****

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 №4**  
Метод сплайн-інтерполяції функцій

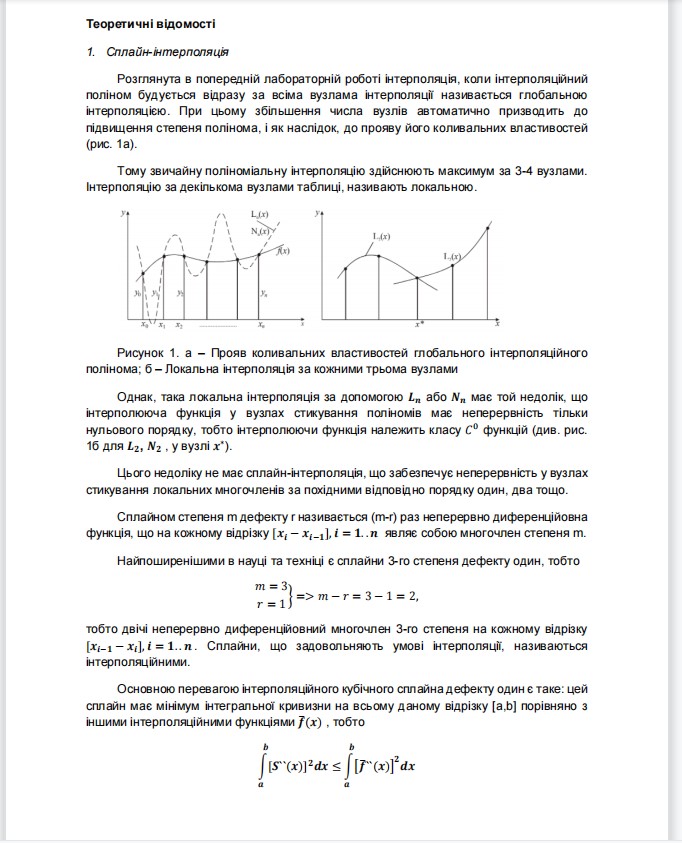
Виконав: Перевірила:

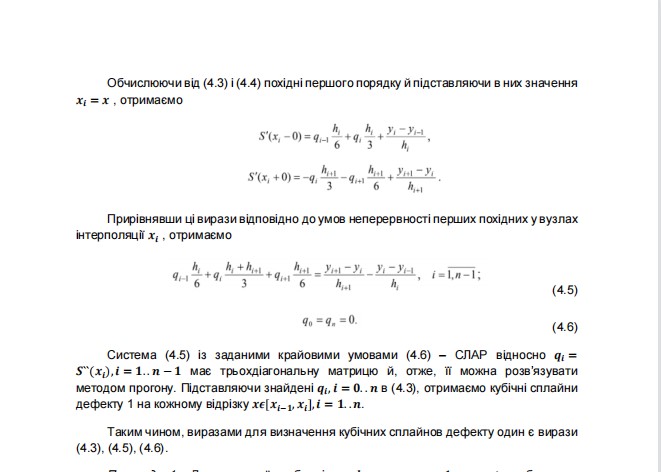
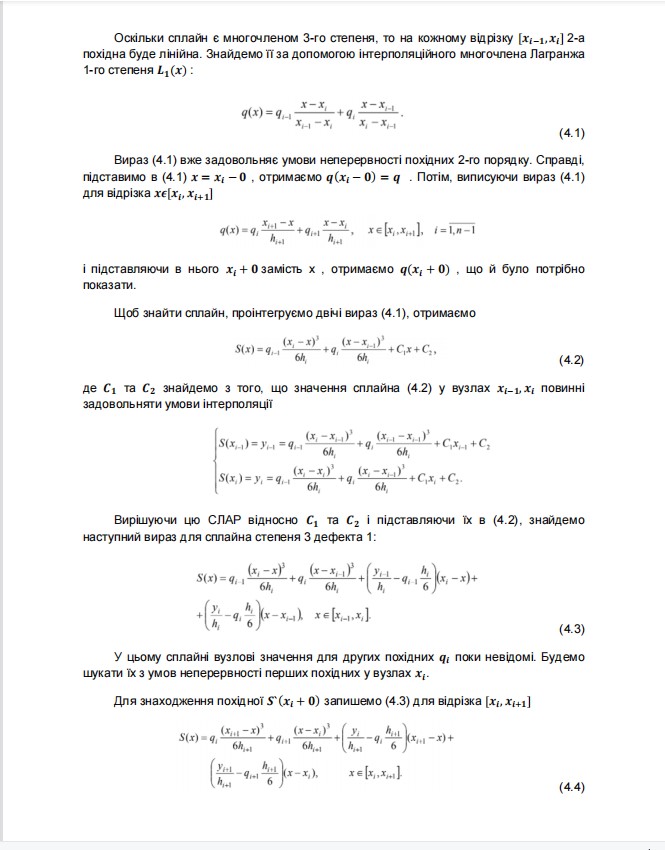
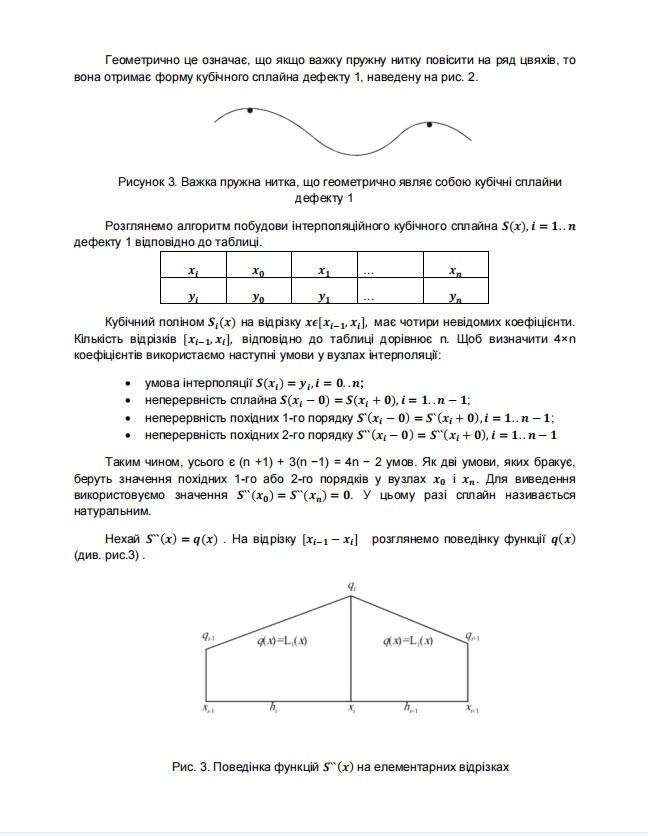
студент групи ІА – 11: Вітюк А.Є.

