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 36 группы

Агаджанян А.С.

Гаранина Л.В.

Преподаватель:

Нигодин Е.А.

Краснодар 2025

**Цель работы:** разработать B-алгоритм минимизации функций Розенброка, Химмельблау, Растригина.

**Ход работы:**

Пчелиным алгоритмом называют (В-алгоритм) *Bees algorithm*, предложенный в работах Фама (D. Т. Pham) с соавторами в 2005 г. Концептуально, схема В-алгоритма имеет следующий вид. Сначала из улья вылетает в случайном направлении некоторое число пчел-разведчиков, которые пытаются отыскать участки, где есть нектар. Через какое-то время пчелы возвращаются в улей и сообщают другим пчелам, где и сколько они нашли нектара. После этого на найденные участки отправляются рекрутированные (рабочие) пчелы, причем чем больше на данном участке предполагается найти нектара, тем больше пчел летит к этому участку. Пчелы-разведчики опять улетают искать другие участки, после чего процесс повторяется.

Пусть дана задача минимизации фитнес-функции. Количество нектара в некоторой точке пространства поиска полагаем пропорциональным значению фитнес-функции в этой точке. Одна итерация алгоритма включает в себя следующие основные шаги.

1. В случайные точки пространства поиска отправляем пчел-разведчиков.

2. На основании значений фитнес-функции, вычисленных в указанных точках, выделяем некоторое число элитных (лучших) участков - подобластей пространства поиска, соответствующих минимальным значениям этой функции. Аналогично определяем некоторое число перспективных участков, которые соответствуют значениям фитнес-функции, близким к минимальным.

3. На каждый из элитных и перспективных участков посылаем определенное число рабочих пчел. Используя соответствующие значения фитнес-функции, находим новые элитные и перспективные участки. При этом выборе учитываем результаты, полученные как пчелами-разведчиками, так и рабочими пчелами.

В качестве текущего приближения к решению задачи принимаем точку с минимальным значением фитнес-функции. Размеры элитных и перспективных участков уменьшаем с ростом числа итераций, так что на завершающих итерациях поиск ведется только в окрестностях минимумов целевой функции.

**Особенности реализации генетического алгоритма.**

Для создания программы использовался язык программирования Python. Для графической визуализации были подключены графические фреймворки Tkinter и Matplotlib.

Интерфейс программы имеет вид:

