НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

про виконання лабораторної роботи №2

по дисципліні «Алгоритми і системи комп’ютерної математики - 1 »

на тему

«Чисельне диференціювання та інтегрування»

Виконав: Перевірив:

студент групи КМ-73 асистент кафедри ПМА

Звягін М.С. Дрозденко О.М.

Київ — 2020

Зміст

[1 Вступ](#_30j0zll) 3

[2 Вимоги до ПЗ](#_1fob9te) 4

[**3**](#_3dy6vkm) **Тестові випадки** 5

[**4**](#_lnxbz9) **Опис методів** 10

[**4.1 Метод трапецій**](#_35nkun2) 10

[**4.2 Чисельне диференціювання**](#_1ksv4uv) 10

[**5**](#_44sinio) **Висновки** 11

[Додаток А](#_2jxsxqh) 12

[Графи потоку керування для модулів програм 1](#_z337ya)2

[Додаток B 1](#_3j2qqm3)4

[Скріншоти виконання тестових випадків та модульних тестів 1](#_1y810tw)4

[Додаток С](#_4i7ojhp) 16

[Код програм](#_2xcytpi) 16

# 

# 1 Вступ

Мета виконання цієї лабораторної роботи полягає у написанні та тестуванні програмного забезпечення на мовах програмування Python та Octave, що здатне проводити операції чисельного інтегрування та диференціювання.

На мові Python та Octave був реалізований метод трапецій та центральна схема диференціювання 4 порядку точності.

Графи потоку керування для кожного модуля надані у додатку А.

Усі скріншоти виконання тестових випадків та модульних тестів наведено у додатку В. Проте читачу рекомендовано повторити їх власноруч у програмах, які додаються до цього звіту. Самі тестові випадки наведені у відповідному розділі.

Код кожної програми наведено у додатку С.

# Вимоги до ПЗ

1. **Валідація вхідних даних**

**А.1** При порожньому уведенні, програма повинна видати повідомлення про помилку та запросити значення ще раз.

**А.2** У кожному з полів допускаються символи «+», «-», «\*», «/», «.», «()», а також числа. При цьому поле повинне містити у собі лише одну змінну.

**А.3** Допускається функція log, яка відповідає натуральному логарифму. Тобто користувач не може ввести випадковий набір букв.

**А.4** Для методу трапецій допускається вибір інтервалу, який не більше 5. При цьому мінімальне допустиме число – це (-99999), а максимальне можливе – (999999). Це, з одного боку, не допускає переповнення розрядної сітки, а з іншого – забезпечує стислість введення.

**А.5** При виборі інтервалу користувач має можливість задавати лише числові значення. Таким чином, при введені у будь-яке з полів значень відмінних від числових, користувач повинен отримати повідомлення про помилку та запит на введення даних заново.

**А.6** Програма має можливість побудови графіку функції та первісної.

1. **Графічний інтерфейс користувача**

**B.1** Програма на мові Python повинна мати віконний графічний інтерфейс.

**B.2** Усі повідомлення про помилку (в обох програмах), а також результати обчислень повинні надаватися користувачу за допомогою діалогових вікон.

1. **Тестові випадки**

**3.1 Програма на мові Python**

1. **Валідація вхідних даних**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Сценарій | Кроки | Вхідні дані | Очікуваний результат | Експ. результат | Pass? |
| A1 | Перевірити введення порожніх полів | 1. Залишити поле порожным | 1.Порожне поле | 1. Повідомлення про помилку. | 1. Повідомлення про помилку | Так |
| А2 | Перевірити обробку буквенно-числових рядків | 1. Увести рядок, що містить некоректне використання симолів | 1. «x\*(\*(» | 1. Повідомлення про помилку. | 1. Повідомлення про помилку. | Так |
| A3 | Перевірити обробку функції log | 1. Увести випадковий набор букв  2. Увести функцію log | 1. «fgr»  2. «log(1)» | 1. Повідомлення про помилку.  2. Нормальна робота. | 1. Повідомлення про помилку.  2. Нормальна робота. | Так |
| А4 | Перевірити гранично допустимі числа | 1. Увести граничні значення | 1. «-100000»  2. «-99999»  3. «999999»  4. «1000000»  5. «0», «5»  6. «0», «6» | 1. Повідомлення про помилку  2. Нормальна робота  3. Нормальна робота  4. Повідомлення про помилку  5. Нормальна робота  6. Повідомлення про помилку | 1. Повідомлення про помилку  2. Нормальна робота  3. Нормальна робота  4. Повідомлення про помилку  5. Нормальна робота  6. Повідомлення про помилку | Так |
| A5 | Увести букви при виборі інтервалу | 1. Увести букви при виборі інтервалу | 1. «fgr» | 1. Повідомлення про помилку | 1. Повідомлення про помилку | Так |

### **B. Графічний інтерфейс користувача**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Сценарій | Кроки | Вхідні дані | Очікуваний результат | Експ. результат | Pass? |
| B1 | Перевірити наявність віконного інтерфейсу | 1.Запусти-ти програму | 1. - | 1. Віконний інтерфейс є | 1. Віконний інтерфейс є | Так |
| B2 | Перевірити вивід результатів | 1. Увести коректні дані і отримати результат | 1.func = 1  in\_func = x  interval = [1, 5] | 1. Результат обчислень (1) відображенні в окремих діалогових вікнах | 1. , відображено у діалогових вікнах | Так |

