Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра ІПІ

Звіт

з лабораторної роботи № 7 з дисципліни
«Методи та технології штучного інтелекту»

„Знаходження мінімуму та максимуму функцій за допомогою генетичних алгоритмів”

Виконав(ла) *ІП-14 Шляхтун Денис Михайлович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Шимкович Володимир Миколайович*
(прізвище, ім'я, по батькові)

Київ 2023

Мета: знайти мінімум (мінімізація) і максимум (максимізація) функцій одно- і двох змінних за допомогою генетичних алгоритмів.

Постановка задачі.

1. Знайти мінімум функції однієї змінної за варіантом з таблиці Б.

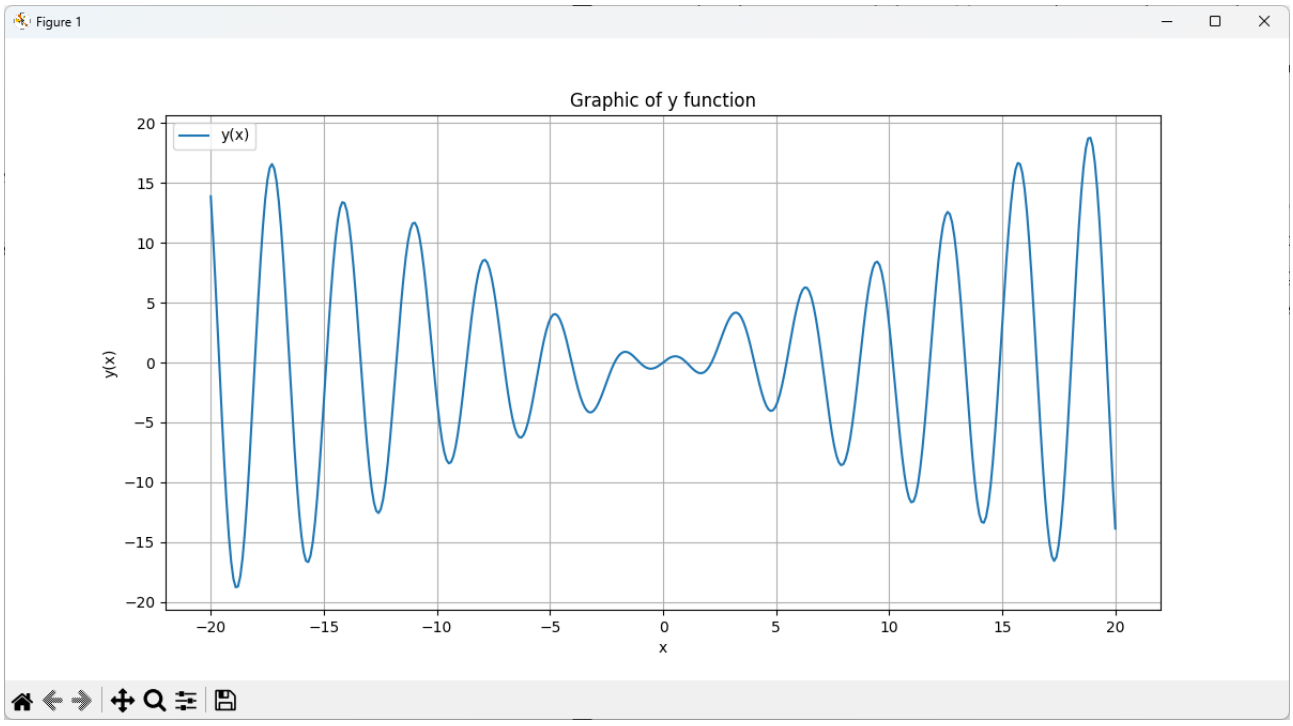
2. Знайти максимум функції двох змінних за варіантом з таблиці Б.

