Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до виконаної лабораторної роботи № 7

з дисципліни

“Чисельні методи”

на тему:

***«Наближене обчислення визначених інтегралів»***

Виконав

студент групи *ОІ-11 сп*

*Вальчевський П. В.*

Викладач:

*Сенета М. Я.*

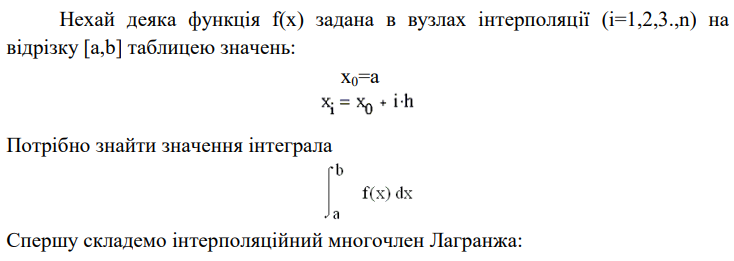
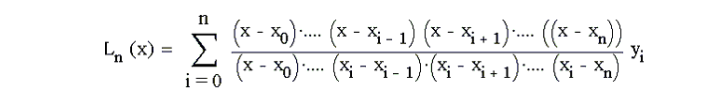
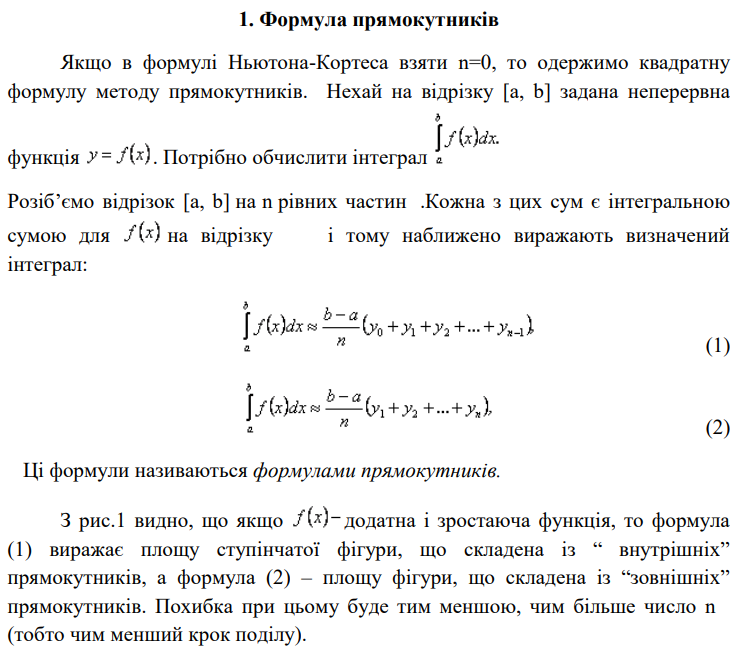
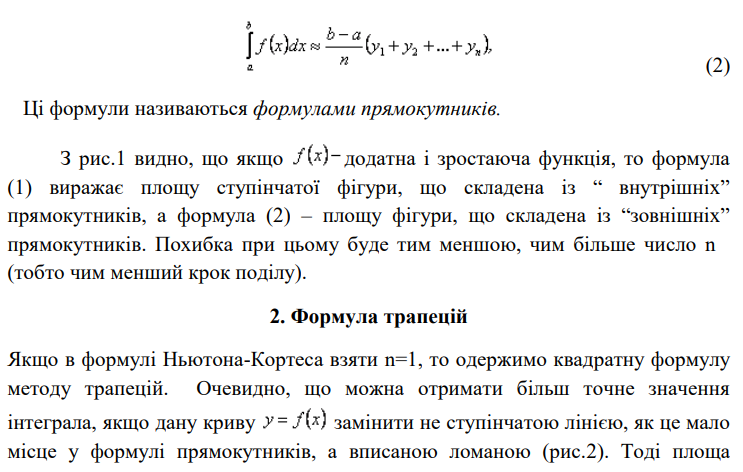
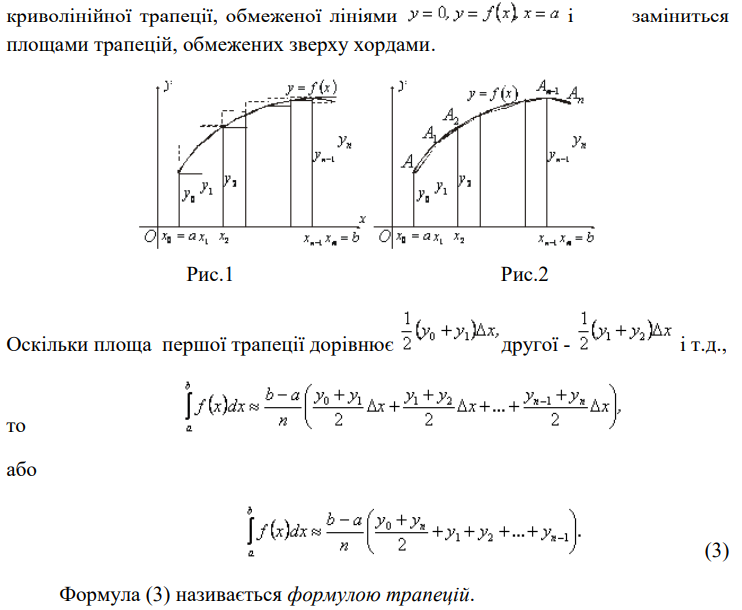
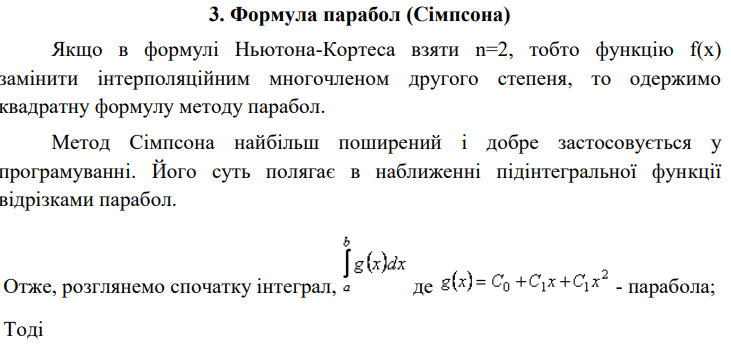
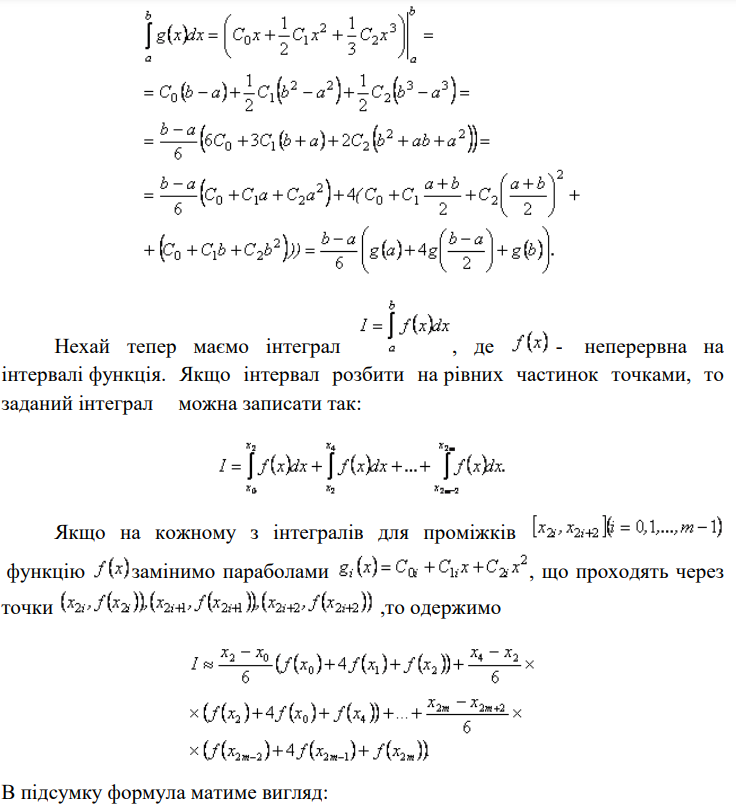
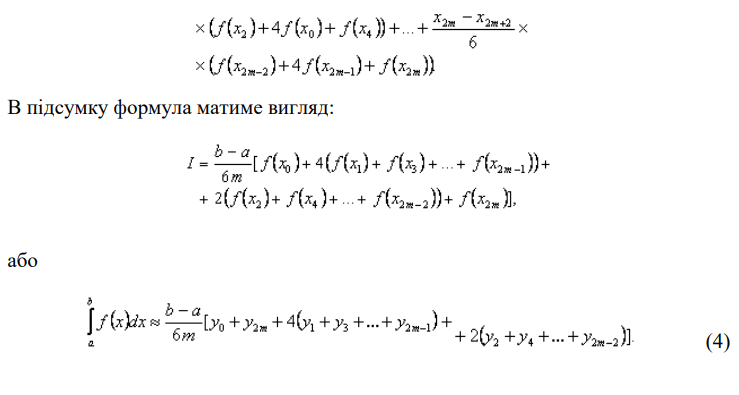
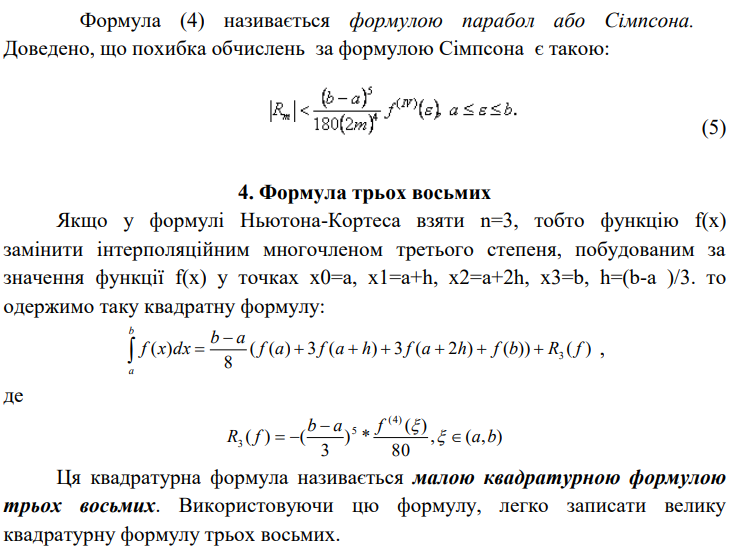
Львів – 2023

