UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Integración numérica (parte II)

Análisis numérico

Autor:
David Gómez Torres

3 Noviembre del 2021

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1. Integrales múltiples

Del cálculo, sabemos que las integrales múltiples se pueden calcular como integrales iteradas

$$\int_c^d \left(\int_a^b f(x,y)dx \right) dy = \int_a^b \left(\int_c^d f(x,y)dy \right) dx \tag{1.1}$$

Primero se evalúa la integral en una de las dimensiones y el resultado de esta primera integración se incorpora en la segunda dimensión.

Una integral numérica doble estará basada en la misma idea. Primero se aplican métodos, como la regla de Simpson o del trapecio para segmentos múltiples, a la primera dimensión manteniendo constantes los valores de la segunda dimensión. Después, se aplica el método para integrar la segunda dimensión.

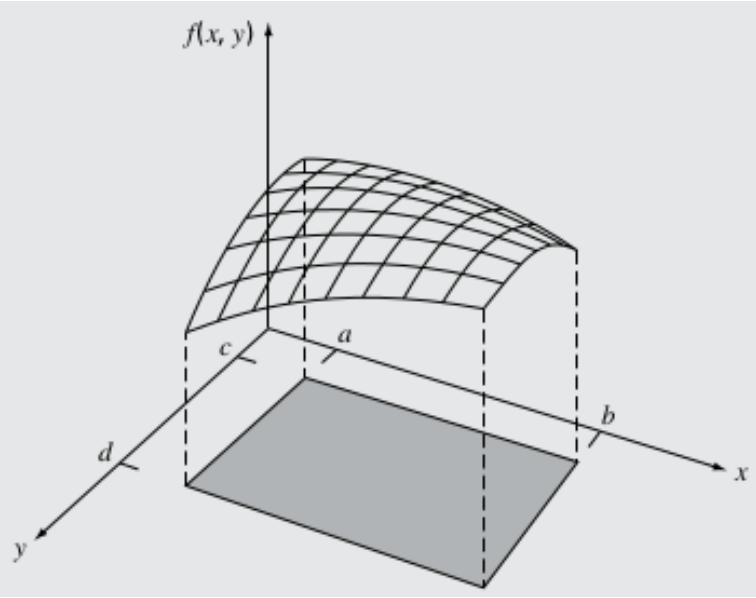

Figura 1.1: Ejemplo de aplicación de una integral doble

1.2. Integración de ecuaciones

1.2.1. Integración de Romberg

Extrapolación de Richardson

La *integración de Romberg* es una técnica diseñada para obtener integrales numéricas de funciones de manera eficiente.

Hay técnicas de corrección del error para mejorar los resultados de la integración numérica con base en la misma estimación de la integral. Dichos métodos usan dos estimaciones de una integral para calcular una tercera más exacta y, en general, se les conoce como **extrapolación de Richardson**.

Se puede demostrar que el error de esta estimación es $O(h_4)$. Así, hemos combinado dos estimaciones con la regla del trapecio de $O(h_2)$ para obtener una nueva estimación de $O(h_4)$. En el caso especial donde el intervalo es dividido a la mitad ($h_2 = h_{1/2}$), esta ecuación se convierte en

$$I \cong I(h_2) + \frac{1}{2^2 + 1}[I(h_2)I(h_1)]$$

o, agrupando términos,

$$I \cong \frac{4}{3}I(h_2)\frac{1}{3}I(h_1) \tag{1.2}$$

El algoritmo de integración de Romberg

$$I_{j,k} = \frac{4^{k1}I_{j+1,k1}I_{j,k1}}{4^{k-1} - 1} \tag{1.3}$$

1.2.2. Cuadratura de Gauss

Supongamos que se elimina la restricción de los puntos fijos y se tuviera la libertad de evaluar el área bajo una línea recta que uniera dos puntos cualesquiera de la curva. Al ubicar esos puntos en forma inteligente, definiríamos una línea recta que equilibrara los errores negativo y positivo. Así que, como en la figura , llegaríamos a una mejor estimación de la integral.

