Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №4

Чисельні методи

Чисельне інтегрування функцій

Варіант 17

Виконала
студентка групи ІТ-91:

Луцай К. А.

Перевірила:

ас. Тимофєєва Ю. С.

Київ 2021

Мета: навчитися обчислювати заданий інтеграл за допомогою мови програмування Python, використати три методи (прямокутників, трапецій, Сімпсона) та методі уточнення цих значень (Рунге, Ромберта).

Теоретичні відомості:

При числовому інтегруванні по заданій підінтегральній функції будується сіткова функція, яка потім за допомогою формул локальної інтерполяції з контрольованою похибкою замінюється інтерполяційним поліномом.

Нехай на відрізку $x \in [a,b]$ задано неперервну функцію $y = f(x)$, і потрібно на цьому відрізку обчислити визначений інтеграл

$$I = \int_a^b f(x)dx .$$

Замінимо дану функцію на сіткову. Замість точного значення інтеграла I шукатимемо його наближене значення за допомогою суми $I \approx I_h = \sum_{i=0}^{n} A_i h_i$, де $h_i = x_i - x_{i-1}$, $i = \overline{1,n}$, $x_0 = a, x_n = b$, в якій необхідно визначити коефіцієнти A_i і похибку формули.

1. Метод прямокутників

Найбільш простою (і неточною) є формула прямокутників. Вона може бути отримана на основі визначення інтеграла, як границі послідовності інтегральних сум. Алгоритм запропонованого методу складається з таких основних кроків.

1) Весь відрізок [a, b] ділиться на n рівних частин із кроком h = (b−a)/n

2) Знаходиться значення yi підінтегральної функції f (x) у кожній частині, тобто yi = *f*(xi), I = 0,n

3) У кожній частині підінтегральна функція апроксимується інтерполяційним поліномом степеня n = 0, тобто прямою, що паралельна осі Ox. У результаті вся підінтегральна функція на ділянці [a, b] апроксимується ламаною лінією.

4) Для кожної частини визначається площа Si прямокутника.

5) Сумуються всі площі. Наближене значення інтеграла I дорівнює сумі площ прямокутників.

2. Метод трапецій

Розглянемо інтеграл $I = \int_a^b f(x)dx$ на відрізку $x \in [x_{i-1}, x_i]$ і на цьому відрізку обчислюватимемо його приблизно, замінюючи підінтегральну функцію інтерполяційним поліномом Лагранжа першого степеня. Отримаємо

$$\int_{x_{i-1}}^{x_i} f(x)dx = \int_{x_{i-1}}^{x_i} L_1(x)dx + R_i,$$

Де Ri – похибка яка підлягає визначенню.

Числове інтегрування за методом трапецій у разі заданої точності ε можна провести так:

1) визначається крок числового інтегрування h;

2) за допомогою цього кроку складається сіткова функція для підінтегральної функції f(x);

3) обчислюється наближене значення інтеграла за формулою $\int_a^b f(x)dx = \frac{h}{2}(y_0 + y_n + 2\sum_{i=1}^{n-1} y_i$, крок h в якій гарантує задану точність ε.

3. Метод Сімпсона

Розіб'ємо відрізок [a, b] на m пар відрізків b-a = nh = 2mh, $\frac{x_1}{x_2}$ і через кожні три вузли проведемо інтерполяційний поліном Лагранжа. Тоді:

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \int_{x_{i-1}}^{x_{i+1}} L_2(x)dx + R_i,$$

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \approx \int_{x_{i-1}}^{x_{i+1}} L_2(x)dx = \frac{h}{3}(y_{i-1} + 4y_i + y_{i+1}).$$

На всьому відрізку вираз необхідно скласти m разів, оскільки є m пар відрізків завдовжки h. Отримаємо *формулу Симпсона числового інтегрування*:

$$\int_a^b f(x)dx \approx \frac{h}{3}(y_0 + y_n + 4\sum_{i=1}^{m} y_{2i-1} + 2\sum_{i=1}^{m-1} y_{2i}).$$

