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 36 группы

Гаранина Л.В.

Агаджанян А.С.

Преподаватель:

Нигодин Е.А.

Краснодар

2025

**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. Он состоит из 9 основных шагов.

**Шаги алгоритма**

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

2) Положить k = 0.

3) Вычислить f(xk).

4) Если |f(x\*)| < ε 1, то расчет закончен и х\* = xk, иначе шаг 5.

5) Если k ≥ M, то расчет закончен и х\* = xk, иначе шаг 6.

6) Задать величину шага tk.

7) Вычислить xk+1 = xk - tkf(xk).

8) Если f(xk+1) - f(xk) < 0 (или |f(xk+1) - f(xk) |< ε ||f(xk)||2), то шаг 9, иначе положить  и перейти к шагу 7.

9) Если ||xk+1 - xk|| < ε 2, ||f(xk+1) - f(xk)|| < ε 2 верно при текущем значении k и k = k -1, то расчет окончен и x\* = xk+1, иначе положить k = k +1 и перейти к шагу 3.

**Реализация алгоритма градиентного спуска с постоянным шагом**

Программа написана на языке Python в VS Code. Для вывода графика используется библиотека matplotlib, для создания графического интерфейса – tkinter.

Программа выводит главное окно, состоящее из области для построения графика и меню ввода данных и вывода результатов (рисунок 1).

