UNIVERSIDAD GALILEO DE GUATEMALA

03 DE NOVIEMBRE DEL 2023

MATEMATICA VI - FISICC

PROYECTO 1 - MAP ”MINI APLICACION”

REPORTE PROYECTO 1 - MAP ”MINI APLICACION”

SPEED DESCENDENT ALGORITHM.

Jimmy Israel Mendoza Coquix - 17002925

Pablo Daniel Agreda Barrios - 17001969

RESUMEN

Como objetivo general del MAP, fue desarrollar y aplicar las SERIES DE FOURIER a funciones periódicas, en este caso para funciones periódicas definida a trozos, hasta un máximo de 3.

En este caso, haciendo uso del lenguaje de computación ”Python”, aplicando distintos métodos numéricos de integración y operaciones matemáticas básicas, tratando de simular la ”SERIE DE FOURIER TRUNCADA”. Creando un sistema de comunicación con el usuario ”FRONT END”.

CONTENTS

1 DESCRIPCIÓN GENERAL

1.1 SERIES DE FOURIER

Las series de Fourer describen señales periódicas como la combinación de señales/funciones armónicas. El objetivo y la practica de estas series, es describir dichas señales en una suma de funciones trigonométricas de senos y cosenos, con destinas frecuencias con múltiplos de la ”frecuencia fundamental”.

Serie de Fourier trigonométrica:

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\left(nw_n t\right) + b_n \sin\left(nw_n t\right)\right) \tag{1.1}$$

Coeficientes de Fourier:

$$\begin{aligned} a_0 &= \frac{1}{2L} \int_{-L}^{L} f(t)\,dt \\ a_n &= \frac{1}{L} \int_{-L}^{L} f(t) \cos(n\omega_n t)\,dt \\ b_n &= \frac{1}{L} \int_{-L}^{L} f(t) \sin(n\omega_n t)\,dt \end{aligned} \tag{1.2}$$

1.2 SERIE DE FOURIER TRUNCADA A 2N + 1 TÉRMINOS:

Al querer aproximar una función periódica $f(t)$ la cual es continua por partes y posee infinitos armónicos, tendremos que truncar la función hasta el armónico N de forma tal que el error sea el mínimo establecido.

La forma de calcular la energıa de la señal:

$$E_f = \int_{-L}^{L} f^2(t)\,dt \tag{1.3}$$

1.3 INTEGRAL CUADRADA DEL ERROR (ICE)

El calculo de ICE se utiliza para encontrar el valor N que luego será utilizado para obtener una representación de la Sumatorio de Fourier Truncada.

Serie de Fourier trigonométrica:

$$ICE(N) = E_f - \left\{ a_0^2 T + \frac{T}{2} \sum_{n=1}^{N} \left(a_n^2 + b_n^2 \right) \right\} \tag{1.4}$$

$$ICE(N) \leq 0.02 E_f$$

2 PARTE PRACTICA

2.1 INTERFAZ DE USUARIO

La interfaz de usuario, se basa en un sistema tipo consola, breve instrucción te uso:

- Ingreso de funciones(Ingreso recursivo):
    1. Ingreso de Función/Definición de Función a trozos + ENTER.
- Ingreso de Rango de Operación de Función Ingresada(Ingreso recursivo):
    1. Inicio de rango + ENTER.
    2. Fin de rango + ENTER.
- Despliegue de resultados, se podra visualizar .
- Al Finalizar en Ingreso de la Función a Trozos, PRESIONAR ENTER, cuando la consola solicite ingreso de nueva función.
- Tomar en cuenta las indicaciones de uso en listadas en la interfaz de usuario.

2.1.1 FUNCIÓN A)

$$f(t) = \begin{cases} 0 & \text{si } -2 < t < -1 \\ 1 & \text{si } -1 < t < 1 \\ 0 & \text{si } 1 < t < 2 \end{cases} \tag{2.1}$$

Ingreso de función:

```
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: 0
Inicio de Intervalo: -2
Fin de Intervalo: -1
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: 1
Inicio de Intervalo: -1
Fin de Intervalo: 1
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: 0
Inicio de Intervalo: 1
Fin de Intervalo: 2
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria:
```

Figure 2.1. Se puede percibir el ingreso de función y la secuencia a seguir para el ingreso.