Процедура Рунге дозволяє оцінити похибку і підвищити на одиницю

порядок методу шляхом багаторазового (у найпростішому разі дворазового) прорахунку з різними кроками. Нехай використовується будь-який метод чисельного інтегрування із кроками h і h/2. І нехай порядок обраного методу дорівнює p, тоді

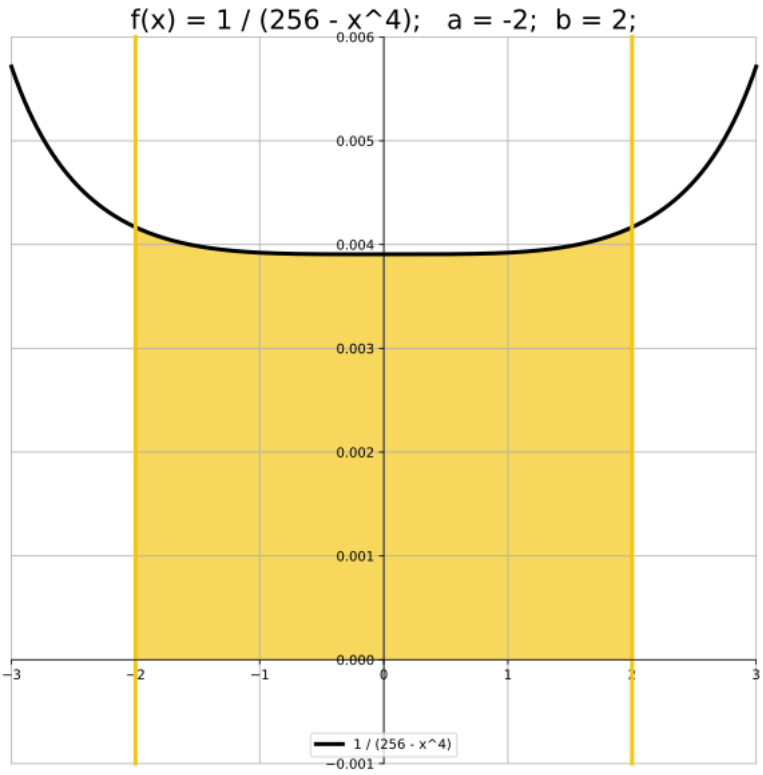
$$I = I_{\frac{h}{2}} + \frac{I_{\frac{h}{2}} - I_h}{2^p - 1} + O(h^{p+1}).$$

Розв'язок задачі в аналітичній формі:

$$f(x) = \frac{1}{256 - x^4}$$

a = -2; b = 2;

$$\int_{-2}^{2} \frac{1}{256 - x^4} dx = \frac{1}{128}(\ln 4 + 2\tan^{-1}\frac{1}{2})$$



Чорним показано сам графік функції;

Жовтим позначено межі та зафарбовано область, площу якої потрібно обчислити.

Лістинги програм розв'язування задачі:

В якості аргументів методи приймають початок та кінець інтегралу, і крок для уточнення.

```
def pram(a, b, h):
    m = int((b-a)/h)
    s = np.zeros(m)
    for i in range(m):
        x = a + i*h
        s[i] = s[i-1] + func17(x)*h
        #print(i,x, func17(x), s[i], s[i-1])
    return s[-1]

def trapez(a, b, h):
    def trap(a, b, old, k):
        if k == 1:
            new = (func17(a) + func17(b))*(b - a)/2.0
        else:
            n = 2**(k-2)
            h = (b-a)/n
            x = a + h/2.0
            s = 0
            for i in range(n):
                s += func17(x)
                x += h
                new = (old + h*s)/2.0
        return new

    m = int((b-a)/h)
    old = 0.0
    for k in range(1, int(math.log2(m) + 1)):
        new = trap(a, b, old, k)
        old = new
    return old

def simphson(a, b, h):
    m = int((b-a)/h)
    x = a
    s = 0
    se = 0
    so = 0
    for i in range(m):
        if i == 0 or i == m:
            s += func17(x)
        else:
            if i % 2 == 0:
                se += func17(x)
            else:
                so += func17(x)
        x += h
    return (h/3)*(s + 2*se + 4*so)
```

