**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з домашньої контрольної роботи з дисципліни

«Методи оптимізації та прийняття рішень»

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Прищепа В.С.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Фіногенов О.Д.*

Київ 2023

ДОМАШНЯ КОНТРОЛЬНА РОБОТА  
МЕТОДИ КООРДИНАТНОГО ТА НАЙШИВДШОГО СПУСКУ

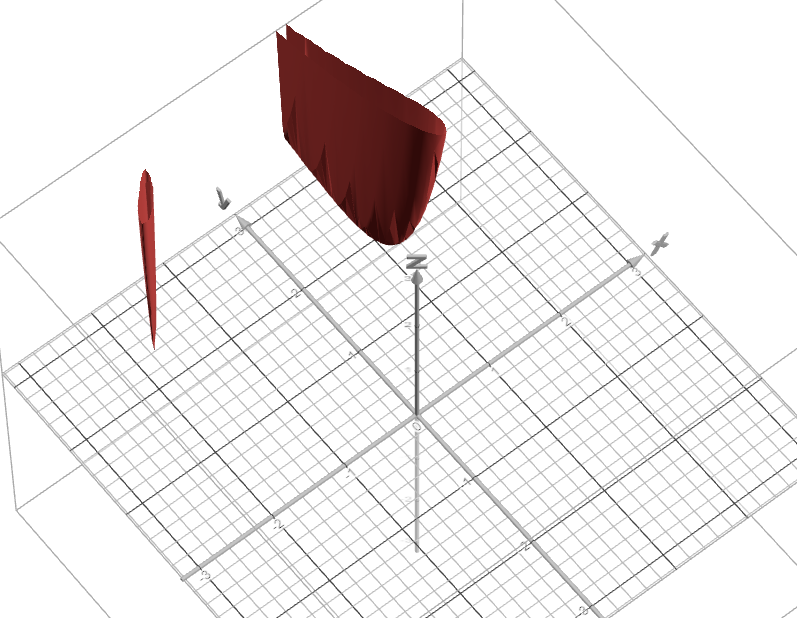
**1. Мета роботи**

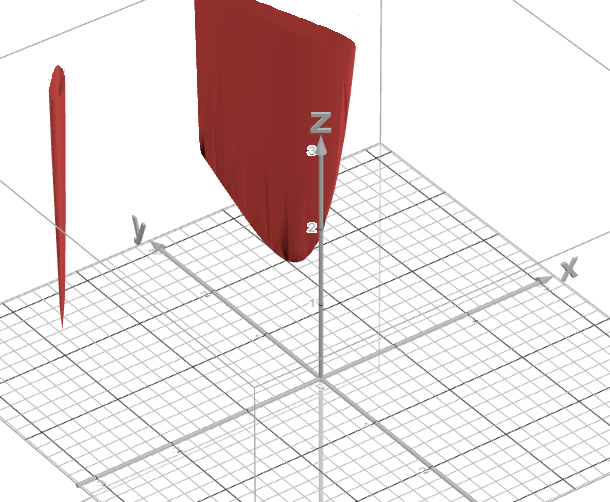
Ознайомитись з методами координатного та найшвидшого спуску для пошуку мінімуму функції багатьох змінних.

