



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal II

“Desarrollo de una aplicación web para el cálculo y graficación de series de Fourier”

2025-A075

QUE PARA OBTENER EL GRADO DE:

Ingeniero en Sistemas Computacionales

PRESENTA:

Sebastián Morales Palacios

DIRECTORES:

M. en C. Jesús Alfredo Martínez Nuño

M. en C. Gisela González Albarrán

Ciudad de México
Noviembre 2024

Resumen

Dentro del presente trabajo terminal se lleva a cabo el desarrollo del prototipo de una aplicación web diseñada para calcular y graficar el desarrollo en series de Fourier de funciones periódicas. El desarrollo en serie puede ser tanto en forma trigonométrica como en forma exponencial compleja, para funciones continuas y discretas.

La aplicación pretende ser de utilidad como herramienta de apoyo para estudiantes y docentes en el tema del análisis de Fourier.

Palabras Clave - aplicación web, análisis de Fourier, cálculo, matemáticas avanzadas para la ingeniería.

Correo de Contacto

smoralesp1700@alumno.ipn.mx

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

1.	Código en Matlab para graficar la serie de Fourier trigonométrica de ??.	123
2.	Código en Matlab para graficar la serie de Fourier compleja de ??.	124
3.	Código en Maple para calcular y graficar la serie de Fourier trigonométrica de ??.	126
4.	Código en Maple para calcular y graficar la serie de Fourier compleja de ??.	127
5.	Código en Maxima para calcular y graficar la serie de Fourier trigonométrica de ??.	128
6.	Código en Maxima para calcular y graficar la serie de Fourier compleja de ??.	129
7.	Código en Python con matplotlib y sympy para graficar la serie de Fourier trigonométrica de ??.	131
8.	Código en Python con matplotlib y sympy para calcular y graficar la serie de Fourier compleja de ??.	132
9.	Código en Python con Manim para graficar la serie de Fourier trigonométrica de ??.	136
10.	Código en Python con Manim para graficar la serie de Fourier compleja de ??.	140
11.	Parte del código en Javascript responsable de generar la gráfica y sus funcionalidades	142
12.	Parte del código en Javascript responsable de graficar funciones matemáticas	142
13.	Parte del código en Javascript responsable de graficar series de funciones	143
14.	Código en Maxima para calcular los coeficientes y su expansión de la serie Trigonométrica de Fourier	144
15.	Código en Maxima para calcular los coeficientes y su expansión de la serie compleja de Fourier	146
16.	API en NodeJS para ejecutar comandos de Maxima	149
17.	Script en NodeJS para probar la biblioteca de tex2max	150
18.	Código para usar MathQuill para introducir datos en T _E X	151
19.	Código para usar MathQuill y MathJax para crear un teclado e introducir datos en T _E X	152

CAPÍTULO 1

Introducción

En este capítulo, examinaremos el problema que se abordará, la necesidad e impulso para hacerlo, así como los objetivos previstos para este proyecto y nuestras propuestas de solución para esta problemática.

1.1. Motivación

Las matemáticas son fundamentales en la formación de un ingeniero, proporcionando herramientas esenciales para modelar y resolver problemas complejos. Durante mi formación en la Escuela Superior de Cómputo, al estudiar el análisis de Fourier y resolver series de Fourier, noté la ausencia de métodos eficientes para verificar los resultados de manera directa. A diferencia de otros cálculos matemáticos que cuentan con calculadoras especializadas, las series de Fourier requieren procesos un tanto más elaborados que implican el uso de múltiples softwares para calcular coeficientes y graficar funciones, consumiendo tiempo valioso que podría dedicarse a una comprensión más profunda de los conceptos. Esta situación motiva el desarrollo de herramientas más óptimas para facilitar y agilizar la verificación de series de Fourier, mejorando así la eficiencia en el aprendizaje y aplicación de estos conceptos.

1.2. Planteamiento del Problema

Las series de Fourier, desde su concepción por Jean-Baptiste Joseph Fourier a principios del siglo XIX [1], han desempeñado un rol crucial en el análisis y la comprensión de señales y fenómenos periódicos. Las series de Fourier son parte esencial en campos como la ingeniería eléctrica, la física teórica y el procesamiento de señales, imágenes y audio, permitiendo descomponer funciones periódicas en sumas de senos y cosenos, lo que facilita su análisis y manipulación. Estas herramientas matemáticas no solo han impulsado avances significativos en la ciencia y tecnología, transformando nuestra interacción con el mundo, sino que también son cruciales en la enseñanza de los fundamentos teóricos de la ingeniería. Esta capacidad de simplificar señales complejas en componentes básicos no solo mejora el análisis, sino que también facilita la síntesis de nuevas tecnologías que se adaptan a necesidades y entornos cambiantes.

Este amplio espectro de aplicaciones destaca la importancia crítica de las series

de Fourier no solo en la investigación avanzada, sino también en la formación académica de futuros ingenieros y científicos. Sin embargo, a pesar de su prevalencia en el currículo educativo, la implementación práctica de este análisis en entornos de aprendizaje a menudo revela áreas de oportunidad dentro de las herramientas disponibles, lo que impacta directamente en la eficacia con la que los estudiantes pueden aplicar y profundizar su comprensión de estos conceptos esenciales. La necesidad de una herramienta más eficiente, como se mencionó en la motivación, es crucial para superar estos desafíos y mejorar la experiencia educativa y profesional en el análisis de series de Fourier.

1.3. Justificación

El análisis de Fourier es fundamental en áreas como la ciencia, la física y la ingeniería, a pesar de esto, el acceso a herramientas que combinen eficientemente el cálculo y la visualización de series de Fourier sigue siendo limitado. Actualmente, hay herramientas que independientemente ayudan a resolver y comprobar algunos problemas pero, ciertos problemas resaltan el valor de esta herramienta ya que los usuarios deben recurrir a múltiples plataformas, lo que añade complejidad y tiempo al proceso, especialmente para aquellos que necesitan resultados rápidos y visualizaciones claras.

Por ello, este proyecto propone el desarrollo de una aplicación web que integre el cálculo simbólico y la representación gráfica interactiva en una sola plataforma. Esto permitirá a estudiantes, ingenieros y profesionales realizar cálculos y visualizar resultados de manera práctica, rápida e intuitiva, sin requerir conocimientos técnicos avanzados o el uso de herramientas separadas.

1.3.1. Encuesta

Para validar aún mas la relevancia del proyecto, se realizó una encuesta dirigida a estudiantes de ingeniería de diferentes planteles universitarios (véase Apéndice A). Los resultados permitieron identificar los principales retos y necesidades al trabajar con asignaturas de matemáticas avanzadas, particularmente con el análisis de Fourier:

A continuación, se presentan los resultados más destacados de la encuesta, organizados por categorías clave. Cada punto incluye el porcentaje de participantes que seleccionaron dicha opción, proporcionando una visión clara de las preferencias y necesidades de los encuestados.

- **Dificultades en asignaturas de matemáticas:** Los encuestados señalaron como principales desafíos:
 - Tener la certeza de que sus resultados son correctos (52.4 %).
 - Entender que es lo que están haciendo (51.2 %).
 - Resolver muchos ejercicios en periodos cortos de tiempo (46.3 %).
 - Comprender los conceptos abstractos (46.3 %).
- **Uso de software de cálculo y graficación:** La mayoría de los participantes ha utilizado herramientas como *Geogebra* (72 %) y *Symbolab*

(53.7 %) las cuales están diseñadas específicamente para ser didácticas y accesibles en el aprendizaje y enseñanza de matemáticas, proporcionando interfaces intuitivas y funcionalidades orientadas a la educación, mientras que herramientas como *MATLAB* (5%) aunque potentes, están más enfocadas a aplicaciones técnicas avanzadas y tienen una curva de aprendizaje más pronunciada, por lo tanto, son menos populares entre los estudiantes.

■ **Características esenciales en herramientas digitales:** Los encuestados destacaron la importancia de:

- Resoluciones paso a paso de los problemas (86.6%).
- Interfaces fáciles e intuitivas (69.5%).
- Visualización gráfica interactiva (62.2%).

■ **Perspectiva sobre las series de Fourier:**

- El 60 % de los encuestados indicó haber trabajado con problemas relacionados con análisis de Fourier. Dentro de este grupo, los principales retos identificados fueron:
 - Dificultad para interpretar sus resultados, una vez llegaban al resultado no sabían que significaba o qué hacer con él (56 %).
 - Los problemas a resolver se tornaban demasiado largos (48 %).
 - No tuvieron acceso en su momento a herramientas para la resolución de este tipo de problemas (44 %).
 - No entendieron el propósito de las series de Fourier (40 %).
- El 40 % de los encuestados no ha trabajado con Fourier, pero al ver una demostración visual, destacaron que desearían una herramienta que:
 - Explique paso a paso los procesos (78.1 %).
 - Permita “jugar” y manipular gráficas interactivas (68.8 %).
 - Sea fácil de usar (53.1 %).
 - Sea accesible incluso para quienes no conocen Fourier (46.9 %).

■ **Importancia de la visualización gráfica:** Ante la pregunta “*¿En qué medida crees que la visualización gráfica interactiva te ayuda a comprender los conceptos?*”, el 56 % de los encuestados calificó con un 10 esta característica, mientras que el 34 % otorgó una puntuación de entre 8 y 9.

Los resultados de esta encuesta sugieren una necesidad percibida por los estudiantes de contar con una herramienta que no solo resuelva problemas matemáticos, sino que también integre gráficos interactivos así como la interpretación de los resultados podría facilitar el aprendizaje y la comprensión de las series de Fourier.

1.4. Solución Propuesta

La solución que se plantea en este proyecto es el desarrollo de una aplicación web que permita el cálculo y graficación de series de Fourier de manera integrada, eficiente e intuitiva, cubriendo la brecha existente en las herramientas actuales, que tienden a separar el cálculo matemático de la visualización gráfica en una sola interfaz. La solución propuesta unificará el cálculo de los resultados con la graficación interactiva de las mismas en una sola plataforma, diseñada para ser simple de usar, práctica y accesible desde cualquier navegador web, para usuarios que deseen analizar y visualizar series de Fourier, eliminando el problema de usar múltiples programas para resolver y graficar las series de Fourier.

La aplicación web contará con las siguientes características clave:

- **Cálculo automático de series de Fourier para diferentes tipos de funciones:**
 - La aplicación calculará **series de Fourier** para **funciones continuas** en un intervalo o para **funciones definidas a trozos**. El usuario podrá ingresar funciones matemáticas, incluso aquellas con discontinuidades o que estén definidas por partes en distintos intervalos, y la aplicación manejará estos casos automáticamente.
 - Se implementará tanto la **serie trigonométrica** como la **serie exponencial compleja**. La serie trigonométrica descompondrá la función en términos de senos y cosenos, mientras que la versión compleja lo hará en términos de exponenciales imaginarios, lo cual es útil en el contexto de análisis más avanzado y en aplicaciones de ingeniería.
- **Extensiones de medio rango:**
 - La herramienta permitirá calcular **extensiones de medio rango** para funciones en un medio intervalo. Esto incluirá la posibilidad de obtener:
 - **Serie de senos** para funciones impares.
 - **Serie de cosenos** para funciones pares.
 - **Serie completa** para funciones periódicas en un intervalo definido, permitiendo extender la función para construir series de Fourier en un medio rango específico.
- **Visualización gráfica interactiva:**
 - Una vez calculados los coeficientes de Fourier, la aplicación generará una **gráfica interactiva** que mostrará la aproximación de la serie de Fourier a la función original. El usuario podrá ajustar el número de términos de la serie para observar cómo mejora la aproximación conforme se incluyen más términos en la suma.
- **Interfaz intuitiva y amigable:**
 - La aplicación ofrecerá una interfaz sencilla y accesible, en la que el usuario podrá ingresar las funciones, definir los intervalos y

elegir el tipo de serie de Fourier que desea calcular. La experiencia estará diseñada para minimizar la curva de aprendizaje, permitiendo a usuarios sin experiencia en programación obtener resultados rápidamente.

1.5. Objetivos

1.5.1. Objetivo General

Desarrollar un prototipo de una aplicación web capaz de calcular las series de Fourier en su forma trigonométrica o exponencial compleja de una función continua en un intervalo o definida a trozos, así como ser capaz de obtener la extensión par o impar de dicha función y, finalmente, graficar tanto la función original como la función expandida como una serie de Fourier y poder añadir o eliminar términos de dicha forma para apreciar como esta se aproxima a la función original.

1.5.2. Objetivos Específicos

- Implementar una interfaz de usuario que permita ingresar la función continua o a trozos, seleccionar intervalos y definir el tipo de serie de Fourier (trigonométrica o exponencial) o tipo de expansión (par o impar).
- Implementar los módulos para el cálculo de coeficientes de la serie trigonométricas y serie exponencial compleja, adaptándose a funciones continuas o definidas a trozos.
- Desarrollar un módulo de visualización que permita graficar tanto la función original como la aproximación de la serie de Fourier. Este módulo debería incluir opciones para añadir o eliminar términos y visualizar cómo estas modificaciones afectan la aproximación a la función original

CAPÍTULO 2

Estado del Arte

Para la correcta comprensión del trabajo presente, se necesita conocer el estado actual de las herramientas y tecnologías disponibles para el cálculo y la visualización de series de Fourier, así como los estudios y proyectos previos que abordan la implementación de soluciones similares. En este sentido, se revisarán diversas plataformas de uso común que permiten realizar estos procesos de manera separada, además de calcular y/o graficar la serie de Fourier para la función $f(x) = x$ en el intervalo de $-\pi$ a π utilizando cada una de las herramientas previamente descritas. El cálculo se realizará en su forma trigonométrica y, en los casos en que la herramienta lo permita, también se obtendrá la forma exponencial compleja. Asimismo, se graficará la serie de Fourier en las plataformas que lo permitan, lo que nos permitirá comparar tanto el proceso como los resultados obtenidos en cada herramienta.

Esta comparación servirá para identificar las capacidades, ventajas y limitaciones de cada una de las plataformas en el contexto del cálculo y la visualización de series de Fourier, evaluando también su facilidad de uso y precisión en la representación gráfica.

- **Forma Trigonométrica:**

- Coeficientes:

$$a_0 = 0, \quad a_n = 0, \quad b_n = \frac{2(-1)^{n+1}}{n}$$

- Serie trigonométrica de Fourier para $f(x) = x$:

$$f(x) = 2 \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(nx)$$

- **Forma Exponencial Compleja:**

- Coeficiente complejo:

$$c_n = \frac{i(-1)^n}{n}, \quad \text{para } n \neq 0.$$

- Serie exponencial compleja de Fourier para $f(x) = x$:

$$f(x) = \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} \frac{i(-1)^n}{n} e^{inx}$$

(Los cálculos para llegar a estos coeficientes se encuentran desarrollados en el Apéndice B.)

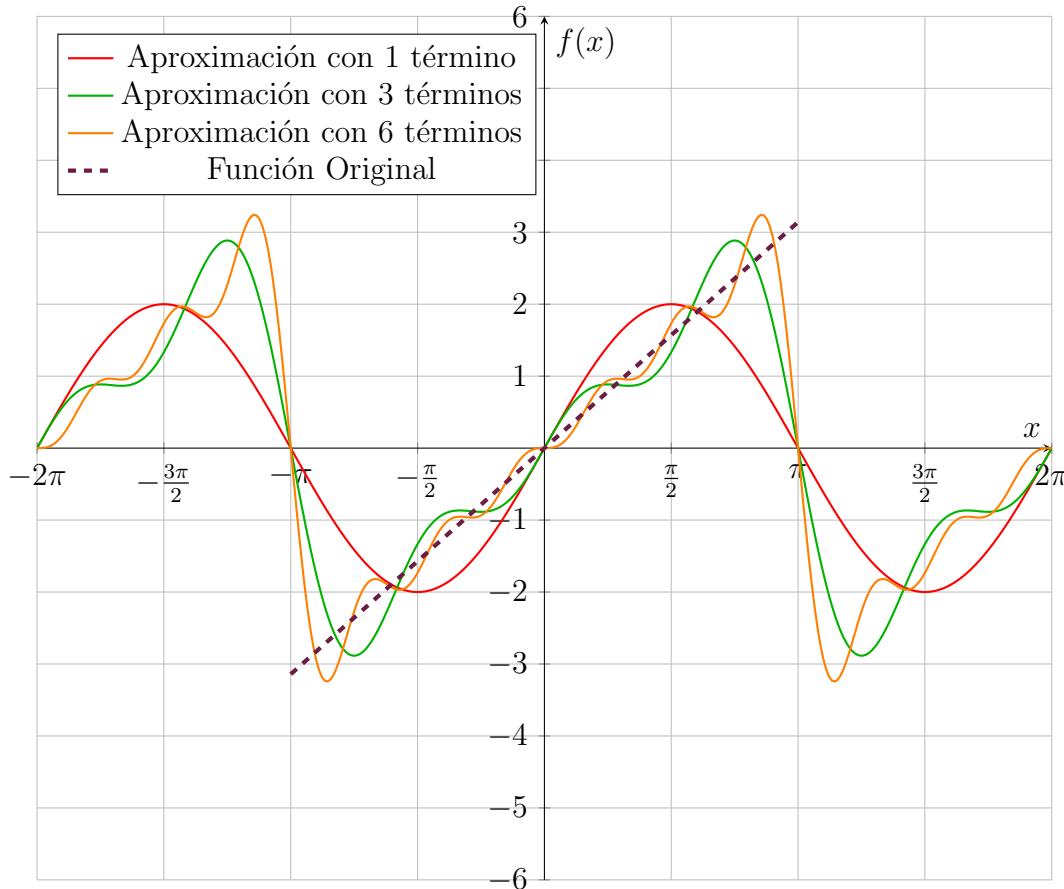


Figura 2.1: Gráfica de los coeficientes de Fourier calculados en el Apéndice B
Fuente: Elaboración propia

Podemos ver en la ??, la gráfica de la función y sus aproximaciones de Fourier, donde estas aproximaciones son equivalentes entre la serie trigonométrica y la serie exponencial. Esta revisión permitirá contextualizar la propuesta de una aplicación web que integre ambas funcionalidades en una sola plataforma.

2.1. Herramientas y Tecnologías Actuales

Las series de Fourier al formar parte importante en diversas áreas de aprendizaje como la ingeniería, la física y las matemáticas aplicadas, ha impulsado el desarrollo de numerosas soluciones tecnológicas para su cálculo y visualización. En consecuencia, hoy en día disponemos de una gran variedad de plataformas y herramientas que facilitan estos procesos, permitiendo resolver y graficar series de Fourier de forma parcial. En esta sección, se presenta un análisis de las herramientas más significativas para esto, así como sus características, funcionalidades y limitaciones en cuanto a la incorporación de ambas capacidades en una única plataforma.

2.1.1. WolframAlpha

Wolfram Alpha es una aplicación comercial que proporciona respuestas tanto a preguntas como cálculos basados en el motor de Mathematica, un potente software para cálculos simbólicos y numéricos [2].



Figura 2.2: Logotipo de WolframAlpha. Fuente: [2]

En contraste con otros motores de búsqueda, en vez de ofrecer una lista de sitios web o documentos, proporciona respuestas precisas y exhaustivas basándose en los conceptos introducidos en su motor de búsqueda. Entre algunas de sus capacidades, este potente software tiene funciones propias para resolver y graficar series de Fourier [3]:

- `FourierSeries[exp, t, n]`
- `FourierTrigSeries[exp, t, n]`
- `FourierSinSeries[exp, t, n]`
- `FourierCosSeries[exp, t, n]`

Estas funciones nos permiten obtener expansión de la serie de Fourier, ya sea compleja, trigonométrica, en senos o cosenos de una función `exp` con periodo de 2π donde esta puede ser de un solo trozo o definida a varios trozos, para esto se usaría la función `Piecewise[val1,cond1,val2,cond2,...]` [4], en términos de la variable `t` hasta el `n`-ésimo término. Además nos mostrará una gráfica estática de una aproximación con `n` términos. También tiene funciones para calcular los coeficientes de la serie [2]:

- `FourierSeriesCoefficient[exp, t, n]`
- `FourierSinCoefficient[exp, t, n]`
- `FourierCosCoefficient[exp, t, n]`

Estas funciones nos permiten calcular el `n`-ésimo coeficiente de la serie de Fourier exponencial compleja, de senos o de cosenos, de una función `exp` con periodo de 2π , recordando que puede ser a trozos o de un solo trozo.

Podemos observar que las funciones propias de Wolfram Alpha para problemas que impliquen series de Fourier limitan el periodo en el que las funciones matemáticas son definidas, limitándolo en 2π , para estos casos se podría resolver el coeficiente directamente usando la función de `Integrate[f,x,xmin,xmax]` [4] para calcular individualmente cada coeficiente, ya sea a_0, a_n y b_n para las series trigonométricas o c_0 y c_n para la serie compleja, pero así no nos proporciona la pequeña gráfica que si nos da al usar las funciones para expandir la serie, además de que la gráfica que se proporciona no es interactiva, y si queremos verla con más detalle debemos pagar su suscripción.

2.1.1.1. Prueba Wolfram Alpha

Para nuestra primer prueba, calcularemos la serie de Fourier trigonométrica de la función calculada en el Apéndice B, para hacerlo usaremos la función `FourierTrigSeries(exp, t, n)` de WolframAlpha, que nos permite calcular la serie trigonométrica de Fourier de la función `exp` respecto a la variable `t` obteniendo la serie hasta el término `n`.

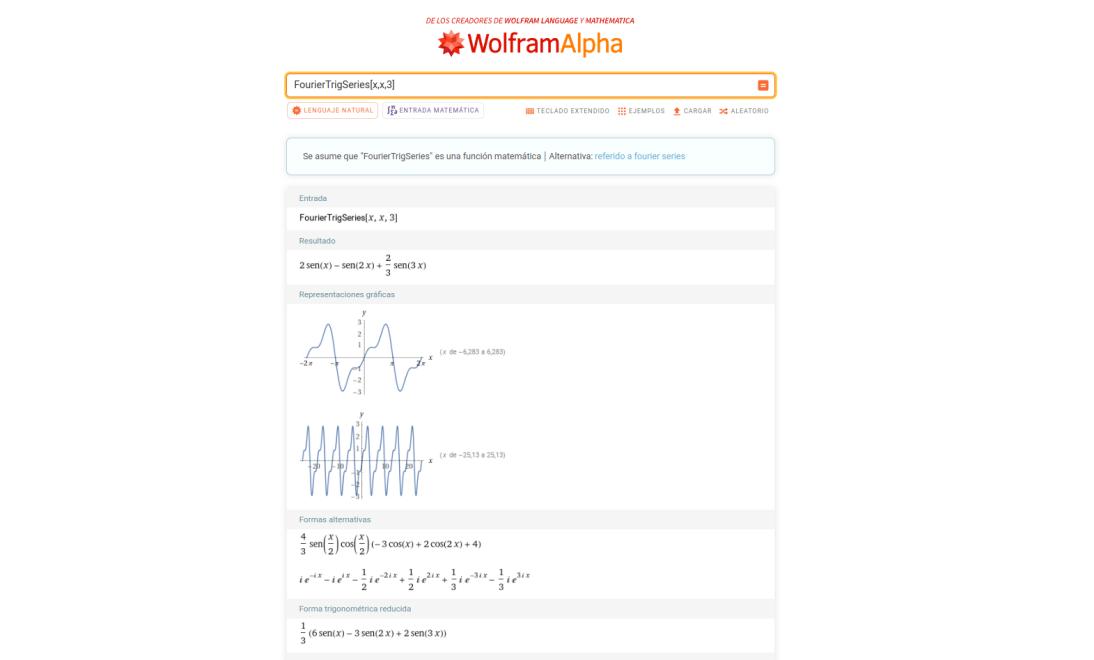


Figura 2.3: Calculo de una serie trigonométrica de Fourier en WolframAlpha

Ahora usaremos la función de `FourierSeries(exp, t, n)` de WolframAlpha, esta función no permite calcular la serie exponencial compleja de Fourier de la función `exp` respecto a la variable `t` obteniendo la serie hasta el término `n`.

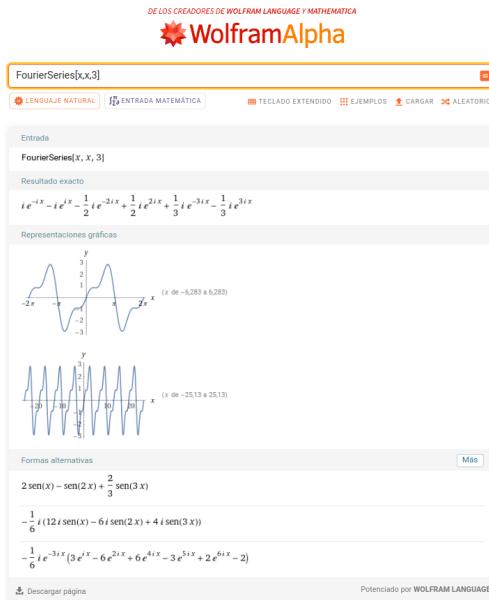


Figura 2.4: Calculo de una serie trigonométrica de Fourier en WolframAlpha

Wolfram nos devuelve la expansión de cada serie en su forma mas reducida, ademas de dos pequeñas gráficas de nuestra aproximación vista desde no nos devuelve los coeficientes de la serie ni tampoco su expresión final en notación de serie.

2.1.2. Symbolab

Symbolab es una calculadora digital que ayuda a resolver problemas matemáticos, como ecuaciones, álgebra o cálculo. Se trata de una herramienta que cuyo principal propósito es ayudar a sus usuarios a aprender matemáticas, ya que ofrece resoluciones paso a paso y conocimientos impulsados por inteligencia artificial. [5].



Figura 2.5: Logotipo de Symbolab. Fuente: [5]

Symbolab se centra en la enseñanza de procedimientos matemáticos mediante explicaciones detalladas. Su interfaz es especialmente útil para estudiantes, ya que presenta guías paso a paso para temas como derivadas, integrales y ecuaciones algebraicas complejas, lo cual es ideal para el aprendizaje autodidacta. Este cuenta con una función para resolver series de Fourier en su forma trigonométrica denominada `fourier series (f)`, `[-L, L]` [5] en donde `f` es una función definida en el intervalo de `[-L, L]`. A pesar de contar con funciones para definir funciones matemáticas a trozos y funciones para hacer gráficas, no es posible unificarlas en la misma aplicación, ademas de no contar con funciones para otras

series de Fourier como extensiones de medio rango o exponencial compleja, pero, al igual que en Wolfram Alpha, se pueden calcular individualmente los coeficientes con la función `integral from a to b of x [5]`.

2.1.2.1. Prueba Symbolab

Para nuestra siguiente prueba, calcularemos la serie de Fourier trigonométrica de la función calculada en el Apéndice B, para hacerlo usaremos la función `fourier series (f), [-L, L]` de Symbolab, que nos permite calcular la serie trigonométrica de Fourier de la función `f` respecto a la variable obteniendo desde el intervalo `-L` a `L`.

The screenshot shows the Symbolab interface with the search bar containing "serie de fourier x, [-pi, pi]". The left sidebar has categories like Pre-Algebra, Algebra, Precalculus, Calculus, Functions, Matrices and vectors, Trigonometry, Statistics, and Conversions. The main area shows the input "serie de fourier x, [-pi, pi]" and the solution: $\sum_{n=1}^{\infty} \frac{2(-1)^n \sin(nx)}{n}$. Below it, the "Pasos de solución" (Steps of solution) section shows the derivation of the formula, including the application of the Fourier series formula and the simplification of the resulting integral expression. A note on the right says "Guardar en el cuaderno!" (Save to notebook!).

Figura 2.6: Calculo de una serie trigonométrica de Fourier en Symbolab

Symbolab nos dará la expresión en forma de serie, y solo podremos hacerlo para la serie trigonométrica, ademas de que no nos muestra ninguna gráfica.

2.1.3. MATLAB

MATLAB es un entorno de programación y un lenguaje de alto nivel comercial ampliamente utilizado en ingeniería, física y matemáticas aplicadas para el análisis numérico, la visualización de datos y el desarrollo de algoritmos. Su robusta caja de herramientas, especialmente la *Signal Processing Toolbox* y la *Symbolic Math Toolbox*, facilitan el cálculo y la graficación de series de Fourier [6].



Figura 2.7: Logotipo de Matlab. Fuente: [6]

A pesar de que MATLAB cuenta con funciones especiales para el análisis de Fourier como la transformada de Fourier (`fourier(f)`) o la Transformada rápida de Fourier (`fft(f)`) no cuenta con funciones específicas para series de Fourier, sin embargo, podemos definir nuestros coeficientes para hacer el gráfico de ambas series.

2.1.3.1. Prueba Matlab

Para la prueba con Matlab, primero ejecutaremos el código en Apéndice C, que nos graficará la serie de Fourier a partir de sus coeficientes trigonométricos ??.

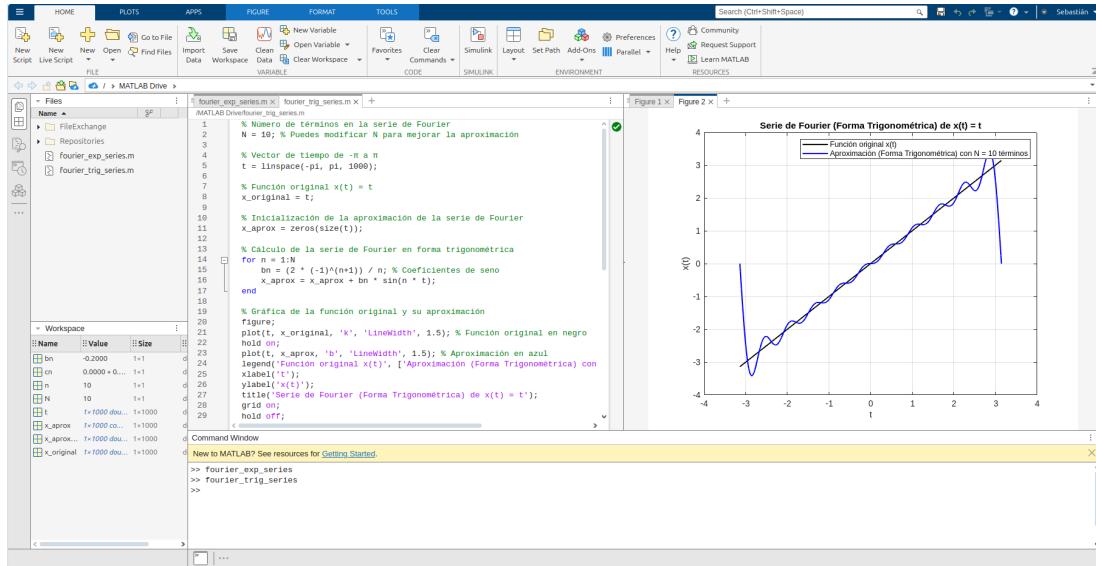


Figura 2.8: Graficación de una serie trigonométrica de Fourier en Matlab

Ahora usamos el código en Apéndice C para graficar la misma serie pero ahora usando su coeficiente complejo ??.

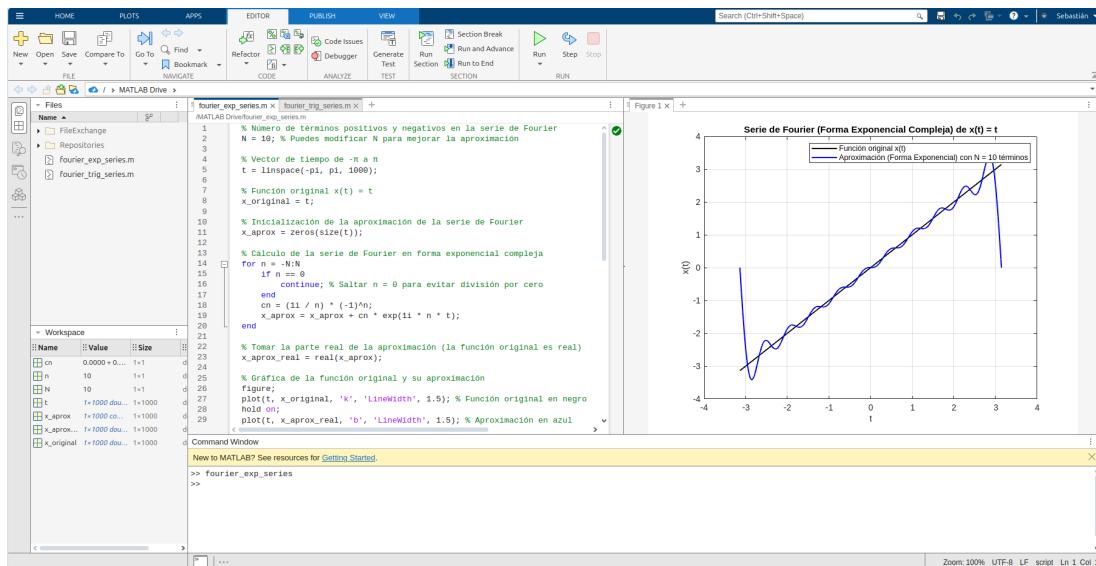


Figura 2.9: Graficación de una serie compleja de Fourier en Matlab

Matlab es capaz de graficar muy buenas aproximaciones de ambas series en una gráfica que nos brinda cierto nivel de interactividad pero todo dependerá de nuestros cálculos y de que tanto detalle plasmemos en nuestro código para aumentar las funcionalidades

2.1.4. Maple

Maple, desarrollado por Maplesoft, es un Sistema de Álgebra Computacional comercial que permite realizar cálculos matemáticos simbólicos y numéricos, resolver ecuaciones, analizar datos y crear visualizaciones gráficas en 2D y 3D. Maple utiliza el paradigma REPL (Read-Eval-Print Loop), un entorno interactivo donde cada línea de código ingresada es leída, evaluada y su resultado es impreso inmediatamente, facilitando así la experimentación y el desarrollo incremental de cálculos complejos. [7].



Figura 2.10: Logotipo de Maple. *Fuente:* [7]

Si bien este CAS cuenta con funciones dedicadas al cálculo de series de Fourier, estás vienen definidas en paquetes desarrollados por la comunidad de Maplesoft [8], así que resulta más eficiente y conveniente usar las funciones de integración (`int(f, x = a..b)`) calcula simbólicamente la integral de `f`, que depende de la variable `x` desde el punto `a` hasta el `b`) para calcular los coeficientes, y para expandir series (`seq(m, n = a..b)`) en donde `m` es la función a sumar,sobre la variable `n` desde el punto `a` hasta el `b`) las funciones de expansión, además de poder declarar una función a trozos con la función `piecewise(cond-1, f-1, cond-2, f-2, ..., cond-n, f-n, f-otherwise)`.

2.1.4.1. Prueba Maple

Para la prueba con Maple, primero ejecutaremos el código en Apéndice C, que, primeramente, calculará los coeficientes de la serie trigonométrica ??.

The screenshot shows the Maple 2024 interface with the following code:

```

fune := x
T := 2 π
series_cozime_core := cos(n-x)
series_sine_core := sin(n-x)
a0 := 0
an := 0
bn := -2(-1)^n/n
a0_simp := 0
an_simp := 0
bn_simp := -2(-1)^n/n
Coeff_A0 := 0
Coeff_An := 0
Coeff_Bn := -2(-1)^n/n
nI := 1
n2 := 5
lista_An := [0, 0, 0, 0]
lista_Bn := [2 sin(x), -sin(2x), 2 sin(3x)/3, -sin(4x)/2, 2 sin(5x)/5]
lista_completa := [0, 0, 0, 0]
serie_final := [0, 0, 0, 0]
serie_factor := [0, 0, 0, 0]
serie_funcion := 2 sin(x) - sin(2x) + 2 sin(3x)/3 - sin(4x)/2 + 2 sin(5x)/5
serie_funcion_simplificada := 2(48 cos(x)^4 - 30 cos(x)^3 - 16 cos(x)^2 + 13) sin(x)/15

```

Figura 2.11: Cálculos de coeficientes trigonométricos de Fourier en Maple

Posteriormente, este código nos dará una gráfica estática con los valores que le establecimos Apéndice C ??.

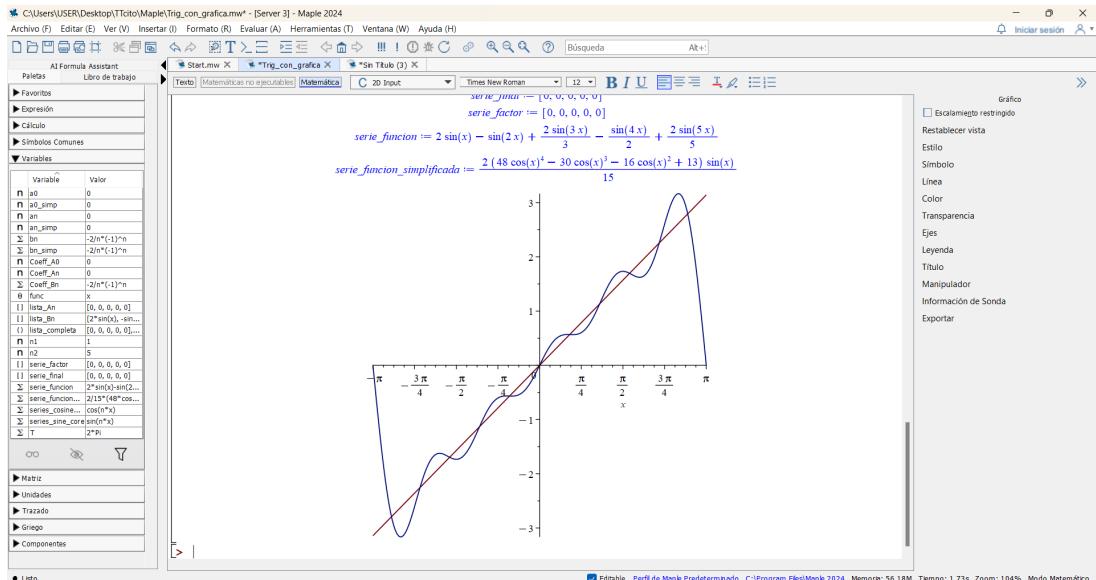


Figura 2.12: Grafica de la serie de Fourier trigonométrica en Maple

Podemos ver que puede resolver la serie trigonométrica sin problema alguno, ahora veamos la misma función pero desarrollada en una serie exponencial compleja Apéndice C.

2.1. HERRAMIENTAS Y TECNOLOGÍAS ACTUALES

15

```

# Graficamos la función y la serie aproximada
plot([func,serie_funcion_simplificada],x = -T/2..T/2);

func := x
T := 6
series_core := e^(1/3*x)
cn := 31((-1)^(n+1))/pi*n
c0 := 0
c0_simp := 0
cn_simp := 31((-1)^n)/pi*n
Coeff_0 := 0
Coeff_n := 31((-1)^n)/pi*n
n := 1
n1 := 1
n2 := 5
lista_completa := [
-31e^(1/3*x), 31/2 e^(1/3*x), -1e^(1/3*x), 31/4 e^(41/3*x), -31/5 e^(51/3*x),
31/3 e^(-1/3*x), -31/2 e^(-2/3*x), 31/4 e^(-41/3*x), -31/5 e^(-51/3*x),
31/1 e^(-11/3*x), +31/2 e^(-10/3*x), -1e^(-1/3*x), +31/4 e^(-41/3*x), -31/5 e^(-51/3*x),
-1/20 (60 e^(1/3*x) - 30 e^(21/3*x) + 20 e^(41/3*x) - 15 e^(51/3*x) + 12 e^(11/3*x))
]
serie_funcion := -31e^(1/3*x) + 31/2 e^(1/3*x) - 1e^(1/3*x) + 31/4 e^(41/3*x) - 31/5 e^(51/3*x)
serie_funcion_simplificada := -2520e^(1/3*x) + 1260e^(2/3*x) - 840e^(3/3*x) + 630e^(4/3*x) - 504e^(5/3*x) + 420e^(11/3*x) - 360e^(12/3*x) + 315e^(13/3*x) - 280e^(14/3*x) + 252e^(15/3*x)
Warning, unable to evaluate 1 of the 2 functions to numeric values in the region; complex values were detected

```

Figura 2.13: Cálculo del coeficiente complejo de Fourier en Maple

De igual modo, este código nos dará una gráfica estática con los valores que le establecimos Apéndice C ??.

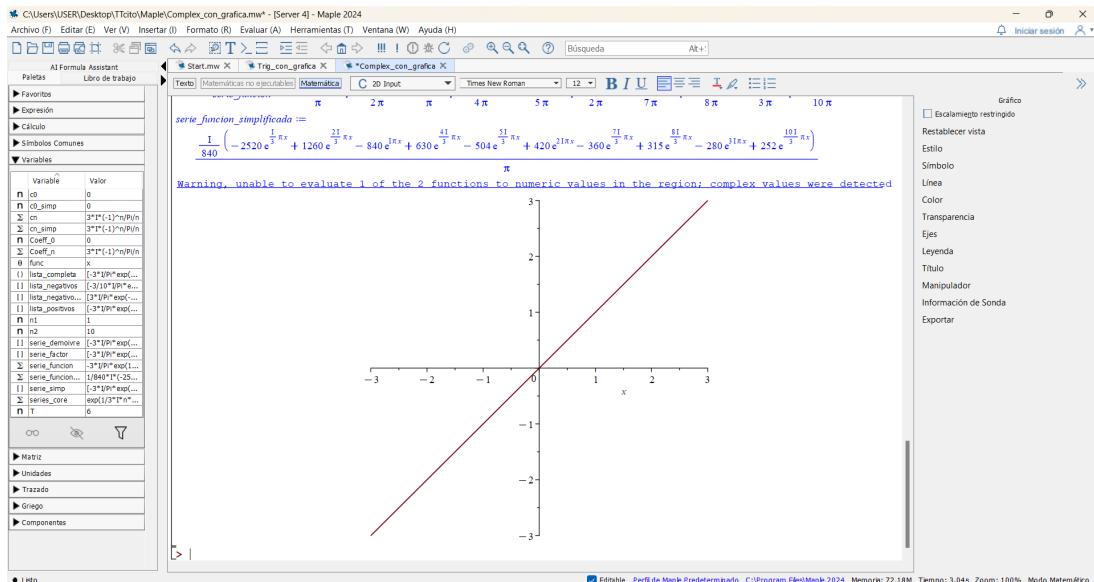


Figura 2.14: Grafica de la serie de Fourier compleja en Maple

El detalle está en que no hay forma de graficar nuestra función serie compleja ya que no puede hacer las cancelaciones entre los números imaginarios al expandir la serie, por lo tanto, la función la mantiene en el dominio de los números complejos y no la puede graficar en el plano real. Una alternativa sería graficar la serie trigonométrica que obtuvimos en ?? ya que son equivalentes.

2.1.5. Maxima

Maxima es un Sistema de Álgebra Computacional de código abierto y gratuito, derivado del sistema Macsyma desarrollado en la década de 1980. Permite

realizar cálculos simbólicos y numéricos avanzados, como derivadas, integrales, simplificación de expresiones algebraicas, resolución de ecuaciones y generación de gráficos en 2D y 3D. Maxima emplea el paradigma REPL (Read-Eval-Print Loop), proporcionando un entorno interactivo donde cada comando ingresado se procesa de inmediato, lo que facilita la verificación y corrección rápida de cálculos [9].



Figura 2.15: Logotipo de Maxima. Fuente: [9]

Si bien, este CAS al igual que Maple, cuenta con módulo que importa funciones especiales para el análisis de Fourier, este si es propio de Maxima. Sin embargo, igual que en Maple, resulta más práctico trabajar con las funciones propias de Maxima para integrar (`integrate(expr, x, a, b)`) Calcula simbólicamente la integral de `expr` con límites en `a` y `b`) así como su función para expandir una función en serie (`makelist(expr, i, i_min, i_max, step)`) Genera una lista evaluando `expr` para cada valor de `i` desde `i_min` hasta `i_max`, incrementando `i` en `step` en cada iteración).

2.1.5.1. Prueba Maxima

Para la prueba con Maxima, ejecutaremos el código en Apéndice C, que primero calculará los coeficientes de la serie trigonométrica ??.

Figura 2.16: Cálculos de coeficientes trigonométricos de Fourier en Maxima

Posteriormente, este código nos dará una gráfica dinámica con los valores que le establecimos.

