Міністерство освіти і науки України  
НТУУ «КПІ ім. І. Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЛАБОРАТОРНА РОБОТА № 3  
з дисципліни «МЕТОДОЛОГІЯ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ КОМП’ЮТЕРНИХ ПРОГРАМ»

Тема: «Оптимізація функції із застосуванням генетичних алгоритмів»

ТІ-92 Черноусов Денис

Перевірив д.т.н Мусієнко А. П.

КИЇВ 2021

**Мета роботи:**

Написати програму оптимізації функції з використанням

генетичного алгоритму

**Завдання:**

Вибрати функцію й діапазон відповідно до варіанта. Побудувати графік

функції. Написати програму знаходження максимуму й мінімуму функції на

заданому діапазоні. Проаналізувати отримані результати.

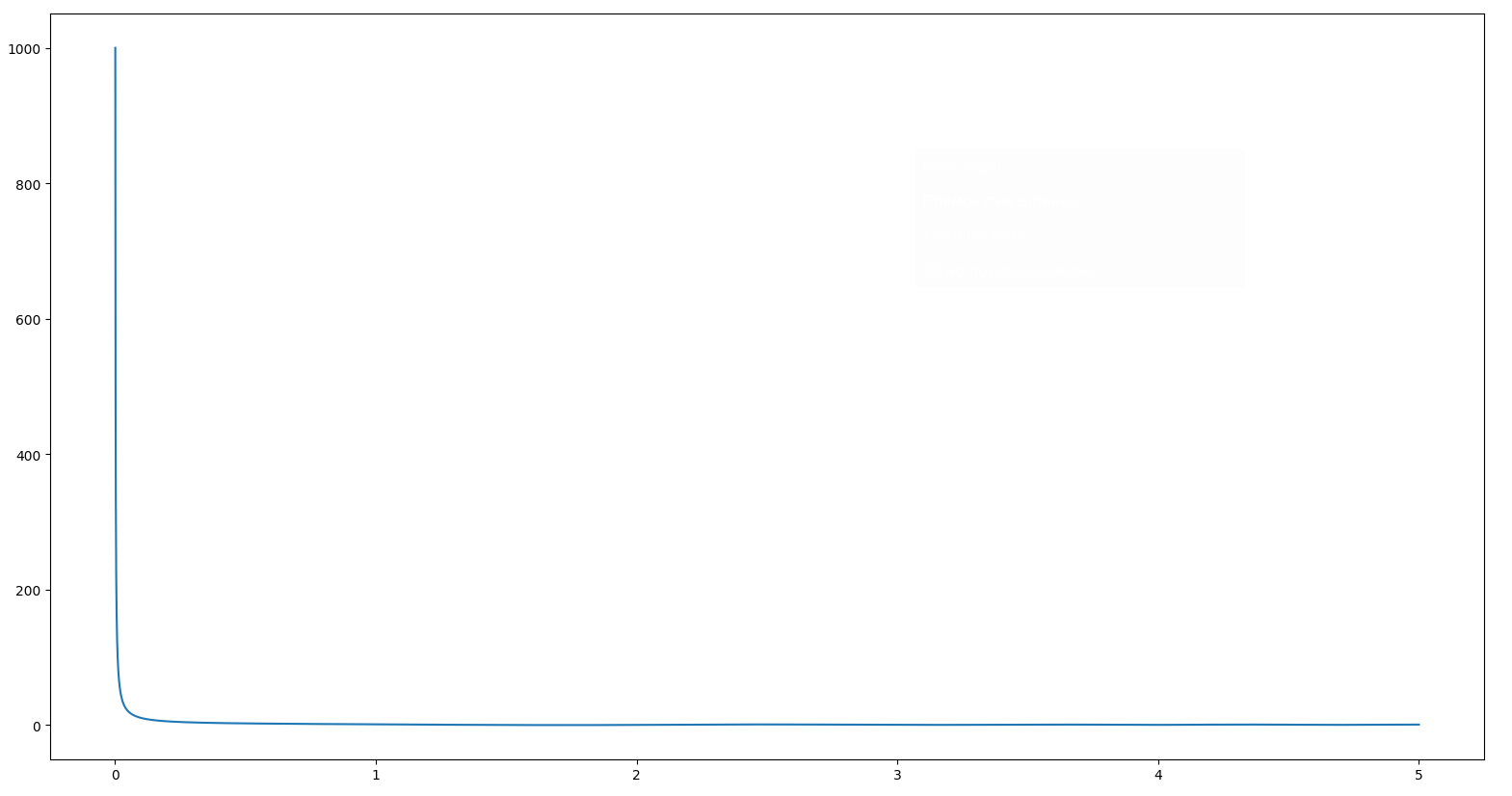
**Варіант 16**. Y(x)=cos(x^2)/x,x=[0...5]

