Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Лабораторна робота №4

З курсу «Чисельні методи»

З теми «Наближення функцій»

Варіант №5

Виконав студент 2 курсу групи КА-01

Вагін Олександр Вікторович

Перевірила старший викладач

Хоменко Ольга Володимирівна

Київ-2022

Завдання №1

$y = 2x - x^2 - 2\cos(x-1)$

Виберемо відрізок інтерполяції $[-2; 4]$. Виберемо 4 вузли так, щоб відстань між ними була однакова:

$x_0 = -2, \qquad x_1 = 0, \qquad x_2 = 2, \qquad x_3 = 4$

Визначимо значення функції в обраних вузлах:

x	-2	0	2	4
y	$-6{,}02002$	$-1{,}08060$	$-1{,}08060$	$-6{,}02002$

Побудуємо таблицю скінченних різниць:

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
-2	$-6{,}02002$	$4{,}93941$	$-4{,}93941$	0
0	$-1{,}08060$	0	$-4{,}93941$	
2	$-1{,}08060$	$-4{,}93941$		
4	$-6{,}02002$			

Знайдемо поліном Лагранжа:

$$L_3(x) = \sum_{i=0}^{3} y_i \prod_{j=0, j\neq i}^{3} \frac{x - x_j}{x_i - x_j} = -6{,}02002 \cdot \frac{x(x-2)(x-4)}{-2 \cdot (-4) \cdot (-6)} - 1{,}08060 \cdot$$

$$\cdot \frac{(x+2)(x-2)(x-4)}{2 \cdot (-2) \cdot (-4)} - 1{,}08060 \cdot \frac{(x+2)x(x-4)}{4 \cdot 2 \cdot (-2)} - 6{,}02002 \cdot$$

$$\cdot \frac{(x+2)x(x-2)}{6 \cdot 4 \cdot 2} = -6{,}02002 \frac{x(x-2)}{48}(x + 2 - x + 4) - 1{,}08060 \cdot$$

$$\cdot \frac{(x+2)(x-4)}{16}(x - 2 - x) = -6{,}02002 \frac{x(x-2)}{8} + 1{,}08060 \frac{(x+2)(x-4)}{8}$$

$$= \frac{1}{8}(-4{,}93942x^2 + 9{,}87884x - 8{,}6448) = -0{,}61743x^2 + 1{,}23486x + 1{,}0806$$

Поліноми Ньютона будемо обчислювати програмно за допомогою формул:

Перший інтерполяційний поліном Ньютона:

$$f(x) \approx y_0 + \sum_{i=1}^{n} \frac{q \cdot \ldots \cdot (q - (i-1))}{i!} \Delta^i y_0$$

Другий інтерполяційний поліном Ньютона:

$$f(x) \approx y_n + \sum_{i=1}^{n} \frac{q \cdot \ldots \cdot (q + (i-1))}{i!} \Delta^i y_{n-i}$$

```
def first_newton_polynom(x, nodes, diffs):
    """Returns value of first newton interpolation polynom"""
    h = nodes[1] - nodes[0]
    q = (x - nodes[0]) / h
    result = func1(nodes[0])
    k = 1
    for i in range(len(diffs) - 1):
        k *= (q - i) / (i + 1)
        result += k * diffs[i][0]
    return result


def second_newton_polynom(x, nodes, diffs):
    """Returns value of second newton interpolation polynom"""
    h = nodes[1] - nodes[0]
    q = (x - nodes[-1]) / h
    result = func1(nodes[-1])
    k = 1
    for i in range(len(diffs) - 1):
        k *= (q + i) / (i + 1)
        result += k * diffs[i][-1]
    return result
```

Використовуючи отримані поліноми, обчислимо значення функції у невузлових точках $\widetilde{x_1} = -1{,}5$, $\widetilde{x_2} = -1$, $\widetilde{x_3} = 0{,}3$, $\widetilde{x_4} = 3{,}2$

x	f(x)	Lagrange	First Newton	Second Newton
-1.5	-3.64771	-2.16091	-4.32209	-4.32209
-1	-2.16771	-0.77169	-2.93288	-2.93288
0.1	-1.05322	1.19791	-0.96329	-0.96329
3.9	-5.46808	-3.49456	-5.65573	-5.65573

