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 №5 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студенты 46 группы

Нагалевский А.М.

Прозоров М.С.

Преподаватель:

Нигодин Е.А.

Краснодар 2023

**Цель работы:** разработать алгоритм оптимизации пчелиным роем на функциях Химмельблау и Розенброка.

**Ход работы:**

Алгоритм пчелиного роя (Artificial Bee Colony, ABC) — это оптимизационный алгоритм, инспирированный поведением пчелиных роев при сборе пищи. Алгоритм был предложен Дереком Каравайем в 2005 году. Он включает в себя понятия разведчиков (scouts), элитных пчёл (elite bees) и перспективных пчёл (perspective bees), которые работают совместно для поиска оптимальных решений задачи оптимизации.

Рассмотрим подробнее аспекты данного алгоритма.

1. Вдохновение природой:

- Идея алгоритма возникла из наблюдений за поведением пчелиных роев в природе. Пчёлы коллективно и эффективно ищут оптимальные источники пищи, используя механизм коммуникации и адаптивного поиска.

1. Первоначальная формулировка:

- Дерек Каравайем предложил алгоритм пчелиного роя в 2005 году в своей работе "An Idea Based on Honey Bee Swarm for Numerical Optimization". Он адаптировал поведение пчёл для разработки эффективного метода оптимизации.

1. Основные компоненты:

- Разведчики (Scouts): Пчёлы, которые исследуют новые области пространства параметров.

- Элитные пчёлы (Elite Bees): Пчёлы, обнаружившие лучшие решения, сохраняют их и остаются в районе лучших точек.

- Перспективные пчёлы (Perspective Bees): Пчёлы, обнаружившие потенциально хорошие решения, сохраняют их и также остаются в районе лучших точек.

1. Принцип работы:

- Алгоритм имитирует процесс сбора пищи пчелиным роем. Разведчики ищут новые источники пищи, а элитные и перспективные пчёлы сосредотачиваются вокруг хороших решений.

- Разведчики и случайные пчёлы отправляются в разные области пространства параметров для исследования.

- Значения функции в каждой точке оцениваются, и лучшие точки сохраняются.

- Элитные и перспективные пчёлы направляют рабочих пчёл вокруг областей, содержащих хорошие решения, чтобы углубить исследование в этих областях.

1. Применение в оптимизации:

- Алгоритм пчелиного роя применяется для решения задач оптимизации в различных областях, таких как инженерия, финансы, машинное обучение и другие. Он проявляет хорошую производительность при решении сложных и многомерных задач оптимизации.

1. Развитие и модификации:

- С момента своего предложения алгоритм был подвергнут различным модификациям и улучшениям. Исследователи вносят изменения в параметры, стратегии выбора пчёл и другие аспекты, чтобы адаптировать его к конкретным задачам.

1. Значение и перспективы:

- Алгоритм пчелиного роя продолжает привлекать внимание исследователей в области оптимизации, и его применение охватывает все больше сфер. Его простота и способность к глобальной оптимизации делают его привлекательным инструментом в исследованиях и практике.

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.

