Universidad Tecnológica Nacional: Facultad Regional Santa Fe

**Trabajo Práctico N°1:**

Análisis de Señales de Electroencefalograma en Epilepsia.

**Carrera**: Ingeniería en Sistemas de Información

**Cátedra**: Análisis Numérico

**Comisión**: 3A

**Año**: 2025

**Profesores**: Pablo Kler, Luis Bianculli, Nicolás Franck

**Integrantes**:

* Exequiel Farías - correo@gmail.com
* Juan Diego Paduli- correo@hotmail.com
* Bautista - [correo@gmail.com](mailto:correo@gmail.com)
* Lautaro - correo@gmail.com
* Diego Fernando Danelone – dr.danelone@gmail.com

Índice

[**Introducción 4**](#_ye5gxwy2w1mo)

[**¿Qué es un acelerograma? 4**](#_36cvh5fzyzoh)

[Gráficos de las señales 4](#_f6icmkwa2cva)

[**a. Serie discreta de Fourier y transformada discreta de Fourier 5**](#_zgwu2ni16ky)

[Coeficientes de la serie discreta de Fourier 5](#_ouu9hvw0gek)

[Transformada discreta de Fourier 6](#_9i3gxyg2t98i)

[**b. Suavizado de las señales 8**](#_8gjnv67phudw)

[**c. Frecuencias más afectadas 11**](#_7ht7ewyg0wqf)

[**d. Frecuencia más acelerada 13**](#_o28dysucbghk)

[Primera solución: sumar las TFDs 14](#_suyflge2buvs)

[Segunda solución: sumar las señales 14](#_rn1v40xhn3cz)

[Por teorema de la convolución 15](#_z9jtniycdzfe)

[**e. Ubicación de terremoto 3 16**](#_iywnfi374ezp)

[**Bibliografía 18**](#_vpjzr91uvyzx)

# 

# Introducción

En el presente trabajo práctico se expondrán las soluciones encontradas para el enunciado planteado. Para ello, utilizaremos como fuente de información los libros brindados por la cátedra, y como principal herramienta Python, empleando distintas librerías, las cuales serán especificadas a lo largo del informe. Además, se proporcionará la base teórica que fundamenta dichas resoluciones, logrando así una explicación detallada de las mismas.

El trabajo emplea datos provenientes de acelerogramas, a los cuales aplicaremos métodos matemáticos con el fin de responder una serie de incisos, y también lograr una mayor comprensión sobre las señales discretas y su funcionamiento.

# ¿Qué es un acelerograma?

Un acelerograma es una representación gráfica de la aceleración en función del tiempo, que se utiliza principalmente en ingeniería sísmica para estudiar y entender el comportamiento dinámico de estructuras ante un terremoto. En otras palabras, muestra cómo varía la aceleración experimentada por un punto específico de una estructura o suelo durante un evento sísmico.

En un acelerograma, la abscisa (eje horizontal) representa el tiempo en segundos, mientras que la ordenada (eje vertical) representa la aceleración en unidades de gravedad (g), donde 1g es la aceleración de la gravedad terrestre estándar (aproximadamente 9.81 m/s^2).

La amplitud y la duración de las sacudidas sísmicas se reflejan en la forma y la altura de las curvas del acelerograma. Este tipo de información es crucial para el diseño sísmico de estructuras y para evaluar su resistencia y seguridad ante terremotos.

## Gráficos de las señales

Para poder visualizar mejor las señales de los distintos acelerogramas brindados, utilizamos la librería matplotlib en Python.

