UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

GRAFICADOR DE FUNCIONES MATEMÁTICAS

CURSO: MÉTODOS OPTIMIZACION

DOCENTE:

ING. Fred Torres Cruz

PRESENTADO POR:

Edilfonso Muñoz Anccori

SEMESTRE: V NIV

PUNO-PERÚ
2025

Introducción

El desarrollo de herramientas computacionales para la visualización y análisis de funciones matemáticas es un aspecto fundamental en diversas áreas de las ciencias, la ingeniería y la educación. Estas herramientas permiten explorar de manera interactiva conceptos clave como variables, funciones y restricciones, esenciales para la resolución de problemas matemáticos y de optimización.

En este contexto, se presenta el desarrollo de un graficador de funciones matemáticas. Este programa facilita la comprensión visual y analítica de funciones de una sola variable, permitiendo:

- **Definir variables:** Trabajar con la variable independiente de la función.
- **Establecer funciones:** Ingresar expresiones matemáticas para describir fenómenos o relaciones.
- **Gestionar restricciones:** Configurar los límites del intervalo de graficación.

El tema central de este trabajo, **"Variables, funciones y restricciones"**, se desarrolla mediante un programa en Python que combina bibliotecas como `SymPy` y `Matplotlib`. Estas herramientas permiten al usuario interactuar con los datos de entrada, procesarlos y generar gráficos que muestran el comportamiento de la función en un intervalo definido.

Este proyecto tiene aplicaciones en diversas áreas:

- **Educación:** Ayuda a los estudiantes a visualizar conceptos como continuidad, derivadas y puntos críticos.
- **Ciencia e ingeniería:** Permite representar modelos matemáticos de manera gráfica.
- **Optimización:** Facilita la representación gráfica de restricciones en problemas matemáticos.

El objetivo principal del proyecto es proporcionar una herramienta versátil e intuitiva que combine facilidad de uso con funcionalidades avanzadas, haciendo de este programa una excelente opción para estudiantes y profesionales.

Desarrollo

El programa sigue un enfoque modular y está compuesto por las siguientes características principales:

1. **Entrada de Datos:** Permite al usuario ingresar la función matemática, la variable independiente y los límites del intervalo.
2. **Procesamiento de Datos:** Convierte la función simbólica ingresada en una función evaluable numéricamente.
3. **Visualización:** Genera la gráfica correspondiente en el rango indicado, mostrando etiquetas en los ejes y un título descriptivo.

El programa utiliza tecnologías modernas, tales como:

- **Python:** Lenguaje de programación principal.

- **SymPy:** Para el manejo simbólico de expresiones matemáticas.
- **Matplotlib:** Para la generación de gráficos.
- **Tkinter:** Para la interfaz gráfica.

Resultados

Al ejecutar el programa, el usuario puede interactuar con la interfaz gráfica para ingresar datos. Por ejemplo:

- **Función:** $f(x) = x^2$
- **Variable:** x
- **Intervalo:** $[-5, 5]$

El gráfico generado incluye:

- La curva de la función en el intervalo indicado.
- Ejes claramente etiquetados.
- Líneas de referencia para los ejes x y y.
- Un título descriptivo del gráfico.

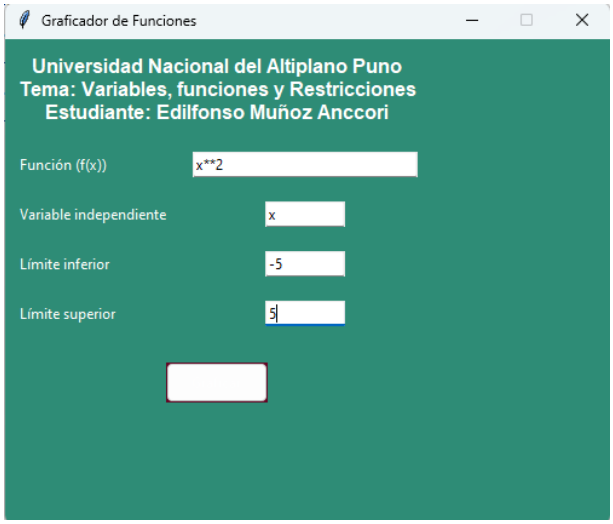

Figura 1: El usuario tiene que ingresar los datos

El gráfico generado incluye:

- La curva de la función en el intervalo indicado.
- Ejes claramente etiquetados.
- Líneas de referencia para los ejes x y y.
- Un título descriptivo del gráfico.

