Procesamiento Señales

Práctica

OPERACIONES CON SEÑALES

NO. DE PRACTICA:	4(CUATRO)	Fecha de entrega:	29 de septiembre, 2023
INTEGRANTES DE EQUIPO: 19 NAVIL PINEDA RUGERIO		Practica:	Individual / En equipo

OBJETIVO: REALIZAR OPERACIONES CON SEÑALES DIGITALES DE AUDIO

MATERIAL Y EQUIPO: COMPUTADORA CON MATLAB

DESARROLLO:

1. Ejecuta el comando **load('handel.mat').**

```
load('handel.mat');
```

Al llamar a la función load en el workspace se carga automáticamente un vector y, el cual contiene muestras de una señal de audio.

2. Abrir uno de los archivos para verificar que se trata de datos numéricos.

```
disp(y);
```

Se utiliza disp(y) para mostrar el vector con los valores de amplitud de la señal, en 73113 muestras de tiempo.

3. Cargar los datos del sonido digitalizado (archivo de texto) al 'Workspace' de MATLAB. Usar la función load y guardar en una variable con el mismo nombre.

```
y = y;
```

Se utiliza disp(y) para mostrar el vector con los valores de amplitud de la señal, en 73113 muestras de tiempo.

4. Después de hacer esto deben aparecer las variables en el Workspace.

Y como se puede observar en la siguiente imagen se tienen cargados los dos archivos de texto.

5. Generar el vector **'np', 'nd', 'nde', 'ns'** correspondiente al tiempo de las variables. Deben tener el mismo número de elementos que los que tienela variable.
   Por ejemplo: Si la variable **y** tiene 14001 elementos. Esto se puede ver en la ventana workspace. Entonces el vector de tiempo debe tener 14001 elementos

```
y = y;
```

Para este apartado se divide la señal en tres partes iguales, creando vectores de tiempo de tamaño de un tercio de la imagen original, en los vectores np, nd y nde, además se grafican esas tres nuevas partes de la señal.

```
np = y(1:round(length(y)/3));
```

```
nd = y(round(length(y)/3):round(length(y)/3)*2);
```

```
nde = y(round(length(y)/3)*2:round(length(y)/3)*3);
```

6. Graficar cada una de las señales usando la función stem, recuerda que debe graficarse y contra np.

```
stem(np,'b')
xlabel('Longitud de la señal (np)')
ylabel('Amplitud de la señal (y)')
title('Señal 1')

stem(nd,'b')
xlabel('Longitud de la señal (nd)')
ylabel('Amplitud de la señal (y)')
title('Señal 2')

stem(nde, 'b')
xlabel('Longitud de la señal (np)')
ylabel('Amplitud de la señal (y)')
title('Señal 3')
```

Se utiliza la función stem para graficar las tres nuevas señales, que forman parte de la señal original, generadas anteriormente.

7. Reproducir las señales como sonido a una frecuencia de muestreo de 8000 hz utilizando la función de MATLAB que te permita realizar este proceso. Fs es la frecuencia de muestreo y debes desarrollarlo para los siguientes valores de Fs:
   - ✓ 4000
   - ✓ 5000
   - ✓ 8000
   - ✓ 20000

Primero generamos una variable para guardar la frecuencia de muestreo, en el primer caso será de 8000 hz, un vector de tamaño de la longitud de los tiempos la muestra original entre Fs, que es la frecuencia de muestreo original, y un vector de tiempo para la nueva muestra resampleada. Una vez que se tienen los tiempos y la frecuencia deseada se utiliza la función interp1, para interpolar la señal original (una tercera parte de la señal de audio completa), para ajustarla a la nueva frecuencia de muestreo. Después se reproduce el sonido con la función sound de Matlab, la cual recibe la señal con la nueva frecuencia, y el dato numérico de la frecuencia.

Para 8000 hz.

```
fs_new = 8000; % frecuencia de muestreo

t_original = (0:length(np) - 1) / fs;
t_new = (0:(length(np) - 1) * (fs_new / fs)) / fs_new;

% interpolar la señal np para ajustarla a la nueva frecuencia de muestreo
np_resampled = interp1(t_original, np, t_new);
sound(np_resampled1, 4000);
```

Para 4000 Fs.

```
fs_new = 4000; % frecuencia de muestreo
t_original = (0:length(np) - 1) / fs;
t_new = (0:(length(np) - 1) * (fs_new / fs)) / fs_new;
np_resampled1 = interp1(t_original, np, t_new);
```

Para 5000 Fs.

```
fs_new = 5000; % frecuencia de muestreo
t_original = (0:length(np) - 1) / fs;
t_new = (0:(length(np) - 1) * (fs_new / fs)) / fs_new;
np_resampled2 = interp1(t_original, np, t_new);
sound(np_resampled2, 5000);
```

