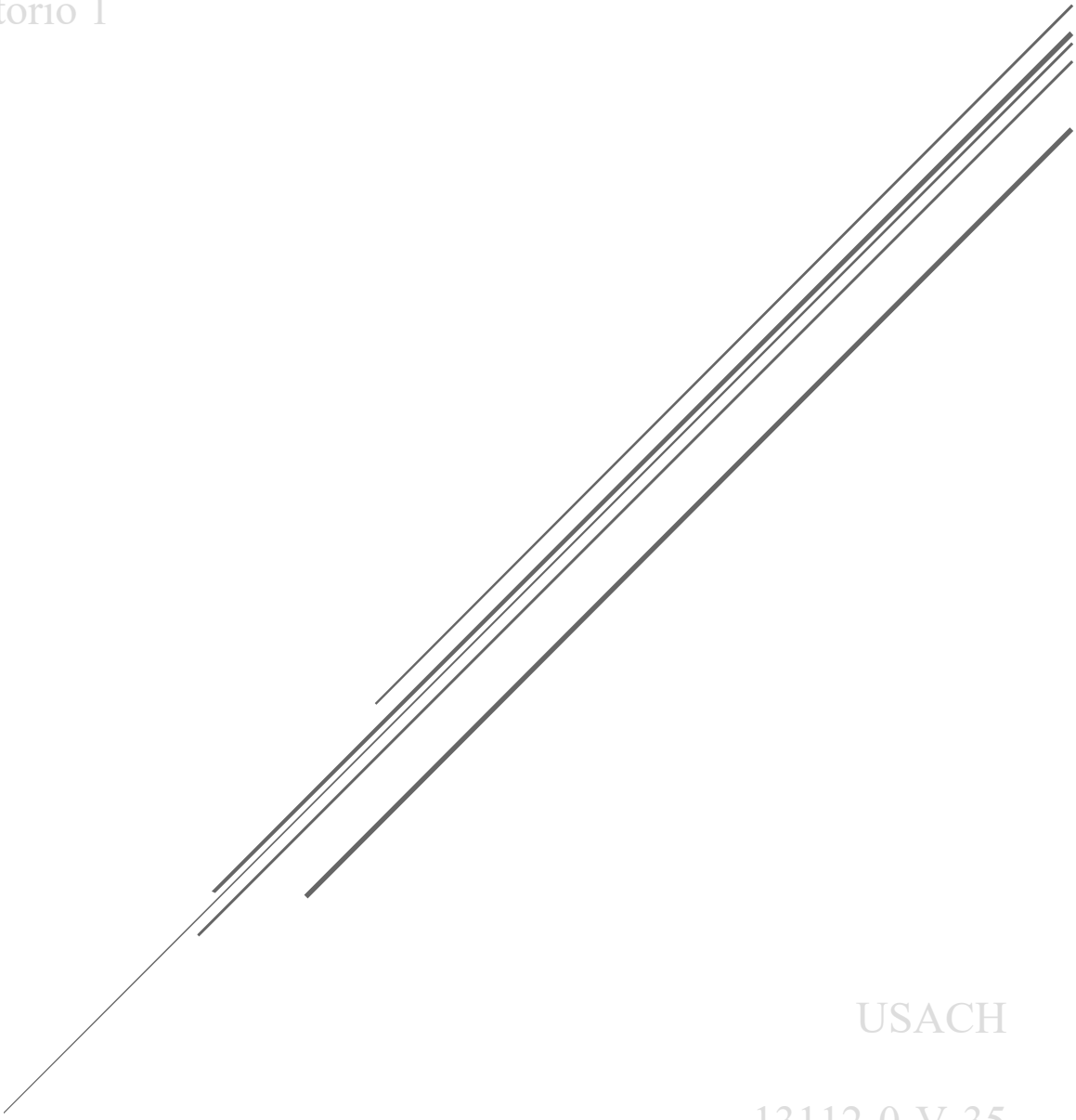
LABORATORIO 1: SEÑALES ANÁLOGAS Y DIGITALES

Laboratorio 1



USACH

13112-0-V-35

Laboratorio 1: Señales Análogas Y Digitales

Mónica Araneda

USACH

Author Note

El diseño es el alma de todo lo creado por el hombre S.J.

Tabla de Contenidos

Laboratorio 1: Señales Análogas Y Digitales

Introducción

En la naturaleza, el conjunto de señales que percibimos son analógicas, así la luz, el sonido, la energía etc, son señales que tienen una variación continua. Incluso la descomposición de la luz en el arco iris vemos como se realiza de una forma suave y continúa. En este laboratorio primero trabajaremos en señales en el dominio del tiempo. Porque se requiere medir la señal de audio en un intervalo en el tiempo.

Marco Teórico

Una señal digital, en el dominio del tiempo, incluye segmentos horizontales y verticales conectados. Una línea vertical en el dominio de tiempo significa una frecuencia infinita. Mientras que el tramo horizontal representa una frecuencia cero. Ir de una frecuencia cero a una frecuencia infinito (y viceversa) implica que todas las frecuencias en medio son parte del dominio.

