

Homework 4 of Computational Mathematics

Chang, Yung-Hsuan

111652004

Department of Applied Mathematics

May 7, 2024

Problem 1. Use the following data and the knowledge that the first five derivatives of f are bounded on $[1, 5]$ by 2, 3, 6, 12, and 23, respectively, to approximate $f'(3)$ as accurately as possible. Find a bound for the error.

x	1	2	3	4	5
$f(x)$	2.4142	2.6734	2.8974	3.0976	3.2804

Solution. By the five-point midpoint formula,

$$\begin{aligned}
 f'(3) &= \frac{1}{12 \cdot 1} (f(1) - 8 \cdot f(2) + 8 \cdot f(4) - f(5)) + \frac{1^4}{30} \cdot f^{(5)}(\xi) \\
 &= \frac{1}{12} (2.4142 - 8 \cdot 2.6734 + 8 \cdot 3.0976 - 3.2804) + \frac{1}{30} \cdot f^{(5)}(\xi) \\
 &= 0.2106 + \frac{1}{30} \cdot f^{(5)}(\xi),
 \end{aligned}$$

where $\xi \in (1, 5)$. The bound for the error is

$$\left| \frac{1}{30} \cdot f^{(5)}(\xi) \right| \leq \frac{23}{30}.$$

□

Problem 2. Let $f(x) = 3xe^x - \cos x$. Use the following data and equation (4.9) to approximate $f''(1.3)$ with $h = 0.1$ and with $h = 0.01$.

x	1.20	1.29	1.30	1.31	1.40
$f(x)$	11.59006	13.78176	14.04276	14.30741	16.86187

Compare your results to $f''(1.3)$.

Solution. We first deal with the case $h = 0.1$. By the second derivative midpoint formula,

$$\begin{aligned}
 f''(1.3) &= \frac{1}{(0.1)^2} (f(1.20) - 2 \cdot f(1.30) + f(1.40)) - \frac{(0.1)^2}{12} \cdot f^{(4)}(\xi_1) \\
 &= \frac{1}{0.01} \cdot (11.59006 - 2 \cdot 14.04276 + 16.86187) - \frac{0.01}{12} \cdot f^{(4)}(\xi_1) \\
 &= 36.641 - \frac{0.01}{12} \cdot f^{(4)}(\xi_1),
 \end{aligned}$$

where $\xi_1 \in (1.20, 1.40)$. We now deal with the case $h = 0.01$. Again by the second derivative midpoint formula,

$$\begin{aligned}
 f''(1.3) &= \frac{1}{(0.01)^2} (f(1.29) - 2 \cdot f(1.30) + f(1.31)) - \frac{(0.01)^2}{12} \cdot f^{(4)}(\xi_2) \\
 &= \frac{1}{0.0001} \cdot (13.78176 - 2 \cdot 14.04276 + 14.30741) - \frac{0.0001}{12} \cdot f^{(4)}(\xi_2) \\
 &= 36.5 - \frac{0.0001}{12} \cdot f^{(4)}(\xi_2),
 \end{aligned}$$

where $\xi_2 \in (1.29, 1.31)$. The actual value of $f''(1.3)$ can be calculated as follows:

$$\begin{aligned}
 f'(x) &= 3e^x + 3xe^x + \sin x \\
 \implies f''(x) &= 6e^x + 3xe^x + \cos x \\
 \implies f''(1.3) &= 6e^{1.3} + 3 \cdot 1.3 \cdot e^{1.3} + \cos(1.3) \\
 &= 36.59354.
 \end{aligned}$$

The case with $h = 0.01$ is closed to the true value, and the other is farther to the true value. □

Problem 3. Derive an $\mathcal{O}(h^4)$ five-point formula to approximate $f'(x_0)$ that uses $f(x_0 - h)$, $f(x_0)$, $f(x_0 + h)$, $f(x_0 + 2h)$, and $f(x_0 + 3h)$. [Hint: Consider the expression $A \cdot f(x_0 - h) + B \cdot f(x_0 + h) + C \cdot f(x_0 + 2h) + D \cdot f(x_0 + 3h)$. Expand in fourth Taylor polynomials, and choose A , B , C , and D , appropriately.]