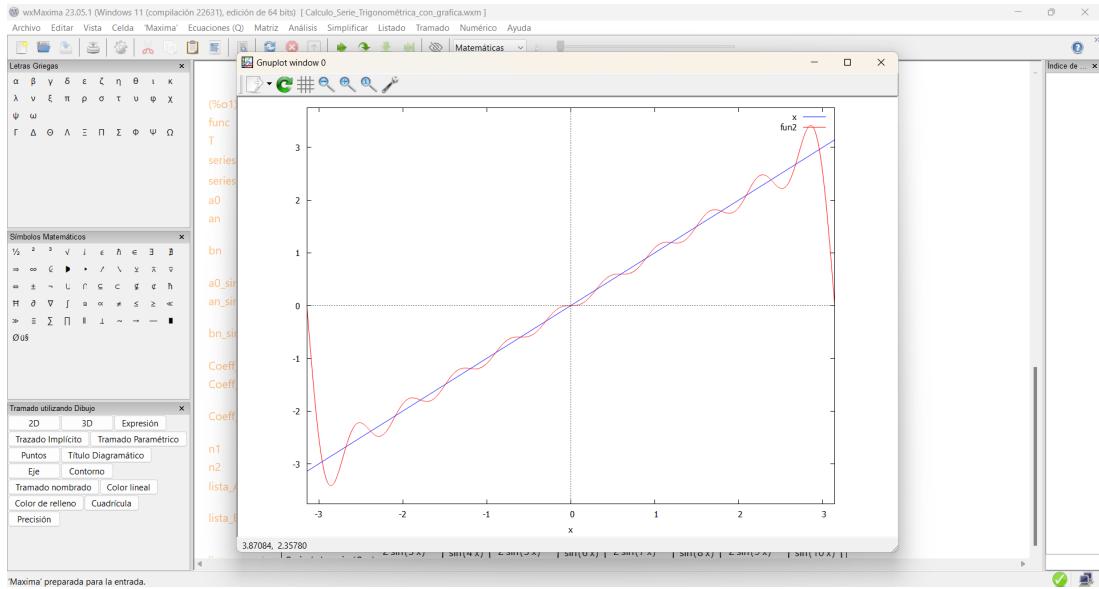


Figura 2.17: Grafica de la serie de Fourier trigonométrica en Maxima

Podemos ver que sin problema alguno se construye la gráfica, ahora, probemos haciendo el problema de la misma función pero para la serie compleja Apéndice C.

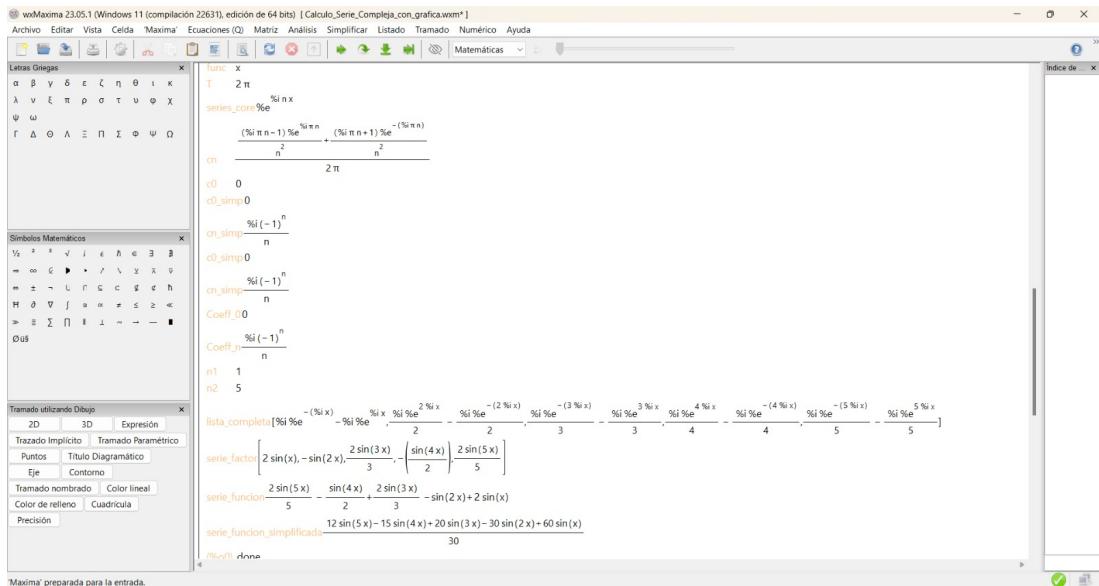


Figura 2.18: Cálculos de coeficientes compleja de Fourier en Maxima

De igual modo, este código nos dará una gráfica dinámica con los valores que le establecimos.

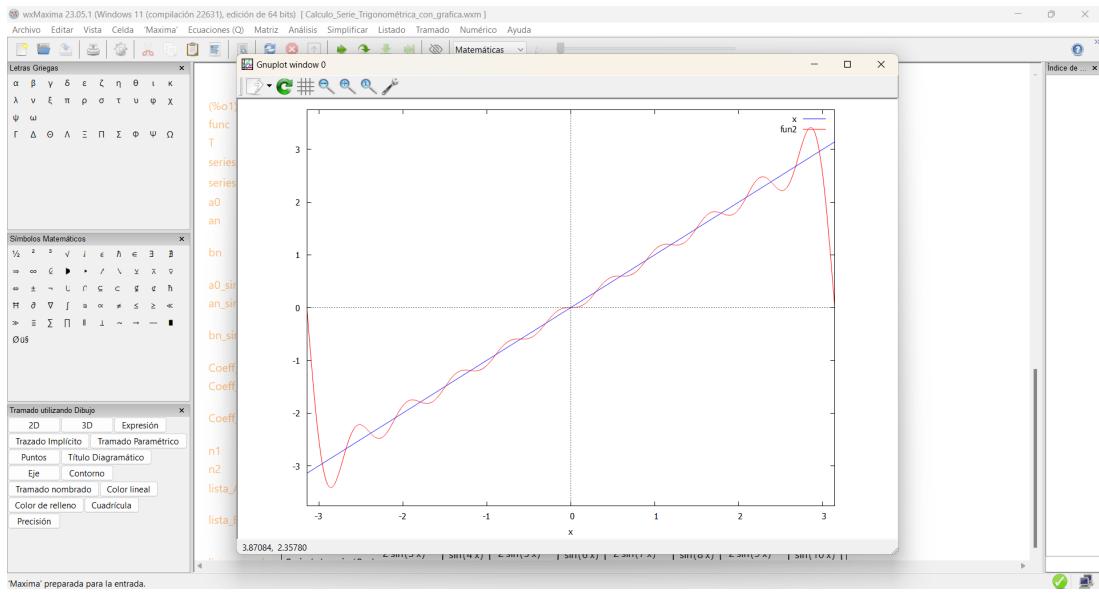


Figura 2.19: Grafica de la serie de Fourier compleja en Maxima

Podemos observar que sin problemas podemos hacer el gráfico ya que la función `demoivre()` se encarga de separar la parte real e imaginaria y con esto, hacer la gráfica equivalente. A pesar de que Maxima no cuenta con una función para declarar funciones a trozos, podemos calcular las integrales por separado o usar matrices (`matrix (fila_1, ..., fila_n)`), pero esto se verá más adelante.

2.1.6. Geogebra / Desmos

GeoGebra y Desmos son dos herramientas educativas digitales, gratuitas y de código abierto, utilizadas para la enseñanza y el aprendizaje de las matemáticas. Ambas permiten a los usuarios graficar funciones, trabajar conceptos algebraicos y geométricos de manera interactiva, facilitando la visualización y comprensión de diversos temas matemáticos. Estas presentan algunas diferencias: GeoGebra ofrece una suite más completa que incluye módulos para álgebra, geometría, cálculo y estadística, lo que la hace especialmente útil para crear construcciones geométricas dinámicas y realizar análisis más complejos [10]. Por otro lado, Desmos se destaca por su interfaz intuitiva y facilidad de uso, enfocándose principalmente en la gráfica de funciones y proporcionando herramientas sencillas para crear visualizaciones rápidas y efectivas, lo que la hace ideal para estudiantes y educadores que buscan una plataforma accesible y potente para la enseñanza de conceptos fundamentales [11].

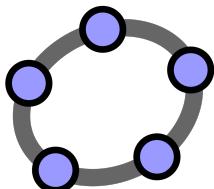


Figura 2.20: Logotipo de Geogebra. Fuente: [10]



Figura 2.21: Logotipo de Desmos. Fuente: [11]

Si bien, no tienen funciones específicas para el análisis de Fourier, ambas pueden hacer las gráficas de sumas correspondientes a los coeficientes. A continuación mostraremos la gráfica de la serie de Fourier de ?? con ambas herramientas.

2.1.6.1. Prueba Geogebra

Para la prueba con Geogebra, crearemos un slider desde 1 hasta el valor que queramos con la única condición de que este de saltos de 1 en 1, es decir, sea un slider de números enteros, también usamos la función de **Suma(Expresión, Variable, Valor Inicial, Valor Final)** en donde **Expresión** será la función de la serie, **Variable** es el n-ésimo término en la serie, **Valor Inicial** será 1 y **Valor Final** será el valor que toma el slider.

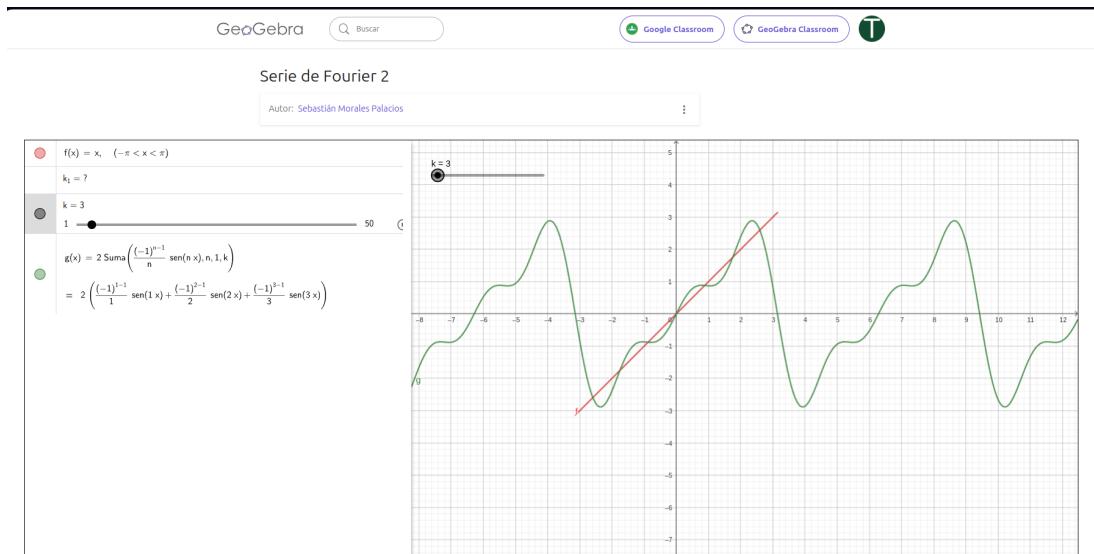


Figura 2.22: Gráfica de la serie trigonométrica de Fourier en Geogebra

Geogebra nos dará una gráfica completamente dinámica pero será un tanto pesada, ya que al añadir más de 5 términos la página comenzará a tener problemas con el dinamismo.

2.1.6.2. Prueba Desmos

Para Desmos será algo más sencillo que en Geogebra, ya que solo tendremos que computar la expresión de la serie sin definir funciones, al terminar de computar la serie nos generará en automático el slider, solo

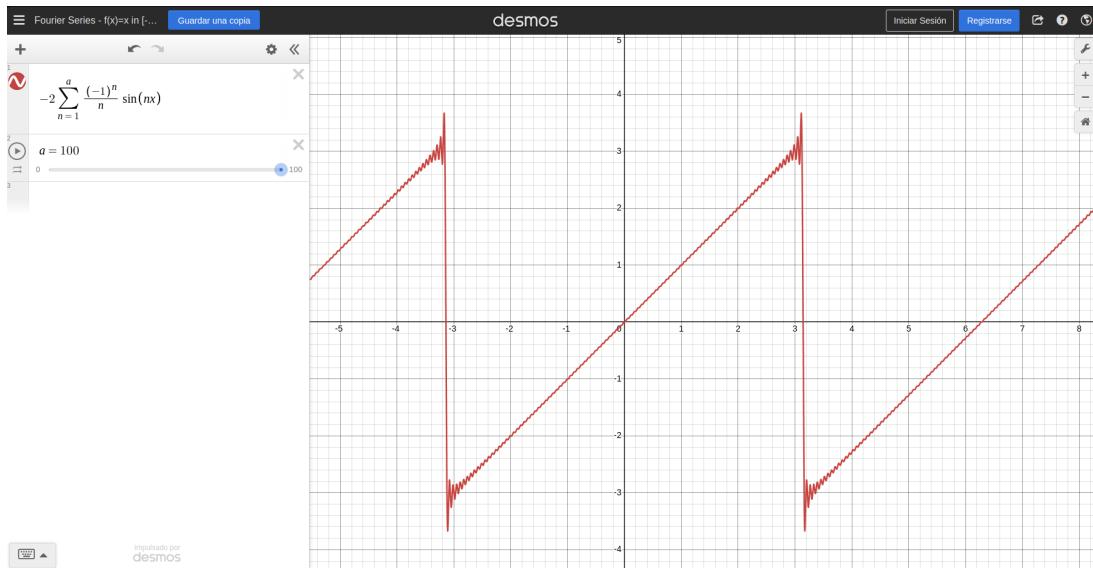


Figura 2.23: Gráfica de la serie trigonométrica de Fourier en Geogebra

A diferencia de geogebra, la gráfica tendrá un todo un tanto más simple pero será mucho más rápida, ya que podremos hacer una serie con 100 términos y no perderá fluidez como lo haría geogebra.

2.1.7. Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo [12].



Figura 2.24: Logo de Python

La versatilidad de python permiten que se creen bibliotecas para todo tipo de tareas con él, una de ellas es la de hacer cálculos matemáticos y creación de gráficos, de los cuales, podemos tomar funciones específicas para ayudarnos en el cálculo de series de Fourier.

2.1.7.1. Sympy / Matplotlib

SymPy y Matplotlib son dos bibliotecas de Python utilizadas en el dentro de las matemáticas y la visualización de datos. SymPy es una biblioteca de matemáticas simbólicas que permite realizar manipulaciones algebraicas, resolver ecuaciones, derivar e integrar funciones [13]. Por otro lado, Matplotlib es una biblioteca de visualización que permite crear gráficos estáticos, interactivos y animados [14].

Al combinar SymPy con Matplotlib, es posible realizar cálculos simbólicos complejos y representar visualmente los resultados de manera clara y efectiva.

2.1.7.2. Prueba Python con Sympy y Matplotlib

Para este caso, utilizaremos las funciones de integración numérica de Sympy para calcular los coeficientes de la serie trigonométrica y usando las funciones de graficación de Matplotlib, crearemos un gráfico para visualizar la función original con su aproximación de Fourier, para esto usamos el código en Apéndice C.

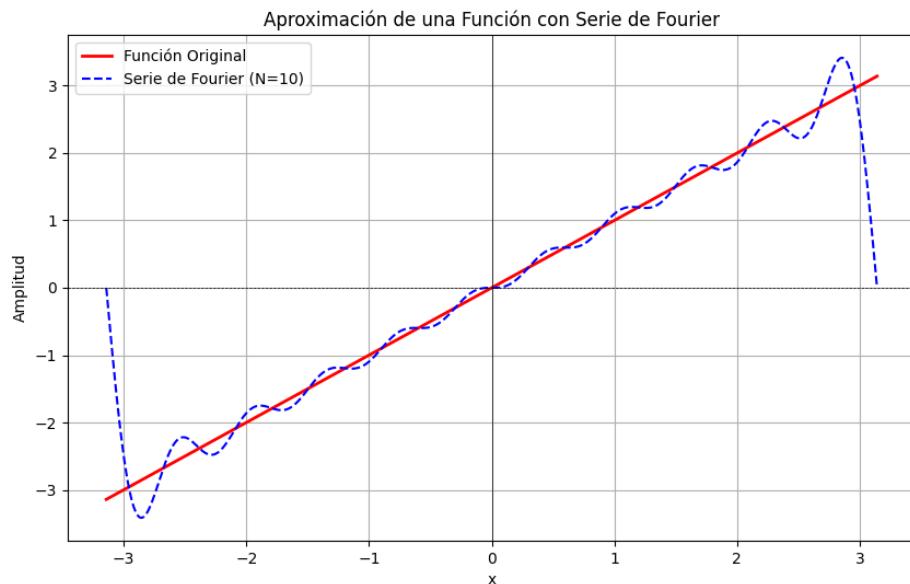


Figura 2.25: Gráfica de la serie trigonométrica y/o exponencial de Fourier en Python con Sympy y Matplotlib

Ahora, si hacemos unos cambios en el código para que calcule el coeficiente de la serie exponencial compleja , obtendremos exactamente la misma gráfica en ??, notando que Sympy es capaz de simplificar la parte imaginaria de la serie exponencial compleja para poder ver la parte real resultante en la gráfica.

2.1.7.3. Manim

Manim es una biblioteca de Python de código abierto diseñada para la creación de animaciones matemáticas de alta calidad. Desarrollada inicialmente por Grant Sanderson para su canal de YouTube "3Blue1Brown", Manim permite crear animaciones dinámicas y precisas de manera intuitiva y atractiva. La herramienta ofrece un control detallado sobre cada aspecto de la animación [15].

2.1.7.4. Prueba Python con Manim

Para poder usar a manim, debemos definir toda la expresión de la serie trigonométrica y usar las funciones que nos ofrece manim para hacer la animación como se detalla en el código en Apéndice C.

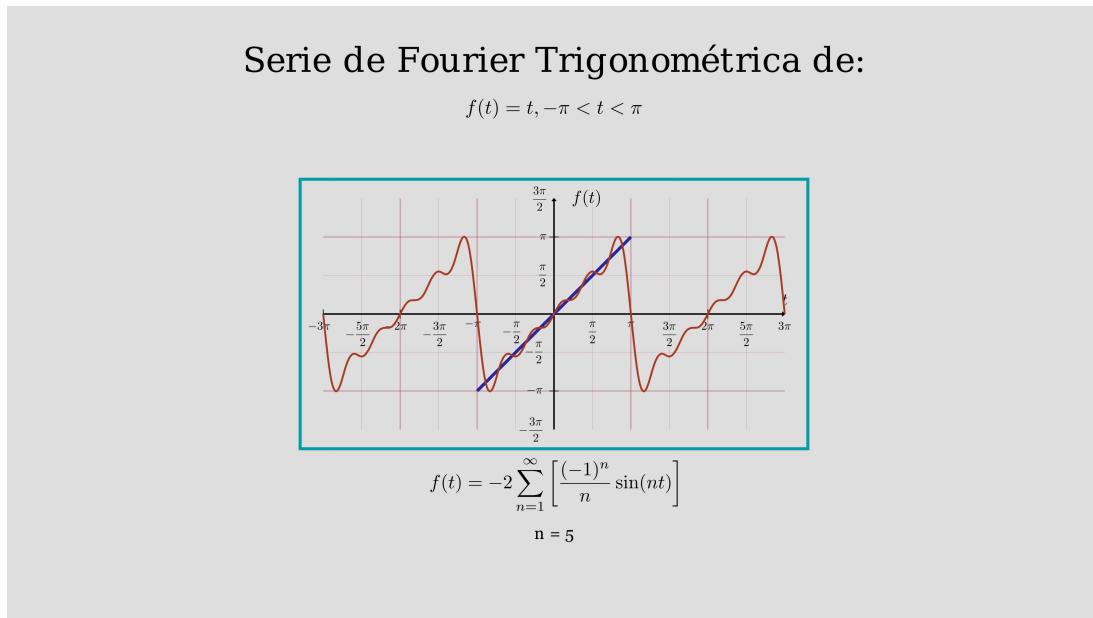


Figura 2.26: Gráfica de la serie trigonométrica de Fourier en Python con Manim

Ahora, ejecutamos el código en Apéndice C para la serie compleja y obtenemos la siguiente salida.

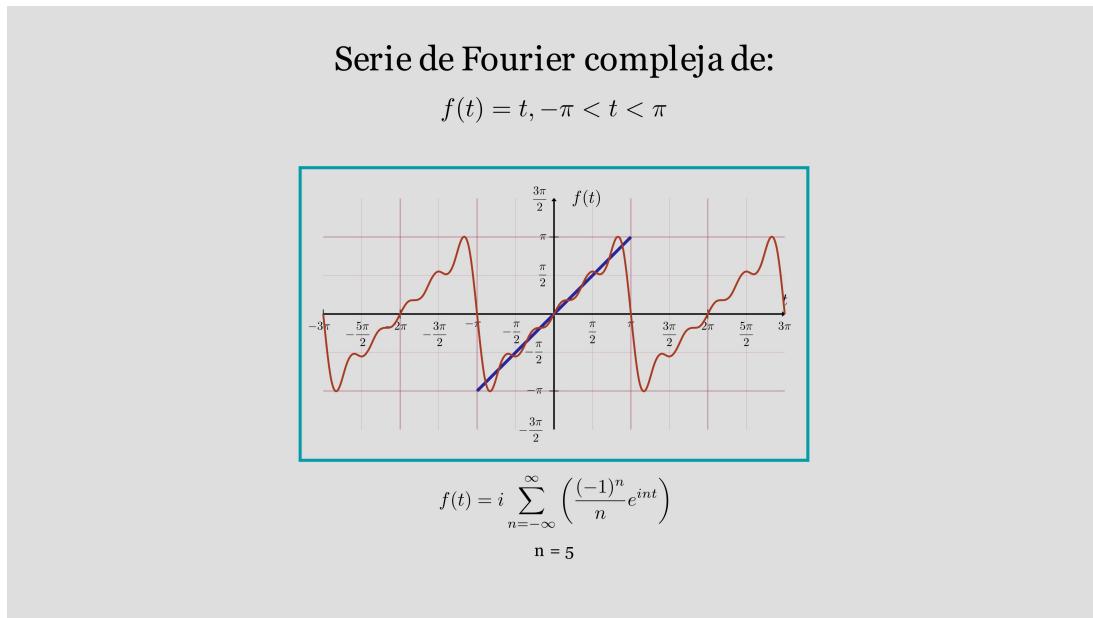


Figura 2.27: Gráfica de la serie compleja de Fourier en Python con Manim

Manim genera gráficas impecables con animaciones fluidas y limpias, sin embargo debemos recordar que los coeficientes deben ser calculados previamente ademas de que tardará bastante tiempo en renderizar el video o imagen dependiendo de la cantidad de calculo que tenga que resolver o la calidad de la imagen.

2.2. Comparativa del Funcionamiento de las Herramientas

En base a la evaluación de cada herramienta, se muestra la tabla ?? a modo de resumen.

Herramienta	Resumen del Funcionamiento	Costo (Noviembre 2024)
Wolfram Alpha	Con sus funciones especiales para series de Fourier se pueden obtener diferentes expansiones de funciones de un solo trozo o varios, esta función debe ser de un periodo de 2π , además de mostrar una gráfica estática de la aproximación en serie de Fourier. Para obtener los n -ésimos coeficientes o para funciones de otro periodo, se pueden usar sus funciones de integración.	Tiene un plan gratuito suficiente para hacer los cálculos. También cuenta con un plan mensual de MXN\$239 que habilita soluciones paso a paso, además de soporte. Está bajo una licencia comercial propietaria. No permite modificación ni redistribución; solo los usuarios que tienen una licencia activa pueden utilizar el software.
Symbolab	Cuenta con una función para calcular la serie trigonométrica de una función de un solo trozo de cualquier periodo. Para otro tipo de series, se pueden usar sus funciones de integración para obtener los resultados	Tiene un plan gratuito suficiente para hacer los cálculos. También cuenta con un plan mensual de MXN\$140 que habilita soluciones paso a paso. Al ser un servicio en linea propietario no permite redistribución o modificación del código ni el software.

Continúa en la siguiente página

Tabla 2.1 – continuación

SOFTWARE	Resumen del Funcionamiento	PRECIO
Matlab	Se puede diseñar un código que integre numéricamente para obtener los coeficientes y luego graficarlos, o tener el resultado y directamente graficarlo.	Tiene licencia gratuita para alumnos de diversas universidades. Fuera de esto, tiene planes desde USD\$99 anuales para estudiantes. Está bajo una licencia comercial propietaria. No permite modificación ni redistribución; solo los usuarios que tienen una licencia activa pueden utilizar el software.
Maple	Si bien no tiene funciones propias para series de Fourier, con sus funciones de integración es posible integrar simbólicamente cualquier función matemática (mientras sea válida) ya sea de uno o varios trozos, para obtener los coeficientes, expandir la serie y reconstruir la gráfica. Aunque tiene problemas para graficar la serie exponencial por los números complejos, si obtiene sus coeficientes.	Cuenta con planes desde USD\$75 para estudiantes. Está bajo una licencia comercial propietaria. No permite modificación, redistribución ni acceso al código fuente.
Maxima	Cuenta con funciones para trabajar con diversos tipos de problemas de series de Fourier, además de que cuenta con funciones de integración simbólica, lo que permite calcular los coeficientes y la expansión trigonométrica y la exponencial de cualquier función matemática (mientras sea válida) ya sea de uno o varios trozos sin problemas, además de poder generar un gráfico.	Es un sistema gratuito bajo la Licencia Pública General (GNU), además de ser de código abierto.

Continúa en la siguiente página

Tabla 2.1 – continuación

SOFTWARE	Resumen del Funcionamiento	PRECIO
Geogebra	Cuenta con funciones para graficar series de funciones, en las que se puede graficar la serie trigonométrica de forma dinámica, habiendo realizado previamente los cálculos de los coeficientes necesarios	Es un sistema gratuito bajo la Licencia Pública General (GNU), además de ser de código abierto.
Desmos	Al igual que con Geogebra, cuenta con una función para graficar series de funciones sobre un lienzo dinámico, con la diferencia de tener una interfaz mas sencilla y con mayor rendimiento que Geogebra	Es un sistema gratuito bajo la Licencia Pública General (GNU), además de ser de código abierto.
Matplotlib / Sympy	Si bien estas bibliotecas no tienen funciones propias para trabajar con series de Fourier, es posible crear un programa en python que use las funciones de integración numérica de Sympy para luego hacer una gráfica en matplotlib de la serie de Fourier obtenida.	Ambas son bibliotecas gratuitas licenciadas bajo BSD, además de ser de código abierto.
Manim	Si bien Manim no cuenta con ninguna función de integración, es posible crear un programa en python que genere gráficos como imagen o video de cualquier serie de Fourier, mientras ya hayamos calculado los coeficientes, los recursos que crea son altamente personalizables desde el código	Es una biblioteca gratuita bajo la Licencia MIT, además de ser de código abierto.

Tabla 2.2: Comparación de software para cálculos matemáticos y visualización de datos

De esta tabla podemos observar que si bien, hay opciones que para cálculo simbólico sumamente potentes que nos permiten calcular los coeficientes, éstas se ven limitadas en la visualización de generar una gráfica dinámica e intuitiva, mientras que las que ofrecen una mejor visualización de las series en las gráficas no siempre pueden hacer los cálculos o algunas no tienen forma de graficar cuando la serie presenta números complejos, sin mencionar que la mayoría requiere de conocimientos en programación especializado en el lenguaje de la herramienta.

CAPÍTULO 3

Marco Teórico

En este capítulo se presentará la teoría necesaria para el desarrollo de nuestra calculadora de series de Fourier. Comenzaremos explorando la historia y evolución de las series de Fourier, se detallarán las fórmulas matemáticas para comprender la descomposición de funciones periódicas en sumas de senos y cosenos, resaltando los conceptos de coeficientes de Fourier. Además, se presentan las tecnologías necesarias para la elaboración del proyecto.

3.1. Origen e historia de las series de Fourier

Uno de los problemas del que se ocuparon los matemáticos del siglo XVIII es el que se conoce con el nombre del *problema de la cuerda vibrante*. Este problema fue estudiado por D'Alambert y Euler (usando el método de propagación de las ondas) y un poco más tarde, concretamente en 1753, por Daniel Bernoulli. La solución dada por este difería de la proporcionada por los anteriores y consistió básicamente en expresar la solución del problema como superposición (en general infinita) de ondas sencillas.

Las ideas de Bernoulli fueron aplicadas y perfeccionadas por Fourier, en 1807, en el estudio de problemas relacionados con la conducción del calor. Quedaron plasmadas por escrito en el libro clásico *Théorie analytique de la chaleur*, publicado en 1822. Los razonamientos realizados por Fourier en este libro plantearon de manera inmediata numerosas controversias y cuestiones que han tenido una influencia significativa en la historia de la Matemática [16].

3.1.1. El problema de la cuerda oscilante

Uno de los problemas más interesantes que abordaron los científicos del siglo XVIII, y que aparece con frecuencia en problemas físicos relacionados con procesos oscilatorios, es el conocido como *problema de la cuerda vibrante*. Este se puede describir en su forma más elemental de la siguiente manera: Supongamos que tenemos una cuerda flexible y tensa, cuyos extremos están fijos, convenientemente, en los puntos $(0, 0)$ y $(\ell, 0)$ sobre el eje horizontal. Si la cuerda se tira de modo que su forma inicial corresponde a la curva definida por $y = f(x)$, y luego se suelta, la pregunta es: ¿Cuál será el movimiento resultante de la cuerda? Los desplazamientos de la cuerda siempre se encuentran en un mismo plano, y el vector desplazamiento es perpendicular en cualquier momento. Para describir

este movimiento se utiliza una función $u(x, t)$, donde $u(x, t)$ representa el desplazamiento vertical de la cuerda en la posición x (con $0 \leq x \leq \ell$) y en el instante t (con $t \geq 0$). El problema es determinar $u(x, t)$ a partir de $f(x)$ [17].

3.1.1.1. D'Alambert y Euler

El primer matemático que propuso un modelo adecuado para este problema fue Jean Le Rond D'Alambert [18]. Bajo diversas hipótesis (asumiendo, por ejemplo, que las vibraciones son “pequeñas”), en 1747 D'Alambert dedujo la ecuación de onda en la siguiente forma:

$$\frac{\partial^2 u(x, t)}{\partial t^2} = \frac{\partial^2 u(x, t)}{\partial x^2}, \quad 0 < x < \ell, \quad t > 0 \quad (3.1)$$

en donde:

- $u(x, t)$: Es la función que describe el desplazamiento de la onda en función de la posición x y el tiempo t .
- $\frac{\partial^2 u(x, t)}{\partial t^2}$: Segunda derivada parcial de $u(x, t)$ respecto al tiempo t , que representa la aceleración del desplazamiento de la onda.
- $\frac{\partial^2 u(x, t)}{\partial x^2}$: Segunda derivada parcial de $u(x, t)$ respecto a la posición x , que describe la curvatura espacial de la onda.

y esta debe satisfacer las condiciones siguientes:

$$\begin{aligned} u(x, 0) &= f(x), \quad 0 \leq x \leq \ell \\ \frac{\partial u(x, 0)}{\partial t} &= 0, \quad 0 \leq x \leq \ell \\ u(0, t) &= u(\ell, t) = 0, \quad t \geq 0. \end{aligned} \quad (3.2)$$

En donde la primera ecuación establece la posición inicial de la cuerda, mientras que la segunda indica que la velocidad inicial de la cuerda es cero (recordando que, una vez desplazada a la posición $f(x)$, la cuerda es liberada). La última condición expresa que, para cualquier tiempo, los extremos de la cuerda permanecen fijos. En la figura ?? La variable $u = u(x, t)$ mide el desplazamiento sobre la vertical a tiempo $t > 0$ en la posición $x \in [0, \ell]$.

Y apartir de esto, construyó la solución [19]:

$$u(x, t) = F(x + t) + G(x - t) \quad (3.3)$$

Que, estando sujeta a las condiciones de frontera ?? se podía reducir a [19]:

$$u(x, t) = F(x + t) + F(x - t) \quad (3.4)$$

siempre y cuando la función F fuese periódica, impar y diferenciable en todas partes. D'Alambert demostraba con esto que la ecuación de onda admitía “un número infinito de soluciones”.

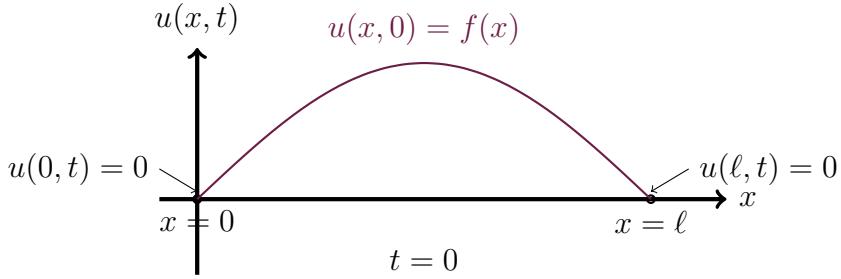


Figura 3.1: Ilustración del problema de la cuerda vibrante. *Fuente: Elaboración propia*

Euler entra en escena al criticar varias de las suposiciones que D'Alambert hiciera para encontrar su solución. Una de particular interés es la de suponer que la forma inicial de la curva o cuerda era continua y diferenciable “como una anguila”. Esta suposición para Euler, le quitaba generalidad a la solución de D' Alambert pues omitía funciones, notablemente la que representa una cuerda al ser pulsada, como se muestra en la figura ???. Euler derivó una ecuación de onda ligeramente más general que la de D'Alembert ??:

$$\frac{1}{c^2} \frac{\partial^2 u(x, t)}{\partial t^2} = \frac{\partial^2 u(x, t)}{\partial x^2} \quad (3.5)$$

con la solución [19]:

$$u(x, t) = F(x + ct) + G(x - ct) \quad (3.6)$$

que con las mismas condiciones de frontera resulta en [19]:

$$u(x, t) = F(x + ct) + F(x - ct) \quad (3.7)$$

Euler argumentaba, a diferencia de D'Alembert, que la función F quedaba determinada por la posición y velocidad inicial de la cuerda. Si denotamos por $w(x)$ y $v(x)$ a éstas, respectivamente, entonces la solución puede representarse como:

$$u(x, t) = \frac{1}{2} \left(w(x + ct) + w(x - ct) + \frac{1}{c} \int_{x-ct}^{x+ct} v(s) ds \right) \quad (3.8)$$

Las funciones w y v podían ser arbitrarias (en el sentido que podían representar cualquier curva que pudiera “dibujarse a mano”) e incluía, implícitamente, a la curva “triangular” de la figura. Como dicen los autores [20] a D'Alembert no le quitaba el sueño sacrificar realismo físico por pureza matemática; Euler, por el contrario, usaba la realidad física para contraargumentar la generalidad de las soluciones obtenidas por el francés aunque su interés no fuera tanto realismo físico sino la generalidad de las soluciones admitidas por la ecuación de onda.

3.1.1.2. Bernoulli

Otra manera de obtener la solución del problema ??, distinta a la de sus distinguidos interlocutores, fue propuesta por Daniel Bernoulli en 1753 [20] La

idea clave es obtener la solución de ?? como superposición de ondas más sencillas, concretamente aquellas que son de la forma [21]:

$$u(x) = \alpha \sin \frac{\pi x}{\ell} + \beta \sin \frac{2\pi x}{\ell} + \gamma \sin \frac{3\pi x}{\ell} + \delta \sin \frac{4\pi x}{\ell} + \dots \quad (3.9)$$

que incorporaban el hecho experimental de que cualquier modo de vibración de una cuerda puede obtenerse superponiendo modos vibratorios simples representados por cada término. Estas funciones representan, para $n = 1$, el modo fundamental, y para $n > 1$, sus armónicos. De esta manera, cualquier vibración de la cuerda puede describirse como una superposición de estos armónicos, como se puede ver en la figura ??.

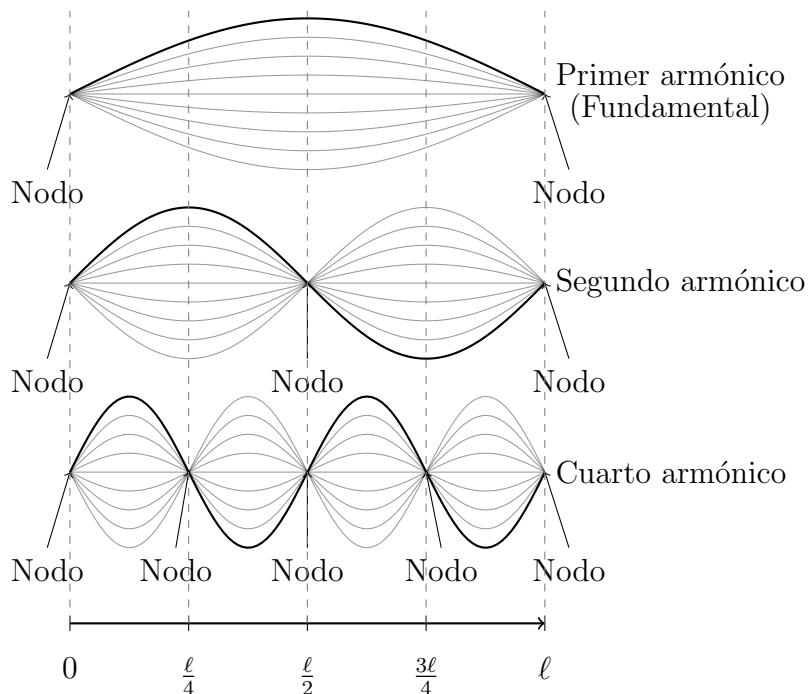


Figura 3.2: Ilustración de 3 armónicos y sus nodos de la cuerda al vibrar. *Fuente: Elaboración propia*

Si la solución propuesta por Bernouilli fuese correcta, ello obligaría a que

$$u(x, 0) = \sum_{n=1}^{\infty} C_n \sin(nx) \quad (3.10)$$

y por tanto a que

$$f(x) = \sum_{n=1}^{\infty} C_n \sin(nx), \quad \forall x \in [0, \ell], \quad (3.11)$$

para “una adecuada elección de los coeficientes C_n ”.

Pero esta fórmula no es correcta del todo pues le falta, a cada término, el producto por la función coseno (función del tiempo). Más aún, Bernoulli fue incapaz de mostrar cómo podían obtenerse los coeficientes $\alpha, \beta, \gamma, \delta, \dots$ cosa que sí hizo Euler un poco después con el método de integrar la expansión después

de multiplicarla por senos o cosenos. Es claro que la solución de Bernoulli no es tan general como la de D'Alembert pero algo que aún el gran Euler no pudo realmente entender es que la solución de Bernoulli permitía funciones iniciales más generales que, a decir de otros autores, fue tema de críticas por parte de Euler a la propuesta de D'Alembert [21]. Bernoulli y Euler mantuvieron una correspondencia epistolar productiva, y se puede afirmar que ambos dependían del otro para realizar su trabajo: Bernoulli necesitaba de Euler para orientar sus estudios matemáticos, mientras que Euler dependía de Bernoulli para comprender los fenómenos físicos que sustentaban su investigación matemática. A pesar de que partían de estudios matemáticos comunes, al final no lograron resolver ni juntos, ni de manera individual, ni desde una perspectiva experimental, donde la matemática jugara un papel metodológico predominante, ni desde el punto de vista matemático donde la física del problema guiara y definiera la teoría. Euler nunca consideró especialmente relevante el trabajo matemático de Daniel Bernoulli, ya que sus métodos eran algo imprecisos, aunque con gran intuición física. Por su parte, Bernoulli tampoco valoraba mucho los estudios matemáticos de Euler, pues estaban distantes de los experimentos. Bernoulli incluso llegó a comentar que Euler “tomaba sus pruebas de la naturaleza y no de algún principio de análisis” [21].

3.1.2. La ecuación de calor

Hubo que esperar 54 años hasta que las ideas de Bernoulli fueron tomadas en cuenta por el barón Jean Baptiste-Joseph Fourier, matemático y físico francés, quien, entre otras actividades acompañó a Napoleón, en calidad e científico, en la campaña de éste en Egipto. Allí, como secretario del Instituto de Egipto, hizo gala de gran competencia en diversos asuntos administrativos. [1]

3.1.2.1. Fourier

Al regresar a Francia, y como profesor de Análisis de la Escuela Politécnica, Fourier se interesó por la teoría de la conducción del calor en los cuerpos sólidos. En 1807 envió un artículo a la Academia de Ciencias de París, que trataba sobre dicho tema. Más concretamente, Fourier consideró una varilla delgada de longitud dada ℓ , cuyos extremos se mantienen a 0° centígrados y cuya superficie lateral está aislada. Si la distribución inicial de temperatura en la varilla viene dada por una función $f(x)$ (se supone que la temperatura de la varilla en cada sección transversal de la misma es constante), ¿Cuál será la temperatura de cualquier punto x de la varilla en el tiempo t ? Suponiendo que la varilla satisface condiciones físicas apropiadas, Fourier demostró que si $u(x, t)$ representa la temperatura en la sección x y en el tiempo t , entonces la función u debe tener la siguiente forma:

$$\frac{\partial u(x, t)}{\partial t} = \alpha^2 \frac{\partial^2 u(x, t)}{\partial x^2}, \quad 0 < x < \ell, \quad 0 < t < \infty \quad (3.12)$$

en donde:

- $\frac{\partial u}{\partial t}(x, t)$: Representa la derivada parcial de u con respecto al tiempo t , es decir, cómo cambia la temperatura $u(x, t)$ en el punto x a lo largo del tiempo t .

- $u(x, t)$: Es la función que describe la temperatura en un punto x en el espacio y en un tiempo t . Depende tanto de la posición espacial x como del tiempo t .
- α^2 : Es el coeficiente de difusión térmica, que depende del material en cuestión. Este coeficiente es constante y está relacionado con la capacidad del material para difundir el calor. La unidad de α es metros cuadrados por segundo (m^2/s).
- $\frac{\partial^2 u}{\partial x^2}(x, t)$: Representa la derivada parcial segunda de u con respecto a la posición x , es decir, cómo cambia la pendiente (o curvatura) de la temperatura a lo largo del espacio. Esta cantidad indica cómo el calor se distribuye espacialmente.

Esta ecuación modela la difusión del calor en un medio a lo largo del tiempo. La derivada en el tiempo ($\frac{\partial u}{\partial t}$) está relacionada con la derivada segunda en el espacio ($\frac{\partial^2 u}{\partial x^2}$), lo que refleja que el cambio en la temperatura con el tiempo depende de cómo está distribuido el calor en el espacio. y esta debe satisfacer las siguientes condiciones:

$$\begin{aligned} u(0, t) &= u(\ell, t) = 0, \quad 0 \leq t \leq \infty \\ u(x, 0) &= f(x), \quad 0 \leq x \leq \ell. \end{aligned} \tag{3.13}$$

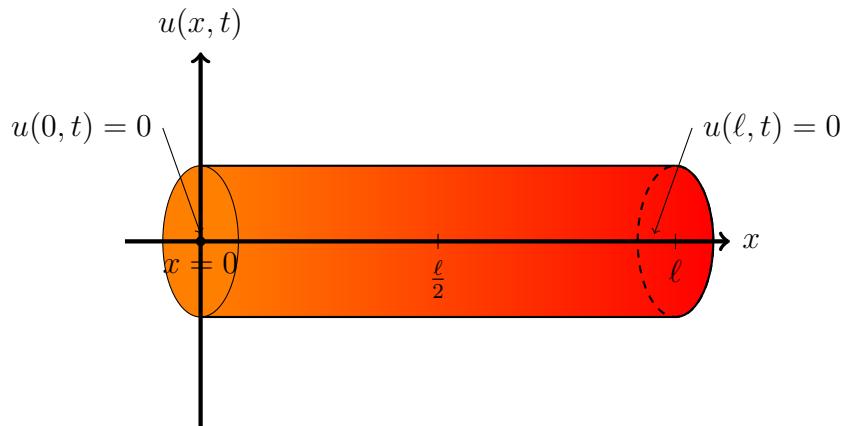


Figura 3.3: Ilustración del problema de transferencia de calor en una barra de longitud ℓ . *Fuente: Elaboración propia*

La primera condición en ?? es una Ecuación en Derivadas Parciales de segundo orden, conocida con el nombre de Ecuación del Calor. La segunda significa que la temperatura, en los extremos de la varilla, se mantiene a 0° centígrados en cualquier tiempo, mientras que la última relación representa la distribución inicial de temperatura en la varilla considerada.