El análisis de Fourier se puede usar para descomponer una señal. Si la señal digital es periódica, lo que es raro en comunicaciones, la señal descompuesta tiene una representación en el dominio de frecuencia con un ancho de banda infinito y frecuencias discretas. Si la señal digital es aperiódica, la señal descompuesta todavía tiene un ancho de banda infinito, pero las frecuencias son continuas.

Objetivo

Esta experiencia tiene como objetivos reforzar los contenidos vistos en clases sobre señales y transformada de Fourier.

Herramientas[1]

Se utilizará el lenguaje de programación Python 3 y algunos módulos de utilidad como: Numpy, Scipy, Matplotlib.

El IDE utilizado para el desarrollo de mí trabajo es : Jupyter Notebook 6.0.1, de Anaconda Navigator 1.9.12.

Desarrollo

Utilizando las herramientas mencionadas por este laboratorioy la señal de audio "handel.wav" publicada en el foro del curso, realicé las siguientes actividades. Utilice las preguntas planteadas como guía para el análisis de mis resultados en el informe.

La señal de audio y los parámetros retornados.

```
import scipy.io.wavfile as waves

archivo = '/Users/monicaaraneda/Documents/CAPACITACI

muestreo, sonido = waves.read(archivo)

size= sonido.shape[0] / muestreo
print(f"duracion = {size}s")
time = np.linspace(0., size, sonido.shape[0])
plt.plot(time, sonido, label="Sonido PCM")
```

Figure 1

La instrucción de lectura de archivo (waves.read(archivo)), entrega dos variables:

Se obtiene la frecuencia de muestreo en muestreo y los datos en sonido.

El valor de muestreo es la frecuencia de los datos por segundo, el arreglo sonido contiene los datos del sonido. Las dimensiones del arreglo sonido indican la cantidad de muestras y el número de canales de audio (mono, estéreo, etc).

La función de audio en el tiempo.

```
duracion = 8.9249267578125s
```

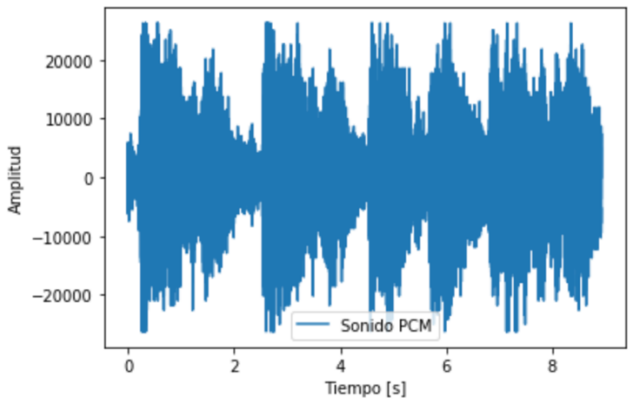


Figure 2

La transformada de Fourier de la señal de audio:

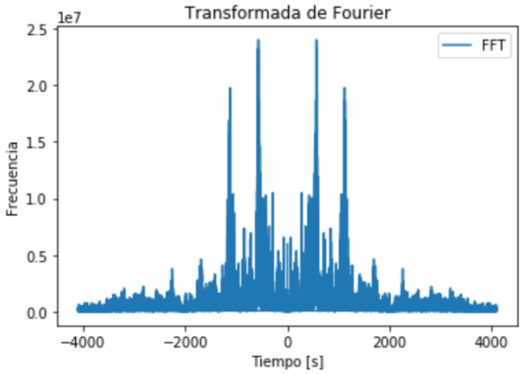


Figure 3

Comparando la Figura 3 con la señal leída en figura 2. Balabala bala

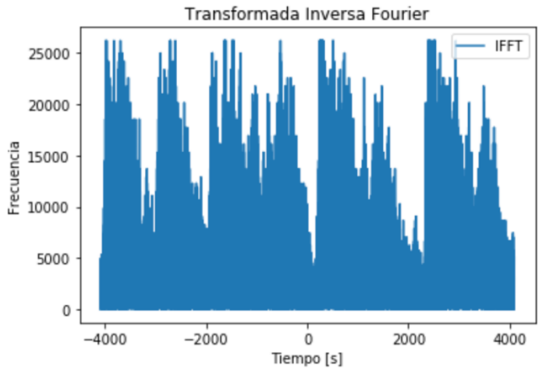


Figure 4

```
fft_x = np.fft.fft(sonido)
fft_i = np.fft.ifft(fft_x)
freqs = fft.fftfreq(len(fft_x)) * muestreo
```

Figure 5

Comparando la Figura 4 con la señal leída en figura 3. La transformada FFT deconstruye una representación en el dominio del tiempo de una señal en la representación de dominio de frecuencia para analizar las diferentes frecuencias en una señal.

Una vez que haya calculado la magnitud de cada coeficiente de FFT, se puede averiguar a qué frecuencia de audio pertenece cada coeficiente de FFT. Una FFT de punto N da el contenido de frecuencia de la señal en N frecuencias igualmente espaciadas, comenzando en 0. Debido a que su frecuencia de muestreo es 73113 Hz. y el número de puntos en su FFT es 459, su espaciado de frecuencia es 73113 / 459 = 159 Hz (aproximadamente).

