UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de ingeniería
Programa de ingeniería mecatrónica

LABORATORIO 2 – CONVOLUCIÓN

PROPIEDADES DE SISTEMAS – ANÁLISIS DE FOURIER

PROCESAMIENTO DIGITAL DE SEÑALES E IMÁGENES

ESTUDIANTE(S) :
CORTEZ GOMEZ BRAJAN LEONEL
ESPINOZA LEÓN KARL ALEJANDRO
GUTIÉRREZ GUTIÉRREZ ITALO AARÓN

DOCENTE :
MS. ING. EMERSON MÁXIMO ASTO RODRIGUEZ

CICLO :
2023 - II

Trujillo, Perú
2023

Resumen

El presente informe tiene como finalidad lograr la resolución del problema planteado en la guía de laboratorio 02. Dicha resolución proporciona el diseño de un algoritmo para modificar un audio usando la convolución. Específicamente, se diseñó un algoritmo para leer cualquier audio wav utilizando la librería pysoundfile, luego se creó un kernel gaussiano de 31 muestras y se realizó la convolución. Esto se llevó a cabo con la finalidad de modificar las propiedades de una señal digital, esencial en el estudio del procesamiento digital de señales e imágenes, además de comparar y analizar las gráficas del audio original y del audio modificado.

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 4. En resumen el informe presenta una revisión de los temas abordados en clase relacionados a la modificación de señales, proporcionando así una comprensión inicial de los conceptos base del curso.

Índice

RESULTADOS

4.1. Importación de Librerías

- **Librerias a utilizar:**

```
[100] import numpy as np
      import matplotlib.pyplot as plt
      import pylab as pl
      import time
      from IPython import display
      plt.style.use(['dark_background']) #Para gráficas para temas oscuros

      #agregamos el audio que usaremos y la librería pysoundfile
      #!wget -nc /content/hello_moto.wav
      !wget -nc https://vvestman.github.io/summerschool19/sounds/Im_Superman.wav

      !pip install pysoundfile
      !pip install bitstring

      audio_file = "Im_Superman.wav"
```

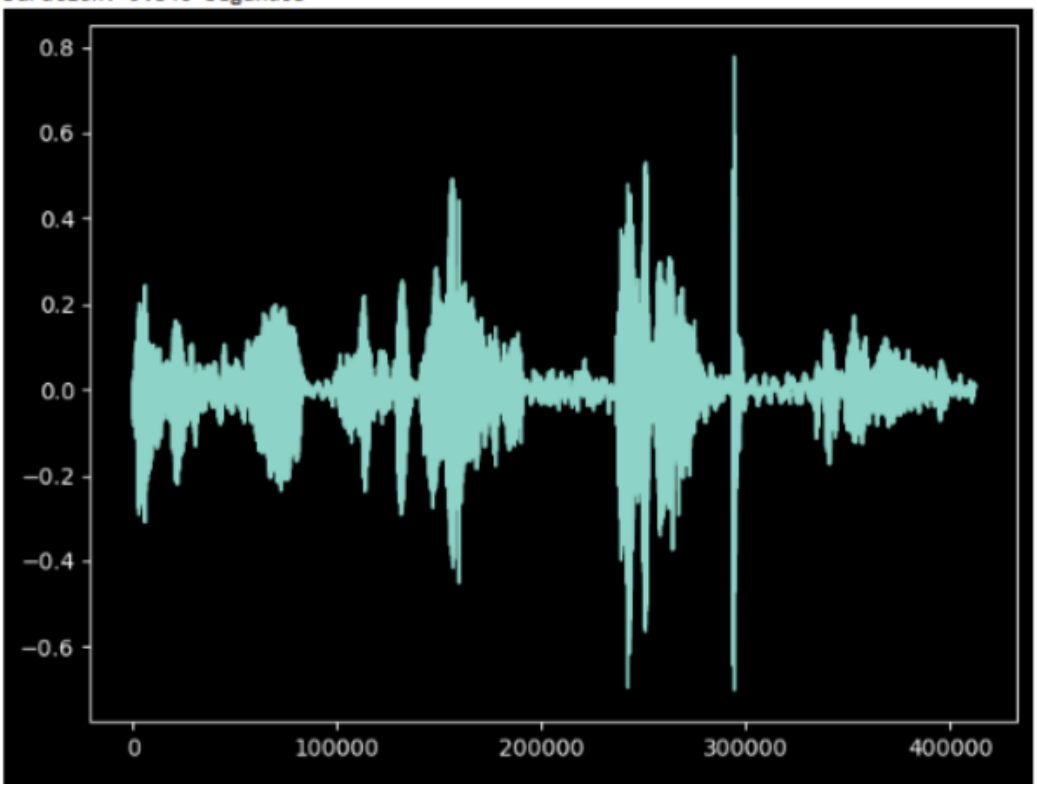
```
File ‘Im_Superman.wav’ already there; not retrieving.

Requirement already satisfied: pysoundfile in /usr/local/lib/python3.10/dist-packages (0.9.0.post1)
Requirement already satisfied: cffi>=0.6 in /usr/local/lib/python3.10/dist-packages (from pysoundfile) (1.15.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=0.6->pysoundfile) (2.21)
Requirement already satisfied: bitstring in /usr/local/lib/python3.10/dist-packages (4.1.2)
Requirement already satisfied: bitarray<3.0.0,>=2.8.0 in /usr/local/lib/python3.10/dist-packages (from bitstring) (2.8.1)
```

