = 2.9000

1.1.3 Ejemplo 3:

Entrada:

[.3 .15 .25 .2 .05 .025 .015 .01]

Salida:

codigo =

[0.3000] '11'

[0.2000] '00'

[0.1500] '011'

[0.2500] '10'

[0.0500] '0101'

[0.0250] '01001'

[0.0150] '010001'

[0.0100] '010000'

H = 2.4025

longitudMedia = 2.4250

Nota: Téngase en cuenta que dependiendo de si se aplica la funcion huffmanRecursivo modificada o no podrian intercambiarse los ceros por unos y viceversa.

1.2 Extensión de fuente.

1.2.1 Ejemplo 1:

Entrada:

[.4 .3 .2 .1]

Orden de la extensión de fuente: 2

Salida:

codigo =

[0.1600] '111'

[0.1200] '100'

[0.0800] '1101'

[0.0800] '1100'

[0.1200] '011'

[0.0400] '0010'

[0.0600] '0101'

[0.0400] '10111'

[0.0900] '000'

[0.0400] '10110'

[0.0300] '00111'

[0.0600] '0100'

[0.0200] '00110'

[0.0300] '10100'

[0.0200] '101011'

[0.0100] '101010'

H = 3.6929

longitudMedia = 3.7300

1.3 Extensión de fuente.

1.3.1 Ejemplo 1:

Entrada:

[.4 .3 .2 .1]

Orden de la extensión de fuente: 3

Salida:

codigo =

[0.0640] '1010'

[0.0240] '00100'

[0.0480] '0100'

[0.0360] '11001'

[0.0360] '11000'

[0.0180] '111000'

[0.0160] '011011'

[0.0480] '11110'

[0.0320]	'01111'
[0.0120]	'001101'
[0.0160]	'011010'
[0.0180]	'110101'
[0.0160]	'100111'
[0.0360]	'11011'
[0.0320]	'01110'
[0.0120]	'001100'
[0.0090]	'1011110'
[0.0160]	'100101'
[0.0160]	'100110'
[0.0090]	'1011101'
[0.0480]	'0001'
[0.0270]	'01100'
[0.0240]	'111111'
[0.0120]	'001111'
[0.0080]	'1000100'
[0.0060]	'11100110'
[0.0180]	'110100'
[0.0120]	'010110'
[0.0120]	'1111100'
[0.0320]	'10110'
[0.0240]	'111011'
[0.0080]	'1011100'
[0.0080]	'1000111'
[0.0060]	'0011101'
[0.0040]	'01011111'
[0.0160]	'100100'
[0.0120]	'000000'
[0.0040]	'01011110'
[0.0240]	'01010'
[0.0240]	'111010'
[0.0090]	'1011111'
[0.0080]	'1000110'
[0.0080]	'1000000'
[0.0060]	'0011100'
[0.0040]	'10000111'
[0.0120]	'1111101'
[0.0120]	'000011'
[0.0080]	'1000101'
[0.0060]	'0000011'
[0.0040]	'10000101'
[0.0030]	'01011100'
[0.0240]	'00101'
[0.0080]	'1000001'
[0.0060]	'0000010'
[0.0040]	'11100100'
[0.0030]	'111001011'
[0.0020]	'100001101'
[0.0120]	'000010'
[0.0040]	'10000100'
[0.0020]	'100001100'
[0.0060]	'11100111'
[0.0030]	'111001010'
[0.0020]	'010111011'
[1.0000e-003]	'010111010'

H = 5.5393

longitudMedia = 5.5770

2 Práctica 2: Codificación Aritmética

2.1 Codificador Aritmético

```
function [X] = codificacionAritmetica(P, secuencia, d, n)
```

2.1.1 Ejemplo 1

Entrada:

P = [.4 .3 .2 .1]

secuencia = [1 2 3]

d = 3

n = 8

Salida:

El valor decimal a codificar es: 0.244000

El valor entero a codificar es: 244

X = 11110100

2.1.2 Ejemplo 2

Entrada:

P = [.1 .1 .2 .3 .3]

secuencia = [1 3 2 1 2 3]

d = 6

n = 16

Salida:

El valor decimal a codificar es: 0.022024

El valor entero a codificar es: 22024

X = 0101011000001000

2.2 Decodificador Aritmético

```
function [V] = decodificacionAritmetica(B, P, d)
```

2.2.1 Ejemplo 1

Entrada:

