



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Ingeniería en Inteligencia Artificial

Proyecto fase 4

Unidad de Aprendizaje: Cómputo Paralelo
Grupo 6BV1

Integrantes

Ibarra Gonzalez Emilio Francisco
Tinoco Videgaray Sergio Ernesto
Sánchez De Los Ríos Flavio Josué

Fecha: 11/06/2023

Contexto de la serie de Fourier

La Serie de Fourier, es una herramienta muy poderosa que tiene una amplia variedad de aplicaciones, en diversas áreas como en las ciencias, matemáticas, ingenierías entre otras. Las series de Fourier reciben su nombre en honor a Jean-Baptiste Joseph Fourier, quien hizo importantes contribuciones al estudio de las series trigonométricas, que previamente habían sido consideradas por Leonhard Euler, Jean le Rond d'Alembert y Daniel Bernoulli en el problema de las cuerdas de violín. En 1807 Fourier perfeccionó y aplicó las ideas de Bernoulli, con el fin de resolver la ecuación de conducción del calor en una lámina de metal. En 1822 publicó "Théorie analytique de la Chaleur" donde hizo su primera aparición formal, aunque en ese entonces existía escepticismo respecto a este nuevo método.

La ecuación del calor es una ecuación en derivadas parciales. La cual era difícil dar solución de manera general, a menos que la fuente de calor se comportara de manera sencilla, en particular, si la fuente era una onda de seno o coseno[0]. A estas soluciones simples se les conocen como valores propios. La idea de Fourier era modelar una fuente de calor compleja con una suma de ondas simples sinusoidales, para escribir la solución como una suma de los correspondientes valores propios.

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos \frac{2n\pi}{T} t + b_n \sin \frac{2n\pi}{T} t \right]$$

Figura 1. Serie de Fourier, donde a_n y b_n se denominan coeficientes de Fourier de la serie de la función $f(t)$.

De la serie de Fourier surgen dos conceptos importantes. La Transformada de Fourier Discreta (TFD) la cual es una transformación discreta que transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo discreto. En otras palabras es una aplicación de la serie de Fourier en el tiempo discreto. Y la Transformada Rapida de Fourier (TRF) el cual es un algoritmo para calcular de manera eficiente la TFD. Estos son de gran utilidad de manera práctica, en especial la TFD, ya que permite a los computadores realizar los cálculos de manera rápida lo que permite hacer análisis y procesamiento de señales.

Como se ha aplicado el análisis de Fourier en la ciencia

1. En ciencias exactas como en la física: Para el modelado de fenómenos físicos generalmente se hace uso de las ecuaciones diferenciales. Para dar soluciones a estas ecuaciones se utiliza la transformada de Fourier[0], como originalmente Fourier hizo para la ecuación del calor. Otro ejemplo es la ecuación de las ondas la cual describe el movimiento de las ondas y es de utilidad para campos como la acústica, el electromagnetismo y la dinámica de fluidos. De manera muy general, en el caso de la ecuación de la onda se aplicó de la siguiente manera:

La ecuación de ondas está dada por:

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}, \quad t > 0, \quad x \in \mathbf{R},$$

Para resolver esta ecuación, primero se le debe aplicar la transformada de Fourier. Quedando de la siguiente manera:

$$\frac{\partial^2 \hat{u}}{\partial t^2} = -4\pi^2\gamma^2 \hat{u},$$

después encontramos \hat{u} :

$$\begin{aligned}\hat{u}(t, \gamma) &= \cos 2\pi\gamma t \hat{f}(\gamma) + \frac{\sin 2\pi\gamma t}{2\pi\gamma} \hat{g}(\gamma) \\ &= \frac{1}{2}[e^{2\pi i \gamma t} + e^{-2\pi i \gamma t}] \hat{f}(\gamma) + \frac{1}{2} \int_{-t}^t e^{2\pi i \gamma y} dy \hat{g}(\gamma),\end{aligned}$$

y luego invertimos para finalmente obtener:

$$u(t, x) = \frac{1}{2}[f(x+t) + f(x-t)] + \frac{1}{2} \int_{x-t}^{x+t} g(y) dy.$$

Esta es la llamada fórmula de D'Alembert, la cual describe la variación en el tiempo y el espacio de una cantidad ondulada.

2. En ciencias aplicadas como la cartografía: Antes, la profundidad del fondo marino era medida por medio de el Escandallo que constaba de una soga graduada llamada sondaleza, en cuyo extremo lleva un peso de plomo llamado escandallo con el extremo inferior socavado para que con auxilio de cebo o grasa se puedan extraer muestras para evaluar la calidad del fondo. Hasta que en 1872 en que Sir William Thomson inventó la máquina Sigsbee, que sustituyó la monótona labor de sondeo que requería dos personas lanzando y leyendo continuamente los valores de la sonda. [1]

Como se ha aplicado el análisis de Fourier en la tecnología

1. Telecomunicaciones: El análisis de Fourier nos permite determinar la amplitud y la fase de cada una de las componentes de frecuencia que tiene una señal.
2. Procesamiento de Señales de Audio: Compactar señales de audio (mp3, mp4), producir efectos de sonido, diseñar sintetizadores de audio, diseñar ecualizadores. [2]

Como se ha aplicado el análisis de Fourier en la ingeniería

1. Análisis espectral: Matemáticamente el análisis espectral está relacionado con una herramienta llamada transformada de Fourier o análisis de Fourier. Dada una señal o fenómeno ondulatorio de amplitud (t) esta se puede escribir matemáticamente como la siguiente combinación lineal generalizada. [3]

$$s(t) = \int_{\mathbb{R}} A(v) e^{-2\pi i vt} dw$$

2. Audio: El análisis de Fourier permite conservar el sonido digitalizado para así ser almacenado en una computadora o pendrive. [4]

Como se ha aplicado el análisis de Fourier en la inteligencia artificial

En el área de la inteligencia artificial existen múltiples modelos y algoritmos que trabajan con datos provenientes de algún tipo de señal. Un tipo de señal muy común son las imágenes, ya que se definen como una función de dos variables x, y que mapean un conjunto de elementos formados en una región llamada imagen. Dicha imagen está conformada por pixeles que adquieren ciertos valores en cada punto de la misma de acuerdo a la información que estos presenten como lo puede ser la intensidad de color, la matiz o el nivel de gris, dependiendo el modelo de color que se aplique. Este tipo de señales puede ser ya sea analógico o digital, más comúnmente el digital. [5]

El análisis de Fourier se utiliza para el procesamiento de imágenes en la detección de bordes, ya que para detectar o clasificar algún elemento dentro de nuestra imagen, se necesita reconocer el contorno del objeto de interés, de igual forma podemos aplicar el análisis de Fourier para la eliminación de algún tipo de ruido, como lo puede ser un ruido Gaussiano o sal y pimienta, esto se logra a partir de un filtro como pasa alto, pasa bajo o pasa banda. De igual forma se puede aplicar dicho análisis para la compresión de imágenes.

La transformada de Fourier es fundamental en la convolución en el dominio de la frecuencia, lo que permite mejorar la eficiencia del procesamiento de imágenes o la aplicación de algún filtro.

Dicho filtro se aplica por medio de la multiplicación matricial de una función H con la imagen obtenida de la transformada de Fourier para posteriormente aplicar la transformada inversa de Fourier y así obtener la imagen filtrada como se ilustra en la figura 5.1 [6]

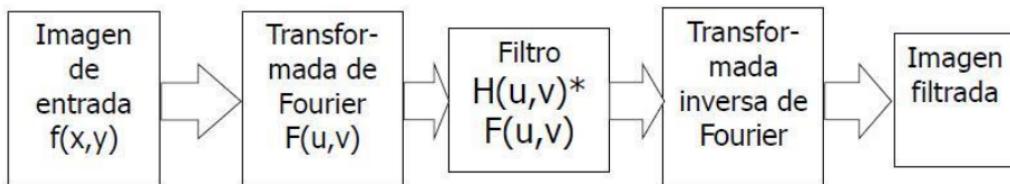


Figura 2. Proceso para filtrar una imagen.

Otra de las aplicaciones más comunes en el análisis de Fourier es en el procesamiento de audio.

El cual se utiliza para la clasificación de sonidos a través de la detección de patrones de sonido, y de igual forma que en el procesamiento de imágenes, aplicar un filtrado que nos permita por ejemplo la eliminación de ruido, la separación de fuentes de audio y por supuesto la compresión de audio por medio de un método conocido como la transformada de Wavelet.

Esto es más simple si consideramos que el sonido se representa por un conjunto de frecuencias o bien, ondas que responden a un comportamiento sinusoidal. [7]

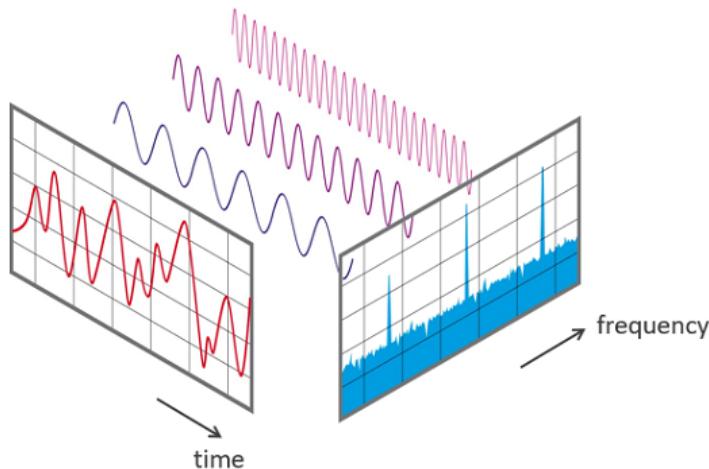


Figura 3.
Descomposición de una señal
de audio en el dominio del tiempo
y de frecuencia.

Una de las aplicaciones más recientes del análisis de Fourier es para el aprendizaje automático por medio de la extracción de características de las señales y para la reducción de la dimensionalidad de los datos. Aquí se utiliza la transformada de Fourier para la extracción de dichas características de las señales de audio y para la reducción de la dimensionalidad de los datos en la clasificación de imágenes o de audios. [8]

Como se ha aplicado el análisis de Fourier en el cómputo paralelo

El análisis de Fourier consta de múltiples operaciones que pueden llegar a requerir una gran serie de cálculos y con ello, un gran coste computacional, sin embargo estos cálculos pueden desarrollarse de manera eficiente con ayuda de una herramienta tan poderosa como lo es el cómputo paralelo.

La transformada de Fourier se puede calcular en sistemas de computación paralela, lo que permite un procesamiento más rápido para grandes conjuntos de datos. Para ello, se dividen los datos en segmentos y se calcula la transformada de Fourier para cada segmento en paralelo utilizando múltiples procesadores o núcleos.

De igual forma, los filtros digitales en audios, que utilizan el análisis de Fourier, se pueden implementar en sistemas de cómputo paralelo para mejorar el rendimiento del procesamiento de señales. Cada procesador se encarga de procesar una parte de la señal, lo que permite un procesamiento más rápido de grandes conjuntos de datos. [9]

$$y = -2x^3 + 4x^2 - x + 4, \quad \pi \leq x \leq 2\pi$$

Tinoco Videgaray Sergio Ernesto

Calcular los coeficientes de Fourier

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} (-2x^3 + 4x^2 - x + 4) dx$$

$$= \frac{1}{\pi} \left[-2 \int x^3 dx + 4 \int x^2 dx - \int x dx + 4 \int dx \right]$$

$$= \frac{1}{\pi} \left[-2 \left[\frac{x^4}{4} \right]_{-\pi}^{\pi} + 4 \left[\frac{x^3}{3} \right]_{-\pi}^{\pi} - \left[\frac{x^2}{2} \right]_{-\pi}^{\pi} + 4x \Big|_{-\pi}^{\pi} \right]$$

$$= \frac{1}{\pi} \left[-\frac{1}{2} (\pi^4 - (-\pi)^4) + \frac{4}{3} (\pi^3 - (-\pi)^3) + 4(\pi - (-\pi)) \right]$$

$$= \frac{1}{\pi} \left[\frac{4}{3} (2\pi^3) + 4(2\pi) \right] = \frac{1}{\pi} \left[\frac{8\pi^3}{3} + 8\pi \right]$$

$$= \frac{1}{\pi} \left[\frac{8\pi^3}{3} + \frac{24\pi}{3} \right] = \frac{1}{\pi} \left[\frac{8\pi^3 + 24\pi}{3} \right] = \frac{8\pi^2 + 24}{3}$$

$$a_u = \frac{1}{\pi} \int_{-\pi}^{\pi} (-2x^3 + 4x^2 - x + 4) \cos(ux) dx$$

$$= \frac{1}{\pi} \left[\int -2x^3 \cos(ux) dx + \int 4x^2 \cos(ux) dx - \int x \cos(ux) dx + \int 4 \cos(ux) dx \right]$$

$$= \frac{1}{\pi} \left[-2 \int x^3 \cos(ux) dx + 4 \int x^2 \cos(ux) dx - \int x \cos(ux) dx + 4 \int \cos(ux) dx \right]$$

Aplicando la ecuaciones 3,5 y 7 se obtienen las integrales por partes.

$$\int x \cos(ax) dx = \frac{1}{a^2} \cos(ax) + \frac{x}{a} \sin(ax) + c \quad (3)$$

$$\int x^2 \cos(ax) dx = \frac{2x}{a^2} \cos(ax) + \left(\frac{x^2}{a} - \frac{2}{a^3} \right) \sin(ax) + c \quad (5) \quad [10]$$

$$\int x^3 \cos(ax) dx = \left(\frac{3x^2}{a^2} - \frac{6}{a^4} \right) \cos(ax) + \left(\frac{x^3}{a} - \frac{6x}{a^3} \right) \sin(ax) + c \quad (7)$$

$$= \frac{1}{\pi} \left[-2 \left(\frac{3x^2}{u^2} - \frac{6}{u^4} \right) \cos(ux) - 2 \left(\frac{x^2}{u} - \frac{6x}{u^3} \right) \sin(ux) \right. \\ + 4 \left(\frac{2x}{u^2} \cos(ux) \right) + 4 \left(\frac{x^2}{u} - \frac{2}{u^3} \right) \sin(ux) \\ - \frac{1}{u^2} \cos(ux) - \frac{x}{u} \sin(ux) \\ \left. + \frac{4}{u} \sin(ux) \right]_{-\pi}^{\pi}$$

$$= \frac{1}{\pi} \left[\left(-\frac{6x^2}{u^2} + \frac{12}{u^4} \right) \cos(ux) + \left(-\frac{2x^3}{u} + \frac{12x}{u^3} \right) \sin(ux) \right. \\ + \left(\frac{8x}{u^2} \right) \cos(ux) + \left(\frac{4x^2}{u} - \frac{8}{u^3} \right) \sin(ux) \\ - \frac{1}{u^2} \cos(ux) - \frac{x}{u} \sin(ux) \\ \left. + \frac{4}{u} \sin(ux) \right]_{-\pi}^{\pi}$$

$$= \frac{1}{\pi} \left[\left(-\frac{6(-\pi)^2}{u^2} + \frac{12}{u^4} \right) \cos(u\pi) + \left(-\frac{2\pi^3}{u} + \frac{12\pi}{u^3} \right) \sin(u\pi) \right. \\ - \left(-\frac{6(-\pi)^2}{u^2} + \frac{12}{u^4} \right) \cos(u\pi) - \left(\frac{-2(-\pi)^3}{u} + \frac{12(-\pi)}{u^3} \right) \sin(u\pi) \\ + \left(\frac{8\pi^2}{u^2} \right) \cos(u\pi) + \left(\frac{4\pi^2}{u} - \frac{8}{u^3} \right) \sin(u\pi) \left. \right]$$

$$= \left(\frac{8(-\pi)}{u^2} \right) \cos(u\pi) - \left(\frac{4(-\pi)^2}{u} - \frac{8}{u^3} \right) \sin(u\pi) \\ - \frac{1}{u^2} \cos(u\pi) - \left(\frac{\pi}{u} \right) \sin(u\pi) + \frac{1}{u^2} \cos(u\pi) + \left(\frac{\pi}{u} \right) \sin(u\pi) \\ + \frac{4}{u} \sin(u\pi) - \frac{4}{u} \sin(u\pi) \left. \right]$$

Notese que $(-\pi)^2 = \pi^2$

$$\begin{aligned}\operatorname{sen}(u\pi) &= \operatorname{sen}(u\pi) \\ \cos(u\pi) &= \cos(u\pi)\end{aligned}$$

$$\begin{aligned}
 &= \frac{1}{\pi} \left[\left(-\frac{6\pi^2}{u^2} + \frac{12}{u} \right) \cos(u\pi) + \left(-\frac{2\pi^3}{u} + \frac{12\pi}{u^3} \right) \operatorname{sen}(u\pi) \right. \\
 &\quad - \left(-\frac{6(\pi)^2}{u^2} + \frac{12}{u} \right) \cos(u\pi) + \left(-\frac{2(\pi)^3}{u} + \frac{12(\pi)}{u^3} \right) \operatorname{sen}(u\pi) \\
 &\quad + \left(\frac{8\pi}{u^2} \right) \cos(u\pi) + \left(\frac{4\pi^2}{u} - \frac{8}{u} \right) \operatorname{sen}(u\pi) \\
 &\quad + \left(\frac{8(\pi)}{u^2} \right) \cos(u\pi) - \left(\frac{4\pi^2}{u} - \frac{8}{u^3} \right) \operatorname{sen}(u\pi) \\
 &\quad - \cancel{\frac{1}{u^2} \cos(u\pi)} - \cancel{\frac{(\pi)}{u} \operatorname{sen}(u\pi)} + \cancel{\frac{1}{u^2} \cos(u\pi)} - \cancel{\frac{(\pi)}{u} \operatorname{sen}(u\pi)} \\
 &\quad \left. + \frac{u}{u} \operatorname{sen}(u\pi) - \cancel{\frac{u}{u} \operatorname{sen}(u\pi)} \right] \\
 &= \frac{1}{\pi} \left[2 \left(-\frac{2\pi^3}{u} + \frac{12\pi}{u^3} \right) \operatorname{sen}(u\pi) + 2 \left(\frac{8\pi}{u^2} \cos(u\pi) \right) - \frac{2\pi}{u} \operatorname{sen}(u\pi) \right]
 \end{aligned}$$