Solution. We follow the hint and expand them in fourth Taylor polynomials, obtaining

$$\begin{aligned} f(x_0 - h) &= f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_1)h^4 \\ f(x_0 + h) &= f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_2)h^4 \\ f(x_0 + 2h) &= f(x_0) + 2f'(x_0)h + 2f''(x_0)h^2 + \frac{4}{3}f'''(x_0)h^3 + \frac{2}{3}f^{(4)}(\xi_2)h^4 \\ f(x_0 + 3h) &= f(x_0) + 3f'(x_0)h + \frac{9}{2}f''(x_0)h^2 + \frac{9}{2}f'''(x_0)h^3 + \frac{27}{8}f^{(4)}(\xi_2)h^4. \end{aligned}$$

Then, we have

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 2 & 3 \\ \frac{1}{2} & \frac{1}{2} & 2 & \frac{9}{2} \\ \frac{1}{24} & \frac{1}{24} & \frac{2}{3} & \frac{27}{8} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

which implies $(A, B, C, D) = \left(-\frac{17}{24}, -\frac{5}{4}, \frac{4}{3}, -\frac{3}{8}\right)$. Hence,

$$\begin{aligned} f'(x_0)h &= -\frac{17}{24}f(x_0 - h) + f(x_0) - \frac{5}{4}f(x_0 + h) + \frac{4}{3}f(x_0 + 2h) - \frac{3}{8}f(x_0 + 3h) \\ f'(x_0) &= \frac{1}{h} \cdot \left(-\frac{17}{24}f(x_0 - h) + f(x_0) - \frac{5}{4}f(x_0 + h) + \frac{4}{3}f(x_0 + 2h) - \frac{3}{8}f(x_0 + 3h)\right). \end{aligned}$$

□

Problem 4. The following data give approximations to the integral

$$M = \int_0^\pi \sin x \, dx.$$

$$N_1(h) = 1.570796, \quad N_1\left(\frac{h}{2}\right) = 1.896119, \quad N_1\left(\frac{h}{4}\right) = 1.974232, \quad N_1\left(\frac{h}{8}\right) = 1.993570.$$

Assuming $M = N_1(h) + K_1h^2 + K_2h^4 + K_3h^6 + K_4h^8 + \mathcal{O}(h^{10})$, construct an extrapolation table to determine $N_4(h)$.

Solution. By the formula for the $\mathcal{O}(h^{2j})$ approximation

$$N_j(h) = N_{j-1}\left(\frac{h}{2}\right) + \frac{N_{j-1}\left(\frac{h}{2}\right) - N_{j-1}(h)}{4^{j-1} - 1}$$

and calculator, we can have the following table:

$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$
1.570796			
1.896119	2.004560		
1.974232	2.000270	1.999984	
1.993570	2.000016	1.999991	1.999999

□

Problem 5. The forward-difference formula can be expressed as

$$f'(x_0) = \frac{1}{h} (f(x_0 + h) - f(x_0)) - \frac{h}{2} f''(x_0) - \frac{h^2}{6} f'''(x_0) + \mathcal{O}(h^3).$$

Use extrapolation to derive an $\mathcal{O}(h^3)$ formula for $f'(x_0)$.

Solution. By the given forward-difference formula, we have

$$f'(x_0) = \frac{1}{h} (f(x_0 + h) - f(x_0)) - \frac{h}{2} f''(x_0) - \frac{h^2}{6} f'''(x_0) + \mathcal{O}(h^3). \quad (5.1)$$

Replacing the parameter h by half its value in (5.1) yields

$$f'(x_0) = \frac{2}{h} \left(f \left(x_0 + \frac{h}{2} \right) - f(x_0) \right) - \frac{h}{4} f''(x_0) - \frac{h^2}{24} f'''(x_0) + \mathcal{O}(h^3). \quad (5.2)$$

Subtracting (5.1) from twice (5.2) yields

$$f'(x_0) = \frac{4}{h} \left(f \left(x_0 + \frac{h}{2} \right) - f(x_0) \right) - \frac{1}{h} (f(x_0 + h) - f(x_0)) + \frac{h^2}{12} f'''(x_0) + \mathcal{O}(h^3). \quad (5.3)$$

Replacing the parameter h by half its value in (5.3) yields

$$f'(x_0) = \frac{8}{h} \left(f \left(x_0 + \frac{h}{4} \right) - f(x_0) \right) - \frac{2}{h} \left(f \left(x_0 + \frac{h}{2} \right) - f(x_0) \right) + \frac{h^2}{48} f'''(x_0) + \mathcal{O}(h^3). \quad (5.4)$$

Subtracting (5.3) from four-time (5.4) yields

$$\begin{aligned} f'(x_0) &= \frac{32}{h} \left(f \left(x_0 + \frac{h}{4} \right) - f(x_0) \right) - \frac{4}{h} \left(f \left(x_0 + \frac{h}{2} \right) - f(x_0) \right) \\ &\quad - \frac{8}{h} \left(f \left(x_0 + \frac{h}{2} \right) - f(x_0) \right) + \frac{1}{h} (f(x_0 + h) - f(x_0)) + \mathcal{O}(h^3). \end{aligned}$$

□

Problem 6. Find the degree of precision of the quadrature formula

$$\int_{-1}^1 f(x) \, dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right).$$

Solution. For $f(x) = P_0(x) = 1$, it is clear that

$$\int_{-1}^1 1 \, dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) = 2.$$

For $f(x) = P_1(x) = x$, it is also clear that

$$\int_{-1}^1 x \, dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) = 0.$$

