НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №2

із дисципліни «Методи обчислення»

на тему

«Інтерполяція слайнами»

2 варіант

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-81 | *викладач Костюшко І.* |
| Верзун П.В. |  |

Київ — 2020

**ВСТУП**

Мета роботи – здобути практичні навички розробки алгоритмів та програм, які здійснюють інтерполювання кубічними поліномами. Для виконання роботи була використана мова програмування Python ті бібліотеки numpy та matploltib, для спрощення розрахунків та візуалізації даних, відповідно.

**ЗАВДАННЯ 1**

**Завдання 1.** Побудувати кубічний сплайн для таблично заданої функції, з граничними умовами для першої похідної.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | -2 | -1 | 1 | 3 | Sa=2  Sb=4 |
| y | 2 | 1 | 3 | 1 |

Для побудови кубічного сплайма потрібно на кожному з відрізків побудувати сплайм наступного вигляду:

Для знахождення

Для .

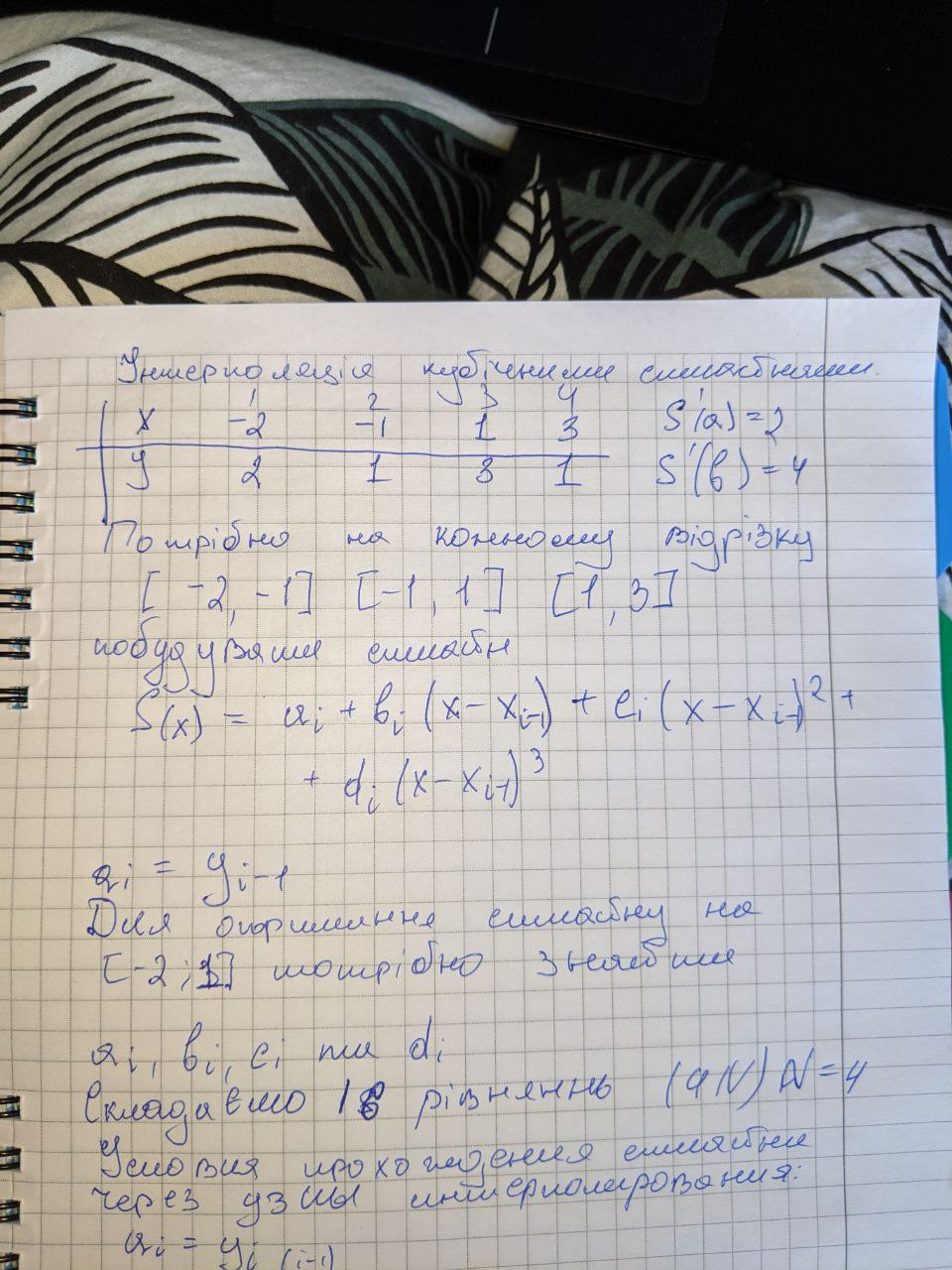
Для визначення використовуються наступні формули :

