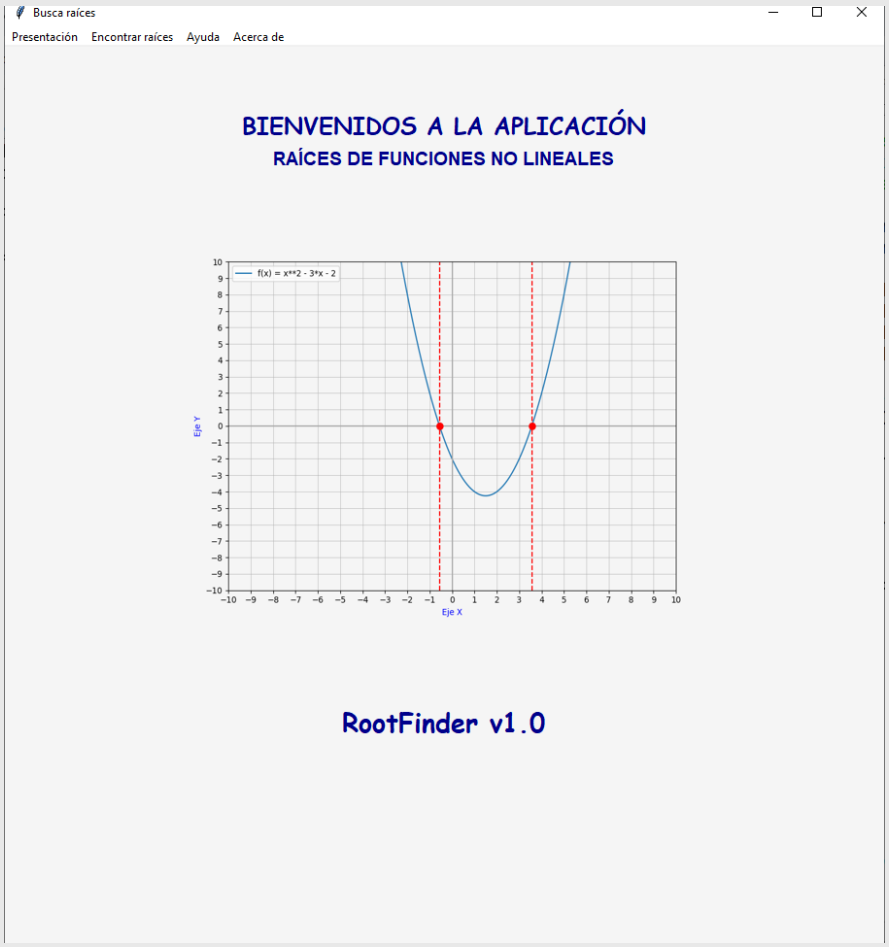
SOFTWARE DE APLICACIÓN

RAÍCES DE UNA ECUACIÓN POLINOMICA NO LINEAL



MANUAL DE USUARIO

SOFTWARE DE APLICACIÓN

RootFinder v1.0: Determina los valores de x que satisfacen f(x)=0, donde f representa una función no lineal.

AUTORES:

- LEONEL COYLA IDME
- ALFREDO MAMANI CANQUI
- ELQUI YEYE PARI CONDORI
- JUAN REYNALDO PAREDES QUISPE
- JOSÉ PÁNFILO TITO LIPA

ÍNDICE

1. INTRODUCCIÓN

RootFinder v1.0 es un software educativo diseñado para facilitar el aprendizaje de los estudiantes que cursan la asignatura de Métodos Numéricos. Su principal función es graficar ecuaciones no lineales y calcular de manera aproximada la raíz de dichas ecuaciones. Estos procedimientos son fundamentales en el análisis numérico, ya que permiten encontrar soluciones a problemas matemáticos que no se resuelven fácilmente de forma algebraica.

La aplicación ofrece una interfaz gráfica amigable y didáctica, donde el usuario puede ingresar una función matemática y observar su comportamiento en un sistema de coordenadas. El gráfico generado permite identificar visualmente el punto donde la curva corta al eje x, es decir, la raíz de la ecuación. Además, el software realiza el cálculo de la raíz, lo que ayuda al estudiante a comprobar sus resultados y mejorar su comprensión de los métodos numéricos.

Gracias a su diseño práctico y funcional, RootFinder v1.0 se convierte en una herramienta de apoyo ideal para el proceso de enseñanza-aprendizaje. Su uso permite reforzar conceptos teóricos mediante la experimentación visual e interactiva, contribuyendo al desarrollo de habilidades analíticas y al entendimiento de la importancia de los métodos numéricos en la resolución de problemas matemáticos reales.

Adicionalmente, RootFinder v1.0 está pensado para ser utilizado tanto en entornos académicos como de autoaprendizaje. Su facilidad de instalación, sus herramientas intuitivas y sus resultados inmediatos permiten que cualquier estudiante o docente pueda utilizar el software sin complicaciones, promoviendo un aprendizaje autónomo y dinámico dentro y fuera del aula.

