Guillermo Aldrey Pastor 4ºA GITT

Práctica 1 - Digitalización

Fecha de Entrega : 04/10/2020

Índice

Descripción

La práctica trata del primer tema de la asignatura, “digitalización y comprensión”.

Consta de tres partes en las que se va a trabajar con lo aprendido en clase sobre PCM (Pulse Code Modulation), discretización diferencial DPCM, y ADPCM (Adaptive Differential Pulse Code Modulation).

En este informe se reflejan los comentarios sobre el código llevado a cabo en los scripts de matlab y se adjuntan imágenes para comprender y analizar los resultados.

Objetivo

Conocer de manera aplicada el proceso de digitalización de una señal analógica con diferentes métodos de digitalización tales como PCM, ley mu y ley A.

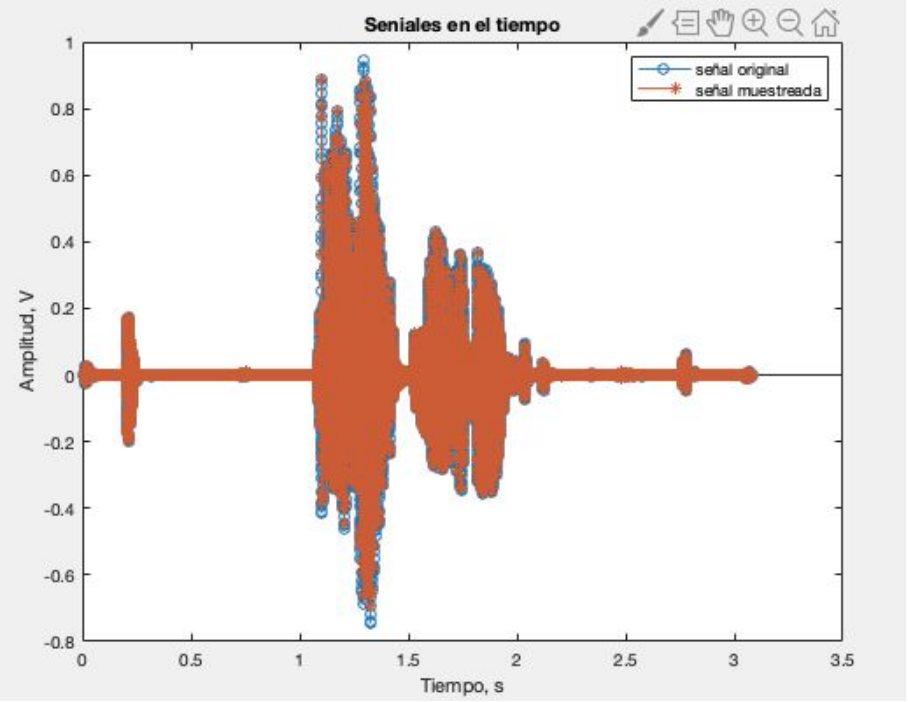
Parte A

En este apartado vamos a usar PCM , método para representar de forma digital señales analógicas como el audio.

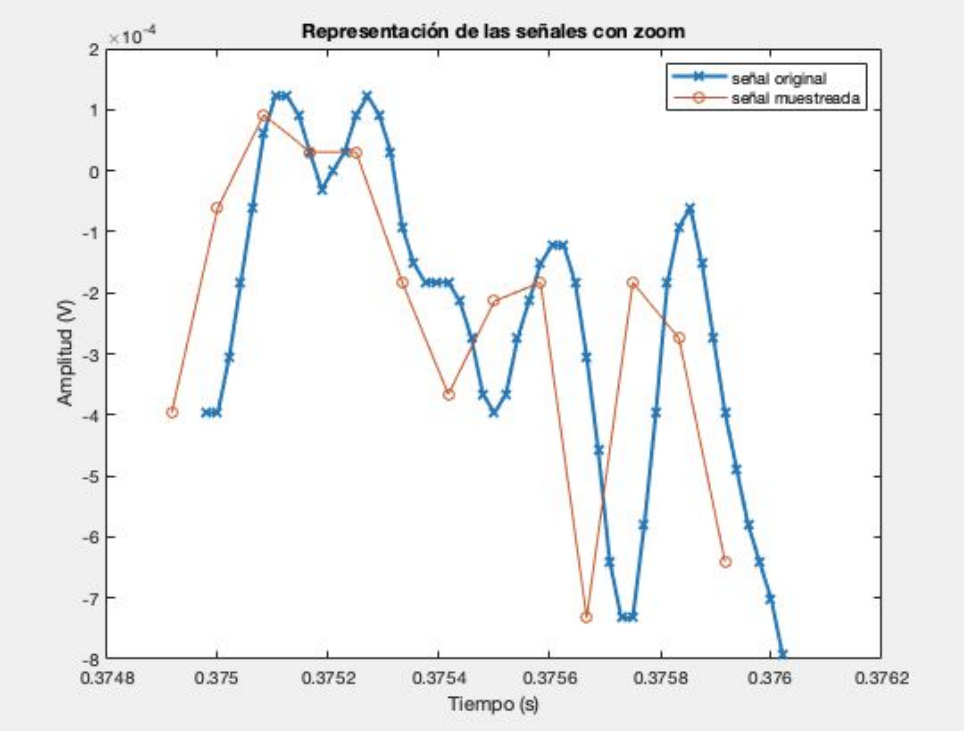
a) Usando el comando **audioread** obtengo tanto mi señal m como la frecuencia de muestreo. La señal, que proviene de un archivo de audio que he grabado y he convertido a formato wav, ya es una señal digital, es decir, ya está muestreada.