Para 8000 Fs.

```
fs_new = 8000; % frecuencia de muestreo
t_original = (0:length(np) - 1) / fs;
t_new = (0:(length(np) - 1) * (fs_new / fs)) / fs_new;
np_resampled2 = interp1(t_original, np, t_new);
sound(np_resampled3, 8000);
```

Para 20000 Fs.

```
fs_new = 20000; % frecuencia de muestreo
t_original = (0:length(np) - 1) / fs;
t_new = (0:(length(np) - 1) * (fs_new / fs)) / fs_new;
np_resampled4 = interp1(t_original, np, t_new);
sound(np_resampled4, 20000);
```

8. Observe las diferencias de sonido y gráfico para cada uno de los anteriores incisos dando una explicación de la diferencia.

Para mostrar gráficamente las señales con diferente frecuencia de muestreo se utiliza la función stem.

```
stem(np_resampled1, 'k')
xlabel('np')
ylabel('y (resampleada)')
title('4000 fs')

stem(np_resampled2, 'k')
xlabel('np')
ylabel('y (resampleada)')
title('5000 fs')

stem(np_resampled3, 'k')
xlabel('np')
ylabel('y (resampleada)')
title('8000 fs')

stem(np_resampled4, 'k')
xlabel('np')
ylabel('y (resampleada)')
title('20000 fs')
```

Al observar las gráficas de la señal se puede ver que las muestras si varían de acuerdo con el tiempo que se muestra en la gráfica, esto quiere decir, que dependiendo de la variación de la frecuencia de muestreo es como se comportará la señal, y pueden ocurrir dos cosas: si se aumente la frecuencia de muestreo quiere decir que se aumentan más muestras por segundo, por lo tanto, la representación de la señal es más detallada, cuenta con más información; o bien si disminuye la frecuencia de muestreo se toman menos muestras por segundo, lo que puede resultar en pérdida de información o bien en distorsión de la señal original, dependiendo de qué tanto se reduzca.

9. Obtener la suma de las cuatro señales y guardarla en una variable con nombre **sumaSenal**.

Para ello se debe modificar la longitud de las señales, pues en la suma todas las señales deben ser del mismo tamaño, por lo que se interpolan hasta la frecuencia más alta (20000 hz).

```
% Frecuencia de muestreo común (20000 hz)
fs_common = 20000;

t_original = (0:length(np) - 1) / fs_new;
t_new = (0:(length(np) - 1) * (fs_common / fs_new)) / fs_common;

max_length = max([length(np_resampled1), length(np_resampled2),
length(np_resampled3), length(np_resampled4)]);
np_resampled1 = [np_resampled1, zeros(1, max_length - length(np_resampled1))];
np_resampled2 = [np_resampled2, zeros(1, max_length - length(np_resampled2))];
np_resampled3 = [np_resampled3, zeros(1, max_length - length(np_resampled3))];
np_resampled4 = [np_resampled4, zeros(1, max_length - length(np_resampled4))];

t_original = (0:max_length - 1) / fs_new;
t_new = (0:(max_length - 1) * (fs_common / fs_new)) / fs_common;

% Interpolar hasta 20000 hz
np_resampled1 = interp1(t_original, np_resampled1, t_new);
np_resampled2 = interp1(t_original, np_resampled2, t_new);
np_resampled3 = interp1(t_original, np_resampled3, t_new);
np_resampled4 = interp1(t_original, np_resampled4, t_new);

% Suma
summed_signal = np_resampled1 + np_resampled2 + np_resampled3 + np_resampled4;
```

10. Graficar la señal **sumaSenal** y reproducirla como sonido.

Se grafica el vector de que contiene la suma de las cuatro señales y se reproduce el sonido a una frecuencia de 20000Fs.

```
sound(summed_signal, fs_common);
stem(summed_signal, 'g')
xlabel('np')
ylabel('y (sumada)')
title('suma de señales')
```

11. Realizar las operaciones necesarias para concatenar las cuatro señales en el mismo orden que en (1) dándoles un espaciamiento de 2000 muestras y guardarlas en la señal **datosConcatenados**. Graficar y reproducir.

