Taller de Derivación e Integración

Integrantes:

-Carlos Barón

-Santiago Chaustre

-Andrés Cocunubo

**DERIVACIÓN**

**F.** Realice una modificación de la fórmula de los tres puntos, tomando valores entre $(x_0 - h)$ y $(x_0 + h)$ y compare la magnitud del error con la fórmula de tres puntos.

Fórmula de tres puntos con $x_0; x_1 = x_0 + h; x\_2 = x_0 + 2h$

**Funciones usadas:**

- f(x): Función usada para el ejercicio, recibe por parámetro el valor a calcular.
$$\log(x)$$
- DerTresPuntoSinModificar($x_0$,h) : Función sin modificar que recibe por parámetro el valor a calcular y el tamaño del paso.
$$\frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0 + h) - \frac{1}{2}f(x_0 + 2h)\right] + \frac{f^3(x_0)}{3}h^2$$
- DerTresPuntoModificada($x_0$,h): Función con el cambio de los puntos, recibe por parámetro el valor y el paso.
$$\frac{1}{2h}[-f(x_0 - h) + f(x_0 + h)] + \frac{f^3(x_0)}{3}h^2$$

**Código Sin modificar:**

```
import numpy as np
import sympy as sp
from math import *

def f(x):
    return log(x)

def DerTresPuntoSinModificar(x0,h):
    c = "log(x)"
    pD = str(sp.diff(c))
    sD = str(sp.diff(pD))
    tD = str(sp.diff(sD))
    x=x0
    res = float(eval(tD))
    error = abs(res/3*h**2)
    r = 1/h*(-3/2*f(x0)+2*f(x0+h)-1/2*f(x0+2*h)) + error
    print ( "H: %.5f"%h, " | Valor de Error : %.5f "%error,"| Resultado :
%.5f"%r)

value = 1.8
DerTresPuntoSinModificar(value,0.1)
DerTresPuntoSinModificar(value,0.01)
DerTresPuntoSinModificar(value,0.0011)
DerTresPuntoSinModificar(value,0.0001)
```

```
DerTresPuntoSinModificar(value,0.00001)
```

**Código con la modificación:**

```
import numpy as np
import sympy as sp
from math import *
def f(x):
    return log(x)

def DerTresPuntoModificada(x0,h):
    c = "log(x)"
    pD = str(sp.diff(c))
    sD = str(sp.diff(pD))
    tD = str(sp.diff(sD))
    x=x0
    res = float(eval(tD))
    error = abs(res/3*h**2)
    r = 1/(2*h)*(-f(x0-h)+f(x0+h))+error
    print ( "H: %.5f"%h, " | Valor de Error : %.5f "%error,"| Resultado :
%.5f"%r)

value = 1.8
DerTresPuntoModificada(value,0.1)
DerTresPuntoModificada(value,0.01)
DerTresPuntoModificada(value,0.0011)
DerTresPuntoModificada(value,0.0001)
DerTresPuntoModificada(value,0.00001)
```

**Salida fórmula sin modificar:**

```
H: 0.10000  | Valor de Error : 0.00114311842706904445  | Resultado : 0.5556849655
H: 0.01000  | Valor de Error : 0.00001143118427069044  | Resultado : 0.5555556970
H: 0.00110  | Valor de Error : 0.00000013831732967535  | Resultado : 0.5555555557
H: 0.00010  | Valor de Error : 0.00000000114311842707  | Resultado : 0.5555555556
H: 0.00001  | Valor de Error : 0.00000000001143118427  | Resultado : 0.5555555556
```

**Salida fórmula modificada:**

```
H: 0.10000  | Valor de Error : 0.00114311842706904445  | Resultado : 0.5572712940
H: 0.01000  | Valor de Error : 0.00001143118427069044  | Resultado : 0.5555727024
H: 0.00110  | Valor de Error : 0.00000013831732967535  | Resultado : 0.5555557630
H: 0.00010  | Valor de Error : 0.00000000114311842707  | Resultado : 0.5555555573
H: 0.00001  | Valor de Error : 0.00000000001143118427  | Resultado : 0.5555555556
```

La magnitud del error es la misma, sin embargo, según el resultado la fórmula sin modificar es mucho más exacta que la modificada.