* 1. **Програма у середовищі Octave**

**А. Валідація вхідних даних**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Сценарій | Кроки | Вхідні дані | Очікуваний результат | Експ. результат | Pass? |
| A1 | Перевірити введення порожніх полів | 1. Залишити поле порожным | 1.Порожне поле | 1. Повідомлення про помилку. | 1. Повідомлення про помилку | Так |
| А2 | Перевірити обробку буквенно-числових рядків | 1. Увести рядок, що містить некоректне використання симолів | 1. «x\*(\*(» | 1. Повідомлення про помилку. | 1. Повідомлення про помилку. | Так |
| A3 | Перевірити обробку функції log | Неможливо перевірити, оскільки користувач працює із наперед заданою функцією | - | - | - | Так |
| А4 | Перевірити гранично допустимі числа | 1. Увести граничні значення | 1. «-100000»  2. «-99999»  3. «999999»  4. «1000000»  5. «0», «5»  6. «0», «6» | 1. Повідомлення про помилку  2. Нормальна робота  3. Нормальна робота  4. Повідомлення про помилку  5. Нормальна робота  6. Повідомлення про помилку | 1. Повідомлення про помилку  2. Нормальна робота  3. Нормальна робота  4. Повідомлення про помилку  5. Нормальна робота  6. Повідомлення про помилку | Так |
| A5 | Увести букви при виборі інтервалу | 1. Увести букви при виборі інтервалу | 1. «fgr» | 1. Повідомлення про помилку | 1. Повідомлення про помилку | Так |

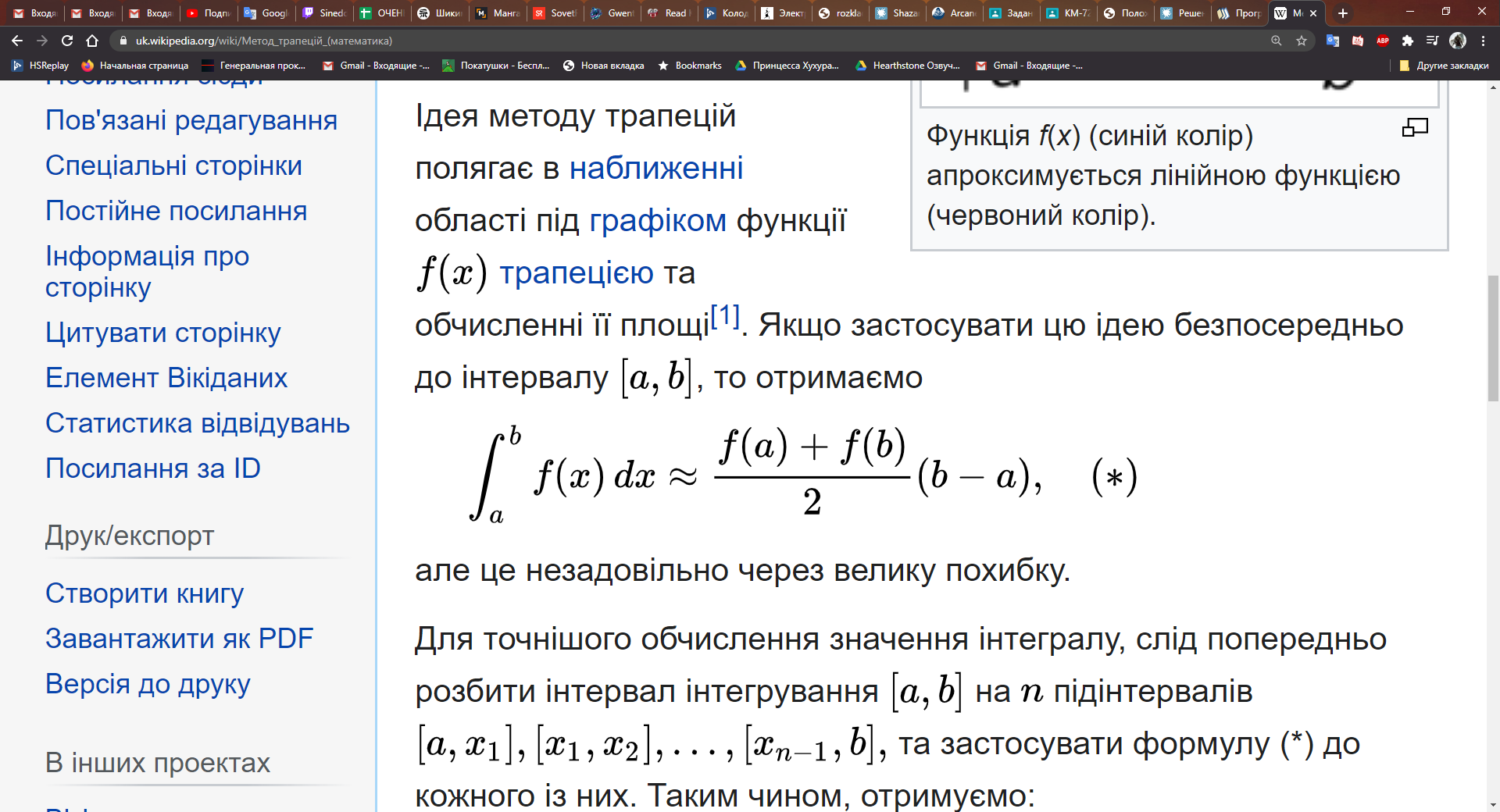
### **B. Графічний інтерфейс користувача**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Сценарій | Кроки | Вхідні дані | Очікуваний результат | Експ. результат | Pass? |
| B1 | Перевірити наявність віконного інтерфейсу | 1.Запусти-ти програму | 1. - | 1. Віконний інтерфейс є | 1. Віконний інтерфейс є | Так |
| B2 | Перевірити вивід результатів | 1. Увести коректні дані і отримати результат | 1.  interval = [1, 5]  step = 0.08 | 1. Результат обчислень (1.3896) | 1. | Так |

