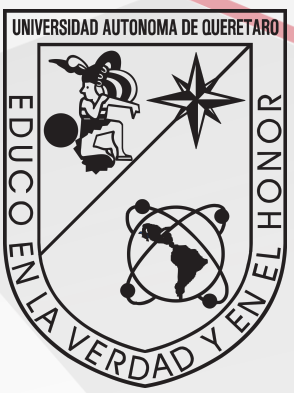
Universidad Autónoma de Querétaro

Facultad de Ingeniería



Tarea 12: Integración numérica pt.2

Análisis numérico

Autor:
J.A. Salinas Sánchez
Abril 2022

Índice general

Capítulo 1

Introducción

Siguiendo con la parte de integración numérica, la indudabilidad del impacto e importancia en el desarrollo de la humanidad del cálculo infinitesimal ha impulsado a matemáticos, físicos y demás personas a buscar manera de resolver integrales numéricas de manera mucho más eficiente, así como más precisa. Es aquí donde entran otros métodos de integración: los métodos compuestos.

Los métodos compuestos toman su nombre del simple hecho de que utilizan varios métodos matemáticos (no sólo de integración), como extrapolaciones o interpolaciones, para refinar sus aproximaciones. Entre dichos métodos se tiene los siguientes:

1.1. Integración de Romberg

Este método de integración consiste en combinar el método del trapecio con la extrapolación de Richardson. El primer método ya lo conocemos. El segundo consiste en obtener n aproximaciones de la integral de la función $f(x)$, I,y relacionándolas en pares, obtener $n-1$ aproximaciones; así hasta obtener una última. La manera en la que la integración de Romberg logra aumentar su exactitud, no sólo es por obtener más aproximaciones, sino, poder asignar un peso a cada una; pues, las n aproximaciones de inicio se hace una más exacta que la anterior; mediante el incremento en las iteraciones de dichas aproximaciones. Entonces, tiene en cuenta que el peso de cada aproximanción debe ser mayor que la anterior. Entonces, el método se divide en dos etapas:

- Obtener las aproximaciones: se obtiene n aproximaciones de I mediante la regla del trapecio compuesta, duplicando el número de iteraciones para cada aproximación subsecuente.

Aplicar la extrapolación de Richardson: es empareja cada aproximación con su subsecuente y se obtiene una nueva aproximación mediante la siguiente fórmula:

$$I_{j,k} \equiv \frac{4^{k-1} I_{j+1,k} - I_{j,k-1}}{4^{k-1} - 1} \tag{1.1}$$

Cuando se llegue a una única aproximación, el proceso habrá finalizado.

1.2. Cuadratura de Gauss-Legendre

En el siguiente método, como siempre, Carl Friedrich Gauss tuvo algo que ver. Contemporáneo de éste, el matemático francés Adrien-Marie Legendre había descubierto un grupo de polinomios que satisfacen una cierta ecuación diferencial:

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}P_n(x)\right] + n(n+1)P_n(x) = 0 \tag{1.2}$$

Y se pueden calcular mediante la siguiente ecuación:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right] \tag{1.3}$$

Estos polinomios tienen la característica de ser ortogonales entre sí, por lo que se podrían definir como linealmente independientes, así como sus soluciones. Es por eso que se utilzan apliamente en la expansión analítica de funciones, en especial para interpolaciones mediante polinomios, utilizando el concepto de que las todo vector (refiriéndonos a espacios vectoriales) resulta una suma (combinación lineal) e vectores linealmente independientes.

Gauss sabiendo esto, los utilizó, no para interpolar; sino para extrapolar datos de una función $f(x)$ a una fórmula de integración fija. Es decir: si, con una interpolación, uno genera una función o más datos a partir de unos ya obtenidos; con una extrapolación (cuadratura), uno ajusta los datos a una función predeterminada (los cuadra). Y eso es lo que hizo Gauss.

Gauss se basó en la regla del trapecio; pero, en lugar de escoger dos puntos exteriores del intervalo y de cada subintervalo de integración, planteó utilizar dos puntos interiores. Para ello, panteó cambiar el intervalo de integración de $[a,b]$ a $[1,1]$ mediante la siguiente sustitución:

$$I = \int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}u + \frac{b+a}{2})du \tag{1.4}$$

Lo que, aplicando suma de Riemann implicaría que:

$$I \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f(\frac{b-a}{2}u_i + \frac{a+b}{2}) \tag{1.5}$$

Con u_i siendo las raíces del polinomio de Legendre que resulte del número de iteraciones deseadas. Así que Gauss definió que, según el número n que se utilice para aproximar una integral; se puede obtener las raíces del polinomio de Legendre del mismo grado que n como los x_i en los que evaluar la función y el peso de cada término de la suma w_i mediante la siguiente fórmula:

$$w_i = \frac{2}{(1-x^2)\left[P_n(x_i)\right]^2} \tag{1.6}$$

Capítulo 2

Desarrollo y método

Los problemas se resolvieron de la siguiente manera: se construyó tres códigos: uno para la integración numérica múltiple, otro para la cuadratura de Gauss y otro para la integración de Romberg. Además, se utilizó como apoyo los códigos para interpolaciones mediante polinomios, trabajados anteriormente.

