ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA COMPUTACION

ASIGNATURA: ICCD412 Métodos Numéricos **GRUPO:** GR2CC
TIPO DE INSTRUMENTO: Deber
FECHA DE ENTREGA LÍMITE: 04/05/2025
ALUMNO: Contreras Carrión Anthony Alexander

Tipo de Errores

OBJETIVOS

- Aplicar el método de bisección para encontrar soluciones numéricas precisas de ecuaciones no lineales. Visualizar las funciones involucradas para comprender mejor el comportamiento de las raíces dentro de intervalos dados.

DESARROLLO

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

def biseccion(f, a, b, tol, iter):
    i = 1
    FA = f(a)

    while i <= iter:
        p = a + (b - a) / 2
        FP = f(p)

        # Paso 4: Verificación de éxito
        if FP == 0 or (b - a) / 2 < tol:
            print(f"Procedimiento completado exitosamente en la iteración {i}.")
            return p

        i += 1

        # Paso 6: Determinar nuevo intervalo
        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    # Paso 7: Si no converge
    print(f"El método fracasó después de {iter} iteraciones, N0 = {iter}")
    return None
```

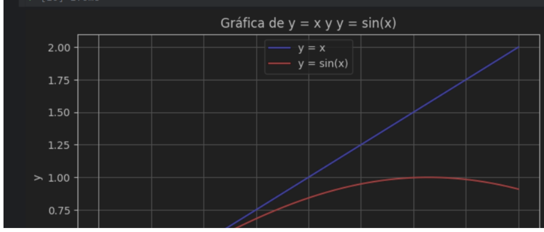
```python
def f(x):
    return x ** 3 - 7* x ** 2 + 14* x - 6

print(biseccion(f,0,1, 1e-2, 100))
print(biseccion(f,1,3.2, 1e-2, 100))
print(biseccion(f, 3.2,4, 1e-2, 100))
```

✓ [9] < 10 ms

```
Procedimiento completado exitosamente en la iteración 7.
0.5859375
Procedimiento completado exitosamente en la iteración 8.
3.0023437500000005
Procedimiento completado exitosamente en la iteración 7.
3.41875
```

```python
y1 = x
y2 = np.sin(x)

# Graficar
plt.figure(figsize=(8, 5))
plt.plot(x, y1, label="y = x", color='blue')
plt.plot(x, y2, label="y = sin(x)", color='red')
plt.title("Gráfica de y = x y y = sin(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.show()
```

[10] 196ms



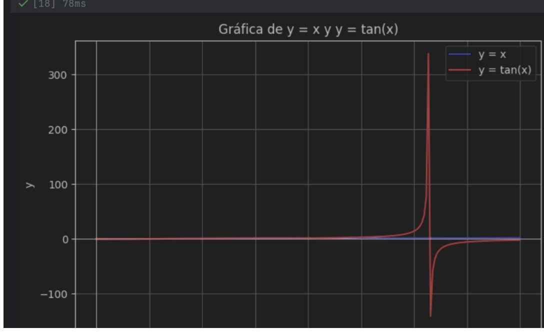

```python
def f(x):
    return x - 2 * np.sin(x)

raiz = biseccion(f, 1, 2, 1e-5, 100)

if raiz is not None:
    print(f"La raíz aproximada es: {raiz}")
```

[11] < 10 ms

```
Procedimiento completado exitosamente en la iteración 17.
La raíz aproximada es: 1.8955001831054688
```

```python
plt.plot(x, y1, label="y = x", color='blue')
plt.plot(x, y2, label="y = tan(x)", color='red')
plt.title("Gráfica de y = x y y = tan(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.show()
```

[18] 78ms




```python
def f(x):
    return x - 2 * np.tan(x)

raiz = biseccion(f, 1, 2, 1e-5, 100)

if raiz is not None:
    print(f"La raíz aproximada es: {raiz}")
```

[16] < 10 ms

```
Procedimiento completado exitosamente en la iteración 17.
La raíz aproximada es: 1.5707931518554688
```

```python
plt.plot(x, y1, label="y = x² - 1", color='blue')
plt.plot(x, y2, label="y = e^(1 - x²)", color='red')
plt.title("Gráfica de y = x² - 1 y y = e^(1 - x²)")
plt.xlabel("x")
plt.ylabel("y")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True)
plt.legend()
plt.show()
```

[19] 74ms


```python
def f(x):
    return x**2 - 1 - np.exp(1 - x**2)

raiz = biseccion(f, -2, 0, 1e-3, 100)

if raiz is not None:
    print(f"La raíz aproximada es: {raiz}")
```

[20] 14ms

```
Procedimiento completado exitosamente en la iteración 11.
La raíz aproximada es: -1.2509765625
```

```python
def f(x):
    return (x + 2) * (x + 1)**2 * x * (x - 1)**3 * (x - 2)

intervalos = {
    "a": (-1.5, 2.5),
    "b": (-0.5, 2.4),
    "c": (-0.5, 3),
    "d": (-3, -0.5)
}


for nombre, (a, b) in intervalos.items():
    raiz = biseccion(f, a, b,1e-5,100)
    if raiz is not None:
        print(f"Intervalo {nombre}: converge a raíz ≈ {raiz:.5f}")
```

[22] 10ms

```
Procedimiento completado exitosamente en la iteración 3.
Intervalo a: converge a raíz ≈ 0.00000
Procedimiento completado exitosamente en la iteración 19.
```

CONCLUSIONES

El método de bisección demostró ser una herramienta confiable y sencilla para encontrar raíces reales de funciones continuas. Su efectividad depende de identificar correctamente los intervalos donde ocurre un cambio de signo, garantizando así la convergencia hacia una solución con la precisión deseada.