For $f(x) = P_2(x) = x^2$,

$$\int_{-1}^1 x^2 \, dx = \frac{2}{3}$$

and

$$f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}.$$

For $f(x) = P_3(x) = x^3$, it is clear that

$$\int_{-1}^1 x^3 \, dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) = 0.$$

For $f(x) = P_4(x) = x^4$,

$$\int_{-1}^1 x^4 \, dx = \frac{2}{5}$$

and

$$f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) = \frac{1}{9} + \frac{1}{9} = \frac{2}{9}.$$

Hence, the degree of precision of this quadrature formula is 3. □

Problem 7. Find the constants x_0 , x_1 , and c_1 so that the quadrature formula

$$\int_0^1 f(x) \, dx = \frac{1}{2}f(x_0) + c_1f(x_1)$$

has the highest possible degree of precision.

Solution. Set $f(x) = P_0(x) = 1$. Then

$$1 = \frac{1}{2} + c_1.$$

This implies that c_1 must be $\frac{1}{2}$. Set $f(x) = P_1(x) = x$. Then

$$\frac{1}{2} = \frac{1}{2}x_0 + \frac{1}{2}x_1.$$

Set $f(x) = P_2(x) = x^2$. Then

$$\frac{1}{3} = \frac{1}{2}x_0^2 + \frac{1}{2}x_1^2.$$

Since there are two unknowns and two equations, two unknowns may be solved:

$$\frac{1}{3} = \frac{1}{2}(1 - x_1)^2 + \frac{1}{2}x_1^2$$

$$2 = 3(1 - x_1)^2 + 3x_1^2$$

$$x_1 = \frac{3 \pm \sqrt{3}}{6}.$$

Choose $x_0 < x_1$. Then $(x_0, x_1, c_1) = \left(\frac{3 - \sqrt{3}}{6}, \frac{3 + \sqrt{3}}{6}, \frac{1}{2}\right)$. □

Problem 8. Determine the values of n and h required to approximate

$$\int_0^2 e^{2x} \sin 3x \, dx$$

to within 10^{-4} . Use

- a. the composite trapezoidal rule;
- b. composite Simpson's rule; and
- c. the composite midpoint rule.

Solution. The real value of the integral is

$$\begin{aligned} \int_0^2 e^{2x} \sin 3x \, dx &= \left[\frac{1}{13} e^{2x} (2 \sin 3x - 3 \cos 3x) \right]_0^2 \\ &= -14.21397712986 \end{aligned}$$

- a. It is clear that $f \in C^2[0, 2]$. Choose $n = 20000$. Then $h = 0.0001$. Using the composite trapezoidal rule,

$$\begin{aligned} \int_0^2 e^{2x} \sin 3x \, dx &\approx \frac{0.0001}{2} \left[0 + 2 \cdot \sum_{j=1}^{19999} e^{2x_j} \sin(3x_j) + e^4 \sin(6) \right] \\ &= -14.213977026729639, \end{aligned}$$

where $x_j = 0.0001 \cdot j$.

- b. It is clear that $f \in C^4[0, 2]$. Choose $n = 20000$. Then $h = 0.0001$. Using composite Simpson's rule,

$$\begin{aligned} \int_0^2 e^{2x} \sin 3x \, dx &\approx \frac{0.0001}{3} \left[0 + 2 \cdot \sum_{j=1}^{9999} e^{2x_{2j}} \sin(3x_{2j}) + 4 \cdot \sum_{j=1}^{10000} e^{2x_{2j-1}} \sin(3x_{2j-1}) + e^2 \sin(6) \right] \\ &= -14.213977129862458, \end{aligned}$$

where $x_j = 0.0001 \cdot j$.

- c. It is clear that $f \in C^2[0, 2]$. Choose $n = 19998$. Then $h = 0.0001$. Using the composite midpoint

rule,

$$\int_0^2 e^{2x} \sin 3x \, dx \approx 2 \cdot 0.0001 \cdot \sum_{j=0}^{9999} f(x_{2j})$$
$$= -14.213977336128231,$$

where $x_j = 0.0001 \cdot (j + 1)$.

It can be seen that all of the three methods are accurate within 10^{-4} (even more precise).