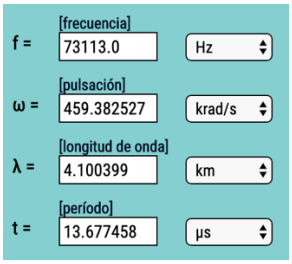


Figure 6

Calculando y graficando el espectrograma de la función. El espectrograma permite visualizar información en el dominio de la frecuencia y del tiempo a la vez.

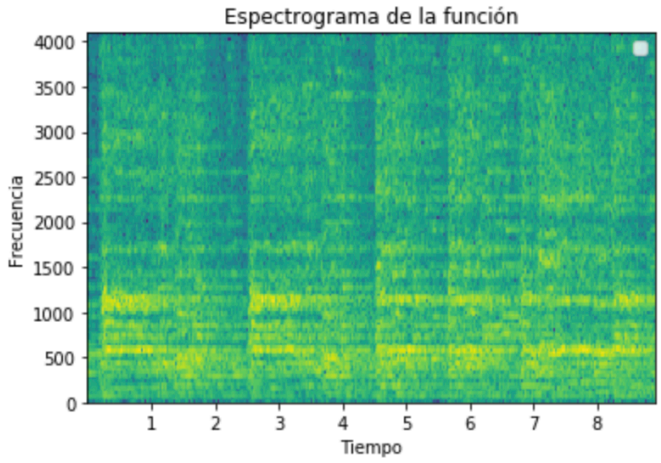


Figure 7

```
# Plot the signal
plot.figure(figsize=(8,8))
plot.subplot(211)
plot.plot(sonido, label="dominio tiempo")
plot.xlabel('Tiempo')
plot.ylabel('Amplitud')
plot.legend()
plot.title('Espectrograma de la función')

# Plot the spectrogram
plot.subplot(212)
powerSpectrum, freqenciesFound, time, imageAxis = plot.specgram(sonido, Fs=Fs1)
plot.xlabel('Tiempo')
plot.ylabel('Frecuencia')

plt.show()
```

Figure 8

WAV file: Fs = 8192, x.shape = (73105,), x.dtype = float32

Análisis de resultados

La señal se clasifica como analógica cuando su amplitud puede tomar un infinito numero de valores en un rango contínuo de tiempo. Por ejemplo, la señal de sonido como la del archivo: handel.wav

```
# Señales analógicas   http://blog.espol.edu.ec/telg1001/senales-analogicas-y-digitales/
# propuesta: edelros@espol.edu.ec

import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as waves

# INGRESO
# archivo = input('archivo de audio: ')
archivo = 'handel.wav'
muestreo, sonido = waves.read(archivo)

# SALIDA - Observacion intermedia
print('frecuencia de muestreo: ', muestreo)
print('dimensiones de matriz: ', np.shape(sonido))
print('datos de sonido: ')
print(sonido)
```

```
frecuencia de muestreo:  8192
dimensiones de matriz:  (73113,)
datos de sonido:
[    0  -202 -2459 ...  7452  4930     0]
```

Con los resultados mostrados, se observa que:

La señal de audio tiene frecuencia de **muestreo** de 8192 Hz

Las dimensiones de la matriz **sonido** (73113,) indican que es de tipo estéreo por usar columnas

El valor de 73113 corresponde a la cantidad de muestras por canal.

Los índices para seleccionar un canal son 0 ó 1.

Para observar la forma del **sonido** se extrae solo un fragmento usando el canal 0.

Señales digitales

Una señal se clasifica como digital cuando la amplitud puede tomar solo un número finito de valores, para un número finito de muestras en el tiempo. Por ejemplo, datos del audio del muestran el carácter finito, discreto, por muestras de la señal.

