Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №2

Чисельні методи

Інтерполяція функцій

Варіант 17

Виконала
студентка групи ІТ-91:

Луцай К. А.

Перевірила:

ас. Тимофєєва Ю. С.

Київ 2021

Мета: навчитися інтерполяції функцій за допомогою мови програмування Python, використати два методи (Лангранжа, Ньютона).

Теоретичні відомості:

У теорії наближень вивчаються методи наближення функцій більш простими, добре вивченими функціями, методи чисельного диференціювання та чисельного інтегрування. При цьому досліджувана наближувана функція може бути задана як в аналітичному, так і дискретному виді (у вигляді експериментальної таблиці).

Нехай дано деяку функцію f (x) на відрізку x∈[a,b], що є досить складною для досліджування. Потрібно замінити цю функцію деякою простою, але добре досліджуваною функцією (наприклад, многочленом). Для цього за допомогою f (x) будують таблицю (її називають сітковою функцією), яку можна замінити (згладити) простою функцією з контрольованою похибкою.

x_i	x_0	x_1	...	x_n
y_i	y_0	y_1	...	y_n

Інтерполяційний поліном Лагранжа

Для поліноміальної інтерполяції можна скласти многочлен у такий спосіб:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}.$$

Інтерполяційний поліном Ньютона

Щоб побудувати інтерполяційний поліном у формі Ньютона, використовується поняття поділеної різниці, що являє собою аналог поняття похідної стосовно сіткових функцій.

$$f(x_0, x_1, ..., x_n) = \frac{f(x_1, ..., x_n) - f(x_0, ..., x_{n-1})}{x_n - x_0}.$$

$$N_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) +$$
$$+ ... + f(x_0, x_1, ..., x_n)(x - x_0)(x - x_1)...(x - x_{n-1}).$$

Верхня оцінка похибки поліноміальної інтерполяції на відрізку [a,b]:

$$\left| f(\bar{x}) - L_n(\bar{x}) \right| \le \frac{\max\limits_{x \in [a,b]} \left| f^{(n+1)}(x) \right|}{(n+1)!} \left| (\bar{x} - x_0)(\bar{x} - x_1)...(\bar{x} - x_n) \right|$$

Розв’язок задачі в аналітичній формі:

$f(x) = e^x + x$

X = 0.5

x_i	-2	-1	0	1
y_i	$\frac{1}{e^2} - 2$	$\frac{1}{e} - 1$	1	e+1

$$L_3(0.5) = \frac{1}{16}\left(\frac{1}{e^2} - \frac{5}{e} + \frac{5e}{16} + 2\frac{1}{16}\right)$$

$$N_3(0.5) = \frac{1}{16}\left(\frac{1}{e^2} - \frac{5}{16e} + \frac{5e}{16} + \frac{23}{16}\right)$$

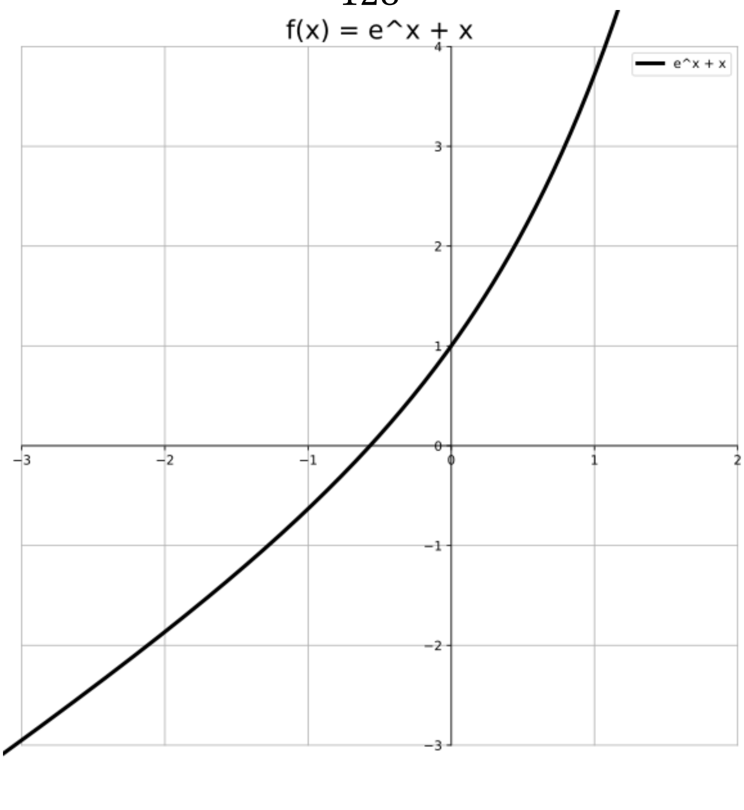
x_i	-2	-1	0.2	1
y_i	$\frac{1}{e^2} - 2$	$\frac{1}{e} - 1$	$\sqrt[5]{e} + 0.2$	e+1

