GAUSS LAGUERRE QUADRATURE

Pawanpreet Kaur
(2020PHY1092)
(20068567038)

Kabir Sethi
(2020PHY1098)
(20068567031)

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

1.1 Laguerre-Gauss Quadrature

The gaussian quadrature is used to calculate various types of integral having discontinuties and infinite limit for that purpose we introduce a new family of orthogonal polynomials Laguerre Polynomials which are known to be the solution of laguerre's second order linear differential equation.
Using this set of polynomials we can calculate the integrals of the below type by just including the weighting function specific to the laguerre polynomials.

$$xy" + (1-x)y' + ny = 0 \qquad \text{where } n \in \mathbb{Z}_+$$

The weighting function and integration interval of these polynomials is -

$$w(x) = e^{-x} \qquad \text{on } [0,\infty)$$

We can write it as

$$\int_0^\infty f(x)dx$$

with the weighting function e^{-x} and in Summation form, we can write it as

$$\int_0^\infty f(x)e^{-x}dx = \sum_1^n w_i f(x_i)$$

where n is the n point quadrature used.We can write

$$\int_0^\infty f(x)dx = \int_0^\infty f(x)e^x e^{-x}dx = \int_0^\infty g(x)e^{-x}dx$$

$$g(x) := f(x)e^x$$

1.2 Laguerre polynomials

We can wrute the recurrence relation from the solution of the given above equation as

$$L_{n+1}(x) = (1 - x + 2n)L_n - n^2 L_{n-1}(x)$$

With the knowledge of polynomials $L_0 and L_1$ we can determine all the other polynomials

$$L_n(x) = \sum_{k=0}^{n} \frac{(-1)^{n-k}(n!)^2}{(n-k)!^2 k!} x^{n-r}$$

From these relations we can derive first five Laguerre polynomials

$$L_0 = 1$$
$$L_1 = -x + 1$$
$$L_2 = \frac{1}{2}(x^2 - 4x + 2)$$
$$L_3 = \frac{1}{6}(-x^3 + 9x^2 - 18x + 6$$
$$L_4 = \frac{1}{24}(x^4 - 16x^3 + 72x^2 - 96x + 24$$
$$L_5 = \frac{1}{120}(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120$$

Orthogonality relation

The orthogonality relation for Laguerre Polynomials is

$$\int_0^\infty L_m(x) L_n(x) e^{-x} dx = (n!)^2 \delta_{mn}$$

Where δ_{mn} is the kronecker delta function

1.3 2-point Laguerre gauss quadrature

The degree to which we can find accurate result can be found out by

$$2n-1$$

Where n is the n-point we are using.

Since this formula is to have degree of precision equal to 3,Then the weights and abcissas must satisfy for constant,linear,quadratic and cubic functions.

1. **For** $f(x)=1$

$$\int_0^\infty 1e^{-x}dx = w_1 f(x_1) + w_2 f(x_2)$$
$$w_1 + w_2 = [-e^{-x}]_0^\infty$$
$$w_1 + w_2 = 1 \tag{1}$$

2. **For** $f(x)=x$

$$\int_0^\infty xe^{-x}dx = w_1 f(x_1) + w_2 f(x_2)$$
$$w_1x_1 + w_2x_2 = [-(x+1)e^{-x}]_0^\infty$$
$$w_1x_1 + w_2x_2 = 1 \tag{2}$$

3. **For** $f(x)=x^2$

$$\int_0^\infty x^2e^{-x}dx = w_1 f(x_1) + w_2 f(x_2)$$
$$w_1x_1^2 + w_2x_2^2 = [-(x^2+2x+2)e^{-x}]_0^\infty$$
$$w_1x_1^2 + w_2x_2^2 = 2 \tag{3}$$

4. **For** $f(x)=x^3$

$$\int_0^\infty x^3e^{-x}dx = w_1 f(x_1) + w_2 f(x_2)$$
$$w_1x_1^3 + w_2x_2^3 = [-(x^3+3x^2+6x+6)e^{-x}]_0^\infty$$
$$w_1x_1^3 + w_2x_2^3 = 6 \tag{4}$$

