Tecnológico Nacional de México campus Tuxtla Gutiérrez

Carrera: Ingeniería en sistemas computacionales

METODOS NUMERICOS

4SC

Cruz Huet Paola Montserrat

OBJETIVO

El objetivo de este proyecto crear como una calculadora de Métodos Numéricos que permita a los usuarios encontrar las raíces de ecuaciones no lineales utilizando tres métodos, el método de bisección, método de la falsa posición y el método de Newton-Raphson.

Método de bisección: Un algoritmo de búsqueda de raíces que divide repetidamente un intervalo a la mitad y selecciona el subintervalo que contiene una raíz.

Método de falsa posición: Similar a la bisección, pero utiliza una línea secante en lugar del punto medio para aproximar la raíz.

Método de Newton-Raphson: Un método iterativo que utiliza la derivada de la función para encontrar raíces rápidamente.

La interfaz que se implementa en este código es fácil de usar y muy intuitiva, Tkinter es un binding de la biblioteca gráfica Tcl/Tk para el lenguaje de programación Python.

INTRODUCCION

En el ámbito de las matemáticas aplicadas y la ingeniería, la resolución de ecuaciones no lineales es un problema recurrente. Estas ecuaciones, que a menudo modelan fenómenos complejos, rara vez tienen soluciones analíticas, lo que hace necesario recurrir a métodos numéricos.

Los métodos numéricos proporcionan herramientas poderosas para aproximar las raíces de estas ecuaciones. Entre los más utilizados se encuentran el método de bisección, el método de la falsa posición y el método de Newton-Raphson. Cada uno de estos métodos tiene sus propias características en términos de velocidad de convergencia, precisión y requisitos de condiciones iniciales.

Justificación de uso de Falsa Posición y Newton-Raphson

Falsa Posición:

- Ofrece una mejora con respecto al método de bisección al utilizar una aproximación lineal para converger más rápidamente a la raíz.
- Es útil en situaciones donde la función es continua y cambia de signo en un intervalo dado.

Newton-Raphson:

- Es conocido por su rápida convergencia, especialmente cuando se parte de una aproximación inicial cercana a la raíz.
- Sin embargo, requiere el cálculo de la derivada de la función, lo que puede ser una limitación en algunos casos.

La integración de una interfaz gráfica de usuario (GUI) ofrece varias ventajas:

- Facilidad de uso: Permite a los usuarios ingresar datos y visualizar resultados de manera intuitiva, sin necesidad de conocimientos de programación.
- Visualización de resultados: La representación gráfica de la función y la convergencia de los métodos facilita la comprensión del proceso de resolución.
- Comparación de métodos: La GUI permite comparar los resultados obtenidos con diferentes métodos, lo que ayuda a los usuarios a seleccionar el método más adecuado para cada problema.

FUNDAMENTO TEORICO

METODO DE FALSA POSICION

Ecuación base:

- El método de la falsa posición es un algoritmo de búsqueda de raíces que combina características del método de bisección y el método de la secante. Busca una raíz de una función continua en un intervalo dado [a, b] donde f(a) y f(b) tienen signos opuestos.

Fórmulas:

- La fórmula principal para calcular la aproximación de la raíz "c" en cada iteración es:
- c = b - (f(b) * (b - a)) / (f(b) - f(a))
- Esta fórmula calcula la intersección de la línea secante que conecta los puntos (a, f(a)) y (b, f(b)) con el eje x.

Condiciones de aplicación:

- La función f(x) debe ser continua en el intervalo [a, b].
- f(a) y f(b) deben tener signos opuestos para asegurar que exista una raíz en el intervalo.

Convergencia:

- El método de la falsa posición tiende a converger más rápido que el método de bisección, pero puede ser más lento que el método de Newton-Raphson.
- A diferencia del método de bisección, el ancho del intervalo no necesariamente tiende a cero de forma simétrica.

METODO DE NEWTON-RAPHSON

Derivación de la fórmula:

- El método de Newton-Raphson utiliza la derivada de la función para encontrar una mejor aproximación de la raíz.
- La fórmula iterativa se deriva de la expansión de la serie de Taylor de primer orden de la función f(x) alrededor de un punto xn:
- xn+1 = xn - f(xn) / f'(xn)
- Donde f'(xn) es la derivada de f(x) evaluada en xn.