   Lo que voy a hacer entonces es muestrearla de nuevo, usando un diezmador. Al igual que en el ejemplo “cuantificacion_uniforme_1”, lo que estamos haciendo es tomar una muestra de cada M, y en este caso, M será 4, por lo tanto tomamos una muestra de cada 4.

A continuación represento tanto mi señal original **m**, como mi señal muestreada con el diezmador, **m_samp**.



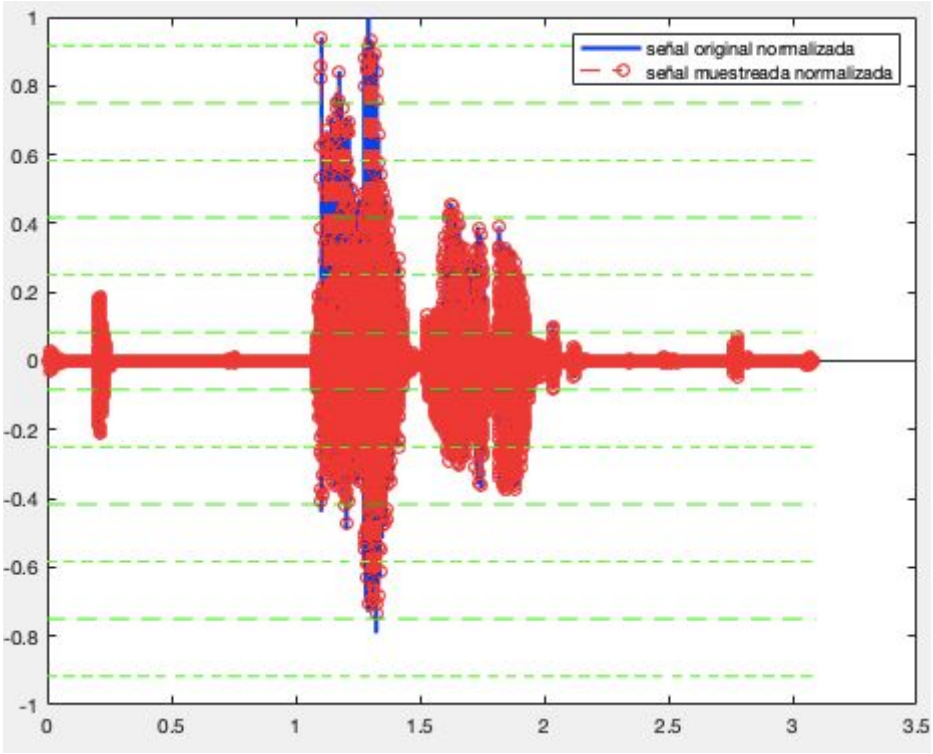
Aplico zoom para observar mejor que el proceso de diezmado se ha realizado de manera correcta.



b) A continuación verificaré la presencia de los codewords asignados

Para ello, definiremos L (número de niveles de discretización) como 12, y utilizamos la función otorgada por el profesor **uniform_pcm**.

