UNIVERSIDAD NACIONAL DE TRUJILLO

facultad de ingeniería
programa de ingeniería mecatrónica

LABORATORIO 1 – FILTRO MEDIA MÓVIL

OPERACIONES - SEÑALES FUNDAMENTALES - SISTEMAS

PROCESAMIENTO DIGITAL DE SEÑALES E IMÁGENES

ESTUDIANTE(S) :
CORTEZ GOMEZ, BRAJAN LEONEL
ESPINOZA LEÓN KARL ALEJANDRO
GUTIERREZ GUTIERREZ ITALO AARON

DOCENTE :
MS. ING. EMERSON MÁXIMO ASTO RODRIGUEZ

CICLO :
2023 - II

Trujillo, Perú
2023

Resumen

Este siguiente informe tiene como finalidad la resolución del problema planteado en la guía de práctica y/o laboratorio, proporciona el diseño de un algoritmo para el filtrado de un audio, y en específico, a la implementación de un filtro de media móvil y el filtro gaussiano a un audio cualquiera de formato wav. Esto con la finalidad de lograr una mejor comprensión de la transformación y modificación de las propiedades de una señal digital, esencial en el estudio del procesamiento digital de señales e imágenes.

En el informe también se resuelven unas preguntas planteadas en el test de comprobación, con la finalidad de absolver toda duda y lograr una mejor comprensión de las bases de la clase de la semana 2. En resumen el informe presenta una revisión de los temas abordados en clase correspondientes al filtrado de señales, proporcionando así una comprensión inicial de los conceptos base del curso.

Índice

RESULTADOS

Filtro de media móvil

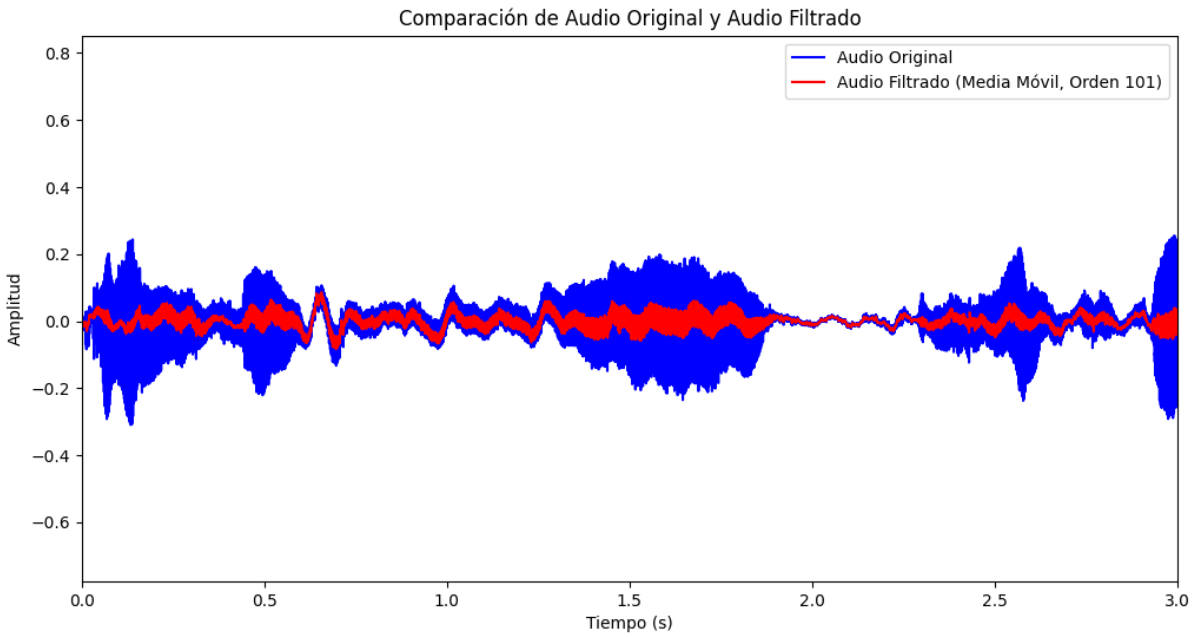
A Continuación se encuentra el código implementado para la filtración de la señal de un audio cualquier en formato wav (para este caso en específico llamado “Im_Superman.wav”).

```
import soundfile as sf

import numpy as np

import matplotlib.pyplot as plt



# Cargamos el archivo de audio WAV

audio_file = "Im_Superman.wav"

audio_data, sample_rate = sf.read(audio_file)



# Aplicamos el filtro de media móvil de orden 101

def media_movil(audio_data, orden):

    filtro = np.ones(orden) / orden

    audio_filtrado = np.convolve(audio_data, filtro, mode='same')

    return audio_filtrado



orden_filtro = 101

k = 50  # Cambiamos el valor de k a 50 para obtener un filtro de orden
101

audio_filtrado = media_movil(audio_data, 2 * k + 1)



# Definimos el rango de tiempo de 0 a 3 segundos

inicio_tiempo = 0

fin_tiempo = 3
```

```
# Creamos un arreglo de tiempo para el rango especificado

tiempo = np.arange(0, len(audio_data)) / sample_rate



# Graficamos el audio original y el audio filtrado en un solo gráfico

plt.figure(figsize=(12, 6))

plt.plot(tiempo, audio_data, color='b', label='Audio Original')

plt.plot(tiempo,  audio_filtrado,  color='r',  label=f'Audio  Filtrado
(Media Móvil, Orden {orden_filtro})')



# Limitamos el rango de tiempo en el gráfico

plt.xlim(inicio_tiempo, fin_tiempo)



# Agregamos etiquetas y leyenda

plt.title("Comparación de Audio Original y Audio Filtrado")

plt.xlabel("Tiempo (s)")

plt.ylabel("Amplitud")

plt.legend()



plt.show()
```

