Міністерство освіти й науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЗВІТ

Про виконання лабораторної роботи №2

З дисципліни «Комп’ютерне моделювання та оптимізація»

**Методи оптимізації нульового порядку для функції однієї змінної**

TI-92 Черноусова Дениса

Перевірив проф. д.т.н. Шушура О. М.

Київ – 2021



g = 3

k = 2

g – остання цифра у номері студентського квитка, а k – передостання.

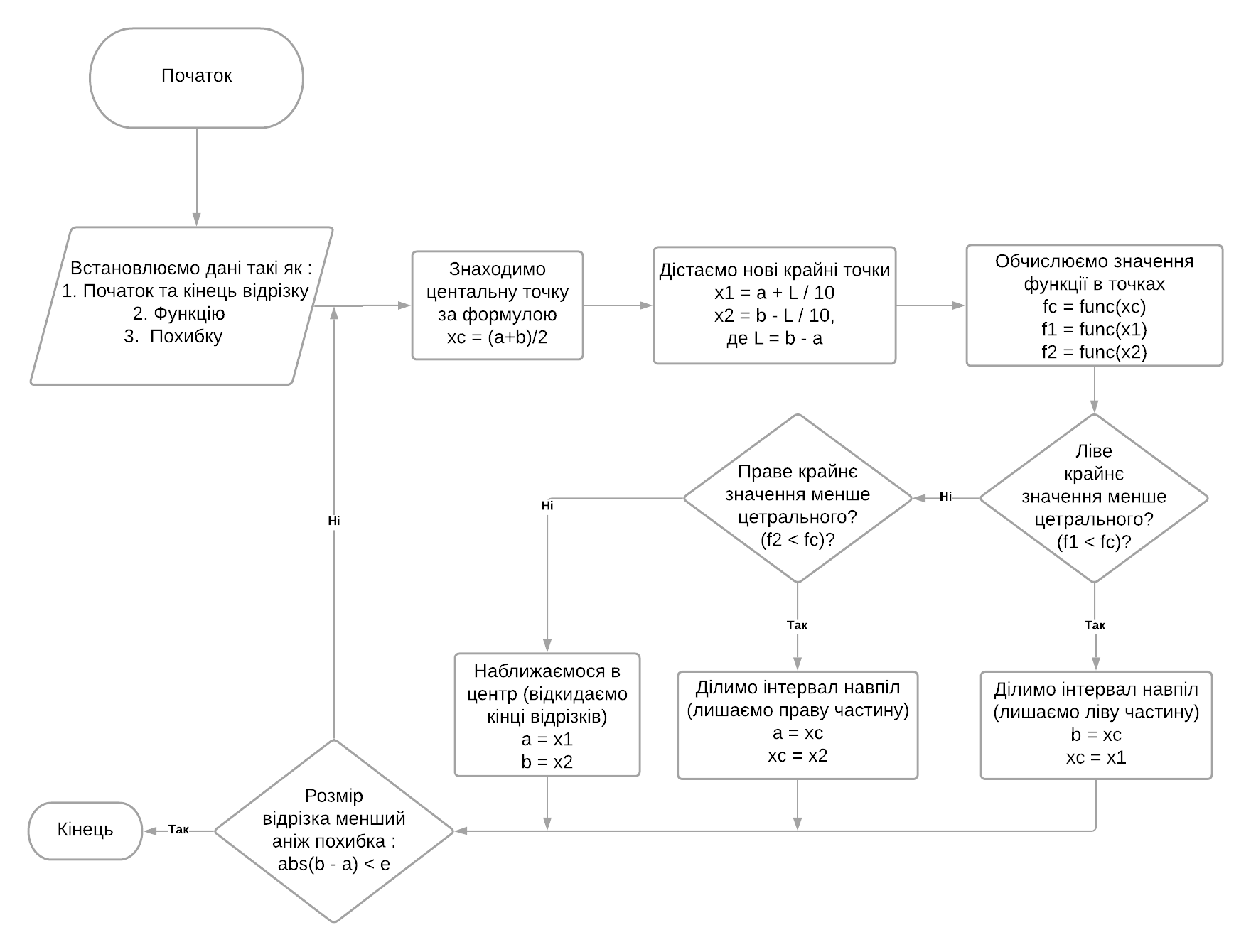
Розробити алгоритми та програмне забезпечення для розв’язку наведених задач оптимізації. Алгоритми представити у вигляді блок-схем або діаграм діяльності UML. Програмне забезпечення розробити на будь-якій сучасній мові програмування. Знайти точне значення розв’язку задачі за допомогою математичних бібліотек та порівняти його зі значенням, отриманим в результаті роботи розробленого програмного забезпечення.