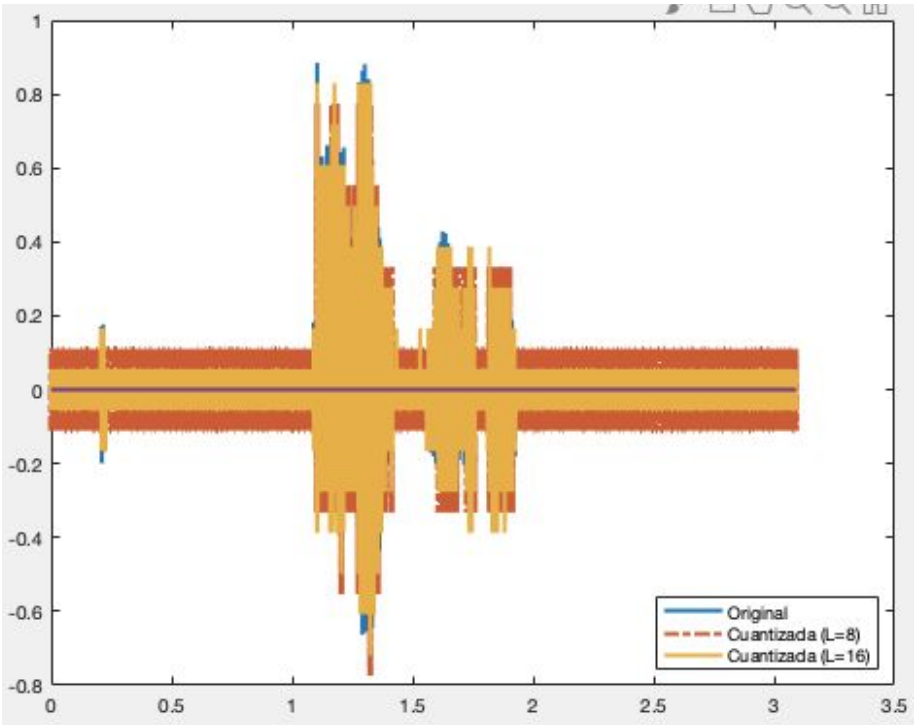
Se normalizan las señales.



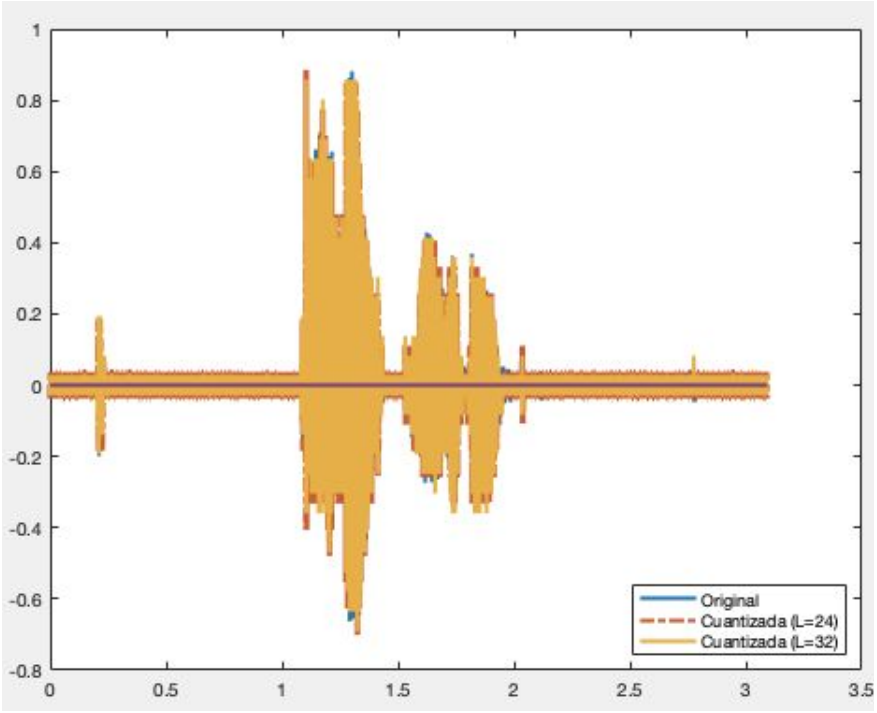
Llegados a este punto comprobamos con **sound** como se escucha nuestra señal discretizada y nos damos cuenta de que se escucha un poco peor que la señal original. Esto se debe a la comprensión que hemos llevado a cabo.

c) En este apartado vamos a ir variando el valor de L, siguiendo :
L=2i-1, donde i=3,4...8

Una vez llevada a cabo la función de PCM con cada valor de L, representaremos las respectivas salidas de la función para comprobar cómo cambian las señales con respecto a L.



En esta imagen podemos apreciar la señal original junto con las señales cuantificadas con 8 y 16 valores. Se puede ver como la cuantificación no ha sido del todo precisa , por lo tanto seguimos subiendo el valor de L.



Ahora, con L igual a 24 y 32, los escalones no son tan grandes cuando la señal tiene amplitud nula. Esto quiere decir que la precisión es mejor.

Como conclusión a este apartado, podemos decir que según aumenta L, el error SNR de la señal también aumenta y la precisión de la cuantificación es mejor.

Además, hay que mencionar que hemos imprimido por pantalla los 5 primeros valores de la señal original, la señal cuantificada y el code. Con esto hemos comprobado cómo los valores originales son todos cero y los códigos de los niveles corresponden a la mitad, lo cual tiene sentido.

d) Una vez hicimos la primera discretización comprobamos como nuestro **audio.wav** se escuchaba algo peor que al principio. Pero según vamos incrementando el número de niveles de cuantificación se escucha un poco mejor, concordando con lo que hemos visto en el apartado anterior respecto al error.