Figura 2

Gráfica que muestra el código implementado



Comparación de la gráfica de la señal original (con ruido) y la filtrada por método media móvil

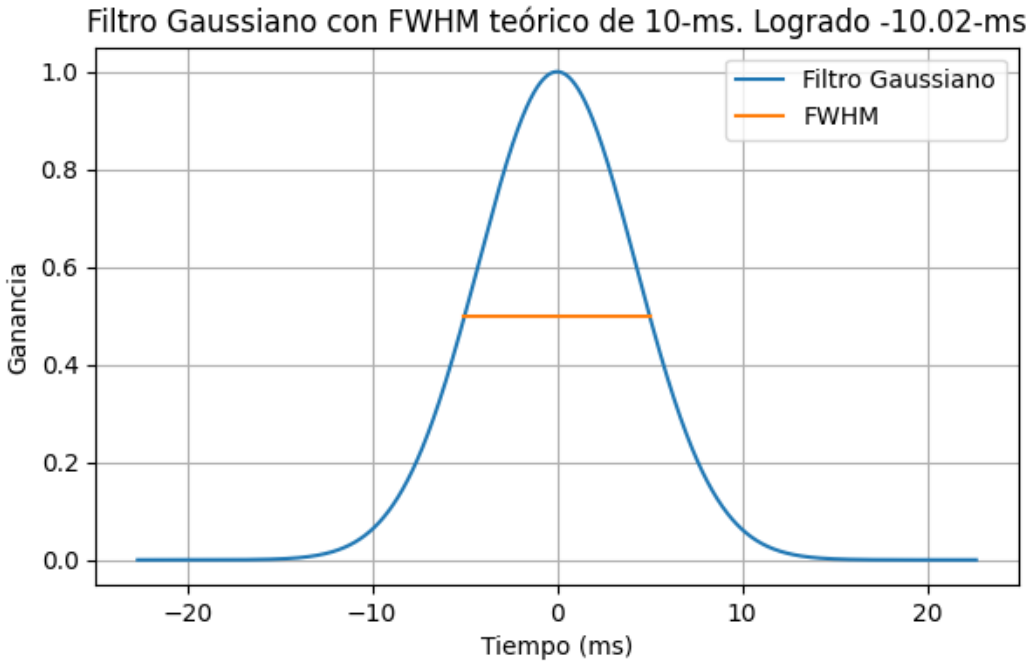
En la gráfica de ambas señales de audio (original y modificado) se puede notar que la señal modificada presenta menos perturbaciones con respecto a la señal original, el ruido es mucho menor en la señal filtrada, por ende, tiene menos distorsión. Ambos presentan efectos de sonido de fondo pero la señal filtrada presenta menos, debido a que fue alterada usando comandos de la librería pysoundfile y el algoritmo realizado por el grupo para obtener dicha comparación de señales.