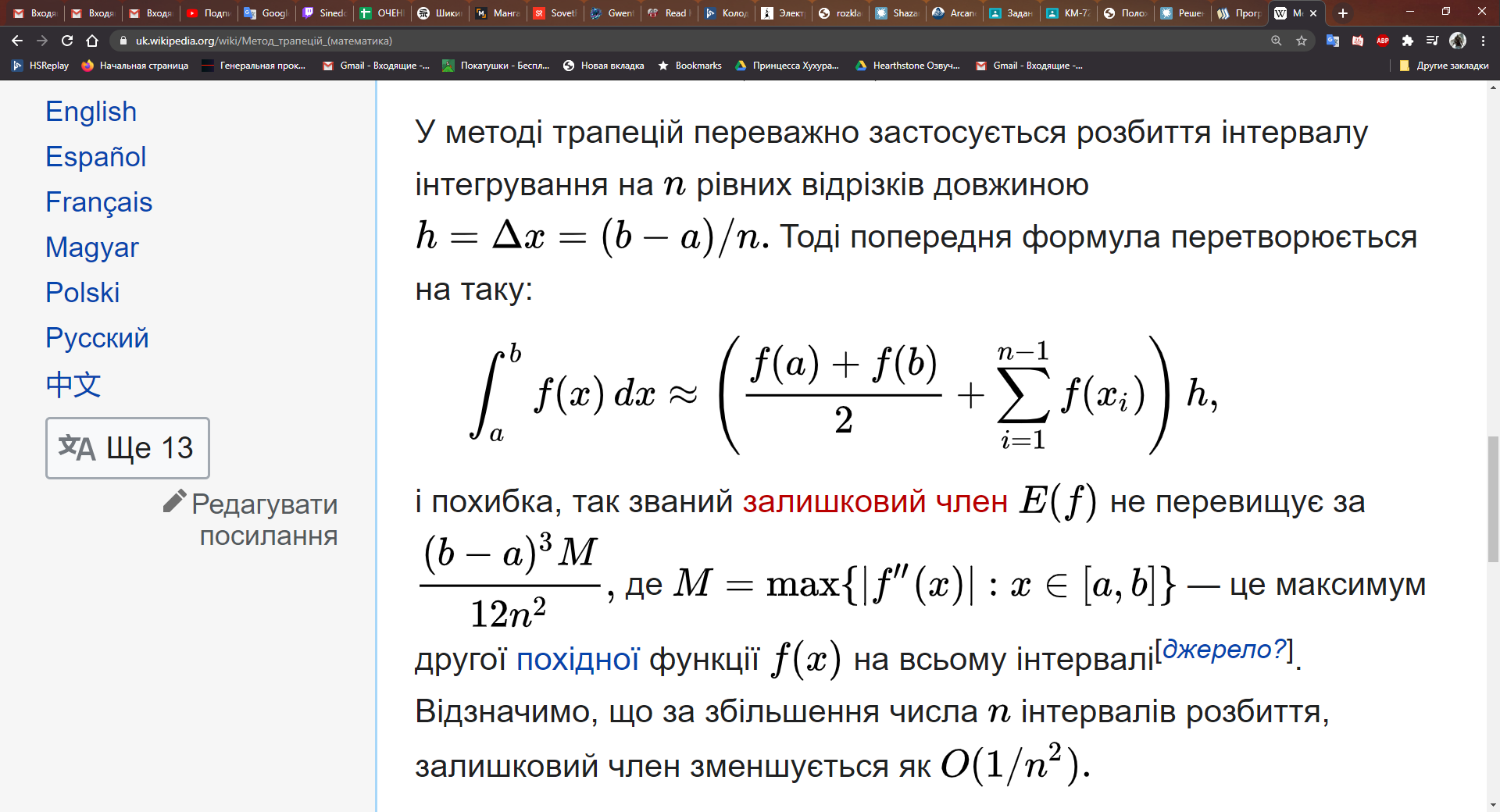
1. **Опис методів**

## **4.1 Метод трапецій**

Ідея методу трапецій полягає в наближенні області під графіком функції f(x) трапецією та обчисленні її площі. Якщо застосувати цю ідею безпосередньо до інтервалу [a,b], то отримаємо