PROYECTO REGISTRADO CON URL:

https://github.com/leonelcoyla/RootFinder

2. REQUISITOS DEL SISTEMA

Para garantizar un rendimiento óptimo de RootFinder v1.0, el software diseñado para construir gráficos de ecuaciones no lineales y calcular la raíz de dicha ecuación, es fundamental que tu sistema cumpla con los requisitos mínimos y recomendados. Asegúrate de que tanto el hardware como el software de tu equipo estén correctamente configurados antes de instalar y utilizar el programa.

2.1. Sistema Operativo

Al aplicativo RootFinder v1.0 es compatible con los siguientes sistemas operativos:

- Windows 7, 8, 10 o superior
- Linux (Ubuntu 18.04 o superior)
- macOS (versión 10.12 o superior)

2.2. Hardware

- Procesador: Intel Core i3 o superior
- Memoria RAM: 2 GB mínimo (recomendado 4 GB)
- Espacio en Disco: 100 MB libres
- Resolución de pantalla mínima: 1024x768

2.3. Software

- Python 3.10 o superior
- Librerías de Python necesarias:
  - matplotlib
  - tkinter
  - pandas

3. INSTALACIÓN Y CONFIGURACIÓN

Para poner en funcionamiento RootFinder v1.0, debes asegurarte de que los siguientes componentes estén correctamente instalados en tu equipo:

3.1. Instalar Python

- Descargar Python desde la página oficial: https://www.python.org/downloads/
- Durante la instalación, marcar la opción "Add Python to PATH"
- Finalizar el proceso de instalación.
-

3.2. Ejecutar la Aplicación

- Guarde el código dado en un archivo con el nombre RootFinder.py.
- Abra la terminal, diríjase a la carpeta donde almacenó el archivo y ejecútelo.:

python RootFinder.py

- También es posible ejecutar el archivo RootFinder.exe directamente desde su ubicación.
- Otra opción es instalar el programa ejecutando mysetup.exe desde la carpeta donde se encuentra.

Archivo principal

RootFinder.py #código fuente

RootFinder.exe archivo ejecutable

mysetup.exe archivo instalador de la aplicación

CONTACTO DEL DESARROLLADOR: +51951679658

3.3. Configuración opcional

RootFinder v1.0 está configurado para funcionar con una resolución de pantalla mínima de **1024x768 px**. Si deseas usar una resolución mayor, puedes modificar el tamaño de la ventana en el código.

- Abre el archivo **RootFinder.py** en un editor de texto.
- Busca la línea donde se establece el tamaño de la ventana (en el código proporcionado, es root.geometry("900x900"))
- Cambia las dimensiones a tu preferencia. Por ejemplo:

```
root.geometry("900x900") # Ajusta el tamaño de la ventana
```

3.4. Solución de problema

Problema	Solución
Python no se reconoce como comando	Asegurarse de haber marcado *Add Python to PATH* en la instalación.
Error al instalar librerías	Ejecutar `pip install --upgrade pip` antes de instalar.
La aplicación no abre	Verificar que el archivo esté completo y en la ubicación correcta.

3.5. Actualización

- Para actualizar RootFinder v.1.0:
- Verificar si existe una nueva versión disponible en el sitio oficial o repositorio.

- Descargar la nueva versión.
- Reemplazar los archivos antiguos.
- Ejecutar nuevamente la aplicación.

4. DESCRIPCIÓN DE LA APLICACIÓN RootFinder v1.0

RootFinder v1.0 es una aplicación educativa desarrollada con el objetivo de facilitar el aprendizaje de conceptos matemáticos fundamentales, como la representación gráfica de funciones no lineales y el cálculo de raíces de ecuaciones. Esta herramienta está orientada principalmente a estudiantes que desean comprender de manera visual y práctica cómo se comportan las funciones en el plano cartesiano.

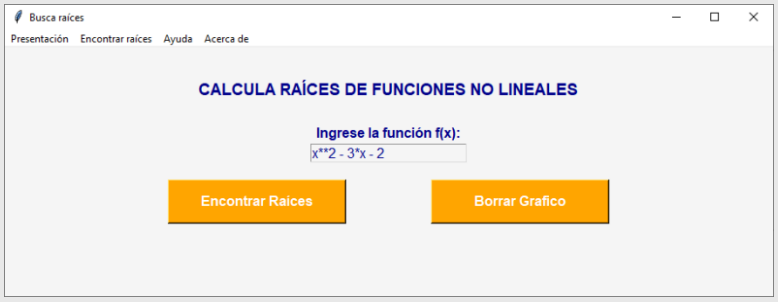
El software permite al usuario ingresar una ecuación no lineal, visualizar su gráfica de forma precisa y determinar de manera aproximada la raíz o solución de dicha ecuación. RootFinder v1.0 ofrece una interfaz intuitiva, amigable y con colores llamativos, lo que hace que la experiencia de aprendizaje sea más dinámica y accesible.