Partiendo de las ideas de Bernoulli, para la ecuación de ondas, Fourier buscó las soluciones más sencillas que puede presentar la ecuación del calor: aquellas que son de la forma:

$$u(x, t) = X(x)P(t) \tag{3.14}$$

Imponiendo la condición de que tales funciones satisfagan, formalmente, dicha ecuación, obtenemos, como en el caso de la ecuación de ondas, los dos problemas siguientes de ecuaciones diferenciales ordinarias [22]:

$$\begin{aligned} X''(x) + \mu X(x) &= 0, \quad x \in (0, \ell), \quad X(0) = X(\ell) = 0 \\ P'(t) + \mu P(t) &= 0, \quad t > 0 \end{aligned} \quad (3.15)$$

Utilizando el método de separación de variables [23], Fourier propuso que la solución a la ecuación del calor podría expresarse como el producto de dos funciones, una dependiente de la posición $X(x)$ y otra del tiempo $P(t)$, como se indica en (??). Esto descompone la ecuación en dos ecuaciones diferenciales ordinarias: una espacial (??) y una temporal (??). Fourier resolvió cada una de estas ecuaciones por separado. La solución espacial $X(t)$ resulta en una serie de funciones sinusoidales que satisfacen las condiciones de frontera en los extremos de la varilla, mientras que la solución temporal $P(t)$ resulta en exponentes negativos que representan la disipación de calor en el tiempo. Así, disponemos de un procedimiento que nos permite calcular infinitas “soluciones elementales” de la ecuación del calor, a saber, las funciones de la forma $b_n v_n$, donde v_n se define como:

$$v(x, t) = \sin\left(\frac{n\pi}{\ell}x\right) e^{-\alpha^2 \frac{n^2 \pi^2}{\ell^2} kt} \quad (3.16)$$

Es trivial que, si la distribución inicial de temperatura, f , es algún múltiplo de $\sin(nx)$ (o una combinación lineal finita de funciones de este tipo), entonces la solución buscada de (??) es un múltiplo adecuado de v_n (respectivamente, una adecuada combinación lineal de funciones de esta forma).

Ahora bien, $f(x)$ no es, en general, de la forma justo mencionada, pero, y aquí demostró Fourier, como Bernouilli, una enorme intuición, ¿Será posible obtener la solución $u(x, t)$ de (??), para cualquier $f(x)$ dada, como superposición de las anteriores soluciones sencillas v_n ? Es decir, ¿Será posible elegir adecuadamente los coeficientes b_n tal que la única solución de (??) sea de la forma:

$$u(x, t) = \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi}{\ell}x\right) e^{-\alpha^2 \frac{n^2 \pi^2}{\ell^2} kt} \quad (3.17)$$

Fourier afirmó en su artículo que esto era correcto. Observemos que nuevamente llegamos a que, entonces, se debe satisfacer la relación (??). Esto plantea la misma cuestión para dos problemas completamente distintos, el problema (??) y el problema (??), en donde los coeficientes b_n toman la forma:

$$b_n = \frac{2}{\ell} \int_0^{\ell} f(x) \sin\left(\frac{n\pi x}{\ell}\right) dx \quad (3.18)$$

Los coeficientes b_n se determinan en función de la distribución inicial de temperatura $f(x)$ a través de la expresión en (??), lo que permite que la solución refleje la distribución inicial específica de la varilla. Así, la solución general toma la forma de una serie de Fourier, como se observa en (??), donde cada término

de la serie combina una función sinusoidal en el espacio con un decaimiento exponencial en el tiempo.

El artículo Fourier fue evaluado por Lagrange, Laplace y Legendre y fue rechazado por la Academia Francesa, principalmente debido a la forma en que dedujo la ecuación del calor y por la falta de rigor en sus conclusiones (según la opinión de los académicos mencionados). Sin embargo, los miembros de dicha institución reconocían la relevancia de los problemas relacionados con la propagación del calor, y los resultados teóricos que Fourier presentó mostraban gran concordancia con varios experimentos realizados previamente [16].

Debido a esto, la Academia estableció un premio sobre el tema. Dicho premio fue otorgado a Fourier en 1812, pero, a pesar de esto, los académicos continuaron criticando la falta de rigor, de modo que, aunque ganó el premio, Fourier no logró publicar su trabajo en la famosa serie “*Mémoires*” de la Academia Francesa. Con gran perseverancia, Fourier continuó trabajando en el tema, y en 1822 publicó su célebre libro *Théorie Analytique de la Chaleur*, Firmin Didot, Père et Fils, 1822, París, donde incluyó gran parte de su artículo de 1812 casi sin modificaciones. Este libro es actualmente una de las obras clásicas en matemáticas [16].

Dos años después, obtuvo el puesto de Secretario de la Academia Francesa, lo que le permitió finalmente publicar su artículo en la serie “*Mémoires*” [16].

3.2. Series de Fourier

Luego de que las soluciones para la ecuación de onda y de calor fueron desarrolladas, Joseph Fourier propuso un nuevo método para expresar funciones periódicas en función de sinusoides. Las bases de lo que hoy conocemos como series de Fourier fueron establecidas por esta propuesta.

A continuación, se describen los conceptos matemáticos en los cuales se basan las series de Fourier, ofreciendo una base firme para entenderlas y aplicarlas en el estudio de funciones que se repiten periódicamente.

3.2.1. Funciones Periódicas

Una función es llamada *función periódica* si existe una constante $T > 0$ tal que

$$f(t) = f(t \pm T) \quad (3.19)$$

para todo valor de t en el dominio de definición de $f(t)$, donde la constante T se denomina *periodo* de la función. El periodo también se puede definir como el tiempo transcurrido entre dos puntos equivalentes de una función [24].

Las funciones periódicas aparecen en muchas aplicaciones de matemáticas para problemas de física e ingeniería. Es evidente que la suma, diferencia, producto o cociente de dos funciones con período T también será una función con período T [24].

Si graficamos una función periódica $f(t)$ en un intervalo cerrado $a \leq t \leq a \pm T$, podemos obtener la gráfica completa de $f(t)$ repitiendo periódicamente la porción de la gráfica correspondiente a $a \leq t \leq a \pm T$??.

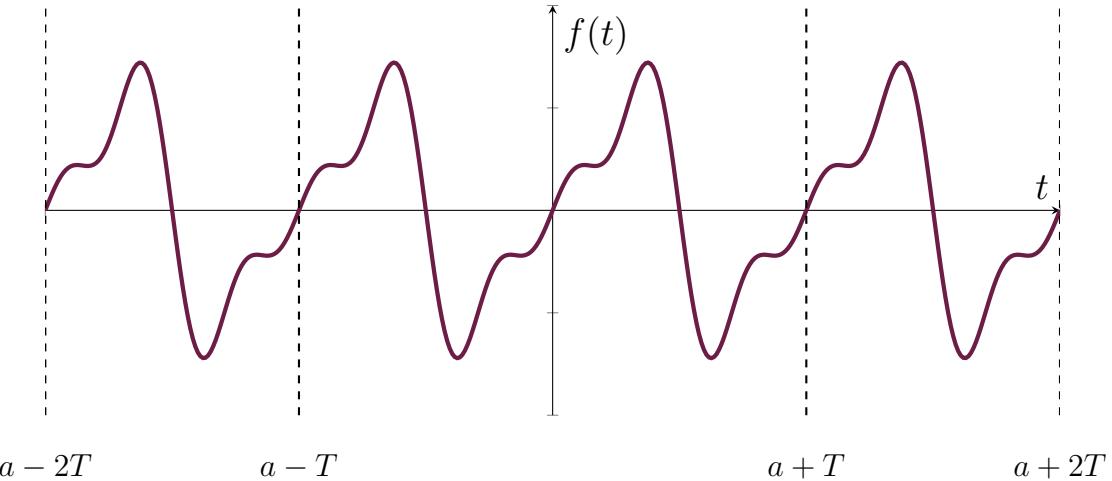


Figura 3.4: Gráfica de una función periódica $f(t)$ con periodo T . Fuente: Elaboración propia

Si T es un período de la función $f(t)$, entonces los números $2T, 3T, 4T, \dots$ también son períodos. Esto se deduce fácilmente al inspeccionar la gráfica de una función periódica o a partir de la serie de igualdades [25]

$$f(t) = f(t \pm T) = f(t \pm 2T) = f(t \pm 3T) = \dots = f(t \pm nT), \quad n = 0, \pm 1, \pm 2, \dots \quad (3.20)$$

que se obtiene mediante el uso repetido de la condición (??). Así, si T es un período, también lo es nT , donde n es cualquier entero positivo, es decir, si existe un período, no es único [25]. Además, podemos definir como el *periodo fundamental* T_0 de $f(t)$ a el valor positivo más pequeño de T para el cual se satisface (??). [26]

Ahora, consideremos la siguiente propiedad de cualquier función $f(t)$ con período T :

Si $f(t)$ es integrable en cualquier intervalo de longitud T , entonces es integrable en cualquier otro intervalo de la misma longitud, y el valor de la integral es el mismo. Es decir,

$$\int_a^{a+T} f(t) dx = \int_b^{b+T} f(t) dt \quad (3.21)$$

para cualquier a y b .

Esta propiedad es una consecuencia directa de la interpretación del área bajo la curva como la integral. De hecho, cada integral de la ecuación anterior representa el área entre la curva $f(t)$, el eje t , y las líneas verticales en los puntos extremos del intervalo. Las áreas sobre el eje t se consideran positivas y las áreas bajo el eje t como negativas. En este caso, las áreas representadas por ambas integrales son iguales debido a la periodicidad de $f(t)$ [24] ??.

A partir de ahora, cuando afirmemos que una función $f(t)$ de período T es integrable, queremos decir que es integrable en un intervalo de longitud T . Esto implica, a partir de la propiedad recién demostrada, que $f(t)$ es integrable en cualquier intervalo de longitud finita [24].

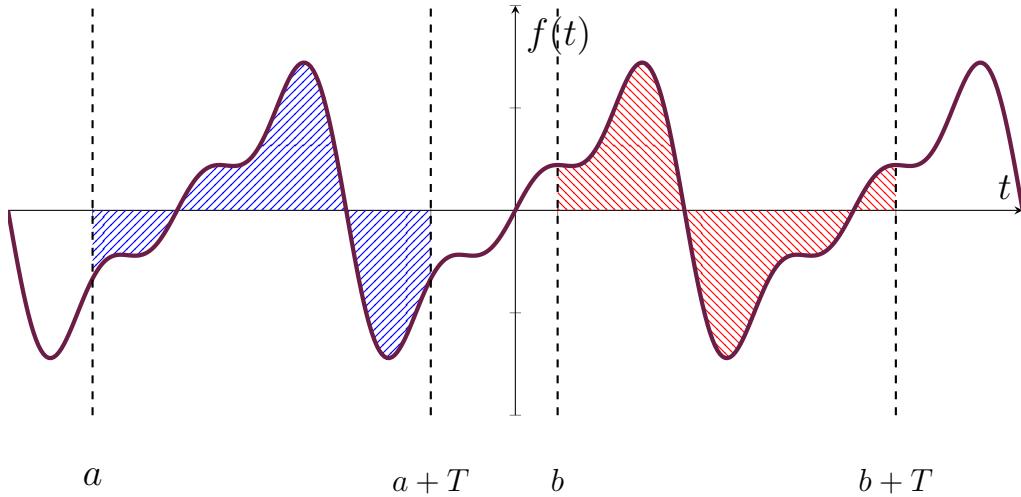


Figura 3.5: Gráfica del área bajo la curva de $f(t)$ entre los intervalos desde a hasta $a + T$ y desde b hasta $b + T$. *Fuente: Elaboración propia*

Las funciones periódicas más representativas son las funciones trigonométricas $\sin(t)$ y $\cos(t)$, ambas con periodo $T = 2\pi$ (Figura ??). Por lo tanto, de acuerdo con (??), se tiene que $\sin(t + 2\pi) = \sin(t)$ y $\cos(t + 2\pi) = \cos(t)$ [26].

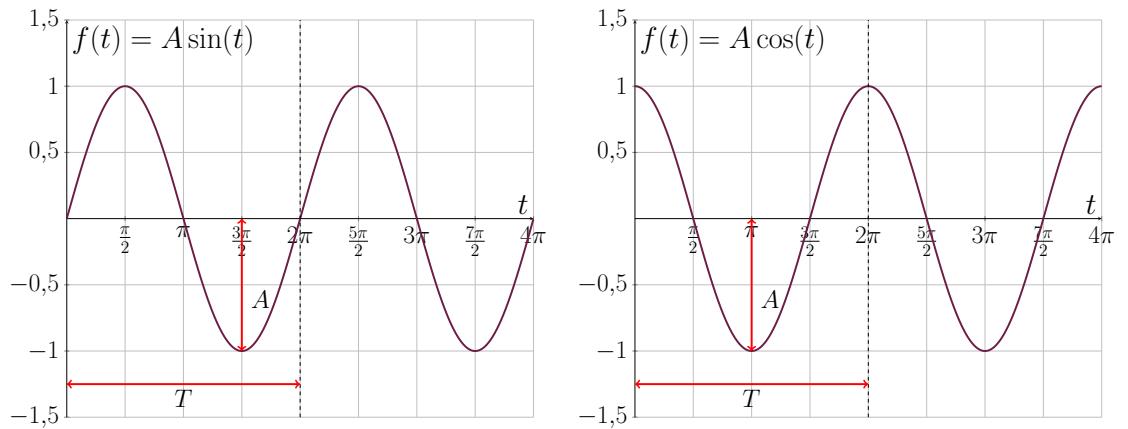


Figura 3.6: Gráfica de las funciones trigonométricas $\sin(t)$ y $\cos(t)$. *Fuente: Elaboración propia*

3.2.2. Frecuencia

La relación entre la *frecuencia* f y el *periodo* T de una función oscilatoria está dada por

$$f = \frac{1}{T} \quad (3.22)$$

lo cual indica el número de oscilaciones de la función por unidad de tiempo. La unidad de medida de la frecuencia es el hertz (Hz). Un hertz quiere decir que una función oscila una vez por segundo [26]. La Figura ?? muestra una función senoidal con tres frecuencias diferentes.

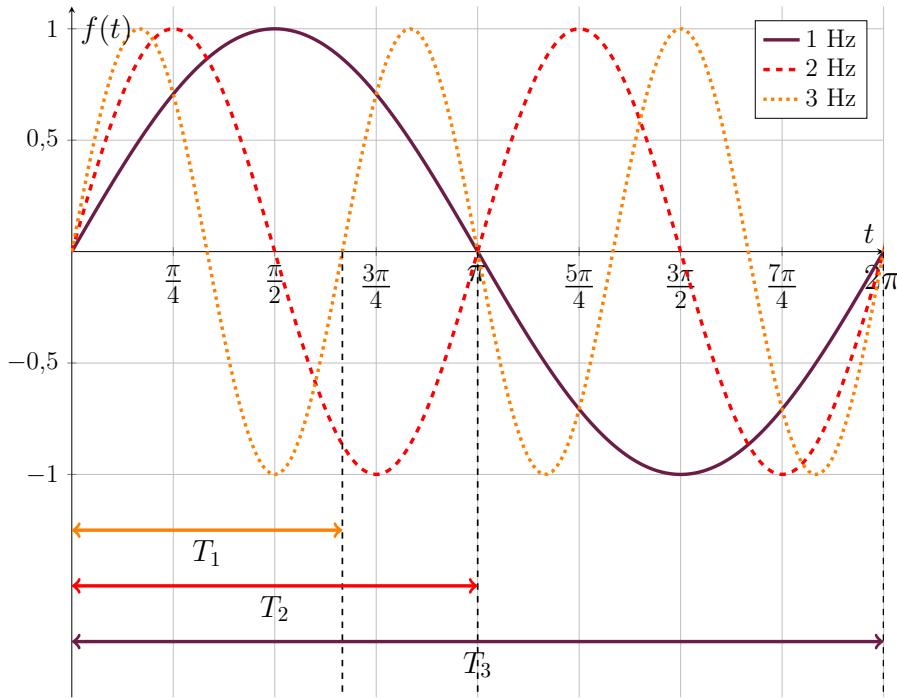


Figura 3.7: Función $\sin(t)$ con diferentes frecuencias. Fuente: Elaboración propia

La *frecuencia angular*, usualmente denotada por ω_0 , se define como la razón de cambio del desplazamiento angular θ durante la oscilación (o rotación) [26]:

$$\frac{d\theta}{dt} = \omega_0 = 2\pi f = \frac{2\pi}{T}. \quad (3.23)$$

La frecuencia angular se mide comúnmente en radianes por segundo (rad/s). La Figura?? ejemplifica la relación de una onda senoidal en el intervalo $[0, 2\pi]$ con una frecuencia angular ω_0 . [26]

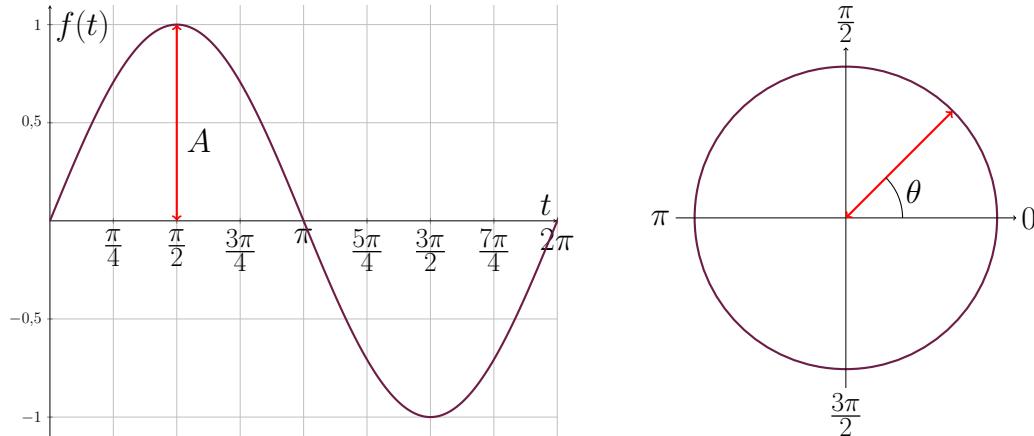


Figura 3.8: Relación de la frecuencia angular entre la función $\sin(t)$ con el círculo unitario. Fuente: Elaboración propia

3.2.3. Armónicos

La función periódica de mayor importancia es:

$$f(t) = A \sin(\omega_0 t + \varphi) \quad \text{o} \quad f(t) = A \cos(\omega_0 t + \varphi) \quad (3.24)$$

Cabe señalar que, dado que las funciones seno y coseno solo se diferencian por un deslizamiento de fase, este movimiento podría modelarse usando la función coseno o la función seno, en donde A , ω_0 , y φ son constantes. En una función armónica, la *amplitud* A representa el valor máximo de la oscilación. La *frecuencia angular* ω_0 determina la rapidez con la que se realiza la oscilación y viene dada por $\frac{2\pi}{T}$, recordando que T es el periodo de la función, y la *fase inicial* φ indica el desplazamiento de la función en el tiempo al inicio de la observación, es decir, indica con qué desfase comienza. ?? [27].

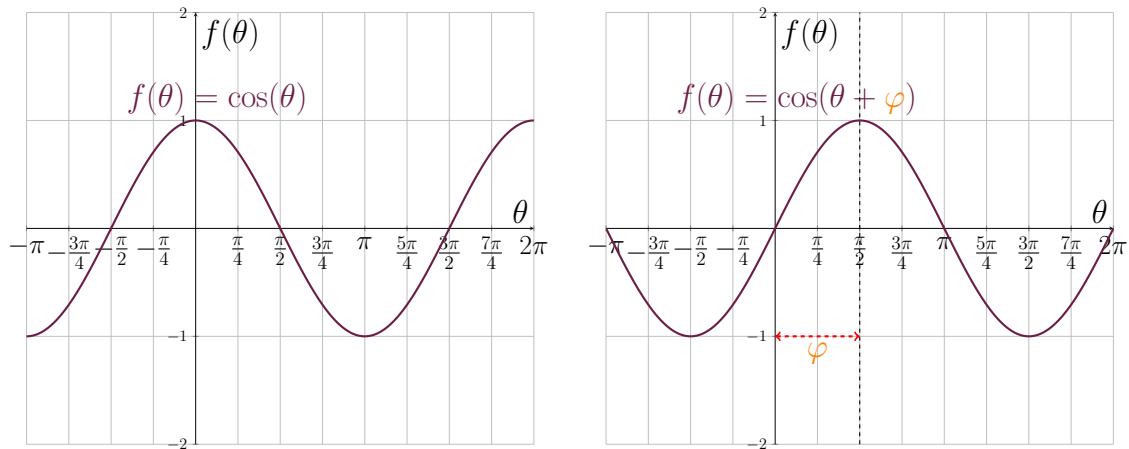


Figura 3.9: Una función coseno y la misma función coseno desplazada hacia la izquierda por un ángulo φ . Fuente: *Elaboración propia*

3.2.4. Funciones Pares e Impares

Más adelante se verá como es que el hecho de que una función $f(t)$ sea par o impar puede ahorrar diversos cálculos, comenzamos definiendo que una función es par si se cumple que: [28]

$$f(t) = f(-t), \quad -L < t < L \quad (3.25)$$

En la Figura ?? se muestra un ejemplo de una función para en donde se puede observar la definición (??)

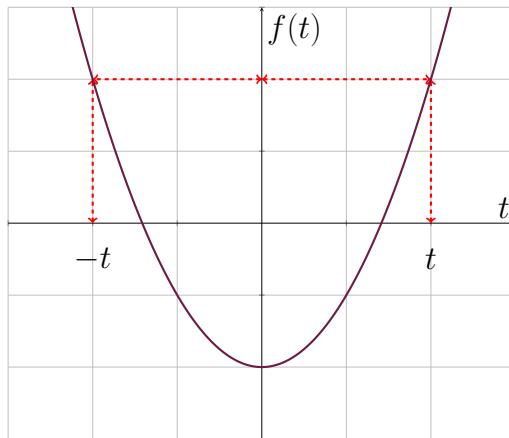


Figura 3.10: Función $f(t) = t^2$, una función par. *Fuente: Elaboración propia*

Ahora, decimos que una función es impar si se cumple que: [28]

$$f(t) = -f(-t), \quad -L < t < L \quad (3.26)$$

En la Figura ?? se muestra un ejemplo de una función para en donde se puede observar la definición (??)

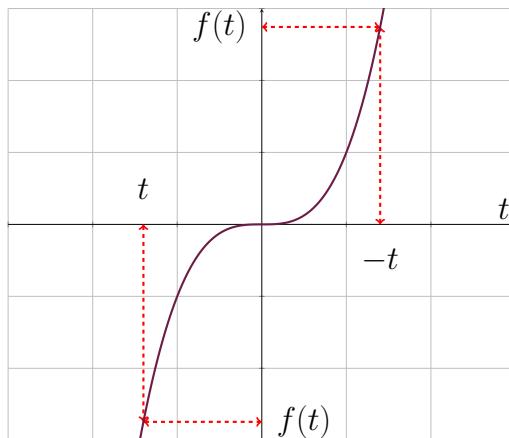


Figura 3.11: Función $f(t) = t^3$, una función impar. *Fuente: Elaboración propia*

Podemos decir entonces que en un intervalo simétrico $-L < t < L$, la gráfica de una función par tiene simetría respecto al eje y, mientras que la gráfica de una función impar tiene simetría en relación con el origen. Además, las funciones pares e impares cumplen con los siguientes teoremas [28]:

- a) El producto de dos funciones pares es par.
- b) El producto de dos funciones impares es par.
- c) El producto de una función par y una impar es impar.
- d) La suma (resta) de dos funciones pares es par.
- e) La suma (resta) de dos funciones impares es impar.

f) Si f es par, entonces $\int_{-a}^a f(t) dt = 2 \int_0^a f(t) dt$.

g) Si f es impar, entonces $\int_{-a}^a f(t) dt = 0$.

En la Figura ?? podemos observar como es que se cumplen los teoremas f y g, ya que en la primera el área de la función desde $-a$ hasta 0 es el negativo a su contraparte que se encuentra desde 0 hasta a , lo que significa que el área conjunta desde $-a$ hasta a se anulará dando 0, por otra parte, la segunda imagen nos muestra que el área desde $-a$ hasta 0 es la misma que la que hay desde 0 hasta a , lo que implica que el área desde $-a$ hasta a es lo doble de el área desde 0 hasta $\pm a$.

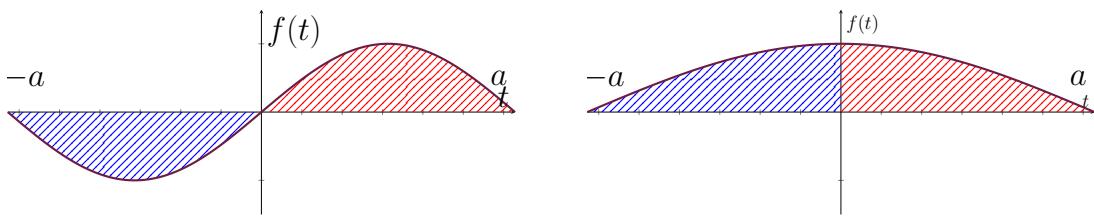


Figura 3.12: Ejemplo de los teoremas f y g de funciones pares e impares *Fuente: Elaboración propia*

3.2.5. Funciones suaves y funciones suaves a trozos

Una función $f(t)$ se dice que es *suave* en el intervalo $[a, b]$ si tiene una derivada *continua* en $[a, b]$. En términos geométricos, esto significa que la dirección de la tangente cambia de forma *continua*, sin saltos, mientras se desplaza a lo largo de la curva $f(t)$ [ver Figura ??]. Así, el gráfico de una función suave es una curva suave sin “saltos”. [24]

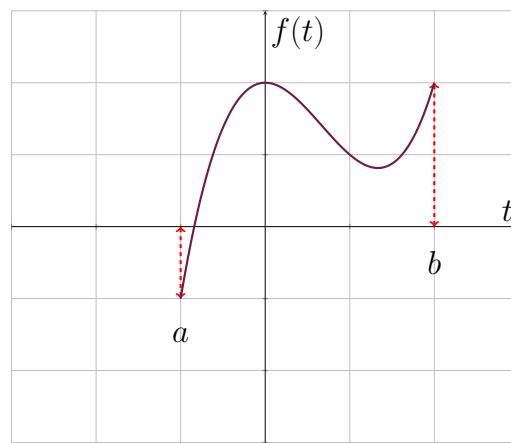


Figura 3.13: Ejemplo de una función suave. *Fuente: Elaboración propia*

La función $f(t)$ se dice que es *suave a trozos* en el intervalo $[a, b]$ si tanto $f(t)$ como su derivada son continuas en $[a, b]$, o si tienen solamente un número finito de discontinuidades de salto en $[a, b]$. Es fácil ver que el gráfico de una función suave por partes es o bien una curva continua o una curva discontinua que puede tener un número finito de saltos (en las cuales la derivada tiene saltos). A medida

que nos acercamos a cualquier discontinuidad o salto (desde un lado o el otro), la dirección de la tangente se aproxima a una posición límite definida, ya que la derivada sólo puede tener discontinuidades de salto. Las figuras ?? ilustran las gráficas de funciones suaves a trozos continuas y discontinuas. [24]

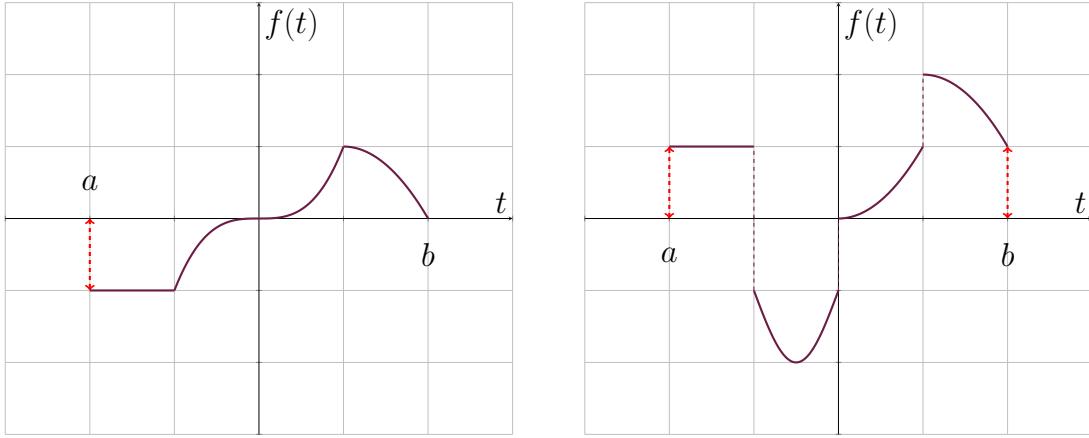


Figura 3.14: Ejemplo de una función suave a trozos completamente continua y una función a trozos con discontinuidades (saltos). *Fuente: Elaboración propia*

Entonces, una función continua o discontinua $f(t)$ definida en todo el eje t se dice que es *suave a trozos* si, en cualquier intervalo finito, se comporta de manera que su derivada es continua o tiene pocas discontinuidades. Esto significa que la función puede tener ciertos “saltos” o cambios bruscos en su forma, pero entre estos puntos, su comportamiento es regular y suave. Este concepto es útil para funciones periódicas, que se repiten en ciclos. Toda función $f(t)$ suave a trozos (ya sea continua o discontinua) está acotada, lo que implica que no puede crecer indefinidamente y tiene una derivada acotada en todas partes, excepto en sus puntos de discontinuidad (es decir, en los puntos donde la función cambia abruptamente o tiene saltos), donde la derivada $f'(t)$ no existe. [24]

3.2.6. Funciones Ortogonales

En álgebra lineal, el producto escalar (también conocido como producto interno o producto punto) entre dos vectores **a** y **b** se basa en la proyección de un vector sobre el otro. Esto permite medir “cuánto el vector **a** apunta en la misma dirección que el vector **b**”. Si ambos vectores apuntan en direcciones similares, el producto escalar es positivo; en cambio, si apuntan en direcciones opuestas, el producto escalar será negativo. Cuando los vectores son ortogonales, su producto escalar resulta en cero, lo cual indica que no se pueden representar uno en función del otro. [29]

El concepto de ortogonalidad de funciones es análogo al de perpendicularidad de vectores; es decir, funciones ortogonales proporcionan información mutuamente excluyente. Por lo tanto, se puede decir que dos funciones diferentes $f_1(t)$ y $f_2(t)$ son ortogonales en el intervalo $a < t < b$ cuando su producto interno es cero:

$$\langle f(t)_1, (t)_2 \rangle = \int_a^b f_1(t) f_2(t) dt = 0. \quad (3.27)$$

En general, se dice que el conjunto de funciones de valor real $\{\phi_0(t), \phi_1(t), \phi_2(t), \dots, \phi_n(t)\}$ es ortogonal en el intervalo $[a, b]$ si [28]

$$\langle \phi_n(t), \phi_m(t) \rangle = \int_a^b \phi_n(t) \phi_m(t) dt = 0, \quad m \neq n. \quad (3.28)$$

3.2.6.1. Ortogonalidad del seno y del coseno

Ahora, consideremos que tenemos conjuntos de funciones senoidales y cosenoidales recordando que $\omega_0 = \frac{2\pi}{T}$, a través del cálculo integral podemos encontrar que:

$$\int_{-T/2}^{T/2} \cos(m\omega_0 t) dt = 0, \quad \text{para } m \neq 0 \quad (3.29)$$

$$\int_{-T/2}^{T/2} \sin(m\omega_0 t) dt = 0, \quad \text{para todo valor de } m \quad (3.30)$$

$$\int_{-T/2}^{T/2} \cos(m\omega_0 t) \cos(n\omega_0 t) dt = \begin{cases} 0, & m \neq n \\ T/2, & m = n \neq 0 \end{cases} \quad (3.31)$$

$$\int_{-T/2}^{T/2} \sin(m\omega_0 t) \sin(n\omega_0 t) dt = \begin{cases} 0, & m \neq n \\ T/2, & m = n \neq 0 \end{cases} \quad (3.32)$$

$$\int_{-T/2}^{T/2} \sin(m\omega_0 t) \cos(n\omega_0 t) dt = 0, \quad \text{para todo valor de } m \text{ y } n \quad (3.33)$$

Estas relaciones demuestran que el conjunto de funciones $\{1, \cos \omega_0 t, \cos 2\omega_0 t, \dots, \cos n\omega_0 t, \dots, \sin \omega_0 t, \sin 2\omega_0 t, \dots, \sin n\omega_0 t, \dots\}$ son ortogonales en el intervalo $-\frac{T}{2} < t < \frac{T}{2}$.

3.2.7. Serie Trigonométrica de Fourier

Como ya vimos anteriormente, las series de Fourier, fueron introducidas por el matemático francés Jean Baptiste Joseph Fourier con el objetivo de aproximar funciones periódicas utilizando un conjunto de funciones ortogonales como $\{1, \cos \omega_0 t, \cos 2\omega_0 t, \dots, \cos n\omega_0 t, \dots, \sin \omega_0 t, \sin 2\omega_0 t, \dots, \sin n\omega_0 t, \dots\}$ [28].

La idea fundamental detrás de las series de Fourier es descomponer una función periódica en términos de una suma infinita de funciones básicas, senos y cosenos, cuyas frecuencias son múltiplos de la frecuencia de la función original. La ortogonalidad de las funciones seno y coseno, usadas como “bases”, es fundamental porque permite descomponer la función original en componentes que representan diferentes armónicos o frecuencias sin que estos interfieran entre sí, asegurando que cada término de la serie tenga una contribución bien definida. Esta descomposición permite analizar las propiedades de una función o señal y facilita la síntesis de objetos o fenómenos.

3.2.7.1. Series de Fourier de una función con período T

Si $f(t)$ es una función periódica de periodo T ??, se puede expresar mediante una serie trigonométrica como:

$$f(t) = \frac{a_0}{2} + a_1 \cos(\omega_0 t) + a_2 \cos(2\omega_0 t) + \dots + b_1 \sin(\omega_0 t) + b_2 \sin(2\omega_0 t) + \dots \quad (3.34)$$

o de forma general como [25]:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)) \quad (3.35)$$

donde $\omega_0 = \frac{2\pi}{T}$ es la frecuencia angular fundamental y los coeficientes a_0 , a_n y b_n son los coeficientes de Fourier dados por (??) [25]:

$$a_0 = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) dt, \quad a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(n\omega_0 t) dt, \quad b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(n\omega_0 t) dt \quad (3.36)$$

La ecuación (??) describe cómo una función periódica arbitraria puede representarse como una suma infinita de componentes sinusoidales con diferentes frecuencias. El componente de la frecuencia $\omega_n = n\omega_0$ se denomina el n -ésimo armónico de la función periódica, donde este n -ésimo componente os dice cuánto de la n -ésima frecuencia está presente o aporta a la función original.

Otra forma muy común de representar una función $f(t)$ en una serie de Fourier es cuando el periodo T se encuentra definido en un intervalo de $-p$ a p , que es completamente equivalente a cuando está definida sobre $-\frac{T}{2}$ a $\frac{T}{2}$, para este caso, la fórmula de la serie sigue siendo la misma establecida previamente en (??), lo que cambia levemente es como se definen nuestros coeficientes: [28]

$$a_0 = \frac{1}{p} \int_{-p}^p f(t) dt, \quad a_n = \frac{1}{p} \int_{-p}^p f(t) \cos(n\omega_0 t) dt, \quad b_n = \frac{1}{p} \int_{-p}^p f(t) \sin(n\omega_0 t) dt \quad (3.37)$$

donde también cambia un poco como se define nuestra frecuencia fundamental $\omega_0 = \frac{\pi}{p}$. [28]:

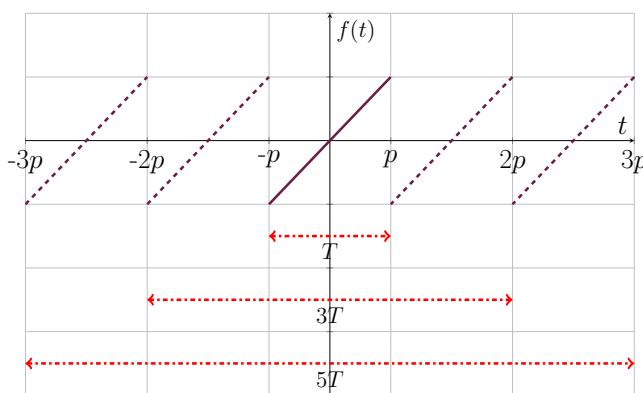


Figura 3.15: Gráfica de una función periódica $f(t)$ con periodo T , repitiéndose en intervalos de T , $2T$, y $3T$. Fuente: Elaboración propia

3.2.7.2. Criterios para funciones pares e impares

Como vimos anteriormente, el que una función sea par o impar podrá ahorrar potenciales cálculos, en primera, podemos decir que si nuestra función $f(t)$ es

una función par, entonces los únicos coeficientes que será necesario calcular serán los coeficientes a_0 y a_n (??). Pero, ¿qué pasa con el coeficiente b_n ? Recordemos que el coeficiente b_n en la serie de Fourier está dado por la integral:

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(\sin(n\omega_0 t)) dt$$

Dado que $f(t)$ es par y $\sin(\sin(n\omega_0 t))$ es una función impar, al multiplicarlos obtenemos una función impar [28], como ya vimos anteriormente. Entonces, esto implica que cuando integremos esta función impar resultante sobre un intervalo simétrico alrededor del origen (como $[-\frac{T}{2}, \frac{T}{2}]$), la integral resultante será cero:

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(\sin(n\omega_0 t)) dt = 0$$

Por lo tanto, todos los coeficientes b_n se cancelan automáticamente cuando $f(t)$ es una función par, ya que la contribución de los términos de seno se anula debido a la simetría impar del integrando. Esto simplifica la serie de Fourier, ya que solo es necesario calcular los coeficientes a_0 , correspondiente al promedio de la función y a_n , correspondiente a los términos de coseno, que son funciones pares.

Algo similar sucede cuando nuestra función $f(t)$ es una función impar, en este caso el único coeficiente que sobrevive es el coeficiente b_n . Sabemos que el coeficiente a_n en la serie de Fourier se calcula mediante la integral:

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(n\omega_0 t) dt$$

Dado que $f(t)$ es una función impar y $\cos(n\omega_0 t)$ es una función par, el producto $f(t) \cos(n\omega_0 t)$ será una función impar. Esto es porque la multiplicación de una función impar con una función par produce una función impar. [28]

Al integrar una función impar en un intervalo simétrico alrededor del origen, como $[-T/2, T/2]$, la integral resultará en cero. Esto implica que:

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(n\omega_0 t) dt = 0$$

Por un razonamiento análogo, el coeficiente a_0 , que representa el promedio de la función y se calcula mediante la integral:

$$a_0 = \frac{2}{T} \int_{-T/2}^{T/2} f(t) dt$$

también será igual a cero, ya que la función $f(t)$ es impar y, al integrarla sobre un intervalo simétrico, la integral se anula.

Por lo tanto, cuando $f(t)$ es una función impar, todos los coeficientes a_n y a_0 se cancelan automáticamente, ya que la contribución de los términos de seno y del promedio de la función se anula debido a la simetría impar del integrando. Así, en este caso, solo es necesario calcular los coeficientes b_n , correspondientes a los términos de coseno, los cuales son funciones pares y, por lo tanto, sobreviven

en la serie de Fourier.

Entonces, podemos decir que la serie de Fourier para una función $f(t)$ cuando esta función es **par** es:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) \quad (3.38)$$

Y la serie de Fourier para una función $f(t)$ cuando esta función es **ímpar** es:

$$f(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega_0 t), \quad (3.39)$$

donde los coeficientes de Fourier cuando se mantienen igual ??, solo es cuestión de aplicar lo previamente visto para saber cuando calcular los necesarios:

3.2.8. Extensiones de Medio Intervalo

En la sección anterior observamos que para desarrollar una función $f(t)$ en una serie de Fourier debía estar declarada en un intervalo simétrico $-\frac{T}{2}, \frac{T}{2}$, sin embargo, en muchos casos nos interesa representar una función definida para $0 < x < p$, es decir, comenzando en 0 a algún punto positivo p mediante una serie trigonométrica. Este tipo de casos se conocen como *extensiones de medio rango*. De aquí existen 3 posibilidades, asumir que la función a expandir en serie de Fourier es par, impar o periódica [30].

3.2.8.1. Extensión Par (Serie de Cosenos)

La extensión par hace que la función sea simétrica respecto al eje y (Figura ??), permitiendo extender el análisis a un intervalo completo de manera simétrica.

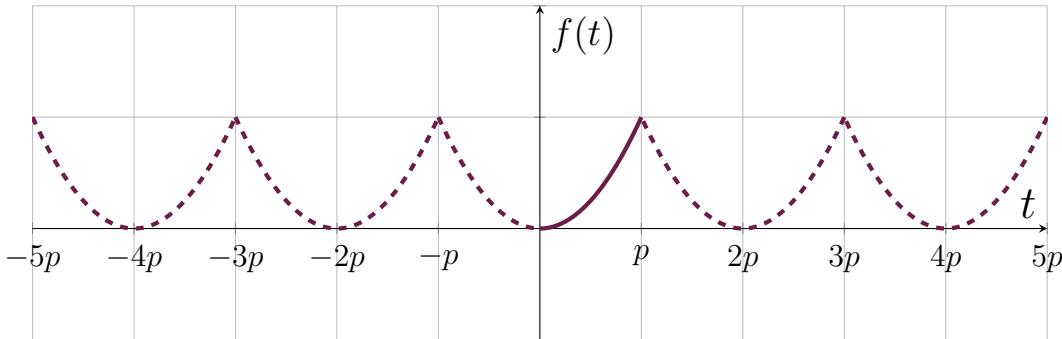


Figura 3.16: Extensión par en cosenos de una función $f(t)$. Fuente: Elaboración propia

Podemos observar que la sección de la función que se repite, es decir, el periodo de la función se encuentra de $-p$ a p , por lo tanto su periodo es de $2p$ y recordando que $\omega_0 = \frac{2\pi}{\text{periodo}}$ es la frecuencia angular fundamental, para este caso podemos decir que $\omega_0 = \frac{2\pi}{2p} \therefore \omega_0 = \frac{\pi}{p}$.

Para la extensión par, la serie de Fourier se define de la misma manera que cuando tenemos una función par $f(t)$ (??):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) \quad (3.40)$$

Pero los coeficientes se verán afectados de la siguiente forma: [30]

$$a_0 = \frac{2}{p} \int_0^p f(t) dt, \quad a_n = \frac{2}{p} \int_0^p f(t) \cos(n\omega_0 t) dt, \quad (3.41)$$

3.2.8.2. Extensión Impar (Serie de Senos)

La extensión impar hace que la función sea simétrica respecto al origen (Figura ??), permitiendo analizarla en un intervalo completo con una continuidad en la transición de signo.

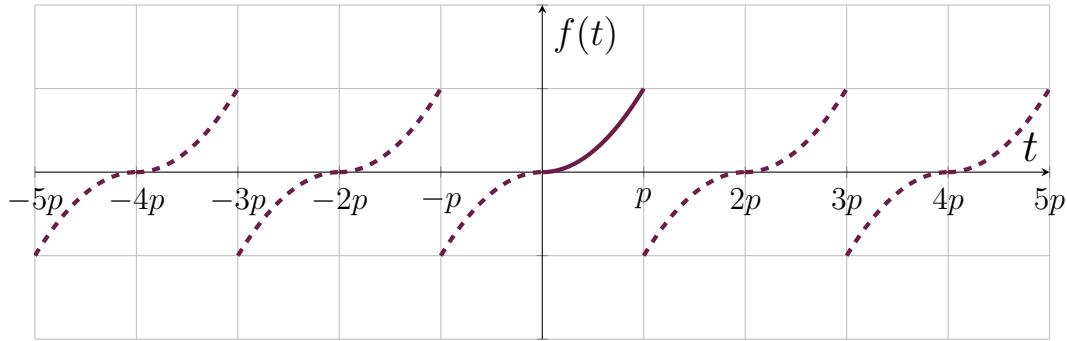


Figura 3.17: Extensión impar en senos de una función $f(t)$. Fuente: Elaboración propia

Al igual que con la extensión en serie de senos, el rango en donde se repite la función (su periodo) es igual de $2p$, por lo tanto su frecuencia angular se define para este caso de igual manera como $\omega_0 = \frac{\pi}{p}$.

