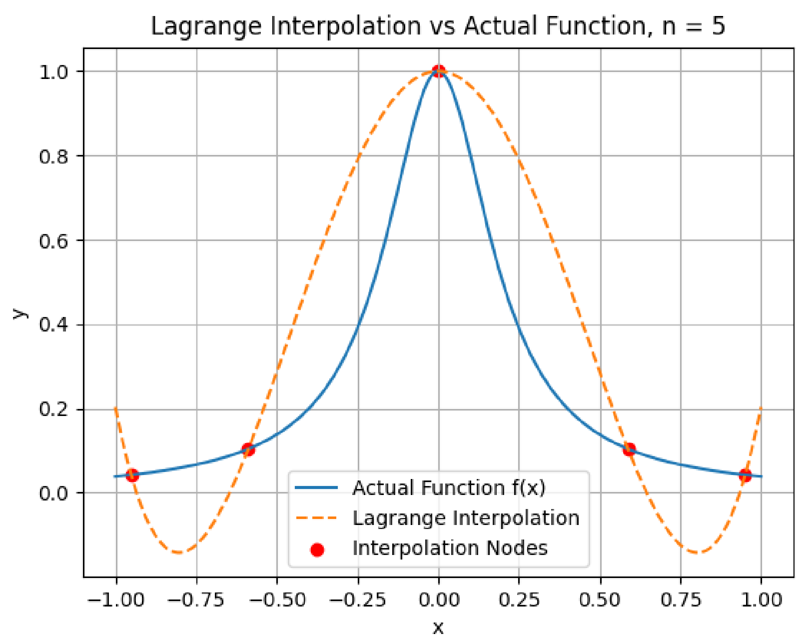
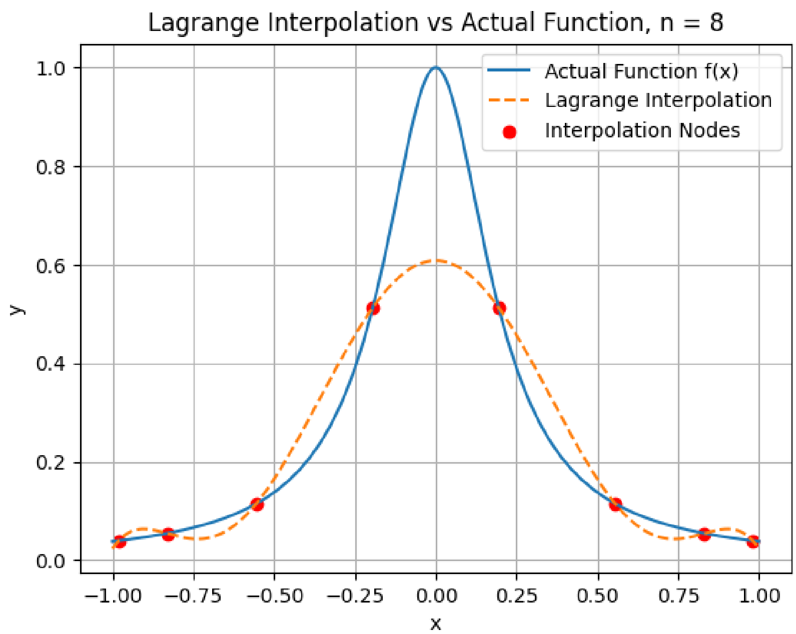
MATH 417 502
Homework 4