4.1. Menú Presentación

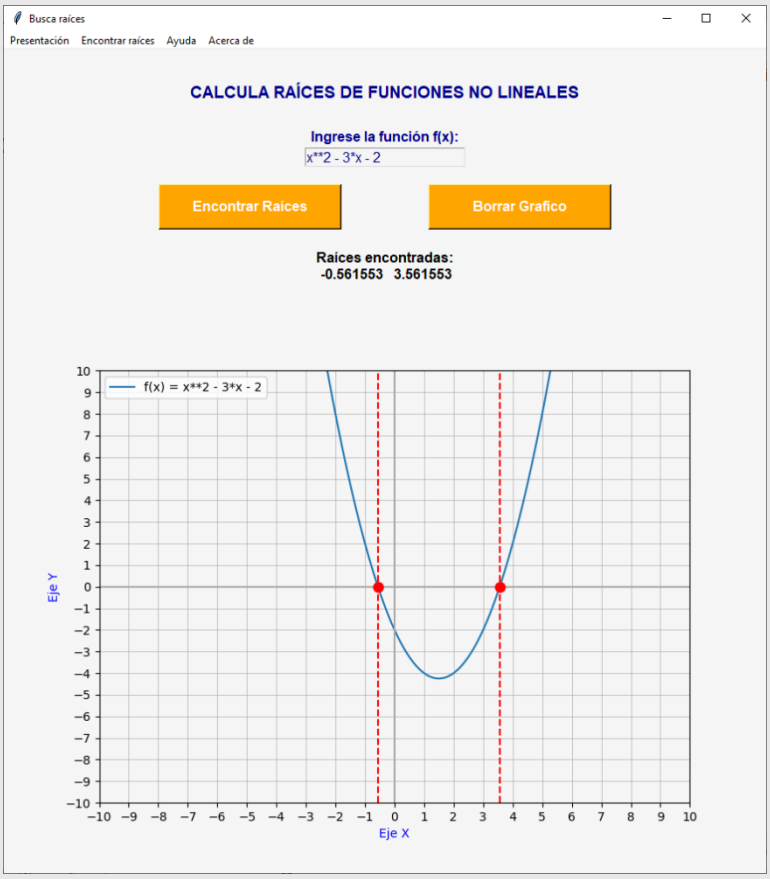
Al iniciar RootFinder v1.0, los usuarios son recibidos en el menú de presentación, donde se muestra un mensaje de bienvenida, acompañado del logotipo representativo de la aplicación y su nombre resaltado. Este espacio está diseñado para introducir al usuario a la interfaz y al propósito principal del software, permitiéndole conocer sus funciones antes de proceder a graficar ecuaciones no lineales y calcular sus respectivas raíces.

4.2. RootFinder v1.0

La pestaña "Gráfico" de *RootFinder v1.0* es la sección central de la aplicación, diseñada para interactuar con ecuaciones no lineales. En primer lugar, el usuario debe ingresar la ecuación matemática que desea graficar en un campo de texto específico.

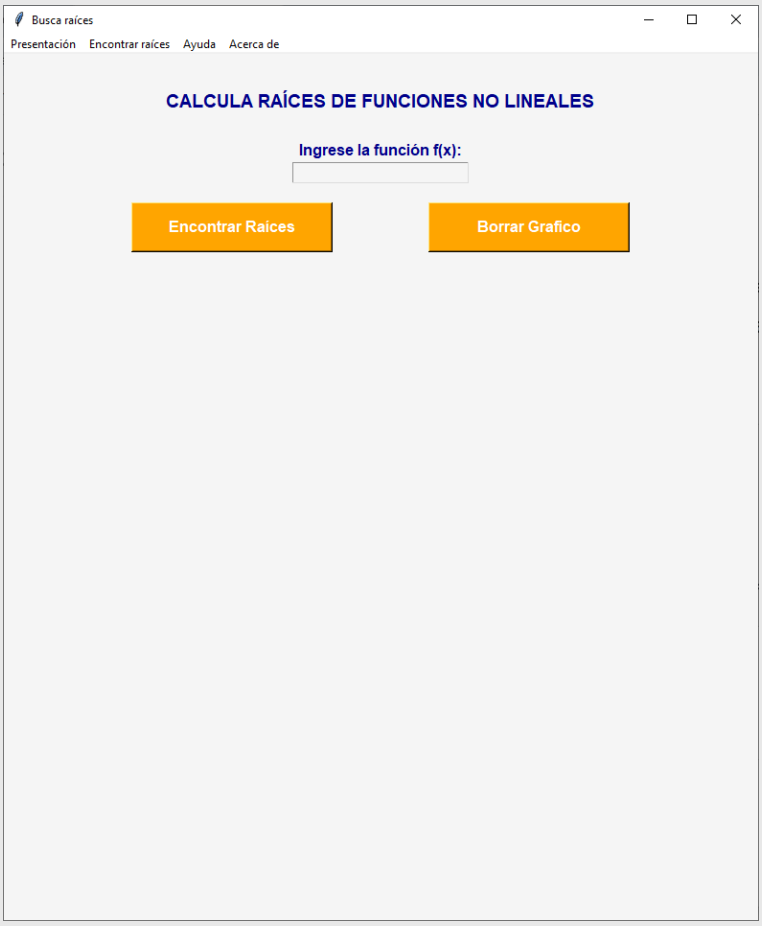


Una vez ingresada la ecuación, haga clic en el botón 'Encontrar Raíces'. Esto generará el gráfico de la ecuación y mostrará la(s) raíz(es) en el plano cartesiano, permitiendo al usuario visualizar el comportamiento de la función de manera clara y precisa.