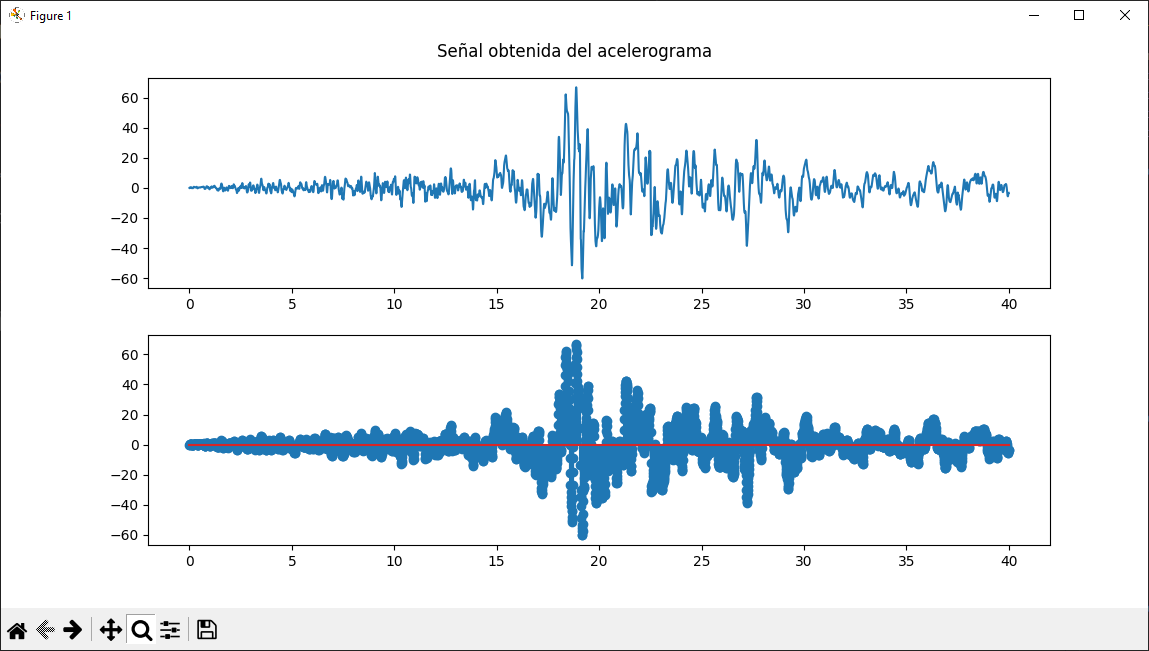


Figura 1: acelerograma del terremoto1.txt

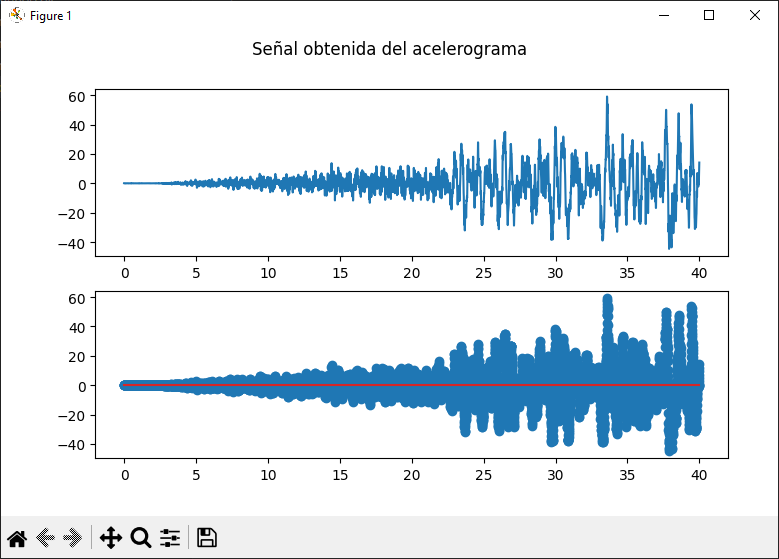


Figura 2: acelerograma del terremoto2.txt

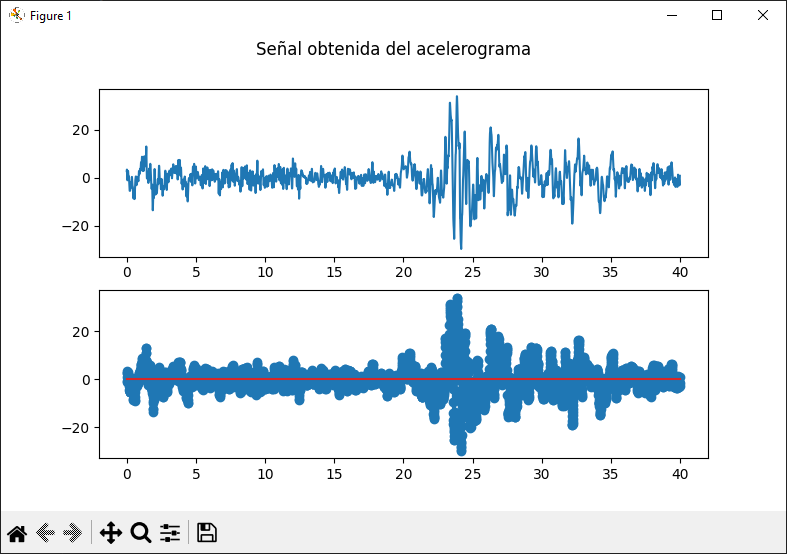


Figura 3: acelerograma del terremoto3.txt

Aunque los gráficos anteriores se muestran de manera continua, en realidad son funciones definidas en forma discreta. Para mayor claridad, hemos decidido presentarlas de este modo.

# a. Serie discreta de Fourier y transformada discreta de Fourier

## Coeficientes de la serie discreta de Fourier

Para determinar los coeficientes de la serie discreta de Fourier, nos basamos en el libro de Oppenheim: "Señales y sistemas" - 1998 capítulo 3 sección 3.6.2.

La expresión cerrada para obtener los coeficientes de la serie discreta de Fourier es la siguiente:

[1]

[2]

Este conjunto de ecuaciones determina cómo se pueden obtener los coeficientes de la serie discreta de Fourier a partir de los datos que tenemos. Específicamente, contamos con x[n]: la señal discreta del acelerograma. Sin embargo, no contamos con un período porque la señal no es periódica, por lo que vamos a considerar todo el tiempo de muestreo como nuestro período.

El código en Python es el siguiente:

# Funcion para calcular los coeficientes de la serie de Fourier de la senal

def calcular\_coeficientes(datos):

N = len(datos) #Calcula la cantidad de entradas dentro del archivo

n = np.arange(N) #Hace un arreglo con valores desde 0 a N-1

k = n.reshape((N, 1)) #Hace el arreglo n como un vector columna

M = np.exp(-2j \* np.pi \* k \* n / N) #Calcula el numero exponencial que tendra que ser multiplicado por la entrada x[n] (Datos)

return np.dot(M, [dato[1] for dato in datos]) \* (2/N)

La función calcular\_coeficientes recibe como parámetro los datos del archivo en forma de array. El objetivo de la función es calcular la serie de los distintos y retornarlos en otro array.

En primera instancia, calculamos la cantidad de entradas o cantidad de datos dentro del archivo (serán los puntos de la función discreta). Seguidamente, ordenamos los datos leídos de manera que se pueda realizar el producto punto entre la función compleja definida y los datos de entrada x[n]. La serie está representada como ese mismo producto punto entre vectores, y la exponencial compleja está definida exactamente como la expresión indicada más arriba.

El resultado serán los distintos . En total, tendremos 4000; los normalizamos dividiéndolos por 1/N y multiplicando por 2, ya que tenemos el cuenta el espectro simétrico que se espeja a la mitad de los datos.