Іванніков Максим Тимурович

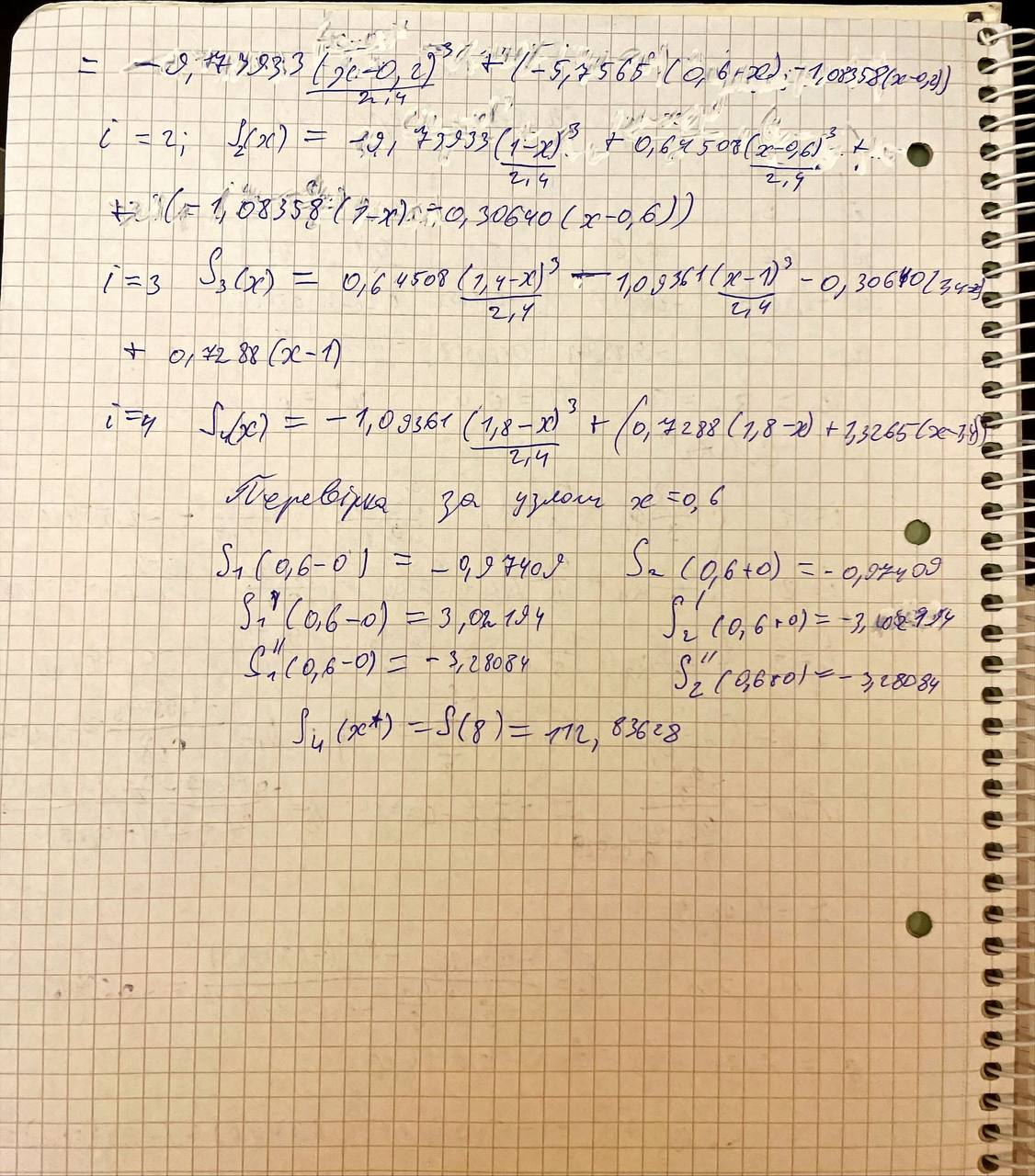