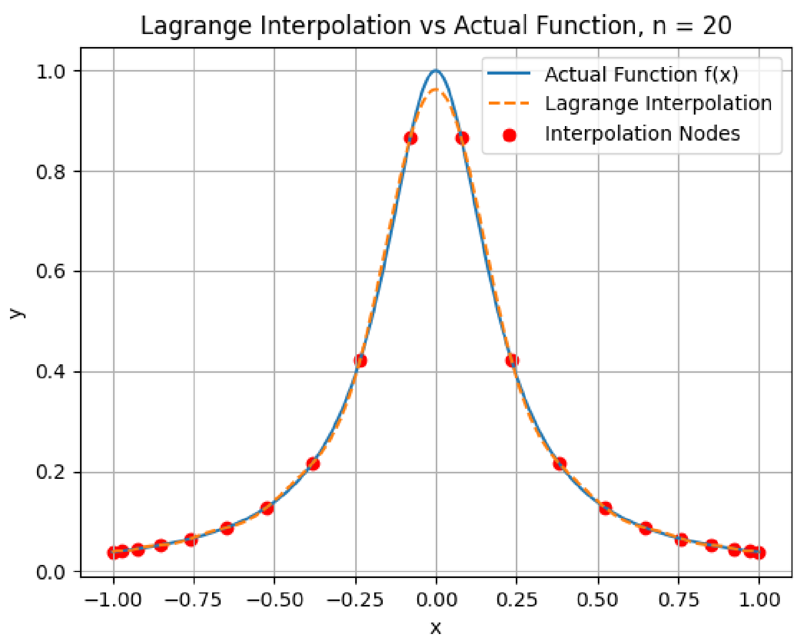
Keegan Smith

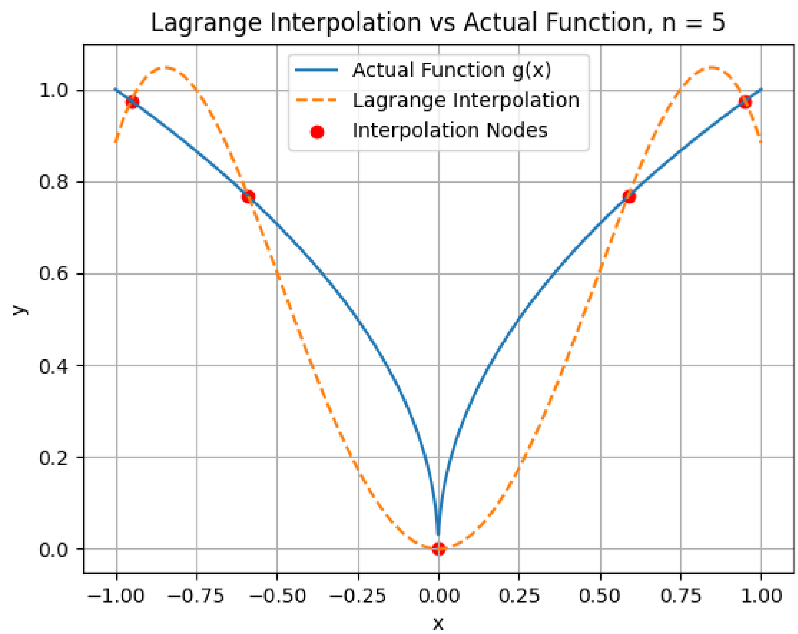
September 30, 2024

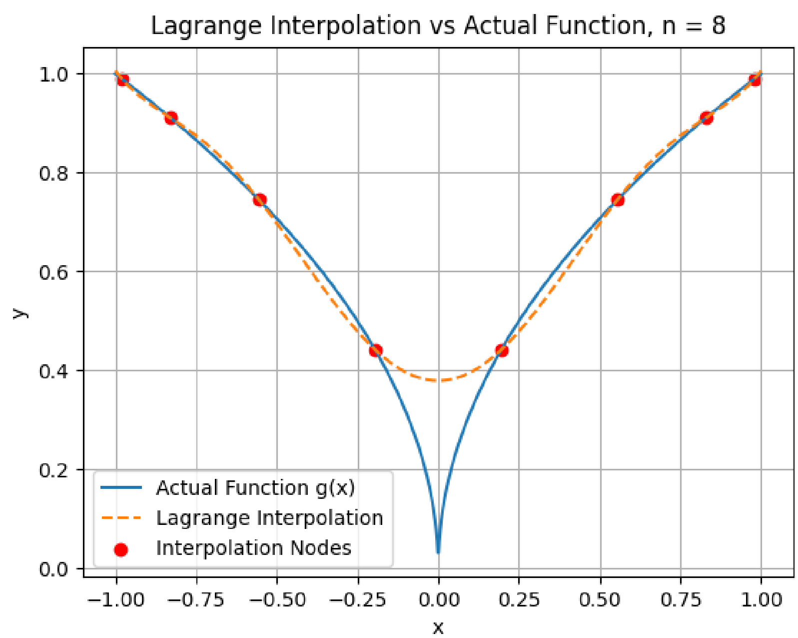
Problem 1

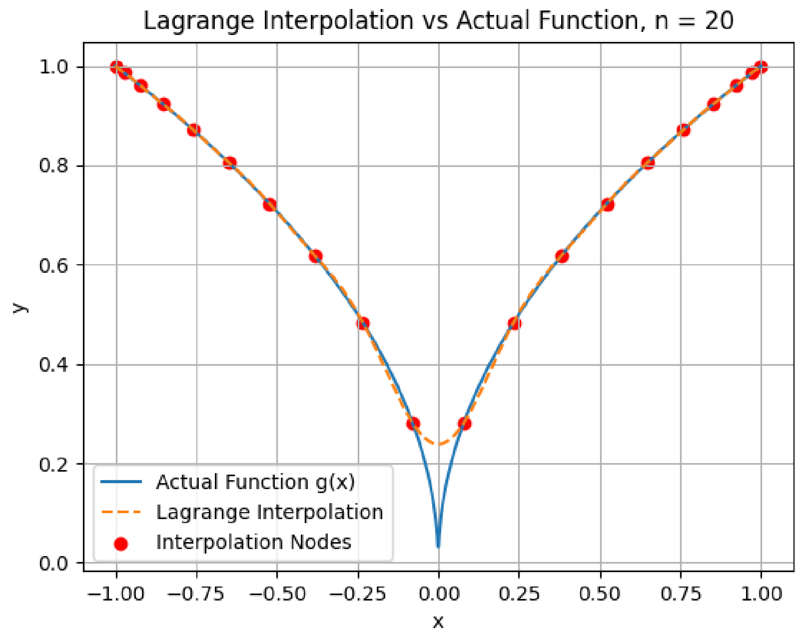



Lagrange Interpolation vs Actual Function, n = 8
Actual Function f(x)
Lagrange Interpolation
Interpolation Nodes
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Lagrange Interpolation vs Actual Function, n = 20
Actual Function f(x)
Lagrange Interpolation
Interpolation Nodes
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Lagrange Interpolation vs Actual Function, n = 5
Actual Function g(x)
Lagrange Interpolation
Interpolation Nodes
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Lagrange Interpolation vs Actual Function, n = 8
Actual Function g(x)
Lagrange Interpolation
Interpolation Nodes
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