Al generar el gráfico y mostrar las raíces de la ecuación en el intervalo de interés, el software indica los puntos donde la curva intersecta el eje x, lo que representa visualmente las soluciones de la ecuación. Además de crear el gráfico, RootFinder v1.0 resalta las raíces con marcas visuales, ayudando a los estudiantes a comprender cómo se encuentran las soluciones de las ecuaciones no lineales en el contexto de los métodos numéricos

Al finalizar el análisis, el usuario tiene la opción de borrar el gráfico generado, lo que le permite ingresar una nueva ecuación y realizar un nuevo cálculo. Este ciclo interactivo facilita el aprendizaje, permitiendo que los estudiantes experimenten con diferentes funciones y valores, y mejoren su comprensión de los conceptos teóricos mediante la visualización dinámica y el cálculo de raíces.



El código de RootFinder v1.0 está diseñado para trabajar con una amplia variedad de **funciones matemáticas**, siempre que sean compatibles con la sintaxis de **SymPy** y **NumPy**. A continuación, te menciono los principales tipos de funciones que puedes ingresar:

Tipos de funciones compatibles:

1. **Funciones polinómicas:**

   Ejemplo: x**2 - 4*x + 3, x**5 + 2*x - 7

2. **Funciones racionales:**

   Ejemplo: (x**2 - 1)/(x + 1)

3. **Funciones trigonométricas:**

   Ejemplo: sin(x), cos(x), tan(x), sin(x) - x/2

4. **Funciones exponenciales y logarítmicas:**

   Ejemplo: exp(x) - 2, log(x + 1)

5. **Funciones combinadas:**

   Ejemplo: sin(x) + x**2 - log(x + 2)

6. **Funciones raíz o radicales:**

   Ejemplo: sqrt(x + 4), x**(1/3)

Importante:

- La variable debe ser escrita como x.
- Usa ** para potencias, no ^ (por ejemplo, escribe x**2 y no x^2).
- No ingreses funciones fuera del dominio válido (por ejemplo, log(x) con $x \leq 0$) ya que puede generar errores.

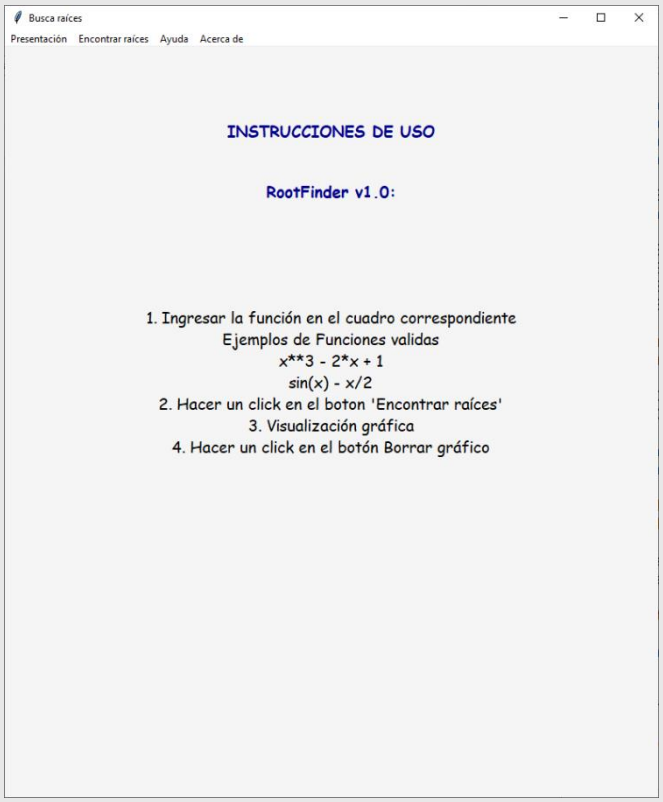
Método de cálculo de raíces

Para hallar las raíces utilizamos el método de Newton, mediante la función opt.newton pertenece al módulo scipy.optimize de Python y se utiliza para **encontrar raíces de funciones no lineales** mediante el **método de Newton-Raphson** o una variante de él. Aplicando la librería from scipy import optimize as opt.

4.3. Menú Ayuda

El menú "Ayuda" ofrece una guía práctica y accesible que facilita a los estudiantes la comprensión y el uso adecuado del software RootFinder v1.0. Este recurso está diseñado para proporcionar información clara y directa a los usuarios.