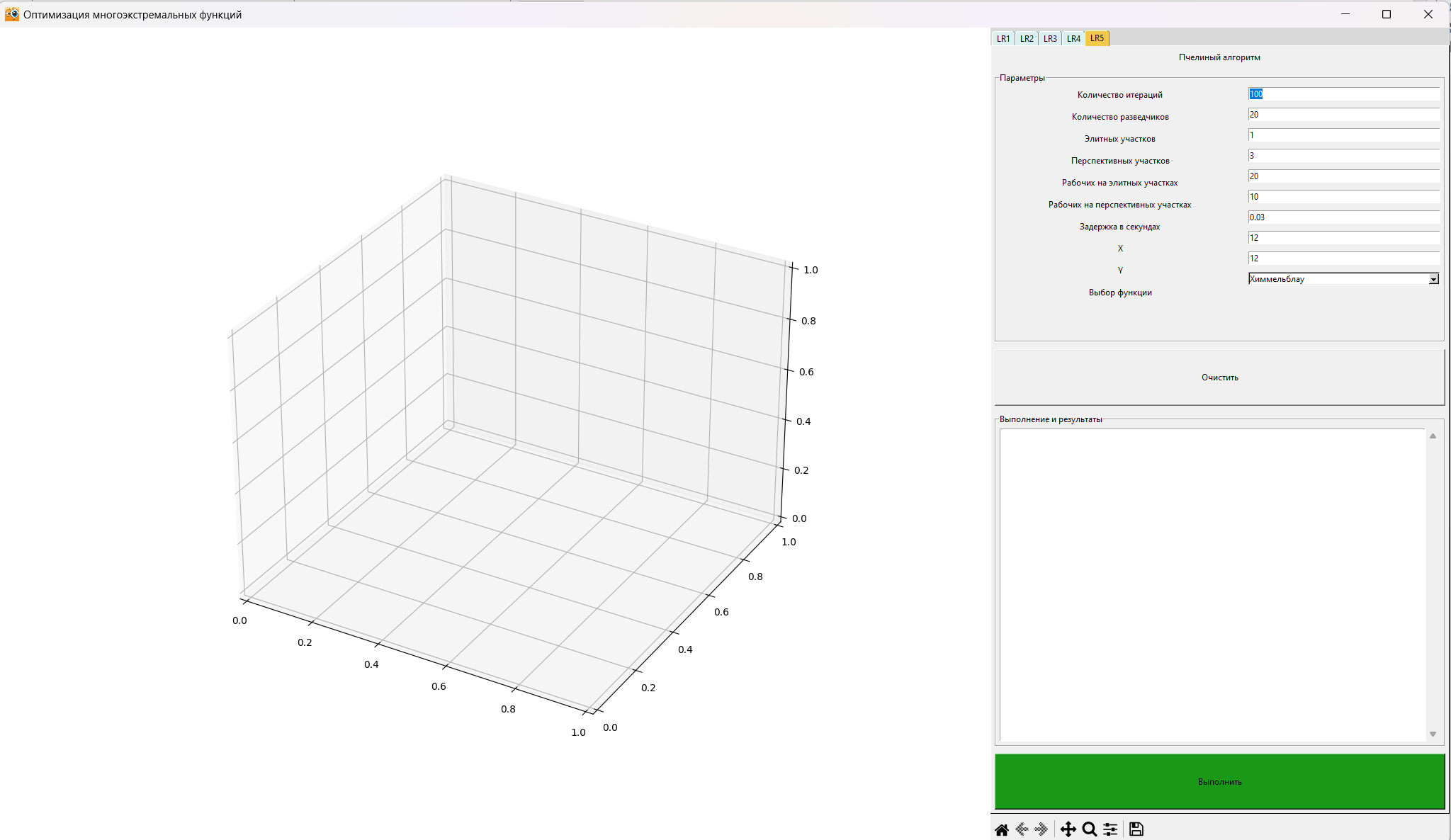


Рисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую лабораторную работу, в данном случае «5», ввести количество итераций алгоритма, количество разведчиков, элитных участков, перспективных участков, рабочих на элитных участках, рабочих на перспективных участках и задержку в секундах.

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.

