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

1. Ejercicios y Desarrollo

Se plantearon 10 ecuaciones no lineales con una sola variable. Utilizar la librería *math* en *Python*. Se aplica bisección y secante para la solución de cada una de ellas, se proponen 4 ejercicios para su codificación, finalmente se establece las conclusiones

2. Sistemas de Ecuaciones No lineales de una variable

Ejercicio 1

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

Ejercicio 2

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

Ejercicio 3

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

Ejercicio 4

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

Ejercicio 5

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

Ejercicio 6

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

Ejercicio 7

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

Ejercicio 8

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

Ejercicio 9

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

Ejercicio 10

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

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

Funciones principales

```
Función Bisección(f, a, b, tol, maxit):
    fa ← f(a)
    fb ← f(b)
    Si fa * fb > 0:
      Error "No hay cambio de signo en [a,b]"

    lista ← []

    Para k desde 1 hasta maxit:
      r ← (a + b) / 2
      fr ← f(r)
      Agregar (k, a, b, r, fr) a lista

      Si |fr| < tol O (b - a)/2 < tol:
        Salir del bucle

      Si fa * fr < 0:
        b ← r
        fb ← fr
      Sino:
        a ← r
        fa ← fr

    Retornar lista
```

```python
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        raise ValueError("No hay 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
```

```
Función Secante(f, x0, x1, tol, maxit):
    lista ← []

    Para k desde 1 hasta maxit:
      fx0 ← f(x0)
      fx1 ← f(x1)

      Si fx1 = fx0:
        Error "Denominador cero"

      x2 ← x1 - fx1 * (x1 - x0) / (fx1 - fx0)
      fx2 ← f(x2)

      Agregar (k, x0, x1, x2, fx1, fx2) a lista

      Si |x2 - x1| < tol O |fx2| < tol:
        Salir del bucle

      x0 ← x1
      x1 ← x2

    Retornar lista
```

```python
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
```

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

```
Inicio
  Definir f(x) = x^3 - x - 2

  # Intervalo inicial
  a ← 1.0
  b ← 2.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
FinAlgoritmo
```

```python
import math

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

# Método de Bisección
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

# Método de la Secante
def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial
a, b = 1.0, 2.0

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)


# Mostrar resultados
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 $[0, 1]$

```
Inicio
  Definir f(x) = cos(x) - x

  # Intervalo inicial
  a ← 0.0
  b ← 1.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Fin
```

```python
import math

def f(x):
    return math.cos(x) - x

# Método de Bisección
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

# Método de la Secante
def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial
a, b = 0.0, 1.0

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)

# Mostrar resultados
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 $[0, 1]$

```
Inicio
  Definir f(x) = exp(x) - 3*x

  # Intervalo inicial
  a ← 0.0
  b ← 1.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

def f(x):
    return math.exp(x) - 3*x

# Método de Bisección
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

# Método de la Secante
def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial
a, b = 0.0, 1.0

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)

# Mostrar resultados
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 $[1, 2]$

```
Inicio
  Definir f(x) = x * sen(x) - 1

  # Intervalo inicial
  a ← 1.0
  b ← 2.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

# Definición de la función
def f(x):
    return x * math.sin(x) - 1

# Método de Bisección
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

# Método de la Secante
def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial
a, b = 1.0, 2.0

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)

# Mostrar resultados
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 $[1, 2]$

```
Inicio
  Definir f(x) = ln(x+1) + x^2 - 3

  # Intervalo inicial
  a ← 1.0
  b ← 2.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

# Definición de la función
def f(x):
    return math.log(x+1) + x**2 - 3

def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial
a, b = 1.0, 2.0

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)

# Mostrar resultados
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 $[4.4, 4.6]$

```
Inicio
  Definir f(x) = tan(x) - x

  # Intervalo inicial (evitar la asíntota en x = 3π/2 ≈ 4.712...)
  a ← 4.4
  b ← 4.6

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, a, b, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