Resultados de la Funcion:

```
RESULTADOS
PERIODO DE LA FUNCION: 4.0
Desea enlisatar los An y Bn de las N iteraciones: Ingrese, SI para enlistar y NO/ENTER para proseguir:
Coeficientes de Fourier para la N termino,  An: -0.05787452476068926 Bn: 0.0
Valor N para la serie: 11
```

Figure 2.2. Se puede percibir el despliegue de resultados para la sumatoria de Fourier para dicha función.

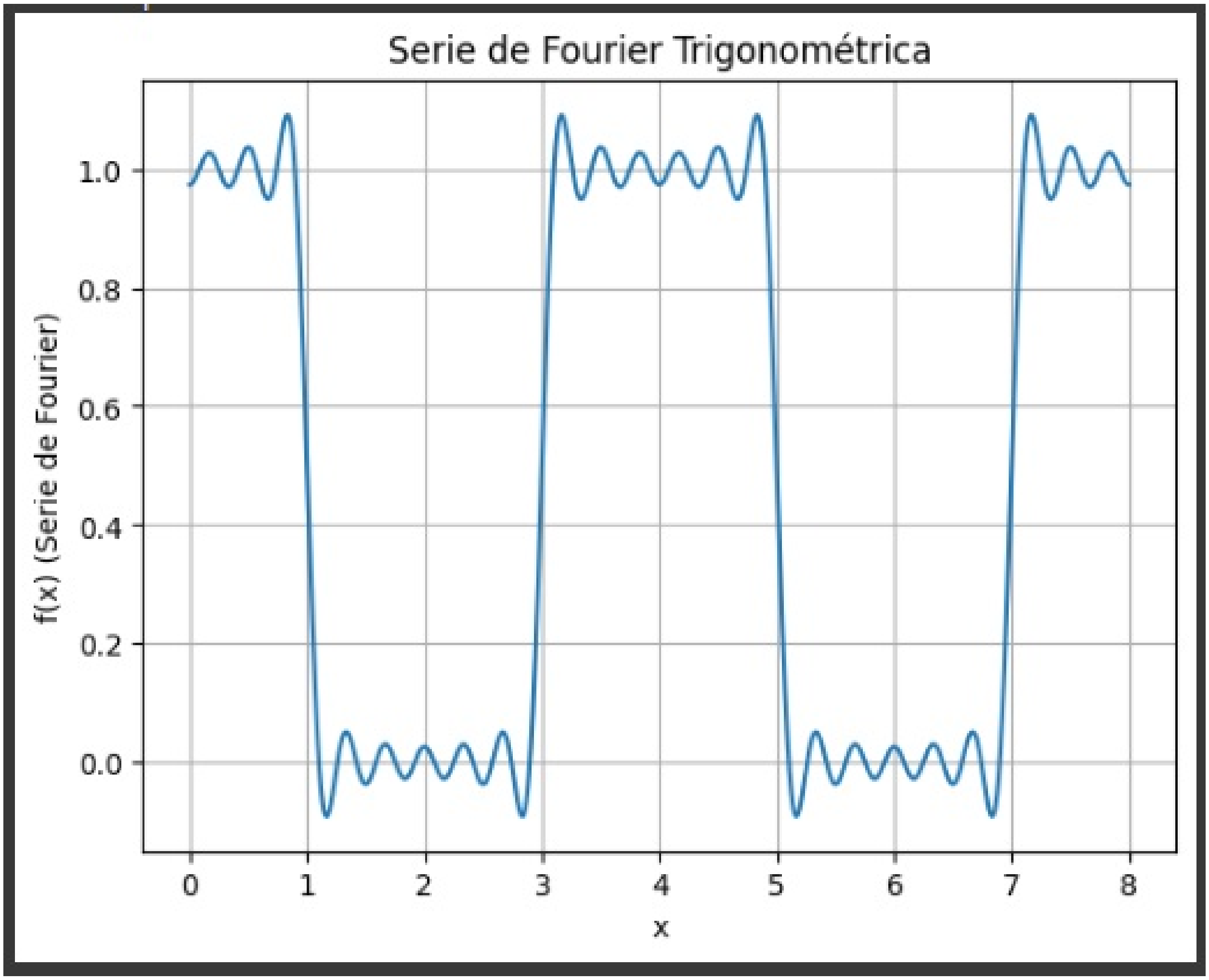

Figure 2.3. Se puede percibir la gráfica resultante de la Función A.

```
Desea enlisatar los An y Bn de las N iteraciones: Ingrese, SI para enlistar y NO/ENTER para proseguir: SI
[0.6366197723675814, 0.0]
[4.163336342344337e-17, 0.0]
[-0.21220659078919377, 0.0]
[-5.551115123125783e-17, 0.0]
[0.12732395447351622, 0.0]
[2.0541712908535467e-17, 0.0]
[-0.09094568176679739, 0.0]
[0.0, 0.0]
[0.07073553026306459, 0.0]
[-1.0175854106141663e-16, 0.0]
[-0.05787452476068926, 0.0]
```