Filtro Gaussiano

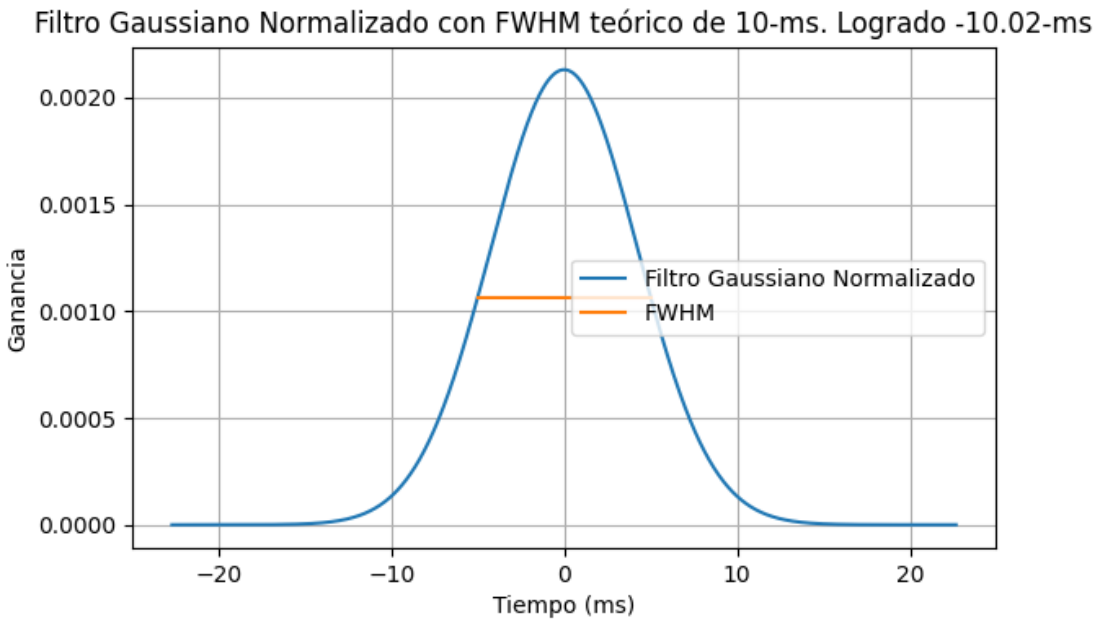
A Continuación, se presenta el código para la filtración de una señal de audio en formato wav (para este caso en específico llamado “Im_Superman.wav”).

```
[ ]  FWHM =int(10)   # Establecemos un FWHM teórico de 25 ms
     fm = int(sample_rate)
     k = 1000
     gt = 1000*np.arange(-k,k)/fm  # tiempo normalizado de la función gaussiana en ms.

     filtro_gaussiano = np.exp( -(4*np.log(2)*gt**2) / FWHM**2)   #Creación del filtro gaussiano.
     filtro_gaussiano_normalizado = filtro_gaussiano / np.sum(filtro_gaussiano)    #Normalizado de la ganancia a 1

     # Cálculo empírico del tamaño de FWHM en ms

     ind_flanco_bajada = k + np.argmin( (filtro_gaussiano[k:]-.5)**2)    #Índice de la mitad del flanco de subida
     ind_flanco_subida = np.argmin( (filtro_gaussiano-.5)**2)    #Índice de la mitad del flanco de subida.

     FWHM_calculado = gt[ind_flanco_subida] - gt[ind_flanco_bajada]    #Duración del FWHM en ms

     sen_filtrada_gauss = np.zeros_like(audio_data)

     n = len(audio_data)
```

```
[38] plt.subplots(1, 2, figsize=(15,4))

     plt.subplot(121)
     plt.plot(gt,filtro_gaussiano, label="Filtro Gaussiano") #Gráfica del a función gaussiana construida
     plt.plot([gt[ind_flanco_subida], gt[ind_flanco_bajada]],
              [filtro_gaussiano[ind_flanco_subida],filtro_gaussiano[ind_flanco_bajada]],
              label = "FWHM")    #Gráfica de la línea FWHM
     plt.title(f"Filtro Gaussiano con FWHM teórico de {FWHM}-ms. Logrado {round(FWHM_calculado,2)}-ms")
     plt.xlabel("Tiempo (ms)")
     plt.ylabel("Ganancia")
     plt.grid()
     plt.legend()

     plt.subplot(122)
     plt.plot(gt,filtro_gaussiano_normalizado, label="Filtro Gaussiano Normalizado")   #Gráfica de la función gaussiana
     plt.plot([gt[ind_flanco_subida], gt[ind_flanco_bajada]],
              [filtro_gaussiano_normalizado[ind_flanco_subida], filtro_gaussiano_normalizado[ind_flanco_bajada]],
              label = "FWHM")    #Gráfica de la línea FWHM
     plt.title(f"Filtro Gaussiano Normalizado con FWHM teórico de {FWHM}-ms. Logrado {round(FWHM_calculado,2)}-ms")
     plt.xlabel("Tiempo (ms)")
     plt.ylabel("Ganancia")
     plt.grid()
     plt.legend()

     plt.show()
```



