Guía de laboratorio 5: Resolución de Sistemas de Ecuaciones No Lineales (Método bisección y secante).

1. Introducción

En el análisis numérico, la resolución de ecuaciones no lineales constituye un problema fundamental con aplicaciones en física, ingeniería, economía y ciencias computacionales. Dado que muchas funciones no pueden resolverse de manera analítica, se requieren métodos iterativos para aproximar sus raíces.

Entre los más importantes se encuentran el *Método de la Bisección*, que garantiza convergencia al aprovechar el *Teorema del Valor Intermedio en intervalos* donde ocurre un cambio de signo, y el *Método de la Secante*, que, aunque no siempre asegura convergencia, suele ser más rápido al aproximar la pendiente de la función sin necesidad de derivadas.

En esta práctica de laboratorio se abordará la resolución de diversas funciones no lineales de *una sola variable* utilizando ambos métodos, con el fin de comparar su eficiencia, precisión y número de iteraciones necesarias. Además, se implementarán en Python para observar la evolución de las aproximaciones y analizar los resultados obtenidos.

2. Objetivos

2.1 Objetivo general

- Aplicar los métodos numéricos de Bisección y Secante para resolver ecuaciones no lineales de una variable, implementando las soluciones mediante programación en Python.

2.2 Objetivos específicos

- Comprender los fundamentos teóricos de los métodos de Bisección y Secante.
- Aplicar paso a paso cada método en la resolución de funciones no lineales seleccionadas.
- Implementar algoritmos en Python que permitan automatizar los cálculos iterativos.
- Comparar el número de iteraciones, velocidad de convergencia y precisión alcanzada por ambos métodos.
- Analizar los resultados obtenidos para identificar ventajas, limitaciones y posibles aplicaciones de cada técnica.

3. Materiales y Recursos

- Computadora personal o laboratorio de cómputo.
- Python 3.10+ instalado (preferible en Anaconda).
- Librería *Math* instalada (pip install *nombre librería*).
- Editor de código (Sublime Text, Visual Studio Code o Jupyter Notebook).
- Opcional (crear un ambiente virtual)

4. Fundamentación Teórica

La resolución de ecuaciones no lineales de la forma:

$$f(x) = 0$$

Es un problema central en el análisis numérico y en aplicaciones de la ingeniería, la física y la economía. Como muchas funciones no admiten soluciones exactas, se utilizan *métodos iterativos* que generan aproximaciones sucesivas a la raíz buscada.

En esta práctica se abordará dos de los métodos más empleados: el *Método de la Bisección* y el *Método de la Secante*.

Método de la Bisección

El método de la bisección se fundamenta en el *Teorema del Valor Intermedio*, el cual establece que si $f(x)$ es continua en el intervalo [a, b] y

$$f(a) \cdot f(b) < 0,$$

Entones existe al menos una raíz en (a, b)

El procedimiento consiste en:

1. Calcular el punto medio del intervalo:

$$r_k = \frac{a_k + b_k}{2}$$

2. Evaluar la función en el punto medio:

$$f(r_k)$$

3. Seleccionar el subintervalo donde la función cambia de signo y repetir el proceso:
   - Si $f(a_k) \cdot f(r_k) < 0 \Rightarrow b_{k+1} = r_k$
   - Si $f(r_k) \cdot f(b_k) < 0 \Rightarrow a_{k+1} = r_k$

El error en cada iteración se reduce aproximadamente a la mitad:

$$|e_k| \leq \frac{b-a}{2^k}$$

Este método es **lento pero seguro**, ya que garantiza convergencia siempre que se cumpla la condición de cambio de signo.

Método de la Secante

El método de la secante es una aproximación al *Método de Newton-Raphson*, pero sin necesidad de calcular la derivada. En lugar de usar la tangente, se emplea una recta secante que pasa por los puntos $(x_{k-1}, f(x_{k-1}))$ y $(x_k, f(x_k))$.

La fórmula iterativa es:

$$x_{k+1} = x_k - f(x_k) \cdot \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

El proceso requiere dos valores iniciales x_0 y x_1.

A diferencia de la bisección, el método de la secante ***no garantiza convergencia***, pero cuando converge lo hace más rápidamente, con un orden aproximado de convergencia de **1.618...** (número áureo), mayor que el orden lineal del método de bisección.

4. Ejercicios y Desarrollo

Se plantearon seis ejercicios correspondientes a funciones no lineales de una sola variable, cada bloque de código Python, se aplica bisección y secante, se imprime una tabla de iteraciones (las primeras y últimas filas) y la raíz final. Además, se proponen 4 ejercicios para su codificación, finalmente se establece las conclusiones.

