НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №1

із дисципліни «Методи обчислення»

на тему

«Наближення функції»

2 варіант

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-81 | Бай.Ю.П. |
| Верзун П.В. |  |

Київ — 2020

**ВСТУП**

Метою роботи є здобути практичні навички розробки алгоритмів та програм, які здійснюють інтерполювання поліномами Лагранжа та Ньютона та схему Ейткена. Функція та необхідна точність приведені нижче. Для обчислення була взята точка -0.5, тому що f(0)=e0=1.

**Варіант №2**

|  |  |  |
| --- | --- | --- |
| Варіант | Функція | Точність |
| 2 | e2x+1 | 0.002 |

**ОСНОВНА ЧАСТИНА**

Всі алгоритми використовують точку -0.5, *x*є[-1.5, 0.5] та 3 вузла.

Схема Ейткена дає можливість наближено підрахувати значення функції в точці, результат виконання алгоритму наведено на рис.1.

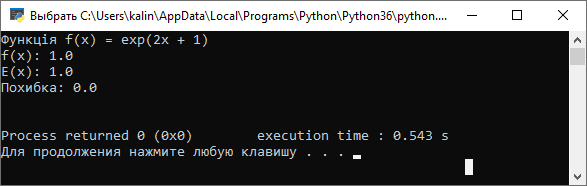


Рисунок 1

Виходить, що при описаних вище умовах метод Ейткена обчислює значення, яке Python не може відрізнити від 1.0, тобто метод ідеально виконує свою роботу.

Метод Ньютона виявився не менш точним, за схему Ейткена(рис.2):

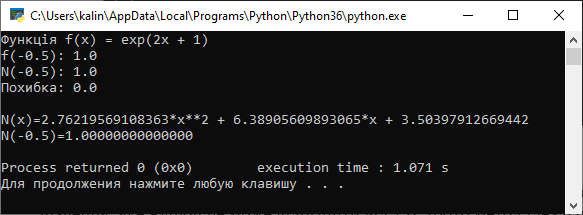


Рисунок 2

Так, як поліноми Лагранжа та Ньютона це, по суті, один поліном, то для демонстрації роботи алгоритму виведемо(формула 1) поліном Ньютона та підставимо в нього шукане значення (програмні розрахунки приведені на рисунку 2).



Формула 1

Власне поліном, що переведено в біль читабельний запис наведено нижче:

N(x) = 2.76219569108363x2 + 6.38905609893065x + 3.50397912669442

Програмно було підраховано значення N(-0.5), що показано на рисунку 2, проте перевіримо це на калькуляторі ще раз (рис.3).