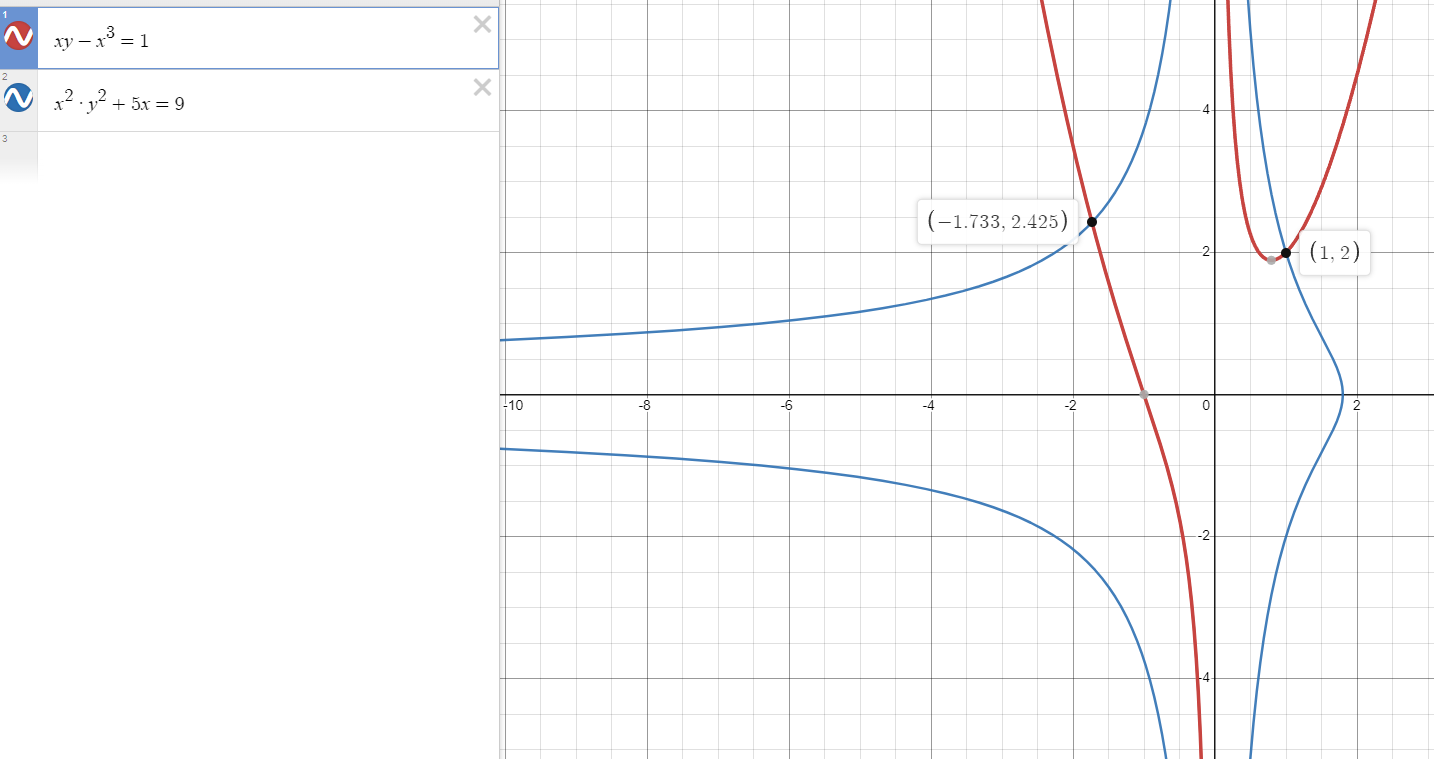
**2. Виконання завдання**

Варіант завдання — 19.

Цільова функція:

Графік ЦФ:

Так, як цільова функція представляє собою суму квадратів двох багаточленів, то, звертаючись до наданого зображення графіку ЦФ, можна припустити, що мінімум цієї функції є нуль і досягаються вони тоді, коли обидва багаточлени рівні нулю. Результат побудови графіків двох багаточленів, за умови, що вони рівні нулю, наступні:

Бачимо, що припущення підтвердилося. Маємо дві точки мінімумів: **(1, 2)** та **(-1.733, 2.425)**.

Для подальших методів мінімізації визначимо стартові точки: **А0(0, 0)** та **А1(-2, 3)**.

Виходячи з того, що варіант 19-тий, то для мінімізації ЦФ будуть використані наступні методи:

- Метод координатного спуску, де вісь обирається випадковим чином, а мінімум локальної функції на вісі шукається методом дихотомії (для цього методу, виходячи з графіку ЦФ, x є [-2, 2], а y є [-1, 3]);

- Метод найшвидшого спуску, де мінімум локальної функції шукається за допомогою методу золотого перетину (крайніми точками є поточна та та, що вирахувана чистим методом найшвидшого спуску).

Код реалізації двох методів наведений нижче у лістингу.

Результати виконання представимо у вигляді зведеної таблиці:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № ітерації | метод координатного спуску | | метод найшвидшого спуску | |
| 0 | A0(0, 0) | A1(-2, 3) | A0(0, 0) | A1(-2, 3) |
| 1 | 16.283323 | 6.885518 | 26.781407 | 56.123840 |
| 2 | 16.283323 | 0.054144 | 18.580010 | 16.848089 |
| 3 | 16.283323 | 0.054144 | 16.315183 | 5.58615 |
| N | 0.000094 | 0.000098 | 0.000095 | 0.000093 |
| Координати знайденого мінімуму | (1.002231, 1.992587) | (0.997592, 2.007235) | (1.002113, 1.992614) | (1.002274, 1.992655) |
| Похибка | 0.000094 | 0.000098 | 0.000095 | 0.000093 |
| КОЦФ | 142 | 115 | 123 | 113 |

