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
Романецький М.С.

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 15

Підінтегральна функція	Проміжок інтегрування	Кількість частин розбиття	Крок h	Первісна функція
$x2^{3x}$	[1; 4]	50	0.06	$\frac{x2^{3x}}{3ln2} - \frac{2^{3x}}{9(ln2)^2}$

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 – 0

Похибка диференціювання мовою Octave:

- Мінімальна – 6.078432e-08
- Максимальна – 4.660402e-04

Похибка інтегрування мовою Python:

- Мінімальна – 0
- Максимальна – 0.095040

Похибка інтегрування мовою Octave:

- Мінімальна – 0
- Максимальна – 4.547474e-12

Це досить хороші результати, оскільки в половині випадків 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 0.3811 сек

	Функція	Похідна від первісної	Різниця
1.00	8.00000000000000	8.00000000000000	0
1.06	9.60685534730837	9.60685534730837	0
1.12	11.4994960421636	11.4994960421636	0
1.18	13.7255005635037	13.7255005635037	0
1.24	16.3400457825887	16.3400457825887	0
1.30	19.4070862239656	19.4070862239656	0
1.36	23.0007125626157	23.0007125626157	0
1.42	27.2067161689421	27.2067161689421	0
1.48	32.1243904739609	32.1243904739609	0
1.54	37.8686044669367	37.8686044669367	0
1.60	44.5721888407616	44.5721888407616	0
1.66	52.3886812626873	52.3886812626873	0
1.72	61.4954840794951	61.4954840794951	0
1.78	72.0974955925472	72.0974955925472	0
1.84	84.4312850031721	84.4312850031721	0
1.90	98.7698913969185	98.7698913969185	0
1.96	115.428338894385	115.428338894385	0
2.02	134.769973561540	134.769973561540	0
2.08	157.213743089499	157.213743089499	0
2.14	183.242557903506	183.242557903506	0
2.20	213.412892564665	213.412892564665	0
2.26	248.365809452714	248.365809452714	0
2.32	288.839613183553	288.839613183553	0

Рис. 1 – Диференціювання за допомогою Python (частина 1)

2.38	335.684374500652	335.684374500652	0
2.44	389.878597032953	389.878597032953	0
2.50	452.548339959392	452.548339959392	0
2.56	524.989154977137	524.989154977137	0
2.62	608.691247854328	608.691247854328	0
2.68	705.368334192197	705.368334192197	0
2.74	816.990726892436	816.990726892436	0
2.80	945.823270442015	945.823270442015	0
2.86	1094.46882588056	1094.46882588056	0
2.92	1265.91811179448	1265.91811179448	0
2.98	1463.60682270172	1463.60682270172	0
3.04	1691.48107882457	1691.48107882457	0
3.10	1954.07241286106	1954.07241286106	0
3.16	2256.58367266166	2256.58367266166	0
3.22	2604.98741678089	2604.98741678089	0
3.28	3006.13860622722	3006.13860622722	0
3.34	3467.90365439868	3467.90365439868	0
3.40	3999.30819275770	3999.30819275770	0
3.46	4610.70624750501	4610.70624750501	0
3.52	5313.97390834145	5313.97390834145	0
3.58	6122.73101118173	6122.73101118173	0
3.64	7052.59486019827	7052.59486019827	0
3.70	8121.47058971506	8121.47058971506	0
3.76	9349.88342338643	9349.88342338643	0
3.82	10761.3588383618	10761.3588383618	0
3.88	12382.8574989630	12382.8574989630	0
3.94	14245.2728028578	14245.2728028578	0
4.00	16384.0000000001	16384.0000000001	0

Рис. 2 – Диференціювання за допомогою Python (частина 2)

```
Інтегрування за допомогою Python
Час виконання 0.3142 сек

      м.Сімпсона  м.Ньютона-Лейбніца   Різниця
1.00     0.000000           0.000000  0.000000
1.06     0.528206           0.526885  0.001321
1.12     1.158538           1.158534  0.000004
1.18     1.913621           1.913487  0.000135
1.24     2.813369           2.813359  0.000010
1.30     3.883521           3.883335  0.000185
1.36     5.152751           5.152733  0.000018
1.42     6.655912           6.655659  0.000253
1.48     8.431791           8.431762  0.000028
1.54    10.527448          10.527105  0.000343
1.60    12.995211          12.995168  0.000042
1.66    15.898472          15.898010  0.000462
1.72    19.307659          19.307598  0.000060
1.78    23.307972          23.307351  0.000621
1.84    27.993994          27.993909  0.000085
1.90    33.480005          33.479174  0.000831
1.96    39.892777          39.892661  0.000117
2.02    47.385311          47.384201  0.001110
2.08    56.127217          56.127058  0.000159
2.14    66.322981          66.321502  0.001478
2.20    78.199154          78.198939  0.000216
2.26    92.028602          92.026636  0.001966
2.32   108.113469         108.113179  0.000290
```