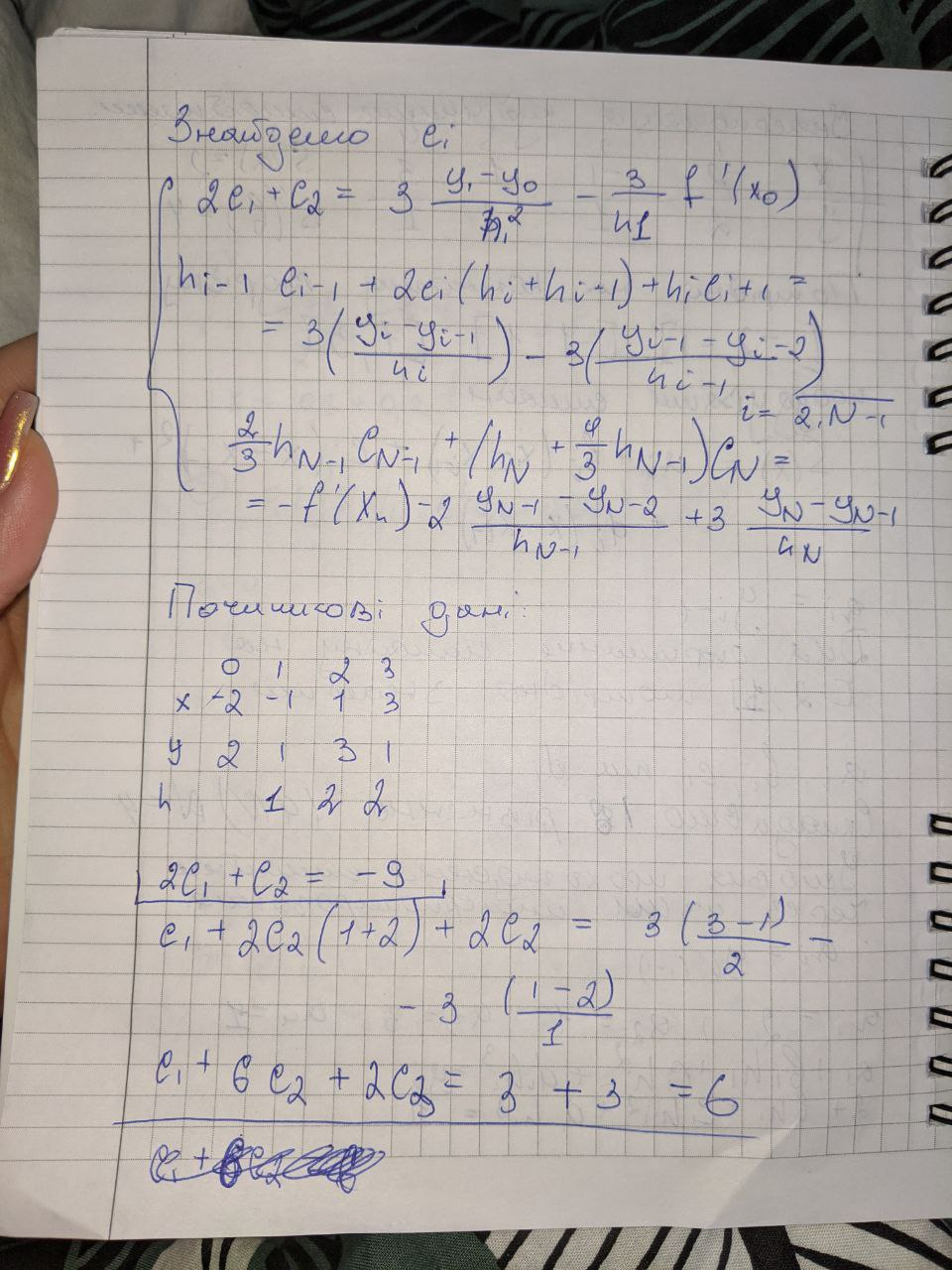
Для визначення :