Побудуємо інтерполяційний кубічний сплайн за такою таблицею:

x	-2	0	2
y	$-6{,}02002$	$-1{,}08060$	$-1{,}08060$

$$g(x) = \begin{cases} a_1 + b_1(x-0) + c_1(x-0)^2 + d_1(x-0)^3, & x \in [-2{,}0] \\ a_2 + b_2(x-2) + c_2(x-2)^2 + d_2(x-2)^3, & x \in [0{,}2] \end{cases}$$

За умови $g(x_k) = f_k$ маємо:

$$\begin{cases} a_1 + b_1(-2-0) + c_1(-2-0)^2 + d_1(-2-0)^3 = -6{,}02002 \\ a_1 + b_1(0-0) + c_1(0-0)^2 + d_1(0-0)^3 = -1{,}0806 \\ a_2 + b_2(0-2) + c_2(0-2)^2 + d_2(0-2)^3 = -1{,}0806 \\ a_2 + b_2(2-2) + c_2(2-2)^2 + d_2(2-2)^3 = -1{,}0806 \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} a_1 - 2b_1 + 4c_1 - 8d_1 = -6{,}02002 \\ a_1 = -1{,}0806 \\ a_2 - 2b_2 + 4c_2 - 8d_2 = -1{,}0806 \\ a_2 = -1{,}0806 \end{cases} \Rightarrow \begin{cases} 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ a_1 = -1{,}0806 \\ 2b_2 - 4c_2 + 8d_2 = 0 \\ a_2 = -1{,}0806 \end{cases}$$

З неперервності сплайну: $g_1(0) = g_2(0), g_1'(0) = g_2'(0), g_1''(0) = g_2''(0)$

$g_1'(x) = b_1 + 2c_1x + 3d_1x^2, \qquad g_2'(x) = b_2 + 2c_2(x-2) + 3d_2(x-2)^2$

$g_1''(x) = 2c_1 + 6d_1x, \qquad g_2''(x) = 2c_2 + 6d_2(x-2)$

Також використаємо умову, що $g''(a) = g''(b) = 0$. Маємо:

$$\begin{cases} 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ a_1 = -1{,}0806 \\ 2b_2 - 4c_2 + 8d_2 = 0 \\ a_2 = -1{,}0806 \\ b_1 = b_2 - 4c_2 + 12d_2 \\ 2c_1 = 2c_2 - 12d_2 \\ 2c_1 - 12d_1 = 0 \\ 2c_2 = 0 \end{cases} \Rightarrow \begin{cases} a_1 = -1{,}0806 \\ a_2 = -1{,}0806 \\ c_2 = 0 \\ c_1 = 6d_1 \\ 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ 2b_2 - 4c_2 + 8d_2 = 0 \\ b_1 = b_2 - 4c_2 + 12d_2 \\ 2c_1 = 2c_2 - 12d_2 \end{cases} \Rightarrow$$