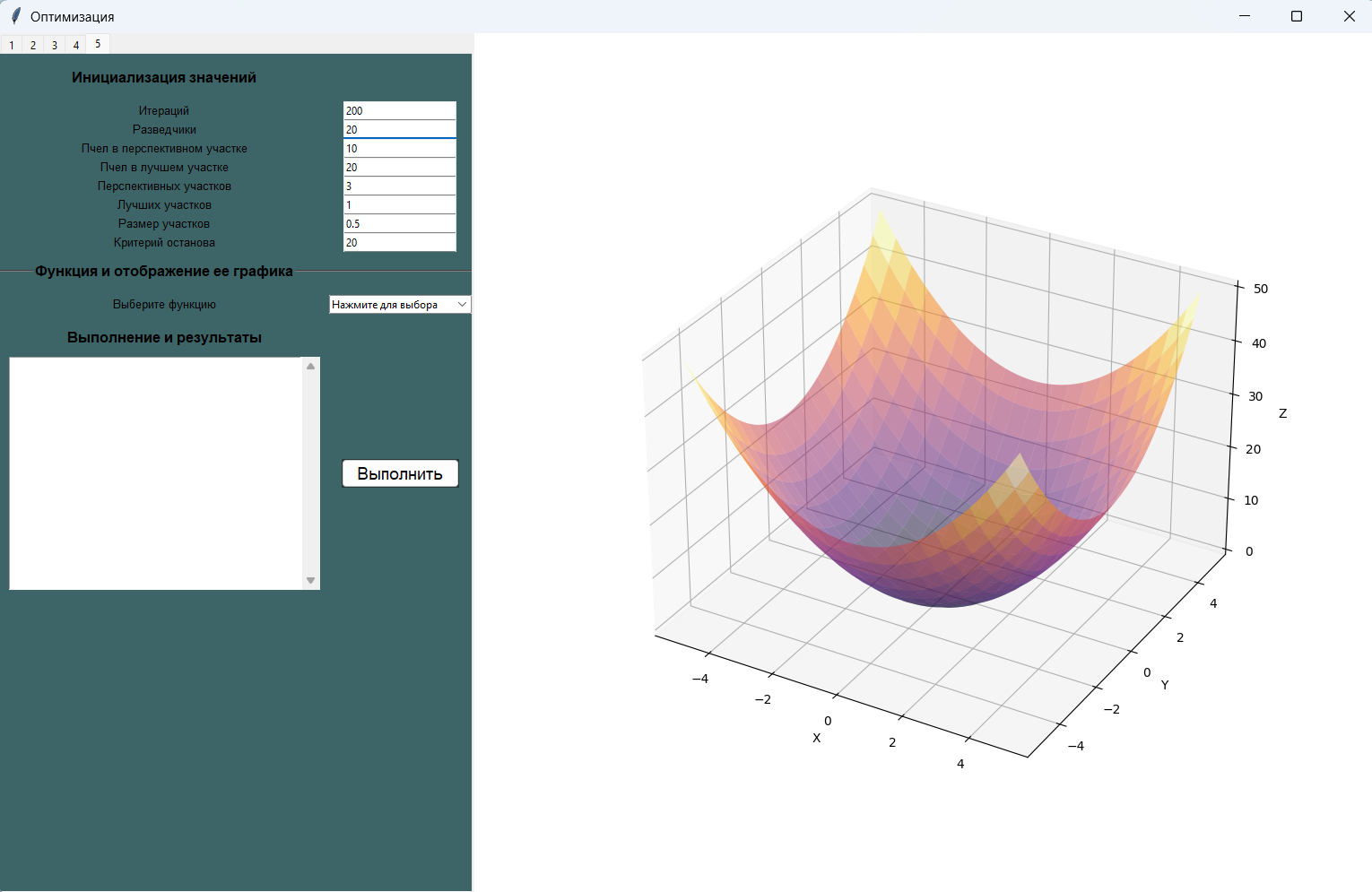


Рисунок 1 – Главное окно программы

В данном окне можно выбрать необходимую вкладку соответствующей лабораторной работы, в данном случае «5», ввести количество итераций, количество пчёл-разведчиков, пчёл в перспективном участке, пчёл в лучшем участке, количество перспективных участков, количество лучших участков, размер участков и критерий останова. 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, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Выполнение и результаты

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

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

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.

