VNiVERSiDAD
D SALAMANCA

PRÁCTICAS DE SEÑALES Y SISTEMAS - GRADO EN INGENIERÍA INFORMÁTICA

SESIÓN 5: Simulación de sistemas PCM

En esta sesión vamos a trabajar con sistemas PCM aplicados a señales de audio. En la primera parte vamos a trabajar con una señal de audio creada artificialmente de forma algo rudimentaria. En la segunda, trabajaremos un fichero de audio real.

Ejercicio 1:

a) Abrir el fichero en MATLAB titulado "Ejercicio 1". La primera parte del ejercicio consiste en crear una señal compuesta por tres armónicos. Estos tres armónicos corresponden a los tres armónicos de la nota La4 del piano, cuya frecuencia fundamental es, por convenio, 442Hz. Mediante el dibujo de la señal en el dominio del tiempo (`plot(t,y)`), podemos señalar el **periodo de la señal** completa, así como los **niveles de voltaje máximo y mínimo**.

b) Utilizando la función `espectro()` como ya hemos visto, podemos representar la función en el dominio de la frecuencia. Presentar dicho espectro en amplitud, identificando el **ancho de banda de la señal** y las **frecuencias y amplitudes** de los tres armónicos principales, comprobando que efectivamente, la frecuencia principal corresponde a un La.

Ejercicio 2:

En este ejercicio, vamos a diseñar un sistema PCM.

a) Teóricamente, deberíamos muestrear la señal con algún sistema. En este caso, puesto que tenemos que dibujar la señal de alguna manera, ya estamos muestreando la señal al darle valores al vector `t` y dibujar el valor de `y`. Podemos observar lo que ocurre con la señal en el dominio del tiempo y el de la frecuencia si cambiamos el incremento del vector `t`. (Utilizar **frecuencias de muestreo** superiores e inferiores a la frecuencia de Nyquist).

b) El siguiente paso es **cuantizar** la señal. Para ello, tenemos una función auxiliar: `[q,index]=cuantizar(y,vmin,vmax,m)`, donde `y` es la señal muestreada, `vmin` y `vmax` son los niveles mínimo y máximo de voltaje, respectivamente, y `m` es el número de bits de codificación para los niveles (grupos de `m` pulsos que aparecerán en la señal codificada). Por ejemplo, si queremos 4 niveles, el valor de `m` sería 2, porque 2^m=4 niveles. Esta función devuelve dos vectores: en *q* la señal cuantizada y en *index* el número del nivel de cuantización utilizado en cada periodo. Presentar la señal que resulta en el dominio del tiempo y observar el ruido

de cuantización cuando se considera un número de niveles de cuantización muy bajo. Probar con diferente número de niveles de cuantización.

Considerar como **límites de cuantización** los valores máximo y mínimo de la señal original en el dominio del tiempo. ¿Qué ocurre si estos valores no son los adecuados?

c) Ahora vamos a codificar y decodificar la señal. Para codificar utilizamos la función `c = codificar(index, m, b)`, donde `index` es una de las salidas de la función cuantizar, 2^n son los niveles de cuantización como se explicó en a) y `b` es el tipo de codificación (binaria, ternaria o cuaternaria). Para decodificar, utilizamos `q= decodificar(c,m,b,vmin,vmax)`, donde c es la señal codificada. Entre estas dos etapas se encontraría el medio de transmisión, que en este caso se considera que no produce ningún efecto sobre la señal. Observar la señal PCM en el dominio del tiempo, para distintos tipos de codificación (binaria, ternaria, cuaternaria).

d) Añadir un **filtro pasabaja**, utilizando la función `lowpass(q,f,fs,…)` donde `q` es la señal decodificada, `f` es la frecuencia de corte y `fs` la frecuencia de muestreo. Cambia los valores de la frecuencia `f` con valores por encima y por debajo del ancho de banda, y describe lo que pasa en el dominio del tiempo y de la frecuencia.

Ejercicio 3:

Con un sistema PCM, podemos transmitir una señal de audio a través de un canal. En este ejercicio vamos a simular este proceso con un fichero que podemos encontrar en la carpeta auxiliar llamado “violinA.wav” y que contiene la grabación de un sonido del violín tocando la misma nota que hemos simulado anteriormente, un La3, una octava inferior a la vista en el ejercicio anterior, con una frecuencia fundamental alrededor de los 220Hz. Hay varios aspectos a tener en cuenta a la hora de trabajar con audio “real”:

1) Normalmente los ficheros de audio están grabados en estéreo, lo que significa que tenemos dos canales. Para este ejercicio, nos quedaremos con el canal 1.

2) Los ficheros de audio son digitales, así que la señal tiene que estar obligatoriamente muestreada. Normalmente en las propiedades de los ficheros suelen indicar cuál es la frecuencia de muestreo. En este caso, podemos recuperar esta información mediante la función `audioread('violinA.wav')`, que devuelve la señal muestreada y la frecuencia de muestreo.

3) Podemos escuchar la señal en todo momento, incluso una vez cuantizada, filtrada o decodificada, simplemente utilizando la función `sound(y,fs)`, donde `y` es la señal y `fs` es la frecuencia de muestreo que nos ha devuelto `audioread`.

a) Antes de construir el sistema PCM, vamos a mostrar la señal en el dominio del tiempo y en el dominio de la frecuencia. En el dominio del tiempo vamos a identificar los niveles de voltaje máximo y mínimo. Para calcular el espectro, utilizaremos la función auxiliar `espectro_s(y,fs)`, donde `y` es el audio y `fs` es la frecuencia de muestreo. Presentar el espectro en amplitud, identificando el **ancho de banda de la señal** y las **frecuencias y amplitudes** de los tres armónicos

principales, comprobando que efectivamente, la frecuencia principal corresponde a un La. Con esta información, comprobar mediante el teorema de Nyquist que la señal en violinA.wav está bien muestreada.

b) Por último, volveremos a construir el mismo sistema que anteriormente. Escucha los sonidos que se producen en cada fase, para comprobar qué pasa físicamente con la señal. Prueba al menos con tres niveles y escucha el ruido de cuantización en el audio. Prueba con varias frecuencias de corte del filtro pasabaja y escucha cómo se distorsiona la señal de audio.