Київський національний університет
імені Т.Шевченка

Звіт

до лабораторної роботи №1
з дисципліни:

«Комп'ютерне Моделювання»

Студента третього курсу
Групи МІ-32
Факультету комп'ютерних наук
та кібернетики
Федорича Андрія

Київ-2023

Варіанти: 10, 23, 36

Мова виконання: Python

GitHub Code: [StopFuture]

Постановка задачі 10:

10. Наближено обчислити довжину дуги еліпса за формулою $L = 4\int\limits_0^{\pi/2} \sqrt{a^2 \sin^2 t + b^2 \cos^2 t} dt$ за допомогою таблиці Ромберга, використавши 4 кроки.

Теоретичні відомості:

Таблиця Ромберга. За допомогою таблиці Ромберга можна з високою точністю обчислити інтеграл від достатньо гладкої функції.

$$
\begin{array}{ccccc}
I_h^{(0)} & & & & \\
I_{\frac{h}{2}}^{(0)} & I_{\frac{h}{2}}^{(1)} & & & \\
I_{\frac{h}{4}}^{(0)} & I_{\frac{h}{4}}^{(1)} & I_{\frac{h}{4}}^{(2)} & & \\
I_{\frac{h}{8}}^{(0)} & I_{\frac{h}{8}}^{(1)} & I_{\frac{h}{8}}^{(2)} & I_{\frac{h}{8}}^{(3)} & \\
\ldots & \ldots & \ldots & \ldots & \ldots
\end{array}
$$

Для побудови таблиці обчислюються інтеграли за допомогою складених квадратурних формул із сталим кроком. Перший стовпчик таблиці – інтеграли, які обчислюються за допомогою ділення кроку навпіл, вони мають порядок точності p. Другий стовпчик – інтеграли, які обчислюються за допомогою формули екстраполяції Річардсона (19) та мають порядок точності $(p+s)$. Якщо до отриманих інтегралів знов застосувати формулу Річардсона, то отримаємо інтеграли третього стовпця, точність яких підвищується до $(p+2s)$, процес продовжується, поки не досягається необхідна точність.

Похибку кожного з отриманих інтегралів (крім останнього), можна оцінити за правилом Рунге (16):

$$|I - I_{\frac{h}{2^k}}^{(l)}| \approx \left| \frac{I_{\frac{h}{2^k}}^{(l)} - I_{\frac{h}{2^{k-1}}}^{(l)}}{2^{p+sl} - 1} \right|.$$

Зауваження. Для складених формул середніх прямокутників, трапецій та Сімпсона $s = 2$.

Екстраполяційна формула Річардсона. З формули (16) безпосередньо випливає *наближення Річардсона*, яке дає змогу отримати більш точне значення інтеграла:

$$I \approx I_{\frac{h}{2}} + \frac{I_{\frac{h}{2}} - I_h}{2^p - 1} = \frac{2^p I_{\frac{h}{2}} - I_h}{2^p - 1}. \tag{19}$$

Попередні розрахунки/ Перевірка:

Перш за все можемо сказати, що функція є достатньо гладкою, адже має неперервну похідну на всій області визначення.

Тепер щоб спростити обчислення, замість змінних a та b підставимо певні значення, для прикладу, a = 3 і b = 2, потім в коді перевіримо для інших.

Крок 0:

h = (π/2 - 0) = π/2

R[0][0] = I_pi/2 = 5pi / 4= 3.9269908169872414

Крок 1:

h = (π/2 - 0) / 2 = π/4

R[1][0] = R[0][0] / 2 + (f(π/8) +f(3π/8)) * (π/4) = 3.965875689045382

R[1][1] = (4^1 * R[1][0] - R[0][0]) / (4^1 - 1) = (4*3.965875689045382 - 3.9269908169872414) / 3 = 3.978837313064762

Все це і так обчислювалалося технікою, тому дозволимо програмі знайти все далі.