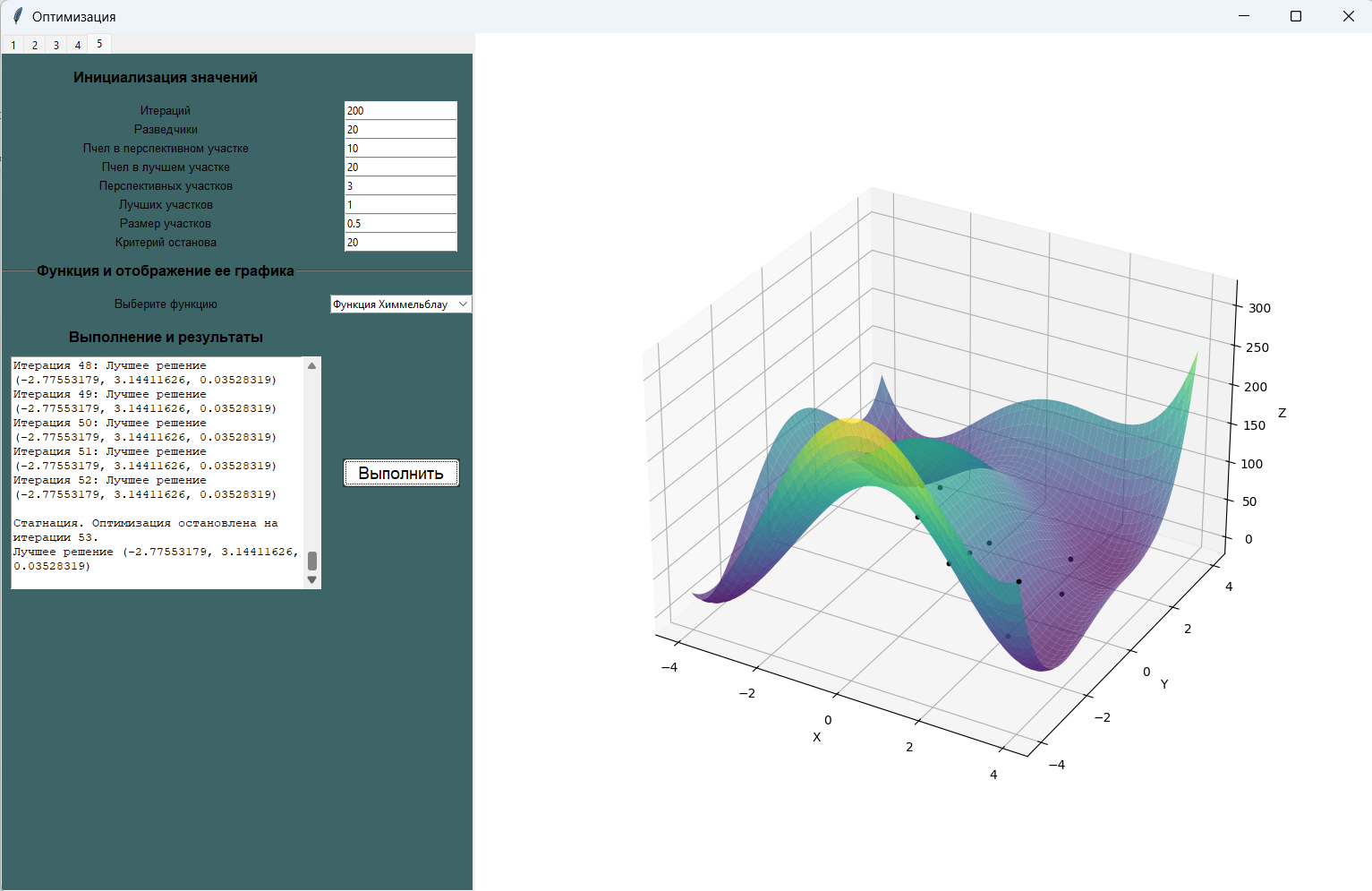


Рисунок 4 – Результат работы программы

**Вывод:** в ходе работы были изучен пчелиный алгоритм (B-алгоритм), а затем разработан B-алгоритм минимизации функций Розенброка, Химмельблау, Растригина.

**Листинг программы**

Файл bee.py

import tkinter as tk

from tkinter import ttk

from tkinter import scrolledtext

import numpy as np

import random

# Класс для представления пчелы

class Bee:

    """

     coords - координаты поля для пчелы

     fitness - значение фитнесс-функции для центра данного участка

    """

    def \_\_init\_\_(self, coords, fitness):

        self.coords = coords

        self.fitness = fitness

class BeeAlgorithm:

    def \_\_init\_\_(self, num\_scouts, elite\_radius, perspective\_radius, num\_elite, num\_perspective, agents\_per\_perspective,

                 agents\_per\_elite, bounds, max\_epochs, stagnation\_limit, fitness\_function):

        '''

        - num\_scouts (int): Количество пчел-разведчиков в популяции.

        - elite\_radius (float): Радиус элитных участков для каждой пчелы.

