VNiVERSiDAD D SALAMANCA

PRÁCTICAS DE SEÑALES Y SISTEMAS - GRADO EN INGENIERÍA INFORMÁTICA

SESIÓN 5: Simulación de sistemas PCM

Ejercicio 1:

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.

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, 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. 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).

a) El siguiente paso es **cuantizar** la señal. Para ello, tenemos una función auxiliar: `cuantizar(y,vmin,vmax,n)`, donde `y` es la señal muestreada, `vmin` y `vmax` son los niveles mínimo y máximo de voltaje, respectivamente, y `n` es el número de bits de codificación para los niveles. De esta manera, si queremos 4 niveles, el valor de n sería 2, porque 2^n=4 niveles. Esta función devuelve dos parámetros: la señal cuantizada y los niveles de cuantización de la señal. 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?

b) Ahora vamos a codificar y decodificar la señal. Para codificar utilizamos la función `codificar(index, n, b)`, donde `index` es una de las salidas de la función cuantizar, `n` son los niveles de cuantización y `b` es el tipo de codifcación (binaria, ternaria o cuaternaria). Para decodificar, utilizamos `decodificar(c,n,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).

c) Añadir un **filtro pasabaja**, utilizando la función `lowpass(q,f,fs,…)` donde `q` es la señal decodificada, `f` es la frecuencia de paso 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 La4. 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 dicho espectro, 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.

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.

Ejercicio 4:

En este ejercicio, vamos a trabajar con espectrogramas para una señal muy sencilla, un chirp que va desde 220Hz hasta 880Hz con una duración de 2 segundos. Vamos a utilizar la función spectrogram, cuyo prototipo principal es:

```
spectogram(señal,ventana,samples_overlapping);
```

`señal`: Nuestra señal muestreada en el dominio del tiempo

`ventana`: función de ventaneo

`samples_overlapping`: valores que se van a repetir en cada STFT calculada

d) Completa el código para generar el chirp sinusoidal y escúchalo con la función sound. Dibuja la señal en el dominio del tiempo y el espectro en amplitud.

e) Dibuja el espectrograma del chirp. El único argumento obligatorio es el primero, la señal muestreada en el dominio del tiempo. Si no le indicamos más, `spectrogram` aplicará una ventana de tipo hamming de tamaño N/8 (N es la longitud de la señal muestreada) con una superposición (overlapping) del 50%.

f) Compara los espectrogramas aplicando una ventana hamming frente a una normal (rectangular), para un tamaño relativamente pequeño pero estable (el tamaño de la ventana, dado por L, lo hemos puesto a 600 muestras, pero podéis cambiarlo). La función `hamming(L)` dada por MATLAB nos crea una ventana haming de tamaño L. La función `rectwin(L)`, nos creará una ventana rectangular de tamaño L. ¿Aprecias alguna diferencia entre ambas? ¿Cuáles? ¿Por qué ocurre eso?

g) Vamos a quedarnos con la ventana hamming, la más utilizada por sus ventajas que ya comentamos en clase. En este apartado, cambia el tamaño de la ventana, utiliza uno muy grande y uno muy pequeño (solo hay que cambiar la variable `L`). ¿Qué diferencias aprecias? A la vista de los resultados, ¿cuál crees que es más adecuado? ¿Por qué?

h) Ahora quédate con un tamaño de ventana fijo, vamos a modificar el overlapping. Para que se vea mejor, vamos a utilizar un tamaño un poquito más grande, por ejemplo, 2000. El tamaño de la superposición en el espectrograma se mide en número de muestras, y siempre debe ser más pequeño que el tamaño de la ventana. Para ayudarnos, os aconsejo multiplicar el tamaño de la ventana (L) por un número que vaya entre 0 y 1. Elegid dos números, uno pequeño y otro grande. ¿Crees, a la vista de los resultados, que el overlapping es importante? ¿Por qué?