code:

```
import numpy as np
import matplotlib.pyplot as plt
import math
def lagrange(x, points):
    lagrange_results = [];
    for i in range(0, len(points)):
        numerator = 1;
        denominator = 1;
        for j in range(0, len(points)):
            if(i == j):
                continue;
            numerator *= (x - points[j][0]);
            denominator *= (points[i][0] - points[j][0]);
        lagrange_results.append(numerator / denominator);
    result = 0;
    for i in range(0, len(points)):
        result += points[i][1] * lagrange_results[i]
    return result;

def f(x):
    return 1 / (1 + 25 * x**2);
def g(x):
```

```
    return (abs(x)) ** (1/2);

def compute_cheby_points(n):
    x_coords = [];
    for i in range(0, n):
        x_coords.append(math.cos((2* i + 1) * math.pi / (2*(n - 1) + 2)))
    return x_coords;
def get_x_coords(interval, num_points):
    start = interval[0];
    orig_start = start
    end = interval[1];
    result = compute_cheby_points(num_points)

    return result;
def do_the_thing(my_function, n):
    x_coords = get_x_coords([-1, 1], n);
    actual_function_values = []
    for x in x_coords:
        actual_function_values.append([x, my_function(x)]);

    x_plot = np.linspace(-1, 1, 1000)
    y_actual = []
    y_interp = []
    for i in range(0, len(x_plot)):
        y_actual.append(my_function(x_plot[i]));
        y_interp.append(lagrange(x_plot[i], actual_function_values));

    plt.plot(x_plot, y_actual, label=f'Actual Function {my_function.__name__}(x)')
    plt.plot(x_plot, y_interp, '--', label='Lagrange Interpolation')
    plt.scatter(x_coords, [my_function(x) for x in x_coords], color='red', label='Interpolation Nodes')
    plt.title(f'Lagrange Interpolation vs Actual Function, n = {n}')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.grid(True)
    plt.show()
def main():
    functions = [f, g]
    nums = [5, 8, 20]
    for function in functions:
        for num in nums:
            do_the_thing(function, num)
if __name__ == "__main__":
    main();
```

Problem 2

From the lecture notes, Simpson's $\frac{3}{8}$'s rule is a result of computing Newton Cotes for $n = 3$. Newton Cotes rule gives us a quadrature rule where:

$$w_i = \int_a^b L_i(x)dx$$

for n equidistant points from a to b. We will define $h = \frac{b-a}{n}$, such that $a + h = x_1, a + 2h = x_2$, etc. So for $n = 3$ we have:

$$
\begin{aligned}
w_0 &= \int_a^b \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}dx \\
&= \frac{1}{(-h)(-2h)(-3h)} \int_a^b (x^3 - x^2\cdot x_3 - x^2\cdot x_2 + x\cdot x_2\cdot x_3 - x^2\cdot x_1 + x\cdot x_1\cdot x_3 + x_1\cdot x_2\cdot x - x_1\cdot x_2\cdot x_3)dx \\
&= \frac{1}{(-h)(-2h)(-3h)} \int_a^b (x^3 - x^2(x_3+x_2+x_1) + x(x_2\cdot x_3 + x_1\cdot x_3 + x_1\cdot x_2) - x_1\cdot x_2\cdot x_3)dx \\
&= \frac{1}{(-h)(-2h)(-3h)}(\frac{1}{4}(b^4-a^4) - (x_3+x_2+x_1)(\frac{1}{3}(b^3-a^3)) + (x_2\cdot x_3 + x_1\cdot x_3 + x_1\cdot x_2)(\frac{1}{2})(b^2-a^2) \\
&\quad - (b-a)(x_1\cdot x_2\cdot x_3)) \\
&= \frac{1}{-6h^3}(\frac{1}{4}((a+3h)^4 - a^4) - (3a+6h)(\frac{1}{3}((a+3h)^3 - a^3)) - (b-a)(x_1\cdot x_2\cdot x_3) \\
&\quad + ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)) \\
&= \frac{1}{-6h^3}(\frac{1}{4}(a^4 + 12a^3h + 54a^2h^2 + 108ah^3 + 81h^4 - a^4) - (3a+6h)(\frac{1}{3}((a+3h)^3 - a^3)) \\
&\quad - (b-a)(x_1\cdot x_2\cdot x_3) \\
&\quad + ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)) \\
&= \frac{1}{-6h^3}(\frac{1}{4}(12a^3h + 54a^2h^2 + 108ah^3 + 81h^4) - (3a+6h)(\frac{1}{3}(9ha^2 + 27ah^2 + 27h^3)) \\
&\quad - (b-a)(x_1\cdot x_2\cdot x_3) \\
&\quad + ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)) \\
&= \frac{1}{-6h^3}(\frac{1}{4}(12a^3h + 54a^2h^2 + 108ah^3 + 81h^4) - (54h^4 + 9ha^3 + 27a^2h^2 + 81ah^3 + 18h^2a^2) \\
&\quad - (b-a)(x_1\cdot x_2\cdot x_3) \\
&\quad + ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)) \\
&= \frac{1}{-6h^3}((3a^3h + \frac{27}{2}a^2h^2 + 27ah^3 + \frac{81}{4}h^4) - (54h^4 + 9ha^3 + 27a^2h^2 + 81ah^3 + 18h^2a^2) \\
&\quad - (b-a)(x_1\cdot x_2\cdot x_3) \\
&\quad + ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2))
\end{aligned}
$$