Parte B

En este apartado trabajaremos con discretización perceptual (μ-Law), basada en una función logarítmica capaz de asociar valores de intensidad de voz sobre N niveles de discretización; y también debemos desarrollar dos funciones del tipo A-LAW.

a) Desarrollo A-LAW

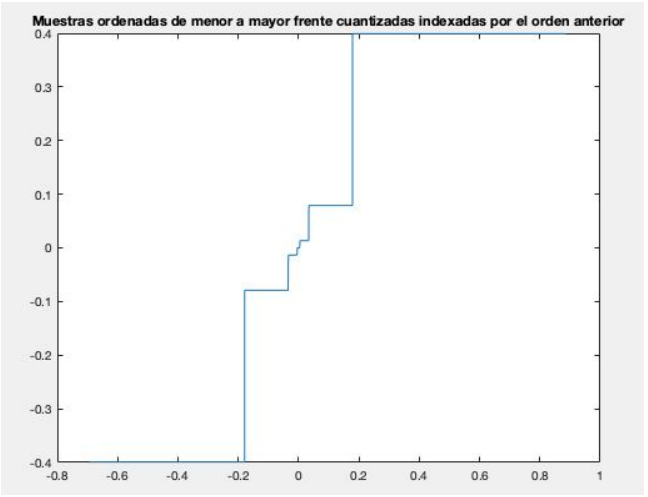
```
function [y,m_max]=alaw(m_samp,A)

m_max = max(abs(m_samp));
y = zeros(size(m_samp,1),1);
for i =1:size(m_samp,1)

    if abs(m_samp(i)/m_max) < (1/A)
        y(i)    = ((A*abs(m_samp(i)/m_max))/(1+log(A)))*signum(m_samp(i));
    else
        y(i)    =
((1+log(A*abs(m_samp(i)/m_max)))/(1+log(A)))*signum(m_samp(i));
    end
end
```

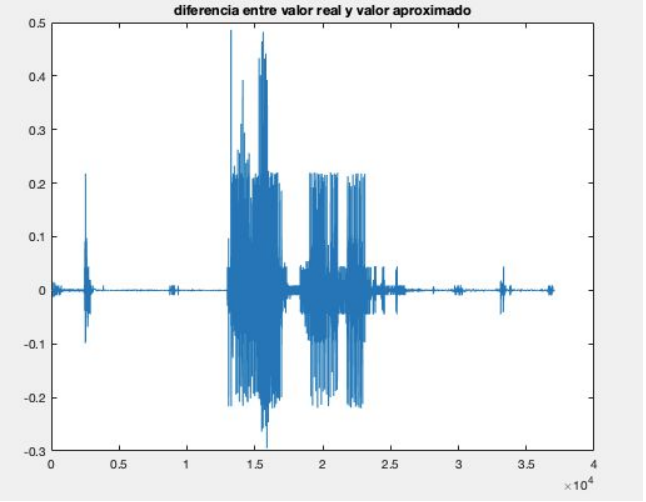
```
% Funcion A-LAW

function [sqnr,m_quan,code]=alaw_pcm(m_samp,L,A)

% Call the alaw function:
[y,m_max]=alaw(m_samp,A);

% Call the uniform_pcm function:
[sqnr,m_quan,code]=uniform_pcm(y,L);

% Call the invalaw function:
x = invalaw(m_quan,A);

m_quan = m_max*x;
sqnr = 20*log10(norm(m_samp)/norm(m_samp - m_quan));
```

b) Vamos a reutilizar los casos de digitalización no-uniforme utilizando tanto la ley mu como la ley A, para comprobar las diferencias.