Отримаємо такі результати:

```python
# Example usage
a = 3
b = 2
accuracy = 1e-8
number_of_steps = 4

arc_length = ellipse_arc_length(a, b, accuracy, number_of_steps)
print(f"The length of the ellipse arc with a={a} and b={b} is approximately {4*arc_length:.8f}")
```

```
[3.9269908169872414, 0, 0, 0, 0]
[3.965875689045382, 3.978837313064762, 0, 0, 0]
[3.9663596385164506, 3.966520955006807, 3.9656998644696104, 0, 0]
[3.9663598973224192, 3.9663599835910754, 3.96634925216336, 3.966359559904531, 0]
[3.966359897322647, 3.966359897322723, 3.9663598915714995, 3.966360060450994, 3.9663600624139215]
The length of the ellipse arc with a=3 and b=2 is approximately 15.86544025
```

Побудована таблиця Ромберга та бажаний результат для дуги еліпса = 4 * R[4][4] = 15.86544025.

Також перевіримо нашу програму з WolframAlpha:

Definite integral

$$\int_0^{\frac{\pi}{2}} \sqrt{9 \sin(t) \sin(t) + 4 \cos(t) \cos(t)}\, dt = 2\, E\left(-\frac{5}{4}\right) \approx 3.96636$$

Проте слід зазначити, що формула з умови, шукає не довжину дуги еліпса задану між двома точками(межами інтегрування), а довжину всього еліпса, це легко перевірити, якщо підставити в формулу коло, тобто параметри a та b = 1.

```python
# Example usage
a = 1
b = 1
accuracy = 1e-8
number_of_steps = 4

arc_length = ellipse_arc_length(a, b, accuracy, number_of_steps)
print(f"The length of the ellipse arc with a={a} and b={b} is approximately {4*arc_length:.8f}")
```

```
The length of the ellipse arc with a=1 and b=1 is approximately 6.28318531
```

L = 2 * pi * R = 2 * pi = 6.2831…

Код програми:

```python
import math

def romberg_integration(f, a, b, n):
    h = b - a
    R = [[0] * (n+1) for _ in range(n+1)]
    R[0][0] = (f(a) + f(b)) * h / 2

    for i in range(1, n+1):
        h /= 2
        R[i][0] = R[i-1][0] / 2 + sum(f(a + j*h) for j in range(1, 2**i, 2)) * h

        for k in range(1, i+1):
            R[i][k] = (4**k * R[i][k-1] - R[i-1][k-1]) / (4**k - 1)

    if n == 4:
         for r in R:
            print(r)
    return R[n][n]

def ellipse_arc_length(a, b, accuracy, cnt):
    def integrand(t):
        return math.sqrt(a**2 * math.sin(t)**2 + b**2 * math.cos(t)**2)

    n = 1
    previous_result = 0
    current_result = romberg_integration(integrand, 0, math.pi/2, n)

    while abs(current_result - previous_result) > accuracy and n != cnt:
        n += 1
        previous_result = current_result
        current_result = romberg_integration(integrand, 0, math.pi/2, n)

    return current_result

# Example usage
a = 2
b = 3
accuracy = 1e-8
number_of_steps = 4

arc_length = ellipse_arc_length(a, b, accuracy, number_of_steps)
print(f"The length of the ellipse arc with a={a} and b={b} is approximately {4*arc_length:.8f}")
```

```
[3.9269908169872414, 0, 0, 0, 0]
[3.965875689045382, 3.978837313064762, 0, 0, 0]
[3.9663596385164506, 3.966520955006807, 3.9656998644696104, 0, 0]
[3.9663598973224192, 3.9663599835910754, 3.96634925216336, 3.966359559904531, 0]
[3.966359897322647, 3.966359897322723, 3.9663598915714995, 3.966360060450994, 3.9663600624139215]
The length of the ellipse arc with a=2 and b=3 is approximately 15.86544025
```

P.s. Бачимо, що при зміні a та b місцями, отримуємо те саме значення, адже це просто еліпс сплющений відносно іншої осі.

Постановка задачі 23:

23. Наближено обчислити інтеграл $I = \int\limits_0^1 \frac{e^x}{\sqrt{x}} dx$ методом Канторовича. Використати квадратурну формулу Сімпсона.

