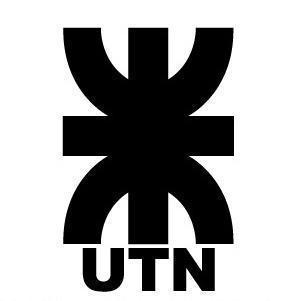
**UNIVERSIDAD TECNOLÓGICA NACIONAL**

**FACULTAD REGIONAL DE SANTA FE**

**TRABAJO PRÁCTICO N° 1**

**Señales Biológicas: ECG**

**Cátedra:** Matemática Superior

**Carrera:** Ingeniería en Sistemas de Información

**Comisión:** 3A

**Año:** 2021 - 1° Cuatrimestre

**Integrantes:** Albani, Juan - juanalbani48@gmail.com

Musso, Emilio - emilio.musso2898@gmail.com

Nuñez, Nahuel - nahuelnunezz14@gmail.com

Peiretti, Tomás - tomaspeiretti@gmail.com

Riquelme, Nahuel - nahuelriquelme00@gmail.com

**ÍNDICE**

Desarrollo…………………………………..………………………………………………………..…………1

Acondicionamiento de la horizontalidad de la señal…………………………………………..….1.1

Aproximación de la primera y segunda derivada por convolución……………………………....1.2

Primera derivada………………………………………………………………………....1.2.1

Segunda derivada………………………………………………………………...……...1.2.2

Posible modelo para generar la señal de un ECG (Respuesta a la pregunta 3)………….…...1.3

Filtrado y reconstrucción de la señal……………………………………………………..……….1.4

Separando las componentes de la señal…………………………………..…………..1.4.1

Separación de las componentes………………………………………….….1.4.1.1

Filtrado de las ondas…………………………………………………….…….1.4.1.2

Reconstrucción de la señal……………………………………………...…...1.4.1.3

Utilizando la señal completa……………………………………………………...……..1.4.2

**1 - DESARROLLO**

Para realizar las gráficas a partir del archivo .csv brindado por la cátedra, se hace uso de los paquetes pandas, matplotlib.pyplot y numpy en un script de Python, el cual se adjunta con este informe.

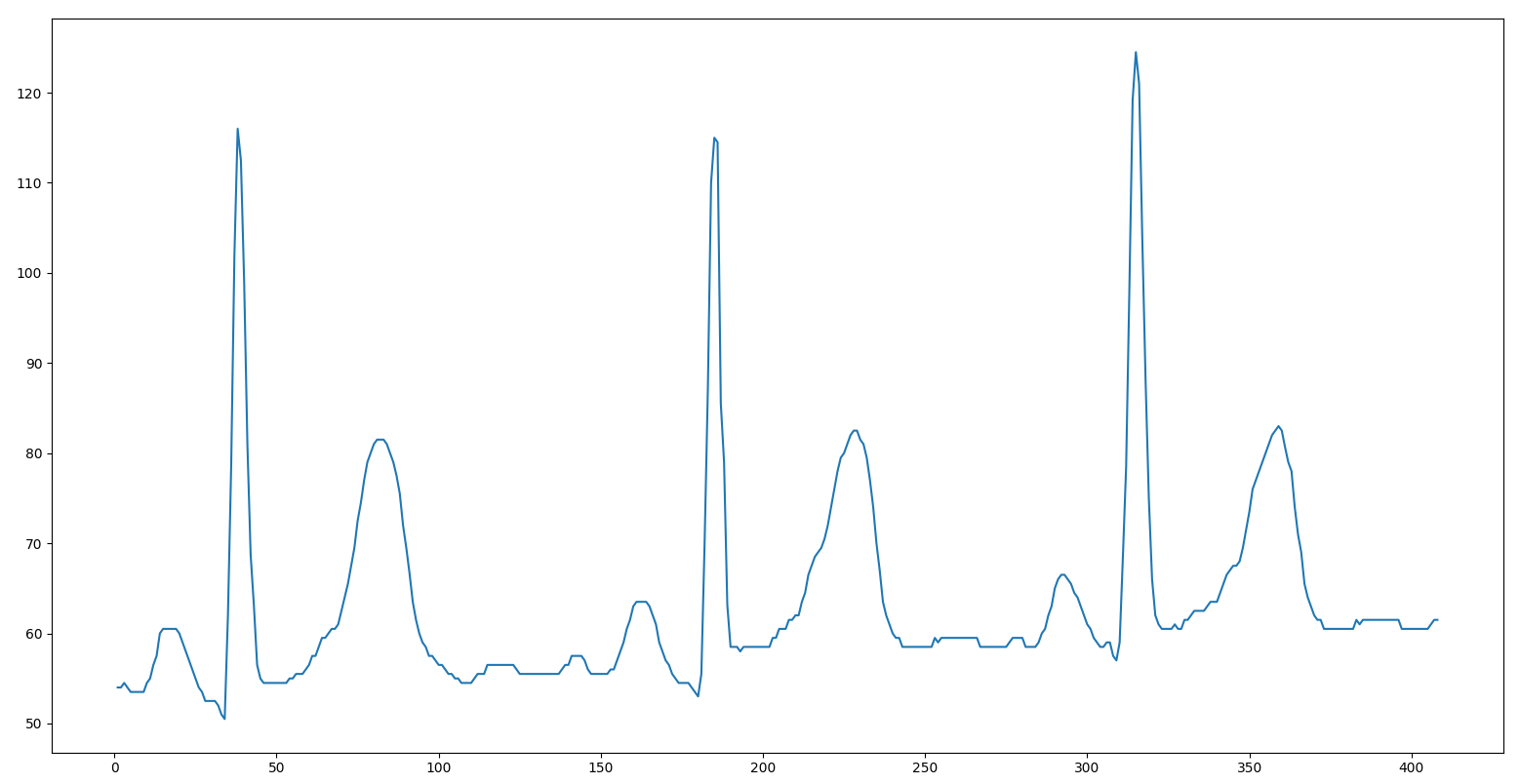


Imagen 1 - Señal del electrocardiograma dado.

**1.1 - ACONDICIONAMIENTO DE LA HORIZONTALIDAD DE LA SEÑAL**

En la imagen 1 se logra observar que la señal obtenida no se encuentra ‘horizontal’, sino que parece ir ascendiendo con cierta pendiente. Para contrarrestar esto, se tomaron dos puntos **P1**(1,54) y **P2**(408,61.5) pertenecientes a la zona de reposo en donde la señal se quiere que sea nula y se forma la recta que pasa por P1 y P2.  
Sea **f[n]** la señal del ECG y **r[n]** la recta que pasa por **P1** y **P2**, restando punto a punto **f[n]**-**r[n]**, se logra obtener la señal horizontal y con los valores correspondientes al reposo nulos (o aproximadamente nulos).