□

```
TeX 111652004_CM_HW4.tex M 111652004_CH_MW4_P8.py U x
HW4 > 111652004_CH_MW4_P8.py > ...
1  import math
2
3  a = 0
4  b = 2
5  n = 20000
6
7  h = (b-a)/n
8
9  def x(i):
10 |     return a + h*i
11
12  def f(x):
13 |     return math.e ** (2 * x) * math.sin(3 * x)
14
15  ctSum = f(a)+f(b)
16  for i in range(n-1):
17 |     ctSum += 2 * f(x(i+1))
18  print("Composite trapezoidal rule:", h * ctSum / 2)
19
20  cSSum = f(a)+f(b)
21  for i in range(n//2-1):
22 |     cSSum += 2 * f(x(2*(i+1)))
23  for i in range(n//2):
24 |     cSSum += 4 * f(x(2*(i+1)-1))
25  print("Composite Simpson's rule:", h * cSSum / 3)
26
27
28  def x_cm(i):
29 |     return a + h*(i+1)
30  n = 19998
31  h = (b-a)/(n+2)
32  cmSum = 0
33  for i in range(n//2+1):
34 |     cmSum += f(x_cm(2*i))
35  print("Composite midpoint rule:", 2 * h * cmSum)
36
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
● thub_Clone/CM_HW/HW4/111652004_CH_MW4_P8.py"
Composite trapezoidal rule: -14.213977026729639
Composite Simpson's rule: -14.213977129862458
Composite midpoint rule: -14.213977336128231
```

Problem 9. Determine to within 10^{-6} the length of the graph of the ellipse with $4x^2 + 9y^2 = 36$.

Solution. We use composite Simpson's rule to answer. The desired integral is

$$2 \cdot \int_0^\pi \sqrt{(-3 \sin \theta)^2 + (2 \cos \theta)^2} d\theta.$$

The function $f(x)$ can be re-written as $\sqrt{4 + 5(\sin \theta)^2}$. Choose $n = 1000$, so that the error term

$$\begin{aligned} \frac{\pi - 0}{180} \cdot \left(\frac{\pi - 0}{1000}\right)^4 \cdot f^{(4)}(\mu) &\leq \frac{\pi - 0}{180} \cdot \left(\frac{\pi - 0}{1000}\right)^4 \cdot \sup_{x \in \mathbb{R}} |f^{(4)}(x)| \\ &< \frac{\pi}{180} \cdot 0.01^4 \cdot 20 \\ &< 3.5 \times 10^{-9}. \end{aligned}$$

By Python and composite Simpson's rule,

$$\begin{aligned} 2 \cdot \int_0^\pi \sqrt{(-3 \sin \theta)^2 + (2 \cos \theta)^2} d\theta &\approx 2 \cdot \frac{\pi}{1000} \cdot \frac{1}{3} \left[f(0) + 2 \cdot \sum_{j=1}^{499} f(x_{2j}) + 4 \cdot \sum_{j=1}^{500} f(x_{2j-1}) + f(\pi) \right] \\ &= 15.8654393826, \end{aligned}$$

where $x_i = a + hi$. The true value is around $8E\left(-\frac{5}{4}\right) = 15.8654395893$; thus the absolute error is

$$|15.8654393826 - 15.8654395893| = 0.0000002067$$

$$< 3 \times 10^{-7}$$

$$< 10^{-6}.$$

□

```
111652004_CM_HW4.tex M 111652004_CM_HW4_P9.py U X
HW4 > 111652004_CM_HW4_P9.py > ...
1  import math
2
3  a = 0
4  b = math.pi
5  n = 1000
6
7  h = (b - a) / n
8
9  def x(n):
10 |     return a + n * h
11
12 def f(x):
13 |     a = math.sqrt(4 + 5 * math.sin(x) ** 2)
14 |     return a
15
16 sum = f(a) + f(b)
17
18 for i in range(n//2 - 1):
19 |     sum += 2 * f(x(2 * i))
20
21 for i in range(n//2):
22 |     sum += 4 * f(x(2 * i-1))
23
24 print(2 * h * sum / 3)
25

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
/usr/bin/python3 "/Users/eiken/Visual Studio/Github_Clone/CM_HW/HW4/111652004_CM_HW4_P9.py"
eiken@Eikens-MacBook-Air CM_HW % /usr/bin/python3 "/Users/eiken/Visual Studio/Github_Clone/CM_HW/HW4/111652004_CM_HW4_P9.py"
15.865439382587326
```

Problem 10. Show that the error $E(f)$ for composite Simpson's rule can be approximated by

$$-\frac{h^4}{180} (f'''(b) - f'''(a)) .$$

[Hint: $\sum_{j=1}^{n/2} f^{(4)}(\xi_j)(2h)$ is a Riemann sum for $\int_a^b f^{(4)}(x) \, dx$.]

Solution. By textbook, the error for composite Simpson's rule is

$$-\frac{h^5}{90} \sum_{j=1}^{n/2} f^{(4)}(\xi_j) = -\frac{h^4}{180} \sum_{j=1}^{n/2} f^{(4)}(\xi_j)(2h)$$

for $\xi_j \in (x_{2j-2}, x_{2j})$. By hint, we have

$$\begin{aligned} -\frac{h^4}{180} \sum_{j=1}^{n/2} f^{(4)}(\xi_j)(2h) &= -\frac{h^4}{180} \int_a^b f^{(4)}(x) \, dx \\ &= -\frac{h^4}{180} (f'''(b) - f'''(a)) \end{aligned}$$

by the fundamental theorem of calculus, as desired. □

Problem 11. Use the following data to approximate $\int_1^5 f(x) \, dx$ as accurately as possible.

