Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

****

**ЗВІТ**

**Про виконання лабораторної роботи № 10**

«Чисельні методи інтегрування»

**з дисципліни «Чисельні методи»**

**Лектор:**

Мельник Н.Б.

**Виконав:**

студент групи ПЗ-11

Солтисюк Д.А.

**Прийняла:**

Мельник Н.Б.

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема роботи:** Чисельні методи інтегрування.

**Мета роботи:** Ознайомлення на практиці з методами чисельного інтегрування.

**Теоретичні відомості**

**Метод прямокутників**

Найпростішим методом наближеного обчислення інтеграла є метод прямокутників, суть якого зводиться до знаходження означеного інтеграла як суми площ n прямокутників висотою та основою , отриманих шляхом розбиття відрізка інтегрування [a,b] на n рівних частин.

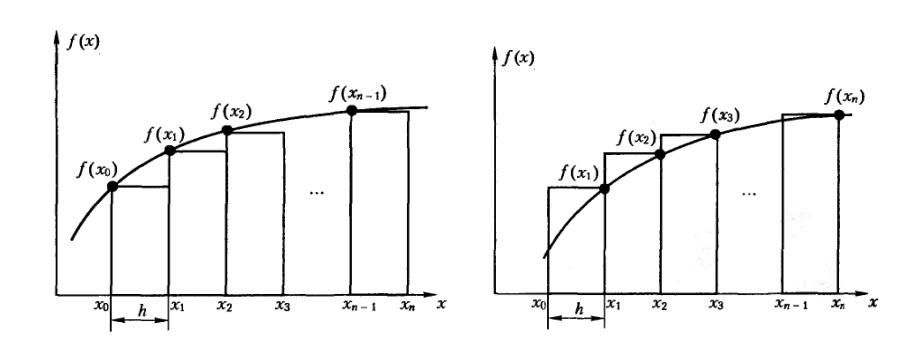


Рис.1 Метод лівих прямокутників Рис.2 Метод правих прямокутників

Формула лівих прямокутників:

Формула правих прямокутників:

Де

Однак ці методи не є дуже точними, оскільки в першому є нестача, а в другому надлишок інтегралу. Тому існує метод середніх прямокутників.

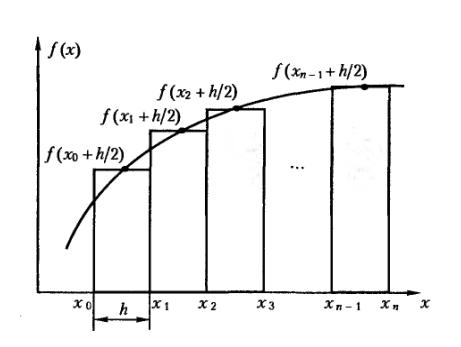


Рис.3 Метод середніх прямокутників.

Формула середніх прямокутників:

**Метод трапецій**

Метод трапецій полягає в тому, що відрізок інтегрування [а,b] розбивають на n рівних відрізків, а криву, описану підінтегральну функцією f (x), замінюють на кожному із цих відрізків кусково-лінійною функцією j(x), отриманою стягуванням хорд, які проходять через точки .

Значення інтеграла знаходять як суму площ прямокутних трапецій з висотою .

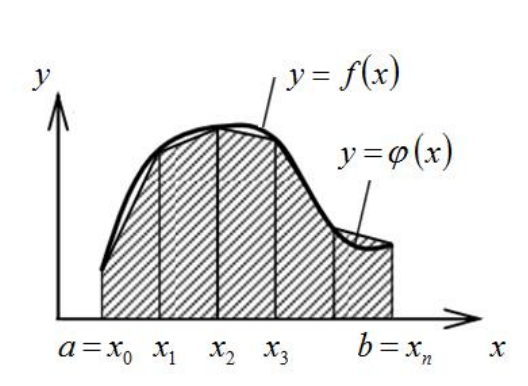


Рис.4 метод трапецій

Площу кожної i-ої елементарної трапеції визначають за формулою

Відповідно отримуємо таку формулу трапецій для обчислення інтегралу:

**Метод Сімпсона**

Даний метод полягає в тому, що криву, описану підінтегральною функцією f (x) , на елементарних відрізках заміняють параболою.