def hacerHorizontal(x,fx):

recta\_y = []

ejex = []

for i in range (1 , 409):

recta\_y.append((i-1)\*((fx[0]-fx[-1])/(x[0]-x[-1]))+fx[0])

for i in range (0,fx.\_\_len\_\_()):

fx[i] -= recta\_y[i]

ejex.append(0)

return fx

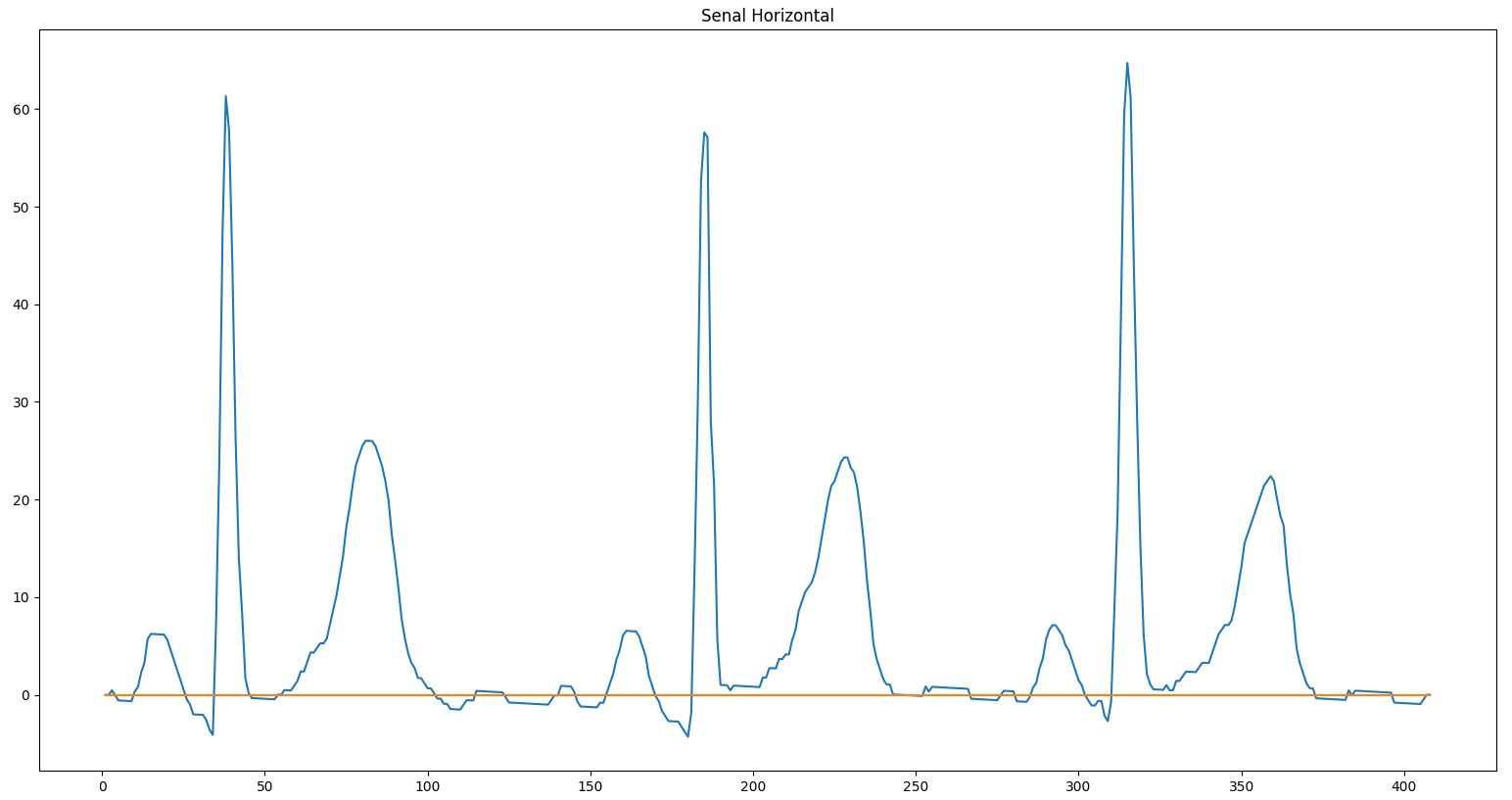


Imagen 2 - Señal horizontal.

**1.2 - APROXIMACIÓN DE LA PRIMERA Y SEGUNDA DERIVADA POR CONVOLUCIÓN**

**1.2.1 - PRIMERA DERIVADA**

La derivada de una función en el campo continuo se define de la siguiente manera:

Si a la función se la muestrea con un paso **h**, podemos realizar dos aproximaciones haciendo uso de la primer diferencia:

* Aproximación por izquierda:
* Aproximación por derecha:

Pero, para obtener una mejor aproximación a la primer derivada, se pueden promediar ambas aproximaciones y obtener la siguiente fórmula:

Para poder realizar la aproximación por convolución, se aplica la siguiente propiedad de la función **Impulso Unitario Discreto***:* *La convolución entre una función y un impulso desplazado k unidades, es equivalente a tener la función desplazada k unidades.*

Luego, se puede plantear lo siguiente:

Es decir, se puede aproximar la primer derivada de la señal del electrocardiograma f[n] realizando la convolución entre el vector que representa **f[n]** y el vector [1, 0, -1], y luego a la señal resultante dividirla por **2h**, donde **h** es el paso con el que se realizó el muestreo.

Como la columna en el archivo .csv dado, no posee indicada la unidad, se considera que es el número de muestra. Y, como la señal dada posee 408 muestras, las cuales fueron tomadas en un período de 180 segundos, se puede calcular el paso **h** como:

**h** = 180/408 = 0.441176

def primeraDerivada(f,paso):

'''

convolución con d[n+1]-d[n-1] (aproximación de la primera derivada)

'''

convolucion = np.convolve(f,[1,0,-1])

for x in convolucion:

x = x/(2\*paso)

