Київський національний університет імені Тараса
Шевченка
Факультет комп’ютерних наук та кібернетики

ЛАБОРАТОРНА РОБОТА №2
З курсу “Комп’ютерне моделювання”

Виконав:
Студент 3 курсу, групи ТТП-31
Спеціальності “Комп’ютерні науки”
Масич Нікіта Дмитрович

Київ - 2023

Лабораторна робота №2

Завдання №1

$y = \sqrt{x^2+2}\cos(2,14x)$, $x \in [1; 3,5]$

Побудувати середньоквадратичне наближення для $f(x)$, використовуючи поліноми Чебишева ($x \in [-1; 1]$, $n = 4$):

1. Нормалізуємо інтервал $[1; 3,5]$ до $[-1; 1]$

Введемо заміну $t = \frac{4x-9}{5}$, $x \in [1; 3,5]$, $t \in [-1; 1]$.

Тоді $x = \frac{5t+9}{4}$, отже $y = \sqrt{\left(\frac{5t+9}{4}\right)^2 + 2}\cos\left(2,14 \cdot \frac{5t+9}{4}\right)$

2. Перші 4 поліноми Чебишева:

$T_0(t) = 1$ $\qquad T_2(t) = 2t^2 - 1$

$T_1(t) = t$ $\qquad T_3(t) = 4t^3 - 3t$

3. Обчислимо перші 4 коефіцієнти середньоквадратичного наближення:

$$C_k = \frac{2}{\pi}\int_{-1}^{1}\frac{f(t)\,T_k(t)}{\sqrt{1-t^2}}\,dt$$

- Маємо обрахунки в програмі на python 3

4. Отримуємо середньоквадратичне наближення:

$$f(t) \approx \sum_{k=0}^{3} c_k T_k(t) =$$

$$= 0,28998 + 1,34006\,t - 0,17368\,(2t^2-1) - 1,47187\,(4t^3-3t)$$

5. Проведемо зворотню заміну.

$$f(x) \approx 0,28998 + 1,34006\left(\frac{4x-9}{5}\right) - 0,17368\left(2\cdot\left(\frac{4x-9}{5}\right)^2 - 1\right) -$$

$$- 1,47187\left(4\left(\frac{4x-9}{5}\right)^3 - 3\left(\frac{4x-9}{5}\right)\right)$$

Вище наведене 1 завдання та відповідний розв’язок.

Допоміжний код для обрахування коефіцієнтів апроксимації:

```
# Функція для обчислення коефіцієнтів c_k
def calculate_coefficients(k):
    # Функція, яку інтегруємо
    integrand = lambda t: f((b - a) * t / 2 + (a + b) / 2) * chebyt(k)(t)

    # Інтеграл для обчислення коефіцієнта
    result, _ = quad(integrand, -1, 1)

    # Норма полінома Чебишева
    norm_squared = quad(lambda t: chebyt(k)(t)**2 / np.sqrt(1 - t**2), -1, 1)[0]

    # Обчислення коефіцієнта c_k
    coefficient = result / norm_squared

    return coefficient
```

Помітимо, що аргумент функції приведений відповідно до t (нормалізація інтервалу).