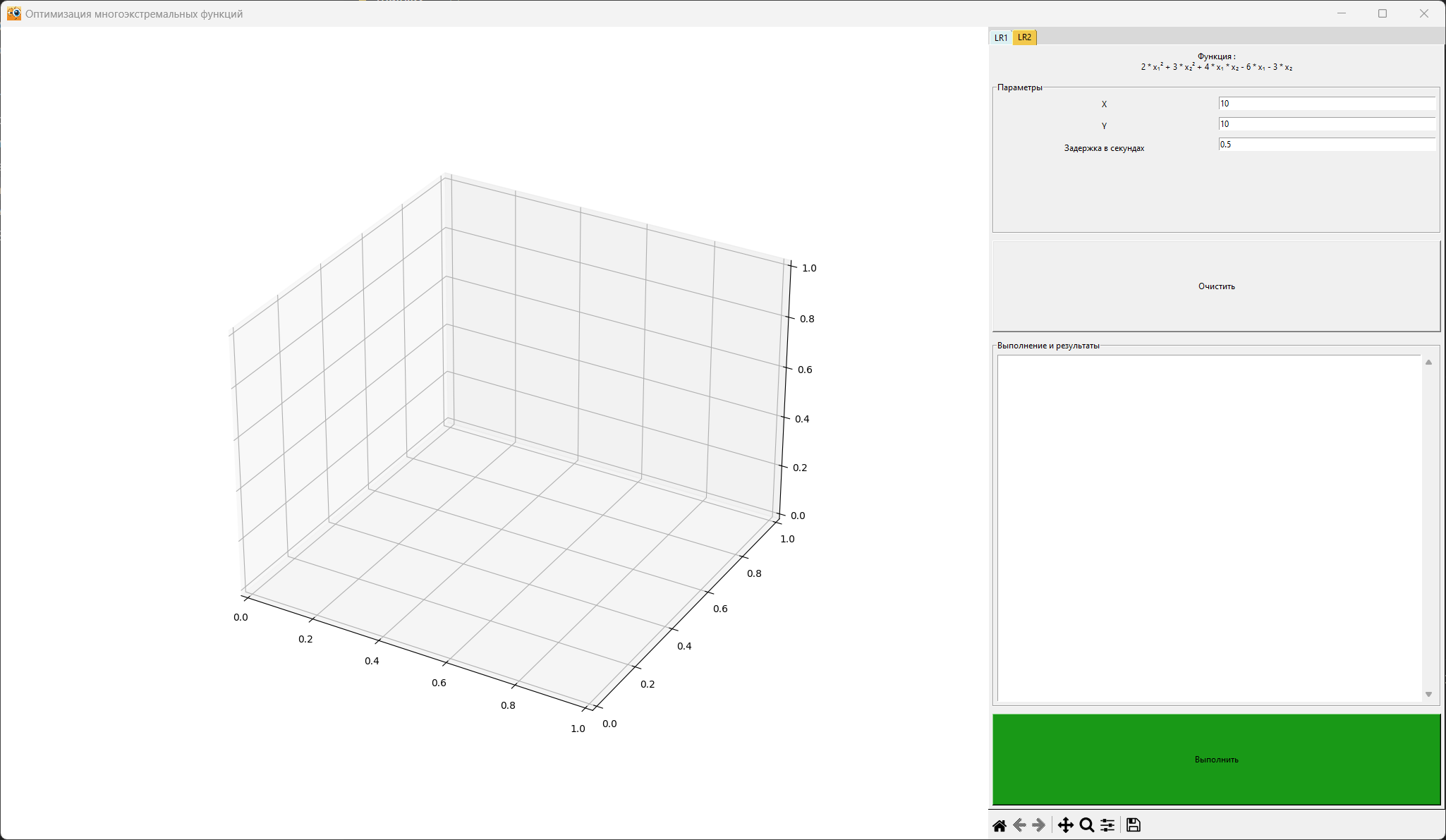


Рисунок 2 – Выполнение и результаты.

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.

