Gauss Hermite

Prateek Bhardwaj
(2020PHY1110)
(20068567042)

Monu Chaurasiya
(2020PHY1102)
(20068567035)

S.G.T.B. Khalsa College, University of Delhi, Delhi-110007, India.

Practical Report Submitted to

Dr. Mamta

"32221401 - Mathematical Physics III"

Table of Contents

1 Theory

(a) Explain Hermite Gauss Quadrature method for integration. What kind of integrals are evaluated by this method?

- Hermite Gauss quadrature is a Gaussian quadrature over the interval $(-\infty, \infty)$ with weighting function $W(x) = e^{(-x^2)}$.

  The method is used for evaluating the integrals of the following kind:

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx$$

(b)Write down the Hermite differential equation and first five Hermite polynomials

- The second order Hermite Differential Equation is:

$$\frac{d^2y}{dx^2} - (2x)\frac{dy}{dx} + 2ky = 0$$

  Where,the constant k can be any real number.

- The Rodrigues representation for the Hermite polynomials is

$$H_n(y) = (-1)^n e^{y^2} \frac{d^n}{dy^n}(e^{-y^2})$$

  And first five Hermite Polynomials are as follows:

$$H_0(y) = 1 \tag{1}$$

$$H_1(y) = 2y \tag{2}$$

$$H_2(y) = 4y^2 - 2 \tag{3}$$

$$H_3(y) = 8y^3 - 12y \tag{4}$$

$$H_4(y) = 16y^4 - 48y^2 + 12 \tag{5}$$

(c)Write down the recursion formulae and orthogonality conditions for these polynomials.

- The recurrence relations are:

  1. $H_n'(x) = 2nH_{n-1}(x)$
  2. $2xH_n(x) = 2nH_{n-1}(x) + H_{n+1}(x)$

- The Orthogonal Properties:

  The Hermite polynomials are orthonormal to each other making a complete orthonormal set

$$\int_{-\infty}^{\infty} H_n(x)H_m(x)e^{-x^2}dx = 2^n n!\sqrt{\pi}\delta_{nm}$$

  Where δ_{nm} is the Kronecker Delta Function

$$\delta_{nm} = 0 \,\forall\; n \neq m$$

$$\delta_{nm} = 1 \,\forall\; n = m$$

(d) Explicitly derive the 2-point quadrature formula for this method.

We have,

$$H_o(x) = 1$$

$$H_1(x) = 2x$$

$$H_2(x) = 4x^2 - 2$$

Gauss-Hermite Quadrature formula

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

For 2-point Gauss-Hermite (n=2)

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = W_1 f(x_1) + W_2 f(x_2)$$

Where x_1 and x_2 are the abscissa and w_1 and w_2 are the weights.

The abscissa for n-point rule are the roots of the hermite function of degree n.

We have,$H_2(x) = 4x^2 - 2$. The roots of $H_2(x) = 0$ are the abscissa for 2-point Gauss-Hermite rule.

$$\implies 4x^2 - 2 = 0$$

$$x_1 = \frac{1}{\sqrt{2}}\ ,\ x_2 = \frac{-1}{\sqrt{2}}$$

To find the w_1 and w_2 weights, we use $H_o(x)$ and $H_1(x)$ to find the relationship eq.

Using $H_o(x) = 1$

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = w_1 f(x_1) + w_2 f(x_2)$$

$$\int_{-\infty}^{\infty} e^{-x^2}(1)dx = w_1 + w_2 \tag{6}$$

Integration of $\int_{-\infty}^{\infty} e^{-x^2}dx \quad (I_1)$is:

$I_1 = \int_{-\infty}^{\infty} e^{-x^2}dx = \int_{-\infty}^{\infty} e^{-y^2}dy$ (dummy variables)

$$I_1^2 = \int_{-\infty}^{\infty} e^{-x^2}dx. \int_{-\infty}^{\infty} e^{-y^2}dy$$

$$= \iint_{-\infty}^{\infty} e^{-(x^2+y^2)}dxdy$$

Convert to polar coordinate, $r = \sqrt{x^2+y^2}$, element of area $dx.dy = r.dr.d\theta$

Limits are $r = 0$ to ∞ , $\theta = 0$ to $\theta = 2\pi$

$$I_1^2 = \int_0^{2\pi}\int_0^{\infty} e^{-r^2}.r.dr.d\theta = 2\pi \int_0^{\infty} re^{-r^2}dr$$