Considerando las expresiones 1 y 2

$$\cos(n\pi) = (-1)^n \text{ donde } n \in \mathbb{N} \quad (1)$$

$$\sin(n\pi) = 0 \text{ donde } n \in \mathbb{N} \quad (2)$$

$$\begin{aligned}
 &= \frac{1}{\pi} \left[2 \left(-\frac{2\pi^3}{u} + \frac{12\pi}{u^3} \right) \operatorname{sen}(u\pi) + 2 \left(\frac{8\pi}{u^2} \cos(u\pi) \right) - \frac{2\pi}{u} \operatorname{sen}(u\pi) \right] \\
 &= \frac{1}{\pi} \left[-\frac{16\pi^4}{u^2} (-1)^u \right] \Rightarrow \frac{16}{u^2} (-1)^u \\
 \text{but } u &= \frac{1}{\pi} \int_{\pi}^{\pi} (-2x^3 + 4x^2 - x + 4) \operatorname{sen}(ux) dx
 \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\pi} \left[\int -2x^3 \operatorname{sen}(ux) dx + \int 4x^2 \operatorname{sen}(ux) dx - \int x \operatorname{sen}(ux) dx + \int 4 \operatorname{sen}(ux) dx \right] \\
&= \frac{1}{\pi} \left[-2 \int x^3 \operatorname{sen}(ux) dx + 4 \int x^2 \operatorname{sen}(ux) dx - \int x \operatorname{sen}(ux) dx + 4 \int \operatorname{sen}(ux) dx \right]
\end{aligned}$$

Aplicando la ecuaciones 4, 6 y 8 se obtienen las integrales por partes.

$$\int x \sin(ax) dx = \frac{1}{a^2} \sin(ax) - \frac{x}{a} \cos(ax) + c \quad (4)$$

$$\int x^2 \sin(ax) dx = \frac{2x}{a^2} \sin(ax) - \left(\frac{x^2}{a} - \frac{2}{a^3} \right) \cos(ax) + c \quad (6)$$

$$\int x^3 \sin(ax) dx = \left(\frac{3x^2}{a^2} - \frac{6}{a^4} \right) \sin(ax) - \left(\frac{x^3}{a} - \frac{6x}{a^3} \right) \cos(ax) + c \quad (8)$$

$$\begin{aligned}
&= \frac{1}{\pi} \left[-2 \left(\frac{-2x^2}{u^2} - \frac{6}{u^4} \right) \operatorname{sen}(ux) - (-2) \left(\frac{x^3}{u} - \frac{6x}{u^3} \right) \cos(ux) \right. \\
&\quad \left. + 4 \left(\frac{2x}{u^2} \operatorname{sen}(ux) \right) - 4 \left(\frac{x^2}{u} - \frac{2}{u^3} \right) \cos(ux) \right. \\
&\quad \left. - \frac{1}{u^2} \operatorname{sen}(ux) + \frac{x}{u} \cos(ux) \right. \\
&\quad \left. - \frac{4}{u} \cos(ux) \right] \underbrace{\quad}_{\text{...}}
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\pi} \left[\left(-\frac{6x^2}{u^2} + \frac{12}{u^4} \right) \operatorname{sen}(ux) - \left(-\frac{2x^3}{u} + \frac{12x}{u^3} \right) \cos(ux) \right. \\
&\quad \left. + \left(\frac{8x}{u^2} \operatorname{sen}(ux) \right) - \left(\frac{4x^2}{u} - \frac{8}{u^3} \right) \cos(ux) \right. \\
&\quad \left. - \frac{1}{u^2} \operatorname{sen}(ux) + \frac{x}{u} \cos(ux) \right. \\
&\quad \left. - \frac{4}{u} \cos(ux) \right] \underbrace{\quad}_{\text{...}}
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\pi} \left[\left(-\frac{6u^2}{n^2} + \frac{12}{n^4} \right) \operatorname{sen}(un) - \left(-\frac{2u^3}{n} + \frac{12u}{n^3} \right) \cos(un) \right. \\
&\quad \left. - \left(-\frac{6u^2}{n^2} + 12 \right) \operatorname{sen}(un) + \left(-2u^3 + 12u \right) \cos(un) \right]
\end{aligned}$$

$$\begin{aligned}
& \text{LHS} = \frac{1}{n^2} \left[-\left(\frac{-6u^2}{n^2} + \frac{12}{n^4} \right) \sin(u^n) + \left(-\frac{2u^3}{n} + \frac{12u}{n^3} \right) \cos(u^n) \right. \\
& \quad \left. - \left(\frac{8u}{n^2} \sin(u^n) \right) - \left(\frac{4u^2}{n} - \frac{8}{n^3} \right) \cos(u^n) \right. \\
& \quad \left. - \left(-\frac{8u}{n^2} \sin(u^n) \right) + \left(\frac{4u^2}{n} - \frac{8}{n^3} \right) \cos(u^n) \right. \\
& \quad \left. + \frac{1}{n^2} \sin(u^n) - \frac{(-1)}{n} \cos(u^n) \right. \\
& \quad \left. - \frac{4}{n} \cos(u^n) + \frac{4}{n} \cos(u^n) \right] \\
= & \frac{1}{n^2} \left[-\left(\frac{-2u^3}{n} + \frac{12u}{n^3} \right) \cos(u^n) - \left(-\frac{2u^3}{n} + \frac{12u}{n^3} \right) \cos(u^n) \right. \\
& \quad \left. + \left(\frac{8u}{n^2} \sin(u^n) \right) + \left(\frac{8u}{n^2} \sin(u^n) \right) \right. \\
& \quad \left. + \frac{4}{n} \cos(u^n) + \frac{4}{n} \cos(u^n) \right] \\
= & \frac{1}{n^2} \left[-2 \left(\frac{-2u^3}{n} + \frac{12u}{n^3} \right) \cos(u^n) \right. \\
& \quad \left. + 2 \left(\frac{8u}{n^2} \sin(u^n) \right) \right. \\
& \quad \left. + 2 \frac{4}{n} \cos(u^n) \right] \\
= & \frac{1}{n^2} \left[\left(\frac{4u^3}{n} - \frac{24}{n^3} \right) (-1)^n + \frac{32}{n} (-1)^n \right] = \left(\frac{4u^2}{n} - \frac{24}{n^3} \right) (-1)^n + \frac{2}{n} (-1)^n \\
= & \left(\frac{4u^2}{n^3} - \frac{24}{n^3} \right) (-1)^n + \frac{2}{n} (-1)^n = \left(\frac{4u^2 n^2 - 24}{n^3} + \frac{2}{n} \right) (-1)^n \\
= & \left(\frac{4u^2 n^2 - 24 + 2n^2}{n^3} \right) (-1)^n \Rightarrow 2(-1)^n \left(\frac{2u^2 n^2 - u^2 - 12}{n^3} \right)
\end{aligned}$$

Aplicando la definición de
la serie de Fourier

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

$$f(x) = \frac{4x^2 + 12}{3 \cdot 2} + \sum_{n=1}^{\infty} \left[\frac{16}{n^2} (-1)^n \cos(nx) + 2(-1)^n \left(\frac{2\pi n^2 - n^2 - 12}{n^3} \right) \sin(nx) \right]$$

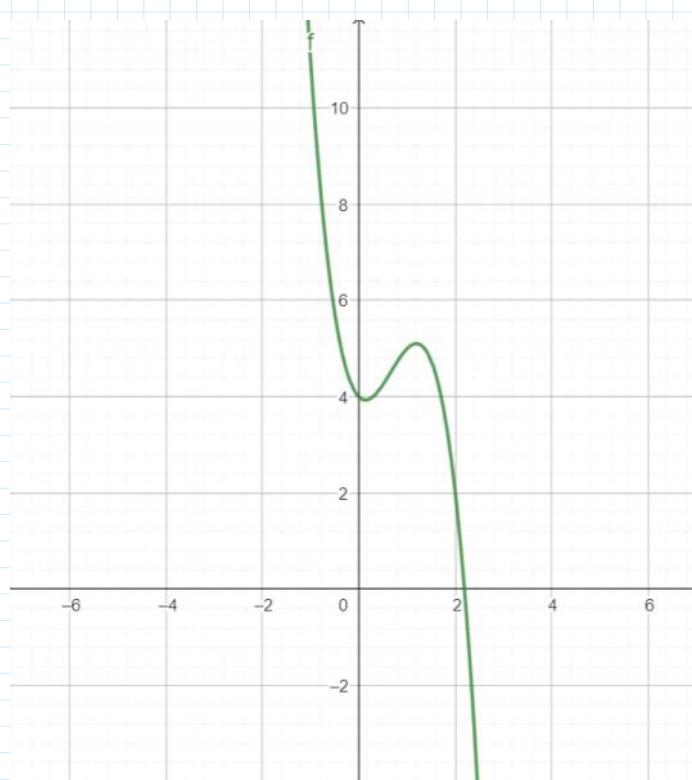
$$\frac{4x^2 + 12}{3} + \sum_{n=1}^{\infty} \left[\frac{16}{n^2} (-1)^n \cos(nx) + \left(\frac{4\pi n^2 - 2n^2 - 24}{n^3} \right) \sin(nx) \right]$$

Solución

Vista gráfica en Geogebra

Función algebraica inicial

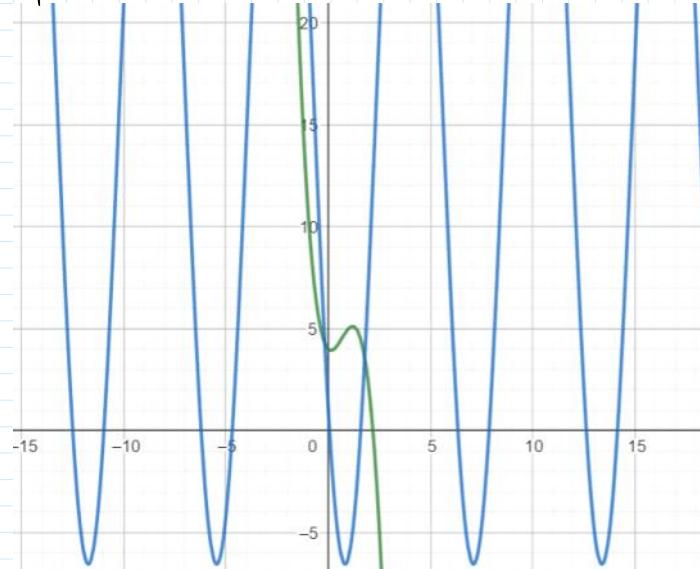
$$f : y = -2x^3 + 4x^2 - x + 4$$



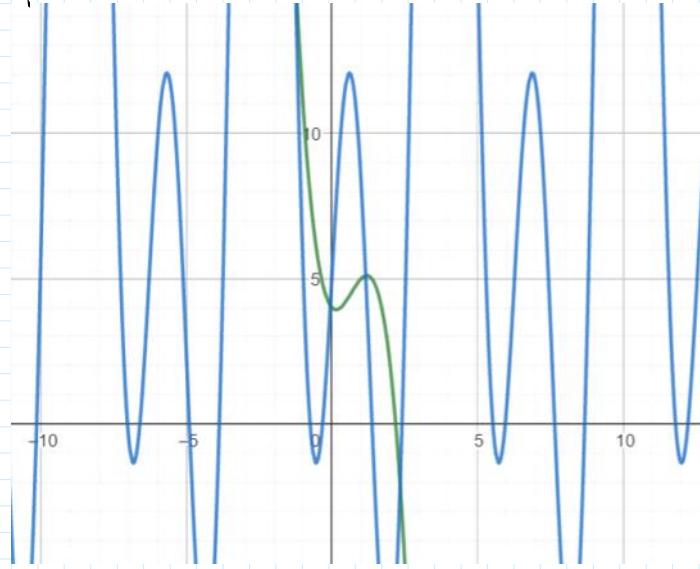
Se define la Serie de Fourier por medio del comando "Suma"

$$s_1(x) = \frac{4\pi^2 + 12}{3} + \text{Suma}\left(\frac{16}{n^2} (-1)^n \cos(nx) + (-1)^n \cdot \frac{4\pi^2 n^2 - 24 + 2n^2}{n^3} \sin(nx), n, 1, a\right)$$

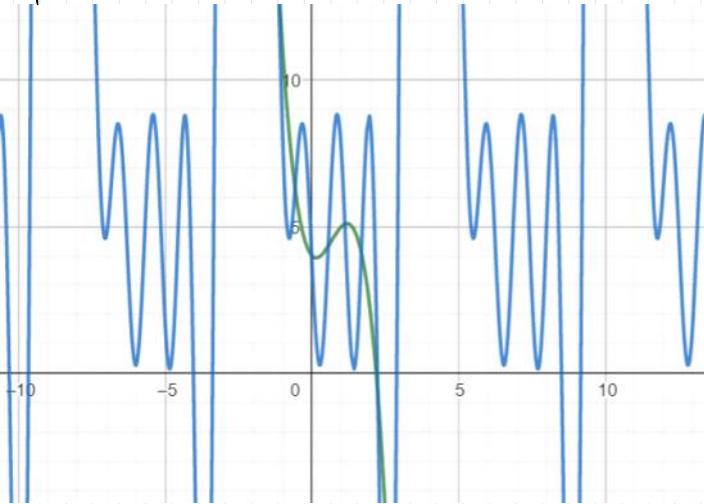
Función con $\omega=1$



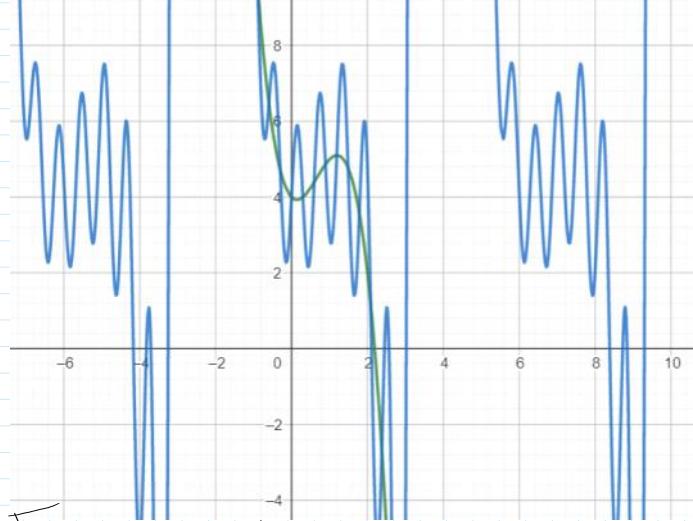
Función con $\omega=2$



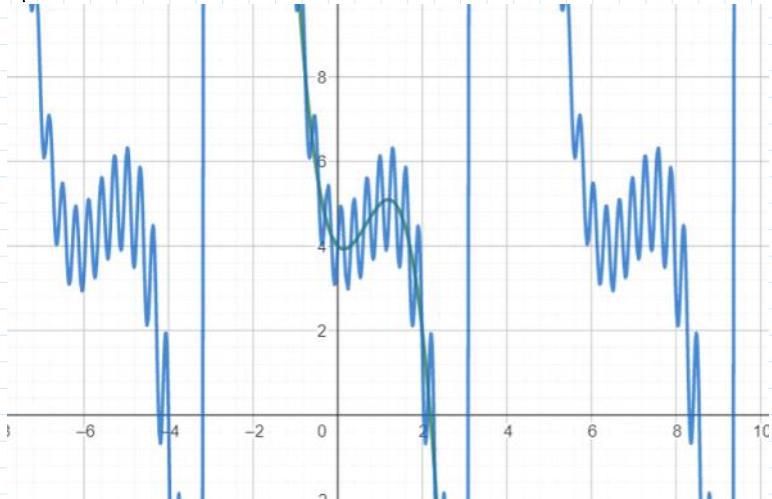
Función con $\omega=5$



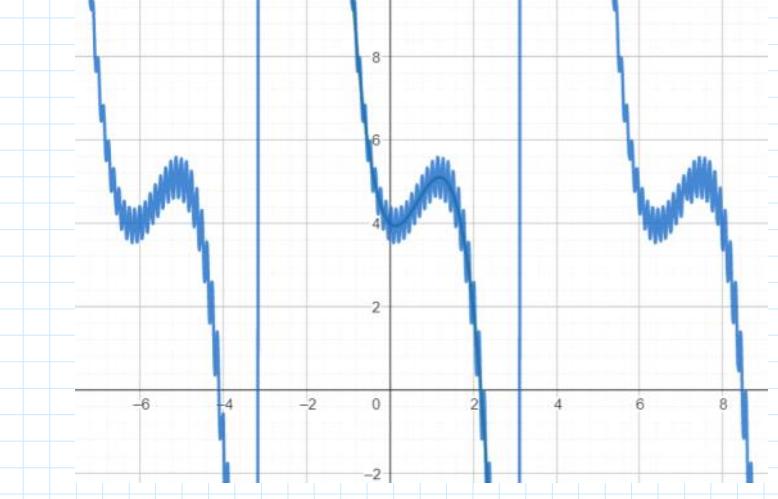
Función con $\omega=10$



Función con $\omega=20$



Función con $\omega=50$



$$f(x) = x^2 - x + 3, -2 \leq x \leq 2$$

usando a formula: $a_0 = \frac{1}{L} \int_{-L}^L f(x) dx$

$$a_0 = \frac{1}{2} \int_{-2}^2 (x^2 - x + 3) dx$$

$$a_0 = \frac{1}{2} \left(\frac{x^3}{3} - \frac{x^2}{2} + 3x \right) \Big|_{-2}^2 =$$