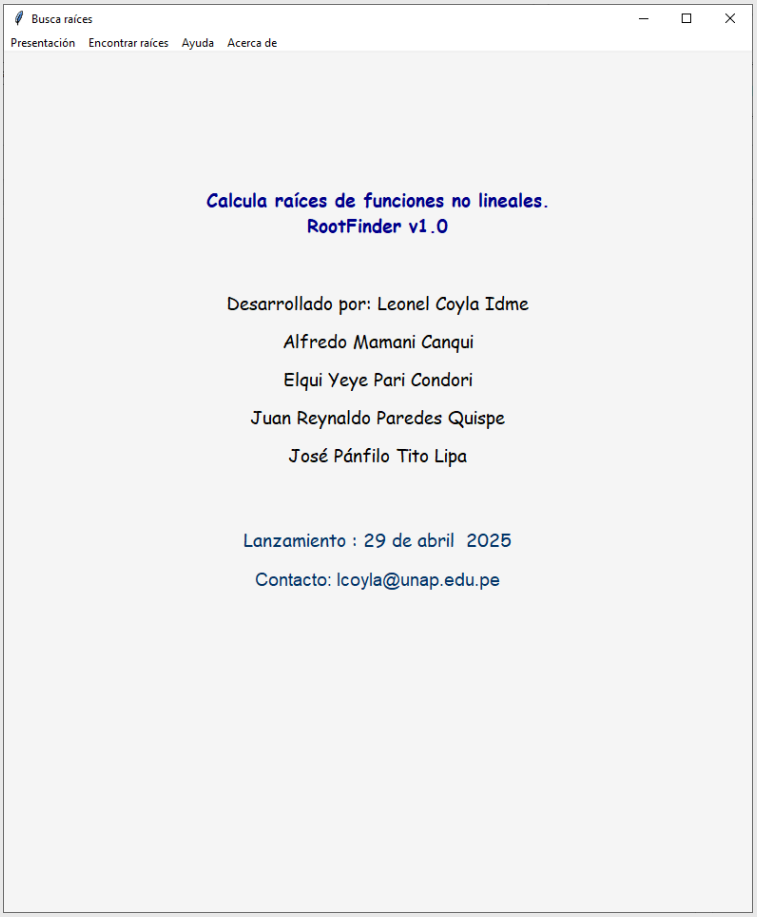
Su objetivo principal es brindar orientación de manera sencilla, permitiendo a los estudiantes familiarizarse con las principales funciones de la aplicación y así aprovechar al máximo las herramientas que ofrece el programa.



4.4. Menú Acerca de

En la sección "Acerca de", los usuarios podrán encontrar información importante sobre RootFinder v1.0, una aplicación educativa desarrollada para facilitar la elaboración de gráficos de ecuaciones no lineales y determinar sus respectivas raíces. Este apartado proporciona detalles como el nombre completo del software, la versión actual, el nombre del desarrollador principal, Leonel Coyla Idme, la fecha de lanzamiento, así como los datos de contacto disponibles para consultas, sugerencias o soporte técnico.

Este apartado tiene como finalidad brindar detalles sobre la autoría y procedencia de la herramienta, promoviendo la confianza de los usuarios al utilizarla. Además, facilita un canal de comunicación directo con el desarrollador para resolver dudas, realizar consultas o compartir sugerencias. Gracias a esta sección, los estudiantes, docentes y padres pueden conocer de manera clara la procedencia del software y confirmar que están utilizando una herramienta confiable y diseñada con fines educativos específicos.



5. PREGUNTAS FRECUENTES (FAQS) SOBRE EL SOFTWARE

5.1. ¿Qué es RootFinder y cuál es su propósito?

RootFinder v1.0 es un software educativo diseñado para facilitar el aprendizaje de los estudiantes que cursan la asignatura de Métodos Numéricos. Su propósito principal es graficar ecuaciones no lineales y calcular de manera aproximada la raíz de dichas ecuaciones. Estos procedimientos son fundamentales en el análisis numérico, ya que permiten encontrar soluciones a problemas matemáticos que no se resuelven fácilmente de forma algebraica.

5.2. ¿Cuáles son los requisitos mínimos del sistema para ejecutar?

Para ejecutar correctamente la aplicación RootFinder v1.0, se requiere un sistema operativo Windows 7 o superior, aunque también es compatible con macOS y distribuciones de Linux. Es necesario contar con al menos 2 GB de memoria RAM y un procesador de doble núcleo a 1.5 GHz o superior. Se recomienda una resolución de pantalla mínima de 1024x768 píxeles para una visualización adecuada de la interfaz. Además, debe tener instalado Python 3.8 o superior, junto con las bibliotecas Tkinter, SymPy, NumPy, Matplotlib y Pillow. Es fundamental tener acceso a un entorno gráfico que permita ejecutar interfaces basadas en Tkinter. El espacio en disco requerido es mínimo, aproximadamente 100 MB. Se sugiere mantener el sistema actualizado para garantizar compatibilidad y rendimiento óptimo.