Cuadratura de Gauss es el nombre de una clase de técnicas para realizar tal estrategia. Las fórmulas particulares de cuadratura de Gauss descritas en esta sección se denominan fórmulas de Gauss-Legendre.

Desarrollo de la fórmula de Gauss- Legendre de dos puntos

Así como en el caso anterior para la obtención de la regla del trapecio, el objetivo de la cuadratura de Gauss es determinar los coeficientes de una ecuación de la forma

$$I \cong c_0 f(x_0) + c_1 f(x_1) \tag{1.4}$$

donde las $c =$ los coeficientes desconocidos.

A diferencia de la regla del trapecio que utiliza puntos extremos fijos a y b, los argumentos de la función x_0 y x_1 no están fijos en los extremos. De esta manera, ahora se tienen cuatro incógnitas

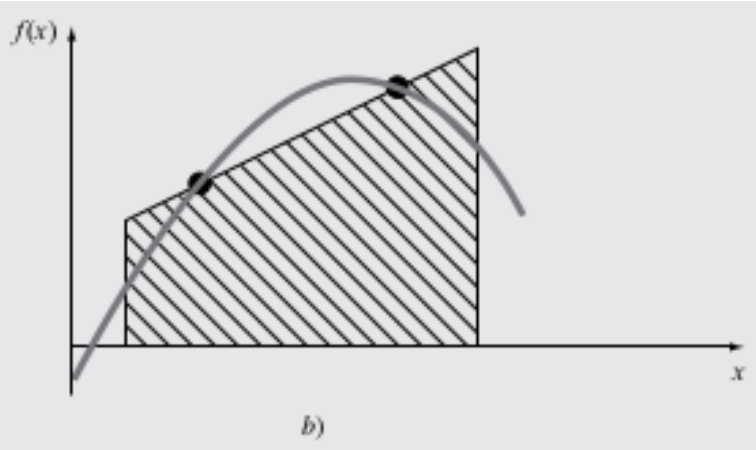

Figura 1.2: Representación gráfica de la cuadratura de Gauss

que deben evaluarse y, en consecuencia, se requieren cuatro condiciones para determinarlas con exactitud.

La fórmula de Gauss-Legendre de dos puntos

$$I \cong f\left(\frac{-1}{\sqrt{3}} + f\left(\frac{1}{\sqrt{3}}\right)\right) \tag{1.5}$$

Suponiendo una nueva variable x_d relacionada con la variable original x en una forma lineal, como sigue

$$x = a_0 + a_1 x_d \tag{1.6}$$

Podemos sustituir los coeficientes a_0 y a_1 en (1.6)

$$x = \frac{(b+a) + (b-a)x_d}{2} \tag{1.7}$$

y diferenciar esta ecuación para obtener

$$dx = \frac{b-a}{2} dx_d \tag{1.8}$$

Las ecuaciones (1.7) y (1.8) pueden sustituirse ahora por x y dx, respectivamente, en la ecuación que se habrá de integrar. Tales sustituciones efectivamente transforman el intervalo de integración sin cambiar el valor de la integral.

Fórmulas con más puntos

Aparte de la fórmula de dos puntos descrita en la sección anterior, se pueden desarrollar versiones con más puntos en la forma general

$$I \cong c_0 f(x_0) + c_1 f(x_1) + \cdots + c_{n-1} f(x_{n-1}) \tag{1.9}$$

donde n = número de puntos.

