Facultad de Ciencias Físico Matemáticas (2023), **Febrero**, 1–9

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.

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

Puede ver la implementación de este método numérico en Python en el apéndice "Programa integracion doble en Python" y su implementación en Fortran en el apéndice "Programa integracion doble en Fortran".

4. Resolver la ecuación del Cohete

Problema. Encuentra el campo $\vec{E}$ debido a un disco cargado de radio $2m$ con una carga total $Q = 5C$ a una distancia de $5m$ del disco colocado sobre el eje del disco.

Solución.

Conocemos la ley de Coulomb para distribuciones continuas de carga.

$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0}\int_S \frac{\sigma(\vec{r})\left(\vec{r}-\vec{r}'\right)}{\left\|\vec{r}-\vec{r}'\right\|} dA' \tag{10}$$

En nuestro caso

$$\sigma(\vec{r}') = \frac{Q}{\pi R^2} \quad , \quad \vec{r} = z_0\hat{k} \quad , \quad \vec{r}' = x\hat{i} + y\hat{j}$$

$$\vec{r}-\vec{r}' = -x\hat{i} - y\hat{j} + z_0\vec{k} \quad , \quad \left\|\vec{r}-\vec{r}'\right\| = \left(x^2 + y^2 + z_0^2\right)^{1/2}$$

Sustituyendo las ecuaciones anteriores en la ecuación número 10.

$$\vec{E}(\vec{r}) = \frac{\sigma}{4\pi\epsilon_0}\int_S \frac{-x\hat{i} - y\hat{j} + z_0\vec{k}}{\left(x^2 + y^2 + z_0^2\right)^{3/2}} dA' \tag{11}$$

Como estamos integrando sobre la superficie definida por el interior de la circunferencia de la ecuación $x^2 + y^2 = 4$, reescribimos a la integral de la siguiente forma.

$$\vec{E}(\vec{r}) = \frac{\sigma}{4\pi\epsilon_0}\int_{-2}^{2}\int_{-\sqrt{4-x^2}}^{\sqrt{4-x^2}} \frac{-x\hat{i} - y\hat{j} + z_0\vec{k}}{\left(x^2 + y^2 + z_0^2\right)^{3/2}} dydx \tag{12}$$

De esta manera, las componentes rectangulares del campo eléctrico estan dadas por:

$$\longrightarrow E_x(\vec{r}) = \int_{-2}^{2}\int_{-\sqrt{4-x^2}}^{\sqrt{4-x^2}} \frac{\sigma}{4\pi\epsilon_0}\frac{-x}{\left(x^2 + y^2 + z_0^2\right)^{3/2}} dydx \tag{13}$$

$$\longrightarrow E_y(\vec{r}) = \int_{-2}^{2}\int_{-\sqrt{4-x^2}}^{\sqrt{4-x^2}} \frac{\sigma}{4\pi\epsilon_0}\frac{-y}{\left(x^2 + y^2 + z_0^2\right)^{3/2}} dydx \tag{14}$$

$$\longrightarrow E_z(\vec{r}) = \int_{-2}^{2}\int_{-\sqrt{4-x^2}}^{\sqrt{4-x^2}} \frac{\sigma}{4\pi\epsilon_0}\frac{z_0}{\left(x^2+y^2+z_0^2\right)^{3/2}}dydx \tag{15}$$

Calculamos las tres ecuaciones anteriores con el programa escrito en el apéndice titulado "Programa del problema físico en Fortran".

Vea una comparación de las componentes del campo eléctrico obtenidas en con el programa en Fortran con la solución teórica en la tabla 1.

Table 1. Se utilizó $h_x = 0.0001$ y $h_y = 0.0001$ para la integración.

	Solución numérica N/C	Solución teórica N/C
E_x	-4.57552147	0
E_y	-7215.09961	0
E_z	1.60803021×10^9	1.6×10^9

5. Conclusión

Podemos observar que las componentes E_x y E_y, parecen diferir mucho del valor exacto cálculado teóricamente, sin embargo, dichas componentes comparadas con la E_z son muy peuqueñas.

$$\frac{E_x}{E_z} \approx 0 \quad y \quad \frac{E_y}{E_z} \approx 0 \tag{16}$$

Por ello podemos considerar que el método de integración sigue siendo muy útil al capturar la esencia del problema físico, es decir, las componentes E_x y E_y son despreciables comparadas con E_z. Además este método de integración resulta especialmente útil sobre regiones de integración complicadas de parametrizar.

References

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

Appendix 1. Programa de integración doble en 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.sin(x+y)

# Definimos el limite superior de integracion
def limit_sup(x):
```

```
    return 3 + np.exp(x/5.0)

