НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи
з дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»
на тему
"Чисельне диференціювання та інтегрування"

Виконав:
студент групи КМ-01
Іваник Ю.П.

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

Київ — 2023

Зміст

Вступ

Метою роботи є ознайомитися з програмними засобами чисельного диференціювання й інтегрування функцій; практичне розв’язання задач з використанням СКМ, порівняльний аналіз методів чисельного диференціювання й інтегрування.

Основна частина

Варіант 9

Підінтегральна функція	Проміжок інтегрування	Кількість частин розбиття	Крок h	Первісна функція
$\frac{x}{sin^2 3x}$	[0.2; 1]	25	0.04	$-\frac{x}{3}ctg3x + \frac{1}{9}\ln\sin 3x$

Вимоги до ПЗ

1. Реалізувати перевірки на некоректний ввід (порожнє введення, символи замість числа, тощо).
2. У програмі повинно бути передбачено можливість гнучкого налагодження розмірності розв'язуваної задачі.
3. Має зберігатись логічно правильне розв'язання (нижня межа інтегрування не має перевищувати верхню, тощо)

Алгоритми розв'язання

Програма складається з двох частин – диференціювання та інтегрування, реалізованих на мові Python та мові Octave.

- Диференціювання
    - ***Python***: Для реалізації роботи програми використовувались наступні бібліотеки – Numpy [1], Pandas [2], Sympy [3]. Програма читає текстовий файл із вхідними даними. За допомогою функції Sympy.sympify [6] перетворює текстову формулу в пайтон-об'єкт із яким можна математично взаємодіяти. Диференціювання відбувається за допомогою функції Sympy.diff [7]. В циклі проходить по масиву точок і підставляє дані точки в формулу, отримані результати записуються в табличку, яка потім виводиться на екран користувача
    - ***Octave***: Для реалізації роботи програми використовувались наступні бібліотеки – Oct2py [5], Pandas [2]. Програма читає текстовий файл із вхідними даними. За допомогою вбудованої функції eval() інтерпретує формулу як октав-об'єкт із яким можна математично

взаємодіяти. Диференціювання відбувається за допомогою правої границі [9]. Результати та різниця між ними виводяться на екран користувача у вигляді таблиці.

- Інтегрування
  - ***Python*:** Для реалізації роботи програми використовувались наступні бібліотеки – Numpy [1], Pandas [2], Sympy [3], Scipy [4]. Програма читає текстовий файл із вхідними даними. За допомогою функції Sympy.sympify [6] перетворює текстову формулу в пайтон-об'єкт із яким можна математично взаємодіяти. Інтегрування відбувається за допомогою метода Сімпсона [8], а саме його реалізації в бібліотеці Scipy. В циклі проходить по масиву точок і підставляє дані точки в формулу, отримані результати записуються в табличку, яка потім виводиться на екран користувача разом із їх різницею.
  - **Octave:** Для реалізації роботи програми використовувались наступні бібліотеки – Oct2py [5], Pandas [2]. Програма читає текстовий файл із вхідними даними. За допомогою вбудованої функції eval() інтерпретує формулу як октав-об'єкт із яким можна математично взаємодіяти. Інтегрування відбувається методом Сімпсона. Нижня межа лишається не змінною, а верхня змінюється з заданим кроком h. Результати та різниця між ними виводяться на екран користувача.

Висновки

У результаті виконання роботи ми досягли таких результатів:

Похибка диференціювання мовою Python:

- Мінімальна – 0
- Максимальна – 5.55111512312578e-17

Похибка диференціювання мовою Octave:

- Мінімальна – 1.922221e-09
- Максимальна – 1.046345e-05

Похибка інтегрування мовою Python:

- Мінімальна – 0
- Максимальна – 7.328638e-02

Похибка інтегрування мовою Octave:

- Мінімальна – 0
- Максимальна – 2.664535e-15

Це досить хороші результати, оскільки в 3-х випадках ми отримали нульову похибку, а в решті випадків похибка була не суттєвою.