Para la extensión impar, la serie de Fourier se define de la misma manera que cuando tenemos una función impar $f(t)$ (??):

$$f(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega_0 t), \quad (3.42)$$

Pero el coeficiente se verá afectado de la siguiente forma: [30]

$$b_n = \frac{2}{p} \int_0^p f(t) \sin(n\omega_0 t) dt \quad (3.43)$$

3.2.8.3. Extensión Periódica

La extensión periódica replica la función en intervalos sucesivos, creando una función periódica que se extiende indefinidamente en ambos sentidos (Figura ??).

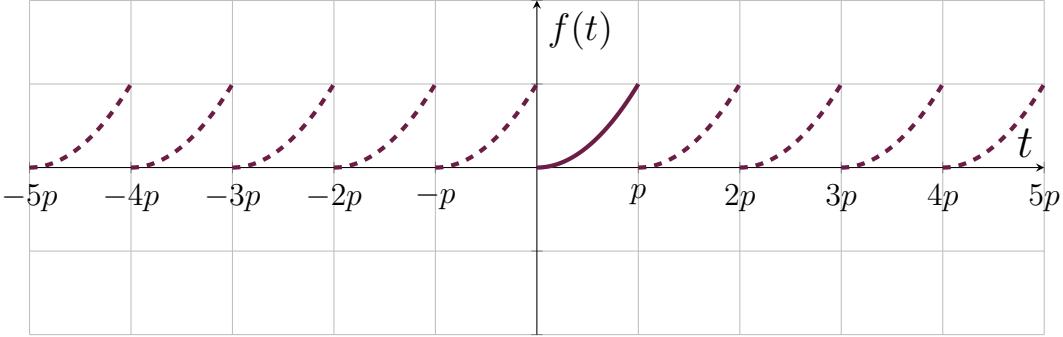


Figura 3.18: Extensión periódica de medio rango de una función $f(t)$. *Fuente: Elaboración propia*

Para esta extensión, podemos ver que ahora la sección de la función que se repite (su periodo) es la sección que se encuentra entre 0 y p , por lo tanto, su periodo es de p , así que su frecuencia angular se define como $\omega_0 = \frac{2\pi}{p}$.

Para la extensión periódica, al no ser par ni impar, requerirá de los 3 coeficientes, la única diferencia es que estos se calculan añadiendo un 2 al argumento del seno y el coseno, lo que permitirá que la extensión de medio rango ahora sea puramente periódica: [30]

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)], \quad (3.44)$$

$$a_0 = \frac{2}{p} \int_0^p f(t) dt, \quad a_n = \frac{2}{p} \int_0^p f(t) \cos(n\omega_0 t) dt, \quad b_n = \frac{2}{p} \int_0^p f(t) \sin(n\omega_0 t) dt \quad (3.45)$$

3.2.9. Serie Compleja de Fourier

Una forma compacta de expresar a la serie trigonométrica de Fourier (??) es definiría en términos de una exponencial compleja. Para lograr esto, se representan a las funciones de seno y coseno usando la notación de Euler $e^{i\theta} = \cos(\theta) + i\sin(\theta)$, donde $i^2 = -1$. [31]

$$\cos t = \frac{e^{it} + e^{-it}}{2}, \quad \sin t = \frac{e^{it} - e^{-it}}{2i} \quad (3.46)$$

Al utilizar (??) para reemplazar $\cos(n\omega_0 t)$ y $\sin(n\omega_0 t)$ en (??), la serie de Fourier de una función f puede escribirse como

$$\begin{aligned} f(t) &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \frac{e^{i(n\omega_0 t)} + e^{-i(n\omega_0 t)}}{2} + b_n \frac{e^{i(n\omega_0 t)} - e^{-i(n\omega_0 t)}}{2i} \right] \\ &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\frac{1}{2} (a_n - ib_n) e^{i(n\omega_0 t)} + \frac{1}{2} (a_n + ib_n) e^{-i(n\omega_0 t)} \right] \\ &= c_0 + \sum_{n=1}^{\infty} c_n e^{i(n\omega_0 t)} + \sum_{n=1}^{\infty} c_{-n} e^{-i(n\omega_0 t)} \end{aligned} \quad (3.47)$$

donde $c_0 = \frac{1}{2}a_0$, $c_n = \frac{1}{2}(a_n - ib_n)$ y $c_{-n} = \frac{1}{2}(a_n + ib_n)$. Los símbolos a_n , b_n son los coeficientes de la definición (??). Cuando la función $f(t)$ es real, c_n y c_{-n} son complejos conjugados y pueden escribirse también en términos de las funciones exponenciales complejas:

$$c_0 = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) dt \quad (3.48)$$

$$\begin{aligned} c_n &= \frac{2}{T} (a_n - ib_n) = \frac{2}{T} \left(\frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(n\omega_0 t) dt - i \frac{T}{2} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(n\omega_0 t) dt \right) \\ &= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) (\cos(n\omega_0 t) - i \sin(n\omega_0 t)) dt \\ &= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-i(n\omega_0 t)} dt \end{aligned} \quad (3.49)$$

$$\begin{aligned} c_{-n} &= \frac{2}{T} (a_n + ib_n) = \frac{2}{T} \left(\frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(n\omega_0 t) dt + i \frac{T}{2} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(n\omega_0 t) dt \right) \\ &= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) (\cos(n\omega_0 t) + i \sin(n\omega_0 t)) dt \\ &= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{i(n\omega_0 t)} dt \end{aligned} \quad (3.50)$$

Puesto que los subíndices de coeficientes y exponentes se encuentran en el rango de todo el conjunto de enteros no negativos... $-3, -2, -1, 0, 1, 2, 3, \dots$, podemos escribir los resultados de (??), (??), (??) y (??) de manera más compacta al sumar tanto enteros negativos como no negativos. En otras palabras, es posible utilizar *una* suma y *una* integral que defina todos los coeficientes c_0 , c_n y c_{-n} . [28]

3.2.9.1. Serie Compleja de Fourier de una función con período T

Si $f(t)$ es una función periódica de periodo T ??, se puede expresar mediante una serie exponencial compleja como: [25]

$$f(t) = c_0 + \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} c_n e^{in\omega_0 t} \quad (3.51)$$

recordando que $\omega_0 = \frac{2\pi}{T}$ es la frecuencia angular fundamental y los coeficientes c_0 y c_n están dados por:

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-in\omega_0 t} dt \quad c_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (3.52)$$

Se calcula el coeficiente c_0 por aparte del coeficiente general c_n ya que en varios casos, el coeficiente c_n presenta una indeterminación en $n = 0$, por lo tanto, calculamos el coeficiente por aparte para evitar este problema.

Recordemos que otra forma en la que nos podemos encontrar la función $f(t)$ es que esté definida sobre un intervalo $[-p, p]$??, recordamos que la frecuencia fundamental toma la forma de $\omega_0 = \frac{\pi}{p}$ y los coeficientes ahora tienen la siguiente forma

$$c_n = \frac{1}{2p} \int_{-p}^p f(t) e^{-in\omega_0 t} dt \quad c_0 = \frac{1}{2p} \int_{-p}^p f(t) dt \quad (3.53)$$

3.2.9.2. Serie Compleja de Fourier de 0 a p

Así como con la serie trigonométrica, también se puede obtener una extensión de medio intervalo usando la serie exponencial compleja, con la diferencia de que solo podemos obtener la extensión de medio rango periódica ?? de una función definida en el intervalo 0 a p , ya que las extensiones par e impar se obtienen únicamente con componentes trigonométricos. Esta serie tiene la misma forma de (??), pero los coeficientes quedan de la siguiente forma [26]

$$c_n = \frac{1}{p} \int_0^p f(t) e^{-in\omega_0 t} dt \quad c_0 = \frac{1}{p} \int_0^p f(t) dt \quad (3.54)$$

Recordando que la frecuencia angular es $\omega_0 = \frac{2\pi}{periodo}$, la frecuencia angular de esta expansión se define como $\omega_0 = \frac{2\pi}{p}$.

3.2.10. Fenómeno de Gibbs

Cuando una función dada se aproxima mediante una serie de Fourier, ya sea trigonométrica o exponencial compleja, habrá un error considerable en la vecindad de la discontinuidad, no importa cuantos términos se quieran emplear. Este efecto se conoce como el fenómeno de Gibbs. Para ilustrar este fenómeno se puede ver en la Figura ?? el resultado de aproximar una onda cuadrada por una serie finita de Fourier [32]

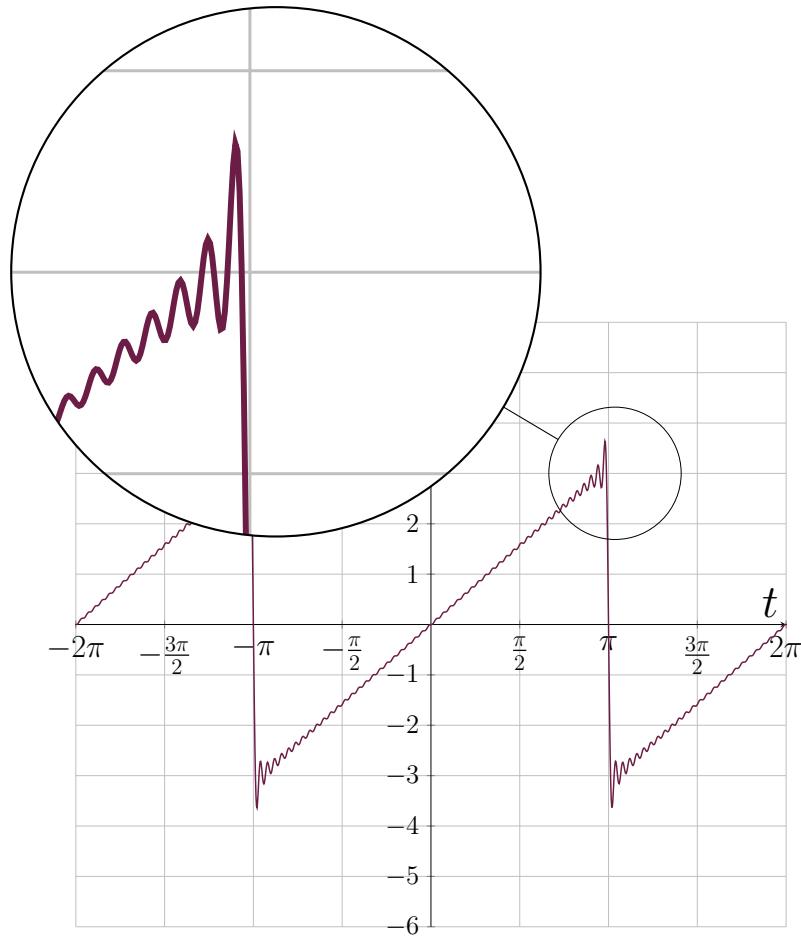


Figura 3.19: Aproximación de una función $f(t) = t$ con $-\pi < t < \pi$ en sus 50 primeros armónicos. *Fuente: Elaboración propia*

3.2.11. Aplicaciones de las series de Fourier

Las series de Fourier tienen aplicaciones que parecerían infinitas en diversas disciplinas, desde la ingeniería hasta la medicina y las ciencias naturales. A continuación, se describen algunos casos destacados que ejemplifican su utilidad en problemas prácticos y teóricos.

3.2.11.1. Ciencias Naturales y Física

Las series de Fourier son utilizadas para modelar fenómenos físicos y naturales como el flujo de calor, vibraciones y propagación de ondas. Algunos ejemplos incluyen:

- **Temperatura de la Tierra:** Cálculo de la distribución de temperatura en la profundidad terrestre mediante series de Fourier [33].
- **Ecuación de calor unidimensional:** Resolución de problemas relacionados con el flujo de calor en barras y otros objetos [33].
- **Detección de profundidad y fondo marino:** Uso de ondas acústicas para medir profundidades y estudiar fondos oceánicos [33].

3.2.11.2. Medicina

En el campo médico, Fourier es una herramienta esencial para el análisis de señales y procesamiento de imágenes:

- **Electroencefalogramas:** Análisis de ondas cerebrales para detectar anomalías [33].
- **Resonancia magnética:** Generación de imágenes médicas a partir de señales complejas [33].
- **Ecodoppler:** Estudio del flujo sanguíneo mediante transformada de Fourier [33].

3.2.11.3. Ingeniería Electrónica y Telecomunicaciones

La ingeniería eléctrica y de telecomunicaciones se beneficia ampliamente del análisis de Fourier para el diseño y análisis de señales y sistemas:

- **Circuitos eléctricos:** Análisis de circuitos con señales periódicas [33].
- **Modulación y demodulación:** Uso de Fourier en señales sinusoidales portadoras [33].
- **Telecomunicaciones:** Estudio de la transmisión de datos en señales eléctricas [33].
- **Redes de computadoras:** Análisis matemático de la capa física del modelo de cinco etapas utilizando series de Fourier. [33].

3.2.11.4. Procesamiento Digital

Las series de Fourier son fundamentales para transformar señales analógicas a digitales, mejorar su calidad mediante filtros y modelar patrones recurrentes en datos. Algunos ejemplos destacados son:

- **Procesamiento de imágenes digitales:** Filtrado y compresión de imágenes utilizando la transformada de Fourier [33].
- **Digitalización del sonido:** Conversión de señales analógicas en digitales [33].
- **Compresión de imágenes:** Uso de DFT para reducir el tamaño de los archivos digitales [33].
- **Modelado estacional con Facebook Prophet:** Uso de series de Fourier para descomponer patrones recurrentes en series temporales, como tendencias anuales y mensuales, facilitando la predicción de datos mediante una suma de términos sinusoidales [34].

3.2.11.5. Matemáticas y Educación

Fourier también tiene aplicaciones pedagógicas y conceptuales que ayudan a entender propiedades matemáticas complejas:

- **Matemáticas detrás de la música:** Estudio de señales acústicas y análisis armónico [33].
- **Método de Fourier para formas de onda:** Aplicaciones en ingeniería eléctrica y diseño de circuitos [33].
- **Funciones descriptivas:** Análisis de propiedades matemáticas específicas [33].

3.2.11.6. Aplicaciones Industriales

Finalmente, las series de Fourier tienen impacto directo en campos como la música, diseño de sistemas y clasificación de patrones:

- **Clasificación de géneros musicales:** Uso de Fourier en redes neuronales para identificar géneros musicales [33].
- **Efectos de sonido en guitarras:** Análisis matemático de las transformaciones del sonido [33].
- **Diseño de sistemas de control:** Predicción del comportamiento de sistemas no lineales [33].

3.3. Aplicaciones Web

Una aplicación web es un software que se ejecuta desde un navegador web, como Google Chrome o Firefox. Estas aplicaciones no necesitan instalarse para poder ser utilizadas, ya que al ejecutarse sobre el navegador solo requieren una conexión a una red de internet. Las aplicaciones web están basadas en una arquitectura cliente-servidor (véase la imagen ??), en la cual el código se divide en dos partes: scripts del lado del cliente y scripts del lado del servidor [35]. Estos dos elementos también se conocen como frontend y backend.



Figura 3.20: Diagrama de una arquitectura cliente - servidor. Fuente: [36]

3.3.1. Frontend

El frontend corresponde a la parte del cliente, el cual se encarga de gestionar la funcionalidad de la interfaz de usuario. Cuando el usuario final accede a la aplicación web mediante un enlace, el navegador carga el script del lado del cliente y renderiza tanto los elementos visuales como el texto necesario para la interacción con el usuario. [35].

3.3.2. Backend

El backend corresponde a la parte del lado del servidor, esta se encarga la lógica de la aplicación. Este servidor gestiona las solicitudes realizadas por el cliente y devuelve una respuesta. Estas solicitudes pueden incluir acciones como la obtención de datos adicionales, la edición de datos o el almacenamiento de nuevos datos. [35].

3.3.3. Tecnologías base de las aplicaciones web

El desarrollo de aplicaciones web se sustenta en un conjunto de tecnologías fundamentales que permiten crear, diseñar y gestionar tanto el frontend como el backend de dichas aplicaciones. Las 3 tecnologías fundamentales de cualquier aplicación web son 3: HTML, CSS y JavaScript.

3.3.3.1. HTML

HTML es el acrónimo de inglés de lenguaje de etiquetas de hipertexto (HyperText Markup Language) es el lenguaje estándar fundamental para estructurar y presentar contenido en la web, siendo la base sobre la cual se construyen todas las páginas web. Su función principal es organizar el contenido de manera semántica, es decir, emplea etiquetas específicas para definir la estructura de los distintos elementos, como encabezados, párrafos, listas, imágenes y enlaces, permitiendo que los navegadores interpreten y muestren adecuadamente el contenido al usuario [37].

3.3.3.2. CSS

CSS es el acrónimo de inglés de hojas de estilo en cascada (Cascading Style Sheets) es básicamente un lenguaje que maneja el diseño y presentación de una aplicación web, es decir, cómo lucen cuando un usuario las visita. Funciona junto con el lenguaje HTML que se encarga del contenido básico de los sitios. Se les denomina hojas de estilo “en cascada” porque se puede tener varias y una de ellas con las propiedades heredadas (o en cascada) de otras [38].

3.3.3.3. JavaScript

JavaScript es un lenguaje de programación que permite brindar interactividad a las aplicaciones web. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web. Además, JavaScript funciona como lenguaje de scripting del lado del servidor a través de entornos de ejecución [39].

3.4. API

API es el acrónimo de inglés de Application Programming Interface (interconexión de programación de aplicaciones), se trata de un software que permite que sistemas separados compartan información entre sí, actuando como un puente que conecta a la aplicación que envía la solicitud (cliente) con un servidor que le proporciona una respuesta [40].

3.4.1. API Rest

Las API de REST (Representational State Transfer o Transferencia del Estado de representación, por su acrónimo del inglés) siguen un conjunto de reglas que hacen que este tipo de API sea mucho más fácil de trabajar; en el núcleo, una API de REST requiere URLs que respondan a las solicitudes HTTP con un elemento de datos o un recurso. Estos datos regularmente están formateados en el marcado de Javascript (JSON en inglés, o Notación de Objetos Javascript).

3.4.2. Protocolo HTTP

El HTTP, que viene del acrónimo de inglés de Protocolo de Transferencia de Hiper Texto (HyperText Transfer Protocol) es el protocolo esencial para la transmisión de información en la World Wide Web, facilitando la comunicación entre servidores, clientes y proxies mediante un lenguaje común que opera principalmente a través de los puertos 80 y 8080. Al ser un protocolo sin estado, no guarda registro de interacciones anteriores, pero utiliza cookies para almacenar información de visitas previas en el cliente. Su funcionamiento se basa en un esquema de petición-respuesta, permitiendo una comunicación eficiente y flexible entre el usuario (como navegadores o rastreadores web) y los servidores web [41].

3.4.2.1. Métodos de petición HTTP

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustitutivos, estos métodos de solicitud a veces son llamados HTTP verbs [42]. Los códigos de respuesta son 9:

- **GET:** El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
- **HEAD:** El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
- **POST:** El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
- **PUT:** El método PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
- **DELETE:** El método DELETE borra un recurso en específico.

- **CONNECT**: El método CONNECT establece un túnel hacia el servidor identificado por el recurso.
- **OPTIONS**: El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.
- **TRACE**: El método TRACE realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.
- **PATCH**: El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.

3.4.2.2. Códigos de estado de respuesta HTTP

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica [43]. Las respuestas se agrupan en cinco clases:

- **Respuestas informativas (100–199)**
- **Respuestas satisfactorias (200–299)**
- **Redirecciones (300–399)**
- **Errores de los clientes (400–499)**
- **Errores de los servidores (500–599)**

3.4.3. Tecnologías base para las APIs

La programación de servicios y microservicios REST simplifica el desarrollo y mantenimiento de aplicaciones web. Sin embargo, su desarrollo y testeo puede resultar incómodo, siempre y cuando, claro está, no se disponga de las herramientas necesarias para el desarrollo de APIs.

3.4.3.1. Desarrollo de APIs

Aunque es perfectamente posible realizar el desarrollo de APIs desde cero, sea cual sea el lenguaje elegido, es conveniente partir de frameworks o plantillas. En algunos casos, serán específicos para el desarrollo de APIs. La lista es interminable y existen para prácticamente todos los lenguajes.

3.4.3.2. NodeJS

Node.js es un entorno de ejecución de un solo hilo, de código abierto y multiplataforma diseñado para desarrollar aplicaciones de red y del lado del servidor rápidas y escalables de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Está escrito en C/C++ y Javascript y emplea una arquitectura de E/S basada en eventos y sin bloqueos, lo que la vuelve eficaz y apropiada para aplicaciones en tiempo real [44].

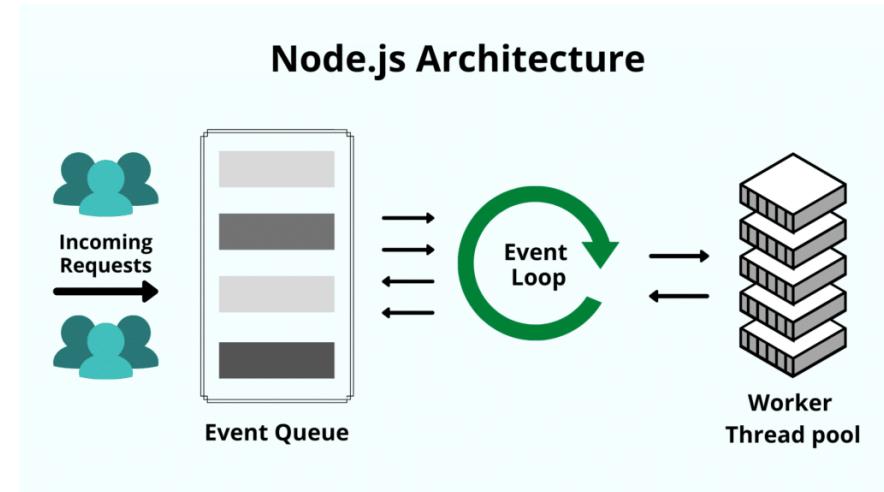


Figura 3.21: Cómo procesa node.js las peticiones entrantes utilizando el bucle de eventos. *Fuente: [44]*

Node.js utiliza una arquitectura de “Single Threaded Event Loop” ?? que maneja múltiples solicitudes mediante un único bucle de eventos y un pool de hilos limitado para operaciones de E/S bloqueantes. A diferencia del modelo multihilo de lenguajes como Java, donde cada solicitud se asigna a un hilo individual del pool, Node.js coloca las solicitudes entrantes en una cola y las procesa de manera no bloqueante, asignando hilos del pool de trabajadores solo cuando es necesario. Esta arquitectura reduce el consumo de recursos y memoria, permitiendo una ejecución más rápida de tareas ligeras y haciendo que Node.js sea ideal para aplicaciones en tiempo real. Sin embargo, para tareas que requieren un procesamiento intensivo de datos, los lenguajes multihilo como Java son más adecuados [44]. Gracias a su arquitectura, bibliotecas propias de NodeJS como lo es ExpressJS y su amplia demanda en la industria, hacen a NodeJS una gran opción para desarrollar una API.

3.4.3.3. Prueba de APIs

Las herramientas para testear o probar una API son fundamentales para garantizar su máximo desempeño una vez esta sea desplegada. Es importante realizar pruebas de porque garantizan que la comunicación de los datos entre sistemas de software sea precisa. Además, esto ayuda a prevenir inconsistencias de datos o interpretaciones erróneas. Los desarrolladores de API están trabajando en nuevas funciones, mejoras, así como corrección de errores, en consecuencia, las APIs suelen sufrir cambios. Las pruebas de API no solo evalúan las nuevas funcionalidades, sino también aseguran que los nuevos cambios no estén afectando las existentes [45].

3.4.3.4. Postman

Es una de las herramientas más utilizadas para probar y desarrollar APIs. Permite realizar solicitudes HTTP de manera sencilla y visualizar las respuestas para verificar su precisión. Postman soporta pruebas automatizadas mediante scripts, lo que facilita la validación de la funcionalidad y el rendimiento de la API en

diferentes escenarios [45].

3.5. Seguridad en aplicaciones web

La seguridad en aplicaciones web es fundamental para proteger la integridad, confidencialidad y disponibilidad de los datos manejados, así como para garantizar la confianza de los usuarios. Algunas medidas que se implementan en una aplicación web para implementar dicha seguridad son las siguientes.

3.5.0.1. Protocolo HTTPS

El protocolo de transferencia de hipertexto seguro, por sus siglas en inglés, (HyperText Transfer Protocol Secure) es una versión segura del protocolo HTTP que utiliza el SSL/TLS para cifrado y autenticación. HTTPS está especificado por RFC 2818 [46] y utiliza el puerto 443 o el 8443 de forma predeterminada en lugar del puerto 80/8080 de HTTP [47].

HTTPS se diferencia de HTTP al agregar cifrado, autenticación e integridad al protocolo original. Mediante SSL/TLS, HTTPS cifra los datos para protegerlos de interceptaciones y ataques de intermediarios, utilizando criptografía de clave pública para establecer una conexión segura entre el servidor y el navegador. Además, HTTPS autentica la identidad del servidor mediante certificados digitales emitidos por autoridades de certificación confiables, garantizando que los documentos provienen de una fuente legítima. También asegura la integridad de los datos mediante firmas digitales que verifican que el contenido no ha sido alterado durante la transmisión. Estas mejoras hacen que HTTPS sea un protocolo mucho más seguro para navegar y realizar transacciones en la web en comparación con HTTP [47], además de volverse un estándar para cualquier sitio web, ya sea que este intercambie o no datos sensibles con los usuarios.

3.5.0.2. Certificado SSL/TLS

El certificado SSL (Secure Sockets Layer) es una pieza clave para establecer una conexión segura entre el servidor y el cliente. Este establece un enlace cifrado entre un servidor y un cliente. Esto permite que información confidencial se transmita de forma segura a través de Internet. El certificado contiene una clave pública que autentica la identidad del sitio web y permite la transferencia de datos cifrados mediante criptografía asimétrica o de clave pública. La clave privada correspondiente se mantiene en secreto en el servidor [48].

3.5.1. CORS

CORS (Cross-Origin Resource Sharing) es un mecanismo de seguridad usado en los navegadores web que permite a un servidor indicar qué dominios pueden acceder a sus recursos. Por defecto, debido a la política de seguridad del mismo origen, los navegadores restringen las solicitudes HTTP realizadas desde scripts a un dominio diferente al de la página que los ejecuta. CORS agrega encabezados HTTP que indican a los navegadores que permitan a una aplicación

web ejecutarse en un origen y acceder a recursos de otro origen diferente. Este tipo de acción se denomina una solicitud HTTP de origen cruzado [49].

3.6. Frameworks

Los frameworks son un conjunto de herramientas, estilos y bibliotecas dispuestas a través de una estructura o esqueleto base, para el desarrollo de aplicaciones web más escalables y sencillas de mantener. Son ampliamente utilizados puesto que permiten acelerar el trabajo, reducir los errores, fomentar la colaboración, y obtener un resultado de más calidad. Consiguiendo así que cualquier proceso de trabajo sea más rápido y eficaz, manteniendo la misma calidad. De este modo, gracias a los frameworks web, es posible ahorrar grandes cantidades de tiempo y costes.

3.6.1. Angular

Angular es un Framework de JavaScript de código abierto escrito en TypeScript. Su objetivo principal es desarrollar aplicaciones de una sola página. Google se encarga del mantenimiento y constantes actualizaciones de mejoras para este framework [50]. Angular es un framework popular debido a su arquitectura basada en componentes, que permite la reutilización de código y facilita el mantenimiento de aplicaciones grandes y complejas. Además, su potente sistema de enlazado de datos bidireccional (two-way data binding) sincroniza automáticamente la interfaz de usuario con los datos subyacentes, mejorando la experiencia del usuario y la eficiencia en el desarrollo [50].

3.6.1.1. TypeScript

TypeScript es un superset de JavaScript que añade tipado estático y características avanzadas al lenguaje de programación. Desarrollado por Microsoft, TypeScript se compila a JavaScript, lo que permite utilizar sus funcionalidades en cualquier entorno que soporte JavaScript. Al incorporar tipos estáticos, TypeScript facilita la detección temprana de errores durante el desarrollo, mejora la legibilidad del código y facilita el mantenimiento de proyectos a gran escala. Además, ofrece soporte para programación orientada a objetos y otras mejoras que potencian la productividad y la robustez de las aplicaciones web [51].

3.6.1.2. Tailwind CSS

Tailwind CSS es un framework de CSS que permite a construir interfaces de usuario personalizadas de manera rápida y eficiente. A diferencia de los frameworks tradicionales que proporcionan componentes predefinidos, Tailwind ofrece clases de utilidad que pueden combinarse directamente en el HTML sin necesidad de escribir CSS personalizado. Esto facilita la creación de diseños responsivos y altamente personalizables, promoviendo la consistencia y reduciendo la cantidad de código CSS necesario [52].

3.7. Sistema de álgebra computacional

Un Sistema de Álgebra Computacional (CAS acrónimo del inglés Computer Algebra System) es un software que facilita las operaciones matemáticas y el cálculo simbólico, es decir, cálculos matemáticos usando valores simbólicos (variables y expresiones) en lugar de valores numéricos [9].

3.7.1. Maxima

Maxima es un Sistema de Álgebra Computacional (CAS, por sus siglas en inglés: Computer Algebra System) diseñado en Lisp, de código abierto diseñado para realizar manipulaciones simbólicas y numéricas de expresiones matemáticas. Originalmente desarrollado a partir de una versión de Macsyma del MIT en 1982, Maxima ha evolucionado gracias a la contribución de una comunidad de desarrolladores y usuarios que continúan mejorándolo y ampliando sus capacidades [9].

Se destaca por ser muy rápido y ligero, lo que permite realizar manipulaciones simbólicas y numéricas de manera eficiente, incluyendo simplificación, derivadas, integrales y resolución de ecuaciones algebraicas y diferenciales. Además, maneja operaciones avanzadas con matrices y vectores, genera gráficos bidimensionales y tridimensionales, y ofrece un lenguaje de programación propio para automatizar cálculos y crear scripts personalizados [9].

3.8. Gráficos en aplicaciones web

Los gráficos en las aplicaciones web se vuelven necesarios cuando hay que representar visualmente datos y mejorar la interacción del usuario. A través de gráficos, es posible mostrar información compleja de manera intuitiva, facilitando el análisis y la comprensión de los datos por parte de los usuarios. Los gráficos también permiten crear visualizaciones interactivas que enriquecen la experiencia de usuario, proporcionando funciones como zoom, desplazamiento, y selección de datos en tiempo real. Existen diversas herramientas y bibliotecas que permiten generar gráficos avanzados y personalizables para una amplia variedad de aplicaciones web.

3.8.1. Canvas

El elemento `<canvas>` es una etiqueta de HTML que permite dibujar gráficos, crear animaciones y renderizar imágenes en tiempo real a través de scripting. El elemento `<canvas>` es sólo un contenedor de gráficos. Es necesario usar JavaScript para crear los gráficos. Canvas cuenta con varios métodos para dibujar trazados, cuadros, círculos, texto así como agregar imágenes. Canvas también es compatible con todos los principales navegadores [53].

3.9. Representación y captura de funciones matemáticas en sitios web

Cuando se trata de visualizar elementos que incluyen notación matemática como fórmulas y expresiones algebraicas, es importante utilizar herramientas que permitan renderizar estas expresiones de manera que sea compatible en distintos navegadores, además de facilitar al usuario poder interpretarlos. Esto ayuda a comunicar conceptos complejos de forma clara y accesible, especialmente en aplicaciones educativas o científicas o financieras. Para ello, existen bibliotecas, que permiten tanto la captura como la visualización de fórmulas matemáticas, facilitando la interacción y manipulación de expresiones directamente en el navegador.

3.9.1. MathQuill

MathQuill es un editor de fórmulas de código abierto para la Web mantenido por Mary Stufflebeam y Han Seoul-Oh. Este editor funciona como una biblioteca que importa módulos necesarios para generar un campo de entrada para el usuario, en donde puede escribir expresiones matemáticas de forma que la notación matemática se mantenga lo mejor posible, ya que usa Latex para mostrar los valores ingresados [54].

3.9.2. MathJax

MathJax es un motor de visualización en JavaScript para notación matemática en LaTeX, que facilita la inclusión de expresiones matemáticas en aplicaciones web sin necesidad de instalar plugins o software adicional. Permite fórmulas en diversos formatos y genera resultados en HTML, CSS o SVG, lo cual asegura que las matemáticas sean textuales y, por tanto, indexables y accesibles en motores de búsqueda [55].

3.10. Parsing

El parsing o parseo es un proceso que implica que un programa analice una cadena de texto, divida sus elementos y extraiga información de utilidad de la misma. Una aplicación del parsing, es el de leer una cadena en determinado lenguaje para convertirlo o entregar una equivalencia del mismo en otro lenguaje.

3.10.1. text2max

Se trata de una biblioteca de código abierto desarrollada por André Storhaug, un estudiante de doctorado en Ciencias de la Computación en la Universidad Noruega de Ciencia y Tecnología (NTNU) [56]. Esta biblioteca está desarrollada en Javascript y permite hacer el proceso de parseo entre la sintaxis de L^AT_EXa el lenguaje del CAS Maxima.

3.11. Servicios de Nube

Los servicios de nube, también conocidos como *cloud services*, son recursos informáticos que se ofrecen a través de Internet. Estos servicios permiten almacenar, gestionar y procesar datos de manera remota, eliminando la necesidad de infraestructura física local. Los principales beneficios de los servicios de nube incluyen escalabilidad, flexibilidad, accesibilidad desde cualquier ubicación, y reducción de costos operativos. Además, facilitan la colaboración y el acceso a aplicaciones y herramientas avanzadas sin requerir instalaciones complejas, lo que los hace fundamentales para el desarrollo y despliegue de aplicaciones web modernas [57].

3.11.1. Microsoft Azure

Microsoft Azure es una plataforma de servicios en la nube proporcionada por Microsoft, que ofrece una amplia gama de soluciones para computación, almacenamiento, bases de datos, análisis, redes, inteligencia artificial y más. Azure permite a los desarrolladores crear, desplegar y gestionar aplicaciones de manera eficiente utilizando herramientas y frameworks familiares. Además, Azure garantiza alta disponibilidad, seguridad robusta y cumplimiento de normativas, lo que lo convierte en una opción confiable para empresas de todos los tamaños [57]’.

3.12. Interfaz gráfica de usuario

Las Interfaces Gráficas de Usuario (GUI) son omnipresentes en nuestra vida cotidiana, ya sea al utilizar una computadora o un celular entre otros dispositivos. La eficacia de una GUI es crucial para determinar si un producto será competitivo o no. Un producto puede fracasar si el usuario no logra completar una acción, como una transacción económica, si no comprende la secuencia de pasos requeridos, si no encuentra fácilmente cómo realizar una acción necesaria, como hacer una compra, o si no encuentra atractivo el diseño de la aplicación [58].

3.12.1. Principios de Usabilidad

Los principios de usabilidad web son la base de cualquier página web para que sea “user friendly”. O lo que es lo mismo, es un tipo de diseño centrado en el usuario para conseguir mejorar la experiencia del mismo. La usabilidad se refiere a la facilidad con la que los usuarios interactúan con una herramienta con el fin de lograr un propósito específico. Así pues, la usabilidad web indica hasta qué punto un sitio web resulta sencillo de utilizar. Esto significa que para que una aplicación web [59].

Los principios de usabilidad, son diez fundamentos que facilitan el diseño de productos más aceptados por los usuarios al enfocarse en sus necesidades y comportamientos [59]:

1. **Visibilidad del estado del sistema.** El usuario siempre debe de estar informado de lo que está pasando en la aplicación web y ofrecerle una

respuesta en el menor tiempo posible. Ejemplos de esto son las barras de carga o notificaciones sobre el navegador.

2. **Relación entre el sistema y el mundo real.** El sistema tiene que “hablar” el lenguaje del usuario usando palabras, frases inclusive símbolos con los que el esté familiarizado y que pueda reconocer con facilidad. Para esto la información debe mostrarse en un orden lógico y usando las palabras y símbolos correctos, sin darle oportunidad al usuario de equivocarse.
3. **Control y libertad del usuario.** Los usuarios pueden equivocarse al realizar una acción dentro de la aplicación, así que este debe tener la oportunidad de corregir su error y no sentirse frustrado por ello. Claro ejemplo de esto es un botón de deshacer.
4. **Consistencia y estándares.** Las aplicaciones deben de seguir los estándares y convenios establecidos para iconos o colores, por ejemplo un ícono con líneas verticales indica un menú, o el color verde y rojo se asocian con aceptar y cancelar, respectivamente.
5. **Prevención de errores.** La aplicación debe de prevenir cualquier posible error que el usuario pueda cometer, y encaso de que este cometa uno, debe de disponer de diversas herramientas para corregirlo, limitar caracteres o entradas no validas por el sistema es un ejemplo de esto.
6. **Reconocer antes que recordar.** La aplicación debe ayudar al usuario a no tener que memorizar acciones u objetos para que pueda usarla fácilmente, usar botones asignados y personalizados para acciones específicas es un ejemplo de esto.
7. **Flexibilidad y eficiencia de uso.** La aplicación debe de estar preparada para recibir a todo tipo de usuarios, si alguien inexperto en el sitio o en el tema del mismo se pueden brindar ayudas como ventanas con mensajes o tutoriales.
8. **Diseño estético y minimalista.** La aplicación no debe de tener información innecesaria que distraiga al usuario y perturbe su experiencia al usarla.
9. **Ayudar a los usuarios a reconocer, diagnosticar y corregir los errores.** En caso de que se produzca un error o excepción dentro de la aplicación, los mensajes de error deben ser compresibles para el usuario, como pasar de un ERROR 404 a un mensaje más entendible como ERROR DEL SERVIDOR.
10. **Ayuda y documentación.** Es preciso que la aplicación cuente con un manual de funcionamiento, esto ayuda al usuario a que le sea más fácil usar la aplicación. Este manual debe ser fácil de localizar, definir los pasos claramente y no de manera extensa.

El cumplir con estos principios de usabilidad ayudan a que el sitio cuente con un tráfico más recurrente, es decir, aumentamos las posibilidades de que un usuario,

después de utilizar la aplicación, la vuelva a usar en un futuro, así como que más personas la usen, demás de que se disminuye el porcentaje de rebote, que no es otra cosa que conseguir que el tiempo de estancia del usuario en la aplicación sea más alto y que explore mas partes del sitio.

CAPÍTULO 4

Análisis del sistema

En este capítulo se evaluaron los datos del marco teórico para seleccionar los componentes más adecuados para el proyecto. También se realizó un análisis detallado de los requerimientos y riesgos asociados al desarrollo del sistema, asegurando un diseño eficiente y viable para su implementación.

4.1. Metodología de Desarrollo

La metodología a usar La metodología propuesta para el desarrollo del proyecto es Prototipos Evolutivos, en la cual se emplearán dos iteraciones mínimas. Esta metodología es adecuada para proyectos en los que la comprensión completa de los requisitos puede surgir gradualmente y en donde es importante recibir retroalimentación continua [60]. Al utilizar prototipos, se permite una aproximación más ágil al desarrollo, ya que se pueden generar soluciones tempranas que serán ajustadas en función de los comentarios de los usuarios y evaluaciones constantes.

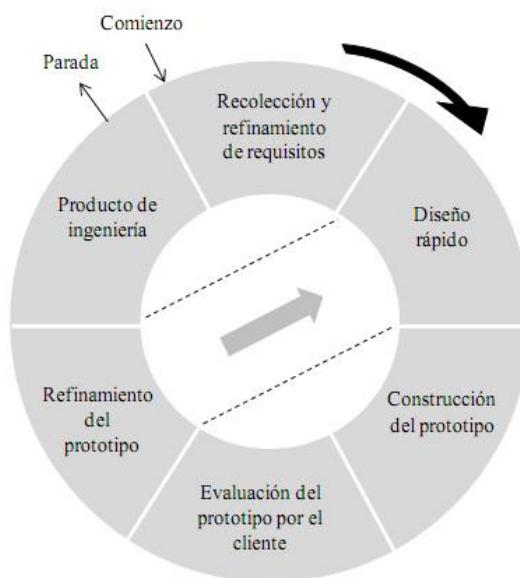


Figura 4.1: tapas de la metodología por prototipos. *Fuente: [60]*

El uso de prototipos permite abordar los requerimientos del sistema de manera iterativa. Cada prototipo no es desecharlo sino que se mejora progresivamente

hasta obtener una versión completa del sistema. Esto asegura que el producto final no solo cumpla con los requisitos establecidos inicialmente, sino también con las expectativas del cliente o usuario.

La retroalimentación temprana que se recibe desde las primeras etapas del proyecto es invaluable para identificar mejoras y posibles problemas antes de que se conviertan en obstáculos mayores. Esto permite un ahorro significativo de tiempo y recursos, ya que se pueden realizar correcciones antes de que los problemas se propaguen en el ciclo de desarrollo. Algunas de las principales ventajas de usar esta metodología son:

- Permite obtener retroalimentación temprana y frecuente del cliente, lo que ayuda a ajustar los requerimientos de manera rápida.
- Facilita la detección temprana de errores o problemas de diseño, lo que reduce costos de desarrollo en fases posteriores.
- Mejora la comprensión de los requisitos por parte del equipo de desarrollo y el cliente, reduciendo malentendidos.
- Aumenta la satisfacción del cliente al ver un producto tangible en las primeras fases del desarrollo.
- Fomenta la iteración y mejora continua del producto, asegurando que se ajuste mejor a las necesidades reales del cliente.

4.2. Análisis de Requerimientos

El análisis de requerimientos tiene la finalidad de establecer las bases del proyecto, ya que permiten conocer si realmente se está entendiendo la problemática principal que se está tratando para garantizar que la solución propuesta es ideal para dar una solución fiable. Con esto en mente, partimos de los resultados de la encuesta y la investigación realizada en el estado del arte ??, se han considerado los requerimientos que se detallan a continuación.

4.2.1. Requerimientos Funcionales

Los requerimientos funcionales describen lo que el sistema debe hacer, es decir, las funcionalidades o tareas específicas que debe realizar para cumplir su propósito. A continuación, se muestran en las siguientes tablas los requerimientos funcionales para el desarrollo del sistema cuyo propósito es calcular y graficar series de Fourier.

Requerimiento	Entrada para la función f
Identificador	RF1

Continúa en la siguiente página

Tabla 4.1 – continuación

Requerimiento	Entrada para la función f
Prioridad de desarrollo	Alta
Entrada	Función f ingresada por el usuario, ya sea en términos de x o de t , en formato de notación matemática, como una sola función o como una función a trozos incluyendo los intervalos de t o x .
Salida	Captura y representación de la función ingresada en notación matemática en la interfaz.
Descripción	El sistema permitirá al usuario ingresar una función f , en un campo de entrada fácil de entender. Proporcionar botones de añadir (\oplus) y quitar (\ominus) para agregar o eliminar trozos de la función sin necesidad de preguntar explícitamente si es una función a trozos.
Precondición	El usuario debe estar en la interfaz de entrada de función.
Postcondición	La función ingresada se mostrará en notación matemática y estará lista para validación y procesamiento.

Tabla 4.2: Requerimiento funcional No. 1

Requerimiento	Teclado en pantalla para entrada de funciones
Identificador	RF2
Prioridad de desarrollo	Media
Entrada	Interacción del usuario con el teclado en pantalla para ingresar funciones y símbolos matemáticos como $\sin()$, $\cos()$, $\sinh()$, $\cosh()$, $\exp()$, exponentes y fracciones.
Salida	Función correctamente ingresada en notación matemática en la interfaz de usuario, utilizando los símbolos seleccionados del teclado en pantalla.

Continúa en la siguiente página

Tabla 4.3 – continuación

Requerimiento	Teclado en pantalla para entrada de funciones
Descripción	La interfaz debe mostrar un teclado en pantalla que incluya funciones matemáticas y símbolos comunes, tales como funciones trigonométricas ($\sin()$, $\cos()$), hiperbólicas ($\cos()$, $\sinh()$), exponenciales ($\exp()$), operaciones de potencia y fracciones. Esto permitirá al usuario ingresar funciones de manera precisa y eficiente.
Precondición	El usuario debe tener acceso a la interfaz de entrada de funciones con el teclado en pantalla activado.
Postcondición	La función ingresada aparece en el campo de entrada en notación matemática, lista para su procesamiento.

