Facultad de Ciencias Físico Matemáticas (2023), **Febrero**, 1–6

REPORTE DE LABORATORIO COMPUTACIONAL

Aplicación de la regla del trapecio para integrales dobles en un problema físico.

Jesús Eduardo Loera Casas*

Facultad de Ciencias Físico Matemáticas, San Nicolás de los Garzas, Nuevo León, México
*Corresponding author. Email: colab.git@gmail.com

Abstract

En este reporte implementamos la regla del trapecio para el cálculo de intregrales en los lenguajes de Fortran y Python. El método funciona para resolver integrales definidas en regiones rectangulares para funciones de dos variables reales. Una vez implementado el método, este se aplicó para resolver la ecuación del cohete, donde se mostró que el método de extrapolación de Richardson es eficiente para iterar los resultados numéricos de obtenidos con el método de Romberg y mejorar la precisión de las integrales calculadas.

Keywords: Integración numérica, Regla del trapecio, Integrales dobles, Fortran, Python

1. Introducción

En prácticamente todas las ramas de la física la integración tiene un papel muy importante, conocemos muchas magnitudes físicas cuya definición implica una integral, particularmente el flujo eléctrico, el flujo magnético y la normalización de la función de onda bidimensional son ejemplos de definiciones que implican una integral doble.

$$\langle \psi | \psi \rangle = \int_S \psi^* \psi \, dA$$

$$\Phi_B = \int_S \vec{B} \cdot d\vec{A} \qquad \Phi_E = \int_S \vec{E} \cdot d\vec{A}$$

El cálculo de dichas integrales se puede volver complicado, por ello solemos recurrir a técnicas de integración numérica para resolver el problema de manera computacional.

2. La regla del trapecio

La regla del trapecio (Nakamura 1993) es un método de integración numérica que permite calcular el valor aproximado de las integrales de la forma

$$I = \int_a^b f(x)dx$$

donde f(x) es una función escalar de una variable real X.

El discretizar la región de integración $[a, b]$ en N subintervalos de igual tamaño h delimitados por los puntos $\{x_0, x_1, x_2, ..., x_{N-1}, x_N\}$ donde $x_{i+1} > x_i$, $x_0 = a$ y $x_N = B$, se puede aproximar el valor de la integral de la siguiente forma:

$$\int_a^b f(x)dx = \frac{h}{2}\left\{f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i)\right\} \tag{1}$$

$$h = \frac{b-a}{N} \tag{2}$$

Es decir, podemos escribir

$$\int_a^b f(x)dx = \frac{h}{2}\left\{f(a) + f(b) + 2\sum_{i=1}^{N-1} f(a + i\frac{b-a}{N})\right\} \tag{3}$$

3. Integración doble mediante la regla del trapecio

Consideremos una integral doble definida I sobre una región rectangular.

$$I = \int_a^b \int_c^d f(x,y)dydx \tag{4}$$

Podemos definir a la función G como sigue

$$G(x) \equiv \int_c^d f(x,y)dy \tag{5}$$

De esta manera, podemos reescribir la integral I como a continuación.

$$I = \int_a^b G(x)dx \tag{6}$$

Observe que de esta forma I puede ser calculada con la regla del trapecio ordinaria.

$$I = \int_a^b G(x)dx = \frac{h}{2}\left\{G(a) + G(b) + 2\sum_{i=1}^{N-1} G(a + i\frac{b-a}{N})\right\} \tag{7}$$

Donde, empleando la definición de G nos queda la siguiente ecuación.

$$I = \frac{h}{2}\left\{\int_c^d f(a,y)dy + \int_c^d f(b,y)dy + 2\sum_{i=1}^{N-1} N \int_c^d f(a + i\frac{b-a}{N}, y)dy\right\} \tag{8}$$

Observe que $G(x_i)$ también puede ser cálculado fácilmente por la regla del trapecio.

$$G(x_i) = \frac{h}{2}\left\{f(x_i, c) + f(x_i, d) + 2\sum_{i=1}^{N-1} f(x_i, c + i\frac{d-c}{N})\right\} \tag{9}$$

4. Resolver la ecuación del Cohete

Problema. The upward velocity of a rocket can be computed by the following formula:

$$v = uln\left(\frac{m_o}{m_o - qt}\right) - gt$$

where v = upward velocity, u = velocity at which fuel is expelled relative to the rocket, m_o = initial mass of the rocket at time $t = 0$, q = fuel consumption rate, and g = downward acceleration of gravity (assumed constant = $9.8m/s^2$). If $u = 1800m/s$, $m_o = 160000kg$ and $q = 2500kg/s$, use six-segment trapezoidal and Simpson's 1/3 rule, six-point Gauss quadrature, and $O(h^8)$ Romberg mehtods to determine how high the rocket will fly in 30 s.

- Solución con Python

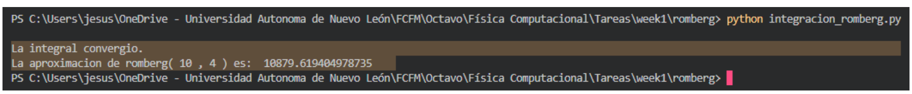

Figure 1. Se ejecutó el script de Python que resuelve el problema físico.