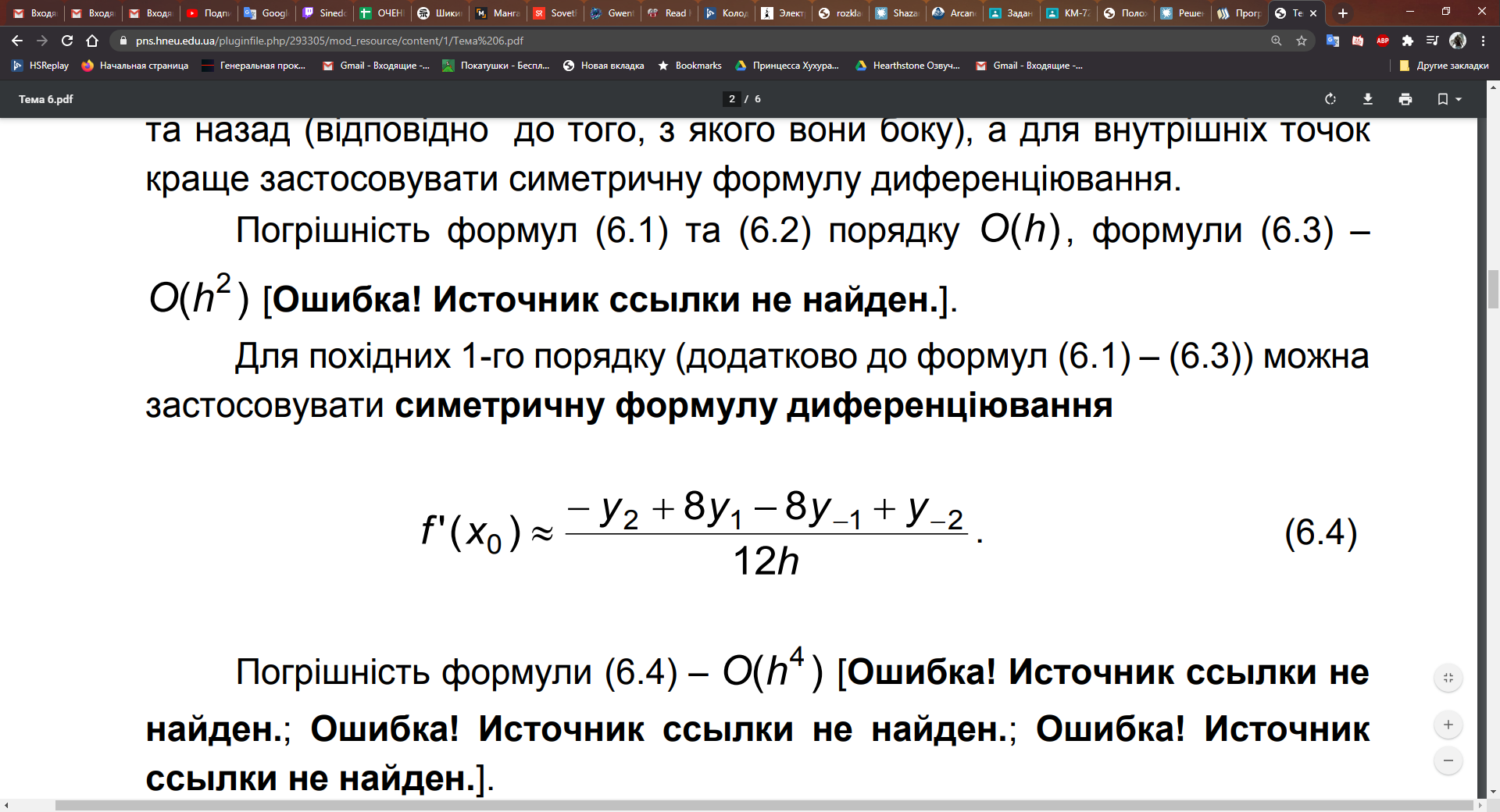
Для даної лабораторної роботи ми використовуємо формулу Котеса.



## **4.2 Чисельне диференціювання**

Задача чисельного диференціювання полягає в знаходженні значень похідних функції y = f(x) в заданих точках у випадках, коли аналітичний вид функції f(x) невідомий (задана неявно), дуже складний або функція f(x) задана таблицею. Привабливість чисельного підходу пояснюється наявністю простих формул, за допомогою яких похідні в заданих точках можна приблизно обчислити за кількома значеннями функції f(x) в цих та близьких до них точках.

У цій роботі ми застосовуємо таку формулу



1. **Висновки**

Насамкінець, продемонструємо роботу програм на індивідуальному завданні для варіанту №5, а саме таких функціях:



Значення визначеного інтегралу дорівнює 1.3896, значення похідної в точці 1 дорівнює 0.