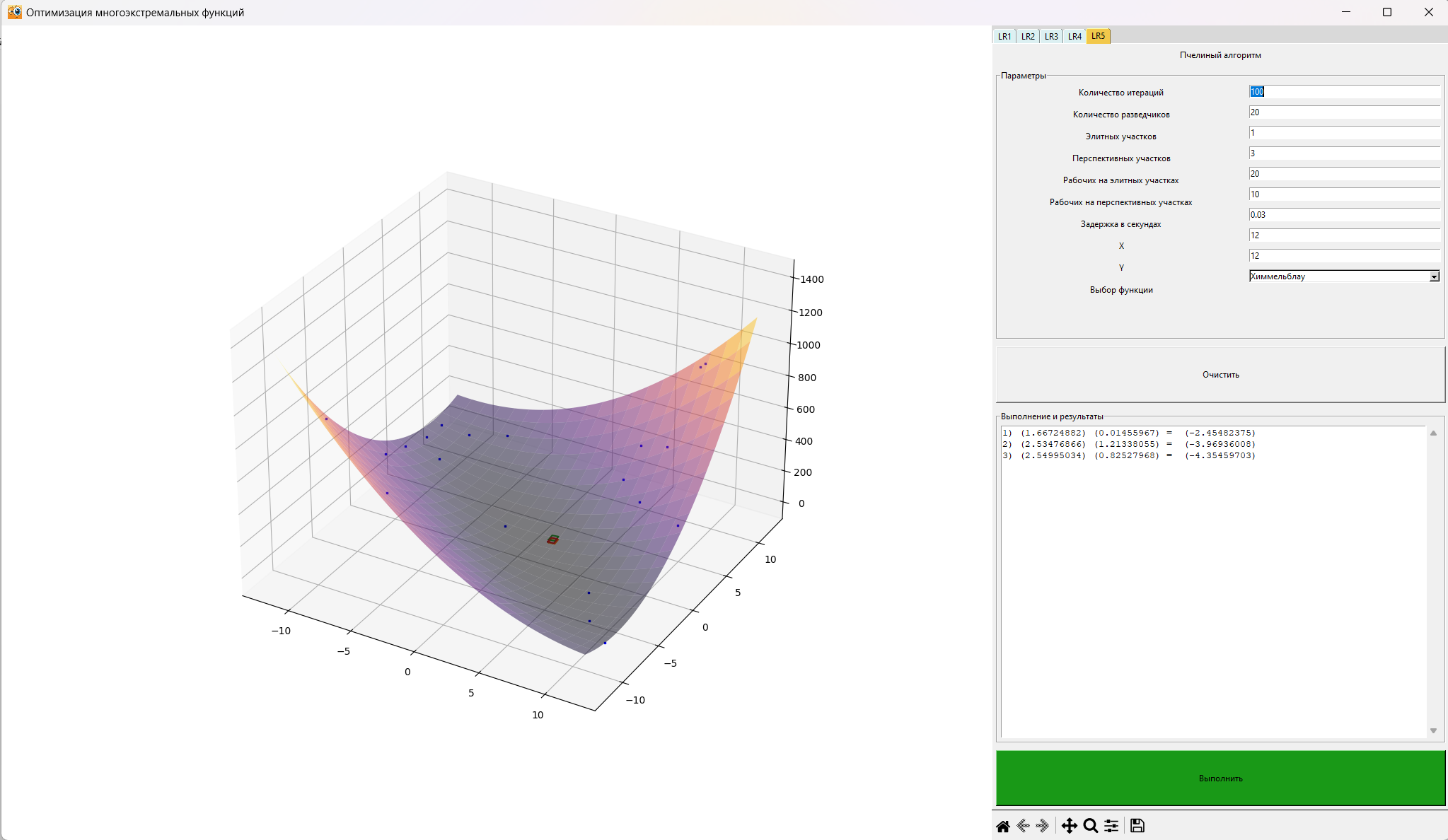


Рисунок 4 – Функция Химмельблау.

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.