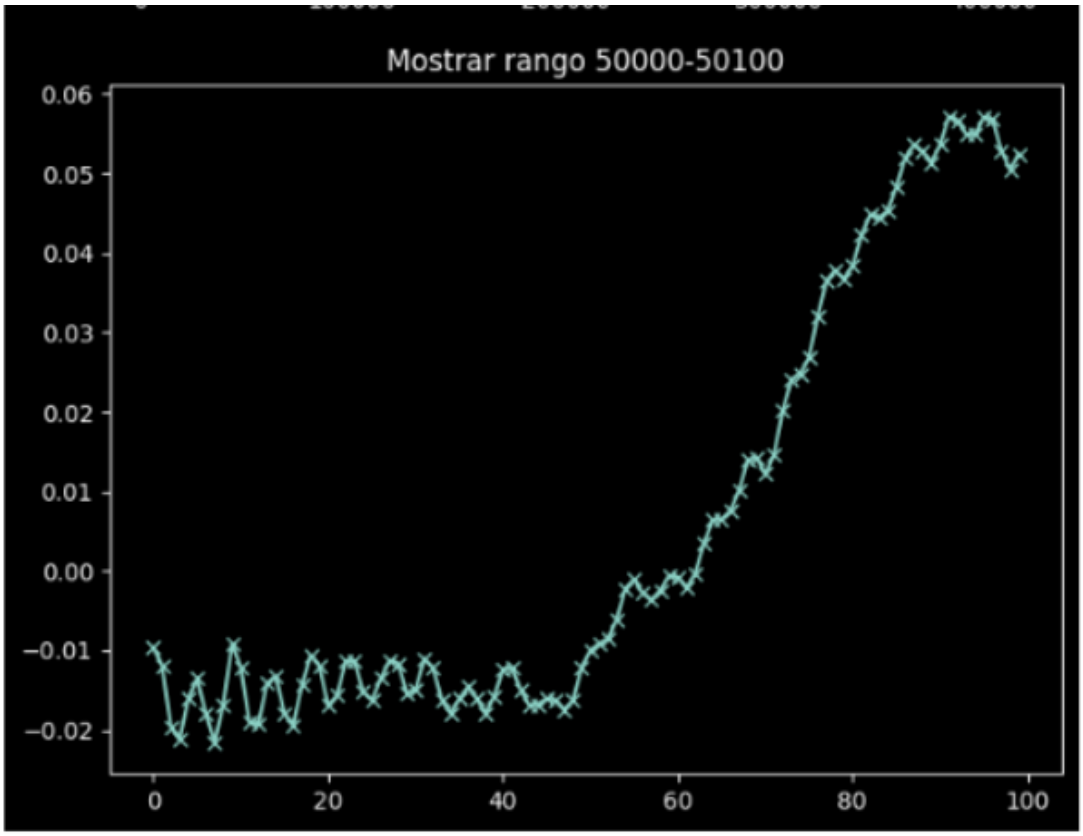
```
[101] import IPython
      IPython.display.Audio("Im_Superman.wav")
```

```python
import soundfile
#import matplotlib.pyplot as plt

# Cargamos el archivo de audio WAV
audio_senial, sampling_rate = soundfile.read(audio_file)

# Imprimimos información sobre la señal
print('Tasa de muestreo: {} muestras/segundo'.format(sampling_rate))
print('Tamaño de la señal: {} muestras'.format(audio_senial.shape[0]))
print('Duración: {:.3f} segundos'.format(audio_senial.shape[0] / sampling_rate))

# Graficamos la señal de audio
plt.plot(audio_senial)
plt.tight_layout()
plt.figure()
plt.plot(audio_senial[50000:50100], marker='x')
plt.title("Mostrar rango 50000-50100")
plt.tight_layout()

# abajo se muestra lo codificado:
```

```
[102] # abajo se muestra lo codificado:
```

```
Tasa de muestreo: 44100 muestras/segundo
Tamaño de la señal: 411889 muestras
Duración: 9.340 segundos
```