With the given equations and solving then, we found out

x_i	w_i
$2-\sqrt{2}$	$\frac{1}{4}(2+\sqrt{2})$
$2+\sqrt{2}$	$\frac{1}{4}(2-\sqrt{2})$

Table 1: Result

2 Algorithm

Algorithm 1 MyLaguQuad

function MYLAGUQUAD:(f,n)

▷ *MyLaguQuad function takes the parameter f and n* ◁

▷ *f: function* ◁

▷ *n: no.of points* ◁

Integral=0 ▷ *Declaring variable to store the value of integral*

xi, wi=np.polynomial.laguerre.laggauss(n) ▷ *Inbuilt function np.polynomial.laguerre.laggauss(n) takes n as a parameter and it returns two arrays of weights and points.*

for all (Xi, Wi) in zip (xi, wi): **do**

Integral+=$Wi * f(Xi)$ ▷ *n-point gauss-laguerre quadrature Integration formula is the sum of the product of weight and points*

return Integral ▷ *Returns the value of integral*

3 Discussion

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

METHOD USED : Gauss Laguerre quadrature (TWO POINT)
   f(x)  Calculated  Exact
0     1         1.0      1
1     x         1.0      1
2  x**2         2.0      2
3  x**3         6.0      6
4  x**4        20.0     24
5  x**5        68.0    120

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

Figure 1: *Two-Point Gauss Laguerre Quadrature for different functions.*

From the above table , it can be understood that two point quadrature gives exact results for the polynomials of degree 3 and less but for the polynomials of degree 4 or more , it does not give exact results.

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

METHOD USED : Gauss Laguerre quadrature (FOUR POINT)
   f(x)  Calculated   Exact
0     1         1.0       1
1     x         1.0       1
2  x**2         2.0       2
3  x**3         6.0       6
4  x**4        24.0      24
5  x**5       120.0     120
6  x**6       720.0     720
7  x**7      5040.0    5040
8  x**8     39744.0   40320
9  x**9    339264.0  362880

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

Figure 2: *Four-Point Gauss Laguerre Quadrature for different functions.*

From the above table , it can be understood that four point quadrature gives exact results for the polynomials of degree 7 and less but for the polynomials of degree 8 or more , it does not give exact results.

So, it can be concluded that npoint quadrature formula gives exact result when $f(x)$ is a polynomial of order 2n 1 or less.

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

     n        I1        I2
0    2  0.647059  1.493257
1    4  0.636427  1.501190
2    8  0.620075  1.533760
3   16  0.621507  1.553738
4   32  0.621449  1.562483
5   64  0.621450  1.566725
6  128  0.621450  1.568789

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

Figure 3: *Gauss Laguerre Quadrature for different values of n for integral I1 and I2.*

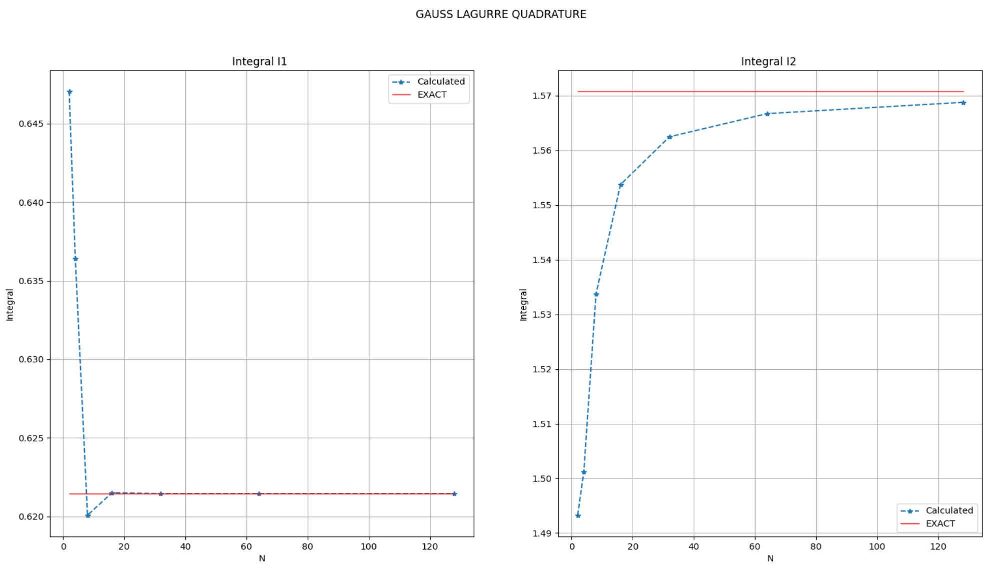

Figure 4: *Integral vs n plot for Gauss Laguerre Quadrature.*

From the graph we can see that the integral value I1 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 I2 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 I2 graph line shows exponential growth due to presence of exponential term in the integrand.