## Transformada discreta de Fourier

Para obtener la transformada de Fourier a partir de estos coeficientes, observaremos dos ecuaciones:

* La ecuación [2], utilizada anteriormente para el cálculo de los coeficientes de la serie de Fourier.
* Nuevamente revisamos el libro Oppenheim: "Señales y sistemas" - 1998. En el capítulo 5, sección 5.1, ecuación 5.9, se define la transformada de Fourier en tiempo discreto como:

Teniendo en cuenta que plantear la sumatoria de n desde infinito negativo hasta infinito positivo es equivalente a plantear la sumatoria de K para los distintos N, notamos que si multiplicamos a [2] por N, obtenemos [3].

Es decir,

Entonces, para obtener la transformada discreta de Fourier, calcularemos la cantidad de coeficientes que tenemos y lo multiplicaremos por los coeficientes que nos devuelve la función calcular\_coeficientes. El código queda de la siguiente manera:

# Función para calcular la TFD a partir de los coeficientes de la SFD

def calcular\_tfd(coeficientes\_sfd):

#Calcula la TFD multiplicando los coeficientes de la serie de Fourier por N

N = len(coeficientes\_sfd)

return coeficientes\_sfd \* N

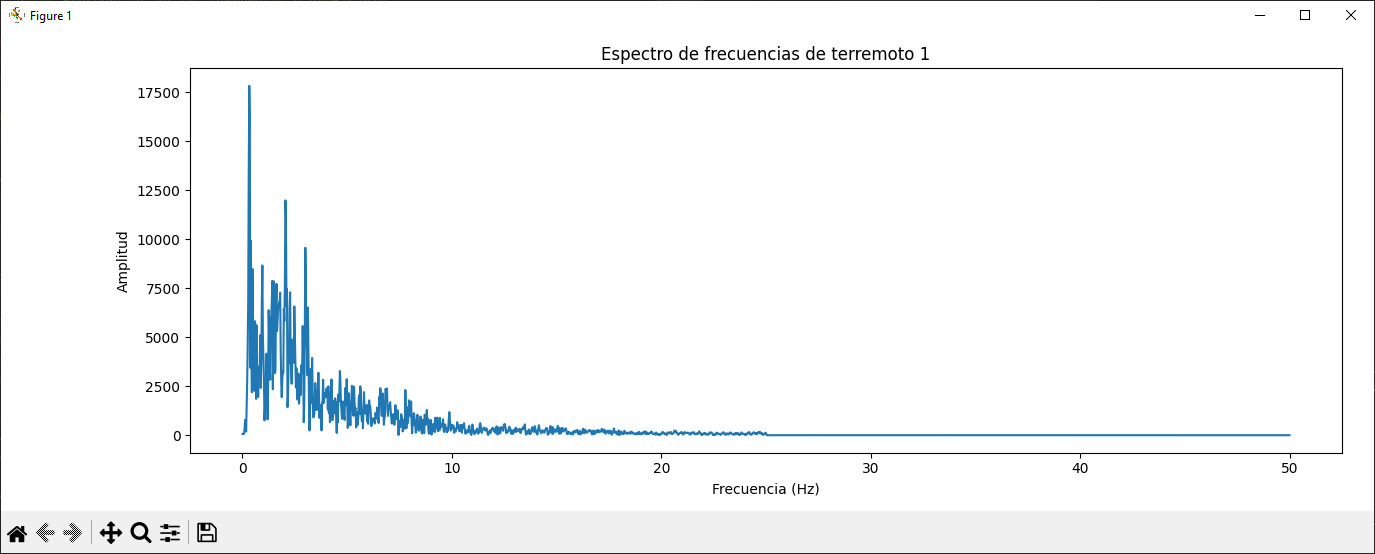


Figura 4: espectograma del terremoto1.txt

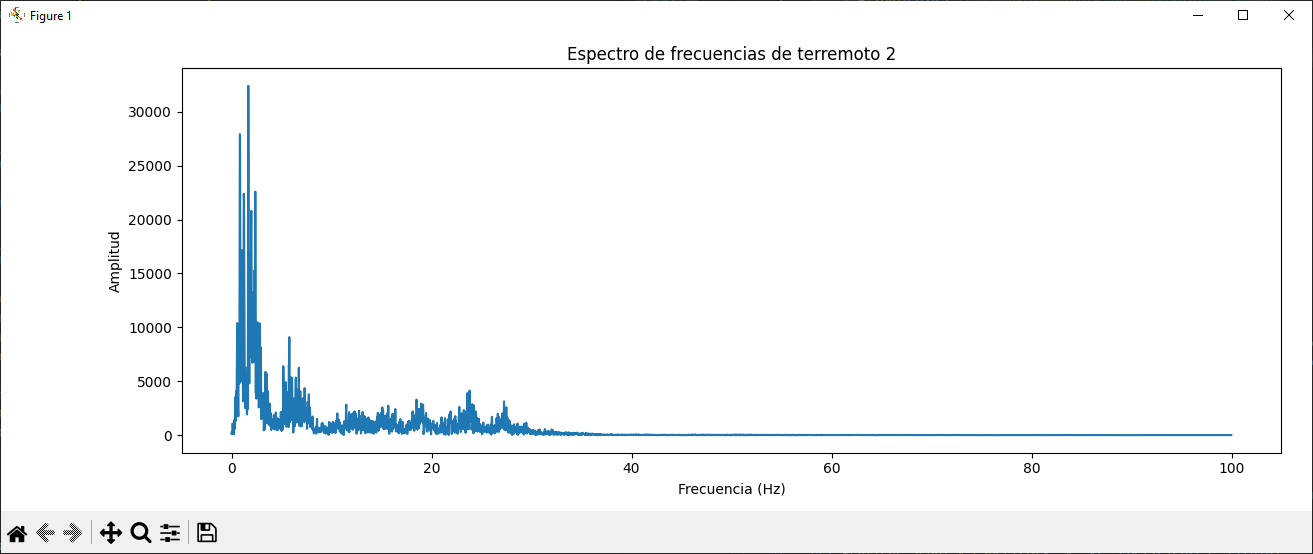


Figura 5: espectograma del terremoto2.txt

# b. Suavizado de las señales

Para suavizar las altas frecuencias de las distintas señales, utilizaremos una ventana de convolución, específicamente la ventana de Hanning.