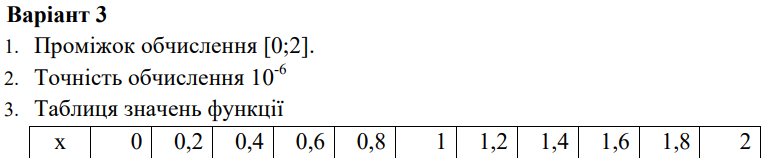
## Лабораторна робота № 7

*Тема роботи:*  **Наближене обчислення визначених інтегралів.**

*Мета роботи:* ***засвоїти методи наближених обчислень визначених інтегралів.***

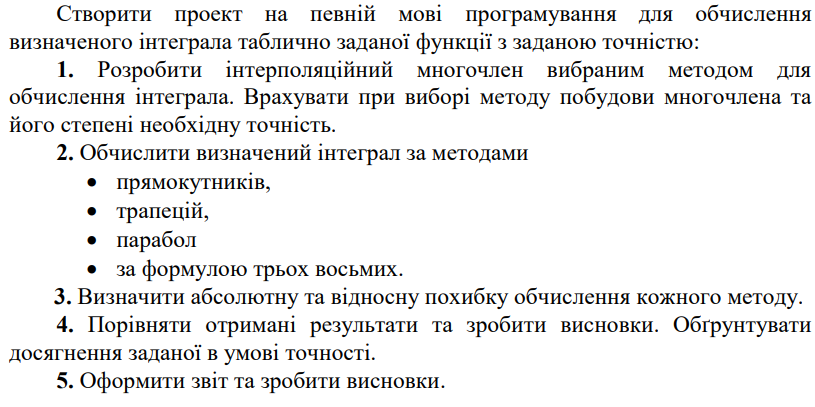
***Порядок виконання роботи***

1. **Основні теоретичні відомості**
2. ****
3. ****
4. ****
5. ****
6. ****
7. ****
8. ****
9. ****
10. ****
11. **Номер варіанту – 3.**