x	1	2	3	4	5
$f(x)$	2.4142	2.6734	2.8974	3.0976	3.2804

Solution. We use the Romberg extrapolation for higher accuracy.

$$R_{1,1} = 4 \cdot (2.4142 + 3.2804) / 2 = 11.3892;$$

$$R_{2,1} = 2 \cdot (2.4142 + 2 \cdot 2.8974 + 3.2804) / 2 = 11.4894;$$

$$R_{3,1} = 1 \cdot (2.4142 + 2.6734 + 2.8974 + 3.0976 + 3.2804) = 11.5157;$$

$$R_{2,2} = R_{2,1} + \frac{1}{4^1 - 1} (R_{2,1} - R_{1,1})$$

$$= 11.5228;$$

$$R_{3,2} = R_{3,1} + \frac{1}{4^1 - 1} (R_{3,1} - R_{2,1})$$

$$= 11.5245;$$

$$R_{3,3} = R_{3,2} + \frac{1}{4^2 - 1} (R_{3,2} - R_{2,2})$$

$$= 11.5246.$$

$R_{3,3}$ is the desired answer.

□

Problem 12. Show that the approximation obtained from $R_{k,2}$ is the same as that given by the composite Simpson's rule described in Theorem 4.4 with $h = h_k$.

Solution. By textbook,

$$\begin{aligned}
 R_{k,2} &= R_{k,1} + \frac{R_{k,1} - R_{k-1,1}}{3} \\
 &= \frac{1}{3}(4R_{k,1} - R_{k-1,1}) \\
 &= \frac{1}{3} \left(4 \cdot \frac{1}{2} \left(R_{k-1,1} + 2h_k \sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k) \right) - R_{k-1,1} \right) \\
 &= \frac{1}{3} \left(R_{k-1,1} + 2h_k \sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k) \right) \\
 &= \frac{h_k}{3} \left(f(a) + f(b) + 4 \sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k) + 2 \sum_{i=1}^{2^{k-2}-1} f(a + 2ih_k) \right),
 \end{aligned}$$

which is the same as the one given by composite Simpson's rule. □

Problem 13. Use composite Simpson's rule with $n = 4, 6, 8, \dots$, until successive approximations to the following integrals agree to within 10^{-6} . Determine the number of nodes required. Use the adaptive quadrature algorithm to approximate the integral to within 10^{-6} , and count the number of nodes. Did the adaptive quadrature produce any improvement?

a. $\int_0^\pi x \sin(x^2) \, dx$; and

b. $\int_0^\pi x^2 \sin x \, dx$.

