Gauss-Laguerre

SGTB Khalsa College, University of Delhi

Preetpal Singh(2020PHY1140)(20068567043)

Ankur Kumar(2020PHY1113)(20068567010)

Unique Paper Code: 32221401

Paper Title: Mathematical Physics III

Submitted on: February 8, 2022

B.Sc(H) Physics Sem IV

Submitted to: Dr. Mamta

Contents

1 Theory

Laguerre Gauss Quadrature method for Integration

Laguerre-Gauss quadrature is an extension of Gaussian quadrature over the interval $[0, \infty)$. It fits all polynomials of degree $2m-1$ exactly. This method approximates the value of integrals of the following kind:

$$\int_0^\infty f(x)dx = \int_0^\infty f(x)e^x e^{-x}dx = \int_0^\infty g(x)e^{-x}dx \approx \sum_{i=1}^n w_i f\left(x_i\right)$$

here $g(x) = e^x f(x)$, x_i is the i-th root of Laguerre polynomial $L_n(x)$ and the weight w_i is,

$$w_i = \frac{x_i}{(n+1)^2 \left[L_{n+1}\left(x_i\right)\right]^2}$$

Laguerre differential equation

The Laguerre polynomials are solutions of second-order linear Laguerre's differential equation:

$$xy'' + (1-x)y' + ny = 0$$

The Laguerre polynomials are defined by given formula,

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}\left(e^{-x}x^n\right)$$

The first five Laguerre polynomials are denoted by L_0, L_1, L_2, L_3, L_4, which are following,

$$L_0 = 1$$

$$L_1 = -x + 1$$

$$L_2 = \frac{1}{2}\left(x^2 - 4x + 2\right)$$

$$L_3 = \frac{1}{6}\left(-x^3 + 9x^2 - 18x + 6\right)$$

$$L_4 = \frac{1}{24}\left(x^4 - 16x^3 + 72x^2 - 96x + 24\right)$$

Recursion Formulae for Laguerre Polynomials

$$L_{n+1}(x) = (2n+1-x)L_n(x) - n^2 L_{n-1}(x)$$
$$nL_{n-1}(x) = nL'_{n-1}(x) - L'_n(x)$$
$$xL'_n(x) = -nL_n(x) - n^2 L_{n-1}(x)$$

Orthogonality properties for Laguerre Polynomials

$$\int_0^\infty e^{-x} L_m(x) L_n(x) dx = 0 \quad when \quad m \neq n$$
$$\int_0^\infty e^{-x} \left[L_n(x)\right]^2 dx = (n!)^2 \quad when \quad m = n$$

2-point Quadrature Formula for Gauss Laguerre Polynomials

As we know for N point method,

$$I_n = \sum_{i=1}^{N} w_i g\left(x_i\right)$$
$$I_n = \int_0^\infty g(x) e^{-x} dx = \omega_0 g\left(x_0\right) + \omega_1 g\left(x_1\right) + \cdots + \omega_n g\left(x_n\right)$$

For 2 point formula,

$$I(g) = \int_0^\infty g(x) e^{-x} dx = \omega_0 g\left(x_0\right) + \omega_1 g\left(x_1\right) = I_2(g)$$

As there are 4 unknowns $x_0, x_1, \omega_0, \omega_1$ so approximating these with 4 polynomials of degree less than or equal to 3.

$$g(x) = 1, x, x^2, x^3$$

$$g(x) = 1 \quad \int_0^\infty g(x) e^{-x} dx = \int_0^\infty e^{-x} dx = \omega_0 g\left(x_0\right) + \omega_1 g\left(x_1\right) = \omega_0 + \omega_1 = 1$$
$$g(x) = x \quad \omega_0 x_1 + \omega_1 x_2 = 1$$
$$g(x) = x^2 \quad \omega_0 x_1^2 + \omega_1 x_2^2 = 2$$
$$g(x) = x^3 \quad \omega_0 {x_1}^3 + \omega_1 {x_2}^3 = 6$$

Solving these 4 equations, we'll get

$$x_1 = 0.585786$$
$$x_2 = 3.41421$$
$$\omega_1 = 0.853553$$
$$\omega_2 = 0.146447$$

2 Algorithm

Algorithm 1 n-point Gauss Laguerre Quadrature rule

function MyLaguQuad(f, n)
  $[lagu\_zer, w] = l\_roots(n)$ ▷ Store the x values and weights in two lists
  lagu = 0 ▷ initialize the summation
  for i in range(1, n+1):
    $lagu+ = f(lagu\_zer[i-1]) * w[i-1]$ ▷ loop to sum all values for the integral
  return $lagu$ ▷ Returns the value of integral

