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 46 группы

Нагалевский А.М.

Прозоров М.С.

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 3.11 и среда разработки PyCharm. Для графической визуализации используется графический фреймворк Tkinter и Matplotlib.

В созданной программе одно главное активное окно.

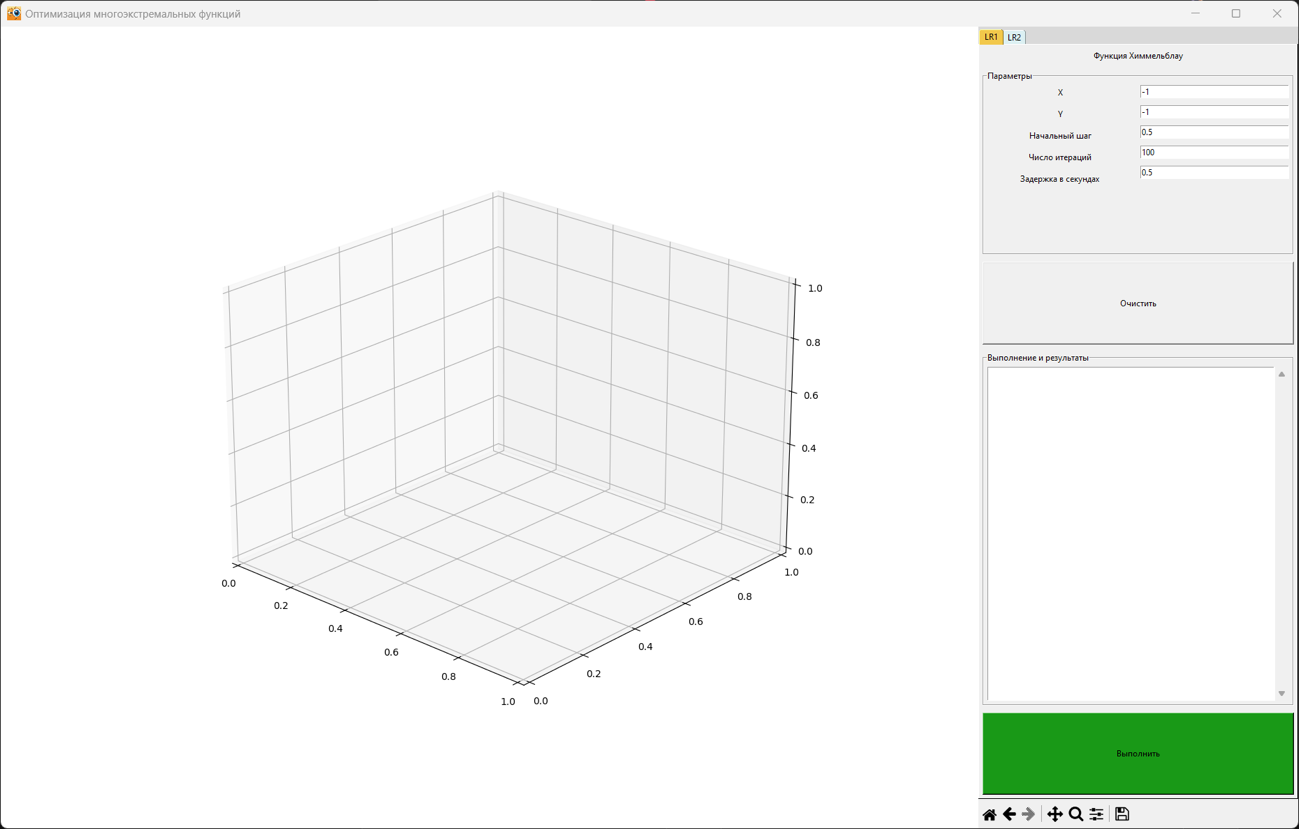


Рисунок 1 – Главное окно программы.

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.