Solution. The true value for a is $\left(\sin\left(\frac{\pi^2}{2}\right)\right)^2$. The true value for b is $\pi^2 - 4$. By Python, for a, it takes 148 steps to obtain successive approximations to the integral agree to within 10^{-6} . It takes 55 steps using the adaptive quadrature for a; for b, it takes 36 steps to obtain successive approximations to the integral agree to within 10^{-6} . It takes 28 steps using the adaptive quadrature for a. The adaptive quadrature did produce improvements. □


```
111652004_CM_HW4.tex M 111652004_CM_HW4_P13.py U x
HW4 > 111652004_CM_HW4_P13.py > ...
1 import math
2
3 TOL = 0.000001
4 true_a = (math.sin(math.pi ** 2 / 2)) ** 2
5 true_b = math.pi ** 2 - 4
6
7 > def cSimpson(f, n, a, b): ...
17
18 def adaptiveQuadrature(f, a, b, TOL, num_of_recc):
19     def recursive(f, a, b, TOL, fa, fb, fc, approx_0, num_of_recc):
20         c = (a + b) / 2; h = (b - a) / 2
21         d = (a + c) / 2; e = (c + b) / 2
22         fd = f(d); fe = f(e)
23         approx_l = h * (fa + 4 * fd + fc) / 6
24         approx_r = h * (fc + 4 * fe + fb) / 6
25         approx = approx_l + approx_r
26         if abs(approx - approx_0) <= TOL:
27             num_of_recc[0] += 1
28             return approx
29         else:
30             return (recursive(f, a, c, TOL / 2, fa, fc, fd, approx_l,
31                             num_of_recc) +
32                     recursive(f, c, b, TOL / 2, fc, fb, fe, approx_r,
33                             num_of_recc))
34
35     fa = f(a); fb = f(b); fc = f((a + b) / 2)
36
37     approx_0 = (b - a) * (fa + 4 * fc + fb) / 6
38
39     return (recursive(f, a, b, 10 * TOL, fa, fb, fc, approx_0,
40                     num_of_recc), num_of_recc[0])
41
42 def f_a(x):
43     return x * math.sin(x ** 2)
44
45 def f_b(x):
46     return x * math.sin(x ** 2)
47
48 def f_c(x):
49     return x * math.sin(x ** 2)
50
51 def f_d(x):
52     return x * math.sin(x ** 2)
53
54 def f_e(x):
55     return x * math.sin(x ** 2)
56
57 def f_f(x):
58     return x * math.sin(x ** 2)
59
60 def f_g(x):
61     return x * math.sin(x ** 2)
62
63 def f_h(x):
64     return x * math.sin(x ** 2)
65
66 def f_i(x):
67     return x * math.sin(x ** 2)
68
69 def f_j(x):
70     return x * math.sin(x ** 2)
71
72 def f_k(x):
73     return x * math.sin(x ** 2)
74
75 def f_l(x):
76     return x * math.sin(x ** 2)
77
78 def f_m(x):
79     return x * math.sin(x ** 2)
80
81 def f_n(x):
82     return x * math.sin(x ** 2)
83
84 def f_o(x):
85     return x * math.sin(x ** 2)
86
87 def f_p(x):
88     return x * math.sin(x ** 2)
89
90 def f_q(x):
91     return x * math.sin(x ** 2)
92
93 def f_r(x):
94     return x * math.sin(x ** 2)
95
96 def f_s(x):
97     return x * math.sin(x ** 2)
98
99 def f_t(x):
100    return x * math.sin(x ** 2)
101
102 def f_u(x):
103    return x * math.sin(x ** 2)
104
105 def f_v(x):
106    return x * math.sin(x ** 2)
107
108 def f_w(x):
109    return x * math.sin(x ** 2)
110
111 def f_x(x):
112    return x * math.sin(x ** 2)
113
114 def f_y(x):
115    return x * math.sin(x ** 2)
116
117 def f_z(x):
118    return x * math.sin(x ** 2)
119
120 def f_aa(x):
121    return x * math.sin(x ** 2)
122
123 def f_ab(x):
124    return x * math.sin(x ** 2)
125
126 def f_ac(x):
127    return x * math.sin(x ** 2)
128
129 def f_ad(x):
130    return x * math.sin(x ** 2)
131
132 def f_ae(x):
133    return x * math.sin(x ** 2)
134
135 def f_af(x):
136    return x * math.sin(x ** 2)
137
138 def f_ag(x):
139    return x * math.sin(x ** 2)
140
141 def f_ah(x):
142    return x * math.sin(x ** 2)
143
144 def f_ai(x):
145    return x * math.sin(x ** 2)
146
147 def f_aj(x):
148    return x * math.sin(x ** 2)
149
150 def f_ak(x):
151    return x * math.sin(x ** 2)
152
153 def f_al(x):
154    return x * math.sin(x ** 2)
155
156 def f_am(x):
157    return x * math.sin(x ** 2)
158
159 def f_an(x):
160    return x * math.sin(x ** 2)
161
162 def f_ao(x):
163    return x * math.sin(x ** 2)
164
165 def f_ap(x):
166    return x * math.sin(x ** 2)
167
168 def f_aq(x):
169    return x * math.sin(x ** 2)
170
171 def f_ar(x):
172    return x * math.sin(x ** 2)
173
174 def f_as(x):
175    return x * math.sin(x ** 2)
176
177 def f_at(x):
178    return x * math.sin(x ** 2)
179
180 def f_au(x):
181    return x * math.sin(x ** 2)
182
183 def f_av(x):
184    return x * math.sin(x ** 2)
185
186 def f_aw(x):
187    return x * math.sin(x ** 2)
188
189 def f_ax(x):
190    return x * math.sin(x ** 2)
191
192 def f_ay(x):
193    return x * math.sin(x ** 2)
194
195 def f_az(x):
196    return x * math.sin(x ** 2)
197
198 def f_ba(x):
199    return x * math.sin(x ** 2)
200
201 def f_bb(x):
202    return x * math.sin(x ** 2)
203
204 def f_bc(x):
205    return x * math.sin(x ** 2)
206
207 def f_bd(x):
208    return x * math.sin(x ** 2)
209
210 def f_be(x):
211    return x * math.sin(x ** 2)
212
213 def f_bf(x):
214    return x * math.sin(x ** 2)
215
216 def f_bg(x):
217    return x * math.sin(x ** 2)
218
219 def f_bh(x):
220    return x * math.sin(x ** 2)
221
222 def f_bi(x):
223    return x * math.sin(x ** 2)
224
225 def f_bj(x):
226    return x * math.sin(x ** 2)
227
228 def f_bk(x):
229    return x * math.sin(x ** 2)
230
231 def f_bl(x):
232    return x * math.sin(x ** 2)
233
234 def f_bm(x):
235    return x * math.sin(x ** 2)
236
237 def f_bn(x):
238    return x * math.sin(x ** 2)
239
240 def f_bo(x):
241    return x * math.sin(x ** 2)
242
243 def f_bp(x):
244    return x * math.sin(x ** 2)
245
246 def f_bq(x):
247    return x * math.sin(x ** 2)
248
249 def f_br(x):
250    return x * math.sin(x ** 2)
251
252 def f_bs(x):
253    return x * math.sin(x ** 2)
254
255 def f_bt(x):
256    return x * math.sin(x ** 2)
257
258 def f_bu(x):
259    return x * math.sin(x ** 2)
260
261 def f_bv(x):
262    return x * math.sin(x ** 2)
263
264 def f_bw(x):
265    return x * math.sin(x ** 2)
266
267 def f_bx(x):
268    return x * math.sin(x ** 2)
269
270 def f_by(x):
271    return x * math.sin(x ** 2)
272
273 def f_bz(x):
274    return x * math.sin(x ** 2)
275
276 def f_ca(x):
277    return x * math.sin(x ** 2)
278
279 def f_cb(x):
280    return x * math.sin(x ** 2)
281
282 def f_cc(x):
283    return x * math.sin(x ** 2)
284
285 def f_cd(x):
286    return x * math.sin(x ** 2)
287
288 def f_ce(x):
289    return x * math.sin(x ** 2)
290
289
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

a.
It takes 148 steps to obtain successive approximations with composite Simson's rule for a.
For apadative quadrature, it takes 55 steps.

b.
It takes 36 steps to obtain successive approximations with composite Simson's rule for b.
For apadative quadrature, it takes 28 steps.

Problem 14. Approximate the following integral using Gaussian quadrature with $n = 2, 3, 4$. Compare your answers with the exact values of the integral.

$$\int_1^{1.6} \frac{2x}{x^2 - 4} dx.$$

Solution. We first transform the interval $[1, 1.6]$ to $[-1, 1]$ by using $t = \frac{2x - 2.6}{0.6}$. Hence the integral becomes

$$\int_{-1}^1 \frac{0.6t + 2.6}{\left(\frac{0.6t + 2.6}{2}\right)^2 - 4} \cdot \frac{0.6}{2} dt = \int_{-1}^1 \frac{6t + 26}{3t^2 + 26t - 77} dt.$$

We now deal with the case $n = 2$. Using Table 4.12, the approximation is

$$\begin{aligned} & 1 \cdot \frac{12 \cdot 0.57735 + 26}{(0.57735)^2 + 26 \cdot 0.57735 - 77} + 1 \cdot \frac{12 \cdot (-0.57735) + 26}{(-0.57735)^2 + 26 \cdot (-0.57735) - 77} \\ &= -0.73072. \end{aligned}$$

We now deal with the case $n = 3$. Using Table 4.12, the approximation is

$$\begin{aligned} & 0.55556 \cdot \frac{12 \cdot 0.77460 + 26}{(0.77460)^2 + 26 \cdot 0.77460 - 77} \\ & + 0.88889 \cdot \frac{12 \cdot 0 + 26}{(0)^2 + 26 \cdot 0 - 77} \\ & + 0.55556 \cdot \frac{12 \cdot (-0.77460) + 26}{(-0.77460)^2 + 26 \cdot (-0.77460) - 77} \\ &= -0.73380. \end{aligned}$$

We now deal with the case $n = 4$. Using Table 4.12, the approximation is

$$\begin{aligned} & 0.34785 \cdot \frac{12 \cdot 0.86114 + 26}{(0.86114)^2 + 26 \cdot 0.86114 - 77} \\ & + 0.65215 \cdot \frac{12 \cdot 0.33998 + 26}{(0.33998)^2 + 26 \cdot 0.33998 - 77} \\ & + 0.65215 \cdot \frac{12 \cdot (-0.33998) + 26}{(-0.33998)^2 + 26 \cdot (-0.33998) - 77} \\ & + 0.34785 \cdot \frac{12 \cdot (-0.86114) + 26}{(-0.86114)^2 + 26 \cdot (-0.86114) - 77} \\ &= -0.73396. \end{aligned}$$

The exact value of the integral is

$$\begin{aligned}\int_1^{1.6} \frac{2x}{x^2 - 4} dx &= \int_{-3}^{-1.44} \frac{1}{u} du \\ &= \ln(0.48) \\ &= -0.73397.\end{aligned}$$

The relative errors are 0.0044228, 0.00022582, and 0.000012200, in the order of $n = 2$, $n = 3$, and $n = 4$.

The process of calculation is done by Python. □