$$\frac{1}{2} \left[\left(\frac{(2)^3}{3} - \frac{(2)^2}{2} + 3(2) \right) - \left(\frac{(-2)^3}{3} - \frac{(-2)^2}{2} + 3(-2) \right) \right]$$

$$= \frac{1}{2} \left[\frac{8}{3} - \frac{4}{2} + 6 - \left(-\frac{8}{3} - \frac{4}{2} - 6 \right) \right] = \frac{1}{2} \left[\frac{8}{3} + 4 + \frac{8}{3} + 6 \right] =$$

$$\frac{1}{2} \left[\frac{52}{3} \right] = \frac{26}{3}$$

usando a formula: $a_n = \frac{1}{L} \int_{-L}^L f(x) \cos\left(\frac{n\pi x}{L}\right) dx$

$$a_n = \frac{1}{2} \int_{-2}^2 (x^2 - x + 3) \cos\left(\frac{n\pi x}{2}\right) dx$$

$$x^2 - x + 3 \quad x^2 - x + 3 \\ 2x+1 \quad 2x+1 \quad + \cos\left(\frac{n\pi x}{2}\right)$$

$$2 \quad - \frac{2}{n\pi} \sin\left(\frac{n\pi x}{2}\right)$$

$$0 \quad + \frac{-4}{n^2\pi^2} \cos\left(\frac{n\pi x}{2}\right)$$

$$\frac{8}{n^3\pi^3} \sin\left(\frac{n\pi x}{2}\right)$$

$$a_n = \frac{1}{2} (2x-1) \left(\frac{4}{n^2\pi^2} \cos\left(\frac{n\pi x}{2}\right) \right) \Big|_{-2}^2$$

$$a_n = \frac{2}{n^2\pi^2} (2x-1) \left(\cos\left(\frac{n\pi x}{2}\right) \right) \Big|_{-2}^2 \quad \begin{aligned} \cos(n\pi) &= (-1)^n \\ \cos(-n\pi) &= (-1)^n \end{aligned}$$

$$a_n = \frac{2}{n^2\pi^2} [3(\cos(n\pi)) - (-5)\cos(-n\pi)]$$

$$a_n = \frac{2}{n^2\pi^2} [3(-1)^n + 5(-1)^n]$$

$$a_n = \frac{2}{n^2\pi^2} [8(-1)^n] \quad a_n = \frac{16(-1)^n}{n^2\pi^2}$$

$$b_n = \frac{1}{2} \int_{-2}^2 (x^2-x+3) \sin\left(\frac{n\pi x}{2}\right) dx$$

$$\begin{aligned} & x^2-x+3 + \sin\left(\frac{n\pi x}{2}\right) \\ & 2x-1 \quad \frac{-2}{n\pi} \cos\left(\frac{n\pi x}{2}\right) \\ & 2 \quad \frac{-4}{n^2\pi^2} \sin\left(\frac{n\pi x}{2}\right) \\ & 0 \quad \frac{8}{n^3\pi^3} \cos\left(\frac{n\pi x}{2}\right) \end{aligned}$$

$$b_n = \frac{1}{2} \left[-\frac{2}{n\pi} (x^2-x+3) \cos\left(\frac{n\pi x}{2}\right) + \frac{16}{n^3\pi^3} \cos\left(\frac{n\pi x}{2}\right) \right] \Big|_{-2}^2$$

$$b_n = \frac{1}{2} \left[-\frac{2}{n\pi} (5) \cos(n\pi) + \frac{16}{n^3\pi^3} \cos(n\pi) \right] - \frac{1}{2} \left[\frac{2}{n\pi} (9) \cos(-n\pi) + \frac{16}{n^3\pi^3} \cos(-n\pi) \right]$$

$$b_n = \frac{1}{2} \left[-\frac{10}{n\pi} (-1)^n + \frac{16}{n^3\pi^3} (-1)^n \right] - \frac{1}{2} \left[\frac{18}{n\pi} (-1)^n + \frac{16}{n^3\pi^3} (-1)^n \right]$$

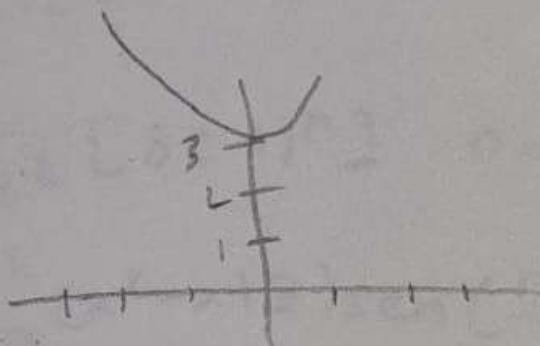
$$b_n = -\frac{5}{n\pi} (-1)^n + \frac{8}{n^3\pi^3} (-1)^n + \frac{9}{n\pi} (-1)^n - \frac{8}{n^3\pi^3} (-1)^n$$

$$b_n = -\frac{5}{n\pi} (-1)^n + \frac{8}{n^3\pi^3} (-1)^n + \frac{9}{n\pi} (-1)^n - \frac{8}{n^3\pi^3} (-1)^n$$

$$b_n = \frac{4}{n\pi} (-1)^n$$

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)]$$

$$f(x) = \frac{13}{3} + \sum_{n=1}^{\infty} \left[\frac{16(-1)^n}{n^2\pi^2} \cos\left(\frac{n\pi x}{2}\right) + \frac{4(-1)^n}{n\pi} \sin\left(\frac{n\pi x}{2}\right) \right]$$



$$a_0 = \frac{26}{3}$$

$$a_n = \frac{16(-1)^n}{n^2\pi^2}$$

$$b_n = \frac{4}{n\pi} (-1)^n$$

Izquira González Emilio Francisco

$$f(x) = -\frac{(\pi+x)}{2}, \quad -\pi < x < 0$$

$$= \frac{\pi-x}{2}, \quad 0 < x < \pi$$

$$f(x) = -\frac{(\pi+x)}{2}, \quad -\pi < x < 0 \quad (1)$$

$$f(-x) = -\frac{(\pi-x)}{2}, \quad -\pi < -x < 0$$



$$f(-x) = -\frac{(\pi-x)}{2}, \quad 0 < x < \pi \quad (2)$$

de 1 v 2 y 3 v 4

$$f(x) = \frac{\pi-x}{2}, \quad 0 < x < \pi \quad (3)$$

$$f(-x) = \frac{\pi+x}{2}, \quad 0 < -x < \pi$$



$$f(-x) = \frac{\pi+x}{2} \quad -\pi < x < 0$$

$$f(x) = -f(-x) \quad -\pi < x < 0$$

$$f(x) = -f(-x) \quad 0 < x < \pi$$

∴ $f(x)$ es impar

$$a_0 = \frac{1}{2} \int_{-\pi}^{\pi} f(x) dx = 0 \quad y \quad a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx = 0$$

$$\begin{aligned} b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx = \frac{2}{\pi} \int_0^{\pi} f(x) \sin nx dx = \frac{2}{\pi} \int_0^{\pi} \frac{\pi-x}{2} \sin nx dx \\ &= \frac{2}{\pi} \int_0^{\pi} \frac{\pi}{2} \sin nx dx - \frac{2}{\pi} \int_0^{\pi} \frac{x}{2} \sin nx dx = \int_0^{\pi} \frac{\pi}{2} \sin nx dx - \frac{1}{\pi} \int_0^{\pi} x \sin nx dx \\ &= \left[-\frac{\cos nx}{n} \right]_0^{\pi} - \frac{1}{\pi} \left[x \cdot -\frac{\cos nx}{n} \Big|_0^{\pi} - 1 \left(\frac{-\sin nx}{n^2} \right) \Big|_0^{\pi} \right] \end{aligned}$$

$$-\frac{\cos n\pi + \cos 0}{n} - \frac{1}{\pi} \left[-\frac{\pi \cos n\pi + 0}{n} \right] = \frac{\cos n\pi}{n} + \frac{1}{n} + \frac{\cos n\pi}{n}$$

$$b_n = \frac{1}{n}$$

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$$

$$f(x) = \sum_{n=1}^{\infty} \frac{1}{n} \sin nx$$

$$f(x) = \sin x + \frac{\sin 2x}{2} + \frac{\sin 3x}{3} + \dots$$

Resultados de las series de Fourier realizadas en Excel (figura 4).

Sergio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-3	17.74195435	8.238460015	6.401975821	5.670115088	5.235696846	4.901000007	4.59981307	4.306051269	4.008827074	3.703953654	81.96731973	
-2	18.91423983	9.296898012	-4.94928364	-9.04438607	-3.43635224	3.66707764	5.189808358	-3.2330217	-3.33437052	31.32594195		
-1	2.696860441	-15.9762094	3.397531518	6.153495434	-7.18522322	2.141037971	3.225403371	-4.62494653	1.882580738	1.891597087	10.76159993	
0	-16	4	-1.7777778	1	-0.64	0.44444444	-0.32653061	0.25	-0.19753086	0.16	4.07207725	
1	-19.9865342	12.64703472	0.122441803	-7.46078268	6.822135624	-1.28755327	-3.7177477	4.552196515	-1.522652804	-2.16009998	5.167935299	
2	-5.59754106	-14.526047	1.535344845	8.753386006	4.510363801	-2.91698523	-5.27910613	-1.57468949	2.972155838	3.46495678	8.501310905	
3	13.93780554	-0.55709772	-3.16240156	-3.98240717	-4.26329632	-4.31405182	-4.24211233	-4.09396177	-3.89341421	-3.65459319	-1.06605801	
Flavio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	
-1.5	2.046634653	0.636619772	0.172736735	-0.10132118	-0.225916	-0.12220659	-0.10522236	0.025330296	0.114187224	0.127323954	6.811499832	
-1	1.273239545	-0.40528473	-0.424441318	0.101321184	0.254647909	-0.04503164	-0.18189136	0.025330296	0.141471061	-0.01621139	5.056511021	
0	-1.62113894	0.405284735	-0.18012655	0.101321184	-0.06484556	0.045031637	-0.03308447	0.025330296	-0.02001406	0.016211389	3.007303	
1	-1.27323954	-0.40528473	0.424441318	0.101321184	-0.25464791	-0.04503164	0.181891364	0.025330296	-0.14147106	-0.01621139	2.930403083	
1.5	0.24600202	-0.63661977	-0.42747474	-0.10132118	0.13421053	0.212206591	0.152010868	0.025330296	-0.08588307	-0.12732395	3.724471517	
2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	
Emilio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
0	0	0	0	0	0	0	0	0	0	0	0	
0.5	0.479425539	0.420735492	0.332498329	0.227324357	0.119694429	0.023520001	-0.05011189	-0.09460031	-0.10861446	-0.09589243	1.25397906	
1	0.841470985	0.454648713	0.047040003	-0.18920062	-0.19178485	-0.04656925	0.093855228	0.123669781	0.045790943	-0.05440211	1.124518813	
1.5	0.997494987	0.070560004	-0.32584337	-0.06985387	0.187599995	0.068686414	-0.12567082	-0.06707161	0.089309381	0.065028784	0.89023988	
2	0.909297427	-0.37840125	-0.0931385	0.247339562	-0.10880422	-0.08942882	0.141515337	-0.03598791	-0.08344303	0.091294525	0.600243119	
2.5	0.598472144	-0.47946214	0.312666659	-0.13600528	-0.01326438	0.108381307	-0.13937514	0.114118156	-0.0541305	-0.01323518	0.298165652	
3	0.141120008	-0.13970775	0.137372828	-0.13414323	0.130057568	-0.12516454	0.119522234	-0.1131973	0.106263992	-0.09880316	0.023320653	

Figura 4. Matriz de términos de Fourier.

Comparación de gráficas obtenidas (figuras 5.1, 5.2 y 5.3).

Comparación Original/Serie de Fourier

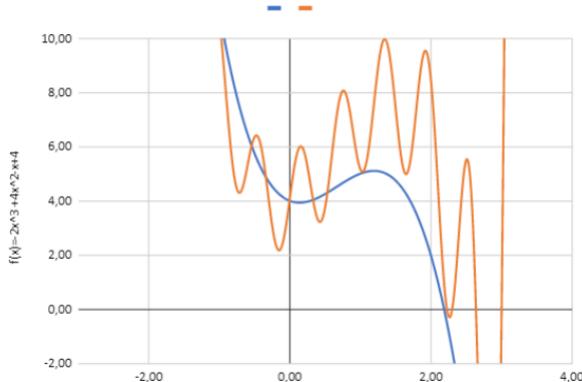


Figura 5.1 Gráfica de la serie 1.

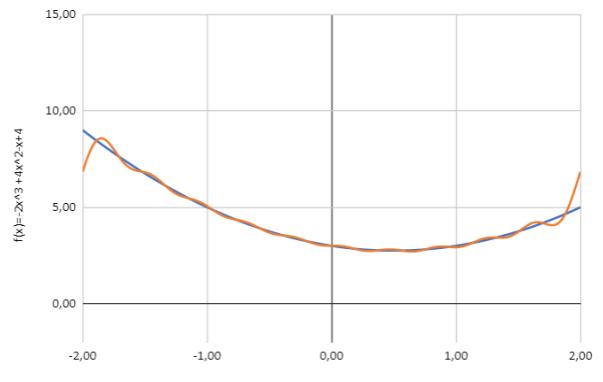


Figura 5.2 Gráfica de la serie 2.



Figura 5.3 Gráfica de la serie 3.

Desarrollo de la Fase 3

Primeramente se van a generar los archivos para cada uno de los procesos.

Para ello se va a abrir una ventana de la terminal dentro de la carpeta de la práctica (figura 5).

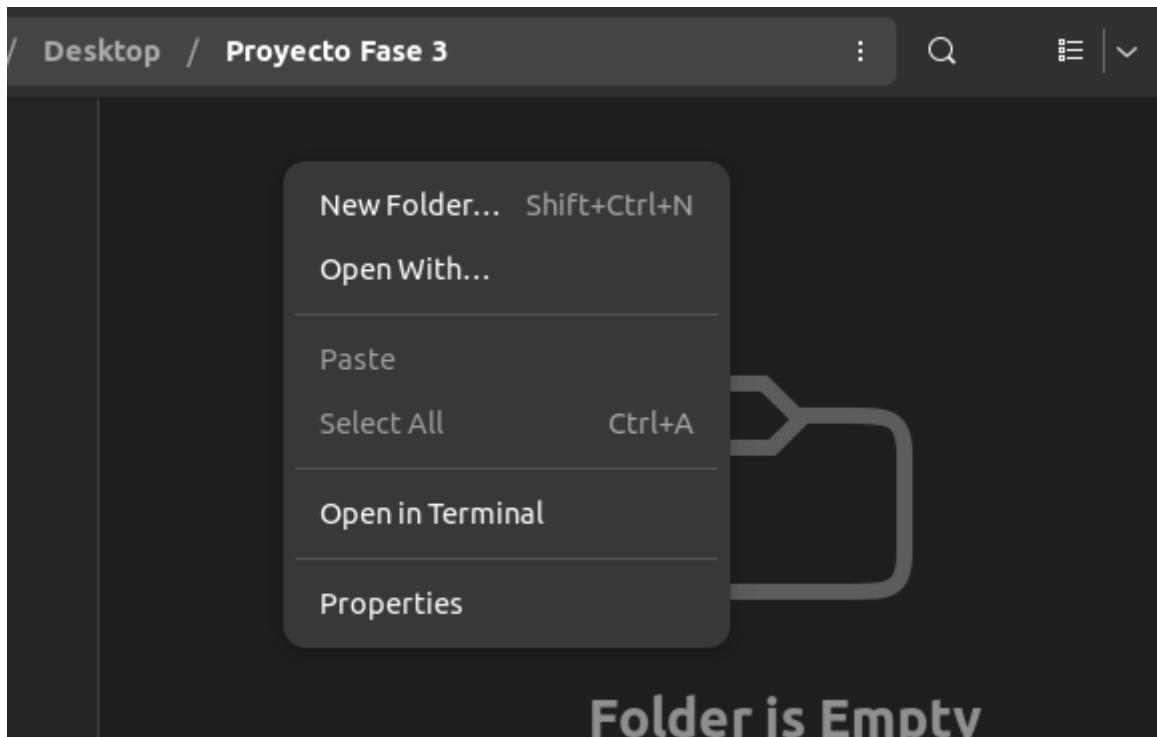


Figura 5. Vista de la carpeta Proyecto Fase 3.

