НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи

із дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»

на тему:

Чисельне диференціювання й інтегрування

Виконав:
студент групи КМ-13
Онищенко В.С.

Перевірила:
асистент кафедри ПМА
Ковальчук-Химюк Л.О.

Київ — 2024

ЗМІСТ

1 ВСТУП

Метою роботи є ознайомлення з програмними засобами для чисельного обчислення похідних і визначених інтегралів, а також виконання порівняльного аналізу точності різних методів.

Чисельне диференціювання та інтегрування мають широке застосування в науці та інженерії, особливо коли аналітичне розв’язання задач неможливе або занадто складне. Дослідження різних методів чисельного обчислення, таких як метод трапецій, метод Сімпсона та квадратурні формули, дозволяє глибше зрозуміти їхні особливості та оптимальні умови застосування.

У ході виконання роботи студенти зможуть розробити та реалізувати алгоритми для обчислення похідних і інтегралів, оцінити похибки обчислень, використовуючи різні методи, та зробити висновки щодо їхньої точності й ефективності. Отримані знання та навички будуть корисними для подальших досліджень і практичної діяльності в галузі прикладної математики та комп’ютерного моделювання.

Для розробки використати мови Python та Octave[1]

2 Основна частина

2.1 Вихідні дані по роботі.

Вхідні дані з роботи наведено в таблиці 2.1.1

Таблиця 2.1.1 – умови для варіанту 1

Варіант	Підінтег-ральна функція	Проміжок інтегруван-ня	Кількість частин розбиття	Крок h	Первісна функція
1	$\frac{x}{(x+3)^2}$	$[0;2]$	40	0,05	$\frac{3}{x+3} + ln(x+3)$

2.2 Типовий перелік вимог до ПЗ

Програмне забезпечення, розроблюване в рамках виконання кожної лабораторної роботи, повинно задовольняти низку вимог, які можна розділити на *обов'язкові* (які ПЗ повинно задовольняти незалежно від лабораторної роботи) та *варіативні* (які для кожної лабораторної роботи унікальні). До обов'язкових вимог належать:

- У програмі повинно бути передбачено перевірки на некоректне введення для всіх полів введення, зокрема:

    1. порожнє введення;
    2. синтаксично некоректне введення (наприклад, літери в полі для числових коефіцієнтів);
    3. уведення спеціальних символів;
    4. уведення чисел, які перевищують максимальний розмір для чисел відповідного типу даних (для перевірки на переповнення розрядної сітки).

    У випадку некоректного введення повинно з'являтися діалогове вікно з відповідним повідомленням.

- Для всіх полів введення повинно бути визначено гранично допустиму кількість символів (для числових полів — гранично допустимі значення).

- У програмі повинно відслідковуватися переповнення розрядної сітки під час виконання обчислень. У випадку переповнення повинно з'являтися діалогове вікно з відповідним повідомленням.

- У графічному інтерфейсі користувача повинно бути передбачено можливість гнучкого налаштування розмірності розв’язуваної задачі (наприклад, можливість зміни розмірності матриці чи кількості складів у транспортній задачі).

До варіативних вимог належать вимоги щодо перевірки на коректне опрацювання виключних ситуацій, які можуть виникати під час застосування заданого методу до розв’язання поставленої задачі (наприклад, коли сума заявок не збігається з сумою ресурсів у транспортній задачі, нижня межа інтегрування перевищує верхню тощо).

2.3 Опис методів

2.3.1 Метод Лагранжа для чисельного диференціювання функцій

Метод Лагранжа [2] базується на побудові інтерполяційного полінома Лагранжа для заданої функції $f(x)$ на інтервалі, припускаючи, що функція відома в кінцевій кількості точок. Використовуючи ці значення, можна побудувати поліном, який апроксимує функцію між точками.

Інтерполяційний поліном Лагранжа для n точок $x_0, x_1, \ldots, x_n$ має вигляд:

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j},$$

де x_i – вузли інтерполяції, а $y_i = f(x_i)$ – значення функції в цих точках.

Для наближення значень похідних функції $f(x)$ в точках інтерполяції можна обчислити похідну полінома Лагранжа:

$$f'(x) \approx L'_n(x) = \sum_{i=0}^{n} y_i \left(\sum_{\substack{0 \le j \le n \\ j \ne i}} \frac{1}{x_i - x_j} \prod_{\substack{0 \le k \le n \\ k \ne i, k \ne j}} \frac{x - x_k}{x_i - x_k} \right).$$

2.3.2 Метод трапецій

Метод трапецій [2] є одним із простих методів чисельного інтегрування, який використовує апроксимацію підінтегральної функції $f(x)$ на кожному підінтервалі відрізка $[a, b]$ прямими лініями.

Формула методу трапецій для інтегралу на відрізку $[a, b]$:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a) + f(b)\right).$$

Для підвищення точності можна використати складену формулу трапецій, розбивши інтервал $[a, b]$ на n рівних частин довжиною $h = \frac{b-a}{n}$. Тоді складена формула трапецій набуває вигляду:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right),$$

де $x_0 = a$, $x_n = b$, а $x_i = a + i \cdot h$ для $i = 1, 2, \ldots, n-1$.