Поділимо відрізок інтегрування [a,b] на парну кількість n рівних частин з кроком

. На кожному елементарному відрізку підінтегральну функцію f (x) замінимо інтерполяційним поліномом другого степеня (квадратичною параболою). Тоді обчислення означеного інтеграла зводиться до обчислення суми площ , (i =1,n) криволінійних трапецій.

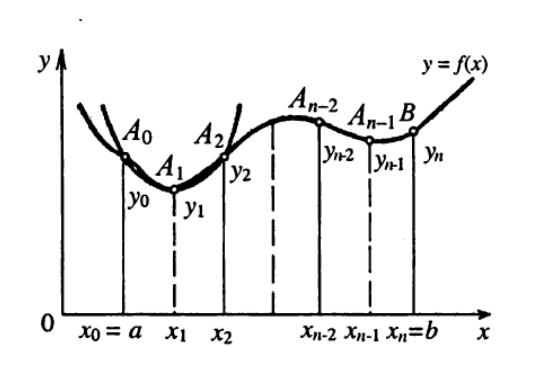


Рис.5 Метод Сімпсона

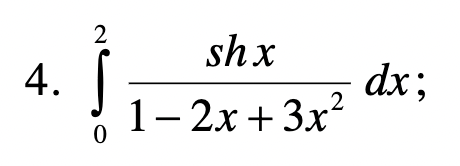
Площу кожної елементарної криволінійної трапеції визначають за формулою Сімпсона

Формула Сімпсона:

**Індивідуальне завдання**

Скласти програму чисельного інтегрування у відповідності до варіанту:

1. Методом лівих, правих та центральних прямокутників;
2. Методом трапецій;
3. Методом Сімпсона

**Варіант 4**

**Результат виконання роботи**

rectangles.py

**from** typing **import** Callable, Dict, Literal, Tuple

**import** numpy **as** np

**from** common.main **import** IntegrationOrientedMethod

RectangleMethodType = Literal["right", "left", "centered", "trapezium"]

**class** RectangleMethod(IntegrationOrientedMethod):

method\_mappings: Dict[RectangleMethodType, Tuple[Callable, list[int]]]

method\_type: RectangleMethodType

**def** \_\_init\_\_(self, method\_type: RectangleMethodType, f: Callable, a: float,

b: float, eps: float) -> **None**:

super().\_\_init\_\_(f"{method\_type.capitalize()} rectangles", f, a, b,

eps)

self.method\_type = method\_type

self.method\_mappings = {

"left": (**lambda** i: self.f\_xi(i), list(range(self.n - 1))),

"right": (**lambda** i: self.f\_xi(i), list(range(1, self.n))),

"centered":

(**lambda** i: self.f\_xi(i + 0.5), list(range(self.n - 1))),

"trapezium": (**lambda** i: (self.f\_xi(i) + self.f\_xi(i + 1)) / 2,

list(range(self.n - 1)))

}

**def** execute\_method(self):

**return** self.h \* np.sum([

self.method\_mappings[self.method\_type][0](i)

**for** i **in** self.method\_mappings[self.method\_type][1]

])

simpson.py

**from** typing **import** Callable

**import** numpy **as** np

**from** common.main **import** IntegrationOrientedMethod

**class** SimpsonMethod(IntegrationOrientedMethod):

**def** \_\_init\_\_(self, f: Callable, a: float, b: float, eps: float) -> **None**:

super().\_\_init\_\_("Simpson", f, a, b, eps)

**def** execute\_method(self):

**return** (self.h / 3) \* (

self.f\_xi(0) + self.f\_xi(self.n) +

4 \* np.sum([self.f\_xi(i - 1) **for** i **in** range(1, self.n) **if** i % 2]) +

