Hermite-Gauss Quadrature
Lab Report for Assignment No. 5

SHASHVAT JAIN
(2020PHY1114)

HARSH SAXENA
(2020PHY1162)

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

March 10, 2022

Submitted to Dr. Mamta
"32221401 - MATHEMATICAL PHYSICS III"

Contents

1 Theory

1.1 Hermite-Gauss Quadrature

One such family of Polynomials is Hermite Polynomials which are known to be the solution of Hermite differential equation which is a second order linear differential equation.

$$y" + -2xy' + \lambda y = 0 \qquad \text{where} \lambda \in \mathbb{R} \tag{1}$$

Whenever $\lambda \in \mathbb{N}$ the solution of this differential equation gives the Hermite polynomial.
The weighting function and integration interval of these polynomials is -

$$w(x) = e^{-x^2} \qquad \text{on} \qquad [-\infty, \infty) \tag{2}$$

It is evident by seeing the limit of integration that this gaussian quadrature rule is used in calculating the improper of integral of type.

$$\int_{-\infty}^{\infty} f(x)dx \tag{3}$$

Specifically when we include the weighting function e^{-x^2} then.

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

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

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} f(x)e^{x^2}e^{-x^2}dx = \int_{-\infty}^{\infty} g(x)e^{-x^2}dx$$
$$g(x) := f(x)e^{x^2}$$

1.2 Hermite polynomials

The Solution of above differential equation results in the following recurrence relation.

$$L_{n+1}(x) = (1 - x + 2n)L_n - n^2 L_{n-1}(x) \tag{4}$$

If we know the polynomials $L_0 and L_1$ we can determine all the other polynomials

$$L_n(x) = \sum_{k=0}^{n} \frac{(-1)^{n-k}(n!)^2}{(n-k)!^2 k!} x^{n-r} \tag{5}$$

From these relations we can derive first five Laguerre polynomials

$$L_0 = 1 \qquad L_1 = -x + 1 \tag{6}$$
$$L_2 = \frac{1}{2}(x^2 - 4x + 2) \tag{7}$$
$$L_3 = \frac{1}{6}(-x^3 + 9x^2 - 18x + 6 \tag{8}$$
$$L_4 = \frac{1}{24}(x^4 - 16x^3 + 72x^2 - 96x + 24 \tag{9}$$
$$L_5 = \frac{1}{120}(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120 \tag{10}$$

Orthogonality relation

$$\int_0^\infty L_m(x)L_n(x)e^{-x}dx = (n!)^2\delta_{mn} \tag{11}$$

Where δ_{mn} is the kronecker delta function

1.3 Construction of 2-point Hermite gauss quadrature

Since this formula is to have degree of precision equal to 3,Then the weights and abcissas must satisfy-that for constant,linear,quadratic and cubic functions the integral must be exatly equal to the weights multiplied by the function evaluation at these decided abcissa.

1. $f(x) = 1$

$$\int_{-\infty}^{\infty} 1e^{-x}dx = w_1f(x_1) + w_2f(x_2)$$

$$w_1 + w_2 = \sqrt{\pi} \tag{12}$$

2. $f(x) = x$

$$\int_{-\infty}^{\infty} xe^{-x^2}dx = w_1f(x_1) + w_2f(x_2) \tag{13}$$

$$w_1x_1 + w_2x_2 = [\frac{-e^{-x^2}}{2}]_{-\infty}^{\infty} \tag{14}$$

$$w_1x_1 + w_2x_2 = 0 \tag{15}$$

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

$$\int_{-\infty}^{\infty} x^2e^{-x^2}dx = w_1f(x_1) + w_2f(x_2) \tag{16}$$

$$\tag{17}$$

$$w_1x_1^2 + w_2x_2^2 = \frac{\sqrt{\pi}}{2} \tag{18}$$

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

$$\int_{-\infty}^{\infty} x^3e^{-x^2}dx = w_1f(x_1) + w_2f(x_2) \tag{19}$$

$$w_1x_1^3 + w_2x_2^3 = [-(x^2+1)e^{-x^2}]_{-\infty}^{\infty} \tag{20}$$

$$w_1x_1^3 + w_2x_2^3 = 0 \tag{21}$$

After solving the following equations we get the following weights-

x_i	w_i
-0.7071067811865475244008	0.8862269254527580136491
0.7071067811865475244008	0.8862269254527580136491