```
HW4 > 111652004_CM_HW4_P14.py > ...
1  import math
2
3  def f(t):
4      return (6 * t + 26) / (3 * t ** 2 + 26 * t - 77)
5
6  n_values = []
7
8  node_data = {
9      2: ([0.57735, -0.57735], [1, 1]),
10     3: ([0.77460, 0, -0.77460], [0.55556, 0.88889, 0.55556]),
11     4: ([0.86114, 0.33998, -0.33998, -0.86114], [0.34785, 0.65215, 0.65215, 0.34785])
12 }
13
14 for num_nodes in range(2, 5):
15     sum = 0
16     roots, coeffs = node_data[num_nodes]
17     for i in range(num_nodes):
18         sum += coeffs[i] * f(roots[i])
19     n_values.append(sum)
20
21 rel_error = []
22
23 for i in range(3):
24     rel_error.append(abs(n_values[i] - math.log(0.48)) / abs(math.log(0.48)))
25
26 print(n_values)
27 print(rel_error)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

```
/usr/bin/python3 "/Users/eiken/Visual Studio/Github_Clone/CM_HW/HW4/111652004_CM_HW4_P14.py"
eiken@Eikens-MacBook-Air CM_HW % /usr/bin/python3 "/Users/eiken/Visual Studio/Github_Clone/CM_HW/HW4/111652004_CM_HW4_P14.py"
[-0.7307229812146365, -0.7338034323569402, -0.7339602207957825]
[0.0044227931850260565, 0.00022581700824436885, 1.2199809912944473e-05]
```