Рис. 3 – Інтегрування за допомогою Python (частина 1)

```
2.38   126.817334          126.814724  0.002610
2.44   148.542583          148.542195  0.000388
2.50   173.773014          173.769553  0.003461
2.56   203.043806          203.043288  0.000518
2.62   236.997911          236.993329  0.004583
2.68   276.346255          276.345567  0.000688
2.74   321.942300          321.936239  0.006062
2.80   374.729345          374.728432  0.000913
2.86   435.839043          435.831034  0.008010
2.92   506.521679          506.520470  0.001209
2.98   588.276230          588.265656  0.010574
3.04   682.758209          682.756610  0.001599
3.10   791.951228          791.937280  0.013948
3.16   918.045297          918.043186  0.002111
3.22  1063.662967         1063.644584  0.018383
3.28  1231.698736         1231.695952  0.002783
3.34  1425.616934         1425.592723  0.024212
3.40  1649.239964         1649.236297  0.003667
3.46  1907.140430         1907.108564  0.031866
3.52  2204.362106         2204.357279  0.004827
3.58  2546.935806         2546.893893  0.041914
3.64  2941.511962         2941.505613  0.006349
3.70  3396.038866         3395.983771  0.055096
3.76  3919.279175         3919.270830  0.008345
3.82  4521.701130         4521.628748  0.072382
3.88  5214.842700         5214.831740  0.010961
3.94  6012.482026         6012.386986  0.095040
4.00  6929.801675         6929.787286  0.014389
```

Рис. 4 – Інтегрування за допомогою Python (частина 2)

Диференціювання за допомогою Octave
Час виконання 52.5321 сек

		Функція	Похідна від первісної	Різниця
1.00		8.000000	8.000000	8.460700e-08
1.06		9.606855	9.606855	1.415889e-07
1.12		11.499496	11.499496	6.873688e-08
1.18		13.725501	13.725501	8.365109e-08
1.24		16.340046	16.340046	6.712783e-08
1.30		19.407086	19.407087	2.916337e-07
1.36		23.000713	23.000713	6.078432e-08
1.42		27.206716	27.206716	3.553652e-07
1.48		32.124390	32.124391	4.955959e-07
1.54		37.868604	37.868605	6.736086e-07
1.60		44.572189	44.572189	6.540676e-07
1.66		52.388681	52.388681	1.361290e-07
1.72		61.495484	61.495485	8.914242e-07
1.78		72.097496	72.097498	2.512448e-06
1.84		84.431285	84.431286	1.137157e-06
1.90		98.769891	98.769890	1.707653e-06
1.96		115.428339	115.428340	7.032717e-07
2.02		134.769974	134.769978	4.371506e-06
2.08		157.213743	157.213745	1.866198e-06
2.14		183.242558	183.242554	3.972966e-06
2.20		213.412893	213.412893	8.370452e-07
2.26		248.365809	248.365814	4.151609e-06
2.32		288.839613	288.839614	7.320171e-07

Рис. 5 – Диференціювання за допомогою Octave (частина 1)

2.38	335.684375	335.684371	3.889163e-06
2.44	389.878597	389.878602	4.903336e-06
2.50	452.548340	452.548355	1.474193e-05
2.56	524.989155	524.989156	8.152182e-07
2.62	608.691248	608.691238	9.996545e-06
2.68	705.368334	705.368348	1.336111e-05
2.74	816.990727	816.990752	2.478064e-05
2.80	945.823270	945.823246	2.494027e-05
2.86	1094.468826	1094.468797	2.868912e-05
2.92	1265.918112	1265.918092	1.930882e-05
2.98	1463.606823	1463.606873	4.997416e-05
3.04	1691.481079	1691.481145	6.654709e-05
3.10	1954.072413	1954.072388	2.451729e-05
3.16	2256.583673	2256.583764	9.127601e-05
3.22	2604.987417	2604.987526	1.091751e-04
3.28	3006.138606	3006.138536	7.026134e-05
3.34	3467.903654	3467.903616	3.821536e-05
3.40	3999.308193	3999.308092	1.009511e-04
3.46	4610.706248	4610.706424	1.760862e-04
3.52	5313.973908	5313.973998	8.920351e-05
3.58	6122.731011	6122.730883	1.278195e-04
3.64	7052.594860	7052.595129	2.684999e-04
3.70	8121.470590	8121.470819	2.291631e-04
3.76	9349.883423	9349.883203	2.204283e-04
3.82	10761.358838	10761.358590	2.487003e-04
3.88	12382.857499	12382.857312	1.866311e-04
3.94	14245.272803	14245.273269	4.660402e-04
4.00	16384.000000	16384.000537	5.370729e-04

