Name:	Aneesh Panchal	**Homework No:**	Homework 5
SR No:	06-18-01-10-12-24-1-25223	**Course Code:**	DS288
Email ID:	aneeshp@iisc.ac.in	**Course Name:**	Numerical Methods
Date:	November 18, 2024	**Term:**	AUG 2024

Solution 1

We are given exact function as $f(x) = x^n$ along with equation,

$$\int_{x_0}^{x_2} f(x)dx = a_0 f(x_0) + a_1 f(x_1) + a_2 f(x_2) + k f^{(4)}(\xi) \tag{1}$$

Now, for different values of $n = 0, 1, 2, 3$ we get,

1. $n = 0$, we have $f(x) = 1$ and hence,
$$\int_{x_0}^{x_2} dx = x\Big|_{x_0}^{x_2} = x_2 - x_0 = a_0 + a_1 + a_2 + \cancelto{0}{k(0)} \tag{2}$$

2. $n = 1$, we have $f(x) = x$ and hence,
$$\int_{x_0}^{x_2} xdx = \frac{x^2}{2}\Big|_{x_0}^{x_2} = \frac{x_2^2}{2} - \frac{x_0^2}{2} = a_0x_0 + a_1x_1 + a_2x_2 + \cancelto{0}{k(0)} \tag{3}$$

3. $n = 2$, we have $f(x) = x^2$ and hence,
$$\int_{x_0}^{x_2} x^2dx = \frac{x^3}{3}\Big|_{x_0}^{x_2} = \frac{x_2^3}{3} - \frac{x_0^3}{3} = a_0x_0^2 + a_1x_1^2 + a_2x_2^2 + \cancelto{0}{k(0)} \tag{4}$$

4. $n = 3$, we have $f(x) = x^3$ and hence,
$$\int_{x_0}^{x_2} x^3dx = \frac{x^4}{4}\Big|_{x_0}^{x_2} = \frac{x_2^4}{4} - \frac{x_0^4}{4} = a_0x_0^3 + a_1x_1^3 + a_2x_2^3 + \cancelto{0}{k(0)} \tag{5}$$

As we only require 3 variables and hence, we can use only use equations (2), (3) and (4). Substituting values of $x_2 = x_0 + 2h$ in these equations we get,

$$2h = a_0 + a_1 + a_2$$
$$2h(x_0 + h) = x_0(a_0 + a_1 + a_2) + h(a_1 + 2a_2)$$
$$\frac{2h}{3}(3x_0^2 + 4h^2 + 6x_0h) = x_0^2(a_0 + a_1 + a_2) + h^2(a_1 + 4a_2) + x_0h(2a_1 + 4a_2)$$

Comparing coefficients of x_0 and x_0^2 we get,

$$a_1 + 2a_2 = 2h$$
$$a_1 + 4a_2 = \frac{8h}{3}$$

Solving these equations we get,

$$\boxed{\mathbf{a_1 = \frac{4h}{3}} \quad \text{and, } \mathbf{a_2 = \frac{h}{3}}}$$

Substituting these values in equation (2) we get,

$$\boxed{\mathbf{a_0 = \frac{h}{3}}}$$

These values of a_0, a_1 and a_2 also satisfy equation (5) and hence, are correct required values.

For $n = 4$, we have $f(x) = x^4$ and $f^{(4)}(x) = 4!$, and hence,

$$\int_{x_0}^{x_2} x^4dx = \frac{x^5}{5}\Big|_{x_0}^{x_2} = \frac{x_2^5}{5} - \frac{x_0^5}{5} = a_0x_0^4 + a_1x_1^4 + a_2x_2^4 + k(4!) \tag{6}$$

Substituting values of $x_2 = x_0 + 2h$ and values of a_0, a_1 and a_2,

Neglecting Terms with $\mathbf{x_0, x_0^2, x_0^3}$ and $\mathbf{x_0^4}$
because we are interested in finding value of k which doesn't involve any x_0 term we get,

$$\frac{(2h)^5}{5} = \frac{4h}{3}h^5 + \frac{h}{3}(2h)^4 + 24k$$

$$\frac{96h^5 - 20h^5 - 80h^5}{15} = 24k$$

Solving which we get,

$$\boxed{\mathbf{k = -\frac{h^5}{90}}}$$

Solution 2

Given assumptions for solving problems are as follows,
Initial Point: $a = 0$
Final Point: $b = 1$
Tolerence: $|R_{n,n} - R_{n-1,n-1}| \leq 10^{-5}$