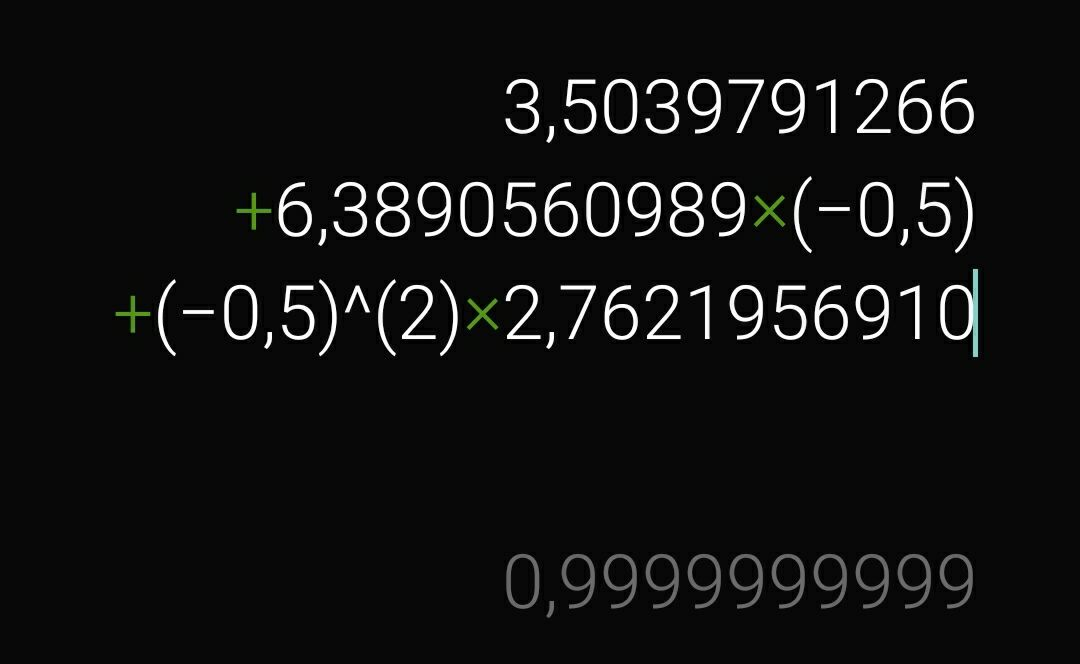


Рисунок 3

Я видно результат не 1, проте, скорше за все це через те, що телефонний калькулятор дозволяє ввести лише 10 знаків після коми, а в поліномі їх 14. Однак, це все ще дуже близько до 1, що підтверджує правильність обчислень.

Наступним кроком буде демонстрація роботи алгоритму Лагранжа(рис.4).

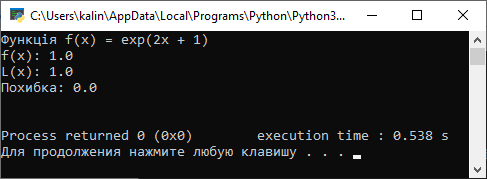
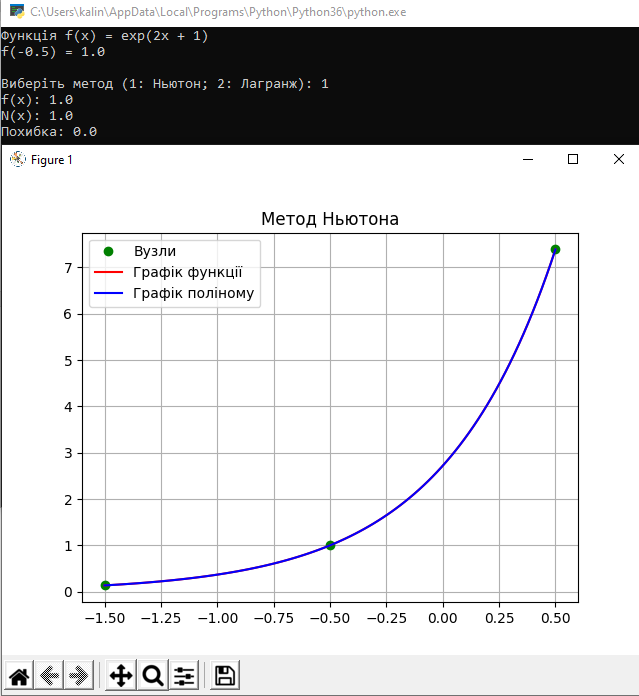


Рисунок 4

Далі, по завданню, необхідно було реалізувати програму, що дає вибір користувачеві, графік якого методу будувати: Ньютона чи Лагранжа, що і було зроблено та продемонстровано на рисунках 5-6 відповідно.

