Frecuencia de Muestreo Señal de Audio

Camilo Acosta Acosta
Instagram/GitHub: @acosmilo

Julio 2019

Objetivo

Grabar una señal de voz con frecuencia de muestreo de 16kHz, guardar esa señal en un archivo y graficarla en el tiempo y en la frecuencia. A la misma señal cambiar de frecuencia de muestreo a 8kHz, 4kHz, 2kHz y 1kHz. A esas señales guardarlas y graficarlas en tiempo y frecuencia.

Resolución

Se cierra figuras, se limpia variables y consola.

```
close all
clear all
clc
```

Parámetros iniciales

Se asignan valores a 3 parámetros.

1. Fs : La frecuencia de muestreo.
2. Bits : Número de bits, con los que se otendrá la grabación.
3. Canal : Cuantos canales se audio de entrada se usarán.

```
Fs = 16000; % Frecuencia de muestreo 16kHz.
Bits = 16;
Canal = 1; % Un unico canal audio mono.
```

Grabación

Usando la función *audiorecorder()* se graba la información del audio en un objeto llamado **recObj**. Usando la función *recordblocking()* ordenamos que grabe únicamente 5 segundos.

```
disp('Comienza grabacion REC'); % Mensaje de inicio en consola.
recObj = audiorecorder (Fs, Bits, Canal);
recordblocking (recObj, 5);
disp('Finaliza grabacion STOP'); % Mensaje de finalizado en consola.
```

Usando *getaudiodata()* convertimos la informacion grabada en **recObj** en un vector y se la almacena en la variable audio. *audio=getaudiodata(recObj)*; Se guarda el audio a Fs=16kHz con formato .wav.

```
audio16=audio;
audiowrite ('audio16kHz.wav', audio , Fs);
```

Comienza grabación REC. Finaliza grabación STOP.

Gráfica señal a 16kHz

Se crea un eje temporal t que permita ver la señal en 5 segundos, por ello al vector que va de 0 hasta 79999 se lo divide para Fs = 16000 para que el vector vaya desde 0 hasta 4.9, entonces el audio estaría en función del tiempo.

```
t=(0:length(audio16)-1)/Fs;
```

Se crea un eje frecuencial f con 80000 muestras y que vaya desde 0 hasta Fs dividido para la longitud del vector audio.
A Y se le da los valores de usando *abs()* del vector resultante de *fft()* dividido para la longitud del audio.

```
f=Fs/length(audio16)*(0:length(audio16)-1);
Y=abs(fft(audio16)/length(audio16));
```

Se plotea la señal temporal y su espectro en frecuencia.

```
figure
subplot(1,2,1)
plot (t,audio16,'Color',[0,0,1]);
xlim([0 5])
ylim([-1.25 1.25])
```

```
xlabel('Tiempo[s]');
ylabel('Amplitud')
title('Senal a 16k[Hz]')

subplot(1,2,2)
plot (f,Y,'Color',[0,0.5,1])
xlim([0 8000])
ylim([0 0.005])
xlabel('Frecuencia 16k[Hz]');
ylabel('|Y(f)|')
title('Espectro de Frecuencia')
```

Gráficas señal a 8kHz, 4kHz, 2kHz y 1kHz.