        - perspective\_radius (float): Радиус перспективных участков для каждой пчелы.

        - num\_elite (int): Количество элитных участков для обновления координат пчелы.

        - num\_perspective (int): Количество перспективных участков для обновления координат пчелы.

        - agents\_per\_perspective (int): Количество агентов, отправляемых на каждый перспективный участок.

        - agents\_per\_elite (int): Количество агентов, отправляемых на каждый элитный участок.

        - bounds (listусловие останова).

        - fitness\_function: заданная фитнесс-функция

        '''

        self.num\_scouts = num\_scouts

        self.elite\_radius = elite\_radius

        self.perspective\_radius = perspective\_radius

        self.num\_elite = num\_elite

        self.num\_perspective = num\_perspective

        self.agents\_per\_perspective = agents\_per\_perspective

        self.agents\_per\_elite = agents\_per\_elite

        self.bounds = bounds

        self.max\_epochs = max\_epochs

        self.stagnation\_limit = stagnation\_limit

        self.best\_bees = []

        self.fitness\_function = fitness\_function

    def set\_options(self, root, ax, canvas, results\_text,bound\_start,bound\_end,target\_func):

        self.canvas = canvas

        self.root = root

        self.ax = ax

        self.results\_text = results\_text

        self.ax = ax

        self.bound\_start = bound\_start

        self.bound\_end = bound\_end

        self.target\_func = target\_func

    def initialize\_bees(self):

        bees = []

        for \_ in range(self.num\_scouts):

            # Инициализация случайных координат для каждой пчелы в пределах заданных границ

            coords = np.array([random.uniform(self.bounds[i][0], self.bounds[i][1]) for i in range(len(self.bounds))], dtype='float')

            bees.append(Bee(coords, self.fitness\_function(coords)))

        return bees

    def optimize(self):

        # инициализация начальной популяции пчел

        bees = self.initialize\_bees()

        stagnation\_count = 0

        best\_fitness = float('inf')

        for epoch in range(self.max\_epochs):

            # Сортировка пчел по их приспособленности (лучшие впереди)

            bees = sorted(bees, key=lambda bee: bee.fitness)

            self.best\_bees = bees[:self.num\_elite]

            x\_range = np.linspace(self.bound\_start, self.bound\_end, 100)

            y\_range = np.linspace(self.bound\_start, self.bound\_end, 100)

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

            Z = np.zeros\_like(X)

            for i in range(X.shape[0]):

                for j in range(X.shape[1]):

                    Z[i, j] = self.target\_func(np.array([X[i, j], Y[i, j]]))

            self.ax.cla()

            self.canvas.draw()

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

            self.ax.set\_xlabel('X')

            self.ax.set\_ylabel('Y')

            self.ax.set\_zlabel('Z')

            self.ax.set\_xticks(np.arange(self.bound\_start, self.bound\_end + 1, 2))

            self.ax.set\_yticks(np.arange(self.bound\_start, self.bound\_end + 1, 2))

            for i in range(self.num\_scouts):

                # Исследование окружения для каждой пчелы-разведчика

                self.explore(bees[i])

                # print(bees[i].fitness,bees[i].coords[0],bees[i].coords[1])

                self.ax.scatter(bees[i].coords[0], bees[i].coords[1], bees[i].fitness, color='black',

                           s=10)

            # Выбор лучших пчел из текущей эпохи

            bees = self.select\_best(bees)

            # print(bees[i].fitness)

            # Проверка условия стагнации

            current\_best\_fitness = self.best\_bees[0].fitness

            if current\_best\_fitness < best\_fitness:

                best\_fitness = current\_best\_fitness

                stagnation\_count = 0

            else:

                stagnation\_count += 1

            if stagnation\_count >= self.stagnation\_limit:

                self.results\_text.insert(tk.END,f"\nСтагнация. Оптимизация остановлена на итерации {epoch}.\n")

                break

            # self.ax.scatter(self.best\_bees[0].coords[0], self.best\_bees[0].coords[1], self.best\_bees[0].fitness, c="black")

            self.results\_text.insert(tk.END,

                                f"Итерация {epoch}: Лучшее решение ({self.best\_bees[0].coords[0]:.8f}, {self.best\_bees[0].coords[1]:.8f}, {self.best\_bees[0].fitness:.8f})\n")

            self.canvas.draw()

            self.results\_text.yview\_moveto(1)

            self.root.update()