****

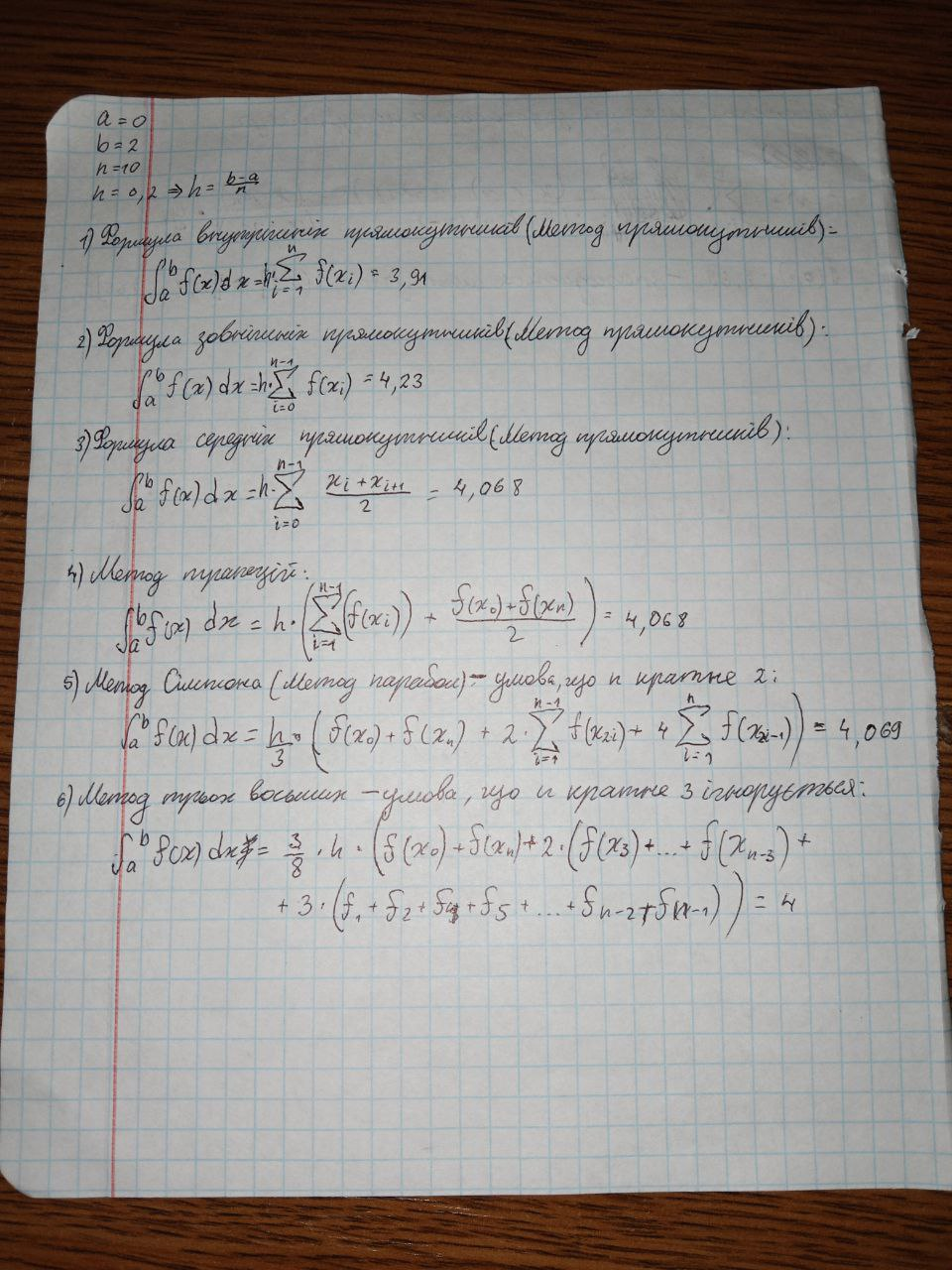
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0,2 | 0,4 | 0,6 | 0,8 | 1 | 1,2 | 1,4 | 1,6 | 1,8 | 2 |
| Y | 2,76 | 2,65 | 2,53 | 2,39 | 2,24 | 2,08 | 1,91 | 1,72 | 1,53 | 1,34 | 1,14 |

**Умова завдання**

1. 
2. **Алгоримт розв’язання завдання**

* Алгоритм розв’язання задачі:
  1. Визначити точність, крок та межі х і у.
  2. Визначити кількість вузлів.
  3. Створити відповідний інтервал х та за допомогою інтерполяції обрахувати відповідний інтервал у.
  4. Використовуючи відповідні формули обчислити значення інтеграла від функції (Метод прямокутників, Метод парабол, Метод трапецій, Метод трьох восьмих).
  5. Зауважу, що для Методу парабол повинна виконуватись умова про парну кількість вузлів та для Методу трьох восьмих кількість вузлів має бути кратна трьом.
* Щодо програмної реалізації слід:
  1. Спроектувати приблизний вигляд програмної системи;
  2. Обрати підходящу мову програмування для реалізації поставленої задачі (Python);
  3. Реалізувати програмну систему на обраній мові:
     + Створити методи виводу;
     + Зобразити графіки;
     + Забезпечити виконання потрібних розрахунків для роботи системи;
  4. Провести тестування розробленої програмної системи та виправити помилки виявлені під час тестування;
  5. Сформувати програму виконання для використання створеної програмної системи.
  6. Провести тестування програми виконання та виправити помилки при їх наявності.
  7. Зауважу, що для Методу парабол повинна виконуватись умова про парну кількість вузлів та для Методу трьох восьмих кількість вузлів має бути кратна трьом. Для програмної реалізації це буде проігноровано, оскільки це ускладнить алгоритм у плані знаходження абсолютної та відносної похибок (відбуватиметься зміна кількості вузлів). Також, задана точність 10-6 в програмі не продемонстрована через довгість розрахунку, але продемонстровано точність 10-2 та час за який були виконані ці розрахунки.

**Виконане завдання вручну**



1. **Код програмної реалізації**

*Файл main.py*

from math import fabs, log10 *# Для потрібних обрахунків.*from tabulate import tabulate *# Для побудови таблиці.*import time *# Для обрахунку час виконання програми.  
  