$$L_3(0.5) = -\frac{3}{88e^2} - \frac{5}{32e} + \frac{15e}{64} + \frac{1875\sqrt[5]{e}}{2112}$$

$$N_3(0.5) = \frac{15}{8e^2} - \frac{27}{8e} + \frac{5e}{12} + \frac{73\sqrt[5]{e}}{24} + \frac{1}{2}$$

Верхня оцінка похибки:

$$|f(0.5) - L_3(0.5)| \leq \frac{5e}{128}$$



Лістинги програм розв’язування задачі:

Методи для інтерполяції функції на основі масивів даних (x координати точок, y координати точок) для обраної точки x:

```
def lagranj(dipX, dipY, x):
    z = 0
    for j in range(len(dipY)):
        p1 = 1
        p2 = 1
        for i in range(len(dipX)):
            if i == j:
                p1 *= 1
                p2 *= 1
            else:
                p1 *= x - dipX[i]
                p2 *= dipX[j] - dipX[i]
        z += dipY[j] * p1 / p2
    return z

def coef(X):
    x = np.array(X, dtype=float)
    y = np.array([func17(x) for x in X], dtype=float)
    m = len(x)
    a = np.copy(y)
    for k in range(1, m):
        a[k:m] = (a[k:m] - a[k - 1])/(x[k:m] - x[k - 1])
    return a

def evalN(a, dipX, x):
    X = np.array(dipX, dtype=float)
    n = len(a) - 1
    y = a[n]
    for i in range(1, n+1):
        y = a[n-i] + y * (x - X[n-i])
    return y
```

Функції для виклику цих методів та графічного представлення результатів. Вхідними даними є масив x координат та колір бажаного графіку, генерація масиву y координат відбувається за відомою функцією після виклику.

```
def lagranDraw(X, clr):
    x = np.array(X, dtype=float)
    y = np.array([func17(x) for x in X], dtype=float)

    newX = np.linspace(np.min(x), np.max(x), 1000)
    newY = [lagranj(x, y, i) for i in newX]

    fig = plt.figure(figsize=(10, 10))
    ax = plt.subplot(ylim=(-3, 4), xlim=(-3, 2))
    plt.axvline(tX, color="yellow")
    ax.plot(tX, lagranj(x, y, tX), "yo", markersize=10)
    ax.plot(x, y, clr[0] + "o")
```

```
    ax.plot(newX, newY, color=clr, lw=3)
    ax.grid(True)
    ax.spines["left"].set_position("zero")
    ax.spines["right"].set_color("none")
    ax.spines["bottom"].set_position("zero")
    ax.spines["top"].set_color("none")
    plt.title("Лагранжа " + str(X), fontsize=20)
    plt.show()

def newtonDraw(X, clr):
    a = coef(X)
    newX = np.linspace(np.min(X), np.max(X), 1000)
    newY = [evalN(a, X, i) for i in newX]

    fig = plt.figure(figsize=(10, 10))
    ax = plt.subplot(ylim=(-3, 4), xlim=(-3, 2))
    plt.axvline(tX, color="yellow")
    ax.plot(tX, evalN(a, X, tX), "yo", markersize=10)
    ax.plot(X, [func17(x) for x in X], clr[0] + "o")
    ax.plot(newX, newY, color=clr, lw=3)
    ax.grid(True)
    ax.spines["left"].set_position("zero")
    ax.spines["right"].set_color("none")
    ax.spines["bottom"].set_position("zero")
    ax.spines["top"].set_color("none")
    plt.title("Ньютона " + str(X), fontsize=20)
    plt.show()
```

Задані величини: функція, два масиви тестових координат, точка у якій потрібно обрахувати похибку інтерполяцій.