Substitute $t = r^2.dt = 2\pi r.dr$ limits of $t = 0$ to $t = 2pi$

$$I_1^2 = 2\pi \int_0^{\infty} \frac{e^{-t}}{2}dt = \pi[-(e^{\infty} - e^0)] = \pi$$

$$I_1 = \sqrt{\pi}$$

Putting in equation 6. we get,

$$w_1 + w_2 = \sqrt{\pi} \tag{7}$$

Using $H_1(x) = 2x$

$$\int_{-\infty}^{\infty} e^{-x^2}(2x)dx = w_1 f\left(\frac{1}{\sqrt{2}}\right) + w_2 f\left(\frac{-1}{\sqrt{2}}\right) \tag{8}$$

Integration of $\int_{-\infty}^{\infty} e^{-x^2}(2x)dx \quad (I_2)$

$$I_2 = \int_{-\infty}^{\infty} e^{-x^2}(2x)dx$$

$$x^2 = t \implies 2xdx = dt$$

$$I_2 = \int_{-\infty}^{\infty} e^{-t}dt = [-e^{-t}]_{\infty}^{\infty}$$

$$I_2 = [-(e^{-\infty} - e^{\infty}] = 0$$

$$I_2 = 0$$

Putting back in equation 8 we get,

$$w_1(\sqrt{2}) + w_2(\sqrt{-2}) = 0 \tag{9}$$

Solving eq.7 and eq.9 we get

$$w_1 = \frac{\sqrt{\pi}}{2} \ , \ w_2 = \frac{\sqrt{\pi}}{2}$$

<u>2-point Gauss-Hermite Quadrature Formula:</u>

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = \frac{\sqrt{\pi}}{2}\left[f\left(\frac{1}{\sqrt{2}}\right) + f\left(\frac{-1}{\sqrt{2}}\right)\right]$$

2 Algorithm

Algorithm 1 Hermite Polynomials

function MyHermiteQuad(f, n)

▷ *MyHermiteQuad function takes the parameter f and n*

▷ *f: function , n: no.of points*

▷ *Inbuilt function np.polynomial.hermite.hermgauss(n) takes n as a parameter and it returns two arrays of weights and points.*

```
    Integral=0
    xi,wi=np.polynomial.hermite.hermgauss(n)
```

▷ *n-point gauss-hermite quadrature Integration formula is the sum of the product of weight and points*

```
    for ( doXi,Wi) in zip (xi,wi):
        Integral+=Wi*f(Xi)
        return Integral
```