La convolución se basa en combinar dos funciones para obtener una tercera función, la cual representa cómo una de ellas filtra o influye en la otra. El filtrado por ventana de convolución consiste en multiplicar la función original por una función de ventana (filtro). Las ventanas son de longitud finita, con una amplitud que varía suave y gradualmente hacia cero en los bordes, y da como resultado una forma de onda continua sin transiciones bruscas.

La ventana de Hanning se basa en la combinación de dos ventanas más simples: la suma de una ventana rectangular y una función coseno. Su fórmula es la siguiente: [4]

La característica más importante de la ventana de Hanning es la supresión de la información de frecuencia demasiado elevada, esto significa que reduce la amplitud de las componentes de alta frecuencia en la señal. Además esta ventana suaviza las discontinuidades en los extremos de la señal, posee un buen pico de lóbulo principal, el cual permite obtener un espectro de buena resolución, y reduce bastante las pérdidas no deseadas en este último. Por otra parte, corrige las amplitudes en el dominio de la frecuencia, proporcionando una transición suave entre las muestras de la señal.

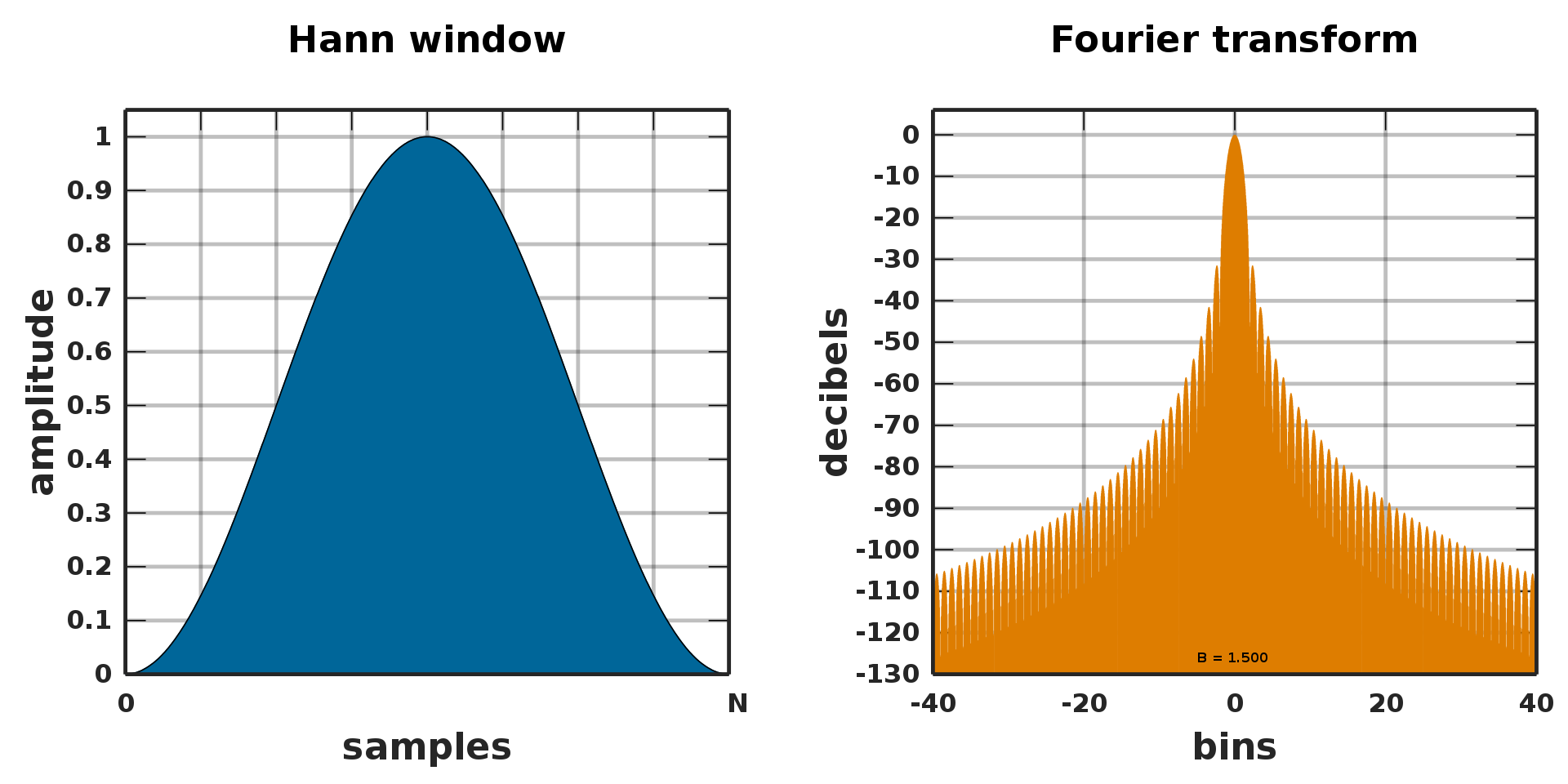


Figura 6: ventana de Hanning.

El código es el siguiente:

#Funcion que filtra frecuencias. convolucion con hanning

def filtrar(datos):

ventana = np.hanning(50) #ventana hanning

y = [p[1] for p in datos]

y\_suavizado = convolve(y, ventana, mode='same') / sum(ventana)

datos\_suavizados = [(datos[i][0], y\_suavizado[i]) for i in range(len(datos))]

return datos\_suavizados

N es el valor de la longitud de la ventana, y mientras más grande sea N, más suavizada estará la señal resultante. En este caso empleamos una longitud de 50 puntos.

