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 46 группы

Стасюк К.В.

Баева Д.Н.

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.

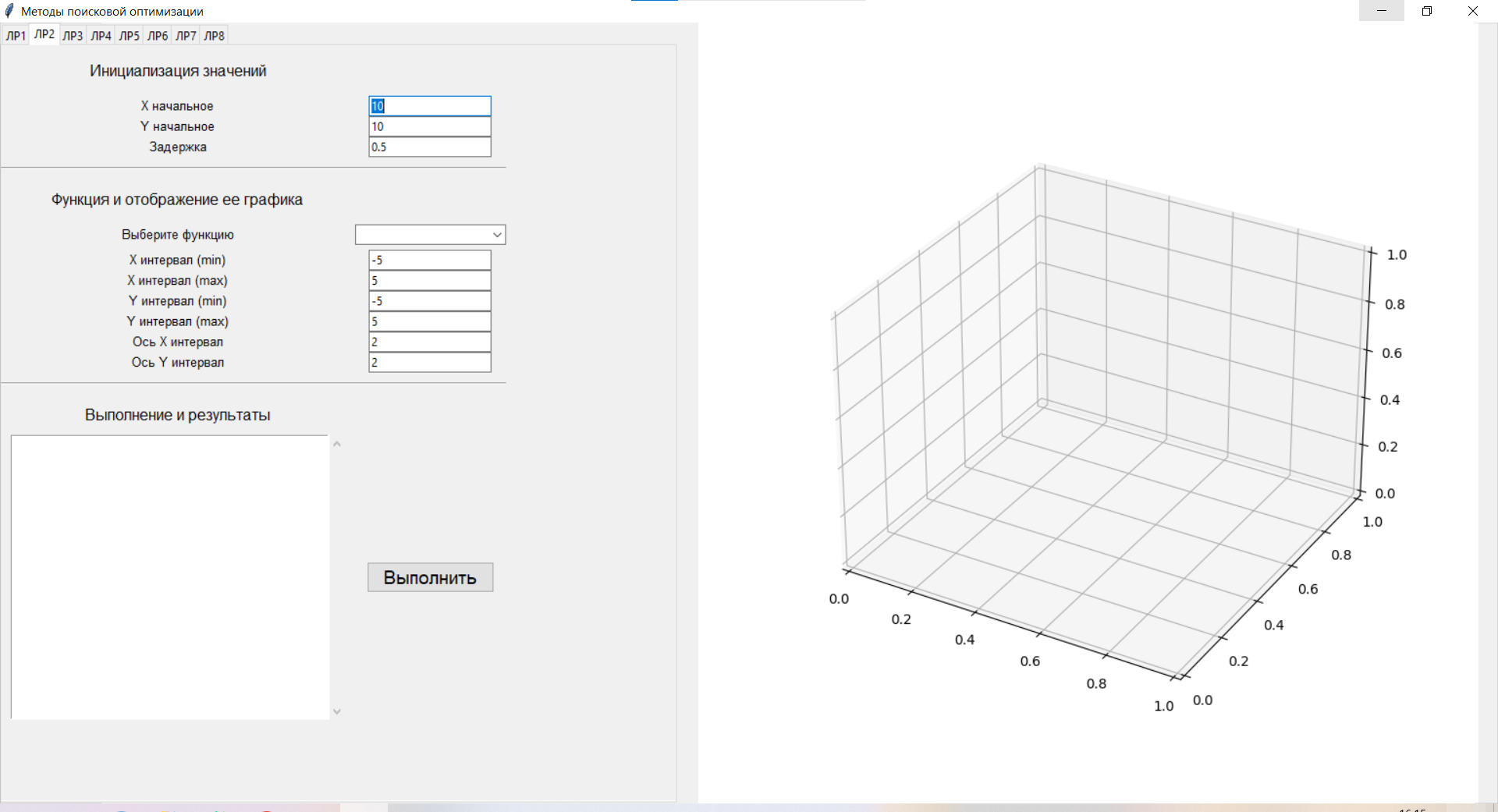
*Шаг 5.* **Проверка условий оптимальности.** Проверить, что полученное решение удовлетворяет ограничениям: . Затем проверить условие ККТ (условия Каруша-Куна-Таккера), включая неотрицательность множителей Лагранжа: .

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

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

Для создания программы использовался язык программирования Python 3.10 и среда разработки PyCharm Professional 2023.2. Для графической визуализации были подключены графические фреймворки Tkinter и Matplotlib.