$$= \frac{1}{-6h^3}(-\frac{135}{4}h^4 - 6ha^3 - \frac{27}{2}a^2h^2 - 54ah^3 - 18h^2a^2 - (b-a)(x_1 \cdot x_2 \cdot x_3)$$
$$+ ((a+2h)\cdot(a+3h) + (a+h)\cdot(a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)))$$
$$= \frac{1}{-6h^3}(-\frac{135}{4}h^4 - 6ha^3 - \frac{27}{2}a^2h^2 - 54ah^3 - 18h^2a^2 - (b-a)(x_1 \cdot x_2 \cdot x_3)$$
$$+ ((a+3h)\cdot(a+2h+a+h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)))$$
$$= \frac{1}{-6h^3}(-\frac{135}{4}h^4 - 6ha^3 - \frac{27}{2}a^2h^2 - 54ah^3 - 18h^2a^2 - (b-a)(x_1 \cdot x_2 \cdot x_3)$$
$$+ ((a+3h)\cdot(2a+3h) + (a+h)\cdot(a+2h))\cdot\frac{1}{2}((a+3h)^2 - a^2)))$$
$$= \frac{1}{-6h^3}(-\frac{135}{4}h^4 - 6ha^3 - \frac{27}{2}a^2h^2 - 54ah^3 - 18h^2a^2 - (b-a)(x_1 \cdot x_2 \cdot x_3)$$
$$+ \frac{27h^2a^2 + 99h^4}{2} + 87ah^3 + 9ha^3 + 36a^2h^2)$$
$$= \frac{1}{-6h^3}(-\frac{135}{4}h^4 - 6ha^3 - \frac{27}{2}a^2h^2 - 54ah^3 - 18h^2a^2 + \frac{27h^2a^2 + 99h^4}{2} + 87ah^3 + 9ha^3 + 36a^2h^2$$
$$- (b-a)(x_1 \cdot x_2 \cdot x_3))$$
$$= \frac{1}{-6h^3}(-\frac{135h^4}{4} + 3ha^3 + \frac{99h^4}{2} + 33ah^3 - (b-a)(x_1 \cdot x_2 \cdot x_3))$$
$$= \frac{1}{-6h^3}(\frac{63}{4}h^4 + 3ha^3 + 33ah^3 - (3h)((a+h)\cdot(a+2h)\cdot(a+3h)))$$
$$= \frac{1}{-6h^3}(\frac{63}{4}h^4 + 3ha^3 + 33ah^3 - (3h)(a^3 + 6ha^2 + 11ah^2 + 6h^3))$$
$$= \frac{1}{-6h^2}(\frac{63}{4}h^3 + 3a^3 + 33ah^2 - (3a^3 + 18ha^2 + 33ah^2 + 18h^4))$$
$$= \frac{1}{-6h^2}(\frac{63}{4}h^3 - 18ha^2 - 18h^4)$$
$$= \frac{1}{-6h}(\frac{63}{4}h^2 - 18a^2 - 18h^3)$$