1.2.3. Integrales impropias

En esta sección nos ocuparemos de un tipo de integral impropia. Es decir, una con límite inferior $-\infty$ y/o límite superior $+\infty$.
Tales integrales a menudo se resuelven con un cambio de variable, que transforma el límite infinito en uno finito.

$$\int_a^b f(x)dx = \int_{1/b}^{1/a} \frac{1}{t^2} f\left(\frac{1}{t}\right) dt \tag{1.10}$$

Por lo tanto, se utiliza sólo cuando a es positiva y b es ∞, o cuando a es $-\infty$ y b es negativa. En los casos donde los límites son desde $-\infty$ a un valor positivo o desde un valor negativo a ∞, la integral puede calcularse en dos partes.

$$\int_{-\infty}^b f(x)dx = \int_{-\infty}^{-A} f(x)dx + \int_{-A}^b f(x)dx \tag{1.11}$$

donde –A se elige como un valor negativo lo suficientemente grande para que la función comience a aproximarse a cero.

Las fórmulas abiertas de aplicación múltiple podrán combinarse con las fórmulas cerradas para los segmentos interiores y fórmulas abiertas para los extremos. Por ejemplo, si se combinan la regla del trapecio de segmentos múltiples y la regla del punto medio se obtiene

$$\int_{x_0}^{x_n} f(x)dx = h\left[\frac{3}{2}f(x_1) + \sum_{i=2}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right] \tag{1.12}$$

Aunque se pueden usar estas relaciones, una fórmula preferida es

$$\int_{x_0}^{x_n} f(x)dx = h[f(x_{1/2} + f(x_{3/2}) + \cdots + f(x_{n-3/2}) + f(x_{n-1/2})] \tag{1.13}$$

que se conoce como la **regla extendida del punto medio**.

CAPÍTULO 2

METODOLOGÍA

2.1. Problema 21.15: Integral múltiple

Problema 21.15

Evalúe la siguiente integral triple,

$$\int_{-2}^{2}\int_{0}^{2}\int_{-3}^{1}(x^3-3yz)dxdydz$$

a) en forma analítica y

b) con el uso de aplicaciones únicas de la regla de Simpson 1/3. Para el inciso b) calcule el error relativo porcentual (ϵ_t).

a) De forma analítica:

$$\begin{aligned}
&\int_{-2}^{2}\int_{0}^{2}\int_{-3}^{1}(x^3-3yz)dxdydz \\
&\qquad = \int_{-2}^{2}\int_{0}^{2}\left(\frac{x^4}{4}-3xyz\right)\Big|_{x=-3}^{x=1}dydz \\
&\qquad = \int_{-2}^{2}\int_{0}^{2}(-20-12yz)dydz \\
&\qquad = \int_{-2}^{2}\left(-20y-\frac{12}{2}y^2z\right)\Big|_{y=0}^{y=2}dz \\
&\qquad = \left(-40z-\frac{24}{2}z^2\right)\Big|_{y=0}^{y=2} \\
&\qquad = -160
\end{aligned}$$

b) Uso de la regla de Simpson

```
import math

cadena = "Regla de Simpson 1/3".capitalize()
print(cadena.center(50, " "))


a = float(input("Ingrese el limite inferior de integracion: "))
b = float(input("Ingrese el limite superior de integracion: "))
n = int(input("Numero de iteraciones 'n': "))

def simpson(a,b,n):
    sum = 0
    int = 0
    h = (b-a)/n #delta

#regla de simpson1/3
    for i in range(n+1):
        x = (a + (i*h))
#funcion
        f = x**3-3yz
#casos
        if (x == a or x == b):
            sum += f
        elif (i % 2 != 0):
            f = 4*f
            sum += f
        elif (i % 2 == 0):
            f = 2*f
            sum += f

#resultado
    int = sum*(h/3)

    print("El resultado de la integral es: ", int)

simpson(a,b,n)
```

Ejecución del método

```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /usr/b
integracion2/impropia.py"
              Regla de simpson 1/3


Primera integral

Ingrese el limite inferior de integracion: -3
Ingrese el limite superior de integracion: 1


Segunda integral

Ingrese el limite inferior de integracion: 0
Ingrese el limite superior de integracion: 2


Tercera integral

Ingrese el limite inferior de integracion: -2
Ingrese el limite superior de integracion: 2

El resultado de la integral es: 159.3142
El error relativo porcentual es 0.4312
```