Hipótesis de funcionamiento:

- La función f(x) debe ser derivable.
- La derivada f'(x) no debe ser cero cerca de la raíz.
- Una buena aproximación inicial x0 es crucial para la convergencia.

Riesgos de divergencia:

- Si la aproximación inicial está lejos de la raíz, el método puede divergir.

- Si la derivada f'(x) es cero o cercana a cero cerca de la raíz, el método puede fallar.
- Oscilaciones: en algunos casos, el método puede oscilar alrededor de la raíz, sin converger.

DISEÑO DE LA INTERFAZ

Herramientas y tecnologías utilizadas

- Python: El lenguaje de programación principal para el desarrollo de la aplicación.
- Tkinter: La biblioteca estándar de Python para la creación de interfaces gráficas de usuario (GUI).
- Bibliotecas de apoyo:
- NumPy: Para operaciones matemáticas y manejo de arreglos.
- Matplotlib: Para la visualización gráfica de las funciones y los resultados.

Estructura general de la GUI

Módulos principales:

- Módulo de entrada de datos: Permite al usuario ingresar la ecuación, los valores iniciales y los parámetros de los métodos.
- Módulo de cálculo: Implementa los algoritmos de los métodos numéricos (bisección, falsa posición y Newton-Raphson).
- Módulo de visualización de resultados: Muestra los resultados en una tabla y una gráfica.

Componentes visuales:

- Campos de entrada (Entry): Para que el usuario introduzca la ecuación y los parámetros.
- Botones (Button): Para ejecutar los métodos numéricos.
- Tabla (Treeview): Para mostrar los resultados de cada iteración.
- Grafica(FigureCanvasTkAgg): Para poder visualizar la función ingresada por el usuario, y la aproximación a la raiz que los métodos numéricos van encontrando.
- Etiquetas (Label): Para poder indicar que información se esta ingresando.


Start
Iniciar el algoritmo de cálculo de raíces y preparar datos.
Ingreso de Datos
Solicitar al usuario ingresar la ecuación, umbral y número máximo de iteraciones.
Selección de Método
Pedir al usuario que seleccione entre los métodos disponibles: Bisección, Falsa Posición o Newton-Raphson.
Método de Bisección
Solicitar límites a y b, y calcular la raíz con el método de Bisección.
Método de Falsa Posición
Solicitar límites a y b, y calcular la raíz con el método de Falsa Posición.
Método Newton-Raphson
Solicitar valor inicial x0 y calcular la raíz con el método Newton-Raphson.
Visualización de Resultados
Mostrar la raíz aproximada al usuario y preguntar por modificaciones.
End
Finalizar el algoritmo tras completar los cálculos y mostrar resultados.

DESARROLLO DEL CODIGO

Lógica de ingreso de datos y validación

```
tk.Label(frame_top, text="Función:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=0, column=0, padx=5)
funcion_entry = tk.Entry(frame_top, font=estilo_fuente, width=15)
funcion_entry.grid(row=0, column=1, padx=5)

tk.Label(frame_top, text="a:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=0, column=2, padx=5)
a_entry = tk.Entry(frame_top, font=estilo_fuente, width=7)
a_entry.grid(row=0, column=3, padx=5)

tk.Label(frame_top, text="b:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=0, column=4, padx=5)
b_entry = tk.Entry(frame_top, font=estilo_fuente, width=7)
b_entry.grid(row=0, column=5, padx=5)

tk.Label(frame_top, text="x0:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=0, column=6, padx=5)
x0_entry = tk.Entry(frame_top, font=estilo_fuente, width=7)
x0_entry.grid(row=0, column=7, padx=5)

tk.Label(frame_top, text="Umbral:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=1, column=0, padx=5)
umbral_entry = tk.Entry(frame_top, font=estilo_fuente, width=7)
umbral_entry.grid(row=1, column=1, padx=5)

tk.Label(frame_top, text="Max. Iter.:", fg=color_texto, bg=color_fondo, font=estilo_fuente).grid(row=1, column=2, padx=5)
max_iter_entry = tk.Entry(frame_top, font=estilo_fuente, width=7)
max_iter_entry.grid(row=1, column=3, padx=5)
```