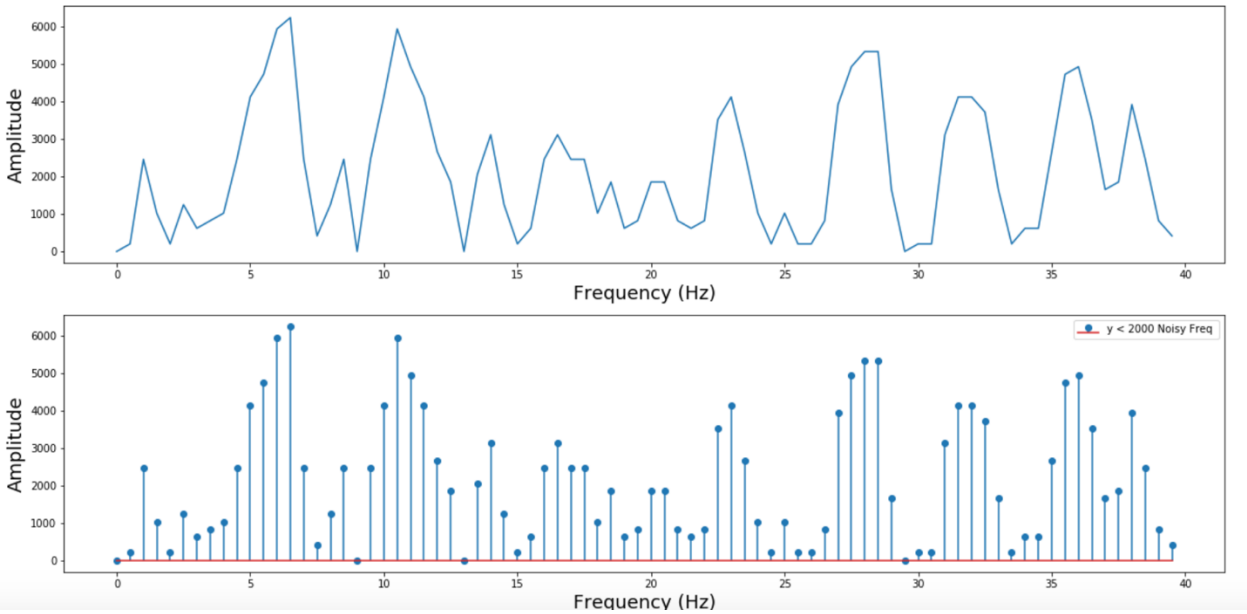


Figure 9

La información se puede obtener de cada gráfico es que las tasas de muestreo más altas nos permiten observar mejor las señales subyacentes. Esto da como resultado una vista irregular de los datos, pero aún podemos observar con precisión la frecuencia y observar que las frecuencias menores a 2000 corresponden a frecuencias de ruido, las que se eliminan aplicando el filtro con una frecuencia de corte de 400 Hz.

Los filtros se pueden clasificar como respuesta de impulso finita (FIR) o respuesta de impulso infinita (IIR). En este laboratorio se utiliza el FIR.

Los filtros se construyen en el dominio de la frecuencia y se deben considerar varias propiedades.

Tomé la FFT inversa y la tracé para ver cómo se ve como un núcleo en el dominio temporal.

Observé cómo la ganancia se escala desde [0,1] Los filtros se pueden multiplicar por la FFT de una señal para aplicar el filtro en el dominio de la frecuencia. Cuando la señal resultante se transforma nuevamente en el dominio del tiempo usando la FFT inversa, la nueva señal se filtrará. Esto puede ser mucho más rápido que aplicar filtros en el dominio del tiempo.

El parámetro filter_order se usa para ajustar la nitidez del corte en el dominio de frecuencia. Intenté usar diferentes valores para ver cómo cambia la trama del filtro.

```
filter_order = 2
frequency_cutoff = 25
sampling_frequency = 500

b, a = butter(filter_order, frequency_cutoff, btype='high', output='ba', fs=sampling_frequency)

plot_filter(b, a, sampling_frequency)
```

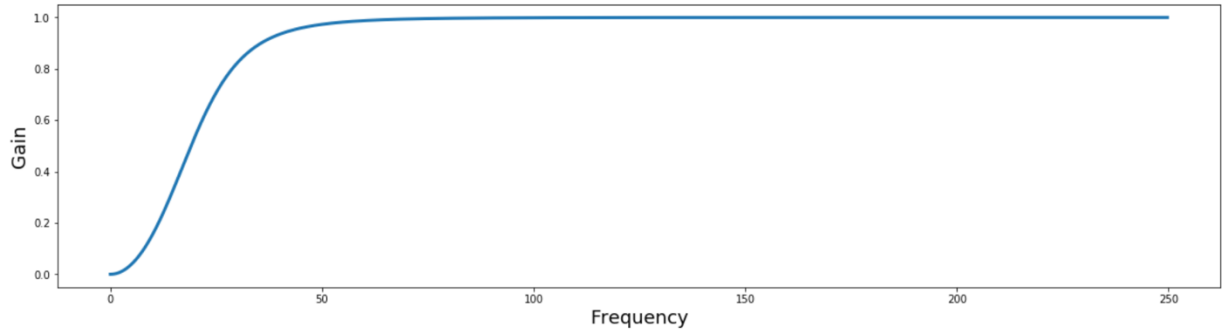


Figure 10

El parámetro ba es para la compatibilidad con versiones anteriores.

Cómo se ve el filtro en el dominio temporal. Tomé la FFT inversa y la trace para ver cómo se ve como un núcleo en el dominio temporal. Observe cómo cambiar el orden del filtro agrega más ondas en el dominio del tiempo.

```
from scipy.signal import sosfreqz

filter_order = 8
sos = butter(filter_order, frequency_cutoff, btype='high', output='sos', fs=sampling_frequency)
w_sos, h_sos = sosfreqz(sos)

plt.plot(ifft(h_sos)[0:100], linewidth=3)
plt.ylabel('Amplitude', fontsize=18)
plt.xlabel('Time', fontsize=18)
```

```
/Users/monicaaraneda/opt/anaconda3/lib/python3.7/site-packages/numpy/core/_asarray.py:85: ComplexWarning: Casting com
plex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

```
Text(0.5, 0, 'Time')
```

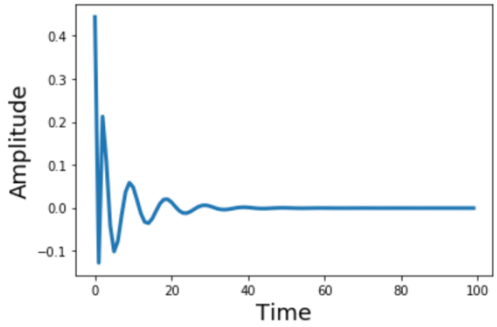


Figure 11

Ahora apliqué el filtro a mis datos. Apliqué el filtro a la señal en el dominio del tiempo utilizando la función filtfilt. Esta es una buena opción predeterminada, aunque hay varias otras funciones para aplicar el filtro. filtfilt aplica el filtro hacia adelante y luego hacia atrás asegurando que haya una distorsión de fase cero.