Теоретичні відомості:

Квадратурна формула Сімпсона:

Формула Сімпсона. Поклавши у формулі Ньютона-Котеса замкненого типу $n = 2$, отримуємо формулу парабол (Сімпсона)

$$\int\limits_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right),$$

з оцінкою залишкового члена

$$|R(f)| \leqslant \frac{M_4(b-a)^5}{2880}.$$

Складена формула Сімпсона з оцінкою залишкового члена має вигляд:

$$I \approx \frac{h}{6}\left(f(x_0) + 4\sum_{i=1}^{n} f(x_{i-\frac{1}{2}}) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right), \quad (14)$$

$$|R(f)| \leqslant \frac{M_4(b-a)h^4}{2880}.$$

Враховуючи зауваження про парні степені інтерполяції, алгебраїчний степінь точності квадратурної формули дорівнює 3. Порядок точності складеної формули – 4, а по одному проміжку – 5.

Метод Канторовича:

Метод виділення особливостей (Канторовича) знов використовує представлення інтеграла у вигляді суми:

$$I = \int_a^b f(x)dx = \int_a^b g(x)dx + \int_a^b (f(x) - g(x))dx,$$

де функція $g(x)$ має таку ж особливість, як $f(x)$; функція $(f(x) - g(x))$ – достатньо гладка: $(f(x) - g(x)) \in C^{(m)}_{[a;b]}$, $m \geqslant 1$.

Розглянемо метод для інтегралів вигляду

$$I = \int_a^b \frac{\varphi(x)}{(x - x_0)^\alpha} dx$$

з особливою точкою $x_0 \in [a; b]$, $\alpha \in (0; 1)$.

Функцію $\varphi(x)$ розкладають в ряд Тейлора:

$$\varphi(x) = \sum_{k=0}^{m} \frac{\varphi^{(k)}(x_0)}{k!}(x - x_0)^k + \Psi(x) = P_m(x) + \Psi(x) \Rightarrow$$

$$\Psi(x) = \varphi(x) - \sum_{k=0}^{m} \frac{\varphi^{(k)}(x_0)}{k!}(x - x_0)^k;$$

$$\begin{aligned}
I &= \int_a^b \frac{\varphi(x)}{(x - x_0)^\alpha} dx = \int_a^b \frac{P_m(x) + \Psi(x)}{(x - x_0)^\alpha} dx = \\
&= \int_a^b \frac{1}{(x - x_0)^\alpha} \left(\sum_{k=0}^{m} \frac{\varphi^{(k)}(x_0)}{k!}(x - x_0)^k + \Psi(x) \right) dx = \\
&= \sum_{k=0}^{m} \frac{\varphi^{(k)}(x_0)}{k!} \int_a^b (x - x_0)^{k-\alpha} dx + \int_a^b \frac{\Psi(x)}{(x - x_0)^\alpha} dx = \\
&= \sum_{k=0}^{m} \frac{\varphi^{(k)}(x_0)}{k!(k + 1 - \alpha)} \left((b - x_0)^{k+1-\alpha} - (a - x_0)^{k+1-\alpha} \right) + \\
&\qquad + \int_a^b \frac{\Psi(x)}{(x - x_0)^\alpha} dx = I_1 + I_2.
\end{aligned}$$

Отже, інтеграл I_1 обчислюється аналітично, а інтеграл I_2 – наближено, наприклад, за допомогою квадратурних формул.

Попередні розрахунки:

$x_0 = 0$

$phi(x) = e^x$

$alpha = 0.5$

Записуємо ряд Тейлора до x^4:

$P_4(x) = 1 + x + (x^2)/(2!) + (x^3)/(3!) + (x^4)/(4!)$

$phi(x) \approx 1 + x + (x^2)/(2!) + (x^3)/(3!) + (x^4)/(4!) + psi(x)$

$\Rightarrow$

$psi(x) = e^x - (x^2)/(2) + (x^3)/(6) + (x^4)/(24)$

Тепер наш інтеграл може бути розписаний, як сума:

$I = I_1 + I_2 =$ (integral 0 to 1)$P_4(x)/sqrt(x)$ + (integral 0 to 1) $Psi(x)/sqrt(x)$

Тепер інтеграл I_1 можна шукати аналітично, I_2 використавши квадратурну формулу Сімпсона.

