Instituto Politécnico Nacional

Escuela Superior de Cómputo

Análisis Vectorial

Calculadora de integrales dobles y triples (numérica vs exacta)

Profesor: Dr. David Correa Coyac

Integrantes: Bonilla Hernandez Ximena Sofia

Castillo Vidal Carmen Andrea

Cruz Rodríguez Bruno Aron

Introducción y motivación del proyecto

La resolución de integrales múltiples, tanto dobles como triples, es uno de los mayores desafíos dentro del aprendizaje. Esta dificultad no solo radica en la abstracción matemática necesaria para comprender las regiones de integración, sino también en la extrema complejidad analítica y el elevado tiempo de ejecución que se requiere para llegar a resultados precisos de forma manual. En la práctica, un solo error en un paso intermedio de una integral triple puede invalidar horas de trabajo, lo que convierte a este proceso en un cuello de botella para estudiantes y profesionales.

Considerando que las integrales son una herramienta fundamental e indispensable para la ingeniería se vuelve una prioridad tecnológica el poder optimizar estos procesos. No se trata solo de obtener un número, sino de agilizar los tiempos de respuesta en el diseño y análisis de proyectos, donde la eficiencia es un factor crítico.

Modelado matemático: explicación de los conceptos de Análisis Vectorial utilizados.

Integrales dobles: son una extensión de las integrales simples para funciones de dos variables sobre una región bidimensional, usándose principalmente para calcular el volumen bajo una superficie o el área de una región plana

Integrales triples: una extensión a tres dimensiones de las integrales simples y dobles, que permiten sumar infinitas cantidades infinitesimales sobre una región tridimensional para calcular propiedades como volumen, masa, centro de masa y momentos de inercia de sólidos

Factor de escala jacobiano: indica cómo cambian las áreas o volúmenes bajo una transformación, midiendo el estiramiento o compresión de las coordenadas en un punto

Coordenadas Polares: representan puntos mediante una distancia radial (r) y un ángulo (θ) desde un origen (polo)

Coordenadas cilíndricas: escriben puntos en el espacio usando el radio desde el eje z (ρ), un ángulo azimutal (ϕ) en el plano xy, y la altura (z).

Coordenadas esféricas: describen un punto en el espacio usando la distancia radial ρ desde el origen, un ángulo azimutal θ (similar a la longitud) y un ángulo polar ϕ (similar a la latitud, desde el eje Z positivo).

Cálculo Simbólico (Exacto): Utiliza la librería SymPy para realizar la integración analítica mediante reglas de integración fundamentales.

Sumas de Riemann (Numérico): Para la aproximación numérica, el código divide la región en una malla (usando np.meshgrid) y suma las contribuciones de cada pequeño elemento de área o volumen.

Coordenadas:

Cartesianas: Basadas en los ejes perpendiculares (x, y, z).

Curvilíneas: Maneja ángulos y radios como θ para rotaciones polares/cilíndricas, y ϕ junto con ρ para la geometría esférica.

Descripción de la implementación (arquitectura del programa, módulos, librerías emplea-das).

Arquitectura del programa:

Clases principales:

Clase CalculadoraIntegrales (se encarga de realizar los cálculos matemáticos del programa)

Clase app (gestiona toda la experiencia del usuario)

Librerias:

customtkinter: entorno grafico

sympy: logica simbolica

numpy: arreglos multidimensionales y calculos numericos rapidos

matplotlib: graficos 3d

PIL: recursos visuales

Funcionamiento clave:

Transformaciones Implícitas: Utiliza implicit_multiplication_application de la librería SymPy. Esto permite que si el usuario escribe 2x, el programa lo interprete correctamente como 2 * x.

Limpieza de Sintaxis: El código tiene una función dedicada a reemplazar caracteres comunes por sintaxis de Python (ej. ^ por **, sen por sin, tg por tan).

Variables Simbólicas: Define objetos sp.Symbol para x, y, z, r, theta, phi, rho. Esto permite que SymPy realice operaciones algebraicas como si fuera un humano resolviendo el problema en papel.

Coordenadas Polares/Cilíndricas: Si el usuario selecciona este modo, el código multiplica la función f por el factor r (Jacobiano).

