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 39 группы

Миленченко А.Р.

Малышев Д.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 Python 3.7 и среда разработки PyCharm. Для графической визуализации используется графический фреймворк PyQt и OpenGL.

В созданной программе одно главное активное окно.

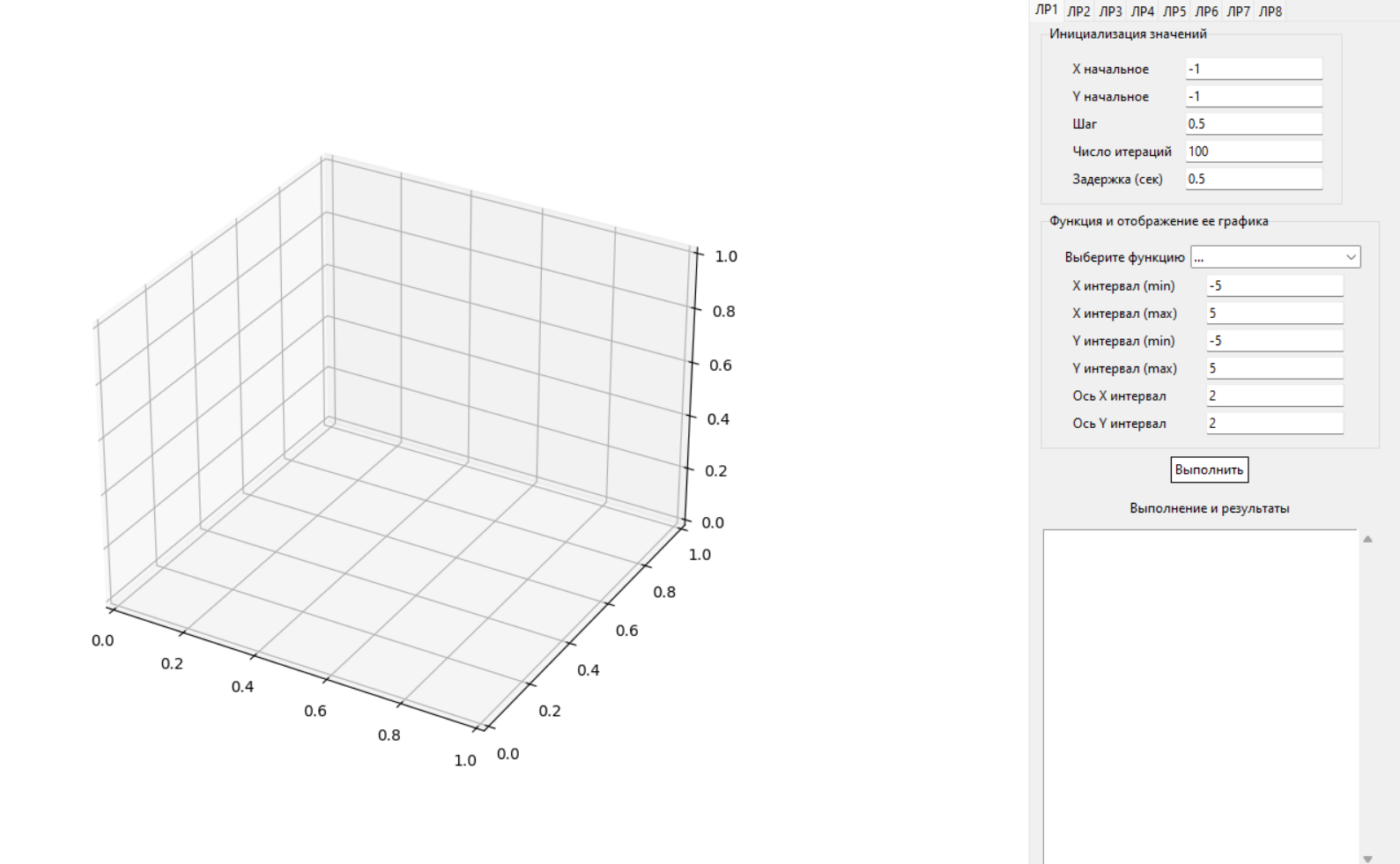


Рисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую лабораторную работу, в данном случае «1», ввести начальную точку метода градиентного спуска с координатами «X» и «Y», задать начальный шаг алгоритма, число итераций (шагов) алгоритма, и задержку между этими итерациями в секундах (используется для наглядного представления работы алгоритма).