return convolucion

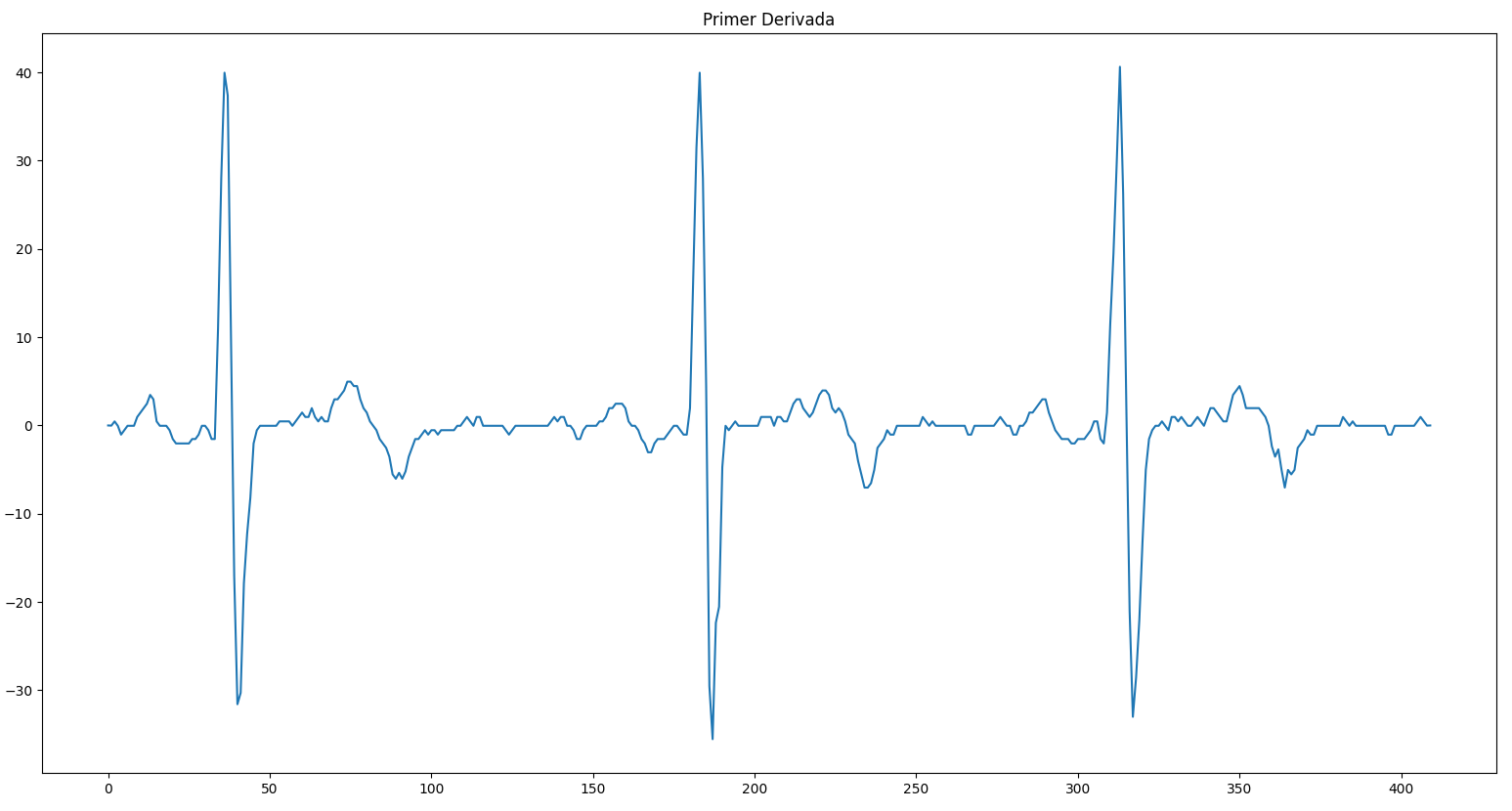
****

Imagen 3 - Primera derivada de la señal.

**1.2.2 - SEGUNDA DERIVADA**

Si se plantea la misma idea que en el punto anterior, aplicando la primera diferencia se puede aproximar la segunda derivada de la señal de la siguiente forma:

Donde, se puede escribir en función de una convolución como:

Es decir, se puede aproximar la segunda derivada de la señal del ECG realizando la convolución entre el vector que representa la señaly el vector [1, -2, 1], y luego a la señal resultante dividirla por **h^2**, donde **h** es el paso con el que se realizó el muestreo (180/408).

def segundaDerivada(f,paso):

'''

convolución con d[n+1]-2d[n]+d[n-1] (aproximación de la segunda derivada)

'''

convolucion = np.convolve(f,[1,-2,1])

for x in convolucion:

x = x/(paso\*paso)

return convolucion

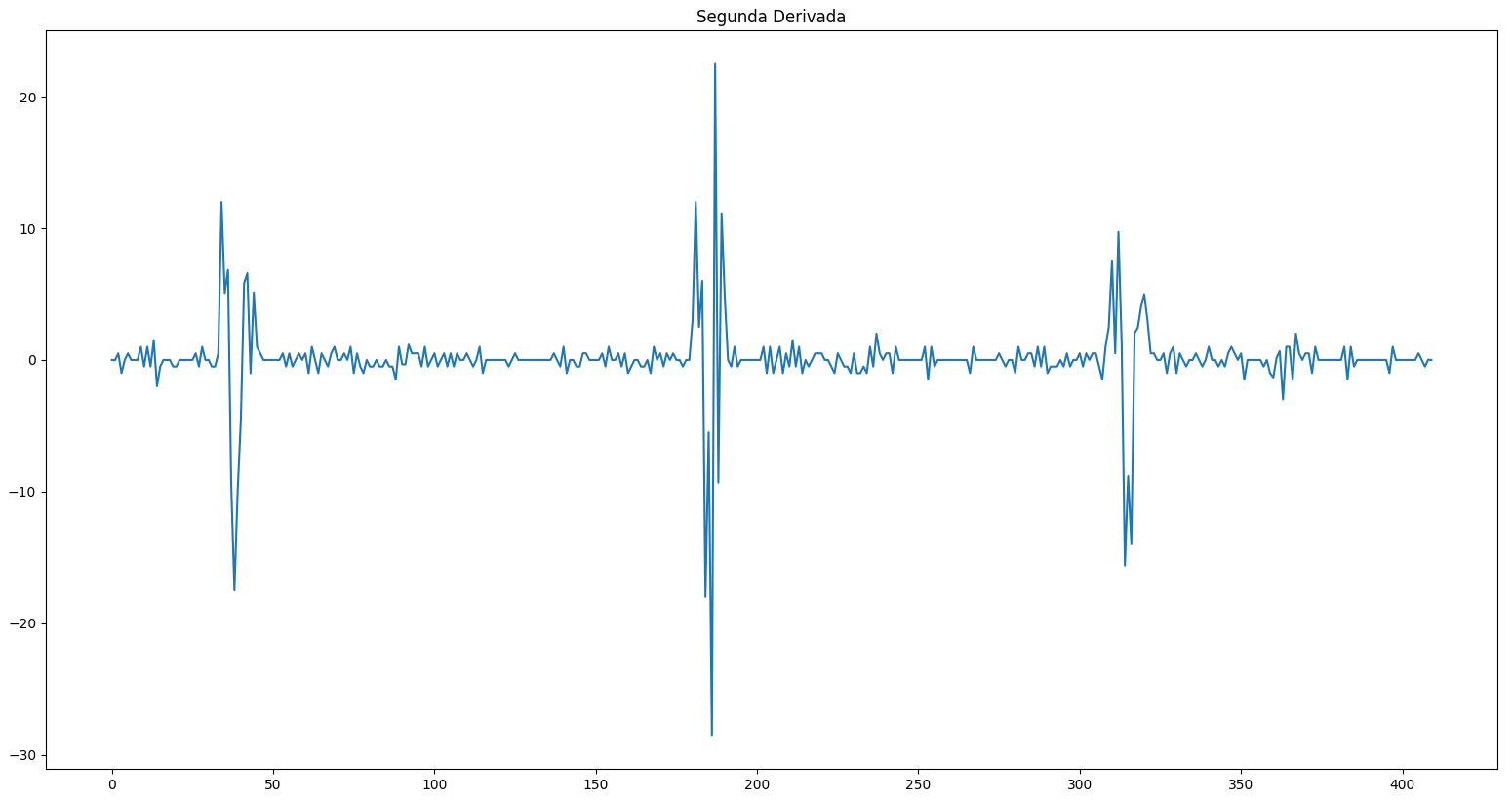


Imagen 4 - Segunda derivada de la señal.

**1.3 - POSIBLE MODELO PARA GENERAR LA SEÑAL DE UN ECG (Respuesta a la pregunta 3)**

Habiendo obtenido las derivadas de la señal ECG, se terminó optando por la primera derivada para trabajarla, ya que brinda una visión más clara de dónde se encuentran los picos de la señal.

Se sabe que los puntos críticos de una función se encuentran donde la primera derivada es nula (o no existe). Luego, esto es aplicable para determinar los picos (máximos y mínimos) de cada onda. Además, en la primera derivada del ECG, es posible saber la ubicación de la onda QRS observando la amplitud entre los picos que rodean a los puntos críticos (ya que la amplitud será muy grande). Luego, sabemos que para cada latido del corazón, la onda QRS se encuentra entre la onda P y la T. Entonces, si se desplaza un poco hacia la izquierda, se puede encontrar fácilmente la onda P, y desplazándose un poco hacia la derecha, la onda T.