```
fourier_fft = fft(signal)
fourier_ifft = ifft(fourier_fft)

filtered = filtfilt(b, a, signal)

plt.figure(figsize=(20,5))
plt.plot(signal, linewidth=2)
plt.plot(filtered, linewidth=2)
plt.ylabel('Intensity', fontsize=18)
plt.xlabel('Time', fontsize=18)
plt.legend(['Original','Filtered'], fontsize=18)
```

```
<matplotlib.legend.Legend at 0x222ab8ead0>
```

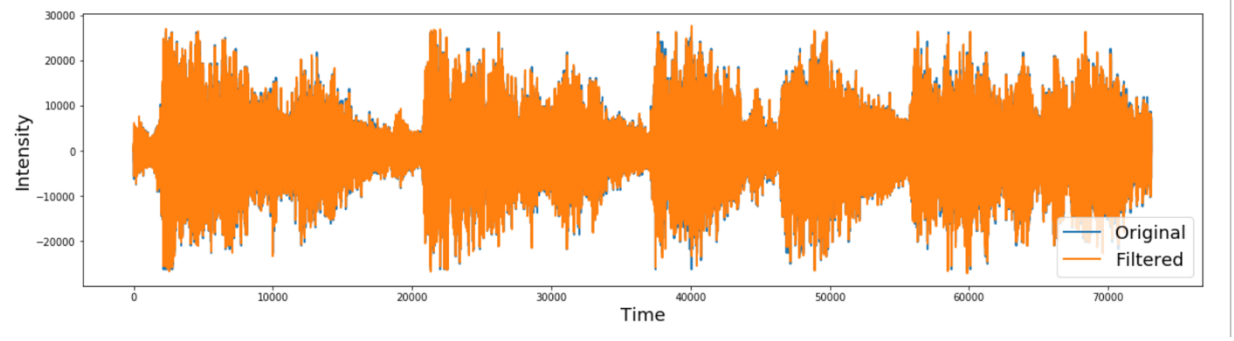


Figure 12

Los filtros Low Pass solo retienen las señales de baja frecuencia, lo que elimina cualquier información de alta frecuencia.

```
from scipy.signal import butter, filtfilt
filter_order = 2
frequency_cutoff = 10
sampling_frequency = 500
# Create the filter
b, a = butter(filter_order, frequency_cutoff, btype='low', output='ba', fs=sampling_frequency)
# Apply the filter
filtered = filtfilt(b, a, signal)
plt.figure(figsize=(20,5))
plt.plot(signal, linewidth=2)
plt.plot(filtered, linewidth=4)
plt.ylabel('Intensity', fontsize=18)
plt.xlabel('Time', fontsize=18)
plt.legend(['Original','Filtered'], fontsize=18)
```

```
<matplotlib.legend.Legend at 0x3a5f42f50>
```

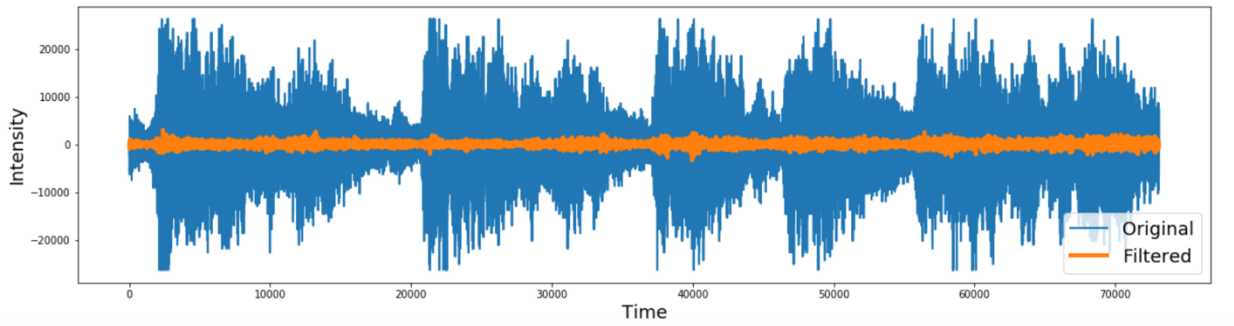


Figure 13

Ahora como se ve el Filtro.