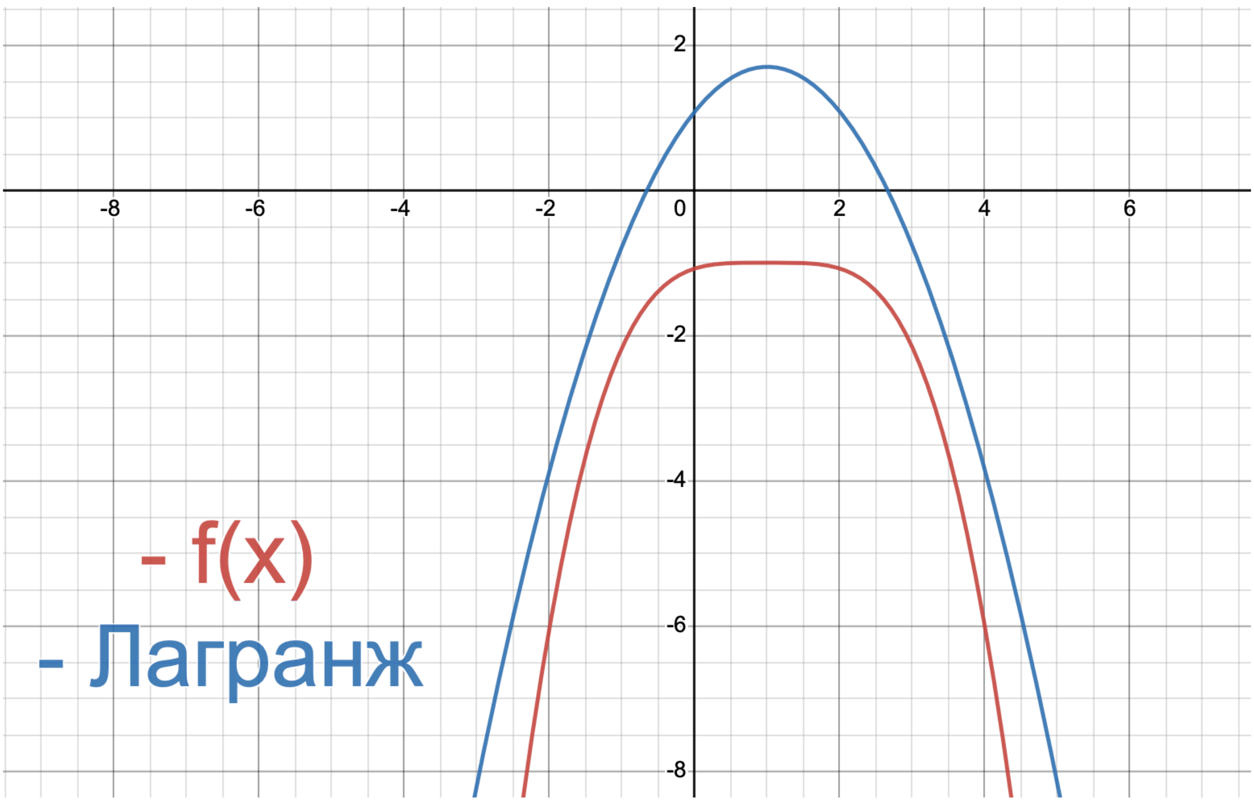
$$\Rightarrow \begin{cases} a_1 = -1{,}0806 \\ a_2 = -1{,}0806 \\ c_2 = 0 \\ c_1 = 6d_1 \\ 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ 2b_2 + 8d_2 = 0 \\ b_1 = b_2 + 12d_2 \\ 2c_1 = -12d_2 \end{cases} \Rightarrow \begin{cases} a_1 = -1{,}0806 \\ a_2 = -1{,}0806 \\ c_2 = 0 \\ c_1 = 6d_1 \\ 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ b_2 + 4d_2 = 0 \\ b_1 = b_2 + 12d_2 \\ c_1 = -6d_2 \end{cases} \Rightarrow$$