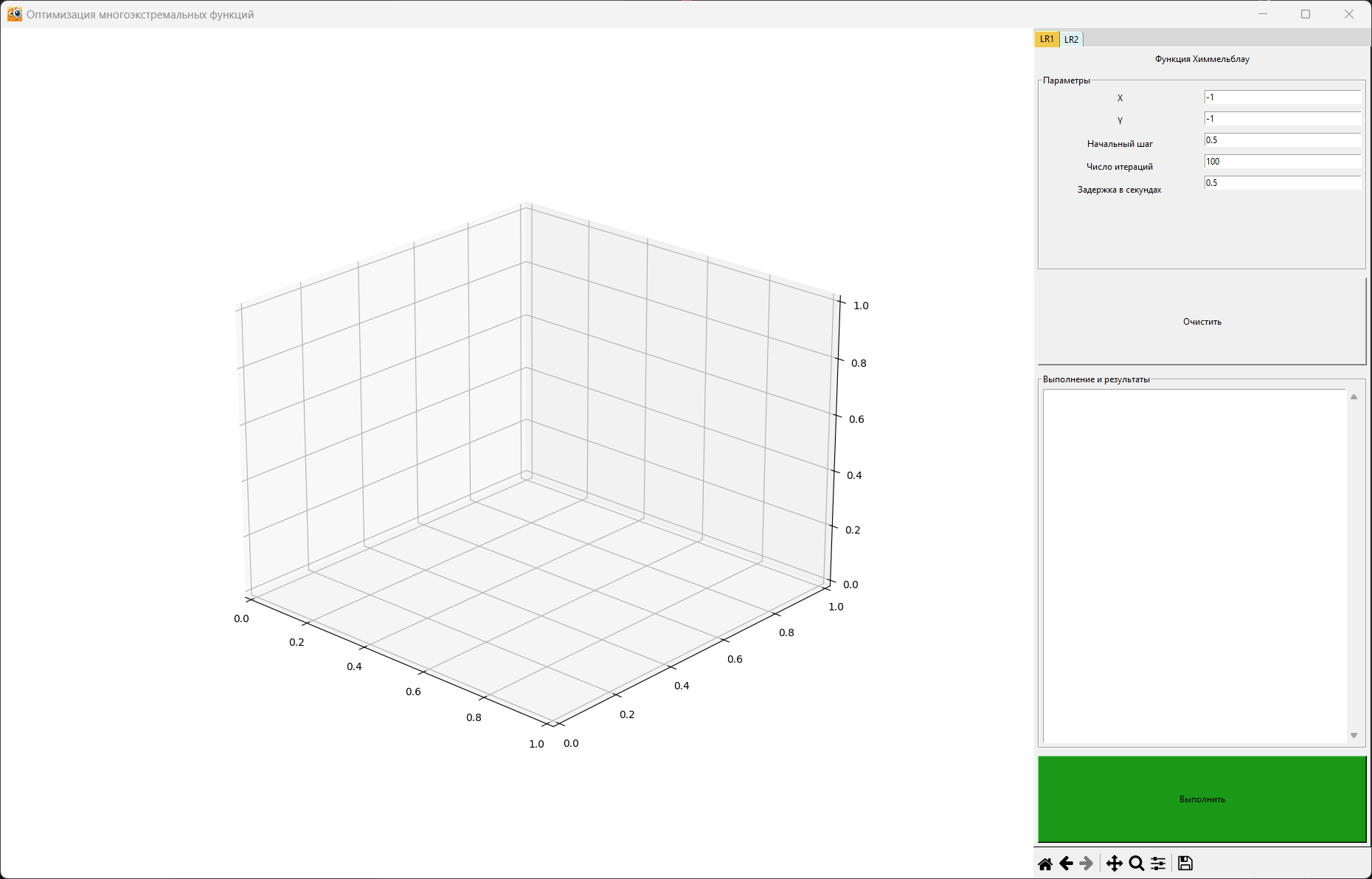


Рисунок 2 – Выполнение и результаты.