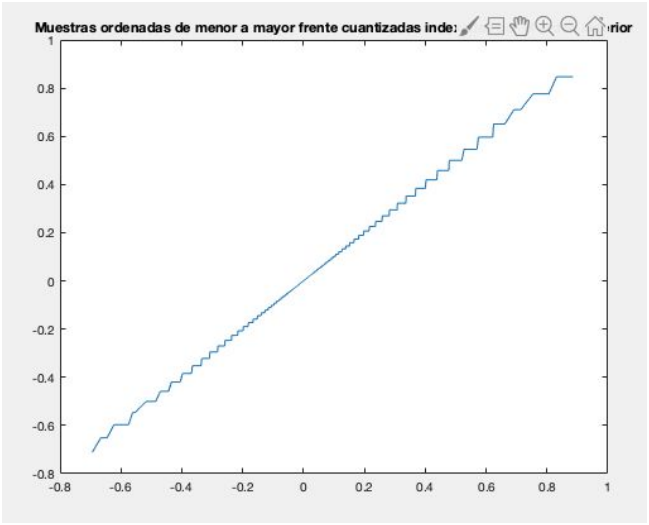
- Ley mu → mu = 255

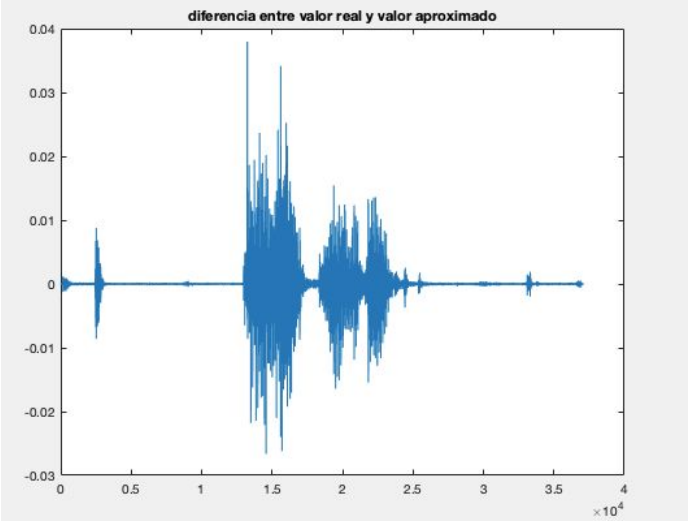
  Utilizando este valor, L va a ir incrementando su valor hasta llegar a 255. Por ejemplo, mostramos tanto el primer valor de L como el último:





Aquí observamos una mayor diferencia entre el valor real y el valor aproximado, y por lo tanto peor codificación.

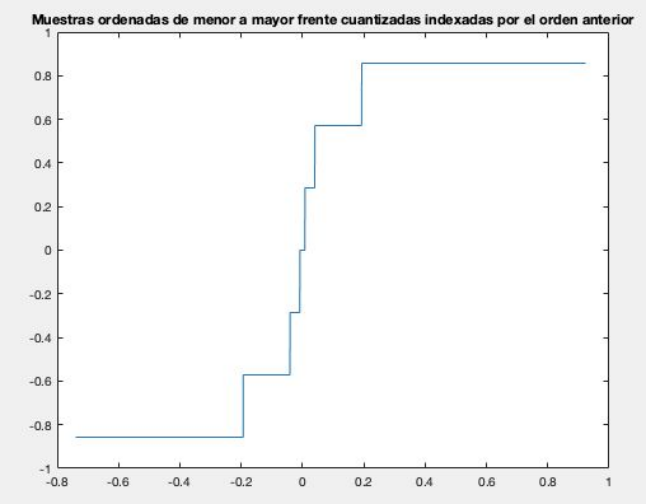


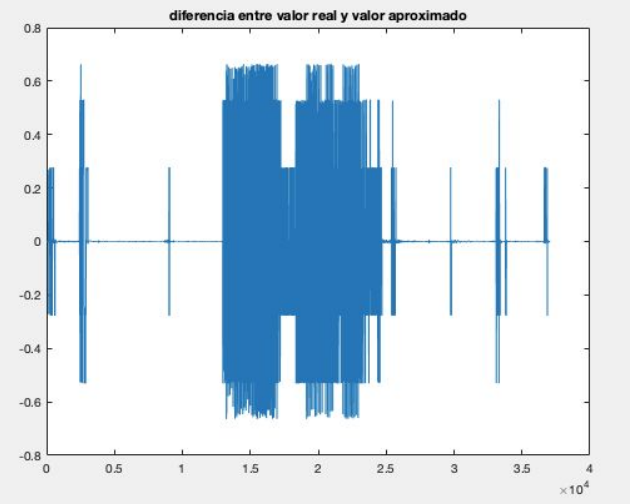


Aquí, siendo L=255, la codificación es casi perfecta, y se refleja en cómo la diferencia entre el valor real y aproximado es casi nula.