Tabla 4.4: Requerimiento funcional No. 2

Requerimiento	Selección del tipo de serie de Fourier
Identificador	RF3
Prioridad de desarrollo	Media
Entrada	El usuario selecciona el tipo de serie a utilizar entre opciones de serie trigonométrica o serie exponencial compleja. Si selecciona la serie trigonométrica, puede optar por extensiones periódicas de rango completo o de medio rango (extensión par usando serie de cosenos o extensión impar usando serie de senos).
Salida	El tipo de serie y extensión seleccionados quedan registrados y configurados en la interfaz, listos para el cálculo de la serie correspondiente.
Descripción	El sistema permite al usuario seleccionar entre opciones de series de Fourier: serie trigonométrica o serie exponencial compleja. Para la serie trigonométrica, el usuario puede elegir entre una extensión periódica de rango completo (opción por defecto) o una extensión de medio rango (par o impar), con la condición de que la función comience en $t = 0$ para las extensiones de medio rango. Si para estas no se selecciona una extensión impar o par, automáticamente se tomará la extensión periódica que incluye senos y cosenos.

Continúa en la siguiente página

Tabla 4.5 – continuación

Requerimiento	Selección del tipo de serie de Fourier
Precondición	El usuario debe estar en la interfaz de selección de series de Fourier y tener una función cargada para análisis de serie.
Postcondición	El tipo de serie y la extensión seleccionada quedan configurados y visibles en la interfaz para su validación y procesamiento.

Tabla 4.6: Requerimiento funcional No. 3

Requerimiento	Validación de funciones compatibles
Identificador	RF4
Prioridad de desarrollo	Alta
Entrada	Función ingresada por el usuario en el campo de entrada de funciones.
Salida	Mensaje de notificación en la interfaz si la función ingresada es inválida para el cálculo de la Serie de Fourier.
Descripción	El sistema debe contar con un catálogo de funciones no válidas para el cálculo de Series de Fourier, (como $\ln(t)$, $csc(t)$). Si el usuario ingresa una función inválida, el sistema debe informarle y evitar el proceso de cálculo, asegurando que solo funciones compatibles continúen al procesamiento.
Precondición	El usuario ha ingresado una función para la validación previa al cálculo de la Serie de Fourier.
Postcondición	La función es verificada y, en caso de ser inválida, el sistema notifica al usuario y bloquea el cálculo de la Serie de Fourier para dicha función.

Tabla 4.8: Requerimiento funcional No. 4

Requerimiento	Parseo de entrada de funciones para procesamiento.
Identificador	RF5
Prioridad de desarrollo	Alta
Entrada	Expresión matemática ingresada por el usuario en formato de notación matemática (LATEX).
Salida	Expresión convertida a un formato compatible para procesamiento matemático en el backend, enviada mediante una solicitud en formato JSON.
Descripción	El sistema debe interpretar y convertir las entradas matemáticas en notación LATEX a un formato compatible con el motor de cálculo. Los datos convertidos se envían al backend a través de una solicitud para su procesamiento.
Precondición	El usuario ha ingresado una expresión matemática en la interfaz.
Postcondición	La expresión se encuentra en el backend en un formato adecuado para su procesamiento y evaluación matemática.

Tabla 4.10: Requerimiento funcional No. 5

Requerimiento	Cálculo de coeficientes y expansión de la Serie de Fourier
Identificador	RF6
Prioridad de desarrollo	Alta
Entrada	Función ingresada por el usuario y configuraciones para el cálculo de la Serie de Fourier, como el número de términos deseado en la expansión.
Salida	Serie de Fourier generada, con los coeficientes calculados a_0 , a_n , b_n o c_n , y expansión de la serie hasta el número especificado de términos.

Continúa en la siguiente página

Tabla 4.11 – continuación

Requerimiento	Cálculo de coeficientes y expansión de la Serie de Fourier
Descripción	El sistema utiliza un motor de cálculo simbólico en el backend para resolver los integrales necesarios, calcular los coeficientes de Fourier a_0 , a_n , b_n o c_n , y generar la expansión de la Serie de Fourier hasta 50 términos.
Precondición	El usuario ha ingresado una función válida y configurado los parámetros para el cálculo de la Serie de Fourier.
Postcondición	La Serie de Fourier calculada y sus coeficientes se encuentran disponibles en la interfaz para visualización y análisis.

Tabla 4.12: Requerimiento funcional No. 6

Requerimiento	Recuperación de resultados para visualización
Identificador	RF7
Prioridad de desarrollo	Media
Entrada	Solicitud de recuperación de resultados generados por el cálculo de la Serie de Fourier, en formato JSON.
Salida	Resultados del cálculo, convertidos para visualización gráfica y en formato LATEX para mostrar expresiones matemáticas en la interfaz.
Descripción	El sistema enviará los resultados del cálculo al frontend a través de una solicitud en formato JSON. La interfaz de usuario debe interpretar estos resultados y convertirlos a un formato adecuado para graficación y a formato LATEX para mostrar las expresiones matemáticas en notación visual.
Precondición	Los resultados de los cálculos de la Serie de Fourier están listos para su recuperación y visualización en el frontend.
Postcondición	Los resultados se muestran en la interfaz de usuario en formato gráfico y en notación matemática (LATEX) para una interpretación visual y análisis claros.

Tabla 4.14: Requerimiento funcional No. 7

Requerimiento	Gráfica interactiva de la función y su aproximación
Identificador	RF8
Prioridad de desarrollo	Alta
Entrada	La función original f y su aproximación de Serie de Fourier, así como el número de términos especificado por el usuario.
Salida	Gráfica interactiva en la interfaz que muestra tanto la función original como su aproximación mediante Serie de Fourier.
Descripción	El sistema debe graficar la función original f y su aproximación mediante la Serie de Fourier en una interfaz gráfica interactiva. El usuario podrá ajustar el número de términos de la serie mediante un control deslizante, y contará con herramientas para hacer zoom, panear e interactuar en tiempo real con la gráfica. La representación gráfica debe ser eficiente y de alto rendimiento para asegurar una experiencia de usuario fluida.
Precondición	El usuario ha ingresado la función y solicitado la visualización gráfica de su aproximación mediante Serie de Fourier.
Postcondición	La gráfica interactiva de la función y su aproximación de Fourier se encuentra visible y responde a las interacciones del usuario.

Tabla 4.16: Requerimiento funcional No. 8

Requerimiento	Visualización de coeficientes y expansión de la Serie de Fourier
Identificador	RF9
Prioridad de desarrollo	Media
Entrada	Coeficientes calculados a_0, a_n, b_n, c_n , y la expresión completa de la Serie de Fourier para la función $f(t)$.

Continúa en la siguiente página

Tabla 4.17 – continuación

Requerimiento	Visualización de coeficientes y expansión de la Serie de Fourier
Salida	Visualización clara y legible de los coeficientes calculados, la función original $f(t)$, y la expansión de la Serie de Fourier.
Descripción	El sistema debe mostrar los coeficientes a_0 , a_n , b_n y c_n , junto con la función original f y su expansión en Series de Fourier $f = \sum$. Las expresiones matemáticas deben presentarse de manera clara y legible, permitiendo al usuario comprender y revisar los cálculos y resultados de forma efectiva.
Precondición	Los coeficientes de la Serie de Fourier y la expansión han sido calculados y están listos para su visualización.
Postcondición	Las expresiones matemáticas se muestran de manera clara en la interfaz, facilitando la interpretación y revisión de los resultados por parte del usuario.

Tabla 4.18: Requerimiento funcional No. 9

4.2.2. Requerimientos No Funcionales

Los requerimientos funcionales definen cómo debe funcionar el sistema, además de características y restricciones que debe cumplir la aplicación, además de su funcionalidad básica.

Nombre	Descripción
Rendimiento	El sistema debe ser capaz de calcular las Series de Fourier en un tiempo razonable y devolver los resultados al usuario, asegurando fluidez en la interacción con la aplicación. La visualización de los gráficos debe ser fluida al agregar o quitar términos de la serie.
Modularidad	Cada microservicio debe ser autónomo, encargado de una tarea específica (por ejemplo, cálculo de series de Fourier) y desacoplado del cliente y otros microservicios, permitiendo su reemplazo o actualización sin afectar al sistema completo.

Continúa en la siguiente página

Tabla 4.19 – continuación

Nombre	Descripción
Escalabilidad	El sistema debe poder integrar nuevos microservicios en el futuro (por ejemplo, cálculo de transformadas de Fourier o análisis simbólico avanzado) sin necesidad de modificar la arquitectura base.
Compatibilidad	La aplicación debe ser compatible con los principales navegadores modernos (Chrome, Firefox, Edge). A pesar de que no se definió y no es recomendable, el diseño debe ser responsive para ajustarse a diferentes tamaños de pantalla, incluidos dispositivos móviles y tabletas.
Usabilidad	La interfaz debe ser intuitiva y fácil de usar, con botones y controles claramente identificados. Los mensajes de error y advertencia deben ser comprensibles, evitando el uso de jerga técnica complicada. La notación matemática debe ser clara, precisa y visualmente agradable.
Mantenibilidad	El código del sistema debe estar estructurado siguiendo buenas prácticas de desarrollo, con documentación técnica que facilite su mantenimiento y futuras mejoras.
Interactividad	Las gráficas deben ser dinámicas, permitiendo zoom, desplazamiento y control del número de coeficientes de Fourier mediante un slider interactivo, con actualizaciones instantáneas.
Fiabilidad	La aplicación debe manejar adecuadamente errores de entrada de funciones no válidas, mostrando mensajes claros y precisos al usuario.
Reusabilidad	El código debe estar bien documentado para que futuros desarrolladores puedan entender la lógica y estructura de la aplicación fácilmente.
Seguridad	Todas las comunicaciones entre el frontend y el backend deben estar cifradas (HTTPS).

Tabla 4.20: Tabla de requerimientos no funcionales

4.3. Análisis de Riesgos

En la gestión de proyectos, los riesgos representan cualquier evento o condición incierta que, si ocurre, puede afectar negativamente los objetivos del proyecto. Identificar y gestionar estos riesgos es crucial para minimizar su impacto y asegurar el éxito del proyecto.

4.3.1. Matriz de riesgos

La matriz de riesgos es una herramienta clave en la gestión de proyectos, ya que permite evaluar y visualizar la gravedad y probabilidad de los riesgos que pueden afectar su éxito. Clasifica los riesgos según sus consecuencias (de "bajo.^a "grave") y su frecuencia de ocurrencia (de raro.^a casi seguro"). Al organizar estos factores en una matriz de cinco por cinco, es más fácil identificar los riesgos más críticos y priorizar las acciones correctivas. Los colores de la matriz van del verde (bajo riesgo) al rojo (alto riesgo), lo que proporciona una representación visual clara de la urgencia y el impacto de cada riesgo identificado ??.

Consecuencia	Probabilidad				
	Casi seguro	Probable	Possible	Improbable	Raro
Grave	Extremo	Extremo	Extremo	Alto	Medio
Mayor	Extremo	Extremo	Alto	Medio	Medio
Moderado	Extremo	Alto	Medio	Medio	Bajo
Menor	Alto	Medio	Medio	Bajo	Bajo
Bajo	Medio	Medio	Bajo	Bajo	Bajo

Tabla 4.21: Matriz de clasificación de riesgos. *Fuente: Elaboración propia*

4.3.2. Identificación y jerarquización de riesgos

Para poder comenzar con el análisis de riesgos pertinente, comenzamos identificación los riesgos que pueden ocurrir y, por lo tanto, afectar la correcta ejecución del proyecto, además de su jerarquización en base a su semáforo.

ID Riesgo	Descripción	Probabilidad	Impacto	Semáforo
R1	Incapacidad temporal para continuar el proyecto debido a una enfermedad u otra causa que afecte la salud o disponibilidad.	Possible	Grave	

Continúa en la siguiente página

Tabla 4.22 – continuación

ID Riesgo	Descripción	Probabilidad	Impacto	Semáforo
R2	No cumplir con los objetivos del proyecto	Possible	Mayor	Rojo
R3	Sobrecarga de proyectos académicos	Probable	Mayor	Rojo
R4	Mala gestión del tiempo y actividades a lo largo del desarrollo del proyecto.	Possible	Mayor	Naranja
R5	Mala elección de tecnologías para el desarrollo del proyecto	Possible	Mayor	Naranja
R6	Falta de claridad o mal planteamiento de los requerimientos	Possible	Mayor	Naranja
R7	Atraso en la Entrega de Prototipos.	Possible	Mayor	Naranja
R8	Falta de conocimientos para realizar actividades relacionadas al proyecto.	Possible	Menor	Ambar
R9	Aumento de la complejidad del proyecto.	Possible	Moderado	Ambar

Tabla 4.23: Tabla de identificación de riesgos jerarquizada.

Una vez se tienen identificados y jerarquizados todos los riesgos, se procede a hacer un análisis individual de cada uno.

Hoja de información de riesgo	
Riesgo ID:	R1
Fecha:	10/11/2024
Probabilidad:	Possible
Impacto:	Grave

Continúa en la siguiente página

Tabla 4.24 – continuación

Hoja de información de riesgo

Descripción:	Incapacidad temporal para continuar el proyecto debido a enfermedad u otra causa que afecte mi salud o disponibilidad, como único miembro del equipo.
Tipo de Riesgo:	Rendimiento del equipo (individual).
Contexto:	<ul style="list-style-type: none"> ▪ Sub-Condición 1: Si me enfermo o enfrento una incapacidad, todo el trabajo del proyecto se detiene, ya que no hay otros miembros para continuar las actividades. ▪ Sub-Condición 2: La duración de la incapacidad es incierta, lo cual podría impactar significativamente el progreso del proyecto.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ▪ Aceptar: Reconocer el riesgo de interrupción en caso de incapacidad y comunicar el estado de avance a las partes interesadas regularmente. ▪ Minimizar impacto: Llevar un registro detallado y documentado del proyecto para que cualquier avance pueda ser retomado sin dificultades en caso de una pausa temporal.
Plan de contingencia	Documentar todas las actividades y avances detalladamente, manteniendo versiones de respaldo en la nube para asegurar que el proyecto pueda retomarse rápidamente tras cualquier interrupción. Informar a las partes interesadas sobre el progreso y posibles pausas debido a mi disponibilidad.
Estado Actual:	En evaluación

Tabla 4.25: Hoja de información de riesgo - R1

Hoja de información de riesgo

Riesgo ID:	R2
Fecha:	10/11/2024
Probabilidad	Possible

Continúa en la siguiente página

Tabla 4.26 – continuación

Hoja de información de riesgo

Impacto:	Mayor
Descripción:	No cumplir con los objetivos del proyecto debido a retrasos o falta de recursos.
Tipo de Riesgo:	Cumplimiento de objetivos.
Contexto:	<ul style="list-style-type: none"> ▪ Sub-Condición 1: Los plazos del proyecto son ajustados y cualquier retraso podría afectar el logro de los objetivos establecidos. ▪ Sub-Condición 2: La falta de una adecuada organización podría impedir la finalización de algunas actividades clave.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ▪ Aceptar: Aceptar el riesgo pero monitorear constantemente el avance del proyecto para detectar retrasos a tiempo. ▪ Planificar metas intermedias: Establecer metas de corto plazo para asegurar el progreso constante y mantener el enfoque en los objetivos principales.
Plan de contingencia	Revisar y ajustar el plan de trabajo regularmente para priorizar las tareas críticas. Identificar actividades que puedan ser simplificadas o pospuestas si el tiempo o los recursos se vuelven limitados.
Estado Actual:	En fase de planificación

Tabla 4.27: Hoja de información de riesgo - R2

Hoja de información de riesgo

Riesgo ID:	R3
Fecha:	02/03/24
Probabilidad:	Probable
Impacto:	Mayor

Continúa en la siguiente página

Tabla 4.28 – continuación

Hoja de información de riesgo

Descripción:	Sobrecarga de proyectos académicos que pueda limitar el tiempo dedicado al proyecto actual.
Tipo de Riesgo:	Gestión de tiempo y recursos.
Contexto:	<ul style="list-style-type: none"> ▪ Sub-Condición 1: La carga académica en otras asignaturas o actividades podría restar tiempo al desarrollo del proyecto. ▪ Sub-Condición 2: Las fechas de entrega de otros proyectos coinciden con los períodos importantes de este proyecto, aumentando el riesgo de retrasos.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ▪ Planificación anticipada: Organizar un calendario de trabajo que contemple todas las responsabilidades y anticiparse a los períodos de alta carga académica. ▪ Distribución de tareas: Dividir el proyecto en tareas más pequeñas y alcanzables para avanzar gradualmente, incluso en semanas de alta demanda.
Plan de contingencia	Identificar tareas del proyecto que puedan realizarse en paralelo con otras responsabilidades académicas, aprovechando momentos libres o períodos de menor carga. De ser posible, simplificar ciertos entregables para adaptarlos a la disponibilidad de tiempo.
Estado Actual:	En fase de planificación

Tabla 4.29: Hoja de información de riesgo - R3

Hoja de información de riesgo

Riesgo ID:	R4
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Mayor

Continúa en la siguiente página

Tabla 4.30 – continuación

Hoja de información de riesgo

Descripción:	Mala gestión del tiempo y actividades a lo largo del desarrollo del proyecto.
Tipo de Riesgo:	Organización y planificación.
Contexto:	<ul style="list-style-type: none"> ▪ Sub-Condición 1: La falta de planificación detallada podría llevar a demoras en la ejecución de tareas críticas. ▪ Sub-Condición 2: La pérdida de enfoque en las prioridades del proyecto podría afectar la calidad y cumplimiento de los objetivos.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ▪ Establecer prioridades claras: Definir las tareas más importantes y enfocarse en completarlas primero. ▪ Monitoreo semanal: Revisar semanalmente el progreso y ajustar las actividades según los avances logrados.
Plan de contingencia	En caso de identificar retrasos, reasignar tiempos y simplificar tareas menos prioritarias para asegurar que los entregables principales se completen dentro de los plazos definidos.
Estado Actual:	En implementación de estrategia de monitoreo

Tabla 4.31: Hoja de información de riesgo - R4

Hoja de información de riesgo

Riesgo ID:	R5
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Mayor
Descripción:	Mala elección de tecnologías para el desarrollo del proyecto, lo que podría afectar su funcionalidad o escalabilidad.

Continúa en la siguiente página

Tabla 4.32 – continuación

Hoja de información de riesgo

Tipo de Riesgo:	Selección de tecnologías.
Contexto:	<ul style="list-style-type: none"> ■ Sub-Condición 1: La elección de una tecnología inadecuada podría generar dificultades en el desarrollo o falta de soporte. ■ Sub-Condición 2: Cambiar de tecnología a mitad del proyecto podría ocasionar retrasos y pérdida de avances.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ■ Investigación previa: Evaluar diferentes tecnologías antes de iniciar el proyecto y seleccionar la que mejor se adapte a los requerimientos. ■ Pruebas iniciales: Realizar pruebas tempranas para verificar la compatibilidad y funcionalidad de las tecnologías seleccionadas.
Plan de contingencia	En caso de identificar problemas críticos con la tecnología elegida, evaluar la viabilidad de migrar a una alternativa o de implementar soluciones temporales que permitan continuar con el desarrollo hasta que se resuelva el inconveniente.
Estado Actual:	En proceso de evaluación tecnológica

Tabla 4.33: Hoja de información de riesgo - R5

Hoja de información de riesgo

Riesgo ID:	R6
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Mayor
Descripción:	Falta de claridad o mal planteamiento de los requerimientos, lo que podría afectar el desarrollo adecuado del proyecto.

Continúa en la siguiente página

Tabla 4.34 – continuación

Hoja de información de riesgo

Tipo de Riesgo:	Definición de requerimientos.
Contexto:	<ul style="list-style-type: none"> ■ Sub-Condición 1: La ambigüedad en los requerimientos podría generar retrabajo o desvíos en el desarrollo. ■ Sub-Condición 2: Cambios constantes en los requerimientos afectan la estabilidad del proyecto.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ■ Revisión exhaustiva: Revisar los requerimientos con las partes interesadas para asegurar comprensión y claridad antes de comenzar el desarrollo. ■ Documentación detallada: Mantener una documentación clara y detallada de cada requerimiento, incluyendo su justificación y alcance.
Plan de contingencia	En caso de identificar ambigüedades o cambios en los requerimientos, realizar una reunión de revisión para redefinir los objetivos y ajustar el enfoque del proyecto, minimizando el impacto en los avances previos.
Estado Actual:	En revisión de requerimientos

Tabla 4.35: Hoja de información de riesgo - R6

Hoja de información de riesgo

Riesgo ID:	R7
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Mayor
Descripción:	Atraso en la entrega de prototipos que pueda afectar la evaluación del proyecto.
Tipo de Riesgo:	Gestión de entregas.

Continúa en la siguiente página

Tabla 4.36 – continuación

Hoja de información de riesgo

Contexto:	<ul style="list-style-type: none"> Sub-Condición 1: La falta de tiempo para completar los prototipos puede retrasar su entrega para revisión y comentarios. Sub-Condición 2: Los prototipos son esenciales para evaluar el avance y realizar ajustes en etapas tempranas.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> Definición de entregas parciales: Establecer entregables parciales que permitan verificar el avance y corregir posibles desvíos. Monitoreo constante: Evaluar semanalmente el progreso en los prototipos para asegurar que los plazos se cumplen.
Plan de contingencia	Si se anticipa un retraso en la entrega de prototipos, priorizar las funcionalidades clave para desarrollar una versión preliminar. Esto permite realizar pruebas básicas y obtener retroalimentación, mientras se completan las secciones faltantes.
Estado Actual:	En fase de desarrollo de prototipos

Tabla 4.37: Hoja de información de riesgo - R7

Hoja de información de riesgo

Riesgo ID:	R8
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Menor
Descripción:	Falta de conocimientos para realizar actividades específicas del proyecto, lo cual podría afectar la calidad o retrasar el desarrollo.
Tipo de Riesgo:	Competencias técnicas.

Continúa en la siguiente página

Tabla 4.38 – continuación

Hoja de información de riesgo

Contexto:	<ul style="list-style-type: none"> Sub-Condición 1: La falta de experiencia en ciertas tecnologías o metodologías podría complicar la ejecución de ciertas tareas. Sub-Condición 2: Aprender nuevas habilidades durante el proyecto podría demorar su avance.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> Capacitación anticipada: Identificar áreas de mejora y capacitarse en estas áreas antes de iniciar tareas específicas. Consultas puntuales: Buscar apoyo de expertos o recursos en línea para resolver dudas específicas rápidamente.
Plan de contingencia	En caso de enfrentar dificultades técnicas, priorizar el autoaprendizaje mediante recursos de aprendizaje en línea. Además, evaluar la posibilidad de simplificar ciertas funcionalidades para evitar demoras.
Estado Actual:	Identificación de áreas de mejora técnica

Tabla 4.39: Hoja de información de riesgo - R8

Hoja de información de riesgo

Riesgo ID:	R9
Fecha:	02/03/24
Probabilidad:	Possible
Impacto:	Moderado
Descripción:	Aumento de la complejidad del proyecto, lo cual podría requerir más tiempo o recursos de lo inicialmente planificado.
Tipo de Riesgo:	Complejidad del proyecto.

Continúa en la siguiente página

Tabla 4.40 – continuación

Hoja de información de riesgo

Contexto:	<ul style="list-style-type: none"> ■ Sub-Condición 1: La ampliación del alcance o de los requisitos puede hacer que el proyecto sea más complejo de lo previsto. ■ Sub-Condición 2: La complejidad adicional podría requerir ajustes en el diseño y la implementación.
Mitigación / Monitoreo:	<ul style="list-style-type: none"> ■ Control de alcance: Revisar regularmente el alcance para evitar la adición de requisitos que aumenten la complejidad. ■ Evaluación de impacto: Analizar los posibles efectos de cualquier cambio en el diseño antes de su implementación.
Plan de contingencia	Si la complejidad del proyecto aumenta, evaluar y priorizar las funcionalidades esenciales. Considerar implementar las funciones adicionales en fases posteriores si es necesario para cumplir con los objetivos iniciales en el tiempo previsto.
Estado Actual:	En monitoreo de alcance y requisitos

Tabla 4.41: Hoja de información de riesgo - R9

4.4. Estudio de factibilidad

El estudio de factibilidad sirve para evaluar si el proyecto puede llevarse a cabo de manera exitosa considerando aspectos operativos, tecnológicos y económicos en el tiempo establecido que son 12 meses. Este análisis asegura que los recursos disponibles sean suficientes y que el proyecto sea viable en todos los sentidos.

4.4.1. Factibilidad operativa

Analiza si el proyecto cumple con las necesidades de los usuarios finales y si las operaciones requeridas pueden ser ejecutadas de manera eficiente dentro del entorno previsto.

Mes	Días	Fin de semana	Días no laborales	Días hábiles	Horas de trabajo por día (media)	Horas totales	Días laborales (8 horas al día)
Agosto 2024	31	10	16	5	4	20	2.5
Septiembre 2024	30	9	1	20	4	80	10
Octubre 2024	31	8	1	22	4	88	11
Noviembre 2024	30	9	2	19	4	76	9.5
Diciembre 2024	31	9	7	15	4	60	7.5
Enero 2025	31	8	3	20	4	80	10
Febrero 2025	28	8	1	19	4	76	9.5
Marzo 2025	31	10	1	20	4	80	10
Abril 2025	30	8	7	15	4	60	7.5
Mayo 2025	31	9	3	19	4	76	9.5
Junio 2025	30	4	0	26	4	104	13
Total de días laborales							100

Tabla 4.43: Tiempo de ejecución para el proyecto

La tabla muestra la distribución del tiempo disponible para el desarrollo del proyecto, considerando los días totales de cada mes, fines de semana, días no laborales, y días hábiles con sus respectivas horas promedio de trabajo. Se estima un total de 100 días laborales efectivos (considerando jornadas de 8 horas), lo cual proporciona una visión clara del tiempo real disponible para cumplir con las tareas del proyecto.

Cantidad	Rol	Descripción	Sueldo x Mes	Subtotal
1	Full Stack Developer	Desarrollo de frontend y backend, integración de APIs y diseño de bases de datos	\$18,000	\$18,000
Subtotal del proyecto				\$20,000
Ajuste por día				\$666
Total				\$66,666

Tabla 4.44: Factibilidad tecnológica de software.

El valor del salario es un promedio obtenido de ofertas laborales para dicho puesto en las páginas CompuTrabajo, Talentmx, y LinkedIn a la fecha del 10 de noviembre de 2024. Estos salarios pueden cambiar de acuerdo con las fechas consultadas y al mercado versátil global.

4.4.2. Factibilidad tecnológica

Evaluá la disponibilidad y adecuación de las tecnologías necesarias para desarrollar y ejecutar el proyecto, asegurando que los recursos tecnológicos sean compatibles con los requisitos del sistema.

Cantidad	Tecnología	Descripción	Costo	Subtotal
1	Microsoft Azure	Suscripción a Azure para alojamiento del proyecto durante 6 meses (\$500/mes) para alojar la aplicación y garantizar su disponibilidad.	MXN\$500	MXN\$3000
Total			MXN\$3,000	MXN\$3,000

Tabla 4.46: Factibilidad tecnológica de software.

Servicio	Costo
Luz	\$2,700 MXN
Agua	\$1,200 MXN
Internet	\$2,400 MXN
Transporte	\$1,500 MXN
Alimentos	\$6,600 MXN

Continúa en la siguiente página

Tabla 4.47 – continuación	
Servicio	Costo
Total	\$14,400 MXN

Tabla 4.48: Factibilidad tecnológica en otros gastos.

Para obtener el total de la factibilidad tecnológica se debe sumar el total de la tabla ?? con la tabla ??, lo que nos da:

$$\text{Total de factibilidad tecnológica} = (\text{software}) + (\text{others})$$

$$\text{Total de factibilidad tecnológica} = \text{MXN\$}2,400 + \text{MXN\$}14,400$$

Por lo tanto:

$$\text{Total de factibilidad tecnológica} = \text{MXN\$}17,400 \quad (4.1)$$

4.4.3. Factibilidad económica

Para obtener la factibilidad económica, es decir, el total del proyecto es necesario sumar la factibilidad operativa junto con la tecnológica.

$$\text{Total de factibilidad económica} = (\text{Factibilidad operativa}) + (\text{Factibilidad tecnológica})$$

$$\text{Total de factibilidad económica} = \text{MXN\$}66,666 + \text{MXN\$}17,400$$

Por lo tanto:

$$\text{Total de factibilidad económica} = \text{MXN\$}84,066 \quad (4.2)$$

CAPÍTULO 5

Diseño del sistema

El diseño del sistema define la estructura y organización de los componentes de software necesarios para la operatividad de la aplicación web, estableciendo cómo interactúan y cooperan las tecnologías seleccionadas para cumplir con los requisitos funcionales. Este diseño se ilustra mediante diagramas que muestran la arquitectura y flujo de trabajo del sistema, junto con mockups de interfaz de usuario que proporcionan una visión general de la experiencia e interacción del usuario con la aplicación.

5.1. Tabla de tecnologías

En la tabla ?? se muestra un resumen de las tecnologías usadas dentro de este proyecto. Cada una especifica la tecnología utilizada, su función dentro del proyecto, su categoría y su integración.

Tecnología	Función	Categoría	Integración
HTML	Lenguaje de marcado para estructura web	Desarrollo Web	Define la estructura de la aplicación
JavaScript	Lenguaje de programación para interactividad	Desarrollo Web	Lógica y funcionalidades interactivas
CSS	Lenguaje para estilización web	Desarrollo Web	Mejora el diseño visual
Angular	Framework para desarrollo web frontend	Desarrollo Web	Gestiona la interfaz y su lógica
TypeScript	Superset de JavaScript con tipado estático	Lenguaje de Programación	Facilita el desarrollo seguro en Angular
Tailwind CSS	Framework para estilos CSS utilitario	Estilización Web	Facilita estilos rápidos y responsivos

Continúa en la siguiente página

Tabla 5.1 – continuación

Tecnología	Función	Categoría	Integración
MathQuill	Renderización de expresiones matemáticas de entrada en páginas web	Matemáticas en aplicaciones web	Captura las fórmulas de entrada del usuario
MathJax	Renderización de fórmulas matemáticas en el sitio web	Matemáticas en aplicaciones web	Mostrar resultados matemáticos correctamente
Node.js	Entorno de ejecución para JavaScript en backend	Desarrollo Web	Procesar cálculos y operaciones backend
Maxima	Software para cálculos algebraicos simbólicos	Matemáticas	Calcular los coeficientes y expansiones de serie
Postman	Herramienta para pruebas de APIs REST	Desarrollo Backend	Valida el funcionamiento del backend
tex2max	Librería para convertir fórmulas TeX a formato Maxima	Parser	Convierte entrada TeX a formato computable
Microsoft Azure	Plataforma en la nube para despliegue y hospedaje de la aplicación	Infraestructura en la nube	Aloja la aplicación web
Git	Sistema de control de versiones	Controlador de versiones	Permite gestionar cambios en el código
GitHub	Plataforma para alojar proyectos Git	Controlador de versiones	Almacena y colabora en el repositorio

Tabla 5.2: Tabla de tecnologías y su uso en la aplicación

5.2. Diagrama de arquitectura general del sistema

El diagrama de la figura ?? ilustra la arquitectura general de la aplicación de cálculo y graficación mediante el modelo de cliente servidor

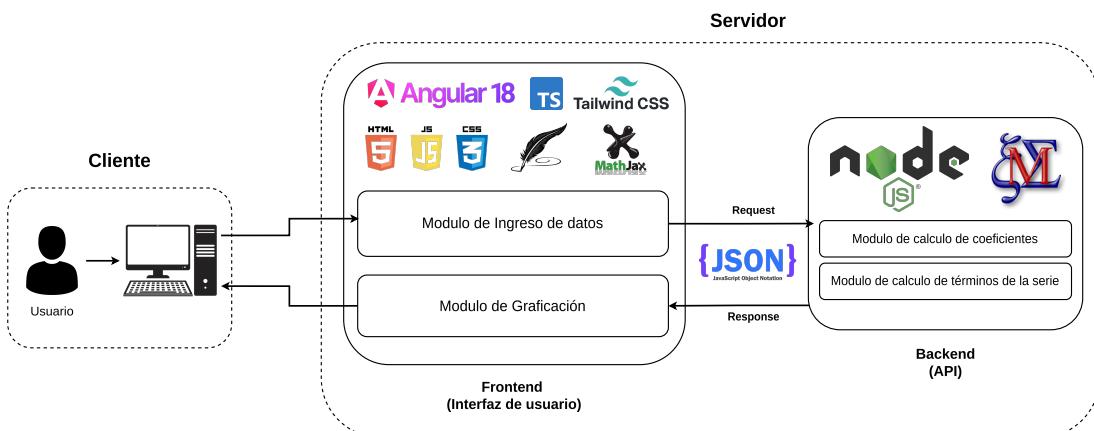


Figura 5.1: Diagrama de arquitectura del sistema. *Fuente: Elaboración propia*

El diagrama de la figura ?? ilustra la arquitectura general de la aplicación web para el cálculo y graficación de las series de Fourier. Este sistema sigue una arquitectura cliente-servidor, complementada con el modelo de microservicios para garantizar modularidad y escalabilidad en el desarrollo futuro. Dentro de este diagrama se describen 3 principales elementos:

- **Cliente:** Representa al usuario final que interactúa con el sistema a través de un navegador web. Incluye herramientas de ingreso de datos, selección de parámetros y graficación interactiva.
- **Servidor:** Implementado en NodeJS, actúa como intermediario entre el cliente y el sistema de cálculo matemático Maxima. Maneja las solicitudes entrantes, procesa los cálculos y devuelve los resultados.
- **JSON:** Actúa como formato estándar para la comunicación entre el cliente, el servidor y los microservicios, asegurando interoperabilidad y facilidad de integración.

5.2.1. Arquitectura cliente-servidor

El sistema se divide en dos componentes principales: el cliente y el servidor.

- **Cliente:**
 - El cliente es la interfaz de usuario construida en Angular, donde los usuarios pueden interactuar con el sistema mediante un navegador web.
 - Incluye los siguientes módulos clave:

- **Módulo de Ingreso de Datos:** Diseñado para que el usuario ingrese la función f en términos de x o de t , ya sea de manera implícita (función continua) o por partes (trozos de la función). Además, cuenta con herramientas visuales como un botón (\oplus) para agregar trozos adicionales o (\ominus) para eliminarlos.
- **Módulo de Selección de Parámetros:** Permite configurar el tipo de serie de Fourier que se calculará (trigonométrica, exponencial compleja o extensiones de medio rango). Se presentan las opciones para extensiones pares o impares en caso de la serie trigonométrica.
- **Módulo de Graficación:** Utilizando Canvas, se genera una visualización interactiva de la función original y su aproximación mediante series de Fourier. También incluye controles como un slider para ajustar el número de coeficientes utilizados en la gráfica.
- **MathQuill/MathJax:** Herramientas integradas para que el usuario pueda capturar las funciones y la aplicación muestre los resultados en un formato amigable utilizando notación TeX.
- **tex2max:** Se parsean todas las entradas necesarias (la función y sus intervalos) para ser enviadas al servidor y realizar los cálculos.

■ **Servidor:**

- El servidor está implementado en NodeJS y se comunica con el cliente a través de una API RESTful.
- Se encarga de procesar las solicitudes enviadas desde el cliente y realizar el cálculo de los coeficientes y la extensión de la serie mediante el sistema algebraico computacional (CAS) Maxima. Además de también convertir estas expresiones a formato de Tex.
- Principales responsabilidades:
 - Recepción y parseo de las entradas en formato JSON, enviadas desde el cliente con la ayuda de la librería tex2max.
 - Ejecución de comandos en Maxima para calcular las integrales definidas y obtener los coeficientes a_0 , a_n , b_n (para la serie trigonométrica) o c_n (para la exponencial compleja). Además de almacenar estos resultados en formato Maxima y en formato Tex
 - Generación de resultados en formato JSON para su devolución al cliente, incluyendo los coeficientes calculados y las series de Fourier en formato Maxima y Tex.

5.2.2. Arquitectura basada en microservicios

Aunque actualmente solo se cuenta con un microservicio (el API para cálculo de series de Fourier), el diseño sigue una arquitectura de microservicios para permitir la escalabilidad futura. Esto implica que:

- El servicio de cálculo de series de Fourier está desacoplado del resto del sistema, permitiendo su mantenimiento, actualización o reemplazo sin afectar otras partes de la aplicación.

- En el futuro, se pueden agregar otros microservicios para cálculos relacionados sin alterar la arquitectura actual.
- Cada microservicio puede ser independiente y comunicarse con el servidor principal mediante una API RESTful bien definida.

5.3. Diagrama de casos de uso

En la figura ?? se presenta el diagrama de casos de uso general de la aplicación, el cual describe las interacciones entre el usuario y las principales funcionalidades del sistema.

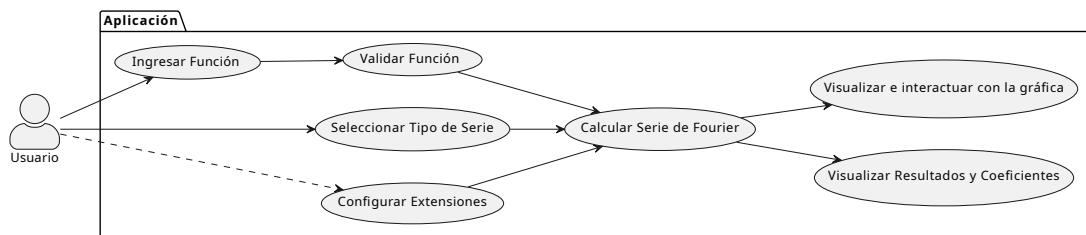


Figura 5.2: Diagrama general de casos de uso del sistema. *Fuente: Elaboración propia*

El diagrama de casos de uso describe las interacciones entre usuario:

- **Usuario:** Representa a la persona que interactúa con la aplicación, como estudiantes o docentes interesados en analizar y graficar series de Fourier.

Con las funcionalidades principales de la *Aplicación Web para el Cálculo y Graficación de Series de Fourier*.

1. **Ingresar Función:** El usuario introduce la función matemática que desea analizar.
2. **Validar Función:** El sistema verifica si la función ingresada es válida y puede ser procesada.
3. **Seleccionar Tipo de Serie:** El usuario elige el tipo de serie de Fourier a calcular (por ejemplo, serie completa, solo senos, solo cosenos, etc.).
4. **Configurar Extensiones** (opcional): El usuario ajusta configuraciones adicionales, como los límites de integración o la cantidad de términos en la serie.
5. **Calcular Serie de Fourier:** Se realiza el cálculo de los coeficientes de la serie y se generan los datos necesarios.
6. **Visualizar Resultados y Coeficientes:** El sistema presenta los valores calculados, incluyendo los coeficientes de la serie, en un formato comprensible.

7. **Visualizar e Interactuar con la Gráfica:** El sistema genera una gráfica interactiva que permite al usuario explorar visualmente la serie de Fourier y su aproximación a la función original.

El flujo principal del diagrama es el siguiente:

1. El **Usuario** comienza **Ingresando una Función** (*CU1*) que es **Validada** automáticamente (*CU4*).
2. Luego, **Selecciona el Tipo de Serie** (*CU2*) y, opcionalmente, **Configura Extensiones** (*CU3*).
3. Despues de estos pasos, la aplicación **Calcula la Serie de Fourier** (*CU5*), generando **Resultados y Coeficientes** (*CU6*) y **Gráficas Interactivas** (*CU7*) que el usuario puede explorar visualmente.

5.4. Diagramas de secuencia UML

Los diagramas de secuencia UML son una representación gráfica que ilustra la interacción de objetos en un sistema de forma cronológica, capturando la secuencia de mensajes intercambiados entre objetos y el orden en que ocurren estas interacciones.

5.4.1. Ingreso y validación de una función

El diagrama de la figura ?? ilustra como el usuario ingresa una función, donde esta función puede estar formada de múltiples trozos y el usuario puede añadir o quitar estos trozos, definiendo su valor además del intervalo en el que está definido. Estos datos validarán para indicar si se ingresó algún valor o función no válidos. También ingresará el tipo de extensión de la serie.

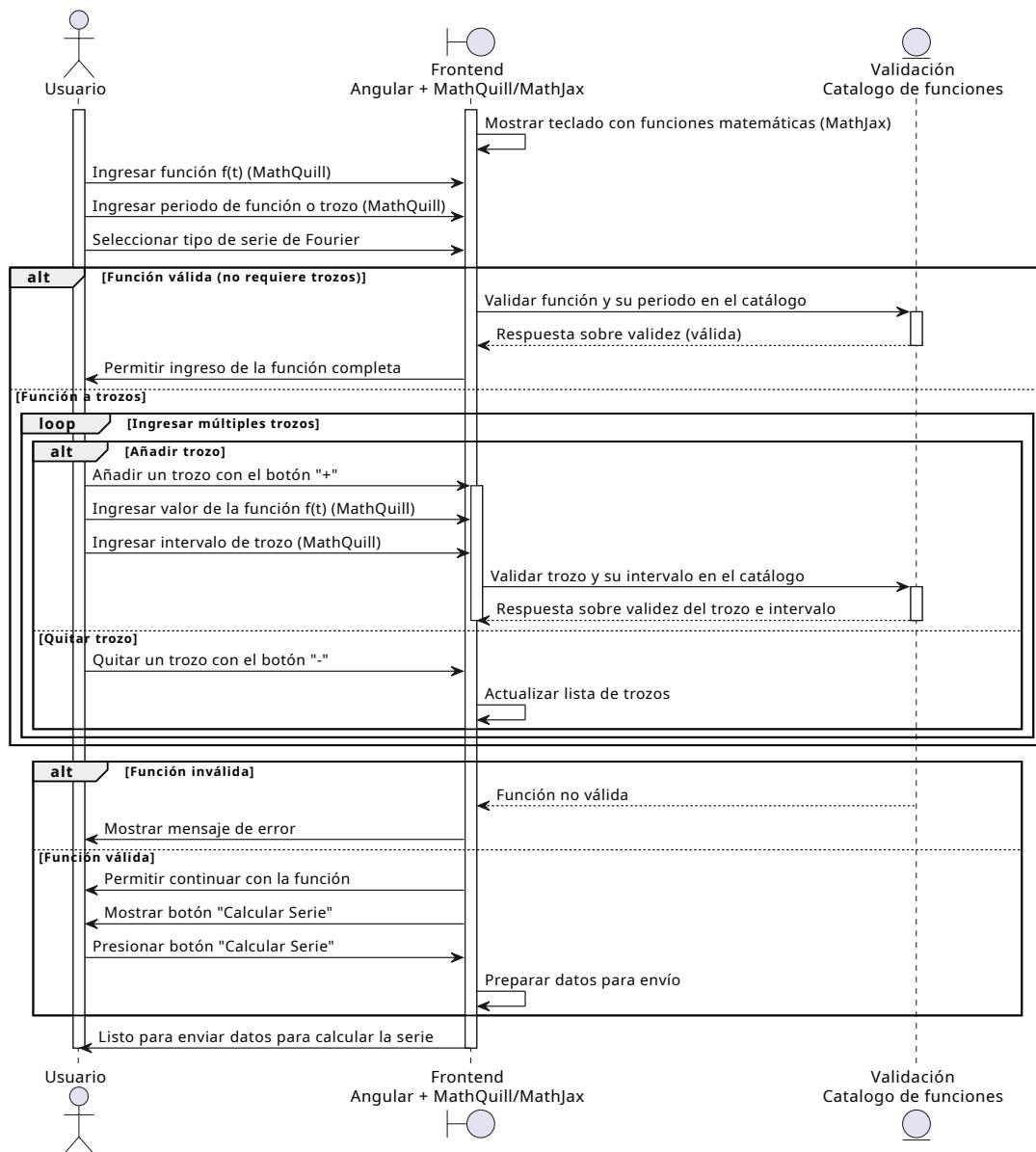


Figura 5.3: Diagrama de Secuencia 1. Fuente: Elaboración propia

5.4.2. Parseo de función para la API

El diagrama de la figura ?? ilustra como, una vez confirmada la función ingresada por el usuario, se parsea la expresión y sus intervalos de formato LaTeX y a Maxima utilizando la librería de parseo. Luego, los datos se estructuran en JSON indicando la función, sus intervalos y el tipo de extensión y se envían al la API para realizar los cálculos.