El código para la integración de Romberg pide el grado de complejidad al que se quiere llegar $h = 2n$; luego, realiza n aproximaciones de I con la regla del trapecio $\frac{1}{2}$, duplicando el número de iteraciones hasta llegar a h. Finalmente, aplica la fórmula de Romberg hasta reducir todas las aproximaciones a una sola, y devuelve el resultado.

Luego; el código de cuadratura gaussiana recibe una función y, dependiendo del n dado, va realizando la sumatoria obteniendo los coeficientes w_i y las raíces de los polinomios de Legendre previamente cargados en una lista. Por último, realiza la sumatoria y regresa el resultado.

Concluyendo, la integración numérica múltiple se realiza mediante integración trapezoidal simple, primero sobre una variable, y luego, sobre otra.

2.1. Código

2.1.1. Romberg

```
#coding:utf8
import numpy as np
import sympy as sp
import math as mt

x=sp.symbols('x')

def main():
    aproximaciones=[]
    f=str(input('Introduzca su funci n: '))
    a=float(input('Introduzca su l mite inferior de integraci n:
    '))
    b=float(input('Introduzca su l mite superior de integraci n:
    '))
    h=int(input('Introduzca el orden h del m todo: '))

```

```
    func=getfunction(f)

    for i in range(2,h+1,2):
        aproximaciones.append(trapeciocomp(func,a,b,i))
    count=2
    res=romberg(aproximaciones,count)
    print(res)

def romberg(l,c):

    if (len(l)==1):
        return(l)

    v=[]
    for i in range(0,len(l)-1):
        ap=(4**(c-1)*l[i+1]-l[i])/(4**(c-1)-1)
        v.append(ap)
    c+=1
    return(romberg(v,c))


def trapeciocomp(f,a,b,n):
    eq=getfunction(f)
    sum=0
    for i in range(1,n):
        xi=a+i*((b-a)/n)
        sum=sum+eq.evalf(subs={x:xi})
    c=(eq.evalf(subs={x:a})+eq.evalf(subs={x:b}))/2
    return(((b-a)/n)*(c+sum))

def getfunction(f):
    global x
    return(sp.sympify(f))

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

2.1.2. Cuadratura Gaussiana

```
#coding:utf8
import numpy as np
import sympy as sp
import math as mt

x=sp.symbols('x')
w2=[1,1]
w3=[0.55555,0.88888,0.55555]
w4=[0.3478548451,0.6521451549,0.3478548451,0.6521451549]
w5=[0.2369268851,0.4786286705,0.2369268851,0.4786286705,0.5688]
x2=[0.5773502692,-0.5773502692]
x3=[0.7745966692,0,-0.7745966692]
x4=[0.8611363113,0.3399810536,-0.3399810536,-0.8611363113]
x5=[0.9061798459,0.5384693101,0,-0.5384693101,-0.9061798459]

def main():
    f=str(input('Introduzca su funci n: '))
    a=float(input('Introduzca su l mite inferior de integraci n:
    '))
    b=float(input('Introduzca su l mite superior de integraci n:
    '))
    func=getfunction(f)
    n=int(input('Introduzca el n mero de iteraciones deseada: '))
    if n<=5 and n>1:
```

```
        gauss(func,a,b,n)

def gauss(func,a,b,n):
    global w2
    global w3
    global w4
    global w5
    global x2
    global x3
    global x4
    global x5
    w=[]
    z=[]
    k=(b-a)/(2)
    sum1=0

    if(n==2):
        w=w2
        z=x2
    elif(n==3):
        w=w3
        z=x3
    elif(n==4):
        w=w4
        z=x4
    elif(n==5):
        w=w5
        z=x5

    for i in range(0,len(w)):
        xi=((b-a)/(2))*z[i]+((a+b)/(2))
        sum1=w[i]*func.evalf(subs={x:xi})

    res=sum1*k
    print(res)


def getfunction(f):
    global x
    return(sp.sympify(f))