Скріншоти виконання програми наведені в Додатку А.

Відповіді на контрольні питання

1. ***Як залежить похибка усікання і похибка округлення від розміру кроку при чисельному диференціюванні ?*** – Похибка усікання і похибка округлення в чисельному диференціюванні залежать від розміру кроку. Похибка усікання зазвичай зменшується при зменшенні розміру кроку, оскільки більша кількість точок дозволяє краще апроксимувати функцію. Однак похибка округлення може збільшуватися при зменшенні розміру кроку через накопичення помилок округлення при виконанні більшої кількості обчислень.

2. ***Оцініть похибку заданої формули чисельного диференціювання за допомогою дослідження на апроксимацію*** – Оцінка похибки заданої формули чисельного диференціювання може бути виконана за допомогою дослідження на апроксимацію. Це включає в себе порівняння результатів, отриманих за допомогою формули, з точними значеннями, якщо вони доступні, або з результатами, отриманими за допомогою більш точних методів.

3. ***Проведіть порівняння точності формул трапецій і Сімпсона на підставі аналізу залишкових членів формул*** – Точність формул трапецій і Сімпсона можна порівняти на основі аналізу залишкових членів формул. Зазвичай формула Сімпсона дає більш точні результати, ніж формула трапецій, оскільки вона базується на квадратичному апроксимаційному поліномові, тоді як формула трапецій використовує лінійний апроксимаційний поліном.

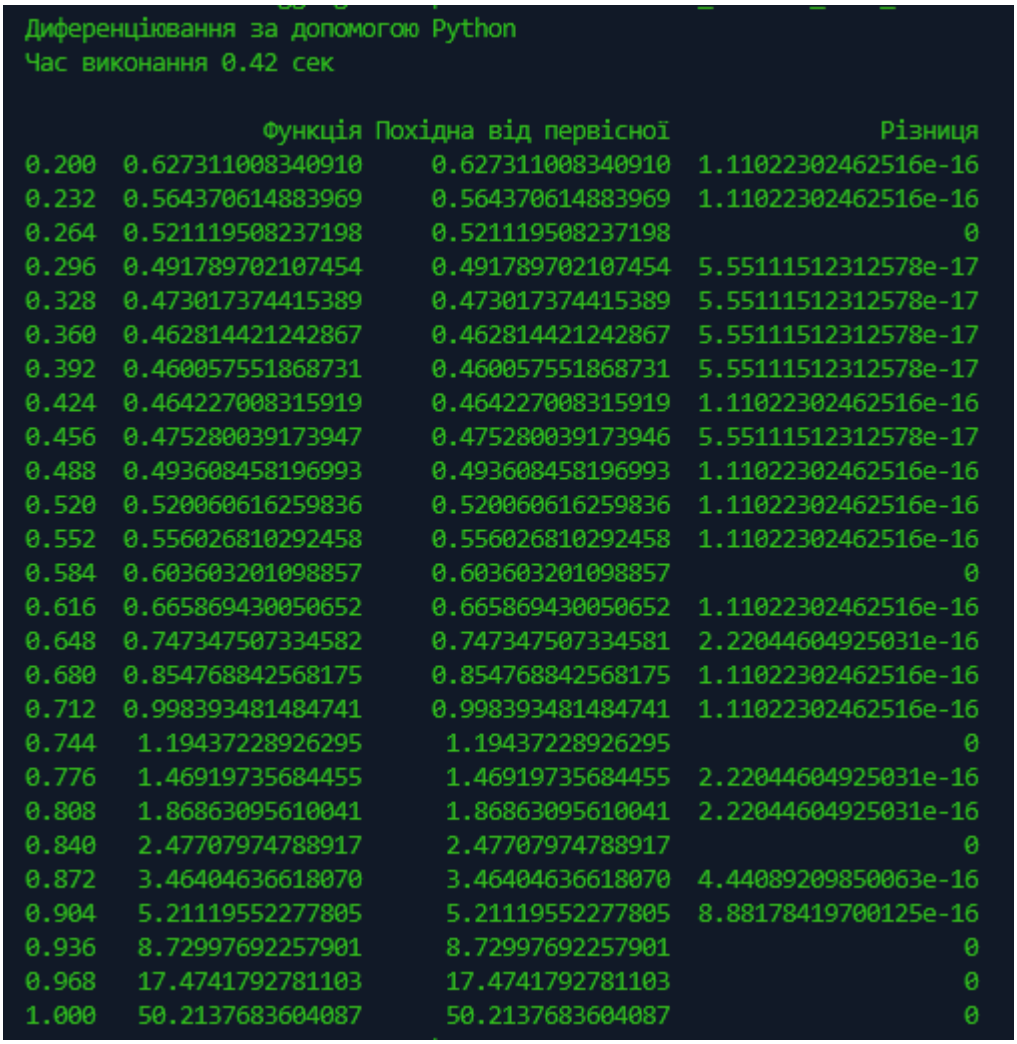
4. ***У яких випадках доцільно використовувати квадратні формули Гауса ?*** – Квадратурні формули Гауса доцільно використовувати, коли функція є достатньо гладкою і не має особливостей в межах інтегрування. Вони також ефективні для інтегрування поліноміальних функцій, оскільки вони можуть надати точний результат для поліномів до певного степеня.