Рисунок 5

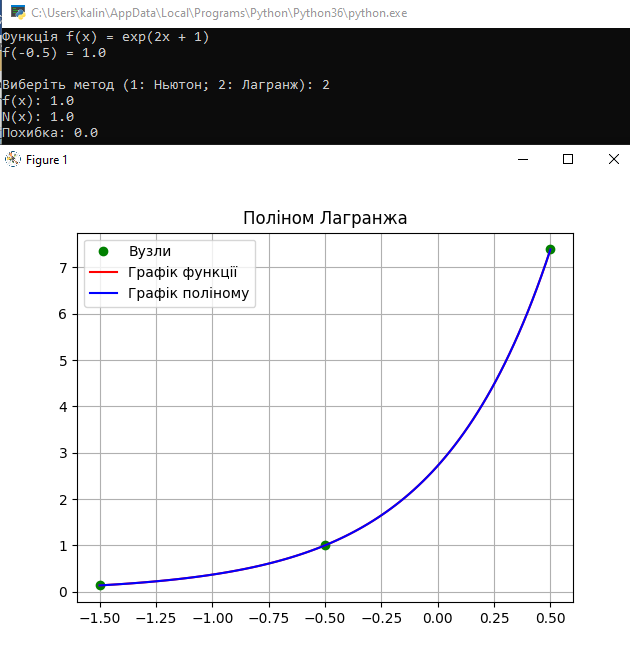


Рисунок 6

**ВИСНОВКИ**

При виконанні лабораторної роботи вивчено метод інтерполяції за схемою Ейткена та за формулами Лагранжа і Ньютона. Всі методи показали себе чудово: наближені значення функцій були настільки близькі до істинних, що Python не «побачив» різниці між ними і похибка становила 0.0 для всіх методів, з використанням всього трьох вузлів.

Було обчислено поліном Ньютона, який в шуканій точці також показав вірний результат.

**ЛІТЕРАТУРА**

1. Фельдман Л.П., Петренко А.І., Дмитрієва О.А. Чисельні методи в інформатиці. - К.: Видавнича група BHV, 2006. - 480 c.
2. Волков Е.А. Численные методы. – М.:Наука, 1982. – 256 с.

**Додаток**

**Інтерполяційна схема Ейткена**

import numpy as np

import matplotlib.pyplot as plt

def func(x):

return np.exp(2\*x + 1)

X = -0.5

n = 3

xi = np.linspace(X-1, X+1, n)

x\_and\_xi = X- xi

yi = func(xi)

polinoms = []

first\_nums\_in\_polinoms = []

def polinom\_calc(x, y, count):

cur\_polinom\_val = []

for i in range(len(x) - count):

cur\_polinom\_val.append((x[i] \* y[i + 1] - x[i + count] \* y[i]) / abs(x[i + count] - x[i]))

polinoms.append(cur\_polinom\_val)

first\_nums\_in\_polinoms.append(cur\_polinom\_val[0])

polinom\_calc(x\_and\_xi, yi, 1)

for i in range(n-2):

polinom\_calc(x\_and\_xi, polinoms[i], i+2)

difference = np.diff(first\_nums\_in\_polinoms)

difference = np.absolute(difference)

min\_value = difference.argmin()

print('Функція f(x) = exp(2x + 1)')

real\_val = func(X)

n\_val = first\_nums\_in\_polinoms[min\_value]

error = np.absolute(real\_val - n\_val)

print(f'f(x): {real\_val}\nE(x): {n\_val}\nПохибка: {error}\n')

**Алготим Ньютона**

import numpy as np

import matplotlib.pyplot as plt

from sympy import \*

def func(x):

return np.exp(2\*x + 1)

def Newton(func, x, n):

xi = np.linspace(x-1, x+1, n)

h = abs(xi[1] - xi[0])

yi = func(xi)

N = np.diff(yi, n=0)[0]

for i in range(1, n):

brackets = 1

for j in range(i):

brackets \*= (x - xi[j])

N += (np.diff(yi, n=i)[0] / np.math.factorial(i) / h\*\*i \* brackets)

return N

def poli\_info(func, X, n):

xi = np.linspace(X-1, X+1, n)

h = abs(xi[1] - xi[0])

yi = func(xi)

x = Symbol('x')

pol = np.diff(yi, n=0)[0]

for i in range(1, n):

brackets = 1

for j in range(i):

brackets \*= (x - xi[j])

pol += (np.diff(yi, n=i)[0] / np.math.factorial(i) / h\*\*i \* brackets)

pol = expand(pol)

print(f'N(x)={pol}')

print(f'N({X})={pol.subs(x, X)}')

if \_\_name\_\_ == '\_\_main\_\_':