Dentro de la terminal se van a crear los archivos correspondientes para cada uno de los procesos que van a realizar la suma de los elementos de cada fila a través del comando touch (figura 6).

```
~/Desktop/Proyecto Fase 3$ touch programaA.c
~/Desktop/Proyecto Fase 3$ touch programaB.c
~/Desktop/Proyecto Fase 3$ touch parametrosFourier.h
~/Desktop/Proyecto Fase 3$ touch gestionarSemaforos.h
~/Desktop/Proyecto Fase 3$ █
```

Figura 6. Creación de los archivos para cada uno de los programas.

Una vez ejecutados los comandos se verifica que se hayan creado todos los archivos (figura 7).

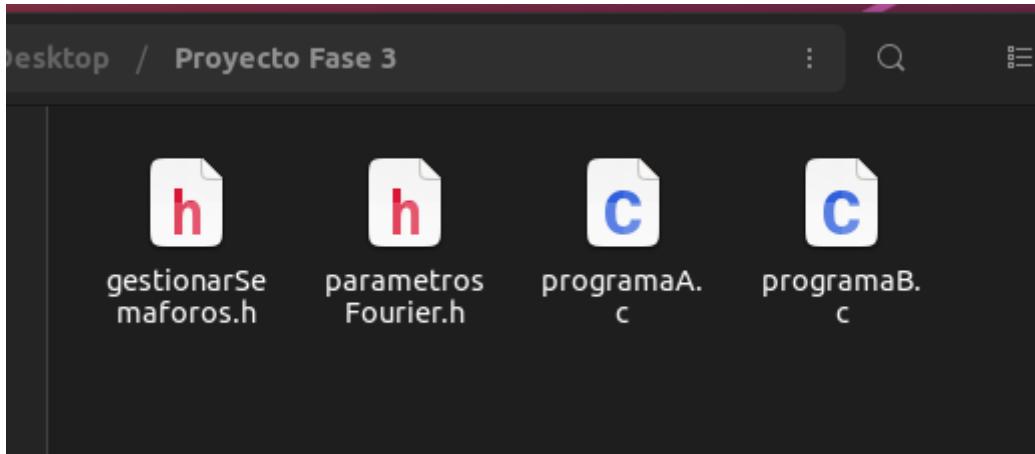


Figura 7. Ventana de la carpeta Proyecto Fase 3 con los archivos creados.

Una vez creados los archivos, se van a programar cada uno de los procesos correspondientes.

- Archivos de cabecera.

Para los programas 1 y 2 se van a crear e importar dos archivo de cabecera (extensión .h) que van a contener las funciones, estructuras y variables que les van a permitir hacer uso de los semáforos y realizar los cálculos de los términos de fourier (figura 8).

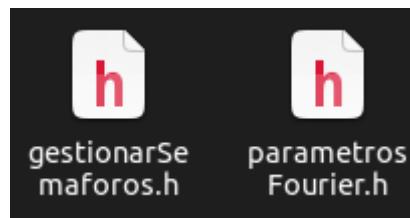


Figura 8. Archivos de cabecera.

El primer archivo de cabecera es el de “gestionarSemaforos” y este archivo va a contener las funciones y variables necesarias para utilizar este recurso fundamental en el cómputo paralelo.

Dentro de este archivo de cabecera se va a utilizar la biblioteca sem.h para el uso de los semáforos (figura 9).

```
#include <stdio.h>
#include <stdlib.h>
#include<sys/sem.h>
```

Figura 9. Código de las bibliotecas a utilizar dentro del archivo gestionarSemaforos.h

Se define un valor constante para los permisos que se van a otorgar a las llaves de cada semáforo (figura 10).

```
#define PERMISOS 0644
```

Figura 10. Código de la constante PERMISOS.

Se define la función para generar los semáforos, la cual recibe como parámetros la llave del semáforo correspondiente y un valor inicial para dicho semáforo. Dentro de dicha función se va a hacer uso de la función “semget” para obtener una ID del semáforo a generar y posteriormente se evalúa si dicha ID fue generada correctamente. Posteriormente se va a inicializar dicho semáforo con el valor inicial dado a través de la función semctl. Finalmente se retorna la ID del semáforo creado (figura 11).

```
int Crea_semaforo(key_t llave,int valor_inicial)
{
    int semid=semget(llave,1,IPC_CREAT|PERMISOS);
    if(semid==-1)
    {
        return -1;
    }
    semctl(semid,0,SETVAL,valor_inicial);
    return semid;
}
```

Figura 11. Código de la función que crea el semáforo.

Posteriormente se define la función que va a establecer un estado bajo para el semáforo que se le indique por medio de la ID, la cual se va a pasar como argumento para dicha función esto a través de la función “semop” que aplica una operación con el semáforo (figura 12).

```
void down(int semid)
{
    struct sembuf op_p[]={0,-1,0};
    semop(semid,op_p,1);
}
```

Figura 12. Código de la función que establece un estado bajo para el semáforo.

De igual forma se define la función que va a establecer un estado alto para el semáforo indicado por su ID haciendo uso de la función “semop” (figura 13).

```

void up(int semid)
{
    struct sembuf op_v[]={0,+1,0};
    semop(semid,op_v,1);
}

```

Figura 13. Código de la función que establece un estado alto en el semáforo.

El segundo archivo de cabecera es parametrosFourier y se va a encargar de aplicar las operaciones con los coeficientes de fourier así como para almacenar los parámetros correspondientes a cada serie de Fourier propuestas en este trabajo.

Primeramente se definen las bibliotecas necesarias para este archivo, en este caso la biblioteca más importante es math.h que nos va a permitir trabajar con operadores matemáticos como la constante PI (figura 14).

```

#include <stdio.h>
#include <stdlib.h>
#include<math.h>

```

Figura 14. Código de las bibliotecas del archivo de cabecera parametrosFourier.

Posteriormente se definen las constantes que corresponden con los coeficientes de la serie de Fourier correspondiente a cada integrante del equipo (figura 15).

```

#define ao_Sergio (((4.0*M_PI*M_PI)+12.0)/3.0)
#define ao_Emilio 0.0
#define ao_Flavio 13.0/3.0

```

Figura 15. Constantes del archivo de cabecera parametrosFourier.

Posteriormente se define un tipo de dato abstracto haciendo uso de una estructura en C, con el fin de encapsular los parámetros de cada serie de Fourier en un solo tipo de dato.

Dicha estructura va a contener los apuntadores a las funciones que le van a permitir al programa inicial realizar los cálculos de cada término de Fourier así como un arreglo de 7 valores para la variable X (figura 16).

```

typedef struct
{
    float (*a_Funct)(int n);
    float (*b_Funct)(int n);
    float x[7];
}Parametros_Serie;

```

Figura 16. Estructura con los parámetros de la serie de Fourier.

Se definen las funciones que van a aplicar la multiplicación de cada término con sus coeficientes de Fourier (figura 17)..

```
float a_Sergio(int n){
    return (16.0/(n*n))*(pow(-1.0,n));
}

float b_Sergio(int n){
    return pow(-1.0,n)*((4.0*M_PI*M_PI*n*n-(24.0+(2.0*n*n)))/(pow(n,3.0)));
}

float a_Emilio(int n){
    return 0.0;
}

float b_Emilio(int n){
    return (1.0/n);
}

float a_Flavio(int n){
    return ((16.0*pow(-1.0,n))/((n*n)*M_PI*M_PI));
}

float b_Flavio(int n){
    return (((pow(-1.0,n))*4.0)/(n*M_PI));
}
```

Figura 17. Funciones del archivo de cabecera parametrosFourier para el cálculo de los términos de cada serie.

Finalmente se definen las variables de este archivo de cabecera.

La primera variable corresponde con un arreglo que contiene los 3 coeficientes “ao” de cada serie (figura 18).

La segunda variable corresponde con el tipo de dato abstracto que se definió previamente usando la estructura y va a contener un arreglo de 3 elementos con todos los parámetros indicados de acuerdo al tipo de dato (figura 18).

```
float ao_arr[3]={ao_Sergio,ao_Flavio,ao_Emilio};
Parametros_Serie series[]={

    [0]={a_Sergio, b_Sergio,{-3.0,-2.0,-1.0,0.0,1.0,2.0,3.0}}, //Serie de Sergio
    [1]={a_Flavio, b_Flavio,{-2.0*M,-1.5*M,-1.0*M,0.0,1.0*M,1.5*M,2.0*M}}, //Serie de Flavio
    [2]={a_Emilio, b_Emilio,{0.0,0.5,1.0,1.5,2.0,2.5,3.0}} //Serie de Emilio
};
```

Figura 18. Declaración e inicialización de funciones dentro del archivo de cabecera parametrosFourier.

- Programa A.

El programa A se va a encargar generar los procesos necesarios para realizar los cálculos de cada serie de Fourier así como aplicar la definición de la serie de Fourier utilizando los elementos definidos previamente en los archivos de cabecera (figura 14 - 18).

Para ello se va a incluir en las bibliotecas los archivos de cabecera que se programaron previamente (Figura 19).

```
#include "gestionarSemaforos.h"
#include "parametrosFourier.h"
```

Figura 19. Código del programa A donde se importan los archivos de cabecera.

Se codifica la función para calcular los términos de la serie de Fourier. Dicha función recibe como argumentos un arreglo tridimensional en forma de apuntador triple, el ID del semáforo, un índice k para indicar la serie actual, un arreglo con los valores de la variable X y los apuntadores a las funciones de los coeficientes an y bn (figura 20).

```
void calcularTerminos(float*** terminos, int sem, int k, float x[], float(*a)(int), float(*b)(int))
{
    pid_t pid_hijos_terminos[10];
    int i, j;
```

Figura 20. Funciona para calcular los términos de cada serie de Fourier.

Luego de esto se define un bucle for de 10 iteraciones para los 10 términos a calcular y por cada iteración se va a crear un proceso hijo que se va a encargar de realizar los cálculos por medio de la definición de la serie de Fourier haciendo uso del semáforo que recibe la función como parámetro (figura 21).

```
for(i=0;i<10;i++)
{
    pid_hijos_terminos[i] = fork(); //Llamada al sistema para crear un proceso hijo
    if(pid_hijos_terminos[i]==-1) //En caso de que no se pudo crear el proceso hijo
    {
        perror("\nError al crear el proceso hijo\n");
        exit(-1);
    }
    if(pid_hijos_terminos[i]==0) //En caso de que se logro crear el proceso hijo
    {
        int n=i+1;
        printf("\nProceso %d cuyo padre es %d, fue creado exitosamente\n",getpid(),getppid());
        //Calculo de los terminos de la serie
        for(j=0;j<7;j++)
        {
            down(sem); //Semaforo en bajo
            terminos[k][j][n-1]=(*a)(n)*cos(n*x[j])+(*b)(n)*sin(n*x[j]); //Aplicar definicion
            up(sem); //Semaforo en alto
        }
        exit(0);
    }
    else
    {
        wait(NULL); //Esperar a los hijos
    }
}
```

Figura 21. Código de la función “calcularTerminos” la cual va a generar 10 procesos hijos que van a realizar los cálculos de cada términos de la serie.

Se define la función “calcularSeries” que va a generar un proceso hijo por cada serie para realizar los cálculos correspondientes de forma simultánea. Dicha función va a recibir como parámetros un arreglo tridimensional en forma de apuntador triple y el ID del semáforo a utilizar (figura 22).

```
int calcularSeries(float*** terminos,int sem)
{
    pid_t pid[3];
    int i;
```

Figura 22. Código de la función “calcularSeries” la cual recibe como parámetros el arreglo de los términos y la ID del semáforo.

Dentro de la función se va a definir un bucle for que va a iterar 3 veces y por cada iteración va a crear un proceso hijo que se va a encargar de invocar a la función para calcular los términos de la serie correspondiente. (figura 23).

```
for(i=0;i<3;i++)
{
    pid[i]=fork();
    if(pid[i]==-1)
    {
        perror("\nError al crear el proceso hijo\n");
        exit(-1);
    }
    if(pid[i]==0)
    {
        printf("\nProceso %d cuyo padre es %d, fue creado exitosamente\n",getpid(),getppid());
        calcularTerminos(terminos,sem,i,series[i].x,series[i].a_Funct,series[i].b_Funct);
        exit(0);
    }
    else
    {
        wait(NULL);
    }
}
```

Figura 23. Código de la función calcularSeries donde se crean los procesos hijos para calcular los términos de cada serie de forma paralela.

Se define una función para liberar espacio en memoria por cada matriz de términos de cada serie (figura 24).

```

void liberarMemoria(float*** terminos)
{
    int i;
    for(i=0;i<3;i++)
    {
        free(terminos[i]);
    }
}

```

Figura 24. Funciona para liberar la memoria del arreglo de términos.

Finalmente se define la función principal.

Inicialmente se declaran las variables necesarias para la creación de la región de memoria y los semáforos así como algunos índices (figura 25).

```

int main()
{
    //Declaracion de variables
    int shmid, i,j,k, sem;
    float *datos_matriz;
    float **m1,**m2,**m3;
    float ***terminos;
    key_t llave,llave_sem;
    pid_t pid;
}

```

Figura 25. Código de la función main donde se declaran las variables necesarias.

Se inicializan las variables de las llaves para la memoria compartida y el semáforo por medio de la función “ftok” y de igual forma se inicializa el semáforo por medio de su ID que se obtiene al invocar la función Crea_semaforo del archivo de cabecera “gestionarSemaforos” (figura 26).

```

llave=ftok("Terminos",'k');

//Semaforo
llave_sem=ftok("S1",'k');
sem=Crea_semaforo(llave_sem,1);

```

Figura 26. Inicialización de las llaves y el semáforo.

Se crea la región de la memoria compartida por medio de las funciones “shmget” y “shmat” conformada por 210 valores decimales o “flotantes” la cual va a almacenar las matrices de los términos de las series de Fourier (figura 27).

```

shmid=shmget(llave,3*7*10*sizeof(datos_matriz[0]),IPC_CREAT|0777);
datos_matriz=(float*)shmat(shmid,0,0);

```

Figura 27. Creación del espacio de memoria compartida para las matrices de términos.

Se reserva el espacio en memoria necesario para almacenar los datos de las matrices de términos por medio de 3 apuntadores dobles que van a hacer referencia a cada 70 celdas dentro de la matriz para que los 210 espacios reservados queden distribuidos de manera uniforme en las matrices, es decir, se van a definir 3 matrices de 70 elementos cada una (3×70), y posteriormente se va a utilizar un apuntador triple para cada apuntador doble de tal forma que se va a crear un arreglo tridimensional (figura 28).

```

m1=(float**)malloc(7*sizeof(m1[0]));
m2=(float**)malloc(7*sizeof(m2[0]));
m3=(float**)malloc(7*sizeof(m3[0]));

for(i=0;i<7;i++)
{
    m1[i]=datos_matriz+i*10;
    m2[i]=datos_matriz+70+i*10;
    m3[i]=datos_matriz+140+i*10;
}

//Reservar memoria para el arreglo que agrupa las 3 matrices
terminos=(float***)malloc(3*sizeof(float**));
terminos[0]=m1;
terminos[1]=m2;
terminos[2]=m3;

```

Figura 28. Reservación de memoria para las matrices de términos por medio de apuntadores.

Se invoca a la función para calcular las series pasando como parámetro el triple apuntador de términos y el ID del semáforo. Posteriormente se pausa el flujo de proceso por 2 segundos por medio de la función sleep y finalmente se libera la memoria reservada (figura 29).

```

printf("\nSoy el proceso padre A con PID %d\n",getpid());
calcularSeries(terminos,sem);           //Se invoca a la función
sleep(2);
liberarMemoria(terminos);
printf("Proceso padre A finalizado\n");

```

Figura 29. Ejecución de la función para calcular las series de Fourier y liberación de memoria.

Finalmente, por medio de la función “shmdt” se van a disociar los segmentos de memoria compartida de los apuntadores y posteriormente se van a eliminar dichos segmentos de memoria compartida a través de la función “shmctl” como se muestra en la figura 30.

```
shmdt(&datos_matriz);
shmctl(shmid,IPC_RMID,0);
return 0;
```

Figura 30. Destrucción de los segmentos de memoria existentes.

- Programa B.

El programa B se va a encargar de consultar las matrices de términos para realizar la suma de los mismos y obtener el resultado de la función de la serie de Fourier en base a la figura 1.

Primeramente se van a incluir los archivos de cabecera gestionarSemaforos y parametrosFourier (figura 31).

```
#include "gestionarSemaforos.h"
#include "parametrosFourier.h"
```

Figura 31. Código del programa B donde se incluyen los archivos de cabecera.

Se define la función para calcular la suma de términos de cada serie recibiendo como parámetros el arreglo de términos, la matriz para los resultados y el ID del semáforo.

```
void calcularResultados(float ***terminos, float**resultados, int sem)
{
    int i,j,k;
    // Cálculo de resultados
}
```

Figura 32. Función calcular resultados del programa B.

Dentro de la función se va a declarar un triple bucle for anidado para recorrer el arreglo tridimensional de los términos de cada serie y se va a inicializar en cada uno el arreglo de resultados con el valor del coeficiente ao de la serie correspondiente para luego realizar la suma de los términos haciendo uso de las funciones de los semáforos definidas en el archivo de cabecera “gestionarSemáforos” como se muestra en la figura 33.

```

for(i=0;i<3;i++)
{
    printf("\n Matriz de terminos de la serie %d\n",i+1);
    for(j=0;j<7;j++)
    {
        resultados[i][j]=ao_arr[i];
        for(k=0;k<10;k++)
        {
            down(sem);
            printf("%f ",terminos[i][j][k]);
            resultados[i][j]+=terminos[i][j][k];
            up(sem);
        }
        printf("\n");
    }
    printf("\n\n");
}

```

Figura 33. Función calcular resultados donde se hace la suma de términos de Fourier.

