НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Лабораторна робота №3

Тема: «Опукла оболонка»

Виконала: студентка групи КМ-31
Буряк Софія

КИЇВ — 2024

Мета

Розробити програмний засіб, який знаходить опуклу оболонку множини точок заданих своїми координатами та відображує її на координатній площині і зберігає зображення в одному з графічних форматів.

Обрана бібліотека

Для розв'язання завдання з побудови опуклої оболонки та візуалізації даних були використані три основні бібліотеки Python: NumPy, SciPy та Matplotlib. Кожна з них виконує певну роль у загальному процесі. NumPy є однією з найбільш популярних бібліотек для роботи з числовими даними в Python. Вона забезпечує ефективну роботу з багатовимірними масивами та зручні інструменти для обробки даних. SciPy — це потужна бібліотека для числових і наукових обчислень, побудована на основі NumPy. У даному проєкті була використана її підмодуль scipy.spatial, який включає функціонал для геометричних обчислень. Matplotlib є стандартом у Python для створення графіків і візуалізації даних. Ця бібліотека дозволяє легко і зручно представляти дані у вигляді графіків.

Усі методи, які використовувалися з бібліотек:

Matplotlib

- plt.figure — створення полотна, де буде виконуватись побудова графіка. Це дозволяє налаштувати розмір полотна, роздільну здатність та інші параметри.
- plt.scatter — нанесення точок на графік у вигляді розсіювання (scatter plot), що використовується для візуалізації вихідного набору даних.
- plt.plot — побудова лінійного графіка для з'єднання точок опуклої оболонки та формування замкненого контуру.
- plt.xlim, plt.ylim — встановлення меж графіка за осями X та Y для оптимального масштабу відображення.
- plt.title, plt.xlabel, plt.ylabel — додавання заголовка та підписів до осей графіка для більшої зрозумілості.
- plt.grid — активація сітки на графіку для полегшення аналізу розташування точок.
- plt.legend — додавання легенди для пояснення графічних елементів (точок вихідного датасету та контуру опуклої оболонки).

- plt.savefig — збереження графіка у файл у заданому графічному форматі (наприклад, PNG), що дозволяє використовувати його для звітів.
- plt.show — виведення графіка на екран для інтерактивного перегляду результатів.

NumPy

- np.loadtxt — зчитування координат точок із текстового файлу, перетворюючи їх у числовий масив для подальшої обробки.
- np.savetxt — збереження опуклої оболонки у новий текстовий файл із заданим форматом координат.
- np.vstack — об'єднання масивів, зокрема додавання початкової точки до кінця масиву для замикання контуру опуклої оболонки.

SciPy

- ConvexHull — побудова опуклої оболонки для множини точок, що визначає індекси точок, які утворюють цю оболонку.

Хід виконання

1. Для читання координат була розроблена функція read_coordinates. Функція відповідає за зчитування даних із текстового файлу, який містить пари чисел — координати точок. Її завдання полягає в обробці текстових даних і перетворенні їх у зручний для обробки формат.

   Ключові етапи:

   - Відкриття файлу у режимі читання.
   - Зчитування кожного рядка файлу.
   - Використання методу split() для розділення координат у рядку.
   - Перетворення текстових значень у числовий формат за допомогою функції map(int, ...).
   - Формування масиву точок, який згодом використовується для візуалізації та обчислення опуклої оболонки.

2. Для візуалізації точок та опуклої оболонки була створена функція plot_with_convex_hull. Ця функція реалізує процес побудови графіка з двома основними елементами: точками вихідного набору даних і лінією, що представляє опуклу оболонку.

   Ключові етапи:

   - Налаштування розміру полотна через метод plt.figure, що дозволяє створити полотно відповідного розміру для відображення даних.