X = -0.5

n = 3

real\_val = func(X)

n\_val = Newton(func, X, n)

error = np.absolute(real\_val - n\_val)

print('Функція f(x) = exp(2x + 1)')

print(f'f({X}): {real\_val}\nN({X}): {n\_val}\nПохибка: {error}\n')

poli\_info(func, X, n)

**Алгоритм Лагранжа**

import numpy as np

import matplotlib.pyplot as plt

from pylab import \*

def func(x):

return np.exp(2\*x + 1)

def Lagranzh(func, x, n):

xi = np.linspace(x-1, x+1, n)

yi = func(xi)

L = 0

for i in range(n):

Li = 1

for j in range(n):

if i == j:

continue

else:

Li = Li \* (x - xi[j]) / (xi[i] - xi[j])

L = L + (yi[i] \* Li)

return L

if \_\_name\_\_ =='\_\_main\_\_':

X = -0.5

n = 3

real\_val = func(X)

l\_val = Lagranzh(func, X, n)

error = np.absolute(real\_val - l\_val)

print('Функція f(x) = exp(2x + 1)')

print(f'f(x): {real\_val}\nL(x): {l\_val}\nПохибка: {error}\n')

**Задача за вибором та графіками**

import numpy as np

import matplotlib.pyplot as plt

from lagrange import Lagranzh

from newton import Newton

def func(x):

return np.exp(2\*x + 1)

X = -0.5

n = 3

real\_val = func(X)

print('Функція f(x) = exp(2x + 1)')

print(f'f({X}) = {real\_val}\n')

choice = input('Виберіть метод (1: Ньютон; 2: Лагранж): ')

if choice == '1':

n\_val = Newton(func, X, n)

error = np.absolute(real\_val - n\_val)

r\_error = error / real\_val

print(f'f(x): {real\_val}\nN(x): {n\_val}\nПохибка: {error}\n')

x\_graph\_func = np.linspace(X-1, X+1, 100)

y\_graph\_func = [func(x) for x in x\_graph\_func]

x\_graph\_newton = np.linspace(X-1, X+1, 100)

y\_graph\_newton = [Newton(func, x\_graph\_newton[i], n) for i in range(len(x\_graph\_newton))]

xi = np.linspace(X-1, X+1, n)

plt.plot(xi, func(xi), 'o', color="green")

plt.plot(x\_graph\_func, y\_graph\_func, color="red")

plt.plot(x\_graph\_newton, y\_graph\_newton, color="blue")

plt.title("Метод Ньютона")

plt.legend(('Вузли', 'Графік функції', 'Графік поліному'))

plt.grid(True)

plt.show()

elif choice == '2':

l\_val = Lagranzh(func, X, n)

error = np.absolute(real\_val - l\_val)

r\_error = error / real\_val

print(f'f(x): {real\_val}\nN(x): {l\_val}\nПохибка: {error}\n')

x\_graph\_func = np.linspace(X-1, X+1, 100)

y\_graph\_func = [func(x) for x in x\_graph\_func]

x\_graph\_pol = np.linspace(X-1, X+1, 100)

y\_graph\_pol = [Lagranzh(func, x\_graph\_pol[i], n) for i in range(len(x\_graph\_pol))]

xi = np.linspace(X-1, X+1, n)

plt.plot(xi, func(xi), 'o', color="green")

plt.plot(x\_graph\_func, y\_graph\_func, color="red")

plt.plot(x\_graph\_pol, y\_graph\_pol, color="blue")

plt.title("Поліном Лагранжа")

plt.legend(('Вузли', 'Графік функції', 'Графік поліному'))

plt.grid(True)

plt.show()