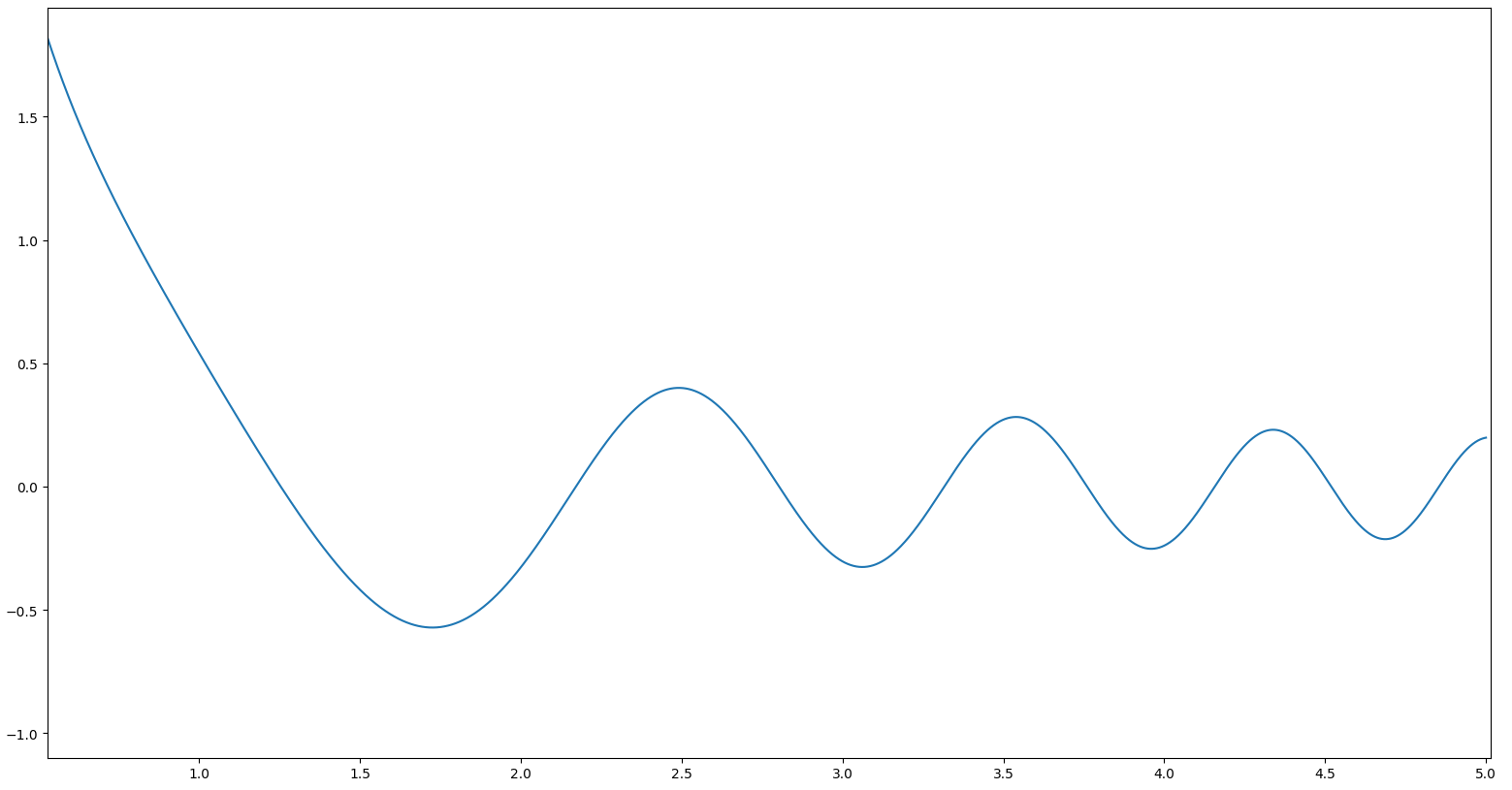
**Хід роботи**

**1. Визначення завдання**

У функції Y(x)=cos(x^2)/x є певна особливість, а саме коли х прямує до нуля, тоді значення функції прямує до безкінечності. Оскільки комп’ютер “мислить” дискретно, тобто поділити на ноль та отримати y=+∞ , то слід змінити область визначення на [0.001 ... 5]. Замінюємо 0 на достатньо малу величину 0.001