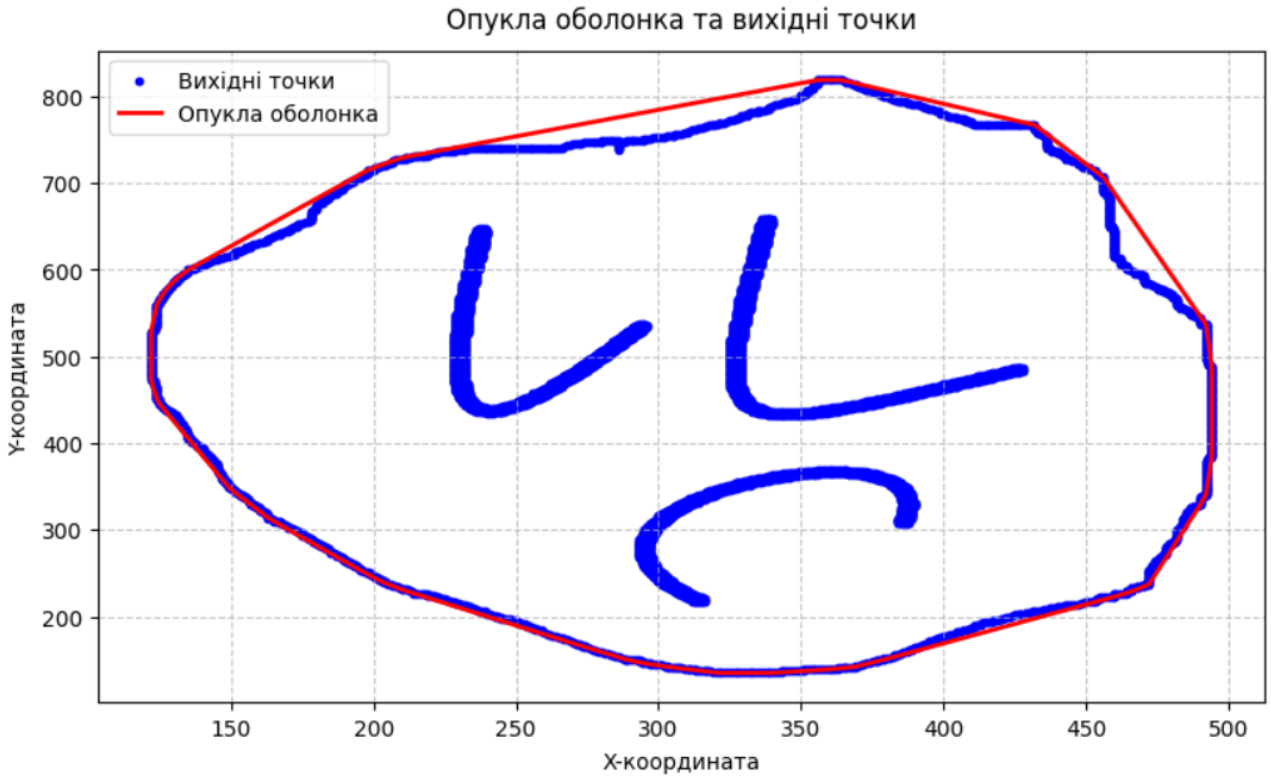
- Відображення точок набору даних за допомогою методу plt.scatter, з можливістю налаштування кольору, розміру та прозорості точок.
- Побудова опуклої оболонки за допомогою методу plt.plot, що малює лінію між точками оболонки, створюючи замкнутий контур.
- Налаштування графіка: додавання заголовка, підписів осей, сітки, а також автоматичного визначення меж осей графіка з урахуванням діапазону координат точок. Зазначені межі додатково коригуються на 5%, щоб точки не розташовувалися на краю полотна.
- Додавання легенди для пояснення графічних елементів.

3. Збереження графіка у файл формату PNG. Фінальний етап роботи передбачає збереження створеного графіка у зручному форматі для подальшого використання.

   Ключові етапи:

   - Використання методу plt.savefig для збереження графіка у форматі PNG із заданими параметрами (роздільна здатність, кольори фону, межі зображення).
   - Перевірка успішності збереження: у разі успіху виводиться повідомлення, у разі виникнення помилки — обробляється виняток із відповідним повідомленням.
   - Відображення графіка на екрані за допомогою plt.show для візуального контролю.

Результат



Код

```
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
import numpy as np

# Функція для зчитування координат із файлу
def read_coordinates(file_name):
    try:
        points = np.loadtxt(file_name, dtype=int)
        return points
    except FileNotFoundError:
        print(f"Файл {file_name} не знайдено!")
        return np.array([])

# Функція для збереження опуклої оболонки у новий датасет
def save_convex_hull(hull_points, output_file="convex_hull.txt"):
    try:
        np.savetxt(output_file, hull_points, fmt='%d %.8f')
        print(f"Опуклу оболонку збережено у файл: {output_file}")
    except Exception as e:
        print(f"Помилка при збереженні опуклої оболонки: {e}")

# Функція для побудови графіка
def plot_with_convex_hull(points, hull_indices, canvas_size=(960, 540),
output_file="output_image.png"):
    if points.size == 0:
        print("Список точок порожній, нічого виводити.")
        return

    # Отримуємо точки опуклої оболонки за індексами
    hull_points = points[hull_indices]
    hull_points = np.vstack((hull_points, hull_points[0]))

    # Створюємо полотно
    fig, ax = plt.subplots(figsize=(canvas_size[0] / 100, canvas_size[1] / 100),
dpi=100)

    # Відображаємо точки вихідного датасету
    ax.scatter(points[:, 0], points[:, 1], c='blue', s=10, label="Вихідні точки")

    # Відображаємо опуклу оболонку
    ax.plot(hull_points[:, 0], hull_points[:, 1], c='red', linewidth=2,
label="Опукла оболонка")

    # Налаштування вигляду графіка
    ax.set_title("Опукла оболонка та вихідні точки", pad=10)
    ax.set_xlabel("X-координата", labelpad=5)
    ax.set_ylabel("Y-координата", labelpad=5)
    ax.grid(True, linestyle='--', alpha=0.7)
    ax.legend()
    ax.set_aspect('auto')
```

```
    # Зберігаємо результат у файл
    try:
        plt.savefig(output_file, bbox_inches='tight', dpi=100)
        print(f"Графік успішно збережено у файл: {output_file}")
    except Exception as e:
        print(f"Помилка при збереженні графіка: {e}")

    plt.show()
    plt.close()

file_name = "DS1.txt"
output_hull_file = "convex_hull.txt"
output_image_file = "output_image.png"

points = read_coordinates(file_name)

# Перевірка, чи є точки для обробки
if points.size > 0:
    hull = ConvexHull(points)
    hull_indices = hull.vertices

    save_convex_hull(points[hull_indices], output_hull_file)

    plot_with_convex_hull(points, hull_indices, canvas_size=(960, 540),
output_file=output_image_file)
```