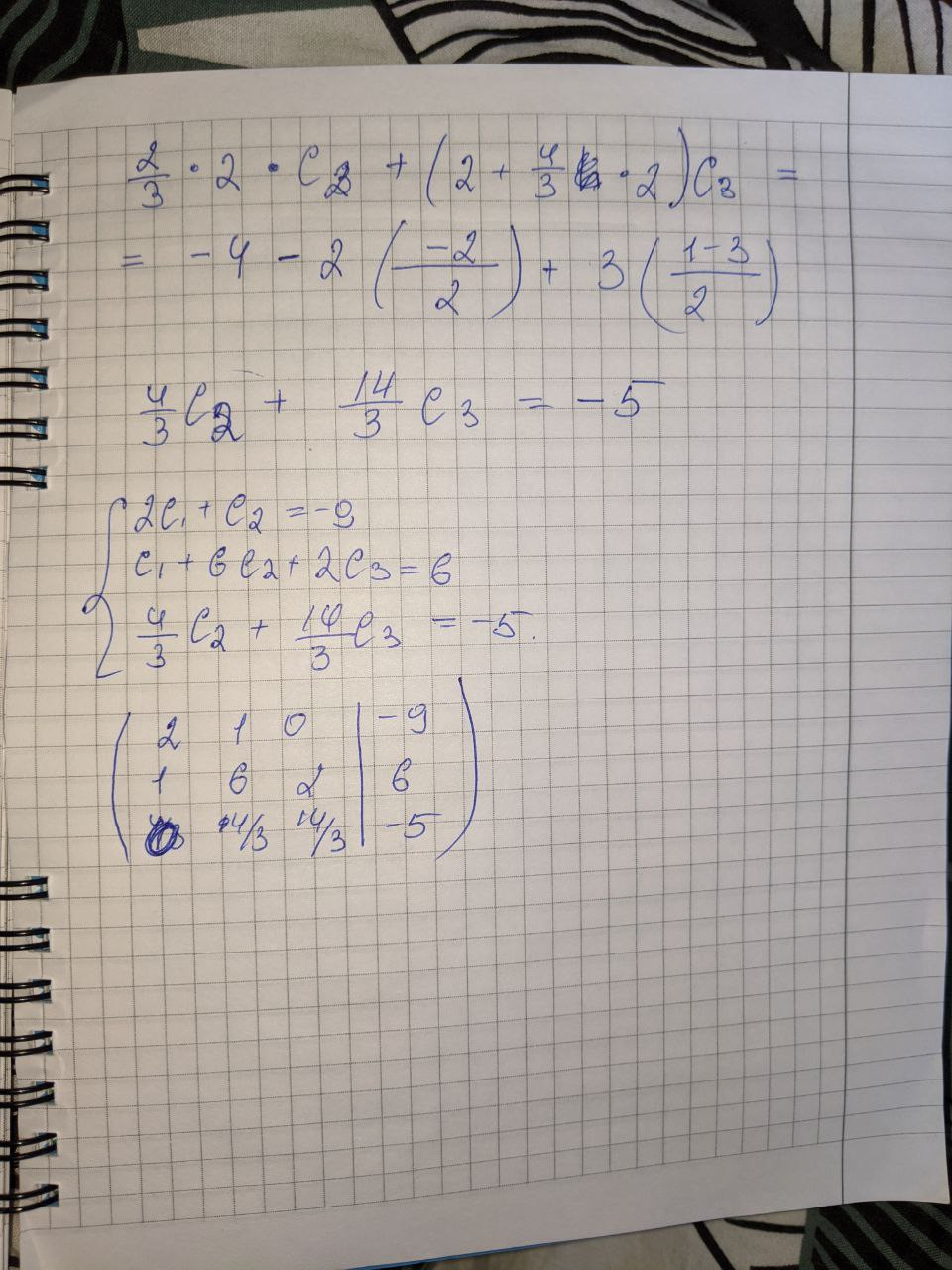
Також слід зауважити , що при першій граничній умові .

