EM 215 – NUMERICAL METHODS

LAB ASSIGNMENT 2

MADHUSHAN R.

E/17/194

SEMESTER 4

26.12.2020

Question:

Use four iterations of Romberg integration to estimate $\int_0^1 \frac{4}{1+x^2}dx = \pi$

Comment on the accuracy of your result.

You may use the given following MatLab codes for generating the Romberg table and the error table.

(Used python to the entire lab)

Exact result will be $\pi = 3.141592653589793 \ldots$

Romberg Table

$R_{(i,j)}$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	3.0			
$i = 2$	3.1	3.1333333333333333		
$i = 3$	3.1311764705882354	3.1415686274509804	3.1421176470588237	
$i = 4$	3.138988494491089	3.141592502458707	3.1415940941258884	3.1415857837618737

Error Table

$E_{(i,j)}$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	0.14159265358979312			
$i = 2$	0.04159265358979303	0.0082593202564598		
$i = 3$	0.01041618300155767	2.402613881269e-05	-0.000524993469031	
$i = 4$	0.00260415909870426	1.511310863122e-07	-1.4405360953e-06	6.86982791942e-06

The numerical approximation is quite close to the actual value as the 4th iterative error is in the order of 10^{-6}. Therefore the result is more accurate and acceptable.

When applying 10 iterations the error reaches 10^{-15}

Question:

i. **Compare the convergence of trapezoidal rule and the Simpsons rule using the given codes for the same integral as in 1.**

ii. **Use the code comp_error.m to plot the logarithm of the error versus the logarithm of h. Use this code to plot the errors for N = 10; 20; 40; 60 80**
Using the same interval and function as in problem 1. How does the slope of your log-log plot compare to the error term given for composite Simpson's rule and composite trapezoid rule?

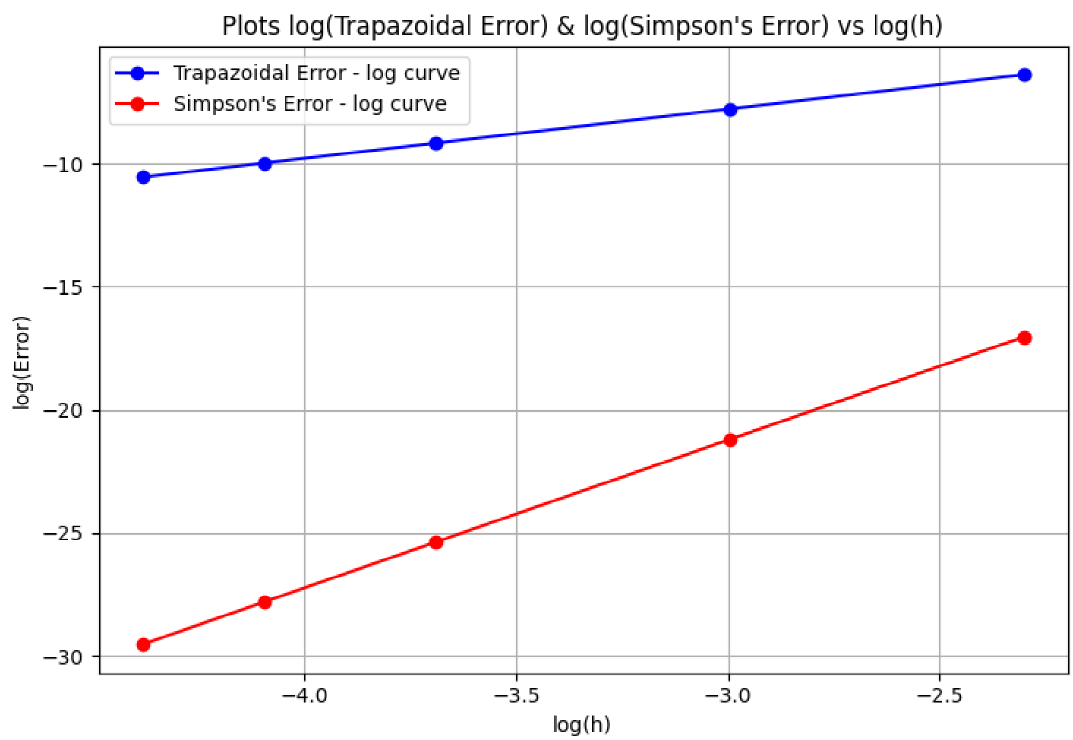
$i)$

Comparison of the convergence of trapezoidal and Simpson's rules

n	Approximation		Error	
	Trapezoidal	Simpson's	Trapezoidal	Simpson's
10	3.1399259889071587	3.141592613939215	0.0016666646826344333	3.9650577932093256e-08
20	3.1411759869541287	3.141592652969785	0.00041666663566441997	6.200080449048073e-10
40	3.1414884869236115	3.141592653580106	0.00010416666618162651	9.687362023669266e-12
60	3.1415463572935387	3.141592653588943	4.629629625441112e-05	8.504308368628699e-13
80	3.141566611923134	3.141592653589642	2.6041666659093465e-05	1.509903313490213e-13

From the results from the above table it is obvious that Simpson's rule converges exponentially faster than Trapezoidal rule.
The errors for each n value by both rules have a clear cut difference.