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

RESULTS USING SIMPSON METHOD
Tolerance for MYSimp defined in MyIntegration Module = 0.1e-5
Tolerance for the value of Integral with respect to value of b(upper limit) = 0.1e-7

   a(lower limit)  b(upper limit)  Integral I1
0               0              10     0.621449
1               0             100     0.621450
2               0            1000     0.621450
3               0           10000     0.621450
4               0          100000     0.621450

   a(lower limit)  b(upper limit)  Integral I2
0               0              10     1.471128
1               0             100     1.560797
2               0            1000     1.569796
3               0           10000     1.570696
4               0          100000     1.570786
5               0         1000000     1.570795

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

Figure 5: *Calculation of Integral I1 and I2 using Simpson Method*

It can be seen that required tolerance is achieved for R=1000000

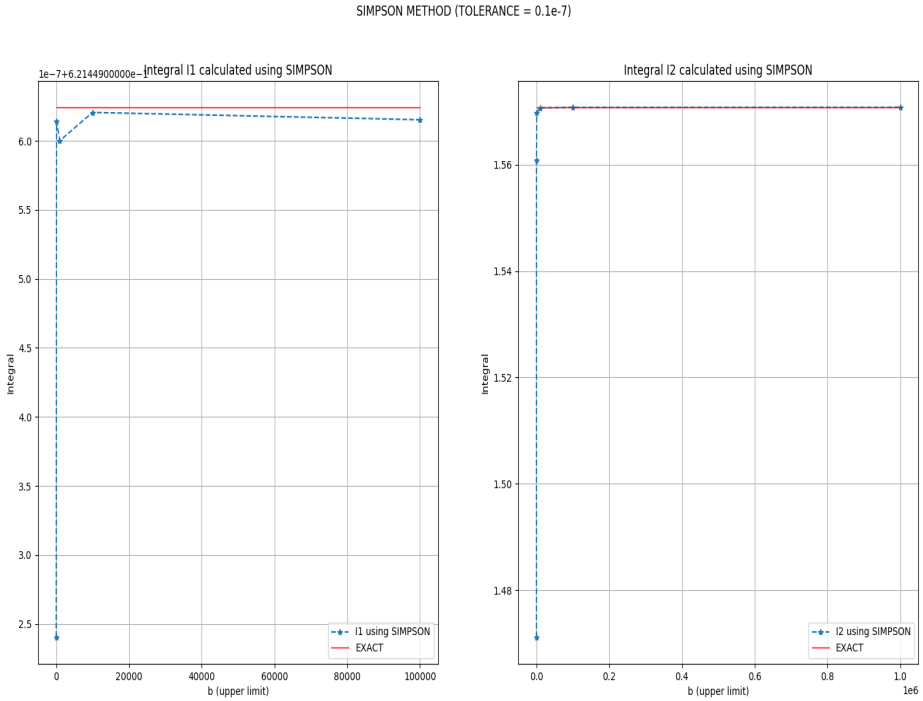

Figure 6: *Integral vs R plot for Integral I1 and I2 calculated using Simpson Method.*

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 value correct upto certain significant digits.