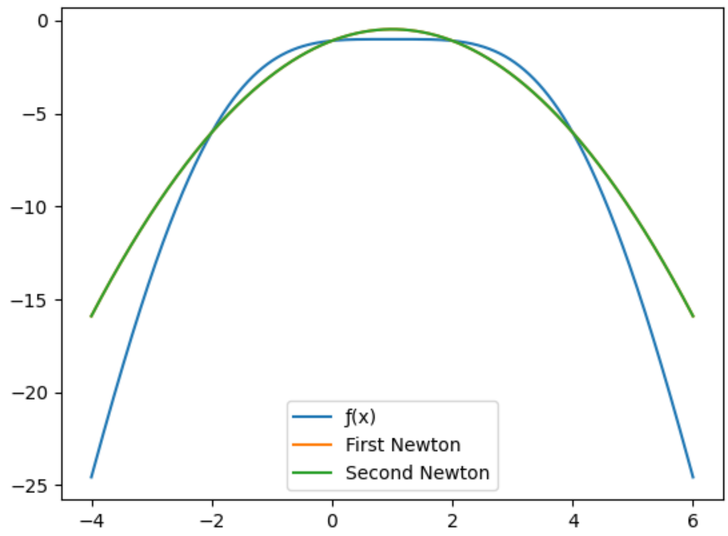
$$\Rightarrow \begin{cases} a_1 = -1{,}0806 \\ a_2 = -1{,}0806 \\ c_2 = 0 \\ c_1 = 6d_1 \\ 2b_1 - 4c_1 + 8d_1 = 4{,}93942 \\ b_2 = -4d_2 \\ b_1 = 8d_2 \\ d_1 = -d_2 \end{cases} \Rightarrow \begin{cases} a_1 = -1{,}0806 \\ a_2 = -1{,}0806 \\ c_2 = 0 \\ c_1 = -0{,}92614 \\ d_1 = -0{,}15436 \\ d_2 = 0{,}15436 \\ b_2 = -0{,}61743 \\ b_1 = 1{,}23486 \end{cases}$$

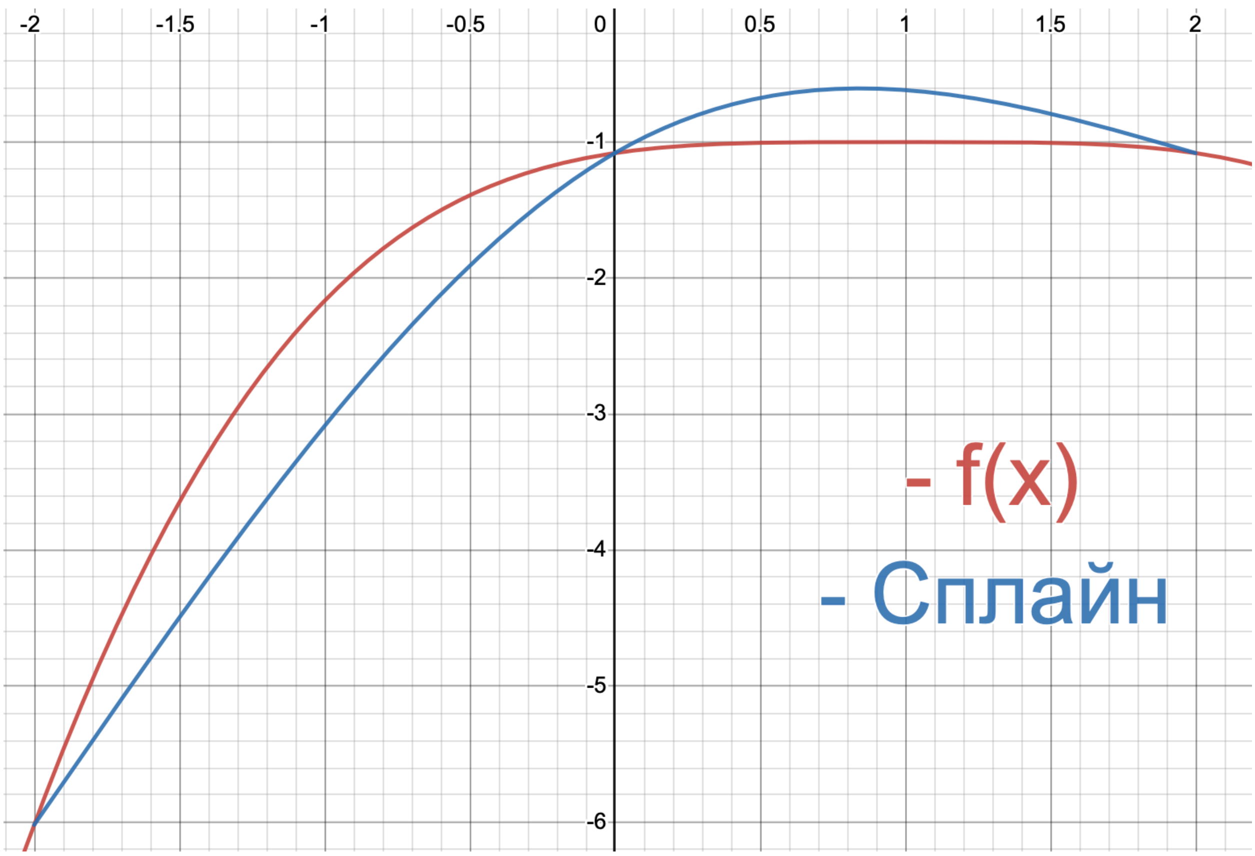
А отже:

$$g(x) = \begin{cases} -1{,}0806 + 1{,}23486x - 0{,}92614x^2 - 0{,}15436x^3, & x \in [-2,0] \\ -1{,}0806 - 0{,}61743(x-2) + 0{,}15436(x-2)^3, & x \in [0,2] \end{cases}$$

Побудуємо графіки отриманих наближень:



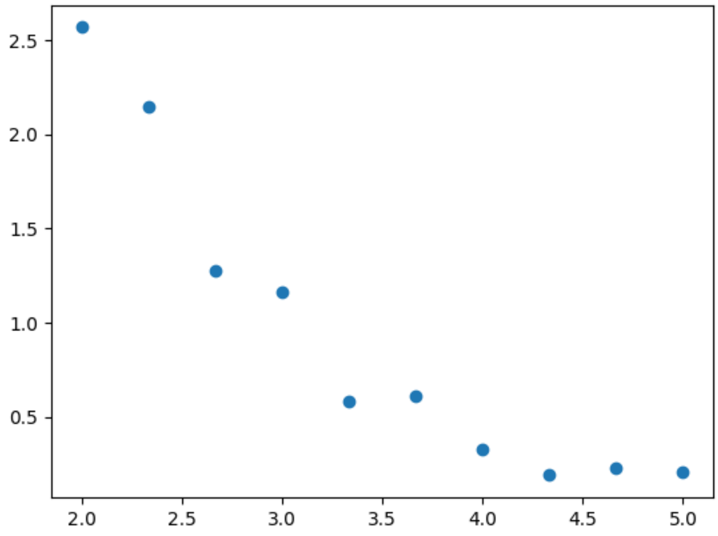




Завдання №2

x	2,00	2,33	2,67	3,00	3,33	3,67	4,00	4,33	4,67	5,00
y	2,57	2,15	1,28	1,16	0,58	0,61	0,33	0,19	0,23	0,21

Нанесемо точки задані в таблиці на графік:



З графіку можна побачити, що за апроксимувальну функцію доцільно брати:

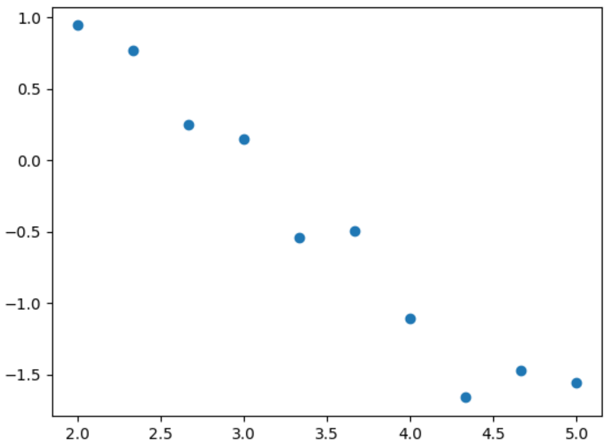
$y = ae^{bx}$

Прологарифмуємо цю рівність:

$\ln y = \ln a + bx$

Введемо $Y_i = \ln y_i$ та переробимо таблицю:

x	2,00	2,33	2,67	3,00	3,33	3,67	4,00	4,33	4,67	5,00
y	0,94	0,77	0,25	0,15	−0,54	−0,49	−1,11	−1,66	−1,47	−1,56



