INTERFAZ METODOS ABIERTOS FINAL

METODOS
NUMERICOS

INGENIERIA EN SISTEMAS COMPUTACIONALES

HORACIO IRAN SOLIS CISNEROS

03 de marzo del 2025

SEMESTRE: 4º GRUPO: B

Alumno:

Moguel González Ángel Adrián

2. RESUMEN

Este proyecto implementa métodos numéricos para la resolución de ecuaciones no lineales a través de una interfaz gráfica desarrollada en Tkinter. Se incluyen los métodos de Secante y Newton-Raphson, permitiendo la visualización de la convergencia mediante gráficos. La interfaz facilita la introducción de parámetros y valida las condiciones necesarias antes de ejecutar los cálculos.

3. Introducción

Los métodos numéricos son herramientas fundamentales en la resolución de ecuaciones no lineales donde las soluciones analíticas no son viables. El método de la Secante y Newton-Raphson son ampliamente utilizados debido a su rapidez y precisión en la aproximación de raíces.

La implementación de una interfaz gráfica mejora la accesibilidad de estos métodos al usuario, proporcionando una forma intuitiva de ingresar valores y visualizar los resultados, minimizando errores y facilitando la interpretación de los datos.

Objetivo General: Desarrollar una herramienta interactiva para calcular raíces de ecuaciones no lineales usando los métodos de Newton-Raphson y Secante, asegurando la validación de datos y la representación gráfica de la convergencia.

4. Fundamento Teórico

4.1 Método de la Secante

Ecuación base: xn+1=xn−f(xn)(xn−xn−1) f(xn)−f(xn−1) xn+1=xn−f(xn)−f(xn−1) f(xn)(xn−xn−1)

Condiciones de aplicación:

Se requieren dos valores iniciales distintos.

La función debe ser continua en el intervalo de evaluación.

Convergencia:

Converge más rápido que el método de bisección, pero puede fallar si la elección de valores iniciales es inadecuada.

4.2 Método de Newton-Raphson

Fórmula: xn+1=xn−f(xn)f′(xn)xn+1=xn−f′(xn)f(xn)

Hipótesis de funcionamiento:

Requiere la derivada de la función.

Se necesita un valor inicial razonablemente cercano a la raíz buscada.

Riesgos de divergencia:

Si la derivada es cero o cercana a cero, el método puede fallar.

Puede oscilar o no converger si el valor inicial no está bien elegido.

5. Diseño de la Interfaz

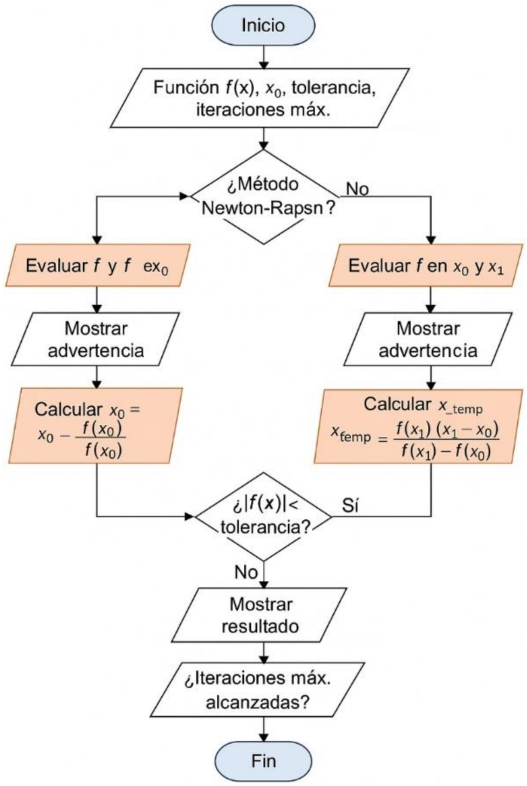
5.1 Herramientas y tecnologías utilizadas

- **Python**: Lenguaje de programación principal.
- **Tkinter**: Biblioteca para la creación de interfaces gráficas.
- **SymPy**: Biblioteca para manipulación simbólica y derivación de funciones matemáticas.
- **Matplotlib**: Biblioteca para la generación de gráficos y visualización de resultados.