$ii)$



Comment:
The slope of the log(Simpson's curve) is higher than the slope of the log(Trapezoidal Curve) and both the curves show linear gradient.
Theoretically the order of error of Simpson's $\frac{1}{3}$ rule is higher than the order of error of Trapezoidal rule. The plot well illustrates that.

Question:

i. **Observe the convergence of Gaussian Quadrature rule using the given codes for the same integral as in 1.**
ii. **Use the code given to plot the logarithm of the error versus the logarithm of h. Use this code to plot the errors for N = 10; 20; 40; 60 80**
Using the same interval and function as in problem 1. How does this quadrature compare to composite Simpson's method from problem1?

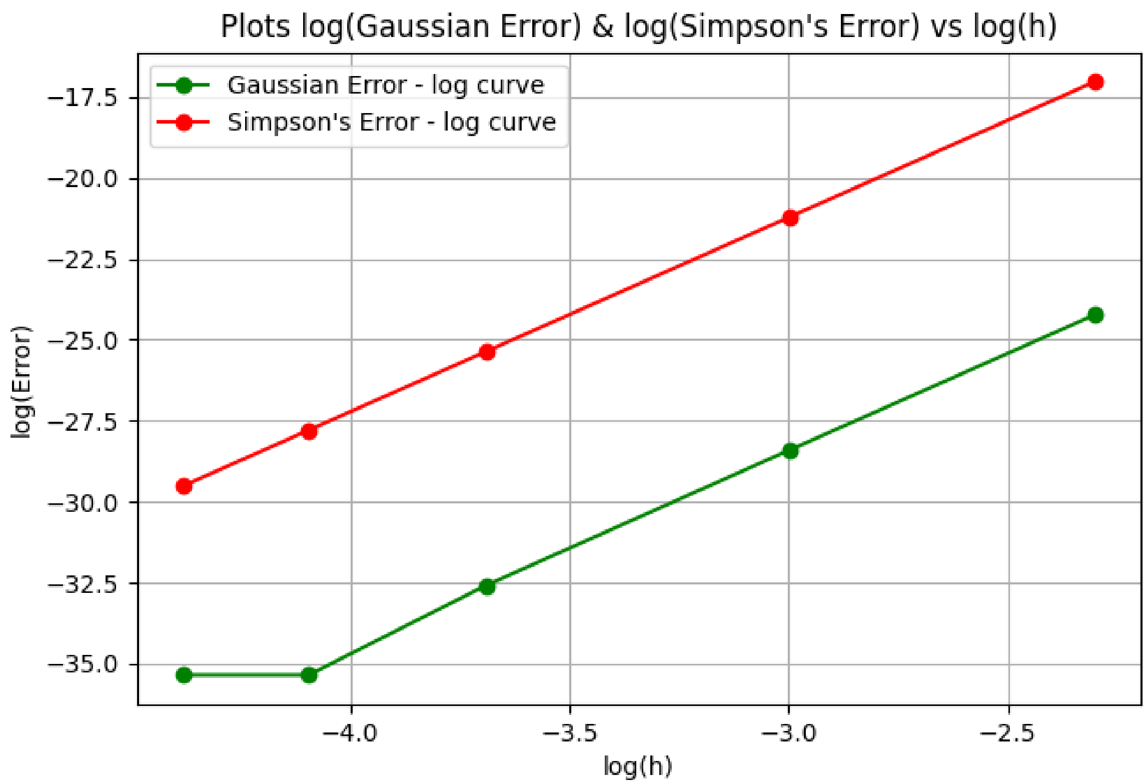
$i)$

Observation of the convergence of Gaussian Quadrature rule

n	Gaussian Approximation	Error
10	3.141592653560034	2.9759306130472396e-11
20	3.1415926535893273	4.658495811327157e-13
40	3.141592653589786	7.105427357601002e-15
60	3.1415926535897927	4.440892098500626e-16
80	3.1415926535897927	4.440892098500626e-16

Comment:
The Gaussian Quadrature approximates the result better than Trapezoidal and Simpson's Rules.
For higher n values the error become too small. The convergence of Gaussian quadrature rule is similar to the Simpson's Rule.

$ii)$



Both Gaussian Quadrature and Simpson's rule have a similar convergence but GQ rule produces lesser error than Simpson's rule

Find the code used to plot and compare these error and approximations in the Appendix section at the end.