Figura 2.1: Integral múltiple

2.2. Problema 21.19: aplicación de la integral

Problema 21.19

Se recolectaron los siguientes datos para una sección transversal de un río (y = distancia de una ribera, H = profundidad y U = velocidad):

y, m	0	1	3	5	7	8	9	10
H, m	0	1	1.5	3	3.5	3.2	2	0
U, m/s	0	0.1	0.12	0.2	0.25	0.3	0.15	0

Use integración numérica para calcular

a) la profundidad promedio,

b) el área de la sección transversal,

c) la velocidad promedio y

d) el caudal.

Observe que el área de la sección transversal (A_c) y el caudal (Q) se pueden calcular como

$$A_c = \int_0^y H(y)dy, \qquad Q = \int_0^y H(y)U(y)dy$$

a) la profundidad promedio,

```
Metodo del trapecio
Ingresa el limite inferior de la funcion: 0
Ingresa el limite superior de la funcion: 10
Ingresa el numero de particiones 'n': 10
El resultado de la profundidad es:  25.396428571415065
```

b) el área de la sección transversal,

```
Metodo del trapecio
Ingresa el limite inferior de la funcion: 0
Ingresa el limite superior de la funcion: 10
Ingresa el numero de particiones 'n': 20
El resultado de seccion transversal es:  21.168048967562125
```

c) la velocidad promedio y

```
            Metodo del trapecio
Ingresa el limite inferior de la funcion: 0
Ingresa el limite superior de la funcion: 10
Ingresa el numero de particiones 'n': 10
El resultado de la velocidad es:  1.3602678571395284
```

d) el caudal.

```
            Metodo del trapecio
Ingresa el limite inferior de la funcion: 0
Ingresa el limite superior de la funcion: 10
Ingresa el numero de particiones 'n': 10
El resultado del caudal es: 5.013641
```

2.3. Problema 22.1: Integración de Romberg

Problema 22.1

Utilice la integración de Romberg de orden h^8 para evaluar

$$\int_0^3 xe^{2x}dx$$

Compare ϵ_a y ϵ_t.