Задана функція та величини, які використано при розв’язанні:

```
def func17(x):
    return 1 / (256 - x**4)

aX = -2
bX = 2
h1 = 1.0
h2 = 0.5
```

Для уточнення та виклику основних функцій було розроблено метод Рунге, що приймає початок, кінець, крок 1, крок 2, кількість знаків після коми до якого потрібно провести уточнення:

```
def runge_p(a, b, h1, h2, n):
    print("1. Метод прямокутників")
    eps = 10**-n
    I1 = pram(a, b, h1)
    I2 = pram(a, b, h2)
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    delt = 1/3*math.fabs(I2 - I1)
    while delt > eps:
        #print( h1, h2, delt)
        h1 = h2
        h2 = h2/2.0
        I1 = pram(a, b, h1)
        I2 = pram(a, b, h2)
        delt = 1/3*math.fabs(I2 - I1)
    print("Точність:", eps, "Похибка:", delt, "за методом Рунге")
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    return round(I1, n), round(I2, n)

def runge_t(a, b, h1, h2, n):
    print("2. Метод трапецій")
    eps = 10**-n
    I1 = trapez(a, b, h1)
    I2 = trapez(a, b, h2)
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    delt = 1/3*math.fabs(I2 - I1)
    while delt > eps:
        #print( h1, h2, delt)
        h1 = h2
        h2 = h2/2.0
        I1 = trapez(a, b, h1)
        I2 = trapez(a, b, h2)
        delt = 1/3*math.fabs(I2 - I1)
    print("Точність:", eps, "Похибка:", delt, "за методом Рунге")
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    return round(I1, n), round(I2, n)
```

```
def runge_s(a, b, h1, h2, n):
    print("3. Метод Сімпсона")
    eps = 10**-n
    I1 = simphson(a, b, h1)
    I2 = simphson(a, b, h2)
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    delt = 1/15*math.fabs(I2 - I1)
    while delt > eps:
        #print( h1, h2, delt)
        h1 = h2
        h2 *= 0.5
        I1 = simphson(a, b, h1)
        I2 = simphson(a, b, h2)
        delt = 1/15*math.fabs(I2 - I1)
    print("Точність:", eps, "Похибка:", delt, "за методом Рунге")
    print("Значення", I1, "за кроком", h1)
    print("Значення", I2, "за кроком", h2)
    return round(I1, n), round(I2, n)
```

Окремо було розроблено метод екстраполяції Річардсона (Ромберга) для інтегралів з використанням матриці попередніх результатів обчислення. Функція приймає метод для обчислення інтегралу (один з трьох заданих за умовою), кінець, початок, похибку, розрядність матриці для обчислень (точність):

```
def romb(inter, a, b, h, m):
    eps = 10**-m
    print("Точність:", eps, "за методом Ромберга")
    r = np.array([[0] * (m+1)] * (m+1), float)
    h = b-a
    r[0, 0] = inter(a, b, h)
    power = 1
    for i in range(1, m + 1):
        h *= 0.5
        s = 0
        power *= 2
        for k in range(1, power, 2):
            s += func17(a + k*h)
        r[i, 0] = 0.5 * r[i-1, 0] + s*h
        power4 = 1
        for j in range(1, i+1):
            power4 *= 4
            r[i, j] = r[i, j-1] + (r[i, j-1] - r[i-1, j-1])/(power4 - 1)
    print("Значення", r[-1][-1], "за кроком", h)
    return r[-1][-1]
```