Se define una función para liberar la memoria de los arreglos términos y resultados (figura 34).

```

void liberarMemoria(float*** terminos, float** resultados)
{
int i;
for(i=0;i<3;i++)
{
    free(terminos[i]);
}
free(resultados);
}

```

Figura 34. Función para liberar el espacio en memoria de los apuntadores.

Finalmente se define la función main.

Primeramente se declaran las variables necesarias para utilizar la memoria compartida y el semáforo (figura 35).

```

int main() {
    //Declaracion de variables
    int shmid,i,j,sem;
    float *datos_matriz;
    float **resultados,**m1,**m2,**m3;
    float ***terminos;
    key_t llave, llave_sem;

```

Figura 35. Código de la función main donde se declaran las variables principales.

Se inicializan las variables de las llaves para la memoria compartida y el semáforo por medio de la función “ftok” y de igual forma se inicializa el semáforo por medio de su ID que se obtiene al invocar la función Crea_semaforo del archivo de cabecera “gestionarSemaforos” (figura 36).

```

llave=ftok("Terminos",'k');

//Semaforo
llave_sem=ftok("S1",'k');
sem=Crea_semaforo(llave_sem,1);

```

Figura 36. Inicialización de las llaves y el semáforo.

Se crea la región de la memoria compartida por medio de las funciones “shmget” y “shmat” conformada por 210 valores decimales o “flotantes” la cual va a almacenar las matrices de los términos de las series de Fourier (figura 37).

```

shmid=shmget(llave,3*7*10*sizeof(datos_matriz[0]),IPC_CREAT|0777);
datos_matriz=(float*)shmat(shmid,0,0);

```

Figura 37. Creación del espacio de memoria compartida para las matrices de términos.

Al igual que el proceso A, se reserva el espacio en memoria necesario para almacenar los datos de las matrices de términos por medio de 3 apuntadores dobles que van a hacer referencia a cada 70 celdas dentro de la matriz para que los 210 espacios reservados queden distribuidos de manera uniforme en las matrices, es decir, se van a definir 3 matrices de 70 elementos cada una (3x70), y posteriormente se va a utilizar un apuntador triple para cada apuntador doble de tal forma que se va a crear un arreglo tridimensional (figura 38).

```

m1=(float**)malloc(7*sizeof(m1[0]));
m2=(float**)malloc(7*sizeof(m2[0]));
m3=(float**)malloc(7*sizeof(m3[0]));

for(i=0;i<7;i++)
{
    m1[i]=datos_matriz+i*10;
    m2[i]=datos_matriz+70+i*10;
    m3[i]=datos_matriz+140+i*10;
}

//Reservar memoria para el arreglo que agrupa las 3 matrices
terminos=(float***)malloc(3*sizeof(float**));
terminos[0]=m1;
terminos[1]=m2;
terminos[2]=m3;

```

Figura 38. Reservación de memoria para las matrices de términos por medio de apuntadores.

De igual forma se reserva memoria para los resultados (figura 39).

```

//Reservar memoria para la matriz de resultados
resultados=(float**)malloc(3*sizeof(float*));
for(i=0;i<3;i++)
{
    resultados[i]=(float*)malloc(7*sizeof(float));
}

```

Figura 39. Reservación de memoria para los resultados.

Se invoca a la función para calcular los resultados pasando como parámetros los términos, la matriz de resultados y el semáforo (figura 40).

```

printf("\nSoy el proceso padre B con PID %d\n",getpid());
//Se invoca a la función para calcular la suma de términos
sleep(2);
calcularResultados(terminos,resultados,sem);

```

Figura 40. Ejecución de la función calcularResultados

Se imprimen los resultados por medio de un bucle for anidado (figura 41).

```

printf("\nResultados de la suma de terminos:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<7;j++)
    {
        printf("%f ",resultados[i][j]);
    }
    printf("\n");
}

```

Figura 41. Visualización de los resultados.

Finalmente, se libera la memoria y por medio de la función “shmdt” se van a disociar los segmentos de memoria compartida de los apuntadores y posteriormente se van a eliminar dichos segmentos de memoria compartida a través de la función “shmctl” como se muestra en la figura 42.

```

liberarMemoria(terminos,resultados);
shmdt(&datos_matriz);
shmctl(shmid,IPC_RMID,0);
return 0;

```

Figura 42. Destrucción de los segmentos de memoria existentes.

- Resultados

Para la ejecución de los programas se van a abrir 2 ventanas de la terminal (una para programa) dentro de la carpeta del Proyecto Fase 3. Una vez creadas las ventanas de la terminal se va a compilar cada archivo por separado como se muestra en la figura 43.

```

~/Desktop/Proyecto$ gcc -o pA programaA.c -lm
~/Desktop/Proyecto$ 

sergio@Ubuntu: ~/Desktop/Proyecto

~/Desktop/Proyecto$ gcc -o pB programaB.c -lm
~/Desktop/Proyecto$ 

```

Figura 43. Compilación de los programas en cada ventana de la terminal.

Se ejecutan los programas por separado como se muestra en la figura 44.

```

sergio@Ubuntu:~/Desktop/Proyecto$ gcc -o pA programaA.c -lm
sergio@Ubuntu:~/Desktop/Proyecto$ ./pA
Soy el proceso padre A con PID 9751
Proceso 9752 cuyo parente es 9751, fue creado exitosamente
Proceso 9753 cuyo parente es 9752, fue creado exitosamente
Proceso 9754 cuyo parente es 9752, fue creado exitosamente
Proceso 9755 cuyo parente es 9752, fue creado exitosamente
Proceso 9756 cuyo parente es 9752, fue creado exitosamente
Proceso 9757 cuyo parente es 9752, fue creado exitosamente
Proceso 9758 cuyo parente es 9752, fue creado exitosamente
Proceso 9759 cuyo parente es 9752, fue creado exitosamente
Proceso 9760 cuyo parente es 9752, fue creado exitosamente
Proceso 9761 cuyo parente es 9752, fue creado exitosamente
Proceso 9762 cuyo parente es 9752, fue creado exitosamente
Proceso 9763 cuyo parente es 9751, fue creado exitosamente
Proceso 9764 cuyo parente es 9763, fue creado exitosamente
Proceso 9765 cuyo parente es 9763, fue creado exitosamente
sergio@Ubuntu:~/Desktop/Proyecto$ ./pB
Soy el proceso padre B con PID 9785
Matriz de terminos de la serie 1
17.741955 8.238461 6.401976 5.670115 5.235697 4.901000 4.599813 4.306051 4.008827 3.703954
18.914240 9.296898 -4.949284 -9.044386 -3.436352 3.667078 5.189808 1.095860 -3.233021 -3.334370
2.696860 -15.976210 3.397532 6.153495 -7.185224 2.141038 3.225403 -4.624947 1.882581 1.891597
-16.000000 4.000000 -1.777778 1.000000 -0.640000 0.444444 -0.326531 0.250000 -0.197531 0.160000
-19.986534 12.647035 0.122442 -7.468783 6.822136 -1.287553 -3.717747 4.552197 -1.522628 -2.160100
5.597541 -14.526048 1.535345 8.753386 4.510364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.65459

```



```

Matriz de terminos de la serie 2
1.621139 0.405285 0.180127 0.101321 0.064845 0.045032 0.033084 0.025330 0.020014 0.016211
2.046635 0.636620 0.172737 -0.101321 -0.225916 -0.212207 -0.105223 0.025330 0.114187 0.127324
1.273240 -0.405285 -0.424413 0.101321 0.254648 -0.045032 -0.181891 0.025330 0.141471 -0.016211
-1.621139 0.405285 -0.180127 0.101321 -0.064846 0.045032 -0.033084 0.025330 -0.020014 0.016211
-1.273239 -0.405285 0.424413 0.101321 -0.254648 -0.045032 0.181891 0.025330 -0.141471 -0.016211
0.246002 -0.636620 -0.427474 -0.101321 0.134211 0.212207 0.152011 0.025330 -0.085883 -0.127324
1.621139 0.405285 0.180127 0.101321 0.064846 0.045032 0.033085 0.025330 0.020014 0.016212

```



```

Matriz de terminos de la serie 3
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.479426 0.420735 0.332498 0.227324 0.119694 0.023520 -0.050112 -0.094600 -0.108614 -0.095892
0.841471 0.454649 0.047040 -0.189201 -0.191785 -0.046569 0.093855 0.123670 0.045791 -0.054402
0.997495 0.070560 -0.325843 -0.069854 0.187600 0.068686 -0.125671 -0.067072 0.089309 0.065029
0.909297 -0.378401 -0.093139 0.247340 -0.108804 -0.089429 0.141515 -0.035988 -0.083443 0.091295
0.598472 -0.479462 0.312667 -0.136005 -0.013264 0.108381 -0.139375 0.114118 -0.054131 -0.013235

```

Figura 44. Ejecución de los programas de forma simultánea.

Se termina la ejecución de ambos programas mostrando su resultado (figura 45).

```

Proceso 9752 cuyo parente es 9751, fue creado exitosamente
Proceso 9770 cuyo parente es 9763, fue creado exitosamente
Proceso 9771 cuyo parente es 9763, fue creado exitosamente
Proceso 9772 cuyo parente es 9763, fue creado exitosamente
Proceso 9773 cuyo parente es 9763, fue creado exitosamente
Proceso 9774 cuyo parente es 9751, fue creado exitosamente
Proceso 9775 cuyo parente es 9774, fue creado exitosamente
Proceso 9776 cuyo parente es 9774, fue creado exitosamente
Proceso 9777 cuyo parente es 9774, fue creado exitosamente
Proceso 9778 cuyo parente es 9774, fue creado exitosamente
Proceso 9779 cuyo parente es 9774, fue creado exitosamente
Proceso 9780 cuyo parente es 9774, fue creado exitosamente
Proceso 9781 cuyo parente es 9774, fue creado exitosamente
Proceso 9782 cuyo parente es 9774, fue creado exitosamente
Proceso 9783 cuyo parente es 9774, fue creado exitosamente
Proceso 9784 cuyo parente es 9774, fue creado exitosamente
Proceso padre A finalizado
sergio@Ubuntu:~/Desktop/Proyecto$ 

-5.597541 -14.526048 1.535345 8.753386 4.510364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.65459

Matriz de terminos de la serie 2
1.621139 0.405285 0.180127 0.101321 0.064845 0.045032 0.033084 0.025330 0.020014 0.016211
2.046635 0.636620 0.172737 -0.101321 -0.225916 -0.212207 -0.105223 0.025330 0.114187 0.127324
1.273240 -0.405285 -0.424413 0.101321 0.254648 -0.045032 -0.181891 0.025330 0.141471 -0.016211
-1.621139 0.405285 -0.180127 0.101321 -0.064846 0.045032 -0.033084 0.025330 -0.020014 0.016211
-1.273239 -0.405285 0.424413 0.101321 -0.254648 -0.045032 0.181891 0.025330 -0.141471 -0.016211
0.246002 -0.636620 -0.427474 -0.101321 0.134211 0.212207 0.152011 0.025330 -0.085883 -0.127324
1.621139 0.405285 0.180127 0.101321 0.064846 0.045032 0.033085 0.025330 0.020014 0.016212

Matriz de terminos de la serie 3
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.479426 0.420735 0.332498 0.227324 0.119694 0.023520 -0.050112 -0.094600 -0.108614 -0.095892
0.841471 0.454649 0.047040 -0.189201 -0.191785 -0.046569 0.093855 0.123670 0.045791 -0.054402
0.997495 0.070560 -0.325843 -0.069854 0.187600 0.068686 -0.125671 -0.067072 0.089309 0.065029
0.909297 -0.378401 -0.093139 0.247340 -0.108804 -0.089429 0.141515 -0.035988 -0.083443 0.091295
0.598472 -0.479462 0.312667 -0.136005 -0.013264 0.108381 -0.139375 0.114118 -0.054131 -0.013235
0.141120 -0.139708 0.137373 -0.134143 0.130058 -0.125165 0.119522 -0.113197 0.106264 -0.098803

Resultados de la suma de terminos:
81.967331 31.325947 10.761600 4.072079 5.167937 8.501312 -1.066057
6.845722 6.811500 5.056510 3.007303 2.930403 3.724472 6.845724
0.000000 1.253979 1.124519 0.890240 0.600243 0.298166 0.023321
sergio@Ubuntu:~/Desktop/Proyecto$ 

```

Figura 45. Resultados de la ejecución.

Discusiones de los resultados:

Como se puede observar en la ejecución de los programas, se generaron múltiples procesos hijos por cada proceso padre, como el caso del proceso padre A con el PID 9751 que fue el proceso inicial que generó los 3 procesos hijos para realizar los cálculos de las series como el

caso de los procesos 9752, 9763 y 9774 cuyo padre es el 9751. Dichos procesos hijos se encargaron de generar otros 10 procesos hijos cada uno para realizar los cálculos de cada término de la serie y de ahí resultaron en 33 procesos diferentes más el proceso padre A, y en general no se presentó ningún problema en la creación y ejecución de estos 33 procesos hijos.

En cuanto al proceso B tampoco presentó algún inconveniente al acceder a las regiones de memoria compartida ni al realizar los cálculos pertinentes, por lo que la comunicación entre los procesos se realizó de forma adecuada y sincrónica gracias al uso de semáforos.

Finalmente con respecto a los resultados los cálculos obtenidos por el proceso B resultaron concordar con los obtenidos en la fase 2 de este trabajo como se muestra en la figura 46, 47 y 48.

Sergio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-3	17.74195435	8.238460015	6.401975821	5.670115088	5.235696846	4.901000007	4.59981307	4.306051269	4.008827074	3.703953654	81.96731973	
-2	18.91423983	9.296898012	-4.94928364	-9.04438607	-3.43635224	3.66707764	5.189808358	1.095859753	-3.2330217	-3.33437052	31.32594195	
-1	2.696860441	-15.9762094	3.397531518	6.153495434	-7.18522322	2.141037971	3.225403371	-4.62494653	1.882580738	1.891597087	10.76159993	
0	-16	4	-1.7777778	1	-0.64	0.444444444	-0.32653061	0.25	-0.19753086	0.16	4.07207725	
1	-19.9865342	12.64703472	0.122441803	-7.46078268	6.822135624	-1.28755327	-3.7177477	4.552196515	-1.52262804	-2.16009998	5.167935299	
2	-5.59754106	-14.526047	1.535344845	8.753386006	4.510363801	-2.91698523	-5.27910613	-1.57468949	2.972155838	3.46495678	8.501310905	
3	13.93780554	-0.55709772	-3.16240156	-3.98240717	-4.26329632	-4.31405182	-4.24211233	-4.09396177	-3.89341421	-3.65459319	-1.06605801	

Flavio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	
-1.5	2.046634653	0.636619772	0.172736735	-0.10132118	-0.225916	-0.21220659	-0.10522236	0.025330296	0.114187224	0.127323954	6.811499832	
-1	1.273239545	-0.40528473	-0.42441318	0.101321184	0.254647909	-0.04503164	-0.18189136	0.025330296	0.141471061	-0.01621139	5.056511021	
0	-1.62113894	0.405284735	-0.18012655	0.101321184	-0.064845556	0.045031637	-0.03308447	0.025330296	-0.02001406	0.016211389	3.007303	
1	-1.27323954	-0.40528473	0.424413182	0.101321184	-0.25464791	-0.04503164	0.181891364	0.025330296	-0.14147106	-0.01621139	2.930403083	
1.5	0.24600202	-0.63661977	-0.42747414	-0.10132118	0.13421053	0.212206591	0.152010868	0.025330296	-0.08588307	-0.12732395	3.724471517	
2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	

Emilio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
0	0	0	0	0	0	0	0	0	0	0	0	
0.5	0.479425539	0.420735492	0.332498329	0.227324357	0.119694429	0.023520001	-0.05011189	-0.09460031	-0.10861446	-0.09589243	1.25397906	
1	0.841470985	0.454648713	0.047040003	-0.18920062	-0.19178485	-0.04656925	0.093855228	0.123669781	0.045790943	-0.05440211	1.124518813	
1.5	0.997494987	0.070560004	-0.32584337	-0.06985387	0.187599995	0.068686414	-0.12567082	-0.06707161	0.089309381	0.065028784	0.89023988	
2	0.909297427	-0.37840125	-0.0931385	0.247339562	-0.10880422	-0.08942882	0.141515337	-0.03598791	-0.08344303	0.091294525	0.600243119	
2.5	0.598472144	-0.47946214	0.312666659	-0.13600528	-0.01326438	0.108381307	-0.13937514	0.114118156	-0.0541305	-0.01323518	0.298165652	
3	0.141120008	-0.13970775	0.137372828	-0.13414323	0.130057568	-0.12516454	0.119522234	-0.1131973	0.106263992	-0.09880316	0.023320653	

Figura 46. Matriz de términos de la fase 2.

```

Matriz de terminos de la serie 1
17.741955 8.238461 6.401976 5.670115 5.235697 4.901000 4.599813 4.306051 4.008827 3.703954
18.914240 9.296898 -4.949284 -9.044386 -3.436352 3.667078 5.189808 1.095860 -3.233021 -3.334370
2.696860 -15.976210 3.397532 6.153495 -7.185224 2.141038 3.225403 -4.624947 1.882581 1.891597
-16.000000 4.000000 -1.777778 1.000000 -0.640000 0.444444 -0.326531 0.250000 -0.197531 0.160000
-19.986534 12.647035 0.122442 -7.460783 6.822136 -1.287553 -3.717747 4.552197 -1.522628 -2.160100
-5.597541 -14.526048 1.535345 8.753386 4.510364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.654593

Matriz de terminos de la serie 2
1.621139 0.405285 0.180127 0.101321 0.064845 0.045032 0.033084 0.025330 0.020014 0.016211
2.046635 0.636620 0.172737 -0.101321 -0.225916 -0.212207 -0.105223 0.025330 0.114187 0.127324
1.273240 -0.405285 -0.424413 0.101321 0.254648 -0.045032 -0.181891 0.025330 0.141471 -0.016211
-1.621139 0.405285 -0.180127 0.101321 -0.064846 0.045032 -0.033084 0.025330 -0.020014 0.016211
-1.273239 -0.405285 0.424413 0.101321 -0.254648 -0.045032 0.181891 0.025330 -0.141471 -0.016211
0.246002 -0.636620 -0.427474 -0.101321 0.134211 0.212207 0.152011 0.025330 -0.085883 -0.127324
1.621139 0.405285 0.180127 0.101321 0.064846 0.045032 0.033085 0.025330 0.020014 0.016212

Matriz de terminos de la serie 3
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.479426 0.420735 0.332498 0.227324 0.119694 0.023520 -0.050112 -0.094600 -0.108614 -0.095892
0.841471 0.454649 0.047040 -0.189201 -0.191785 -0.046569 0.093855 0.123670 0.045791 -0.054402
0.997495 0.070560 -0.325843 -0.069854 0.187600 0.068686 -0.125671 -0.067072 0.089309 0.065029
0.909297 -0.378401 -0.093139 0.247340 -0.108804 -0.089429 0.141515 -0.035988 -0.083443 0.091295
0.598472 -0.479462 0.312667 -0.136005 -0.013264 0.108381 -0.139375 0.114118 -0.054131 -0.013235
0.141120 -0.139708 0.137373 -0.134143 0.130058 -0.125165 0.119522 -0.113197 0.106264 -0.098803

```

Figura 47. Matriz de términos de los resultados actuales.