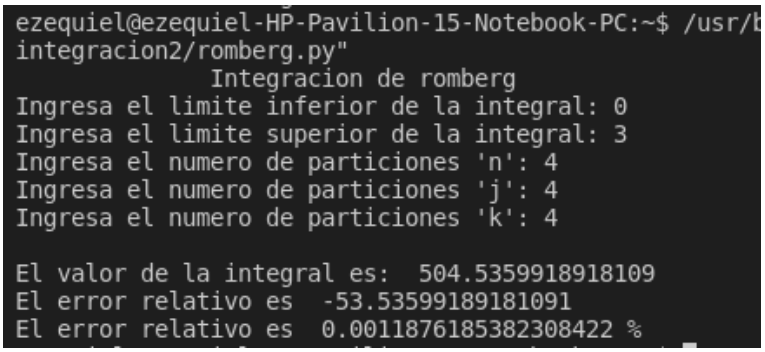

Figura 2.2: Integración mediante el método de Romberg

```
import numpy as np

cadena = "Integracion de Romberg".capitalize()
print(cadena.center(50, " "))

#pedir al usuario "a"
a = float(input("Ingresa el limite inferior de la integral: "))
#pedir al usuario "b"
b = float(input("Ingresa el limite superior de la integral: "))
 #pedir al usuario "n"
n = int(input("Ingresa el numero de particiones 'n': "))
j = int(input("Ingresa el numero de particiones 'j': "))
#pedir al usuario "k"
k = int(input("Ingresa el numero de particiones 'k': "))
#nivel de integracion (k*2)

def f(x):
    return x*np.exp(2*x)
def trapecio(a,b,n):
#calculo de delta x
    h = (b-a)/n
    s = 0
    for i in range(n-1):
        s += f(a+h*(i+1))
    return h/2*(f(a)+2*s+f(b))
def romberg(k,j,a,b):
    if k==1:
        return trapecio(a,b,2**(j-1))
    else:
        return (4**(k-1)*romberg(k-1,j+1,a,b)-romberg(k-1,j,a,b))/
        (4**(k-1)-1)
orden = 5
#crear matriz de ceros
rom = np.zeros((orden,orden))
#regla del trapecio a la primera columna
for i in range(orden):
    rom[i,0] = trapecio(0,5,2**i)
for i in np.arange(1,orden):
    for j in np.arange(i,orden):
        rom[j,i] = rom[j,i-1]+(rom[j,i-1]-rom[j-1,i-1])/(4**i-1)
print("\nEl valor de la integral es: ",romberg(k,j,a,b))
#error ea y et
ea = 504-53-romberg(k,j,a,b)
et = abs((504.53-romberg(k,j,a,b))/504.53)*100

print("El error relativo es ",ea)
print("El error relativo es ",et,"%")
```

2.4. Problema 22.5: Formulas de Gauss-Legendre

Problema 22.5

Obtenga una estimación de la integral del problema 22.1, pero use las fórmulas de Gauss-Legendre con dos, tres y cuatro puntos.

Calcule ϵ_t para cada caso sobre la base de la solución analítica.

Solución analítica

$$\begin{aligned}\int_0^3 xe^{2x}dx &= \int_0^6 \frac{e^u \cdot u}{4}du \\ &= \frac{1}{4}\int_0^6 e^u \cdot udu \\ &= \frac{1}{4}\left(e^u \cdot u - e^u\right)\Big|_0^6 \\ &= \frac{5e^6+1}{4} = 504.53\end{aligned}$$

```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /usr/bi
integracion2/gauss-legendre.py"
 Integral mediante la formula de gauss-legrende

Ingrese el limite inferior de la integral: 0
Ingrese el limite superior de la integral: 3


 formula de gauss-legrende con dos puntos

El valor de la integral es  406.2950215783018
El error relativo porcentual es  19.470592119734835 %


 formula de gauss-legrende con tres puntos

El valor de la integral es  455.64928983198536
El error relativo porcentual es  9.688365442692133 %


 formula de gauss-legrende con cuatro puntos

El valor de la integral es  504.13046817859555
El error relativo porcentual es  0.07918891273153693 %
```

Figura 2.3: Calculo de la integral mediante la fórmula de Gauss-Legendre

```python
import numpy as np

cadena1 = "Integral mediante la formula de Gauss-Legrende\n"
.capitalize()
print(cadena1.center(50, " "))

a = float(input("Ingrese el limite inferior de la integral: "))
b = float(input("Ingrese el limite superior de la integral: "))

def f(x):
    return x*np.exp(2*x)

def gauss2puntos(a,b,f):
    cadena = "\n\n Formula de Gauss-Legrende con dos puntos\n"
    .capitalize()
    print(cadena.center(50, " "))

    x = np.array([np.sqrt(1/3),-np.sqrt(1/3)])
    #arg.funcion
    w = np.array([1,1])
    #factor de ponderacion

    u = (b-a)*x/2+(a+b/2) #cambio de variable

#formula gauss-legendre
    integral = (b-a)*np.sum(w*f(u))/2
#error
    et = abs((504.53-integral)/504.53)*100

    print("El valor de la integral es ",integral)
    print("El error relativo porcentual es ",et)

def gauss3puntos(a,b,f):
    cadena = "\n\n Formula de Gauss-Legrende con tres puntos\n"
    .capitalize()
    print(cadena.center(50, " "))

    x = np.array([np.sqrt(3/5),0,-np.sqrt(3/5)]) #arg.funcion
    w = np.array([5/9,0,5/9]) #factor de ponderacion

    u = (b-a)*x/2+(a+b/2) #cambio de variable

#formula gauss-legendre
    integral = (b-a)*np.sum(w*f(u))/2
```

```
#error
    et = abs((504.53-integral)/504.53)*100

    print("El valor de la integral es ",integral)
    print("El error relativo porcentual es ",et)

def gauss4puntos(a,b,f):
    cadena = "\n\n Formula de Gauss-Legrende con cuatro puntos\n"
    .capitalize()
    print(cadena.center(50, " "))

    x = np.array([-0.861136312,-0.339981044,0.339981044,0.861136312])
    #arg.funcion
    w = np.array([0.3478548,0.6521452,0.6521452,0.3478548])
    #factor de ponderacion

    u = (b-a)*x/2+(a+b/2) #cambio de variable

#formula gauss-legendre
    integral = (b-a)*np.sum(w*f(u))/2

#error
    et = abs((504.53-integral)/504.53)*100

    print("El valor de la integral es ",integral)
    print("El error relativo porcentual es ",et)

gauss2puntos(a,b,f)
gauss3puntos(a,b,f)
gauss4puntos(a,b,f)
```