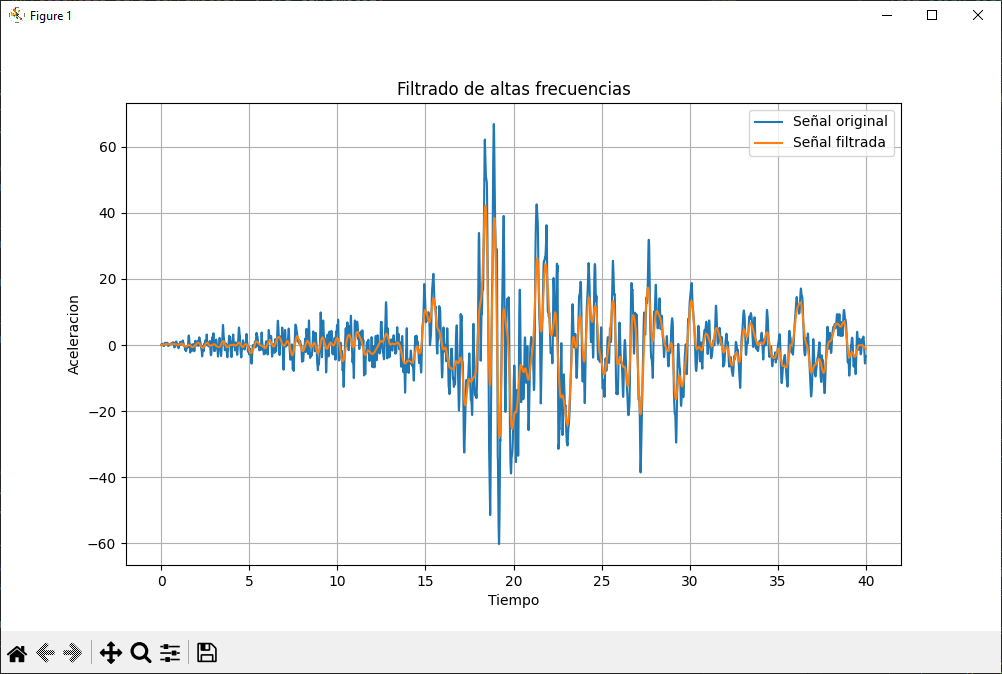


Figura 7: señal original de (azul) y filtrada (naranja) del terremoto1.

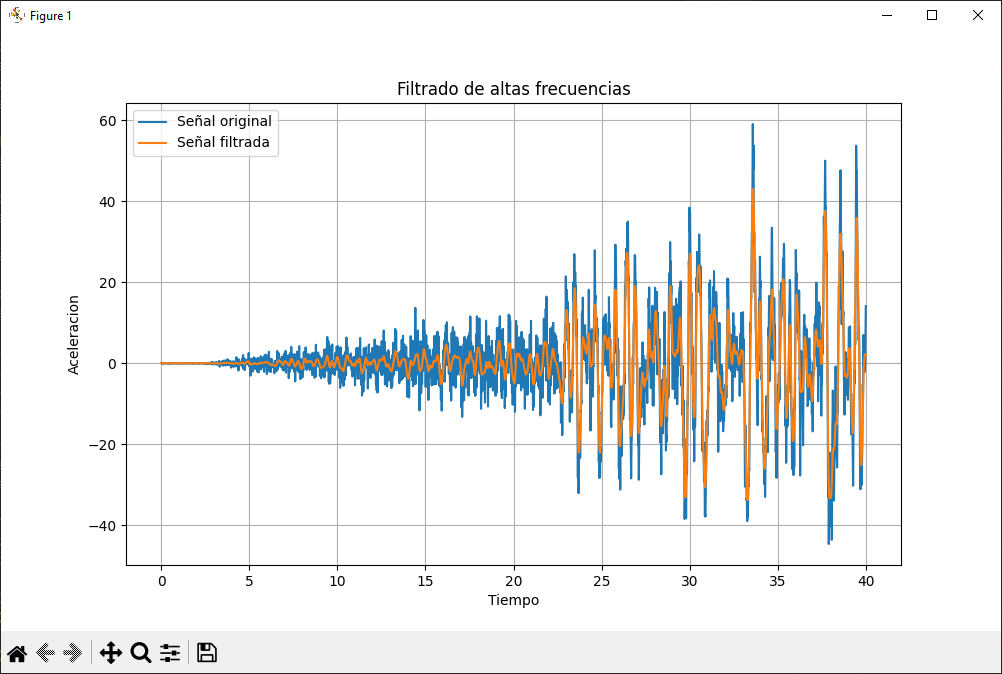


Figura 8: señal original de (azul) y filtrada (naranja) del terremoto2.

