Práctica 8: Ejercicios de Integración Numérica

José Josef Medellin Reséndez

6 de abril de 2025

Introducción

El método de cálculo de Romberg es una técnica numérica basada en la regla del trapecio y un proceso llamado extrapolación de Richardson. Esta técnica permite mejorar la precisión en la estimación de integrales definidas utilizando una secuencia de aproximaciones cada vez más precisas, sin requerir un número excesivo de evaluaciones de la función.

Gracias a su estructura iterativa y uso de valores previos, el método logra una rápida convergencia, siendo una herramienta eficiente dentro del análisis numérico.

Desarrollo de la práctica

Proceso paso a paso

Importación de módulos:
Cargue dos bibliotecas: numpy, para trabajar con arreglos numericos y operaciones matematicas eficientes, y math, que ofrece funciones matematicas como sin, cos, pi, etc.

Definición de función:
Esta funcion evalua una expresion matematica que puedo ingresar como texto. Use eval para interpretar ese texto como codigo Python, permitiendo usar variables como x, y funciones de math o numpy.

Evaluación de límites:
Hay otra funcion que tambien usa eval, pero esta vez para convertir los limites de integracion escritos como texto (por ejemplo, math.pi/4) en valores numericos reales.

Función del método de Romberg:

```
def romberg(f, a, b, n, tol=None):
    R = [[0] * (n + 1) for _ in range(n + 1)]
    h = b - a
    R[0][0] = (f(a) + f(b)) * h / 2
```

```
    for i in range(1, n + 1):
        h /= 2
        sum_f = sum(f(a + (2 * k - 1) * h) for k in range(1, 2 ** (i
            - 1) + 1))
        R[i][0] = 0.5 * R[i - 1][0] + h * sum_f

        for j in range(1, i + 1):
            R[i][j] = R[i][j - 1] + (R[i][j - 1] - R[i - 1][j - 1])
               / (4 ** j - 1)

        if tol and i > 0 and abs(R[i][i] - R[i - 1][i - 1]) < tol:
            return R[i][i]
    return R[n][n]
```

Función principal:

```
def main():
    import math
    expr = input("Ingresa la funci n a integrar (en t rminos de x)
       : ")
    a = eval(input("L mite inferior: "))
    b = eval(input("L mite superior: "))
    n = int(input("N mero de iteraciones: "))
    tol = input("Tolerancia (opcional): ")
    tol = float(tol) if tol else None

    f = lambda x: eval(expr, {"x": x, "math": math, "np": __import__
       ("numpy")})
    result = romberg(f, a, b, n, tol)
    print(f"Resultado: {result}")
```

Ejecución del programa:

```
if __name__ == "__main__":
    main()
```

Conclusión

El método de Romberg me permitió resolver integrales con una gran presición, combinando simpleza y potencia matemática. La implementación en Python, utilizando `numpy` y `math`, facilitó enormemente el manejo de funciones, límites y la estructura iterativa del método. Fue una buena practíca, además que fue de gran ayuda para comprender métodos numéricos aplicados al cálculo.