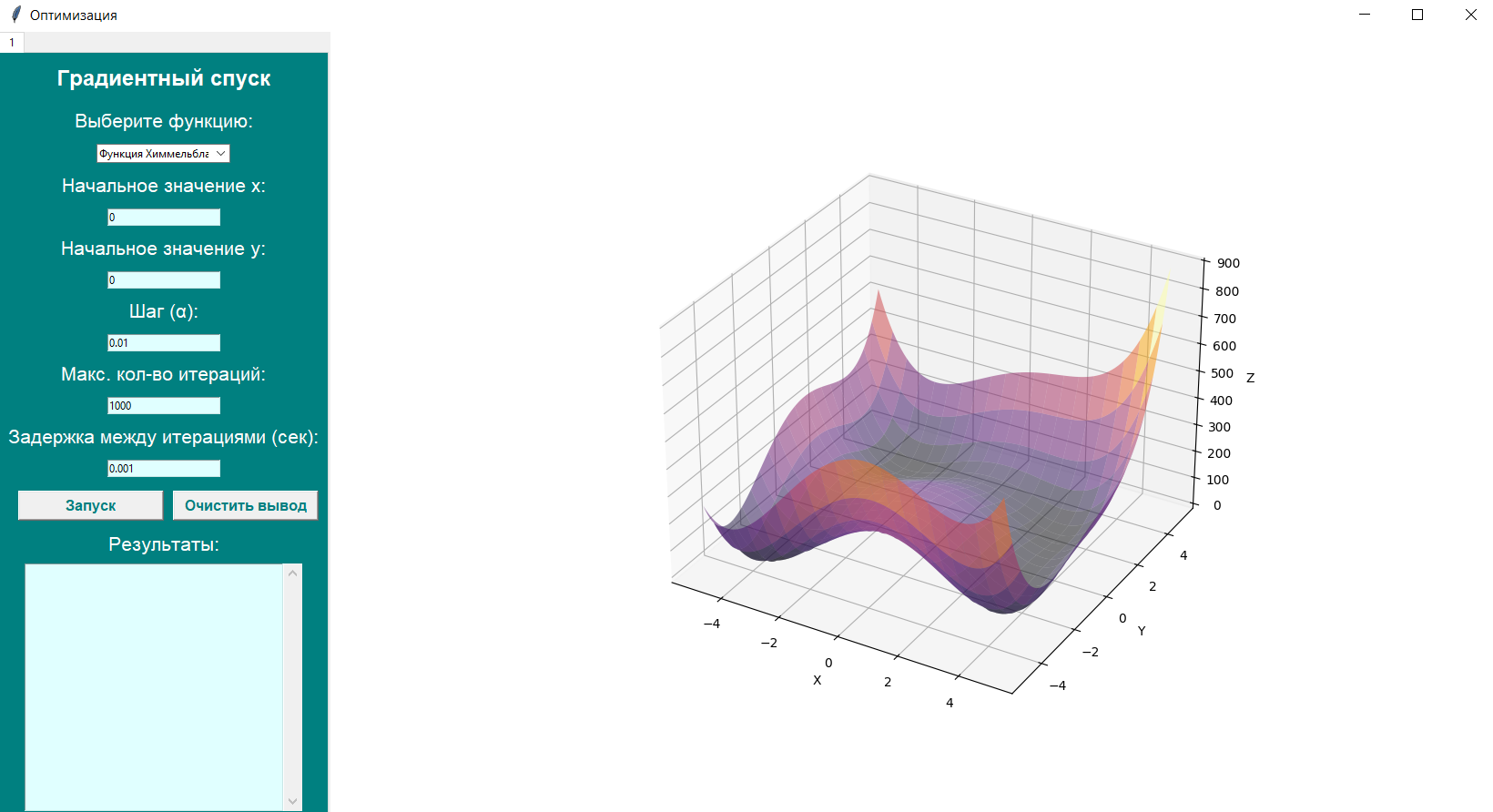


Рисунок 1 – Главное окно программы

В меню можно выбрать функцию, задать значения x, y, шаг, количество итераций и задержку между итерациями в секундах. На рисунке 2 показано основное меню для ввода параметров.

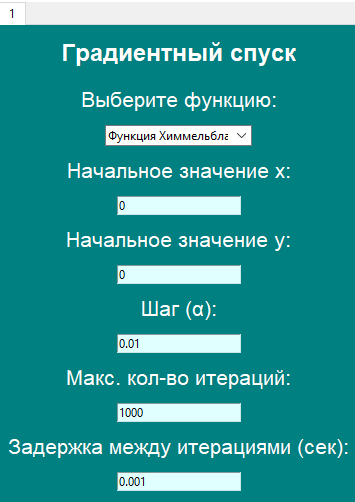


Рисунок 2 – Окно ввода значений

На рисунке 3 изображены основные кнопки. Чтобы запустить алгоритм, необходимо нажать на "Запуск". При необходимости поле вывода можно очистить, нажав "Очистить вывод".



Рисунок 3 – Командные кнопки

Под вводом значений и кнопками находится окно вывода результатов метода. Это можно увидеть на рисунке 4.

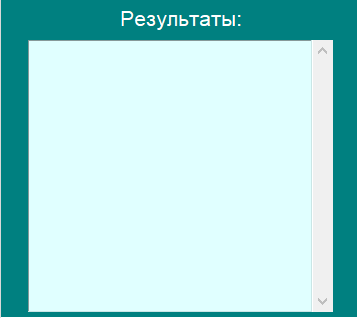


Рисунок 4 – Окно вывода результатов

При выборе функции сразу в своей области отображается график, в котором самые высокие значения показаны желтым цветом, самые низкие – темно-фиолетовым. Это показано на рисунке 5.

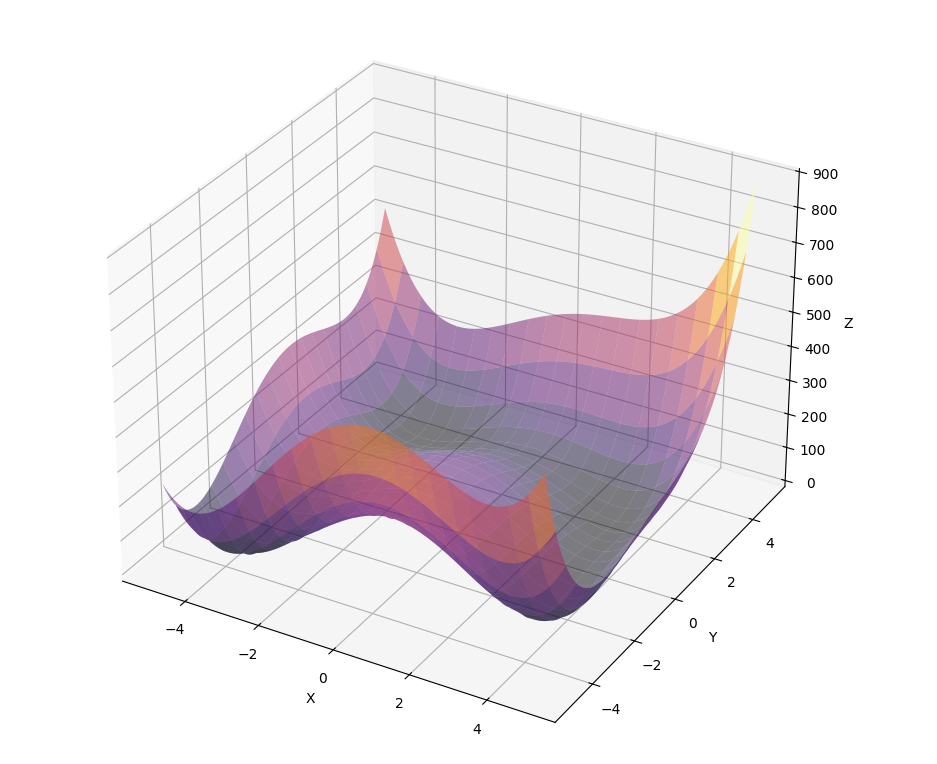


Рисунок 5 – Вывод графика

**Вывод:** проделав лабораторную работу, были изучены различные методы безусловной поисковой оптимизации с использованием производных, реализован метод градиентного спуска с постоянным шагом и сам графический интерфейс программы.