5. ***Які формули чисельного інтегрування зручніше програмувати для ЕОМ ?*** – Для програмування на ЕОМ зручно використовувати ті формули чисельного інтегрування, які легко реалізуються і не вимагають складних обчислень. Це можуть бути такі методи, як метод прямокутників, метод трапецій та метод Сімпсона. Однак оптимальний вибір методу залежить від конкретної задачі і вимог до точності.

Перелік посилань

1. Numpy - https://numpy.org
2. Pandas - https://pandas.pydata.org
3. Sympy - https://www.sympy.org/en/index.html
4. Scipy - https://scipy.org
5. Oct2py - https://pypi.org/project/oct2py/
6. Sympy.sympify - https://gamma.sympy.org/input/?i=sympify
7. Sympy.diff - https://docs.sympy.org/latest/tutorials/intro-tutorial/calculus.html
8. Метод Сімпсона - https://www.blacksacademy.net/texts/SimpsonMethod.pdf

Додаток А – Скріншоти роботи програми



```
Диференціювання за допомогою Python
Час виконання 0.42 сек

                Функція Похідна від первісної              Різниця
0.200  0.627311008340910     0.627311008340910  1.11022302462516e-16
0.232  0.564370614883969     0.564370614883969  1.11022302462516e-16
0.264  0.521119508237198     0.521119508237198                    0
0.296  0.491789702107454     0.491789702107454  5.55111512312578e-17
0.328  0.473017374415389     0.473017374415389  5.55111512312578e-17
0.360  0.462814421242867     0.462814421242867  5.55111512312578e-17
0.392  0.460057551868731     0.460057551868731  5.55111512312578e-17
0.424  0.464227008315919     0.464227008315919  1.11022302462516e-16
0.456  0.475280039173947     0.475280039173946  5.55111512312578e-17
0.488  0.493608458196993     0.493608458196993  1.11022302462516e-16
0.520  0.520060616259836     0.520060616259836  1.11022302462516e-16
0.552  0.556026810292458     0.556026810292458  1.11022302462516e-16
0.584  0.603603201098857     0.603603201098857                    0
0.616  0.665869430050652     0.665869430050652  1.11022302462516e-16
0.648  0.747347507334582     0.747347507334581  2.22044604925031e-16
0.680  0.854768842568175     0.854768842568175  1.11022302462516e-16
0.712  0.998393481484741     0.998393481484741  1.11022302462516e-16
0.744   1.19437228926295      1.19437228926295                    0
0.776   1.46919735684455      1.46919735684455  2.22044604925031e-16
0.808   1.86863095610041      1.86863095610041  2.22044604925031e-16
0.840   2.47707974788917      2.47707974788917                    0
0.872   3.46404636618070      3.46404636618070  4.44089209850063e-16
0.904   5.21119552277805      5.21119552277805  8.88178419700125e-16
0.936   8.72997692257901      8.72997692257901                    0
0.968   17.4741792781103      17.4741792781103                    0
1.000   50.2137683604087      50.2137683604087                    0
```