For I1 we can see that the integral approaching towards exact value for b=1000 but after that it start diverging . And I2 converges for the value of b=0.1*10

4 Program

```
'''

Name-Kabir Sethi
Roll No. - 2020PHY1097

Partner -
Name-Pawanpreet Kaur
Roll No. - 2020PHY1092
'''


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 MyLaguQuad

#(c)
print("Name-Kabir Sethi \n Roll No. - 2020PHY1097")
def new_simp(f,a,R0,R_max,tol):
    lis=[]
    R_a=[]
    w=0
    a_a=[]
    while R0<=R_max:
        j=MySimp(f,a,R0,2,key1=True,N_max=10**8,key2=True,tol=0.1e-5)
        #j=MySimp(f,a,R0,2,key1=False)
        lis.append(j[0])
        R_a.append(R0)
        a_a.append(a)
        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
```

```
52
53
54 #Q3(b)
55 #(i)
56 n=2
57 f_x=["1","x","x**2","x**3","x**4","x**5"]
58 Calc=[]
59 Exact=[1,1,2,6,24,120]
60 for i in range(0,6):
61     print(i+1,"th function")
62     f=eval("lambda x:"+input("Enter the value of the FUNCTION F(x): ")
       )
63     Calc.append(MyLaguQuad(f, n))
64 data={"f(x)":f_x,"Calculated":Calc,"Exact":Exact}
65 print()
66 print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
67 print()
68 print("METHOD USED : Gauss Laguerre quadrature (TWO POINT)")
69 print(pd.DataFrame(data))
70 print()
71 print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
72 print()
73
74
75 n=4
76 f_x=["1","x","x**2","x**3","x**4","x**5","x**6","x**7","x**8","x**9"]
77 Calc=[]
78 Exact=[1,1,2,6,24,120,720,5040,40320,362880]
79 for i in range(0,10):
80     print(i+1,"th function")
81     f=eval("lambda x:"+input("Enter the value of the FUNCTION F(x): ")
       )
82     Calc.append(MyLaguQuad(f, n))
83
84 data={"f(x)":f_x,"Calculated":Calc,"Exact":Exact}
85 print()
86 print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
87 print()
88 print("METHOD USED : Gauss Laguerre quadrature (FOUR POINT)")
89 print(pd.DataFrame(data))
90 print()
91 print("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*")
92 print()
93
94
95 #(ii)
96 I1_a=[]
97 I2_a=[]
98 n_a=[2,4,8,16,32,64,128]
99 f1=lambda x : 1/(1+x**2)
100 f2=lambda x : np.exp(x)/(1+x**2)
101 for n in n_a:
102     I1_a.append(MyLaguQuad(f1, n))
103     I2_a.append(MyLaguQuad(f2, n))
104
```

```python
DataOut = np.column_stack((n_a,I1_a,I2_a))
np.savetxt("quad-lag-1092", DataOut,delimiter=',')

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


q1=np.array([0.621449624235]*len(I1_a))
q2=np.array([np.pi/2]*len(I2_a))
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('GAUSS LAGURRE 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-7
F1=lambda x : np.exp(-1*x)/(1+x**2)
F2=lambda x : 1/(1+x**2)
s1=new_simp(F1,a,R0,R_max,tol)
s2=new_simp(F2,a,R0,R_max,tol)

#d
print()
print("#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
")
print()
print("RESULTS USING SIMPSON 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-7")
print()
data={"a(lower limit)":s1[5],"b(upper limit)":s1[4],"Integral I1":s1[3]}
print(pd.DataFrame(data))
print()
data={"a(lower limit)":s2[5],"b(upper limit)":s2[4],"Integral I2":s2[3]}
print(pd.DataFrame(data))
```

```
print()
print("
    #-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
    ")
print()

q1=np.array([0.621449624235]*len(s1[3]))
q2=np.array([np.pi/2]*len(s2[3]))
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('SIMPSON METHOD (TOLERANCE = 0.1e-7)')
ax1.plot(s1[4],s1[3],marker="*",label="I1 using SIMPSON",linestyle='
    dashed')
ax1.plot(s1[4],q1,label="EXACT",c="red",linewidth=1)
ax2.plot(s2[4],s2[3],marker="*",label="I2 using SIMPSON",linestyle='
    dashed')
ax2.plot(s2[4],q2,label="EXACT",c="red",linewidth=1)
ax1.grid()
ax1.legend()
ax1.set(xlabel="b (upper limit)",ylabel="Integral",title="Integral I1
    calculated using SIMPSON")
ax2.set(xlabel="b (upper limit)",ylabel="Integral",title="Integral I2
    calculated using SIMPSON")
ax2.grid()
ax2.legend()
plt.show()
```