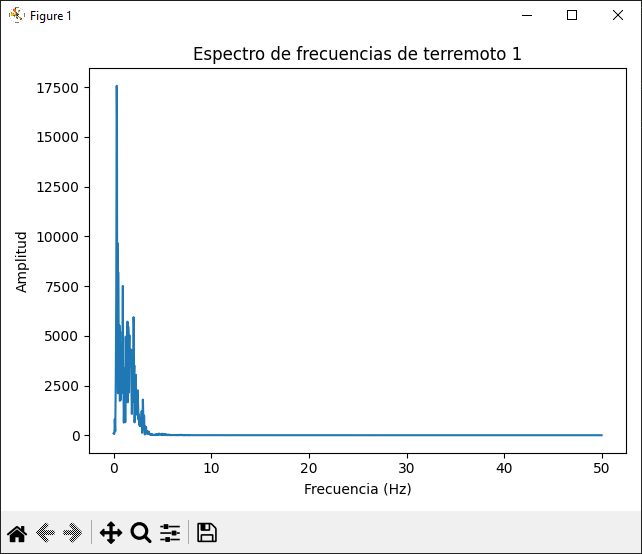


Figura 9: espectograma del terremoto1 suavizado

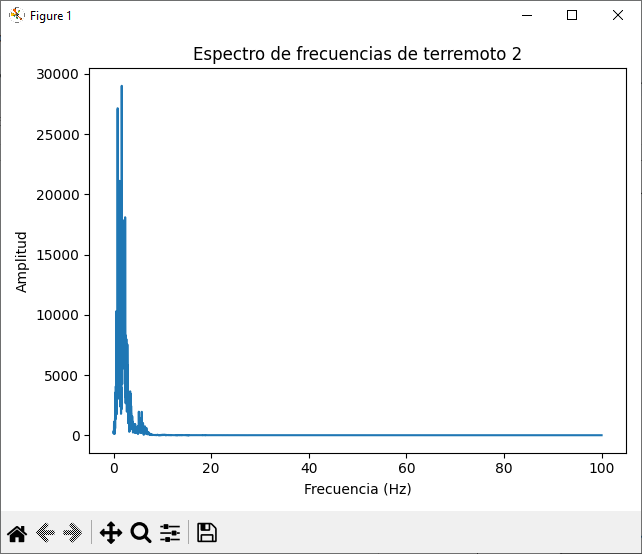


Figura 10: espectograma del terremoto2 suavizado

# c. Frecuencias más afectadas

Para determinar cuáles fueron las frecuencias más afectadas en cada una de las señales de los terremotos, es necesario trabajar con el espectrograma generado en el inciso a. Dicho espectrograma se forma a partir de las frecuencias y la amplitud (o valores de la transformada de Fourier).

Para obtener las frecuencias, realizamos el siguiente código:

#Funcion para sacar las frecuencias de la TFD

def frecuencias\_tfd(frecuencia\_muestreo, datos):

N=len(datos)

frecuencias = np.fft.fftfreq(N, 1/frecuencia\_muestreo)

return frecuencias

Hacemos uso de la función fft.fftfreq de numpy para que nos retorne el arreglo de las frecuencias en Hz. Además, le pasamos como parámetro la frecuencia de muestreo, que será el resultado de hacer , donde *t* es el intervalo de muestreo.

Para las amplitudes, utilizamos el resultado de la función calcular\_tfd del inciso a.

#Encuentra la freq maxima para una lista de coeficientes dados

def freq\_max(freqs, coef):

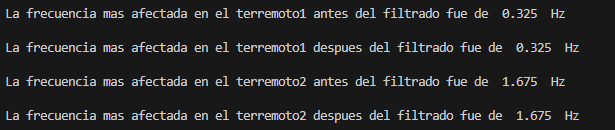
coef = abs(coef)

indice = np.argmax(coef)

return np.abs(freqs[indice])

La función recibe como parámetros el arreglo de las frecuencias y el arreglo de los coeficientes de la transformada discreta de Fourier. La estrategia aplicada es encontrar el índice del pico en el espectrograma (en cuanto a amplitud) para después encontrar el valor de frecuencia asociado a ese índice.

La función será invocada dos veces: una para los coeficientes de la transformada de la función original, y otra para los coeficientes de la transformada de la señal suavizada.



Aquí se observa cómo las frecuencias más afectadas se mantienen aunque se haya realizado el filtrado. Esto se debe a que el filtrado afecta linealmente a todos los coeficientes de la transformada, eliminando el ruido existente y filtrando las altas frecuencias.