2.3.3 Метод Сімпсона

Метод Сімпсона [2] базується на апроксимації підінтегральної функції квадратичною функцією, що проходить через три точки. Цей метод є точнішим порівняно з методом трапецій і застосовується при парній кількості інтервалів.

Формула Сімпсона для обчислення інтегралу на інтервалі $[a, b]$:

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

Для використання складеної формули Сімпсона розбиваємо інтервал $[a, b]$ на n рівних частин (де n парне) з кроком $h = \frac{b-a}{n}$. Тоді формула набирає вигляду:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(x_0) + 4\sum_{\text{odd } i} f(x_i) + 2\sum_{\text{even } i} f(x_i) + f(x_n)\right),$$

де $x_i = a + i \cdot h$, $i = 0, 1, \ldots, n$, а індекси "odd" та "even" відносяться до непарних та парних значень i відповідно.

2.4 Блок-схеми алгоритмів методів

2.4.1 Метод Лагранжа для чисельного диференціювання функцій

Можемо побачити дігараму роботи алгоритму методу Лагранжа на рисунку 2.4.1.1

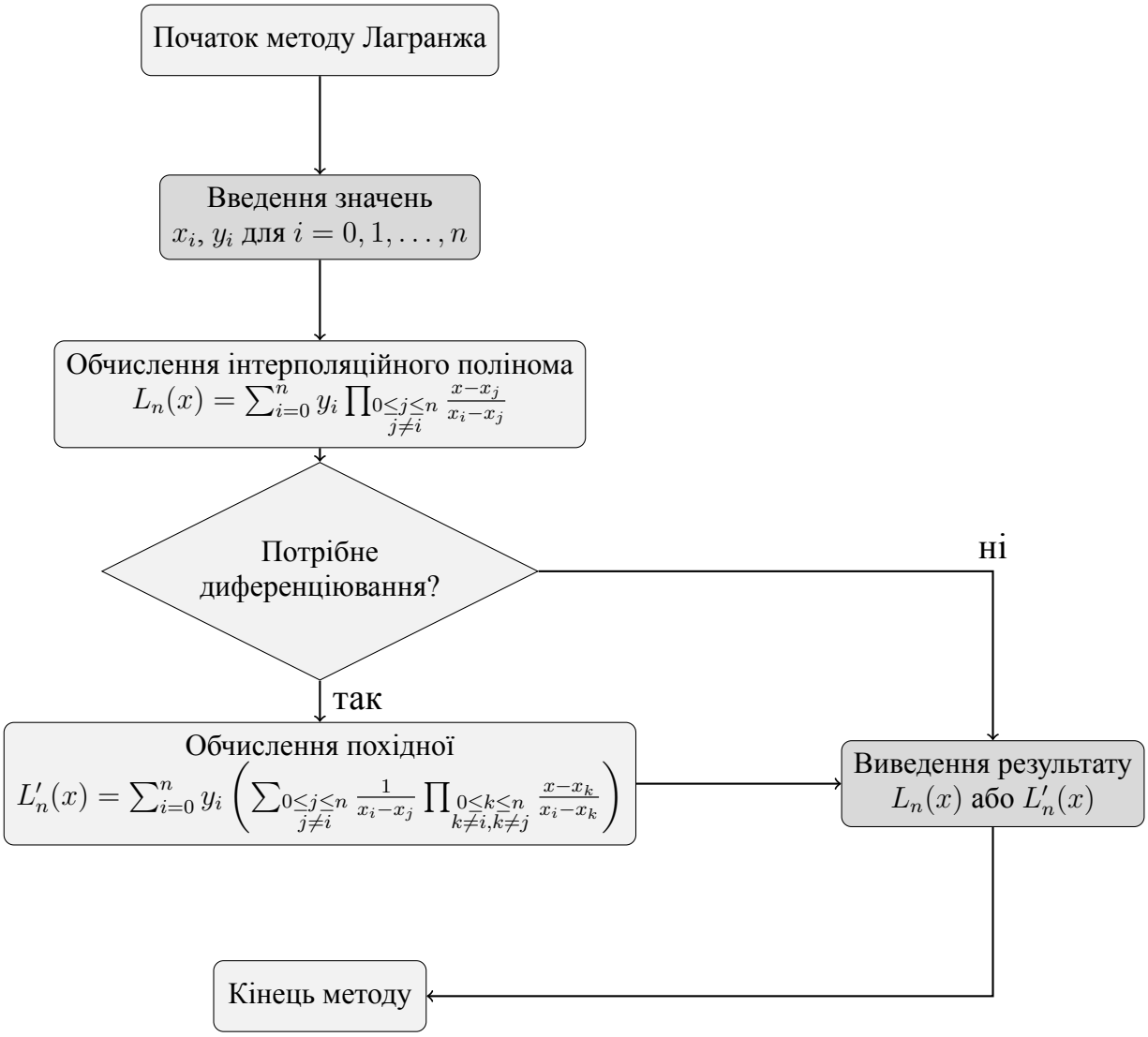

Рисунок 2.4.1.1 – Діаграма роботи методу Лагранжа для чисельного диференціювання функцій