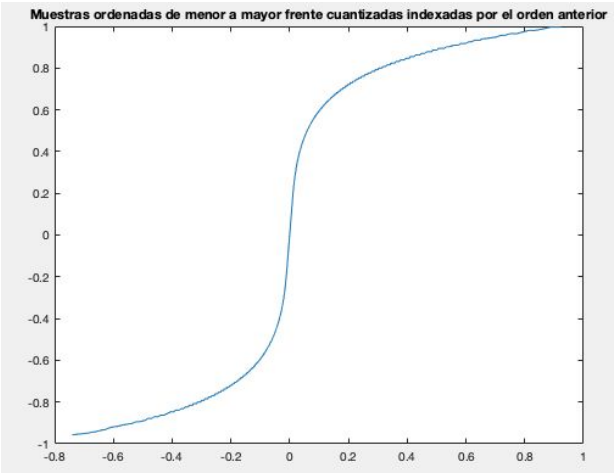
- Ley A → A = 87.7

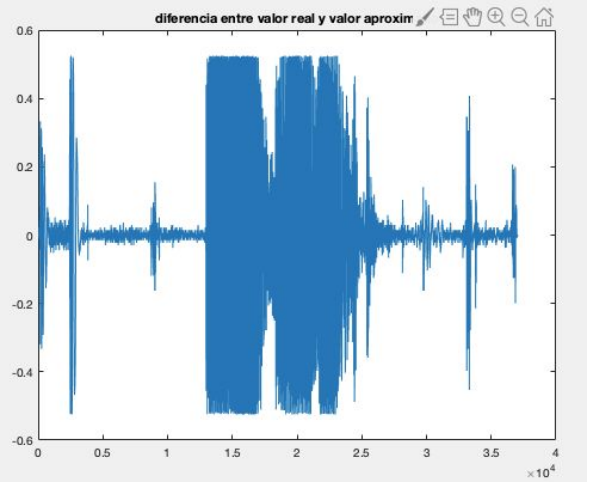
  Repetimos el resultado utilizando la ley A que hemos diseñado en el apartado a. En este caso se repite el hecho de que la codificación mejora según aumenta el valor de L, pero sigue siendo una mala codificación, a diferencia de lo que pasaba con la ley mu.





Aquí vemos como con L=7, la codificación no es buena, como pasaba antes.





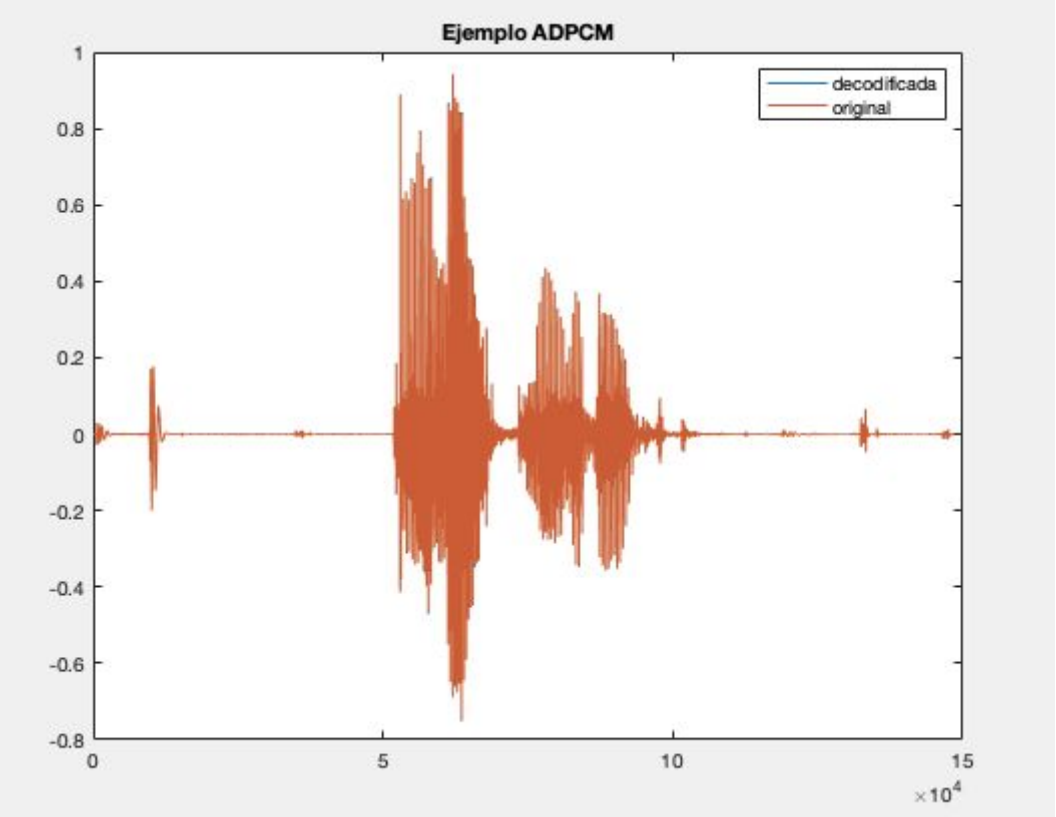
Y aquí, con L=255, esta ha mejorado si,pero con la ley mu era mucho más precisa. He comprobado escuchando el audio y para este caso no se escucha muy bien, algo que con la ley mu si sucedía.