**Висновки:** під час виконання ДКР було застосовано методи координатного та найшвидшого спусків у вигляді алгоритмів для знаходження мінімуму цільової функції за допомогою мови програмування Python. Результатом роботи стала зведена таблиця протоколу роботи алгоритмів. Виходячи з результатів, можна стверджувати, що, принаймні для заданої цільової функції та за заданих умов, кращим методом мінімізації є найшвидший спуск, бо кількість обчислень ЦФ дещо менша за інший метод. Треба відзначити, що за заданих умов метод координатного спуску не проявив себе на повну, бо вісь, згідно до варіанту завдання, обиралася випадково. Як можна побачити з результатів зведеної таблиці, це призвело до того, що на наступному кроці могла обратися попередня вісь, через що ітерація змарновувалася на дублювання результату попередньої ітерації. Тому можна стверджувати, що ці два методи спуску є приблизно однаковими за своєю загальною ефективністю.

**Лістинг програм:**

*Метод координатного спуску:*

import random

#parameters

x = -2.0 #x of initial point

y = 3.0 #y of initial point

eps = 0.0001 #epsilon

it = 0 #iterator

#my function

def f(x, y):

return (x \* y - x \*\* 3 - 1) \*\* 2 + (x \*\* 2 \* y \*\* 2 + 5 \* x - 9) \*\* 2

#minimization

while(True):

#random choise between Ox and Oy

chois = random.choice(['x', 'y'])

#case Ox

if chois == 'x':

#calculation of needed parameters

a = -2.0

b = 2.0

d = eps / 2

x1 = (a + b - d) / 2

x2 = (a + b + d) / 2

f1 = f(x1, y)

f2 = f(x2, y)

#method of "dikhotomia"

while(True):

if f1 >= f2:

a = x1

else:

b = x2

x1 = (a + b - d) / 2

x2 = (a + b + d) / 2

f1 = f(x1, y)

f2 = f(x2, y)

if abs(b - a) < eps:

break

x = (a + b) / 2

#case Oy

else:

#calculation of needed parameters

a = -1.0

b = 3.0

d = eps / 2

y1 = (a + b - d) / 2

y2 = (a + b + d) / 2

f1 = f(x, y1)

f2 = f(x, y2)

#method of "dikhotomia"

while(True):

if f1 >= f2:

a = y1

else:

b = y2

y1 = (a + b - d) / 2

y2 = (a + b + d) / 2

f1 = f(x, y1)

f2 = f(x, y2)

if abs(b - a) < eps:

break

y = (a + b) / 2

it += 1

print("No: %5d; x: %10.6f; y: %10.6f; f(x, y): %11.6f" % (it, x, y, f(x, y)))

#conditions of ending minimization

if(f(x, y) < eps):

break

if(it >= 1000):

break

*Метод найшвидшого спуску:*

#parameters

x = -2.0 #x of initial point

y = 3.0 #y of initial point

eps = 0.0001 #epsilon

it = 0 #iterator

t = (3.0 - 5 \*\* 0.5) / 2.0 #parameter of GRS

#my function

def f(x, y):

return (x \* y - x \*\* 3 - 1) \*\* 2 + (x \*\* 2 \* y \*\* 2 + 5 \* x - 9) \*\* 2

#x derivative of f(x, y)

def dx(x, y):

return -34 \* x \* y \*\* 2 + 6 \* x \*\* 5 - 8 \* x \*\* 3 \* y - 2 \* y + 6 \* x \*\* 2 + 4 \* x \*\* 3 \* y \*\* 4 + 50 \* x + 30 \* x \*\* 2 \* y \*\* 2 - 90

#y derivative of f(x, y)

def dy(x, y):

return -34 \* x \*\* 2 \* y - 2 \* x \*\* 4 - 2 \* x + 4 \* x \*\* 4 \* y \*\* 3 + 20 \* x \*\* 3 \* y

#minimization

while(True):

#calculation of needed parameters

xnew = x - 0.1 \* dx(x, y)

ynew = y - 0.1 \* dy(x, y)

cx = x + t \* (xnew - x)

cy = y + t \* (ynew - y)

bx = x + xnew - cx

by = y + ynew - cy

#golden ratio search

while(True):

if f(cx, cy) <= f(bx, by):

xnew = bx

ynew = by

bx = cx

by = cy

cx = x + xnew - bx

cy = y + ynew - by

else:

x = cx

y = cy

cx = bx

cy = by

bx = bx - t \* (xnew - x)

by = by - t \* (ynew - y)

if abs(xnew - x) < eps:

if abs(ynew - y) < eps:

break

x = xnew

y = ynew

it += 1

print("No: %5d; x: %10.6f; y: %10.6f; f(x, y): %11.6f" % (it, x, y, f(x, y)))

#conditions of ending minimization

if(f(x, y) < eps):

break

if(it >= 1000):

break