Тоді для обчислення :

Далі наведена частина аналіточного вирішення



****

****

**Далі наведено код та скріншот виконання програми:**

****

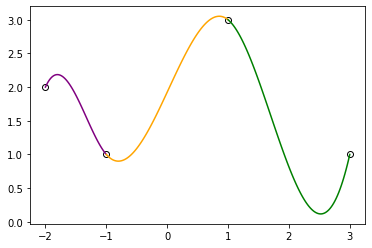


Результат:

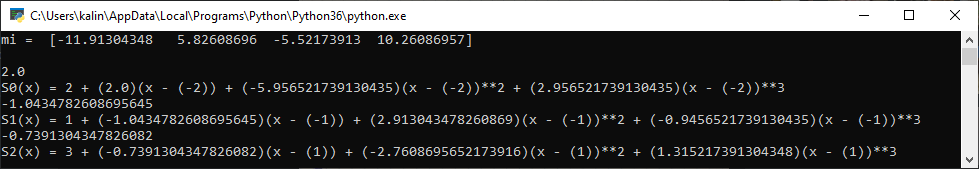
Коефіцієнти с: [-5.95652174 2.91304348 -2.76086957]

Коефіцієнти b: [ 2. -1.04347826 -0.73913043]

Коефіцієнти d: [ 2.95652174 -0.94565217 1.31521739]