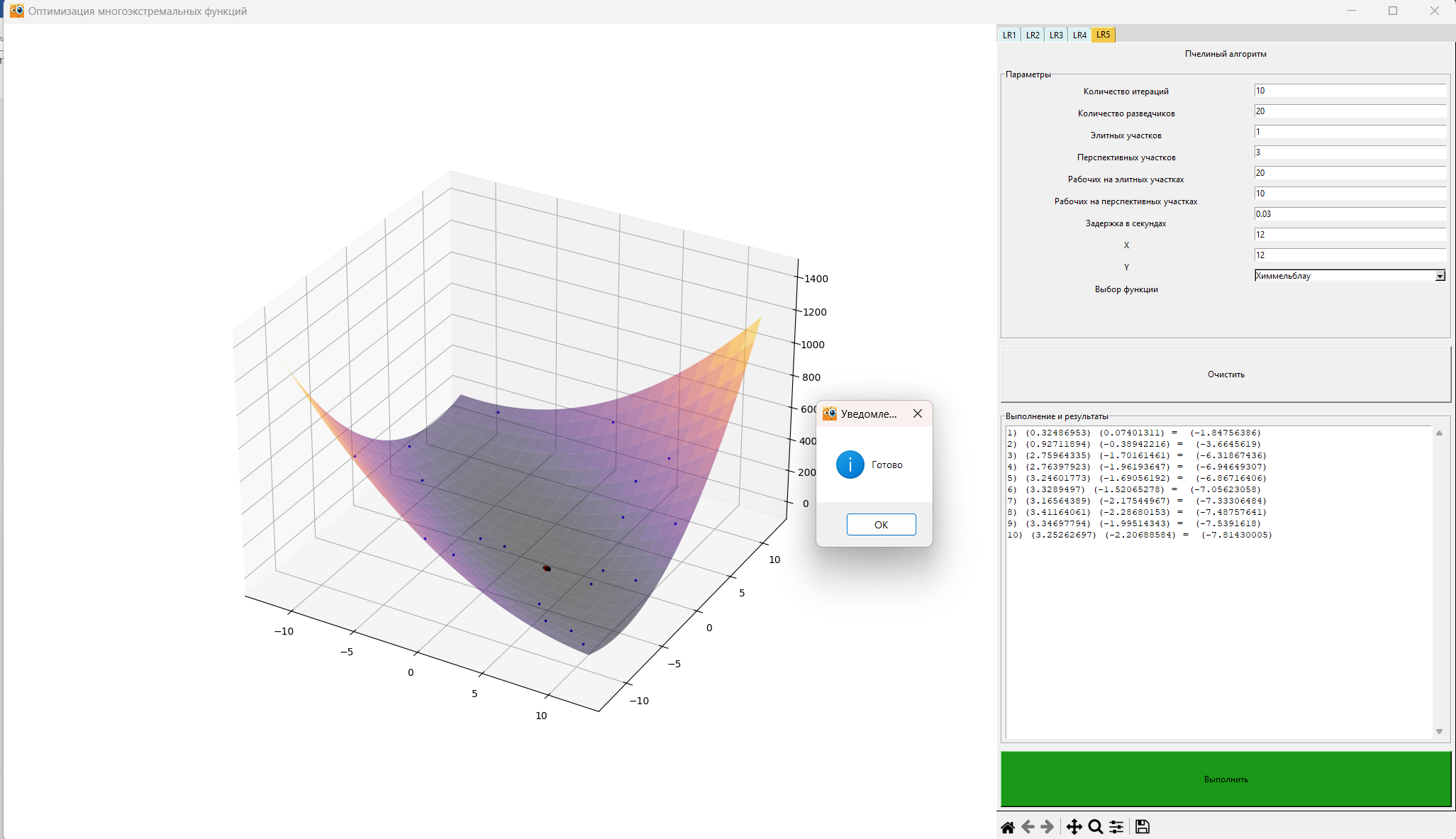


Рисунок 5 – Результат работы программы.

**Вывод:** в ходе работы был изучен и реализован алгоритм пчелиного роя на определенной функции.

**Листинг программы:**

**Файл bees.py**

import random

from operator import itemgetter

class Bees:

def \_\_init\_\_(self, func, scouts, elite, perspect, bees\_to\_leet, bees\_to\_persp, radius, position\_x, position\_y):

self.func = func

self.pos\_x = float(position\_x)

self.pos\_y = float(position\_y)

self.scouts = [[random.uniform(-self.pos\_x, self.pos\_x), random.uniform(-self.pos\_y, self.pos\_y), float(0.0)] for \_ in

range(scouts)]

for i in self.scouts:

i[2] = self.func(i[0], i[1])

self.n\_workers = elite \* bees\_to\_leet + perspect \* bees\_to\_persp

self.e = elite

self.p = perspect

self.b\_leet = bees\_to\_leet

self.b\_persp = bees\_to\_persp

max\_b = max(self.scouts, key=itemgetter(2))

self.workers = [[self.pos\_x, self.pos\_y, max\_b[2]] for \_ in range(self.n\_workers)]

self.bees = list()

self.selected = list()

self.rad = radius

def send\_scouts(self):

for unit in self.scouts:

unit[0] = random.uniform(-self.pos\_x, self.pos\_x)

unit[1] = random.uniform(-self.pos\_y, self.pos\_y)

unit[2] = self.func(unit[0], unit[1])

def research\_reports(self):

self.bees = self.scouts + self.workers

self.bees = sorted(self.bees, key=itemgetter(2), reverse=False)

self.selected = self.bees[:self.e + self.p]

def get\_best(self):

return self.bees[0]

def send\_workers(self, bee\_part, sector, radius):

for bee in bee\_part:

bee[0] = random.uniform(sector[0] - radius, sector[0] + radius)

bee[1] = random.uniform(sector[1] - radius, sector[1] + radius)

bee[2] = random.uniform(sector[2] - radius, sector[2] + radius)

def selected\_search(self, param):

for i in range(self.e):

Bees.send\_workers(self.func,

self.workers[i \* self.b\_leet:i \* self.b\_leet + self.b\_leet],

self.selected[i],

self.rad \* param)

for i in range(self.p):

Bees.send\_workers(self.func,

self.workers[self.e \* self.b\_leet + i \* self.b\_persp:self.e \* self.b\_leet +

i \* self.b\_persp + self.b\_persp],

self.selected[self.e + i],

self.rad \* param)

**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 Rosenbrock\_function import make\_data\_lab\_3

from pso import PSO

from bees import Bees

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)

def draw\_lab\_5():

fig.clf()

iter\_number = int(txt\_1\_tab\_5.get())

scouts\_number = int(txt\_2\_tab\_5.get())

elite = int(txt\_3\_tab\_5.get())

perspective = int(txt\_4\_tab\_5.get())

b\_to\_leet = int(txt\_5\_tab\_5.get())

b\_to\_persp = int(txt\_6\_tab\_5.get())

pos\_x = int(txt\_8\_tab\_5.get())

pos\_y = int(txt\_9\_tab\_5.get())

delay = txt\_7\_tab\_5.get()

if combo\_tab\_5.get() == "Химмельблау":

func = himmelblau\_2

x, y, z = make\_data\_himmelblau(pos\_x, pos\_y)

else:

func = rosenbrock\_2

x, y, z = make\_data\_rosenbrock(pos\_x, pos\_y)

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

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

canvas.draw()

bees\_swarm = Bees(func, scouts\_number, elite, perspective, b\_to\_leet, b\_to\_persp, 1, pos\_x, pos\_y)

for scout in bees\_swarm.scouts:

ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1, marker="s")

bees\_swarm.research\_reports()

bees\_swarm.selected\_search(1)

for worker in bees\_swarm.workers:

ax.scatter(worker[0], worker[1], worker[2], c="black", s=1, marker="s")

b = bees\_swarm.get\_best()

ax.scatter(b[0], b[1], b[2], c="red")

canvas.draw()

window.update()

fig.clf()

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

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

canvas.draw()

for i in range(iter\_number):

bees\_swarm.send\_scouts()

for scout in bees\_swarm.scouts:

ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1, marker="s")

bees\_swarm.research\_reports()

bees\_swarm.selected\_search(1 / (i + 1))

for sec in bees\_swarm.selected:

rx, ry, rz = make\_square(sec[0], sec[1], 1 / (i + 1), func)

ax.plot(rx, ry, rz, label='parametric curve')

canvas.draw()

window.update()

for worker in bees\_swarm.workers:

ax.scatter(worker[0], worker[1], worker[2], c="black", s=1, marker="s")

b = bees\_swarm.get\_best()

ax.scatter(b[0], b[1], b[2], c="red")

txt\_tab\_5.insert(INSERT,

f"{i + 1}) ({round(b[0], 8)})"

f" ({round(b[1], 8)}) = "

f" ({round(b[2], 8)})\n")

canvas.draw()

window.update()

time.sleep(float(delay))

fig.clf()

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

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

canvas.draw()

for scout in bees\_swarm.scouts:

ax.scatter(scout[0], scout[1], scout[2], c="blue", s=1, marker="s")

for worker in bees\_swarm.workers:

ax.scatter(worker[0], worker[1], worker[2], c="black", s=1, marker="s")

b = bees\_swarm.get\_best()

ax.scatter(b[0], b[1], b[2], c="red")

canvas.draw()

window.update()