3 Programming

```
'''
Name-Monu Chaurasiya
Roll No. -2020PHY1102


Partner -
Name-Prateek Bhardwaj
Roll No. - 2020PHY1110
'''


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy import integrate
from sympy import *
from sympy import simplify
import scipy
from MyIntegration import MySimp
from MyIntegration import MyHermiteQuad


#(c)

def new_simp(f,a,R0,R_max,tol):
    lis=[]
    R_a=[]
    w=0
    a_a=[]
    while R0<=R_max:
        j=MySimp(f,-R0,R0,2,key1=True,N_max=10**8,key2=True,tol=0.1e
    -5)
        lis.append(j[0])
        R_a.append(R0)
        a_a.append(-R0)
        if len(lis)>=2:
            if lis[-1]<=0.1e-5:
                err=abs(lis[-1]-lis[-2])
            else:
                err=abs((lis[-1]-lis[-2])/lis[-1])
            if err<=tol:
                w=1
                break
            else:
                pass
        R0=10*R0
    if w==0:
            s=("R_max reached without achieving required tolerance")
    elif w==1:
             s="Given tolerance achieved with R=",R_a[-1]
    return lis[-1],R_a[-1],s,lis,R_a,a_a      #returning integral,
```

```
number of intervals and message


#Q3(b)
#(i)
n=2
f_x=["1","x","x**2","x**3","x**4","x**5"]
Calc=[]
Exact=[1.7724538509,0,0.88622692545,0,1.329340388,0]

for i in range(0,6):
    print(i+1,"th function")
    f=eval("lambda x:"+input("Enter the value of the FUNCTION F(x): "))
    Calc.append(MyHermiteQuad(f, n))


data={"f(x)":f_x,"Calculated":Calc,"Exact":Exact}
print()
print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
print()
print("METHOD USED : Gauss Hermite quadrature (TWO POINT)")
print(pd.DataFrame(data))
print()
print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
print()


n=4
f_x=["1","x","x**2","x**3","x**4","x**5","x**6","x**7","x**8","x**9"]
Calc=[]
Exact=[1.7724538509,0,0.886226925,0,1.329340388
,0,3.32335097044,0,11.63172839,0]
for i in range(0,10):
    print(i+1,"th function")
    f=eval("lambda x:"+input("Enter the value of the FUNCTION F(x): "))
    Calc.append(MyHermiteQuad(f, n))

data={"f(x)":f_x,"Calculated":Calc,"Exact":Exact}
print()
print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
print()
print("METHOD USED : Gauss Hermite quadrature (FOUR POINT)")
print(pd.DataFrame(data))
print()
print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
print()






#(ii)
```

```
I1_a=[]
I2_a=[]
n_a=[2,4,8,16,32,64,128]
f1=lambda x : 1/(1+x**2)
f2=lambda x : np.exp(x**2)/(1+x**2)
for n in n_a:
    I1_a.append(MyHermiteQuad(f1, n))
    I2_a.append(MyHermiteQuad(f2, n))

DataOut = np.column_stack((n_a,I1_a,I2_a))
np.savetxt("quad-herm-1110", DataOut,delimiter=',')

print("#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#")
print()
data={"n":n_a,"I1":I1_a,"I2":I2_a}
print(pd.DataFrame(data))

q1=np.array([1.3432934216]*len(I1_a))
q2=np.array([np.pi]*len(I2_a))
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('GAUSS HERMITE QUADRATURE')
ax1.plot(n_a,I1_a,marker="*",label="Calculated",linestyle='dashed')
ax1.plot(n_a,q1,label="EXACT",c="red",linewidth=1)
ax2.plot(n_a,I2_a,marker="*",label="Calculated",linestyle='dashed')
ax2.plot(n_a,q2,label="EXACT",c="red",linewidth=1)
ax1.grid()
ax1.legend()
ax1.set(xlabel="N",ylabel="Integral",title="Integral I1 ")
ax2.set(xlabel="N",ylabel="Integral",title="Integral I2 ")
ax2.grid()
ax2.legend()
plt.show()


a=0
R0=10
R_max=10**6
tol=0.1e-8
F1=lambda x : np.exp(-1*x**2)/(1+x**2)
F2=lambda x : 1/(1+x**2)


s2=new_simp(F1,-R0,R0,R_max,tol)
s3=new_simp(F2,-R0,R0,R_max,tol)

print()
print("#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#")
print()
print("RESULTS USING SIMPSSON METHOD")
print("Tolerance for MYSimp defined in MyIntegration Module = 0.1e-5")
print("Tolerance for the value of Integral with respect to value of b(upper limit) = 0.1e-8")
print()
```

```
print("INTEGRAL I1")
print()
print("Limit -R to R")
data={"a(lower limit)":s2[5],"b(upper limit)":s2[4],"Integral I1":s2
    [3]}
print(pd.DataFrame(data))


print("INTEGRAL I2")
print()
print("Limit -R to R")
data={"a(lower limit)":s3[5],"b(upper limit)":s3[4],"Integral I2":s3
    [3]}
print(pd.DataFrame(data))

q1=np.array([1.3432934216]*len(s2[3]))
q2=np.array([np.pi]*len(s3[3]))
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('SIMPSON METHOD (TOLERANCE = 0.1e-8)')
ax1.plot(s2[4],s2[3],marker="*",label="I1 using SIMPSON",linestyle='
    dashed')
ax1.plot(s2[4],q1,label="EXACT",c="red",linewidth=1)
ax2.plot(s3[4],s3[3],marker="*",label="I1 using SIMPSON",linestyle='
    dashed')
ax2.plot(s3[4],q2,label="EXACT",c="red",linewidth=1)
ax1.grid()
ax1.legend()
ax1.set(xlabel="R",ylabel="Integral",title="Integral I1 calculated
    using SIMPSON")
ax2.set(xlabel="R",ylabel="Integral",title="Integral I2 calculated
    using SIMPSON")
ax2.grid()
ax2.legend()
plt.show()
```