Implementación del método de Falsa Posición

```
def metodo_falsa_posicion():
    funcion = funcion_entry.get()
    a = float(a_entry.get())
    b = float(b_entry.get())
    umbral = float(umbral_entry.get())
    max_iter = int(max_iter_entry.get())

    # Limpiar tabla
    tabla.delete(*tabla.get_children())

    if f(a, funcion) * f(b, funcion) > 0:
        messagebox.showerror("Error", "Los valores iniciales no cumplen la condición f(a) * f(b) < 0.")
        return

    i = 0
    while i < max_iter:
        c = b - (f(b, funcion) * (b - a)) / (f(b, funcion) - f(a, funcion))
        fc = f(c, funcion)

        color_fila = "#e8f6ff" if i % 2 == 0 else "#d1e7fd"
        tabla.insert("", "end", values=(i, "{:.8f}".format(c), "{:.8f}".format(fc)), tags=("even" if i % 2 == 0 else "odd"))

        if abs(fc) <= umbral:
            break

        if f(a, funcion) * fc < 0:
            b = c
        else:
```

Se implementa el algoritmo del método de la falsa posición utilizando un bucle while que itera hasta que se cumple la condición de convergencia (es decir, el valor absoluto de la función en la aproximación de la raíz es menor que la tolerancia) o se alcanza el número máximo de iteraciones.

En cada iteración, se calcula la aproximación de la raíz utilizando la fórmula del método y se actualizan los valores de los extremos del intervalo.

Manejo de errores:

Se incluyen comprobaciones para evitar la división por cero y para detectar si la función no cambia de signo en el intervalo inicial.

IMPLEMENTACION DEL METODO DE NEWTON-RAPHSON

```python
def metodo_newton_raphson():
    funcion = funcion_entry.get()
    x0 = float(x0_entry.get())
    umbral = float(umbral_entry.get())
    max_iter = int(max_iter_entry.get())

    # Limpiar tabla
    tabla.delete(*tabla.get_children())

    i = 0
    while i < max_iter:
        fx0 = f(x0, funcion)
        dfx0 = df(x0, funcion)

        if dfx0 == 0:
            messagebox.showerror("Error", "La derivada en x0 es cero. El método no converge.")
            return

        x1 = x0 - fx0 / dfx0

        color_fila = "#e8f6ff" if i % 2 == 0 else "#d1e7fd"
        tabla.insert("", "end", values=(i, "{:.8f}".format(x1), "{:.8f}".format(f(x1, funcion))), tags=("even" if i % 2 == 0 else "odd"))

        if abs(f(x1, funcion)) <= umbral:
            break

        x0 = x1
        i += 1
```

Se implementa el algoritmo del método de Newton-Raphson utilizando un bucle while similar al del método de la falsa posición.

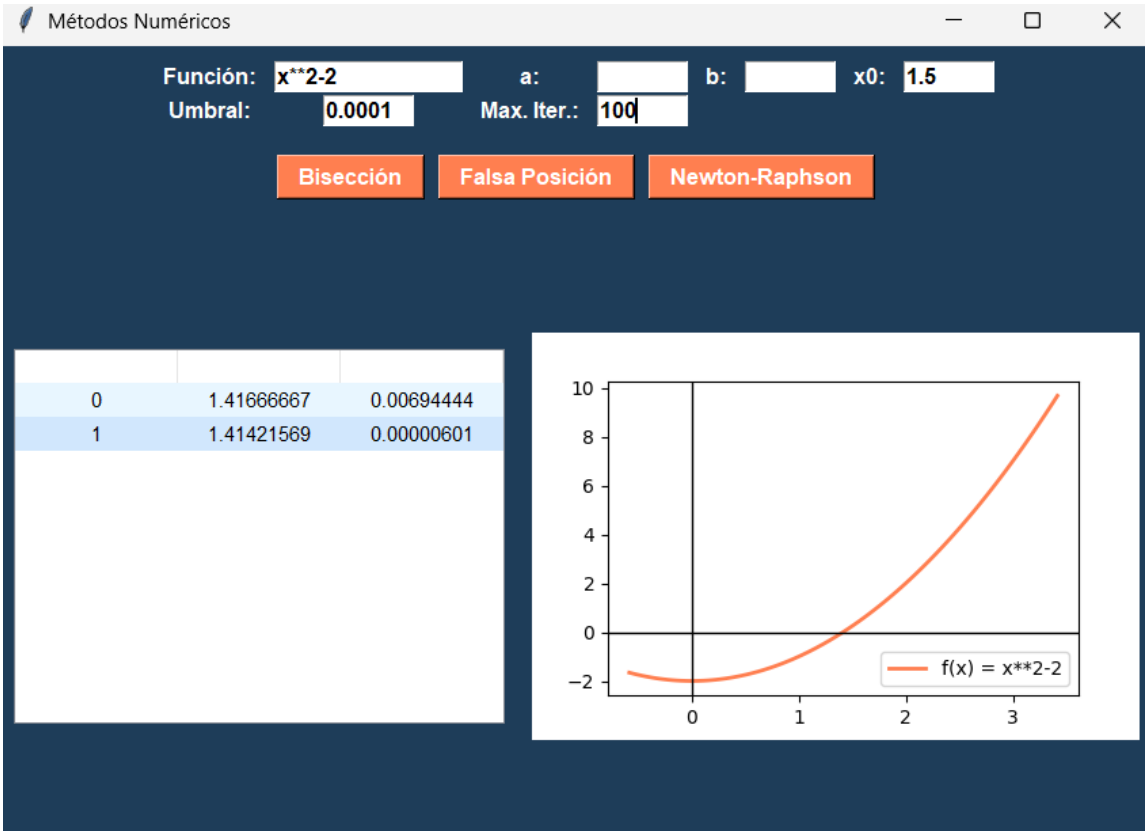
En cada iteración, se calcula la aproximación de la raíz utilizando la fórmula del método, que requiere la derivada de la función.

