UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO
Facultad de Ingeniería Eléctrica

Laboratorio de Procesamiento Digital de Señales

PRÁCTICA 2

"Señales de Audio y el Aliasing"

Objetivo

Implementar funciones básicas para el manejo de señales digitales de audio, usando los sonidos demo de Matlab, y/o archivos .wav

ANTECEDENTES

Señales de audio

Muchos de los sonidos que encontramos todos los días son generados por eventos físicos que involucran una interacción entre objetos (una moneda que se cae al piso, gotas de agua que caen sobre el fregadero) o un cambio de las propiedades de un objeto (un globo que estalla). ¿Pueden los oyentes recuperar las propiedades de estos eventos físicos basándose únicamente en la información auditiva? ¿Cuál es la naturaleza de la información utilizada para recuperar estas características?. Mediante el procesamiento digital de señales es posible acercarse a las respuestas a dichas interrogantes.

Para realizar cualquier forma de procesamiento mediante computadoras digitales, las señales de audio analógicas deben reducirse a muestras discretas de un dominio de tiempo discreto. La operación que transforma una señal del tiempo continuo al tiempo discreto se llama ***muestreo***, y se realiza tomando los valores de la señal de tiempo continuo en instantes de tiempo que son múltiplos de una cantidad T_s, llamado intervalo o periodo de muestreo. La cantidad

$$F_s = \frac{1}{T_s}$$

se denomina frecuencia de muestreo (representada por F_s) y está dada en muestras/segundo o bien en Hz, mientras que T_s está dado en segundos. El proceso de muestreo implica importantes consecuencias en la señal, que serán analizadas de una forma empírica en el desarrollo de esta práctica; sin embargo, se mencionan algunas de ellas a continuación.

- El muestreo de una señal continua en el tiempo $x(t)$, con un periodo de muestreo T_s, produce una función $\hat{x}(n) \triangleq x(nT_S)$ de la variable discreta n.

- El muestreo de una señal de tiempo continuo con frecuencia de muestreo F_s produce una señal de tiempo discreto cuyo espectro de frecuencia es una replicación periódica del espectro de la señal original, y el período de replicación es F_s. La variable de Fourier ω para funciones de variable discreta se convierte en la variable de frecuencia f (en Hz) mediante la relación:

$$\omega = 2\pi f T_s = \frac{2\pi f}{F_s}$$

Las consideraciones anteriores nos pueden ayudar a comprender el teorema de muestreo introducido por Nyquist en la segunda década del siglo 20, y que fue popularizada por Shannon en poco más de 20 años después. El teorema dice que:

Teorema de Nyquist: Una señal de tiempo continuo $x(t)$, cuyo contenido espectral está limitado a frecuencias más pequeñas que F_b (es decir, no contiene ninguna frecuencia mayor o igual a F_b) se puede recuperar de su versión muestreada $\hat{x}(n) = x(nT)$ si la tasa de muestreo $F_s = 1/T_s$ es tal que

$$F_s > 2F_b$$

En otras palabras, para que una señal que ha sido muestreada pueda ser convertida nuevamente al dominio del tiempo continuo, es necesario que ésta haya sido muestreada a una frecuencia de muestreo superior al doble de la frecuencia F_b. En caso de no cumplirse esta relación, la reconstrucción de la señal arrojará una señal distinta a la originalmente muestreada; es decir, se obtendrá un *alias* de la señal original.

DESARROLLO

1.- Usando el comando load, cargue uno de los siguientes archivos de sesión incluidos en MATLAB: “handel.mat”, “chrip.mat” o “gong.mat”.

2.- Con el comando “sound” reproduzca el vector “*y*” a su respectiva “*Fs*” –*sound(y,Fs)*- y escuche atentamente el sonido reproducido.

3.- Vuelva a reproducir el mismo sonido pero a frecuencias $Fs_2 = 2*Fs$ y $Fs_3 = Fs/2$, escuche con atención y describa con sus propias palabras el fenómeno observado, y explique la razón de dicho efecto en el audio. Ponga especial atención al tiempo de reproducción de su secuencia de audio.