B = 11110100

P = [.4 .3 .2 .1]

d = 3

Salida:

1 2 3

2.2.2 Ejemplo 2

Entrada:

B = 0101011000001000

P = [.1 .1 .2 .3 .3]

d = 6

Salida:

1 3 2 1 2 3

2.2.3 Ejemplo 3

Entrada:

B = 100011011111110000000

P = [.2 .4 .4]

d = 7

Salida:

1 2 3 3 3 2 2

3 Práctica 3: Codificación PCM

3.1 Cuantificador Uniforme

```
function [Scuan, Srecon]=PCMUniforme(S, nbits, v)
```

3.1.1 Ejemplo 1

Entrada:

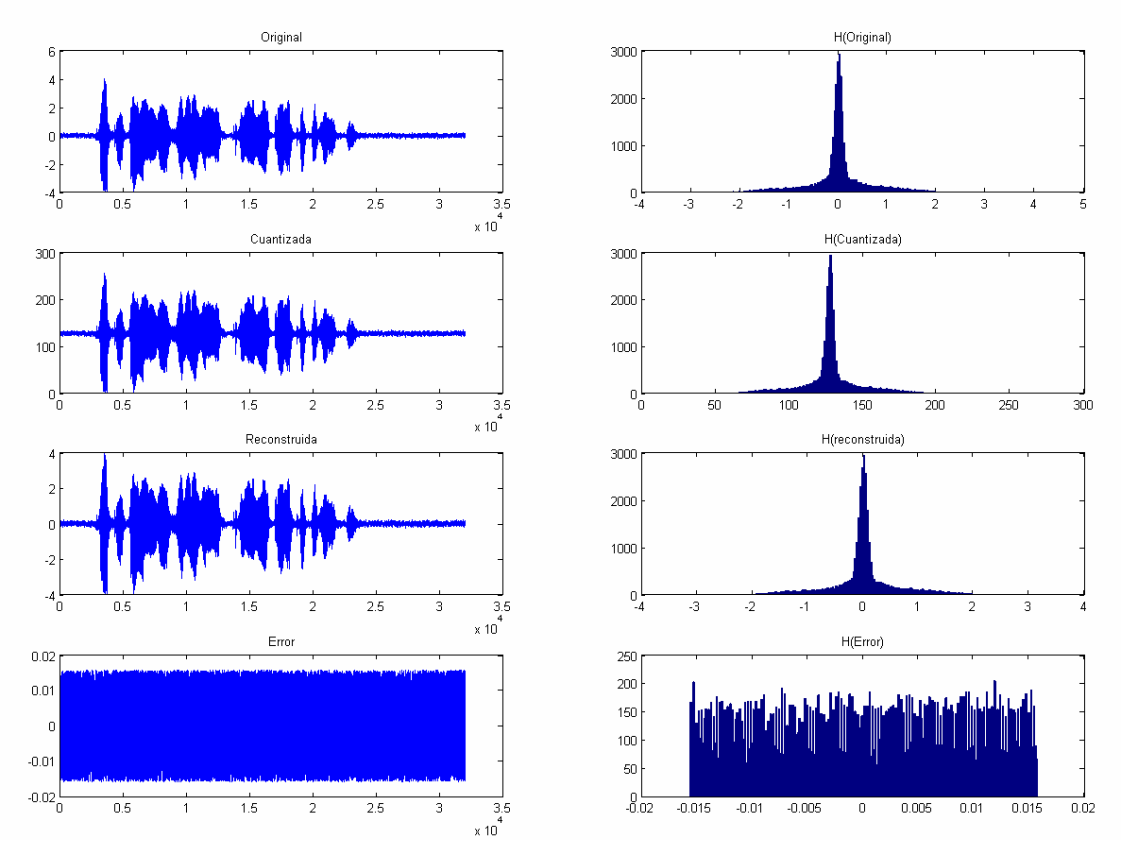
S = Secuencia de voz cargada ‘sample_audio.mat’

nbits = 8

v = 4

Salida:

Potencia de error: 2.6111



3.1.2 Ejemplo 2

Entrada:

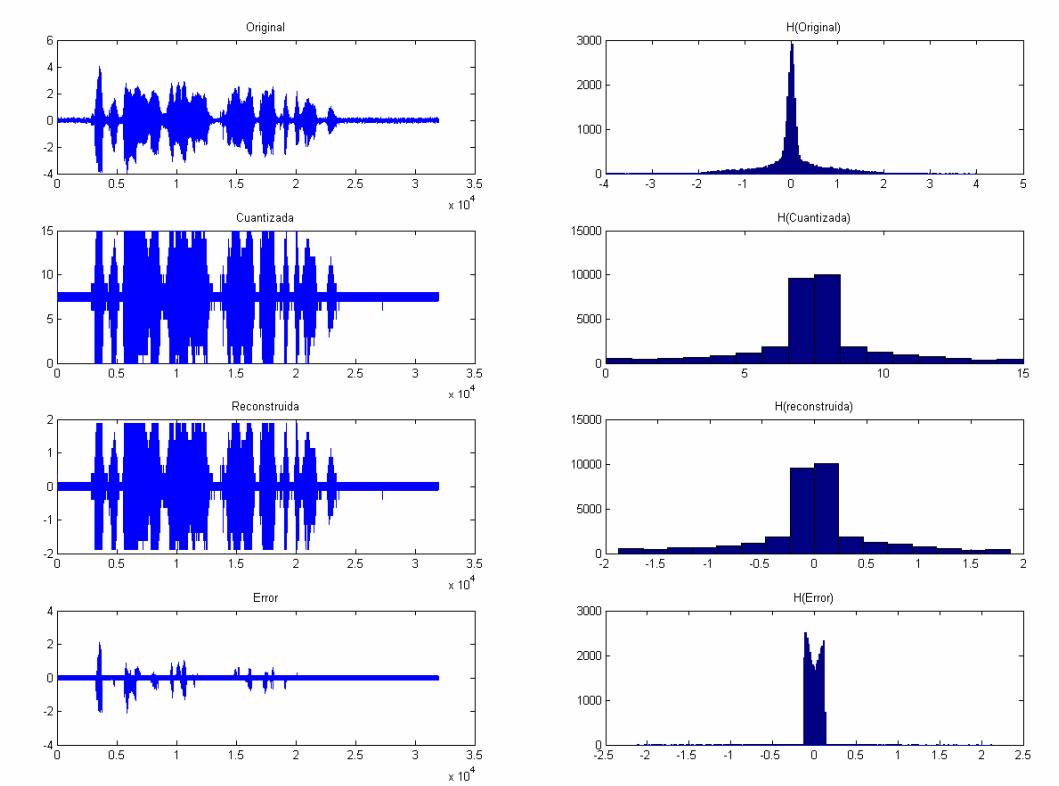
S = Secuencia de voz cargada ‘sample_audio.mat’

nbits = 4

v = 2

Salida:

Potencia de error: 480.7285



3.1.3 Ejemplo 3

Entrada:

S =[-3, -2.75, -2.5, -2.25, -2, -1.75, -1.5, -1.25, -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3]

nbits = 3

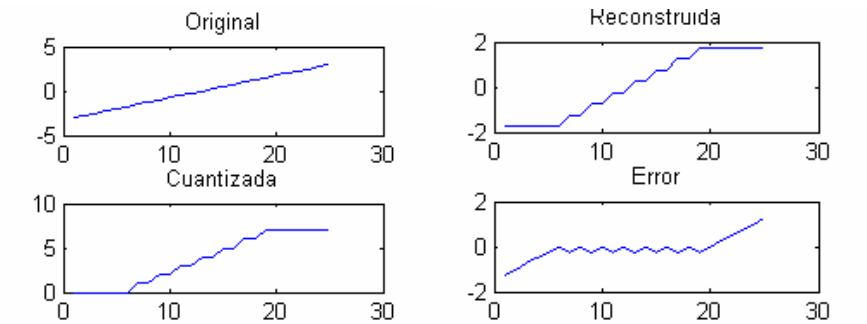
v = 2

Salida:

Scuan = 0 0 0 0 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 7 7 7 7 7

Srecon = -1.7500 -1.7500 -1.7500 -1.7500 -1.7500 -1.7500 -1.2500 -1.2500 -0.7500 -0.7500 -0.2500 -0.2500 0.2500 0.2500 0.7500 0.7500 1.2500 1.2500 1.7500 1.7500 1.7500 1.7500 1.7500 1.7500 1.7500