2 \* np.sum([self.f\_xi(i) **for** i **in** range(1, self.n - 1) **if** i % 2]))

main.py

**from** math **import** sinh

**from** typing **import** List

**from** common.rectangles **import** RectangleMethod, RectangleMethodType

**from** common.simpson **import** SimpsonMethod

**if** \_\_name\_\_ == "\_\_main\_\_":

**def** f(x):

**return** sinh(x) / (1 - 2 \* x + pow(3 \* x, 2))

a = 0

b = 2

eps = 0.01

rectangle\_methods: List[RectangleMethodType] = [

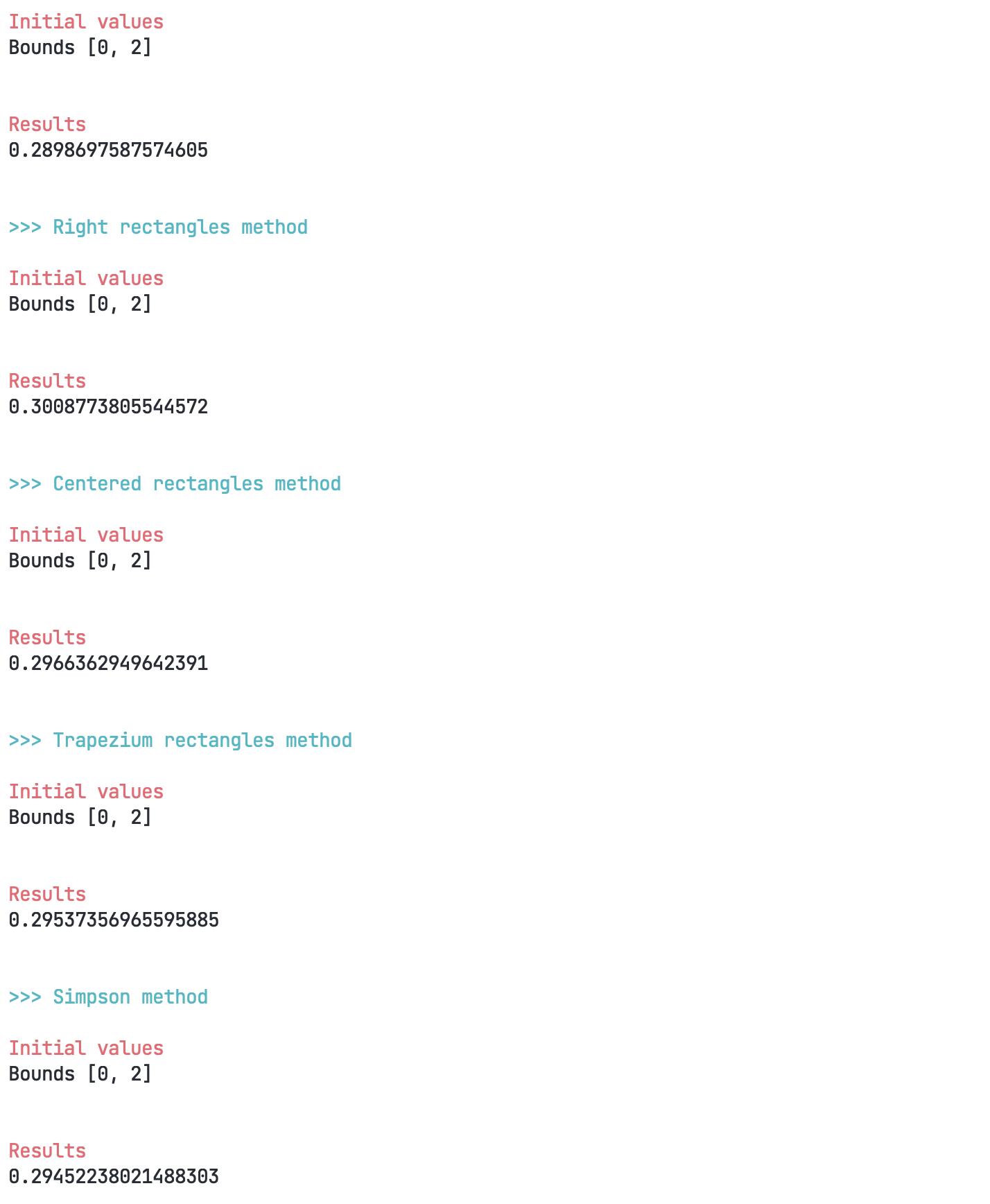
"left", "right", "centered", "trapezium"

]

**for** rectangle\_method **in** rectangle\_methods:

RectangleMethod(rectangle\_method, f, a, b, eps).compile()

SimpsonMethod(f, a, b, eps).compile()

**Результат виконання програми**

**Висновки**

Виконуючи лабораторну роботу №10, я ознайомився на практиці з методами чисельного інтегрування.