Laboratorio 1

Representación en Frecuencia

Joaquín Firpo, Bruno Moreira

Resumen - En este laboratorio se analiza el comportamiento de la serie de Fourier mediante simulaciones computacionales, comparando sus resultados con la teoría. Se exploran los aspectos fundamentales de esta herramienta, sus ventajas y limitaciones, presentando los resultados de manera gráfica y analítica.

Sección 1

I. INTRODUCCIÓN

En sección se desarrolló un código en Python para hallar las sumatorias correspondientes a la de la serie de Fourier [1] para poder investigar si es posible realizar una representación computacional de la serie de Fourier que se aproxime a la señal original.

II. DESARROLLO

Problema 1

Para calcular los coeficientes de la serie de Fourier, primero es necesario ser capaz de encontrar series de intervalo infinito. Para ello se creó un código que sea capaz de calcular la convergencia de una sumatoria determinada como:

$$c = \sum_{k=-N}^{N} b_k$$

Se utilizó una estructura 'for', la cual se encarga de sumar los términos de b_k en el intervalo simétrico seleccionado N. Para modificar el intervalo se utilizaron estructuras condicionales 'if'.

Para verificar su funcionamiento se utilizaron tres casos de prueba:

$$C_1 = \sum_{k=0}^{N} \frac{1}{2^k}$$

$$C_2 = \sum_{k=1}^{-N} \left(\frac{1}{5}\right)^{-k}$$

$$C_3 = \sum_{k=1}^{N} \frac{(-1)^k}{|k|}$$

Como nuestro intervalo N no puede ser infinito, ya que esto nos llevaría mucho tiempo, se optó por tomar valores muy grandes de N y compararlos con la convergencia real de la serie infinita. Para lograr representar de la serie adecuada se utilizó la librería mpmath la cual nos permite trabajar con más cifras decimales. Para determinar cuántas cifras, se contempló el caso que más rápido tiende su valor limite, en este caso C_1 y cuando N es \

mayor. Teniendo en cuenta la ecuación 1 se necesitarán aproximadamente 3010 cifras .

$$Nlog10(2) \approx 0.30103N = 3010 \text{ cifras}$$

Ecuación 1: Cifras necesitas para alcanzar cierta precisión decimal.

N	10	100	1000	10000	Valor límite
C_1	-9.77e-4	-7.89e-31	-9.33e-302	-5.01e-3011	2
C_2	1.73e-17	1.73e-17	1.73e-17	1.73e-17	0.25
C_3	1.291	1.376	1.385	1.386	Diverge

Tabla 1: Verificación de convergencia de la serie implementada en 'Problema 1'.

Para calcular el valor límite de la sumatoria se tuvieron en cuenta las reglas de convergencia de las series infinitas, y se evaluó su diferencia con respecto a el valor obtenido por la sumatoria discreta.

Observando la (**Tabla 1**) podemos deducir que si se toma un valor de N lo suficientemente grande el programa implementado se aproxima al valor de convergencia real de la serie, sin embargo, para el caso C_2 esto no se observa. Para comprobar si efectivamente converge a lo esperado se calculan diferentes ΔN mediante otra estructura 'for' y una lista la cual almacena los distintos valores los cauales serán comparados.

	ΔN(10,100)	ΔN(100,1000)	ΔN(1000,10000)
C_2	2.56e-8	3.169e-71	1.386e-700

Tabla 2: Verificación de convergencia de la serie C_2.

Con esta reinterpretación de los valores obtenidos podemos concluir que C_2 también converge a lo esperado.