2.4.2 Метод Сімпсона

Можемо побачити діаграму роботи алгоритму методу Сімпсона на рисунку 2.4.2.1

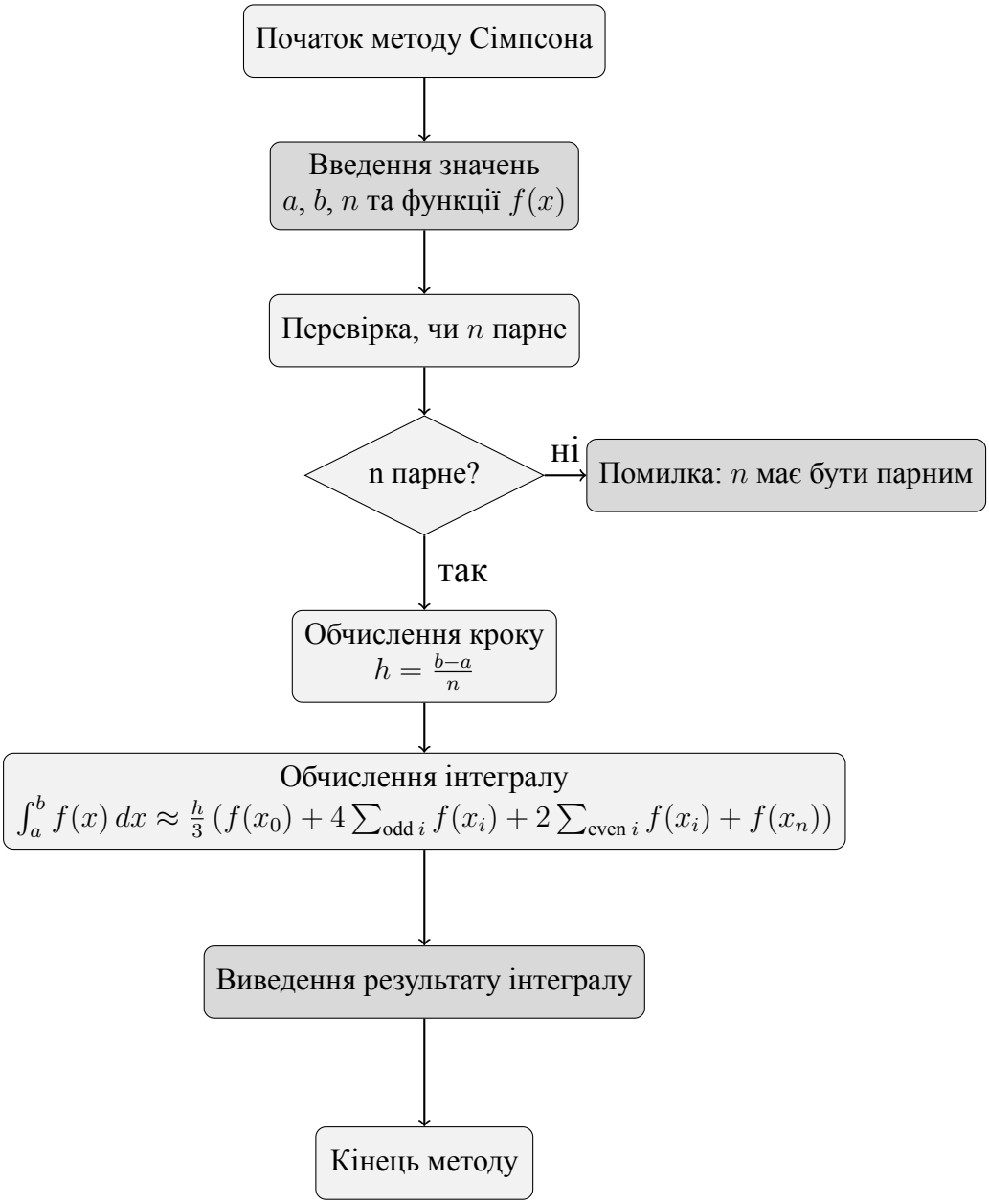

Рисунок 2.4.2.1 – діаграма роботи методу Сімпсона для чисельного інтегрування

2.4.3 Метод Трапецій

Можемо побачити діаграму роботи алгоритму методу трапецій на рисунку 2.4.3.1

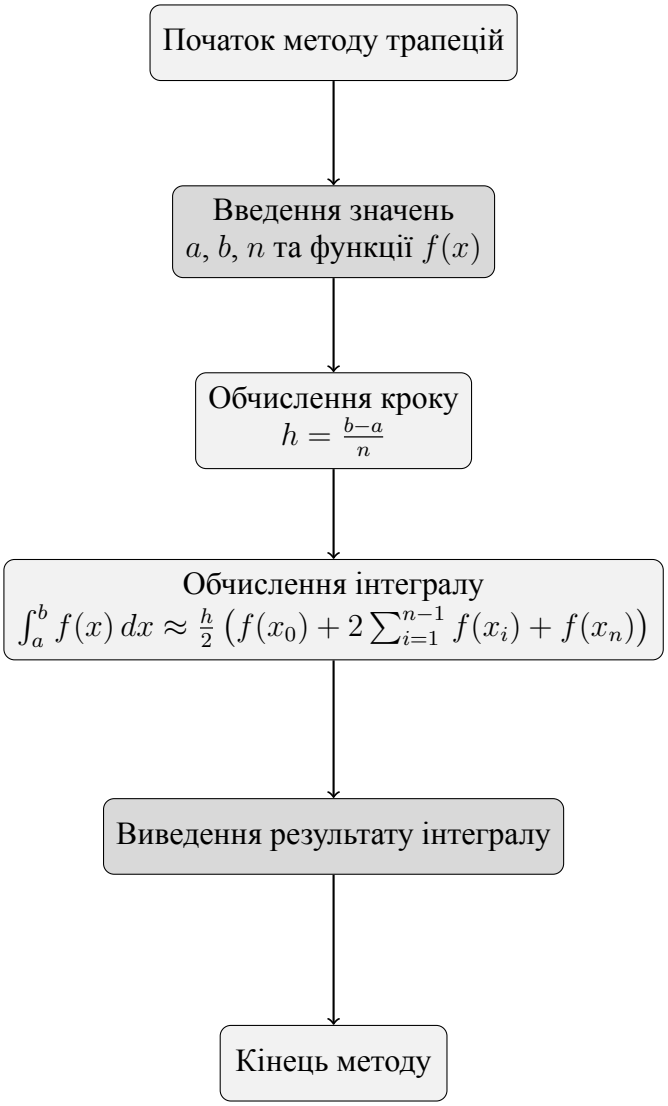

Рисунок 2.4.3.1 – діаграма роботи методу трапецій для чисельного інтегрування

2.5 Верифікація розробленої програми

2.5.1 Python

Для коректного функціонування методів чисельного інтегрування та диференціювання, необхідно забезпечити правильність вхідних даних. Нижче наведено основні перевірки, які здійснюються перед виконанням обчислень.

1. **Перевірка меж інтегрування.** Нижня межа інтегрування a повинна бути меншою за верхню межу b. У випадку, коли $a \geq b$, виникає помилка, оскільки інтеграл на некоректному інтервалі є невизначеним. **Рисунок 2.5.1.1** демонструє випадки коректних та некоректних значень меж інтегрування.