```
filter_order = 10
frequency_cutoff = 10
sampling_frequency = 500

# Create the filter
b, a = butter(filter_order, frequency_cutoff, btype='low', output='ba', fs=sampling_frequency)

plot_filter(b, a, sampling_frequency)
```

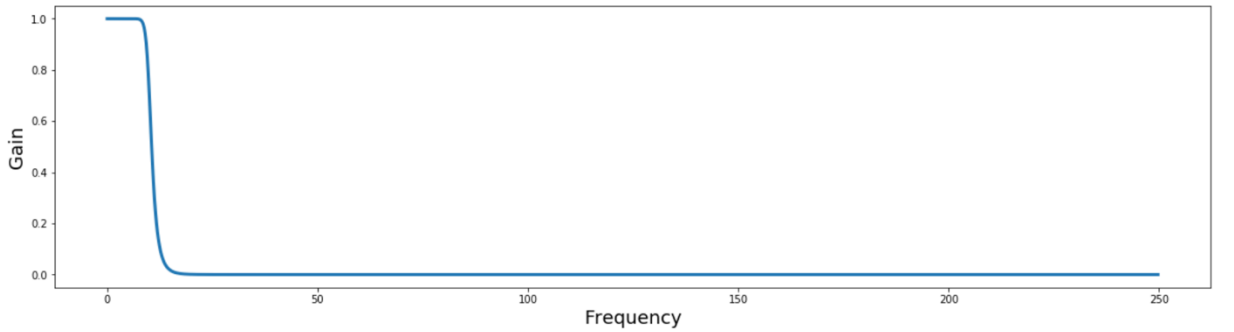


Figure 14

Los filtros de paso de banda permiten retener solo una frecuencia específica. Voy a seleccionar y eliminar frecuencias específicas.

```
filter_order = 2
lowcut = 7
highcut = 13

# Create the filter
b, a = butter(filter_order, [lowcut, highcut], btype='bandpass', output='ba', fs=sampling_frequency)

# Apply the filter
filtered = filtfilt(b, a, signal)

plt.figure(figsize=(20,5))
plt.plot(signal, linewidth=2)
plt.plot(filtered, linewidth=4)
plt.ylabel('Intensity', fontsize=18)
plt.xlabel('Time', fontsize=18)
plt.legend(['Original','Filtered'], fontsize=18)
```

```
<matplotlib.legend.Legend at 0x2280f6e690>
```

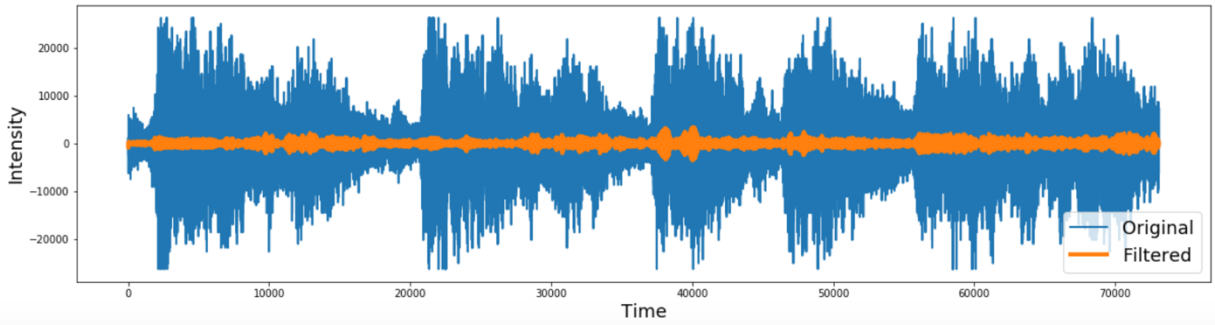


Figure 15

Los filtros Bandstop eliminan una frecuencia específica de la señal

```
b,a = butter(filter_order, [lowcut, highcut], btype='bandstop', output='ba', fs=sampling_frequency)
# Plot the filter
plot_filter(b, a, sampling_frequency)
# Apply the filter
filtered = filtfilt(b, a, signal)
plt.figure(figsize=(20,5))
plt.plot(signal, linewidth=2)
plt.plot(filtered, linewidth=2)
plt.ylabel('Intensity', fontsize=18)
plt.xlabel('Time', fontsize=18)
plt.legend(['Original','Filtered'], fontsize=18)
```