Problema 2

Utilizando una versión simplificada del código anterior, ya estamos en condiciones de calcular la suma parcial de una serie de Fourier.

```python
import cmath
import numpy as np
import matplotlib.pyplot as plt
from math import pi, sin

def suma_parcial_fourier(t, f0, N):
    suma_total = 2/3
    for k in range(-N, N + 1):
        if k % 3 == 0:
            a_k = 0
        elif k != 0:
            a_k =(2*(cmath.exp(-k*(1/3)*pi*1j))*(sin(k*(1/3)*pi)))/(k*pi)
        suma_total += a_k * cmath.exp(2*pi*k*f0*t*1j)
    return suma_total

def señal_original(t):
    t_mod = t % 3  # Tomar la parte fraccionaria para el período
    if 0 <= t_mod < 1:
        return 2
    else:
        return 0
```

Código 2:Función que calcula la serie de Fourier

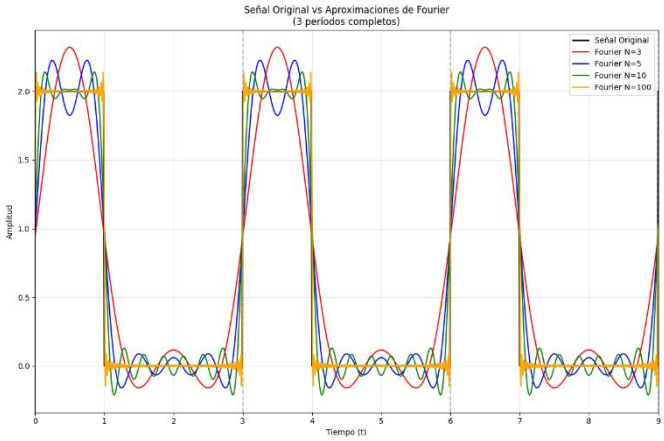
Se creo una función con la estructura anterior y con las variables de entrada, $f_0, N\ y\ t$.
Para calcular serie de Fourier se definió los términos a_k como se muestra en la línea 12 del código 2, teniendo en cuanta (Ejercicio 6 del repartido 4). A su vez se implementaron las librerías presentes en las líneas 1 a la 4 para facilitar la operación con complejos y la implementación de graficas.
Para verificar que el programa funciona correctamente se prueban 5 valores de t diferentes variando el valor de N para aproximarnos al valor real para ello se utilizó otra estructura 'for' para los valores 10, 100, 1000 y 10000 en conjunto con la función presente en la línea 6.

t(s)	0.5	1	1.5	2	2.5
10	2.010	1.034	0.005	0.069	0.005
100	~2.000	1.004	5.455e-5	0.007	5.455e-5
1000	~2.000	~1.000	5.508e-7	0.001	5.508e-7
10000	~2.000	~1.000	5.513e-9	7.351e-5	5.513-9

Tabla 3: Valores de la serie de Fourier para N y t distintos.

Si se tiene en cuenta la señal cuadrada original x(t) del Ejercicio 6 se percibe que nuestra aproximación mediante la serie de Fourier concuerda con lo esperado.
Para observar esto de una forma más visual se optó por graficar las señales producidas por la suma parcial de Fourier para distintos valores de N, 3, 5, 10 y 100.

Para ello se definen algunos parámetros como el tiempo de inicio y final de la representación y los puntos a ser graficados, para definir t con un espaciado predeterminado. Es importante tener en cuenta que este procedimiento es necesario porque estamos intentando hacer una representación n de algo continuo con herramientas que funcionan de forma discreta, hay discretizar la variable del tiempo para ser capaces de representar los distintos valores.
Se implemento una función para graficar la señal original(), la cual devuelve el valor constante 2 o 0 según t. Y se grafica usando la librería de mathplot.
Para calcular y ordenar las distintas representaciones de la serie de Fourier se utilizó una lista y_fourier[N] y mediante un for y la función llamada suma_parcial_fourier se crearon listas con los puntos a graficar.



Grafica 1: Representación de serie de Fourier de x(t) para distintos valores de N **.**
Observando la gráfica 1 podemos concluir que la representación de la serie de Fourier realizada en el Problema 2 se aproxima a la función original y al aumentar N esta aproximación parece tender a la función original como lo muestran los valores de la tabla 3.