Problem 15. Determine constants a , b , c , and d that will produce a quadrature formula

$$\int_{-1}^1 f(x) \, dx = a \cdot f(-1) + b \cdot f(1) + c \cdot f'(-1) + d \cdot f'(1).$$

that has degree of precision 3.

Solution. Set $f(x) = P_0(x) = 1$. Then

$$2 = a + b.$$

Set $f(x) = P_1(x) = x$. Then

$$0 = -a + b + c + d.$$

Set $f(x) = P_2(x) = x^2$. Then

$$\frac{2}{3} = a + b - 2c + 2d.$$

Set $f(x) = P_3(x) = x^3$. Then

$$0 = -a + b + 3c + 3d.$$

Hence, $(a, b, c, d) = (1, 1, 1/3, -1/3)$.

□

Problem 16. The improper integral

$$\int_0^{\infty} f(x) \, dx$$

cannot be converted into an integral with finite limits using the substitution $t = \frac{1}{x}$ because the limit at zero becomes infinite. The problem is resolved by first writing

$$\int_0^{\infty} f(x) \, dx = \int_0^1 f(x) \, dx + \int_1^{\infty} f(x) \, dx.$$

Apply this technique to approximate the following improper integrals to within 10^{-6} .

a. $\int_0^{\infty} \frac{1}{1+x^4} \, dx$; and

b. $\int_0^{\infty} \frac{1}{(1+x^2)^3} \, dx$.

Solution. The true value for a is $\frac{\pi}{2\sqrt{2}}$. The true value for b is $\frac{3\pi}{16}$. Using this technique, the integral becomes

$$\int_0^1 \frac{1}{1+x^4} \, dx + \int_0^1 \frac{t^2}{t^4+1} \, dt$$

and

$$\int_0^1 \frac{1}{(1+x^2)^3} \, dx + \int_0^1 \frac{t^4}{(t^2+1)^3} \, dt.$$

By Python, the values are 1.1107208061 and 0.58904866, respectively. Both of which are approximated to the integral to within 10^{-6} (actually less than 10^{-7}). □

```
111652004_CM_HW4.tex M 111652004_CM_HW4_P16.py U X
HW4 > 111652004_CM_HW4_P16.py > ...
1 import math
2
3 TOL = 1e-6
4
5 def adaptiveQuadrature(f, a, b, TOL):
6     def recursive(f, a, b, TOL, fa, fb, fc, approx_0):
7         c = (a + b) / 2; h = (b - a) / 2
8         d = (a + c) / 2; e = (c + b) / 2
9         fd = f(d); fe = f(e)
10        approx_l = h * (fa + 4 * fd + fc) / 6
11        approx_r = h * (fc + 4 * fe + fb) / 6
12        approx = approx_l + approx_r
13        if abs(approx - approx_0) <= TOL:
14            return approx
15        else:
16            return (recursive(f, a, c, TOL / 2, fa, fc, fd, approx_l) +
17                    recursive(f, c, b, TOL / 2, fc, fb, fe, approx_r))
18
19        fa = f(a); fb = f(b); fc = f((a + b) / 2)
20
21        approx_0 = (b - a) * (fa + 4 * fc + fb) / 6
22
23        return recursive(f, a, b, 10 * TOL, fa, fb, fc, approx_0)
24
25 def f_a(x):
26     return 1 / (1 + x ** 4)
27
28 def f_a_i(t):
29     return t ** 2 / (1 + t ** 4)
30
31 def f_b(x):
32     return 1 / (1 + x ** 2) ** 3
33
34 def f_b_i(t):
35     return t ** 4 / (t ** 2 + 1) ** 3
36
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
eiken@Eikens-MacBook-Air CM_HW % /usr/bin/python3 "/Users/eiken/Visual Studio/Github_Clone/CM_HW/HW4/111652004_CM_H
a: 1.110720806090411, 7.155081949150599e-08
b: 0.5890486599751326, 3.7427046351012905e-08
```