  
# Знайти інтерполяцію у заданій точці х.*def getInterpolation(listX=list, listY=list, xValue=float) -> list:  
 Ln = 0.0  
 LnStr = f"y({xValue}) = "  
 for x in range(len(listX)):  
 multiplyUp, multiplyDown = 1.0, 1.0  
 multiplyUpStr, multiplyDownStr = f"[ ", f"[ "  
 for xI in range(len(listX)):  
 if listX[x] != listX[xI]:  
 multiplyUp \*= xValue - listX[xI]  
 multiplyDown \*= listX[x] - listX[xI]  
 multiplyUpStr += f"({xValue} - {round(listX[xI], 2)}) \* "  
 multiplyDownStr += f"({round(listX[x], 2)} - {round(listX[xI], 2)}) \* "  
 else: *# Заміна '\*' для отсанньої ітерації.* multiplyUpStr = multiplyUpStr[:(len(multiplyUpStr) - 2)] + "]"  
 multiplyDownStr = multiplyDownStr[:(len(multiplyDownStr) - 2)] + "]"  
 Ln += multiplyUp / multiplyDown \* listY[x]  
 LnStr += f"{'{'} {multiplyUpStr} / {multiplyDownStr} {'}'} \* {round(listY[x], 2)}\n\t + "  
 else: *# Заміна '+' для отсанньої ітерації.* LnStr = LnStr[:(len(LnStr) - 2)] + f"= {Ln}"  
 return [Ln, LnStr]  
  
  
*# Сформувати список Х (При h = 0.2 буде список з завдання).*def getListX(h=0.2) -> list:  
 a, b = 0.0, 2.0  
 listX = [a]  
 while listX[-1] < b:  
 listX.append(listX[-1] + h)  
 if round(listX[-1], 2) > b:  
 listX.remove(listX[-1])  
 break  
 return listX  
  
  
*# Сформувати список У (При h = 0.2 буде список з завдання).*def getListY(h=0.2) -> list:  
 listY = [2.76, 2.65, 2.53, 2.39, 2.24, 2.08, 1.91, 1.72, 1.53, 1.34, 1.14]  
 if h != 0.2:  
 tableX = getListX()  
 tableY = listY.copy()  
 listX = getListX(h)  
 listY = [getInterpolation(tableX, tableY, x)[0] for x in listX]  
 return listY  
  
  
*# Отримати крок через кількість вузлів.*def getH(n=int, a=0.0, b=2.0) -> float:  
 return (b - a) / n  
  
  
*# Формула внутрішніх прямокутників.*def formulaInnerRectangles(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = sum(listY[1:])  
 return [h \* sumY, n, h, f"Метод внутрішніх прямокутників", f"{h} \* E{listY[1:]}"]  
  
*# Формула зовнішніх прямокутників.*def formulaOuterRectangles(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = sum(listY[:(len(listY) - 1)])  
 return [h \* sumY, n, h, f"Метод зовнішніх прямокутників", f"{h} \* E{listY[:(len(listY) - 1)]}"]  
  
*# Формула середніх прямокутників.*def formulaMiddleRectangles(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = 0.0  
 strSumY = f"{h} \* ( "  
 for i in range(n):  
 sumY += (listY[i] + listY[i + 1]) / 2.0  
 strSumY += f"({listY[i]} + {listY[i + 1]}) / 2 + "  
 else:  
 strSumY = strSumY[:(len(strSumY) - 2)] + ")"  
 return [h \* sumY, n, h, f"Метод середніх прямокутників", strSumY]  
  
*# Формула трапецій.*def formulaTrapezoids(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = (listY[0] + listY[-1]) / 2.0 + sum(listY[1:(len(listY) - 1)])  
 return [h \* sumY, n, h, f"Метод трапецій",  
 f"{h} \* ( ({listY[0]} + {listY[-1]}) / 2 + E{listY[1:(len(listY) - 1)]} )"]  
  
  
*# Формула парабол або Сімпсона (без врахування умови на виконання кількість вузлів n парне число).*def formulaParabols(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = listY[0] + listY[-1] + 2 \* sum(listY[2:(len(listY) - 2):2]) + 4 \* sum(listY[1:(len(listY) - 1):2])  
 return [h / 3.0 \* sumY, n, h, f"Метод парабол або Сімпсона", f"{h} / 3 \* ( ({listY[0]} + {listY[-1]}) + 2 \* E{listY[2:(len(listY) - 2):2]} + 4 \* E{listY[1:(len(listY) - 1):2]} )"]  
  
  
*# Формула трьох восьмих (без врахування умови на виконання кількість вузлів n кратна 3).*def formula3and8(n=int) -> list:  
 h = getH(n)  
 listY = getListY(h)  
 sumY = listY[0] + listY[-1]  
 sumYStr = f"3 / 8 \* {h} \* ( {listY[0]} + {listY[-1]} + "  
 for i in range(1, n):  
 if i % 3 == 0:  
 sumY += 2 \* listY[i]  
 sumYStr += f"2 \* {listY[i]} + "  
 else:  
 sumY += 3 \* listY[i]  
 sumYStr += f"3 \* {listY[i]} + "  
 else:  
 sumYStr = sumYStr[:len(sumYStr) - 2] + ")" *# Заміна останнього '+' (після отсаннього доданка) на закриття дужки.* return [(3.0 / 8.0) \* h \* sumY, n, h, f"Метод трьох восьмих", sumYStr]  
  