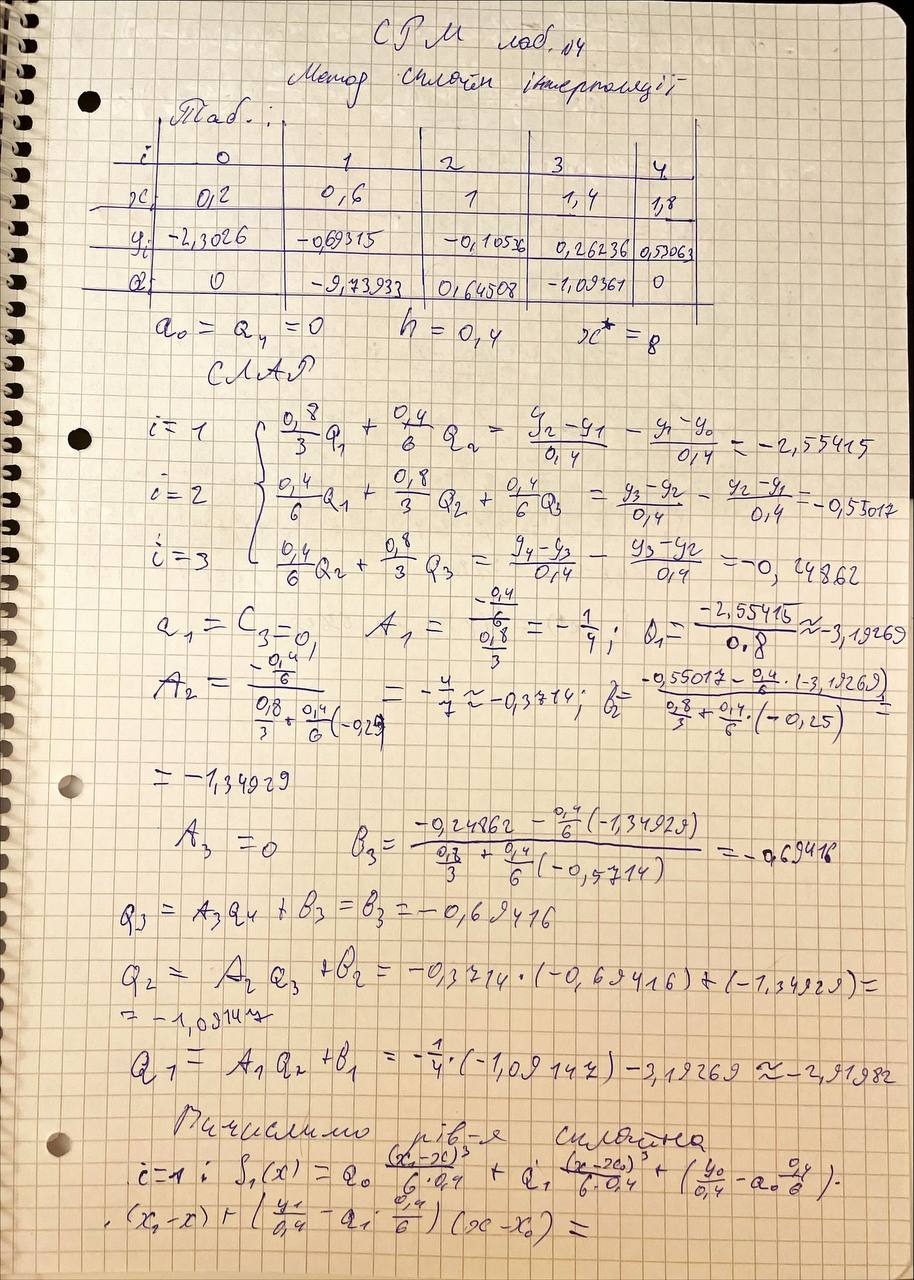
Київ 2021