5.2 Estructura general de la GUI La interfaz está diseñada para facilitar la interacción del usuario con los métodos numéricos, proporcionando campos de entrada y botones para ejecutar los cálculos y visualizar los resultados.

- **Módulos principales**:
  - Entrada de datos: Campos para la función, valores iniciales, tolerancia y número máximo de iteraciones.
  - Selección del método: Botones para elegir entre Newton-Raphson y Secante.
  - Ejecución del método: Botón "Calcular" para iniciar el proceso.
  - Visualización de resultados: Ventanas emergentes con la tabla de iteraciones y gráficas de convergencia.

5.3 Diagrama de flujo o esquema de interacción



6. Desarrollo del Código

6.1 Lógica de ingreso de datos

- Validación de la sintaxis de la función ingresada.
- Verificación de que los valores iniciales sean números válidos.
- Control de errores en la tolerancia y el número de iteraciones.

6.2 Implementación del método de Secante

- Cálculo iterativo de la raíz según la fórmula del método de la secante.
- Manejo de errores cuando los valores iniciales son iguales o cuando la diferencia entre f(x0) y f(x1) es cero.
- Almacenamiento de valores intermedios para su posterior visualización.

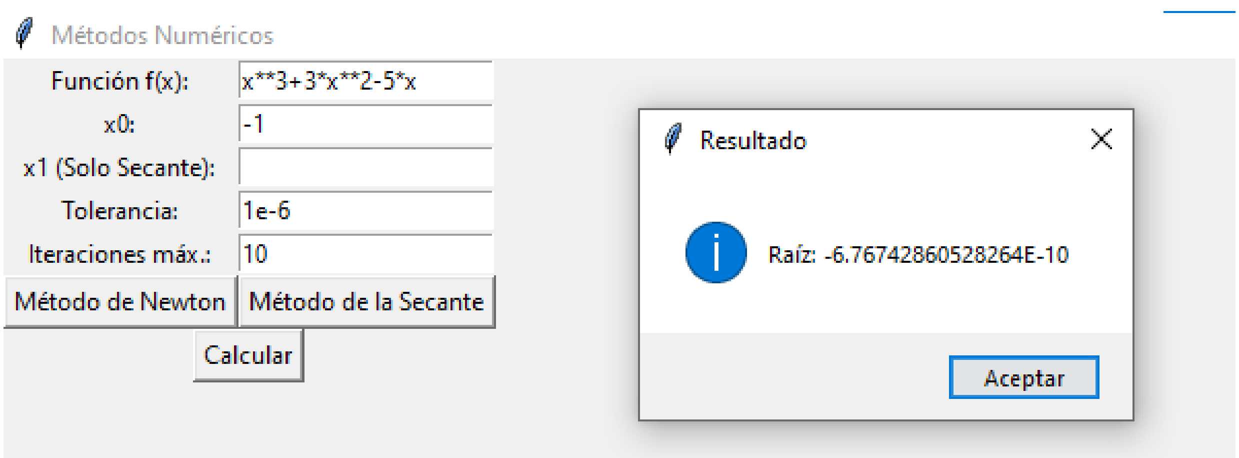
6.3 Implementación del método de Newton-Raphson