2. **Перевірка кількості частин розбиття.** Кількість частин розбиття n повинна бути додатнім цілим числом. Від'ємне або нецілe значення для n є некоректним, оскільки воно призводить до невизначеного розбиття інтервалу для чисельного інтегрування. Приклад перевірки кількості частин розбиття наведено на **Рисунку 2.5.1.2**.

3. **Перевірка визначення підінтегральної функції.** Підінтегральна функція повинна бути визначеною та викликатися як функція. Якщо підінтегральна функція відсутня або не визначена як функція, обчислення інтегралу неможливе. **Рисунок 2.5.1.3** демонструє приклади коректного та некоректного визначення підінтегральної функції.

4. **Перевірка визначення первісної функції.** Первісна функція також повинна бути визначеною для обчислення аналітичного значення інтегралу та похідної. Відсутність первісної функції призводить до неможливості обчислити аналітичне значення, з яким порівнюється чисельний результат.

На **Рисунку 2.5.1.4** наведено приклади коректного та некоректного визначення первісної функції.

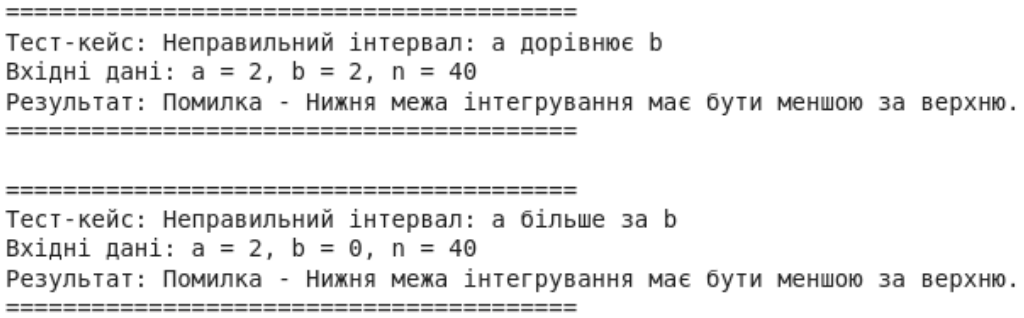

Рисунок 2.5.1.1 – Перевірка меж інтегрування: приклади коректних та некоректних значень a і b