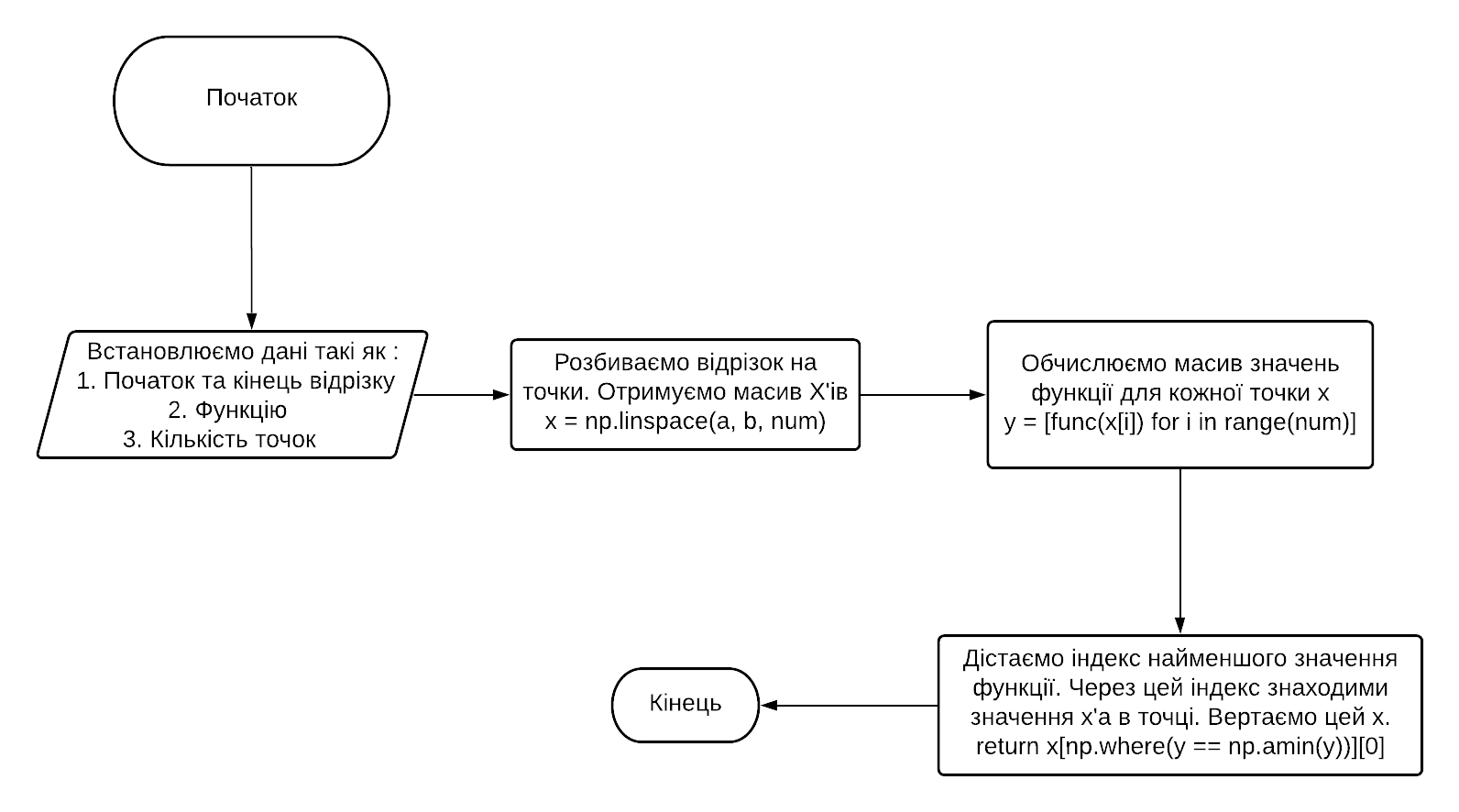
Задачі:

1. Для функції f(x) = *х2+ 2kgx* + *k* знайти точку локального мінімуму на відрізку [-*kg-2, kg+1*] (де *g* – остання цифра у номері студентського квитка, а *k* – передостання) методами:

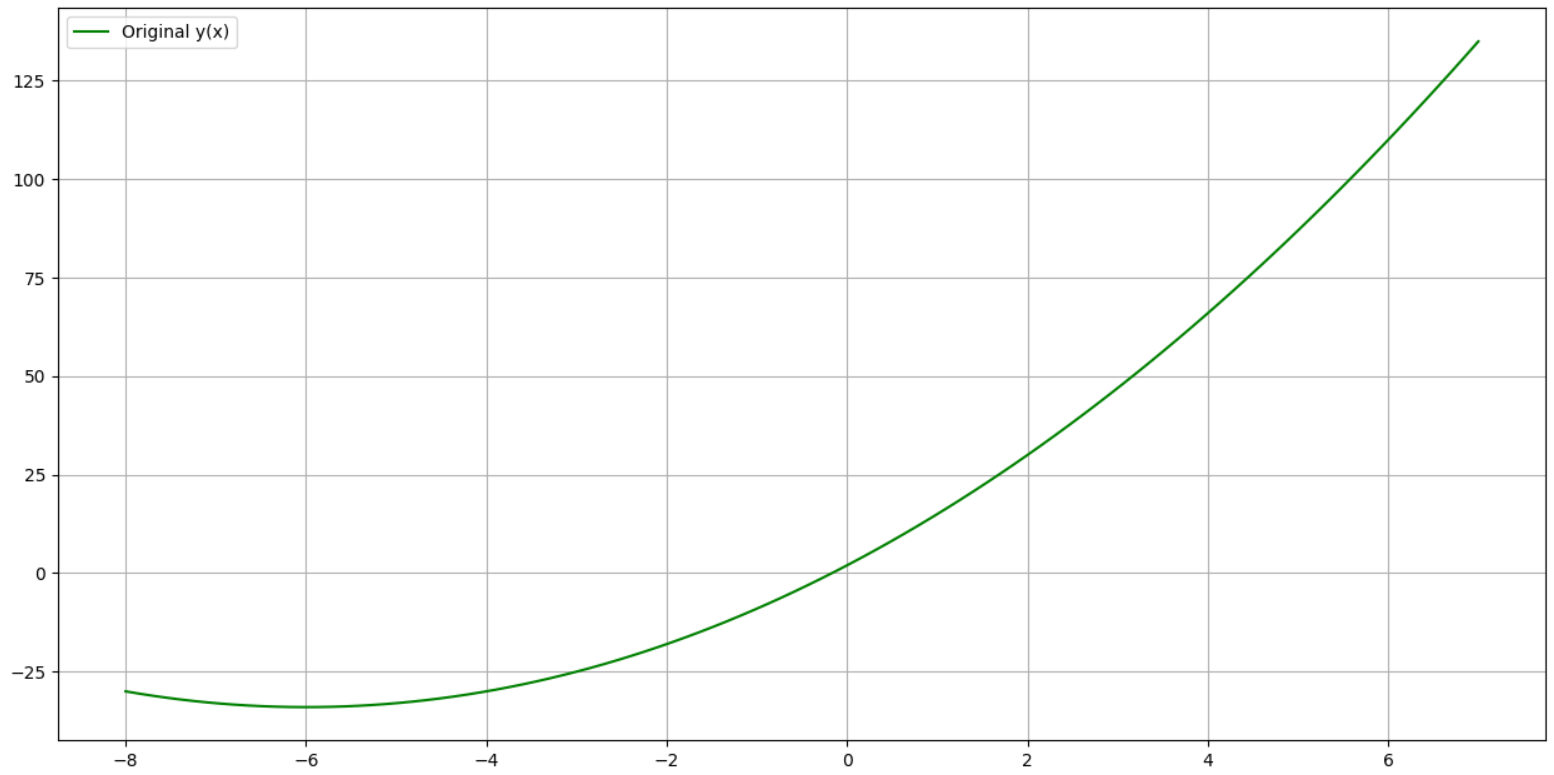
1. поділу інтервалу навпіл (абсолютна точність пошуку має дорівнювати 0,01);
2. випадкового пошуку (кількість точок має дорівнювати 100)