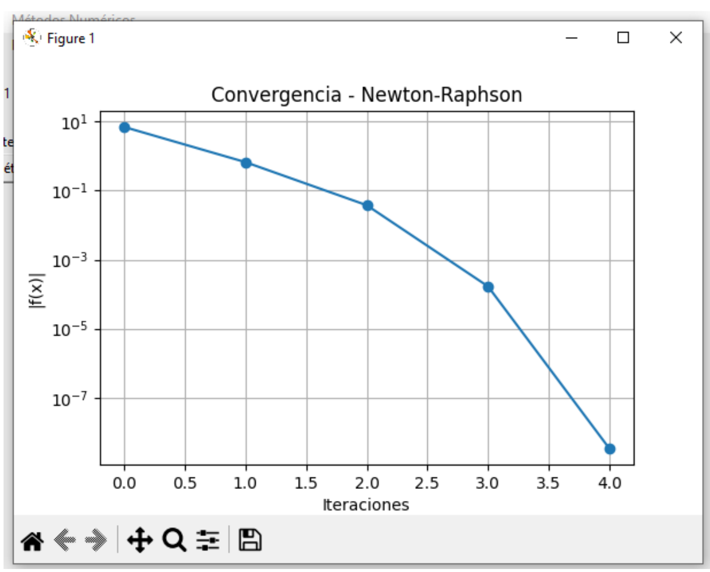
- Derivación simbólica de la función utilizando SymPy.
- Iteración basada en la fórmula de Newton-Raphson.
- Control de errores cuando la derivada es cero en algún punto.

6.4 Salida de resultados y gráficas

- Despliegue de la raíz aproximada en una ventana emergente.
- Tabla de iteraciones con valores intermedios.
- Gráfico de convergencia del error y del valor de la raíz.

7. Ejemplos y Pruebas





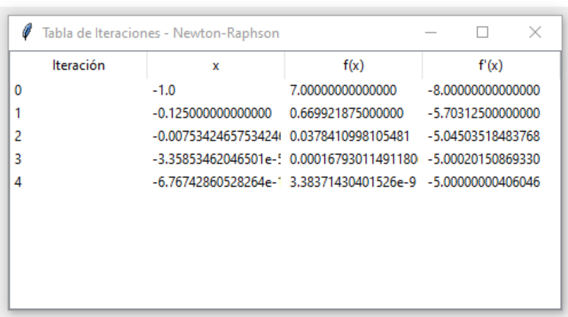
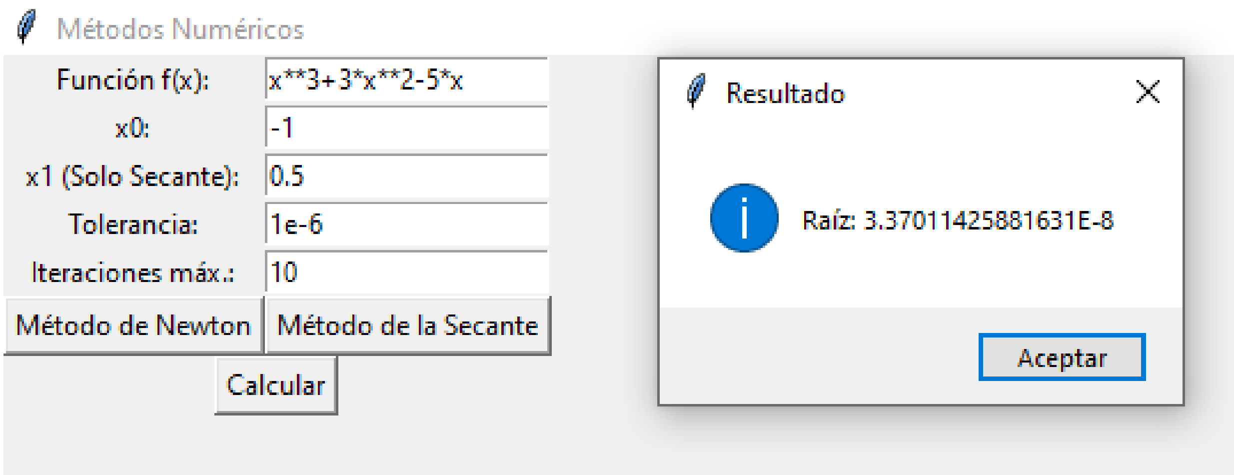
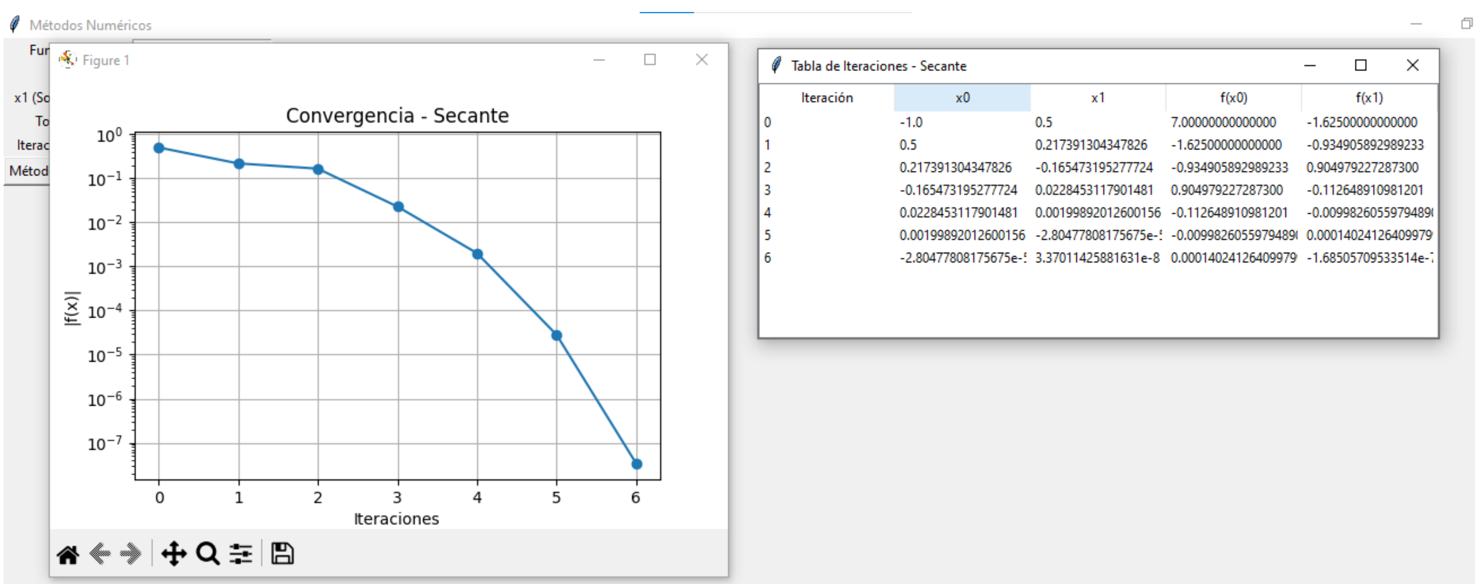