Sección 2

Problema 1

Para calcular si la señal resultante del calculo de la serie de Fourier realmente converge a la señal original se utiliza[1]:

$$e_N(t) = x(t) - \sum_{k=-N}^{N} a_k e^{Jktw}$$

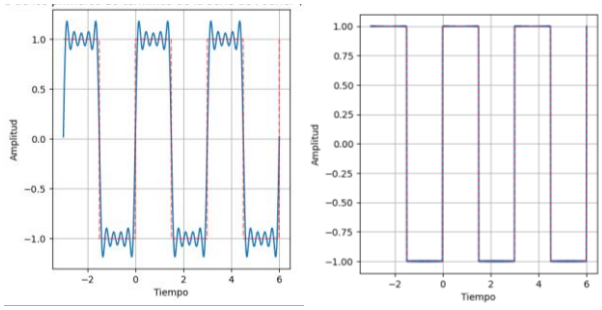
```python
def funcion_error(t, funcion_cuadrada, fourier_sum_real, T=3):
    diferencia = funcion_cuadrada(t, periodo=T) - fourier_sum_real
    #Se calcula la diferencia entre la representacion mediante la serie de fourier de la señal y la señal
    error = (1.0 / T) * np.trapezoid(diferencia**2, t)
    #Calculo de error segun  Sección 3.4"Señales y Sistemas de Oppenheim, Willsky y Young".
    return error
```

Código 3: Función que calcula la serie de Fourier
Se crea una función que se muestra en la línea 8 la cual mediante la función de trapecio y la definición de una función cuadrada es capaz de calcular la energía promedio de $e_N(t)$.

Mediante los valores 'k' se determinó los números de términos a sumar, a continuación, se presenta una tabla que muestran como varia el error al cambiar el valor de 'k'.

k	$E_{e_N(t)}$.
10	1.211e-1
100	1.090e-2
500	1.204e-05

Tabla 4: Valores de la serie de $E_{e_N(t)}$ para distintos valores de 'k'.

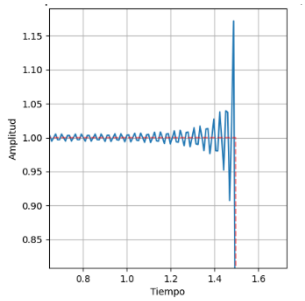


k=10 k=500

Grafica 2: Suma de los primeros 10 y 500 términos de la serie de Fourier (Parte Real)

Al observar la tanto la tabla 4 como la gráfica 2, se percibe como al aumentar los valores de k, la señal efectivamente converge a las serie de Fourier calculada

Problema 2:



Grafica 3: Suma de los primeros 100 términos de la serie de Fourier (Parte Real), con aumento en la discontinuidad.

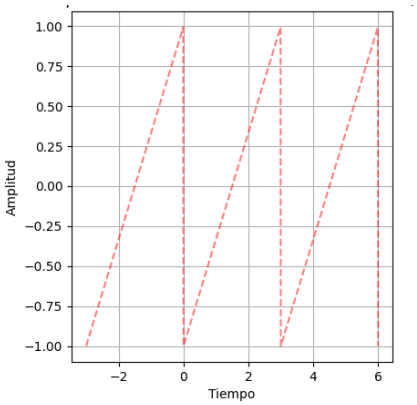
Al observar con más detenimiento que sucede con la gráfica 3 en los puntos de salto de la señal se concluye que presentan el fenómeno de Gibbs. Este ocurre cuando se intenta aproximar mediante el uso de series de Fourier los puntos de discontinuidad de las funciones [2].Este fenómeno no desaparece incluso al aumentar el número de términos de la serie; sin embargo, su amplitud de oscilación se estabiliza, permitiendo calcular que valor real tiene la representación de Fourier en ese punto realizando el promedio entre los lados de la discontinuidad.

Problema 3:

```
8 def funcion_sierra(t, periodo=3.0):
9     # Función sierra: crece linealmente de -1 a 1
  en cada período
10     return 2 * ((t % periodo) / periodo) - 1
11     #Cada vez que t supera periodo, empieza de
  nuevo desde 0
```

Código 4: Función senial de sierra

Se definió una nueva señal, mostrada en la línea 8 del código 4, la cual mediante una función que retorna el valor de la señal '2t' según en que punto múltiplo del periodo fundamental se encuentre construye una señal de sierra como se muestra en la Grafica 4.