Para modelar un electrocardiograma, se podría tomar 3 senos diferentes (con diferentes frecuencias, amplitudes, fase y desplazamientos) y acotarlos en el eje horizontal utilizando lomos de burros, suponga s1[x], s2[x] y s3[x]. Luego, sumando estos tres senos se podría tener una aproximación a un pulso cardíaco, como se ve en las siguiente imagen:

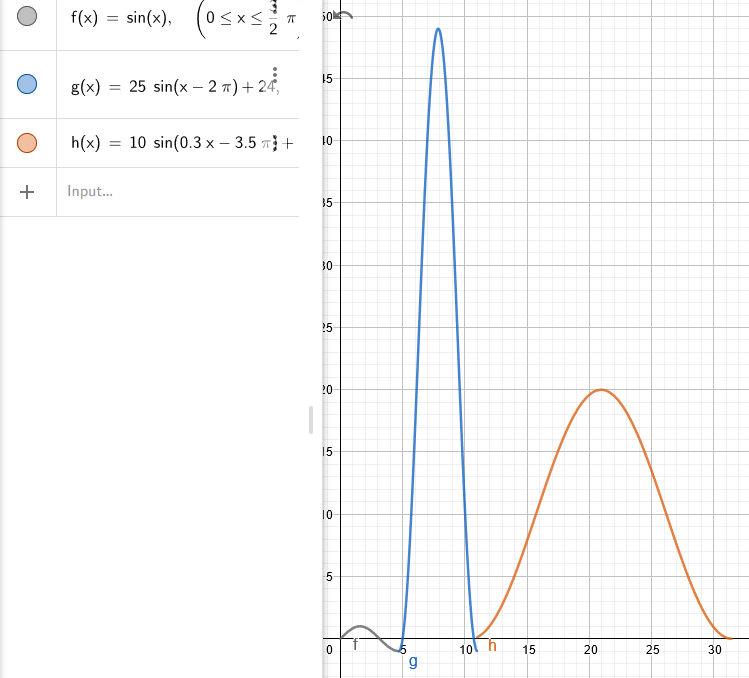


Imagen 5 - Ejemplo de cómo resultaría la suma de 3 senos diferentes.

Por último, habría que realizar la extensión periódica de dicha función, tomando un período un poco mayor al de los tres senos sumados, para considerar la zona de reposo del electrocardiograma. En el caso que se quisiera modelar el posible ruido generado por el equipo que realiza el ECG, se podría tomar la función resultante y sumar un valor aleatorio que vaya variando entre pequeños valores (-0.01 a 0.01 por ejemplo).

Aparte de lo planteado, existe una herramienta en python que permite simular un electrocardiograma que aplica un método muy similar al planteado, solo que a diferencia de sumar 3 senos diferentes, hace uso de la ***“Daubechies" wavelet***.

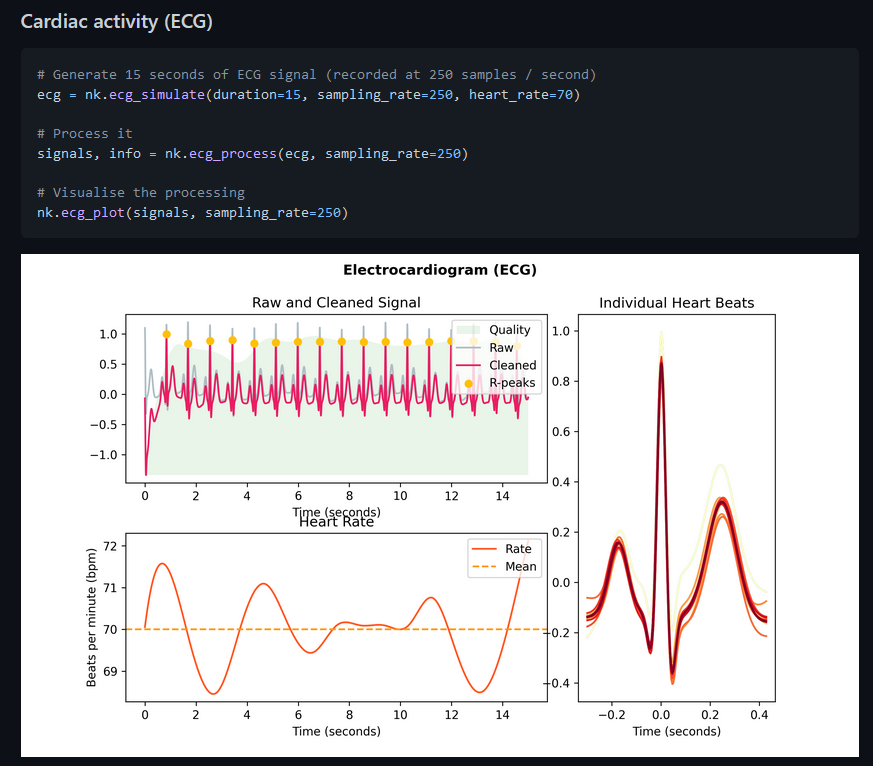


Imagen 6 - <https://github.com/neuropsychology/NeuroKit#citation>

**1.4 - FILTRADO Y RECONSTRUCCIÓN DE LA SEÑAL**

**1.4.1 - SEPARANDO LAS COMPONENTES DE LA SEÑAL**

**1.4.1.1 - SEPARACIÓN DE LAS COMPONENTES**

Para la separación de las componentes, se aplica lo explicado en el punto 1.3. El algoritmo implementado busca los puntos críticos de la señal. Luego, siendo **x** un punto crítico, se evalúa **x** respecto de los puntos vecinos en la derivada y se analiza la diferencia entre ambos. Si la diferencia en valor absoluto es mayor a 20 (valor que se obtuvo de observar la derivada gráficamente) entonces la coordenada x de **x** se considera como la posible posición donde se encuentra el pico de la componente QRS.

Una vez obtenidas todas las abscisas posibles, aquellas que tengan una distancia de 10 unidades o menor, sólo se guarda aquel punto que su imagen se encuentre más cercano al cero en la primera derivada. Esta distancia de 10 unidades fue determinada gráficamente observando todos los puntos críticos, dado que los puntos críticos que se encuentren a una distancia mayor a ésta, no pertenecerán al mismo período (sino a otros).

Finalmente, una vez obtenidas las ubicaciones de los picos de la QRS (supongamos **X**), se define la QRS con todos los puntos que pertenezcan al intervalo [**X** - 10, **X** + 10]. Luego, para las ondas **P** y **T** se toman los intervalos (**X** - 40, **X** - 10) y (**X** + 10, **X** + 70] respectivamente. Todos aquellos puntos que no pertenezcan al intervalo correspondiente, se colocan nulos.

El ancho de cada intervalo fue definido a partir de la observación de la señal gráficamente, tomando aproximadamente el promedio de los tres períodos para cada una de las ondas.

El código de este algoritmo es extenso, pero se encuentra en el archivo .py adjuntado en la función **hallarOndas(f, paso)**.