Tabla de Iteraciones - Newton-Raphson

Iteración	x	f(x)	f'(x)
0	-1.0	7.00000000000000	-8.00000000000000
1	-0.125000000000000	0.669921875000000	-5.70312500000000
2	-0.0075342465753424	0.0378410998105481	-5.04503518483768
3	-3.35853462046501e-	0.000167930114911180	-5.00020150869330
4	-6.76742860528264e-	3.38371430401526e-9	-5.00000000406046





Iteración	x0	x1	f(x0)	f(x1)
0	-1.0	0.5	7.00000000000000	-1.62500000000000
1	0.5	0.217391304347826	-1.62500000000000	-0.934905892989233
2	0.217391304347826	-0.165473195277724	-0.934905892989233	0.904979227287300
3	-0.165473195277724	0.0228453117901481	0.904979227287300	-0.112648910981201
4	0.0228453117901481	0.00199892012600156	-0.112648910981201	-0.0099826055979489
5	0.00199892012600156	-2.80477808175675e-	-0.0099826055979489	0.00014024126409979
6	-2.80477808175675e-	3.37011425881631e-8	0.00014024126409979	-1.68505709533514e-

8. Resultados y Discusión

Análisis del comportamiento de cada método

- Método de Newton-Raphson:
  - Presenta convergencia rápida cuando la derivada en el punto inicial no es cero y el valor inicial está cerca de la raíz.
  - El número de iteraciones es generalmente bajo.
  - Puede fallar si la derivada en algún punto es cero (lo cual está bien manejado en el código mediante advertencias).
- Método de la Secante:
  - No requiere derivadas, lo cual es útil para funciones donde calcular la derivada es complicado.