Рис. 1 – Диференціювання за допомогою Python

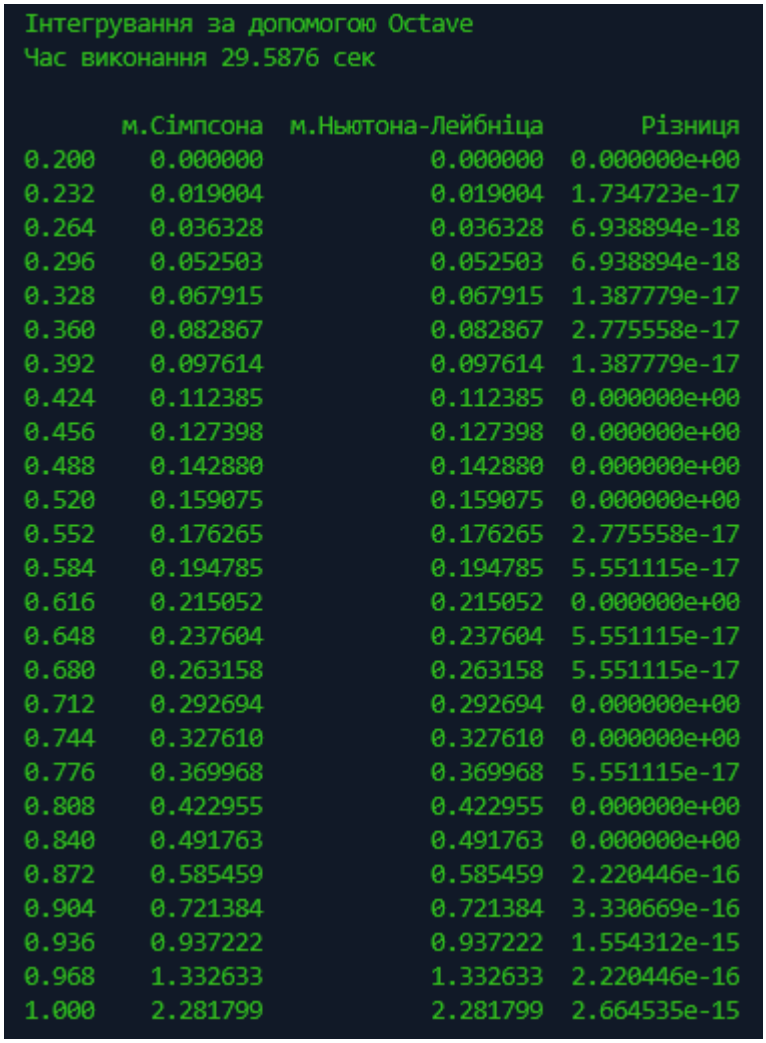
```
Інтегрування за допомогою Python
Час виконання 1.7479 сек

       м.Сімпсона  м.Ньютона-Лейбніца       Різниця
0.200    0.000000            0.000000  0.000000e+00
0.232    0.019067            0.019004  6.327406e-05
0.264    0.036330            0.036328  1.534642e-06
0.296    0.052499            0.052503  3.801487e-06
0.328    0.067917            0.067915  2.004069e-06
0.360    0.082867            0.082867  1.648636e-07
0.392    0.097616            0.097614  2.213321e-06
0.424    0.112387            0.112385  1.940509e-06
0.456    0.127400            0.127398  2.365218e-06
0.488    0.142883            0.142880  3.259086e-06
0.520    0.159078            0.159075  2.554709e-06
0.552    0.176270            0.176265  4.950632e-06
0.584    0.194788            0.194785  2.897073e-06
0.616    0.215060            0.215052  8.073737e-06
0.648    0.237608            0.237604  3.660696e-06
0.680    0.263173            0.263158  1.517911e-05
0.712    0.292699            0.292694  5.674285e-06
0.744    0.327644            0.327610  3.439446e-05
0.776    0.369980            0.369968  1.213836e-05
0.808    0.423053            0.422955  9.855214e-05
0.840    0.491802            0.491763  3.965557e-05
0.872    0.585851            0.585459  3.918304e-04
0.904    0.721610            0.721384  2.259925e-04
0.936    0.939944            0.937222  2.722393e-03
0.968    1.336066            1.332633  3.433194e-03
1.000    2.355086            2.281799  7.328638e-02
```