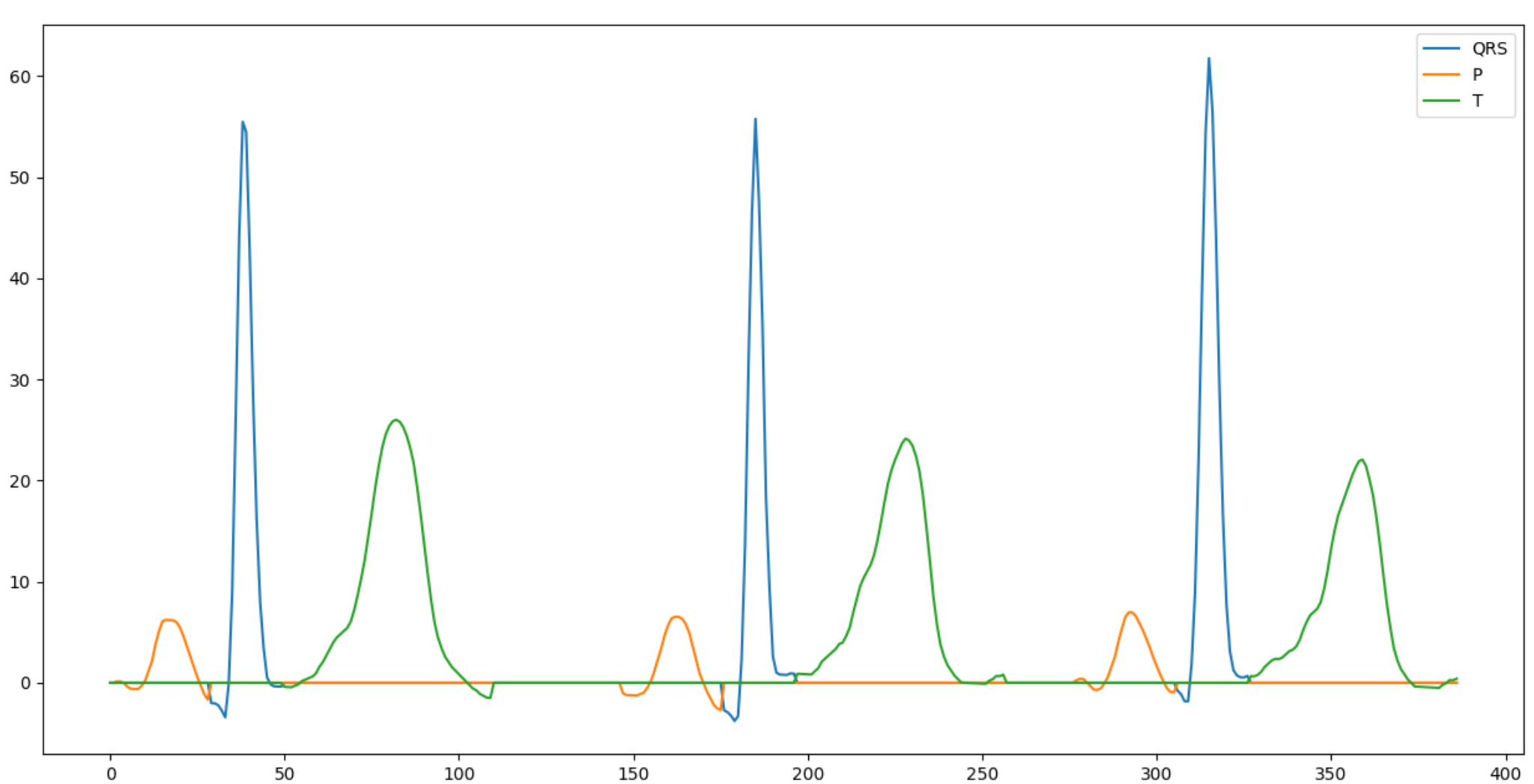
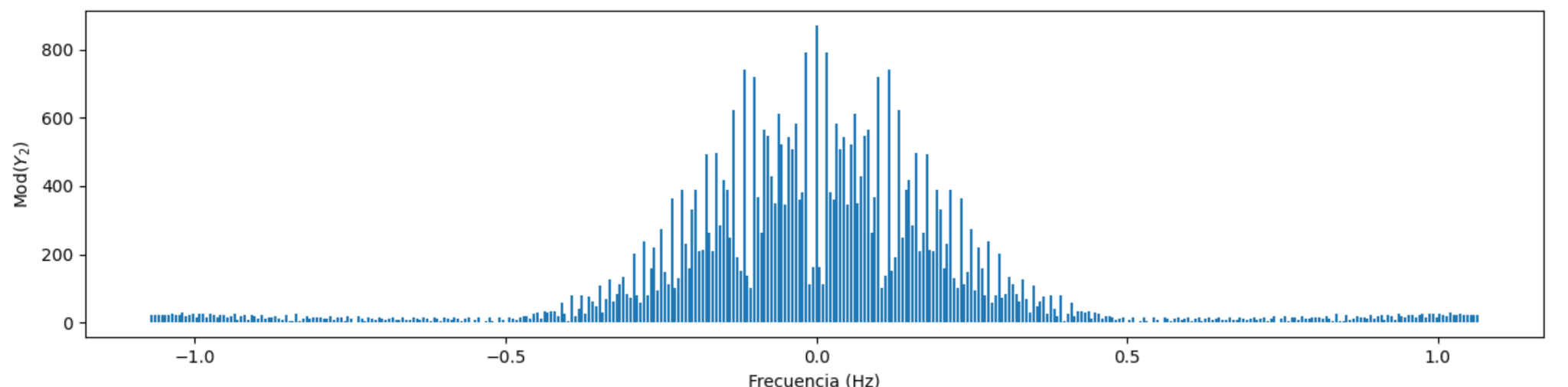
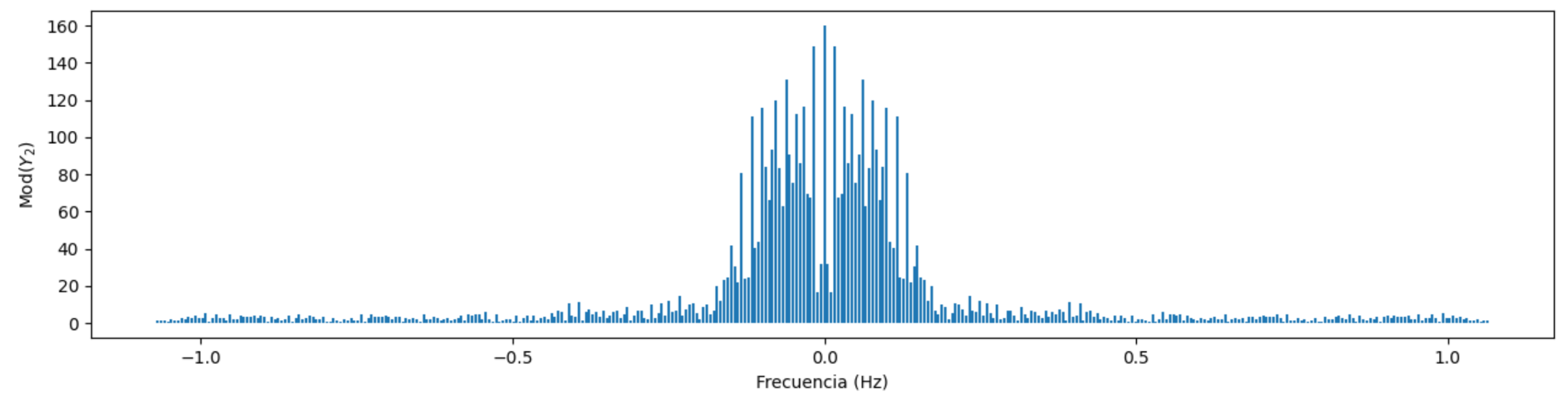


Imagen 7 - Ondas separadas en un mismo gráfico.

**1.4.1.2 - FILTRADO DE LAS ONDAS**

El filtrado de cada onda se realiza a partir de la observación de los espectros de frecuencia de las ondas de las transformadas de fourier de las ondas obtenidas en el punto anterior. En las siguientes imágenes, se muestran los espectros de frecuencia de las ondas QRS, P y T respectivamente.