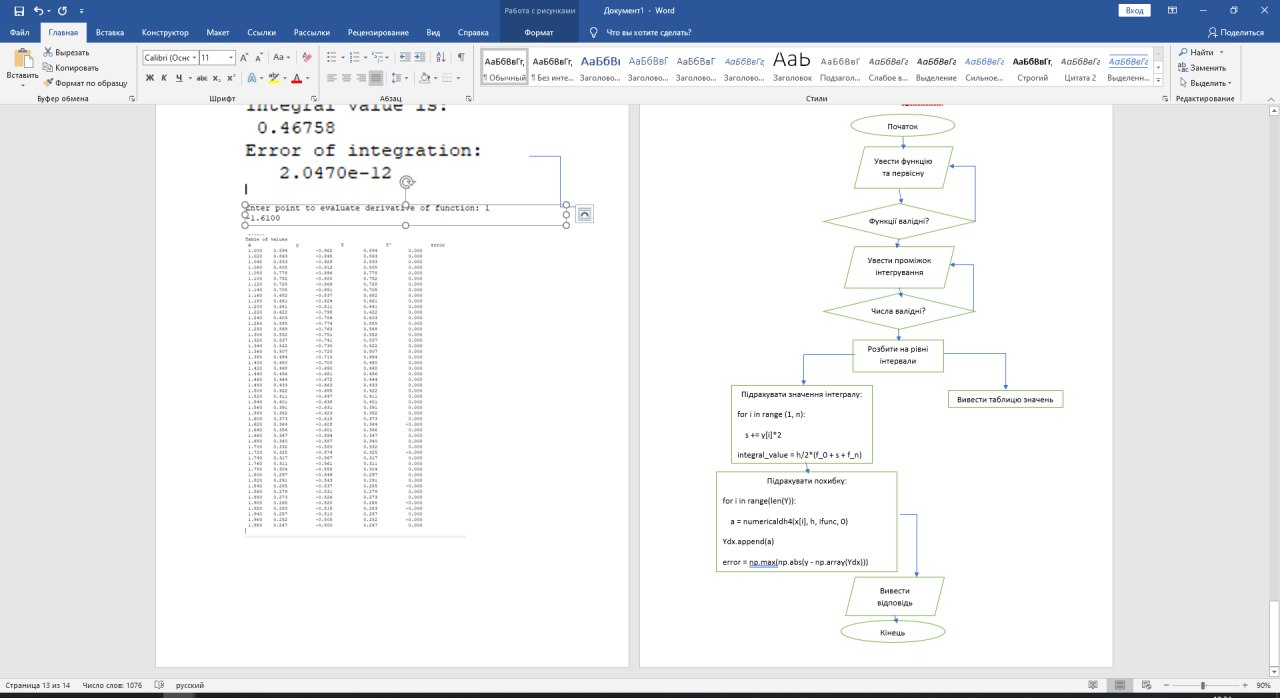
# 

# 

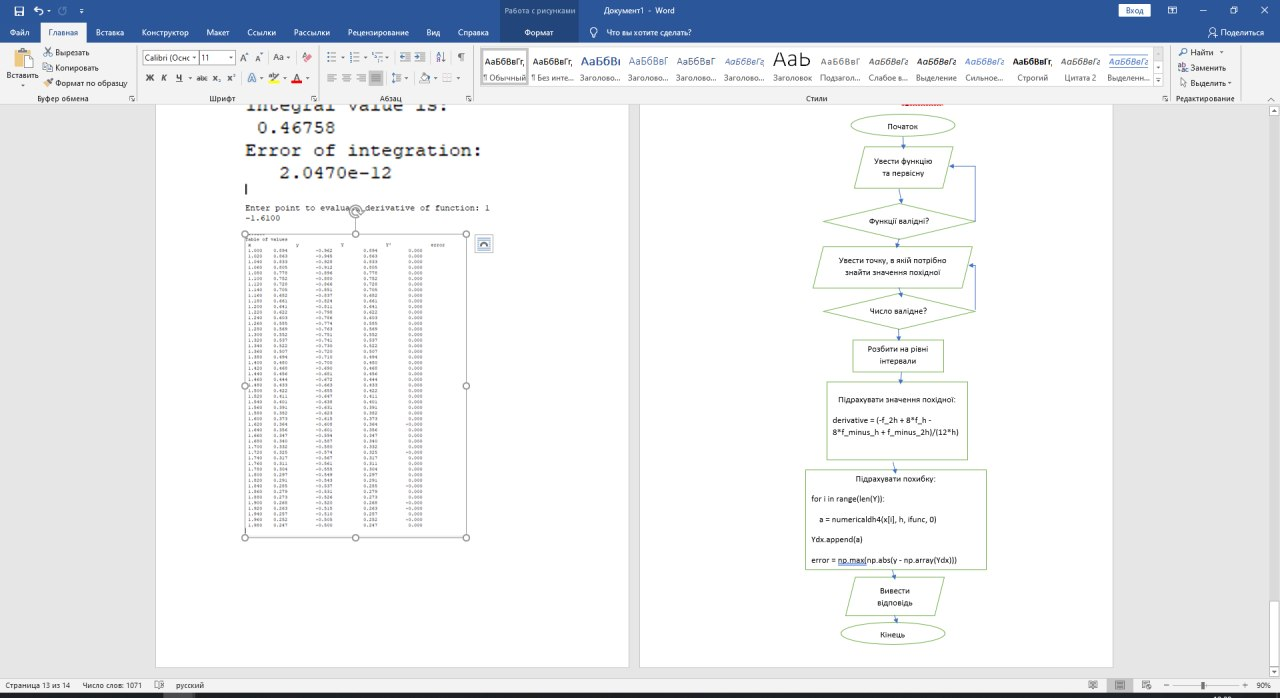
# Додаток А

# Графи потоку керування для модулів програм

Метод трапецій



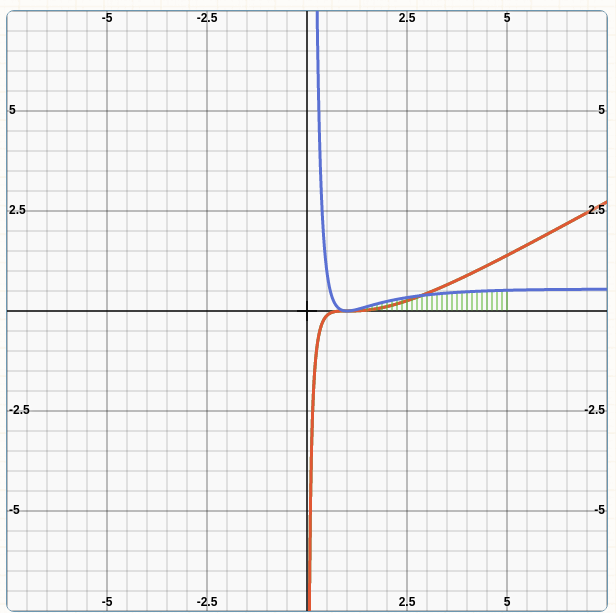
Знаходження похідної



# Додаток B

# Скріншоти виконання програми

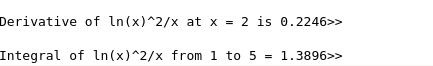
**Знаходження похідної та метод трапецій на Python**