4 Result and Discussion

```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

METHOD USED : Gauss Hermite quadrature (TWO POINT)
   f(x)   Calculated      Exact
0      1     1.772454  1.772454
1      x     0.000000  0.000000
2   x**2     0.886227  0.886227
3   x**3     0.000000  0.000000
4   x**4     0.443113  1.329340
5   x**5     0.000000  0.000000

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
METHOD USED : Gauss Hermite quadrature (FOUR POINT)
   f(x)      Calculated       Exact
0      1  1.772454e+00    1.772454
1      x  2.775558e-17    0.000000
2   x**2  8.862269e-01    0.886227
3   x**3  0.000000e+00    0.000000
4   x**4  1.329340e+00    1.329340
5   x**5  1.110223e-16    0.000000
6   x**6  3.323351e+00    3.323351
7   x**7  0.000000e+00    0.000000
8   x**8  8.973048e+00   11.631728
9   x**9  0.000000e+00    0.000000
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

```
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#

     n        I1        I2
0    2  1.181636  1.948188
1    4  1.306019  2.328295
2    8  1.339187  2.588464
3   16  1.343129  2.761919
4   32  1.343292  2.879063
5   64  1.343293  2.958943
6  128  1.343293  3.013879

#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
```

For gauss-hermite quadrature we verify the results for n=2 and n=4 and we can se it shows correct value upto 2n-1 degree and that we can see clearly in output table.

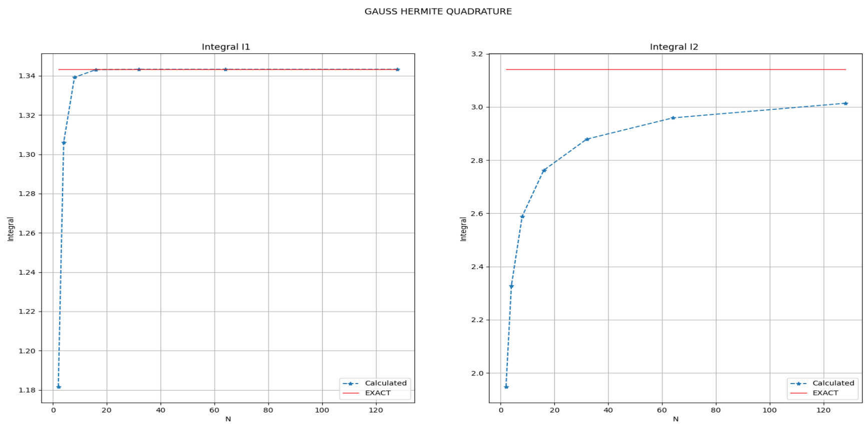

Figure 1: Integral value vs No. of intervals

From the graph we can see that the integral value I_1 approaches to true value very rapidly i.e it converges for increasing value of n. The graph overlap over the exact value line for n=20.

For integral value I_2 graph we can see that the graph approaching towards true value as n increases but it doesn't overlap over the exact value line and the integral I_2 graph line shows exponential growth due to presence of exponential term in the integrand.

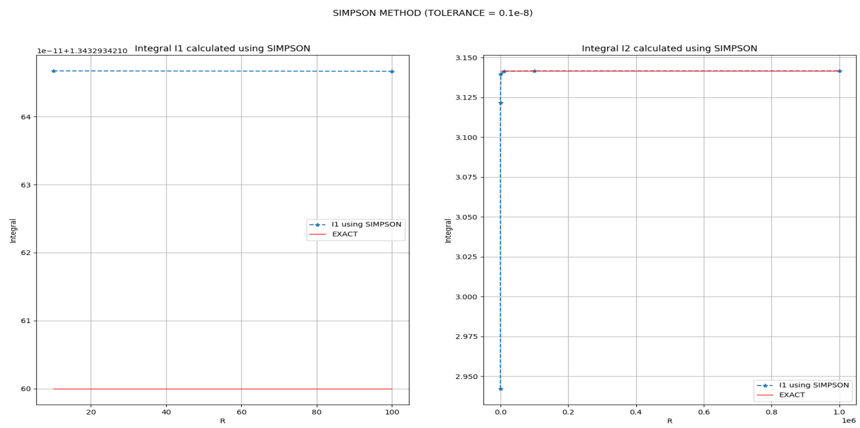

Figure 2: Gibbs Phenomenon

It is the graph between integral value vs the limits that we are using instead of infinity to check for a Given tolerance what is the limit that gives us a exact or true value or we can say that we are finding a value of limit from there the integral start converging.

For I_1 we can see that the integral approaching towards exact value for b=100 i.e the given tolerance is achieved . And I_2 converges for the value of b=0.1*10