  
*# Отрмати значення при певній точності з певної формули.*def getResultFormula(func, eps=float, n=10, step=2, maxIteration=10 \*\* 4) -> list:  
 end = func(n)  
 start = end.copy()  
 counterIteration = 1  
 deltaAbsolute, deltaRelative = 0.0, 0.0  
 for i in range(maxIteration): *# При досягненні максимального числа ітерацій, буде повернуте останнє значення.* start = end.copy()  
 n \*= step  
 end = func(n)  
 deltaAbsolute = fabs(start[0] - end[0])  
 deltaRelative = round(fabs((start[0] - end[0]) / start[0]) \* 100, 2)  
 if deltaAbsolute < eps:  
 counterIteration = i + 1  
 break  
 else:  
 counterIteration = maxIteration  
 *# Модифікація списку даних для подальшого отримання інформації у консолі.* listData = end[:(len(end) - 1)].copy()  
 if counterIteration == maxIteration - 1:  
 listData.append(f"{counterIteration}\nостаннє значення\nпри максимальній ітерації")  
 else:  
 listData.append(counterIteration)  
 listData.append(round(deltaAbsolute, 2 + int(-log10(eps))))  
 listData.append(deltaRelative)  
 listData.append(end[-1])  
 return listData  
  
  
*# Отримати результати з еплсолоном.*def getDataResult(eps=float, n=10, step=100) -> list:  
 listFunc = [formulaInnerRectangles, formulaOuterRectangles, formulaMiddleRectangles, formulaTrapezoids, formulaParabols, formula3and8]  
 return [getResultFormula(func, eps, n, step) for func in listFunc]  
  
*# Отримати результати без епсолона.*def getDataResultNotEps(n=10) -> list:  
 listFunc = [formulaInnerRectangles, formulaOuterRectangles, formulaMiddleRectangles, formulaTrapezoids, formulaParabols, formula3and8]  
 return [func(n) for func in listFunc]  
  
*# Отримати рядок таблиці.*def getTable(nameTable=str, listHeaders=list, listContent=list, headersLeft=False, cutStrFormula=False, lengthCutFormula=20) -> str:  
 *# Для виводу Х і У.* if headersLeft:  
 for j in listContent: *# Округлення до 2 знаків після коми.* for k in range(len(j)):  
 j[k] = round(j[k], 2)  
 for i in range(len(listHeaders)): *# Вставка відповідного знака Х або У.* listContent[i] = [listHeaders[i]] + listContent[i]  
 listHeaders = []  
 *# Скорочення формули (обрізка) й долучення '...' в кінці - використання у методах обрахунку інтеграла.* if cutStrFormula and lengthCutFormula >= 20 :  
 for i in listContent:  
 i[-1] = i[-1][:lengthCutFormula] + "..."  
 return f"{nameTable}\n{tabulate(listContent, listHeaders, tablefmt='pretty')}"  
  
  
*# Отримати дані при заданій точності під час обрахунку кожного методу.*def getTableWithDataResult(eps=float, n=int, step=int, lengthStrFormula=50) -> str:  
 start = time.time()  
 listData = getDataResult(eps, n, step)  
 end = time.time()  
 nameLastTable = f"Результати розрахунків обчислені за {round(end - start, 2)} секунд, при заданому eps: {eps}, почтаковій кількості вузлів n: {n} й значенням зміни кількості вузлів {step}:"  
 listHeadersLastTable = [  
 "\nРезультат",  
 "Кіклькість вузлів\nn",  
 "Крок\nh",  
 "\nНазва методу",  
 f"Кількість необхідних ітерацій\nпри eps: {eps}",  
 "Абсолютна\nпохибка",  
 "Відносна\nпохибка\nу %",  
 "\nФомула обчислення"  
 ]  
 return f"\t\*Зауваження: взята {eps} точність для демонстрації, оскільки 10 \*\* -6 обраховується дуже довго.\n{getTable(nameLastTable, listHeadersLastTable, listData, False, True, lengthStrFormula)}"  
  
*# Отримати дані без точності під час обрахунку кожного методу.*def getTableWithDataResultNotEps(n=int, lengthStrFormula=50) -> str:  
 start = time.time()  
 listData = getDataResultNotEps(n)  
 end = time.time()  
 nameLastTable = f"Результати розрахунків обчислені за {round(end - start, 2)} секунд, при заданій кількості вузлів n: {n}:"  
 listHeadersLastTable = [  
 "Результат",  
 "Кіклькість вузлів n",  
 "Крок h",  
 "Назва методу",  
 "Фомула обчислення"  
 ]  
 return f"{getTable(nameLastTable, listHeadersLastTable, listData, False, True, lengthStrFormula)}"  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 eps = 10 \*\* -4 *# Взята ця точність для демонстрації, оскільки 10 \*\* -6 обраховується дуже довго.* h = 0.1 *# Крок для демострації інтерполяції.* xValue = 0.1 *# Значення для обрахунку інтерполяції.* n = 10 *# Кількість вузлів.* step = 2 *# Крок для зміни кількості вузлів.* print("Програму розробив Вальчевський П. В. для ЛР № 7, варіант № 3 з дисципліни Чисельні методи.\n")  
 print(  
 f"Приклад обрахування значення x: {0.1} за допомогою інтерполяції: {getInterpolation(getListX(), getListY(), xValue)[0]}")  
 print(getInterpolation(getListX(), getListY(), xValue)[1], "\n")  
 print(getTable("Табличні значення Х і У:", ["X", "Y"], [getListX(), getListY()], True), "\n")  
 print(getTable(f"Розраховані значення Х і У при кроці {h}:", ["X", "Y"], [getListX(h), getListY(h)], True), "\n")  
 print(getTableWithDataResult(eps, n, step, 50), "\n")  
 print(getTableWithDataResultNotEps(n, 95))