Resultados de la suma de términos:

```

81.967331 31.325947 10.761600 4.072079 5.167937 8.501312 -1.066057
6.845722 6.811500 5.056510 3.007303 2.930403 3.724472 6.845724
0.000000 1.253979 1.124519 0.890240 0.600243 0.298166 0.023321

```

Figura 46. Suma de términos de cada serie los cuales coinciden con los valores obtenidos en la columna Suma de la figura 46.

Desarrollo de la Fase 4

Primeramente se van a generar los archivos para cada uno de los programas.

Para ello se va a abrir una ventana de la terminal dentro de la carpeta de la práctica (figura 47).

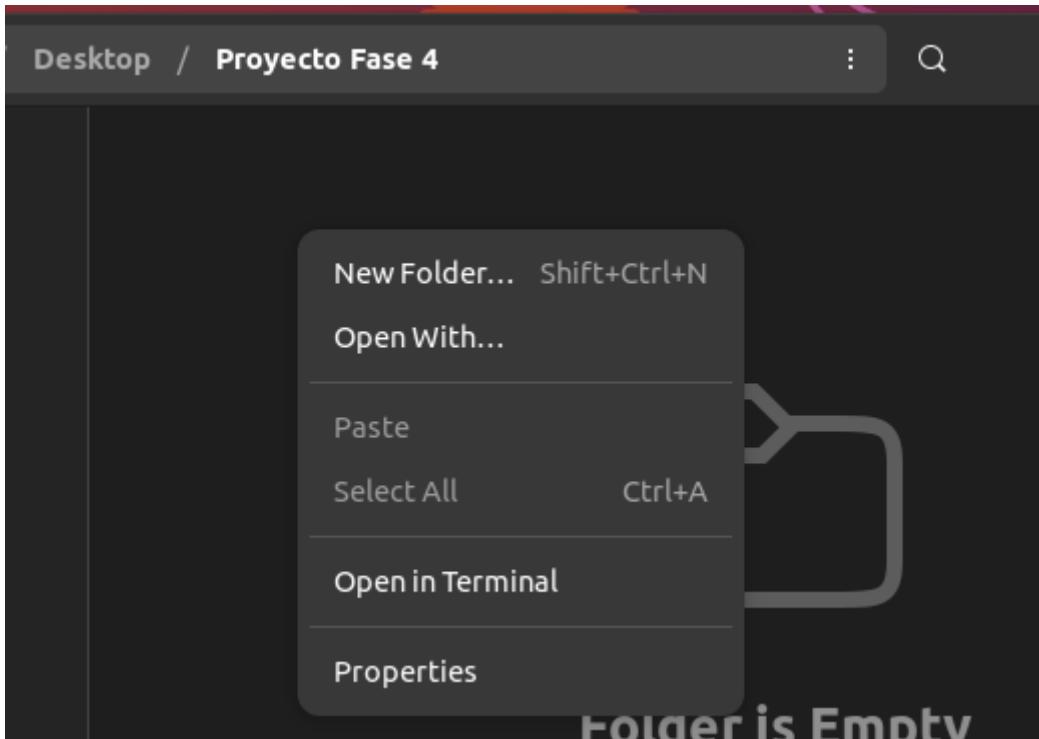


Figura 47. Vista de la carpeta Proyecto Fase 4.

Dentro de la terminal se van a crear los archivos correspondientes para cada uno de los procesos que van a realizar la suma de los elementos de cada fila a través del comando touch (figura 48).

```
~/Desktop/Proyecto Fase 3$ touch programaA.c
~/Desktop/Proyecto Fase 3$ touch programaB.c
~/Desktop/Proyecto Fase 3$ touch parametrosFourier.h
~/Desktop/Proyecto Fase 3$ touch gestionarSemaforos.h
~/Desktop/Proyecto Fase 3$ █
```

Figura 48. Creación de los archivos para cada uno de los programas.

Una vez ejecutados los comandos se verifica que se hayan creado todos los archivos (figura 49).

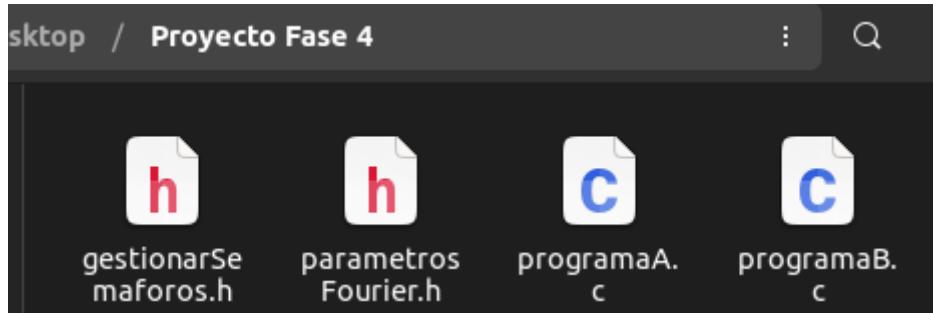


Figura 49. Ventana de la carpeta Proyecto Fase 4 con los archivos creados.

Una vez creados los archivos, se van a programar cada uno de los procesos correspondientes.

- Archivos de cabecera.

Para los programas A y B se van a crear e importar dos archivo de cabecera (extensión .h) que van a contener las funciones, estructuras y variables que les van a permitir hacer uso de los semáforos y realizar los cálculos de los términos de fourier (figura 50).

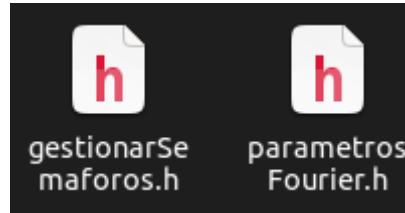


Figura 50. Archivos de cabecera.

El primer archivo de cabecera es el de “gestionarSemaforos” y este archivo va a contener las funciones y variables necesarias para utilizar este recurso fundamental en el cómputo paralelo.

Dentro de este archivo de cabecera se va a utilizar la biblioteca sem.h para el uso de los semáforos (figura 51).

```
#include <stdio.h>
#include <stdlib.h>
#include<sys/sem.h>
```

Figura 51. Código de las bibliotecas a utilizar dentro del archivo gestionarSemaforos.h

Se define un valor constante para los permisos que se van a otorgar a las llaves de cada semáforo (figura 52).

```
#define PERMISOS 0644
```

Figura 52. Valor de la constante PERMISOS.

Se define la función para generar los semáforos, la cual recibe como parámetros la llave del semáforo correspondiente y un valor inicial para dicho semáforo. Dentro de dicha función se va a hacer uso de la función “semget” para obtener una ID del semáforo a generar y posteriormente se evalúa si dicha ID fue generada correctamente. Posteriormente se va a inicializar dicho semáforo con el valor inicial dado a través de la función semctl. Finalmente se retorna la ID del semáforo creado (figura 53).

```
int Crea_semaforo(key_t llave,int valor_inicial)
{
    int semid=semget(llave,1,IPC_CREAT|PERMISOS);
    if(semid== -1)
    {
        return -1;
    }
    semctl(semid,0,SETVAL,valor_inicial);
    return semid;
}
```

Figura 53. Código de la función que crea el semáforo.

Posteriormente se define la función que va a establecer un estado bajo para el semáforo que se le indique por medio de la ID, la cual se va a pasar como argumento para dicha función esto a través de la función “semop” que aplica una operación con el semáforo (figura 54).

```
void down(int semid)
{
    struct sembuf op_p[]={0,-1,0};
    semop(semid,op_p,1);
}
```

Figura 54. Código de la función que establece un estado bajo para el semáforo.

De igual forma se define la función que va a establecer un estado alto para el semáforo indicado por su ID haciendo uso de la función “semop” (figura 55).

```
void up(int semid)
{
    struct sembuf op_v[]={0,+1,0};
    semop(semid,op_v,1);
}
```

Figura 55. Código de la función que establece un estado alto en el semáforo.

El segundo archivo de cabecera es parametrosFourier y se va a encargar de aplicar las operaciones con los coeficientes de fourier así como para almacenar los parámetros correspondientes a cada serie de Fourier propuestas en este trabajo.

Primeramente se definen las bibliotecas necesarias para este archivo, en este caso la biblioteca más importante es math.h que nos va a permitir trabajar con operadores matemáticos como la constante PI (figura 56).

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
```

Figura 56. Código de las bibliotecas del archivo de cabecera parametrosFourier.

Posteriormente se definen las constantes que corresponden con los coeficientes de la serie de Fourier correspondiente a cada integrante del equipo (figura 57).

```
#define ao_Sergio (((4.0*M_PI*M_PI)+12.0)/3.0)
#define ao_Emilio 0.0
#define ao_Flavio 13.0/3.0
```

Figura 57. Constantes del archivo de cabecera parametrosFourier.

Posteriormente se define un tipo de dato abstracto haciendo uso de una estructura en C, con el fin de encapsular los parámetros de cada serie de Fourier en un solo tipo de dato.

Dicha estructura va a contener los apuntadores a las funciones que le van a permitir al programa inicial realizar los cálculos de cada término de Fourier así como un arreglo de 7 valores para la variable X y el índice de la serie (figura 58).

```
typedef struct
{
    float (*a_Funct)(int n);
    float (*b_Funct)(int n);
    float x[7];
    int k;
}ParametrosSerie;
```

Figura 58. Estructura con los parámetros de la serie de Fourier.

De igual forma se define una estructura que contendrá los parámetros para el uso de la memoria compartida y los semáforos (figura 59).

```

typedef struct
{
    float*** terminos;
    float** resultados;
    int sem;
}ParametrosMemoria;

```

Figura 59. Estructura con los parámetros de la memoria compartida.

Se define una tercera estructura donde se almacenen las dos estructuras anteriores y agruparlas en una sola (figura 60).

```

typedef struct
{
    ParametrosSerie serie[3];
    ParametrosMemoria memoria;
}ParametrosHilo;

```

Figura 60. Estructura con los struct definidos previamente.

Se definen las funciones que van a aplicar la multiplicación de cada término con sus coeficientes de Fourier (figura 61).

```

float a_Sergio(int n){
    return (16.0/(n*n))*(pow(-1.0,n));
}

float b_Sergio(int n){
    return pow(-1.0,n)*((4.0*M_PI*M_PI*n*n-(24.0+(2.0*n*n)))/(pow(n,3.0)));
}

float a_Emilio(int n){
    return 0.0;
}

float b_Emilio(int n){
    return (1.0/n);
}

float a_Flavio(int n){
    return ((16.0*pow(-1.0,n))/((n*n)*M_PI*M_PI));
}

float b_Flavio(int n){
    return (((pow(-1.0,n))*4.0)/(n*M_PI));
}

```

Figura 61. Funciones del archivo de cabecera parametrosFourier para el cálculo de los términos de cada serie.

Finalmente se inicializan los valores del estruct Parámetros.

En este caso se van a inicializar únicamente los parámetros de casa serie de Fourier por medio del operador de inicialización de cada estructura (figura 62).

```
ParametrosHilo parametrosHilo={  
    .serie={  
        {a_Sergio, b_Sergio,{-3.0,-2.0,-1.0,0.0,1.0,2.0,3.0},0}, //Serie de Sergio  
        {a_Flavio, b_Flavio,{-2.0*M,-1.5*M,-1.0*M,0.0,1.0*M,1.5*M,2.0*M},1}, //Serie de Flavio  
        {a_Emilio, b_Emilio,{0.0,0.5,1.0,1.5,2.0,2.5,3.0},2} //Serie de Emilio  
    }, //Parametros Series  
    .memoria={NULL,NULL,0} //Parametros Memoria Compartida  
};
```

Figura 62. Declaración e inicialización de funciones dentro del archivo de cabecera parametrosFourier.

- Programa A.

El programa A se va a encargar generar los procesos necesarios para realizar los cálculos de cada serie de Fourier así como aplicar la definición de la serie de Fourier utilizando los elementos definidos previamente en los archivos de cabecera.

Para ello se va a incluir en las bibliotecas los archivos de cabecera que se programaron previamente (Figura 63).

```
#include "gestionarSemaforos.h"  
#include "parametrosFourier.h"
```

Figura 63. Código del programa A donde se importan los archivos de cabecera.

Se define la función de cada hilo que se va a ejecutar.

En dicha función se va a recibir como argumento el TDA que corresponde con la estructura “Parámetros” definida previamente en el archivo de cabecera.

Posteriormente se inicializan las variables a utilizar en esta función tanto para los valores de las series como los apunadores a la memoria compartida y el semáforo (figura 64).

```

void HiloCalcularTerminos(ParametrosSerie *parametros)
{
    int i,j,k=parametros->k;
    float (*a)(int n)=parametros->a_Funct;
    float (*b)(int n)=parametros->b_Funct;
    float *x=parametros->x;

    float*** terminos=parametrosHilo.memoria.terminos;
    int sem=parametrosHilo.memoria.sem;
}

```

Figura 64. Función de cada hilo donde se declaran los valores de cada serie y de memoria compartida.

En cada hilo se va a recorrer la matriz de términos y haciendo uso de semáforos se va a acceder a cada elemento de la matriz para aplicar la definición de la serie de Fourier realizando la suma de productos entre los coeficientes y las funciones seno y coseno (figura 65).

Posteriormente se termina de ejecutar el hilo del proceso correspondiente.

```

for(i=0;i<10;i++)
{
    int n=i+1;
    //Calculo de los terminos de la serie
    for(j=0;j<7;j++)
    {
        down(sem); //Semaforo en bajo
        terminos[k][j][n-1]=(*a)(n)*cos(n*x[j])+(*b)(n)*sin(n*x[j]);
        up(sem); //Semaforo en alto
    }
}

pthread_exit(0); //Terminar hilo

```

Figura 65. Código de la función del hilo donde se calculan y almacenan los términos de cada serie en su espacio de memoria compartida correspondiente.

Se codifica la función para calcular los términos de la serie de Fourier.

Dicha función define un arreglo de identificadores para cada uno de los 3 hilos y las variables para los atributos de cada hilo. Para posteriormente inicializar dichos atributos y disociar cada hilo de su proceso padre (figura 66).

```

int calcularSeries()
{
    pthread_attr_t atributos;
    pthread_t identificadores[3];      //Arreglo de 3 hilos
    int i;

    pthread_attr_init(&atributos);
    pthread_attr_setdetachstate(&atributos, PTHREAD_CREATE_DETACHED);
}

```

Figura 66. Funciona para calcular los términos de cada serie de Fourier.

Luego de esto se define un bucle for de 3 iteraciones para los generar los 3 hilos y por cada iteración se va a ejecutar la función pthread_create cuyos parámetros son el ID del hilo, los atributos analizados previamente, la función de cada hilo y su parámetro que en este caso corresponde con el TDA de cada serie (figura 67).

```

for(i=0;i<3;i++)
{
    //Creacion del hilo y paso de parametros
    pthread_create(&identificadores[i],&atributos,(void*)HiloCalcularTerminos,(void*)&parametrosHilo.serie[i]);
}

```

Figura 67. Bucle for de 3 iteraciones para crear los hilos de proceso en el programa A.

Se define una función para liberar espacio en memoria por cada matriz de términos de cada serie (figura 68).