- Solución con Fortran

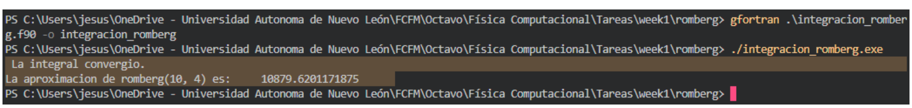

Figure 2. Se ejecutó el script de Python que resuelve el problema físico.

Vea una comparación de las respuestas obtenidas en cada script en la tabla 1.

Table 1. Se resolvió el problema de la altura alcanzada por el cohete con el script en Python y Fortran.

	Altura alcanzada en 30 segundos
Python	10879.61940497 metros
Fortran	10879.62011718 metros

5. Conclusión

Podemos observar que en ambos scripts donde se implementa el método de Romberg (en Python y Fortran), al calcular el valor de la integral numérica se converge a un resultado de manera rápida antes de alcanzar a calcular el valor de $R_{10,4}$ e incluso ambos valores calculados por los distintos programas son muy similares. De manera que queda en evidencia la eficiencia y la rapidez de este método numérico para resolver integrales de manera óptima.

References

Nakamura, Shoichiro. 1993. *Applied numerical methods in c.* Prentice-Hall, Inc.

Appendix 1. Programa en Python

```python
# Programa elaborado por Jesus Eduardo Loera Casas
# Elaborado el dia 13/02/23

import numpy as np


"""
Parametros de integracion
"""

# Definimos el integrando
def f(x, y):
    return np.log(x+2*y)

# definimos la region de integracion
xo = 1.4 ; xf = 2.0
yo = 1.0 ; yf = 1.5

# definimos el tamano de subintervalo de integracion
dx = 0.0001 ; dy = 0.0001

"""
Funciones que definen el metodo de integracion
"""

def trapecio_y(yo, yf, dy, xcte):
    N = int((yf-yo)/dy + 1)
    suma = 0.0
    for i in range(1, N):
        suma = suma + f(xcte, yo+i*dy)
    trapecio_y = (0.5*dy)*(f(xcte,yo) + f(xcte,yf) + 2*suma)
    return trapecio_y

def integracion_doble(xo, xf, yo, yf, dx, dy):
    Nx = int((xf-xo)/dx + 1)
    Ny = int((yf-yo)/dy + 1)
    suma = 0.0
    for i in range(1, Nx):
        suma = suma + trapecio_y(yo, yf, dy, xo + i*dx)
    aux = trapecio_y(yo,yf,dy,xo) + trapecio_y(yo,yf,dy,xf)
    integral = (0.5*dx)*(aux + 2*suma)
    return integral

"""
Mandamos a llamar al metodo
"""

integral = integracion_doble(xo, xf, yo, yf, dx, dy)
print("El valor de la integral es: ", integral)
```

Appendix 2. Programa en Fortran

```
! Programa elaborado por Jesus Eduardo Loera Casas
! Fecha 09/02/23

! En este programa resolvemos integrales dobles con la regla
! del trapecio.

        ! definimos el integrandpo
        real function f(x,y)
            implicit none
            real :: x,y
            f = log(x+2*y)
            return
        end function

        ! programa principal
        program main

            implicit none
            real :: integral, dx, dy, xo, xf, yo, yf

            ! definimos la region de integracion
            xo = 1.4 ; xf = 2.0
            yo = 1.0 ; yf = 1.5

            ! definimos el tamano de subintervalo de integracion
            dx = 0.0001 ; dy = 0.0001

            call integracion_doble(xo, xf, yo, yf, dx, dy, integral)

            write(*,*) "El valor de la integral es: ", integral

        end program main

        ! subrutina que calcula la integral doble con regla del
             trapecio
        subroutine integracion_doble(xo, xf, yo, yf, dx, dy,
             integral)
            implicit none
            real :: xo, xf, yo, yf, dx, dy, integral, suma
            real :: f
            real :: trapecio_y, aux
            integer :: Nx, Ny, i
            Nx = (xf-xo)/dx + 1
            Ny = (yf-yo)/dy + 1
            suma = 0.0
            do i = 1, Nx-1
                suma = suma + trapecio_y(yo, yf, dy, xo + i*dx)
            end do
            aux = trapecio_y(yo,yf,dy,xo) + trapecio_y(yo,yf,dy,xf)
            integral = (0.5*dx)*(aux + 2*suma)
        end subroutine

        ! regla del trapecio para integrar funciones f(cte,y)
        real function trapecio_y(yo, yf, dy, xcte)
            implicit none
            real :: yo, yf, dy, suma
            ! funcion del integrando
            real :: f, xcte
            integer :: N, i

            N = (yf-yo)/dy + 1
            suma = 0.0
            do i = 1, N-1
                suma = suma + f(xcte, yo+i*dy)
            end do
            trapecio_y = (0.5*dy)*(f(xcte,yo) + f(xcte,yf) + 2*suma)
```

```
    return
end function
```