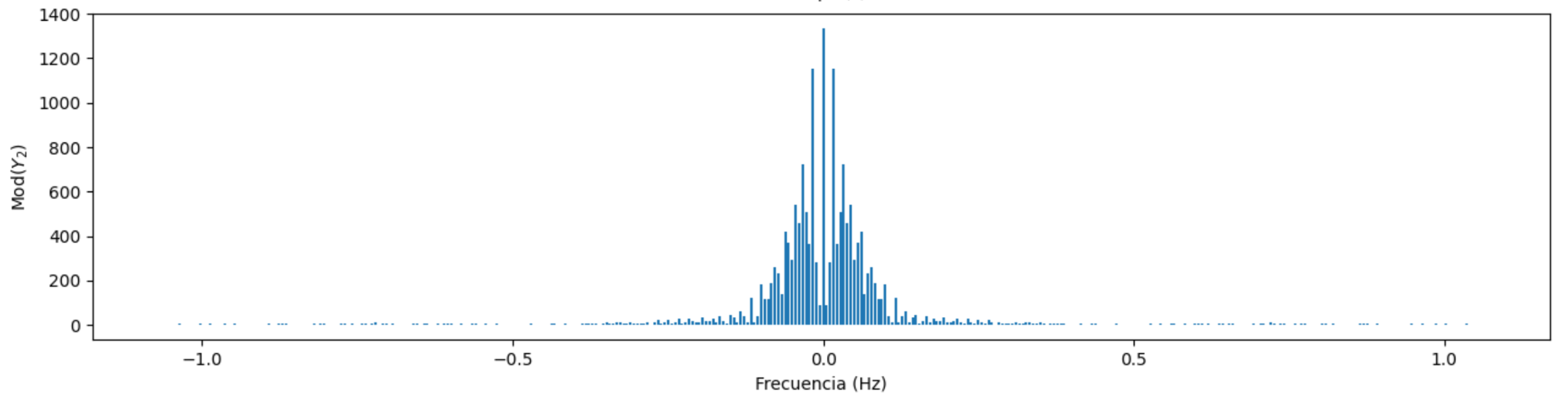


Imagen 8 - Espectro de frecuencias de las ondas QRS (primera), P (segunda) y T (tercera).

El objetivo de filtrado fue conservar las componentes de la transformada que generan una onda que es representativa de la original. Para esto se tomó el criterio de eliminar aquellas componentes cuyo módulo fuese menor a un cierto porcentaje de la armónica con mayor módulo. Se hace uso de este método para poder mantener ciertas componentes que serían filtradas si se haría uso de algún filtro (pasa alto o pasa bajo por ejemplo).

Los porcentajes de corte fueron seleccionados a partir de la observación de la antitransformada filtrada, asegurando que se asemeje a la señal original:

* **QRS:** Porcentaje de corte del 12%.
* **P:** Porcentaje de corte del 40%.
* **T:** Porcentaje de corte del 12%.

def filtrarOnda(f, porcentaje):

F = sc.fft(f)

im = []

for i in F:

im.append(np.sqrt(i.real\*\*2+i.imag\*\*2))

maximo\_modulo = max(im)

for i in range (0, im.\_\_len\_\_()):

if(im[i] < porcentaje\*maximo\_modulo): F[i] = 0

return sc.ifft(F)

**1.4.1.3 - RECONSTRUCCION DE LA SEÑAL**

Para la reconstrucción de la señal, se sumaron las tres ondas filtradas, obteniendo el siguiente resultado:

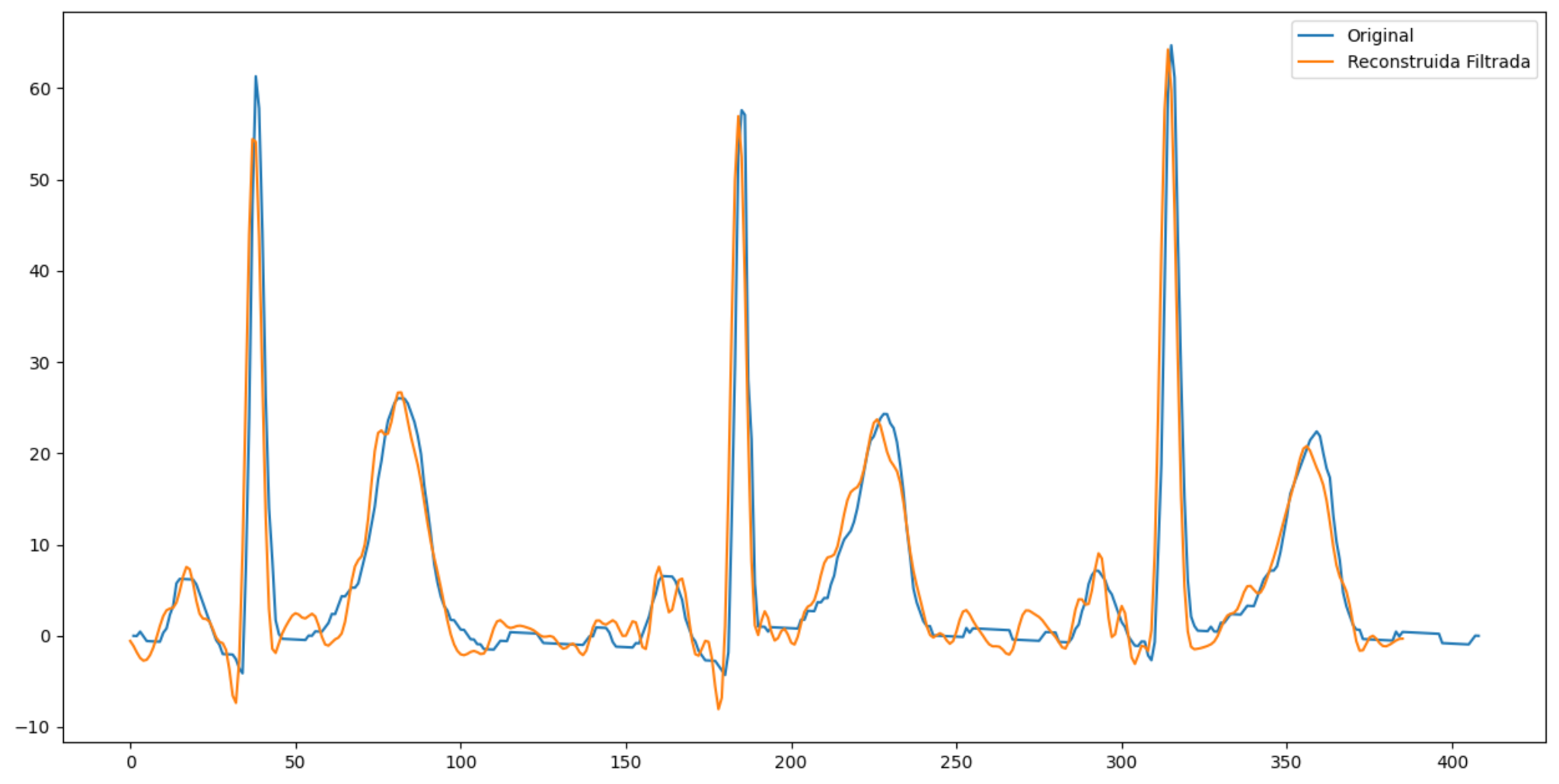


Imagen 9 - Comparación de la señal original con la señal filtrada.