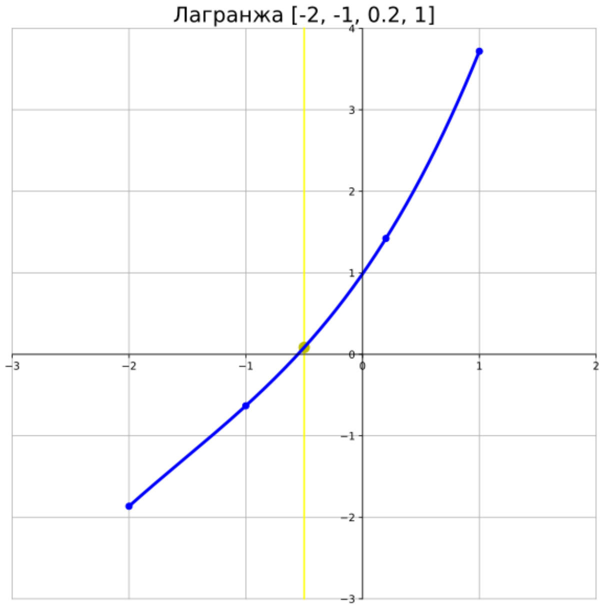
```
import matplotlib.pyplot as plt
import math
import numpy as np

def func17(x):
    return math.exp(x) + x

dipX1 = [-2, -1, 0, 1]
dipX2 = [-2, -1, 0.2, 1]
tX = -0.5
```

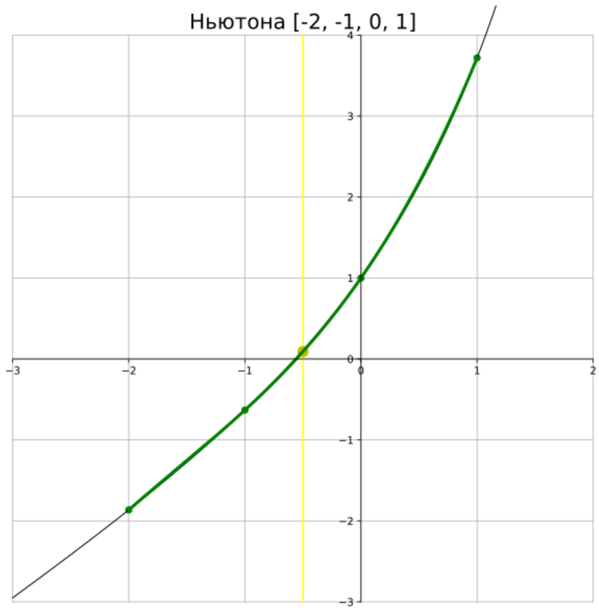
Виклик функцій для графічного представлення результатів та їх порівняння:

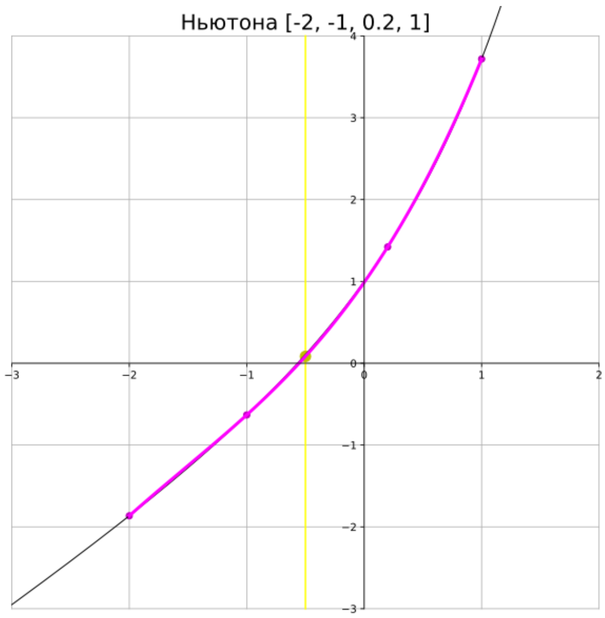
```
lagranDraw(dipX1, "red")
lagranDraw(dipX2, "blue")

newtonDraw(dipX1, "green")
newtonDraw(dipX2, "magenta")

data = [func17(tX)]
print(data[0], "за функцією")
for dip in [dipX1, dipX2]:
    y = lagranj(dip, [func17(x) for x in dip], tX)
```

```python
    data.append(y)
    print(y, "за Лагранжа з похибкою", math.fabs(func17(tX)-y))

for dip in [dipX1, dipX2]:
    a = coef(dip)
    y = evalN(a, dip, tX)
    data.append(y)
    print(y, "за Ньютона з похибкою", math.fabs(func17(tX)-y))

x = np.linspace(-10, 10, 1000)
y = np.exp(x) + x
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(ylim=(0, 1), xlim=(-1, 0))
clr = "yrbgm"
names = ["Оригінал", "Лагранж X1", "Лагранж X2", "Ньютон X1", "Ньютон X2"]
plt.axvline(tX, color="yellow")
for i, Y in enumerate(data):
    ax.plot(tX, Y, clr[i] + "o", label = str(Y) + " " + names[i])
ax.legend(fontsize=16)
ax.grid(True)
ax.spines["left"].set_position("zero")
ax.spines["right"].set_color("none")
ax.spines["bottom"].set_position("zero")
ax.spines["top"].set_color("none")
plt.show()

x = np.linspace(-10, 10, 1000)
y = np.exp(x) + x
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(ylim=(-3, 4), xlim=(-3, 2))
ax.plot(x, y, color="black", label="e^x + x", lw=3)
ax.legend()
ax.grid(True)
ax.spines["left"].set_position("zero")
ax.spines["right"].set_color("none")
ax.spines["bottom"].set_position("zero")
ax.spines["top"].set_color("none")
plt.title("f(x) = e^x + x", fontsize=20)
plt.show()
```

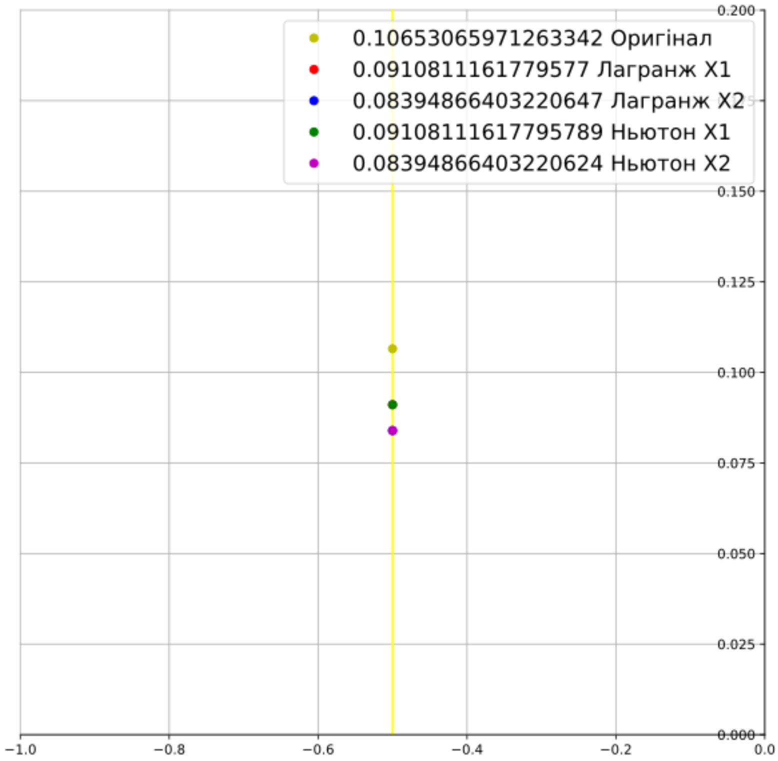
Результати виконання програм:








```
0.10653065971263342 за функцією
0.0910811161779577 за Лагранжа з похибкою 0.01544954353467573
0.08394866403220647 за Лагранжа з похибкою 0.02258199568042696
0.09108111617795789 за Ньютона з похибкою 0.015449543534675536
0.08394866403220624 за Ньютона з похибкою 0.02258199568042718
```



Висновки: було написано програму для інтерполяції функцій на мові Python: було розроблено метод Лагранжа та Ньютона для, було розроблено функції для графічного представлення отриманих многочленів.