Se comienza usando el mismo código para la señal a Fs = 16000, a diferencia de que el nuevo Fs se dividirá para dos en 8kHz, para cuatro en 4 kHz, para ocho en 2 kHz y para dieciséis en 1 kHz. Tambien es necesario entender que para la nueva señal a Fs = 8000, obtiene los datos de su vector a partir de la anterior, para solucionar esto se optó por usar los valores con índice par del vector *audio* de este modo las muestras de audio a Fs=8000 se reduce a 40000, Fs=4000 se reduce a 20000, Fs=2000 se reduce a 10000 y Fs=1000 se reduce a 5000.

```
audio(2:2:end)=[];
audio8=audio;
audiowrite ('audio1.wav', audio8, (Fs/2));


t1=(0:length(audio8)-1)/(Fs/2);

f1=(Fs/2)/length(audio8)*(0:length(audio8)-1);
Y=abs(fft(audio8)/length(audio8));

figure
subplot(1,2,1)
plot(t1,audio8,'Color',[0.5,0.5,1])
xlim([0 5])
ylim([-1.25 1.25])
xlabel('Tiempo[s]');
ylabel('Amplitud')
title('Senal a 8k[Hz]')

subplot(1,2,2)
plot (f1,Y,'Color',[0.5,1,1])
xlim([0 4000])
```

```
ylim([0 0.005])
xlabel('Frecuencia 8k[Hz]');
ylabel('|Y(f)|')
title('Espectro de Frecuencia')

audio(2:2:end)=[];
audio4=audio;
audiowrite ('audio1.wav', audio4, (Fs/4));


t2=(0:length(audio4)-1)/(Fs/4);

f2=(Fs/4)/length(audio4)*(0:length(audio4)-1);
Y=abs(fft(audio4)/length(audio4));

figure
subplot(1,2,1)
plot(t2,audio4,'Color',[1,0,1])
xlim([0 5])
ylim([-1.25 1.25])
xlabel('Tiempo[s]');
ylabel('Amplitud')
title('Senal a 4k[Hz]')

subplot(1,2,2)
plot (f2,Y,'Color',[1 0.4 0.6])
xlim([0 2000])
ylim([0 0.005])
xlabel('Frecuencia[Hz]');
ylabel('|Y(f)|')
title('Espectro de Frecuencia')

audio(2:2:end)=[];
audio2=audio;
audiowrite ('audio1.wav', audio2, (Fs/8));


t3=(0:length(audio2)-1)/(Fs/8);

f3=(Fs/8)/length(audio2)*(0:length(audio2)-1);
Y=abs(fft(audio2)/length(audio2));

figure
subplot(1,2,1)
plot(t3,audio2,'Color',[0.3 1 0.7])
xlim([0 5])
```

```
ylim([-1.25 1.25])
xlabel('Tiempo[s]');
ylabel('Amplitud')
title('Senal a 2k[Hz]')

subplot(1,2,2)
plot (f3,Y,'Color',[0.5 1 0])
xlim([0 1000])
ylim([0 0.005])
xlabel('Frecuencia[Hz]');
ylabel('|Y(f)|')
title('Espectro de Frecuencia')

audio(2:2:end)=[];
audio1=audio;
audiowrite ('audio1.wav', audio1, (Fs/16));


t4=(0:length(audio1)-1)/(Fs/16);

f4=(Fs/16)/length(audio1)*(0:length(audio1)-1);
Y=abs(fft(audio1)/length(audio1));

figure
subplot(1,2,1)
plot(t4,audio1,'Color',[1 0 0.5])
xlim([0 5])
ylim([-1.25 1.25])
xlabel('Tiempo[s]');
ylabel('Amplitud')
title('Senal a 1k[Hz]')

subplot(1,2,2)
plot (f4,Y,'Color',[1 0.5 0])
xlim([0 500])
ylim([0 0.005])
xlabel('Frecuencia[Hz]');
ylabel('|Y(f)|')
title('Espectro de Frecuencia')
```

Recomendaciones

Para poder escuchar los audios correspondientes es necesario la función *sound(audio,Fs)* con Fs igual a 16000, 8000, 4000, 2000 o 1000 según corresponda.

Conclusiones

- Con lo que tiene que ver con el dominio del tiempo la señal sigue durando 5 segundos, pero al perder la mitad de información cada vez que se reduce a la mitad la frecuencia van a dar gráficas similares si el audio graba frecuencias menores a 500Hz. Si solo son frecuencias menores a 1000Hz y mayores a 500 Hz las cuatro primeras gráficas se pareceran. Si son frecuencias menores a 2000Hz y mayores a 1000 Hz las tres primeras gráficas seran muy similares, y asi en cada gráfica. Esto se debe al Teorema de Muestreo donde $Fs >= 2F_{MAX}$, por eso la frecuencia máxima que se puede graficar en cada figura cambia debido a que sera la mitad de la frecuencia de muestreo.

- Con respecto al dominio de la frecuencia en cada gráfica debido al Teorema del Muestreo, la función *fft()* grafica frecuencias en un rango desde cero hasta su respectiva frecuencia máxima.