Интерфейс программы имеет вид:

Рисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую вкладку соответствующей лабораторной работы, в данном случае «ЛР2», ввести начальную точку для функции с координатами «X» и «Y» и задержку между 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.

Изображение выглядит как текст, снимок экрана, Прямоугольник

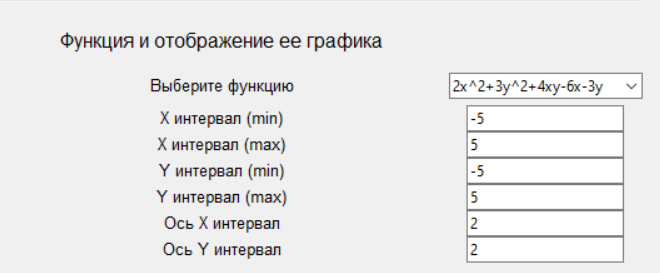
Автоматически созданное описание

Рисунок 2 – Выполнение и результаты.

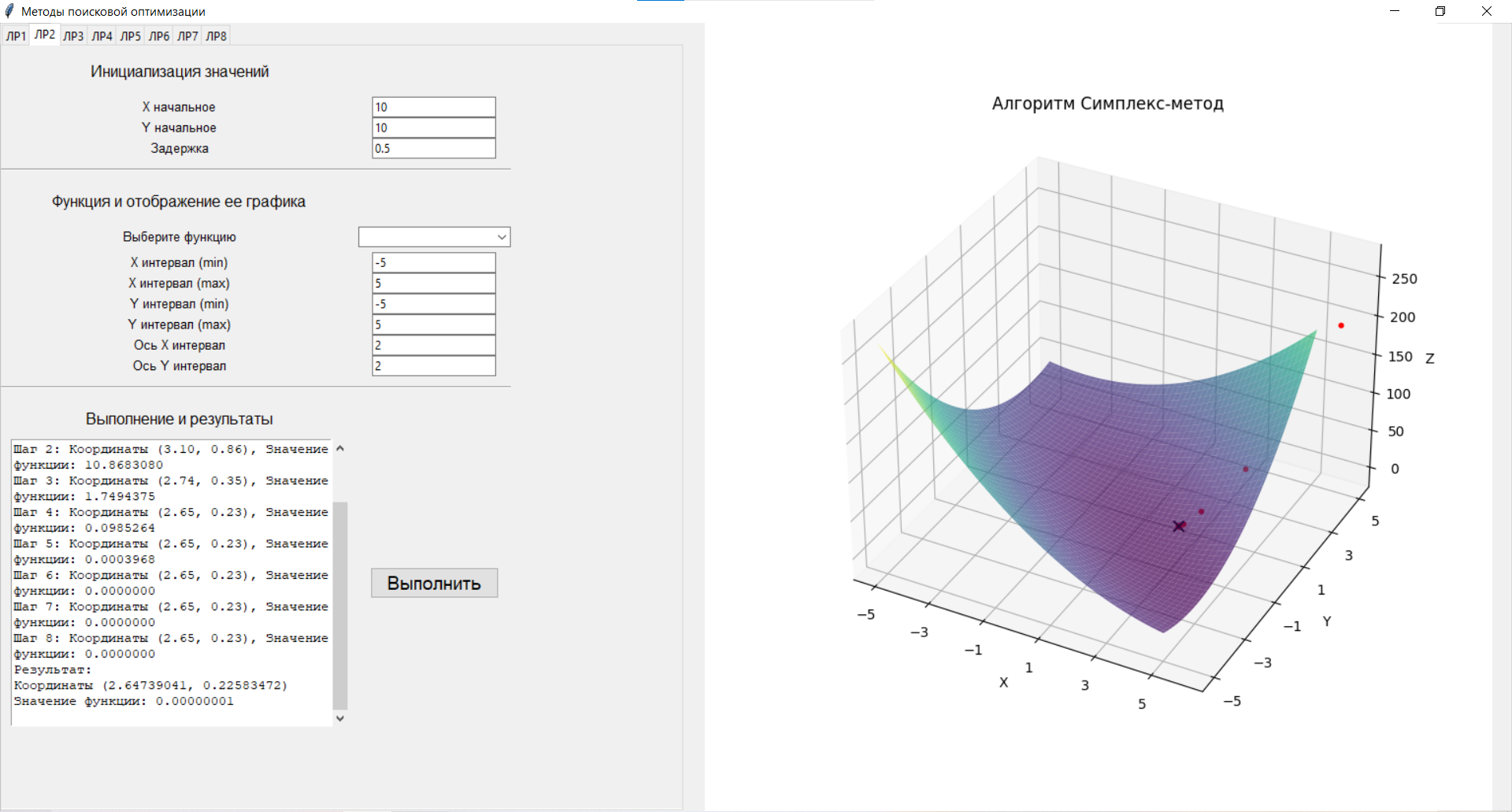
Также, в интерфейсе программы есть панель управления «Функция и отображения её графика», в котором можно выбрать необходимую функцию и задать её параметры:

1. X интервал (интервал на котором строится и минимизируется функция);
2. Y интервал (интервал на котором строится и минимизируется функция);
3. Ось X интервал (размер отображаемой сетки по оси X);
4. Ось Y интервал (размер отображаемой сетки по оси Y).

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

Рисунок 3 – Функция и отображение её графика.

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Выполнение и результаты» начнёт динамически заполняться точками, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки, которые также отображаются динамически как показано на рисунке 4. Результирующая точка выделена на графике крестом. Значения результирующей функции имеют разный окрас в зависимости от положения: самые высокие точки показаны ярко-жёлтым цветом, а самые низкие – тёмно-синим.

Рисунок 4 – Результат работы программы.

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

**Листинг программы**

Файл LR2.py

import tkinter as tk

from tkinter import ttk

import numpy as np

import time

from tkinter import scrolledtext

from scipy.optimize import minimize

def Simplex\_method(frame,root,ax,canvas):

def target\_function(x, y):

return (2\*x \*\* 2 + 3\*y\*\*2 + 4\*x\*y - 6\*x - 3\*y)

def simplex\_method(x, y):

# global points

points = []

def fun(x\_i): # Функция

x1 = x\_i[0]

x2 = x\_i[1]

return 2 \* x1 \* x1 + 3 \* x2 \* x2 + 4 \* x1 \* x2 - 6 \* x1 - 3 \* x2

def callback(x\_w):

g\_list = np.ndarray.tolist(x\_w)

g\_list.append(fun(x\_w))

points.append(g\_list)

b = (0, float("inf")) # диапазон поиска

bounds = (b, b)

x0 = (x, y) # начальная точка

con = {'type': 'eq', 'fun': fun}

# основной вызов

res = minimize(fun, x0, method="SLSQP", bounds=bounds,

constraints=con, callback=callback)

glist = np.ndarray.tolist(res.x)

glist.append(res.fun)

points.append(glist)

for iteration, point in enumerate(points):

yield iteration, point

def run\_optimization():

res\_x = x\_var.get()

res\_y = y\_var.get()

delay = delay\_var.get()

x\_cs = []

y\_cs = []

z\_cs = []

# Очистка текущего графика

ax.cla()

# Генерация сетки для графика целевой функции

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)

Z = target\_function(X, Y)

# Построение поверхности графика целевой функции

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

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()))

ax.set\_title("Алгоритм Симплекс-метод")

# Инициализация списка для хранения результатов оптимизации

results = []

# Вывод результатов в текстовое поле

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

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

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

x\_cs.append(point[0])

y\_cs.append(point[1])

z\_cs.append(point[2])

print(point)

# Сохранение результатов и обновление графика

results.append((point[0], point[1], i, point[2]))

ax.scatter(point[0], point[1], point[2], color='red', s=10)

results\_text.insert(tk.END,

f"Шаг {i}: Координаты ({point[0]:.2f}, {point[1]:.2f}), Значение функции: {point[2]:.7f}\n")

results\_text.yview\_moveto(1)

canvas.draw()

root.update()

time.sleep(delay)

# Вывод окончательного результата

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

results\_text.yview\_moveto(1)