В поле «Выполнение и результаты» динамически выводятся результаты работы алгоритма в виде шагов, представленных координатами и значениями оптимизируемой функции в этих координатах. На рисунке 2 показано поле «Выполнение и результаты» до запуска программы.

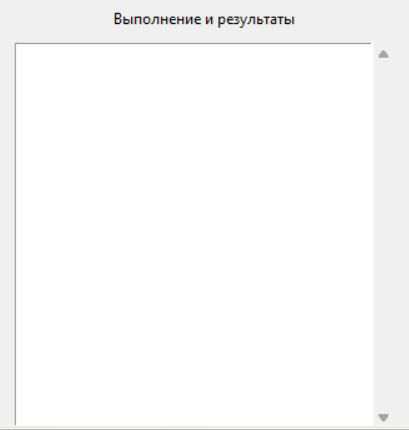


Рисунок 2 – Выполнение и результаты.

Также, в интерфейсе программы есть панель управления «Функция и отображения её графика», в котором можно выбрать необходимую функцию, и задать её параметры:

1. X интервал (интервал на котором строится и минимизируется функция)
2. Y интервал (интервал на котором строится и минимизируется функция)
3. Z масштаб (для масштабирования значений оптимизируемой функции)
4. Оси (здесь можно отключить отображение осей)
5. Ось X интервал (размер отображаемой сетки по оси X)
6. Ось Y интервал (размер отображаемой сетки по оси Y)
7. Сетка (здесь можно отключить отображение координатной сетки)

На рисунке 3 показана панель «Функция и отображение её графика»

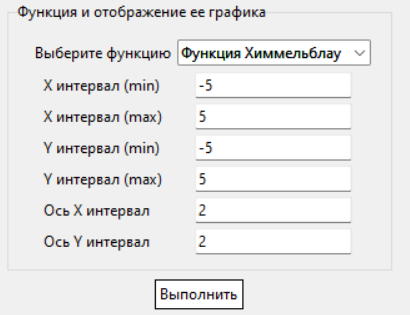


Рисунок 3 – Функция и отображение её графика.

При выборе необходимой функции в меню «Функция и отображение её графика» на главном окне программы отображается необходимая нам функция, как показано на рисунке 4. Причем, более высокие значения функции показана ярко красным цветом, а самые низкие темно синим цветом.