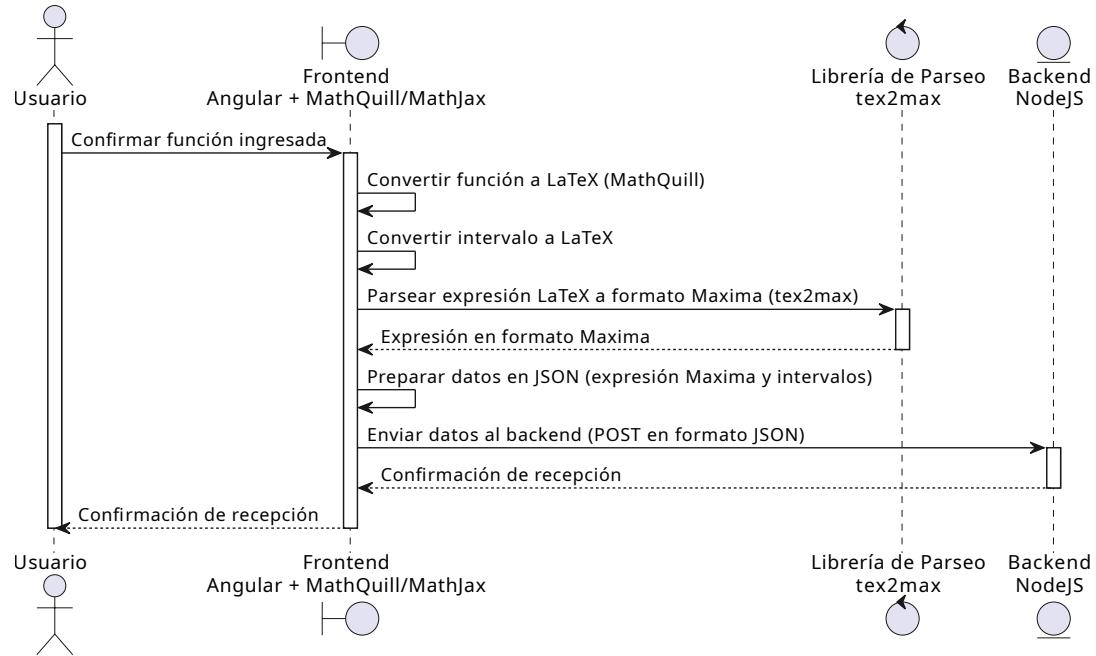


Figura 5.4: Diagrama de Secuencia 2. Fuente: *Elaboración propia*

5.4.3. Cálculo de la Serie de Fourier en la API

En el diagrama de la figura ?? se detalla el proceso en la API para calcular la serie de Fourier. Dependiendo del tipo de serie seleccionado (trigonométrica o exponencial), se calculan los coeficientes correspondientes. Se realizan integraciones en cada intervalo si la función es a trozos y finalmente se generan los resultados, tanto en formato Maxima como en formato LaTeX para su visualización.

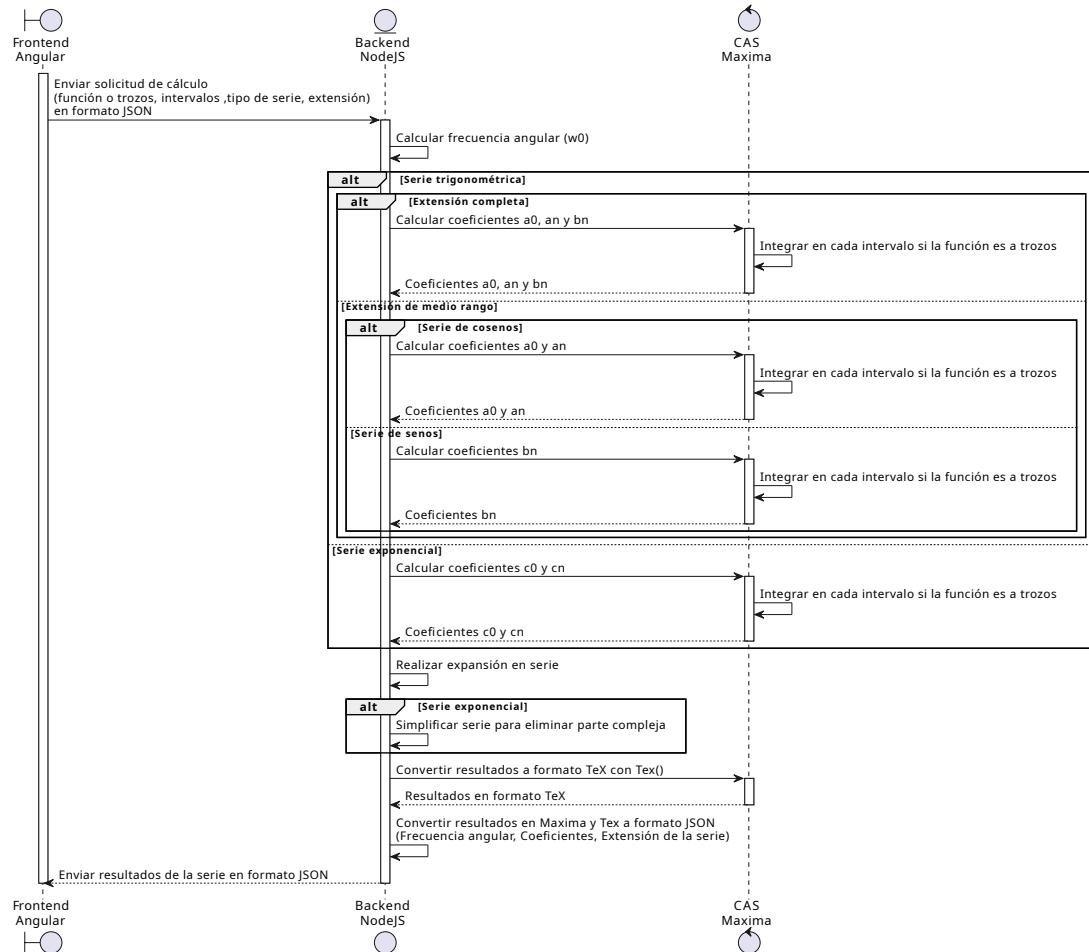


Figura 5.5: Diagrama de Secuencia 3. Fuente: *Elaboración propia*

5.4.4. Preparación de resultados para visualización

El diagrama de la figura ?? muestra cómo se preparan los resultados para su visualización en el frontend. Los datos recibidos del backend en formato JSON se deserializan y se convierten en formatos JavaScript y LaTeX, listos para ser consumidos por el usuario en diferentes para la graficación y ver los resultados en su notación matemática, respectivamente.

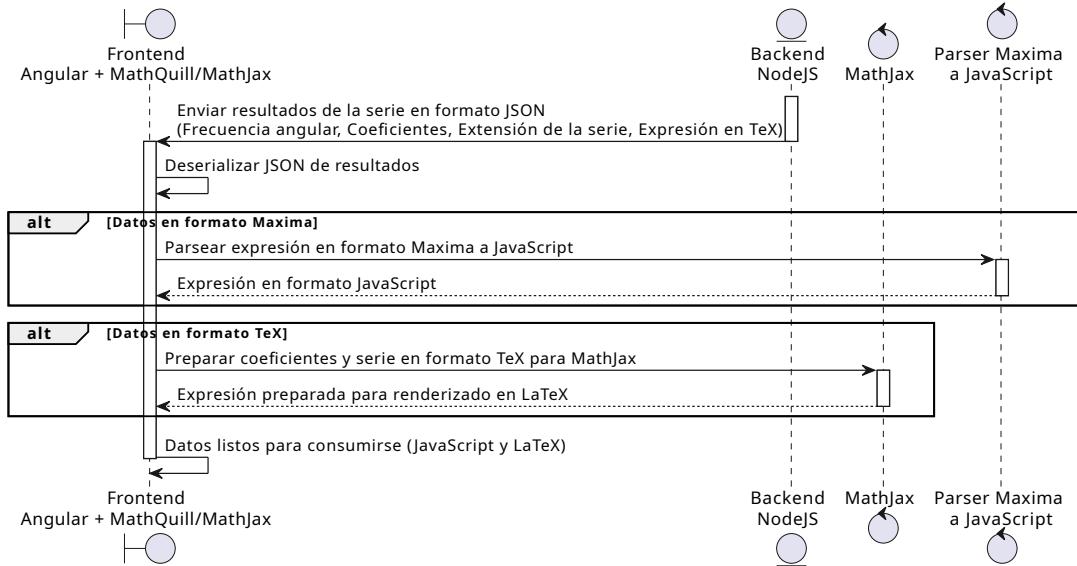


Figura 5.6: Diagrama de Secuencia 4. Fuente: *Elaboración propia*

5.4.5. Visualización de los resultados e interactividad con la gráfica

El diagrama de la figura ?? representa cómo el frontend toma los resultados para para graficar la serie de Fourier de manera interactiva, permitiendo al usuario ajustar el número de términos y ver la actualización de la gráfica en tiempo real. Además, los coeficientes y el resultado en forma de serie se muestran en formato matemático usando MathJax.

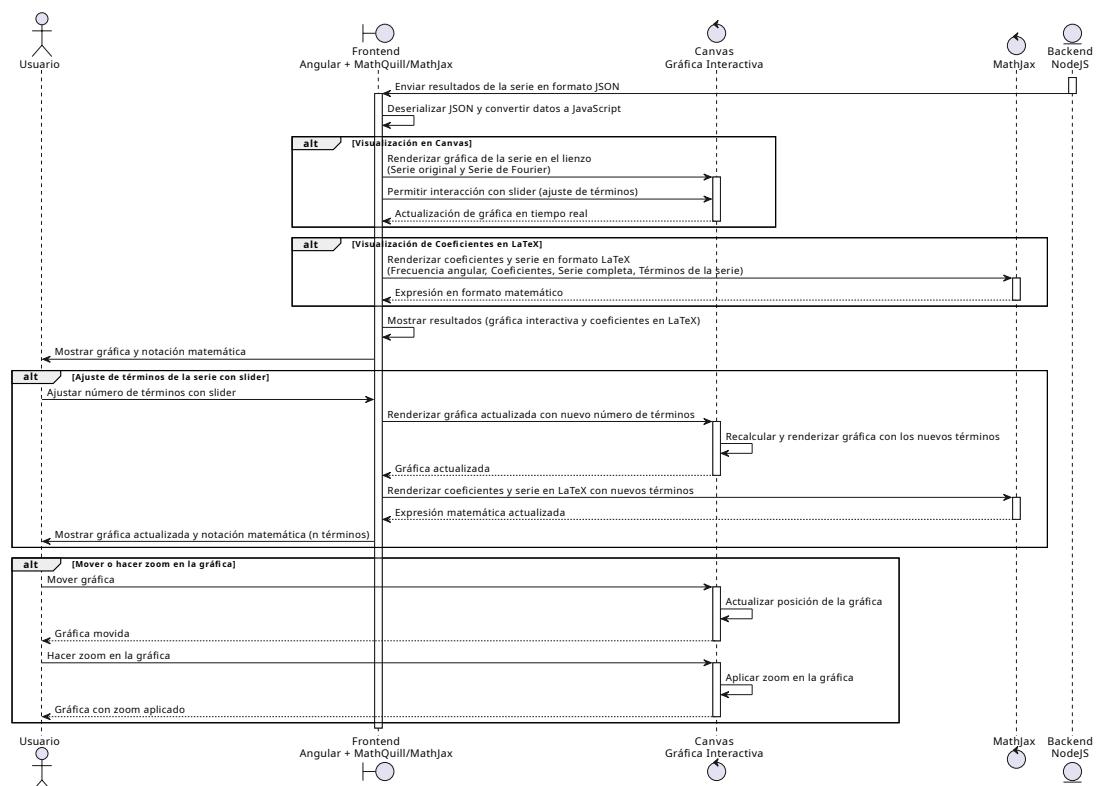


Figura 5.7: Diagrama de Secuencia 5. Fuente: *Elaboración propia*

5.5. Diagramas de clases UML

A pesar de que este proyecto no utiliza el paradigma de programación orientada a objetos (POO), se maneja un objeto que es crucial para la comunicación entre el cliente y el servidor, el JSON, ya que se usa como un objeto de transferencia de datos. Para la arquitectura planteada para el proyecto, se cuenta con dos JSON, el que envía el cliente al servidor (Request) y el que envía el servidor al cliente (Response).

5.5.1. Diagrama de clases ClientRequest

ClientRequest es la clase principal que representa el JSON que el cliente envía al servidor, en el, se captura toda la información necesaria para que el servidor pueda calcular la serie de Fourier solicitada.

La estructura de la clase principal **ClientRequest** y sus relaciones es la siguiente:

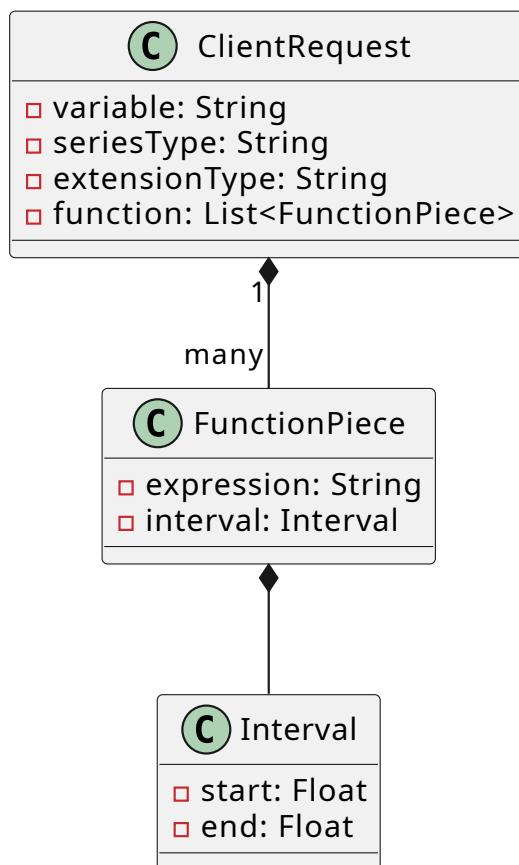


Figura 5.8: Diagrama de clases del JSON del cliente. Fuente: *Elaboración propia*

En donde la clase principal **ClientRequest** tiene los siguientes atributos:

- **variable**: La variable independiente de la función (**String**) que puede ser **x** o **t**.
- **seriesType**: El tipo de Serie de Fourier (**String**) que puede ser **trigonometric** o **complex**.

- **extensionType**: Define si es una extensión par, impar o periódica (String) donde los valores pueden ser `odd`, `even` o `periodic`.
- **function**: Lista de objetos `FunctionPiece` que representan los trozos de la función (`List<FunctionPiece>`).

La clase **FunctionPiece** tiene los siguientes atributos:

- **expression**: La expresión matemática de un segmento de la función (String).
- **interval**: Objeto de tipo `Interval` que define el intervalo de validez del segmento.

La clase **Interval** tiene los siguientes atributos:

- **start**: Inicio del intervalo (Float).
- **end**: Fin del intervalo (Float).

Las relaciones que se tienen en estas clases son:

- `ClientRequest` tiene una relación de composición con `FunctionPiece`, indicando que un `ClientRequest` puede contener múltiples segmentos de función.
- `FunctionPiece` tiene una relación de composición con `Interval`, indicando que cada segmento de función tiene un intervalo definido.

5.5.2. Diagrama de clases ServerResponse

`ServerResponse` es la clase principal que representa el JSON que el servidor devuelve al cliente después de calcular la Serie de Fourier.

La estructura de la clase principal `ServerResponse` es la siguiente:

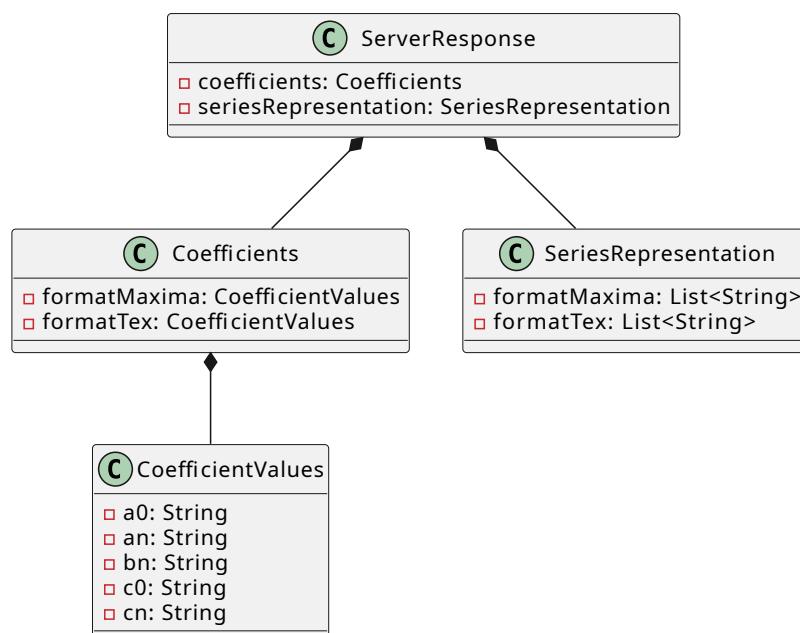


Figura 5.9: Diagrama de clases del JSON del servidor. Fuente: Elaboración propia

En donde la clase principal **ServerResponse** tiene los siguientes atributos:

- **coefficients**: Objeto de tipo **Coefficients** que contiene los valores de los coeficientes calculados.
- **seriesRepresentation**: Objeto de tipo **SeriesRepresentation** que incluye las representaciones de la serie calculada.

La clase **Coefficients** tiene los siguientes atributos:

- **formatMaxima**: Objeto de tipo **CoefficientValues** que contiene los coeficientes en formato Maxima.
- **formatTex**: Objeto de tipo **CoefficientValues** que contiene los coeficientes en formato LaTeX.

La clase **CoefficientValues** tiene los siguientes atributos:

- **a0**: Coeficiente promedio constante (**String**).
- **an**: Coeficiente para los componentes coseno (**String**).
- **bn**: Coeficiente para los componentes seno (**String**).
- **c0**: Coeficiente promedio constante complejo (**String**).
- **cn**: Coeficiente complejo (**String**).

La clase **SeriesRepresentation** tiene los siguientes atributos:

- **formatMaxima**: Representación de la serie en formato Maxima (**List<String>**).
- **formatTex**: Representación de la serie en formato LaTeX (**List<String>**).

Las relaciones que se tienen en estas clases son:

- **ServerResponse** tiene una relación de composición con **Coefficients** y **SeriesRepresentation**.
- **Coefficients** tiene una relación de composición con **CoefficientValues**, indicando que contiene coeficientes en diferentes formatos.

Estos dos diagramas nos dan la estructura de como es que los datos que se envían y reciben entre el cliente y el servidor.

5.6. Diseño de la interfaz de usuario

El diseño de las interfaces de usuario proporciona una representación visual clara de las funcionalidades principales del sistema. A continuación, se presentan los mockups de las dos interfaces principales: la interfaz de ingreso de datos y la interfaz de gráficos interactivos.

5.6.1. Interfaz de ingreso de datos

Esta interfaz permite al usuario ingresar la función a la cual se le calculará su serie de Fourier, configurar los intervalos de cada sección de la función en caso de añadir más trozos a la misma, además de visualizar la función. Podrá seleccionar si la serie será trigonométrica o exponencial y si es trigonométrica podrá seleccionar si se trata de una expansión de medio rango. Además, incluye un teclado en pantalla para facilitar la entrada de funciones matemáticas.

Calculadora de Series de Fourier

Función $f(x)$:

$$f(x) = \begin{cases} x & , \quad 0 < x < \pi \\ 0 & , \quad \pi < x < 2\pi \end{cases}$$

Elige el tipo de serie:

Trigonométrica

¿Extensión de Medio Rango?

Elige la Extensión de Medio Rango:

Extensión Par (Serie de Coseno)

Enviar

Figura 5.10: Interfaz de ingreso de datos. *Fuente: Elaboración propia*

5.6.2. Interfaz de gráficos interactivos

En esta interfaz, el usuario puede visualizar e interactuar con la gráfica de la función original y su aproximación mediante la Serie de Fourier. Incluye herramientas como un control deslizante para ajustar el número de términos de la serie y opciones para hacer zoom o mover la gráfica. Además de visualizar el resultado en notación matemática.

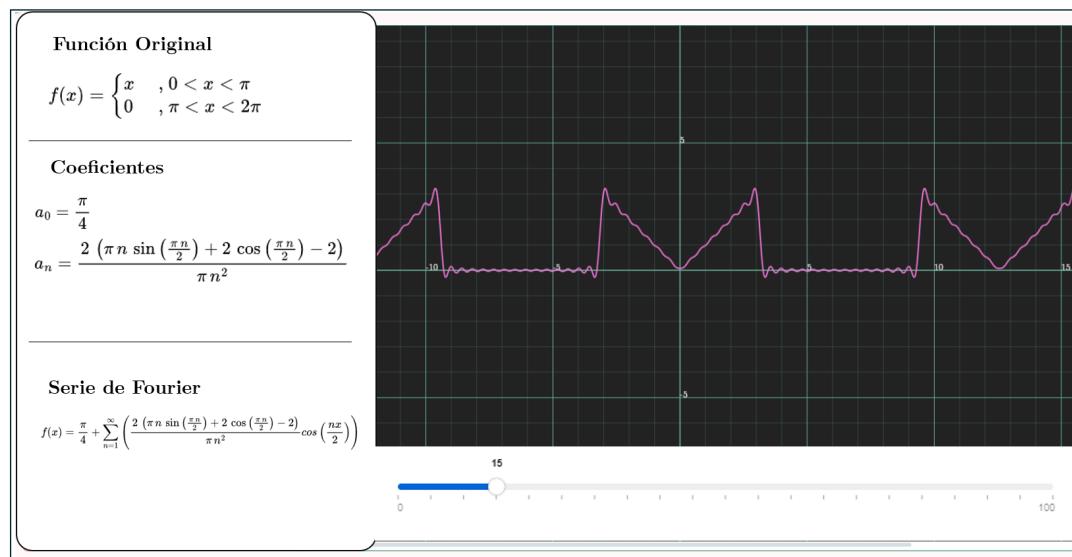


Figura 5.11: interfaz de ingreso de graficación. *Fuente: Elaboración propia*

CAPÍTULO 6

Implementación (Primer Prototipo)

El propósito de la elaboración del primer prototipo fue construir pequeños módulos funcionales para hacer pruebas unitarias con ellos y, posteriormente, integrados para garantizar su interoperatividad. Debido a la complejidad técnica del proyecto y la necesidad de utilizar diversas bibliotecas, surgieron dudas sobre si estas herramientas funcionarían de manera correcta e integrada. Además, al ser un proyecto desarrollado por un único miembro, cualquier problema podría comprometer el avance y la viabilidad del proyecto. Este enfoque permitió validar el funcionamiento independiente de cada módulo y asegurar que el proyecto pudiera avanzar sin bloqueos técnicos. El desarrollo del primer prototipo siguió una serie de etapas que se describen a continuación:

6.1. Construcción inicial del Canvas con JavaScript

EL primer punto clave que se atacó del proyecto fue la implementación de todo el plano donde se hará la visualización interactiva de las funciones y series, para esto, se usó el elemento `<canva>` de HTML, y fué modificado meramente con Javascript para darle su funcionalidad.

6.1.1. Funciones para graficar funciones y series de Fourier

Para lograrlo, se modifico el componente `<canva>` desde Javascript (véase Apéndice C) para lograr dibujar el lienzo y darle toda su funcionalidad de poder moverse y hacer zoom en él. Además de funciones para poder dibujar sobre ese lienzo gráficas de funciones y series de funciones.

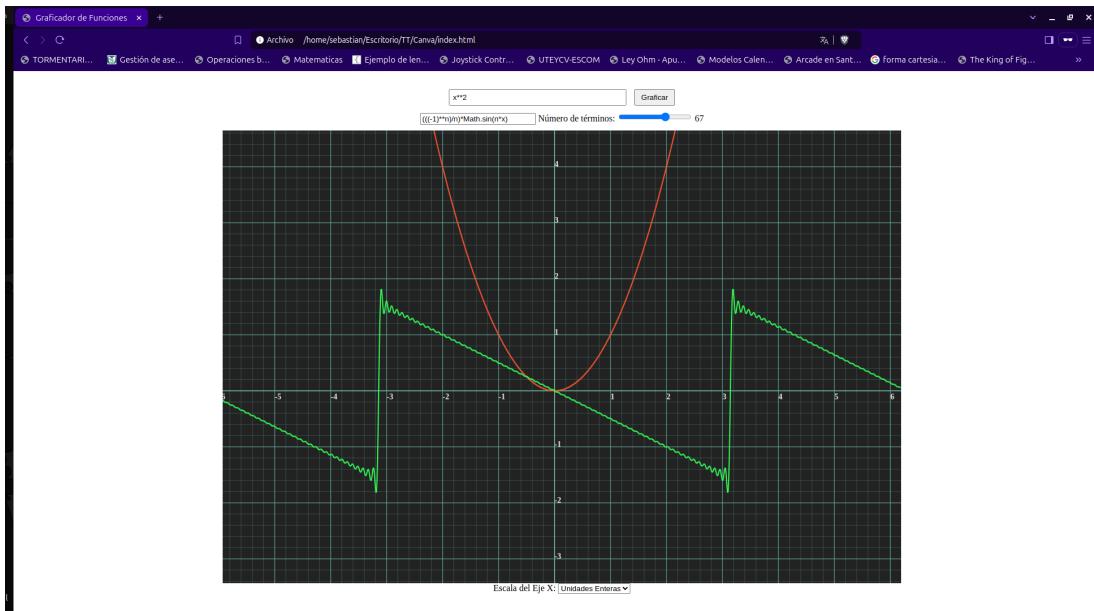


Figura 6.1: Plano hecho con un elemento canva. *Fuente: Elaboración propia*

6.2. Programas en Maxima para cálculo matemático

Lo siguiente fue enfocarse en hacer los cálculos de los coeficientes de la serie trigonométrica y la extensión de su serie para obtener cada término (??) ??.

Figura 6.2: Código en Maxima para calcular los coeficientes de la serie trigonométrica y L^AT_EX. Fuente: Elaboración propia

Sin embargo, la preocupación se tornaba en la serie compleja, ya que al tener números imaginarios su graficación se tronaba complicada, la buena noticia es que Maxima podía separar las partes real e imaginaria de la serie y así, anular la

pare imaginaria dej\'andonos unicamente con su equivalente real, que si podemos graficar(??) ??:

The screenshot shows a Maxima 23.0.1 window with the following details:

- Title Bar:** "maxima-23.0.1 (Windows 11 Compilación 2363), edición de 64 bits [medio_rango_Compleja.wxm]".
- Menu Bar:** Archivo, Editor, Vista, Celda, Maxima, Ecuaciones (Q), Matriz, Análisis, Simplificar, Listado, Tramado, Numérico, Ayuda.
- Toolbar:** Contains icons for New, Open, Save, Print, Copy, Paste, Cut, Undo, Redo, Find, Replace, and Help.
- Text Cell:** Shows the input code for calculating the sum of a series of terms involving trigonometric functions and binomial coefficients.
- Symbolic Mathematics Panel:** Displays the step-by-step derivation of the formula, showing intermediate results and simplifications.
- Tramado (Tramado Diágrámico) Panel:** Shows the graphical representation of the function being analyzed.
- Precisión (Precision) Panel:** Set to 10.
- Status Bar:** "Maxima" preparada para la entrada.

Figura 6.3: Código en Maxima para calcular los coeficientes de la serie compleja y LATEX. Fuente: *Elaboración propia*

6.3. Creación de una API en Node.js para cálculo

Una vez comprobamos que Maxima es una gran opción para hacer los cálculos, la siguiente parte fue implementar esta tecnología en un lenguaje de servidor para poder llamarlo desde otros proyectos, es decir, crear una API que ejecute a Maxima, para esto se usó NodeJS (??), ya que al ser un entorno de ejecución en Javascript resultaba ideal por la curva de aprendizaje no tan alta.

6.3.1. Pruebas de la API con Postman

Una vez se tenía la API desarrollada, era turno de probarla, para esto, se usó Postman, para enviar peticiones a la API para solicitar el cálculo de los coeficientes de la serie trigonométrica ?? y exponencial ??.

The screenshot shows a POST request to `http://localhost:3000/fourier/entirely-continuous/trigonometric`. The request body is:

```

1 {
2   "funcion": "(x**3)+2*x**2+3",
3   "periodo": "2*pi"
4 }

```

The response body is:

```

1 {
2   "a0": "(4*pi^2+18)/3",
3   "an": "(8*(-1)^n)/n^2",
4   "bn": "-((2*pi^2*n^2-12)*(-1)^n)/n^3"
5 }

```

Figura 6.4: Prueba de la API en el endpoint para calcular los coeficientes de la serie trigonométrica. *Fuente: Elaboración propia*

The screenshot shows a POST request to `http://localhost:3000/fourier/entirely-continuous/complex`. The request body is:

```

1 {
2   "funcion": "(x**3)+2*x**2+3",
3   "periodo": "2*pi"
4 }

```

The response body is:

```

1 {
2   "c0": "(2*pi^2+9)/3",
3   "cn": "((6*i*pi^2*n^2+4*n-6*i)*(-1)^n)/n^3"
4 }

```

Figura 6.5: Prueba de la API en el endpoint para calcular los coeficientes de la serie compleja. *Fuente: Elaboración propia*

6.4. Pruebas de entradas matemáticas en formato LATEXcon MathQuill

Ya teniendo como calcular y graficar, tocó el turno de comenzar con los prototipos de interfaces de usuario, para esto, se propuso MathQuill (??)para capturar los datos, al usar TeXpermite una visualización mas agradable de los datos.



Figura 6.6: Prueba de la biblioteca de MathQuill para el ingreso de funciones.
Fuente: Elaboración propia

6.5. Creación de un teclado usando MathJax

Al tener el campo para que el usuario ingrese los datos de su función, surgió el dilema de que los usuarios no tienen porque saber TeX, así que, usando la biblioteca de MathJax (??), que nos permite a nosotros como desarrolladores, escribir texto en TeXen nuestro sitio web, con esto en las manos, se diseñó un teclado imitando al de una calculadora, para que el usuario pueda ingresar las funciones válidas y más comunes para las series de Fourier a solo un botón de distancia ??.

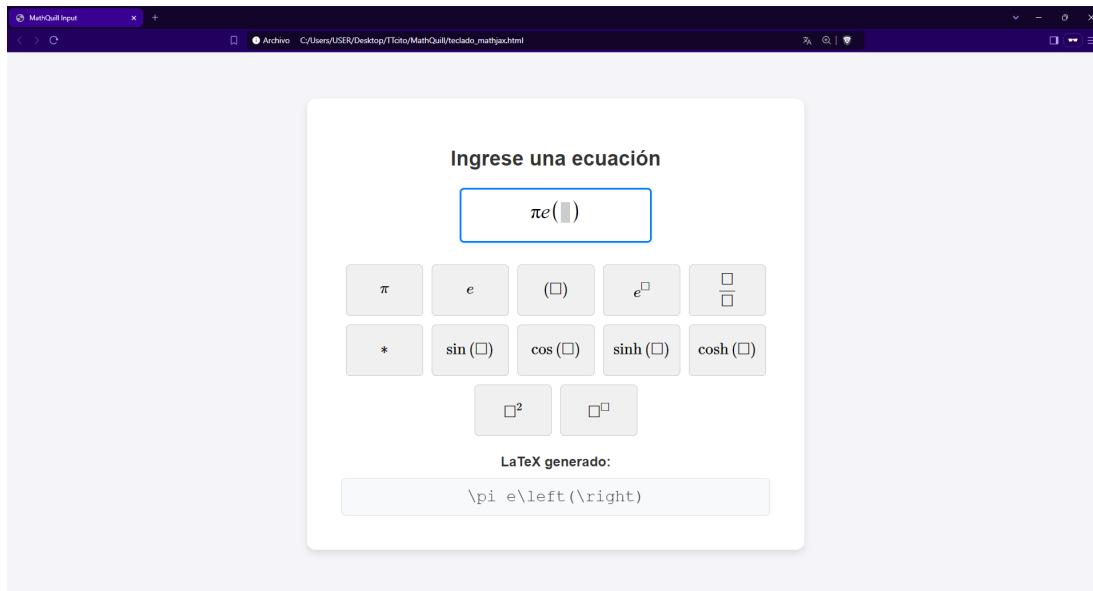


Figura 6.7: Prueba de la biblioteca de MathQuill y un teclado hecho en MathJax para el ingreso de funciones. *Fuente: Elaboración propia*

6.6. Prototipo de interfaz de ingreso de funciones

Ya con lo necesario para una interfaz de ingreso de datos, se comenzó con el diseño inicial de nuestra interfaz para que los usuarios puedan ingresar funciones matemáticas, configurar parámetros y enviar datos para poder calcular una serie de Fourier.

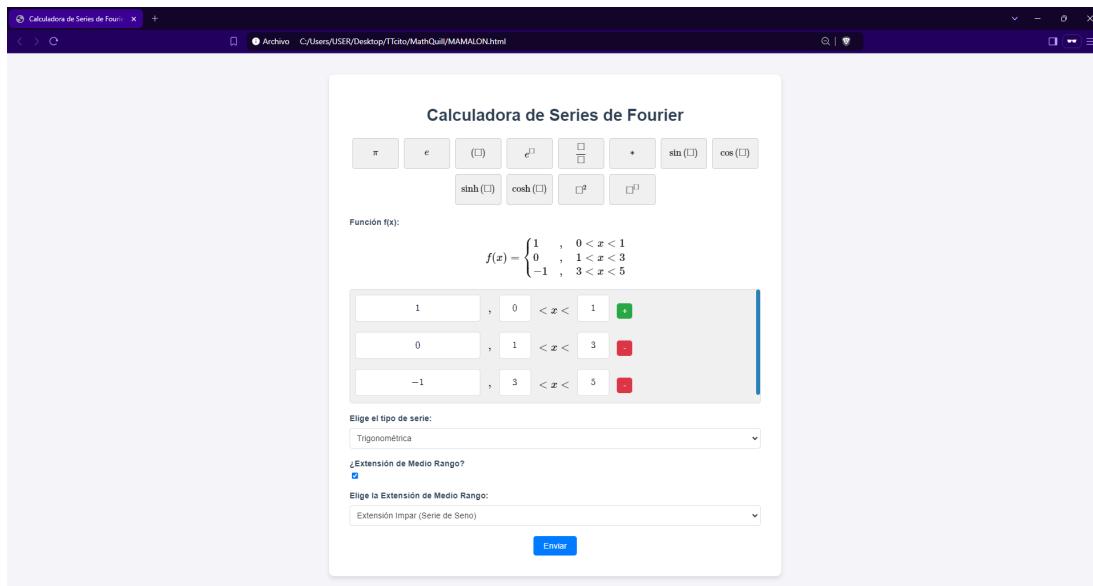
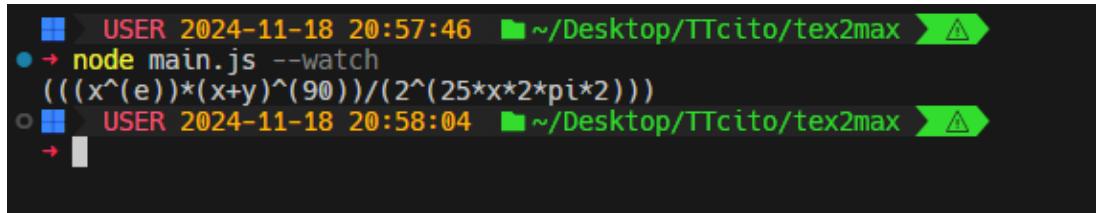


Figura 6.8: Prototipo de la interfaz de ingreso de funciones. *Fuente: Elaboración propia*

6.7. Pruebas de parseo con tex2max

Ya teniendo como ingresar los valores, se cayó en cuenta de que lo que envían estos campos son valores en \TeX , y nuestra API funciona con Maxima, aquí es donde se hicieron pruebas con una biblioteca independiente para poder hacer este parseo (??), para comprobar que de verdad este componente funcionaba y sería de utilidad para este proyecto ??.



```

USER 2024-11-18 20:57:46 ~/Desktop/TTcito/tex2max
● → node main.js --watch
((x^e)*(x+y)^(90))/(2^(25*x^2*pi*2))
○ ■ USER 2024-11-18 20:58:04 ~/Desktop/TTcito/tex2max
→

```

Figura 6.9: Prueba de la biblioteca de tex2max sobre NodeJS. *Fuente: Elaboración propia*

6.8. Migración del Canvas a Angular

Ya teniendo todos los módulos contemplados para el funcionamiento de la aplicación, se procedió a migrar el primero de estos módulos en ser desarrollado, el lienzo al framework de Angular, esto debido a su arquitectura robusta basada en componentes, lo que facilita la reutilización de código y el mantenimiento del proyecto acelerando el proceso de desarrollo. Angular trabaja con un Typescript, un super conjunto de Javascript, por lo que su migración no se tornó complicada ??.

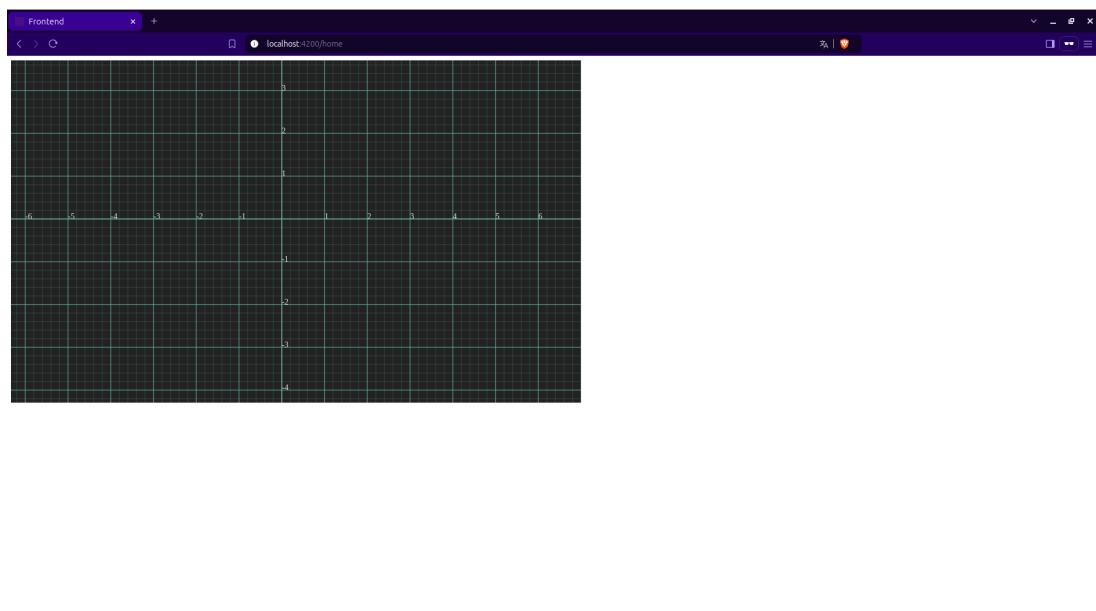
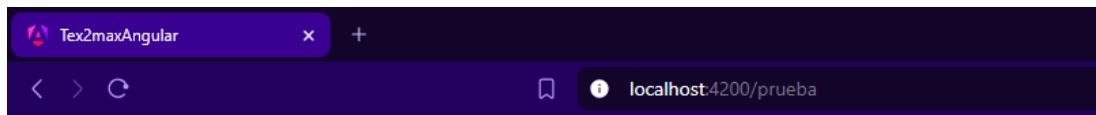


Figura 6.10: Plano hecho en canvas sobre Angular. *Fuente: Elaboración propia*

6.9. Pruebas de tex2max en Angular

La última prueba fue el probar una librería de terceros en el framework de Angular, ya que podrían darse incompatibilidades y si, al principio no se reconocía la biblioteca, la solución fue crear un archivo `.d.ts`, que es una declaración de TypeScript que proporciona información de tipos para bibliotecas JavaScript. Esto permite que permiten que TypeScript entienda las estructuras, funciones y objetos de una biblioteca externa, facilitando así su integración y uso dentro de una aplicación Angular (??).



LaTeX to Maxima Converter

```
\frac{\left( -1 \right)^n}{n!}c
```

Convert

Maxima Output

```
(((-1)^n)/(n*pi))*sin(x*pi)
```

Figura 6.11: Prueba de la biblioteca de tex2max en Angular. *Fuente: Elaboración propia*

CAPÍTULO 7

Trabajo a futuro

Las actividades planeadas para este proyecto durante el plazo del siguiente Trabajo Terminal II son las siguientes:

- **Completar la API:** Ya teniendo los primeros endpoints de la API, se desarrollarán los faltantes para calcular las extensiones de medio rango, así como procesar funciones a trozos. Además de tratar con excepciones.
- **Implementar forma Amplitud - Fase:** La API contará con un nuevo endpoint para calcular la serie de Fourier en su forma de amplitud - fase, así como crear las funciones para graficar este tipo de funciones en el lienzo.
- **Diseñar interfaz para el lienzo:** Se creará una interfaz para colocar el lienzo, además de secciones para mostrar los resultados y los controles para calcular la serie de Fourier.
- **Crear un catalogo de funciones comunes:** A lo largo del estudio de las series de Fourier se presentan problemas comunes como la función diente de sierra, la función escalón, la función piso, un tren de pulsos, estas funciones debe estar en un catalogo para calcularse inmediatamente.
- **Implementar un parser Maxima a Javascript:** Se diseñará e implementara un parser para convertir las cadenas de formato Maxima a formato Javascript, para que el lienzo pueda interpretar y graficar los resultados devueltos por la API.
- **Unificar todos los módulos:** Ya teniendo los módulos de graficación, API de calculo simbólico, parseo de datos, e interfaz de ingreso de datos, todas se unificarán en un solo proyecto de Angular para que operen entre si.
- **Despliegue y pruebas:** Ya teniendo la apliación funcionando, se desplegará sobre la plataforma de servicios de nube de Microsoft Azure, además de obtener un dominio temporal y un certificado SSL para ser compartida con alumnos y profesores para probar su funcionalidad y rendimiento.

CAPÍTULO 8

Conclusiones parciales

El proyecto “Desarrollo de una aplicación web para el cálculo y graficación de series de Fourier” ha mostrado, según los resultados de una encuesta aplicada, ser de potencial utilidad para los estudiantes de áreas de ingeniería en asignaturas relacionadas con el análisis de Fourier. En esta primera etapa del proyecto, se han sentado las bases necesarias para confirmar que su desarrollo puede llevarse a cabo sin problemas de compatibilidad entre los diferentes módulos involucrados.

Los principales retos enfrentados durante el desarrollo han sido la integración de diversas tecnologías, así como la comprensión de sus características y la adaptación para que trabajen de manera conjunta. Estos desafíos han requerido una adecuada administración del tiempo, considerando la carga de otras actividades y factores externos, lo que ha sido crucial para avanzar en el proyecto.

El trabajo realizado hasta ahora demuestra que el proyecto es viable y cuenta con el potencial de ser una herramienta educativa valiosa para complementar el aprendizaje en el análisis de Fourier.

APÉNDICE A

Encuesta

A.1. Datos Generales

Estas preguntas contienen las características generales de los participantes, con el propósito de contextualizar los resultados obtenidos.