$I_1 = 2 + ⅔ + ⅕ + 1/21 + 1/108 = 2{,}9235449735.$

Також наближено можемо обчислити і перший інтеграл:

```
approximation_i1 = approximate_integral(phi, a, b)
approximation_i2 = approximate_integral(psi, a, b)

print(f"The approximate value of the integral I1 is: {(approximation_i1):.6f}")
print(f"The approximate value of the integral I2 is: {(approximation_i2):.6f}")
print(f"The approximate value of the integral I is: {(approximation_i1 + approximation_i2):.6f}")

The approximate value of the integral I1 is: 2.921549
The approximate value of the integral I2 is: 0.001759
The approximate value of the integral I is: 2.923308
```

Тоді необхідний нам результат аналітичного + наближеного інтегралів $= 2{,}9235449735 + 0{,}001759 =$ **2,9253039735**

Також перевіримо нашу програму з WolframAlpha:

Definite integral

$$\int_0^1 \frac{e^x}{\sqrt{x}}\, dx = 2\, e\, F(1) \approx 2.9253$$

Код програми:

```python
import math
def simpsons_formula(f, a, b):
    h = (b - a) / 2
    return (h / 3) * (f(a) + 4 * f((a + b) / 2) + f(b))

def approximate_integral(f, a, b, epsilon = 1e-4):
    n = 1
    previous_result = 0
    current_result = simpsons_formula(f, a, b)
    while abs(current_result - previous_result) > epsilon:
        n *= 2
        h = (b - a) / n
        previous_result = current_result
        current_result = 0
        for i in range(n):
            x0 = a + i * h
            x1 = a + (i + 1) * h
            current_result += simpsons_formula(f, x0, x1)

    return current_result

def g(x):
    return 1 + x + x**2 / 2 + x**3 / 6 + x**4 / 24
def phi(x):
    return (1 + x + x**2 / 2 + x**3 / 6 + x**4 / 24)/(x**(0.5))
def psi(x):
    return (math.exp(x) - g(x))/ math.sqrt(x)


a = 1e-6
b = 1
approximation_i1 = approximate_integral(phi, a, b)
approximation_i2 = approximate_integral(psi, a, b)
print(f"The approximate value of the integral I is: {(approximation_i1 + approximation_i2):.6f}")
```

```
The approximate value of the integral I is: 2.923308
```

Постановка задачі 36:

36. Обчислити інтеграл $I = \int\limits_1^\infty \frac{\arctan x}{1+x^3}dx$ з точністю 10^{-4}, користуючись формулою Сімпсона. Оцінити верхню межу інтегрування, для оцінки точності головної частини інтегралу застосувати принцип Рунге.

Теоретичні відомості:

Квадратурна формула Сімпсона:

Формула Сімпсона. Поклавши у формулі Ньютона-Котеса замкненого типу $n = 2$, отримуємо формулу парабол (Сімпсона)

$$\int\limits_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right),$$

з оцінкою залишкового члена

$$|R(f)| \leqslant \frac{M_4(b-a)^5}{2880}.$$

Складена формула Сімпсона з оцінкою залишкового члена має вигляд:

$$I \approx \frac{h}{6}\left(f(x_0) + 4\sum_{i=1}^{n} f(x_{i-\frac{1}{2}}) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right), \quad (14)$$

$$|R(f)| \leqslant \frac{M_4(b-a)h^4}{2880}.$$

Враховуючи зауваження про парні степені інтерполяції, алгебраїчний степінь точності квадратурної формули дорівнює 3. Порядок точності складеної формули – 4, а по одному проміжку – 5.

Принцип Рунге:

Апостеріорна оцінка похибки. Розглянемо деяку складену квадратурну формулу, яка має порядок точності p, з кроком h та $h/2$:

$I = I_h + ch^p + o(h^p),\ (I)$

$I = I_{\frac{h}{2}} + c\left(\frac{h}{2}\right)^p + o(h^p),\ (II)$

де ch^p – головний член похибки квадратурної формули, c не залежить від h.

