MATEMÁTICA SUPERIOR

Ingeniería en Sistemas de Información

Trabajo Práctico N°1

2023

Alumnos	Correo electrónico
Briani, Nicolás	nicolasj.briani@hotmail.com
Springer, Nicolas	nicosspringer16@gmail.com
Tardivo, Juan	juanaugustotardivo@hotmail.com
Molina, Jeremías	jere.movale@gmail.com
Mahieu, Mateo	mateomahieu01@gmail.com

ÍNDICE:

Introducción

El filósofo griego Eudoxo, en el siglo IV AC, fue el primero que propuso un modelo del Universo en el que el Sol, la Luna, los planetas y las estrellas giraban en torno a la Tierra siguiendo círculos perfectos. El modelo de Eudoxo, llamado geocéntrico, fue mejorado alrededor del 140 DC por el filósofo alejandrino Claudio Ptolomeo. Ptolomeo imaginó un Universo en el que todos los cuerpos giraban alrededor de la Tierra, pero no describiendo círculos perfectos. Según Ptolomeo, los planetas realizan pequeños movimientos circulares (epiciclos) alrededor de un punto imaginario que se desplazaba alrededor de la Tierra recorriendo un círculo perfecto (deferente).

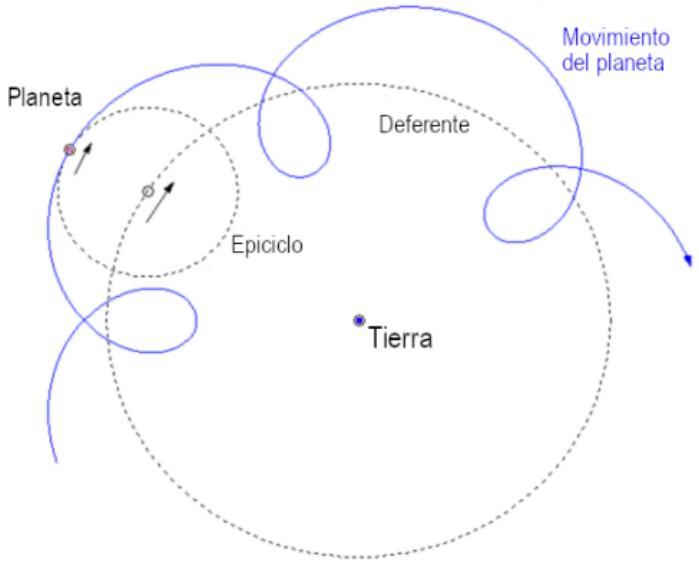


Figura 1 - Los planetas giran sobre un epiciclo que a su vez gira sobre un deferente.

Objetivos

Graficación de Imágenes

A partir de esta misma idea, podemos pensar que una imagen puede ser descrita por los puntos que N epiciclos van dibujando a medida que rotan en el tiempo. Podemos determinar su posición espacial en diferentes instantes de tiempo a través de la siguiente expresión matemática:

$$p_k[t] = \sum_{k=0}^{N} C_k e^{i\omega_k t}$$

donde pk[t] es la coordenada del punto en el plano complejo para el instante de tiempo t, ωk = ω0k es la frecuencia del sistema y Ck es una constante compleja.

1. Realice el desarrollo matemático correspondiente para poder obtener, de forma simplificada, las coordenadas x e y correspondientes a los valores de pk. ¿A qué frecuencia puede asociarse la posición de la tierra?
2. Considere el archivo epiciclos.txt que contiene valores de Ck y los radios Rk de cada epiciclo. Obtenga el parámetro necesario para poder observar la imagen deseada de acuerdo al desarrollo anterior y grafique los resultados. Considere N igual al tamaño del archivo. ¿Cuáles son los tiempos máximo y paso de tiempo ideales para obtener la mejor resolución de imagen? Justifique
3. Realice un análisis de lo que ocurre al variar parámetros fundamentales como el número de epiciclos necesarios para graficar y obtener conclusiones. ¿Deben ser todas las imágenes de contorno cerrado para que puedan graficarse con este método? Justifique.

Aproximación de coeficientes de Fourier

Dada la función compleja C(t) para t real, se desea representar la extensión periódica de dicha función por medio de la serie continua de Fourier:

$$C_{periodica}(t) = \sum_{K=-\infty}^{\infty} C_K e^{i\omega_0 K t}$$

donde CK son los coeficientes complejos. De dicha función solo se tiene la tabla de valores que se adjunta Colibrí.csv.

1. Represente la secuencia de datos en el plano complejo.
2. Aproxime a la función original como:

$$C^*(t) \approx \sum_{K=-M}^{M} \zeta_K e^{i\omega_0 K t}$$

donde ζK es una aproximación a los CK con un M adecuado para que la aproximación sea buena.
Muestre gráficamente la aproximación.
Nota: Para obtener los ζK se debe aproximar a integral que define los CK por medio de la suma de Riemann. Si solo del integrando se conoce los n+1 valores (Z0, Z1, . . . , Zn) distribuidos uniformemente en el intervalo [a ; b] se puede aproximar a la integral definida por medio de:

$$\int_a^b Z(t)\,dt \approx \frac{(b-a)}{n} \sum_{i=1}^{n} \tilde{Z}_i \qquad \text{donde } \tilde{Z}_i = \frac{Z_{i-1}+Z_i}{2}.$$

3. ¿Pasa la aproximación por los puntos? ¿Qué pasa si se toma un M menor al que determinó en 2)? ¿Y si es mayor? Fundamente sus resultados.

Resoluciones

Graficación de imágenes

1) Para poder obtener, de forma simplificada, las coordenadas x e y correspondientes a los valores de pk primero consideraremos un punto que se mueve sobre la circunferencia de un círculo, por lo que podemos definir la ecuación paramétrica:

$$x(t) = R\,cos(wt)$$
$$y(t) = R\,sen(wt)$$

Donde *R* es el radio del disco y *ω* es la velocidad con la cual el punto está rotando.

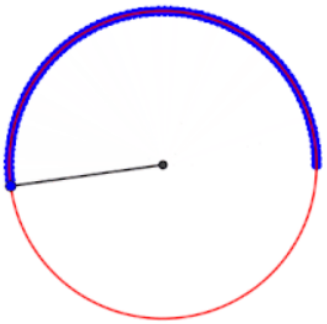


Figura 1 - Punto que se mueve sobre la circunferencia de un círculo.

Ahora, si agregamos un segundo círculo con su propio punto rotando en su circunferencia. podemos observar la siguiente imagen:

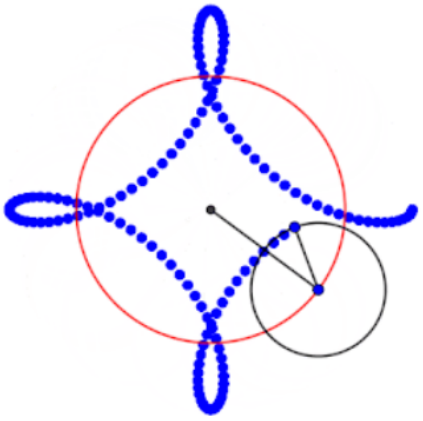


Figura 2 - Segundo círculo con su propio punto rotando en su circunferencia.