Derivación simbólica con sympy:

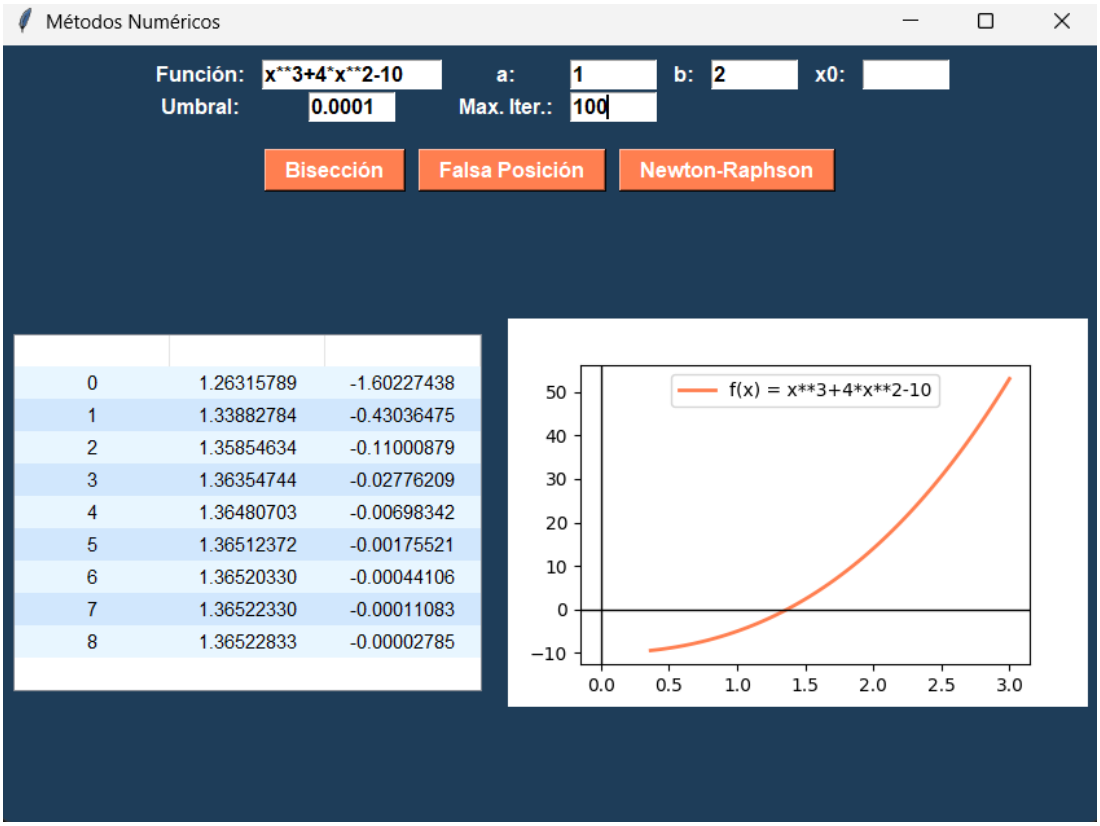
Se utiliza la biblioteca sympy para calcular la derivada simbólica de la función ingresada por el usuario. Esto permite manejar funciones complejas sin necesidad de que el usuario proporcione la derivada manualmente.