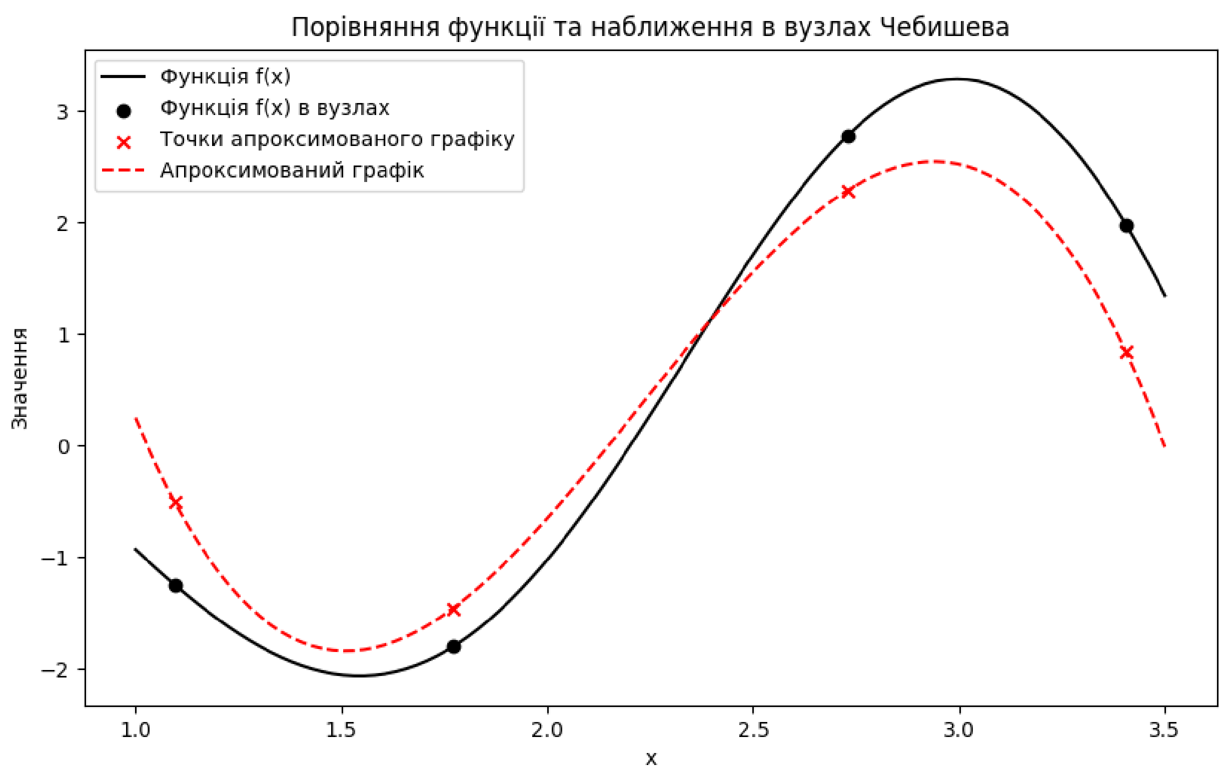
Сама функція та інші вхідні далі декларовано наступним чином:

```
# Задана функція f(x)
def f(x):
    return np.sqrt(x**2 + 2) * np.cos(2.14 * x)

# Нормалізація інтервалу [1, 3.5] до [-1, 1]
a = 1
b = 3.5
normalize = lambda x: (2 * x - a - b) / (b - a)

# Кількість вузлів Чебишева
n = 4
```

Для дослідження точності у вузлах використовуємо наступний код:

```
# Функція для обчислення значень функції та наближення в вузлах Чебишева
def evaluate_approximation():
    # Вузли Чебишева
    cheb_nodes = np.cos((2 * np.arange(1, n + 1) - 1) * np.pi / (2 * n))

    # Обчислення значень функції та наближення в вузлах
    true_values = [f((b - a) * t / 2 + (a + b) / 2) for t in cheb_nodes]
    approx_values = [sum(coefficients[k] * chebyt(k)(t) for k in range(n)) for t in cheb_nodes]

    return cheb_nodes, true_values, approx_values
```

Побудуємо отримані графіки для наглядного спостереження девіації:



Додатково виведемо наступну таблицю конкретних значень:

Вузол	Функція	Апроксимація	Похибка
0.923880	1.982109	0.841969	57.521555
0.382683	2.774461	2.285436	17.625942
-0.382683	-1.804995	-1.459861	19.121068
-0.923880	-1.248783	-0.507623	59.350591

Відповідний код для побудови графіку з використанням бібліотеки matplotlib:

```
# Побудова графіків
x_vals = np.linspace(a, b, 100)
plt.plot(x_vals, f(x_vals), 'black', label='Функція f(x)')

# Точки для функції в вузлах Чебишева
plt.scatter((b - a) * cheb_nodes / 2 + (a + b) / 2, true_values, c='black', marker='o', label='Функція f(x) в вузлах')

# Точки для апроксимованого графіку в тих же вузлах
plt.scatter((b - a) * cheb_nodes / 2 + (a + b) / 2, approx_values, c='red', marker='x', label='Точки апроксимованого графіку')

# Лінія для апроксимованого графіку
plt.plot(x_vals, [sum(coefficients[k] * chebyt(k)((2 * x - a - b) / (b - a)) for k in range(n)) for x in x_vals], 'r--', label='Апроксимований графік')

plt.legend()
plt.xlabel('x')
plt.ylabel('Значення')
plt.title('Порівняння функції та наближення в вузлах Чебишева')
plt.show()
```

<u>Завдання 2</u>

Завдання 2: На основі методу найменших квадратів провести побудову кривих заданих системою точок на площині масивами значень аргументів та функцій (mas_x, mas_y). Передбачається, що відомий функціональний закон для множини точок.