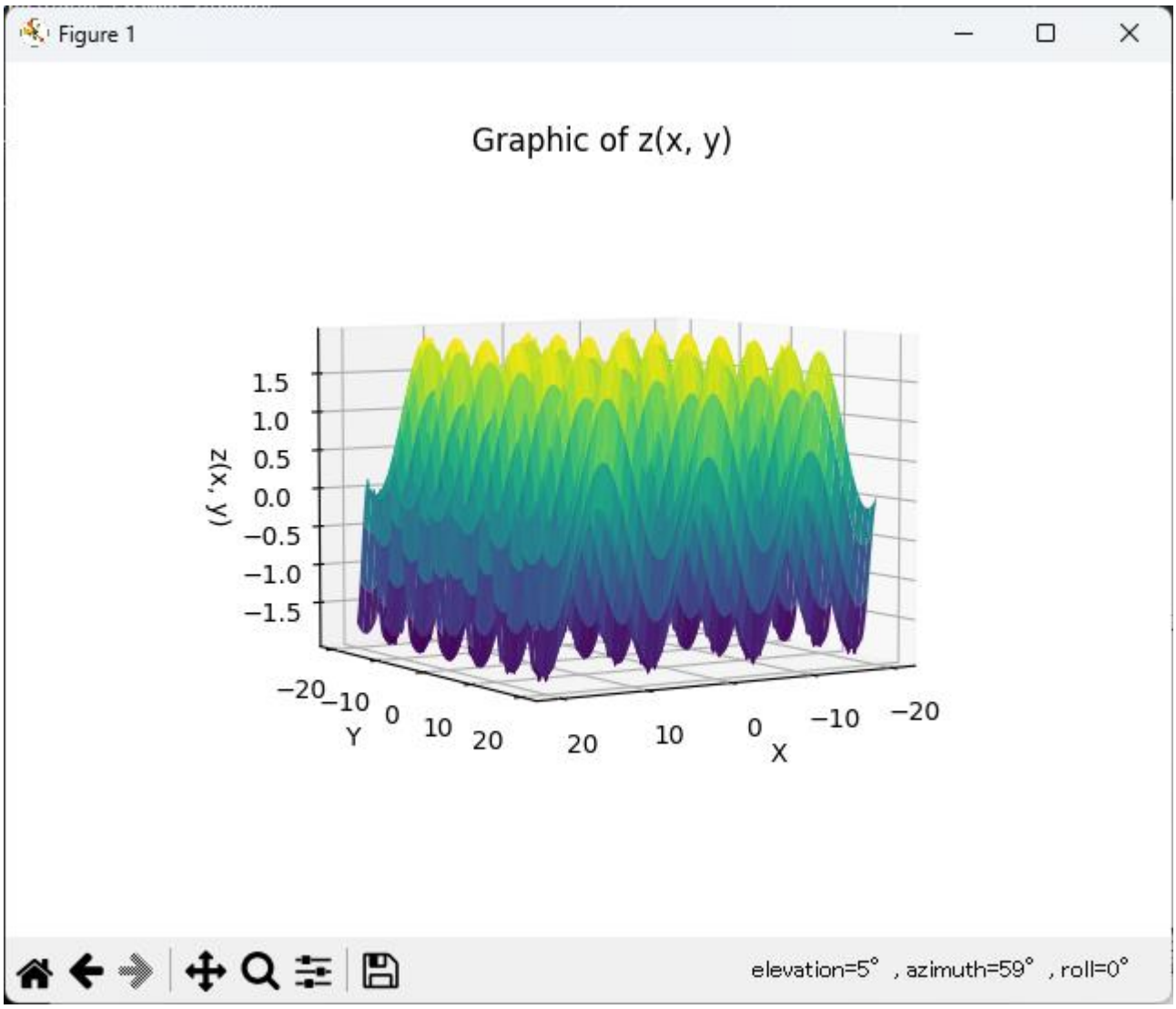
3. Оформити та здати звіт по лабораторній роботі.

24.	$y = x \cdot \cos(2x) + \sin(x/2)$ $z = \sin(y) + \cos(x/2)$	24.	7.	18.