Рис. 2 – Інтегрування за допомогою Python

Диференціювання за допомогою Octave
Час виконання 29.2833 сек

	Функція	Похідна від первісної	Різниця
0.200	0.627311	0.627311	1.074908e-08
0.232	0.564371	0.564371	9.079471e-09
0.264	0.521120	0.521120	5.132015e-09
0.296	0.491790	0.491790	2.865658e-09
0.328	0.473017	0.473017	1.922221e-09
0.360	0.462814	0.462814	3.897120e-09
0.392	0.460058	0.460058	4.255484e-09
0.424	0.464227	0.464227	1.480348e-09
0.456	0.475280	0.475280	3.231766e-09
0.488	0.493608	0.493608	4.587352e-09
0.520	0.520061	0.520061	8.990157e-09
0.552	0.556027	0.556027	1.151498e-08
0.584	0.603603	0.603603	1.422785e-08
0.616	0.665869	0.665869	1.772393e-08
0.648	0.747348	0.747348	1.958471e-08
0.680	0.854769	0.854769	1.451551e-08
0.712	0.998393	0.998394	1.972116e-08
0.744	1.194372	1.194372	2.845140e-08
0.776	1.469197	1.469197	4.023030e-08
0.808	1.868631	1.868631	6.603900e-08
0.840	2.477080	2.477080	1.044585e-07
0.872	3.464046	3.464047	1.723231e-07
0.904	5.211196	5.211196	4.186567e-07
0.936	8.729977	8.729978	8.847412e-07
0.968	17.474179	17.474182	2.423505e-06
1.000	50.213768	50.213779	1.046345e-05

Рис. 3 – Диференціювання за допомогою Octave



```
Інтегрування за допомогою Octave
Час виконання 29.5876 сек

       м.Сімпсона  м.Ньютона-Лейбніца       Різниця
0.200    0.000000           0.000000  0.000000e+00
0.232    0.019004           0.019004  1.734723e-17
0.264    0.036328           0.036328  6.938894e-18
0.296    0.052503           0.052503  6.938894e-18
0.328    0.067915           0.067915  1.387779e-17
0.360    0.082867           0.082867  2.775558e-17
0.392    0.097614           0.097614  1.387779e-17
0.424    0.112385           0.112385  0.000000e+00
0.456    0.127398           0.127398  0.000000e+00
0.488    0.142880           0.142880  0.000000e+00
0.520    0.159075           0.159075  0.000000e+00
0.552    0.176265           0.176265  2.775558e-17
0.584    0.194785           0.194785  5.551115e-17
0.616    0.215052           0.215052  0.000000e+00
0.648    0.237604           0.237604  5.551115e-17
0.680    0.263158           0.263158  5.551115e-17
0.712    0.292694           0.292694  0.000000e+00
0.744    0.327610           0.327610  0.000000e+00
0.776    0.369968           0.369968  5.551115e-17
0.808    0.422955           0.422955  0.000000e+00
0.840    0.491763           0.491763  0.000000e+00
0.872    0.585459           0.585459  2.220446e-16
0.904    0.721384           0.721384  3.330669e-16
0.936    0.937222           0.937222  1.554312e-15
0.968    1.332633           1.332633  2.220446e-16
1.000    2.281799           2.281799  2.664535e-15
```