**k**. Aplicación: En un circuito con un voltaje E(t) y una inductancia L se tiene que:

$E(t) = L\frac{di}{dt} + Ri$ donde R es la resistencia e i es la corriente. La siguiente tabla muestra la medida de la corriente para varios instantes de tiempos (segundos), con R=0.142 ohms L=0.98 henries. Aproxime el voltaje para los valores de t.

| t | 1.00 | 1.01 | 1.02 | 1.03 | 1.04 |
|---|---|---|---|---|---|
| i | 3.10 | 3.12 | 3.14 | 3.18 | 3.24 |

**Funciones usadas:**

- L(x): Función que realiza el polinomio de Lagrange de grado 4, recibe por parámetro el valor a evaluar dentro del mismo.

$$P(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + y_3L_3(x) + y_4L_4(x)$$

**Código:**

```
def L(x):
    l1=       ((x-1.01)*(x-1.02)*(x-1.03)*(x-1.04))/((1-1.01)*(1-1.02)*(1-
1.03)*(1-.04))*3.10
    l2=((x-1)*(x-1.02)*(x-1.03)*(x-1.04))/((1.01-1)*(1.01-1.02)*(1.01-
1.03)*(1.01-1.04))*3.12
    l3=((x-1)*(x-1.01)*(x-1.03)*(x-1.04))/((1.02-1)*(1.02-1.01)*(1.02-
1.03)*(1.02-1.04))*3.14
    l4=((x-1)*(x-1.01)*(x-1.02)*(x-1.04))/((1.03-1)*(1.03-1.01)*(1.03-
1.02)*(1.03-1.04))*3.18
    l5=((x-1)*(x-1.01)*(x-1.02)*(x-1.03))/((1.04-1)*(1.04-1.01)*(1.04-
1.02)*(1.04-1.03))*3.24
    return l1+l2+l3+l4+l5

x,y= [1,1.01,1.02,1.03,1.04],[3.10,3.12,3.14,3.18,3.24]

for i in range(0,4):
    r= L(x[i])
print("El voltaje aproximado en t=%.2f"%x[i], " es %.5f voltios."%r)
```

**Salida:**

```
El voltaje aproximado en t=1.00  es -0.12917 voltios.
El voltaje aproximado en t=1.01  es 3.12000 voltios.
El voltaje aproximado en t=1.02  es 3.14000 voltios.
El voltaje aproximado en t=1.03  es 3.18000 voltios.
El voltaje aproximado en t=1.04  es 3.24000 voltios.
```

**INTEGRACIÓN**

i. Usando la fórmula de cuadratura de Gauss, particionando la integral de la siguiente manera:

$$\int_1^2 x\,e^x dx = \int_1^{1.5} x\,e^x dx + \int_{1.5}^2 x\,e^x dx$$

**Funciones utilizadas:**

- f(x) : Función exponencial, que recibe por parámetro el valor a calcular

$$xe^x$$

- cuadratura(a,b): Implementa la función de cuadratura de Gauss dados los límites de integración

$$\frac{b-a}{2}\left[f\left(-\frac{b-a}{2}*\frac{1}{\sqrt{3}}+\frac{b+a}{2}\right)+f\left(\frac{b-a}{2}*\frac{1}{\sqrt{3}}+\frac{b+a}{2}\right)\right]$$

- scipy.integrate.quad(f,a,b): Función del módulo scipy que realiza la integral usando la cuadratura de Gauss.

**Código:**

```
import numpy as np
import sympy as sp
from math import *
from scipy import integrate
def f(x):
    return x*exp(x)
def cuadratura(a,b):
    return (b-a)/2*(f(-(b-a)/2*1/sqrt(3)+(b+a)/2)+f((b-a)/2*1/sqrt(3)+(b+a)/2))

r = cuadratura(1,2)
print ("La aproximación de 1 a 2 es %.5f"%r)
p1 = cuadratura(1,1.5)
p2 = cuadratura(1.5,2)
r = p1 + p2
print ("La aproximación de 1 a 1.5 más 1.5 a 2 es %.5f"%r)
s = integrate.quad(f,1,2)
print ("La aproximación de scipy es ",s)
```

**Salida:**

```
C:\Users\AndrésFelipe\Desktop\6to Sem\Numérico>py 2i.py
La aproximación de 1 a 2 es 7.38327
La aproximación de 1 a 1.5 más 1.5 a 2 es 7.38868
La aproximación de scipy es  (7.389056098930649, 8.203500211279744e-14)
```

La aproximación realizada con la partición de los límites es mucho más precisa, y comparando con la función de cuadratura de scipy se observa que es más cercana.