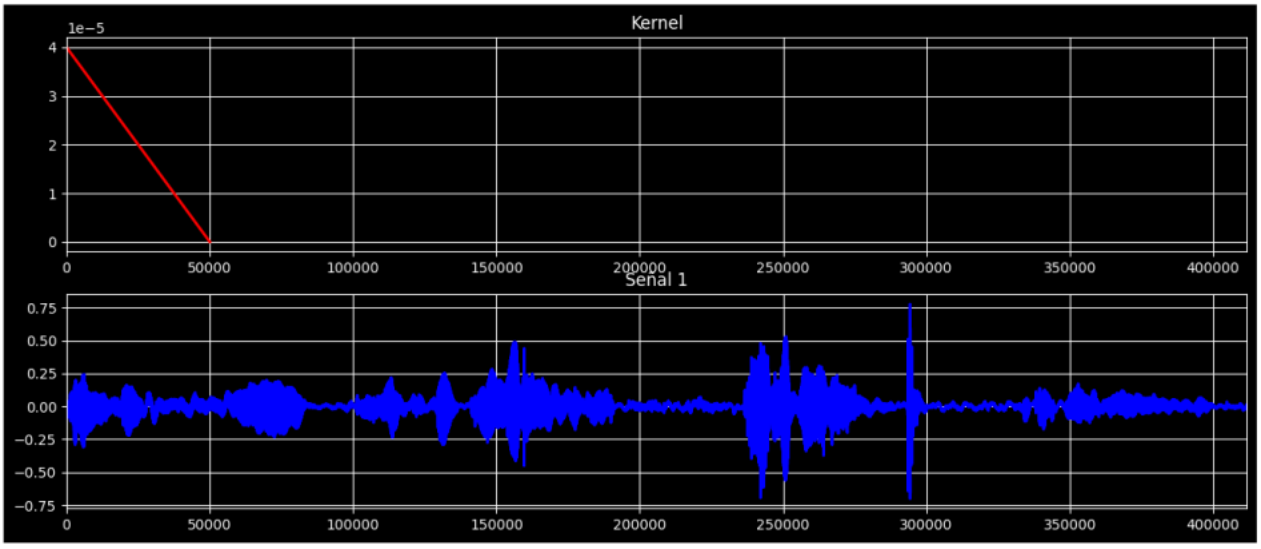


- **Creación de la señal y el kernel**

```
#creando una señal arbitraria
#senal1 = np.concatenate( (np.zeros(30), np.ones(2), np.zeros(20),np.ones(30),2*np.ones(10),np.ones(30),-np.ones(10),np.zeros(40)), axis=0)
senal1 = audio_senial

#Creando el Kernel
#kernel = np.array([0.75,0.5,0.25,0])
kernel = np.linspace(1,0,50000) #Función gaussiana
#kernel = np.exp( -np.linspace(-2,2,50000)**2) #Función gaussiana
kernel = kernel/sum(kernel)
N = len(senal1) #Longitud ded la señal
```