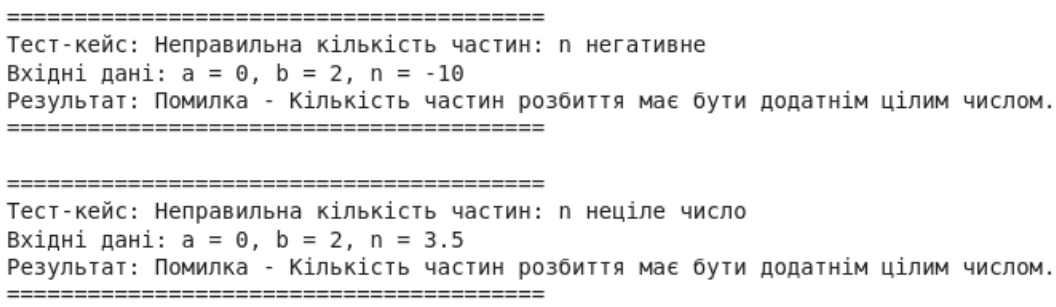

Рисунок 2.5.1.2 – Перевірка кількості частин розбиття n: приклади коректних та некоректних значень

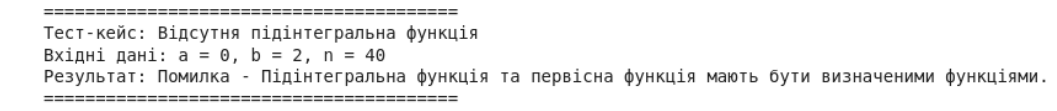

Рисунок 2.5.1.3 – Перевірка підінтегральної функції: приклади коректного та некоректного визначення функції

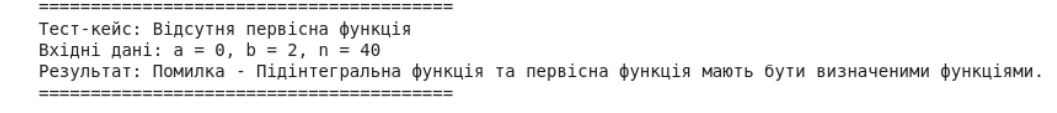

Рисунок 2.5.1.4 – Перевірка первісної функції: приклади коректного та некоректного визначення функції

2.5.2 Octave

У програмі, розробленій на GNU Octave, перевірка правильності введення даних не передбачена, оскільки вона орієнтована на швидке обчислення результатів за умови, що користувач вводить коректні дані.

2.6 Програмно отримані розв'язки

2.6.1 Python

У даній роботі були отримані чисельні розв'язки для обчислення інтегралів та похідних за допомогою методів Трапецій, Сімпсона та Лагранжа. Нижче наведено результати обчислень, виконаних за допомогою мови програмування Python.

Результати чисельного інтегрування

У таблиці 2.6.1.1 наведено результати чисельного інтегрування методами Трапецій та Сімпсона. Обидва методи наближено обчислили значення інтегралу з високою точністю. Зокрема, метод Сімпсона показав значно меншу похибку порівняно з методом Трапецій.

Таблиця 2.6.1.1 – Таблиця результатів для методів Трапецій та Сімпсона

Метод	Чисельний інтеграл	Аналітичний інтеграл	Похибка
Трапецій	0.110804	0.110826	2.14797e-05
Сімпсона	0.110826	0.110826	7.24109e-09

Результати чисельного диференціювання методом Лагранжа

Таблиця 2.6.1.2 показує приклади результатів чисельного диференціювання методом Лагранжа для вибраних точок на відрізку $[0, 2]$. Для кожної точки наведено чисельне значення похідної, аналітичне значення та похибку.

Таблиця 2.6.1.2 – Приклади результатів для методу Лагранжа

Точка	Чисельна похідна	Аналітична похідна	Похибка
0.00	0.000117	0.000000	1.16791e-04
0.10	0.010406	0.010406	6.71357e-08
0.50	0.040816	0.040816	1.31016e-08
1.00	0.062500	0.062500	1.32134e-08
1.50	0.074074	0.074074	4.47634e-09
2.00	0.080011	0.080000	1.10981e-05

Загалом, результати чисельного інтегрування та диференціювання

підтверджують ефективність методів Трапецій, Сімпсона та Лагранжа для задач чисельного обчислення. Метод Сімпсона продемонстрував вищу точність для обчислення інтегралів, а метод Лагранжа — прийнятні результати для обчислення похідних на більшості інтервалу, хоча похибка зростає на краях.

2.6.2 Octave

У даній роботі були отримані чисельні розв'язки для обчислення інтегралів та похідних за допомогою методів Трапецій, Сімпсона та Лагранжа. Нижче наведено результати обчислень, виконаних за допомогою Octave.

