МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №2

з дисципліни: «Чисельні методи»

16 Варіант

СПЕЦІАЛЬНОСТІ 121 – Інженерія програмного забезпечення

Виконав: Левак О. О.

Група: ІТ-91

Перевірила: Тимофєєва Ю.С.

Київ 2020

**Лабораторна робота №2. Інтерполяція функцій**

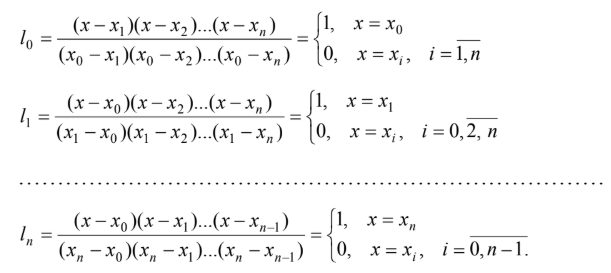
**Мета:** навчитися інтерполяції функцій, використовуючи поліноми Лагранжа та Ньютона; знаходити похибку обчислень та визначати вигідне розміщення інтерполяційних вузлів.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

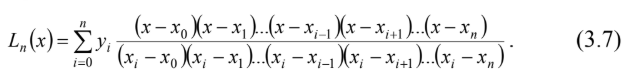
У теорії наближень вивчаються методи наближення функцій більш простими, добре вивченими функціями, методи чисельного диференціювання та чисельного інтегрування. При цьому досліджувана наближувана функція може бути задана як в аналітичному, так і дискретному виді (у вигляді експериментальної таблиці).

**Інтерполяційний поліном Лагранжа:** для поліноміальної інтерполяції можна не розв’язувати СЛАР, а скласти многочлен у такий спосіб:

запишемо систему поліномів n-го степеня



Складемо лінійну комбінацію цих поліномів (їхня кількість дорівнює n+1) з коефіцієнтами лінійної комбінації, рівними значенням зi сіткової функції, отримаємо многочлен n-го степеня:



Поліном (3.7) називають інтерполяційним поліномом Лагранжа n-го степеня, тому що він, по-перше, задовольняє умові інтерполяції

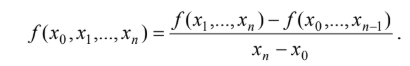


і, по-друге, має n-й степінь.

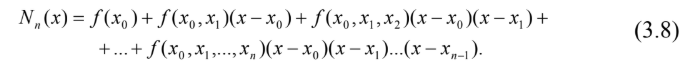
Інтерполяційний многочлен Лагранжа має таку ваду, що у разі, коли додаються нові вузли інтерполяції в таблиці, всі доданки в (3.7) необхідно перераховувати.

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

Поділеною різницею функції n-го порядку у вузлі x0 називають відношення



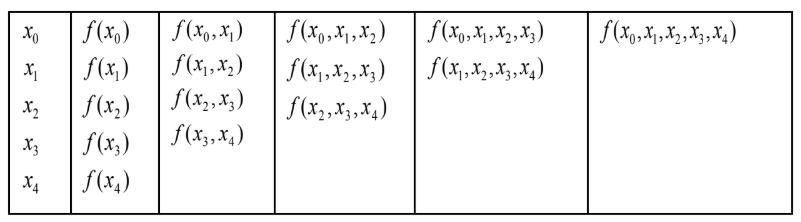
Використовуючи поділені різниці, інтерполяційний поліном Ньютона можна записати у такій формі:



Зазначимо, що коли додаються нові вузли, перші члени многочлена Ньютона залишаються незмінними.

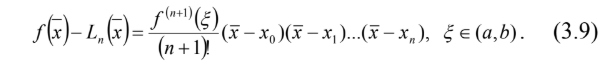
Якщо функцію задано в точках x1, x2, ... , xn, то при побудові інтерполяційного многочлена Ньютона зручно користуватися таблицею, називаною таблицею поділених різниць, приклад якої для n = 4 наведено у табл. 3.1.

Таблиця 3.1.



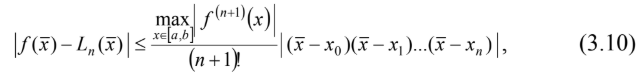
Щоб підвищити точність інтерполяції, в суму (3.8) можна додавати нові члени, для чого необхідно включити додаткові інтерполяційні вузли. При цьому байдуже, у якому порядку додаються нові вузли. Цим формула Ньютона вигідно відрізняється від формули Лагранжа.