Тепер методом найменших квадратів знайдемо оптимальні параметри A і b:

$$Y = bx + A$$

$$\sum_{i=0}^{9} (bx_i + A - Y_i)^2 \to min$$

Прирівняємо до нуля частинні похідні по b та A:

$$\begin{cases} \sum_{i=0}^{9} 2(bx_i + A - Y_i)x_i = 0 \\ \sum_{i=0}^{9} 2(bx_i + A - Y_i) = 0 \end{cases} \Longrightarrow \begin{cases} b\sum_{i=0}^{9} 2x_i^2 + A\sum_{i=0}^{9} 2x_i = \sum_{i=0}^{9} 2x_i\, Y_i \\ b\sum_{i=0}^{9} 2x_i + 20A = \sum_{i=0}^{9} 2\, Y_i \end{cases}$$

Маємо систему лінійних рівнянь:

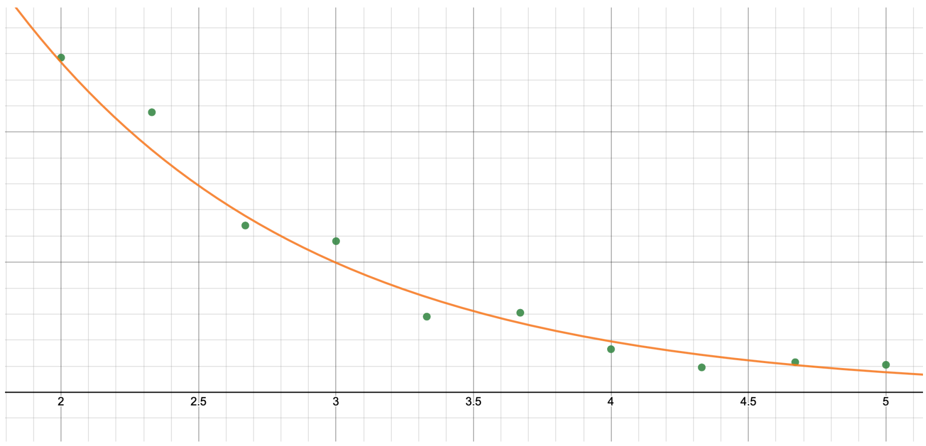
$$\begin{pmatrix} \sum_{i=0}^{9} 2x_i^2 & \sum_{i=0}^{9} 2x_i \\ \sum_{i=0}^{9} 2x_i & 20 \end{pmatrix} \begin{pmatrix} b \\ A \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{9} 2x_i\, Y_i \\ \sum_{i=0}^{9} 2\, Y_i \end{pmatrix}$$

Розв’язавши цю систему програмно, маємо: $\begin{pmatrix} b \\ A \end{pmatrix} = \begin{pmatrix} -0{,}93538 \\ 2.80041 \end{pmatrix}$

$$a = e^A = e^{2.80041}$$

Остаточно маємо $f(x) \approx e^{2.80041} e^{-0{,}93538x} = e^{2.80041-0{,}93538x}$

Побудуємо графік отриманої функції разом з заданими точками:



Код програм:

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


def chebyshev_nodes(a, b, n):
    """Takes segment and number of nodes, returns nodes"""
    nodes = []
    for i in range(n + 1):
        nodes.append((b + a) / 2 - (b - a) / 2 * np.cos((2 * i + 1) / (2 * n
+ 2) * np.pi))
    return nodes


def func1(x):
    """Returns value of function from first task"""
    return 2 * x - x ** 2 - 2 * np.cos(x - 1)


def finite_differences(func_values):
    """Takes values of function in nodes, returns list of finite differences
all orders"""
    n = len(func_values)
    diffs = [func_values]
    while n > 1:
        diffs.append(tuple(diffs[-1][i + 1] - diffs[-1][i] for i in range(n -
1)))
        n -= 1
    del diffs[0]
    return diffs


def lagrange(x):
    """Returns value of lagrange interpolation polynom for first task"""
    return -0.61743 * x ** 2 + 1.23486 * x + 1.0806