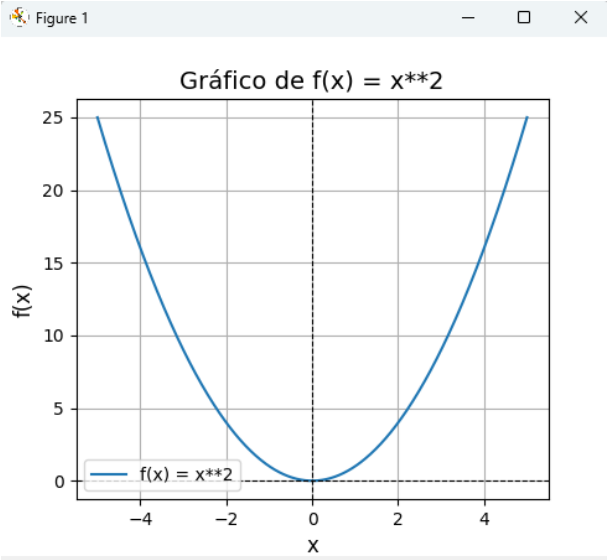

Figura 2: Haz clic en "GRAFICARz muestra el gráfico

Código Fuente

A continuación, se presenta el código fuente del programa:

Listing 1: Graficador de funciones matemáticas

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols, lambdify
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showerror

def graficar_funcion():
    try:
        # Obtener los valores de los campos de entrada
        funcion = entrada_funcion.get()
        variable = entrada_variable.get()
        min_intervalo = float(entrada_min_intervalo.get())
        max_intervalo = float(entrada_max_intervalo.get())

        # Crear la funci n simb lica y convertirla en funci n
           num rica
        x = symbols(variable)
        funcion_simbolica = eval(funcion)
        funcion_numerica = lambdify(x, funcion_simbolica, "numpy"
           )

        # Crear el intervalo de valores
        x_vals = np.linspace(min_intervalo, max_intervalo, 500)
        y_vals = funcion_numerica(x_vals)

        # Graficar
        plt.figure(figsize=(8, 6))
```

```
        plt.plot(x_vals, y_vals, label=f"f({variable}) = {funcion}")
        plt.title(f"Gr fico de f({variable}) = {funcion}", fontsize=14)
        plt.xlabel(variable, fontsize=12)
        plt.ylabel(f"f({variable})", fontsize=12)
        plt.axhline(0, color='black', linewidth=0.8, linestyle="--")
        plt.axvline(0, color='black', linewidth=0.8, linestyle="--")
        plt.legend()
        plt.grid(True)
        plt.show()

    except Exception as e:
        showerror("Error", f"Ha ocurrido un error: {e}")

# Crear la interfaz gr fica
ventana = tk.Tk()
ventana.title("Graficador de Funciones")
ventana.geometry("500x400")
ventana.resizable(False, False)

# Configuraci n de colores
ventana.configure(bg="#2e8c76")  # Fondo de la ventana
estilo = ttk.Style()
estilo.configure("TLabel", background="#2e8c76", foreground="white")  # Colores de las etiquetas
estilo.configure("TButton", background="#630624", foreground="white", padding=5)

# Secci n de encabezado
encabezado = tk.Label(
    ventana,
    text=(
        "Universidad Nacional del Altiplano Puno\n"
        "Tema: Variables, funciones y Restricciones\n"
        "Estudiante: Edilfonso Mu oz Anccori"
    ),
    bg="#2e8c76",
    fg="white",
    font=("Arial", 12, "bold"),
    justify="center"
)
encabezado.grid(row=0, column=0, columnspan=2, pady=10)