3 Programming

```
#Ankur Kumar 2020PHY1113
#Preetpal Singh 2020PHY1140


from IntegrationModule import *
from scipy.integrate import quad
import numpy as np
import pandas as pd
from prettytable import PrettyTable

print("2020PHY1113")

ver_f = eval("lambda x:" + input("function to be integrated by Guass Laguerre, f(x)
    = "))

x = PrettyTable()

x.field_names = ["n-point", "Inbuilt Function", "My Function"]

x.add_row(["2", quad(lambda x: np.exp(-x)*ver_f(x), 0, np.inf)[0], MyLaguQuad(ver_f
    , 2)])
x.add_row(["4", quad(lambda x: np.exp(-x)*ver_f(x), 0, np.inf)[0], MyLaguQuad(ver_f
    , 4)])

print(x)

def Int_1(x):
    return 1/(1+x**2)

def Int_2(x):
    return np.exp(x)/(1+x**2)

I1 = []
I2 = []
n = []
for i in range(1, 8):
    val_1 = MyLaguQuad(Int_1, 2**i)
    I1.append(val_1)


for i in range(1, 8):
    val_2 = MyLaguQuad(Int_2, 2**i)
    I2.append(val_2)
    n.append(2**i)

data = {

  'n': n,
```

```python
  'I_1': I1,
  'I_2': I2

}

df = pd.DataFrame(data)
numpy_array = df.to_numpy()
np.savetxt("quad-lag-1113.out.txt", numpy_array, fmt = "%f")
print(df)

#Comparsion With Simpson Method

func_1 = lambda x: np.exp(-x)/(1+x**2)
func_2 = lambda x: 1/(1+x**2)

simp1 = []
simp2 = []
upperLimit = []
for i in range(0,5):

    temp1 = My_Simp(func_1, 0, 10**i, 100000)
    simp1.append(temp1)

    temp2 = My_Simp(func_2, 0, 10**i, 100000)
    simp2.append(temp2)

    temp3 = 10**i
    upperLimit.append(temp3)


data1 = {

    'Upper Limit': upperLimit,
  'I_1(Simpson)': simp1,
  'I_2(Simpson)': simp2

}

df = pd.DataFrame(data1)
print(df)
```

4 Discussion

Verification of n-point Gauss Laguerre

We know that for n-point Gauss Laguerre the method will give exact results upto $2n-1$ order of polynomials.
For $n = 2$ the integral is exact upto 3rd order polynomials.
For $n = 4$ the integral is exact upto 7th order polynomials.

function to be integrated by Guass Laguerre, f(x) = x**2

n-point	Inbuilt Function	My Function
2	2.0	2.0000000000000004
4	2.0	2.0000000000000018

Figure 1: Polynomial of Order 2

For the given 2nd order polynomial it can be seen that both 2-point and 4-point quadrature gives exact results.

function to be integrated by Guass Laguerre, f(x) = x**5

n-point	Inbuilt Function	My Function
2	120.0	68.00000000000003
4	120.0	120.00000000000023

Figure 2: Polynomial of Order 5

For the given 5th order polynomial it can be seen that 2-point quadrature fails to give an exact result and 4-point is still accurate as predicted.

function to be integrated by Guass Laguerre, f(x) = x**9

n-point	Inbuilt Function	My Function
2	362879.9999999994	9232.000000000007
4	362879.9999999994	339264.00000000116

Figure 3: Polynomial of Order 9

For the given 9th order polynomial it can be seen that both 2-point and 4-point quadrature fail to give exact results.

Evaluating I_1 and I_2

	n	I_1	I_2
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 4: Value of Integrals with change in n

It can be seen that as value of n increases the value of integral approaches the true value.

Comparison with Simpson Method

	Upper Limit	I_1(Simpson)	I_2(Simpson)
0	1	0.524797	0.785398
1	10	0.621449	1.471128
2	100	0.621450	1.560797
3	1000	0.621450	1.569796
4	10000	0.621449	1.570696

Figure 5: Comparison with Simpson Method

Since we know that the lower limit is 0 we keep it constant and change the value of the upper limit towards infinity.
It can be seen that as the value of upper limit goes on increasing both the integrals approach their true value.