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

2.2. Problemas

2.2.1. Unidad 21

21.15

21.23

2.2.2. Unidad 22

22.1

22.5

22.9

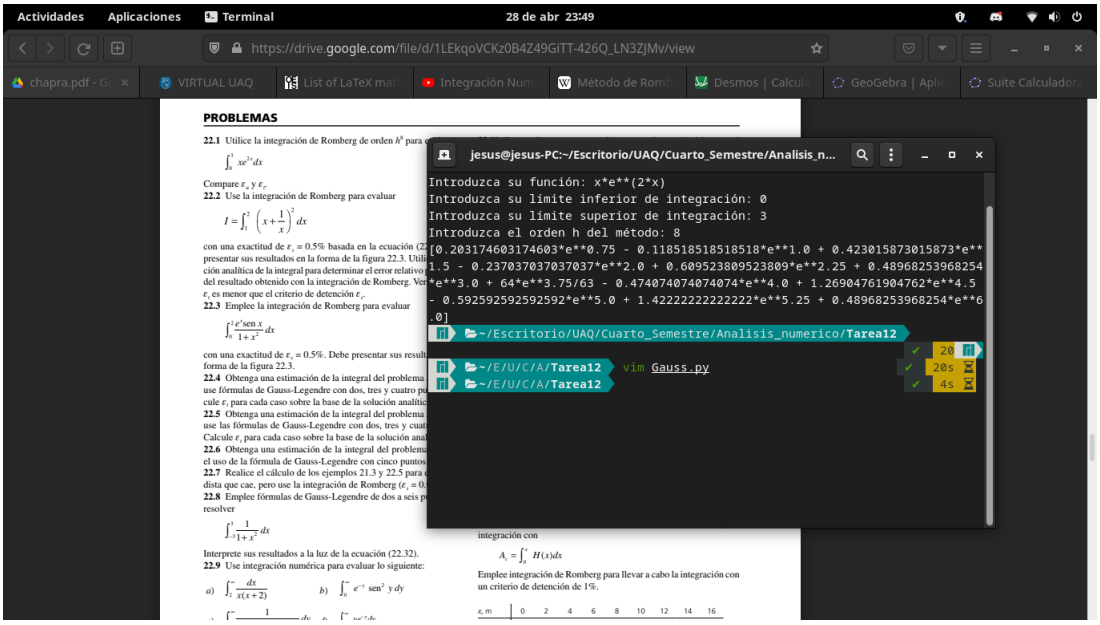


Figura 2.1: Resultado del ejercicio 22.1

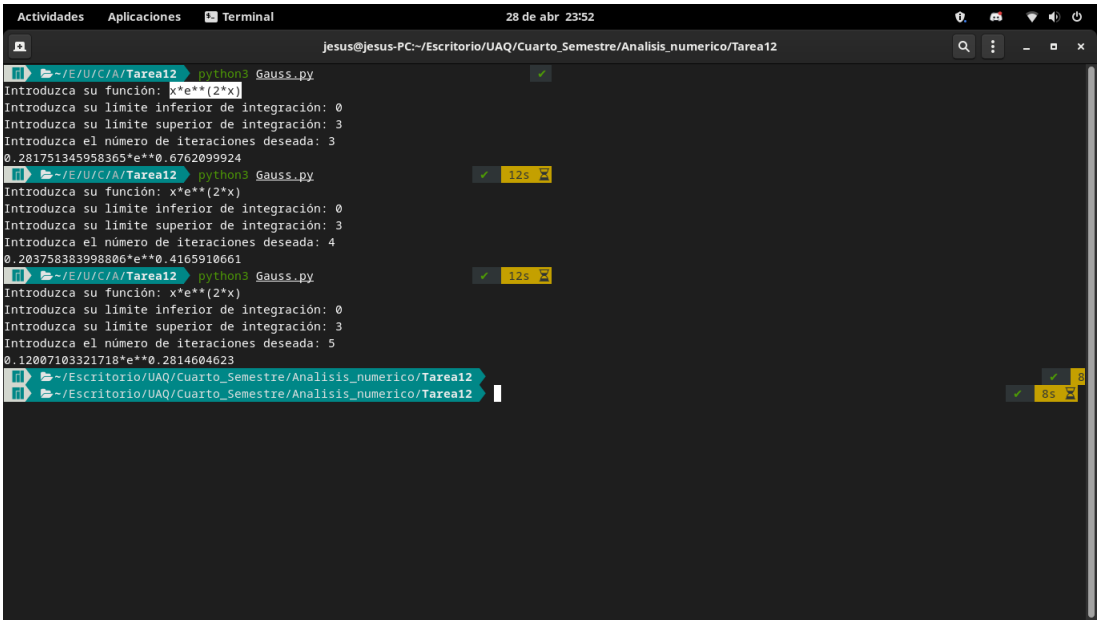


Figura 2.2: Resultado del ejercicio 22.5

Capítulo 3

Conclusión

Contar con métodos que logren refinar tanto la precisión de las aproximaciones obtenidas mediante otros métodos de integración numérica, con un costo computacional fijo; representa un gran logro del análisis numérico; pues, con los métodos simples de integración numérica, para mejorar los resultados se tiene que incrementar el número de iteraciones hasta que, en algunos casos, ya no es informáticamente costeable hacerlo. En cambio, si uno tiene un métodos con costos computacionales fijos, que se pueden controlar al detalle (como la Cuadratura de Gauss, donde uno hasta puede dejar fijo el número de iteraciones), y con resultados tan buenos; en muchos casos las soluciones numéricas son casi equivalentes a las analíticas. Esto último hace que casi literalmente resolvamos lo irresolvible.

Bibliografía

Chapra, S.C. et Canale, R.P (2015) *Métodos numéricos para ingenieros* McGrawHill. pp (494-510)