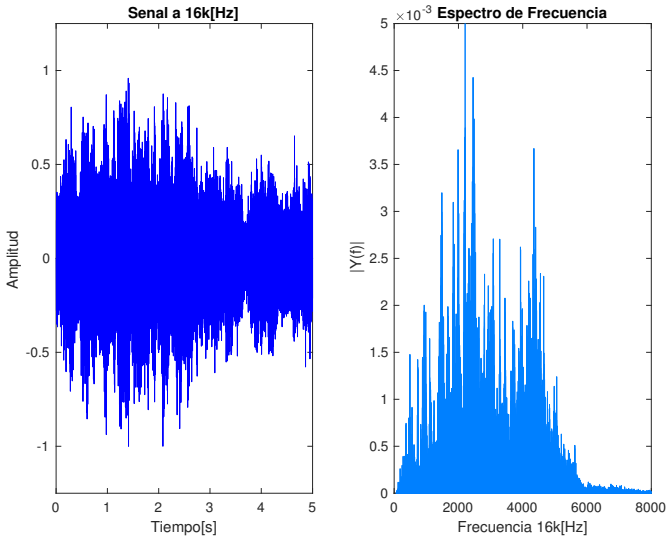
- Si el audio tiene frecuencias mayores a la frecuencia máxima de cualquier gráfica, lo que suele pasar es que los lóbulos vecinos del espectro se intersecan entre si dando un espectro erroneo.

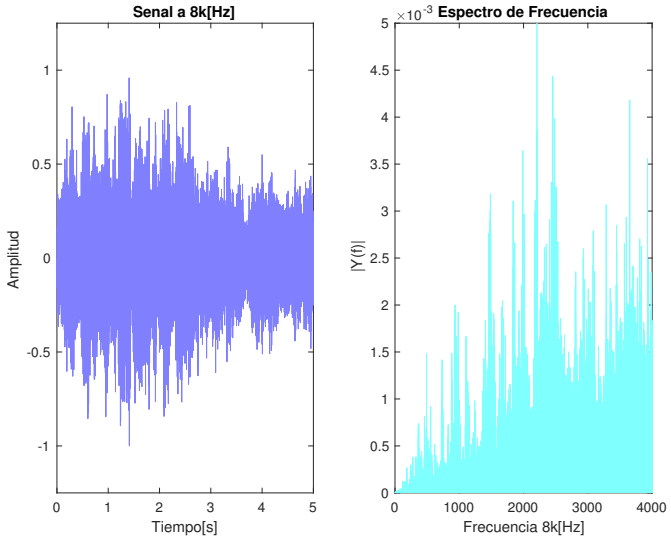
- Lo anterior se puede ver en las 5 primeras figuras donde un audio de voz humana a diferentes frecuencias de muestreo dibuja gráficas muy similares. Pero como se ve en las siguientes 5 figuras, un audio de una señal sinusoidal a 4kHz solo gráfica similar para las gráficas con frecuencia de muestreo 16kHz y 8kHz; para las frecuencias de muestreo menores, grafica erroneamente tanto en el dominio del tiempo como en la frecuencia.

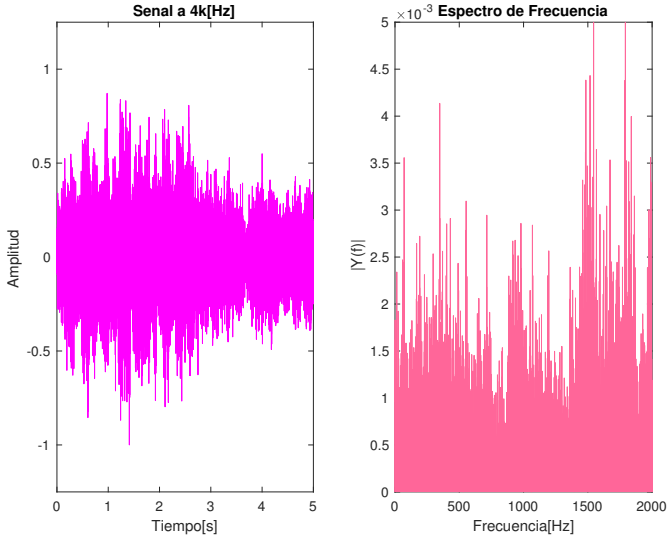
- Las diferencias al escuchar las cinco señales dependeran. Mientras más se baje la frecuencia de muestreo la voz sonorá más grave y puede que no se distinga entre voz de hombre o de mujer.