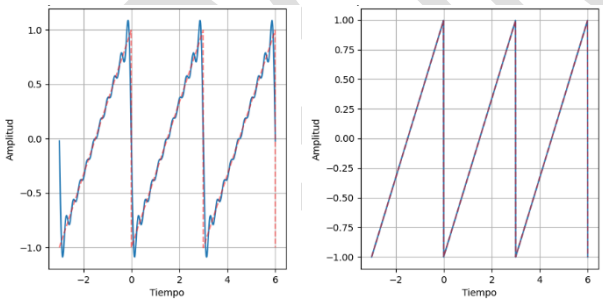


Grafica 4: Señal de sierra con periodo 3.

Se repite el miso procedimiento que el Problema 1 para la señal de sierra.

k	$E_{e_N(t)}$.
10	5.791e-03
100	6.690e-03
500	6.012e-06

Tabla 5: Valores de la serie de $E_{e_N(t)}$ para distintos valores de 'k'.

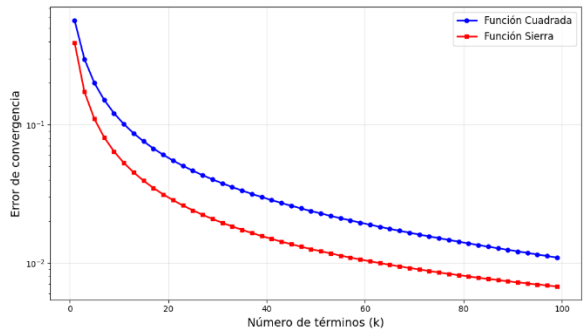


k=10 k=500

Grafica 5: Suma de los primeros 10 y 500 términos de la serie de Fourier para la señal de sierra.

Observando nuevamente la tabla 4 y la gráfica 5 se nota como la serie de Fourier para una señal de sierra también converge a la señal original, sin embargo, si se compara con el caso de la señal cuadrada parece que se necesitan menos valores de k para obtener una aproximación igual de precisa. Esto se deber a que la señal de sierra presenta más 'suavidad', y como la representación mediante la serie de Fourier se realiza con el uso

de funciones, seno y coseno, las cuales son muy suabes, se necesitan menos valores de k para aproximarse a señales que son suabes esto se observa mejor en la gráfica 5 [2].

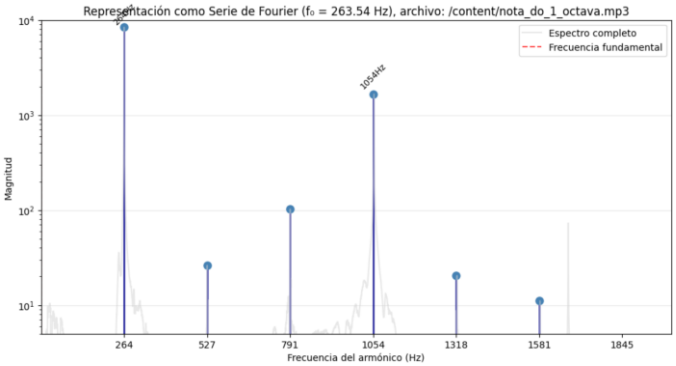


Grafica 5: $E_{e_N(t)}$ en función de, para la señal de sierra y cuadrada.

Sección 3

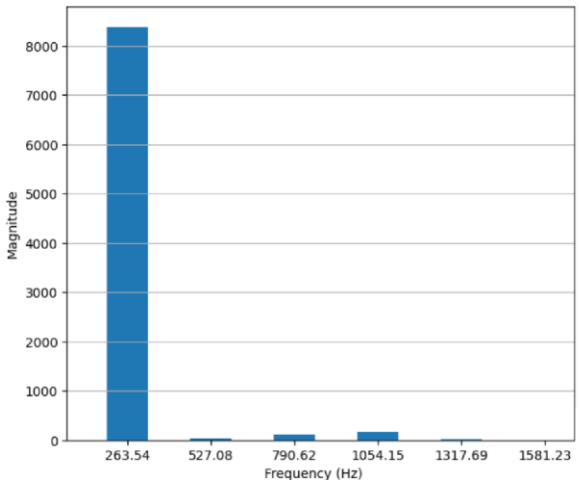
Problema 1

Con el programa provisto se analizaron dos notas, Do (1 octava) y La (2 octava), para el primer caso el programa devuelve diversas graficas, la siguiente grafica muestra la nota descompuesta según sus frecuencias [6].