Por lo que ahora la posición de este punto es solamente la suma de cada una de las partes anteriores. En particular, el punto en un tiempo *t* es la posición ($x_1(t)$, $y_1(t)$) con respecto al primer círculo más la posición ($x_2(t)$, $y_2(t)$) del segundo círculo:

$$x(t) = R_1\,cos(w_1 t) + R_2\,cos(w_2 t)$$
$$y(t) = R_1\,sen(w_1 t) + R_2\,sen(w_2 t)$$

Donde R_1 es el disco de radio del primer círculo y R_2 es el segundo radio.

Una vez obtenidas las ecuaciones anteriores, podemos generalizar para una cantidad mayor de círculos, en los cuales también varían sus radios y velocidades angulares. Dicho lo anterior, se determinan las siguientes ecuaciones paramétricas:

$$x(t) = \sum_i R_i \, cos(w_i t + \phi_i)$$
$$y(t) = \sum_i R_i \, sen(w_i t + \phi_i)$$

Donde ϕ_i determina la rotación *i* inicial en un tiempo $t = 0$, la cual se denomina fase inicial. Si no especificamos esta fase inicial, podremos describir solo epiciclos donde los círculos comienzan alineados, lo cual es solamente una porción de un conjunto de epiciclos.

Cuantos más círculos tengamos, más complejas serán las curvas graficadas. Con esta premisa, si tenemos suficientes círculos, elegimos los tamaños correctos para cada uno de ellos y los giramos a las velocidades adecuadas, podemos crear cualquier curva cerrada. Entonces, usando epiciclos, podemos dibujar cualquier curva cerrada.

Por lo tanto, las coordenadas halladas:

$$x(t) = \sum_{i}^{N} R_i \cos(\omega_i t + \phi_i)$$
$$y(t) = \sum_{i}^{N} R_i \, \mathrm{sen}(\omega_i t + \phi_i)$$

Se aproximan a los puntos tanto como sea posible.
Al observar las ecuaciones, como se mencionó anteriormente, cuanto más círculos tengamos, más nos acercaremos a la curva deseada.

Vimos cómo trazar curvas continuas cerradas utilizando discos giratorios y variando su velocidad y radio. El problema que queremos resolver ahora es: ¿Dada cualquier curva/forma continua cerrada, es posible trazar usando epiciclos?
En lugar de describir la curva mediante ecuaciones paramétricas, supongamos que simplemente especificamos algunos puntos y luego encontramos una fórmula para conectar esos puntos en la forma que queremos. Esto significa que nuestra entrada es un conjunto finito de puntos (x_k, y_k), y queremos encontrar epiciclos que conecten mejor esos puntos. Es decir, queremos determinar los valores de R_k, ω_k y ϕ_k de manera que la expresión

$$\begin{cases} x(t) = \sum_{k=1}^{N} R_k \cos(\omega_k t + \phi_k) \\ y(t) = \sum_{k=1}^{N} R_k \sin(\omega_k t + \phi_k) \end{cases}$$

nos acerque lo más posible a los puntos. Luego podemos usar los puntos para dibujar la forma que queremos y, en consecuencia, tendremos nuestros epiciclos. ¿Pero cómo podemos determinar los valores de R_k, ω_k y ϕ_k?
Podemos utilizar la Transformada Discreta de Fourier para resolver este problema. En este caso, en lugar de pensar en nuestros puntos definidos en el plano real, podemos considerar estos puntos como números complejos. Así podemos escribir

$$p_k = x_k + iy_k$$

y la curva cerrada

$$x(t) + iy(t) = \sum_{k=1}^{N} R_k(\cos(\omega_k t + \phi_k) + i\sin(\omega_k t + \phi_k))$$

(1)

se acercará lo más posible a los puntos complejos p_k. También que la ecuación (1) se puede reescribir, utilizando la fórmula de Euler e^it = cos(t) + i sen(t), de la siguiente manera:

$$x(t) + iy(t) = \sum_{k=1}^{N} R_k e^{i\omega_k t + \phi_k}.$$

Si X_k es un número complejo

$$\sum_{k=1}^{N} X_k e^{i\omega_k t}.$$

Para finalizar esta sección, luego del desarrollo presentado anteriormente, nos enfocaremos en dar respuesta particular a nuestro problema de encontrar las coordenadas x e y correspondientes a los valores de P_k.

Si, en lugar de pensar en nuestros puntos a lo largo de los ejes x e y, pensemos en esos puntos como números complejos a lo largo de los ejes real e imaginario, podemos utilizar la forma polar de los números complejos.

La expresión general para $p_k[t]$ es:

$$p_k[t] = \Sigma\ C_k\ e^{(iwkt)}$$

Podemos escribir Ck en forma polar y reemplazar esto en la expresión de pk[t], obteniendo lo siguiente:

$$Ck = |Ck| * e^{(i\varphi k)}$$

$$pk[t] = \Sigma\ |Ck| * e^{(i(\omega kt + \varphi k))}$$

$$pk[t] = \Sigma\ |Ck| * (\cos(\omega kt + \varphi k) + i*\sin(\omega kt + \varphi k))$$

Las coordenadas x e y corresponden a la parte real e imaginaria de pk[t], respectivamente:

$$\mathbf{x[t]} = \text{Real}(p_k[t]) = \sum_{k=1}^{N} C_k \cos(\omega_k t + \varphi_k)$$

$$\mathbf{y[t]} = \text{Imaginaria}(p_k[t]) = \sum_{k=1}^{N} C_k \sin(\omega_k t + \varphi_k)$$

En cuanto a la frecuencia asociada a la posición de la Tierra, podemos considerar que es la frecuencia fundamental ω_0. En el modelo geocéntrico de Ptolomeo, todos los epiciclos están relacionados alrededor de la Tierra, por lo que comparten la misma frecuencia fundamental. Se asume que todos los epiciclos están relacionados alrededor de la Tierra y comparten la misma frecuencia fundamental ω_0. Esto significa que todos los cuerpos se mueven en sus órbitas con la misma velocidad angular básica ω_0, y las diferencias en sus trayectorias aparentes se logran combinando diferentes radios y fases de los epiciclos.