```
<matplotlib.legend.Legend at 0x8dddeeb50>
```

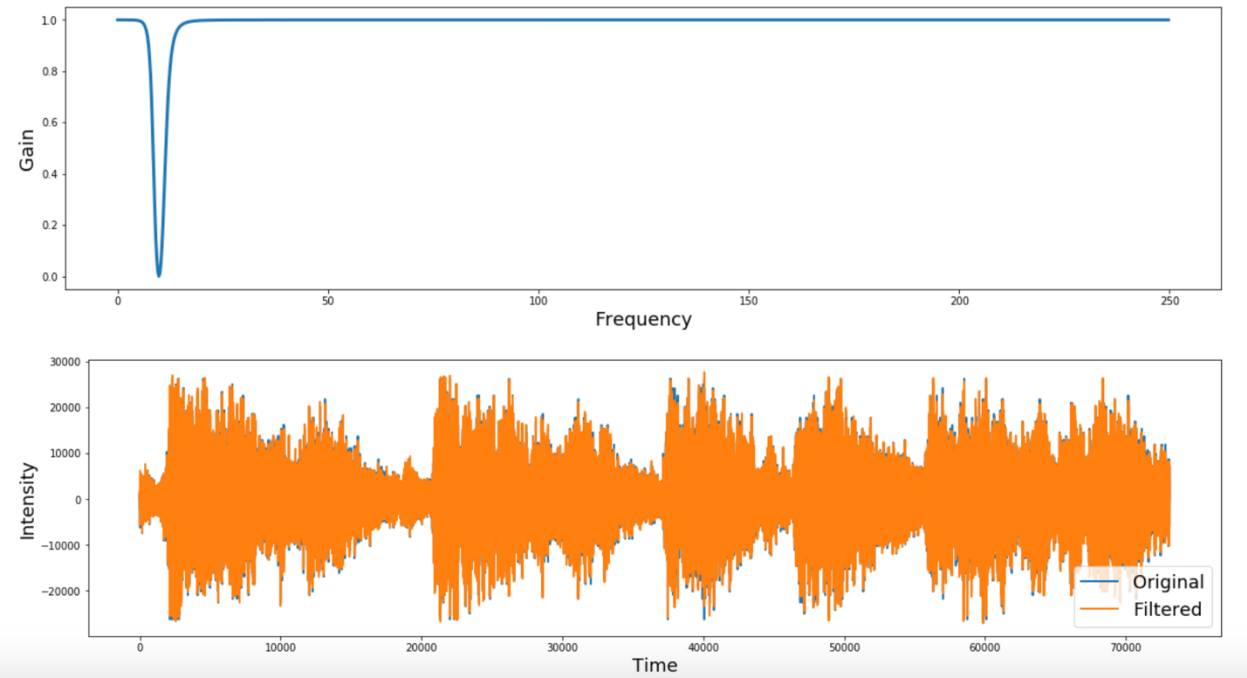


Figure 16

La transformada de Fourier y el espectrograma de la señal filtrada y sus resultados

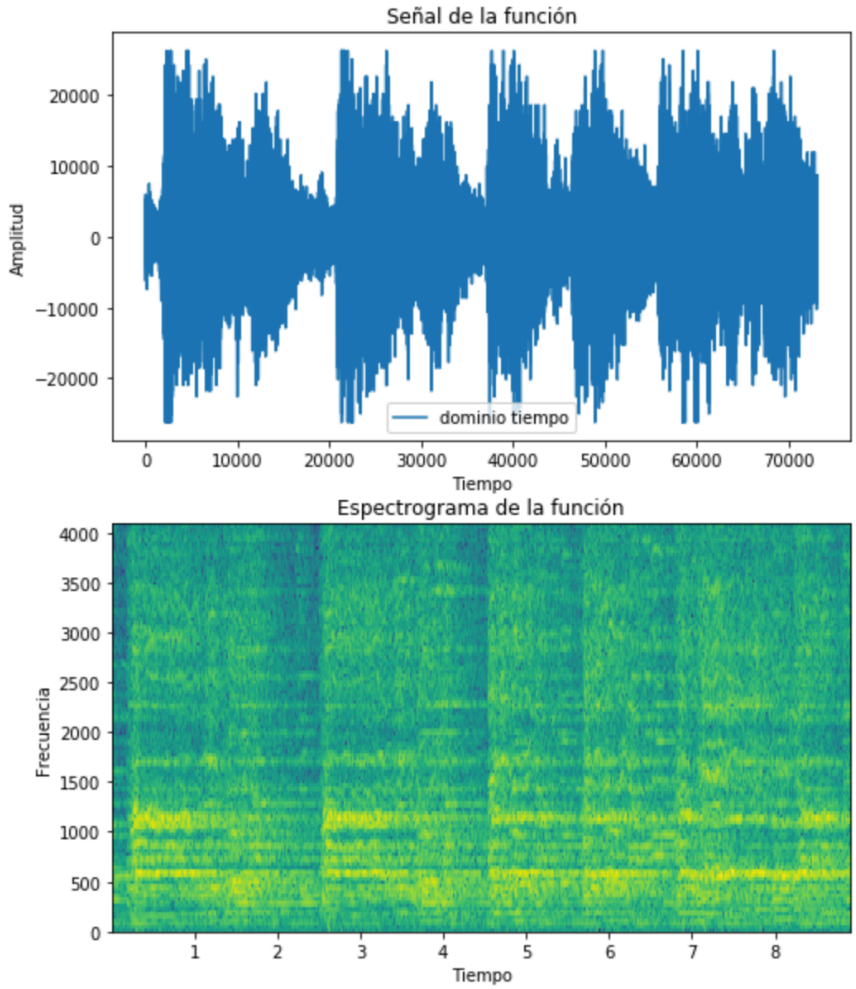


Figure 17

En la figura 16 se observa el espectrograma de una muestra generada por la señal de audio handel.wav.

En la figura se observa el espectrograma del barrido en color amarillo y se observa un pico significativo en la gráfica que representa el punto de coincidencia de la muestra tomada sobre el espectrograma que se intensifica bajo el rango de 1500 Hz. Adicionalmente se puede ver que la señal contiene interferencia, ruido.

La ventana Hamming nos ayuda a tomar un fragmento de la información en la ventana de la señal original, esto ayuda a filtrar frecuencias ruidosas que aparecen en la señal.

Hay diferentes tipos de funciones de ventana que se pueden aplicar dependiendo de la señal. Para comprender cómo una ventana determinada afecta al espectro de frecuencia. En este caso se aplico la ventana se Hamming, porque se acerca a la señal digital. Las funciones de ventana Hamming tienen una forma sinusoidal y estas ventanas resultan en un pico amplio, pero en los laterales son bajos.

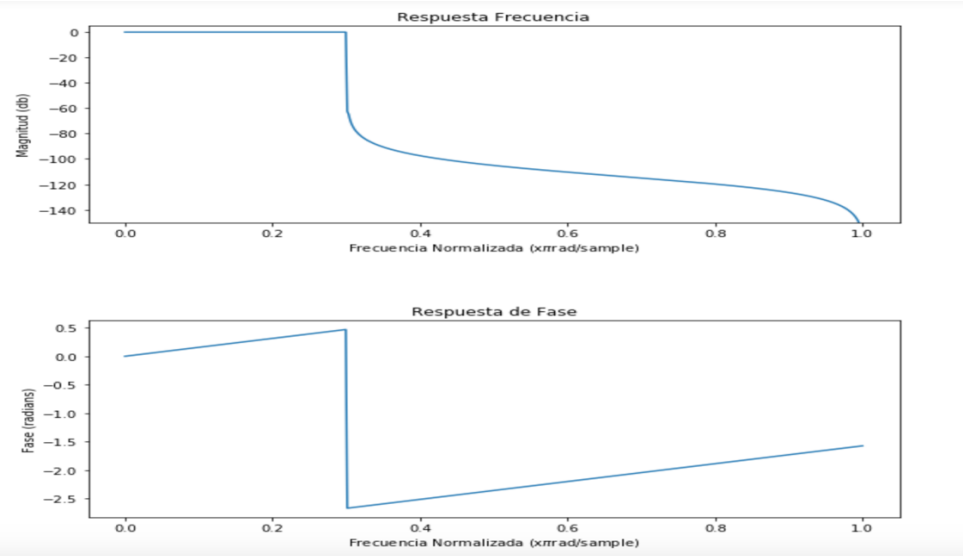


Figure 18

Filtro FIR

Se pasó una señal de entrada con ruido de alta frecuencia a través de un filtro de paso bajo. La salida resultante tiene el ruido de alta frecuencia eliminado, lo que resulta en una señal limpia archivo filtered.wav.

Un filtro toma una señal de dominio de tiempo como entrada, modifica el contenido de frecuencia y emite una nueva señal de dominio de tiempo. Esto puede ser útil en una variedad de aplicaciones.

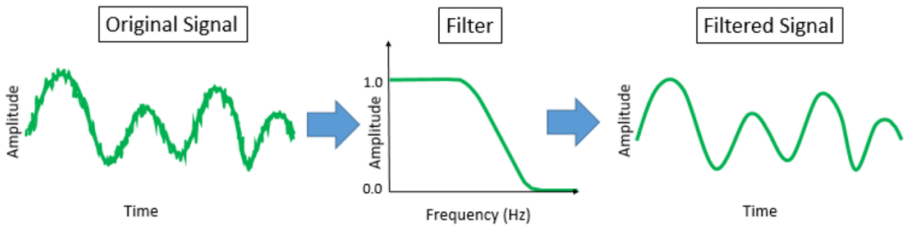


Figure 19