**Знаходження похідної та метод трапецій на Octave**



# Додаток С

# Код програм

**main.py**

main.py

import tkinter as tk

from tkinter import \*

from tkinter import messagebox

import numpy as np

import pandas as pd

import matplotlib

import matplotlib.pyplot as plt

from pandastable import Table

import numexpr as ne

matplotlib.use('TkAgg')

def numericaldh4(x0, h, func, flag=1):

xs = []

x = x0 - h

xs.append(x)

f\_minus\_h = ne.evaluate(func)

x = x0 - 2\*h

xs.append(x)

f\_minus\_2h = ne.evaluate(func)

x = x0 + h

xs.append(x)

f\_h = ne.evaluate(func)

x = x0 + 2\*h

xs.append(x)

f\_2h = ne.evaluate(func)

derivative = (-f\_2h + 8\*f\_h - 8\*f\_minus\_h + f\_minus\_2h)/(12\*h)

if flag == 1:

global enter\_func, enter\_infunc

x = xs

func = enter\_func.get()

ifunc = enter\_infunc.get()

y = ne.evaluate(func)

Y = ne.evaluate(ifunc)

Ydx = []

for i in range(len(Y)):

a = numericaldh4(x[i], h, ifunc, 0)

Ydx.append(a)

error = np.max(np.abs(y - np.array(Ydx)))

messagebox.showinfo(title='Derivative value', message=f"Derivative value is\n {derivative} \n Error is \n {error}")

else:

return derivative

def get\_table():

try:

a = float(aentry.get())

b = float(bentry.get())

if a > 999999 or a < -99999 or b > 999999 or b < -99999:

messagebox.showerror(title="Warning", message="Max number 999999

min number -99999")

return

except:

messagebox.showerror(title="Warning", message="Entered expression is not a number!")

return

if abs(a - b) > 5:

messagebox.showerror(title="Warning", message="Interval is longer than five sorry!")

return

global newWindow

global enter\_func, enter\_infunc

def close\_window():

newWindow1.destroy()

newWindow.deiconify()

newWindow.withdraw()

newWindow1 = tk.Toplevel(newWindow)

newWindow1.title('Table of values')

newWindow1.geometry("440x330")

newWindow1.resizable(False, False)

newWindow1.wm\_protocol("WM\_DELETE\_WINDOW", close\_window)

newWindow1.configure(background='#FFFFFF')

a = float(aentry.get())

b = float(bentry.get())

h = 0.08

func = enter\_func.get()

ifunc = enter\_infunc.get()

x = np.arange(a, b+h, h)

y = ne.evaluate(func)

Y = ne.evaluate(ifunc)

Ydx = []

for i in range(len(Y)):

a = numericaldh4(x[i], h, ifunc, 0)

Ydx.append(a)

table\_of\_values = pd.DataFrame({'X values': x, 'f(x) values': y,

'F(x) values': Y, "F'(x) values": Ydx,

'Error': np.abs(y - np.array(Ydx))})

frame0 = Frame(newWindow1)

frame0.grid(rowspan=3)

pt = Table(frame0, dataframe=table\_of\_values, showscrollbar=True)

pt.show()

def trapecial():

global aentry, bentry, centry

global enter\_func, enter\_infunc

try:

a = float(aentry.get())

b = float(bentry.get())

if a > 999999 or a < -99999 or b > 999999 or b < -99999:

messagebox.showerror(title="Warning", message="Max number 999999 min number -99999")

return

except:

messagebox.showerror(title="Warning", message="Entered expression is not a number!")

return

if abs(a - b) > 5:

messagebox.showerror(title="Warning", message="Interval is longer than five sorry!")

return

func = enter\_func.get()

ifunc = enter\_infunc.get()

a = float(aentry.get())

b = float(bentry.get())

h = 0.08

n = int((b - a) / h)

x = np.arange(a, b+h, h)

y = ne.evaluate(func)

if y.size == 1:

y = np.array([int(func) for \_ in x])

f\_0 = y[0]

f\_n = y[-1]

s = 0

for i in range (1, n):

s += y[i]\*2

integral\_value = h/2\*(f\_0 + s + f\_n)

Y = ne.evaluate(ifunc)

Ydx = []

for i in range(len(Y)):

a = numericaldh4(x[i], h, ifunc, 0)

Ydx.append(a)

error = np.max(np.abs(y - np.array(Ydx)))

messagebox.showinfo(title='Integral value', message=f"Integral value is\n {integral\_value} \n Error is \n {error}")

def integrate\_trap():

global enter\_func, enter\_infunc

try:

x = np.arange(10)

ans = ne.evaluate(enter\_func.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

try:

x = np.arange(10)

ans = ne.evaluate(enter\_infunc.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

def close\_window():

newWindow.destroy()

window.deiconify()

window.withdraw()

global newWindow

newWindow = tk.Toplevel(window)

newWindow.title("Integration with trapezoidal rule")

newWindow.geometry("410x200")

newWindow.resizable(False, False)

newWindow.wm\_protocol("WM\_DELETE\_WINDOW", close\_window)

newWindow.configure(background='#FFFFFF')

newWindow.grid\_rowconfigure(0, weight=1)

newWindow.grid\_columnconfigure(0, weight=1)

iframe = Frame(newWindow, bg='#FFFFFF')

iframe.grid(columnspan=4, sticky='n')