У такому випадку наша графік функції, враховуючи межі, виглядатиме ось так

З цієї картинки очевидно чому дорівнює максимум графіку — 0.001. Приблизимо графік та знайдемо мінімум.

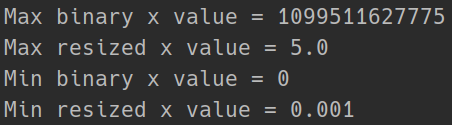
Через бібліотеку scipy знайдемо точне значення мінімуму

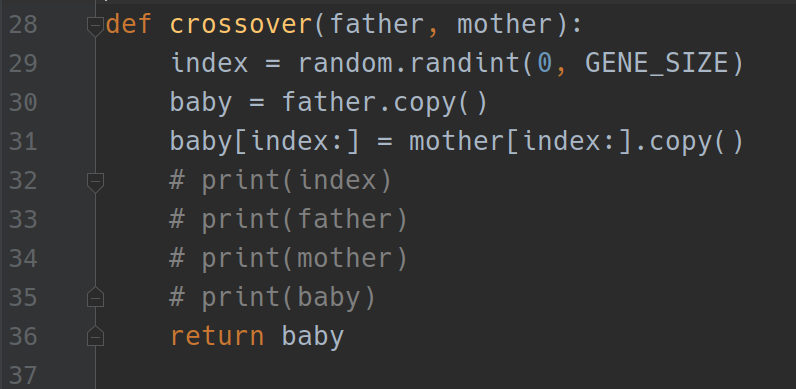


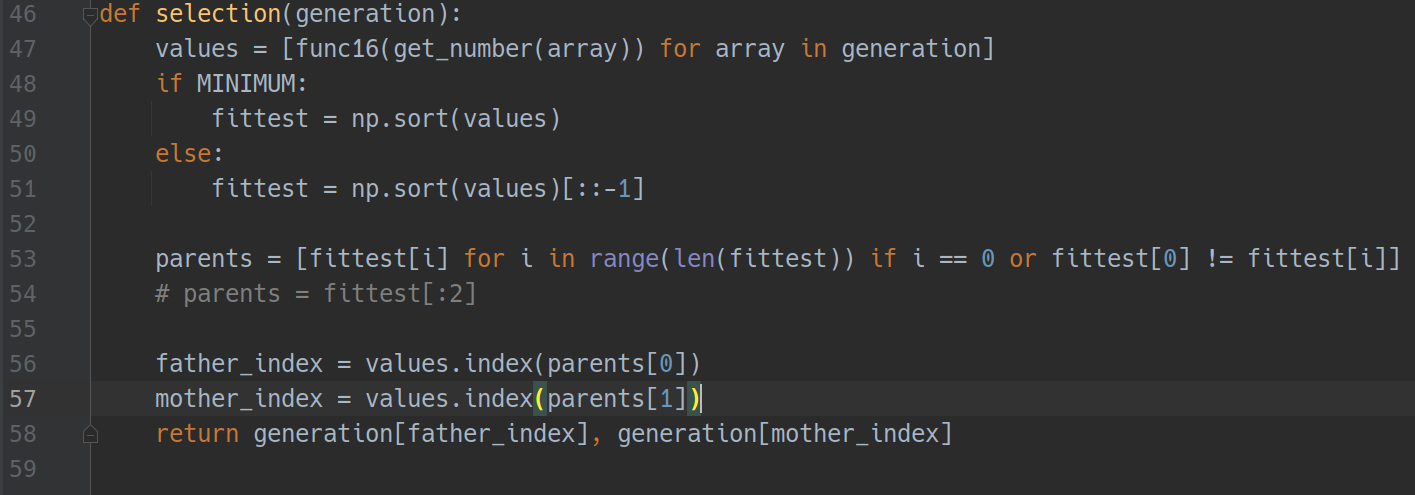
**2. Вирішення завдання**

Генотип, в нашому випадку, це масив з генів-бітів, які можуть дорівнювати 0 або 1. Приклад: [1, 0, 1] => 5. При переведені генотипу в фенотип, тобто з бінарного числа в десяткового цілого числа, значення можуть варіюватися з 0 до достатньо великого числа. Тож проведемо маштабування фенотипу за формулою

 Де END = 5, START = 0.001, MIN\_VALUE = 0, MAX\_VALUE - максимальне значення, що має вигляд (1,1,1...1). Таким чином появляються числа з плавучую точкою.