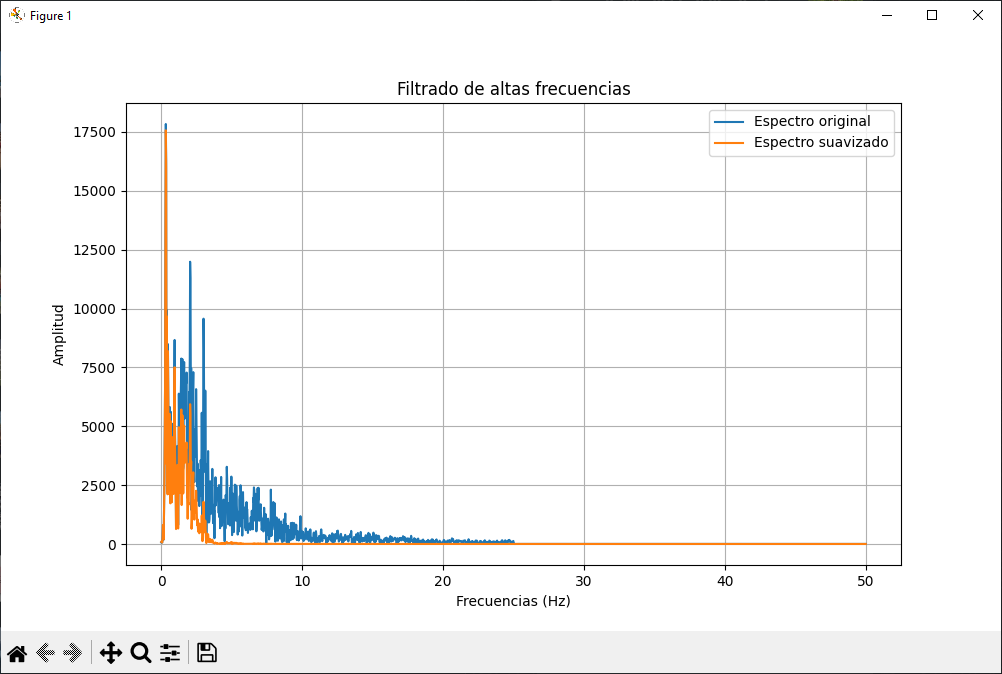


Figura 11: Espectrograma original y filtrado para terremoto 1

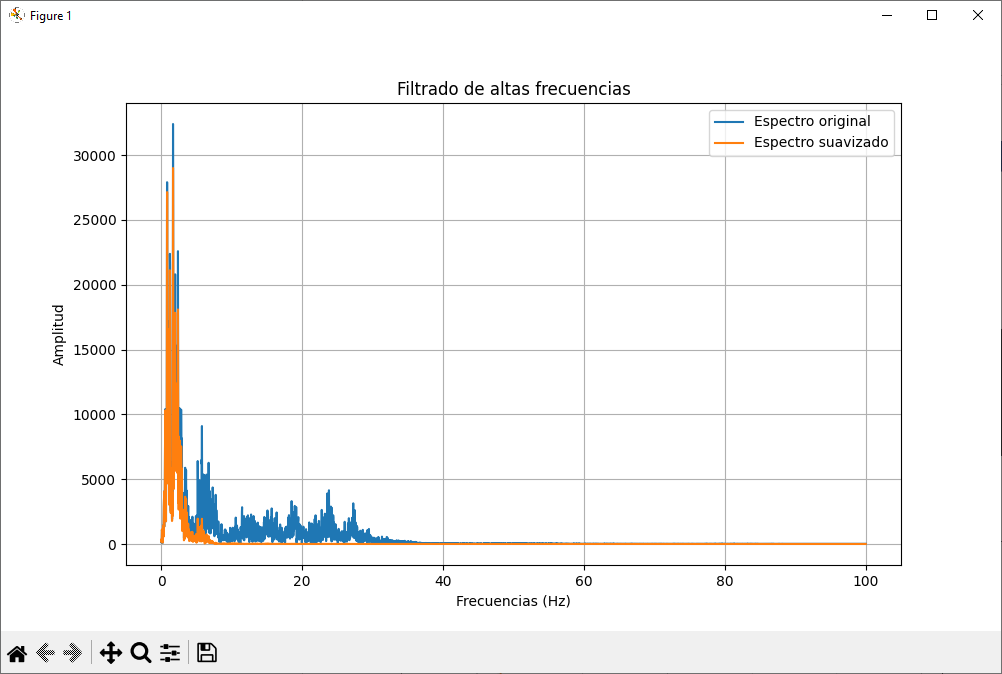


Figura 12: Espectrograma original y filtrado para terremoto 2

Se puede visualizar que las frecuencias correspondientes a los picos no cambian, aunque su amplitud disminuye levemente. Lo que sí cambia son los valores de amplitud para las altas frecuencias, que disminuyen en gran medida.

# d. Frecuencia más acelerada

Para hallar la frecuencia más acelerada de ambos terremotos, recurrimos a la suma de las señales. Cuando sumamos las transformadas de ambas señales, estamos uniendo o combinando su información para así poder llegar a conclusiones acerca de datos en común.

En este caso particular, vamos a hacer uso de una propiedad de los sistemas LTI, que es la linealidad de las transformadas de Fourier. En términos matemáticos, si tenemos dos señales *f* y *g*:

* Donde *a* y *b* son constantes y representa la transformada de Fourier.

Para obtener las dos soluciones requeridas, vamos a verificar esta propiedad resolviendo ambos lados. Primero vamos a sumar las transformadas en forma individual y, por otro lado, sumaremos las señales y transformaremos el resultado. La frecuencia más acelerada para ambas señales será aquella que corresponda al pico más alto del espectrograma. Si coinciden, demostraremos con un ejemplo particular la propiedad mencionada anteriormente.

## Primera solución: sumar las TFDs

Para la primera solución, sumaremos las transformadas de Fourier por separado y posteriormente hallaremos la frecuencia correspondiente al pico más alto del espectrograma.

El código de la solución es el siguiente:

#suma termino a termino dos tfds

def suma\_tfd(tfd1, tfd2):

return tfd1 + tfd2

#Primera solucion freq mas acelerada (SUMAS)

tfd\_terremoto1\_suavizados = calcular\_tfd(coeficientes\_sfd\_terremoto1\_suavizado)

datos\_filtrados2\_acortados= acortar\_terr2(datos\_filtrados2)

tfd\_terremoto2\_suavizados\_acortado = calcular\_tfd(calcular\_coeficientes(datos\_filtrados2\_acortados))

resultado\_suma\_tfd = suma\_tfd(tfd\_terremoto1\_suavizados, tfd\_terremoto2\_suavizados\_acortado)

graficar\_espectro\_continuo(frecuencias\_terremoto1, resultado\_suma\_tfd, "Espectro de frecuencias de suma de tfds")

maximo\_1 = freq\_max(frecuencias\_terremoto1, resultado\_suma\_tfd)

print("\nLa frecuencia mas acelerada de ambas senales es de: ", maximo\_1, " Hz en la primera solucion")



## Segunda solución: sumar las señales

Para la segunda solución, sumaremos las señales (acelerogramas) de forma individual para después calcular la transformada de Fourier. Luego, nos fijaremos en la frecuencia correspondiente al pico más alto del espectrograma.

El código es el siguiente:

#Segunda solucion freq mas acelerada (SUMAS)

suma\_senales = []

for i in range(len(datos\_filtrados1)):

suma\_senales.append(datos\_filtrados1[i][1] + datos\_filtrados2\_acortados[i][1])

tiempo\_suma = np.arange(len(suma\_senales)) / 100

datos\_suma = list(zip(tiempo\_suma, suma\_senales))

graficar\_senal\_singular(suma\_senales, tiempo\_suma)

coeficientes\_sfd\_suma = calcular\_coeficientes(datos\_suma)

tfd\_suma = calcular\_tfd(coeficientes\_sfd\_suma)

graficar\_espectro\_continuo(frecuencias\_terremoto1, tfd\_suma, "Espectro de la suma de senales")

maximo\_2 = freq\_max(frecuencias\_terremoto1, tfd\_suma)

print("\nLa frecuencia mas acelerada de ambas senales es de: ", maximo\_2, " Hz en la segunda solucion")



## Por teorema de la convolución

También quisimos probar una tercera y cuarta solución. Las mismas se basan en verificar el teorema de la convolución, el cual expresa que teniendo dos señales *f* y *g*:

Nos describe que la transformada de la convolución de dos señales es igual al producto punto de las transformadas de las señales por separado.