2)

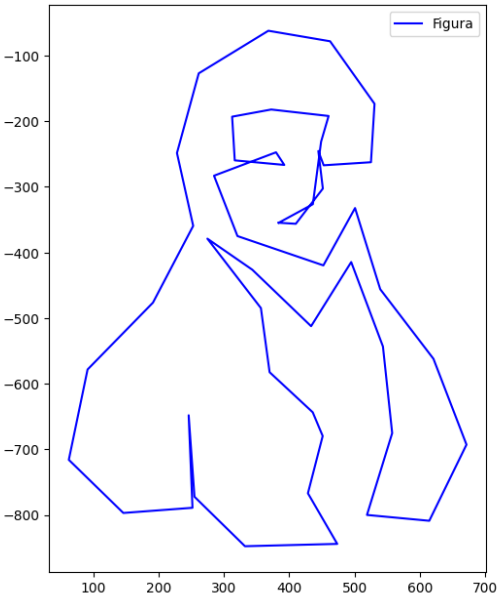
Los tiempos máximo y paso de tiempo ideales para obtener la mejor resolución de imagen en la animación de los epiciclos dependen de varios factores: la figura a graficar, la cantidad de puntos en la curva y la capacidad de procesamiento. No hay una respuesta única es por ello,vamos a hacer algunas aproximaciones generales:

Tiempo máximo (T): El tiempo máximo debe ser lo suficientemente grande para permitir que la animación muestre varios ciclos completos de los epiciclos. Esto permitirá apreciar la evolución completa de la figura. Sin embargo, un tiempo mayor podría hacer que la animación sea tediosa como es el caso de cuando utilizamos un valor largo de este. Un valor adecuado podría ser alrededor de 10 segundos en el cual luego de varias pruebas, concluimos que es un tiempo razonable en donde no debemos esperar demasiado tiempo para la animación completa.

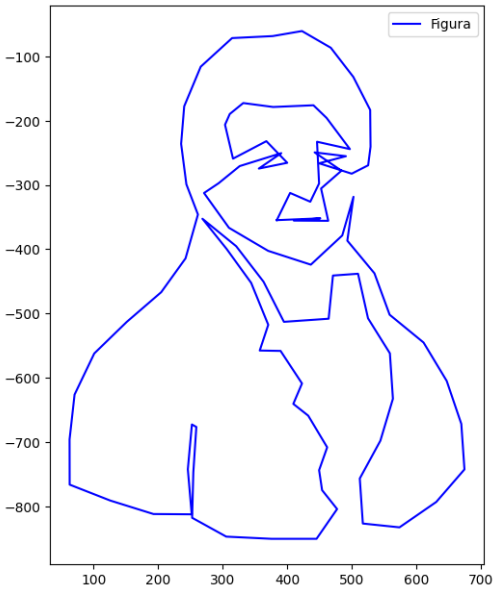
Paso de tiempo (dt): En este caso, es complemento lo inverso a lo descrito anteriormente, debe ser lo suficientemente pequeño para obtener una animación fluida y detallada. Sin embargo, si es demasiado pequeño, la animación puede volverse lenta y requerir mucho tiempo para completarse. Un valor razonable para el paso de tiempo podría ser alrededor de 0.01 segundos. Con la finalidad de realizar las pruebas sin esperar a que la animación finalice, decidimos pre graficar la figura completa y luego sobre esta, realizar la animación de cómo se va dibujando.
Un paso de tiempo más pequeño proporciona una animación más suave y detallada. Esto significa que se muestran más puntos a lo largo de la trayectoria, lo que permite apreciar mejor los detalles de la figura dibujada por los epiciclos.
Un paso más grande, hace que la animación se vea menos fluida, ya que hay menos puntos en la trayectoria.
A medida que el paso de tiempo (dt) se hace más pequeño, la cantidad de puntos en la animación aumenta significativamente. Esto ralentiza el tiempo de cálculo y la visualización, especialmente para figuras complejas con muchos epiciclos. Un paso de tiempo más grande reduce el número de puntos y acelerará el tiempo de cálculo y visualización, pero a expensas de la resolución y la suavidad de la animación, como podemos apreciar en los primeros ejemplos provistos de graficación de la figura.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import re

#Convierte una cadena con formato "real + imag" a un número complejo.
#Tuvimos que formatear el archivo epiciclos.txt para tener uno más fácil
de manejar
def complex_from_str(s):
    real, imag = re.findall(r'[-+]?\d*\.\d+|\d+', s)
    return complex(float(real), float(imag))

#Traza la curva cerrada y los epiciclos basados en los radios y los
coeficientes complejos Xk (Ck). Le pusimos de nombre Xk para seguir con el
desarrollo teórico descrito en el apartado 1,
def plot_epicycles(radios, Xk, num_points=10000, T=10, dt=0.001):
    N = len(radios)
    t_values = np.linspace(0, 2 * np.pi, num_points)
    z_t = np.zeros(num_points, dtype=complex)

    #Calculamos las coordenadas complejas z(t) usando los radios y
coeficientes complejos
    for i, t in enumerate(t_values):
        z_t[i] = np.sum([Xk[k] * np.exp(1j * k * t) for k in range(-N //
2, N // 2)])

    x_values = np.real(z_t)
    y_values = np.imag(z_t)

    #Creamos la figura una sola vez antes de la animación para poder ver
los resultados rápido sin esperar a la animación y luego realizamos la
misma sobre la figura para que pueda observarse.
    fig, ax = plt.subplots(figsize=(8, 8))
    line_fixed, = ax.plot(x_values, y_values, '-b', label='Figura')
    line_animation, = ax.plot([], [], '-r', label='Animación')
    ax.set_aspect('equal', adjustable='box')
    ax.legend()

    #Función que inicia la animación
    def init():
        line_animation.set_data([], [])
```

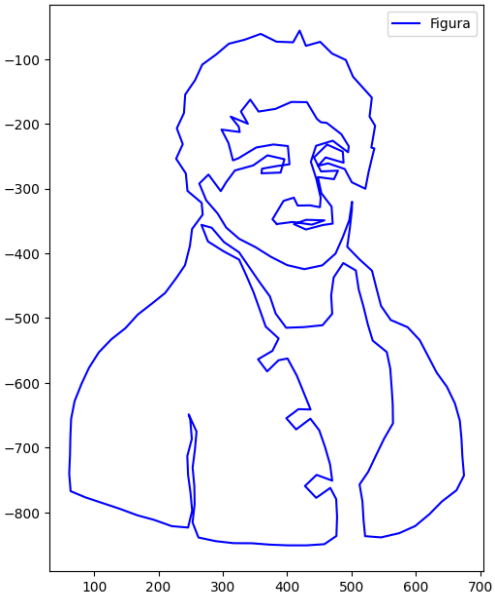
```
        return line_animation,

    #Función para animar los epiciclos
    def animate(i):
        line_animation.set_data(np.real(z_t[:i]), np.imag(z_t[:i]))
        return line_animation,

    #Configuramos los tiempos máximos y el paso de tiempo para la
animación
    T_max = T
    dt_anim = dt

    #Calculamos el número de pasos para la animación
    num_steps = int(T_max / dt_anim)

    #Creamos la animación
    anim = FuncAnimation(fig, animate, init_func=init, frames=num_steps,
interval=dt_anim*1000, blit=True)

    plt.show()

#Lee los datos desde el archivo epiciclos.txt el cual formateamos para que
sea radio + "espacio en blanco" + (parte real+parte compleja j)
file_path = 'epiciclos.txt'

with open(file_path, 'r') as file:
    lines = file.readlines()

radios = []
Xk = []

#Extrae los radios y coeficientes complejos de cada línea del archivo
for line in lines[1:]:
    line_data = line.strip().split()
    radios.append(float(line_data[0]))
    Xk.append(complex_from_str(line_data[1]))