Table 1: Roots and weights for 2-point formula

2 Programming

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar  6 15:41:06 2022

@author: harsh
"""

from Myintegration import *
from scipy import stats
import matplotlib.pyplot as plt

plt.style.use('bmh')

def MyHermiteQuad(func,n):
    x,w = roots_hermite(n)
    return np.dot(w,func(x)*np.exp(x))

MyHermiteQuad = np.vectorize(MyHermiteQuad)

#
# Validation
#
listp = ['one','two','three']
func = []
P = []
K = []
for i in range(len(listp)):
    k = input(f"function {listp[i]}: ")
    K.append(k)
    func.append(lambda x,i = i: eval(K[i],{'x':x,'np':np}))


for j in range(len(func)):
    p1 = lambda x,j=j : func[j](x)*np.exp(-(x**2))
    P.append(p1)

mat = np.zeros((2,4))
n_r = [2,4]
mat[:,0] = n_r
for i in range(1,4):
   mat[:,i] = MyHermiteQuad(P[i-1],n_r)

np.savetxt("validate-herm-1162.out",mat,fmt="%.7g",delimiter=",",header="n,$I_1$,
    $I_2$")
print(mat)

######
I1 = lambda x : np.exp(-x**2)/(1+x**2)
I2 = lambda x : 1/(1+x**2)


dat = np.zeros((7,3))
dat[:,0] = 2**np.arange(1,8)
```

```
n_arr = dat[:,0]
dat[:,1] = MyHermiteQuad(I1,n_arr)
dat[:,2] = MyHermiteQuad(I2,n_arr)
np.savetxt("quad-herm-1114.out",dat,fmt="%.7g",delimiter=",",header="n,$I_1$,$I_2$")

def hermite_tol(func,n_max,rtol):
    n_arr = np.arange(5,n_max,5)
    I = np.zeros(n_arr.shape)
    r_err = np.zeros(len(n_arr)-1)

    for i in range(len(n_arr)):
        I[i] =  MyHermiteQuad(func, n_arr[i])
        if i==0:
           continue
        r_err[i-1] = abs((I[i] - I[i-1])/(I[i]))
        if r_err[i-1] <= rtol:
           return r_err[:i],I[:i],n_arr[:i]


def simpsonHermite(f,max_b= int(2e16),rtol = 0.5*10**(-4)):
    b = 2**np.arange(1,np.floor(np.log2(max_b)))
    print(b)
    I = np.zeros(b.shape)
    r_err = np.zeros(len(b)-1)
    for i,bi in enumerate(b):
        I[i] = MySimp(f,0,bi,m = int(1e7),d =5)[0]
        if i == 0:
            continue
        r_err[i-1] = abs((I[i] - I[i-1])/(I[i]))

        if r_err[i-1]<=rtol:
            return  r_err[:i],I[:i],b[:i]
    return I,b

n_arrt = np.arange(5,100,5)
inty = MyHermiteQuad(I2,n_arrt)

rl_l, inte_l,n_ar = hermite_tol(I2,500,0.5*10**(-12))
rl_s, inte_s,b_ar = simpsonHermite(I2,int(2e16),0.5*10**(-4))

slope, intercept, r_value, p_value, std_err = stats.linregress(rl_s,inte_s)
Y = slope*rl_s + intercept

print('Slope of relative tol vs Integral line for function 1/(1+x^2)',slope)
print(n_ar)
#Plotting and comparison

fig,ax = plt.subplots()
ax.plot(rl_l,inte_l,'--o',label ='improper integral  of 1/(1+x^2) by hermite in range
    -inf to inf')
ax.plot(rl_s,inte_s,'* ',label = 'improper integral by simpson')
ax.plot(rl_s,Y,label = 'Regression line')
ax.set_xlabel('relative tolerence')
ax.set_ylabel('Integral')
ax.set_title('Comparsion between integraal calculated by simpson and Hermite')
ax.legend()

fig,(ax1,ax2) = plt.subplots(1,2)
ax1.plot(n_arrt,inty,'--v')
ax1.set_xlabel('n point formula used')
ax1.set_ylabel('Integral')
ax1.set_title('A')
```