При нажатии на кнопку “Выполнить” на главном окне программы отображается необходимая нам функция, как показано на рисунке 4. Причем, более высокие значения функции показана ярко желтым цветом, а самые низкие темно серым цветом.

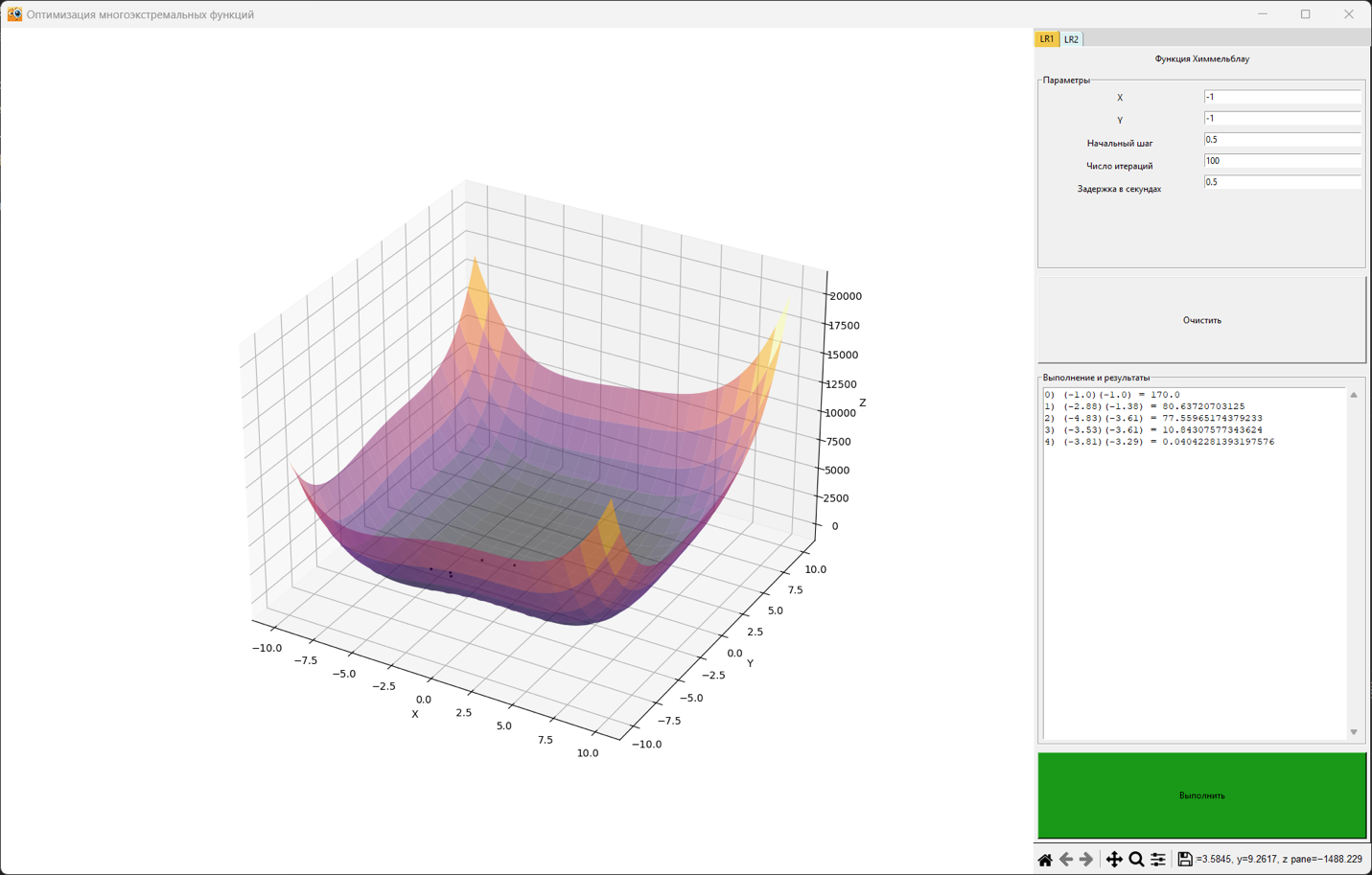


Рисунок 4 – Функция Химмельблау.