Виклик методів (одразу відбувається сортування отриманих значень за масивами даних, що будуть використані для подальшого їх порівняння на діаграмі)

```
H1 = []
H2 = []
H1r = []
H2r = []
H1rr = []
H2rr = []
H1.append(pram(aX, bX, h1))
H2.append(pram(aX, bX, h2))
i1, i2 = runge_p(aX, bX, h1, h2, 8)
H1r.append(i1)
H2r.append(i2)
H1rr.append(romb(pram, aX, bX, h1, 8))
H2rr.append(romb(pram, aX, bX, h2, 8))

H1.append(trapez(aX, bX, h1))
H2.append(trapez(aX, bX, h2))
i1, i2 = runge_t(aX, bX, h1, h2, 8)
H1r.append(i1)
H2r.append(i2)
H1rr.append(romb(trapez, aX, bX, h1, 8))
H2rr.append(romb(trapez, aX, bX, h2, 8))

H1.append(simphson(aX, bX, h1))
H2.append(simphson(aX, bX, h2))
i1, i2 = runge_s(aX, bX, h1, h2, 8)
H1r.append(i1)
H2r.append(i2)
H1rr.append(romb(simphson, aX, bX, h1, 8))
H2rr.append(romb(simphson, aX, bX, h2, 8))
```

Додаткове порівняння отриманих значень для всіх методів. Дані подано у вигляді chart bar діаграми з фіксованим розмахом значень. Для трьох методів (прямокутників, трапецій, Сімпсона) враховано такі значення:

- за заданими кроками h1 та h2;
- за вирахуваними кроками h1 та h2 методом Рунге;
- за вирахуваними кроками h1 та h2 методом Ромберга.

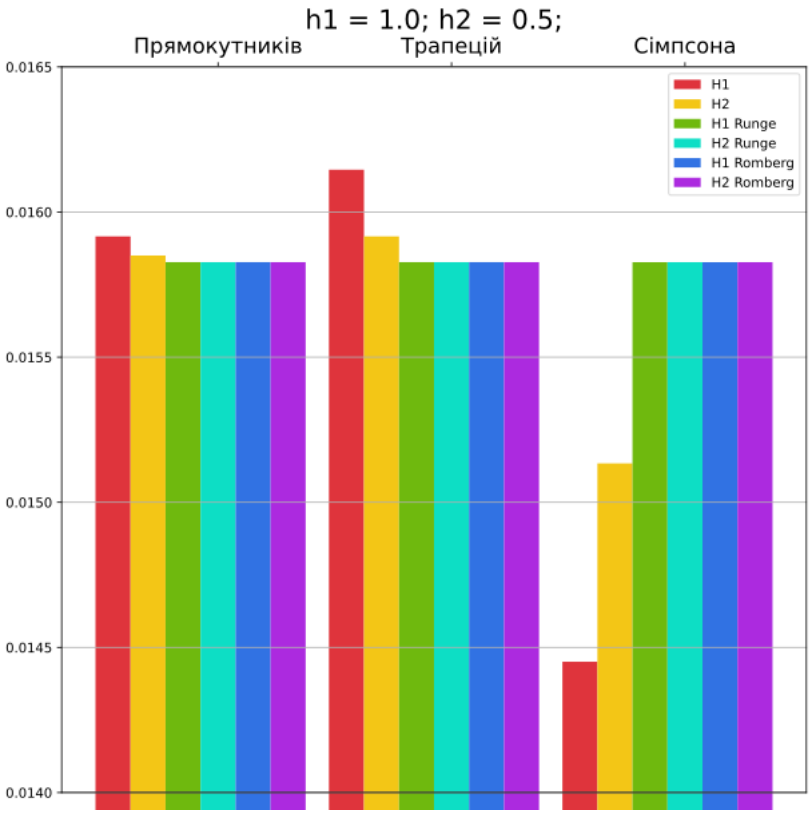
```
barWidth = 0.15
r1 = np.arange(len(H1))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
r4 = [x + barWidth for x in r3]
r5 = [x + barWidth for x in r4]
r6 = [x + barWidth for x in r5]
fig, ax = plt.subplots()
fig.set_size_inches(10, 10)
ax.grid(axis="y")
ax.set(ylim=[0.014, 0.0165])
ax.xaxis.tick_top()
plt.xticks([r + 3*barWidth for r in range(len(H1))],
           ["Прямокутників", "Трапецій", "Сімпсона"], fontsize=16)
```

```
plt.bar(r1, H1, color="#df343c", width=barWidth, label="H1")
plt.bar(r2, H2, color="#f3c616", width=barWidth, label="H2")
plt.bar(r3, H1r, color="#6fb90e", width=barWidth, label="H1 Runge")
plt.bar(r4, H2r, color="#0ddec5", width=barWidth, label="H2 Runge")
plt.bar(r5, H1r, color="#3172e3", width=barWidth, label="H1 Romberg")
plt.bar(r6, H2r, color="#ac2adf", width=barWidth, label="H2 Romberg")
plt.title("h1 = 1.0; h2 = 0.5;", fontsize=20)
plt.legend()
plt.show()
```