```

void liberarMemoria(float*** terminos)
{
    int i;
    for(i=0;i<3;i++)
    {
        free(terminos[i]);
    }
}

```

Figura 68. Funciona para liberar la memoria del arreglo de términos.

Finalmente se define la función principal.

Inicialmente se declaran las variables necesarias para la creación de la región de memoria y los semáforos así como algunos índices (figura 69).

```

int main()
{
    //Declaracion de variables
    int shmid, i,j,k, sem;
    float *datos_matriz;
    float **m1,**m2,**m3;
    float ***terminos;
    key_t llave,llave_sem;
    pid_t pid;

```

Figura 69. Código de la función main donde se declaran las variables necesarias.

Se inicializan las variables de las llaves para la memoria compartida y el semáforo por medio de la función “ftok” y de igual forma se inicializa el semáforo por medio de su ID que se obtiene al invocar la función Crea_semaforo del archivo de cabecera “gestionarSemaforos” y se va a almacenar en el TDA del struct definido en dicho archivo (figura 70).

```

llave=ftok("Terminos",'k');

//Semaforo
llave_sem=ftok("S1",'k');
parametrosHilo.memoria.sem=Crea_semaforo(llave_sem,1);

```

Figura 70. Inicialización de las llaves y el semáforo.

Se crea la región de la memoria compartida por medio de las funciones “shmget” y “shmat” conformada por 210 valores decimales o “flotantes” la cual va a almacenar las matrices de los términos de las series de Fourier (figura 71).

```

shmid=shmget(llave,3*7*10*sizeof(datos_matriz[0]),IPC_CREAT|0777);
datos_matriz=(float*)shmat(shmid,0,0);

```

Figura 71. Creación del espacio de memoria compartida para las matrices de términos.

Se reserva el espacio en memoria necesario para almacenar los datos de las matrices de términos por medio de 3 apuntadores dobles que van a hacer referencia a cada 70 celdas dentro de la matriz para que los 210 espacios reservados queden distribuidos de manera uniforme en las matrices, es decir, se van a definir 3 matrices de 70 elementos cada una (3x70), y posteriormente se va a utilizar el apuntador triple del struct definido en el archivo de cabecera “parametrosFourier” para cada apuntador doble de tal forma que se va a crear un arreglo tridimensional (figura 72).

```

m1=(float**)malloc(7*sizeof(m1[0]));
m2=(float**)malloc(7*sizeof(m2[0]));
m3=(float**)malloc(7*sizeof(m3[0]));

for(i=0;i<7;i++)
{
    m1[i]=datos_matriz+i*10;
    m2[i]=datos_matriz+70+i*10;
    m3[i]=datos_matriz+140+i*10;
}

//Reservar memoria para el arreglo que agrupa las 3 matrices
parametrosHilo.memoria.terminos=(float***)malloc(3*sizeof(float**));
parametrosHilo.memoria.terminos[0]=m1;
parametrosHilo.memoria.terminos[1]=m2;
parametrosHilo.memoria.terminos[2]=m3;

```

Figura 72. Reservación de memoria para las matrices de términos por medio de apuntadores.

Se invoca a la función para calcular las series.

Posteriormente se pausa el flujo de proceso por 3 segundos por medio de la función sleep y finalmente se libera la memoria reservada para el arreglo de términos (figura 73).

```

printf("\nSoy el proceso A con PID %d\n",getpid());
calcularSeries();           //Se invoca a la función para calcular las series de Fourier
sleep(3);
liberarMemoria(parametrosHilo.memoria.terminos);
printf("Proceso A finalizado\n");

```

Figura 73. Ejecución de la función para calcular las series de Fourier y liberación de memoria.

Finalmente, por medio de la función “shmdt” se van a disociar los segmentos de memoria compartida de los apuntadores y posteriormente se van a eliminar dichos segmentos de memoria compartida a través de la función “shmctl” como se muestra en la figura 74.

```

shmdt(&datos_matriz);
shmctl(shmid,IPC_RMID,0);
return 0;

```

Figura 74. Destrucción de los segmentos de memoria existentes.

- Programa B.

El programa B se va a encargar de consultar las matrices de términos para realizar la suma de los mismos y obtener el resultado de la función de la serie de Fourier en base a la figura 1.

Primeramente se van a incluir los archivos de cabecera gestionarSemaforos y parametrosFourier (figura 75).

```
#include "gestionarSemaforos.h"
#include "parametrosFourier.h"
```

Figura 75. Código del programa B donde se incluyen los archivos de cabecera.

Se define la función de cada hilo la cual se va a encargar de realizar la suma de los términos de cada serie. Dicha función recibe como parámetro una variable del TDA definido en el struct del archivo de cabecera “parametrosSerie”.

Dentro de la función se va a declarar e inicializar las variables auxiliares para recorrer el arreglo de términos y resultados de cada serie y el semáforo a utilizar (figura 76).

```
void HiloSumarTerminos(ParametrosSerie* parametros)
{
    int k,j,i=parametros->k;

    float*** terminos=parametrosHilo.memoria.terminos;
    float** resultados=parametrosHilo.memoria.resultados;
    int sem=parametrosHilo.memoria.sem;
```

Figura 76. Función del hilo del programa B donde se reciben y declaran los parámetros a utilizar.

Por medio de un bucle for anidado se va a recorrer el arreglo de términos almacenado en la región de memoria compartida y haciendo uso de semáforos se va a acceder a cada elemento de dicho arreglo para realizar la suma de cada término y almacenarla en el arreglo de resultados (figura 77).

```

for(j=0;j<7;j++)
{
    resultados[i][j]=ao_arr[i];
    for(k=0;k<10;k++)
    {
        down(sem);
        printf("%f ",terminos[i][j][k]);
        resultados[i][j]+=terminos[i][j][k];
        up(sem);
    }
    printf("\n");
}
printf("\n\n");
pthread_exit(0); //Terminar hilo

```

Figura 77. Función del hilo donde se realiza la suma de cada término de la serie.

Se define la función para calcular la suma de términos, la cual va a declarar un arreglo de identificadores para cada uno de los 3 hilos a crear y la variable correspondiente para los atributos de cada hilo (figura 78).

```

pthread_attr_t atributos;
pthread_t identificadores[3]; //Arreglo de 3 hilos
int i;

pthread_attr_init(&atributos);
pthread_attr_setdetachstate(&atributos, PTHREAD_CREATE_DETACHED);

```

Figura 78. Función crear los 3 hilos del programa B.

Dentro de la función se va a declarar un bucle for para crear 3 hilos haciendo uso de la función `pthread_create` y pasando como parámetros el TDA correspondiente con la serie de Fourier definida en el archivo de cabecera “parametrosFourier” como se muestra en la figura 79.

```

for(i=0;i<3;i++)
{
    pthread_create(&identificadores[i],&atributos,(void*)HiloSumarTerminos,(void*)&parametrosHilo.serie[i]);
    sleep(1);
}

```

Figura 79. Creación de los hilos con los parámetros correspondientes.

Se define una función para liberar la memoria de los arreglos términos y resultados (figura 80).

```

void liberarMemoria(float*** terminos, float** resultados)
{
    int i;
    for(i=0;i<3;i++)
    {
        free(terminos[i]);
    }
    free(resultados);
}

```

Figura 80. Función para liberar el espacio en memoria de los apuntadores.

Finalmente se define la función main.

Primeramente se declaran las variables necesarias para utilizar la memoria compartida y el semáforo (figura 81).

```

int main() {
    //Declaracion de variables
    int shmid,i,j,sem;
    float *datos_matriz;
    float **m1,**m2,**m3;
    key_t llave, llave_sem;
}

```

Figura 81. Código de la función main donde se declaran las variables principales.

Se inicializan las variables de las llaves para la memoria compartida y el semáforo por medio de la función “ftok” y de igual forma se inicializa el semáforo por medio de su ID que se obtiene al invocar la función Crea_semaforo del archivo de cabecera “gestionarSemaforos” almacenando en el struct de parametros de memoria (figura 82).

```

llave=ftok("Terminos",'k'); |
//Semaforo
llave_sem=ftok("S1",'k');
parametrosHilo.memoria.sem=Crea_semaforo(llave_sem,1);

```

Figura 82. Inicialización de las llaves y el semáforo.

Se crea la región de la memoria compartida por medio de las funciones “shmget” y “shmat” conformada por 210 valores decimales o “flotantes” la cual va a almacenar las matrices de los términos de las series de Fourier (figura 83).

```

shmid=shmget(llave,3*7*10*sizeof(datos_matriz[0]),IPC_CREAT|0777);
datos_matriz=(float*)shmat(shmid,0,0);

```

Figura 83. Creación del espacio de memoria compartida para las matrices de términos.

Al igual que el proceso A, se reserva el espacio en memoria necesario para almacenar los datos de las matrices de términos por medio de 3 apuntadores dobles que van a hacer referencia a cada 70 celdas dentro de la matriz para que los 210 espacios reservados queden distribuidos de manera uniforme en las matrices, es decir, se van a definir 3 matrices de 70 elementos cada una (3×70), y posteriormente se va a utilizar el apuntador triple de términos para cada apuntador doble de tal forma que se va a crear un arreglo tridimensional (figura 84).

```

m1=(float**)malloc(7*sizeof(m1[0]));
m2=(float**)malloc(7*sizeof(m2[0]));
m3=(float**)malloc(7*sizeof(m3[0]));

for(i=0;i<7;i++)
{
    m1[i]=datos_matriz+i*10;
    m2[i]=datos_matriz+70+i*10;
    m3[i]=datos_matriz+140+i*10;
}

//Reservar memoria para el arreglo que agrupa las 3 matrices
parametrosHilo.memoria.terminos=(float***)malloc(3*sizeof(float**));
parametrosHilo.memoria.terminos[0]=m1;
parametrosHilo.memoria.terminos[1]=m2;
parametrosHilo.memoria.terminos[2]=m3;

```

Figura 84. Reservación de memoria para las matrices de términos por medio de apuntadores.

De igual forma se reserva memoria para los resultados con la variable resultados almacenada en el struct de memoria (figura 85).

```

parametrosHilo.memoria.resultados=(float**)malloc(3*sizeof(float*));
for(i=0;i<3;i++)
{
    parametrosHilo.memoria.resultados[i]=(float*)malloc(7*sizeof(float));
}

```

Figura 85. Reservación de memoria para los resultados.

Se invoca a la función para calcular los resultados y se pausa la ejecución por 1 segundo (figura 86).

```
CalcularResultados();
sleep(1);
```

Figura 86. Ejecución de la función calularResultados

Se imprimen los resultados por medio de un bucle for anidado (figura 87).

```
for(i=0;i<3;i++)
{
    for(j=0;j<7;j++)
    {
        printf("%f ",parametrosHilo.memoria.resultados[i][j]);
    }
    printf("\n");
}
```

Figura 87. Visualización de los resultados.

Finalmente, se libera la memoria y por medio de la función “shmdt” se van a disociar los segmentos de memoria compartida de los apuntadores y posteriormente se van a eliminar dichos segmentos de memoria compartida a través de la función “shmctl” como se muestra en la figura 88.

```
liberarMemoria(parametrosHilo.memoria.terminos, parametrosHilo.memoria.resultados);
shmdt(&datos_matriz);
shmctl(shmid,IPC_RMID,0);
return 0;
```

Figura 88. Destrucción de los segmentos de memoria existentes.

- Resultados

Para la ejecución de los programas se van a abrir 2 ventanas de la terminal (una para programa) dentro de la carpeta del Proyecto Fase 4. Una vez creadas las ventanas de la terminal se va a compilar cada archivo por separado como se muestra en la figura 89.

```

sergio@Fase 4:~/Desktop/Proyecto Fase 4$ gcc -o pA programaA.c -lm
sergio@Fase 4:~/Desktop/Proyecto Fase 4$ 

sergio@Ubuntu: ~/Desktop/Proyecto Fase 4$ gcc -o pB programaB.c -lm
sergio@Ubuntu: ~/Desktop/Proyecto Fase 4$ 

```

Figura 89. Compilación de los programas en cada ventana de la terminal.

Se ejecutan los programas por separado como se muestra en la figura 90.

```

sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ ./pA
Soy el proceso A con PID 4754
Dentro del hilo 1-1 con ID: 140255254537920 cuyo proceso es el 4754
Dentro del hilo 1-2 con ID: 140255246145216 cuyo proceso es el 4754
Dentro del hilo 1-0 con ID: 140255262930624 cuyo proceso es el 4754
Soy el proceso B con PID 4758
Dentro del hilo 2-1 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 1
17.741955 8.238461 6.401976 5.670115 5.235697 4.901000 4.599813 4.306051 4.008827 3.703954
18.914240 9.296898 -4.949284 -9.044386 -3.436352 3.667078 5.189808 1.095860 -3.233021 -3.334370
2.696860 -15.976210 3.397532 6.153495 -7.185224 2.141038 3.225403 -4.624947 1.882581 1.891597
-16.000000 4.000000 -1.777778 1.000000 -0.640000 0.444444 -0.326531 0.250000 -0.197531 0.160000
-19.986534 12.647035 0.122442 -7.460783 6.822136 -1.287553 -3.717747 4.552197 -1.522628 -2.160100
-5.597541 -14.526048 1.535345 8.753386 4.518364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.654593

sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ ./pB
Soy el proceso B con PID 4758
Dentro del hilo 2-1 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 1
17.741955 8.238461 6.401976 5.670115 5.235697 4.901000 4.599813 4.306051 4.008827 3.703954
18.914240 9.296898 -4.949284 -9.044386 -3.436352 3.667078 5.189808 1.095860 -3.233021 -3.334370
2.696860 -15.976210 3.397532 6.153495 -7.185224 2.141038 3.225403 -4.624947 1.882581 1.891597
-16.000000 4.000000 -1.777778 1.000000 -0.640000 0.444444 -0.326531 0.250000 -0.197531 0.160000
-19.986534 12.647035 0.122442 -7.460783 6.822136 -1.287553 -3.717747 4.552197 -1.522628 -2.160100
-5.597541 -14.526048 1.535345 8.753386 4.518364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.654593

```

Figura 90. Ejecución de los programas de forma simultánea.

Se termina la ejecución de ambos programas mostrando su resultado (figura 91).

```

sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ ./pA
Dentro del hilo 2-3 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 3
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.479426 0.420735 0.332498 0.227324 0.119694 0.023520 -0.050112 -0.094600 -0.108614 -0.095892
0.841471 0.454649 0.047040 -0.189201 -0.191785 -0.046569 0.093855 0.123670 0.045791 -0.054402
0.997495 0.070560 -0.325843 -0.069854 0.187600 0.068686 -0.125671 -0.067072 0.089309 0.065029
0.909297 -0.378401 -0.093139 0.247340 -0.108864 -0.089429 0.141515 -0.035988 -0.083443 0.091295
0.598472 -0.479462 0.312667 -0.136005 -0.013264 0.108381 -0.139375 0.114118 -0.054131 -0.013235
0.141120 -0.139708 0.137373 -0.134143 0.130058 -0.125165 0.119522 -0.113197 0.106264 -0.098803
Proceso A finalizado
sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ 

sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ ./pB
Resultados de la suma de terminos:
81.967331 31.325947 10.761600 4.072079 5.167937 8.501312 -1.066057
6.845722 6.811500 5.056510 3.007303 2.930403 3.724472 6.845724
0.000000 1.253979 1.124519 0.890240 0.600243 0.298166 0.023321
Proceso B finalizado
sergio@Ubuntu:~/Desktop/Proyecto Fase 4$ 

```

Figura 91. Resultados de la ejecución.

Discusiones de los resultados:

Como se puede observar en la ejecución de los programas, se generaron 3 hilos para cada proceso padre, como el caso del proceso A con el PID 4754 que fue el proceso inicial que generó 3 hilos para realizar los cálculos y registrar los términos de las series. De igual forma el proceso B con PID 4758 logró generar 3 hilos para realizar la suma de los términos de cada serie obteniendo una matriz con la suma de términos que corresponden con cada valor de X definido en el arreglo de la serie.

Finalmente con respecto a los resultados los cálculos obtenidos por el proceso B resultaron concordar con los obtenidos tanto en la fase 2 como en la fase 3 de este trabajo como se muestra en la figura 92-95.

Sergio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-3	17.74195435	8.238460015	6.401975821	5.670115088	5.235696846	4.901000007	4.59981307	4.306051269	4.008827074	3.703953654	81.96731973	
-2	18.91423983	9.296898012	-4.94928364	-9.04438607	-3.43635224	3.66707764	5.189808358	1.095859753	-3.2330217	-3.33437052	31.32594195	
-1	2.696860441	-15.9762094	3.397531518	6.153495434	-7.18523222	2.141037971	3.225403371	-4.62494653	1.882580738	1.891597087	10.76159993	
0	-16	4	-1.7777778	1	-0.64	0.44444444	-0.32653061	0.25	-0.19753086	0.16	4.072077725	
1	-19.9865342	12.64703472	0.122441803	-7.46078268	6.822135624	-1.28755327	-3.7177477	4.552196515	-1.52262804	-2.16009998	5.167935299	
2	-5.59754106	-14.526047	1.535344845	8.753386006	4.510363801	-2.91698523	-5.27910613	-1.57468949	2.972155838	3.46495678	8.501310905	
3	13.93780554	-0.55709772	-3.16240156	-3.98240717	-4.26329632	-4.31405182	-4.24211233	-4.09396177	-3.89341421	-3.65459319	-1.06605801	
Flavio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
-2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	
-1.5	2.046634653	0.636619772	0.172736735	-0.10132118	-0.225916	-0.2120659	-0.10522236	0.025330296	0.114187224	0.127323954	6.811499832	
-1	1.273239545	-0.40528473	-0.42441318	0.101321184	0.254647909	-0.04503164	-0.18189136	0.025330296	0.141471061	-0.01621139	5.056511021	
0	-1.62113894	0.405284735	-0.18012655	0.101321184	-0.06484556	0.045031637	-0.03308447	0.025330296	-0.02001406	0.016211389	3.007303	
1	-1.27323954	-0.40528473	0.424413182	0.101321184	-0.25464791	-0.04503164	0.181891364	0.025330296	-0.14147106	-0.01621139	2.930403083	
1.5	0.24600202	-0.63661977	-0.42747414	-0.10132118	0.13421053	0.212206591	0.152010868	0.025330296	-0.08588307	-0.12732395	3.724471517	
2	1.621138938	0.405284735	0.180126549	0.101321184	0.064845558	0.045031637	0.033084468	0.025330296	0.020014061	0.016211389	6.845722148	
Emilio												
x	1	2	3	4	5	6	7	8	9	10	Suma	
0	0	0	0	0	0	0	0	0	0	0	0	
0.5	0.479425539	0.420735492	0.332498329	0.227324357	0.119694429	0.023520001	-0.05011189	-0.09460031	-0.10861446	-0.09589243	1.25397906	
1	0.841470985	0.454648713	0.047040003	-0.18920062	-0.19178485	-0.0465695	0.093855228	0.123669781	0.045790943	-0.05440211	1.124518813	
1.5	0.997494987	0.070560004	-0.32584337	-0.06985387	0.187599995	0.068686414	-0.12567082	-0.06707161	0.089309381	0.065028784	0.89023988	
2	0.909297427	-0.37840125	-0.0931385	0.247339562	-0.10880422	-0.08942882	0.141515337	-0.03598791	-0.08344303	0.091294525	0.600243119	
2.5	0.598472144	-0.47946214	0.312666659	-0.13600528	-0.01326438	0.108381307	-0.13937514	0.114118156	-0.0541305	-0.01323518	0.298165652	
3	0.141120008	-0.13970775	0.137372828	-0.13414323	0.130057568	-0.12516454	0.11952234	-0.1131973	0.106263992	-0.09880316	0.023320653	

Figura 92. Matriz de términos de la fase 2.

Matriz de términos de la serie 1												
17.741955	8.238461	6.401976	5.670115	5.235697	4.901000	4.599813	4.306051	4.008827	3.703954			
18.914240	9.296898	-4.949284	-9.044386	-3.436352	3.667078	5.189808	1.095860	-3.233021	-3.334370			
2.696860	-15.976210	3.397532	6.153495	-7.185224	2.141038	3.225403	-4.624947	1.882581	1.891597			
-16.000000	4.000000	-1.777778	1.000000	-0.640000	0.444444	-0.326531	0.250000	-0.197531	0.160000			
-19.986534	12.647035	0.122442	-7.460783	6.822136	-1.287553	-3.717747	4.552197	-1.522628	-2.160100			
-5.597541	-14.526048	1.535345	8.753386	4.510364	-2.916985	-5.279106	-1.574690	2.972156	3.464957			
13.937805	-0.557098	-3.162402	-3.982407	-4.263297	-4.314052	-4.242112	-4.093962	-3.893414	-3.654593			
Matriz de términos de la serie 2												
1.621139	0.405285	0.180127	0.101321	0.064845	0.045032	0.033084	0.025330	0.020014	0.016211			
2.046635	0.636620	0.172737	-0.101321	-0.225916	-0.212207	-0.105223	0.025330	0.114187	0.127324			
1.273240	-0.405285	-0.424413	0.101321	0.254648	-0.045032	-0.181891	0.025330	0.141471	-0.016211			
-1.621139	0.405285	-0.180127	0.101321	-0.064846	0.045032	-0.033084	0.025330	-0.020014	0.016211			
-1.273239	-0.405285	0.424413	0.101321	-0.254648	-0.045032	0.181891	0.025330	-0.141471	-0.016211			
0.246002	-0.636620	-0.427474	-0.101321	0.134211	0.212207	0.152011	0.025330	-0.085883	-0.127324			
1.621139	0.405285	0.180127	0.101321	0.064846	0.045032	0.033085	0.025330	0.020014	0.016212			
Matriz de términos de la serie 3												
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000			
0.479426	0.420735	0.332498	0.227324	0.119694	0.023520	-0.050112	-0.094600	-0.108614	-0.095892			
0.841471	0.454649	0.047040	-0.189201	-0.191785	-0.046569	0.093855	0.123670	0.045791	-0.0544402			
0.997495	0.070560	-0.325843	-0.069854	0.187600	0.068686	-0.125671	-0.067072	0.089309	0.065029			
0.909297	-0.378401	-0.093139	0.247340	-0.108804	-0.089429	0.141515	-0.035988	-0.083443	0.091295			
0.598472	-0.479462	0.312667	-0.136005	-0.013264	0.108381	-0.139375	0.114118	-0.054131	-0.013235			
0.141120	-0.139708	0.137373	-0.134143	0.130058	-0.125165	0.119522	-0.113197	0.106264	-0.098803			

Figura 93. Matriz de términos de los resultados de la fase 3.

```
Dentro del hilo 2-1 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 1
17.741955 8.238461 6.401976 5.670115 5.235697 4.901000 4.599813 4.306051 4.008827 3.703954
18.914240 9.296898 -4.949284 -9.044386 -3.436352 3.667078 5.189808 1.095860 -3.233021 -3.334370
2.696860 -15.976210 3.397532 6.153495 -7.185224 2.141038 3.225403 -4.624947 1.882581 1.891597
-16.000000 4.000000 -1.777778 1.000000 -0.640000 0.444444 -0.326531 0.250000 -0.197531 0.160000
-19.986534 12.647035 0.122442 -7.460783 6.822136 -1.287553 -3.717747 4.552197 -1.522628 -2.160100
-5.597541 -14.526048 1.535345 8.753386 4.510364 -2.916985 -5.279106 -1.574690 2.972156 3.464957
13.937805 -0.557098 -3.162402 -3.982407 -4.263297 -4.314052 -4.242112 -4.093962 -3.893414 -3.654593

Dentro del hilo 2-2 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 2
1.621139 0.405285 0.180127 0.101321 0.064845 0.045032 0.033084 0.025330 0.020014 0.016211
2.046635 0.636620 0.172737 -0.101321 -0.225916 -0.212207 -0.105223 0.025330 0.114187 0.127324
1.273240 -0.405285 -0.424413 0.101321 0.254648 -0.045032 -0.181891 0.025330 0.141471 -0.016211
-1.621139 0.405285 -0.180127 0.101321 -0.064846 0.045032 -0.033084 0.025330 -0.020014 0.016211
-1.273239 -0.405285 0.424413 0.101321 -0.254648 -0.045032 0.181891 0.025330 -0.141471 -0.016211
0.246002 -0.636620 -0.427474 -0.101321 0.134211 0.212207 0.152011 0.025330 -0.085883 -0.127324
1.621139 0.405285 0.180127 0.101321 0.064846 0.045032 0.033085 0.025330 0.020014 0.016212

Dentro del hilo 2-3 con ID: 139833888470720 cuyo proceso es el 4758
Matriz de terminos de la serie 3
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.479426 0.420735 0.332498 0.227324 0.119694 0.023520 -0.050112 -0.094600 -0.108614 -0.095892
0.841471 0.454649 0.047040 -0.189201 -0.191785 -0.046569 0.093855 0.123670 0.045791 -0.054402
0.997495 0.070560 -0.325843 -0.069854 0.187600 0.068686 -0.125671 -0.067072 0.089309 0.065029
0.909297 -0.378401 -0.093139 0.247340 -0.108804 -0.089429 0.141515 -0.035988 -0.083443 0.091295
0.598472 -0.479462 0.312667 -0.136005 -0.013264 0.108381 -0.139375 0.114118 -0.054131 -0.013235
0.141120 -0.139708 0.137373 -0.134143 0.130058 -0.125165 0.119522 -0.113197 0.106264 -0.098803
```

Figura 94. Matriz de términos de los resultados actuales.

Resultados de la suma de términos:									
81.967331	31.325947	10.761600	4.072079	5.167937	8.501312	-1.066057			
6.845722	6.811500	5.056510	3.007303	2.930403	3.724472	6.845724			
0.000000	1.253979	1.124519	0.890240	0.600243	0.298166	0.023321			

Figura 95. Suma de términos de cada serie los cuales coinciden con los valores obtenidos en la columna Suma de la figura 46.

Conclusiones individuales.

El análisis de frecuencia es una herramienta que nos permite conocer más acerca de una señal proveniente de algún medio, yo considero que gracias a este análisis podemos obtener más datos acerca de algún fenómeno u objeto que se encuentre en nuestro entorno, lo cual es fundamental para el análisis y procesamiento de datos que es muy utilizado en el campo ciencia y la ingeniería. De igual forma nos permite conocer acerca de una señal y su comportamiento para aplicar algún método de filtrado o clasificación de esta señal. En el campo de la inteligencia artificial existen múltiples aplicaciones del análisis de frecuencia, ya que se trabajan con datos, los cuales muy comúnmente son provenientes de una señal como una imagen, un video, un audio o algún tipo de sensor que capture dicha señal. Por lo que el uso de una transformada de Fourier o incluso una serie de Fourier nos da la capacidad de conocer a fondo dicha señal para aplicarle algún tipo de filtro o hacer una predicción. Sin duda es una herramienta versátil que, aunque muchos no tengan conocimiento de ella, la realidad es que todos la usamos ya sea en nuestros dispositivos al realizar una compresión de archivos, capturar una imagen, usar algún filtro, o incluso dentro de nuestros mismos cerebros se aplica dicha transformada de forma implícita para reconocer ciertas frecuencias o imágenes. Así que puedo concluir que esta poderosa herramienta nos abre las puertas a muchos avances en investigación y la comunicación de las personas en un nuevo entorno digital.

En cuanto a mi experiencia resolviendo una serie de Fourier para una función algebraica puedo decir que me pareció un proceso un tanto tedioso al principio, pero una vez que logré memorizar las ecuaciones correspondientes a las integrales más comunes como lo son las integrales que se resuelven por partes o incluso las equivalencias de las funciones seno y coseno que tienen cierto comportamiento periódico y aperiódico, pude resolver dicha serie de forma más rápida y con ayuda de herramientas como Geogebra pude comprobar la solución obtenida en el desarrollo.

Respecto al cálculo de una serie de Fourier a través de funciones de Excel me pareció algo interesante y hasta cierto punto didáctico ya que como programadores no estamos acostumbrados a utilizar Excel siendo una herramienta muy utilizada en la industria.

El desarrollo de los programas realizados en la tercera fase me pareció algo complicado pero más que nada una actividad que integra varios de mis conocimiento en programación, lenguaje c, estructuras, uso de Linux y principalmente de cómputo paralelo, lo cual fue gratificante para tener una clara idea de cómo aplicar dicha herramienta en un entorno como Linux y C que operan en un nivel medio de computo.

Finalmente, el desarrollo de los programas de la fase 3 resultó bastante sencillo en comparación a los de la fase 2 ya que partimos de los programas realizados en la fase anterior y los utilizamos como plantilla para el desarrollo de esta fase lo cual facilitó bastante el proceso de desarrollo, realmente no se tuvo mucha dificultad más que la de representar los parámetros con un conjunto de TDA definidos por medio de structs dentro de los archivos de cabecera pero de ahí en fuera lo demás resultó sencillo. De igual forma el rendimiento de los programas a la hora de la ejecución resultó bastante similares a los de la fase 2 por lo que no se presentó ningún inconveniente.

(Tinoco Videgaray Sergio Ernesto)

La serie de Fourier fue un descubrimiento importante que tuvo mucho más alcance del que se había intentado. Se utiliza para descomponer una señal periódica en una serie de ondas sinusoidales de diferentes frecuencias. Esta descomposición permite representar la señal original de una manera más sencilla y compacta, lo que la hace útil en muchas aplicaciones. Puede ser utilizada en ciencias exactas como en matemáticas y física como una herramienta para resolver ecuaciones diferenciales. También en áreas más prácticas en donde se requiera el procesamiento de señales como en procesamiento de imágenes, medicina, inteligencia artificial entre otras.

El cálculo de una serie de fourier a mano puede parecer un proceso tedioso pero la realidad es que es bastante simple dependiendo de la función que se utilice para dicho cálculo.

De igual forma el proceso de realizar una función en Excel cuesta trabajo de implementar en un inicio pero resulta ser todavía más fácil si se conocen las herramientas que este programa ofrece.

Respecto a la implementación de estos programas me pareció interesante la unión de algunos elementos como la memoria compartida con los procesos hijos en la que se llevaron a un nivel más profundo a la hora de crear un árbol de procesos hijos de cada hijo.

Por último, la implementación de hilos facilitó bastante la implementación del algoritmo al no tener que crear tantos procesos hijos como en una fase anterior por lo que el uso de hilos resulta mucho más eficiente en términos de líneas de código y tiempo de ejecución de los programas.

(Ibarra Gonzalez Emilio Francisco)

En este trabajo se realizó el cálculo de los términos de Fourier de forma paralela utilizando recursos como la memoria compartida y los procesos hijos.

Calcular una serie de Fourier a mano resulta difícil si no se tienen los conocimientos de cálculo integral y las fórmulas más comunes para simplificar algunas expresiones trigonométricas.

El cálculo de una serie de Fourier por medio de Excel resulta todavía más difícil si no se conocen las funciones que Excel utiliza y hace que el trabajo sea más complicado.

Yo creo que el trabajo más difícil fue el de implementar un algoritmo que utiliza procesos hijos para calcular los términos de Fourier debido a las matrices de términos y a la sincronicidad de la memoria compartida pues nos planteó un gran desafío, como la incoherencia de datos y la exclusión mutua. Para ello, se han implementado diversos algoritmos y métodos, entre ellos los "semáforos". También considero que trabajar esto en C resulta más tedioso a la hora de pasar datos entre funciones ya que requiere el uso de apuntadores y lo vuelve un trabajo sumamente abstracto.

Al final se tuvo que implementar el mismo algoritmo haciendo uso de hilos de proceso y esto nos facilitó la implementación del algoritmo que ya teníamos codificado previamente y nos ahorró muchas líneas de código y número de procesos en la ejecución de cada programa, por lo que concluimos que utilizar hilos resulta más eficiente que utilizar procesos hijos en términos generales.

(Sánchez De Los Ríos Flavio Josué)

REFERENCIAS

- [1] J. V. Lázaro, "La Transformada de Fourier en Varias Variables y Sus Aplicaciones al Estudio De Las EDPs" Trabajo fin de grado, Universidad de Cantabria, Santander, España, 2016. Available at: <https://repositorio.unican.es/xmlui/bitstream/handle/10902/20494/Lazaro%20Julian%20Victor.pdf?sequence=1&isAllowed=y>
- [2] A. A. Franco, "Detección de la profundidad y fondo marino a través de ondas acústicas" Notas de Aplicación FVC, Universidad Nacional del Sur, Bahía Blanca, Argentina, 2010. Available at: <http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-Andres%20Agustin%20Franco.pdf>
- [3] Herrera, J.R. and Ruiz, V.G. (2014) "Análisis de Fourier". Available at: https://w3.ual.es/~vruiz/Docencia/Apuntes/Signals/Fourier_Analysis/index.html
- [4] Ibaceta I.G. (2021) "SERIES DE FOURIER Y SUS APLICACIONES". Available at: <https://repositorio.unican.es/xmlui/bitstream/handle/10902/23810/GoniIbacetaIrene-TFG-Matematicas.pdf?sequence=1>
- [5] BIG.LAT (2021) "Imagenología analógica vs imagenología digital". Available at: <https://www.technodomus.com/blog/imagenologia-4/imagenologia-analogica-vs-imagenologia-digital-19#:~:text=Qué%20es%20una%20imagen&text=Una%20imagen%20está%20formada%20por,los%20ejes%20espaciales%20x%20e%20y>
- [6] Otarola, L.A. (2015) "Transformada de Fourier para el procesamiento digital de imágenes". Available at: <http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-LucaOtarola.pdf>
- [7] Peña, O.A. (2019) "ANÁLISIS DE AUDIO PARA EXTRACCIÓN DE CARACTERÍSTICAS, SEGMENTACIÓN, CLASIFICACIÓN Y PREDICCIÓN" Available at: <https://cimat.repositoryinstitucional.mx/jspui/bitstream/1008/1029/1/TE%20754.pdf>
- [8] Nagesh S.C. (2020) "Fourier Transformation for a Data Scientist". Available at: <https://www.kdnuggets.com/2020/02/fourier-transformation-data-scientist.html>
- [9] Montes de Oca E. et al. (2020) "Una implementación paralela de las Transformadas DCT y DST en GPU. Análisis de performance" Available at: http://sedici.unlp.edu.ar/bitstream/handle/10915/23624/Documento_completo.pdf?sequence=1
- [10] Márquez, F. (2019) "Integrales Comunes coeficientes series de fourier.pdf" , <https://fisicaymates.com.> Google. Available at: <https://drive.google.com/file/d/1fnPjspVou9pmYNEuP2bRCDsnjXGnTA6D/vie>