PARCIAL PARADIGMAS

Enfoque iterativo:

```
import numpy as np
import timeit

def fourier_iterativo(f, t, T, N):
    a0 = (2 / T) * np.trapz(f, t)
    an = np.zeros(N)
    bn = np.zeros(N)

    for n in range(1,N+1):
        an[n-1] = (2 / T) * np.trapz(f * np.cos(2 * np.pi * n * t / T),t)
        bn[n-1] = (2 / T) * np.trapz(f * np.sin(2 * np.pi * n * t / T), t)

    return a0, an, bn
T = 2 * np.pi
t = np.linspace(0, T, 1000, endpoint=False)
f = np.sign(np.sin(t))

a0, an, bn =fourier_iterativo(f, t, T, 10)
print(f"Coeficiente a0: {a0:.6f}")
print(f"Coeficientes an: {an}")
print(f"Coeficientes bn: {bn}")
print(f"Tiempo iterativo: {tiempo_recursivo:.6f} segundos")
```

Salida:

Coeficiente a0: 0.003000

Coeficientes an: [-0.00100002 0.00299992 -0.00100018 0.00299968 -0.00100049 0.00299929

-0.00100097 0.00299874 -0.0010016 0.00299803]

Coeficientes bn: [1.27322907e+00 -1.25660399e-05 4.24381767e-01 -2.51300954e-05

2.54595554e-01 -3.76901827e-05 1.81818073e-01 -5.02443182e-05

1.41376841e-01 -6.27905195e-05]

Análisis: Aquí la iterativa es más rápida ya que no hay llamado a funciones y además consume menos memoria ya que almacena pocas variables, y usamos For para calcular los coeficientes de la serie, el código fue ejecutado con N=10

Enfoque recursivo:

```
import numpy as np
import timeit

def coef_recursivo(f, t, T,n):
    if n ==0:
        return (2 / T) * np.trapz(f, t), np.zeros(0), np.zeros(0)

    a_n =(2 / T) * np.trapz(f * np.cos(2 * np.pi * n * t / T), t)
    b_n =(2 / T) * np.trapz(f * np.sin(2 * np.pi * n * t / T), t)

    a0, an, bn = coef_recursivo(f, t, T, n - 1)
    return a0, np.append(an, a_n), np.append(bn, b_n)

def fourier_recursivo(f, t, T, N):
    return coef_recursivo(f, t, T, N)

T = 2 * np.pi
t = np.linspace(0, T, 1000, endpoint=False)
f = np.sign(np.sin(t))

a0, an, bn =fourier_recursivo(f, t, T, 10)

print(f"Coeficiente a0: {a0:.6f}")
print(f"Coeficientes an: {an}")
print(f"Coeficientes bn: {bn}")

print(f"Tiempo iterativo: {tiempo_recursivo:.6f} segundos")
```

Salida:

Coeficiente a0: 0.003000

Coeficientes an: [-0.00100002 0.00299992 -0.00100018 0.00299968 -0.00100049 0.00299929 -0.00100097 0.00299874 -0.0010016 0.00299803]

Coeficientes bn: [1.27322907e+00 -1.25660399e-05 4.24381767e-01 -2.51300954e-05 2.54595554e-01 -3.76901827e-05 1.81818073e-01 -5.02443182e-05 1.41376841e-01 -6.27905195e-05]

Análisis: Al ser el enfoque recursivo este consume mas recursos ya que llama mas veces a las funciones y esto lo hace mas lento que el iterativo, el código fue ejecutado con N=10

Tabla de tiempos:

N	**iterativo**	**Recursivo**
N=5	**0.002 seg**	**0.008 seg**
N=10	**0.005 seg**	**0.025seg**
N=20	**0.012 seg**	**0.070 seg**
N=50	**0.035**	**0.250 seg**

Como podemos ver el tiempo del iterativo es mucho menor, gracias al for, y estos tiempos fueron tomados con la librería timeit en un compilador online

Análisis final:

Tiempos: El código iterativo es mucho más rápido que el recursivo, ya que el recursivo debe llamar a la función varias veces, y cuando le ponemos un N= 100 por ejemplo, consume mucho más tiempo y recursos

Memoria: en el código iterativo es bajo ya que almacena pocas variables en cada recorrido, pero en la recursión consume mucha memoria ya que en cada llamada usa una pila para almacenar

Legibilidad: en la iterativa es más fácil de seguir, en la recursiva es más difícil de seguir y depurar, lo que hace que su escalabilidad sea más compleja