**Алгоритми**

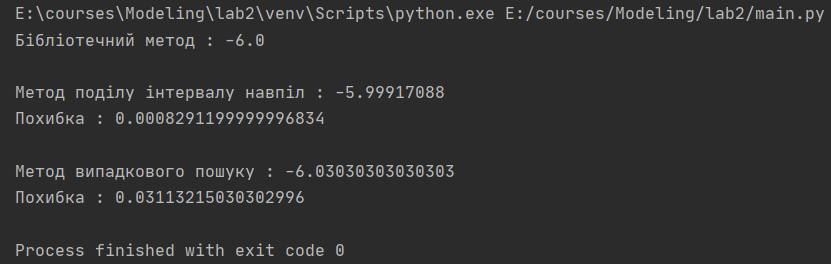
Метод поділу інтервалу навпіл

Метод випадкового пошуку

**Результати роботи**



Графік функції в заданих межах

****

Результати обчислень

**Лістинг**

main.py

from scipy import optimize  
from random\_method import \*  
import matplotlib.pyplot as plt  
from bisection import bisection  
from function import \*  
  
bib = optimize.minimize(MY\_FUNC, MIN, method='SLSQP').x[0] # Sequential Least Squares Programming  
print("Бібліотечний метод :", bib)  
  
bis = bisection(MIN, MAX, MY\_FUNC, 0.01)  
print("\nМетод поділу інтервалу навпіл :", bis )  
print("Похибка :", abs(bis - bib))  
  
ran = random(MIN, MAX, MY\_FUNC, 100)  
print("\nМетод випадкового пошуку :", ran)  
print("Похибка :", abs(bis - ran))  
  
fig, ax1 = plt.subplots(1)  
originalX = np.linspace(MIN, MAX, 100)  
originalY = [MY\_FUNC(originalX[i]) for i in range(100)]  
ax1.plot(originalX, originalY, 'g-', label="Original y(x)")  
ax1.legend()  
ax1.grid()  
plt.show()

function.py

G = 3  
K = 2  
MIN = - K \* G - 2  
MAX = K \* G + 1  
  
  
def MY\_FUNC(x):  
 return x \*\* 2 + 2 \* K \* G \* x + K

random\_method.py

import numpy as np  
  
  
def random(a, b, func, num):  
 x = np.linspace(a, b, num)  
 y = [func(x[i]) for i in range(num)]  
 return x[np.where(y == np.amin(y))][0]

bisection.py

def bisection(a, b, func, e, i=1):  
 # print(i)  
  
 xc = (a + b) / 2  
 L = b - a  
  
 x1 = a + L / 10  
 x2 = b - L / 10

fc = func(xc)  
 f1 = func(x1)  
 f2 = func(x2)  
  
 if f1 < fc:  
 b = xc  
 xc = x1  
 elif f2 < fc:  
 a = xc  
 xc = x2  
 else:  
 a = x1  
 b = x2  
  
 if abs(b - a) < e: return xc  
  
 return bisection(a, b, func, e, i + 1)

distances = []  
  
 for i in range(len(ys\_old[0])):  
 sum = 0  
 for j in range(len(ys\_old)):  
 sum += (ys\_old[j][i] - ys\_new[j][i]) \*\* 2  
 distances.append(sum \*\* 0.5)  
  
 return distances  
  
  
def refine\_array\_over2(cols, rows, ys\_new):  
 ys\_refined = [[], []]  
  
 for i in range(rows):  
 for j in range(cols):  
 ys\_refined[i].append(ys\_new[i][j \* 2])  
  
 return ys\_refined  
  
  
def calculate\_error\_by\_array(arr):  
 error = 0  
 for i in range(len(arr)):  
 error += arr[i] \*\* 2  
  
 return (error / (len(arr) - 1)) \*\* 0.5

**Висновки**

Під час цієї лабораторної роботи були розроблені два алгоритми за методом випадкового пошуку та методом поділу інтервалу навпіл для пошуку мінімуму функції. Згідно спостереженням, метод випадкового пошуку менш точний за метод поділу навпіл та вимагає більше часу для обробки даних, тобто менш продуктивний.