Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

E.P. de Ingeniería Estadística e Informática

Docente: Ing. Torres Cruz Fred

Presentado por: Quispe Ito Luz Leidy

Metodo de la Secante

1. ¿Que es el Metodo de la Secante?

El **método de la secante** es un método numérico utilizado para encontrar aproximaciones de la raíz de una ecuación no lineal $f(x) = 0$. Este método reemplaza la derivada de la función por una *pendiente aproximada*, obtenida a partir de dos puntos cercanos $(x_{n-1}, f(x_{n-1}))$ y $(x_n, f(x_n))$.

La recta que pasa por estos dos puntos es una *secante*, y su intersección con el eje x proporciona la siguiente aproximación de la raíz. A diferencia del método de Newton-Raphson, la secante no requiere calcular derivadas, lo que lo hace especialmente útil cuando la derivada es complicada o no existe.

2. Derivación del método

Aproximamos la derivada $f'(x_n)$ mediante:

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

Sustituyendo esta aproximación en la fórmula de Newton, obtenemos:

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

Análisis de Convergencia

- Orden de convergencia:

$$p = \frac{1 + \sqrt{5}}{2} \approx 1{,}618 \quad \text{(número áureo)}$$

- Error asintótico:

$$|e_{n+1}| \approx C|e_n|^{1,618}$$

- Requiere dos valores iniciales x_0 y x_1.

3. Objetivo del programa

El programa desarrollado en Python implementa el **método numérico de la secante**, cuyo propósito es aproximar una raíz real de una ecuación no lineal $f(x) = 0$. A diferencia del método de bisección, la secante no requiere un intervalo donde la función cambie de signo; en su lugar, utiliza dos valores iniciales x_1 y x_2 para construir una recta secante. La intersección de esta recta con el eje x genera una nueva aproximación a la raíz.

El método permite obtener soluciones con una precisión establecida por medio de una tolerancia de error definida previamente.

2. Funcionamiento general

El programa solicita al usuario los siguientes datos:

- La función $f(x)$ escrita en forma algebraica, por ejemplo: `x^3 - 2*x - 5`, `e^(0.5*x) - 3`, `sin(x) - x/2`.
- Dos valores iniciales x_1 y x_2, necesarios para iniciar el método.

El programa incorpora un preprocesador que convierte la expresión ingresada a un formato compatible con Python, permitiendo el uso de funciones como: `sin, cos, tan, log, exp, pi, e`.

4. Procedimiento del algoritmo

El funcionamiento del programa se divide en varias etapas:

1. Preparación y validación inicial

Se procesa la función ingresada, reemplazando operadores y añadiendo multiplicaciones omitidas (por ejemplo, $3x$ se convierte en $3 * x$). Posteriormente, se evalúan $f(x_1)$ y $f(x_2)$. Si ambos valores son inválidos o coinciden, el programa detiene la ejecución.

2. Cálculo iterativo del Método de la Secante

Mientras el error sea mayor que la tolerancia, el algoritmo ejecuta:

1. Evaluar los valores $f(x_1)$ y $f(x_2)$.
2. Calcular la nueva aproximación mediante:
$$x_r = x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}.$$
3. Calcular el error relativo:
$$E = |x_r - x_2|.$$
4. Registrar los valores e imprimir una tabla con: número de iteración, x_r, error, $f(x_r)$.
5. Actualizar los valores:
$$x_1 \leftarrow x_2, \qquad x_2 \leftarrow x_r.$$

El proceso finaliza cuando el error es menor que la tolerancia o se alcanza el número máximo de iteraciones.

3. Presentación de resultados

Al concluir, el programa muestra:

- La raíz aproximada.
- El valor $f(x_r)$.
- El número total de iteraciones.
- El error final.

Además, el programa genera una gráfica donde se visualizan:

- La función $f(x)$.
- El eje x.
- Los puntos obtenidos en cada iteración del método.

4. Restricciones del programa

Para que el método funcione correctamente, deben cumplirse las siguientes condiciones:

- La función debe ser válida y continua en la región analizada.
- Los valores $f(x_1)$ y $f(x_2)$ deben ser diferentes, ya que la fórmula usa:

$$f(x_2) - f(x_1),$$

  lo cual no puede ser cero.
- Los valores ingresados deben ser numéricos y compatibles con la sintaxis matemática.

El preprocesador del programa permite escribir expresiones de forma natural. Algunos ejemplos:

$$3x \to 3 * x, \qquad \mathrm{sen}(x) \to \sin(x), \qquad \ln(x) \to \log(x), \qquad e^x \to \exp(x).$$

5. Programa en Python

```
import math, re, matplotlib.pyplot as plt

print("=== M TODO DE LA SECANTE ===")

def preparar_funcion(expr):
    expr = expr.lower().strip()
    expr = expr.replace("^", "**").replace("sen", "sin").replace("ln", "log").replace("  ", "pi")
    expr = re.sub(r'(\d)([a-zA-Z\(])', r'\1*\2', expr)
    expr = re.sub(r'([a-zA-Z\)])(\d)', r'\1*\2', expr)
    return expr

funcion_str = input("Ingrese la funci n f(x): ")
funcion_str = preparar_funcion(funcion_str)

def f(x):
    try:
        return eval(funcion_str, {"__builtins__": None}, math.__dict__ | {"x": x})
    except:
        return None

x1 = float(input("Ingrese x1 = "))
x2 = float(input("Ingrese x2 = "))

tol, max_iter = 1e-10, 100
itera, error, xr = 0, 1, x1
```

```python
valores = []

print(f"\n{'Itera':<6} {'xr':<15} {'error':<15} {'f(xr)':<15}")
print("-" * 55)

while itera < max_iter and error > tol:
    fx1, fx2 = f(x1), f(x2)
    if fx1 is None or fx2 is None:
        break
    xr = x2 - fx2 * (x2 - x1) / (fx2 - fx1)
    fxr = f(xr)
    itera += 1
    error = abs(xr - x2)
    valores.append(xr)
    print(f"{itera:<6} {xr:<15.10f} {error:<15.10f} {fxr:<15.10f}")
    x1, x2 = x2, xr

print("\n--- RESULTADOS ---")
print(f"xr = {xr:.10f}")
print(f"f(xr) = {f(xr):.10f}")
print(f"error = {error:.10f}")
print(f"iteraciones = {itera}")

# --- GRAFICAR ---
xs = [x1 + i*(x2-x1)/200 for i in range(201)]
ys = [f(x) for x in xs]
plt.plot(xs, ys, label="f(x)")
plt.axhline(0, color='black')
plt.scatter(valores, [f(v) for v in valores], color='red', label='Aproximaciones')
plt.title("M todo de la Secante")
plt.legend()
plt.show()
```

```
=== MÉTODO DE LA SECANTE ===
Ingrese la función f(x): x^2-4
Ingrese x1 = 1
Ingrese x2 = 3

Itera  xr              error           f(xr)
-------------------------------------------------------
1      1.7500000000    1.2500000000    -0.9375000000
2      1.9473684211    0.1973684211    -0.2077562327
3      2.0035587189    0.0561902978    0.0142475399
4      1.9999525932    0.0036061257    -0.0001896251
5      1.9999999579    0.0000473647    -0.0000001686
6      2.0000000000    0.0000000421    0.0000000000
7      2.0000000000    0.0000000000    0.0000000000

--- RESULTADOS ---
xr = 2.0000000000
f(xr) = 0.0000000000
error = 0.0000000000
iteraciones = 7
```