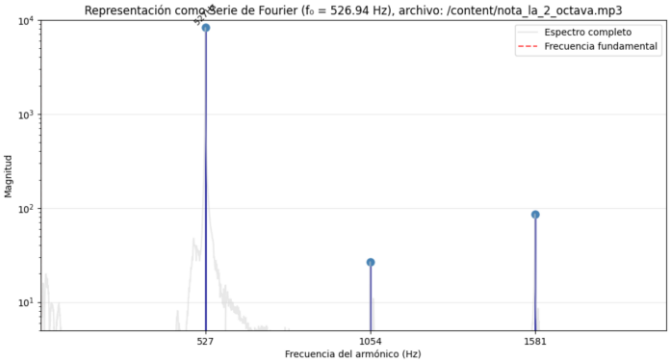
Grafica 6: Descomposición en frecuencias de la nota Do

Como se puede apreciar en la grafica 6 la nota Do está compuesta principalmente de dos frecuencias 264Hz y 1054Hz, también se compone en menor cantidad de las frecuencias 791Hz y 527Hz.
En cuanto a sus armónicos se observa que tiene una predominancia entorno a los 264Hz, el programa también ofrece un gráfico [7] con mayor precisión sobre esto.



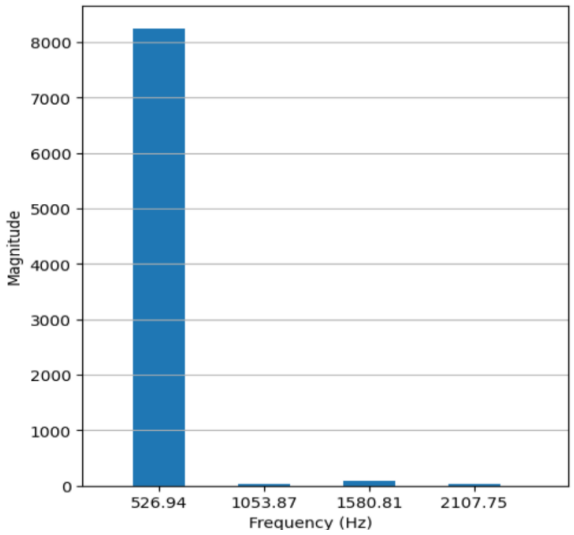
Grafica 7: Contenido Espectral en Frecuencia Fundamental y Armónicos para la nota Do.

Con este grafico se facilita la identificación del armónico dominante, pudiéndose observar que se halla en los 263,54Hz. Ahora para la nota siguiente nota, tratándose de La (2 octava) con el programa se calcula su serie de Fourier entregando la siguiente grafica 8



Grafica 8: Descomposición en frecuencias de la nota La.

Se observa que a diferencia de la nota estudiada anteriormente esta se compone de menos frecuencias, aunque alcanza una magnitud del mismo orden que se veía anteriormente. Principalmente se compone de 527Hz seguido de 1581Hz y 1054Hz.
De igual forma que antes, para identificar el armónico dominante se utiliza otro grafico [9] provisto a continuación por el programa de análisis.



Grafica 9: Descomposición en armónicos

Esta grafica

Relaciones observadas …

REFERENCIAS

[1] A. V. Oppenheim and A. S. Willsky, *Signals and Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1997, ch. 3.

[2] K. Raeen, *A Study of The Gibbs Phenomenon in Fourier Series and Wavelets*, M.S. thesis, Dept. Mathematics, Univ. New Mexico, Albuquerque, NM, USA, 2008.

APÉNDICE

Los apéndices, si son necesarios, aparecen antes del reconocimiento.