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 №1 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 методы безусловной поисковой оптимизации с использованием производных, и применить один из них.

**Ход работы:** для реализации был выбран алгоритм градиентного спуска с постоянным шагом.

**Шаги алгоритма**

*Шаг 1*. Задать *х ,* 0 < ε < 1, ε 1 > 0, ε 2 > 0, *М* –предельное число итераций. Найти градиент функции в произвольной точке .

*Шаг 2.* Положить *k* = 0.

*Шаг 3.* Вычислить *f(xk).*

*Шаг 4.* Проверить выполнение критерия окончания *|f(x\*)| <* ε1*:*

а) если критерий выполнен, то расчёт закончен и *х\* = xk*;

б) если критерий не выполнен, то перейти к шагу 5.

*Шаг 5.* Проверить выполнение неравенства *k ≥ M:*

а) если неравенство выполнено, то расчет окончен: *х\* = xk*;

б) если нет, то перейти к шагу 6.

*Шаг 6*. Задать величину шага *tk.*

*Шаг 7*. Вычислить *xk+1 = xk - tkf(xk)*.

*Шаг 8*. Проверить выполнение условия

*f*(*xk+1*) *- f*(*xk*) *< 0* (или *|f*(*xk+1*) *- f*(*xk*) *|<*  ε *||f(xk)||2*);

а) если условие выполнено, то перейти к шагу 9;

б) если условие не выполнено, положить  иперейти к шагу 7.  
*Шаг 9*. Проверить выполнение условий

*||xk+1 - xk|| <* ε2*, ||f(xk+1) - f(xk)|| <* ε2:

а) если оба условия выполнены при текущем значении *k* и *k = k -*1, то расчет окончен и *x\* = xk+1*;

б) если хотя бы одно из условий не выполнено, положить *k = k* +1 и перейти к шагу 3.

**Особенности реализации алгоритма градиентного спуска с постоянным шагом**

Для создания программы используется язык программирования среда разработки PyCharm. Для работы используются библиотека numpy и matplotlib.

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

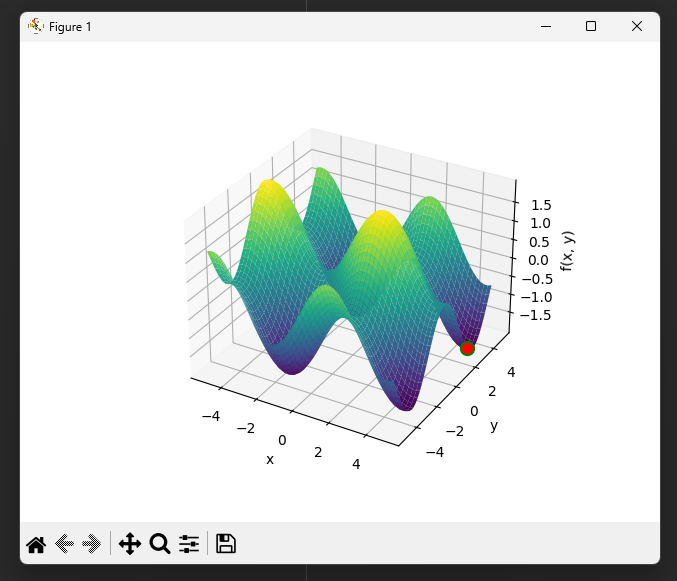
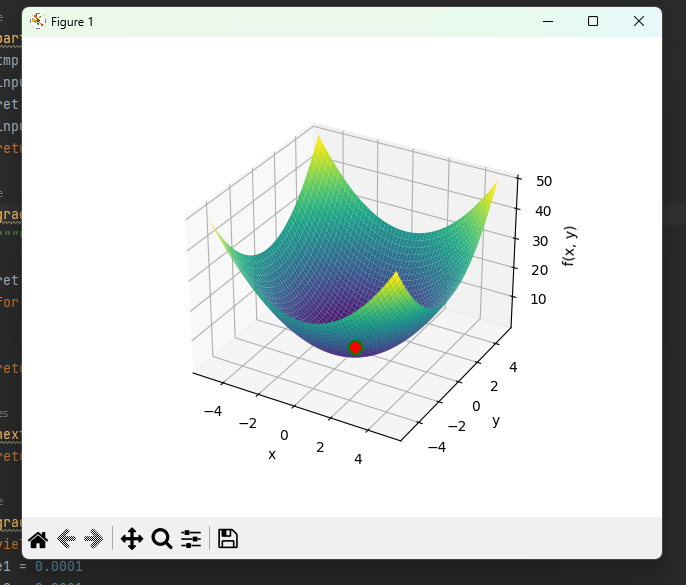


Рисунок 1 – Результат работы программы

**Вывод:** в ходе работы были изучены различные методы безусловной поисковой оптимизации с использованием производных, реализован метод градиентного спуска с постоянным шагом.

**Листинг программы:**

Файл main.py

import numpy as np

import numdifftools as nd

import matplotlib.pyplot as plt

def partial\_function(f\_\_\_, input, pos, value):

tmp = input[pos]

input[pos] = value

ret = f\_\_\_(\*input)

input[pos] = tmp

return ret

def gradient(function, input):

"""Частная произвоздная по каждому из параметров функции f(т.е. градиент)"""

ret = np.empty(len(input))

for i in range(len(input)):

fg = lambda x: partial\_function(function, input, i, x)

ret[i] = nd.Derivative(fg)(input[i])

return ret

def next\_point(x, y, gx, gy, step) -> tuple:

return x - step \* gx, y - step \* gy

def gradient\_descent(function, x0, y0, tk, M):

yield x0, y0, 0, function(x0, y0)

e1 = 0.0001

e2 = 0.0001

k = 0

while True:

(gx, gy) = gradient(function, [x0, y0]) # 3

if np.linalg.norm((gx, gy)) < e1: # Шаг 4. Проверить выполнение критерия окончания

break

if k >= M: # Шаг 5

break

x1, y1 = next\_point(x0, y0, gx, gy, tk) # 7

f1 = function(x1, y1)

f0 = function(x0, y0)

while not f1 < f0: # 8 условие

tk = tk / 2

x1, y1 = next\_point(x0, y0, gx, gy, tk)

f1 = function(x1, y1)

f0 = function(x0, y0)

if np.sqrt((x1 - x0) \*\* 2 + (y1 - y0) \*\* 2) < e2 and abs(f1 - f0) < e2: # 9

x0, y0 = x1, y1

break

else:

k += 1

x0, y0 = x1, y1

yield x0, y0, k, f1

def function(x, y):

return x \*\* 2 + y \*\* 2

def draw\_gradient\_descent(func, grad):

fig = plt.figure()

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

x = np.linspace(-5, 5, 50)

y = x

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

z = func(x, y)

ax.plot\_surface(x, y, z, cmap='viridis')

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.set\_zlabel('f(x, y)')

point, = ax.plot([], [], [], 'ro', markersize=7, zorder=5)

grad = list(grad)

frame = 0

min\_point = min(grad, key=lambda t: t[3])

min\_x, min\_y, min\_f = min\_point[0], min\_point[1], min\_point[3]

ax.plot([min\_x], [min\_y], [min\_f], 'go', markersize=10, zorder=4)

while frame < len(grad):

x\_point, y\_point, \_, \_ = grad[frame]

point.set\_data([x\_point], [y\_point])

point.set\_3d\_properties(func(x\_point, y\_point))

plt.pause(0.01)

frame += 1

plt.show()

x0, y0 = 4, 4

tk = 0.1

M = 50

grad = gradient\_descent(function, x0, y0, tk, M)

draw\_gradient\_descent(function, grad)