Perror = 7.3125



3.2 Cuantificador G711

```
function [Scuan, Srecon] = PCMG711(S)
```

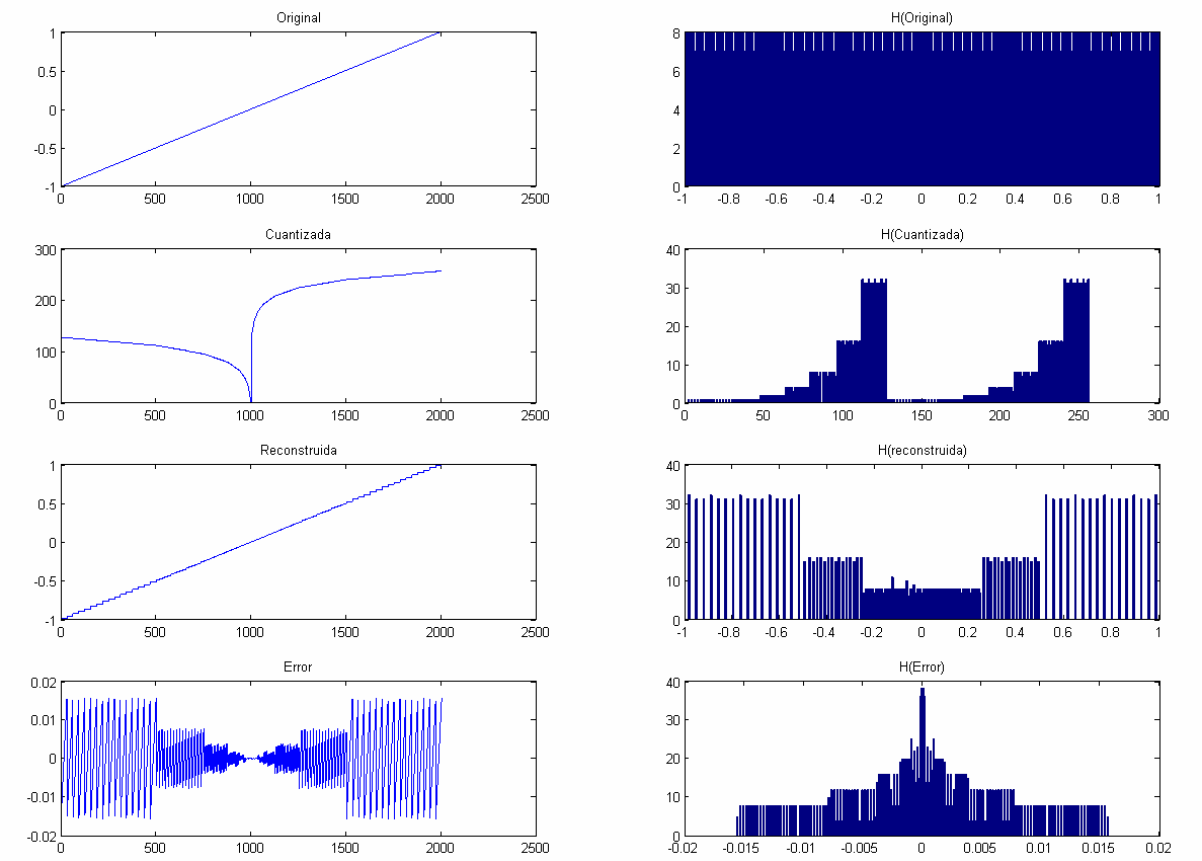
3.2.1 Ejemplo 1

Entrada:

S = [-1:0.001:1]

Salida:

Perror = 0.0935



Nota: *La grafica para el valor cuantizado de la señal se obtiene ploteando los resultados de aplicar la función bin2dec a cada muestra cuantizada.*

3.2.2 Ejemplo 2

Entrada:

S = [-1:0.1:1] = -1.0000 -0.9000 -0.8000 -0.7000 -0.6000 -0.5000 -0.4000 -0.3000 -0.2000 -0.1000 0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000

Salida:

Perror = 0.0014

Scuan= 127 124 121 118 115 112 105 99 89 73 128 201 217 227 233 240 243 246 249 252 255