1. ¿Cuál es tu rol dentro de tu escuela?
82 respuestas

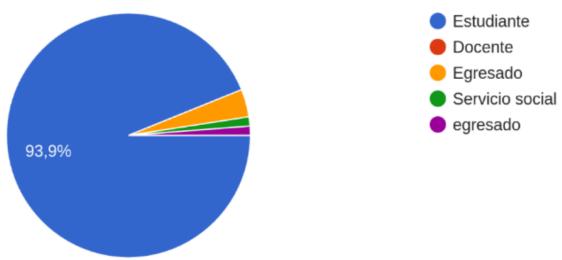


Figura A.1: Roles de los encuestados. *Fuente: Elaboración propia*

2. ¿En que escuela te encuentras? Ex: ESCOM, ESIME, ESFM, etc...
82 respuestas

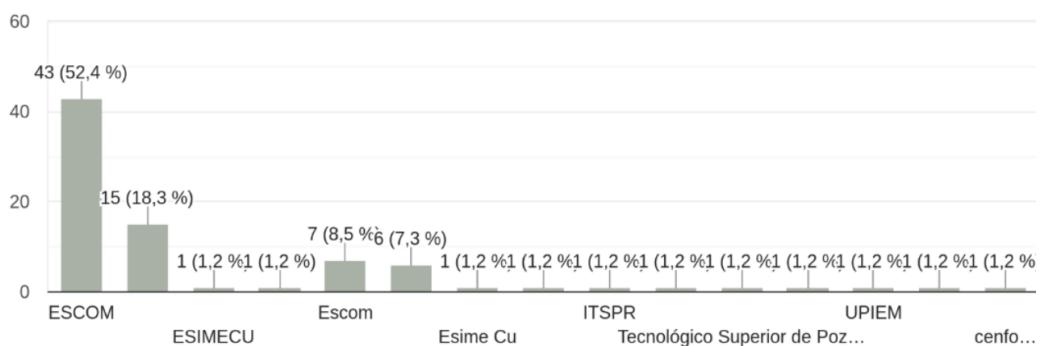


Figura A.2: Cantidad de personas encuestadas por escuela *Fuente: Elaboración propia*

3. ¿Qué ha sido lo más difícil cuando tuviste que estudiar asignaturas de matemáticas (Cálculo, Análisis Vectorial, ...)?
82 respuestas

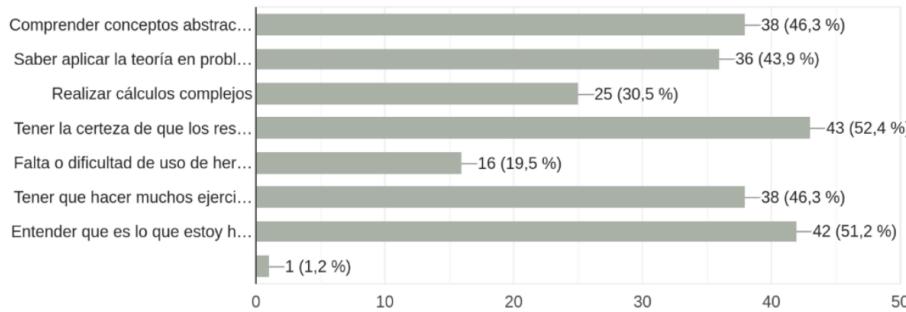


Figura A.3: Problemas más comunes de los encuestados al cursar asignaturas de matemáticas. *Fuente: Elaboración propia*

4. ¿Te has visto en la necesidad de usar software de cálculo o graficación para estas asignaturas? ¿Con cuáles?
82 respuestas

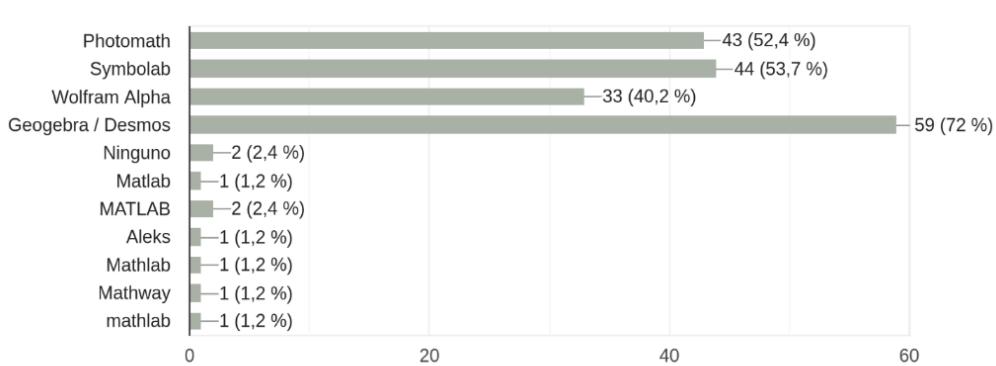


Figura A.4: Sistemas generales usados por los alumnos para apoyarse en asignaturas de matemáticas. *Fuente: Elaboración propia*

5. ¿Qué características consideras esenciales en una herramienta digital para áreas de matemáticas
82 respuestas

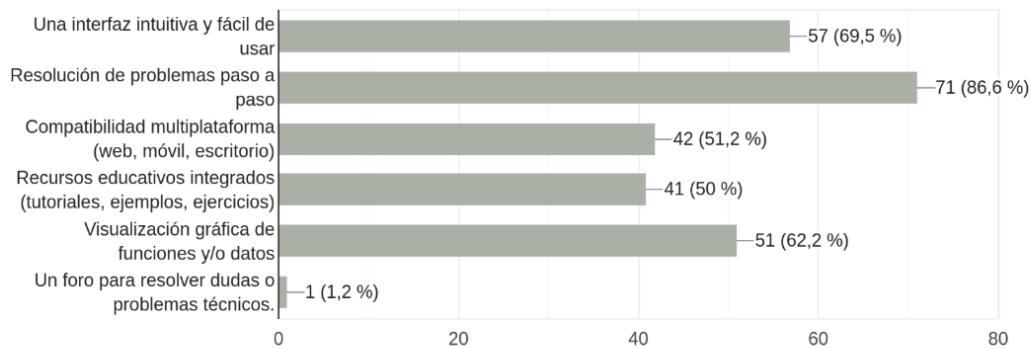


Figura A.5: Elementos que los encuestados consideran necesarios en herramientas auxiliares para las matemáticas. *Fuente: Elaboración propia*

6. ¿Alguna vez te has encontrado o tenido algún acercamiento con temas como "Análisis de Fourier", "Series de Fourier" o "Transformada de Fourier"? (Por ejemplo, en alguna materia o curso)
82 respuestas

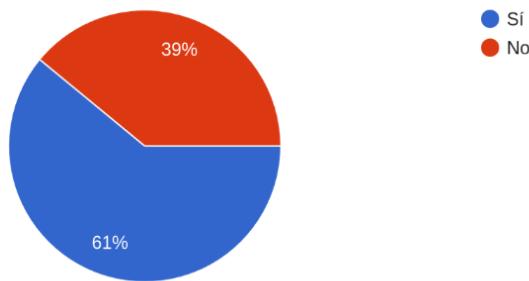


Figura A.6: Cantidad de encuestados que han trabajado con análisis de Fourier. *Fuente: Elaboración propia*

A.2. Participantes con experiencia en Fourier

Se muestran las respuestas de los participantes que han trabajado previamente con problemas relacionados con series de Fourier, destacando las dificultades encontradas y sus preferencias en herramientas.

7. Cuando trabajaste con las series de Fourier, ¿con qué problemas te enfrentaste?

50 respuestas

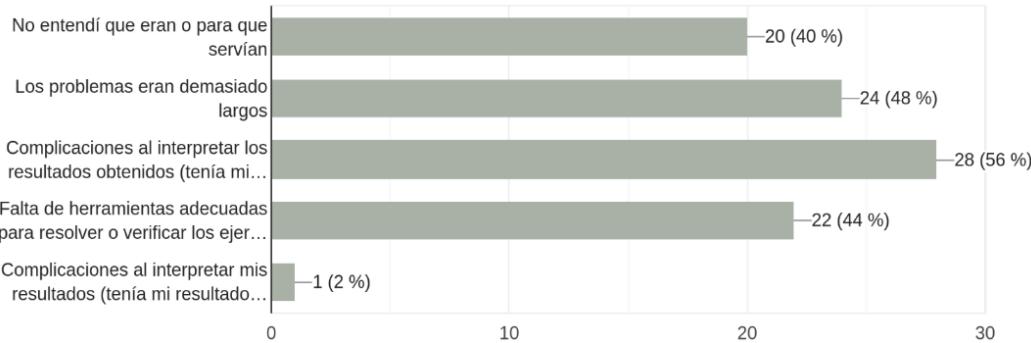


Figura A.7: Problemas que han tenido los encuestados al trabajar con problemas sobre series de Fourier.*Fuente: Elaboración propia*

8. ¿Qué herramientas o recursos usaste para aprender a resolver problemas con series de Fourier?

50 respuestas

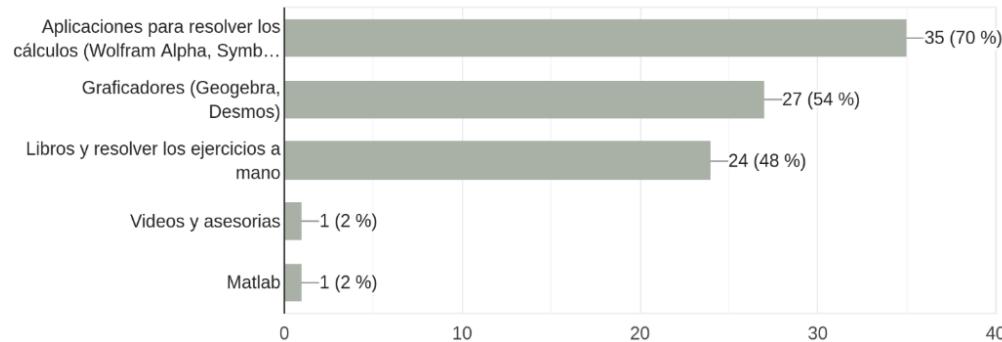


Figura A.8: Herramientas más comunes que han usado los encuestados como auxiliares para resolver series de Fourier.*Fuente: Elaboración propia*

9. En una escala del 1 al 10, ¿En que grado consideras que una herramienta que grafique automáticamente las series de Fourier y te dé los resultados te hubiera ayudado a entender mejor el tema?
 50 respuestas

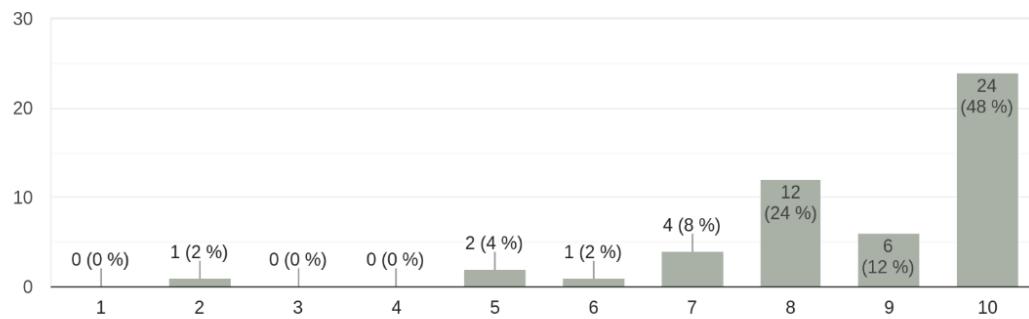


Figura A.9: Escala que muestra la cantidad de encuestados que consideran que una herramienta especializada en series de Fourier les hubiera ayudado en sus cursos previos.*Fuente: Elaboración propia*

10. ¿Qué características crees que serían útiles ver en una aplicación web para trabajar con series de Fourier?
 50 respuestas

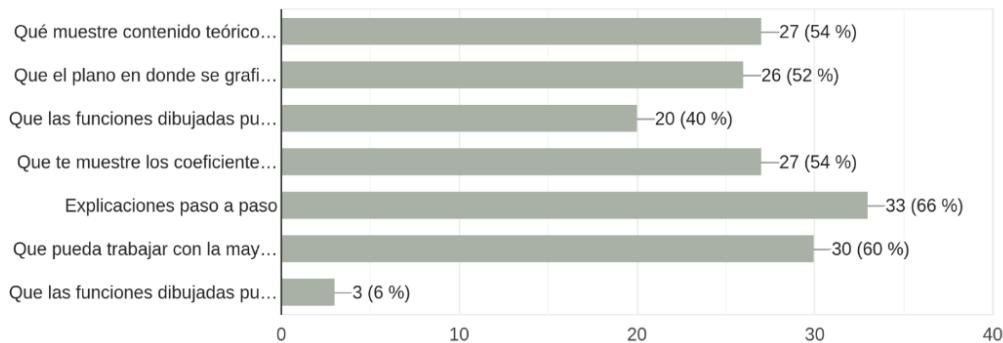


Figura A.10: Características útiles que los usuarios encuentran en una aplicación especializada en series de Fourier.*Fuente: Elaboración propia*

A.3. Participantes sin experiencia en Fourier

Se muestran las respuestas de los participantes que no tienen experiencia previa con series de Fourier, incluyendo sus expectativas y preferencias para una herramienta digital que facilite el aprendizaje.

7. En una escala del 1 al 10, ¿Qué tan difícil te parece este concepto? En base a las gráficas y ecuaciones mostradas en el vídeo?

32 respuestas

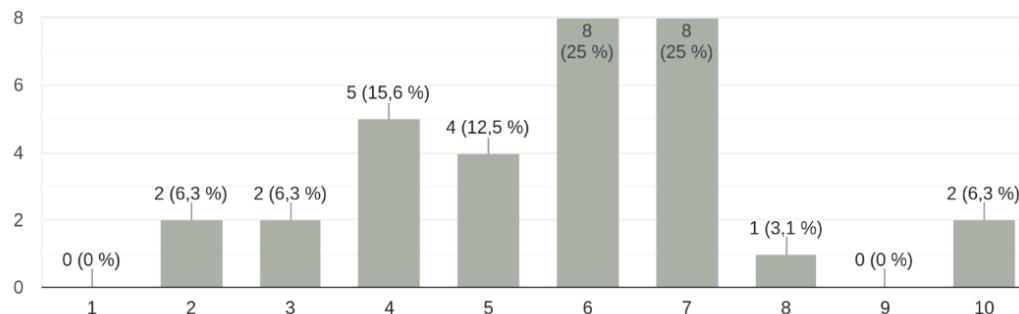


Figura A.11: Opinión de los encuestados sin conocimiento de Fourier sobre un vídeo hecho con Manim. *Fuente: Elaboración propia*

8. Si existiera una aplicación para resolver y visualizar estos problemas, ¿Qué piensas que sería bueno que este tuviera?

32 respuestas

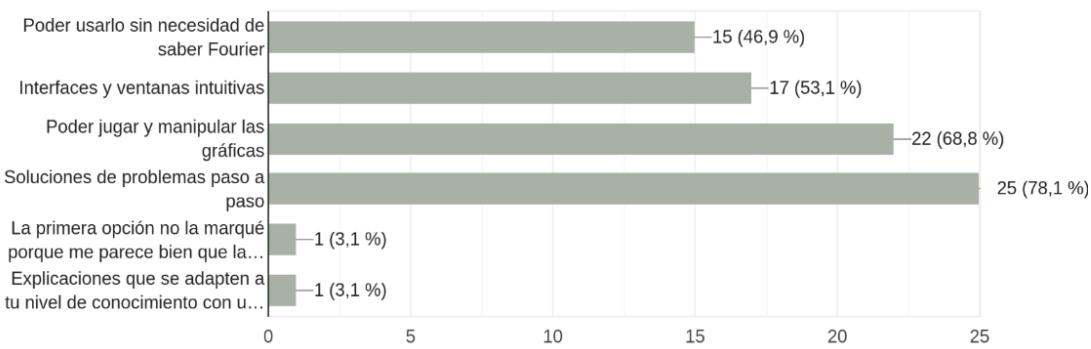


Figura A.12: Elementos que los encuestados sin conocimiento previo de Fourier considerarían útiles en una aplicación dedicada a este tema. *Fuente: Elaboración propia*

A.4. Opinión sobre la utilidad de herramientas digitales

Se muestran las respuestas de ambos grupos sobre la percepción de la utilidad de una herramienta digital interactiva para resolver y visualizar series de Fourier, así como su valoración de la visualización gráfica.

En una escala del 1 al 10, ¿En que medida crees que la visualización con gráficos y la interactividad con estos (como alterar datos, visualizar cambios...o) te ayude a comprender lo que estás haciendo?
82 respuestas

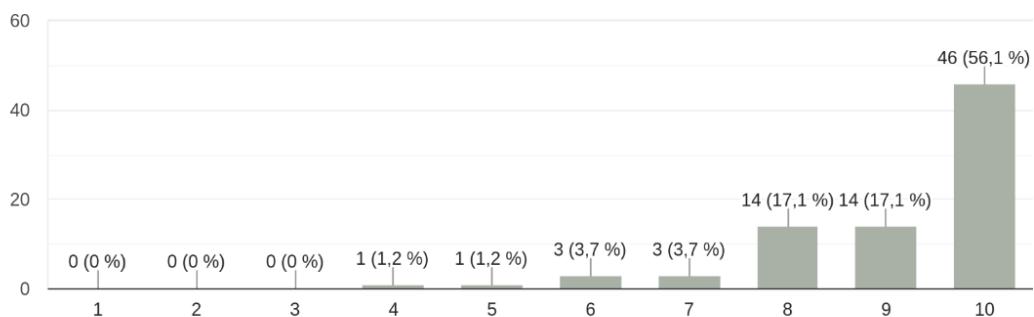


Figura A.13: Escala que muestra el grado de importancia que los encuestados dan a la graficación e interactividad ayuden a aprender. *Fuente: Elaboración propia*

APÉNDICE B

Cálculos

Dada la función $f(x) = x$, vamos a calcular los coeficientes correspondientes a su serie de Fourier en el intervalo $[-\pi, \pi]$.

B.1. Forma Trigonométrica

La serie trigonométrica de Fourier para una función $f(x)$ está dada por la siguiente expresión:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 x) + b_n \sin(n\omega_0 x))$$

Donde los coeficientes a_0 , a_n y b_n se definen como:

$$\begin{aligned} a_0 &= \frac{2}{T} \int_{-T/2}^{T/2} f(x) dx \\ a_n &= \frac{2}{T} \int_{-T/2}^{T/2} f(x) \cos(n\omega_0 x) dx \\ b_n &= \frac{2}{T} \int_{-T/2}^{T/2} f(x) \sin(n\omega_0 x) dx \end{aligned}$$

Donde $\omega_0 = \frac{2\pi}{T}$, y en este caso, $T = 2\pi$, por lo que $\omega_0 = 1$.

Calculamos cada uno de los coeficientes para $f(x) = x$:

- El coeficiente a_0 es:

$$a_0 = \frac{2}{2\pi} \int_{-\pi}^{\pi} x dx = \frac{1}{\pi} \left[\frac{x^2}{2} \right]_{-\pi}^{\pi} = \frac{1}{\pi} \left(\frac{\pi^2}{2} - \frac{(-\pi)^2}{2} \right) = 0$$

Por lo tanto, $a_0 = 0$.

- El coeficiente a_n es:

Dado que $x \cos(nx)$ es una función impar, ya que x es impar y $\cos(nx)$ es par, su integral en un intervalo simétrico es cero:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} x \cos(nx) dx = 0$$

- El coeficiente b_n es:

Dado que $x \sin(nx)$ es una función par (producto de dos funciones impares), podemos calcular:

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} x \sin(nx) dx = \frac{2}{\pi} \int_0^{\pi} x \sin(nx) dx$$

Aplicamos integración por partes, tomando $u = x$ y $dv = \sin(nx) dx$, lo que nos da $du = dx$ y $v = -\frac{1}{n} \cos(nx)$. Entonces:

$$\int_0^{\pi} x \sin(nx) dx = -\frac{x}{n} \cos(nx) \Big|_0^{\pi} + \frac{1}{n} \int_0^{\pi} \cos(nx) dx$$

Evaluamos el primer término:

$$-\frac{\pi}{n} \cos(n\pi) + 0 = -\frac{\pi}{n}(-1)^n$$

La integral restante es:

$$\frac{1}{n} \int_0^{\pi} \cos(nx) dx = \frac{1}{n} \left[\frac{\sin(nx)}{n} \right]_0^{\pi} = \frac{1}{n^2}(0 - 0) = 0$$

Por lo tanto, el coeficiente b_n es:

$$b_n = \frac{2}{\pi} \left(-\frac{\pi}{n}(-1)^n \right) = \frac{2(-1)^{n+1}}{n}$$

Entonces, la serie trigonométrica de Fourier para $f(x) = x$ es:

$$f(x) = 2 \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(nx)$$

B.2. Forma Exponencial Compleja

La serie exponencial compleja de Fourier está dada por la siguiente expresión:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega_0 x}$$

Donde los coeficientes c_n se definen como:

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(x) e^{-in\omega_0 x} dx$$

Calculamos el coeficiente c_n para $n \neq 0$:

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} x e^{-inx} dx$$

Aplicamos integración por partes con $u = x$ y $dv = e^{-inx} dx$, obteniendo $du = dx$ y $v = \frac{e^{-inx}}{-in}$. Entonces:

$$c_n = \frac{1}{2\pi} \left(\left[x \frac{e^{-inx}}{-in} \right]_{-\pi}^{\pi} - \int_{-\pi}^{\pi} \frac{e^{-inx}}{-in} dx \right)$$

El primer término es:

$$\left[x \frac{e^{-inx}}{-in} \right]_{-\pi}^{\pi} = \frac{\pi e^{-in\pi} - (-\pi)e^{in\pi}}{-in} = \frac{2\pi(-1)^n}{-in}$$

El segundo término es cero, ya que:

$$\int_{-\pi}^{\pi} e^{-inx} dx = 0$$

Por lo tanto:

$$c_n = \frac{1}{2\pi} \left(\frac{2\pi(-1)^n}{-in} \right) = \frac{i(-1)^n}{n}$$

Para $n = 0$:

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} x dx = 0$$

Así que la serie exponencial compleja de Fourier para $f(x) = x$ es:

$$f(x) = \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} \frac{i(-1)^n}{n} e^{inx}$$

Ambas representaciones son equivalentes y proporcionan la expansión de Fourier correcta para la función dada.

APÉNDICE C

Códigos

C.1. Código en Matlab para graficar la serie de Fourier trigonométrica

```
1 % Número de términos en la serie de Fourier
2 N = 10; % Puedes modificar N para mejorar la aproximación
3
4 % Vector de tiempo de -pi a pi
5 t = linspace(-pi, pi, 1000);
6
7 % Función original x(t) = t
8 x_original = t;
9
10 % Inicialización de la aproximación de la serie de Fourier
11 x_aprox = zeros(size(t));
12
13 % Cálculo de la serie de Fourier en forma trigonométrica
14 for n = 1:N
15 bn = (2 * (-1)^(n+1)) / n; % Coeficientes de seno
16 x_aprox = x_aprox + bn * sin(n * t);
17 end
18
19 % Gráfica de la función original y su aproximación
20 figure;
21 plot(t, x_original, 'k', 'LineWidth', 1.5); % Función original en negro
22 hold on;
23 plot(t, x_aprox, 'b', 'LineWidth', 1.5); % Aproximación en azul
24 legend('Función original x(t)', ['Aproximación (Forma Trigonométrica)
    ↳ con N = ', num2str(N), ' términos']);
25 xlabel('t');
26 ylabel('x(t)');
27 title('Serie de Fourier (Forma Trigonométrica) de x(t) = t');
28 grid on;
29 hold off;
```

Código 1: Código en Matlab para graficar la serie de Fourier trigonométrica de ???. Fuente: Elaboración propia

C.2. Código en Matlab para graficar la serie de Fourier compleja

```

1 % Número de términos positivos y negativos en la serie de Fourier
2 N = 10; % Puedes modificar N para mejorar la aproximación
3
4 % Vector de tiempo de -pi a pi
5 t = linspace(-pi, pi, 1000);
6
7 % Función original x(t) = t
8 x_original = t;
9
10 % Inicialización de la aproximación de la serie de Fourier
11 x_aprox = zeros(size(t));
12
13 % Cálculo de la serie de Fourier en forma exponencial compleja
14 for n = -N:N
15 if n == 0
16 continue; % Saltar n = 0 para evitar división por cero
17 end
18 cn = (1i / n) * (-1)^n;
19 x_aprox = x_aprox + cn * exp(1i * n * t);
20 end
21
22 % Tomar la parte real de la aproximación (la función original es real)
23 x_aprox_real = real(x_aprox);
24
25 % Gráfica de la función original y su aproximación
26 figure;
27 plot(t, x_original, 'k', 'LineWidth', 1.5); % Función original en negro
28 hold on;
29 plot(t, x_aprox_real, 'b', 'LineWidth', 1.5); % Aproximación en azul
30 legend('Función original x(t)', ['Aproximación (Forma Exponencial) con
   ↳ N = ', num2str(N), ' términos']);
31 xlabel('t');
32 ylabel('x(t)');
33 title('Serie de Fourier (Forma Exponencial Compleja) de x(t) = t');
34 grid on;
35 hold off;

```

Código 2: Código en Matlab para graficar la serie de Fourier trigonométrica de ???. Fuente: Elaboración propia

C.3. Código en Maple para la serie de Fourier trigonométrica

```

1 # Declaramos n como entero
2 assume(n, integer);

```

```

3
4 # Definimos la función
5 func := x;
6 # Periodo de la serie de Fourier
7 T := 2*Pi;
8
9 # Núcleos de las series de Fourier
10 series_cosine_core := cos(n*Pi*x/(T/2));
11 series_sine_core := sin(n*Pi*x/(T/2));
12
13 # Coeficientes de Fourier
14 a0 := (1/(T/2)) * int(func, x = -T/2..T/2);
15 an := (1/(T/2)) * int(func * series_cosine_core, x = -T/2..T/2);
16 bn := (1/(T/2)) * int(func * series_sine_core, x = -T/2..T/2);
17
18 # Simplificamos los coeficientes
19 a0_simp := simplify(a0);
20 an_simp := simplify(an);
21 bn_simp := simplify(bn);
22
23 # Factorizamos los coeficientes
24 Coeff_A0 := factor(a0_simp);
25 Coeff_An := factor(an_simp);
26 Coeff_Bn := factor(bn_simp);
27
28 # Definimos el rango de n positivo y negativo
29 n1 := 1;
30 n2 := 5;
31
32 # Creamos la lista de An
33 lista_An := [seq(simplify(subs(n = i, Coeff_An * series_cosine_core)),
34   i = n1..n2)];
35
36 # Creamos la lista de Bn
37 lista_Bn := [seq(simplify(subs(n = i, Coeff_Bn * series_sine_core)), i
38   = n1..n2)];
39
40 # Sumamos los coeficientes An y Bn
41 lista_completa := lista_An, lista_Bn;
42
43 # Crear la serie final añadiendo A0 al principio de la lista completa
44 serie_final := [Coeff_A0/2, lista_completa];
45 serie_final := simplify(lista_completa);
46
47 # Factorizamos la serie final
48 serie_factor := factor(serie_final);
49
50 # Suma de todos los coeficientes en una sola variable
51 serie_funcion := Coeff_A0/2 + add(lista_An[i], i = 1 .. nops(lista_An))
52   + add(lista_Bn[i], i = 1 .. nops(lista_Bn));

```

```

51 # Simplificamos la expresión para obtener la forma más compacta
52 serie_funcion_simplificada := simplify(serie_funcion);
53
54 # Graficamos la función y la serie aproximada
55 plot([func, serie_funcion_simplificada], x = -T/2..T/2);

```

Código 3: Código en Maple para calcular y graficar la serie de Fourier trigonométrica de ???. Fuente: Elaboración propia

C.4. Código en Maple para la serie de Fourier Compleja

```

1 # Declaramos n como entero
2 assume(n, integer);
3
4 # Definimos la función
5 func := x;
6
7 # Periodo de la serie de Fourier
8 T := 2*Pi;
9
10 # Núcleos de las series de Fourier
11 series_cosine_core := cos(n*Pi*x/(T/2));
12 series_sine_core := sin(n*Pi*x/(T/2));
13
14 # Coeficientes de Fourier
15 a0 := (1/(T/2)) * int(func, x = -T/2..T/2);
16 an := (1/(T/2)) * int(func * series_cosine_core, x = -T/2..T/2);
17 bn := (1/(T/2)) * int(func * series_sine_core, x = -T/2..T/2);
18
19 # Simplificamos los coeficientes
20 a0_simp := simplify(a0);
21 an_simp := simplify(an);
22 bn_simp := simplify(bn);
23
24 # Factorizamos los coeficientes
25 Coeff_A0 := factor(a0_simp);
26 Coeff_An := factor(an_simp);
27 Coeff_Bn := factor(bn_simp);
28
29 # Definimos el rango de n positivo y negativo
30 n1 := 1;
31 n2 := 5;
32
33 # Creamos la lista de An
34 lista_An := [seq(simplify(subs(n = i, Coeff_An * series_cosine_core)),
→ i = n1..n2)];
35
36 # Creamos la lista de Bn

```

```

37 lista_Bn := [seq(simplify(subs(n = i, Coeff_Bn * series_sine_core)), i
40   ↵ = n1..n2)];
38
39 # Sumamos los coeficientes An y Bn
40 lista_completa := lista_An, lista_Bn;
41
42 # Crear la serie final añadiendo A0 al principio de la lista completa
43 serie_final := [Coeff_A0/2, lista_completa];
44 serie_final := simplify(lista_completa);
45
46 # Factorizamos la serie final
47 serie_factor := factor(serie_final);
48
49 # Suma de todos los coeficientes en una sola variable
50 serie_funcion := Coeff_A0/2 + add(lista_An[i], i = 1 .. nops(lista_An))
51   ↵ + add(lista_Bn[i], i = 1 .. nops(lista_Bn));
52
53 # Simplificamos la expresión para obtener la forma más compacta
54 serie_funcion_simplificada := simplify(serie_funcion);
55
56 # Graficamos la función y la serie aproximada
57 plot([func, serie_funcion_simplificada], x = -T/2..T/2);

```

Código 4: Código en Maple para calcular y graficar la serie de Fourier compleja de func . Fuente: Elaboración propia

C.5. Código en Maxima para la serie de Fourier trigonométrica

```

1 declare(n, integer);
2 func: x;
3 /*func: ((3*x**3)-2*x+3);*/
4 T: 2*%pi;
5 series_cosine_core: cos((n*%pi*x)/((T/2)));
6 series_sine_core: sin((n*%pi*x)/((T/2)));
7
8 a0: (1/(T/2)) * integrate(func, x ,-(T/2), (T/2));
9 an: (1/(T/2)) * integrate(func * series_cosine_core, x ,-(T/2),
10   ↵ (T/2));
10 bn: (1/(T/2)) * integrate(func * series_sine_core, x ,-(T/2), (T/2));
11
12 a0_simp: ratsimp(a0);
13 an_simp: ratsimp(an);
14 bn_simp: ratsimp(bn);
15
16 Coeff_A0: factor(a0_simp);
17 Coeff_An: factor(an_simp);
18 Coeff_Bn: factor(bn_simp);
19

```

```

20  /* Definimos el rango de n positivo y negativo */
21  n1 : 1;
22  n2 : 10;
23
24  /* Creamos la lista de An */
25  lista_An : makelist(subst(n=i, Coeff_An * series_cosine_core), i, n1,
→   n2);
26
27  /* Creamos la lista de Bn */
28  lista_Bn : makelist(subst(n=i, Coeff_Bn * series_sine_core), i, n1,
→   n2);
29
30  /* Sumamos los coeficientes An + Bn */
31  lista_completa : lista_An + lista_Bn;
32
33  /* Crear la serie final añadiendo A0 al principio de la lista completa
→   */
34  /*serie_final: cons(Coeff_A0/2, lista_completa);*/
35
36  serie_final: ratsimp(lista_completa);
37
38  serie_factor: factor(serie_final);
39
40  /* Suma de todos los coeficientes en una sola variable */
41  serie_funcion : Coeff_A0/2 + sum(lista_An[i], i, 1, length(lista_An)) +
→   sum(lista_Bn[i], i, 1, length(lista_Bn));
42
43  /* Simplificamos la expresión para obtener la forma más compacta */
44  serie_funcion_simplificada : ratsimp(serie_funcion);
45
46  /* Graficamos la expresión */
47  plot2d([func, serie_funcion_simplificada], [x, -T/2, T/2]);
48
49  kill(all);

```

Código 5: Código en Maxima para calcular y graficar la serie de Fourier trigonométrica de ???. Fuente: Elaboración propia

C.6. Código en Maxima para la serie de Fourier compleja

```

1 declare(n, integer);
2 func: x;
3 /*func: ((3*x**3)-2*x+3);*/
4 T: 2*%pi;
5 series_cosine_core: cos((n*%pi*x)/((T/2)));
6 series_sine_core: sin((n*%pi*x)/((T/2)));
7
8 a0: (1/(T/2)) * integrate((func), x ,-(T/2), (T/2));

```

C.6. CÓDIGO EN MAXIMA PARA LA SERIE DE FOURIER COMPLEJA129

```

9  an: (1/(T/2)) * integrate((func) * series_cosine_core, x ,-(T/2),
10    ↵ (T/2));
11  bn: (1/(T/2)) * integrate((func) * series_sine_core, x ,-(T/2), (T/2));
12
13  a0_simp: ratsimp(a0);
14  an_simp: ratsimp(an);
15  bn_simp: ratsimp(bn);
16
17  Coeff_A0: factor(a0_simp);
18  Coeff_An: factor(an_simp);
19  Coeff_Bn: factor(bn_simp);
20
21  /* Definimos el rango de n positivo y negativo */
22  n1 : 1;
23  n2 : 10;
24
25  /* Creamos la lista de An */
26  lista_An : makelist(subst(n=i, Coeff_An * series_cosine_core), i, n1,
27    ↵ n2);
28
29  /* Creamos la lista de Bn */
30  lista_Bn : makelist(subst(n=i, Coeff_Bn * series_sine_core), i, n1,
31    ↵ n2);
32
33  /* Sumamos los coeficientes An + Bn */
34  lista_completa : lista_An + lista_Bn;
35
36  /* Crear la serie final añadiendo A0 al principio de la lista completa
37    */
38  /*serie_final: cons(Coeff_A0/2, lista_completa);*/
39
40  /* Suma de todos los coeficientes en una sola variable */
41  serie_funcion : Coeff_A0/2 + sum(lista_An[i], i, 1, length(lista_An)) +
42    ↵ sum(lista_Bn[i], i, 1, length(lista_Bn));
43
44  /* Simplificamos la expresión para obtener la forma más compacta */
45  serie_funcion_simplificada : ratsimp(serie_funcion);
46
47  /* Graficamos la expresión */
48  plot2d([func, serie_funcion_simplificada], [x, -T/2, T/2]);
49  kill(all);

```

Código 6: Código en Maxima para calcular y graficar la serie de Fourier compleja de ?? . Fuente: Elaboración propia

C.7. Código en Python usando matplotlib y sympy para la serie de Fourier trigonométrica

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4
5 # Definimos las variables simbólicas
6 x = sp.symbols('x')
7
8 # Pedimos al usuario que ingrese la función
9 user_function = input("Ingresa la función a aproximar (en términos de
→ x, por ejemplo: sin(x), cos(x), x**2, etc.): ")
10 function = sp.sympify(user_function)
11
12 # Definimos el intervalo y el número de términos de la serie de Fourier
13 L = np.pi # Longitud del intervalo
14 N = 10 # Número de términos en la serie de Fourier
15
16 # Calculamos los coeficientes a0, an, bn
17 a0 = (1 / (2 * L)) * sp.integrate(function, (x, -L, L))
18 an = lambda n: (1 / L) * sp.integrate(function * sp.cos(n * np.pi * x /
→ L), (x, -L, L))
19 bn = lambda n: (1 / L) * sp.integrate(function * sp.sin(n * np.pi * x /
→ L), (x, -L, L))
20
21 # Calculamos la serie de Fourier
22 t = np.linspace(-L, L, 1000)
23 fourier_series = a0.evalf()
24
25 for n in range(1, N + 1):
26     fourier_series += an(n).evalf() * np.cos(n * np.pi * t / L) +
→ bn(n).evalf() * np.sin(n * np.pi * t / L)
27
28 # Convertimos la función original a una función NumPy para graficar
29 f_original = sp.lambdify(x, function, modules='numpy')
30
31 # Graficamos la función original y su serie de Fourier
32 plt.figure(figsize=(10, 6))
33 plt.plot(t, f_original(t), label='Función Original', color='red',
→ linewidth=2)
34 plt.plot(t, fourier_series, label='Serie de Fourier (N=10)',
→ color='blue', linestyle='--')
35 plt.title('Aproximación de una Función con Serie de Fourier')
36 plt.xlabel('x')
37 plt.ylabel('Amplitud')
38 plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
39 plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
40 plt.grid()

```

C.8. CÓDIGO EN PYTHON USANDO MATPLOTLIB Y SYMPY PARA LA SERIE DE FOURIER

```
41 plt.legend()
42
43 # Guarda la gráfica como archivo
44 plt.savefig('serie_fourier.png')
45 print("La gráfica se ha guardado como 'serie_fourier.png'.")
```

Código 7: Código en Python con matplotlib y sympy para graficar la serie de Fourier trigonométrica de ???. Fuente: *Elaboración propia*

C.8. Código en Python usando matplotlib y sympy para la serie de Fourier compleja

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4
5 # Definimos las variables simbólicas
6 x = sp.symbols('x')
7
8 # Pedimos al usuario que ingrese la función
9 user_function = input("Ingresa la función a aproximar (en términos de
→ x, por ejemplo: sin(x), cos(x), x**2, etc.): ")
10 function = sp.sympify(user_function)
11
12 # Definimos el intervalo y el número de términos de la serie de Fourier
13 L = np.pi # Longitud del intervalo
14 N = 10 # Número de términos en la serie de Fourier
15
16 # Calculamos los coeficientes cn para la serie de Fourier en su forma
→ exponencial compleja
17 cn = lambda n: (1 / (2 * L)) * sp.integrate(function * sp.exp(-1j * n *
→ sp.pi * x / L), (x, -L, L))
18
19 # Calculamos la serie de Fourier en su forma exponencial compleja
20 t = np.linspace(-L, L, 1000)
21 fourier_series_complex = 0
22
23 for n in range(-N, N + 1):
24     fourier_series_complex += cn(n).evalf() * np.exp(1j * n * np.pi * t /
→ L)
25
26 # Convertimos la función original a una función NumPy para graficar
27 f_original = sp.lambdify(x, function, modules='numpy')
28
29 # Graficamos la función original y su serie de Fourier
30 plt.figure(figsize=(10, 6))
31 plt.plot(t, f_original(t), label='Función Original', color='red',
→ linewidth=2)
```

```

32 plt.plot(t, fourier_series_complex.real, label='Serie de Fourier
33     ↳ Compleja (N=10)', color='blue', linestyle='--')
34 plt.title('Aproximación de una Función con Serie de Fourier Compleja')
35 plt.xlabel('x')
36 plt.ylabel('Amplitud')
37 plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
38 plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
39 plt.grid()
40 plt.legend()
41 # Guarda la gráfica como archivo
42 plt.savefig('serie_fourier_compleja.png')
43 print("La gráfica se ha guardado como 'serie_fourier_compleja.png'.")