El Audio Filtrado, se adjuntará en el informe.

Conclusiones

Con el análisis de fourier se puede demostrar que cualquier señal está constituida por componentes senoidales de distintas frecuencias. Para cada señal hay: una función en el dominio del tiempo s(t) que determina la amplitud de la señal en cada instante de tiempo. Una función en el dominio de la frecuencia s(f) que especifica las frecuencias constitutivas de la señal, la que podrá ser discreta o continua.

El espectro de una señal es el conjunto de frecuencias que la constituyen. El ancho de banda absoluto de una señal es la anchura del espectro; frecuentemente es infinito, si la mayor parte de la energía de la señal se concentra en una banda de frecuencias relativamente estrecha, se la denomina ancho de banda efectivo o ancho de banda.

En lo personal, fue muy interesante unir conceptos matemáticos aplicados a las señales de sonido, creo que es un camino que recién inicio.

Bibliography

Siemens Digital Industry Software Inc. (2019). Article Title. *Journal Title*, Pages From - To https://community.sw.siemens.com/s/article/introduction-to-filters-fir-versus-iir.

Pastell, M. (2013, Mayo 20). *FIR filter design with Python and SciPy*. Retrieved from http://mpastell.com/pweave/_downloads/FIR_design_rst.html

Notas al pie

[1]La distribución de Anaconda viene con 1.500 paquetes seleccionados de PyPI, así como el paquete de conda y el administrador de entorno virtual. También incluye una GUI, Anaconda Navigator, como alternativa gráfica a la interfaz de línea de comando (CLI).

Tabla de Figuras

Recursos

Se adjunta carpeta con lo siguiente:

Código Fuente

El archive es un Jupiter Notebook, similar al que usa el profesor en clases.

Read-WAV-File.ipynb

Recursos de audio

Handel.wav

filtered.wav

audio_2.wav (noisy)