5.3. ¿Cómo instalo o configuro RootFinder?

Para instalar y configurar RootFinder v1.0, primero debes tener Python 3.8 o una versión superior instalado en tu equipo; puedes descargarlo desde la página oficial de Python y asegurarte de marcar la opción "Add Python to PATH" durante la instalación. Luego, crea una carpeta exclusiva donde guardarás el archivo principal RootFinder.py junto con la imagen Graph.png. Abre una terminal y ejecuta el comando pip install sympy numpy matplotlib pillow para instalar las bibliotecas necesarias. Una vez instaladas, navega en la terminal hasta la carpeta del proyecto y ejecuta el comando python RootFinder.py para iniciar la aplicación. RootFinder funcionará correctamente en sistemas Windows, macOS o Linux que cuenten con al menos 2 GB de RAM y un procesador de doble núcleo. Se recomienda una resolución mínima de pantalla de 1024x768 píxeles para una visualización óptima. Es importante contar con un entorno gráfico disponible, especialmente en sistemas Linux. También puedes crear un acceso directo o archivo .bat para facilitar su ejecución.

5.4. ¿Qué se debe realizar si el software no funciona correctamente?

Si el software RootFinder v1.0 no funciona correctamente, lo primero que se debe hacer es verificar que todas las bibliotecas necesarias estén correctamente instaladas, incluyendo sympy, numpy, matplotlib, pillow y que se esté utilizando Python 3.8 o una versión superior. También es importante comprobar que no existan errores de escritura en la función ingresada y que la imagen Graph.png esté ubicada en la misma carpeta del programa. Si el problema persiste, se recomienda reiniciar la aplicación y, de ser necesario, el sistema operativo. En caso de que los errores continúen, se debe contactar al desarrollador principal a través del correo electrónico proporcionado en la sección "Acerca de" del software (lcoyla@unap.edu.pe), describiendo detalladamente el inconveniente, incluyendo capturas de pantalla o mensajes de error si es posible.

6. ADVERTENCIAS Y SUGERENCIAS DEL SOFTWARE DE APLICACIÓN.

6.1. Advertencias

Es importante tener en cuenta que RootFinder v1.0 está diseñado exclusivamente con fines educativos, por lo que no debe utilizarse para aplicaciones técnicas o profesionales que requieran alta precisión en cálculos numéricos. El uso incorrecto del software, como el ingreso de funciones mal escritas o fuera del dominio permitido, puede generar errores o resultados inesperados. Asimismo, modificar el código fuente sin conocimientos adecuados puede afectar su funcionamiento. Se recomienda no eliminar ni cambiar el nombre del archivo de imagen Graph.png, ya que esto podría impedir la correcta visualización en la sección de presentación. Por último, se sugiere utilizar la aplicación en sistemas compatibles y mantener las bibliotecas actualizadas para evitar problemas de ejecución.

6.2. Sugerencias

Se sugiere a los usuarios explorar y practicar con diversas funciones matemáticas dentro de la aplicación RootFinder v1.0 para familiarizarse con el comportamiento gráfico de las ecuaciones no lineales y el proceso de búsqueda de raíces. Esta interacción no solo refuerza el aprendizaje teórico, sino que también permite desarrollar habilidades analíticas al interpretar los resultados visuales obtenidos. Además, se recomienda utilizar funciones reales aplicadas a contextos cotidianos o problemas universitarios, lo que facilitará una comprensión más significativa y motivadora del tema.