Question (a)

$$\int_0^1 x^{\frac{1}{3}} dx$$

Question (b)

$$\int_0^1 x^2 e^{-x} dx$$

Romberg Integration Formulas

$$R_{1,1} = \frac{h}{2}\left(f(a) + f(b)\right) \qquad \text{where, } h_0 = b - a \tag{7}$$

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f\left(a + (2i-1)h_k\right)\right] \qquad \text{where, } h_k = \frac{h_{k-1}}{2} \tag{8}$$

$$R_{k,j} = R_{k,j-1} + \frac{1}{4^{j-1} - 1}\left(R_{k,j-1} - R_{k-1,j-1}\right) \tag{9}$$

Romberg Integration Analysis

Number of function evaluations for,
Eq. (7) is $\mathbf{2}$.
Eq. (8) is $\mathbf{2^{k-2}}$.
Eq. (9) is $\mathbf{0}$

Let the algorithm terminates in n iterations, then total number of function evaluations are,

$$func\_evals = 2 + \left(1 + 2 + 4 + \cdots + 2^k\right) = 1 + 2^k \qquad \text{where, } n = k + 1 \tag{10}$$

Trapezoidal Rule Formula

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\left(f(a+h) + f(a+2h) + \cdots + f(a+(j-2)h)\right) + f(a+(j-1)h)\right] \tag{11}$$

where, $b = a + (j-1)h$, $j = 2^{n-1}$ and $h = (b-a)2^{1-n}$

Number of function evaluations for same n is **same** for Trapezoidal Rule and Romberg Integration formula.

NOTE: We can directly extract the value of Trapezoidal using Romberg Integration table as for particular value of n, $R_{n,1}$ is the value we can obtain from Trapezoidal Rule.
$R_{12,1} = 0.749989342854916$ Trapezoidal $= 0.7499893428549149$
$R_{4,1} = 0.16107989607963955$ Trapezoidal $= 0.16107989607963955$

Method	$f(x)$	Integral Value	n	Func. Eval.	True Value	Abs. Error
Romberg Integration	$x^{\frac{1}{3}}$	0.7499954223896331	12	2049	0.75	$4.57761e-06$
	x^2e^{-x}	0.16060280012980105	4	9	$2-5e^{-1}$	$5.98701e-09$
Trapezoidal Rule	$x^{\frac{1}{3}}$	0.7499893428549149	12	2049	0.75	$1.06571e-05$
	x^2e^{-x}	0.16107989607963955	4	9	$2-5e^{-1}$	$4.77101e-04$

Solution 3

Given assumptions for solving problems are as follows,
Initial Point: $a = 0$
Final Point: $b = 1$
Values of n: $n = 2, 3, 4$ and 5

Gaussian Quadrature Formulas

$$I = \int_a^b f(x)dx = \sum_{i=1}^{n} a_i f(x_i) \qquad \text{where, } x_i \text{ are Gauss Points and } a_i \text{ are weights} \tag{12}$$

We will be using **Legendre Gaussian Quadrature**,

$$\int_{-1}^{1} f\left(\left(\frac{a+b}{2}\right)+\left(\frac{b-a}{2}\right)x\right)\left(\frac{b-a}{2}\right)dx = \left(\frac{b-a}{2}\right)\sum_{i=1}^{n} a_i f(y_i) \qquad \text{where, } y_i = \left(\frac{a+b}{2}\right)+\left(\frac{b-a}{2}\right)x_i \tag{13}$$

$$\text{Gauss Points } x_i \text{ can be found using roots of Legendre Polynomial, } P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left(x^2-1\right)^n \tag{14}$$

$$\text{Weights } a_i \text{ corresponding to Gauss Point } x_i \text{ can be found using, } a_i = \frac{2}{\left(1-x_i^2\right)\left[P_n'\left(x_i\right)\right]^2} \tag{15}$$

Hence, it is clear that **Number of Function Evaluations is $\mathbf{n}$**.