Como se logra observar, la señal resultante es muy similar a la original, con la diferencia que la señal filtrada posee menos frecuencias que la original, es decir, necesita menos componentes en la transformada de fourier para ser representada.

**1.4.2 - UTILIZANDO LA SEÑAL COMPLETA**

El espectro de frecuencias de la señal completa del ECG se muestra en la siguiente imagen:

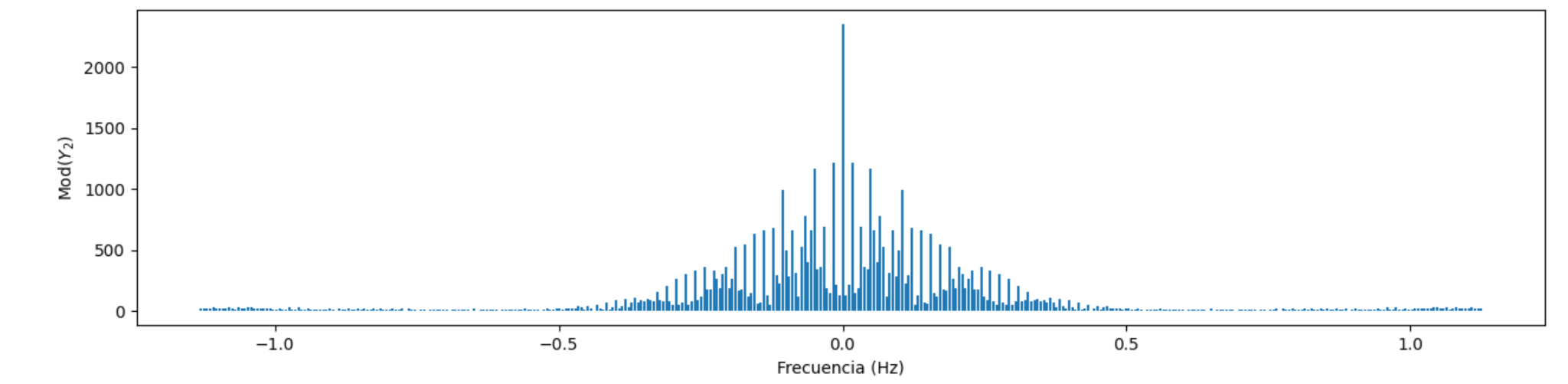


Imagen 10 - Espectro de frecuencias de la señal original.

Para el filtrado de esta señal se aplica el mismo criterio que en el punto anterior. El porcentaje de corte en este caso debe ser muy pequeño para poder obtener una señal que se acerque a la original. Esto se debe a que la componente de mayor módulo, es mucho mayor que el módulo del resto de las componentes.

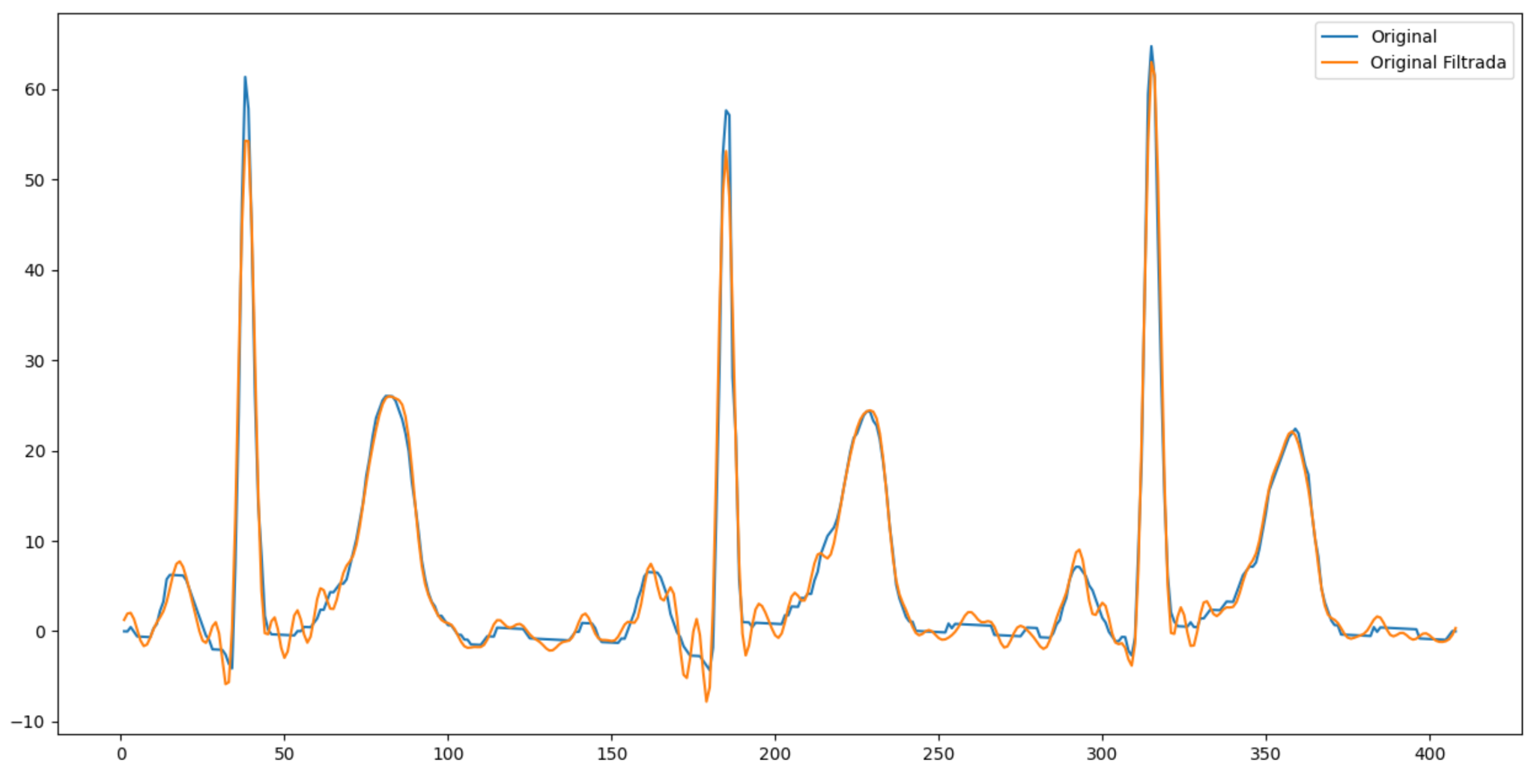


Imagen 11 - Comparación de la señal original con la señal filtrada.

Comparando ambos resultados (la señal entera filtrada y la suma de ondas filtradas) se puede observar que es posible obtener resultados similares aplicando ambos procedimientos, pero que la señal obtenida a partir de separar cada onda, filtrarlas por separado y juntarlas nuevamente (Reconstruida filtrada) requiere de una mayor cantidad de armónicas (para los porcentajes seleccionados).