            # print(f'Лучшая пчела в {self.best\_bees[0].coords} и фитнесс {self.best\_bees[0].fitness}')

        # Последняя сортировка и выбор лучших пчел

        self.best\_bees = sorted(bees, key=lambda bee: bee.fitness)[:self.num\_elite]

        return self.best\_bees[0]

    def explore(self, bee):

        # Случайная фаза исследования соседнего пространства

        phi = random.uniform(-1, 1)

        phi\_elite = random.uniform(0, self.elite\_radius)

        phi\_perspective = random.uniform(0, self.perspective\_radius)

        # Выбор случайного направления для каждой координаты

        directions = [random.uniform(-1, 1) for \_ in range(len(bee.coords))]

        # Новые координаты для текущей пчелы

        new\_coords = [bee.coords[i] + phi \* (bee.coords[i] - self.best\_bees[0].coords[i]) +

                      phi\_elite \* (bee.coords[i] - random.choice(self.best\_bees).coords[i]) +

                      phi\_perspective \* random.choice(self.best\_bees).coords[i] \*

                      directions[i] for i in range(len(bee.coords))]

        # Ограничение координат в пределах заданных границ

        new\_coords = np.array(

            [max(min(new\_coords[i], self.bounds[i][1]), self.bounds[i][0]) for i in range(len(new\_coords))])

        # Вычисление новой приспособленности для пчелы

        new\_fitness = self.fitness\_function(new\_coords)

        # Обновление координат и приспособленности, если новая точка лучше

        if new\_fitness < bee.fitness:

            bee.coords = new\_coords

            bee.fitness = new\_fitness

    def select\_best(self, bees):

        # Сортировка всех пчел

        bees.sort(key=lambda bee: bee.fitness)

        # Выбор лучших мест и инициализация новых пчел для заполнения оставшихся мест

        return bees[:self.num\_perspective] + self.initialize\_bees()[:self.num\_scouts - self.num\_perspective]

def BeesAlgorithm(frame,root,ax,canvas):

        # Функция Розенброка для оптимизации

        def himel\_function(x\_arr):

            x, y = x\_arr[0], x\_arr[1]

            return (x \*\* 2 + y - 11) \*\* 2 + (x + y \*\* 2 - 7) \*\* 2

        def rosenbrock\_function(x\_arr):

            x, y = x\_arr[0], x\_arr[1]

            return (1 - x) \*\* 2 + 100 \* (y - x \*\* 2) \*\* 2

        def rastrigin(x\_arr):

            size = len(x\_arr)

            return 10 \* size + np.sum(x\_arr \*\* 2 - 10 \* np.cos(2 \* np.pi \* x\_arr))

        def run\_optimization():

            ax.cla()

            function\_choice = function\_var.get()

            target\_func = himel\_function

            if function\_choice == "Функция Химмельблау":

                target\_func = himel\_function

            elif function\_choice == "Функция Розенброка":

                target\_func = rosenbrock\_function

            elif function\_choice == "Функция Растригина":

                target\_func = rastrigin

            iterations=int(iteration.get())

            scout = int(scouts.get())  # разведчики

            perspective\_B = int(perspective\_b.get())

            best\_B = int(best\_b.get())  # лучшие пчелы

            perspective\_A = int(perspective\_a.get())

            best\_A  = int(best\_a.get())

            size\_A = int(size\_a.get())

            stop\_entry = int(stop.get())

            bound\_start = float(-4)

            bound\_end = float(4)

            bounds = [(bound\_start, bound\_end) for i in range(2)]

            x\_range = np.linspace(bound\_start, bound\_end, 100)

            y\_range = np.linspace(bound\_start, bound\_end, 100)

