МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по лабораторной работе №2 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студенты 49/1 группы

Епифанцев В.А.

Григорьян А.А.

Преподаватель:

Нигодин Е.А.

Краснодар 2023

**Цель работы:** изучить задачи квадратичного программирования, а затем решить задачу КП на примере функции.

**Ход работы:** для реализации задачи КП используются условия Куна – Таккера и частные производные.

**Шаги алгоритма**

*Шаг 1*. **Формулировка задачи.** Задать квадратичную целевую функцию вида: **,** где x - вектор переменных, Q - матрица квадратичных коэффициентов, c - вектор линейных коэффициентов. Определить ограничения, если они есть, в виде линейных и квадратичных равенств и/или неравенств.

*Шаг 2.* Проверить матрицу на симметричность и положительную (или отрицательную) определенность. Если не является симметричной и положительно определенной, то задачу КП можно сделать симметричной путем взятия средней матрицы и. Далее необходимо проверить наличие ограничений и привести их к стандартной форме КП, если это необходимо.

*Шаг 3.* **Формирование Лагранжиана.** Ввести множители Лагранжа для каждого ограничения. Сформировать Лагранжиан задачи: где λ - вектор множителей Лагранжа, A - матрица коэффициентов ограничений, b - вектор правых частей ограничений.

*Шаг 4.* **Нахождение стационарных точек.** Найти производные Лагранжиана по переменным x и приравнять их к нулю: . Это уравнение называется условием стационарности. Решение этой системы уравнений даст значения переменных x в стационарной точке.

*Шаг 5.* **Проверка условий оптимальности.** Проверить, что полученное решение удовлетворяет ограничениям: . Затем проверить условие ККТ (условия Каруша-Куна-Таккера), включая неотрицательность множителей Лагранжа: .

*Шаг 6*. **Вычисление значения целевой функции.** Вычислить значение квадратичной целевой функции в найденной оптимальной точке: .

**Особенности реализации задачи квадратичного программирования**

Для создания программы использовался язык программирования Python и среда разработки PyCharm. Для графической визуализации были подключены графические фреймворки MplToolkits и Matplotlib.

Программа выводит в качестве результата окно с отображением функции (рис. 1).

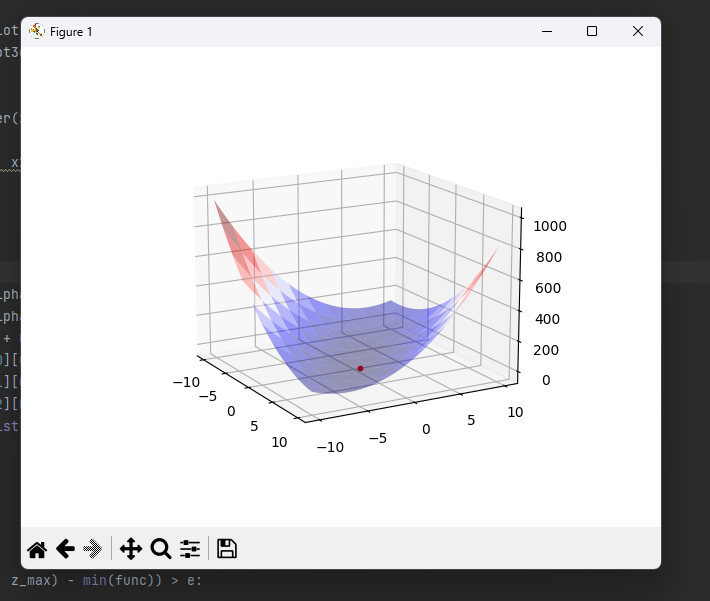


Рисунок 1 – Главное окно программы

**Вывод:** в ходе работы были изучены задачи квадратичного программирования, а затем решена задача КП на примере функции.

**Листинг программы**

Файл lab2.py

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

def f(x, y):

return 2 \* np.power(x, 2) + 3 \* np.power(y, 2) + (4 \* x \* y) - (6 \* x) - (3 \* y)

def simplex\_method(x1, x2):

triangle = []

x0 = float(x1)

z0 = float(x2)

e = 5

alpha = 2

points = [[x0 - alpha / 2, z0 - 0.29 \* alpha],

[x0 + alpha / 2, z0 - 0.29 \* alpha],

[x0, z0 + 0.58 \* alpha]]

func = [f(points[0][0], points[0][1]),

f(points[1][0], points[1][1]),

f(points[2][0], points[2][1])]

triangle.append(list(points))

x\_min = x0

z\_min = z0

y\_min = f(x0, z0)

flag = 0

x\_max = x0

z\_max = z0

while abs(f(x\_max, z\_max) - min(func)) > e:

if flag:

flag = 0

x0, z0 = points[func.index(min(func))]

x0 += points[func.index(max(func))][0]

z0 += points[func.index(max(func))][1]

x0 /= 2

z0 /= 2

points.remove(points[func.index(max(func))])

func.remove(max(func))

x1, z1 = points[func.index(min(func))]

x1 += points[func.index(max(func))][0]

z1 += points[func.index(max(func))][1]

x1 /= 2

z1 /= 2

points.remove(points[func.index(max(func))])

func.remove(max(func))

func.append(f(x0, z0))

points.append([x0, z0])

func.append(f(x1, z1))

points.append([x1, z1])

else:

x\_max, z\_max = points[func.index(max(func))]

points.remove(points[func.index(max(func))])

func.remove(max(func))

x0 = 0 - x\_max

z0 = 0 - z\_max

for value in points:

x0 += value[0]

z0 += value[1]

if f(x0, z0) > max(func):

func.append(f(x\_max, z\_max))

points.append([x\_max, z\_max])

flag = 1

else:

func.append(f(x0, z0))

points.append([x0, z0])

if f(x\_min, z\_min) > min(func):

x\_min, z\_min = points[func.index(min(func))]

y\_min = min(func)

triangle.append(list(points))

x\_min = round(x\_min, 2)

z\_min = round(z\_min, 2)

y\_min = round(y\_min, 2)

return triangle, x\_min, y\_min, z\_min

def makeData():

x = np.linspace(-10, 10, 100)

y = np.linspace(-10, 10, 100)

xgrid, ygrid = np.meshgrid(x, y)

z = f(xgrid, ygrid)

return xgrid, ygrid, z

x, y, z = makeData()

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(x, y, z, rstride=10, cstride=10, alpha=0.4, cmap="seismic")

x1 = 3

x2 = -1

triangle, X, Z, Y = simplex\_method(x1, x2)

for tr in triangle:

if min(tr)[1] > Y:

ax.scatter(tr[0][0], tr[0][1], min(tr), c="red", s=10)

else:

ax.scatter(X, Y, Z, c="red", s=10)

plt.show()