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 №6 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студенты 46 группы

Стасюк К.В.

Баева Д.Н.

Преподаватель:

Нигодин Е.А.

Краснодар 2023

**Цель работы:** разработать алгоритм минимизации функции Розенброка искусственной иммунной сетью.

**Ход работы:**

Иммунная сеть представляет собой математическую структуру, имитирующую некоторые функции иммунной системы человека. Многие внешние черты сближают иммунную сеть с такой современной информационной технологией, как нейронные сети. Подобно нейронным сетям, иммунные сети обладают способностью к обучению, прогнозированию и принятию решений в незнакомой ситуации. Как и нейронные сети, иммунные сети не нуждаются в заранее известной модели задачи, а строят эту модель на основе полученной информации. Наконец, как нейронные, так и иммунные сети наиболее эффективны при решении плохо формализованных (слабо структурированных) задач, таких как прогнозирование, классификация и управление.

Для использования в информационных технологиях наибольший интерес представляют функции приобретенной иммунной системы, которые реализуют, прежде всего, лимфоциты. Лимфоциты— главные клетки иммунной системы, вырабатывающие антитела в ответ на вторжение антигенов. По функциональным признакам различают несколько типов лимфоцитов, главными из которых являются В-лимфоциты, осуществляющие распознавание антигенов и выработку антител. Среди В-лимфоцитов выделяют клетки «памяти», живущие относительно долго и хранящие в себе информацию о встреченных ранее организмом чужеродных белках.

В общих чертах, алгоритм оптимизации с помощью иммунной системы работает следующим образом:

1. Создание начальной популяции антител (кандидатов на решение задачи).
   * Начальная популяция антител создается случайным образом.
2. Оценка каждого антитела в популяции на основе функции оценки (целевой функции) – определение афинности.
   * Каждое антитело оценивается на основе целевой функции, которая определяет, насколько хорошо это антитело решает задачу оптимизации.
3. Выбор наилучших антител для создания новой популяции (клонации и мутации).
   * Наилучшие антитела (с более высокой афинностью) выбираются на основе их оценки целевой функции. Это позволяет сохранять лучшие решения и вносить в них изменения.
4. Модификация выбранных антител (клонирование, мутация и т. д.) для создания новых кандидатов на решение задачи.
   * Выбранные антитела могут быть скопированы, изменены или комбинированы для создания новых кандидатов на решение задачи.
5. Оценка новых кандидатов на решение задачи.
   * Новые кандидаты оцениваются на основе целевой функции.
6. Сравнение новых кандидатов с текущими антителами и замена наихудших антител в популяции на новых кандидатов.
   * Новые кандидаты сравниваются с текущими антителами, и наихудшие антитела заменяются на новых кандидатов.
7. Повторение шагов 3–6 до тех пор, пока не будет достигнуто условие остановки (например, достижение определенного значения целевой функции или превышение максимального количества итераций).

Алгоритм оптимизации искусственной иммунной сетью имеет свои преимущества и недостатки.

**Преимущества:**

* - Алгоритм оптимизации с помощью иммунной системы может быть эффективным для решения сложных задач оптимизации, которые могут быть трудными для других методов оптимизации.
* - Он может быть эффективным для решения задач, которые требуют поиска нескольких оптимальных решений.
* - Алгоритм оптимизации с помощью иммунной системы может быть эффективным для решения задач, которые требуют адаптивности к изменяющимся условиям.