# Definimos el limite superior de integracion
def limit_inf(x):
    return np.log(x)

# definimos la region de integracion
xo = 1.0 ; xf = 3.0

# definimos el tamano de subintervalo de integracion
dx = 0.0001 ; dy = 0.001

"""
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, dx, dy):
    Nx = int((xf-xo)/dx + 1)
    suma = 0.0
    for i in range(1, Nx):
        yo = limit_inf(xo)
        yf = limit_sup(xf)
        suma = suma + trapecio_y(yo, yf, dy, xo + i*dx)
    yo = limit_inf(xo)
    yf = limit_sup(xf)
    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, dx, dy)
print("El valor de la integral es: ", integral)
```

Appendix 2. Programa integracion doble 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 la funcion que limita la region
        ! de integracion superior
        real function limit_sup(x)
            implicit none
            real :: x
            limit_sup = 3.0 + exp(x/5.0)
            return
        end function

        ! definimos la funcion que limita la region
        ! de integracion inferior
        real function limit_inf(x)
            implicit none
            real :: x
```

```
    limit_inf = log(x)
    return
end function

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

! programa principal
program main

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

    ! definimos la region de integracion
    xo = 1.0 ; xf = 3.0


    ! definimos el tamano de subintervalo de integracion
    dx = 0.001 ; dy = 0.001

    call integracion_doble(xo, xf, 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, dx, dy, integral)
    implicit none
    real :: xo, xf, yo, yf, dx, dy, integral, suma
    real :: limit_inf, limit_sup
    real :: trapecio_y, aux
    integer :: Nx, i
    Nx = (xf-xo)/dx + 1
    suma = 0.0
    do i = 1, Nx-1
        yo =  limit_inf(xo + i*dx)
        yf = limit_sup(xo + i*dx)
        suma = suma + trapecio_y(yo, yf, dy, xo + i*dx)
    end do
    aux = trapecio_y(limit_inf(xo),limit_sup(xo),dy,xo)
    aux = aux + trapecio_y(limit_inf(xo),limit_sup(xo),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
```

Appendix 3. Programa del problema físico en Fortran

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

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

        ! programa principal
        program main

            implicit none
            real :: dx, dy, xo, xf
            real :: Ex, Ey, Ez

            ! definimos la region de integracion
            xo = -2.0 ; xf = +2.0

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

            call integracion_doble_fx(xo, xf, dx, dy, Ex)
            write(*,*) "La componente Ex es: ", Ex

            call integracion_doble_fy(xo, xf, dx, dy, Ey)
            write(*,*) "La componente Ex es: ", Ey

            call integracion_doble_fz(xo, xf, dx, dy, Ez)
            write(*,*) "La componente Ex es: ", Ez

        end program main

        ! definimos el integrando Ez
        real function fz(x,y)
            implicit none
            real :: x,y
            real :: pi, sigma, epsilon, zo
            pi = 3.14159265359
            epsilon = 8.85e-12
            sigma = 5.0/(4.0*pi)
            zo =  5
            fz = (sigma*zo)/( 4*pi*epsilon*(x**2 + y**2 + zo**2)**(1.5) )
            return
        end function

        ! definimos el integrando para Ex
        real function fx(x,y)
            implicit none
            real :: x,y
            real :: pi, sigma, epsilon, zo
            pi = 3.14159265359
            epsilon = 8.85e-12
            sigma = 5.0/(4.0*pi)
            zo =  5
            fx = -(sigma*x)/( 4*pi*epsilon*(x**2 + y**2 + zo**2)**(1.5) )
            return
        end function

        ! definimos el integrando para Ey
        real function fy(x,y)
            implicit none
            real :: x,y
            real :: pi, sigma, epsilon, zo
            pi = 3.14159265359
            epsilon = 8.85e-12
            sigma = 5.0/(4.0*pi)
            zo =  5
            fy = -(sigma*x)/( 4*pi*epsilon*(x**2 + y**2 + zo**2)**(1.5) )
```

```
    return
end function

! definimos la funcion que limita la region
! de integracion superior
real function limit_sup(x)
    implicit none
    real :: x
    limit_sup = sqrt(4.0-x**2)
    return
end function

! definimos la funcion que limita la region
! de integracion inferior
real function limit_inf(x)
    implicit none
    real :: x
    limit_inf = -sqrt(4.0-x**2)
    return
end function

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

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

! subrutina que calcula la integral doble con regla del trapecio
subroutine integracion_doble_fz(xo, xf, dx, dy, integral)
    implicit none
    real :: xo, xf, yo, yf, dx, dy, integral, suma
    real ::  limit_inf, limit_sup
    real :: trapeciofz_y, aux
    integer :: Nx, i
    Nx = (xf-xo)/dx + 1
    suma = 0.0
```

```
    do i = 1, Nx-1
        yo =  limit_inf(xo + i*dx)
        yf = limit_sup(xo + i*dx)
        suma = suma + trapeciofz_y(yo, yf, dy, xo + i*dx)
    end do
    aux = trapeciofz_y(limit_inf(xo),limit_sup(xo),dy,xo)
    aux = aux + trapeciofz_y(limit_inf(xo),limit_sup(xo),dy,xf)
    integral = (0.5*dx)*(aux + 2*suma)
end subroutine

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

    N = (yf-yo)/dy + 1
    suma = 0.0
    do i = 1, N-1
        suma = suma + fx(xcte, yo+i*dy)
    end do
    trapeciofx_y = (0.5*dy)*(fx(xcte,yo) + fx(xcte,yf) + 2*suma)
    return
end function

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

    N = (yf-yo)/dy + 1
    suma = 0.0
    do i = 1, N-1
        suma = suma + fy(xcte, yo+i*dy)
    end do
    trapeciofy_y = (0.5*dy)*(fy(xcte,yo) + fy(xcte,yf) + 2*suma)
    return
end function

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

    N = (yf-yo)/dy + 1
    suma = 0.0
    do i = 1, N-1
        suma = suma + fz(xcte, yo+i*dy)
    end do
    trapeciofz_y = (0.5*dy)*(fz(xcte,yo) + fz(xcte,yf) + 2*suma)
    return
end function
```