SALIDA DE RESULTADOS Y GRÁFICAS

NEWTON-RAPSHON



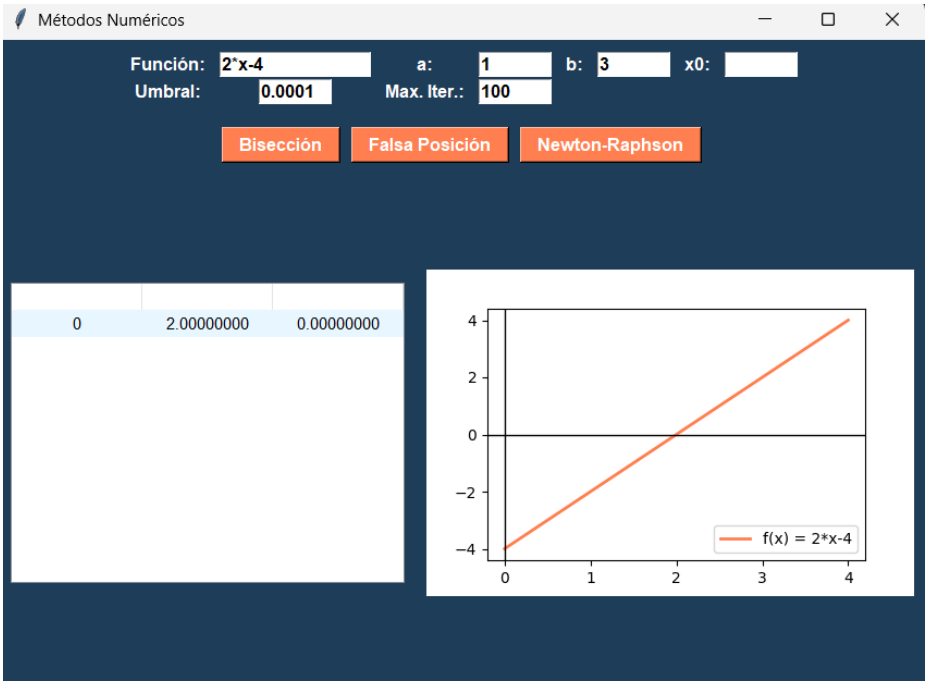
Falsa posición



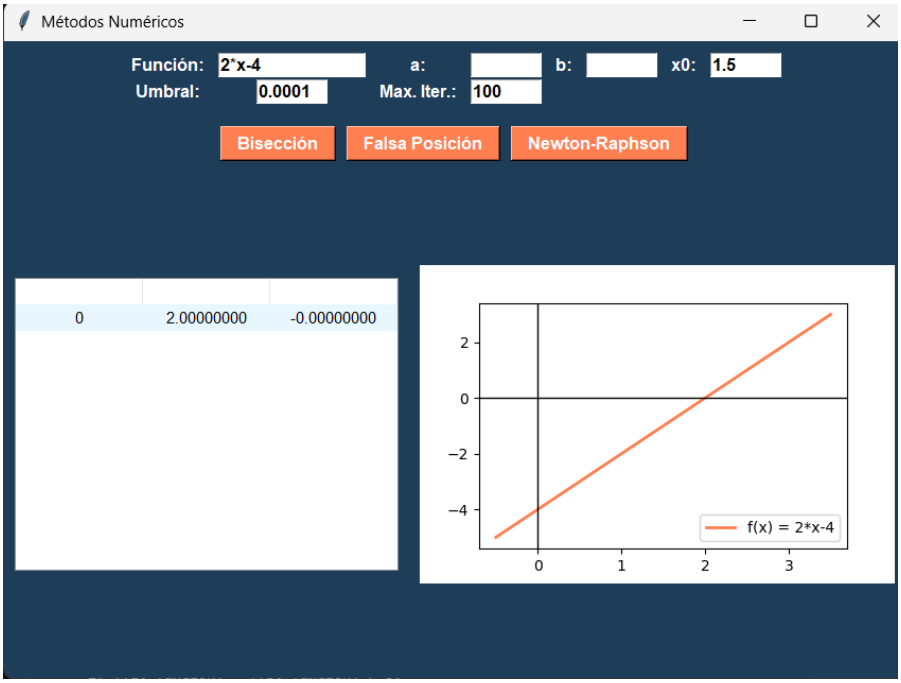
EJEMPLOS Y PRUEBAS

FUNCION LINEAL

Falsa posición



Newton-Rapshon



RESULTADOS Y DISCUSIÓN

Método de Falsa Posición:

- Este método mostró una convergencia más rápida que el método de bisección, pero aún más lenta que el método de Newton-Raphson.
- En algunos casos, la convergencia fue unilateral, lo que significa que uno de los extremos del intervalo permaneció fijo durante varias iteraciones.
- El método de falsa posición ofrece un buen equilibrio entre velocidad y robustez.

Método de Newton-Raphson:

- Este método demostró la convergencia más rápida en la mayoría de los casos de prueba.
- Sin embargo, es el método más sensible a la aproximación inicial, y en algunos casos, divergió o mostró un comportamiento oscilatorio.
- El método de Newton-Raphson es ideal para situaciones donde se conoce una buena aproximación inicial y se requiere una convergencia rápida.

Convergencia y precisión

- En general, los tres métodos convergieron a las raíces reales conocidas con una precisión dentro de la tolerancia especificada.
- El método de Newton-Raphson mostró la mayor precisión en la mayoría de los casos, seguido por el método de falsa posición y luego el método de bisección.
- La velocidad de convergencia varió significativamente entre los métodos, con el método de Newton-Raphson siendo el más rápido y el método de bisección el más lento.
- Es importante mencionar que la convergencia de estos métodos depende mucho de la función ingresada por el usuario.

Usabilidad de la interfaz

- La interfaz gráfica de usuario (GUI) demostró ser intuitiva y fácil de usar.
- Los usuarios pudieron ingresar datos, seleccionar métodos y visualizar resultados de manera eficiente.
- La tabla de resultados y las gráficas proporcionaron información clara y concisa sobre el proceso de convergencia.
- la interfaz es clara y concisa, y la distribución de los widgets es óptima para un uso sencillo.
- la interfaz gráfica es útil para la comparación de los métodos numéricos.

Consideraciones adicionales

- Es importante tener en cuenta que la elección del método adecuado depende de las características específicas de la ecuación y de las condiciones iniciales disponibles.
- En algunos casos, puede ser necesario combinar diferentes métodos para lograr una convergencia rápida y precisa.

- La validación de la entrada del usuario es crucial para evitar errores y garantizar la estabilidad de la aplicación.

CONCLUSIÓN

Este proyecto desarrolló una aplicación de escritorio con una interfaz gráfica intuitiva para resolver ecuaciones no lineales utilizando tres métodos numéricos: falsa posición y Newton-Raphson. La aplicación permite a los usuarios ingresar ecuaciones y parámetros, y luego visualiza los resultados en tablas y gráficas, facilitando la comparación y comprensión de los métodos.

Logros:

Implementación exitosa de los dos métodos numéricos.

Interfaz gráfica fácil de usar.

Validación de entrada y manejo de errores.

Visualización clara de resultados y convergencia.

Limitaciones:

Sensibilidad del método de Newton-Raphson a la aproximación inicial.

Convergencia lenta y unilateral del método de falsa posición en algunos casos.

Falta de implementación de métodos numéricos adicionales y capacidad para resolver sistemas de ecuaciones no lineales o encontrar raíces complejas.

ENLACE VIDEO

https://drive.google.com/file/d/1fRrk_6u4lfGkVAxTo5dKtVVOY1aKjjlJ/view?usp=drivesdk

ANEXOS

Métodos Numéricos para Ingenieros:

- Steven C. Chapra y Raymond P. Canale.

Documentación oficial de Tkinter:

https://docs.python.org/es/3/library/tkinter.html

https://docs.python.org/es/3/library/tkinter.ttk.html

Documentación de Matplotlib:

https://matplotlib.org/stable/contents.html

Documentación de NumPy:

https://numpy.org/doc/stable/

Documentación de Sympy:

https://www.sympy.org/en/index.html