Parte C

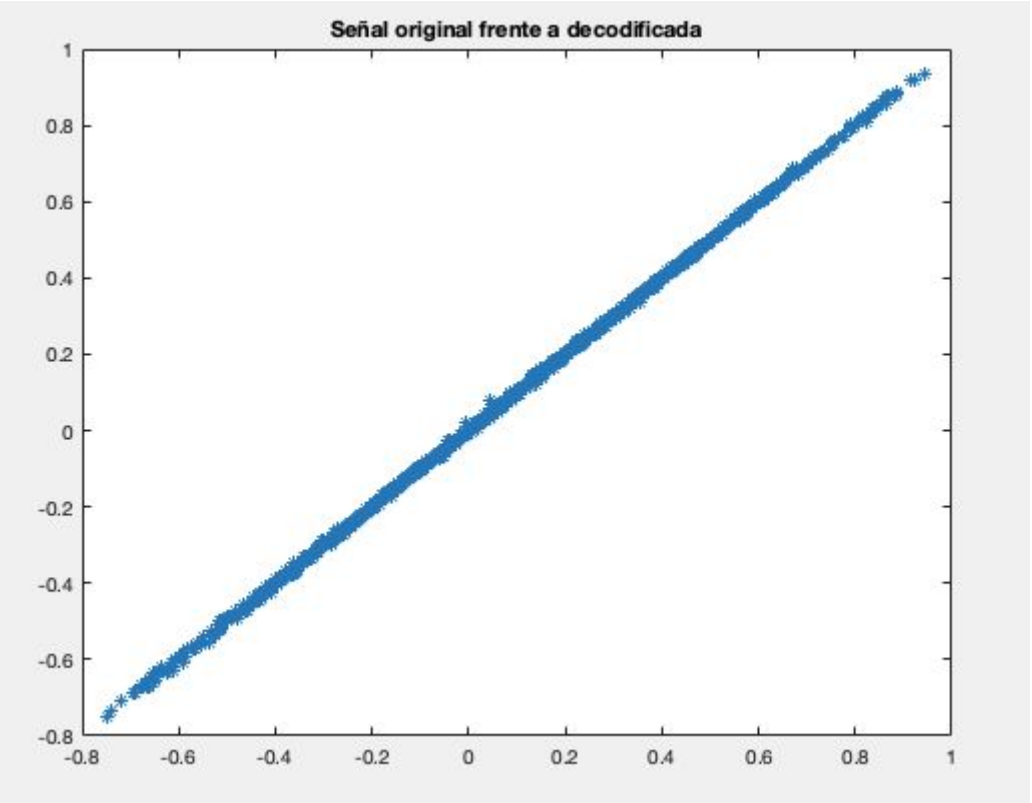
En este apartado vamos a volver a estudiar nuestra grabación de audio, utilizando tanto discretización diferencial **DPCM**, como Adaptive Differential Pulse Code Modulation **ADPCM.**

En ambos casos se ha reutilizado el código otorgado por el profesor.