# Definición de la función
def f(x):
    return math.tan(x) - x

# Método de Bisección
def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

# Método de la Secante
def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

# Intervalo inicial (evitar la asíntota en x = 3π/2 ≈ 4.712...)
a, b = 4.4, 4.6

# Ejecutar métodos
bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, a, b)

# Mostrar resultados
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 7: $f(x) = e^{-x} - x = 0$

```
Inicio
  Definir f(x) = e^(-x) - x

  # Intervalo inicial para Bisección
  a ← 0.0
  b ← 1.0

  # Valores iniciales para Secante
  x0 ← 0.5
  x1 ← 1.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, x0, x1, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

def f(x):
    return math.exp(-x) - x

def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

a, b = 0.0, 1.0
x0, x1 = 0.5, 1.0

bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, x0, x1)

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

```
Inicio
  Definir f(x) = x^3 - 4x + 1

  # Intervalo inicial para Bisección
  a ← 0.0
  b ← 1.0

  # Valores iniciales para Secante
  x0 ← 0.0
  x1 ← 1.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, x0, x1, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

def f(x):
    return x**3 - 4*x + 1

def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

a, b = 0.0, 1.0
x0, x1 = 0.0, 1.0

bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, x0, x1)

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

```
Inicio
  Definir f(x) = cos(x) - x^2

  # Intervalo inicial para Bisección
  a ← 0.0
  b ← 1.0

  # Valores iniciales para Secante
  x0 ← 0.4
  x1 ← 0.6

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, x0, x1, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

def f(x):
    return math.cos(x) - x**2

def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

a, b = 0.0, 1.0
x0, x1 = 0.4, 0.6

bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, x0, x1)

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

```
Inicio
  Definir f(x) = ln(x+2) - x

  # Intervalo inicial para Bisección
  a ← 0.0
  b ← 2.0

  # Valores iniciales para Secante
  x0 ← 0.5
  x1 ← 2.0

  # Ejecutar métodos numéricos
  bis ← Bisección(f, a, b, tol=1e-8, maxit=100)
  sec ← Secante(f, x0, x1, tol=1e-8, maxit=100)

  # Mostrar resultados de Bisección
  Imprimir "Bisección (primeras 6 iter / última):"
  Para cada fila en las primeras 6 de bis:
    Imprimir fila
  Imprimir "...", última fila de bis

  # Mostrar resultados de Secante
  Imprimir "Secante (todas iter mostradas):"
  Para cada fila en sec:
    Imprimir fila

  # Resultados finales
  Imprimir "Resultado final:"
  Imprimir "Bisección raíz ≈", última fila de bis.r, "iter =", última fila de bis.k
  Imprimir "Secante raíz ≈", última fila de sec.r, "iter =", última fila de sec.k
Final
```

```python
import math

def f(x):
    return math.log(x + 2) - x

def bisection_iterations(f, a, b, tol=1e-8, maxit=100):···

def secant_iterations(f, x0, x1, tol=1e-8, maxit=100):···

a, b = 0.0, 2.0
x0, x1 = 0.5, 2.0

bis = bisection_iterations(f, a, b)
sec = secant_iterations(f, x0, x1)

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])
```

3. Conclusiones (Máximo en 10 líneas)

En esta práctica se comprobó la utilidad de los métodos de Bisección y Secante para resolver ecuaciones no lineales de una variable. El método de bisección mostró ser siempre confiable, ya que garantiza convergencia si existe un cambio de signo en el intervalo, aunque su velocidad es lenta. Por el otro lado, el método de la secante, al no requerir derivadas, presentó una convergencia más rápida en la mayoría de los ejercicios, pero con menor estabilidad y riesgo de divergencia si no se eligen adecuadamente los valores iniciales. La comparación de ambos métodos evidenció que la bisección es preferible cuando se busca seguridad en la aproximación, mientras que la secante es más eficiente si se requieren resultados rápidos y se cuenta con buenas estimaciones iniciales. En general, ambos métodos complementan sus fortalezas y limitaciones, siendo herramientas muy importantes en el análisis numérico y en aplicaciones prácticas de la ingeniería.