# Etiquetas y campos de entrada
ttk.Label(ventana, text="Funci n (f(x))").grid(row=1, column=0, padx=10, pady=10, sticky="w")
entrada_funcion = ttk.Entry(ventana, width=30)
entrada_funcion.grid(row=1, column=1, padx=10, pady=10)
```

```
71
72 ttk.Label(ventana, text="Variable independiente").grid(row=2,
      column=0, padx=10, pady=10, sticky="w")
73 entrada_variable = ttk.Entry(ventana, width=10)
74 entrada_variable.grid(row=2, column=1, padx=10, pady=10)
75
76 ttk.Label(ventana, text="L mite inferior").grid(row=3, column=0,
       padx=10, pady=10, sticky="w")
77 entrada_min_intervalo = ttk.Entry(ventana, width=10)
78 entrada_min_intervalo.grid(row=3, column=1, padx=10, pady=10)
79
80 ttk.Label(ventana, text="L mite superior").grid(row=4, column=0,
       padx=10, pady=10, sticky="w")
81 entrada_max_intervalo = ttk.Entry(ventana, width=10)
82 entrada_max_intervalo.grid(row=4, column=1, padx=10, pady=10)
83
84 # Bot n para graficar
85 boton_graficar = ttk.Button(ventana, text="Graficar", command=
      graficar_funcion)
86 boton_graficar.grid(row=5, column=0, columnspan=2, pady=20)
87
88 ventana.mainloop()
89 % Conclusi n
90 \section*{Codigo QR github}
```

Figura 3: Codigo QR github

https://github.com/edilfon/Variables-funciones-y-Restricciones-/tree/main

Conclusión

El desarrollo del graficador de funciones matemáticas ha permitido demostrar cómo las herramientas computacionales pueden facilitar la comprensión y el análisis de conceptos clave en matemáticas, como variables, funciones y restricciones. Este programa no solo cumple con los objetivos planteados, sino que también se posiciona como una herramienta versátil y aplicable en distintos contextos académicos y profesionales.

El proyecto destaca por su facilidad de uso, brindando a los usuarios una interfaz intuitiva que simplifica tareas complejas como la graficación de funciones simbólicas y la gestión de intervalos. Además, su implementación en Python garantiza la robustez del sistema, al apoyarse en bibliotecas de alto rendimiento como SymPy y Matplotlib.

En el ámbito educativo, este programa tiene el potencial de ser utilizado como una herramienta didáctica para estudiantes, ayudándolos a visualizar de forma interactiva el comportamiento de funciones matemáticas, identificar características como máximos, mínimos, puntos de inflexión, y explorar el impacto de las restricciones sobre el dominio de las funciones.

En el contexto profesional y de investigación, el graficador puede ser aplicado en disciplinas como la física, la ingeniería y la economía, donde la representación gráfica de modelos matemáticos es esencial para el análisis de sistemas y la toma de decisiones.

- Analizar visualmente el comportamiento de las funciones.
- Experimentar con diferentes funciones e intervalos.
- Representar gráficamente restricciones y límites en problemas matemáticos.

Se recomienda ampliar este proyecto en el futuro, incorporando soporte para funciones de varias variables, gráficos en 3D y análisis interactivo.

Referencias

[1] Van Rossum, G. (1995). *Python: A programming language for software and applications development*. Computer Science Institute, Amsterdam.

[2] Meurer, A., et al. (2017). *SymPy: Symbolic computation in Python*. Journal of Open Source Software, 6(58), 929.

[3] Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science Engineering, 9(3), 90–95.

[4] Grayson, J. (2000). *Tkinter: Python's de facto standard GUI library*. Python Software Foundation.

[5] Burden, R. L., Faires, J. D. (2010). *Numerical Methods for Engineers*. 7th Edition. Cengage Learning.

[6] Keller, H. B. (2006). *Numerical Methods for Engineers and Scientists*. CRC Press.

[7] Stewart, J. (2008). *Calculus: Early Transcendentals*. 6th Edition. Brooks/Cole.

[8] Nocedal, J., Wright, S. J. (2006). *Numerical Optimization*. 2nd Edition. Springer.