Figure 2.4. Resultados de los ANs y BNs para la funcion A)

2.1.2 FUNCIÓN B)

$$f(t) = \begin{cases} 0 & \text{si } -2 < t < -1 \\ t+2 & \text{si } -1 < t < 1 \\ 0 & \text{si } 1 < t < 2 \end{cases} \tag{2.2}$$

```
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: 0
Inicio de Intervalo: -2
Fin de Intervalo: -1
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: t+2
Inicio de Intervalo: -1
Fin de Intervalo: 1
Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: 0
Inicio de Intervalo: 1
Fin de Intervalo: 2
```

Figure 2.5. Se puede percibir el ingreso de la función B)

```
RESULTADOS
PERIODO DE LA FUNCION: 4.0
Desea enlisatar los An y Bn de las N iteraciones: Ingrese, SI para enlistar y NO/ENTER para proseguir: SI
[1.2732395447351628, 0.4052847345693511]
[1.3877787807814457e-16, 0.31830988618379075]
[-0.42441318157838753, -0.04503163717437229]
[-1.1102230246251565e-16, -0.15915494309189526]
[0.25464790894703243, 0.016211389382773916]
[8.673617379884035e-17, 0.10610329539459692]
[-0.18189136353359478, -0.008271117032027459]
[6.245004513516506e-17, -0.07957747154594766]
[0.1414710605261292, 0.00500351524159693]
[-5.551115123125783e-17, 0.06366197723675814]
[-0.11574904952137854, -0.003349460616275498]
Valor N para la serie: 11
```

Figure 2.6. Resultados de los ANs y BNs para la función B)