**Листинг программы**

Файл gradient.py:

import numpy as np

from functions import \*

def gradient\_descent(x\_start, y\_start, step\_size, max\_iterations, selected\_function):

epsilon1 = 0.001 # Критерий окончания |f(xk)| < ε1

epsilon2 = 0.001 # Критерий остановки ||xk+1 - xk|| < ε2 и ||f(xk+1) - f(xk)|| < ε2

epsilon = 0.001

x = np.array([x\_start, y\_start])

history = []

k = 0

while True:

if selected\_function == "Функция Химмельблау":

f\_xk = himmelblau(x)

grad = himmelblau\_gradient(x)

elif selected\_function == "Функция Бута":

f\_xk = booth(x)

grad = booth\_gradient(x)

elif selected\_function == "Функция Сферы":

f\_xk = sphere(x)

grad = sphere\_gradient(x)

if abs(f\_xk) < epsilon1:

history.append((x[0], x[1], f\_xk))

break

if k >= max\_iterations:

history.append((x[0], x[1], f\_xk))

break

tk = step\_size

while True:

x\_new = x - tk \* grad

if selected\_function == "Функция Химмельблау":

f\_xk\_new = himmelblau(x\_new)

elif selected\_function == "Функция Бута":

f\_xk\_new = booth(x\_new)

elif selected\_function == "Функция Сферы":

f\_xk\_new = sphere(x\_new)

if f\_xk\_new - f\_xk < 0 or np.abs(f\_xk\_new - f\_xk) < epsilon \* (np.linalg.norm(f\_xk) \*\* 2):

break

else:

tk /= 2

if np.linalg.norm(x\_new - x) < epsilon2 and np.linalg.norm(f\_xk\_new - f\_xk) < epsilon2:

history.append((x\_new[0], x\_new[1], f\_xk\_new))

break

else:

x = x\_new

k += 1

history.append((x[0], x[1], f\_xk\_new))

return history

Файл function.py:

# Функция Химмельблау

def himmelblau(x):

return (x[0]\*\*2 + x[1] - 11)\*\*2 + (x[0] + x[1]\*\*2 - 7)\*\*2

def himmelblau\_gradient(x):

df\_dx = 4 \* x[0] \* (x[0] \*\* 2 + x[1] - 11) + 2 \* (x[0] + x[1] \*\* 2 - 7)

df\_dy = 2 \* (x[0] \*\* 2 + x[1] - 11) + 4 \* x[1] \* (x[0] + x[1] \*\* 2 - 7)

return np.array([df\_dx, df\_dy])

# Функция Бута

def booth(x):

return (x[0] + 2\*x[1] - 7)\*\*2 + (2\*x[0] + x[1] - 5)\*\*2

def booth\_gradient(x):

df\_dx = 2 \* (x[0] + 2 \* x[1] - 7) + 4 \* (2 \* x[0] + x[1] - 5)

df\_dy = 4 \* (x[0] + 2 \* x[1] - 7) + 2 \* (2 \* x[0] + x[1] - 5)

return np.array([df\_dx, df\_dy])

# Функция Сферы

def sphere(x):

return sum(xi\*\*2 for xi in x)

def sphere\_gradient(x):

return 2 \* np.array(x)

Файл main.py:

import tkinter as tk

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Notebook, Combobox

from matplotlib import pyplot as plt

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

from gradient import \*

from functions import \*

# Функция для отрисовки поверхности

def plot\_surface(ax, selected\_function):

ax.cla()

x = np.linspace(-5, 5, 100)

y = np.linspace(-5, 5, 100)

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

if selected\_function == "Функция Химмельблау":

z = np.vectorize(lambda x, y: himmelblau(np.array([x, y])))(x, y)

elif selected\_function == "Функция Бута":

z = np.vectorize(lambda x, y: booth(np.array([x, y])))(x, y)

elif selected\_function == "Функция Сферы":

z = np.vectorize(lambda x, y: sphere(np.array([x, y])))(x, y)

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

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

def main():

window = tk.Tk()

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

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

window.title("Оптимизация")

# Меню

menu\_frame = tk.Frame(window, width=300, bg="#008080")

menu\_frame.pack(side=tk.LEFT, fill=tk.Y)

# Для график

plot\_frame = tk.Frame(window, bg="#008080")

plot\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# График

fig = plt.figure(figsize=(10, 10))