$(I) - (II)\ :\ I_{\frac{h}{2}} - I_h \approx ch^p - c\left(\frac{h}{2}\right)^p;\ c\left(\frac{h}{2}\right)^p \approx \frac{(I_{\frac{h}{2}} - I_h)}{2^p - 1}$ – підставляємо в (II) і отримуємо *апостеріорну оцінку похибки* інтеграла I за допомогою наближення $I_{\frac{h}{2}}$ за *правилом Рунге*:

$$|I - I_{\frac{h}{2}}| \approx \frac{\left|I_{\frac{h}{2}} - I_h\right|}{2^p - 1}. \qquad (16)$$

Чисельне визначення порядка точності квадратурної формули. Якщо взяти

$$I = I_{\frac{h}{4}} + c\left(\frac{h}{4}\right)^p + o(h^p) \quad (III)$$

та розглянути (I) - (II), (II) - (III), отримаємо

$$I_{\frac{h}{2}} - I_h \approx \frac{ch^p(2p-1)}{2^p} \quad (IV),$$

$$I_{\frac{h}{4}} - I_{\frac{h}{2}} \approx \frac{ch^p(2p-1)}{2^{2p}} \quad (V).$$

Розділимо (IV) на (V):

$$\frac{I_{\frac{h}{2}} - I_h}{I_{\frac{h}{4}} - I_{\frac{h}{2}}} \approx 2^p.$$

Таким чином можна отримати формулу для *визначення порядка точності* квадратурної формули за допомогою правила Рунге:

$$p \approx \log_2\left|\frac{I_{\frac{h}{2}} - I_h}{I_{\frac{h}{4}} - I_{\frac{h}{2}}}\right|. \qquad (18)$$

Попередні розрахунки:

Приблизно оцінимо верхню межу інтегрування, перевіряючи чи |f(upper_limit)| > eps

1) f(1) = pi/8
2) f(2) = 0.12301652419934338
3) f(4) = 0.02039719482566204
4) …
5) …
6) f(64) = 5.932490027177402e-06
7) f(128) = 7.452884866935295e-07 < eps

Отже, upper_limit = 128;

Результат:

```
print("Approximated integral:", I_h_half)
print("Estimated error:", estimated_error)
print("Upper limit of integration:", b)

```

```
Approximated integral: 0.719026218439031
Estimated error: 5.9457440799252454e-06
Upper limit of integration: 128
```

Також перевіримо нашу програму з WolframAlpha:

Definite integral

$$\int_0^\infty \frac{\tan^{-1}(x)}{1+x^3}\,dx = \frac{1}{144}\left(4\pi\left(\sqrt{3}\,\pi + \log\left(\frac{97}{8} + 7\sqrt{3}\right)\right) - \psi^{(1)}\left(\frac{5}{12}\right) + \psi^{(1)}\left(\frac{11}{12}\right)\right) \approx 0.719073$$

Код програми:

```python
import numpy as np

def f(x):
    return np.arctan(x) / (1 + x**3)

def simpsons_rule(a, b, n):
    h = (b - a) / n
    x = np.linspace(a, b, n + 1)
    y = f(x)
    return h / 3 * (y[0] + 4 * np.sum(y[1:-1:2]) + 2 * np.sum(y[2:-2:2]) + y[-1])

def runge_estimation(I_h, I_h_half, n = 5):
    return np.abs(I_h_half - I_h) / (n ** 2 - 1)


precision = 1e-5
a = 0
b = 1  # initial upper limit
n = 10


I_h = simpsons_rule(a, b, n)
while True:
    b *= 2
    n *= 2
    I_h_half = simpsons_rule(a, b, n)
    estimated_error = runge_estimation(I_h, I_h_half)
    if estimated_error < precision :
        break
    I_h = I_h_half

print("Approximated integral:", I_h_half)
print("Estimated error:", estimated_error)
print("Upper limit of integration:", b)
```

```
Approximated integral: 0.719026218439031
Estimated error: 5.9457440799252454e-06
Upper limit of integration: 128
```