#Utilizamos la función plot_epicycles con los datos del archivo
epiciclos.txt para obtener y graficar las coordenadas x e y.
plot_epicycles(radios, Xk)
```



Cantidad de puntos utilizados: 50

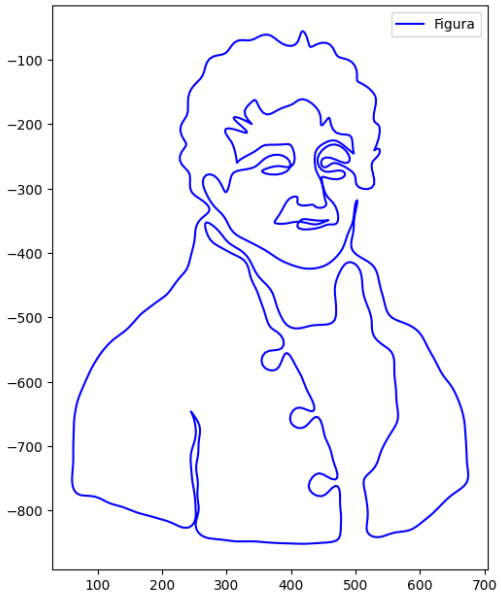


Cantidad de puntos utilizados: 100



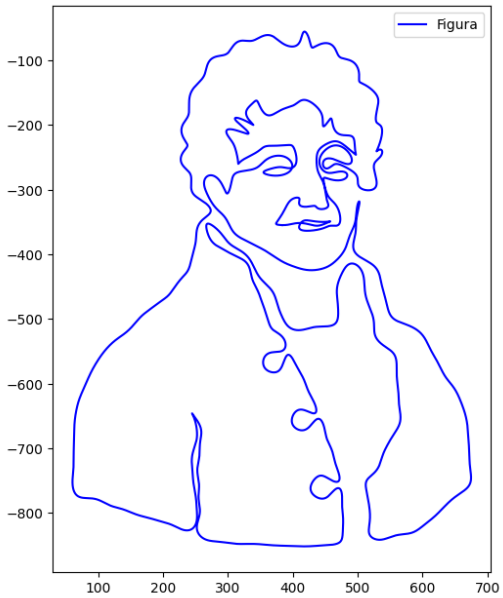
Cantidad de puntos utilizados: 250

Cantidad de puntos utilizados: 500

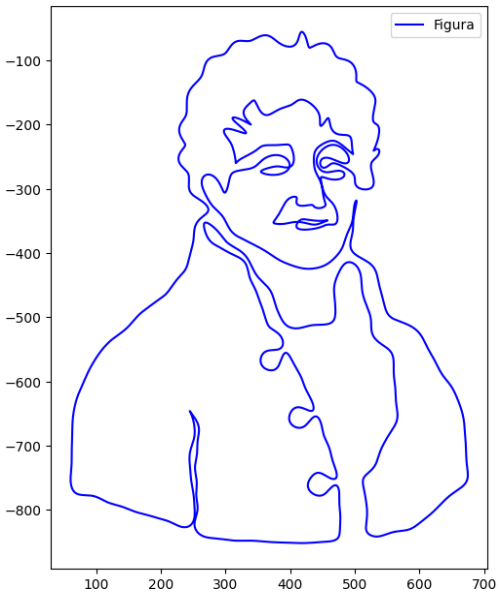


Cantidad de puntos utilizados: 1000

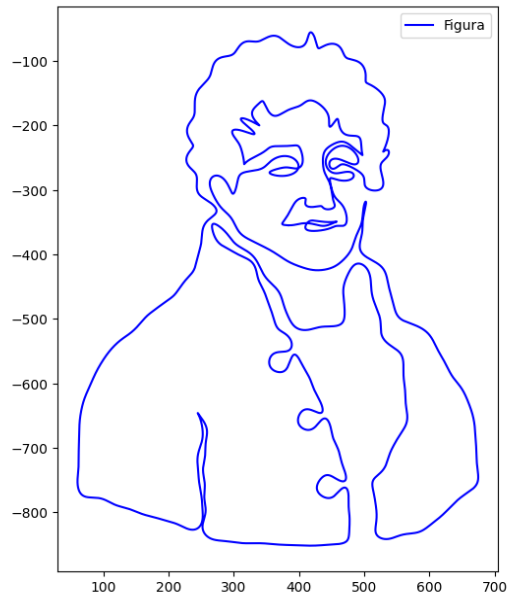
Cantidad de puntos utilizados: 5000



Cantidad de puntos utilizados: 10000



Cantidad de puntos utilizados: 15000



Cantidad de puntos utilizados: 25000

Cantidad de puntos utilizados: 50000

3)

Al variar parámetros fundamentales como el número de epiciclos **N**, podemos observar diferentes comportamientos:
Si N es pequeño, por ejemplo 1 o 2, obtenemos trayectorias simples y cerradas, ya que cada epiciclo realizará una órbita básica. A medida que aumenta N, las trayectorias se vuelven más complejas y se generan patrones más detallados y elaborados en la imagen. Estos valores altos de N aumentan la complejidad computacional.

En cuanto a la pregunta ¿Deben ser todas las imágenes de contorno cerrado para que puedan graficarse con este método?

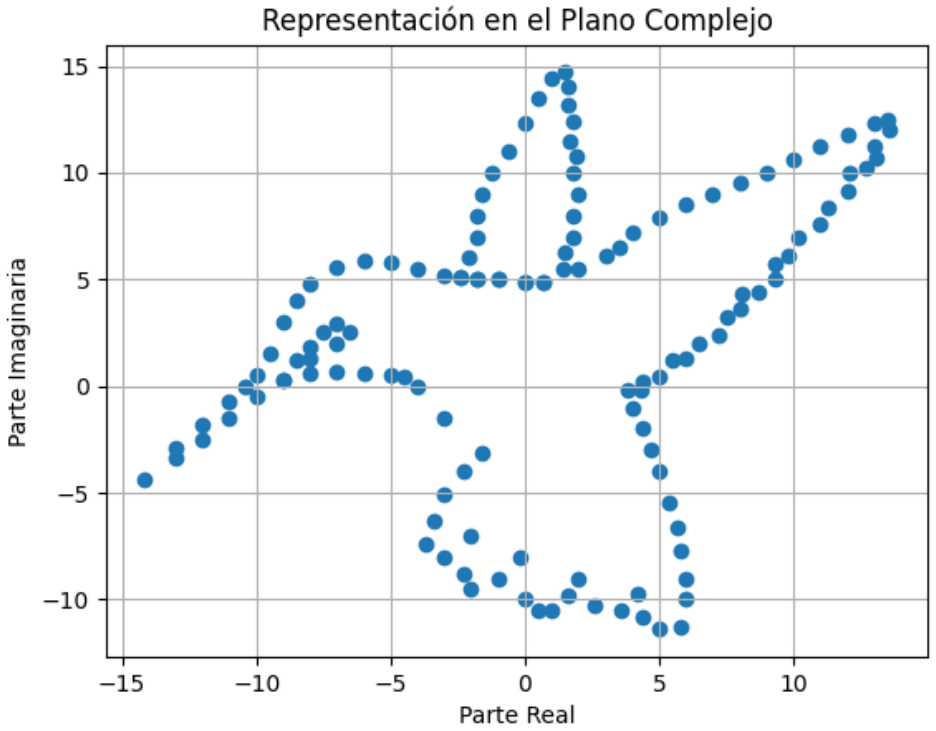
La respuesta es sí, todas las imágenes de contorno deben ser cerradas para que puedan graficarse correctamente con este método. Esto se debe a que el método de los epiciclos se basa en la suma de funciones periódicas (movimiento circular) para crear una trayectoria cerrada.
Si la imagen de contorno no es cerrada, los epiciclos no podrán representarla adecuadamente, ya que los epiciclos están diseñados para repetirse a lo largo del tiempo.
Por lo tanto, para aplicar el método de los epiciclos y obtener una representación significativa de una imagen de contorno, la imagen debe ser cerrada, y modificando el número de epiciclos N, podemos obtener figuras más o menos precisas.

Aproximación de coeficientes de Fourier

El siguiente código, realiza la gráfica de los puntos en el plano complejo como indica el inciso 1 y además muestra la aproximación como indica el inciso 2.

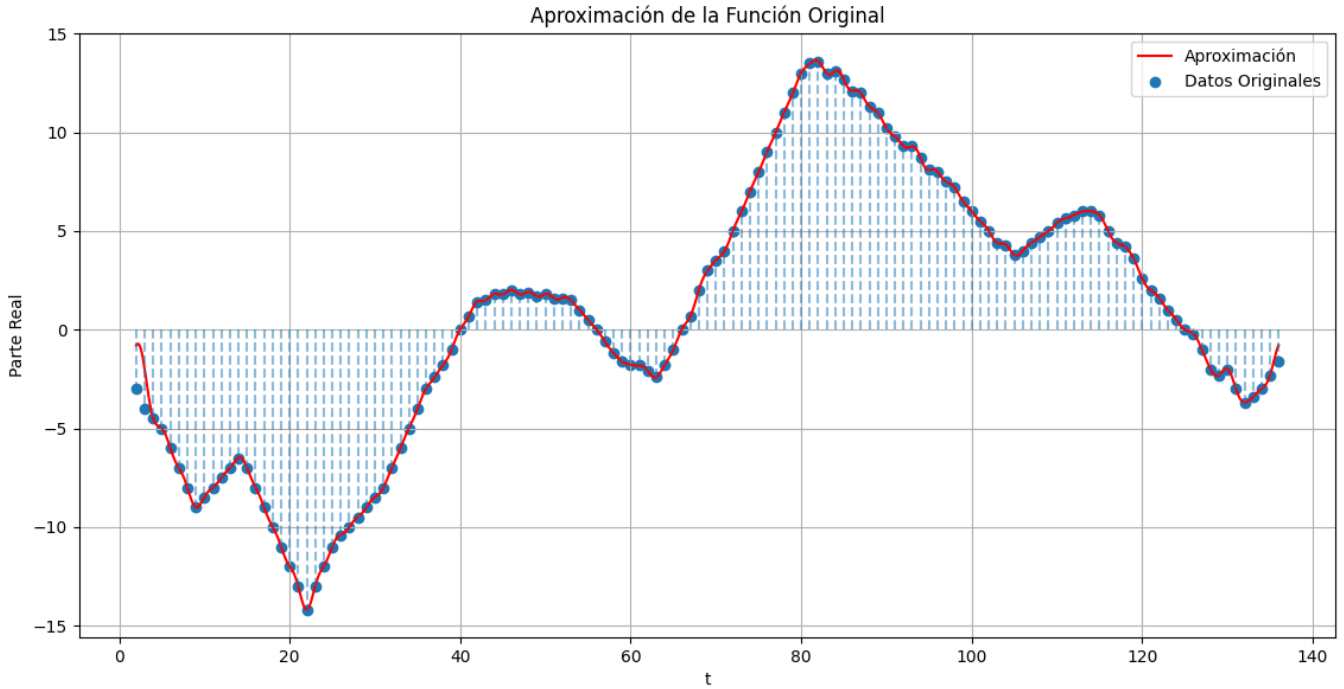
```
import numpy as np
import matplotlib.pyplot as plt