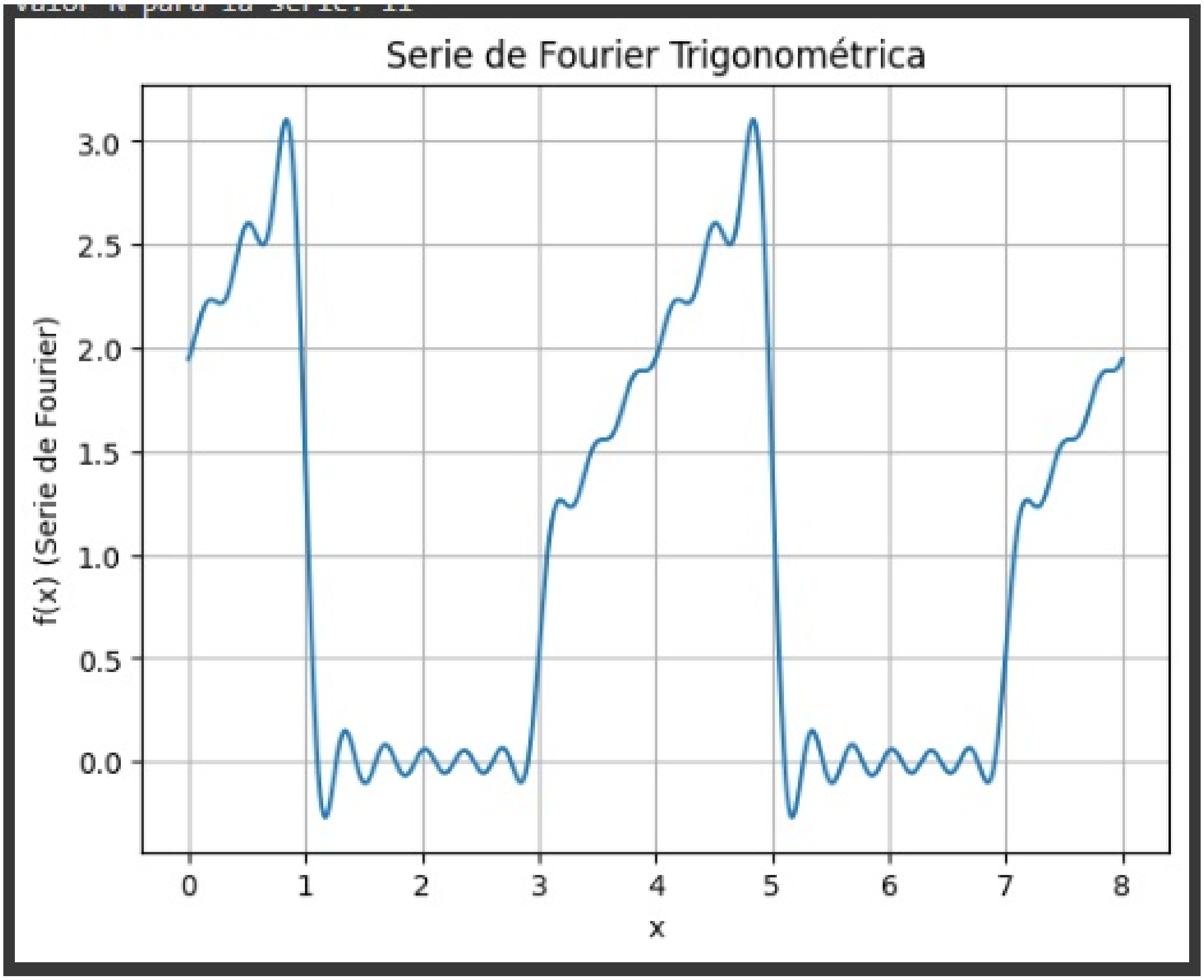

Figure 2.7. Se puede percibir la gráfica resultante de la Función B.

3 DESARROLLO DE CODIGO

3.1 FUNCIÓN ICE

```python
while(True):
#for i in range(1, 13):#ciclo para hacer N iteraciones manuales
# evaluamos cada trozo de la funcion, e iteramos para sacar los n terminos de la sumatoria de An y Bn
# al mismo tiempo almacenamos los valores de los ANs y BNs, para cara iteracion
  for trozo in range(len(matrix)):
    a0 += fourier_a0(matrix[trozo][0], float(matrix[trozo][1]), float(matrix[trozo][2]))
    for i in range(1, N+1):
      an = fourier_an(matrix[trozo][0], float(matrix[trozo][1]), float(matrix[trozo][2]), i)
      bn = fourier_bn(matrix[trozo][0], float(matrix[trozo][1]), float(matrix[trozo][2]), i)
      if(inicio == 1):
        arrayABNs = [float(an), float(bn)]
        matrisABNs.append(arrayABNs)
        if(len(matrisABNs)==len(range(1, N+1))):
          inicio += 1
      else:
        aaa=matrisABNs[i-1][0]+float(an)
        bbb=matrisABNs[i-1][1]+float(bn)
        matrisABNs[i-1][0] = aaa
        matrisABNs[i-1][1] = bbb
      acumulado += (an**2 + bn**2)
    anDesp = matrisABNs[-1][0]
    bnDesp = matrisABNs[-1][1]
    p1 = (a0**2)*(np.abs(float(matrix[0][1])) + np.abs(float(matrix[-1][2])))#PERIODO TOTAL
    a0Desp = a0
  #TERMINA EL FOR DE CADA TROZO
  a0 = 0
  an = 0
  bn = 0

  llaves = ((p1) + (((np.abs(float(matrix[0][1])) + np.abs(float(matrix[-1][2]))))/2)*acumulado)
  ICE_N = energia_final - llaves
  N +=1
  if(ICE_N <= 0.02*energia_final):
    NRetorno= N-1
    break
  else:
    matrisABNs = []
    inicio = 1
  acumulado = 0
  p1 = 0
```

Figure 3.1. Código Python para el algoritmo del calculo de la función ICE.

3.2 FUNCIÓN PARA EL INGRESO DEL USUARIO Y REINICIO DE VARIABLES

```python
def reinicio():
    matrix = []
    matrisABNs = []
    PERIODO_TOTAL = 0
    OMEGA = 0
    PI = 'π'
    NRetorno = 0


def func_input():
    matrix = []
    truncadoGen = 0
    while True:
        function = input("Ingresar valor o la definición de la función/Enter para cálculo de Sumatoria: ")
        if (function == "" and (not not matrix)):
            break
        else:
          if function == "SALIR":
            truncadoGen = -1
            break
          if function != "":
            a = input("Inicio de Intervalo: ")
            b = input("Fin de Intervalo: ")
            a = sp.sympify(a)
            b = sp.sympify(b)
            function = sp.sympify(function)  # Convierte la entrada en una expresión simbólica
            slice_array = [function, float(a), float(b)]
            matrix.append(slice_array)
    return truncadoGen
```

Figure 3.2. Menú de ingreso del usuario, y Matrices utilizadas para el almacenamiento de datos.