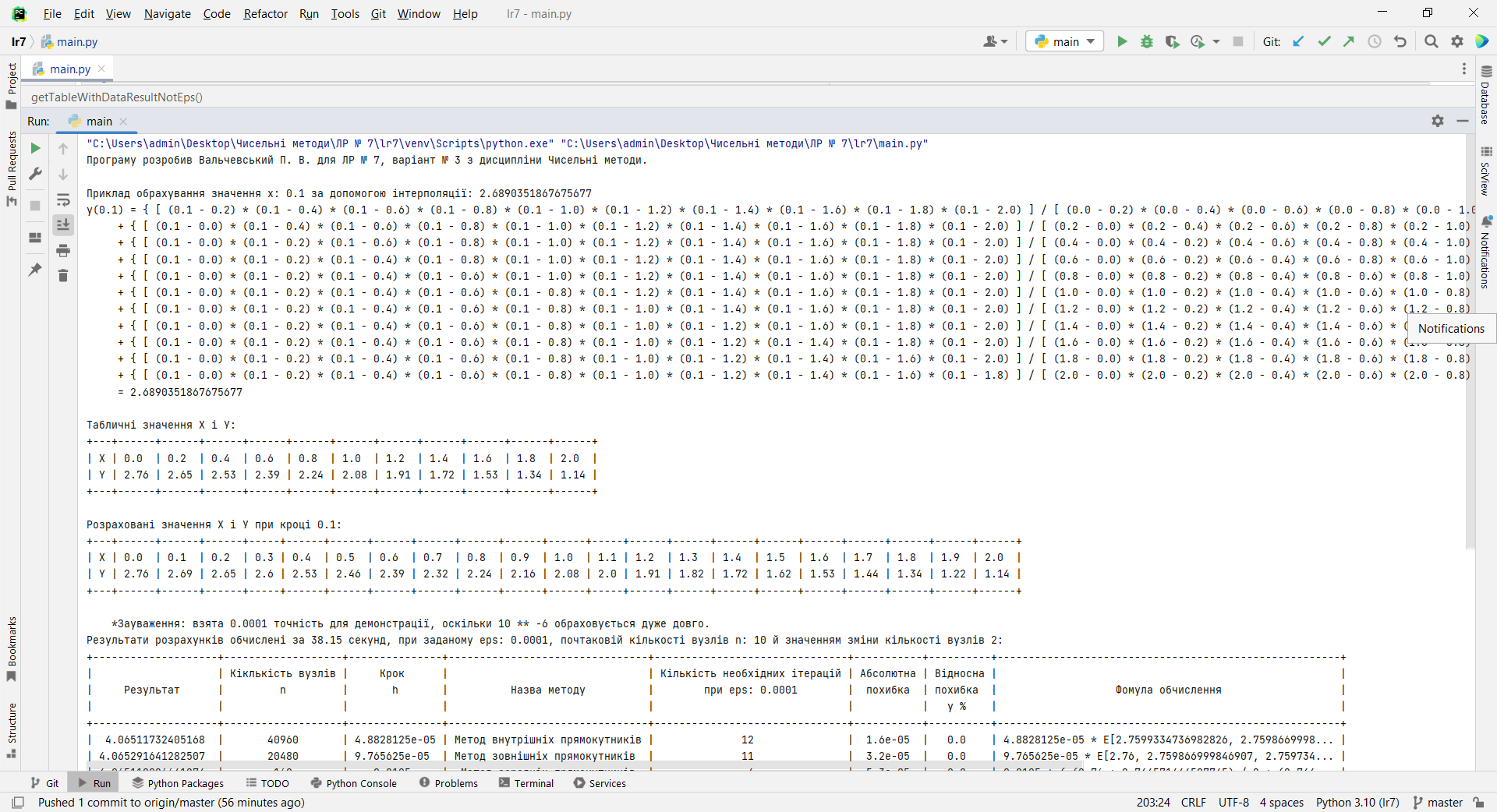
1. **Скріншот отриманого результату**
2. 

Рис. 1 Результат виконання програми.