NOTE: Weights and Gauss Points thus found are given in the Appendix along with code.

n	$f(x)$	Integral Value	Func. Eval.	True Value	Abs. Error	Abs. Error [Romberg]
2	$x^{\frac{1}{3}}$	0.759778022294319	2	0.75	$9.77802e-03$	$4.57761e-06$
	x^2e^{-x}	0.159410430966379	2	$2-5e^{-1}$	$1.19236e-03$	$5.98701e-09$
3	$x^{\frac{1}{3}}$	0.753855469939559	3	0.75	$3.85546e-03$	$4.57761e-06$
	x^2e^{-x}	0.160595386808919	3	$2-5e^{-1}$	$7.40733e-06$	$5.98701e-09$
4	$x^{\frac{1}{3}}$	0.751946476655680	4	0.75	$1.94647e-03$	$4.57761e-06$
	x^2e^{-x}	0.160602777514685	4	$2-5e^{-1}$	$1.66281e-08$	$5.98701e-09$
5	$x^{\frac{1}{3}}$	0.751132312655820	5	0.75	$1.13231e-03$	$4.57761e-06$
	x^2e^{-x}	0.160602794123438	5	$2-5e^{-1}$	$1.93507e-11$	$5.98701e-09$

Results Comparison:

1. For problem $\mathbf{x^{\frac{1}{3}}}$, absolute error using **Gaussian Quadrature** turns out to be very high of order $\mathbf{O(10^{-3})}$. However, for **Romberg Integration** it turns out to be of order $\mathbf{O(10^{-6})}$. This is due to **2049** function evaluations in Romberg Integration while only $\mathbf{2, 3, 4, 5}$ function evaluations in Gaussian Quadrature. So **Gaussian Quadrature** is good measure when we are **restricted by memory**.
2. For problem $\mathbf{x^2e^{-x}}$, absolute error using Gaussian Quadrature outperforms the Romberg Integration method comparative function evaluations. This is due to fact that Gauss Points are specially located which can integrate $\mathbf{2n+1}$ degree polynomial when $\mathbf{n}$ Gauss Points are considered. Due to this reason, **Gaussian Quadrature performs better than any equi-spaced mesh or, grid (eg. Romberg Integration) when comparative points are considered**.

1 Appendix: Assignment 5 Programming

1.1 Ques 2

[1]:

```
import math

def print_row(i, R):
  print(f"R[{i+1:2d}] = ", end="")
  for j in range(i + 1):
    print(f"{R[j]:f} ", end="")
  print()

def romberg(f, a, b, max_steps, tol):
    R1, R2 = [0]*max_steps, [0]*max_steps
    Rprev, Rcurr = R1, R2
    h = b - a
    Rprev[0] = 0.5*h*(f(a) + f(b))
    fun_count = 2
    print_row(0, Rprev)

    for i in range(1, max_steps):
        h = h/2
        c = 0
        exval = 2**(i-1)
        for j in range(1, exval + 1):
            c = c + f(a + (2*j - 1)*h)
            fun_count = fun_count + 1
        Rcurr[0] = h*c + 0.5*Rprev[0]

        for j in range(1, i + 1):
            nval = 4**j
            Rcurr[j] = (nval * Rcurr[j - 1] - Rprev[j - 1]) / (nval - 1)

        print_row(i, Rcurr)
        if i > 1 and abs(Rprev[i - 1] - Rcurr[i]) < tol:
            print("n: ", i + 1)
            return Rcurr[i], fun_count
        Rprev, Rcurr = Rcurr, Rprev

    print("n: ", i+1)
    return Rprev[max_steps - 1], fun_count

def f1(r):
    return r**(1/3)

def f2(r):
    return (r**2)*(math.exp(-r))

r1 = romberg(f1, 0, 1, 10000, 1e-5)
print(r1)
print("\n")
r2 = romberg(f2, 0, 1, 10000, 1e-5)
print(r2)
```

```
R[ 1] = 0.500000
R[ 2] = 0.646850 0.695800
R[ 3] = 0.708055 0.728457 0.730634
R[ 4] = 0.733100 0.741448 0.742314 0.742500
R[ 5] = 0.743230 0.746606 0.746950 0.747023 0.747041
R[ 6] = 0.747297 0.748653 0.748790 0.748819 0.748826 0.748828
R[ 7] = 0.748923 0.749465 0.749520 0.749531 0.749534 0.749535 0.749535
R[ 8] = 0.749572 0.749788 0.749809 0.749814 0.749815 0.749815 0.749815 0.749815
R[ 9] = 0.749830 0.749916 0.749924 0.749926 0.749927 0.749927 0.749927 0.749927
0.749927
R[10] = 0.749932 0.749967 0.749970 0.749971 0.749971 0.749971 0.749971 0.749971
0.749971 0.749971
R[11] = 0.749973 0.749987 0.749988 0.749988 0.749988 0.749988 0.749988 0.749988
0.749988 0.749988 0.749988
R[12] = 0.749989 0.749995 0.749995 0.749995 0.749995 0.749995 0.749995 0.749995
0.749995 0.749995 0.749995 0.749995
n:  12
(0.7499954223896331, 2049)


R[ 1] = 0.183940
R[ 2] = 0.167786 0.162402
R[ 3] = 0.162488 0.160722 0.160611
R[ 4] = 0.161080 0.160610 0.160603 0.160603
n:  4
(0.16060280012980105, 9)
```