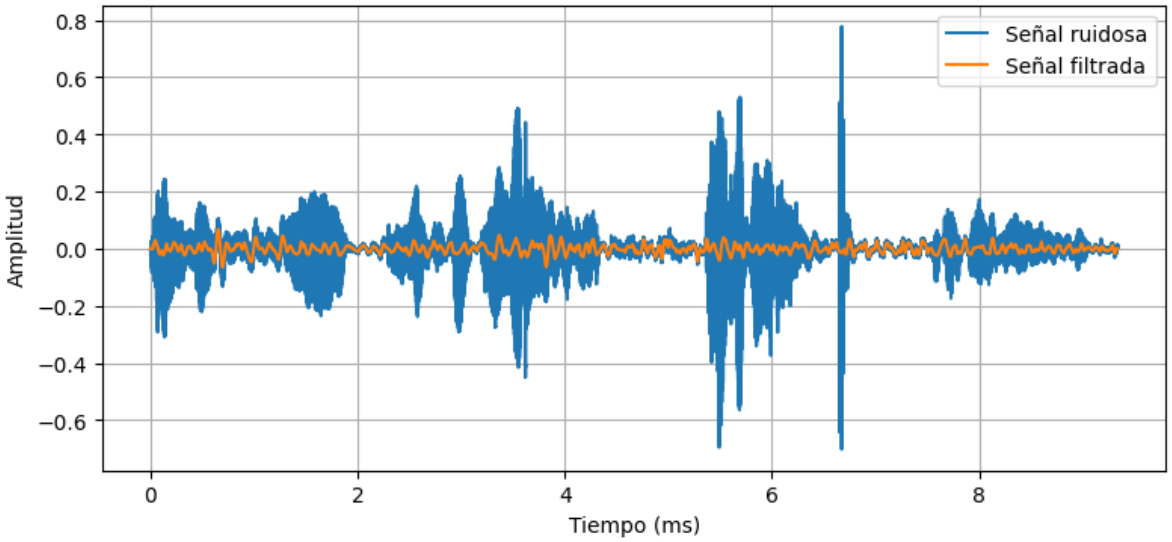
Gráfica de filtro Gaussiano con un FWHM = 10ms teórico