Para este apartado se concatenan las señales en el siguiente orden:

1. Señal 1 (4000Fs) resampleada.
2. Espacio de 2000 muestras.
3. Señal 2 (5000Fs) resampleada.
4. Espacio de 2000 muestras.
5. Señal 3 (8000Fs) resampleada.
6. Espacio de 2000 muestras.
7. Señal 4 (20000Fs) resampleada.

```
space = zeros(1, 2000); % espaciado
% concatenar las señales con el espaciado
datosconcatenados = [np_resampled1, space, np_resampled2, space, np_resampled3, space, np_resampled4];

% graficar y reproducir
sound(datosconcatenados, 20000);
stem(datosconcatenados, 'r')
xlabel('muestras concatenadas')
ylabel('amplitud de la señal')
title('señales concatenadas')
```

12. Invertir la señal **datosConcatenados** y guardalos en **invertida**.

A continuación, se utiliza la función flip de Matlab, que invierte un arreglo, en este caso se invierte el arreglo que contiene la concatenación de las señales resampleadas.

```
invertida = flip(datosconcatenados);
```

13. Graficar y reproducir la señal **invertida**.

Finalmente, se grafica la señal invertida y se reproduce el sonido.

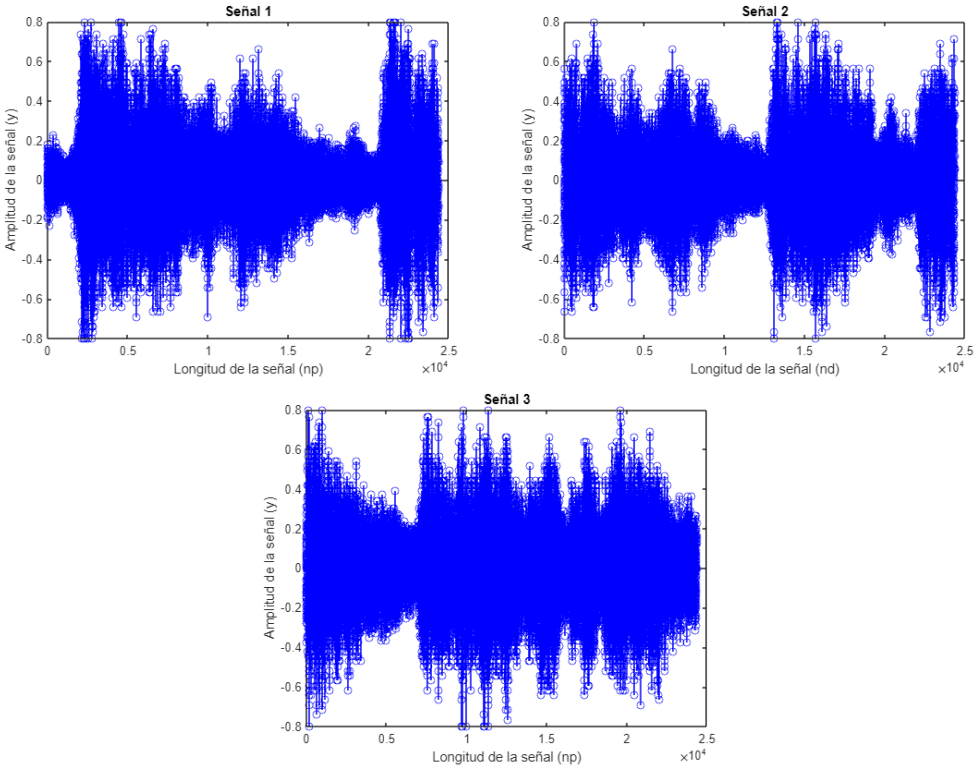
```
sound(invertida, 20000);
stem(invertida, 'c')
xlabel('muestras concatenadas')
ylabel('amplitud de la señal')
title('señal invertida')
```

RESULTADOS:

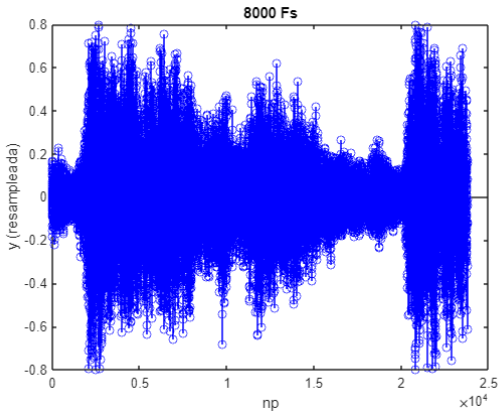
1. El resultado de cargar los archivos de texto se puede ver desde el workspace, y para comprobarlo se imprimió el arreglo “y” que guarda los datos, mostrando algo como lo siguiente:

```
     0
-0.0062
-0.0750
-0.0312
 0.0062
 0.0381
 0.0189
-0.0250
-0.0312
```

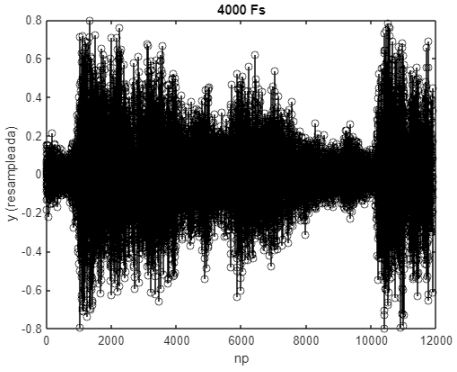
2. A continuación, se divide en tres partes la señal de audio original, y se muestran las gráficas de las tres particiones.