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. При полном выполнении программа выводит окно, уведомляющие нас об окончании работы алгоритма.

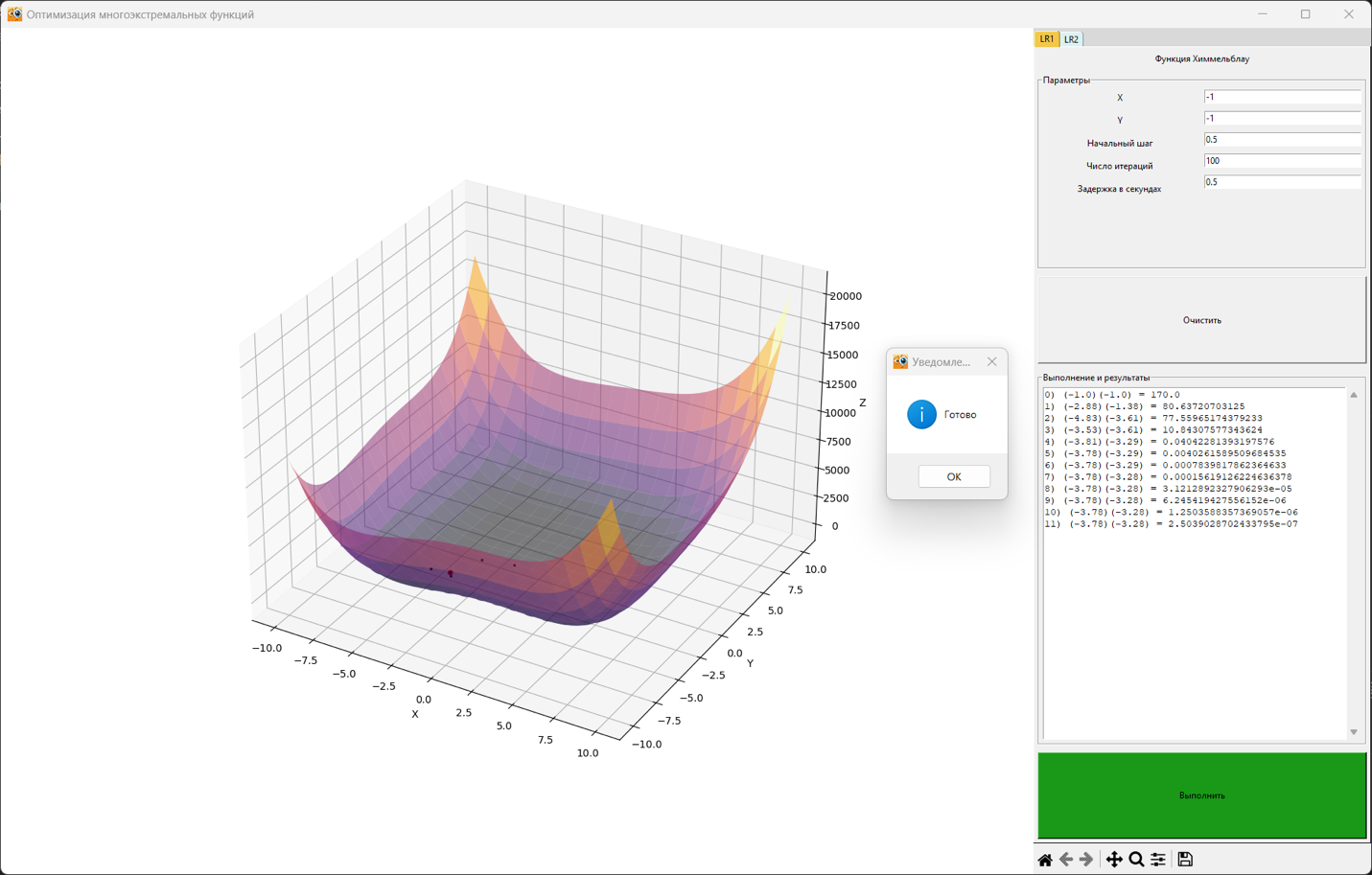


Рисунок 5 – Результат работы программы.