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

canvas = FigureCanvasTkAgg(fig, master=plot\_frame)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Панель инструментов для графика

toolbar = NavigationToolbar2Tk(canvas, plot\_frame)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Функция для выполнения градиентного спуска

def run\_gradient\_descent():

try:

selected\_function = combobox\_functions.get()

x\_start = float(entry\_x.get())

y\_start = float(entry\_y.get())

step\_size = float(entry\_step.get())

max\_iterations = int(entry\_iterations.get())

delay = float(entry\_delay.get())

history = gradient\_descent(x\_start, y\_start, step\_size, max\_iterations, selected\_function)

plot\_surface(ax, selected\_function)

for i, (x, y, z) in enumerate(history):

ax.scatter(x, y, z, c="black", s=10, marker="o")

if i == len(history) - 1 and len(history) - 1 < max\_iterations:

ax.scatter(x, y, z, c="red", s=50, marker="x")

canvas.draw()

output\_text.insert(tk.END, f"{i + 1}) ({x:.2f}, {y:.2f}) = {z:.4f}\n")

output\_text.yview\_moveto(1)

window.update()

window.after(int(delay \* 1000))

if len(history)-1 == max\_iterations:

messagebox.showwarning("Предупреждение", "Достигнуто максимальное количество итераций, возможно, не найден минимум.")

else: messagebox.showinfo("Уведомление", "Градиентный спуск завершен!")

except ValueError:

messagebox.showerror("Ошибка", "Пожалуйста, введите корректные значения.")

# Функция для очистки вывода

def clear\_output():

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

# Функция для обновления графика при выборе функции

def update\_plot(event=None):

selected\_function = combobox\_functions.get()

plot\_surface(ax, selected\_function)

canvas.draw()

# Создание вкладок в меню

tab\_control = Notebook(menu\_frame)

# Вкладка для градиентного спуска

tab1 = tk.Frame(tab\_control, bg="#008080")

tab\_control.add(tab1, text="1")

label\_title = tk.Label(tab1, text="Градиентный спуск", font=("Arial", 18, "bold"), fg="#FFFFFF", bg="#008080")

label\_title.pack(pady=10)

label\_function = tk.Label(tab1, text="Выберите функцию:", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_function.pack(pady=5)

combobox\_functions = Combobox(tab1, values=["Функция Химмельблау", "Функция Сферы", "Функция Бута"])

combobox\_functions.set("Функция Химмельблау")

combobox\_functions.pack(pady=5)

combobox\_functions.bind("<<ComboboxSelected>>", update\_plot)

label\_x = tk.Label(tab1, text="Начальное значение x:", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_x.pack(pady=5)

entry\_x = tk.Entry(tab1, bg="#E0FFFF")

entry\_x.insert(0, "0")

entry\_x.pack(pady=5)

label\_y = tk.Label(tab1, text="Начальное значение y:", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_y.pack(pady=5)

entry\_y = tk.Entry(tab1, bg="#E0FFFF")

entry\_y.insert(0, "0")

entry\_y.pack(pady=5)

label\_step = tk.Label(tab1, text="Шаг (α):", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_step.pack(pady=5)

entry\_step = tk.Entry(tab1, bg="#E0FFFF")

entry\_step.insert(0, "0.01")

entry\_step.pack(pady=5)

label\_iterations = tk.Label(tab1, text="Макс. кол-во итераций:", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_iterations.pack(pady=5)

entry\_iterations = tk.Entry(tab1, bg="#E0FFFF")

entry\_iterations.insert(0, "1000")

entry\_iterations.pack(pady=5)

label\_delay = tk.Label(tab1, text="Задержка между итерациями (сек):", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_delay.pack(pady=5)

entry\_delay = tk.Entry(tab1, bg="#E0FFFF")

entry\_delay.insert(0, "0.001")

entry\_delay.pack(pady=5)

button\_frame = tk.Frame(tab1, bg="#008080")

button\_frame.pack(side=tk.TOP, padx=10)

# Кнопки для запуска и очистки

run\_button = tk.Button(button\_frame, text="Запуск", font=("Arial", 12, "bold"), fg="#008080", width=15, height=1, command=run\_gradient\_descent)

run\_button.pack(side=tk.LEFT, padx=10)

clear\_button = tk.Button(button\_frame, text="Очистить вывод", font=("Arial", 12, "bold"), fg="#008080", width=15, height=1, command=clear\_output)

clear\_button.pack(side=tk.RIGHT,pady=10)

# Вывод результатов

label\_delay = tk.Label(tab1, text="Результаты:", bg="#008080", font=("Arial", 16), fg="#FFFFFF")

label\_delay.pack(pady=1)

output\_text = scrolledtext.ScrolledText(tab1, width=35, bg="#E0FFFF")

output\_text.pack(fill=tk.Y, pady=5)

# Отображение вкладок

tab\_control.pack(expand=1, fill="both")

# Отображаем график по умолчанию

plot\_surface(ax, "Функция Химмельблау")

canvas.draw()

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()