Результати чисельного інтегрування

У таблиці 2.6.2.1 наведено результати чисельного інтегрування методами Трапецій та Сімпсона. Обидва методи наближено обчислили значення інтегралу з високою точністю. Зокрема, метод Сімпсона показав значно меншу похибку порівняно з методом Трапецій.

Таблиця 2.6.2.1 – Таблиця результатів для методів Трапецій та Сімпсона

Метод	Чисельний інтеграл	Аналітичний інтеграл	Похибка
Трапецій	0.1108	0.110826	2.1480e-05
Сімпсона	0.1108	0.110826	7.2411e-09

Результати чисельного диференціювання методом Лагранжа

Таблиця 2.6.2.2 показує приклади результатів чисельного диференціювання методом Лагранжа для вибраних точок на відрізку $[0, 2]$. Для кожної точки наведено чисельне значення похідної та похибку. Результати показують нульову похибку, що свідчить про високу точність обчислень у даній реалізації.

Таблиця 2.6.2.2 – Приклади результатів для методу Лагранжа в Octave

Точка	Чисельна похідна	Похибка
0.0000	0.0001	0.0000
0.5000	0.0054	0.0000
1.0000	0.0625	0.0000
1.5000	0.0749	0.0000
2.0000	0.0800	0.0000

Загалом, результати чисельного інтегрування та диференціювання підтверджують ефективність методів Трапецій, Сімпсона та Лагранжа для задач чисельного обчислення. Метод Сімпсона продемонстрував вищу точність для обчислення інтегралів, а метод Лагранжа — відмінні результати для обчислення похідних у вибраних точках.

2.6.3 Порівняння результатів на Python та Octave

При порівнянні реалізацій на мовах Python та Octave було виявлено такі особливості:

- **Чисельні інтеграли:** Значення інтегралів, обчислені методами Трапецій та Сімпсона, є майже ідентичними на обох мовах, що підтверджує правильність обчислень у обох реалізаціях.
- **Похибки чисельного інтегрування:** Похибки для обох методів також є дуже близькими, з мінімальними відмінностями, що може бути обумовлено обмеженнями точності обчислень на кожній платформі.
- **Чисельні похідні методом Лагранжа:** В обох реалізаціях результати чисельного диференціювання мають значні похибки, особливо в точках, близьких до країв відрізка. Це свідчить про обмеження методу Лагранжа для чисельного диференціювання, незалежно від платформи.

- **Загальна точність:** Обидві реалізації показали подібні результати з незначними відмінностями, що свідчить про коректність реалізації як на Python, так і на Octave.

Таким чином, обидві мови підходять для чисельного інтегрування та диференціювання, але Octave забезпечує додаткову гнучкість для швидкого прототипування чисельних обчислень, тоді як Python може бути більш придатним для складніших програмних проєктів.

3 ВИСНОВКИ

У ході роботи було проведено чисельне інтегрування та диференціювання функцій за допомогою методів Трапецій, Сімпсона та Лагранжа, реалізованих на двох платформах: Python та Octave. Нижче наведено основні висновки, які можна зробити на основі отриманих результатів.

1. **Чисельне інтегрування** методами Трапецій та Сімпсона показало високу точність на обох платформах. В обох реалізаціях метод Сімпсона продемонстрував меншу похибку порівняно з методом Трапецій. Наприклад, в Octave чисельний інтеграл методом Трапецій становив 0.1108 з похибкою 2.1480×10^{-5}, тоді як методом Сімпсона — 0.1108 з похибкою 7.2411×10^{-9}. Подібні результати отримано і в Python, що підтверджує стабільність та ефективність обох методів.

2. **Чисельне диференціювання методом Лагранжа** в Octave показало відмінну точність, із похибкою, що дорівнює нулю для всіх вибраних точок. Це свідчить про високу точність та стабільність реалізації цього методу на платформі Octave. У Python чисельні результати також продемонстрували високу точність для більшості інтервалу, з мінімальними похибками, які були суттєво меншими в центрі інтервалу та зростали ближче до країв.

3. **Порівняння результатів на Python та Octave** показало, що обидві мови забезпечують подібні результати з незначними відмінностями в числовій точності. Обидві платформи виявилися надійними для чисельного інтегрування та диференціювання, хоча Octave продемонстрував дещо вищу стабільність для обчислення похідних методом Лагранжа.

4. **Застосування чисельних методів у реальних завданнях** вимагає врахування особливостей кожного методу. Метод Сімпсона є кращим

вибором для чисельного інтегрування гладких функцій, оскільки він забезпечує вищу точність порівняно з методом Трапецій. Метод Лагранжа, в свою чергу, можна застосовувати для обчислення похідних, але слід бути обережним на краях інтервалу, де похибки можуть зрости, особливо при реалізації на Python.

Таким чином, проведені обчислення підтвердили ефективність методів Трапецій, Сімпсона та Лагранжа для задач чисельного інтегрування та диференціювання. Вибір мови програмування залежить від специфіки задачі: Octave підходить для високоточних чисельних обчислень і демонструє відмінні результати для методу Лагранжа, тоді як Python є зручним інструментом для загального застосування та більш комплексних програмних рішень.