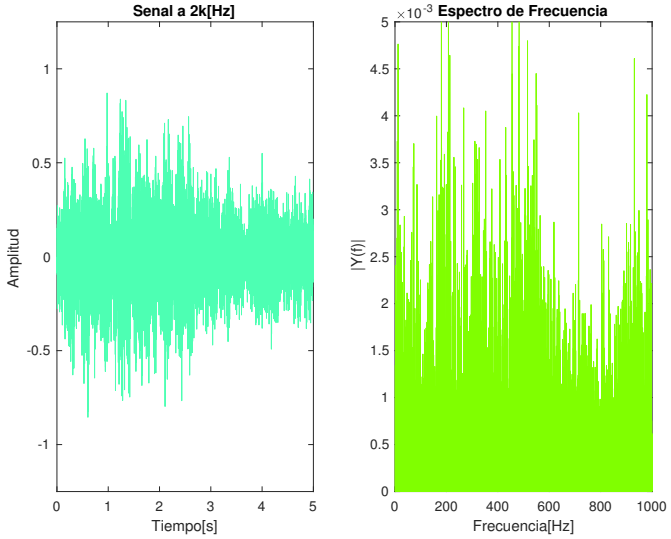
Anexos

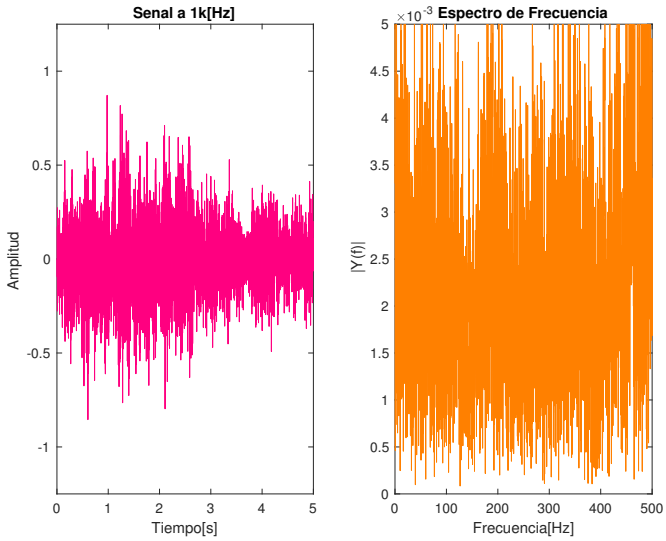
Gráficas de una señal de audio, voz humana, obtenidas a diferentes frecuencias de muestreo.



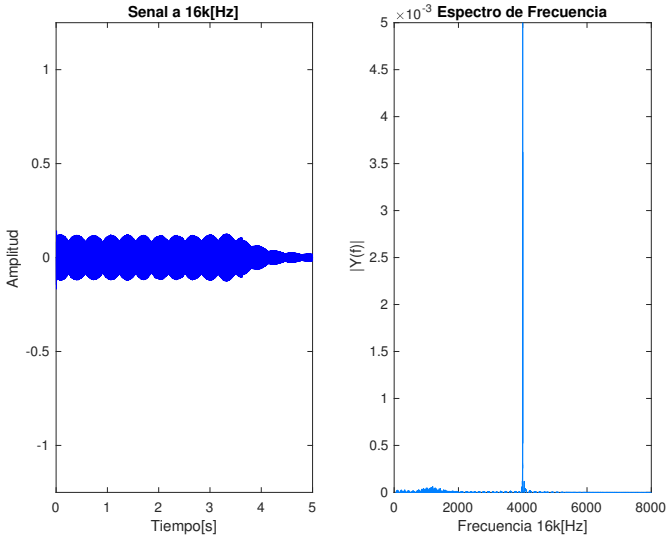


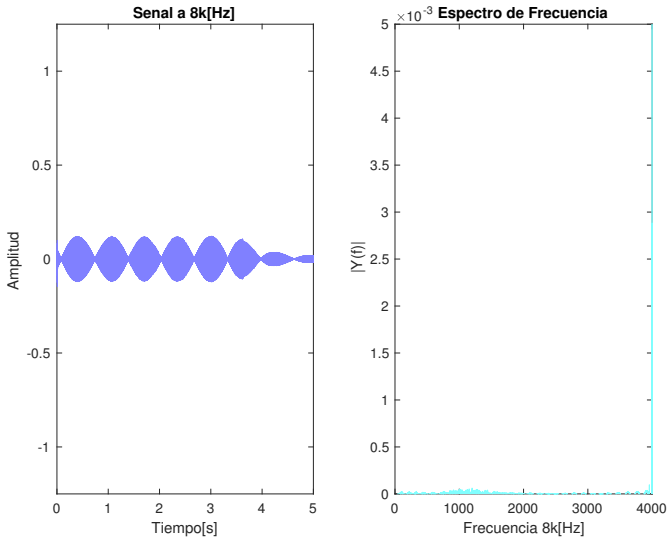

Senal a 4k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia[Hz]