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

            Z = np.zeros\_like(X)

            for i in range(X.shape[0]):

                for j in range(X.shape[1]):

                    Z[i, j] = target\_func(np.array([X[i, j], Y[i, j]]))

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

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

            algorithm = BeeAlgorithm(scout, size\_A, size\_A, best\_A, perspective\_A,

                                     perspective\_B, best\_B, bounds, iterations, stop\_entry,

                                     target\_func)

            algorithm.set\_options(root, ax, canvas, results\_text,bound\_start,bound\_end,target\_func)

            best\_bee = algorithm.optimize()

            ax.scatter(best\_bee.coords[0], best\_bee.coords[1], best\_bee.fitness, c="red")

            results\_text.insert(tk.END,

                                f"Лучшее решение ({best\_bee.coords[0]:.8f}, {best\_bee.coords[1]:.8f}, {best\_bee.fitness:.8f})\n")

            canvas.draw()

            root.update()

        param\_frame2 = frame

        bg\_color = "#3d6466"

        param\_frame2.configure(bg=bg\_color)

        # Параметры задачи

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

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

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

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

        ttk.Label(param\_frame2, text="Пчел в лучшем участке", font=("Helvetica", 10)).grid(row=4, column=0)

        ttk.Label(param\_frame2, text="Перспективных участков", font=("Helvetica", 10)).grid(row=5, column=0)

        ttk.Label(param\_frame2, text="Лучших участков", font=("Helvetica", 10)).grid(row=6, column=0)

        ttk.Label(param\_frame2, text="Размер участков", font=("Helvetica", 10)).grid(row=7, column=0)

        ttk.Label(param\_frame2, text="Критерий останова", font=("Helvetica", 10)).grid(row=8, column=0)

        iteration = tk.IntVar(value=200)

        scouts = tk.IntVar(value=20) #разведчики

        perspective\_b = tk.IntVar(value=10) #перспективных пчел

        best\_b = tk.IntVar(value=20) #лучшие пчелы

        perspective\_a  = tk.IntVar(value=3) #перпективных участков

        best\_a = tk.IntVar(value=1)  # лучших участков

        size\_a = tk.DoubleVar(value=0.5)  # размер участков

        stop = tk.DoubleVar(value=20)  # задержка

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

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

        perspective\_b\_entry = ttk.Entry(param\_frame2, textvariable=perspective\_b)

        best\_b\_entry = ttk.Entry(param\_frame2, textvariable=best\_b)

        perspective\_a\_entry = ttk.Entry(param\_frame2, textvariable=perspective\_a)

        best\_a\_entry = ttk.Entry(param\_frame2, textvariable=best\_a)

        size\_a\_entry = ttk.Entry(param\_frame2, textvariable=size\_a)

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

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

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

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

        best\_b\_entry.grid(row=4, column=1)

        perspective\_a\_entry.grid(row=5, column=1)

        best\_a\_entry.grid(row=6, column=1)

        size\_a\_entry.grid(row=7, column=1)

        stop\_entry.grid(row=8, column=1)

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

        separator.grid(row=9, column=0, columnspan=2, sticky="ew", pady=10)

        # Параметры функции

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

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

        function\_choices = ["Нажмите для выбора","Функция Химмельблау", "Функция Розенброка",

                        "Функция Растригина"]

        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)

        # 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=14, column=1, padx=10, pady=10)

        ttk.Label(param\_frame2, text="Выполнение и результаты", font=("Helvetica", 12,"bold")).grid(row=11, column=0, pady=10)

        results\_text = scrolledtext.ScrolledText(param\_frame2, wrap=tk.WORD, height=16, width=40, padx=2, state=tk.DISABLED)

        results\_text.grid(row=14, column=0, padx=10)