**Мета роботи:**ознайомитись з методологією сплайн-інтерполювання функцій, зокрема побудови кубічного сплайну.

**Теорія:**

****

**Аналітичний розв’язок:**

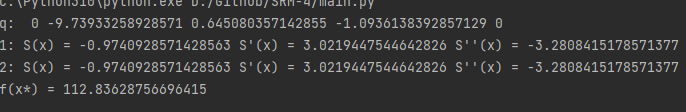
****

**Програмний код:**

from scipy.misc import derivative  
  
  
  
def SLARkoefs(f\_list, i, h):  
 lst = [h/6, 2 \* h / 3, h / 6,  
 (f\_list[i+1] - f\_list[i]) / h - (f\_list[i] - f\_list[i-1]) / h  
 ]  
 if i == 1 or i == 3:  
 lst[i-1] = 0  
 return lst  
  
  
def q\_koefs(f\_list, i, h):  
 lst = []  
 part = SLARkoefs(f\_list, i, h)  
 if i == 1:  
 lst.append( (-part[2]) / part[1] )  
 lst.append( part[3] / part[1] )  
 return lst  
 recursive\_part = q\_koefs(f\_list, i - 1, h)  
 lst.append(  
 (-part[2]) / (part[1] + part[0] \* recursive\_part[0])  
 )  
 lst.append(  
 (part[3] - part[0] \* recursive\_part[1]) / (part[1] + part[0] \* recursive\_part[0])  
 )  
 return lst  
  
  
def calculate\_q(f\_list, h):  
 q\_list = [0 for \_ in f\_list]  
 for i in range(len(q\_list)-2, 0, -1): #qn = q0= 0  
 part = q\_koefs(f\_list, i, h)  
 q\_list[i] = q\_list[i+1] \* part[0] + part[1]  
 return q\_list  
  
  
def spline(x\_list, f\_list, q\_list, h):  
 s\_list = [0, 0, 0, 0]  
 for i in range(len(x\_list) - 1):  
 \_part1 = lambda x: q\_list[i] \* ( x\_list[i+1] - x) \*\* 3  
 \_part2 = lambda x: q\_list[i+1] \* (x - x\_list[i]) \*\* 3  
 half1 = lambda x: \_part1(x) / (h \* 6) + \_part2(x) / (h \* 6)  
  
 part1 = lambda x: (f\_list[i] / h - (q\_list[i] \* h / 6)) \* (x\_list[i+1] - x)  
 part2 = lambda x: (f\_list[i+1] / h - (q\_list[i+1] \* h / 6)) \* (x - x\_list[i])  
 half2 = lambda x: part1(x) + part2(x)  
  
 s\_list[i] = lambda x: half1(x) + half2(x)  
 #print(f"({q\_list[i]} \* ( {x\_list[i+1]} - x) \*\* 3) /{h\*6} + ({q\_list[i+1]} \* (x - {x\_list[i]}) \*\* 3) / {h\*6} + ({(f\_list[i] / h - (q\_list[i] \* h / 6))} \* ({x\_list[i+1]} - x) + {(f\_list[i + 1] / h - (q\_list[i + 1] \* h / 6))} \* (x - {x\_list[i]}))")  
  
 return s\_list  
  
  
def check(x, func):  
 results = []  
 results.append(func(x))  
 results.append(derivative(func, x))  
 results.append(derivative(func, x, n=2))  
 return results

from tools import \*  
  
x\_list = [0.2, 0.6, 1, 1.4, 1.8]  
func\_list = [-2.3026, -0.69315, -0.10536, 0.26236, 0.53063]  
h = 0.4  
X = 8  
  
qs = calculate\_q(func\_list, h)  
splines = spline(x\_list, func\_list, qs, h)  
print("q: ", \*qs)  
  
node2 = x\_list[1]  
for i in range(2):  
 lst = check(node2, splines[i])  
 print(str(i+1) + ": S(x) = " + str(lst[0]) + " S'(x) = " + str(lst[1]) + " S''(x) = " + str(lst[2]))  
print(f"f(x\*) = {splines[3](X)}")

**Результати виконання програми:**



**Висновок:** У ході цієї лабораторної роботи, я розробив додаток для сплайн інтерполювання функції, а також побудував кубічний сплайн, використовуючи мову програмування Python та бібліотеки matplotlib.