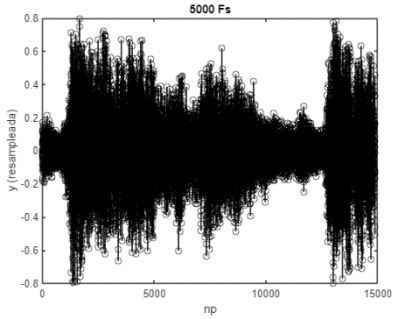
3. Se modifica la frecuencia de muestreo de la señal original y se disminuye a 8000 hz, a continuación, se muestra la gráfica de la señal resampleada.



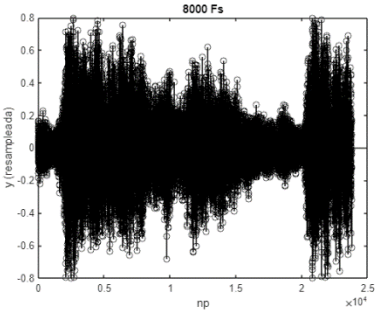
4. Se hace la modificación de los otros cuatro valores de frecuencia de muestreo, y se muestran sus gráficas. Para 4000Fs.



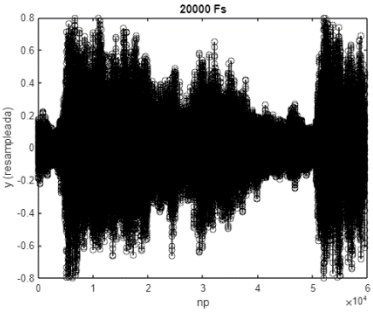
Para 5000Fs.



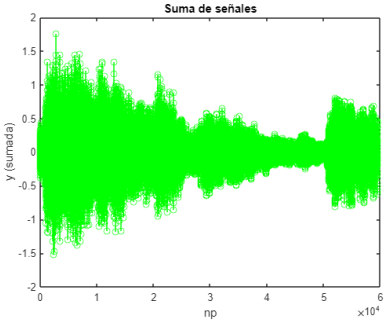
Para 8000Fs.



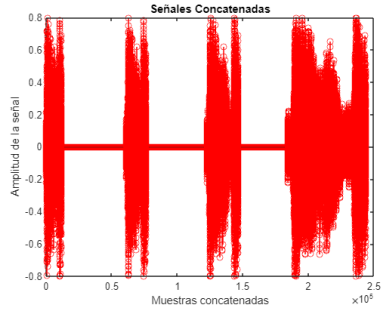
Para 20000Fs.



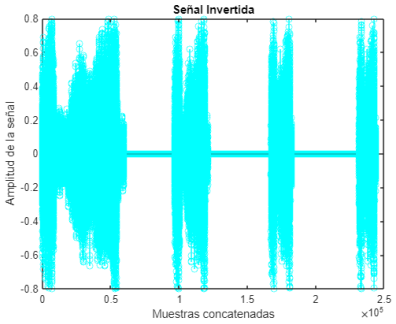
5. Se interpolan y se suman esas cuatro nuevas señales, y se muestra la gráfica de la suma.



6. Después se concatenan las cuatro señales anteriores dejando un espacio de 2000 muestras entre cada una, obteniendo el siguiente resultado.



7. Finalmente, se invierte la señal concatenada anteriormente y se grafica.



CONCLUSIÓN:

Durante esta práctica se ha trabajado con señales de audio y se han podido aplicar operaciones sobre ellas, observando los resultados tanto gráficamente como reproduciendo el mismo audio. Se pudo observar que modificando tan solo la frecuencia de muestreo se obtiene un resultado completamente diferente.

A lo largo de este documento se presentaron diferentes transformaciones a la señal, desde dividirla en intervalos o subseñales de la señal original, hacer el resampling de las mismas a una frecuencia de muestreo diferente, ya sea reduciendo o aumentando el número de muestras, sumando señales o concatenándolas en una sola. Utilizando Matlab se pudo observar que ya tiene incluidas una serie de funciones muy útiles en este aspecto, ya que es sencillo llamar solo a la función para realizar diferentes procesos, como la reproducción de audio o la interpolación, siendo útil para el procesamiento y manipulación de señales de audio para diversas aplicaciones, como en música, procesamiento de audio, comunicación, entre otras.

FECHA FINAL DE ENTREGA: **viernes, 29 de septiembre de 2023**