****

 Такий вигляд має функція випадкового схрещення генотипів батька — найкращий серед усіх генотип, та матері — другий найращих генотип.

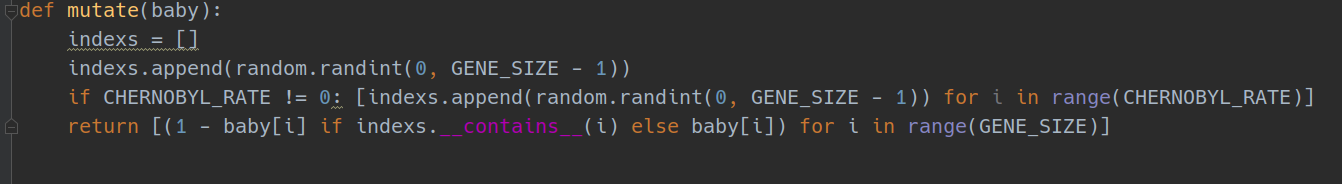
 Залежно від того, що нам треба знайти : мінімум чи максимум — відбувається відповідний відбір через сортування значень функцій та пошуку індексів генотипів батьків. Важливо підмітити, що батько та мати можуть мати одинакові гени(код в коментарі), а можуть і не мати — ця характеритика є взаємнозаміною з підвищенням кількості мутацій, оскільки і то, і то збільшує різноманітність генів.



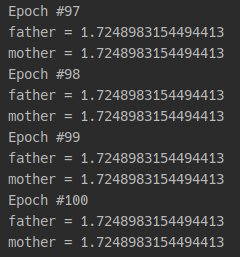
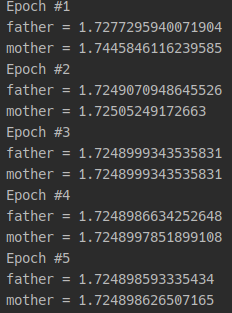
****

Емпірично виявлена залежність, що найточніші результати модель видає тоді, коли розмір популяції в кілька раз більша за кількость генів у генотипі.



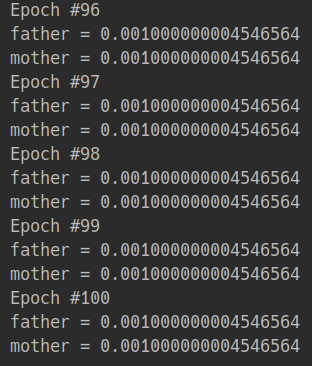
 Так виглядає функція мутації, що залежить від параметрів MUTATION\_RATE — величина, що характерезує у відсотковому вілношені кількість особин популяції, що будуть мутувати. Як мінімум проводиться одна випадкова мутація в генотипі, тобто випадкова 1 стає 0, 0 -> 1. СHERNOBYL\_RATE відповідає за кількість додаткових мутацій. Тобто, згідно вище наведеній конфігурації, загалом мутацій в генотипі, що підпадає під мутацію, максимум відбудеться 4, а мінімум — 1.

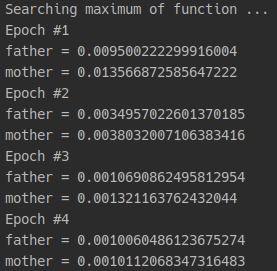
**Результати**

****

Завершення роботи на сотій епосі

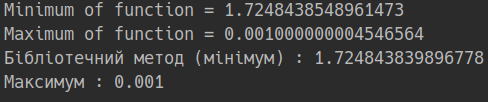
Початок обчислення мінімуму





Початок обчислення мінімуму

Завершення

Результати обчислень та точні значення

**Код**

main.py

from scipy import optimize  
from GA import \*  
  
print("Max binary x value =", MAX\_VALUE, "\nMax resized x value =", resize(MAX\_VALUE))  
print("Min binary x value =", MIN\_VALUE, "\nMin resized x value =", resize(MIN\_VALUE))  
  