Srecon= -0.9844 -0.8906 -0.7969 -0.7031 -0.6094 -0.5156 -0.3984 -0.3047 -0.1992 -0.0996 0.0002 0.0996 0.1992 0.3047 0.3984 0.5156 0.6094 0.7031 0.7969 0.8906 0.9844

3.2.3 Ejemplo 3

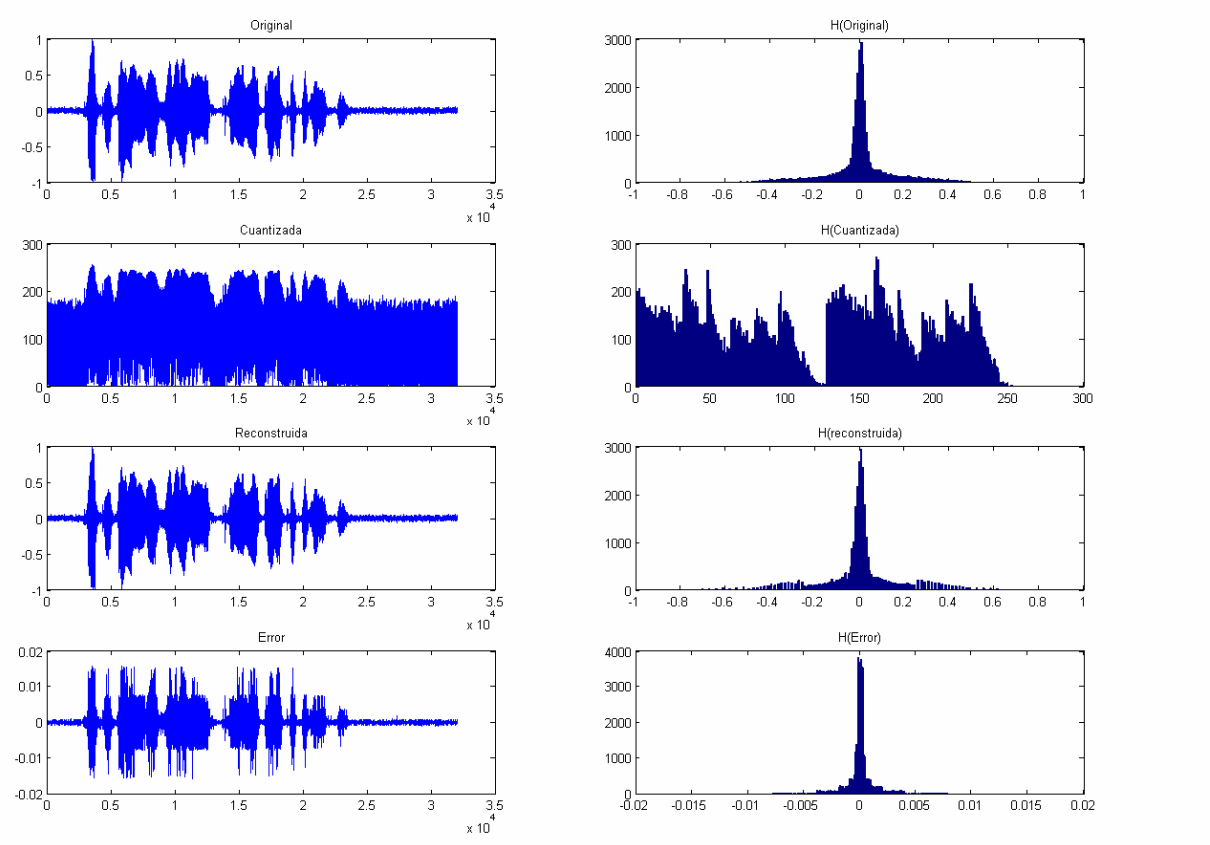
Entrada:

S = Señal de audio ‘sample_audio.mat’ normalizada (*S = leyendo(‘sample_audio.mat’); S = S./max(abs(S));*)

Nota: *Se recuerda que la funcion* `PCMG711` *espera un valor entre -1 y 1*

Salida:

Perror = 0.1536



Nota: *La grafica para el valor cuantizado de la señal se obtiene ploteando los resultados de aplicar la función bin2dec a cada muestra cuantizada.*