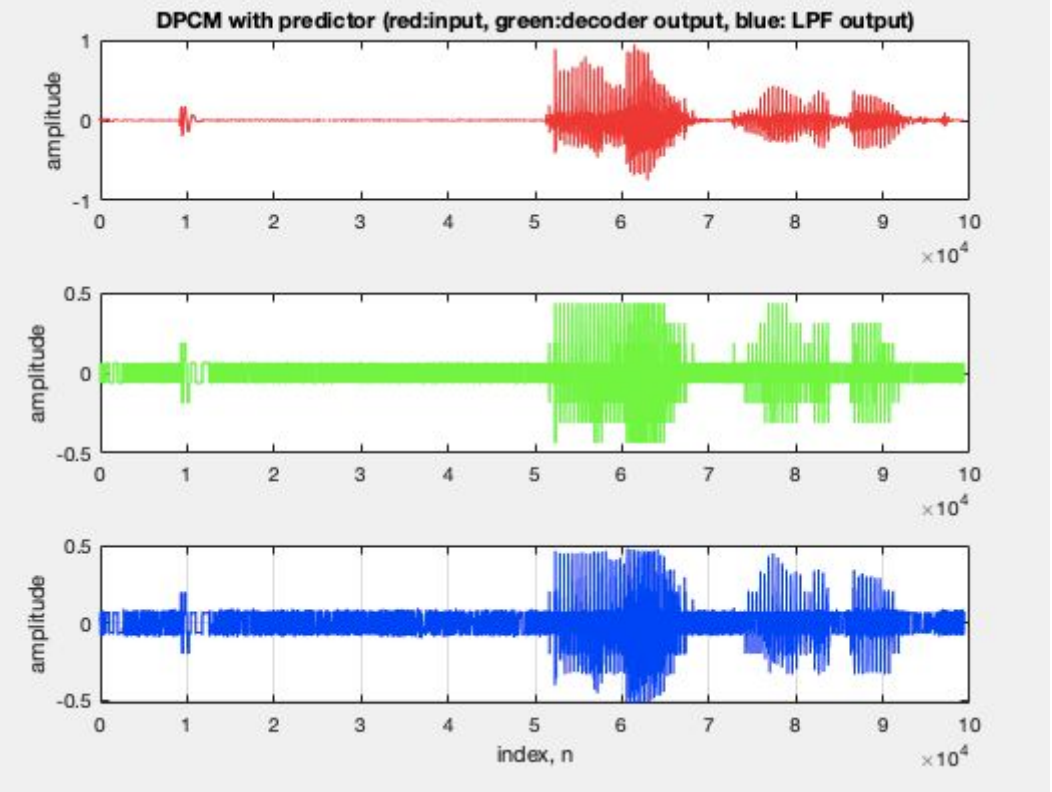
1. ADPCM



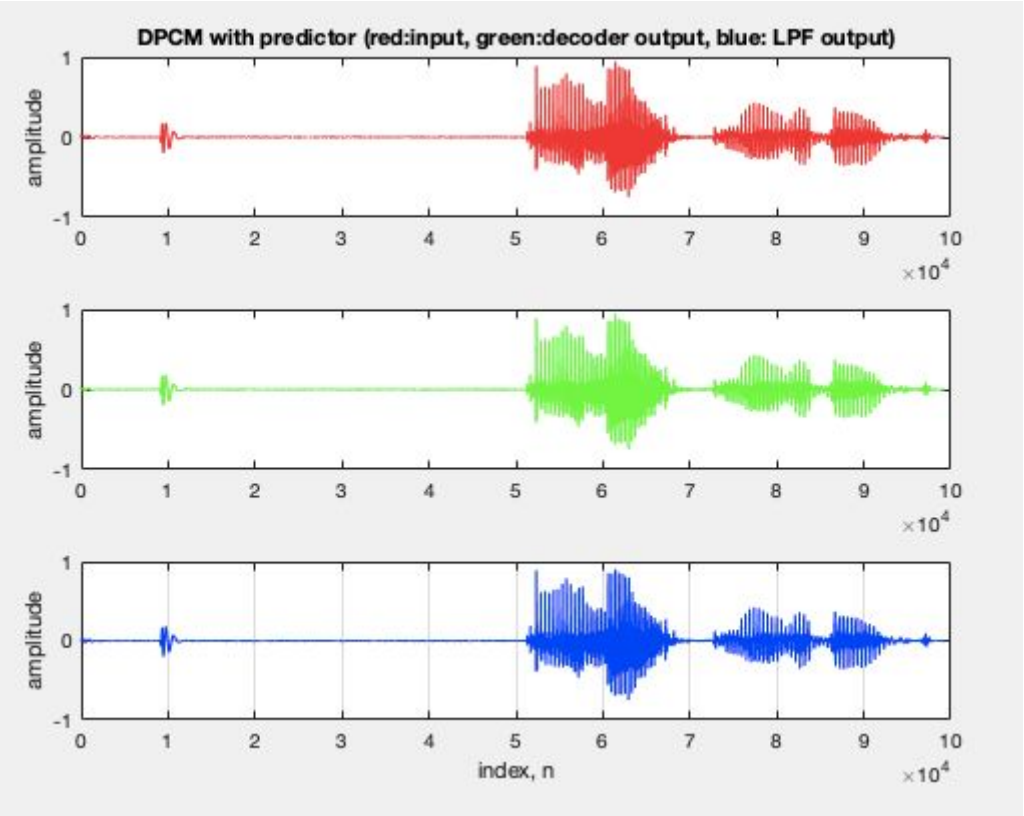
Como se puede observar, la codificación es bastante buena, y la señal decodificada coincide con la original. Lo comprobamos en la siguiente imagen que muestra la señal original frente a la señal codificada.



2. DPCM



Ejemplo de DPCM con bitsize = 3



Ejemplo de DPCM con bitsize = 10

Como podemos observar, la señal está mejor decodificada cuando se usa un mayor bitsize. De hecho, al comprobar cómo se escuchan tanto la original como la codificada con **sound**, se aprecia una notable diferencia entre el caso en el que el bitsize es 3, donde existe una gran cantidad de ruido, y el caso en el que el bitsize es 10, donde la salida de la codificación se asemeja mucho más a la señal original.

Conclusiones

Me ha parecido una práctica muy interesante debido a que todo lo que hemos ido estudiando durante el tiempo que lleva la asignatura ha ido saliendo a lo largo de la misma.

Los ejercicios que hemos entregado durante estos días me han ayudado para enfrentarme a la práctica y entender perfectamente lo que estaba haciendo.