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

**Листинг программы:**

**Файл Gradient.py**

import numpy as np  
import numdifftools as nd  
  
  
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

**Файл main.py**

import tkinter

import time

import sys

from tkinter import \*

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Combobox, Notebook, Style

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from Gradient import make\_data\_lab\_1, funct\_consider

from SLSQP import make\_data\_lab\_2, kp

from functions import \*

def main():

window = Tk()

window.iconbitmap(r'pic/hto.ico')

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

window.geometry("%dx%d" % (width, height))

window.title("Оптимизация многоэкстремальных функций")

fig = plt.figure(figsize=(14, 14))

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

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

toolbar = NavigationToolbar2Tk(canvas, window)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

sky = "#DCF0F2"

yellow = "#F2C84B"

style = Style()

style.theme\_create("dummy", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},

"TNotebook.Tab": {

"configure": {"padding": [5, 1], "background": sky},

"map": {"background": [("selected", yellow)],

"expand": [("selected", [1, 1, 1, 0])]}}})

style.theme\_use("dummy")

tab\_control = Notebook(window)

# Лаба 1

def draw\_lab\_1():

fig.clf()

x, y, z = make\_data\_lab\_1()

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

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

res\_x = txt\_1\_tab\_1.get()

res\_y = txt\_2\_tab\_1.get()

res\_step = txt\_3\_tab\_1.get()

res\_iterations = txt\_4\_tab\_1.get()

x\_cs, y\_cs, z\_cs = funct\_consider(float(res\_x), float(res\_y), float(res\_step), int(res\_iterations))

for i in range(len(x\_cs)):

if i < (len(x\_cs) - 1):

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="black", s=1, marker="s")

else:

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="red")

canvas.draw()

txt\_tab\_1.insert(INSERT, f"{i}) ({round(x\_cs[i], 2)})({round(y\_cs[i], 2)}) = {z\_cs[i]}\n")

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

window.update()

delay = txt\_5\_tab\_1.get()

time.sleep(float(delay))

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_1():

txt\_tab\_1.delete(1.0, END)

tab\_1 = Frame(tab\_control)

tab\_control.add(tab\_1, text="LR1")

main\_f\_tab\_1 = LabelFrame(tab\_1, text="Параметры")

left\_f\_tab\_1 = Frame(main\_f\_tab\_1)

right\_f\_tab\_1 = Frame(main\_f\_tab\_1)

txt\_f\_tab\_1 = LabelFrame(tab\_1, text="Выполнение и результаты")

lbl\_1\_tab\_1 = Label(left\_f\_tab\_1, text="X")

lbl\_2\_tab\_1 = Label(left\_f\_tab\_1, text="Y")

lbl\_3\_tab\_1 = Label(left\_f\_tab\_1, text="Начальный шаг")

lbl\_4\_tab\_1 = Label(left\_f\_tab\_1, text="Число итераций")

lbl\_5\_tab\_1 = Label(tab\_1, text="Функция Химмельблау")

lbl\_6\_tab\_1 = Label(left\_f\_tab\_1, text="Задержка в секундах")

txt\_1\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_1\_tab\_1.insert(0, "-1")

txt\_2\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_2\_tab\_1.insert(0, "-1")

txt\_3\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_3\_tab\_1.insert(0, "0.5")

txt\_4\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_4\_tab\_1.insert(0, "100")