    - Su convergencia es generalmente más lenta que la de Newton-Raphson, pero más estable en ciertos casos si se eligen bien los valores iniciales.
    - Si f(x1) =f(x0) f(x1) = f(x0) f(x1) = f(x0), el método lanza una advertencia y se detiene para evitar división por cero.

Convergencia y Precisión

- Las gráficas de convergencia generadas por Matplotlib permiten observar cómo |f(x)||f(x)||f(x)| disminuye con cada iteración.
- Se utiliza escala logarítmica en el eje Y, lo cual facilita visualizar la rapidez con la que se acerca a cero.
- Ambas funciones alcanzan buena precisión si los valores iniciales son adecuados y la tolerancia es razonable (por ejemplo, 10−510^ {-5}10−5).

Usabilidad de la interfaz

- La interfaz gráfica es clara y sencilla:
    - Permite ingresar todos los parámetros necesarios para cada método.
    - Diferencia entre campos necesarios para Newton y Secante (x1 se marca como “Solo Secante”).
    - Usa botones para seleccionar el método y mostrar los resultados.
- Las advertencias son útiles para evitar errores como:
    - Derivada cero en Newton.
    - Valores iniciales iguales en la Secante.
- Se muestra una tabla de iteraciones y una gráfica de convergencia, lo que mejora la comprensión del proceso iterativo.

9. Conclusiones

Logros alcanzados

- Se implementaron correctamente los métodos de Newton-Raphson y Secante.
- La interfaz gráfica permite que cualquier usuario, sin experiencia en programación, pueda usarlos fácilmente.
- Se manejan adecuadamente errores comunes como divisiones por cero o derivadas nulas.
- La visualización gráfica mejora la comprensión del comportamiento de los métodos.

Limitaciones encontradas (con base en las gráficas mostradas en resultados)

- Ambos métodos dependen fuertemente de la elección de los valores iniciales:
    - Una mala elección puede provocar que el método no converja, se detenga prematuramente o incluso que diverja.
- El método de la secante es más general, pero puede requerir más iteraciones y es sensible a la similitud entre f(x0) y f(x1).
- El método de Newton-Raphson es más eficiente, pero requiere que la función tenga derivada continua y que ésta no sea cero en el proceso.
- Las gráficas muestran que hay casos donde los métodos oscilan o tienen convergencia muy lenta si los puntos iniciales están lejos de la raíz.