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

def make\_square(x, y, rad, func):

r\_1 = [x - rad, x - rad, x + rad, x + rad] # x

r\_2 = [y - rad, y + rad, y + rad, y - rad] # y

r\_3 = [func(r\_1[0], r\_2[0]), func(r\_1[1], r\_2[1]), func(r\_1[2], r\_2[2]), func(r\_1[3], r\_2[3])] # z

r\_1.append(r\_1[0])

r\_2.append(r\_2[0])

r\_3.append(r\_3[0])

return r\_1, r\_2, r\_3

def delete\_lab\_5():

txt\_tab\_5.delete(1.0, END)

tab\_5 = Frame(tab\_control)

tab\_control.add(tab\_5, text="LR5")

main\_f\_tab\_5 = LabelFrame(tab\_5, text="Параметры")

left\_f\_tab\_5 = Frame(main\_f\_tab\_5)

right\_f\_tab\_5 = Frame(main\_f\_tab\_5)

txt\_f\_tab\_5 = LabelFrame(tab\_5, text="Выполнение и результаты")

lbl\_5\_tab\_5 = Label(tab\_5, text="Пчелиный алгоритм")

lbl\_1\_tab\_5 = Label(left\_f\_tab\_5, text="Количество итераций")

lbl\_2\_tab\_5 = Label(left\_f\_tab\_5, text="Количество разведчиков")

lbl\_3\_tab\_5 = Label(left\_f\_tab\_5, text="Элитных участков")

lbl\_4\_tab\_5 = Label(left\_f\_tab\_5, text="Задержка в секундах")

lbl\_6\_tab\_5 = Label(left\_f\_tab\_5, text="Перспективных участков")

lbl\_7\_tab\_5 = Label(left\_f\_tab\_5, text="Выбор функции")

lbl\_8\_tab\_5 = Label(left\_f\_tab\_5, text="Рабочих на элитных участках")

lbl\_9\_tab\_5 = Label(left\_f\_tab\_5, text="Рабочих на перспективных участках")

lbl\_10\_tab\_5 = Label(left\_f\_tab\_5, text="X")

lbl\_11\_tab\_5 = Label(left\_f\_tab\_5, text="Y")

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

txt\_1\_tab\_5.insert(0, "100")

txt\_2\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_2\_tab\_5.insert(0,"20")

txt\_3\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_3\_tab\_5.insert(0,"1")

txt\_4\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_4\_tab\_5.insert(0,"3")

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

txt\_5\_tab\_5.insert(0,"20")

txt\_6\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_6\_tab\_5.insert(0,"10")

txt\_7\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_7\_tab\_5.insert(0,"0.03")

txt\_8\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_8\_tab\_5.insert(0, "12")

txt\_9\_tab\_5 = Entry(right\_f\_tab\_5)

txt\_9\_tab\_5.insert(0, "12")

combo\_tab\_5 = Combobox(right\_f\_tab\_5)

combo\_tab\_5['values'] = ("Химмельблау", "Розенброка")

combo\_tab\_5.set("Химмельблау")

txt\_tab\_5 = scrolledtext.ScrolledText(txt\_f\_tab\_5)

btn\_del\_tab\_5 = Button(tab\_5, text="Очистить", command=delete\_lab\_5)

btn\_tab\_5 = Button(tab\_5, text="Выполнить", foreground="black", background="#199917", command=draw\_lab\_5)

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

main\_f\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_5.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_5.pack(side=RIGHT, fill=BOTH, expand=True)

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

lbl\_2\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_3\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_6\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_8\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_9\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_4\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_10\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_11\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_7\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

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

txt\_2\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_3\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_4\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

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

txt\_6\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_7\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_8\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_9\_tab\_5.pack(side=TOP, padx=5, pady=5, fill=BOTH)

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

txt\_tab\_5.pack(padx=5, pady=5, fill=BOTH, expand=True)

btn\_tab\_5.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_5.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_5.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()