simlabel = tk.Label(iframe, text='Enter interval', font='sans 14', bg='#FFFFFF', justify=CENTER, anchor=CENTER)

simlabel.grid(row=0, column=0, columnspan=4, sticky="nsew")

global func, aentry, bentry

alabel = tk.Label(iframe, text='Enter left boundary', font='sans 10', bg='#FFFFFF', justify=CENTER, anchor=CENTER)

alabel.grid(row=1, column=0, columnspan=2, padx=5)

aentry = tk.Entry(iframe, bd=4)

aentry.grid(row=2, column=0, columnspan=2, padx=5)

blabel = tk.Label(iframe, text='Enter right boundary', font='sans 10', bg='#FFFFFF', justify=CENTER, anchor=CENTER)

blabel.grid(row=1, column=2, columnspan=2, padx=5)

bentry = tk.Entry(iframe, bd=4)

bentry.grid(row=2, column=2, columnspan=2, padx=5)

intagbut = tk.Button(iframe, text="Integrate", command=trapecial, bg='#FFFFFF', width=10)

intagbut.grid(row=3, column=0, columnspan=2, padx=5, pady=10, sticky='nsew')

button\_table = tk.Button(iframe, text="Get Table of values", command=get\_table, bg='#FFFFFF', width=10)

button\_table.grid(row=3, column=2, columnspan=2, padx=5, pady=10, sticky='nsew')

def control():

global dentry, enter\_func

func = enter\_func.get()

try:

x0 = float(dentry.get())

if x0 > 999999 or x0 < -99999:

messagebox.showerror(title="Warning", message="Max number 999999 min number -99999")

return

except:

messagebox.showerror(title="Warning", message="Entered expression is not a number!")

return

h = 0.0

numericaldh4(x0, h, func, flag=1)

def get\_derivative():

global enter\_func, enter\_infunc

try:

x = np.arange(10)

ans = ne.evaluate(enter\_func.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

try:

x = np.arange(10)

ans = ne.evaluate(enter\_infunc.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

def close\_window():

newWindow.destroy()

window.deiconify()

window.withdraw()

global newWindow

newWindow = tk.Toplevel(window)

newWindow.title('Derivative computing window')

newWindow.geometry("410x200")

newWindow.resizable(False, False)

newWindow.wm\_protocol("WM\_DELETE\_WINDOW", close\_window)

newWindow.configure(background='#FFFFFF')

newWindow.grid\_rowconfigure(0, weight=1)

newWindow.grid\_columnconfigure(0, weight=1)

iframe = Frame(newWindow, bg='#FFFFFF')

iframe.grid(columnspan=4, sticky='n')

global func, dentry

dlabel = tk.Label(iframe, text='Enter x point\n to compute derivative', font='sans 10', justify=CENTER, anchor=CENTER)

dlabel.grid(row=1, column=0, columnspan=4, padx=5)

dentry = tk.Entry(iframe, bd=4)

dentry.grid(row=2, column=0, columnspan=4, padx=5, pady=10)

dxbut = tk.Button(iframe, text="Get derivative", command=control, width=10)

dxbut.grid(row=3, column=0, columnspan=4, padx=5, pady=10, sticky='nsew')

def plot\_inital():

global enter\_infunc

try:

x = np.arange(10)

ans = ne.evaluate(enter\_infunc.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

plt.style.use('seaborn')

func = enter\_infunc

x = np.linspace(-2, 2, 301)

func = enter\_infunc.get()

y = ne.evaluate(func)

if y.size == 1:

y = np.array([int(func) for \_ in x])

plt.title('Inital function', fontsize=20)

plt.plot(x, y, linewidth=4, color='royalblue', alpha=0.7)

plt.show()

def plot\_func():

global enter\_func

try:

x = np.arange(10)

ans = ne.evaluate(enter\_func.get())

except:

messagebox.showerror(title="Warning", message="Entered expression is not a function!\n Enter one varible function!")

return

plt.style.use('seaborn')

x = np.linspace(-2, 2, 301)

func = enter\_func.get()

y = ne.evaluate(func)

if y.size == 1:

y = np.array([int(func) for \_ in x])

plt.title('Function', fontsize=20)

plt.plot(x, y, linewidth=4, color='royalblue', alpha=0.7)

plt.show()

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

window = tk.Tk()

window.title('Integration and derivatives')

window.geometry("600x600")

window.resizable(False, False)

window.configure(background='#FFFFFF')

window.grid\_rowconfigure(0, weight=1)

window.grid\_columnconfigure(0, weight=1)

frame1 = tk.Frame(window, bg='#FFFFFF')

frame1.grid(row=0, column=0)

button = tk.Button(frame1, text="Get Integral with trapezoidal rule", command=integrate\_trap, justify=CENTER, anchor=CENTER)

button.grid(row=3, column=0, padx=5, pady=5, columnspan=3)

button2 = tk.Button(frame1, text="Get Derivative", command=get\_derivative, justify=CENTER, anchor=CENTER)

button2.grid(row=4, column=0, padx=5, pady=5, columnspan=3)

button3 = tk.Button(frame1, text="Plot Function", command=plot\_func, justify=CENTER, anchor=CENTER)

button3.grid(row=5, column=0, padx=5, pady=5, columnspan=3)

button4 = tk.Button(frame1, text="Plot Inital\n Function", command=plot\_inital, justify=CENTER, anchor=CENTER)

button4.grid(row=6, column=0, padx=5, pady=5, columnspan=3)

label\_func = tk.Label(frame1, text=' Enter function:', fg='black', font='sans 14')