Рис. 6 – Диференціювання за допомогою Octave (частина 2)

```
Інтегрування за допомогою Octave
Час виконання 53.3846 сек

      м.Сімпсона  м.Ньютона-Лейбніца       Різниця
1.00    0.000000           0.000000  0.000000e+00
1.06    0.526885           0.526885  1.110223e-16
1.12    1.158534           1.158534  8.881784e-16
1.18    1.913487           1.913487  8.881784e-16
1.24    2.813359           2.813359  1.332268e-15
1.30    3.883335           3.883335  4.440892e-16
1.36    5.152733           5.152733  8.881784e-16
1.42    6.655659           6.655659  4.440892e-15
1.48    8.431762           8.431762  3.552714e-15
1.54   10.527105          10.527105  1.776357e-15
1.60   12.995168          12.995168  1.776357e-15
1.66   15.898010          15.898010  1.776357e-15
1.72   19.307598          19.307598  0.000000e+00
1.78   23.307351          23.307351  7.105427e-15
1.84   27.993909          27.993909  3.552714e-15
1.90   33.479174          33.479174  2.131628e-14
1.96   39.892661          39.892661  7.105427e-15
2.02   47.384201          47.384201  7.105427e-15
2.08   56.127058          56.127058  0.000000e+00
2.14   66.321502          66.321502  1.421085e-14
2.20   78.198939          78.198939  2.842171e-14
2.26   92.026636          92.026636  1.421085e-14
2.32  108.113179         108.113179  0.000000e+00
```

Рис. 7 – Інтегрування за допомогою Octave (частина 1)

```
2.38  126.814724         126.814724  1.421085e-14
2.44  148.542195         148.542195  0.000000e+00
2.50  173.769553         173.769553  2.842171e-14
2.56  203.043288         203.043288  2.842171e-14
2.62  236.993329         236.993329  1.136868e-13
2.68  276.345567         276.345567  5.684342e-14
2.74  321.936239         321.936239  5.684342e-14
2.80  374.728432         374.728432  3.979039e-13
2.86  435.831034         435.831034  2.842171e-13
2.92  506.520470         506.520470  5.684342e-14
2.98  588.265656         588.265656  3.410605e-13
3.04  682.756610         682.756610  6.821210e-13
3.10  791.937280         791.937280  4.547474e-13
3.16  918.043186         918.043186  1.136868e-13
3.22 1063.644584        1063.644584  1.136868e-12
3.28 1231.695952        1231.695952  6.821210e-13
3.34 1425.592723        1425.592723  1.136868e-12
3.40 1649.236297        1649.236297  6.821210e-13
3.46 1907.108564        1907.108564  1.818989e-12
3.52 2204.357279        2204.357279  0.000000e+00
3.58 2546.893893        2546.893893  9.094947e-13
3.64 2941.505613        2941.505613  4.547474e-13
3.70 3395.983771        3395.983771  2.273737e-12
3.76 3919.270830        3919.270830  2.273737e-12
3.82 4521.628748        4521.628748  7.275958e-12
3.88 5214.831740        5214.831740  1.818989e-12
3.94 6012.386986        6012.386986  4.547474e-12
4.00 6929.787286        6929.787286  3.637979e-12
```

Рис. 8 – Інтегрування за допомогою Octave (частина 2)

Додаток В – Код програми

Вміст файлу func.txt :

```
1
4
50
x.*2.^(3.*x)
(x.*2.^(3.*x)) / (3.*log(2)) - (2.^(3.*x)) / (9.*(log(2))^2)
```

Вміст файлу main.ipynb :

```
import time
from IPython.display import display, Markdown

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()

func = func.replace('(', '{(')
func = func.replace(')', ')}')
func = func.replace('.', '')
func = func.replace('*', '')

# Відображення формули у Markdown
display(Markdown(f'### Введена формула: $$\int_{{{lower_limit}}}^{{{upper_limit}}}
{func} \, dx$$'))
display(Markdown('### Введена первісна: $$\\frac{x2^{3x}}{3 \ln 2} -
\\frac{2^{3x}}{9 (\ln 2)^2}$$'))
if lower_limit > upper_limit:
    display(Markdown('### <span style="color:red"> lower limit > upper limit
</span>'))
```

```
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:
    display(Markdown('### <span style="color:red"> Syntax error in formula or
antiderivative</span>'))
```

```
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:
    display(Markdown('### <span style="color:red"> Syntax error in formula or antiderivative</span>'))
```

```
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:
    display(Markdown('### <span style="color:red"> Syntax error in formula
</span>'))
```

```
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:
    display(Markdown('### <span style="color:red"> Syntax error in formula
</span>'))
```