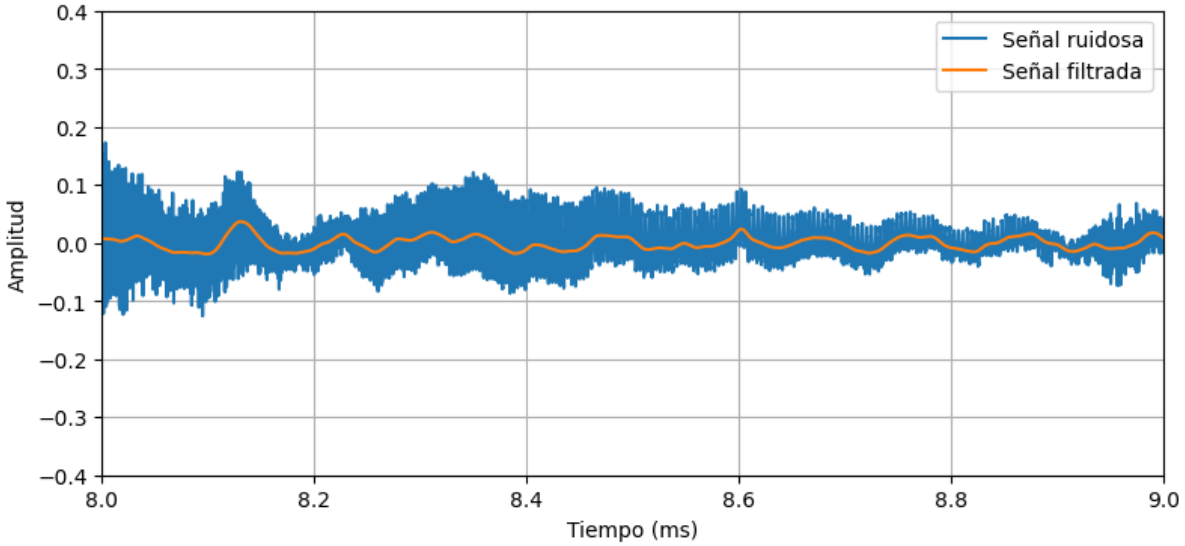
Gráfica de filtro Gaussiano normalizado con un FWHM = 10ms teórico

```
[35] for i in range(k+1,n-k-1):  #Los indices no tienen que salir del rango de la señal
       sen_filtrada_gauss[i] = np.sum(audio_data[i-k:i+k]*filtro_gaussiano_normalizado)   #Cada salida es ponderado
```

```
[36] plt.subplots(1,2, figsize = (20,4))

     plt.subplot(121)
     plt.plot(np.arange(0,len(audio_data))/fm, audio_data, label="Señal ruidosa")
     plt.plot(np.arange(0,len(audio_data))/fm, sen_filtrada_gauss,label="Señal filtrada")
     plt.xlabel("Tiempo (ms)")
     plt.ylabel("Amplitud")
     plt.grid()
     plt.legend()


     plt.subplot(122)
     plt.plot(np.arange(0,len(audio_data))/fm, audio_data, label="Señal ruidosa")
     plt.plot(np.arange(0,len(audio_data))/fm, sen_filtrada_gauss,label="Señal filtrada")
     plt.axis([8, 9, -0.4, 0.4])
     plt.xlabel("Tiempo (ms)")
     plt.ylabel("Amplitud")
     plt.grid()
     plt.legend()
     plt.show()
```



Gráfica “Señal ruidosa” Vs “Señal filtrada”



Gráfica“Señal ruidosa” Vs “Señal filtrada” en dominio de 8 a 9

Comparación de la gráfica de la señal original (con ruido) y la filtrada por método Gaussiano

Se aprecia un cambio significativo entre la señal ruidosa y la filtrada, siendo este la eliminación del ruido de la señal original, obteniendo así, una señal más suave y limpia. Esta señal post-filtro es mucho más suave que la señal filtrada por el método de media móvil, ya que en el filtrado Gaussiano se pondera cada dato de la señal con ruido de forma distinta y correspondiente a que tan cerca esté del dato actual de la señal, a diferencia de la media móvil donde esta ponderación es igual para cada dato de la señal.

TEST DE COMPROBACIÓN

a. ¿Cuál es la diferencia entre una señal estocástica de una señal determinística? Explique.

Se diferencian sobre todo en la naturaleza de su comportamiento, debido a que uno es predecible o previsible, mientras que el otro no. Las determinísticas como su mismo nombre lo dice, se pueden determinar, pues cuentan con ecuaciones matemáticas que las describen como por ejemplo las señales sinusoidales. Mientras que por otro lado se encuentran las estocásticas, que no se pueden predecir con certeza y que a menudo se modelan utilizando conceptos de probabilidad y estadísticas.

b. ¿Cuál es la tasa de muestreo de la señal? Explique.

La tasa de muestreo es la cantidad de muestras o mediciones que se toman por unidad de tiempo de una señal continua para convertirla en una señal discreta. Se expresa típicamente en hercios (Hz) o muestras por segundo (S/s).

c. ¿El filtro media móvil implementado es causal? Explique.

Un filtro de media móvil si es causal, pues calcula la salida en función de un promedio de las muestras de entrada anteriores y actuales, pero no utiliza ninguna muestra de entrada futura. Esto cumple con la definición de un filtro causal, ya que su salida en cualquier punto de tiempo solo depende de datos pasados o presentes, y no de datos futuros. Por lo tanto, teóricamente, un filtro de media móvil es un ejemplo de un filtro causal.

BIBLIOGRAFÍA

Oppenheim, A. V., Willsky, A. S., & Young, I. T. (1983). *Signals and systems*. Englewood Cliffs, N.J: Prentice-Hall.

Kamen, Edward W., y Bonnie S. Heck. (2008). Fundamentos de señales y sistemas usando la Web y MATLAB® PEARSON EDUCACIÓN, México, ISBN: 978-970-26-1187-5