print("\nSearching minimum of function ...")  
x\_min = start(True)  
print("\nSearching maximum of function ...")  
x\_max = start(False)  
print("\nMinimum of function =", x\_min, "\nMaximum of function =", x\_max)  
bib = optimize.minimize(func16, 0.5).x[0] # Sequential Least Squares Programming  
print("Бібліотечний метод (мінімум) :", bib)  
print("Максимум :", START)  
x = np.linspace(START, END, 10000)  
y = [func16(x[i]) for i in range(len(x))]  
plt.plot(x, y)  
plt.show()

function.py

from math import cos  
  
GENE\_SIZE = 40  
START = 0.001  
END = 5  
  
  
def func16(x):  
 return cos(x \*\* 2) / x  
  
  
def get\_raw\_number(binary\_array: [int]):  
 return int(''.join([str(num) for num in binary\_array]), 2)  
  
  
MAX\_VALUE = get\_raw\_number([1 for i in range(GENE\_SIZE)])  
MIN\_VALUE = 0  
  
  
def resize(x):  
 return (x \* (END - START) / (MAX\_VALUE - MIN\_VALUE)) + START  
  
  
def get\_number(binary\_array: [int]):  
 return resize(get\_raw\_number(binary\_array))

GA.py

from random import random  
  
from functions import \*  
  
POP\_SIZE = 200  
MUTATION\_RATE = 0.8  
CHERNOBYL\_RATE = 3  
  
  
def start(minimum: bool):  
 global MINIMUM  
 MINIMUM = minimum  
 generation = [[random.randint(0, 1) for i in range(GENE\_SIZE)] for i in range(POP\_SIZE)]  
 return lifetime(generation, 1)  
  
  
def lifetime(generation, epoch):  
 father, mother = selection(generation)  
 father\_value, mother\_value = get\_number(father), get\_number(mother)  
 print("Epoch #" + str(epoch), "\nfather =", father\_value, "\nmother =", mother\_value)  
 if epoch == 100: return father\_value  
 new\_generation = [crossover(father, mother) for i in range(POP\_SIZE)]  
 new\_generation = [mutate(new\_generation[i]) if i < POP\_SIZE \* MUTATION\_RATE else new\_generation[i] for i in  
 range(POP\_SIZE)]  
 return lifetime(new\_generation, epoch + 1)  
  
  
def crossover(father, mother):  
 index = random.randint(0, GENE\_SIZE)  
 baby = father.copy()  
 baby[index:] = mother[index:].copy()  
 # print(index)  
 # print(father)  
 # print(mother)  
 # print(baby)  
 return baby  
  
  
def mutate(baby):  
 indexs = []  
 indexs.append(random.randint(0, GENE\_SIZE - 1))  
 if CHERNOBYL\_RATE != 0: [indexs.append(random.randint(0, GENE\_SIZE - 1)) for i in range(CHERNOBYL\_RATE)]  
 return [(1 - baby[i] if indexs.\_\_contains\_\_(i) else baby[i]) for i in range(GENE\_SIZE)]  
  
  
def selection(generation):  
 values = [func16(get\_number(array)) for array in generation]  
 if MINIMUM:  
 fittest = np.sort(values)  
 else:  
 fittest = np.sort(values)[::-1]  
  
 # parents = [fittest[i] for i in range(len(fittest)) if i == 0 or fittest[0] != fittest[i]]  
 parents = fittest[:2]  
  
 father\_index = values.index(parents[0])  
 mother\_index = values.index(parents[1])  
 return generation[father\_index], generation[mother\_index]

Висновки

В ході лабораторної роботи було знайдено глобальні максимум та мінімум функції за допомогою розробленого генетичного алгоритму, що складається з селекції, рекомбінації та мутації. Вивчені базові поняття та розгалянуті декілька варіантів реалізації алгоритму.

Точність алгоритму сильно залежить від коефіцієнтів описаних вище. При високому рівні мутації втрачаються найкращі значення та потребується більше епох. Обмеження, що батьки не можуть мати одинакові, робить генотипи більшь різноманітними, проте потребує також більшої кількості епох. Емпірично виявлена залежність, що найточніші результати модель видає тоді, коли розмір популяції в кілька раз більша за кількость генів у генотипі.