```

Código 8: Código en Python con matplotlib y sympy para calcular y graficar la serie de Fourier compleja de ???. Fuente: *Elaboración propia*

C.9. Código en Python usando manim para la serie de Fourier trigonométrica

```

1 # Importamos todas la funciones de manim
2 from manim import *
3
4 # Creamos una escena
5 class Prueba(Scene):
6     def construct(self):
7
8     #Definimos un color de fondo para todo el lienzo
9     self.camera.background_color = "#212121"
10
11    x_labels = [
12        "-3\pi",                      # -3pi
13        "-\frac{5\pi}{2}",              # -5pi/2
14        "-2\pi",                      # -2pi
15        "-\frac{3\pi}{2}",              # -3pi/2
16        "-\pi",                        # -pi
17        "-\frac{\pi}{2}",               # -pi/2
18        "0",                           # Blank
19        "\frac{\pi}{2}",                # pi/2
20        "\pi",                          # pi
21        "\frac{3\pi}{2}",               # 3pi/2
22        "2\pi",                        # 2pi
23        "\frac{5\pi}{2}",               # 5pi/2
24        "3\pi"                         # 3pi
25    ]
26
27    y_labels = [
28        "-\frac{3\pi}{2}",             # -3pi/2
29        "-\pi"                         # -pi

```

C.9. CÓDIGO EN PYTHON USANDO MANIM PARA LA SERIE DE FOURIER TRIGONOMÉTRICA

```

30  "-\\frac{\\pi}{2}",      # -pi/2
31  "0",                  # Blank
32  "\\frac{\\pi}{2}",     # pi/2
33  "\\pi",                # pi
34  "\\frac{3\\pi}{2}"     # 3pi/2
35  ]
36
37  # Creamos los ejes
38  ejes=Axes(
39  x_range = (-3*PI, 3*PI, PI), # Rango del eje x (inicio, fin, de cuanto
    ↪ en cuanto avanza)
40  x_length=(6),           # Tamaño del eje x
41  y_range = (-3*PI/2, 3*PI/2, PI), # Rango del eje y (inicio, fin, de
    ↪ cuanto en cuanto avanza)
42  y_length=(3),           # Tamaño del eje y
43
44  # Configuramos los ejes.
45  axis_config={
46      "include_numbers": False,          # Los ejes se numerarán
47      "font_size": 13,                  # Tamaño de los números
48      "tip_width": 0.05,               # Ancho de la punta
49      "tip_height": 0.03,              # Alto de la punta
50      "tick_size":0.06,               # Tamaño de las lineas de los
    ↪ ejes
51      "color": WHITE,                 # Color de los ejes
52      "stroke_width": 1,              # Grosor de los ejes
53      "line_to_number_buff": SMALL_BUFF # Espacio entre las lineas y
    ↪ los números
54  }
55 )
56
57 x_tex_labels = VGroup(*[
58 MathTex(t, font_size=15).next_to(ejes.x_axis.n2p(x),DOWN*0.4) if x >= 0
    ↪ else
59 # Shift pi<0 labels to left
60 MathTex(t,
    ↪ font_size=15).next_to(ejes.x_axis.n2p(x),DOWN*0.4).shift(LEFT*0.05)
61 for t,x in zip(x_labels,np.arange(-3*PI, 3*PI+PI/2, PI/2)) if t != "0"
62 # Ignore 0 value
63 ])
64
65 y_tex_labels = VGroup(*[
66 MathTex(t, font_size=15).next_to(ejes.y_axis.n2p(x),LEFT*0.4) if x >= 0
    ↪ else
67 # Shift pi<0 labels to left
68 MathTex(t,
    ↪ font_size=15).next_to(ejes.y_axis.n2p(x),LEFT*0.4).shift(LEFT*0.05)
69 for t,x in zip(y_labels,np.arange(-3*PI/2, 3*PI/2+PI/2, PI/2)) if t !=
    ↪ "0"
70 # Ignore 0 value
71 ])

```

```

72
73 # Creamos el plano de números
74 plano_numeros = NumberPlane(
75     x_range = (-3*PI, 3*PI, PI), # Rango del eje x (inicio, fin, de cuanto
    ↪ en cuanto avanza)
76     x_length=(6),           # Tamaño del eje x
77     y_range = (-3*PI/2, 3*PI/2, PI), # Rango del eje y (inicio, fin, de
    ↪ cuanto en cuanto avanza)
78     y_length=(3),           # Tamaño del eje y
79     faded_line_ratio=2,      # Dentro de cada recuadro de unidad, habrá dos
    ↪ de un color mas opaco
80
81 # Configuración de los ejes
82 axis_config={
83     "include_numbers": False,   # Los ejes no se numerarán
84     "tip_width": 0.05,
85     "tip_height": 0.03,
86     "tick_size": 0.06,
87     "color": WHITE
88 },
89
90 # Configuración del plano
91 background_line_style={
92     "stroke_color": TEAL,     # Color de las líneas del plano
93     "stroke_width": 1,        # Grosor de las líneas
94     "stroke_opacity": 0.6     # Opacidad de las líneas
95 }
96 ).set_z_index(-1)
97
98 #ejes.height = 6
99 #ejes.width = 9
100
101 # Título en la parte superior
102 titulo = Text("Serie de Fourier Trigonométrica de:", font_size=32)
103 fun = MathTex(r"f(t)= t, -\pi <t< \pi", font_size=25)
104
105 # Agrupamos ambos textos en un solo grupo y lo posicionamos en la parte
    ↪ superior
106 titulo.next_to(fun, UP)
107 encabezado = VGroup(titulo, fun)
108 encabezado.to_edge(UP).set_z_index(1)
109
110 # Etiquetas de ejes
111 eje_x = ejes.get_x_axis_label("t").next_to(ejes.x_axis.get_end(),
    ↪ UP*0.1).scale(0.45).set_color(WHITE)
112 eje_y = ejes.get_y_axis_label("f(t)").next_to(ejes.y_axis.get_end(),
    ↪ RIGHT*0.1).scale(0.45).set_color(WHITE)
113
114 ejes_etiquetas = VGroup(eje_x, eje_y)
115
116 # Función f(t) en la parte inferior

```

C.9. CÓDIGO EN PYTHON USANDO MANIM PARA LA SERIE DE FOURIER TRIGONOMÉTRICA

```

117  funcion_label = MathTex(
118    r" f(t) = -2 \sum_{n=1}^{\infty} \left[ \frac{(-1)^n}{n} \sin(nt) \right] ",
119    font_size=24
120  )
121  #funcion_label.set_color_by_tex("n", YELLOW)
122  funcion_label.next_to(ejes, DOWN*1.5)
123
124  # Usando dos rectangulos creamos una ventana para dibujar dentro todo
125  # nuestro plano
126  rec1 = Rectangle(height=80, width=40)
127  rec2 = Rectangle(height=3.5, width=6.6)
128  ventana = Cutout(rec1, rec2, fill_opacity=1, color="#212121",
129  # añaden a la suma
130  i_value_text = Text("n = 0", color=WHITE, font_size=16)
131  i_value_text.next_to(funcion_label, DOWN)
132
133  # Creamos la función a graficar en el intervalo establecido
134  func_Original=ejes.plot(
135    lambda x: x, x_range=(-PI,PI), color="#d7da63",
136  )
137
138  # Definimos la función serie compleja
139  def funcion_Serie_Trig(self, x, i):
140    # Inicialmente valdrá 0
141    val=0
142
143    # El coeficiente a0 es 0
144    a0 = 0
145
146    # En cada iteración sumará un elemento k positivo y uno negativo
147    for k in range(1,i+1):
148      try:
149        val += (((-1)**(k)) / k) * np.sin(k * x)
150
151    # Si algún término es indeterminado se sumará un 0
152    except ZeroDivisionError:
153      val += 0
154
155    # EL valor de la suma será multiplicado por su coeficiente
156    return a0 - (2) * val
157
158    self.play(Write(encabezado))
159    self.play(Create(ventana))
160    self.play(Create(ejes), Create(plano_numeros), Create(x_tex_labels),
161    Create(y_tex_labels))
162    self.play(Write(ejes_etiquetas))
163    self.play(Create(func_Original))

```

```

163 self.wait(3)
164
165
166 # Se crean las aproximaciones en la Serie
167 for j in range (0,6):
168     i_value_text.set_text(f"n = {j}")
169     fsC_0=ejes.plot(
170         lambda x: funcion_Serie_Trig(self, x, j), x_range=(-3*PI,3*PI),
171         color=BLUE, stroke_width = 2.5
172     )
173     if j==0:
174         self.play(Create(fsC_0), Write(funcion_label), Write(i_value_text))
175     else:
176         self.play(Transform(old_graph, fsC_0), Transform(i_value_text, Text(f"n
177             = {j}", color=WHITE, font_size=16).next_to(funcion_label, DOWN)))
178     plano = VGroup(encabezado, ejes, ejes_etiquetas, ventana,
179         func_Original, old_graph)
180     plano.move_to(ORIGIN)
181     self.wait(2)

```

Código 9: Código en Python con Manim para graficar la serie de Fourier trigonométrica de ???. Fuente: Elaboración propia

C.10. Código en Python usando Manim para la serie de Fourier compleja

```

1 # Importamos todas la funciones de manim
2 from manim import *
3
4 # Creamos una escena
5 class Prueba(Scene):
6     def construct(self):
7
8         #Definimos un color de fondo para todo el lienzo
9         self.camera.background_color = "#212121"
10
11    x_labels = [
12        "-3\\pi",           # -3pi
13        "-\\frac{5\\pi}{2}", # -5pi/2
14        "-2\\pi",           # -2pi
15        "-\\frac{3\\pi}{2}", # -3pi/2
16        "-\\pi",            # -pi
17        "-\\frac{\\pi}{2}",   # -pi/2
18        "0",               # Blank
19        "\\frac{\\pi}{2}",   # pi/2
20        "\\pi",             # pi
21        "\\frac{3\\pi}{2}"  # 3pi/2

```

C.10. CÓDIGO EN PYTHON USANDO MANIM PARA LA SERIE DE FOURIER COMPLEJA

```

22 "2\\pi",                      # 2pi
23 "\\frac{5\\pi}{2}",            # 5pi/2
24 "3\\pi"                      # 3pi
25 ]
26
27 y_labels = [
28 "-\\frac{3\\pi}{2}",           # -3pi/2
29 "-\\pi",                      # -pi
30 "-\\frac{\\pi}{2}",            # -pi/2
31 "0",                          # Blank
32 "\\frac{\\pi}{2}",             # pi/2
33 "\\pi",                       # pi
34 "\\frac{3\\pi}{2}"            # 3pi/2
35 ]
36 ]
37
38
39 # Creamos los ejes
40 ejes=Axes(
41 x_range = (-3*PI, 3*PI, PI), # Rango del eje x (inicio, fin, de cuanto
    ↪ en cuanto avanza)
42 x_length=(6),               # Tamaño del eje x
43 y_range = (-3*PI/2, 3*PI/2, PI), # Rango del eje y (inicio, fin, de
    ↪ cuanto en cuanto avanza)
44 y_length=(3),               # Tamaño del eje y
45
46 # Configuramos los ejes.
47 axis_config={
48     "include_numbers": False,      # Los ejes se numerarán
49     "font_size": 13,                # Tamaño de los números
50     "tip_width": 0.05,              # Ancho de la punta
51     "tip_height": 0.03,             # Alto de la punta
52     "tick_size": 0.06,              # Tamaño de las lineas de los
        ↪ ejes
53     "color": WHITE,                # Color de los ejes
54     "stroke_width": 1,              # Grosor de los ejes
55     "line_to_number_buff": SMALL_BUFF # Espacio entre las lineas y
        ↪ los números
56 }
57 )
58
59
60 x_tex_labels = VGroup(*[
61 MathTex(t, font_size=15).next_to(ejes.x_axis.n2p(x),DOWN*0.4) if x >= 0
    ↪ else
62 # Shift pi<0 labels to left
63 MathTex(t,
    ↪ font_size=15).next_to(ejes.x_axis.n2p(x),DOWN*0.4).shift(LEFT*0.05)
64 for t,x in zip(x_labels,np.arange(-3*PI, 3*PI+PI/2, PI/2)) if t != "0"
65 # Ignore 0 value
66 ])

```

```

67
68 y_tex_labels = VGroup(*[
69 MathTex(t, font_size=15).next_to(ejes.y_axis.n2p(x), LEFT*0.4) if x >= 0
70 → else
71 # Shift pi<0 labels to left
72 MathTex(t,
73 → font_size=15).next_to(ejes.y_axis.n2p(x), LEFT*0.4).shift(LEFT*0.05)
74 for t,x in zip(y_labels,np.arange(-3*PI/2, 3*PI/2+PI/2, PI/2)) if t != "0"
75 → "# Ignore 0 value
76 ])
77
78 # Creamos el plano de números
79 plano_numeros = NumberPlane(
80 x_range = (-3*PI, 3*PI, PI), # Rango del eje x (inicio, fin, de cuanto
81 → en cuanto avanza)
82 x_length=(6), # Tamaño del eje x
83 y_range = (-3*PI/2, 3*PI/2, PI), # Rango del eje y (inicio, fin, de
84 → cuanto en cuanto avanza)
85 y_length=(3), # Tamaño del eje y
86 faded_line_ratio=2, # Dentro de cada recuadro de unidad, habrá dos
87 → de un color mas opaco
88
89 # Configuración de los ejes
90 axis_config={
91     "include_numbers": False, # Los ejes no se numerarán
92     "tip_width": 0.05,
93     "tip_height": 0.03,
94     "tick_size": 0.06,
95     "color": WHITE
96 },
97
98 # Configuración del plano
99 background_line_style={
100     "stroke_color": TEAL, # Color de las líneas del plano
101     "stroke_width": 1, # Grosor de las líneas
102     "stroke_opacity": 0.6 # Opacidad de las líneas
103 }
104 ).set_z_index(-1)
105
106 # Título en la parte superior
107 titulo = Text("Serie de Fourier compleja de:", font_size=32)
108 fun = MathTex(r"f(t)= t, -\pi < t < \pi", font_size=32)
109
110 # Agrupamos ambos textos en un solo grupo y lo posicionamos en la parte
111 → superior
112 titulo.next_to(fun, UP)
113 encabezado = VGroup(titulo, fun)
114 encabezado.to_edge(UP).set_z_index(1)
115
116 # Etiquetas de ejes

```

C.10. CÓDIGO EN PYTHON USANDO MANIM PARA LA SERIE DE FOURIER COMPLEJA

```

111 eje_x = ejes.get_x_axis_label("t").next_to(ejes.x_axis.get_end(),
112     UP*0.1).scale(0.45).set_color(WHITE)
113 eje_y = ejes.get_y_axis_label("f(t)").next_to(ejes.y_axis.get_end(),
114     RIGHT*0.1).scale(0.45).set_color(WHITE)
115
116 # Función f(t) en la parte inferior
117 funcion_label = MathTex(
118     r"f(t)= \sum_{n=-\infty}^{\infty}\left( \frac{(-1)^n}{n} e^{int} \right)",
119     font_size=24
120 )
121 #funcion_label.set_color_by_tex("n", YELLOW)
122 funcion_label.next_to(ejes, DOWN*2.4)
123
124 # Usando dos rectangulos creamos una ventana para dibujar dentro todo
125     # nuestro plano
126 rec1 = Rectangle(height=80, width=40)
127 rec2 = Rectangle(height=3.8, width=6.6)
128 ventana = Cutout(rec1, rec2, fill_opacity=1, color="#212121",
129     stroke_color=RED)
130
131 # Creamos un contador n para mostrar la cantidad de terminos que se
132     # añaden a la suma
133 i_value_text = Text("n = 0", color=WHITE, font_size=16)
134 i_value_text.next_to(funcion_label, DOWN)
135
136 # Definimos la función serie compleja
137 def funcion_Serie_Compleja(self, x, i):
138     # Inicialmente valdrá 0
139     val=0
140
141     # Como el coeficiente c0 no es indeterminado, lo calculamos aparte
142     c0 = 0
143
144     # En cada iteración sumará un elemento k positivo y uno negativo
145     for k in range(1,i+1):
146         try:
147             val += (((-1)**(k)) / k) * (np.exp(complex(0, (k * x))))
148             val += (((-1)**(-k)) / (-k)) * (np.exp(complex(0, ((-k) * x))))
149
150         # Si algún término es indeterminado se sumará un 0
151     except ZeroDivisionError:
152         val += 0
153
154     # El valor de la suma será multiplicado por su coeficiente
155     return complex(0,1) * val + c0

```

```

156 # Creamos la función a graficar en el intervalo establecido
157 func_Original=ejes.plot(
158 lambda x: x, x_range=(-PI,PI), color="#d7da63",
159 )
160
161 self.play(Write(encabezado))
162 self.play(Create(ventana))
163 self.play(Create(ejes), Create(plano_numeros), Create(x_tex_labels),
164    Create(y_tex_labels))
164 self.play(Write(ejes_etiquetas))
165 self.play(Create(func_Original))
166
167
168 # Se crean las aproximaciones en la Serie
169 for j in range (0,6):
170 i_value_text.set_text(f"n = {j}")
171 fsC_0=ejes.plot(
172 lambda x: funcion_Serie_Compleja(self, x, j),
173    x_range=(-3*np.pi,3*np.pi), color=BLUE, stroke_width = 2.5
174 )
174 if j==0:
175 self.play(Create(fsC_0), Write(funcion_label), Write(i_value_text))
176 old_graph=fsC_0
177 else:
178 self.play(Transform(old_graph, fsC_0), Transform(i_value_text, Text(f"n
179    = {j}", color=WHITE, font_size=16).next_to(funcion_label, DOWN)))
180 plano = VGroup(encabezado, ejes, ejes_etiquetas, ventana,
181    func_Original, old_graph)
181 plano.move_to(ORIGIN)
182 self.wait(2)

```

Código 10: Código en Python con Manim para graficar la serie de Fourier compleja de ???. Fuente: Elaboración propia

C.11. Script para dibujar el plano y sus funcionalidades básicas en el canvas

```

1 // Configuración del canvas y variables iniciales
2 const canvas = document.getElementById("canvas");
3 const ctx = canvas.getContext("2d");
4 let width = canvas.width;
5 let height = canvas.height;
6 let unit = 75; // Escala de unidad en el plano
7 let offsetX = 0, offsetY = 0;
8
9 let origin = { x: width / 2, y: height / 2 };
10
11 function drawScreen() {

```

C.11. SCRIPT PARA DIBUJAR EL PLANO Y SUS FUNCIONALIDADES BÁSICAS EN EL

```
12     ctx.clearRect(0, 0, width, height);
13
14     // Ejes X e Y
15     const XAxis = { start: { x: 0, y: height / 2 }, end: { x:
16         ↵ width, y: height / 2 } };
17     const YAxis = { start: { x: width / 2, y: 0 }, end: { x: width
18         ↵ / 2, y: height } };
19
20     origin = { x: width / 2, y: height / 2 };
21
22     // Dibujar ejes
23     drawAxes(XAxis, YAxis, "#90DCB5");
24     drawGrid(origin, XAxis, YAxis, unit, "#6BBCAC", "#E8E8E8");
25 }
26
27 function drawAxes(XAxis, YAxis, axisColor) {
28     ctx.beginPath();
29     ctx.moveTo(XAxis.start.x, XAxis.start.y - offsetY);
30     ctx.lineTo(XAxis.end.x, XAxis.end.y - offsetY);
31     ctx.moveTo(YAxis.start.x - offsetX, YAxis.start.y);
32     ctx.lineTo(YAxis.end.x - offsetX, YAxis.end.y);
33     ctx.strokeStyle = axisColor;
34     ctx.lineWidth = 1;
35     ctx.stroke();
36 }
37
38 function drawGrid(origin, XAxis, YAxis, unit, gridColor, fontColor) {
39     ctx.strokeStyle = gridColor;
40     ctx.fillStyle = fontColor;
41
42     for (let i = -1000; i < 1000; i++) {
43         const x = origin.x + unit * i - offsetX;
44         const y = origin.y + unit * i - offsetY;
45
46         // Líneas verticales
47         ctx.beginPath();
48         ctx.moveTo(x, YAxis.start.y);
49         ctx.lineTo(x, YAxis.end.y);
50         ctx.lineWidth = (i % 5 === 0) ? 1 : 0.25;
51         ctx.stroke();
52
53         // Líneas horizontales
54         ctx.beginPath();
55         ctx.moveTo(XAxis.start.x, y);
56         ctx.lineTo(XAxis.end.x, y);
57         ctx.lineWidth = (i % 5 === 0) ? 1 : 0.25;
58         ctx.stroke();
59
60         // Etiquetas en los ejes
61         if (i % 5 === 0) {
```

```

60         ctx.fillText(i, x, origin.y - offsetY + 15); //  

61         ↳ Eje X  

62         ctx.fillText(-i, origin.x - offsetX, y); // Eje  

63         ↳ Y  

64     }
}

```

Código 11: Parte del código en Javascript responsable de generar la gráfica y sus funcionalidades

. *Fuente: Elaboración propia*

C.12. Script para graficar Función para graficar funciones en el canva

```

1 function drawFunction(mathFunction, color) {  

2     let previousX = undefined;  

3     let previousY = undefined;  

4  

5     for (let px = 0; px < width; px++) {  

6         const x = (((px + offsetX) / unit) - ((width / unit) /  

7             2));  

8         const y = mathFunction(x);  

9  

10        if (previousX !== undefined && previousY !== undefined)  

11            {  

12                ctx.strokeStyle = color;  

13                ctx.beginPath();  

14                ctx.moveTo(previousX, previousY);  

15                ctx.lineTo(origin.x - offsetX + unit * x,  

16                    ↳ origin.y - offsetY - unit * y);  

17                ctx.lineWidth = 2;  

18                ctx.stroke();  

19            }  

20        previousX = origin.x - offsetX + unit * x;  

21        previousY = origin.y - offsetY - unit * y;  

}
}

```

Código 12: Parte del código en Javascript responsable de graficar funciones matemáticas

. *Fuente: Elaboración propia*

C.13. Script para graficar Función para graficar funciones en el canva

```

1  function drawFunction(mathFunction, color) {
2    let previousX = undefined;
3    let previousY = undefined;
4
5    for (let px = 0; px < width; px++) {
6      const x = (((px + offsetX) / unit) - ((width / unit) / 2));
7      const y = mathFunction(x);
8
9      if (previousX !== undefined && previousY !== undefined) {
10        ctx.strokeStyle = color;
11        ctx.beginPath();
12        ctx.moveTo(previousX, previousY);
13        ctx.lineTo(origin.x - offsetX + unit * x, origin.y -
14          offsetY - unit * y);
15        ctx.lineWidth = 2;
16        ctx.stroke();
17      }
18
19      previousX = origin.x - offsetX + unit * x;
20      previousY = origin.y - offsetY - unit * y;
21    }
}

```

Código 13: Parte del código en Javascript responsable de graficar series de funciones

. Fuente: Elaboración propia

C.14. Código en Maxima para calcular la serie trigonométrica

```

1  /* Función a trozos */
2  func : matrix([1, 0, 1], [0,1, 3], [-1, 3, 5]);
3
4  /* Obtenemos la cantidad de trozos */
5  pieces: length(func);
6
7  /* Obtenemos a -T/2 y T/2 */
8  inicio: func[1][2];
9  fin: func[length(func)][3];
10
11 declare(n, integer);
12
13 /* Calcular el periodo */
14 T: fin - inicio;
15

```

```

16 /* Núcleo de los coeficientes */
17 series_cosine_core: (cos((n%pi*x)/((T)))); 
18 series_sine_core: sin((n%pi*x)/((T)));
19
20 /* Núcleo de los coeficientes para extensión periódica */
21 /*series_cosine_core: (cos((n%pi*x)/((T/2)))); 
22 series_sine_core: sin((n%pi*x)/((T/2)));*/
23
24 a0_acum: 0;
25 an_acum: 0;
26 bn_acum: 0;
27
28 for i:1 thru pieces do
29 (
30 trozo: func[i],
31 piece_func: trozo[1],
32 start: trozo[2],
33 end: trozo[3],
34
35 a0: (2/(T)) * integrate((piece_func), x ,(start), (end)),
36 an: (2/(T)) * integrate((piece_func) * series_cosine_core, x ,(start),
37 → (end)),
37 bn: (2/(T)) * integrate((piece_func) * series_sine_core, x ,(start),
38 → (end)),
38
39 a0_acum: a0 + a0_acum,
40 an_acum: an + an_acum,
41 bn_acum: bn + bn_acum
42 )$ 
43
44 a0_simp: ratsimp(a0_acum);
45 an_simp: ratsimp(an_acum);
46 bn_simp: ratsimp(bn_acum);
47
48 Coeff_A0: factor(a0_simp/2);
49 Coeff_An: factor(an_simp);
50 Coeff_Bn: factor(bn_simp);
51
52 tex(Coeff_A0);
53 tex(Coeff_An);
54 kill(all);

```

Código 14: Código en Maxima para calcular los coeficientes y su expansión de la serie Trigonométrica de Fourier

Fuente: Elaboración propia

C.15. Código en Maxima para calcular la serie compleja

```

1  /* Función a trozos */
2  func : matrix([1, 0, 1], [0,1, 3], [-1, 3, 5]);
3
4  /* Obtenemos la cantidad de trozos */
5  pieces: length(func);
6
7  /* Obtenemos a -T/2 y T/2 */
8  inicio: func[1][2];
9  fin: func[length(func)][3];
10
11 declare(n, integer);
12
13 tellsimpafter(exp(%i * %pi * n), (-1)**n)$
14 tellsimpafter(exp(%i * 2 * %pi * n), 1)$
15
16 /* Calcular el periodo */
17 T: fin - inicio;
18
19 series_core: (exp((%i*n*%pi*x)/((T))));
20 complex_core: (exp(-(%i*n*%pi*x)/((T))));
21
22 c0_acum: 0;
23 cn_acum: 0;
24
25 for i:1 thru pieces do
26 (
27  trozo: func[i],
28  piece_func: trozo[1],
29  start: trozo[2],
30  end: trozo[3],
31
32  c0: (1/(T)) * integrate((piece_func), x, (start), (end)),
33  cn: (1/(T)) * integrate((piece_func) * complex_core, x ,(start),
34   ↳ (end)),
35  c0_acum: c0 + c0_acum,
36  cn_acum: cn + cn_acum
37 )$ 
38
39 c0_simp: ratsimp(c0_acum);
40 cn_simp: ratsimp(cn_acum);
41
42 Coeff_C0: factor(c0_simp/2);
43 Coeff_Cn: factor(cn_simp);
44
45 /* Definimos el rango de n positivo y negativo */
46 n1 : 1;
```

```

47 n2 :15;
48
49 /* Creamos la lista de términos positivos */
50 lista_positivos : makelist(subst(n=i, Coeff_Cn * series_core ), i, n1,
51   ↵ n2);
52
53 /* Creamos la lista de términos negativos */
54 lista_negativos : makelist(subst(n=i, Coeff_Cn * series_core), i, -n2,
55   ↵ -n1);
56
57 /* Invertimos el orden de la lista de términos negativos */
58 lista_negativos_invertida : reverse(lista_negativos);
59
60 /* Sumamos los términos positivos y negativos (con la lista de
61   ↵ negativos invertida) */
62 lista_completa : lista_positivos + lista_negativos_invertida;
63
64 serie_demoivre: demoivre(lista_completa);
65
66 serie_simp: ratsimp(serie_demoivre);
67
68 serie_factor: factor(serie_simp);
69
70 kill(all);

```

Código 15: Código en Maxima para calcular los coeficientes y su expansión de la serie compleja de Fourier

Fuente: Elaboración propia

C.16. API en NodeJS para ejecutar a Maxima

```

1 const express = require('express');
2 const cors = require('cors');
3 const bodyParser = require('body-parser');
4 const { exec } = require('child_process');
5
6 const app = express();
7 const port = 3000;
8
9 // Configurar CORS
10 var corsOptions = {
11     origin: 'http://localhost:4200',
12     optionsSuccessStatus: 200
13 };
14
15 app.use(cors(corsOptions));
16 app.use(bodyParser.json()); // Analizar las solicitudes JSON
17
18 // Middleware para establecer cabeceras CORS
19 function setCorsHeaders(req, res, next) {

```

```

20         res.setHeader('Access-Control-Allow-Origin', '*');
21         res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT,
22                         PATCH, DELETE');
22         res.setHeader('Access-Control-Allow-Headers', 'Content-Type,
23                         Authorization');
23         next();
24     }
25
26 app.use(setCorsHeaders);
27
28 // Función para ejecutar comandos en Maxima y obtener los resultados
29 function execMaxima(command, callback) {
30     exec(command, (error, stdout, stderr) => {
31         if (error) {
32             console.error(`Error executing Maxima command:
33                         ${stderr}`);
33             callback(error, null);
34         } else {
35             const result = stdout.trim();
36             callback(null, result);
37         }
38     });
39 }
40
41 // Ruta para calcular Serie Trigonométrica para una función de un solo
41 // trozo
42 app.post('/fourier/trigonometric', (req, res) => {
43     console.log("Se ha recibido una solicitud para calcular la
43                         serie trigonométrica");
44     const { funcion, periodo } = req.body;
45
46     const command_a0 = `echo "declare(n, integer)$
46                         string(ratsimp(((1/(${periodo})/2)) *
46                         integrate(${funcion}, x, -(${periodo}/2),
46                         ${periodo}/2)))"; | maxima --very-quiet -`;
47     const command_an = `echo "declare(n, integer)$
47                         string(ratsimp(((1/(${periodo})/2)) *
47                         integrate(((${funcion}) * cos((n*pi*x)/(${periodo}/2))),
47                         x, -(${periodo}/2), ${periodo}/2))); | maxima
47                         --very-quiet -`;
48     const command_bn = `echo "declare(n, integer)$
48                         string(ratsimp(((1/(${periodo})/2)) *
48                         integrate(((${funcion}) * sin((n*pi*x)/(${periodo}/2))),
48                         x, -(${periodo}/2), ${periodo}/2))); | maxima
48                         --very-quiet -`;
49
50     execMaxima(command_a0, (error_a0, a0) => {
51         if (error_a0) {
52             res.status(500).send({ error: `Error
52                           calculating a0: ${error_a0.message}` });
53             return;

```

```

54     }
55     execMaxima(command_an, (error_an, an) => {
56       if (error_an) {
57         res.status(500).send({ error: `Error
58           ↪ calculating an:
59           ↪ ${error_an.message}` });
60       }
61       execMaxima(command_bn, (error_bn, bn) => {
62         if (error_bn) {
63           res.status(500).send({ error:
64             ↪ `Error calculating bn:
65             ↪ ${error_bn.message}` });
66           return;
67         }
68         res.json({ a0, an, bn });
69       });
70     });
71   );
72
73 // Ruta para calcular Serie Exponencial para una función de un solo
74 // trozo
74 app.post('/fourier/complex', (req, res) => {
75   const { funcion, periodo } = req.body;
76
77   const command_c0 = `echo "declare(n, integer)$
78     tellsimpafter(exp(%i*%pi*n), (-1)**n)$
79     tellsimpafter(exp(%i*2*%pi*n),1)$
80     string(ratsimp((1/(${funcion}))) * integrate((${funcion}), x
81     ,-((${periodo})/2), ((${periodo})/2)));;" | maxima
82     --very-quiet -`;
83   const command_cn = `echo "declare(n, integer)$
84     tellsimpafter(exp(%i*%pi*n), (-1)**n)$
85     tellsimpafter(exp(%i*2*%pi*n),1)$
86     string(ratsimp((1/(${funcion}))) * integrate((${funcion}) *
87     (exp(-(%i*n*%pi*x)/(((${periodo})/2))), x
88     ,-((${periodo})/2), ((${periodo})/2)));;" | maxima
89     --very-quiet -`;
90
91   execMaxima(command_c0, (error_c0, c0) => {
92     if (error_c0) {
93       res.status(500).send({ error: `Error
94         ↪ calculating c0: ${error_c0.message}` });
95       return;
96     }
97     execMaxima(command_cn, (error_cn, cn) => {
98       if (error_cn) {
99

```

```

87                     res.status(500).send({ error: `Error
88                         ↪ calculating cn:
89                         ↪ ${error_cn.message}` });
90                     return;
91                 }
92             );
93             console.log(`Se ha calculado la serie compleja`);
94         );
95
96 // Iniciar el servidor
97 app.listen(port, () => {
98     console.log(`Server running at http://localhost:${port}`);
99 });

```

Código 16: API en NodeJS para ejecutar comandos de Maxima
Fuente: Elaboración propia

C.17. Script en NodeJS para ejecutar a tex2max

```

1 // Importa la librería TeX2Max.
2 const TeX2Max = require('tex2max');
3
4 // Crea una instancia de TeX2Max con las siguientes opciones de
4   ↪ configuración.
5 const converter = new TeX2Max({
6     onlySingleVariables: false,           // Permite nombres de
6       ↪ variables complejas.
7     handleEquation: true,                // No permite que Maxima
7       ↪ resuelva ecuaciones algebraicas.
8     addTimesSign: true,                  // Añade signos de
8       ↪ multiplicación donde se implique multiplicación.
9     disallowDecimalPoints: false,        // Permite el uso de puntos
9       ↪ decimales en los números.
10    disallowlDecimalCommas: false,        // Permite el uso de comas
10      ↪ decimales en los números.
11    onlyGreekName: false,                // No convierte letras
11      ↪ griegas a nombres.
12    onlyGreekSymbol: false,              // No convierte letras
12      ↪ griegas a símbolos.
13    debugging: false,                   // No produce información
13      ↪ de depuración.
14  });
15 // Configuración del convertidor con opciones personalizadas.
16
17 // Define una cadena LaTeX

```

```

18 const latexInput = "\frac{\left( x^a \right) \left( x+y \right)}{x^{90}}^{25} \cdot x \cdot \pi";
19
20 // Convierte la cadena LaTeX a código Maxima usando el método
21 // 'toMaxima'.
22 const maximaOutput = converter.toMaxima(latexInput);
23
24 // Imprime el resultado convertido a Maxima en la consola.
25 console.log(maximaOutput);

```

Código 17: Script en NodeJS para probar la biblioteca de tex2max

Fuente: *Elaboración propia*

C.18. Script para implementar a MathQuill

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0"
6   />
7
8 <!-- Estilos necesarios para MathQuill -->
9 <link rel="stylesheet" href="styles.css" />
10 <link rel="stylesheet" href=".src/mathquill-0.10.1/mathquill.css" />
11
12 <!-- Cargar jQuery, ya que MathQuill depende de esta librería -->
13 <script
14   src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
15
16 <!-- Cargar el archivo JavaScript principal de MathQuill -->
17 <script src=".src/mathquill-0.10.1/mathquill.js"></script>
18
19 <title>MathQuill Input</title>
20 </head>
21 <body>
22 <div class="container">
23 <h1>Ingrese una ecuación</h1>
24
25 <!-- Campo donde se integrará MathQuill -->
26 <div id="math-field" class="math-field"></div>
27
28 <!-- Sección para mostrar el LaTeX generado -->
29 <p class="latex-text">LaTeX generado:</p>
30 <div id="latex-output" class="latex-output"></div>
31 </div>
32 // Inicializar la interfaz de MathQuill

```

```

33 const MQ = MathQuill.getInterface(2); // Obtener la versión 2 de la
   ↳ interfaz
34
35 // Referencia al contenedor del campo MathQuill
36 const mathFieldSpan = document.getElementById("math-field");
37
38 // Referencia al contenedor donde se mostrará el LaTeX generado
39 const latexOutput = document.getElementById("latex-output");
40
41 // Crear el campo MathQuill dentro del contenedor
42 const mathField = MQ.MathField(mathFieldSpan, {
43     spaceBehavesLikeTab: true, // Presionar espacio actúa como
       ↳ tabulación
44     handlers: {
45         // Controlador de eventos cuando se edita el campo
46         edit: function () {
47             const enteredMath = mathField.latex(); // 
               ↳ Obtener el contenido en formato LaTeX
48             latexOutput.textContent = enteredMath; // 
               ↳ Mostrar el LaTeX generado
49         },
50     },
51 });
52 </script>
53 </body>
54 </html>

```

Código 18: Código para usar MathQuill para introducir datos en T_EX
Fuente: Elaboración propia

C.19. Script para crear un teclado con MathJax y MathQuill

```

1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4  <meta charset="UTF-8" />
5  <meta name="viewport" content="width=device-width, initial-scale=1.0"
   ↳ />
6  <title>Teclado Matemático</title>
7
8  <!-- Estilos de MathQuill -->
9  <link rel="stylesheet" href="mathquill.css" />
10 <!-- MathJax para renderizar fórmulas en el teclado -->
11 <script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js"
   ↳ async></script>
12 <!-- jQuery y MathQuill -->
13 <script
   ↳ src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

```

```

14 <script src="mathquill.js"></script>
15 </head>
16 <body>
17 <div class="container">
18 <h1>Teclado Matemático</h1>
19
20 <!-- Teclado Matemático -->
21 <div class="math-keyboard">
22 <button data-latex="\pi">$$\pi$$</button>
23 <button data-latex="\frac{ }{ }">$$\frac{\square}{\square}$$</button>
24 <button data-latex="\sin\left( \right.
25   \left. \right)">$$\sin\left(\right.\left.\right)$$</button>
26 <button data-latex="\cos\left( \right.
27   \left. \right)">$$\cos\left(\right.\left.\right)$$</button>
28 <button data-latex="\{}^{\{}}">$$\square^{\square}$$</button>
29 <!-- Campo de entrada MathQuill -->
30 <div id="math-field" class="math-field"></div>
31 </div>
32
33 <script>
34 // Inicializar MathQuill
35 const MQ = MathQuill.getInterface(2);
36 const mathField = MQ.MathField(document.getElementById("math-field"), {
37   spaceBehavesLikeTab: true,
38 });
39
40 // Manejar clics en el teclado matemático
41 document.querySelectorAll(".math-keyboard button").forEach((button) =>
42   {
43     button.addEventListener("click", () => {
44       const latexToInsert =
45         button.getAttribute("data-latex");
46       mathField.write(latexToInsert); // Insertar LaTeX en el
47         campo activo
48     });
49   });
50 </script>
51 </body>
52 </html>

```

Código 19: Código para usar MathQuill para introducir datos en T_EX
Fuente: Elaboración propia

Bibliografía

- [1] J.M. Almira. *Fourier: un matemático al servicio de la física*. Genios de las matemáticas. RBA, 2017. ISBN: 9788447387755.
- [2] Wolfram: Computation Meets Knowledge. *Wolfram Mathematica: Modern Technical Computing*. <https://www.wolfram.com/mathematica/>. Accedido: 31 de Octubre de 2024. 2024.
- [3] Wolfram — Alpha: Computational Intelligence. *Fourier Analysis—Wolfram Language Documentation*. <https://reference.wolfram.com/language/guide/FourierAnalysis.html>. Accedido: 17 Junio, 2024. 2024.
- [4] Wolfram Language — System Documentation Center. *Piecewise: Function defined in pieces for different conditions—Wolfram Documentation*. <https://reference.wolfram.com/language/ref/Piecewise.html>. Accedido: 31 de Octubre de 2024. 2024.
- [5] Solucionador matemático Symbolab - calculadora paso a paso. *Solucionador matemático Symbolab - calculadora paso a paso*. <https://es.symbolab.com>. Accedido: 31 de Octubre de 2024. 2024.
- [6] MathWorks. *MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink*. <https://la.mathworks.com/>. Accedido: 14 Junio, 2024. 2024.
- [7] Engineering Maplesoft - Software for Mathematics Online Learning. *Computer Algebra Systems (CAS) - Maplesoft*. https://www.maplesoft.com/documentation_center/. Accedido: 3 Noviembre, 2024. 2024.
- [8] R. J. Lopez. *Classroom Tips and Techniques: Teaching Fourier Series with Maple - Part 3*. <https://www.maplesoft.com/applications/download.aspx?id=4885/TeachingFourierSerieswithMaple-Part3.pdf>. Accedido: 3 Noviembre, 2024. Maplesoft - Software for Mathematics, Online Learning, Engineering, 2024.
- [9] maxima. *Maxima, a Computer Algebra System. Version 5.47.0*. <https://maxima.sourceforge.io/>. Accedido: 27 Octubre, 2024. 2023.
- [10] GeoGebra. *GeoGebra - the world's favorite, free math tools used by over 100 million students and teachers*. <https://www.geogebra.org>. Accedido: 14 Junio, 2024. 2024.
- [11] Desmos. *Desmos — Let's learn together*. <https://www.desmos.com>. Accedido: 14 Junio, 2024. 2024.

- [12] Inc. Amazon Web Services. *¿Qué es Python? - Explicación del lenguaje Python - AWS.* <https://aws.amazon.com/es/what-is/python/>. Accedido: 31 de Octubre de 2024. 2024.
- [13] SymPy Development Team. *Sympy Documentation.* Accessed: 2023-10-01. 2023. URL: <https://docs.sympy.org/latest/>.
- [14] Matplotlib Development Team. *Matplotlib Documentation.* Accessed: 2023-10-01. 2023. URL: <https://matplotlib.org/stable/contents.html>.
- [15] Manim Community. *Manim Community.* <https://www.manim.community>. Accedido: 14 Junio, 2024. 2024.
- [16] A. Cañada. *Una perspectiva histórica de las series de Fourier: de las ecuaciones de ondas y del calor a los operadores compactos y autoadjuntos.* <https://www.nieuwarchief.nl/serie5/pdf/naw5-2000-01-3-242.pdf>. Accedido: 23 Octubre, 2024. 2000.
- [17] H. F. Weinberger. *Ecuaciones diferenciales en derivadas parciales con métodos de variable compleja y de transformadas integrales.* Editorial Reverté, 1970. ISBN: 9788429151602.
- [18] O. L. Campo Bedoya. *Ecuación de D'Alembert, de la cuerda vibrante, bajo la teoría de Lie.* <https://repositorio.unal.edu.co/handle/unal/52162>. Accedido: 23 Octubre, 2024. 2014.
- [19] D. Chamorro. *Lección n°5: Ecuación de Ondas.* https://www.amarun.org/images/amarun/materiales/EDP/edp/leccion_5.pdf. Accedido: 23 Octubre, 2024. 2015.
- [20] I. Grattan-Guinness. *Daniel Bernoulli and the varieties of mechanics.* <https://www.nieuwarchief.nl/serie5/pdf/naw5-2000-01-3-242.pdf>. Accedido: 23 Octubre, 2024. 2000.
- [21] E. Garber. *The Language of Physics: The Calculus and the Development of Theoretical Physics in Europe, 1750-1914.* Springer, 1999. ISBN: 978-1-4612-7272-4.
- [22] J. T. Brereton. *FOURIER SERIES: SOLVING THE HEAT EQUATION.* <https://math.berkeley.edu/~jbrere/heatequation>. Accedido: 23 Octubre, 2024. 2015.
- [23] J. Feldman. *Solution of the Heat Equation by Separation of Variables.* <https://personal.math.ubc.ca/~feldman/m267/heatSln.pdf>. Accedido: 23 Octubre, 2024. 2007.
- [24] G. P. Tolstov. *Fourier Series.* Prentice-Hall, Inc., 1962. ISBN: 9780486633176.
- [25] H. P. Hsu. *Analisis de Fourier.* Addison-Wesley Iberoamericana, 1987. ISBN: 9685000476.
- [26] W. Gómez Flores. *Introducción al Análisis de Fourier.* https://tamps.cinvestav.mx/~wgomez/documentos/analisis_de_fourier.pdf. Accedido: 26 Octubre, 2024. Cinvestav, 2024.

- [27] W. Moebs. *15.1 Movimiento armónico simple - Física universitaria volumen 1 — OpenStax*. <https://openstax.org/books/fsica-universitaria-volumen-1/pages/15-1-movimiento-armonico-simple>. Accedido: 26 Octubre, 2024.
- [28] D. G. Zill. *Matemáticas Avanzadas para la Ingeniería*. McGraw-Hill, 2011. ISBN: 9786071507723.
- [29] D. Poole. *Linear Algebra, A Modern Introduction 3rd ed.* Cengage Learning, 2011. ISBN: 9780538735452.
- [30] C. F. Cruz Fierro. *Ecuaciones Diferenciales / Matemáticas 5 / Series de Fourier*. <https://cruzfierro.com/cursos/2011v/matematicas5/apuntes5.pdf>. Accedido: 27 Octubre, 2024.
- [31] Matemáticas con Luz. *Serie Compleja de Fourier*. <https://www.youtube.com/watch?v=3F4XReHr3i8>. Accedido: 27 Octubre, 2024. Youtube, 2020.
- [32] C. J. Carrillo González. *Fundamentos del Análisis de Fourier*. https://grupo_ene.webs.uvigo.es/wordpress/publicaciones/Apuntes_Fourier.pdf. Accedido: 27 Octubre, 2024. Youtube, 2003.
- [33] UNIVERSIDAD NACIONAL DEL SUR Dto. de Ingeniería Eléctrica y de computadoras. *S. y T. de Fourier*. http://lcr.uns.edu.ar/fvc/series_de_fourier.htm. Accedido: 16 Noviembre, 201P.
- [34] Gowri Shankar. *Fourier Series as a Function of Approximation for Seasonality Modeling - Exploring Facebook Prophet's Architecture*. <https://gowrishankar.info/blog/fourier-series-as-a-function-of-approximation-for-seasonality-modeling-exploring-facebook-prophets-architecture/>. Accedido: 16 Noviembre, 2021.
- [35] ¿Qué es una aplicación web? <https://aws.amazon.com/es/what-is/web-application/>. Accedido: 27 Octubre, 2024. Amazon Web Services, Inc., 2023.
- [36] Introducción a la Arquitectura web. <https://wiki.uqbar.org/wiki/articles/ui-web-intro-arquitectura.html>. Accedido: 29 Octubre, 2024. uqbar-wiki, 2023.
- [37] MDN Web Docs. *HTML: HyperText Markup Language — MDN*. https://www.w3schools.com/graphics/canvas_intro.asp. Accedido: 27 Octubre, 2024.
- [38] W3Schools.com. *CSS: Cascading Style Sheets — MDN*. <https://developer.mozilla.org/en-US/docs/Web/HTML>. Accedido: 29 Octubre, 2024.
- [39] MDN Web Docs. *JavaScript — MDN*. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Accedido: 29 Octubre, 2024.
- [40] Dazzet. *Qué es una API, cómo funciona y para qué sirve - Dazzet*. <https://dazzet.co/que-es/api/>. Accedido: 2 Noviembre, 2024.
- [41] Editorial Etecé. *HTTP - Concepto, para qué sirve y cómo funciona*. <https://concepto.de/http/>. Accedido: 2 Noviembre, 2024.

- [42] MDN Web Docs. *Métodos de petición HTTP - HTTP — MDN*. <https://developer.mozilla.org/es/docs/Web/HTTP/Methods>. Accedido: 13 Noviembre, 2024.
- [43] MDN Web Docs. *Códigos de estado de respuesta HTTP — MDN*. <https://developer.mozilla.org/es/docs/Web/HTTP/Status>. Accedido: 13 Noviembre, 2024.
- [44] Kinsta®. *Qué es Node.js y por qué debería usarlo*. <https://kinsta.com/es/base-de-conocimiento/que-es-node-js/>. Accedido: 2 Noviembre, 2024.
- [45] Software Testing — QA - QAlified. *¿Cómo utilizar Postman para las pruebas de API?* <https://qalified.com/es/blog/postman-para-api-testing/>. Accedido: 13 Noviembre, 2024.
- [46] IETF Datatracker. *RFC 2818: HTTP Over TLS*. <https://datatracker.ietf.org/doc/html/rfc2818>. Accedido: 2 Noviembre, 2024.
- [47] SSL.com. *¿Qué es HTTPS? - SSL.com*. <https://www.ssl.com/es/preguntas-frecuentes/que-es-https/>. Accedido: 2 Noviembre, 2024.
- [48] SSL.com. *Que es SSL /TLS: Una guía detallada - SSL.com*. <https://www.ssl.com/es/article/what-is-ssl-tls-an-in-depth-guide/>. Accedido: 2 Noviembre, 2024.
- [49] Alura. *CORS ¿Qué es y cómo resolver un error de Cross-Origin Resource Sharing (CORS)? — Alura Cursos Online*. <https://www.aluracursos.com/blog/que-es-cors>. Accedido: 15 Noviembre, 2024.
- [50] Angular. *Home • Angular*. <https://angular.dev/overview>. Accedido: 29 Octubre, 2024.
- [51] TypeScript: JavaScript With Syntax For Types. *The starting point for learning TypeScript*. <https://www.typescriptlang.org/docs/>. Accedido: 29 Octubre, 2024.
- [52] Documentation - Tailwind CSS. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. <https://tailwindcss.com/docs/>. Accedido: 29 Octubre, 2024.
- [53] W3Schools.com. *W3Schools Online Web Tutorials*. <https://developer.mozilla.org/en-US/docs/Web/HTML>. Accedido: 29 Octubre, 2024.
- [54] H. Seoul-Oh M. Stufflebeam. *MathQuill: Easily type math into your webapp*. <http://mathquill.com>. Accedido: 13 Noviembre, 2024.
- [55] MathJax Documentation — MathJax 3.2 documentation. *What is MathJax?* — *MathJax 3.2 documentation*. <https://docs.mathjax.org/en/latest/basic/mathjax.html>. Accedido: 13 Noviembre, 2024.
- [56] A. Storhaug. *GitHub - KQMATH/tex2max: :books: JavaScript library for converting LaTeX math into Maxima code*. <https://github.com/KQMATH/tex2max>. Accedido: 13 Noviembre, 2024.
- [57] Microsoft. *Microsoft Azure*. <https://azure.microsoft.com/>. ÚAccedido: 27 Octubre, 2024.

- [58] M. C. Albornoz. *Diseño de interfaz gráfica de usuario*. <https://sedici.unlp.edu.ar/handle/10915/41578>. Accedido: 27 Octubre, 2024. SEDICI, 2014.
- [59] Semrush Blog. *Principios de usabilidad web de Jacob Nielsen y el diseño UX*. <https://es.semrush.com/blog/usabilidad-web-principios-jakob-nielsen/>. Accedido: 2 Noviembre, 2024. 2022.
- [60] MathWorks. *Metodologías de desarrollo*. <https://woodyweb.wordpress.com/2015/08/24/metodologias-de-desarrollo/>. Accedido: 22 Oct, 2024. 2015.