Маємо наступні функції:

- $y = Ax + B$
- $y = Ax^2 + Bx + C$

$k2 = 7;\ k1 = 0.07$

Задекларуємо їх і приведемо датасети (x, y) координат відповідно до значень k:

```
x_data = np.array([-1.6000, -1.2000, -0.8000, -0.4000, 0, 0.4000, 0.8000, 1.2000, 1.6000, 2.0000])

y_data_linear = np.array([-0.2000, 0.6000, 1.4000, 2.2000, 3.0000, 3.8000, 4.6000, 5.4000, 6.2000, 7.0000])
y_data_quadratic = np.array([4.3200, 3.2800, 2.8800, 3.1200, 4.0000, 5.5200, 7.6800, 10.4800, 13.9200, 18.0000])

k2 = 7
k1 = k2 * 0.01

x_data += k2

y_data_linear += k1
y_data_quadratic += k1

def linear_func(x, A, B):
    return A * x + B

def quadratic_func(x, A, B, C):
    return A * x**2 + B * x + C
```

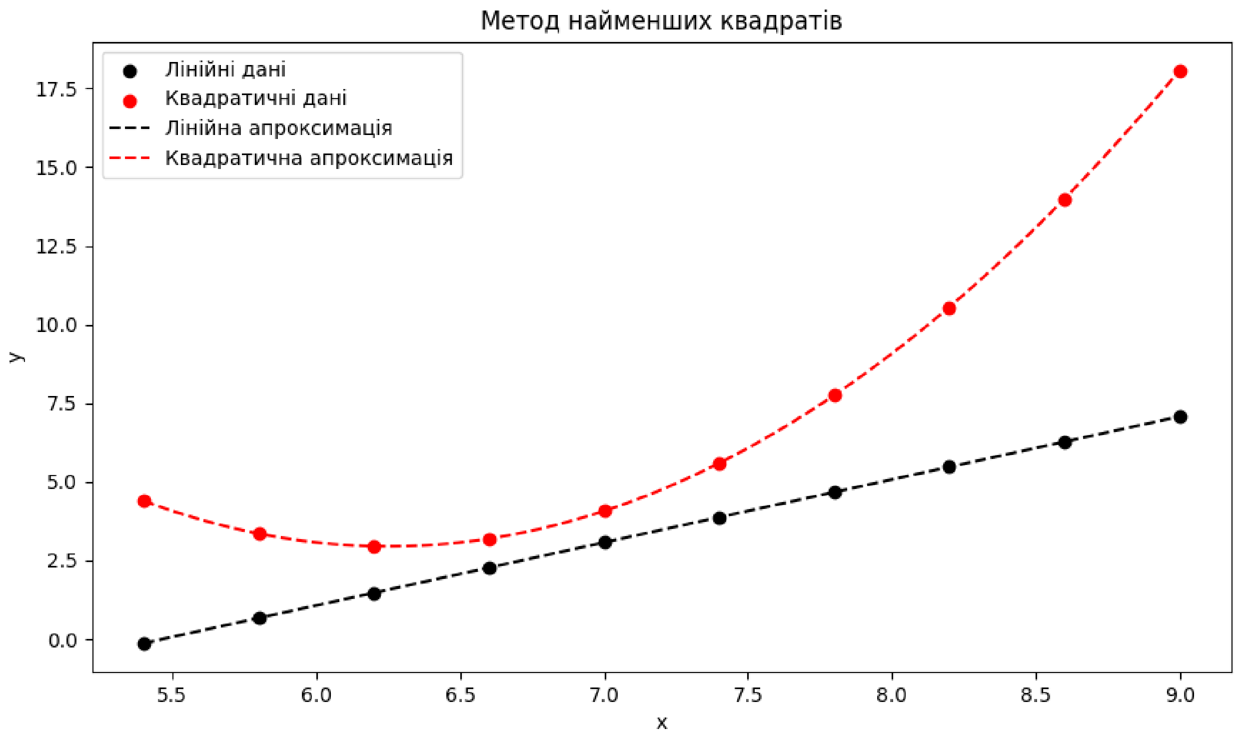
Наступним кроком апроксимуємо методом найменших квадратів дані функції для знаходження відповідних параметрів A, B, C. Використаємо для цього метод curve_fit з бібліотеки scipy:

```
params_linear, covariance_linear = curve_fit(linear_func, x_data, y_data_linear)
A_linear, B_linear = params_linear

params_quadratic, covariance_quadratic = curve_fit(quadratic_func, x_data, y_data_quadratic)
A_quadratic, B_quadratic, C_quadratic = params_quadratic
```

Отримавши параметри, можемо побудувати графіки. Для цього обрахуємо множини координат:

```
x_vals = np.linspace(min(x_data), max(x_data), 100)

y_linear = linear_func(x_vals, A_linear, B_linear)
y_quadratic = quadratic_func(x_vals, A_quadratic, B_quadratic, C_quadratic)
```

Отримали наступну візуалізацію:



Отже, апроксимація вдала. Фінальні функції з параметрами:

```
Лінійна апроксимація: y = 2.0000 * x + -10.9300
Квадратична апроксимація: y = 2.0000 * x^2 + -25.0000 * x + 81.0700
```

Код, використаний для виконання цієї роботи можна побачити за посиланням:

https://github.com/NikitaMasych/km-lab2