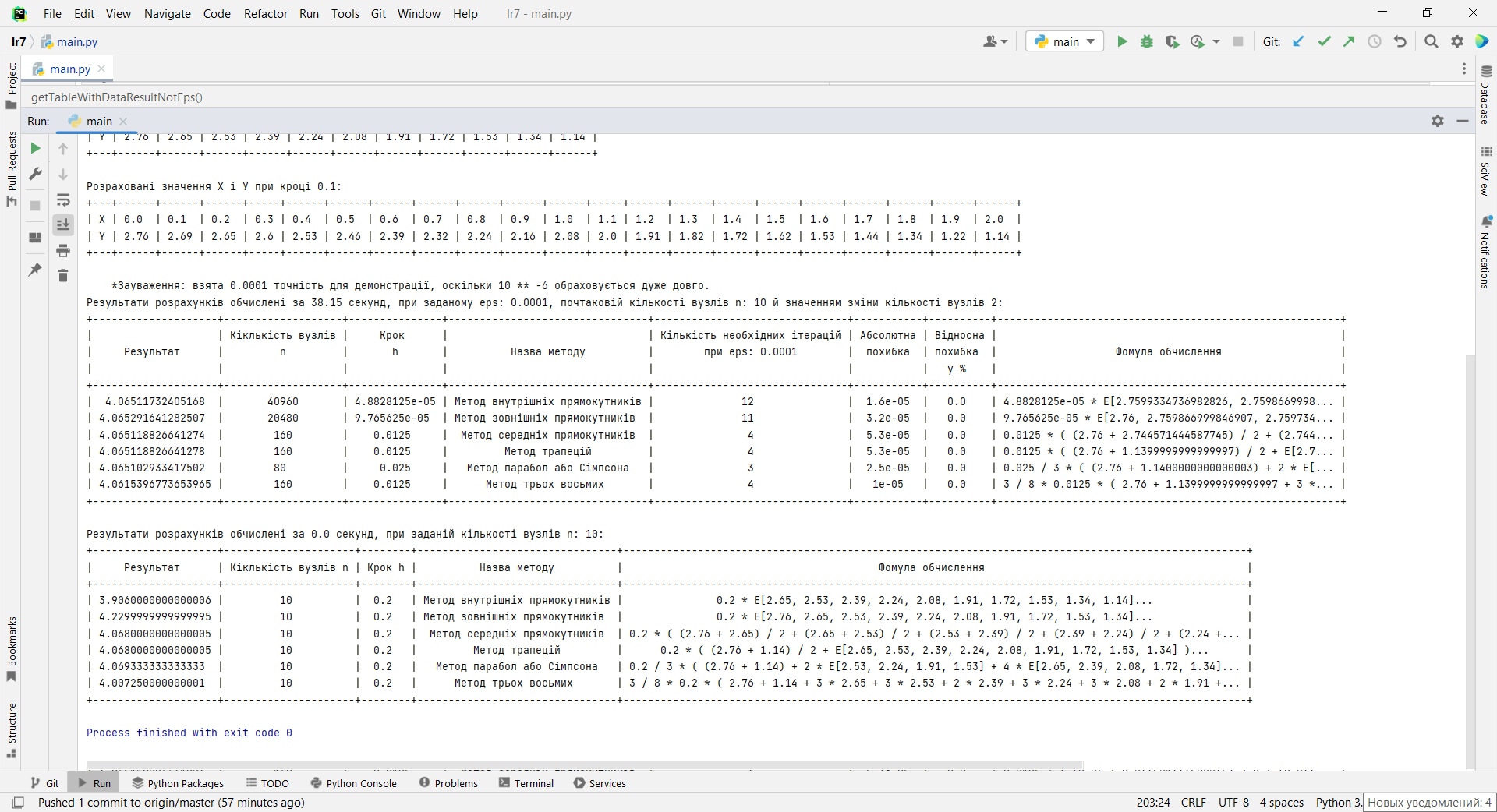
1. 

Рис. 2 Результат виконання програми (продовження).

1. **Висновки (з аналізом результату)**

Згідно результатів та порівняння отриманих під час виконання програми – алгоритм виконується правильно та без помилок.

Реалізовується алгоритм обчислення інтерполяції для заданої точки х для знаходження значення у в цій точці. Також, було знайдено значення інтеграла від заданої функції за допомогою методів: прямокутників, трапецій, парабол, трьох восьмих.

Усі розраховані дані є виведені в консолі у вигляді таблиць.

Зауважу, що для Методу парабол повинна виконуватись умова про парну кількість вузлів та для Методу трьох восьмих кількість вузлів має бути кратна трьом. Для програмної реалізації це буде проігноровано, оскільки це ускладнить алгоритм у плані знаходження абсолютної та відносної похибок (відбуватиметься зміна кількості вузлів). Також, задана точність 10-6 в програмі не продемонстрована через довгість розрахунку, але продемонстровано точність 10-2 та час за який були виконані ці розрахунки.

Загалом увесь код має відповідні коментарі до кожної ключової чи визначної дії, яка впливає на подальші розрахунки.

Програма була відлагоджена й виконана у програмному середовищі PyCharm 2023.2.1 за допомогою мови програмування Python 3.10.4.