4.- Realice un submuestreo mediante el comando "*y2 = y(1 : k : end);*" donde $k=2$ (lo cual equivale a que la frecuencia de muestreo sea $Fs_2 = Fs/k$), y reproduzca la nueva secuencia usando la nueva Fs_2.

5.- Repita el paso anterior para valores de $k = 4$ y $k = 8$. Escuche con atención y describa el fenómeno observado. Explique en que momento el aliasing afectó tanto al audio que éste se volvió irreconocible.

6.- Utilice el comando *wavread* para leer el equivalente de 10 segundos de audio, y reprodúzcalo usando el comando *sound* con sus argumentos adecuados. Tome en cuenta el valor de Fs de dicho audio para calcular el número correcto de muestras necesarias para reproducir los 10 segundos de audio solicitados.

7.- Modifique la secuencia de audio de tal manera que el volumen del audio sea del 50% de su volumen original. Explique el procedimiento realizado para lograr el objetivo.

8.- Grafique la secuencia de audio original y la que se ha modificado e volumen, y compárelas visualmente.

Reportar

- Gráficas y códigos utilizados para obtener dichas gráficas. El código deberá estar debidamente comentado.
- Observaciones y explicaciones en sus propias palabras de todos los fenómenos observados en cada punto.
- Conclusiones.

ANEXO

Manejo de Secuencias de audio en la PC (Uso de MATLAB)

MATLAB cuenta con varios comandos útiles para el manejo de audio. Existen comandos para la grabación y reproducción de audio usando la tarjeta de sonido de la PC, así como también, para la lectura y escritura de secuencias de audio contenidas en archivos en formato WAV.

Reproducción de una Secuencia de Audio

La función *sound(y,Fs)* convierte la secuencia "*y*" en una señal de audio y la envía a la tarjeta de sonido a una frecuencia de muestreo *Fs*. En caso de que no se indique un valor para *Fs*, el sonido se reproducirá a una tasa de 8192 muestras por segundo. Para un audio monoaural, "*y*" es un vector columna de *m* x 1, donde *m* es el número de muestras de audio. En el caso de audio estéreo, "*y*" normalmente es una matriz de *m* x 2, donde la primera columna corresponde al canal izquierdo, y la segunda columna corresponde al canal derecho. La función *sound* asume que y contiene números de punto flotante que varían entre -1 y 1, y satura para valores fuera de dicho rango. Una sobrecarga de la función *sound* agrega un tercer argumento siendo su sintaxis

sound(y,Fs,bits)

El argumento "*bits",* especifica la precisión de los valores de las muestras. Aunque dicho valor depende del hardware, la mayoría de las tarjetas de sonido soportan al menos, precisiones de 8 y 16 bits; si dicho parámetro no se especifica, MATLAB reproduce el audio con una precisión de 8 bits. La mayoría de las tarjetas de sonido soportan frecuencias de muestreo (*Fs*) entre los 5 KHz y los 48 KHz, tenga en cuenta que especificar un *Fs* fuera de ese rango puede ocasionar resultados inesperados.

Recuperando Audio de Archivos wav

MATLAB provee la función *wavread* para leer archivos en formato wav y convertirlos en un vector o matriz de números que representen la señal de audio. Su sintaxis es la siguiente

1 - *y = wavread(NombreArchivo)*
2 - *[y, Fs] = wavread(NombreArchivo)*
3 - *[y, Fs, nbits] = wavread(NombreArchivo)*
4 - *[...] = wavread(NombreArchivo, N)*
5 - *[...] = wavread(NombreArchivo, [N1 N2])*
6 - *siz = wavread(NombreArchivo,'size')*

La primera sobrecarga de la función lee un archivo WAVE especificado por el "*NombreArchivo"* y devuelve el vector/matriz "y" con la señal de audio. La segunda y tercera sobrecarga, regresan los valores de la frecuencia de muestreo (*Fs*) y precisión

respectivamente. La cuarta y quinta sobrecargas pueden devolver los mismos valores que las sobrecargas anteriores, pero además aceptan un segundo argumento "*N*" o [N1 N2] que significan el número de muestras o un rango de muestras entre (N1 y N2) respectivamente.