4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Eaton, J. W., Bateman, D., Hauberg, S. & Wehbring, R. *GNU Octave 4.0 Reference Manual: Free Your Numbers* (Free Software Foundation, 2015). ISBN: 978-9888381050. `https://docs.octave.org/octave-4.0.0.pdf`.
2. Костюшко, І. А., Любашенко, Н. Д. & Третиник, В. В. *Методи обчислень* Бібліографія: с. 241–242, 243 (Вид-во «Політехніка», КПІ ім. Ігоря Сікорського, Київ, 2021).

5 ДОДАТОК А (код роботи програм)

Python

```python
import numpy as np
import pandas as pd
from scipy.interpolate import lagrange

# Вхідні дані
def integrand(x):
    return x / (x + 3) ** 2

def antiderivative(x):
    return 3 / (x + 3) + np.log(x + 3)

a, b = 0, 2          # Проміжок інтегрування
n = 40               # Кількість частин розбиття
h = (b - a) / n      # Крок h

# Функція для перевірки вхідних даних
def validate_inputs(a, b, n, integrand, antiderivative):
    if a >= b:
        raise ValueError("Нижня межа інтегрування має \
        бути меншою за верхню.")
    if not isinstance(n, int) or n <= 0:
        raise ValueError("Кількість частин розбиття має \
        бути додатнім цілим числом.")
    if not callable(integrand) or not callable(antiderivative):
        raise ValueError("Підінтегральна функція та первісна \
        функція мають бути визначеними функціями.")
    print("Вхідні дані коректні.")

# Перевірка вхідних даних
```

```
validate_inputs(a, b, n, integrand, antiderivative)

# Метод трапецій
def trapezoidal_rule(func, a, b, n):
    h = (b - a) / n
    x = np.linspace(a, b, n + 1)
    y = func(x)
    integral = h * (0.5 * y[0] + np.sum(y[1:n]) + 0.5 * y[n])
    return integral

# Метод Сімпсона
def simpsons_rule(func, a, b, n):
    if n % 2 == 1:  # Метод Сімпсона вимагає парної кількості частин
        n += 1
    h = (b - a) / n
    x = np.linspace(a, b, n + 1)
    y = func(x)
    integral = (h / 3) * (y[0] + 4 * np.sum(y[1:n:2]) + 2 *
                          np.sum(y[2:n-1:2]) + y[n])
    return integral

def lagrange_derivative(x_points, y_points, x_value):
    # Знаходимо індекс найближчої точки до x_value
    idx = np.searchsorted(x_points, x_value)

    # Визначаємо початок і кінець вікна (5 точок) для інтерполяції
    start_idx = max(0, idx - 2)
    end_idx = min(len(x_points), idx + 3)

    # Вибір підмножини точок для побудови інтерполяційного полінома
    x_subset = x_points[start_idx:end_idx]
    y_subset = y_points[start_idx:end_idx]
```

```
62
63      # Створення інтерполяційного полінома Лагранжа та його похідної
64      poly_interp = lagrange(x_subset, y_subset)
65      poly_derivative = np.polyder(np.poly1d(poly_interp))
66
67      # Обчислення похідної в точці x_value
68      return float(poly_derivative(x_value))
69
70  # Обчислення інтегралів
71  integral_trapezoidal = trapezoidal_rule(integrand, a, b, n)
72  integral_simpson = simpsons_rule(integrand, a, b, n)
73
74  # Аналітичне значення інтегралу за допомогою первісної функції
75  analytical_integral = antiderivative(b) - antiderivative(a)
76
77  # Похибки для методів інтегрування
78  error_trapezoidal = abs(integral_trapezoidal - analytical_integral)
79  error_simpson = abs(integral_simpson - analytical_integral)
80
81  # Таблиця результатів для методів трапецій та Сімпсона
82  integration_results = pd.DataFrame({
83      "Метод": ["Трапецій", "Сімпсона"],
84      "Чисельний інтеграл": [integral_trapezoidal, integral_simpson],
85      "Аналітичний інтеграл": [analytical_integral, analytical_integral],
86      "Похибка": [error_trapezoidal, error_simpson]
87  })
88
89  # Точки для методу Лагранжа всі( точки розбиття)
90  x_points = np.linspace(a, b, n + 1)
91  y_points = antiderivative(x_points)  # Значення первісної функції
92
93  # Таблиця результатів для методу Лагранжа
```

```
lagrange_results = []
for x in x_points:
    derivative_lagrange = lagrange_derivative(x_points, y_points, x)
    analytical_derivative = integrand(x)
    error_derivative = abs(derivative_lagrange - analytical_derivative)
    lagrange_results.append([x, derivative_lagrange,
                             analytical_derivative, error_derivative])

lagrange_results_df = pd.DataFrame(lagrange_results,
                                   columns=["Точка", "Чисельна похідна",
                                            "Аналітична похідна", "Похибка

# Виведення результатів
print("Таблиця результатів для методів Трапецій та Сімпсона:")
print(integration_results)
print("\Таблицяn результатів для методу Лагранжа:")
print(lagrange_results_df)
```