Рис. 4 – Інтегрування за допомогою Octave

Додаток В – Код програми

Вміст файлу func.txt :

0.2

1

25

x / (sin(3*x).^2)

(1/9) * (-3*x*cot(3*x) + log(sin(3*x)))

Вміст файлу main.py :

```
import time
import numpy as np
import pandas as pd
import sympy as sp
from scipy.integrate import simps
from oct2py import Oct2Py


def get_func():
    '''Читання формули, первісної та меж інтегрування з файлу
    '''
    with open('func.txt', 'r') as f:
        lines = f.readlines()
        lower_limit = float(lines[0].strip())
        upper_limit = float(lines[1].strip())
        n = float(lines[2].strip())
        func = lines[3].strip()
        antiderivative = lines[4].strip()
    return [lower_limit, upper_limit, n, func, antiderivative]

lower_limit, upper_limit, _, func, _ = get_func()

if lower_limit > upper_limit:
    print('lower_limit > upper_limit')


start_timer = time.time()
lower_limit, upper_limit, n, func_str, antiderivative_str = get_func()

try:
    func_str = func_str.replace('.', '')
    func_str = func_str.replace('^', '**')
    antiderivative_str = antiderivative_str.replace('.', '')
    antiderivative_str = antiderivative_str.replace('^', '**')
```

```
    x = sp.symbols('x')
    func = sp.sympify(func_str)  #  Функція
    antiderivative = sp.sympify(antiderivative_str)  #  Первісна

    antiderivative = sp.diff(antiderivative, x)  # Обчислення формули похідної

    h = (upper_limit - lower_limit) / n  #  Крок
    x_values = np.arange(lower_limit, upper_limit+h, h)  # Створюємо масив точок на
інтервалі [a, b] з кроком h

    # Обчислення значень функцій у цих точках
    func_values = [func.subs(x, val).evalf() for val in x_values]
    antiderivative_values = [antiderivative.subs(x, val).evalf() for val in
x_values]

    # Обчислення різниці між значеннями функцій
    difference = np.array(func_values) - np.array(antiderivative_values)

    diff_py = pd.DataFrame({
        'Функція': func_values,
        'Похідна від первісної': antiderivative_values,
        'Різниця': np.abs(difference)
    }, index=x_values)

    print('Диференціювання за допомогою Python')
    print(f'Час виконання {round(time.time() - start_timer, 4)} сек\n')
    print(diff_py)
except:
    print('SYNTAX ERROR')




start_timer = time.time()
lower_limit, upper_limit, n, func_str, antiderivative_str = get_func()

try:
    func_str = func_str.replace('.', '')
    func_str = func_str.replace('^', '**')
    antiderivative_str = antiderivative_str.replace('.', '')
    antiderivative_str = antiderivative_str.replace('^', '**')

    x = sp.symbols('x')
    func = sp.sympify(func_str)  #  Функція
    antiderivative = sp.sympify(antiderivative_str)  #  Первісна

    int_func = sp.integrate(func, x)  #  Інтеграл від функції

    h = (upper_limit - lower_limit) / n  #  Крок
```

```
    x_values = np.arange(lower_limit, upper_limit+h, h)  # Створюємо масив точок на інтервалі [a, b] з кроком h

    # Перетворюємо функцію та первісну у викликаємі функції
    func = sp.lambdify(x, func)
    antiderivative = sp.lambdify(x, antiderivative)

    y_values = [func(point) for point in x_values]  #  Створюємо масив значень функції для кожної точки

    # Обчислюємо чисельне значення визначеного інтегралу для кожної точки
    simpson = [simps(y_values[:i+1], x_values[:i+1], dx=h) for i in range(len(x_values))]

