UNIVERSIDAD DE GRANADA

Procesamiento Digital de Señales

Práctica 1 – Cuantización de Señales

Arturo Alonso Carbonero

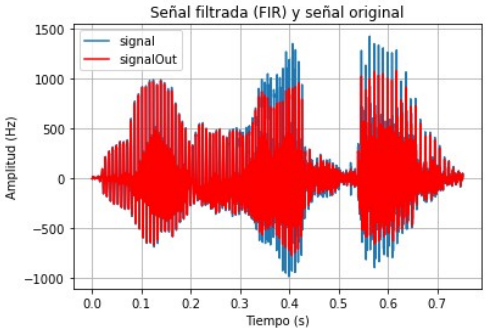
1. Objetivos de la práctica

El objetivo principal de la práctica es realizar ejercicios relacionados con la cuantización de señales, en Python, con el objetivo de comprender en qué consiste y cómo se lleva a cabo. En este caso, se ha realizado la implementación en Python de los bloques (si se tratara de Simulink) que componen un sistema que cuantiza una señal de entrada de diferentes maneras (cuantización uniforme o ley μ), así como la interpretación de los resultados de forma gráfica, numérica (con la medida de la SNR) y en forma de audio.

2. Resultados obtenidos

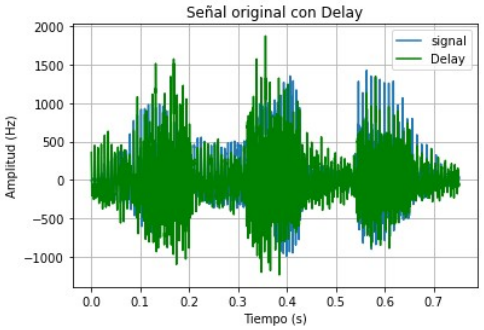
Tarea 3.1 – Filtro FIR

Para aplicar el filtro FIR y obtener la señal resultado, la cual se muestra en rojo en la imagen, se calcula la media de las tres últimas medidas de la señal original (comenzando por la segunda). El filtro se le ha aplicado a la señal de audio ‘muneca.wav’ de la práctica, la cual se muestra en azul para comparar ambas señales.



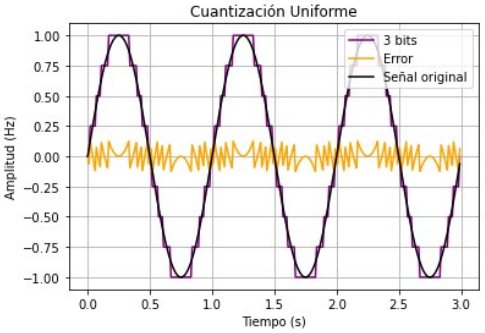
Tarea 3.2 – Efecto *Delay*

Para aplicar un efecto de *delay* a la señal (‘muneca.wav’) he realizado la implementación del sistema indicado en el guion de la práctica teniendo en cuenta el tamaño de la señal resultado. El resultado se muestra en la imagen (en verde) y, al ejecutar el código del ejercicio, genera un archivo ‘.wav’ con el mismo. Dicho audio es una versión de la señal original con algo de eco, el cual se puede modificar para que sea más o menos notable.



Tarea 4.1 – Cuantización uniforme

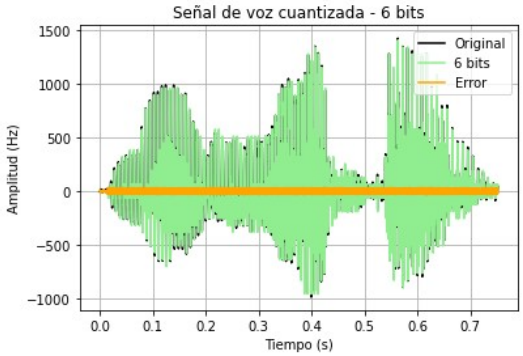
El resultado de la imagen (en morado) es el resultado de aplicar una cuantización uniforme de 3 bits a la señal original que, en este caso, se trata de una señal sinusoidal generada manualmente. En naranja se muestra el error obtenido. La forma cuadrada de la señal surge porque para cada valor que se encuentre por debajo (o por encima) de delta (delta, 2delta, 3delta, etc), se escoge el inmediatamente inferior (o superior).



Cuantización uniforme – 3 bits

Tarea 4.2 – Cuantización de señal de voz

En este caso, la señal original es la del audio de la práctica mencionada anteriormente, a la cual se le aplica una re-cuantización (de 4, 6 y 8bits) basada en el esquema del guion (función ‘quantize(signal, delta)’ en el código del ejercicio). El resultado depende de los bits utilizados, obteniendo una señal de audio con más ruido cuanto menor sea dicho número (y menor SNR).

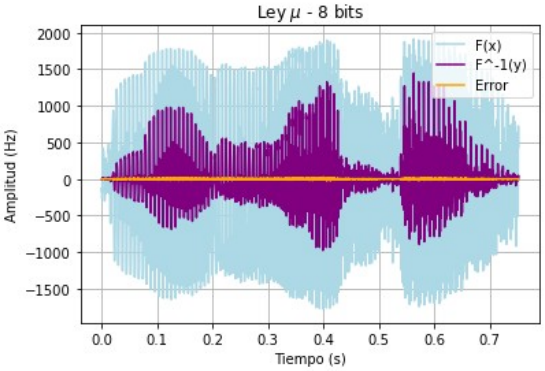


Cuantización – 6 bits

```
SNR teórica para 8 bits -> 49.92571189679381 db
SNR para 8 bits -> 36.063479139910406 db
SNR teórica para 6 bits -> 37.88451207023456 db
SNR para 6 bits -> 23.70771388883538 db
SNR teórica para 4 bits -> 25.84331224367531 db
SNR para 4 bits -> 12.199344265476542 db
```

Tarea 4.3 – Cuantización mediante Ley μ

La cuantización en este caso es no lineal. La implementación se basa en las fórmulas del guion de la práctica. En cuanto a los resultados, obtendremos una mejor calidad que en el apartado anterior para el mismo número de bits. Obtendremos SNR mayores (como se muestra en la imagen) y mejor calidad de audio, aunque para este caso no son cambios significativos.



Cuantización Ley μ – 8 bits

```
SNR teórica para 8 bits -> 49.92571189679381 db
SNR para 8 bits -> 37.865621167102994 db
SNR teórica para 6 bits -> 37.88451207023456 db
SNR para 6 bits -> 25.837900726404417 db
SNR teórica para 4 bits -> 25.84331224367531 db
SNR para 4 bits -> 13.853002298459664 db
```

Reflexión

La práctica en general no cuenta con las características necesarias para generar complicaciones extremas, pero no es trivial (sobre todo las últimas tareas). Tras realizarla, se obtiene un conocimiento práctico sobre lo impartido en teoría, haciendo así que la comprensión sea mayor. Al poder observar los resultados obtenidos con varios ejemplos es, en general, sencilla de comprender.

Cabe destacar que, a la hora de implementar los esquemas de Simulink en Python, es decir, en código, es factible encontrarse con algún problema de compresión de los mismos a la hora de construir las funciones pertinentes. Sin embargo, una vez interpretados dichos esquemas correctamente, esto deja de ser un problema.

Como aclaración decir que, el resto de resultados, las funciones implementadas, los procedimientos seguidos y demás contenido relacionado con las tareas se encuentran en el anexo de la entrega dentro del propio código de cada ejercicio. Para no aumentar considerablemente el tamaño de la entrega, es necesario ejecutar los ejercicios para poder ver esos resultados.