3.3 FUNCIÓN PARA EL CALCULO DE ENERGÍA

```python
#-----------------------ENERGIA-----------------------
def energia(f, Ti_trozo,Tf_trozo):
    func_trozo = lambda x: (f.subs(sp.symbols('t'), x))**2  # Utiliza f como una función simbólica
    integral_result, _ = quad(func_trozo, Ti_trozo,Tf_trozo)
    return integral_result

def calcular_energia():
    energia_total = 0  # Inicializa la energía total en cero
    for i in range(len(matrix)):
        #periodo_trozo = abs(float(matrix[i][2]) - float(matrix[i][1]))
        energia_total += energia(matrix[i][0], float(matrix[i][1]),float(matrix[i][2]))  # Acumula la energía en cada iteración
    return energia_total
```

Figure 3.3. En este caso utilizamos hacemos uso de la matriz de ingreso de la función, para el calculo de la energía.

3.4 CALCULO COEFICIENTES DE FOURIER Y FUNCIÓN PARA GRAFICAR

```python
# VERIFICAR LIMITES DE INTEGRACION
#------------Terminos de de la serie de Fourier-----------
def fourier_a0(f, Ti_trozo,Tf_trozo):
    PERIODO_TOTAL = np.abs(float(matrix[0][1])) + np.abs(float(matrix[-1][2]))
    func_trozo = lambda x: f.subs(sp.symbols('t'), x)
    integral_result, _ = quad(func_trozo, Ti_trozo, Tf_trozo)
    a0 = (1 / PERIODO_TOTAL) * integral_result
    return a0

def fourier_an(f, Ti_trozo,Tf_trozo, n):
    PERIODO_TOTAL = np.abs(float(matrix[0][1])) + np.abs(float(matrix[-1][2]))
    omega_n = 2 * np.pi * n / PERIODO_TOTAL
    func_trozo = lambda x: f.subs(sp.symbols('t'), x) * np.cos(omega_n * x)
    integral_result, _ = quad(func_trozo, Ti_trozo, Tf_trozo)
    an = (2 / PERIODO_TOTAL) * integral_result
    return an

def fourier_bn(f, Ti_trozo,Tf_trozo, n):
    PERIODO_TOTAL = np.abs(float(matrix[0][1])) + np.abs(float(matrix[-1][2]))
    omega_n = 2 * np.pi * n / PERIODO_TOTAL
    func_trozo = lambda x: f.subs(sp.symbols('t'), x) * np.sin(omega_n * x)
    integral_result, _ = quad(func_trozo, Ti_trozo, Tf_trozo)
    bn = (2 / PERIODO_TOTAL) * integral_result
    return bn

# VERIFICAR LIMITES DE INTEGRACION
#----------------------------SFT-------------------------
def serie_fourier_trigonometrica(n,w,t):
  acum = 0
  matrix1 = []
  for trozo in range(len(matrix)):
    acum += fourier_a0(matrix[trozo][0], float(matrix[trozo][1]), float(matrix[trozo][2]))
    for i in range(1, n + 1):
      an = fourier_an(matrix[trozo][0], float(matrix[trozo][1]),float(matrix[trozo][2]), i)
      bn = fourier_bn(matrix[trozo][0], float(matrix[trozo][1]),float(matrix[trozo][2]), i)
      acum += (an * np.cos(i*w * t)) + (bn * np.sin(i * w * t))
      #print("TVALUE: {} | ACUM: {}".format(t_value, acum))

  return acum
```

Figure 3.4. Se aprecia el calculo de los coeficientes de Fourier, Ao, AN Y BN, al mismo tiempo el desarrollo de la función trigonométrica de Fourier.

4 ENLACE DE PROYECTO

Figure 4.1. ENLACE DE PROYECTO EN EL LINK.

ACLARACIONES

DENTRO DEL ENLACE ENCUENTRA EL DESARROLLO DEL CÓDIGO DE PROGRAMACIÓN PARA LECTURA Y EJECUCIÓN EN PYTHON, EL CÓDIGO DE LATEX DEL REPORTE Y EL PDF DEL REPORTE.