**PUNTO J**

Utilice el siguiente código de la regla de Simpson en dos direcciones (para integrales dobles) para resolver el siguiente problema:

Un lago tiene una forma que aproximadamente es rectangular. Las dimensiones son 200 metros de ancho por 400 metros de largo. Se realiza una partición (grilla) para estimar aproximadamente la profundidad en metros en cada cuadricula de la malla como se muestra en la siguiente tabla de datos. Utilice los datos para estimar el volumen aproximado de agua que contiene el lago.

| X / Y | 0 | 100 | 200 | 300 | 400 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 4 | 6 | 0 |
| 50 | 0 | 3 | 5 | 7 | 3 |
| 100 | 1 | 5 | 6 | 9 | 5 |
| 150 | 0 | 2 | 3 | 5 | 1 |
| 200 | 0 | 0 | 1 | 2 | 0 |

**Funciones utilizadas:**

- `f(x,y)`: Función que recibe una coordenada en X y Y, y retorna el respectivo valor.
- `simpson(f,a,b,m)`: Función que recibe una función f, un intervalo a-b y el tamaño-1 de elmentos. Retorna la aproximación del área según la regla de Simpson.

$$S(f,h) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2M-2} + 4f_{2M-1} + f_{2M})$$

- simpson2(f,ax,bx,ay,by,mx,my): Función que recibe los mimos parámetros que la anterior, a diferencia de que los recibe para una variable X y Y.

**Código:**

```
from sympy import *
#Funcion que recibe una coordenada en X y Y y retorna el respectivo valor
def f(x,y):
    ax = [0,100,200,300,400]
    ay = [0,50,100,150,200]
    table= [[0,0,4,6,0],
            [0,3,5,7,3],
            [1,5,6,9,5],
            [0,2,3,5,1],
            [0,0,1,2,0]]
    yi= ax.index(x) #Obtener la posicion de y en la matriz
    xi= ay.index(y) #Obtener la posicion de x en la matriz
    return table[xi][yi] #Retornas el valor segun los indices x,y

#Metodo de simpson para una variable
def simpson(f,a,b,m):
    h = (b - a)/float(m) #Calculo del paso h
    sum1 = 0
    for i in range(1, int(m/2 + 1)):
        sum1 += f(a + (2*i - 1)*h)  #Calculo de la funcion en x1
    sum1 *= 4
    sum2 = 0
    for i in range(1, int(m/2)):
        sum2 += f(a + 2*i*h)  #Calculo de la funcion en x2
    sum2 *= 2
    approx = (b - a)/(3.0*m)*(f(a) + f(b) + sum1 + sum2) #Calculo de la aproximacion
segun metodo de simpson
    return approx

def simpson2(f,ax,bx,ay,by,mx,my):
    x= Symbol('x')
    dy= (by - ay)/my #Calculo del paso en y
    v= ay #Establecer un y fijo
    r= []
    for i in range (0,my+1):
        def g(x): return f(x,v)
        u= simpson(g,ax,bx,mx) #Aproximacion de Simpson en x con un y fijo
        r= r+[u]
        v=v+dy #Mover paso en y
    s=0
    for i in range(1,my):
        s=s+2*(2-(i+1)%2)*r[i] #Suma de los volumenes obtenidos
    s= dy/3*(r[0]+s+r[my]) #Calculo de la aproximacion segun metodo de simpson
    return s

print("El area del lago es: ",simpson2(f,0,400,0,200,4,4)," metros cubicos.")
```

Salida:

```
C:\Users\LICHO\Downloads>python punto2j.py
El area del lago es:  301111.1111111111 metros cubicos.
```

**PUNTO M**

Compare los resultados que obtuvo en la tabla que permite evaluar las probabilidades con la normal estándar si aplica el método de cuadratura.