#Cargamos los datos desde Colibri.csv
with open('Colibri.csv', 'rb') as file:
    data = np.genfromtxt(file, delimiter=';', skip_header=2, usecols=(0, 1,
2), converters={1: lambda s: float(s.decode('utf-8').replace(',', '.')), 2:
lambda s: float(s.decode('utf-8').replace(',', '.'))})

t = data[:, 0]  # Valores de t
Z = data[:, 1] + 1j*data[:, 2]  # Valores complejos de Z

#Graficamos los valores complejos de Z en el plano complejo
plt.figure()
plt.scatter(Z.real, Z.imag)
plt.xlabel('Parte Real')
plt.ylabel('Parte Imaginaria')
plt.title('Representación en el Plano Complejo')
plt.grid(True)

#Aproximamos la función original usando la fórmula de la suma de Riemann
M = 68
omega_0 = 2*np.pi/(t[-1]-t[0])  # Frecuencia fundamental
zeta = np.zeros((2*M+1,), dtype=np.complex128)  # Coeficientes de Fourier
h = (t[-1] - t[0]) / len(t) # (b-a) / n

for k in range(-M, M+1):
    Z_k = Z[1:]*np.exp(-1j*k*omega_0*t[1:])
    zeta[k+M] = h * np.sum((Z_k[1:] + Z_k[:-1])/2)

#Ejemplos:
# Z_k[1:]: [1.0, 2.0, 3.0, 4.0] # Todos los valores de Z_k excepto el último
# Z_k[:-1]:  [2.0, 3.0, 4.0, 5.0] # Todos los valores de Z_k excepto el
primero
# Z_k[1:] + Z_k[:-1] = [3.0, 5.0, 7.0, 9.0]
# (Z_k[:-1] + Z_k[1:]) / 2: =  [1.5, 2.5, 3.5, 4.5]