label\_func.grid(row=1, column=0, padx=5, pady=25, columnspan=2)

global enter\_func

enter\_func = tk.Entry(frame1, bd=4)

enter\_func.grid(row=1, column=2, padx=5, pady=25)

label\_infunc = tk.Label(frame1, text='Enter inital function:', fg='black', font='sans 14')

label\_infunc.grid(row=2, column=0, padx=5, pady=25, columnspan=2)

global enter\_infunc

enter\_infunc = tk.Entry(frame1, bd=4)

enter\_infunc.grid(row=2, column=2, padx=5, pady=25)

exit\_but = tk.Button(frame1, text="Exit", command=exit, padx=5, pady=10, width=10)

exit\_but.grid(row=7, padx=5, pady=20, columnspan=3, sticky='n')

window.mainloop()

**main.m**

function simp\_integ\_dif

function a = get\_double(text2)

a = str2double(input(sprintf(text2), 's'));

while ~isfloat(a) || isempty(a) || isnan(a) || a < -99999 || a > 999999

a = str2double(input(sprintf(['Enter correct float value:']), 's'));

end

end

function x = labfunc(x)

x = ln(x)^2/x end

function x = initalfunc(x)

x = ln(x)^3/3

end

function derivative\_func = numericaldh4\_func(x0)

h0 = 0.001;

if length(x0) == 1

while sum(isinf(labfunc(x0:h0:x0-h0))) > 0

h0 = h0/2;

end

end

f\_minus\_h = labfunc(x0 - h0);

f\_minus\_2h = labfunc(x0 - 2\*h0);

f\_h = labfunc(x0 + h0);

f\_2h = labfunc(x0 + 2\*h0);

derivative\_func = (-f\_2h + 8\*f\_h - 8\*f\_minus\_h + f\_minus\_2h)/(12\*h0);

end

function derivative\_init = numericaldh4\_init(x0)

h0 = 0.001;

if length(x0) == 1

while sum(isinf(initalfunc(x0:h0:x0-h0))) > 0

h0 = h0/2;

end

end

f\_minus\_h = initalfunc(x0 - h0);

f\_minus\_2h = initalfunc(x0 - 2\*h0);

f\_h = initalfunc(x0 + h0);

f\_2h = initalfunc(x0 + 2\*h0);

derivative\_init = (-f\_2h + 8\*f\_h - 8\*f\_minus\_h + f\_minus\_2h)/(12\*h0);

end

function integ = simpson(y, n, h)

f\_0 = y(1);

f\_n = y(end);

s=0;

for k=2:n

s = s + y(k)\*2;

end

integ = h/2\*(f\_0 + f\_n + s);

end

a = 0;

b = 0;

h = 1;

x = 0;

n = 0;

y = 0;

disp('Function to integrate or differentiate is ln(x)^2/x;

selection = menu('Select Action', ...

'Enter parameters',...

'Find solution integral with Simpson method', ...

'Find derivative', ...

'Get table of values', ...

'Plot function', ...

'Exit');

while selection < 6

switch selection

case 1

a\_text = ['Enter starting point of integration interval: '];

a = get\_double(a\_text);

b\_text = ['Enter ending point of integration interval: '];

b = get\_double(b\_text);

h\_text = ['Enter step size of integration interval: '];

h = get\_double(h\_text);

case 2

if h >= abs(a - b) || h == 0

disp('Impossible to count integral. Step size is bigger than interval.');

disp('Try to enter new parameters.');

else

n = abs(b - a)/h;

x = a:h:b-h;

y = labfunc(x);

if sum(isinf(y)) > 0

disp('Impossible to count integral. Break point inside integration interval.');

disp('Try to enter new parameters.');

else

integ = simpson(y, n, h);

disp('Integral value is: ')

disp(integ);

xx = a:h:b-h;

yy = labfunc(xx);

Y\_dxx = numericaldh4\_init(xx);

errorr = yy - Y\_dxx;

disp('Error of integration: ');

disp(max(errorr));

end

end

case 3

x\_text = ['Enter point to evaluate derivative of function: '];

x0 = get\_double(x\_text);

if isinf(labfunc(x0))

disp('Impossible to count integral. Break point inside integration interval.');

disp('Try to enter new point.');

else

derivative = numericaldh4\_func(x0);

disp(derivative);

end

case 4

if (h == 0) || h >= abs(a - b)

disp('Imposible to display table of values.');

disp('Try to enter new parameters.');

else

x = a:h:b-h;

y = labfunc(x);

Y = initalfunc(x);

Y\_dx = numericaldh4\_init(x);

error = y - Y\_dx;

fprintf('Table of values\n');

fprintf(" x \t\t y \t\t Y \t\t Y' \t\t error\n");

fprintf("%6.3f \t %6.3f \t %6.3f \t %6.3f \t %6.3f\n", [x; y; Y; Y\_dx; error]);

end

case 5

%close all;

%xx = 0.1:0.001:1;

%yy = labfunc(xx);

%YY = initalfunc(xx);

%tiledlayout(2,1)

%ax1 = nexttile;

%plot(ax1, xx, yy)

%title(ax1, 'Function')

%ax2 = nexttile;

%plot(ax2, xx, YY)

%title(ax2, 'Inital Function')

end

selection = menu('Select Action', ...

'Enter parameters',...

'Find solution integral with trapezoidal rule', ...

'Find derivative', ...

'Get table of values', ...

'Plot function', ...

'Exit');

end

clear selection;

clear all;

end