**Похибка поліноміальної інтерполяції:** що представляє собою різницю між значенням інтерполяційного многочлена L (x) n і значенням функції f (x) в точці x , що не збігається з вузлом інтерполяції має вигляд:

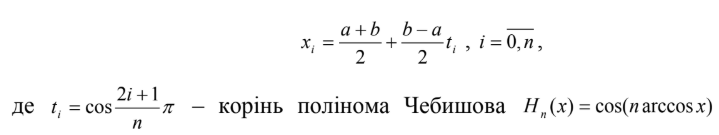


 – точка, у якій шукається похибка (не збігається з вузлами інтерполяції).

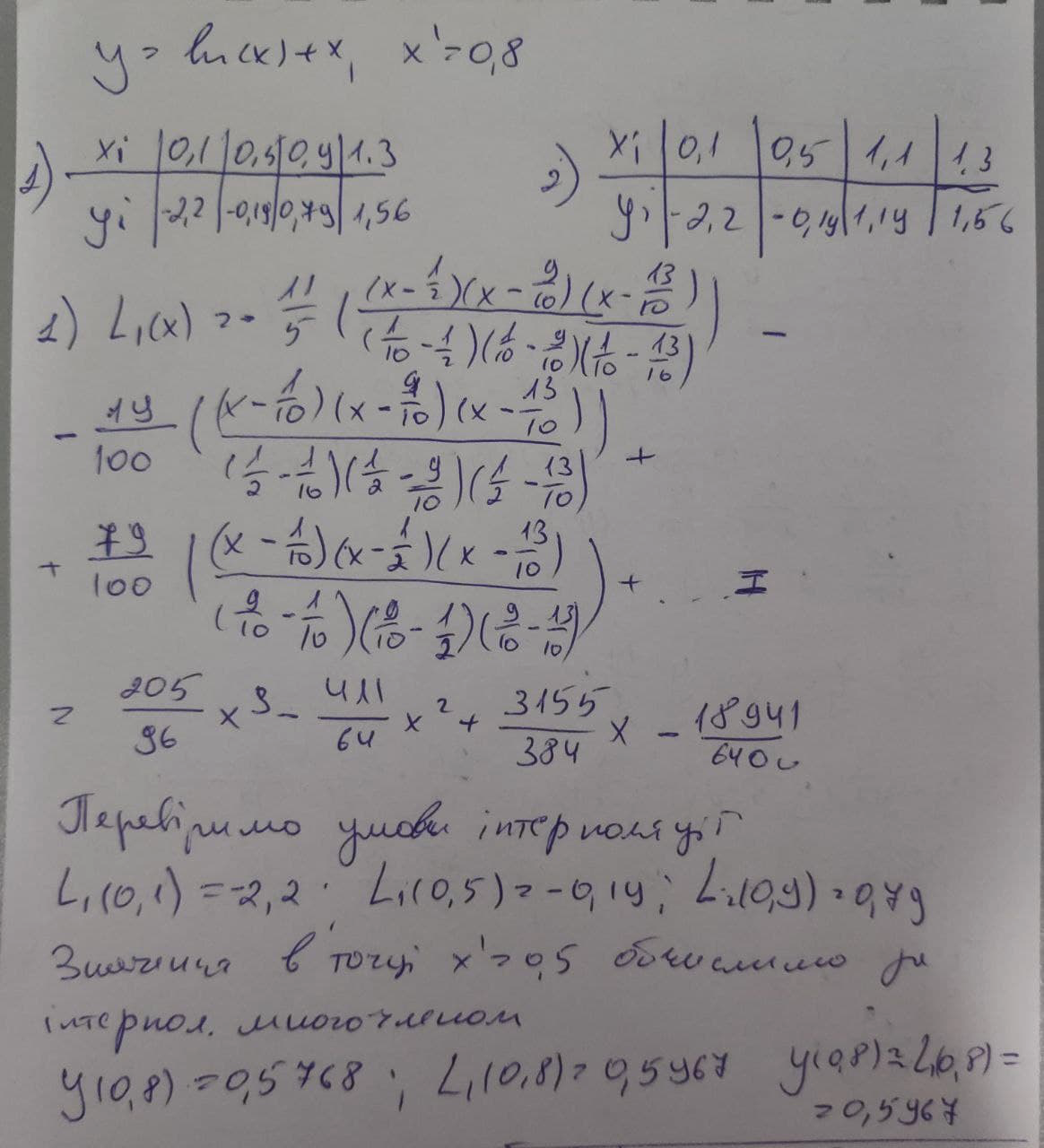
Оскільки точка ξ ∈ (a,b) невідома, то замість похибки (3.9) вводиться верхня оцінка похибки у вигляді:

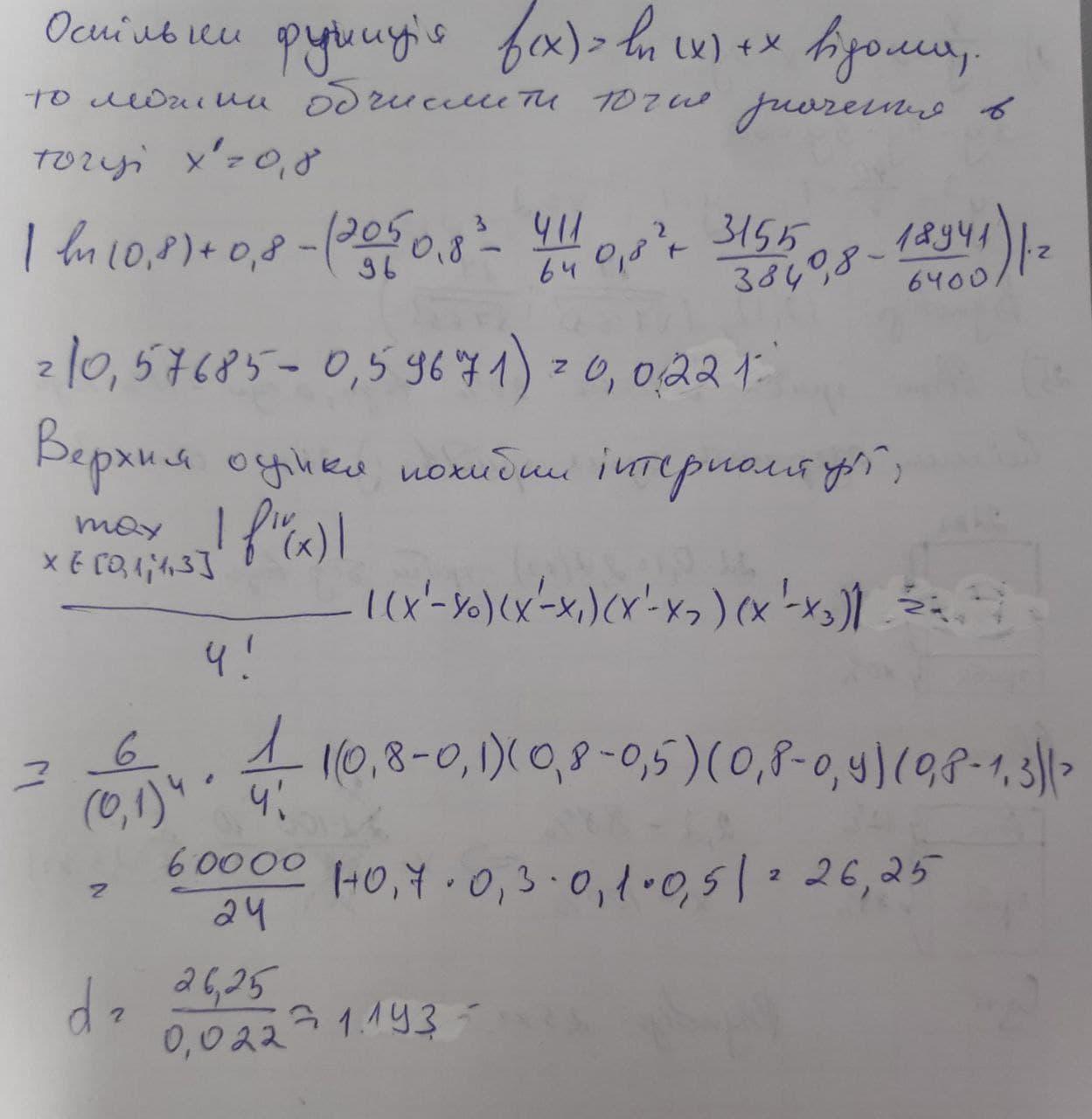


Таким чином, похибка інтерполяції залежить як від величини відповідної похідної наближуваної функції, так і від розташування вузлів. Мінімізувати похибку наближення досить гладкої функції на відрізку [a,b] поліномом степеня n можна, розташувавши вузли інтерполяції в такий спосіб:



**РОЗВ’ЯЗОК ЗАДАЧІ В АНАЛІТИЧНІЙ ФОРМІ**





**ЛІСТИНГ ПРОГРАМИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ**

**import numpy as np**

**import math**

**import matplotlib.pyplot as plt**

**def yValues(xValues, func):**

**return [func(xVal) for xVal in xValues]**

**def lagranz(x, f, xVal):**

**y = yValues(x, f)**

**result = 0**

**for j in range(len(y)):**

**p = 1**

**for i in range(len(x)):**

**if i != j:**

**p \*= (xVal - x[i])**

**p /= (x[j] - x[i])**

**result = result + y[j] \* p**

**return result**

**def nevton(x, f, xVal):**

**y = yValues(x, f)**

**delta = y.copy()**

**n = len(y)**

**for j in range(1, n):**

**for i in range(n - 1, j - 1, -1):**

**delta[i] = (delta[i] - delta[i - 1]) / (x[i] - x[i - j])**

**result = 0**

**for i in range(0, n):**

**count = delta[i]**

**for j in range(0, i):**

**count \*= (xVal - x[j])**

**result += count**

**return result**

**def mistake(xVal, deffRes, f, polinom, x):**

**polMistake = (deffRes / math.factorial(len(x)))**

**for xV in x:**

**polMistake \*= xVal - xV**

**polMistake = abs(polMistake)**

**absPolError = abs(f(xVal) - polinom(x, f, xVal))**

**print("Верхня оцінка похибк", polMistake)**

**print("Абсолютна похибка", absPolError)**

**return polMistake / absPolError**

**xVal = 0.8**

**x1 = [0.1, 0.5, 0.9, 1.3]**

**x2 = [0.1, 0.5, 1.1, 1.3]**

**def f(x):**

**return np.log(x) + x**

**print("Ньютон", nevton(x1, f, xVal))**

**print("Лагранж", lagranz(x1, f, xVal))**

**print("Співвідношення", mistake(xVal, 60000, f, nevton, x1))**

**plotX = [x \* 0.1 for x in range(1, 10)]**

**plotNevton = [nevton(x1, f, x) for x in plotX]**

**plotLagranz = [lagranz(x1, f, x) for x in plotX]**

**plotStart = [f(x) for x in plotX]**

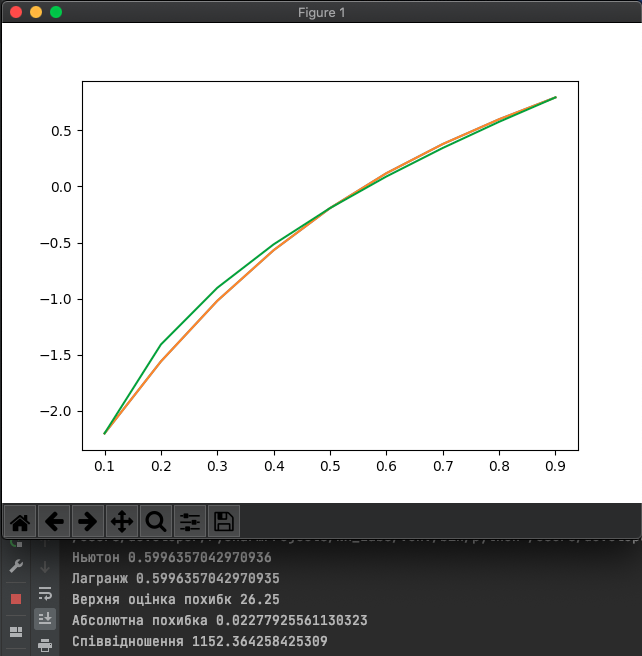
**plt.plot(plotX, plotNevton)**

**plt.plot(plotX, plotLagranz)**

**plt.plot(plotX, plotStart)**

**plt.show()**

**РЕЗУЛЬТАТИ ВИКОНАННЯ ПРОГРАМИ**

****

**Висновки :**

В ході лабораторної роботи ми навчилися інтерполяції функцій, використовуючи поліноми Лагранжа та Ньютона; знаходити похибку обчислень та визначати вигідне розміщення інтерполяційних вузлів..