#Construimos la aproximación y la graficamos junto con los datos originales
```

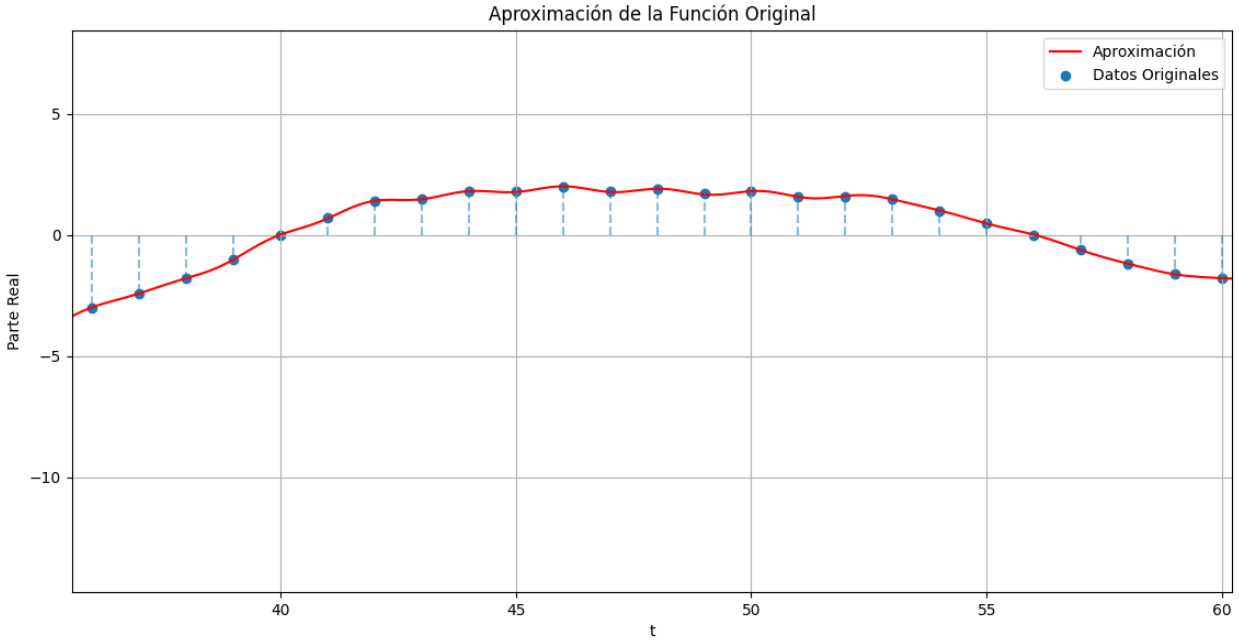
```
t_aprox = np.linspace(t[0], t[-1], 1000)
C_aprox = np.zeros_like(t_aprox, dtype=np.complex128)
for k in range(-M, M+1):
    C_aprox += zeta[k+M]*np.exp(1j*k*omega_0*t_aprox)

#Escalamos la aproximación para que coincida con los datos originales
factor_escala = np.abs(Z).max() / np.abs(C_aprox).max()
C_aprox *= factor_escala

plt.figure()
plt.plot(t_aprox, C_aprox.real, label='Aproximación')
plt.scatter(t, Z.real, label='Datos Originales')
plt.vlines(t, ymin=0, ymax=Z.real, linestyle='--', alpha=0.5)
plt.xlabel('t')
plt.ylabel('Parte Real')
plt.title('Aproximación de la Función Original')
plt.legend()
plt.grid(True)

#Muestra la aproximación graficando el colibrí
plt.figure()
plt.plot(C_aprox.real, C_aprox.imag, label='Aproximación')
plt.scatter(Z.real, Z.imag, label='Datos Originales')
plt.xlabel('Parte Real')
plt.ylabel('Parte Imaginaria')
plt.title('Aproximación formando el Colibri')
plt.legend()
plt.grid(True)