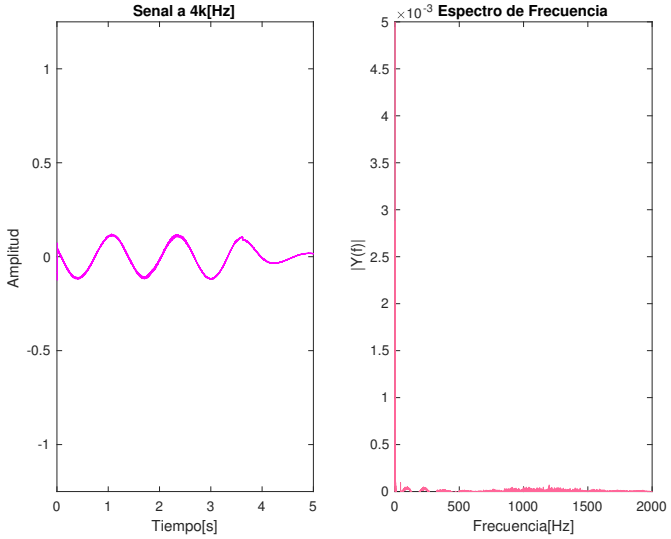

Senal a 2k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia[Hz]

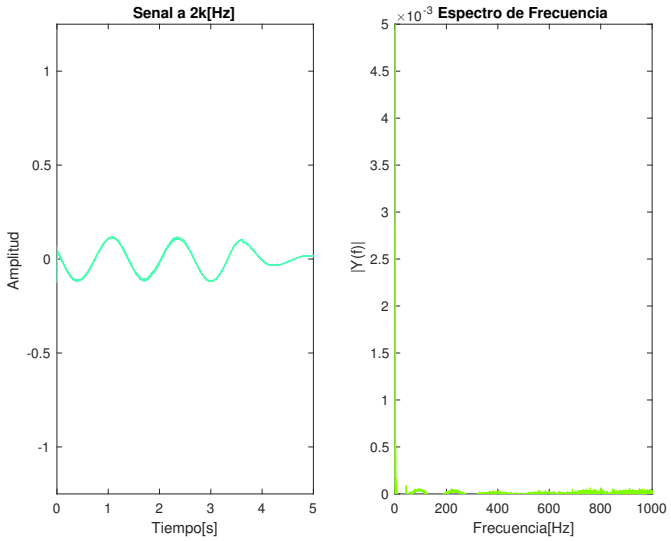


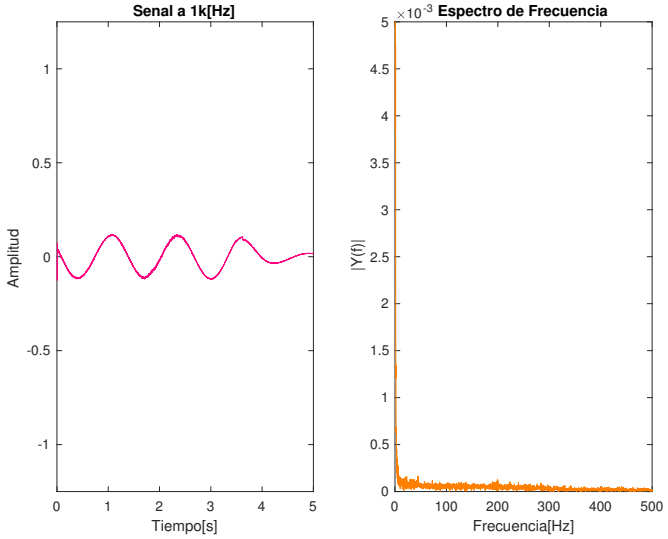
Gráficas de una señal de audio, onda sinusoidal de 4kHz, obtenidas a diferentes frecuencias de muestreo.




Senal a 8k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia 8k[Hz]


Senal a 4k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia[Hz]


Senal a 2k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia[Hz]


Senal a 1k[Hz]
Amplitud
Tiempo[s]
×10^-3 Espectro de Frecuencia
|Y(f)|
Frecuencia[Hz]