txt\_5\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_5\_tab\_1.insert(0, "0.5")

txt\_tab\_1 = scrolledtext.ScrolledText(txt\_f\_tab\_1)

btn\_del\_tab\_1 = Button(tab\_1, text="Очистить", command=delete\_lab\_1)

btn\_tab\_1 = Button(tab\_1, text="Выполнить", foreground="black", background="#199917", command=draw\_lab\_1)

lbl\_5\_tab\_1.pack(side=TOP, padx=5, pady=5)

main\_f\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_1.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_1.pack(side=RIGHT, fill=BOTH, expand=True)

lbl\_1\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_2\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_3\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_4\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_6\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_1\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_2\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_3\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_4\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_5\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_tab\_1.pack(padx=5, pady=5, fill=BOTH, expand=True)

btn\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

# Лаба 2

def draw\_lab\_2():

fig.clf()

x, y, z = make\_data\_lab\_2()

res\_x = txt\_1\_tab\_2.get()

res\_y = txt\_2\_tab\_2.get()

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

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

x\_cs = []

y\_cs = []

z\_cs = []

for i, point in kp(res\_x, res\_y):

x\_cs.append(point[0])

y\_cs.append(point[1])

z\_cs.append(point[2])

for i in range(len(x\_cs)):

if i < (len(x\_cs) - 1):

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="black", s=1, marker="s")

else:

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="red")

txt\_tab\_2.insert(INSERT, f"{i}) ({round(x\_cs[i], 2)})({round(y\_cs[i], 2)}) = {round(z\_cs[i], 4)}\n")

canvas.draw()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

window.update()

delay = txt\_3\_tab\_2.get()

time.sleep(float(delay))

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_2():

txt\_tab\_2.delete(1.0, END)

tab\_2 = Frame(tab\_control)

tab\_control.add(tab\_2, text="LR2")

main\_f\_tab\_2 = LabelFrame(tab\_2, text="Параметры")

left\_f\_tab\_2 = Frame(main\_f\_tab\_2)

right\_f\_tab\_2 = Frame(main\_f\_tab\_2)

txt\_f\_tab\_2 = LabelFrame(tab\_2, text="Выполнение и результаты")

lbl\_1\_tab\_2 = Label(tab\_2, text="Функция :\n2 \* x₁² + 3 \* x₂² + 4 \* x₁ \* x₂ - 6 \* x₁ - 3 \* x₂")

lbl\_2\_tab\_2 = Label(left\_f\_tab\_2, text="X")

lbl\_3\_tab\_2 = Label(left\_f\_tab\_2, text="Y")

lbl\_4\_tab\_2 = Label(left\_f\_tab\_2, text="Задержка в секундах")

txt\_1\_tab\_2 = Entry(right\_f\_tab\_2)

txt\_1\_tab\_2.insert(0, "10")

txt\_2\_tab\_2 = Entry(right\_f\_tab\_2)

txt\_2\_tab\_2.insert(0, "10")

txt\_3\_tab\_2 = Entry(right\_f\_tab\_2)

txt\_3\_tab\_2.insert(0, "0.5")

txt\_tab\_2 = scrolledtext.ScrolledText(txt\_f\_tab\_2)

btn\_del\_tab\_2 = Button(tab\_2, text="Очистить", command=delete\_lab\_2)

btn\_tab\_2 = Button(tab\_2, text="Выполнить", foreground="black", background="#199917", command=draw\_lab\_2)

lbl\_1\_tab\_2.pack(side=TOP, padx=5, pady=5)

main\_f\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_2.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_2.pack(side=RIGHT, fill=BOTH, expand=True)

lbl\_2\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_3\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_4\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_1\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_2\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_3\_tab\_2.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_tab\_2.pack(padx=5, pady=5, fill=BOTH, expand=True)

btn\_tab\_2.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_2.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_2.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

tab\_control.pack(side=RIGHT, fill=BOTH, expand=True)

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()