plt.show()
```

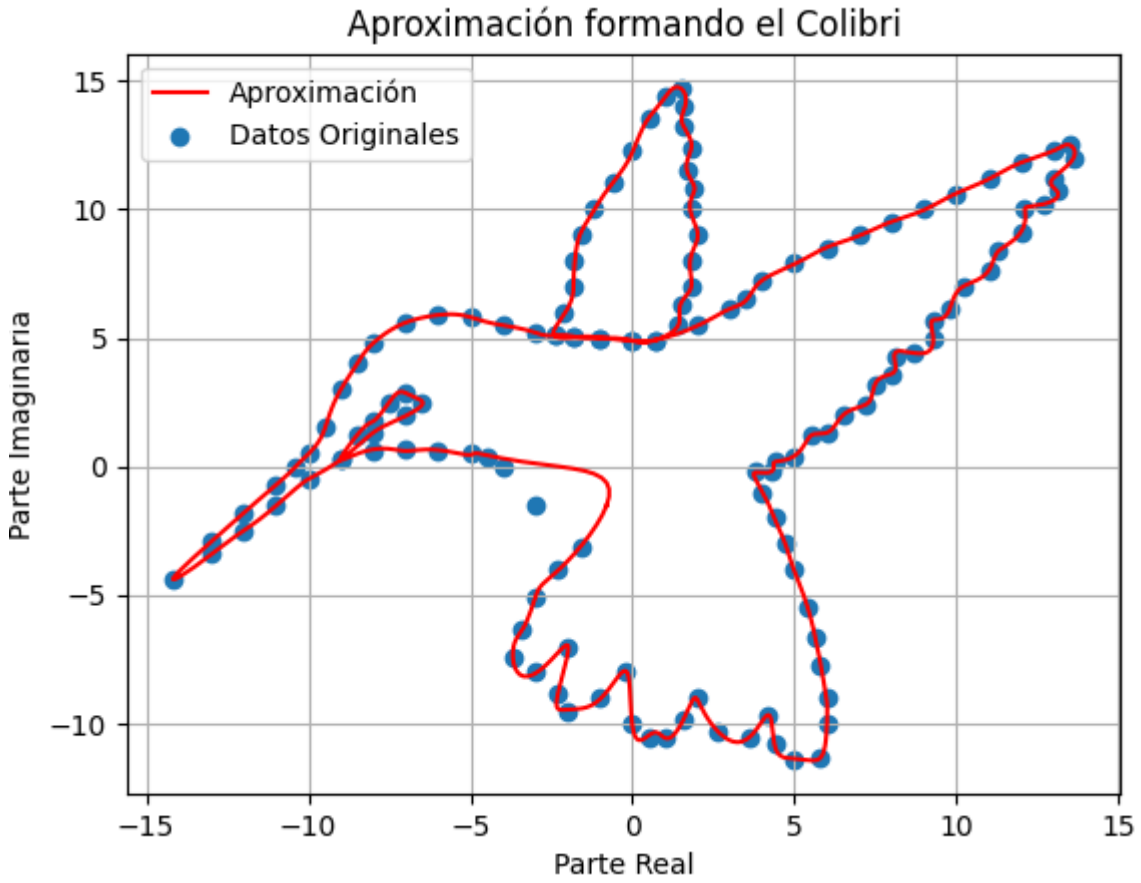


3)

La aproximación a simple vista sugiere que la trayectoria pasa a través de los puntos, como se evidencia en las imágenes; no obstante, al hacer zoom, se revela que en realidad no pasa directamente, sino que se acerca bastante.

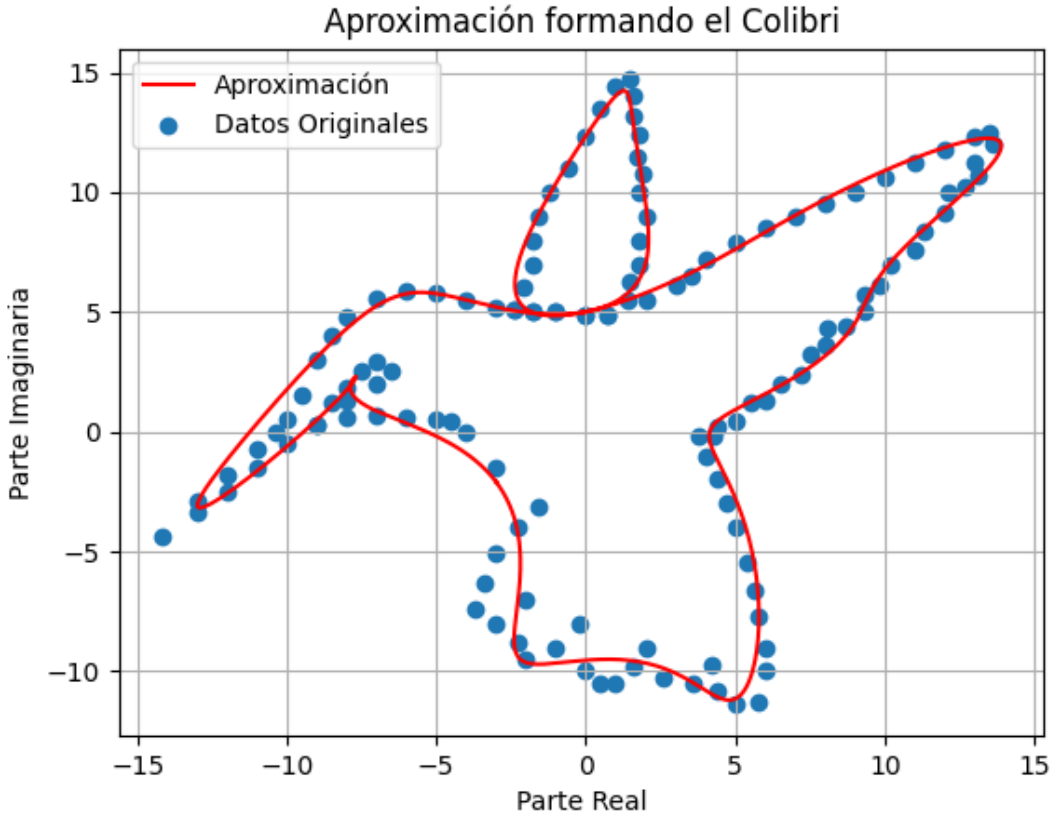


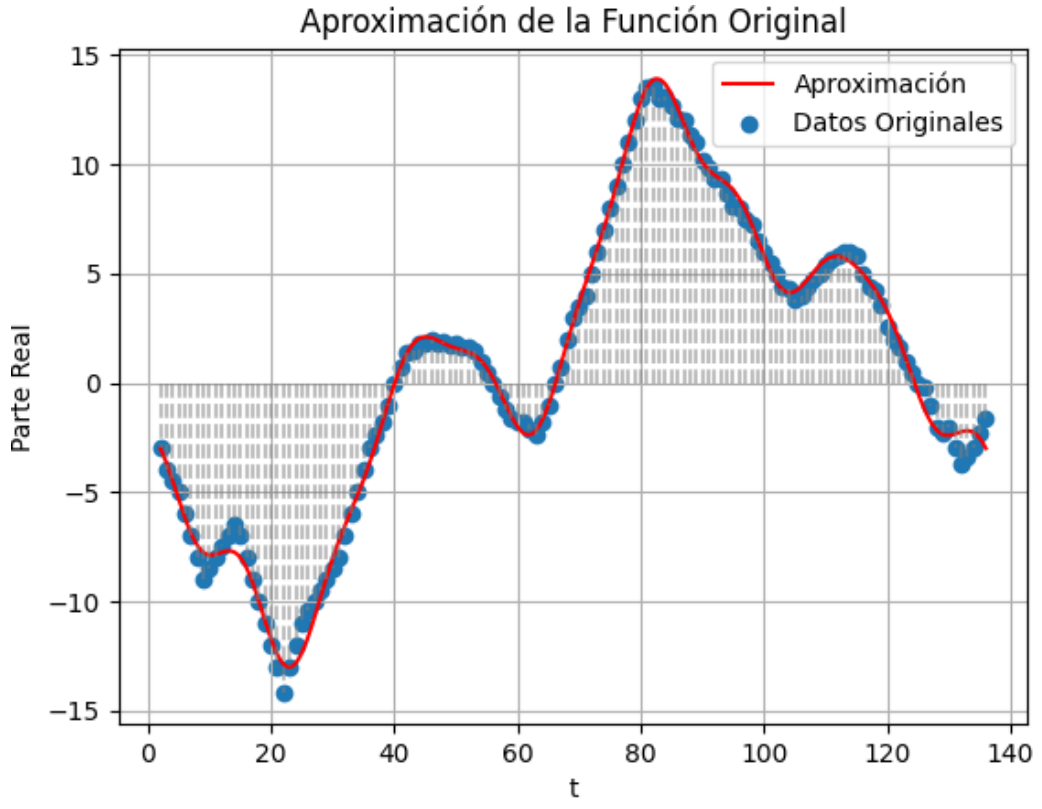

Aproximación de la Función Original
Aproximación
Datos Originales
Parte Real
t
5
0
−5
−10
40
45
50
55
60


Aproximación formando el Colibri
Aproximación
Datos Originales
Parte Imaginaria
Parte Real
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15

Si usas un valor de M menor que 68, la aproximación tendrá menos términos de la serie de Fourier, lo que significa que no capturará todos los detalles finos de la función original. Esto puede resultar en una aproximación más burda y menos precisa.

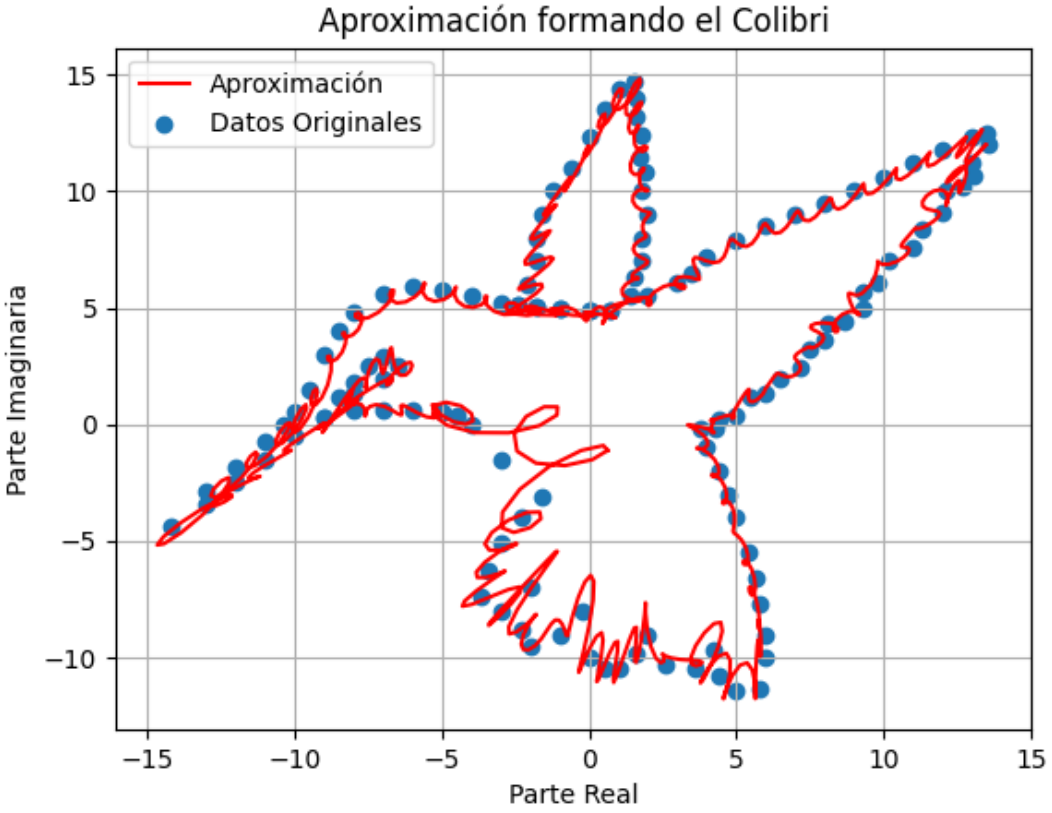
Aproximación con M =10:

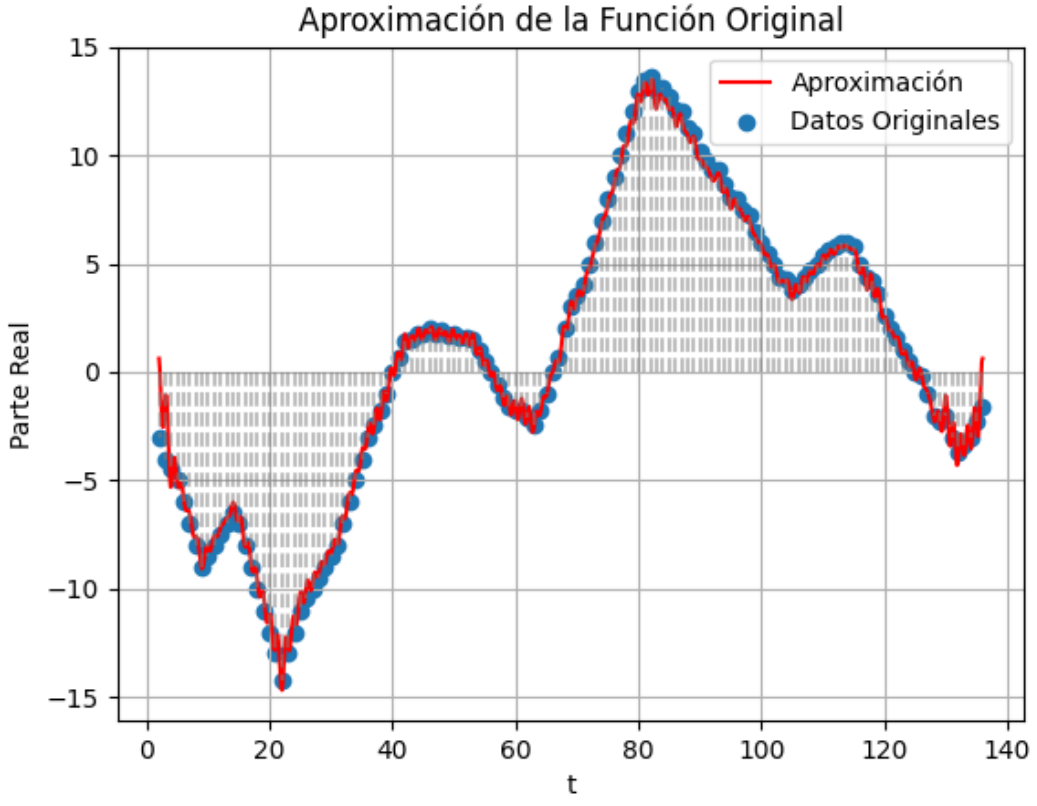