7. Código

```python
import tkinter as tk
import sympy as sp
import scipy.optimize as opt
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import PhotoImage
from PIL import Image, ImageTk

ejemploEcuacion = "x**2 - 3*x - 2"
entry_function = None

def on_entry_focus_in(event):
    if entry_function.get() == ejemploEcuacion:
        entry_function.delete(0, tk.END)
        entry_function.config(fg="black")

def on_entry_focus_out(event):
    if entry_function.get().strip() == "":
        entry_function.insert(0, ejemploEcuacion)
        entry_function.config(fg="gray")

def encontrarRaices():
    global entry_function
    limpiarFrame()
    for widget in canvasFrame.winfo_children():
        widget.destroy()

    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="CALCULA RAÍCES DE FUNCIONES NO LINEALES",
             font=("Arial", 14, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="Ingrese la función f(x):", font=("Arial", 12, "bold"),
             fg="#00008B", bg="whitesmoke").pack()

    entry_function = tk.Entry(frameContenido, font=("Arial", 12), fg="#00008B", bg="whitesmoke")
    entry_function.insert(0, ejemploEcuacion)  # Muestra una ecuación como ejemplo de ingreso
    entry_function.bind("<FocusIn>", on_entry_focus_in)
    entry_function.bind("<FocusOut>", on_entry_focus_out)
    entry_function.pack()

    def newtonRaices():
        try:

            exprCadena = entry_function.get()
            x = sp.symbols('x')
            expr = sp.sympify(exprCadena)

            funcionlamdified = sp.lambdify(x, expr, 'numpy')
            primerad = sp.lambdify(x, sp.diff(expr, x), 'numpy')  # primera derivada

            # Para detectar logaritmos
            if 'log' in exprCadena or 'ln' in exprCadena:
                valoresx = np.linspace(0.01, 2, 10000)
            else:
                valoresx = np.linspace(-10, 10, 4000)

            # Evaluacción f(x)
            valoresy = funcionlamdified(valoresx)

            # Filtrado de valores válidos
            mascara_valida = np.isfinite(valoresy)
            valoresx = valoresx[mascara_valida]
            valoresy = valoresy[mascara_valida]

            ventanas = []
            for i in range(len(valoresx) - 1):
                if valoresy[i] == 0:
                    ventanas.append(valoresx[i])
                elif valoresy[i] * valoresy[i + 1] < 0:
                    # Punto inicial o medio como aproximación inicial
                    x0 = (valoresx[i] + valoresx[i+1]) / 2
                    try:
                        raiz = opt.newton(funcionlamdified, x0, fprime=primerad, tol=1e-10, maxiter=50)
                        ventanas.append(raiz)
```

```
                except Exception:
                    # Si Newton falla, usar simplemente el punto medio
                    ventanas.append(x0)

        # Revisamos el último valor
        if valoresy[-1] == 0:
            ventanas.append(valoresx[-1])

        # Eliminar raíces muy cercanas
        ventanas_filtradas = []
        tolerancia = 1e-5
        for r in ventanas:
            if not any(abs(r - existente) < tolerancia for existente in ventanas_filtradas):
                ventanas_filtradas.append(r)

        if ventanas_filtradas:
            texto_raices = "Raíces encontradas:\n"
            for idx, raiz in enumerate(ventanas_filtradas, start=1):
                texto_raices += f" {raiz:.6f}  "

            ventanaLabel.config(text=texto_raices.strip(), font=("Arial", 12, "bold"))

        else:
            ventanaLabel.config(text="No se encontraron raíces")

        muestraPlot(expr, ventanas_filtradas)

    except Exception as e:
        ventanaLabel.config(text=f"Error: Ingrese la ecuación correctamente", font=("Arial"),fg="red")

def borrarGrafico():
    entry_function.delete(0, tk.END)
    ventanaLabel.config(text="")
    infoLabel.config(text="")
    for widget in canvasFrame.winfo_children():
        widget.destroy()

# Botón Encontrar raíces
tk.Button(
    frameContenido, text="Encontrar Raíces", command=newtonRaices,
    fg="white", bg="orange", font=("Arial", 12, "bold"),
    height=2, width=20
).pack(side="left", padx=50, pady = 20)

# Botón Borrar grafico
tk.Button(
    frameContenido, text="Borrar Grafico", command=borrarGrafico,
    fg="white", bg="orange", font=("Arial", 12, "bold"),
    height=2, width=20
).pack(side="left", padx=50, pady = 20)

def muestraPlot(exprFunc, ventanas):
    x = sp.symbols('x')
    funcionlamdified = sp.lambdify(x, exprFunc, 'numpy')

    valoresx = np.linspace(-10, 10, 400)
    valoresx_validos = valoresx[valoresx > 0]   # positivo
    valoresy = funcionlamdified(valoresx)

    fig, eje = plt.subplots(figsize=(10, 8))
    eje.plot(valoresx, valoresy, label=f'f(x) = {exprFunc}')

    eje.set_xticks(np.arange(-10, 11, 1))
    eje.set_yticks(np.arange(-10, 11, 1))

    # Color de ejes
    eje.axhline(0, color='darkgray', linewidth=1.5)
    eje.axvline(0, color='darkgray', linewidth=1.5)
    #Color de etiqueta en el eje
    eje.set_xlabel("Eje X", color ='blue')
    eje.set_ylabel("Eje Y", color ='blue')
    #Color de fondo
    fig.patch.set_facecolor("whitesmoke")
    eje.set_facecolor("whitesmoke")
    #Color de numeros en los ejes
```