```
ax2.plot(b_ar,inte_s,'r--x')
ax2.set_xlabel('Upper limit on the integral')
ax2.set_ylabel('Integral')
ax2.set_title('B')
```

3 Results and Analysis

From the data obtained, shown below, we notice that both the integrals I_1 and I_2 converge as the Gaussian Quadrature is calculated at greater number of absiccas.(n in the n-point formuala, which is also what we expected to happen as the Gaussian quadrature is able to calculate functions differing from the usual polynomial functions more accurately with an increase in n-points.

n	I_1	I_2
2	0.9034642	1.489561
4	1.101324	1.558053
8	1.153702	1.575974
16	1.159536	1.578104
32	1.159775	1.578192
64	1.159777	1.578193
128	1.159777	1.578193

Table 2: Data from quad-herm-1114.out

However we also expect this change in integral to die out which is explained by the second figure. The simpson method seems to converge much more sharply, but this is really not the case, do not forget that in simpson's case we are increasing the upper-bound thus introducing many more sub-intervals to compute the formula over whereas in the case of n-points, we are varying n-points not so much in comparison to Simpson.

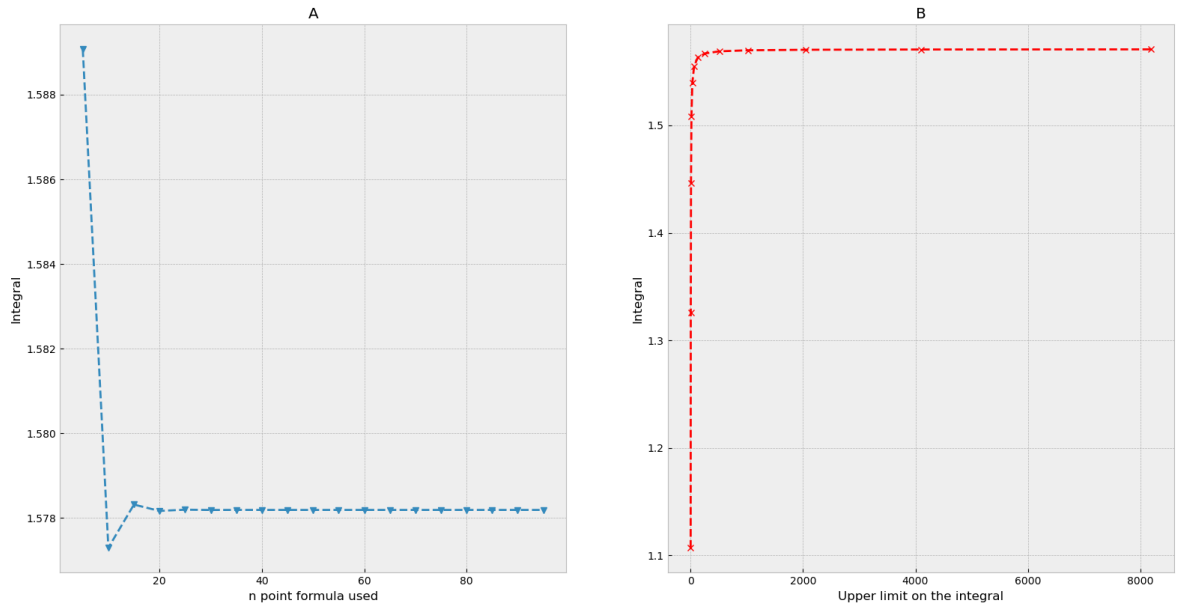

Figure 1: (A) Hermite-Gauss Quadrature (B) Simpson's composite rule to calculate $f(x) = \dfrac{1}{1+x^2}$