[2]:

```
def trapezoidal(f, a, b, n):
    h = (b-a)/n
    s = f(a) + f(b)
    i = 1
    fun_count = 2
    while i < n:
        s = s + 2*f(a + i*h)
        fun_count = fun_count + 1
        i = i + 1
    return ((h/2)*s), fun_count

t1 = trapezoidal(f1, 0, 1, 2**11)
print(t1)
print("\n")
t2 = trapezoidal(f2, 0, 1, 2**3)
print(t2)
```

```
(0.7499893428549149, 2049)


(0.16107989607963955, 9)
```

1.2 Ques 3

[3]:
```python
import sympy as sym
x = sym.symbols('x')

def legendre(x, n):
    val = sym.diff((x**2 - 1)**n, x, n)
    return (1/((2**n)*(math.factorial(n))))*val

def weights_roots(n):
    w = []
    func = legendre(x, n)
    funcdiff = sym.diff(func, x)
    r = sym.solve(func, x, dict=False)
    for i in range(len(r)):
        val = 2/((1 - r[i]**2)*((funcdiff.subs(x, r[i]))**2))
        w.append(val)
    return r, w

def quadrature(f, n, a, b):
    fvalues = []
    for i in range(len(n)):
        funcval = 0
        gausspt, weight = weights_roots(n[i])
        for j in range(len(gausspt)):
            normpt = ((b+a)/2) + ((b-a)/2)*gausspt[j]
            funcval = funcval + ((b-a)/2)*weight[j]*f(normpt)
        print(f"n = {n[i]}: ", funcval)
        fvalues.append(funcval)
        print("Gauss Points: ", gausspt)
        print("Weights: ",weight)
        print("\n")
    return fvalues

n = [2,3,4,5]
a = 0
b = 1
q1 = quadrature(f1, n, a, b)
print("\n")
q2 = quadrature(f2, n, a, b)
```

```
n = 2:  0.759778022294319
Gauss Points:  [-0.577350269189626, 0.577350269189626]
Weights:  [1.00000000000000, 1.00000000000000]


n = 3:  0.753855469939559
Gauss Points:  [-0.774596669241483, 0.0, 0.774596669241483]
Weights:  [0.555555555555555, 0.888888888888889, 0.555555555555555]


n = 4:  0.751946476655680
Gauss Points:  [-0.861136311594053, -0.339981043584856, 0.339981043584856,
0.861136311594053]
```

```
Weights:  [0.347854845137454, 0.652145154862546, 0.652145154862546,
0.347854845137454]


n = 5:  0.751132312655820
Gauss Points:  [-0.906179845938664, -0.538469310105683, 0.0, 0.538469310105683,
0.906179845938664]
Weights:  [0.236926885056189, 0.478628670499367, 0.568888888888889,
0.478628670499367, 0.236926885056189]
```

```
n = 2:  0.159410430966379
Gauss Points:  [-0.577350269189626, 0.577350269189626]
Weights:  [1.00000000000000, 1.00000000000000]


n = 3:  0.160595386808919
Gauss Points:  [-0.774596669241483, 0.0, 0.774596669241483]
Weights:  [0.555555555555555, 0.888888888888889, 0.555555555555555]


n = 4:  0.160602777514685
Gauss Points:  [-0.861136311594053, -0.339981043584856, 0.339981043584856,
0.861136311594053]
Weights:  [0.347854845137454, 0.652145154862546, 0.652145154862546,
0.347854845137454]


n = 5:  0.160602794123438
Gauss Points:  [-0.906179845938664, -0.538469310105683, 0.0, 0.538469310105683,
0.906179845938664]
Weights:  [0.236926885056189, 0.478628670499367, 0.568888888888889,
0.478628670499367, 0.236926885056189]
```