**Отримані сплайни:**

****

**ЗАВДАННЯ 2**

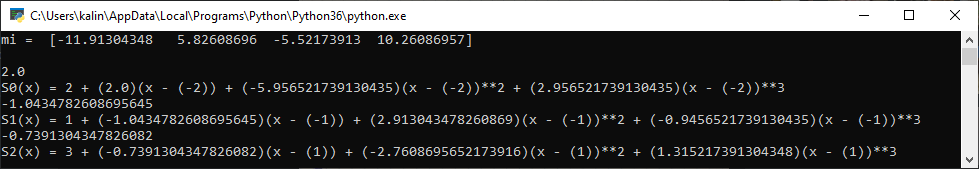
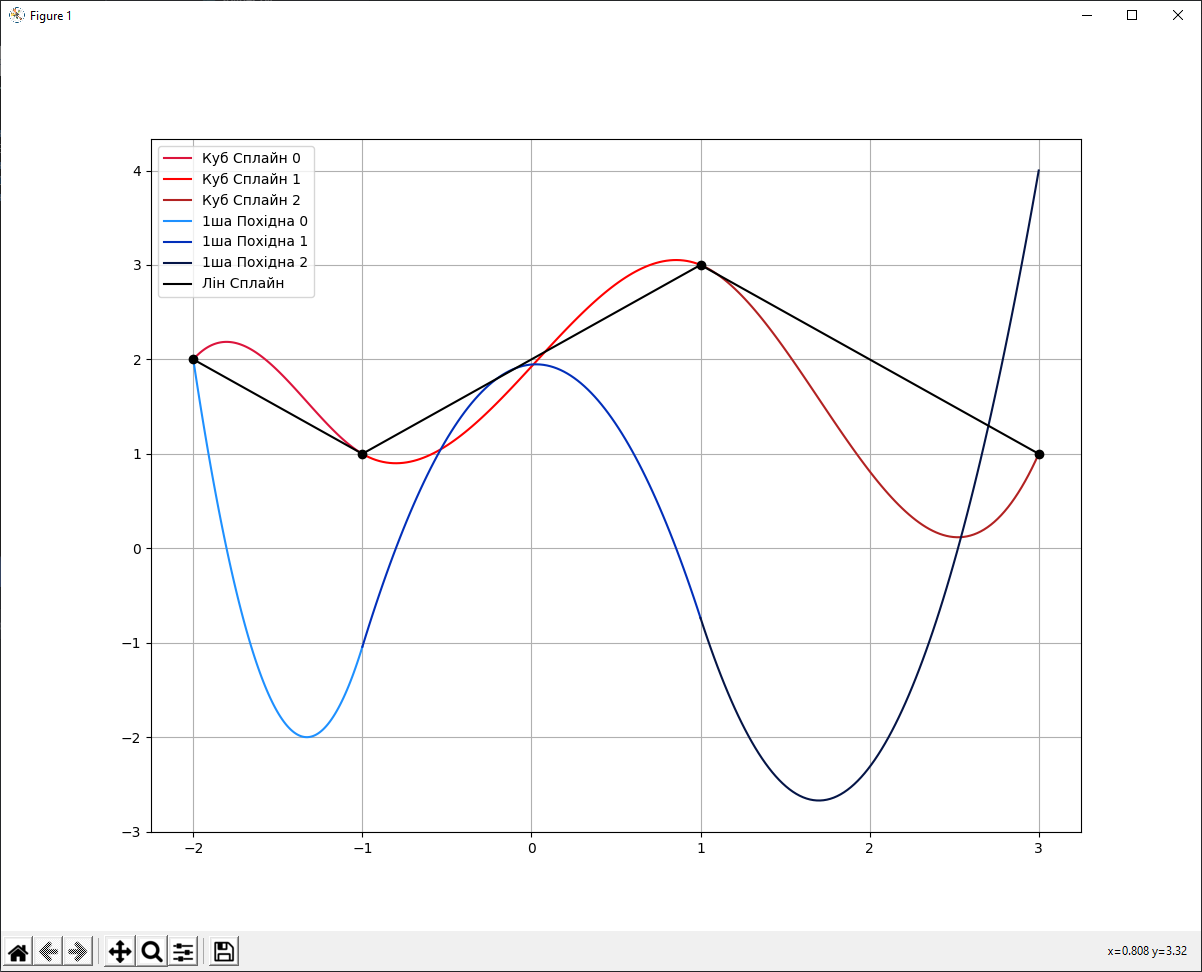
На рисунку 4 приведені програмно знайдені mi та сплайни. 

Рисунок 1

**ЗАВДАННЯ 3**

Значення mi , що були обчислені раніше, будуть використані в даному завданні. Ідея проста: розбити кожен інтервал на 100 підінтервалів та знайти значення кубічного сплайну та його першої похідної. Після необхідних обчислень виводимо результат на графік (рис.5).



Як можна побачити на графіку – лінійний сплайн не найкращим чином відтворює функцію. Кубічний, в свою чергу, справляється зі своєю роботою добре. Необхідно відмітити, що графік похідної починається та закінчується у точках з вихідних даних, що підтверджує правильність обчислень.