Додаткова побудова графіку функції

```
x = np.linspace(-3, 3, 1000)
X = np.linspace(-2, 2, 1000)
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(ylim=(-0.001, 0.006), xlim=(-3, 3))
ax.plot(x, 1 / (256 - x**4), color="black", label="1 / (256 - x^4)", lw=3)
plt.axvline(-2, color="#f3c616", lw=3)
plt.axvline(2, color="#f3c616", lw=3)
plt.fill_between(X, 1 / (256 - X**4), 0, color="#f3c616", alpha=.7)
ax.legend()
ax.grid(True)
ax.spines["left"].set_position("zero")
ax.spines["right"].set_color("none")
ax.spines["bottom"].set_position("zero")
ax.spines["top"].set_color("none")
plt.title("f(x) = 1 / (256 - x^4);    a = -2;  b = 2;", fontsize=20)
plt.show()
```

Результати виконання програм:



```
1. Метод прямокутників
Значення 0.015916053921568626 за кроком 1.0
Значення 0.01585028690783878 за кроком 0.5
Точність: 1e-08 Похибка: 5.651322682968344e-09 за методом Рунге
Значення 0.015827425001212463 за кроком 0.015625
Значення 0.015827408047244414 за кроком 0.0078125
Точність: 1e-08 за методом Ромберга
Значення 0.015827402395857202 за кроком 0.015625
Точність: 1e-08 за методом Ромберга
Значення 0.015827402395857202 за кроком 0.015625
2. Метод трапецій
Значення 0.01614583333333333 за кроком 1.0
Значення 0.015916053921568626 за кроком 0.5
Точність: 1e-08 Похибка: 5.651322681811862e-09 за методом Рунге
Значення 0.01582742500121247 за кроком 0.0078125
Значення 0.015827408047244425 за кроком 0.00390625
Точність: 1e-08 за методом Ромберга
Значення 0.015787744009829602 за кроком 0.015625
Точність: 1e-08 за методом Ромберга
Значення 0.015787744009829602 за кроком 0.015625
3. Метод Сімпсона
Значення 0.014450571895424836 за кроком 1.0
Значення 0.015133920125484389 за кроком 0.5
Точність: 1e-08 Похибка: 5.651403350385535e-09 за методом Рунге
Значення 0.015827232853756548 за кроком 0.0001220703125
Значення 0.015827317624806804 за кроком 6.103515625e-05
Точність: 1e-08 за методом Ромберга
Значення 0.015800963471838802 за кроком 0.015625
Точність: 1e-08 за методом Ромберга
Значення 0.015800963471838802 за кроком 0.015625
```

Висновки: було написано програму для чисельного інтегрування функцій на мові Python: було реалізовано методи прямокутників, трапецій, Сімпсона; було розроблено методи уточнення за Рунге та Ромбергом.