:(

$$
\begin{aligned}
w_1 &= \int_a^b \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}dx \\
&= \frac{1}{(h)(-h)(-2h)} \int_a^b x^3 - x^2x_3 - x^2x_2 + xx_2x_3 - x^2x_0 + xx_0x_3 + xx_0x_2 - x_0x_2x_3dx \\
&= \frac{1}{2h^3} \int_a^b x^3 - x^2x_3 - x^2x_2 + xx_2x_3 - x^2x_0 + xx_0x_3 + xx_0x_2 - x_0x_2x_3dx \\
&= \frac{1}{2h^3} \int_a^b x^3 - x^2(x_3+x_2) + x(x_2x_3 + x_0x_3 + x_0x_2) - x_0x_2x_3dx \\
&= \frac{1}{2h^3}(\frac{1}{4}(b^4-a^4) - \frac{1}{3}(x_3+x_2)(b^3-a^3) + \frac{1}{2}(b^2-a^2)(x_2x_3+x_0x_3+x_0x_2) - (b-a)(x_0x_2x_3)) \\
&= \frac{1}{2h^3}(\frac{1}{4}((a+3h)^4-a^4) - \frac{1}{3}(2a+5h)((a+3h)^3-a^3) \\
&\quad + \frac{1}{2}((a+3h)^2-a^2)((a+2h)(a+3h) + a(a+3h) + a(a+2h)) - (3h)(a(a+2h)(a+3h))) \\
&= \frac{1}{2h^3}(-\frac{99h^4}{4} - 3ha^3 - 36ah^3 - \frac{39a^2h^2}{2} \\
&\quad + \frac{1}{2}((a+3h)^2-a^2)((a+2h)(a+3h) + a(a+3h) + a(a+2h)) - (3h)(a(a+2h)(a+3h))) \\
&= \frac{1}{2h^3}(-\frac{99h^4}{4} - 3ha^3 - 36ah^3 - \frac{39a^2h^2}{2} + \frac{54h^4+57h^2a^2+90ah^3}{2} + 6ha^3) \\
&= \frac{1}{2h^3}(\frac{9h^4}{4} + 9ah^3 + 9a^2h^2 + 3a^3h) \\
&= \frac{1}{2h^2}(\frac{9h^3}{4} + 9ah^2 + 9a^2h + 3a^3)
\end{aligned}
$$

The weights of Newton Cotes are symmetric, so $w_2 = w_1, w_3 = w_0$. Thus in total we have:

$$
\begin{aligned}
I(f) &= w_0 \cdot f(a) + w_1 \cdot f(a+h) + w_2 \cdot f(a+2h) + w_3 \cdot f(b) \\
&= \frac{1}{-6h}(\frac{63}{4}h^2 - 18a^2 - 18h^3) \cdot (f(a)+f(b)) \\
&\quad + \frac{1}{2h^2}(\frac{9h^3}{4} + 9ah^2 + 9a^2h + 3a^3) \cdot (f(a+h)+f(a+2h)) \\
&= \frac{b-a}{8}(f(a) + 3f(a+h) + 3f(a+2h) + f(b))
\end{aligned}
$$

Problem 3

1. The trapezoidal rule (derived from Newton Cotes with $n = 1$):

$$\int_a^b f(x)dx \approx (b-a)\frac{1}{2}(f(a) + f(b))$$

Thus we have:

$$\begin{aligned}\int_a^b f(x)dx &\approx (1-0)\cdot\frac{1}{2}(\frac{1}{1+2\cdot 0} + \frac{1}{1+2\cdot 1})\\ &= \frac{1}{2}(1+\frac{1}{3})\\ &= \frac{1}{2}(\frac{4}{3})\\ &= \frac{2}{3}\end{aligned}$$

2. Simpson's rule (derived from Newton Cotes with $n = 2$):

$$\int_a^b f(x)dx \approx (b-a)\frac{1}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$$