Виконання завдання.

Для виконання завдання була обрана мова програмування високого рівня Python.






```
Expected y minimal value:  -18.787519682523794
Expected z maximal value:  1.9998668238507964
Results of execution of genetic algorithm:
x value: [-18.926086988797763] y(x) minimal value: -18.66656294421129
x and y values: [12.575390171314012, -11.010290347154195] z(x, y) maximal value: 1.9998815517154727
```

Висновок.

При виконанні лабораторної роботи було знайдено мінімум функції однієї змінної та максимум функції двох змінних відповідно до варіанту. Результати генетичного алгоритму близькі до очікуваних, тому задача виконана правильно.

Код програми:

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

# Functions
def yfunction(x):
    return x*np.cos(2*x) + np.sin(x / 2)

def zfunction(x, y):
    return np.sin(y) + np.cos(x/2)

# Initial preparations
x = np.linspace(-20, 20, 400)
Yvals = yfunction(x)
y = np.linspace(-20, 20, 400)
X, Y = np.meshgrid(x, y)
Z = zfunction(X, Y)
real_min_yfunction = np.min(Yvals)
real_max_zfunction = np.max(Z)
print("Expected results: ")
print("Expected y minimal value: ", real_min_yfunction)
print("Expected z maximal value: ", real_max_zfunction)

#Genetic algorithm for searching max/min
class GeneticAlgorithm:
    def __init__(self, func, vars_am, bounds, pop_size=100, mut_rate=0.01, gens=100,
search_max=True):
        self.func = func
        self.vars_am = vars_am
        self.bounds = bounds
        self.pop_size = pop_size
        self.mut_rate = mut_rate
        self.gens = gens
        self.search_max = search_max

    def create_individual(self):
        return [random.uniform(*bound) for bound in self.bounds]

    def create_population(self):
        return [self.create_individual() for _ in range(self.pop_size)]

    def fitness(self, individual):
        return self.func(*individual)

    def selection(self, population):
        fitnesses = [self.fitness(individual) for individual in population]
        if self.search_max:
            return max(zip(population, fitnesses), key=lambda lx: lx[1])[0]
        else:
```

```
            return min(zip(population, fitnesses), key=lambda lx: lx[1])[0]

    def crossover(self, parent1, parent2):
        child = []
        for i in range(self.vars_am):
            if random.random() < 0.5:
                child.append(parent1[i])
            else:
                child.append(parent2[i])
        return child

    def mutation(self, individual):
        for i in range(self.vars_am):
            if random.random() < self.mut_rate:
                individual[i] = random.uniform(*self.bounds[i])
        return individual

    def execute_ga(self):
        population = self.create_population()
        for _ in range(self.gens):
            new_population = []
            for _ in range(self.pop_size):
                parent1 = self.selection(population)
                parent2 = self.selection(population)
                child = self.crossover(parent1, parent2)
                child = self.mutation(child)
                new_population.append(child)
            population = new_population
        return self.selection(population)

# Y function
print("Results of execution of genetic algorithm:")
ga_min = GeneticAlgorithm(yfunction, vars_am=1, bounds=[(-20, 20)],
search_max=False)
result_min = ga_min.execute_ga()
print("x value:", result_min, "y(x) minimal value:", yfunction(*result_min))
plt.figure(figsize=(12, 6))
plt.plot(x, yfunction(x), label='y(x)')
plt.title('Graphic of y function')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.grid(True)
plt.legend()
plt.show()

# Z function
ga_max = GeneticAlgorithm(zfunction, vars_am=2, bounds=[(-20, 20), (-20, 20)],
search_max=True)
result_max = ga_max.execute_ga()
```

```
print("x and y values:", result_max, "z(x, y) maximal value:",
zfunction(*result_max))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')
ax.set_title('Graphic of z(x, y)')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('z(x, y)')
ax.view_init(azim=45, elev=5)
plt.show()
```