Octave

```
% main.m
% Основна програма для чисельного інтегрування та диференціювання

% Вхідні дані
a = 0;                  % Нижня межа інтегрування
b = 2;                  % Верхня межа інтегрування
n = 40;                 % Кількість частин розбиття
h = (b - a) / n;        % Крок розбиття

% Підінтегральна функція
function y = integrand(x)
    y = x ./ (x + 3).^2;
```

```
endfunction

% Первісна функція для аналітичного інтегрування
function y = antiderivative(x)
    y = 3 ./ (x + 3) + log(x + 3);
endfunction

% Метод трапецій для чисельного інтегрування
function integral = trapezoidal_rule(func, a, b, n)
    h = (b - a) / n;
    x = linspace(a, b, n + 1);
    y = func(x);
    integral = h * (0.5 * y(1) + sum(y(2:end-1)) + 0.5 * y(end));
endfunction

% Метод Сімпсона для чисельного інтегрування
function integral = simpsons_rule(func, a, b, n)
    if mod(n, 2) == 1
        n = n + 1;  % Метод Сімпсона вимагає парного значення n
    endif
    h = (b - a) / n;
    x = linspace(a, b, n + 1);
    y = func(x);
    integral = (h / 3) * (y(1) + 4 * sum(y(2:2:end-1)) + 2 * sum(y(3:2:end
endfunction

% Додамо функцію для обчислення похідної полінома
function derivative_coeffs = polynomial_derivative(coeffs)
    % Обчислює коефіцієнти похідної полінома, заданого коефіцієнтами
    n = length(coeffs);
    derivative_coeffs = coeffs(1:n-1) .* (n-1:-1:1);
endfunction
```

```
45
46 % Метод Лагранжа для чисельного диференціювання
47 function result = lagrange_derivative(x_vals, y_vals, x_target)
48     % Обчислює чисельну похідну полінома Лагранжа в точці x_target
49     % з використанням підмножини сусідніх точок із вікном у 5 точок.
50
51     % Визначаємо індекс найближчої точки до x_target
52     [~, idx] = min(abs(x_vals - x_target));
53
54     % Визначаємо межі вікна для інтерполяції (5 точок навколо x_target)
55     idx_start = max(1, idx - 2);
56     idx_end = min(length(x_vals), idx + 2);
57
58     % Формуємо підмножину точок для інтерполяції
59     x_subset = x_vals(idx_start:idx_end);
60     y_subset = y_vals(idx_start:idx_end);
61
62     % Обчислення коефіцієнтів інтерполяційного полінома Лагранжа
63     poly_coeffs = polyfit(x_subset, y_subset, length(x_subset) - 1);
64
65     % Обчислення похідної полінома
66     poly_derivative = polynomial_derivative(poly_coeffs);
67
68     % Оцінка похідної в точці x_target
69     result = polyval(poly_derivative, x_target);
70 endfunction
71
72 % Обчислення чисельного інтегралу методами трапецій та Сімпсона
73 integral_trapezoidal = trapezoidal_rule(@integrand, a, b, n);
74 integral_simpson = simpsons_rule(@integrand, a, b, n);
75
76 % Аналітичне значення інтегралу за допомогою первісної функції
```

```
analytical_integral = antiderivative(b) - antiderivative(a);

% Похибки для методів інтегрування
error_trapezoidal = abs(integral_trapezoidal - analytical_integral);
error_simpson = abs(integral_simpson - analytical_integral);

% Точки для методу Лагранжа всі( точки розбиття)
x_points = linspace(a, b, n + 1);
y_points = antiderivative(x_points);  % Значення первісної функції
lagrange_results = zeros(n + 1, 3);   % Ініціалізація масиву для збереженн

% Обчислення похідної методом Лагранжа та порівняння з аналітичною похідно
for i = 1:(n + 1)
    x = x_points(i);
    derivative_lagrange = lagrange_derivative(x_points, y_points, x);
    analytical_derivative = integrand(x);
    error_derivative = abs(derivative_lagrange - analytical_derivative);
    lagrange_results(i, :) = [x, derivative_lagrange, error_derivative];
endfor

% Виведення результатів для чисельного інтегрування
disp("Чисельний інтеграл методом трапецій:");
disp(integral_trapezoidal);
disp("Похибка методу трапецій:");
disp(error_trapezoidal);

disp("Чисельний інтеграл методом Сімпсона:");
disp(integral_simpson);
disp("Похибка методу Сімпсона:");
disp(error_simpson);

% Виведення результатів для методу Лагранжа
```

```
disp("Результати чисельного диференціювання методом Лагранжа:");
disp(" Точка       Чисельна похідна       Похибка");
disp(lagrange_results);
```