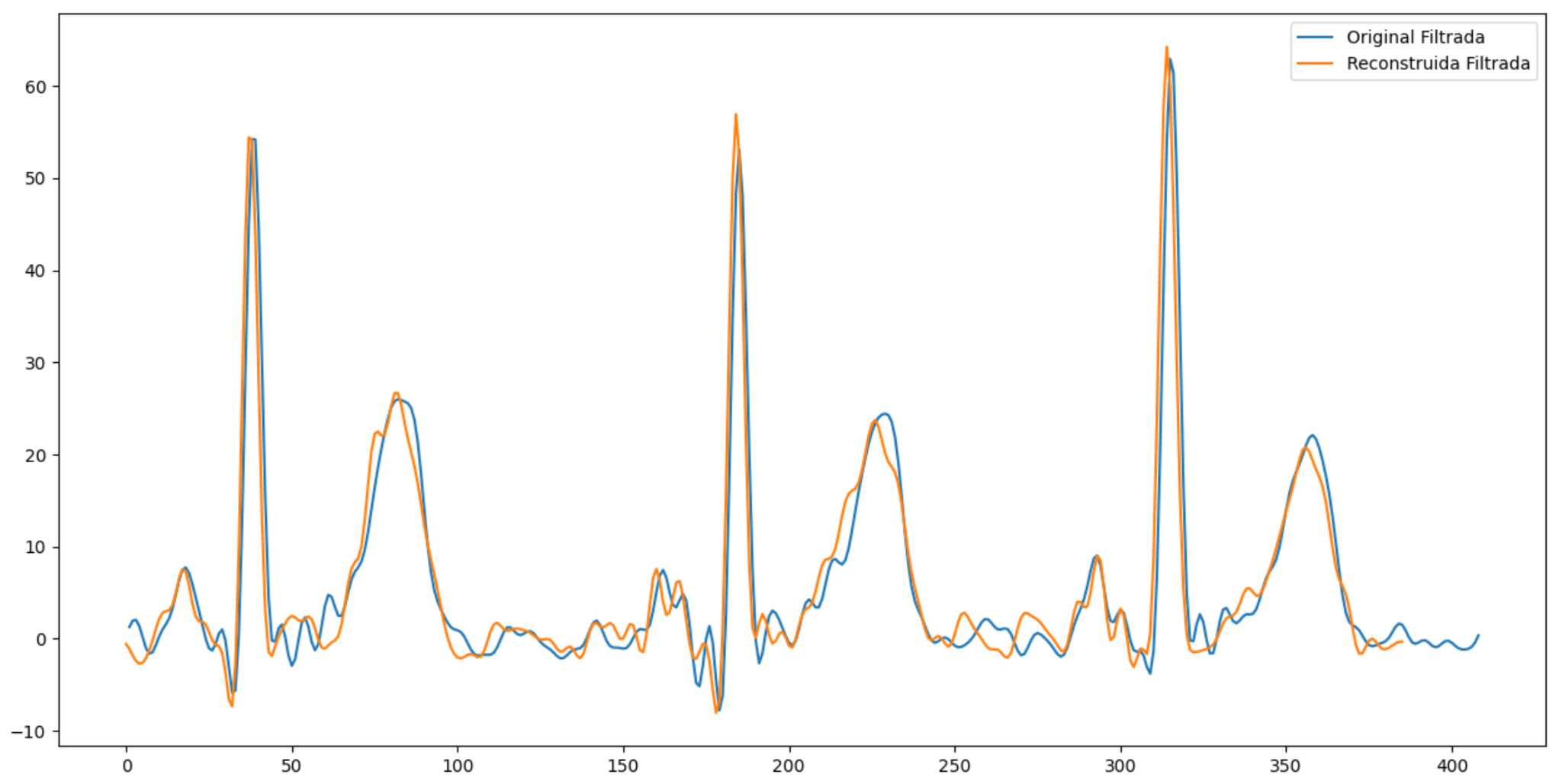
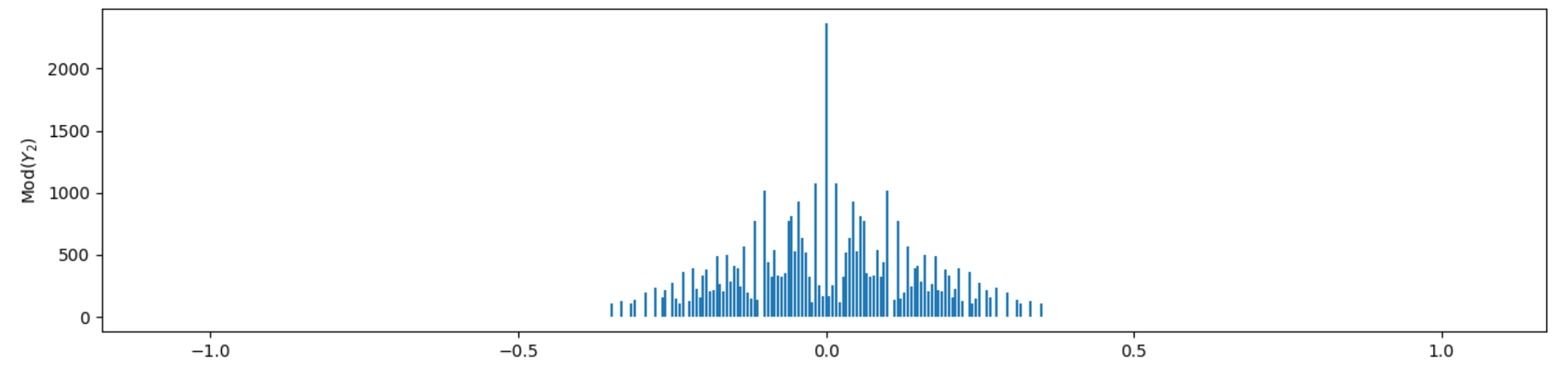


Imagen 12 - Señales Original filtrada y Reconstruida filtrada.



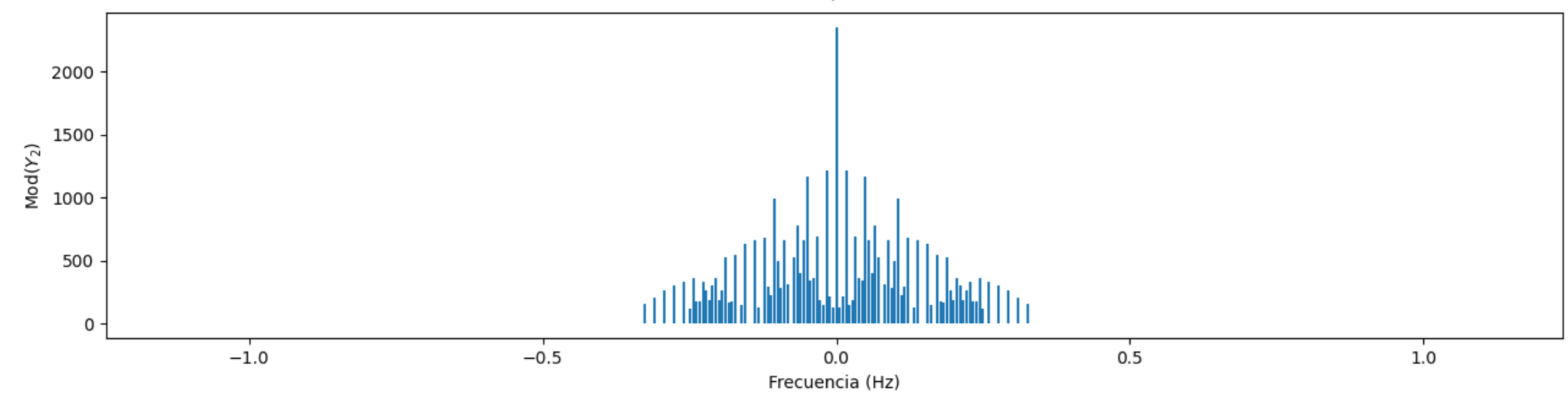


Imagen 13 - Espectro de frecuencias de las señales Reconstruida filtrada (arriba) y Original filtrada (abajo).