Ejercicio 1 — $f(x) = x^3 - x - 2$ en $[1, 2]$

a. **Realizar Pseudocódigo**
b. **Codificar en Python**

```python
# Ejercicio 1: x^3 - x - 2
import math

def f(x): return x**3 - x - 2

def bisection_iterations(f,a,b,tol=1e-8,maxit=100):
    fa, fb = f(a), f(b)
    if fa*fb>0: raise ValueError("No cambio de signo en [a,b]")
    rows=[]
    for k in range(1,maxit+1):
        r=(a+b)/2.0
        fr=f(r)
        rows.append((k,a,b,r,fr))
        if abs(fr)<tol or (b-a)/2.0 < tol: break
        if fa*fr < 0:
            b, fb = r, fr
        else:
            a, fa = r, fr
    return rows

def secant_iterations(f,x0,x1,tol=1e-8,maxit=100):
    rows=[]
    for k in range(1,maxit+1):
        fx0,fx1 = f(x0), f(x1)
        if fx1==fx0: raise ZeroDivisionError("Denominador cero")
        x2 = x1 - fx1*(x1-x0)/(fx1-fx0)
        fx2 = f(x2)
        rows.append((k,x0,x1,x2,fx1,fx2))
        if abs(x2-x1)<tol or abs(fx2)<tol: break
        x0,x1 = x1,x2
    return rows

a,b = 1.0, 2.0
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)
```

```
print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

Ejercicio 2 — $f(x) = \cos x - x$ en $[0, 1]$

```
# Ejercicio 2: cos(x) - x
import math
def f(x): return math.cos(x) - x

# (Puede reutilizar las mismas funciones bisection_iterations y
secant_iterations del bloque anterior)

a,b = 0.0, 1.0
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)

print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

Ejercicio 3 — $f(x) = e^x - 3x$ en $[0, 1]$

```
# Ejercicio 3: exp(x) - 3x
import math
def f(x): return math.exp(x) - 3*x

a,b = 0.0, 1.0
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)

print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
```

```
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

Ejercicio 4 — $f(x) = x \sin x - 1$ en $[1, 2]$

```
import math
def f(x): return x*math.sin(x) - 1

a,b = 1.0, 2.0
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)

print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

Ejercicio 5 — $f(x) = \ln(x + 1) + x^2 - 3$ en $[1, 2]$

```
# Ejercicio 5: ln(x+1) + x^2 - 3
import math
def f(x): return math.log(x+1) + x**2 - 3

a,b = 1.0, 2.0
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)

print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

Ejercicio 6 — $f(x) = \tan x - x$ en $[4.4, 4.6]$

```
# Ejercicio 6: tan(x) - x
import math
def f(x): return math.tan(x) - x

a,b = 4.4, 4.6
# NOTA: tan(x) tiene asintotas; cuidado con el intervalo: aquí [4.4,4.6] evita la asintota cercana (~3π/2=4.712...)
bis = bisection_iterations(f,a,b)
sec = secant_iterations(f,a,b)

print("Bisección (primeras 6 iter / última):")
for row in bis[:6]: print(row)
print("...", bis[-1])
print("\nSecante (todas iter mostradas):")
for row in sec: print(row)
print("\nResultado final:")
print("Bisección raíz ≈", bis[-1][3], "iter=", bis[-1][0])
print("Secante   raíz ≈", sec[-1][3], "iter=", sec[-1][0])
```

NOTA: Complete la codificación en los ejercicios 7, 8, 9 y 10

Ejercicio 7

$$f(x) = x^3 - 4x + 1 = 0$$

- **Bisección:** usar el intervalo $[0, 1]$.
- **Secante:** usar $x_0 = 0$, $x_1 = 1$.

Ejercicio 8

$$f(x) = e^{-x} - x = 0$$

- **Bisección:** usar el intervalo $[0, 1]$.
- **Secante:** usar $x_0 = 0.5$, $x_1 = 1$.

Ejercicio 9

$$f(x) = \cos(x) - x^2 = 0$$

- **Bisección:** usar el intervalo $[0, 1]$.
- **Secante:** usar $x_0 = 0.4$, $x_1 = 0.6$.

Ejercicio 10

$$f(x) = \ln(x+2) - x = 0$$

- **Bisección:** usar el intervalo $[0, 2]$.
- **Secante:** usar $x_0 = 0.5$, $x_1 = 2$.

5. Conclusiones (Máximo en 10 líneas)