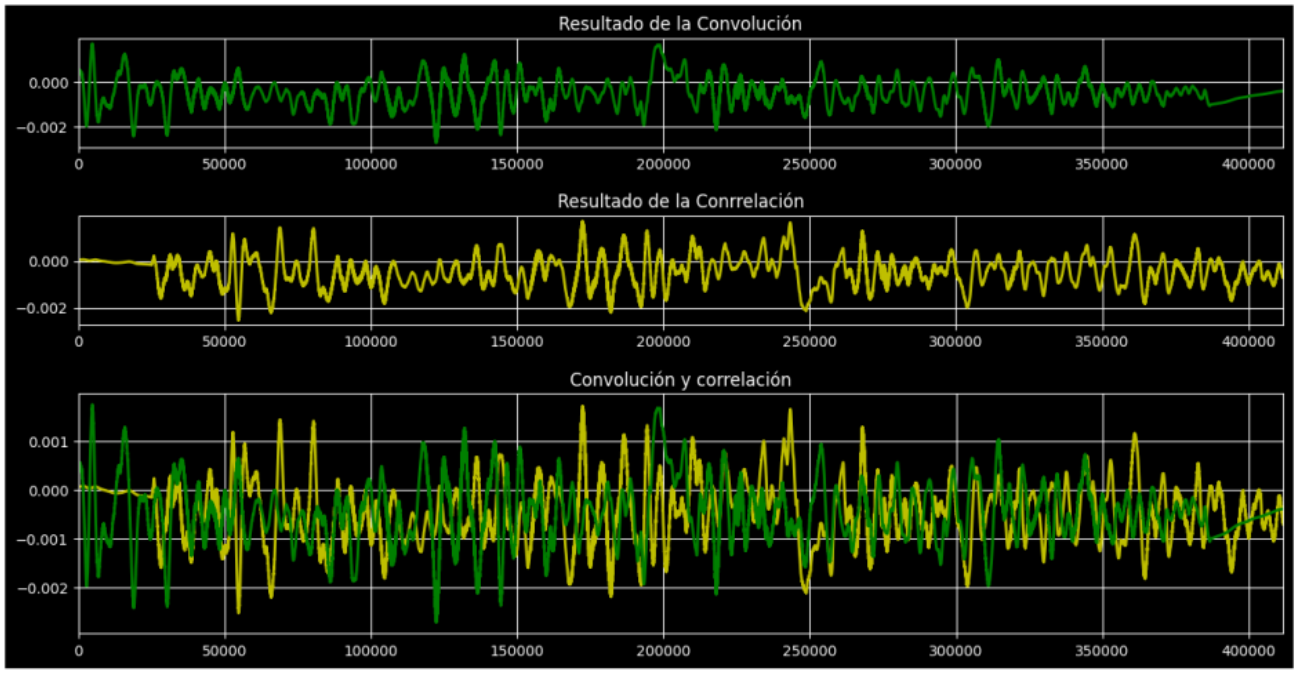
- **Mostrando señales creadas**

```
plt.figure(figsize=(15,6)) #Estableciendo el tamaño de la gráfica

#Gráfica del kernel
plt.subplot(211)
plt.plot(kernel,'r', linewidth=2)
plt.xlim([0,N]) #Limitamos el eje x entre 0 y N
plt.title('Kernel')
plt.grid()

#Gráfica de la señal
plt.subplot(212)
plt.plot(senal1, 'b', linewidth=2,)
plt.xlim([0,N]) #Limitamos el eje x entre 0 y N
plt.title('Señal 1')
plt.grid()
plt.show()
```



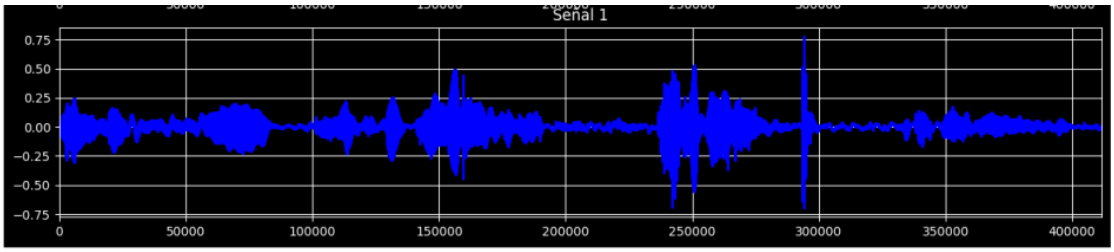
- **Cálculo y gráfica de la convolución**

```
[105] resultado = np.correlate(senal1,kernel,'sane') #El 'same indica que el resultado debe ser recortado y tener
                                                     #las mismas dimensiones que la señal de entrada. Si se dea
                                                     #el resultado completo se puede usar 'full' y no limitar el
                                                     # eje x.
      resultado_2 = (1.01)*np.convolve(senal1,kernel,'sane')
      plt.figure(figsize=(15,3))

      plt.subplot(211)
      plt.plot(resultado_2, 'g', linewidth=2)
      plt.xlim([0,N]) #Limitamos el eje x entre 0 y N
      plt.title('Resultado de la Convolución')
      plt.grid()
      plt.show()
      plt.figure(figsize=(15,3))
      plt.subplot(212)
      plt.plot(resultado, 'y', linewidth=2)
      plt.xlim([0,N]) #Limitamos el eje x entre 0 y N
      plt.title('Resultado de la Conrrelación')
      plt.grid()
      plt.show()

      plt.figure(figsize=(15,3))
      plt.plot(resultado, 'y', linewidth=2)
      plt.plot(resultado_2, 'g', linewidth=2)
      plt.xlim([0,N]) #Limitamos el eje x entre 0 y N
      plt.title('Convolución y correlación')
      plt.grid()
      plt.show()
```