```
    eje.tick_params(axis='x', colors='black')
    eje.tick_params(axis='y', colors='black')

    for ventana in ventanas:
        eje.axvline(ventana, color='red', linestyle='--')
        eje.plot(ventana, 0, 'ro', markersize=8)

    eje.set_xlim([-10, 10])
    eje.set_ylim([-10, 10])
    eje.grid(True, linestyle='-', linewidth=0.5)
    #eje.grid(True)
    eje.legend()

    for widget in canvasFrame.winfo_children():
        widget.destroy()

    canvas = FigureCanvasTkAgg(fig, master=canvasFrame)
    canvas.draw()
    canvas.get_tk_widget().pack()

def limpiarFrame():
    for widget in frameContenido.winfo_children():
        widget.destroy()

def Presentacion():
    global frameContenido, ventanaLabel, infoLabel, canvasFrame
    limpiarFrame()
    for widget in canvasFrame.winfo_children():
        widget.destroy()

    ventanaLabel.config(text="")
    infoLabel.config(text="")

    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="BIENVENIDOS A LA APLICACIÓN", font=("Comic Sans MS", 18, "bold"),
             fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="RAÍCES DE FUNCIONES NO LINEALES",
             font=("Arial", 14, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()

    try:
        image = Image.open("RootFinder.png")
        image = image.resize((550, 400))
        image_tk = ImageTk.PhotoImage(image)
        imageLabel = tk.Label(frameContenido, image=image_tk, bg="whitesmoke")
        imageLabel.image = image_tk
        imageLabel.pack(pady=20)
    except Exception as e:
        print(f"Error al cargar la imagen: {e}")

    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="", font=("Arial", 12, "bold"), fg="#00008B", bg="whitesmoke").pack()
    tk.Label(frameContenido, text="RootFinder v1.0", font=("Comic Sans MS", 20, "bold"),
             fg="#00008B", bg="whitesmoke").pack()

def Ayuda():
    limpiarFrame()
    for widget in canvasFrame.winfo_children():
        widget.destroy()

    ventanaLabel.config(text="")
    infoLabel.config(text="")

    tk.Label(frameContenido, text="\n\nINSTRUCCIONES DE USO", bg="whitesmoke",
             font=("Comic Sans MS", 14,"bold"), fg="#00008B").pack(pady=20)
    tk.Label(frameContenido, text="RootFinder v1.0:\n\n", bg="whitesmoke",
             font=("Comic Sans MS", 14,"bold"), fg="#00008B").pack(pady=20)
    tk.Label(frameContenido, text="\n1. Ingresar la función en el cuadro correspondiente"
             f"\nEjemplos de Funciones validas\nx**3 - 2*x + 1\nsin(x) - x/2\n"
             f"2. Hacer un click en el boton 'Encontrar raíces'"
             f"\n3. Visualización gráfica\n4. Hacer un click en el botón Borrar gráfico",bg="whitesmoke",
             font=("Comic Sans MS", 14)).pack(pady=20)
```

```
225
226 def Acercade():
227     limpiarFrame()
228     for widget in canvasFrame.winfo_children():
229         widget.destroy()
230
231     ventanaLabel.config(text="")
232     infoLabel.config(text="")
233
234     tk.Label(frameContenido, text="\n\n\n\nCalcula raíces de funciones no lineales."
235             f"\nRootFinder v1.0", bg="whitesmoke", font=("Comic Sans MS", 14,"bold"), fg="#00008B").pack(pady=20)
236
237     textos = [
238         "\nDesarrollado por: Leonel Coyla Idme",
239         "Alfredo Mamani Canqui",
240         "Elqui Yeye Pari Condori",
241         "Juan Reynaldo Paredes Quispe",
242         "José Pánfilo Tito Lipa",
243     ]
244
245     for texto in textos:
246         tk.Label(frameContenido, text=texto, bg="whitesmoke", font=("Comic Sans MS", 14)).pack(pady=4)
247
248     labelAcercade = tk.Label(frameContenido, text= "\n\nLanzamiento : 29 de abril  2025",
249                              font=("Comic Sans MS", 14),fg="#003366",bg="whitesmoke")
250     labelAcercade.pack(pady=(1,10))
251     labelAcercade = tk.Label(frameContenido, text= "Contacto: lcoyla@unap.edu.pe",
252                              font=("Comic Sans MSI", 14),fg="#003366",bg="whitesmoke")
253     labelAcercade.pack(pady=(1,10))
254
255 def crearInterfaz():
256     global frameContenido, ventanaLabel, infoLabel, canvasFrame
257
258     ventana = tk.Tk()
259     ventana.title("Busca raíces")
260     ventana.geometry("1000x1000")
261     ventana.config(bg="whitesmoke")
262
263     menuBar = tk.Menu(ventana)
264     ventana.config(menu=menuBar)
265
266     menuBar.add_command(label="Presentación", command=Presentacion)
267     menuBar.add_command(label="Encontrar raíces", command=encontrarRaices)
268     menuBar.add_command(label="Ayuda", command=Ayuda)
269     menuBar.add_command(label="Acerca de", command=Acercade)
270
271     mainFrame = tk.Frame(ventana, bg="whitesmoke")
272     mainFrame.pack(pady=10)
273
274     frameContenido = tk.Frame(mainFrame, bg="whitesmoke")
275     frameContenido.pack()
276
277     ventanaLabel = tk.Label(mainFrame, text="", bg="whitesmoke")
278     ventanaLabel.pack()
279
280     infoLabel = tk.Label(mainFrame, text="", fg="blue", bg="whitesmoke")
281     infoLabel.pack()
282
283     canvasFrame = tk.Frame(mainFrame, bg="whitesmoke")
284     canvasFrame.pack()
285
286     encontrarRaices()
287
288     ventana.mainloop()
289
290 def main():
291     crearInterfaz()
292
293 if __name__ == "__main__":
294     main()
295
```