2.5. Problema 22.19: Integrales impropias

Problema 22.9

Use integración numérica para evaluar lo siguiente:

a)

$$\int_2^\infty \frac{dx}{x(x+2)}$$

b)

$$\int_0^\infty \frac{1}{(1+y^2)(1+y^2/2)} dy$$

c)

$$\int_{-2}^\infty y e^{-y} dy$$

d)

$$\int_0^\infty \frac{1}{\sqrt{2\pi}} e^{-x}$$

```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:
integracion2/impropia.py"
              Regla de simpson 1/3


Ingrese el limite inferior de integracion: 2
Ingrese el limite superior de integracion: 0
Numero de iteraciones 'n': 3

El resultado de la integral es: 0.3465
```

Figura 2.4: Integral a)

```
              Regla de simpson 1/3


Ingrese el limite inferior de integracion: 0
Ingrese el limite superior de integracion: -1
Numero de iteraciones 'n': 4

El resultado de la integral es: 1.047
```

Figura 2.5: Integral b)

```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /u
integracion2/impropia.py"
              Regla de simpson 1/3


Ingrese el limite inferior de integracion: -2
Ingrese el limite superior de integracion: 0
Numero de iteraciones 'n': 5

El resultado de la integral es: 14.7781
```

Figura 2.6: Integral c)

```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /us
integracion2/impropia.py"
              Regla de simpson 1/3


Ingrese el limite inferior de integracion: 0
Ingrese el limite superior de integracion: 4
Numero de iteraciones 'n': 5

El resultado de la integral es: 0.7071
```

Figura 2.7: Integral d)

```python
import numpy as np

cadena1 = "Calculo de integral impropia\n".capitalize()
print(cadena1.center(50, " "))

a = float(input("Ingrese el limite inferior de la integral: "))
b = float(input("Ingrese el limite superior de la integral: "))
n = int(input("Numero de iteraciones 'n': "))

def f(x):
    return (1/x*(x+2))
    #return (1/np.sqrt(2*np.pi)*np.exp(-x**2/2))
    #return (1/(1+x**2)*(1+x**2/2))
    #return (np.exp(-x)*np.sin(x)**2)
    #return (x*np.exp(-x))

def conversion(f,t):
    return (1/t**2*(f(1/t)))

def simpson(a,b,n,conversion):
    sum = 0
    int = 0
    h = (b-a)/n #delta

#regla de simpson1/3
    for i in range(n+1):
        x = (1/a + (i*h))
#casos
        if (x == a or x == b):
            sum += f
        elif (i % 2 != 0):
            f = 4*f
            sum += f
        elif (i % 2 == 0):
            f = 2*f
            sum += f
#resultado
    int = sum*(h/3)
    print("El resultado de la integral es: ", int)

simpson(a,b,n)
```

CAPÍTULO 3

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-hill.