Comparación entre la gráfica del audio original y del audio modificado

Audio original:



Audio modificado:

Se puede notar la diferencia entre el audio original y el audio modificado por convolución, existe un cambio en la acústica, al aplicarse la convolución a dicha señal de audio, esto hace que se perciba menos distante.

¿Qué efecto sonoro identifica entre el audio original y el modificado?

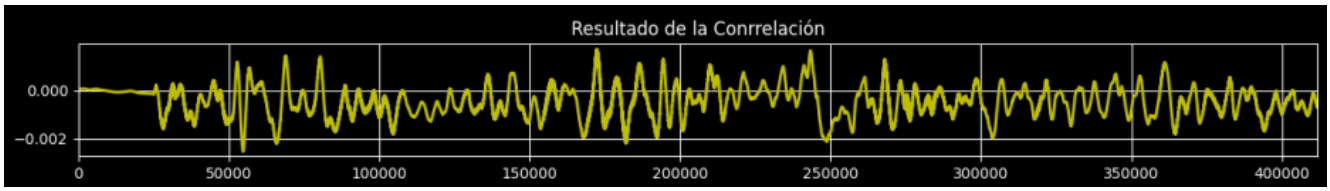
Se puede identificar el efecto sonoro de reverberación, ya que el sonido se percibe como si estuviera dentro de un entorno en específico, lo cual hace que se oiga más fuerte y abultado, a diferencia del audio modificado, el cual resulta que tiene menos reverberación que el original. Además, otro efecto que se puede notar es el filtro de tono a partir de la convolución, la cual hace que la frecuencia de la onda varíe.

TEST DE COMPROBACIÓN

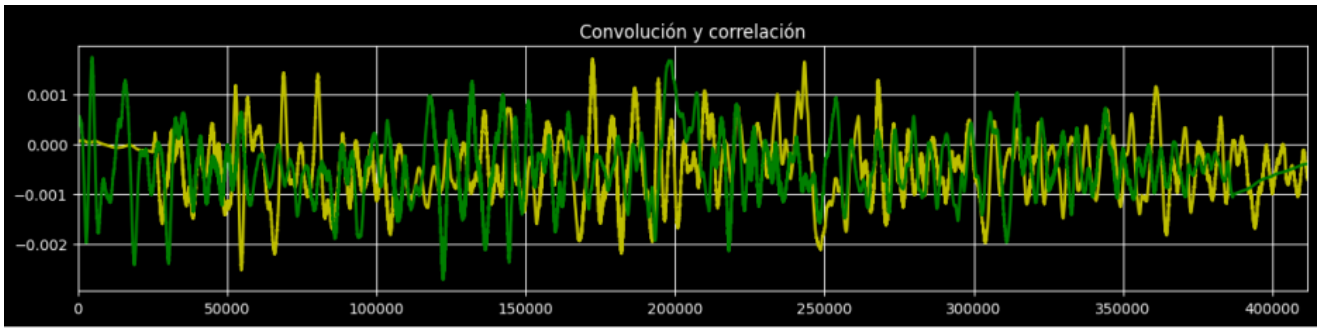
a. ¿Si en el ejercicio 5a usa la función correlate en vez de convolve que diferencias en la forma y sonoras se tendría en la señal de salida? Explique.

<u>Señal modificada con la función "convolve"</u>:

Señal modificada con la función "correlate":



Las 2 señales juntas:



No habría mucha diferencia en la forma, ya que al compararlas, tienen aproximadamente la misma amplitud y la misma frecuencia. Solo se puede notar una cierta diferencia, la cual es que, en la onda con "convolve", la señal en una parte del final es aproximadamente constante, a diferencia de "correlate", donde la señal en una parte al inicio es aproximadamente constante.

No hay mucha diferencia respecto a la reverberación entre ambas ondas, los patrones son casi los mismos.

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