Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Torres Cruz Fred

Alumno:

Ticona Miramira Roberto Angel

Trabajo 6 - Método de la Secante

» Descripción

El método de la secante es un procedimiento iterativo utilizado para encontrar aproximaciones sucesivas de las raíces reales de una función continua $f(x)$. A diferencia del método de Newton-Raphson, no requiere el cálculo de la derivada analítica, ya que la pendiente se estima numéricamente mediante dos puntos cercanos a la raíz.

La idea fundamental consiste en reemplazar la tangente del método de Newton por una **secante**, es decir, la recta que pasa por los puntos $(x_{n-1}, f(x_{n-1}))$ y $(x_n, f(x_n))$. La intersección de esta recta con el eje x proporciona la siguiente aproximación de la raíz, dada por la fórmula:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Este método combina la simplicidad de la bisección con una velocidad de convergencia cercana a la del método de Newton-Raphson, siempre que las condiciones iniciales sean adecuadas.

Importancia de la visualización previa

Antes de aplicar el método de la secante, es recomendable graficar la función $f(x)$ en un intervalo apropiado. Esta visualización permite identificar los puntos donde la función cruza el eje x, y seleccionar dos valores iniciales x_0 y x_1 que estén cercanos a una raíz. Una elección adecuada de estos valores iniciales incrementa la probabilidad de convergencia.

Ventajas y limitaciones

- No requiere conocer la derivada analítica de la función.
- Puede converger más rápidamente que el método de bisección.
- Su velocidad de convergencia es **superlineal** (entre la bisección y Newton-Raphson).
- Si los puntos iniciales x_0 y x_1 no están bien elegidos, el método puede divergir.
- No garantiza convergencia si la función no es continua en el intervalo considerado.

» Aplicación del método en Python

Se desarrolló un programa en Python que permite al usuario ingresar una función $f(x)$ y graficarla en un intervalo definido. Esta etapa inicial facilita la identificación visual de posibles raíces y la selección de los valores iniciales x_0 y x_1.

Luego, el programa aplica el método de la secante iterativamente hasta alcanzar una tolerancia establecida o un número máximo de iteraciones. En cada iteración, se muestran los valores de

x_0, x_1, $f(x_0)$, $f(x_1)$, la nueva aproximación x_2 y el error absoluto. Finalmente, se imprime la raíz aproximada encontrada y el número de iteraciones necesarias.

Este procedimiento combina la exploración gráfica con el razonamiento numérico, fomentando una comprensión más completa de la convergencia y del comportamiento de la función.

» Entrada

Una cadena de texto que representa una función matemática.

$$f(x) = x^3 - x - 1$$

» Salida

- Gráfica para visualizar la función y elegir los puntos iniciales.
- Tabla con los valores de cada iteración.
- Raíz aproximada y número de iteraciones necesarias.

» Restricciones

- $f(x)$ debe ser continua en el intervalo analizado.
- Los valores iniciales x_0 y x_1 deben ser distintos y cercanos a la raíz.