El código para esta solución está comentado en el archivo Python, pero por razones técnicas y desconocimiento de la herramienta, obtuvimos resultados distintos. Nuestra conclusión es que la convolución que hemos hecho fue errónea. Sin embargo, si desea ver el código está indicado al final del archivo.

#Primera solucion freq mas acelerada PRODUCTO

tfd\_terremoto1\_suavizados = calcular\_tfd(coeficientes\_sfd\_terremoto1\_suavizado)

datos\_filtrados2\_acortados= acortar\_terr2(datos\_filtrados2)

tfd\_terremoto2\_suavizados\_acortado = calcular\_tfd(calcular\_coeficientes(datos\_filtrados2\_acortados))

resultado\_prod\_punto\_tfd = prod\_punto\_tfd(tfd\_terremoto1\_suavizados, tfd\_terremoto2\_suavizados\_acortado)

graficar\_espectro\_continuo(frecuencias\_terremoto1, resultado\_prod\_punto\_tfd, "Espectro de frecuencias de prod punto de tfds")

maximo\_1 = freq\_max(frecuencias\_terremoto1, resultado\_prod\_punto\_tfd)

print("\nLa frecuencia mas acelerada de ambas senales es de: ", maximo\_1, " Hz en la primera solucion")

#Segunda solucion freq mas acelerada PRODUCTO

convolucion\_senales = np.convolve([p[1] for p in datos\_filtrados2\_acortados], [p[1] for p in datos\_filtrados1], mode='full')

convolucion\_senales = np.concatenate(([0], convolucion\_senales))

tiempo\_convolucion = np.arange(len(convolucion\_senales)) / 200

datos\_convolucion = list(zip(tiempo\_convolucion, convolucion\_senales))

graficar\_senal\_singular(convolucion\_senales, tiempo\_convolucion)

coeficientes\_sfd\_convolucion = calcular\_coeficientes(datos\_convolucion)

tfd\_convolucion = calcular\_tfd(coeficientes\_sfd\_convolucion)

graficar\_espectro\_continuo(frecuencias\_terremoto1, tfd\_convolucion, "Espectro de la convolucion de senales")

maximo\_2 = freq\_max(frecuencias\_terremoto1, tfd\_convolucion)

print("\nLa frecuencia mas acelerada de ambas senales es de: ", maximo\_2, " Hz en la segunda solucion")

# e. Ubicación de terremoto 3

Para saber qué detector estaría más próximo al detector del terremoto 3, utilizamos un algoritmo de correlación cruzada. La correlación cruzada es una medida de similitud entre dos señales, y consiste en deslizar una señal a través de otra para encontrar un número que simbolice el parecido entre ambas. Esto es conocido también como producto punto deslizante y está íntimamente relacionado con la convolución (en específico el teorema de la convolución expuesto en el punto d.)

Elegimos este método para medir la similitud entre las dos señales porque es posible que los datos del terremoto 1 o del terremoto 2 estén desplazados en el tiempo, es decir, puede que los datos hayan comenzado a medirse en un para el terremoto 1 y en un para el terremoto 2. Deslizar las señales a través de la señal 3 nos brinda una estimación más precisa acerca de su similitud, independientemente del momento en que haya empezado la medición del acelerograma.

La correlación cruzada es utilizada usualmente para buscar en señales más largas otras señales más cortas y determinar algún patrón. Está definida como:

[5]

indica el tiempo por el cual una señal se desplaza con respecto a la otra.

La relación entre la correlación cruzada y la convolución es que la correlación cruzada es esencialmente una convolución con una de las señales invertida en el tiempo.

La correlación cruzada entre y es equivalente a la convolución entre y . Es decir, podemos escribir la correlación cruzada como: .

El código es el siguiente:

#Funcion para determinar el nivel de correlacion entre dos senales

def nivel\_de\_correlacion(senal1, senal2):

aux = correlate(senal1, senal2, mode='full')

return aux

correlacion\_con1 = nivel\_de\_correlacion([p[1] for p in datos\_terremoto1], [p[1] for p in datos\_terremoto3])

correlacion\_con2 = nivel\_de\_correlacion([p[1] for p in datos\_terremoto2], [p[1] for p in datos\_terremoto3])

if(np.max(correlacion\_con1) > np.max(correlacion\_con2)):

print("El detector 3 esta mas proximo al detector 1")

else:

print("El detector 3 esta mas proximo al detector 2")

El objetivo de la función es calcular el número correspondiente al nivel de correlación. Mientras más grande sea la correlación con una señal, más próximo estará el detector a ella.

La salida del programa es:



Con este resultado, podemos concluir que el detector 3 se encuentra más próximo al detector 1, ya que su nivel de correlación es más alto.

# 

# 

# Bibliografía

1. Oppenheim, A. V., & Willsky, A. S. (1998). "Señales y sistemas".

2. ComparingSignals - MakeabilityLab.

<https://makeabilitylab.github.io/physcomp/signals/ComparingSignals/index.html>

3. Cross-correlation. En Wikipedia.

<https://en.wikipedia.org/wiki/Cross-correlation>

4. Teorema de convolución. En Wikipedia.

<https://es.wikipedia.org/wiki/Teorema_de_convoluci%C3%B3n>