APPENDIX A : Python program used for the lab

Implementations function, Trapezoidal rule, Simpson's rule, Composite Gaussian Quadrature and Romberg table.
Used numpy library and matplotlib library for numerical calculations and graphical representation.

```
import math
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return (4/(x**2 + 1))

def trapazoid(n, a, b):
    h = (b - a) / n
    result = f(a) + f(b)
    for i in range(1, n):
        result += 2 * f(a + (i*h))
    result = result * h / 2
    return result

def simpson(n, a, b):
    h = (b - a) / n
    result = f(a) + f(b)
    r1 = 0
    r2 = 0
    for i in range(1, n):
        if(i%2 == 0):
            r2 += f(a + (i*h))
        else:
            r1 += f(a + (i*h))

    result = (result + (4 * r1 + 2 * r2)) * (h / 3)
    return result
```

```
def gaussianQuadrature(a, b):
    nodes = [-math.sqrt(3/5), 0, math.sqrt(3/5)]
    weights = [(5/9), (8/9), (5/9)]

    result = 0
    for i in range(3):
        result += weights[i] * f(nodes[i]*(b-a)/2 + (a+b)/2)
    result *= (b-a)/2

    return result

def comp_gaussQuadrature(n, a, b):
    intervals = list(np.linspace(a,b,n+1))
    result = 0
    for i in range(n):
        result += gaussianQuadrature(intervals[i], intervals[i+1])
    return result

def romberg(a, b, t, type):
    Rs = []
    Es = []
    r = []
    e = []
    exact = math.pi
    for i in range(t):
        if type == "trapazoid":
            p = trapazoid((2**i), a, b)
        elif type == "simpson":
            p = simpson((2**i), a, b)
        r.append(p)
        e.append(exact-p)
    Rs.append(r)
    Es.append(e)
    for k in range(t-1):
        r = []
        e = []
        for j in range(1, t-k):
            v = Rs[-1][j] + ((Rs[-1][j] - Rs[-1][j-1])/(4**(k+1) - 1))
            r.append(v)
            e.append(exact-v)
        Rs.append(r)
        Es.append(e)
    return [Rs,Es]
```

Functions used to plot the curves

```
n_lst = [10,20,40,60,80]

def comp_error_TrapAndSimpson(n_lst, a, b):
    trapError_arr = []
    simpError_arr = []
    h_arr = []
    for n in n_lst:
        trapAprx = trapazoid(n,a,b)
        simpAprx = simpson(n,a,b)
        exact = math.pi
        trapError_arr.append(math.log(exact-trapAprx))
        simpError_arr.append(math.log(exact-simpAprx))
        _h = (b - a) / n
        h_arr.append(math.log(_h))

    y1 = np.array(trapError_arr)
    y2 = np.array(simpError_arr)
    h = np.array(h_arr)

    plt.plot(h,y1, 'bo-')
    plt.plot(h,y2,'ro-')
    plt.title("Plots log(Trapazoidal Error) & log(Simpson's Error) vs log(h)")
    plt.xlabel("log(h)")
    plt.ylabel("log(Error)")
    plt.legend(["Trapazoidal Error - log curve", "Simpson's Error -
 log curve"])
    plt.grid()
    plt.show()

comp_error_TrapAndSimpson(n_lst, 0, 1) #function call
```

```python
def plot_GaussError_vs_h(n_lst, a, b):
    y = []
    h_arr = []
    for n in n_lst:
        gaussAprx = comp_gaussQuadrature(n,a,b)
        exact = math.pi
        _h = (b - a) / n
        h_arr.append(math.log(_h))
        y.append(math.log(exact-gaussAprx))

    plt.plot(np.array(h_arr),np.array(y),'go-')
    plt.title("Plots log(Gaussian Quadrature Error) vs log(h)")
    plt.xlabel("log(h)")
    plt.ylabel("log(Error)")
    plt.grid()
    plt.show()

plot_GaussError_vs_h(n_lst, 0, 1)   #function call
```

```python
def comp_error_GaussAndSimpson(n_lst, a, b):
    gaussError_arr = []
    simpError_arr = []
    h_arr = []
    for n in n_lst:
        gaussAprx = comp_gaussQuadrature(n,a,b)
        simpAprx = simpson(n,a,b)
        exact = math.pi
        gaussError_arr.append(math.log(exact-gaussAprx))
        simpError_arr.append(math.log(exact-simpAprx))
        _h = (b - a) / n
        h_arr.append(math.log(_h))

    y1 = np.array(gaussError_arr)
    y2 = np.array(simpError_arr)
    h = np.array(h_arr)

    plt.plot(h, y1, 'go-')
    plt.plot(h, y2, 'ro-')
    plt.title("Plots log(Gaussian Error) & log(Simpson's Error) vs log(h)")
    plt.xlabel("log(h)")
    plt.ylabel("log(Error)")
    plt.legend(["Gaussian Error - log curve", "Simpson's Error - log curve"])
    plt.grid()
    plt.show()

comp_error_GaussAndSimpson(n_lst, 0, 1)     #function call
```