**FUNCIONES USADAS**

- `f(x)`: Esta función evalúa la función de gauss, usando el parámetro x y es usada para obtener los valores en cada punto.

$$\frac{1}{\sqrt{2\pi} * e\hat{}\left(-\frac{1}{2} * x\hat{}2\right)}$$

- `fixed_quad(f(x), limite Infe,limiteSup, Grado polinomio)` : Esta es una función del módulo integrate de scipy que permite calcular la cuadratura de gauss utilizando la función , el límite inferior , superior y el grado del polinomio.
- `Cuadratura(x)`: Esta función usa la función `fixed_quad` para retornar su aproximación en el punto x

**CODIGO**

```
from scipy import integrate as sp
import numpy as np
import math

def f(x) :
    return 1/(np.sqrt(2*np.pi))*np.exp(-1/2*x**2)
def cuadratura(x) :
    return sp.fixed_quad(f,float(-6),float(x) , n=11)[0]

sumag1 = 0
sumag2 = 0
print ("Ingrese punto a calcular.")
limSup = float(input())
limInf = float (-5.91637424)
print ("Ingrese tamaño del paso. (8 cifras decimales).")
dx = float(input())
print ("Ingrese tamaño de la particion. (8 cifras decimales).")
ddx = float(input())
Z = 0.0
suma = 0
frac = 0.001
print ("\n")
numerofilas =  int ( ((limSup/dx)) +2)
numerocolumnas = int((dx/ddx)+1)
matriz = [None] * numerofilas;
for i in range(numerofilas):
    matriz[i] = [None] * (numerocolumnas)
i = 1

matriz2 = [None] * numerofilas;
for i in range(numerofilas):
    matriz2[i] = [None] * (numerocolumnas)
i = 1
```

```
auxddx = 0
j = 1
while auxddx < (dx-ddx):
    matriz[0][j] = ("%.4f")%auxddx
    auxddx = auxddx + ddx
    j = j+1
auxddx = 0
j = 1
while auxddx < (dx-ddx):
    matriz2[0][j] = ("%.4f")%auxddx
    auxddx = auxddx + ddx
    j = j+1
xx= open("tabla_normal-trapecio.txt","w+")
xx1= open("tabla_normal-cuadratura.txt","w+")
aux = Z
while Z <= (limSup ):
    suma = suma
    matriz[i][0] = ("%.4f")%Z
    matriz2[i][0] = ("%.4f")%Z
    j = 1
    for w in range (0,10):
        suma = suma
        while limInf < aux:
            fInf = float(f(limInf))
            cotSup = limInf + ddx
            fSup = float(f(cotSup))
            area = ((fInf + fSup) * ddx) / 2
            suma = suma + area
            limInf = limInf + ddx
        matriz[i][j] = ("%.8f")%suma
        sumag1 += suma
        suma2 = cuadratura(cotSup)
        sumag2 += suma2
        matriz2[i][j] = ("%.8f")%suma2
        #print(repr("%.8f"%aux).rjust(9), repr("%.8f"%suma).rjust(10))
        #print ("\n")
        aux = aux + ddx
        j = j+1
    Z = aux
    i = i+1
for i in range (0,numerofilas):
    for j in range ( 0,numerocolumnas):
        xx.write("%s ~ " % (matriz[i][j]))
    xx.write("\n")
for i in range (0,numerofilas):
    for j in range ( 0,numerocolumnas):
        xx1.write("%s ~ " % (matriz2[i][j]))
    xx1.write("\n")

error = abs((sumag1 - sumag2)/(sumag1))*100;

print("El error estandar es de : ",error)
```

Este código además de mostrarnos la diferencia entre la integral calculada con la cuadratura y la calculada con el método de trapecios guarda ambas tablas en archivos de texto .txt llamados “tabla_normal-trapecio.txt” y “tabla_normal-cuadratura.txt”

**SALIDA POR PANTALLA**

```
Ingrese punto a calcular.
 3
Ingrese tamaño del paso. (8 cifras decimales).
 0.1
Ingrese tamaño de la particion. (8 cifras decimales).
 0.01


El error estandar es de :  0.000651573719064
```