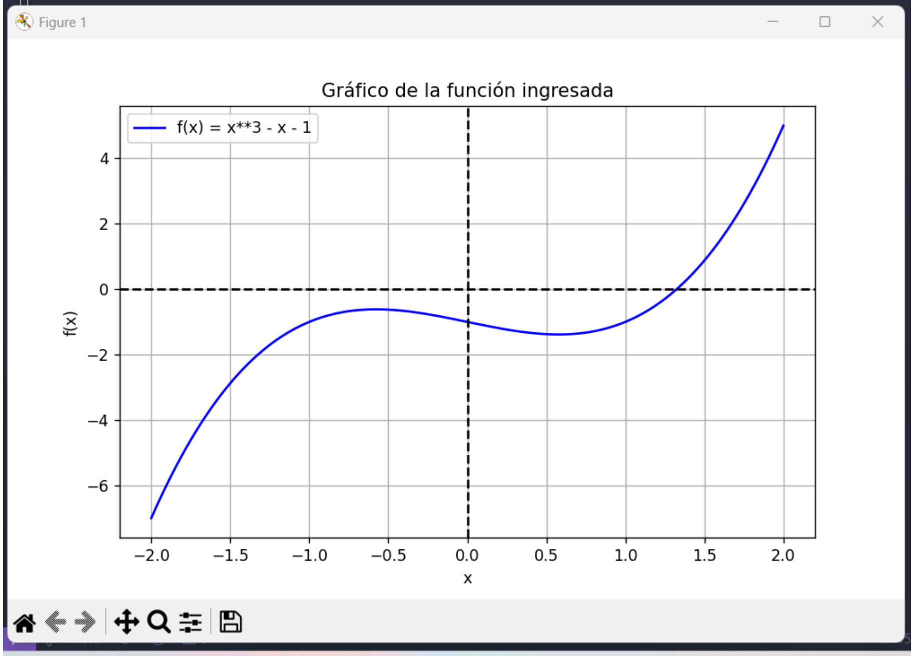
Código

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

# === 1. Ingreso de la función ===
func_str = input("Ingrese la función f(x): ")  # Ejemplo: x**3 - x - 1

# Definimos la función
def f(x):
    return eval(func_str, {"np": np, "x": x})

# === 2. Graficar la función ===
xmin = float(input("Ingrese el valor mínimo de x: "))
xmax = float(input("Ingrese el valor máximo de x: "))

x = np.linspace(xmin, xmax, 400)
y = f(x)

plt.figure(figsize=(8, 5))
plt.plot(x, y, label=f"f(x) = {func_str}", color='blue')
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.title("Gráfico de la función ingresada")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()

# === 3. Pregunta si desea aplicar el método de la secante ===
op = input("¿Desea encontrar una raíz con el método de la Secante? (s/n): ").lower()

if op == "s":
    # === 4. Ingreso de puntos iniciales ===
    x0 = float(input("Ingrese el primer valor inicial x0: "))
```

```python
    x1 = float(input("Ingrese el segundo valor inicial x1: "))

    # Parámetros del método
    tol = 1e-6
    max_iter = 100

    print("\nIteración |      x0       |       x1       |      f(x0)      |      f(x1)      |      x2       |    Error")
    print("----------------------------------------------------------------------")

    for i in range(1, max_iter + 1):
        f0 = f(x0)
        f1 = f(x1)

        if f1 - f0 == 0:
            print(f"\ n División por cero en la iteración {i}. El método no puede continuar.")
            break

        x2 = x1 - f1 * (x1 - x0) / (f1 - f0)
        error = abs(x2 - x1)

        print(f"{i:9d} | {x0:10.6f} | {x1:10.6f} | {f0:12.6f} | {f1:12.6f} | {x2:10.6f} | {error:10.6f}")

        if error < tol:
            print(f"\ n Raíz aproximada encontrada: {x2:.6f}")
            print(f"Iteraciones realizadas: {i}")
            break

        x0, x1 = x1, x2

    else:
        print("\ n No se alcanzó la convergencia después de", max_iter, "iteraciones.")
else:
    print("No se aplicó el método de la Secante.")
```

Ejecución

```
Ingrese la función f(x): x**3 - x - 1
Ingrese el valor mínimo de x: -2
Ingrese el valor máximo de x: 2
```




```
Ingrese la función f(x): x**3 - x - 1
Ingrese el valor mínimo de x: -2
Ingrese el valor máximo de x: 2
¿Desea encontrar una raíz con el método de la Secante? (s/n): s
Ingrese el primer valor inicial x0: 1
Ingrese el segundo valor inicial x1: 2

Iteración |     x0      |     x1      |    f(x0)    |    f(x1)    |     x2      |   Error
-------------------------------------------------------------------------------------------
        1 |   1.000000 |   2.000000 |   -1.000000 |    5.000000 |   1.166667 |   0.833333
        2 |   2.000000 |   1.166667 |    5.000000 |   -0.578704 |   1.253112 |   0.086445
        3 |   1.166667 |   1.253112 |   -0.578704 |   -0.285363 |   1.337206 |   0.084094
        4 |   1.253112 |   1.337206 |   -0.285363 |    0.053881 |   1.323850 |   0.013356
        5 |   1.337206 |   1.323850 |    0.053881 |   -0.003698 |   1.324708 |   0.000858
        6 |   1.323850 |   1.324708 |   -0.003698 |   -0.000043 |   1.324718 |   0.000010
        7 |   1.324708 |   1.324718 |   -0.000043 |    0.000000 |   1.324718 |   0.000000

✅ Raíz aproximada encontrada: 1.324718
Iteraciones realizadas: 7
PS E:\Universidad\Ing. Estadística e Informática\IV Semestre\Programación numérica\Tareas\Pruebas>
```