def first_newton_polynom(x, nodes, diffs):
    """Returns value of first newton interpolation polynom"""
    h = nodes[1] - nodes[0]
    q = (x - nodes[0]) / h
    result = func1(nodes[0])
    k = 1
    for i in range(len(diffs) - 1):
        k *= (q - i) / (i + 1)
        result += k * diffs[i][0]
    return result


def second_newton_polynom(x, nodes, diffs):
    """Returns value of second newton interpolation polynom"""
    h = nodes[1] - nodes[0]
    q = (x - nodes[-1]) / h
    result = func1(nodes[-1])
    k = 1
    for i in range(len(diffs) - 1):
        k *= (q + i) / (i + 1)
        result += k * diffs[i][-1]
    return result


def comparison_table(points):
```

```
    """Prints value of approximation functions"""
    print("-" * 71)
    print('|' + "x".center(9) + '|'
          + 'f(x)'.center(12) + '|'
          + 'Lagrange'.center(12) + '|'
          + 'First Newton'.center(16) + '|'
          + 'Second Newton'.center(16) + '|')
    print("-" * 71)
    for x in points:
        print('|' + str(x).center(9) + '|'
              + '{:.5f}'.format(func1(x)).center(12) + '|'
              + '{:.5f}'.format(lagrange(x)).center(12) + '|'
              + '{:.5f}'.format(first_newton_polynom(x, nodes,
differences)).center(16) + '|'
              + '{:.5f}'.format(second_newton_polynom(x, nodes,
differences)).center(16) + '|')
        print("-" * 71)


def smallest_squares(x, y):
    """Returns value of coefficient of method the smallest squares for linear
function"""
    a00 = sum(2 * t ** 2 for t in x)
    a01 = sum(2 * t for t in x)
    a10 = sum(2 * t for t in x)
    a11 = 20
    a = ((a00, a01), (a10, a11))
    b0 = sum(2 * t * s for t, s in zip(x, y))
    b1 = sum(2 * s for s in y)
    b = (b0, b1)
    return np.linalg.solve(a, b)


# First Task

# Calculating values in nodes
nodes = (-2, 0, 2, 4)
values = tuple(func1(node) for node in nodes)
print("Function value in nodes:")
for node in values:
    print('{:.5f}'.format(node), end=' ')

# Calculating finite differences
differences = finite_differences(values)
print("\n\nFinite differences:")
print(*differences, sep='\n')

# Comparing values
xs = (-1.5, -1, 0.1, 3.9)
comparison_table(xs)

# Drawing newtons
plt.subplots()
ls = np.linspace(-4, 6, 1000)
plt.plot(ls, func1(ls), label='f(x)')
plt.plot(ls, first_newton_polynom(ls, nodes, differences), label='First
Newton')
plt.plot(ls, second_newton_polynom(ls, nodes, differences), label='Second
Newton')
plt.legend(loc='best')
plt.show()

# Second Task
```

```
# Visualisation of initial data
x = np.linspace(2, 5, 10)
y = (2.57, 2.15, 1.28, 1.16, 0.58, 0.61, 0.33, 0.19, 0.23, 0.21)
plt.scatter(x, y)
plt.show()

# Parsing y-values to linearize sample
y1 = tuple(map(np.log, y))
plt.scatter(x, y1)
plt.show()

root = smallest_squares(x, y1)
print(root)
```

Висновок:

У першому завданні ми обирали вузлові точки, а потім будували різні варіанти інтерполяції. Як на мене найкраще з задачею наближення впоралися поліноми Ньютона. Також доволі близьким виявився сплайн. А от поліном Лагранжа виявився доволі далеким. До того ж перевіряючи значення не у вузлових точках значення доволі сильно різнилися. Це пов’язано саме з інтерполяцією

У другому завданні ми апроксимували функцію яка могла породити таблицю значень. І як на мене отримали набагато цікавіший результат ніж у першому завданні. На перший погляд було важко зрозуміти яким саме класом треба було апроксимувати, але шляхом підбору у графічному застосунку було виявлено, що напевно це має бути показникова функція. Після обчислень, виявилось що дійсно, експонента зі зміщенням доволі гарно описує задані точки