So we have:

$$\begin{aligned}\int_a^b f(x)dx &\approx \frac{1}{6}(1 + 4\cdot\frac{1}{1+2\cdot\frac{1}{2}} + \frac{1}{3})\\ &= \frac{1}{6}(1+2+\frac{1}{3})\\ &= \frac{10}{18}\\ &= \frac{5}{9}\end{aligned}$$

3. I found that the error dipped below 10^{-6} for Newton Cotes with $n = 10$ (so 11 evaluations of the integrand are required).
This is the code I used:

```
from scipy.integrate import newton_cotes
import numpy as np
ACTUAL =  0.5493064
def f(x):
    return 1/(1 + 2*x)
def n_point_rule(function, interval, n):
```

```
    a = interval[0]
    b = interval[1]
    x = np.linspace(a, b, n + 1)
    an, B = newton_cotes(n)
    h = (b - a) / n
    return h * np.sum(an * function(x))
def n3(function, interval): #newton cotes, n=3
    h = (interval[1] - interval[0]) / 3;
    return 3/8 * h * (function(interval[0]) + 3 * function(interval[0] + h) + 3* function(interval[0] + 2*h) + function(interval[1]))
def n4(function, interval):
    h = (interval[1] - interval[0]) / 4;
    a = interval[0]
    return 2/45 * h * (7 * function(a) + 32 * function(a + h) + 12 * function(a + 2 * h) + 32 * function(a + 3* h) + 7 * function(interval[1]))
def n5(function, interval):
    h = (interval[1] - interval[0]) / 5;
    a = interval[0]
    x_val = a;
    index = 0
    vals =[]
    while(index < 6):
        vals.append(function(x_val))
        x_val += h
        index += 1
    return 5/288 * h * (19* vals[0] + 75 * vals[1] + 50 * vals[2] + 50 * vals[3] + 75 * vals[4] + 19 * vals[5])
def n6(function, interval):
    h = (interval[1] - interval[0]) / 6;
    a = interval[0]
    x_val = a;
    index = 0
    vals =[]
    while(index < 7):
        vals.append(function(x_val))
        x_val += h
        index += 1
    return 1/140 * h * (41* vals[0] + 216 * vals[1] + 27 * vals[2] + 272 * vals[3] + 27 * vals[4] + 216 * vals[5] + 41 * vals[6])
def n7(function, interval):
    h = (interval[1] - interval[0]) / 7;
    a = interval[0]
    x_val = a;
    index = 0
    vals =[]
    while(index < 8):
```

```
        vals.append(function(x_val))
        x_val += h
        index += 1
    return 7/17280 * h * (751* vals[0] + 3577 * vals[1]
        + 1323 * vals[2] + 2989 * vals[3] + 2989 * vals
        [4] + 1323 * vals[5] + 3577 * vals[6] + 751 *
        vals[7])
def n8(function, interval):
    h = (interval[1] - interval[0]) / 8;
    a = interval[0]
    x_val = a;
    index = 0
    vals =[]
    while(index < 9):
        vals.append(function(x_val))
        x_val += h
        index += 1
    return 4/14175 * h * (989* vals[0] + 5888 * vals[1]
        - 928 * vals[2] + 10496 * vals[3] - 4540 * vals
        [4] + 10496 * vals[5] - 928 * vals[6] + 5888 *
        vals[7] + 989 * vals[8])
def main():
    interval = [0, 1]
    print("n =3 error: ", abs(n3(f, interval) - ACTUAL))
    print("n=4 error:", abs(n4(f, interval) - ACTUAL))
    print("n=5 error:", abs(n5(f, interval) - ACTUAL))
    print("n=6 error:", abs(n6(f, interval) - ACTUAL))
    print("n=7 error:", abs(n7(f, interval) - ACTUAL))
    print("n=8 error:", abs(n8(f, interval) - ACTUAL))
    for i in range(9, 15):
        print(f"n={i} error:", abs(n_point_rule(f,
            interval, i)) - ACTUAL)

if __name__ == "__main__":
    main()
```