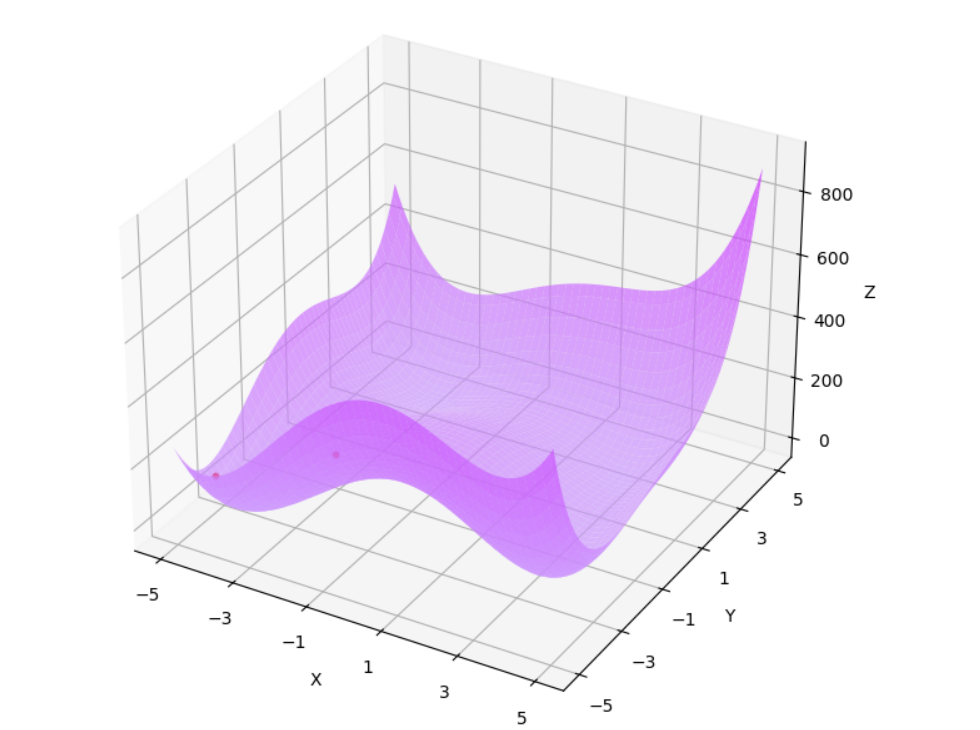


Рисунок 4 – Функция Химмельблау.

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Выполнение и результаты» начнёт динамически заполнятся, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки, которые также отображаются динамически как показано на рисунке 5.