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

param\_frame2 = frame

# Параметры задачи

ttk.Label(param\_frame2, text="Инициализация значений", font=("Helvetica", 12)).grid(row=0, column=0, pady=15)

ttk.Label(param\_frame2, text="Задержка", font=("Helvetica", 10)).grid(row=3, column=0)

ttk.Label(param\_frame2, text="X начальное", font=("Helvetica", 10)).grid(row=1, column=0)

ttk.Label(param\_frame2, text="Y начальное", font=("Helvetica", 10)).grid(row=2, column=0)

x\_var = tk.DoubleVar(value=10)

y\_var = tk.DoubleVar(value=10)

delay\_var = tk.DoubleVar(value=0.5)

x\_entry = ttk.Entry(param\_frame2, textvariable=x\_var)

y\_entry = ttk.Entry(param\_frame2, textvariable=y\_var)

delay\_entry = ttk.Entry(param\_frame2, textvariable=delay\_var)

x\_entry.grid(row=1, column=1)

y\_entry.grid(row=2, column=1)

delay\_entry.grid(row=3, column=1)

separator = ttk.Separator(param\_frame2, orient="horizontal") # Горизонтальная полоса разделения

separator.grid(row=7, column=0, columnspan=2, sticky="ew", pady=10)

# Параметры функции

ttk.Label(param\_frame2, text="Функция и отображение ее графика", font=("Helvetica", 12)).grid(row=9, column=0, pady=10)

ttk.Label(param\_frame2, text="Выберите функцию", font=("Helvetica", 10)).grid(row=10, column=0)

function\_choices = ["2x^2+3y^2+4xy-6x-3y"]

function\_var = tk.StringVar(value=function\_choices[0])

function\_menu = ttk.Combobox(param\_frame2, textvariable=function\_var, values=function\_choices, width=22)

function\_menu.grid(row=10, column=1, pady=5)

ttk.Label(param\_frame2, text="X интервал (min)", font=("Helvetica", 10)).grid(row=11, column=0)

ttk.Label(param\_frame2, text="X интервал (max)", font=("Helvetica", 10)).grid(row=12, column=0)

ttk.Label(param\_frame2, text="Y интервал (min)", font=("Helvetica", 10)).grid(row=13, column=0)

ttk.Label(param\_frame2, text="Y интервал (max)", font=("Helvetica", 10)).grid(row=14, column=0)

ttk.Label(param\_frame2, text="Ось X интервал", font=("Helvetica", 10)).grid(row=16, column=0)

ttk.Label(param\_frame2, text="Ось Y интервал", font=("Helvetica", 10)).grid(row=17, column=0)

separator = ttk.Separator(param\_frame2, orient="horizontal") # Горизонтальная полоса разделения

separator.grid(row=18, column=0, columnspan=2, sticky="ew", pady=10)

x\_interval\_min = tk.DoubleVar(value=-5)

x\_interval\_max = tk.DoubleVar(value=5)

y\_interval\_min = tk.DoubleVar(value=-5)

y\_interval\_max = tk.DoubleVar(value=5)

x\_axis\_interval = tk.IntVar(value=2)

y\_axis\_interval = tk.IntVar(value=2)

x\_interval\_min\_entry = ttk.Entry(param\_frame2, textvariable=x\_interval\_min)

x\_interval\_max\_entry = ttk.Entry(param\_frame2, textvariable=x\_interval\_max)

y\_interval\_min\_entry = ttk.Entry(param\_frame2, textvariable=y\_interval\_min)

y\_interval\_max\_entry = ttk.Entry(param\_frame2, textvariable=y\_interval\_max)

x\_axis\_interval\_entry = ttk.Entry(param\_frame2, textvariable=x\_axis\_interval)

y\_axis\_interval\_entry = ttk.Entry(param\_frame2, textvariable=y\_axis\_interval)

x\_interval\_min\_entry.grid(row=11, column=1)

x\_interval\_max\_entry.grid(row=12, column=1)

y\_interval\_min\_entry.grid(row=13, column=1)

y\_interval\_max\_entry.grid(row=14, column=1)

x\_axis\_interval\_entry.grid(row=16, column=1)

y\_axis\_interval\_entry.grid(row=17, column=1)

# Создание кнопки Выполнить

button\_style = ttk.Style()

button\_style.configure("My.TButton", font=("Helvetica", 14))

# Создание кнопки Выполнить

apply\_settings\_button = ttk.Button(param\_frame2, text="Выполнить",command=run\_optimization, style="My.TButton")

apply\_settings\_button.grid(row=21, column=1, padx=10, pady=10)

ttk.Label(param\_frame2, text="Выполнение и результаты", font=("Helvetica", 12)).grid(row=20, column=0, pady=10)

results\_text = scrolledtext.ScrolledText(param\_frame2, wrap=tk.WORD, height=18, width=40, padx=2, state=tk.DISABLED)

results\_text.grid(row=21, column=0, padx=10)