    # Обчислюємо чисельне значення визначеного інтегралу за допомогою методу Ньютона-Лейбніца
    newton_leibniz = [antiderivative(point) - antiderivative(lower_limit) for point in x_values]

    # Обчислюємо різницю між значеннями, отриманими двома методами
    difference = [abs(simpson[i] - newton_leibniz[i]) for i in range(len(x_values))]

    int_py = pd.DataFrame({
        'м.Сімпсона': simpson,
        'м.Ньютона-Лейбніца': newton_leibniz,
        'Різниця': difference
    }, index=x_values)

    print('Інтегрування за допомогою Python')
    print(f'Час виконання {round(time.time() - start_timer, 4)} сек\n')
    print(int_py)
except:
    print('SYNTAX ERROR')



start_timer = time.time()
lower_limit, upper_limit, n, func_str, antiderivative_str = get_func()

try:
    oc = Oct2Py()
    oc.eval('f1 = @(x) {0};'.format(func_str))
    oc.eval('F = @(x) {0};'.format(antiderivative_str))

    # Обчислюємо похідну функції F
    oc.eval('f2 = @(x) diff(F(x))./diff(x);')

    h = (upper_limit - lower_limit) / n  #  Крок
    x_values = np.arange(lower_limit, upper_limit+h, h)  # Створюємо масив точок на інтервалі [a, b] з кроком h

    # Обчислюємо значення функцій f1 та f2 для кожної точки в x_values
```

```
    results_f1 = []
    results_f2 = []
    delta = 1e-8  # Мале число для обчислення похідної
    for x in x_values:
        result_f1 = oc.eval('f1({0});'.format(x))
        #  Обчислюємо похідну за допомогою методу передньої різниці
        result_f2 = oc.eval('(F({0}+{1}) - F({0})) / {1};'.format(x, delta))
        results_f1.append(result_f1)
        results_f2.append(result_f2)

    # Створюємо DataFrame з результатами
    diff_oct = pd.DataFrame({
        'Функція': results_f1,
        'Похідна від первісної': results_f2,
        'Різниця': np.abs(np.array(results_f1) - np.array(results_f2))
    }, index=x_values)

    # Виводимо результати
    print('Диференціювання за допомогою Octave')
    print(f'Час виконання {round(time.time() - start_timer, 4)} сек\n')
    print(diff_oct)
except:
    print('SYNTAX ERROR')



start_timer = time.time()
lower_limit, upper_limit, n, func_str, antiderivative_str = get_func()

try:
    oc = Oct2Py()
    oc.eval('f1 = @(x) {0};'.format(func_str))
    oc.eval('f2 = @(x) {0};'.format(antiderivative_str))

    h = (upper_limit - lower_limit) / n  #  Крок
    x_values = np.arange(lower_limit, upper_limit+h, h)  # Створюємо масив точок на
інтервалі [a, b] з кроком h

    # Обчислюємо інтеграли за допомогою методу Ньютона-Лейбніца та методу Сімсона
для кожної точки в x_values
    results_simpson = []
    results_newton_leibniz = []
    for x in x_values:
        result_simpson = oc.eval('quad(f1, {0}, {1});'.format(lower_limit, x))
        result_newton_leibniz = oc.eval('f2({1}) - f2({0});'.format(lower_limit,
x))
        results_simpson.append(result_simpson)
        results_newton_leibniz.append(result_newton_leibniz)

    int_oct = pd.DataFrame({
        'м.Сімпсона': results_simpson,
        'м.Ньютона-Лейбніца': results_newton_leibniz,
```

```
        'Різниця': np.abs(np.array(results_simpson) -
np.array(results_newton_leibniz))
    }, index=x_values)

    # Виводимо результати
    print('Інтегрування за допомогою Octave')
    print(f'Час виконання {round(time.time() - start_timer, 4)} сек\n')
    print(int_oct)
except:
    print('SYNTAX ERROR')
```