10. Referencias

https://docs.python.org/es/3.13/library/tk.html

https://docs.python.org/es/3.12/library/tk.html

https://librosoa.unam.mx/bitstream/handle/123456789/3416/MNPython.pdf?sequence=1

11. Anexos

```
import tkinter as tk
from tkinter import ttk, messagebox
import sympy as sp
import matplotlib.pyplot as plt

def newton_raphson_method(f, x0, tol, max_iter):
    x = sp.symbols('x')
    f = sp.sympify(f)
    f_prime = sp.diff(f, x)
    iter_values = []

    for i in range(max_iter):
        fx = f.subs(x, x0)
        f_prime_x = f_prime.subs(x, x0)
        iter_values.append((i, x0, fx, f_prime_x))

        if abs(fx) < tol:
            break
        if f_prime_x == 0:
            messagebox.showwarning("Advertencia", "Derivada en x0 es cero. No se puede continuar.")
```

```
            return None, iter_values

        x0 = x0 - fx / f_prime_x

    return x0, iter_values

def secant_method(f, x0, x1, tol, max_iter):
    x = sp.symbols('x')
    f = sp.sympify(f)
    iter_values = []

    for i in range(max_iter):
        fx0, fx1 = f.subs(x, x0), f.subs(x, x1)
        iter_values.append((i, x0, x1, fx0, fx1))

        if abs(fx1) < tol:
            break
        if fx1 - fx0 == 0:
            messagebox.showwarning("Advertencia", "f(x1) - f(x0) es cero. No
se puede continuar.")
            return None, iter_values

        x_temp = x1 - (fx1 * (x1 - x0)) / (fx1 - fx0)
        x0, x1 = x1, x_temp

    return x1, iter_values

def plot_convergence(iter_values, method_name):
    iterations = [i[0] for i in iter_values]
    errors = [abs(i[2]) for i in iter_values if len(i) > 2]

    plt.figure(figsize=(6, 4))
    plt.plot(iterations, errors, marker='o', linestyle='-')
    plt.yscale("log")
    plt.xlabel("Iteraciones")
    plt.ylabel("|f(x)|")
    plt.title(f"Convergencia - {method_name}")
    plt.grid()
    plt.show()

def display_table(iter_values, method_name):
    table_window = tk.Toplevel(root)
    table_window.title(f"Tabla de Iteraciones - {method_name}")

    if method_name == "Newton-Raphson":
```

```python
        columns = ("Iteración", "x", "f(x)", "f'(x)")
    elif method_name == "Secante":
        columns = ("Iteración", "x0", "x1", "f(x0)", "f(x1)")

    tree = ttk.Treeview(table_window, columns=columns, show="headings")
    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, width=120)

    for row in iter_values:
        tree.insert("", "end", values=row)

    tree.pack(fill=tk.BOTH, expand=True)

def calculate():
    method = method_var.get()
    f = function_entry.get()
    tol = float(tolerance_entry.get())
    max_iter = int(iteration_entry.get())

    try:
        if method == "Newton-Raphson":
            x0 = float(x0_entry.get())
            result, iter_values = newton_raphson_method(f, x0, tol,
max_iter)
            if result is not None:
                messagebox.showinfo("Resultado", f"Raíz: {result}")
                display_table(iter_values, "Newton-Raphson")
                plot_convergence(iter_values, "Newton-Raphson")

        elif method == "Secant":
            x0 = float(x0_entry.get())
            x1 = float(x1_entry.get())
            if x0 == x1:
                messagebox.showwarning("Advertencia", "x0 y x1 no deben ser
iguales.")
                return
            result, iter_values = secant_method(f, x0, x1, tol, max_iter)
            if result is not None:
                messagebox.showinfo("Resultado", f"Raíz: {result}")
                display_table(iter_values, "Secante")
                plot_convergence(iter_values, "Secante")

        else:
            messagebox.showerror("Error", "Método inválido")
```

```python
    except Exception as e:
        messagebox.showerror("Error", str(e))

# GUI setup
root = tk.Tk()
root.title("Métodos Numéricos")

tk.Label(root, text="Función f(x):").grid(row=0, column=0)
function_entry = tk.Entry(root)
function_entry.grid(row=0, column=1)

tk.Label(root, text="x0:").grid(row=1, column=0)
x0_entry = tk.Entry(root)
x0_entry.grid(row=1, column=1)

tk.Label(root, text="x1 (Solo Secante):").grid(row=2, column=0)
x1_entry = tk.Entry(root)
x1_entry.grid(row=2, column=1)

tk.Label(root, text="Tolerancia:").grid(row=3, column=0)
tolerance_entry = tk.Entry(root)
tolerance_entry.grid(row=3, column=1)

tk.Label(root, text="Iteraciones máx.:").grid(row=4, column=0)
iteration_entry = tk.Entry(root)
iteration_entry.grid(row=4, column=1)
iteration_entry.insert(0, "15")  # Valor predeterminado

method_var = tk.StringVar(value="Newton-Raphson")
tk.Button(root, text="Método de Newton", command=lambda:
method_var.set("Newton-Raphson")).grid(row=5, column=0)
tk.Button(root, text="Método de la Secante", command=lambda:
method_var.set("Secant")).grid(row=5, column=1)

tk.Button(root, text="Calcular", command=calculate).grid(row=6, column=0,
columnspan=2)

root.mainloop()
```