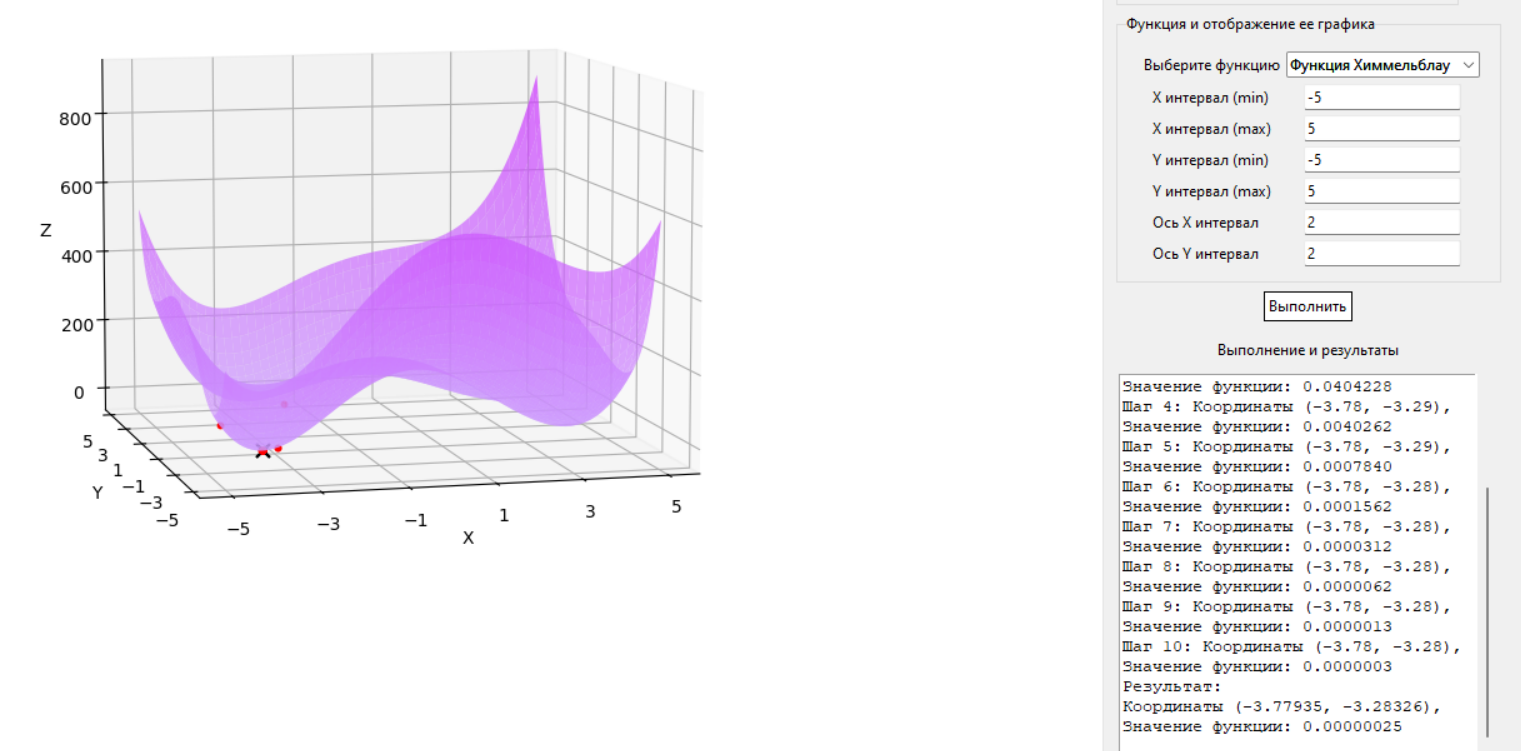


Рисунок 5 – Результат работы программы.

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

**Листинг программы:**

Файл Lab\_1.py

import tkinter as tk

from tkinter import ttk

import time

import numdifftools as nd

from tkinter import scrolledtext

from functions import \*

from matplotlib.colors import LinearSegmentedColormap

def drawLab1(tab, window, ax, canvas):

# Функция для градиента

def gradient(function, input):

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

# Функция для частной производной

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

tmp = input[pos]

input[pos] = value

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

input[pos] = tmp

return ret

# Функция, которая будет выполнена при нажатии кнопки "Выполнить"

def run\_optimization():

x0 = x\_var.get()

y0 = y\_var.get()

step = step\_var.get()

max\_iterations = iterations\_var.get()

delay = delay\_var.get()

ax.cla()

x\_range = np.linspace(x\_interval\_min.get(), x\_interval\_max.get(), 100)

y\_range = np.linspace(y\_interval\_min.get(), y\_interval\_max.get(), 100)

X, Y = np.meshgrid(x\_range, y\_range)

if function\_var.get() != "...":

Z = target\_function(X, Y, function\_var)[0]

else:

return

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_xticks(np.arange(x\_interval\_min.get(), x\_interval\_max.get() + 1, x\_axis\_interval.get()))

ax.set\_yticks(np.arange(y\_interval\_min.get(), y\_interval\_max.get() + 1, y\_axis\_interval.get()))

# Создадим colormap с тремя цветами

colors = [ (0.8, 0.6, 1), (0.8, 0.4, 1), (0.8, 0.2, 1) ]

cmap = LinearSegmentedColormap.from\_list("DX:HR", colors, N=256)

ax.plot\_surface(X, Y, Z, cmap=cmap, alpha=0.7)

target\_func = target\_function(X, Y, function\_var)[1]

results = []

results\_text.config(state=tk.NORMAL)

results\_text.delete(1.0, tk.END)

for k in range(max\_iterations):

(gx, gy) = gradient(target\_func, [x0, y0])

if np.linalg.norm((gx, gy)) < 0.0001:

break

x1, y1 = x0 - step \* gx, y0 - step \* gy

f1 = target\_func(x1, y1)

f0 = target\_func(x0, y0)

while not f1 < f0:

step = step / 2

x1, y1 = x0 - step \* gx, y0 - step \* gy

f1 = target\_func(x1, y1)

f0 = target\_func(x0, y0)

if np.sqrt((x1 - x0) \*\* 2 + (y1 - y0) \*\* 2) < 0.0001 and abs(f1 - f0) < 0.0001:

x0, y0 = x1, y1

break

else:

x0, y0 = x1, y1

results.append((x0, y0, k, f1))

ax.scatter([x0], [y0], [f1], color='red', s=10)

results\_text.insert(tk.END,

f"Шаг {k}: Координаты ({x0:.2f}, {y0:.2f}), Значение функции: {f1:.7f}\n")

results\_text.yview\_moveto(1)

canvas.draw()

window.update()

time.sleep(delay)

length = len(results) - 1

ax.scatter(results[length][0], results[length][1], results[length][3], color='black', marker='x', s=60)

results\_text.insert(tk.END,

f"Результат:\nКоординаты ({results[length][0]:.5f}, "

f"{results[length][1]:.5f}),\nЗначение функции: {results[length][3]:.8f}\n")

results\_text.yview\_moveto(1)

results\_text.config(state=tk.DISABLED)