**ВИСНОВКИ**

В ході даної роботи був розглянута інтерполяція сплайнами, а саме: кубічним та лінійним. Очевидно, до кубічний сплайн несе в собі більше корисної інформацію про функцію, що інтерполюють, аніж лінійний. Це було продемонстровано на графіку. До того ж даний приклад був розв’язаний «вручну», тобто аналітично. Обчислення не були настільки ж точними через округлення, проте коефіцієнти, все ж, близькі до програмно обчислених.

**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 А**

from pylab import \*

xi = [-2, -1, 1, 3]

yi = [2, 1, 3, 1]

n = len(xi) - 1

A = 2

B = 4

hi = np.diff(xi)

di = np.divide(np.diff(yi), hi)

List\_x\_points= []

List\_x\_points\_derivatives = []

List\_Si = []

List\_Si\_derivatives = []

system = np.array([[1, 0.5, 0, 0],

[hi[0], 2 \* (hi[0] + hi[1]), hi[1], 0],

[0, hi[1], 2 \* (hi[1] + hi[2]), hi[2]],

[0, 0, 0.5, 1]]

)

answers = np.array([3 / hi[0] \* (di[0] - A),

6 \* (di[1] - di[0]),

6 \* (di[2] - di[1]),

3 / hi[2] \* (B - di[2])]

)

result = np.linalg.solve(system, answers)

print('mi = ',result,'\n')

for k in range(0, len(xi)-1):

s\_k\_0 = yi[k]

s\_k\_1 = di[k] - hi[k]\*(2\*result[k] + result[k + 1])

print(s\_k\_1)

s\_k\_2 = result[k] / 2

s\_k\_3 = (result[k + 1] - result[k]) / (6 \* hi[k])

spline = f'S{k}(x) = {s\_k\_0} + ({s\_k\_1})(x - ({xi[k]})) + ({s\_k\_2})(x - ({xi[k]}))\*\*2 + ({s\_k\_3})(x - ({xi[k]}))\*\*3'

print(spline)

for i in range(n):

x\_points = np.linspace(xi[i], xi[i+1], 100)

Si = []

Si\_derivative = []

Si\_second\_derivatives = []

for j in range(100):

Si.append(yi[i] + (di[i] - (hi[i] \* (2 \* result[i] + result[i+1]) / 6)) \* (x\_points[j] - xi[i]) + # Кубічний

(result[i] / 2) \* math.pow((x\_points[j] - xi[i]), 2) +

((result[i+1] - result[i]) / 6 / hi[i]) \* (x\_points[j] - xi[i])\*\*3)

Si\_derivative.append((di[i] - (hi[i] \* (2 \* result[i] + result[i + 1]) / 6)) + # Похідна кубічного поліному

2 \* (result[i] / 2) \* (x\_points[j] - xi[i]) +

3 \* ((result[i + 1] - result[i]) / 6 / hi[i]) \* (x\_points[j] - xi[i])\*\*2)

List\_Si.append(Si)

List\_x\_points.append(x\_points)

List\_Si\_derivatives.append(Si\_derivative)

List\_x\_points\_derivatives.append(x\_points)

plt.figure(1, (12, 9))

plt.plot(List\_x\_points[0], List\_Si[0], color="#DC143C", label='Куб Сплайн 0')

plt.plot(List\_x\_points[1], List\_Si[1], color="#FF0000", label='Куб Сплайн 1')

plt.plot(List\_x\_points[2], List\_Si[2], color="#B22222", label='Куб Сплайн 2')

plt.plot(List\_x\_points\_derivatives[0], List\_Si\_derivatives[0], color="#1E90FF", label='1ша Похідна 0')

plt.plot(List\_x\_points\_derivatives[1], List\_Si\_derivatives[1], color="#022fba", label='1ша Похідна 1')

plt.plot(List\_x\_points\_derivatives[2], List\_Si\_derivatives[2], color="#051547", label='1ша Похідна 2')

plt.plot(xi, yi, 'o', color="black")

plt.plot(xi, yi, color="black", label='Лін Сплайн')

plt.legend()

plt.grid(True)

plt.show()