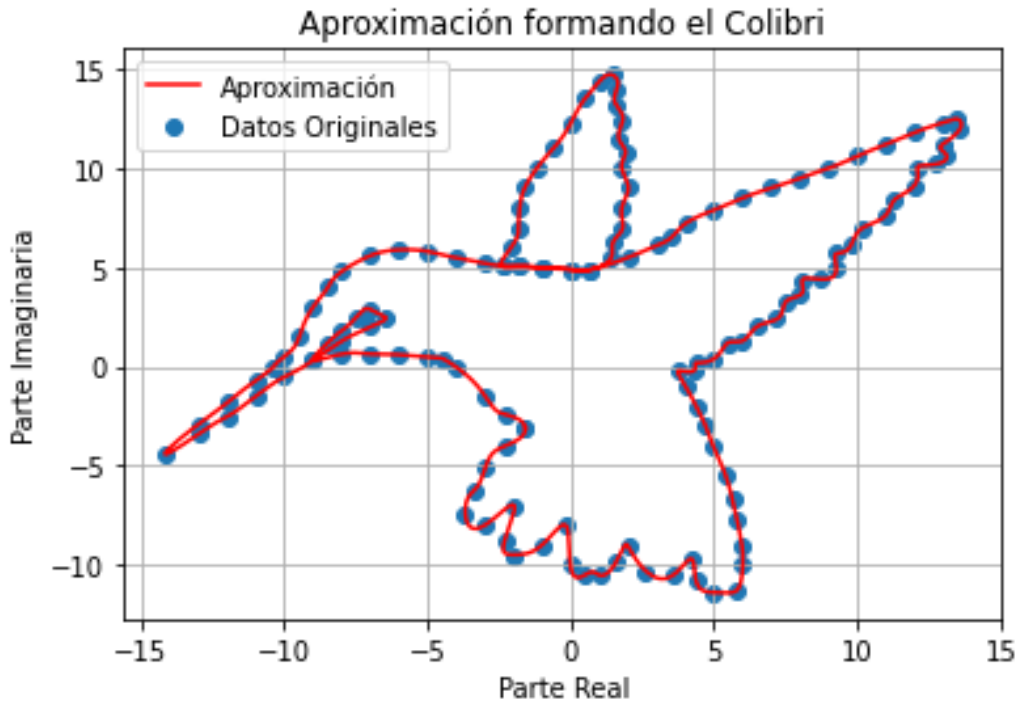
Por otro lado, si usas un valor de M mayor que 68, estarás considerando más términos de la serie de Fourier, lo que podría llevar a una aproximación más precisa y detallada. Sin embargo, un valor de M demasiado grande podría llevar a un aumento en el ruido y la variabilidad en la aproximación.

Aproximación con M =120





Se implementó luego un cambio en la fórmula para calcular las Z_k que resulta en una alternativa que utiliza todos los valores de la señal, ya que en la primer solución se estaban excluyendo los primeros valores y esto se traducía en una aproximación no tan precisa en una de las áreas del colibrí.
Con esta actualización del código las gráficas de la solución se ven de la siguiente manera:
Con M = 68



Referencias bibliográficas

- https://bestiariotopologico.blogspot.com/2018/12/dibujando-curvas-con-epiciclos.html
- https://www.youtube.com/watch?v=r6sGWTCMz2k&t=1026s&ab_channel=3Blue1Brown