Implementación: funcion_final = f_original * r

Coordenadas Esféricas: Es la implementación más compleja. El código aplica el Jacobiano y mapea las variables correspondientes.

Implementación: funcion_final = f_original * rho**2 * sp.sin(phi)

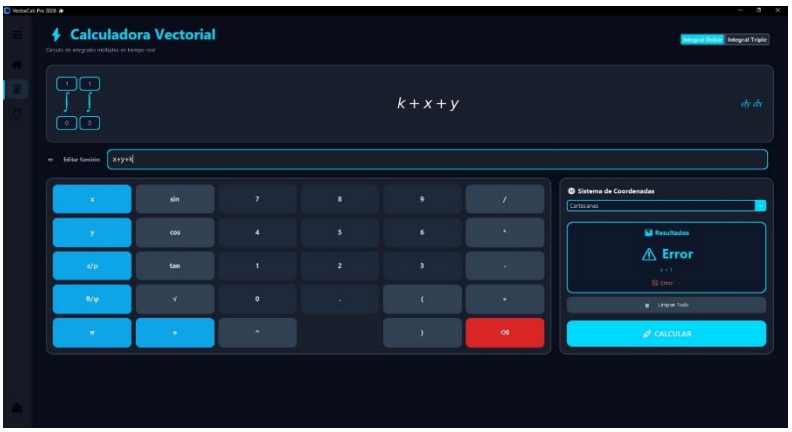
Cuando el usuario desea visualizar la región, el código utiliza plot_surface de Matplotlib.

Calcula los valores z = f (x, y) en un rango determinado por los límites de integración.

Aplica un mapa de colores para representar la magnitud de la función, ayudando a visualizar el volumen que se está integrando.

Capturas:

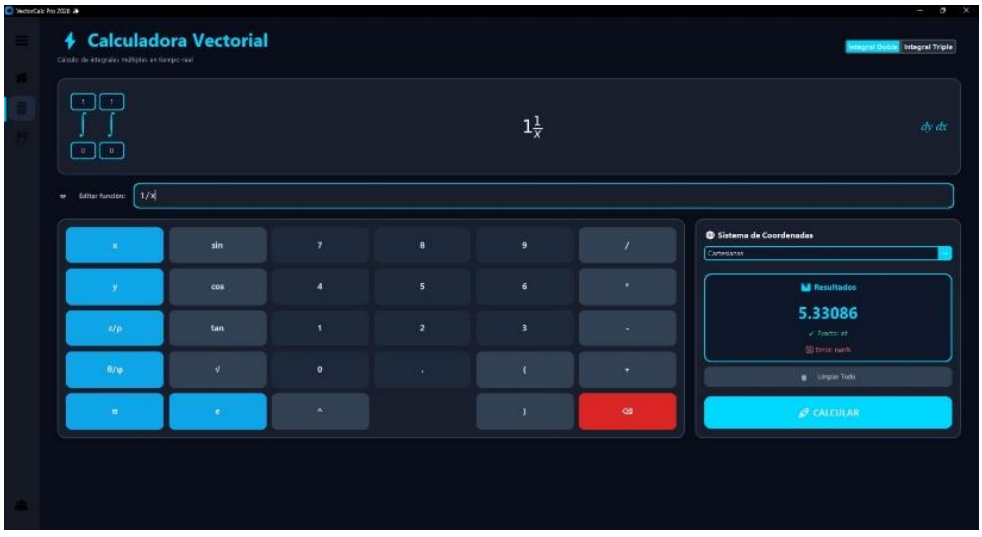
Ejemplo de error



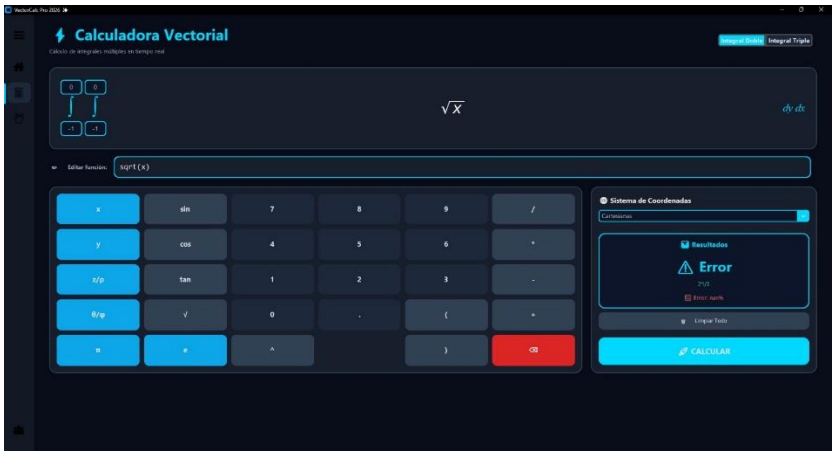
Caso Exitoso

Por qué: ∫01∫01(x2+y2)dydx=∫01(x2+1/3)dx=1/3+1/3=2/3.

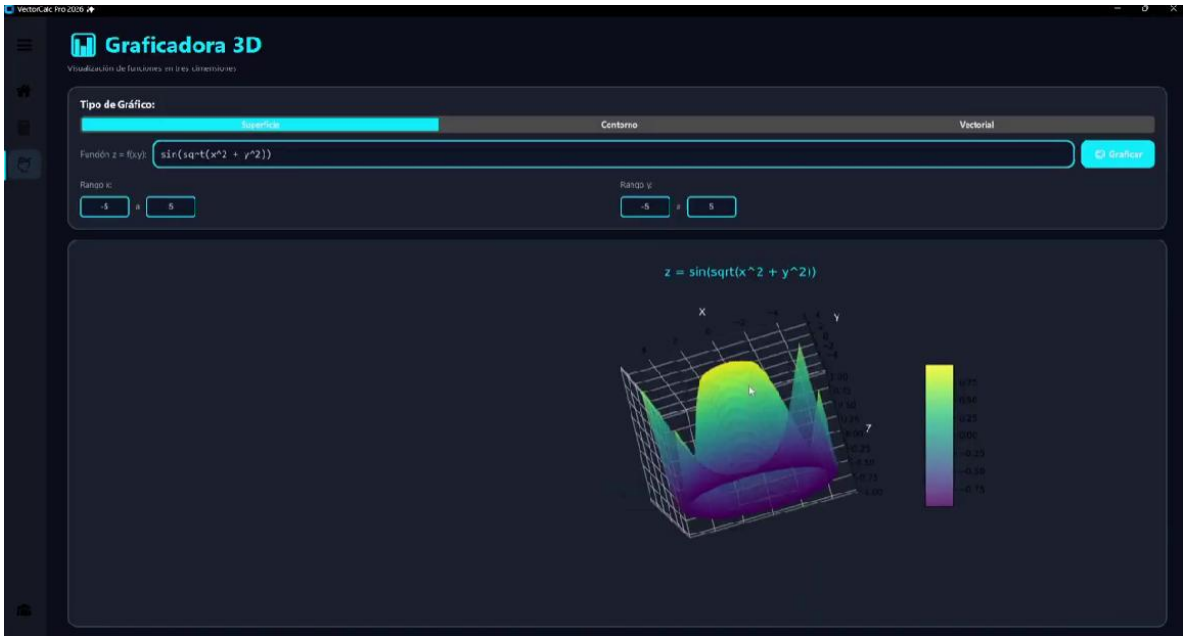
Con sumas de Riemman da un numero raro, pero en el exacto da infinito por lo cual está bien para 1/x



Para la raiz de x de un numero negativo se manejan numeros complejos 2*I/3 pero con sumas de riemman sale error



Grafica



Conclusiones

En conclusión, la principal ventaja de este proyecto es que resuelve automáticamente los pasos más difíciles, como los cambios a coordenadas polares o esféricas mediante el uso de Jacobianos, y ofrece una doble verificación al comparar un resultado exacto con una aproximación numérica. Además, gracias a que muestra las fórmulas en formato profesional y permite visualizar las gráficas en 3D, transforma un proceso que suele ser tedioso y propenso a errores en una experiencia visual, rápida y segura para cualquier estudiante o profesional.