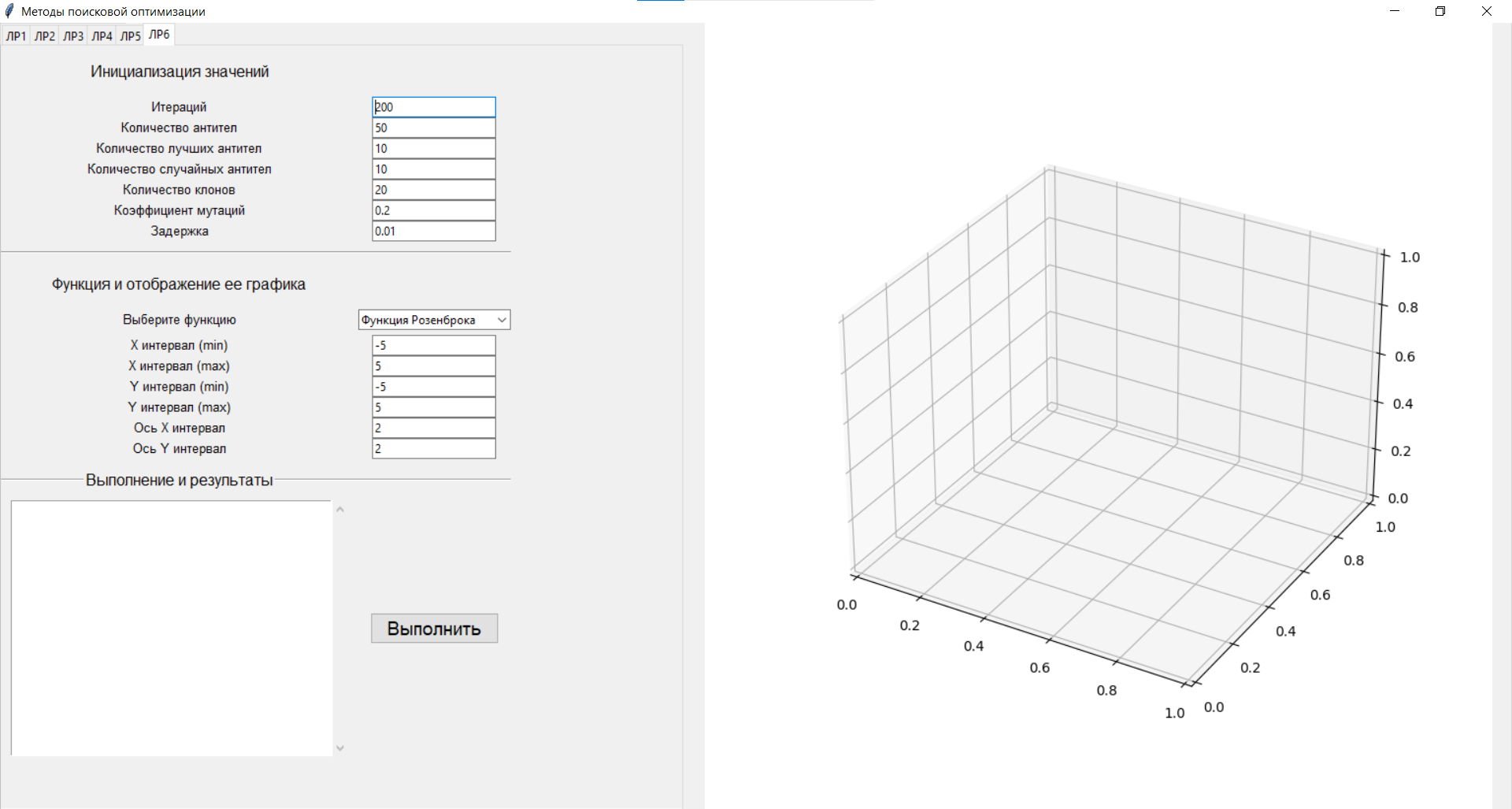
**Недостатки:**

* - Алгоритм оптимизации с помощью иммунной системы может быть сложен для понимания и реализации.
* - Он может потребовать большого количества вычислительных ресурсов и времени для выполнения.
* - Алгоритм оптимизации с помощью иммунной системы может быть чувствителен к выбору параметров и настроек, что может затруднить его использование в некоторых случаях.

**Особенности реализации генетического алгоритма.**

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 «ЛР6», ввести количество итераций, количество антител, количество лучших антител, количество случайных антител, количество клонов, коэффициент мутаций, задержка. 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

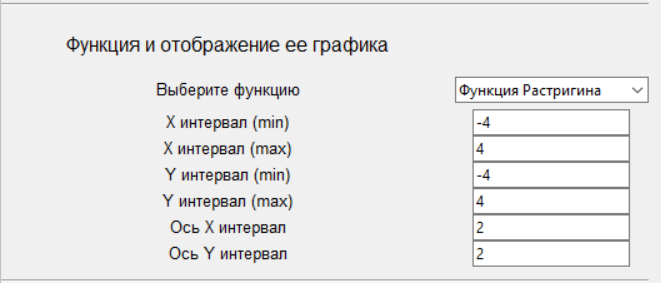
Автоматически созданное описание

Рисунок 2 – Выполнение и результаты.

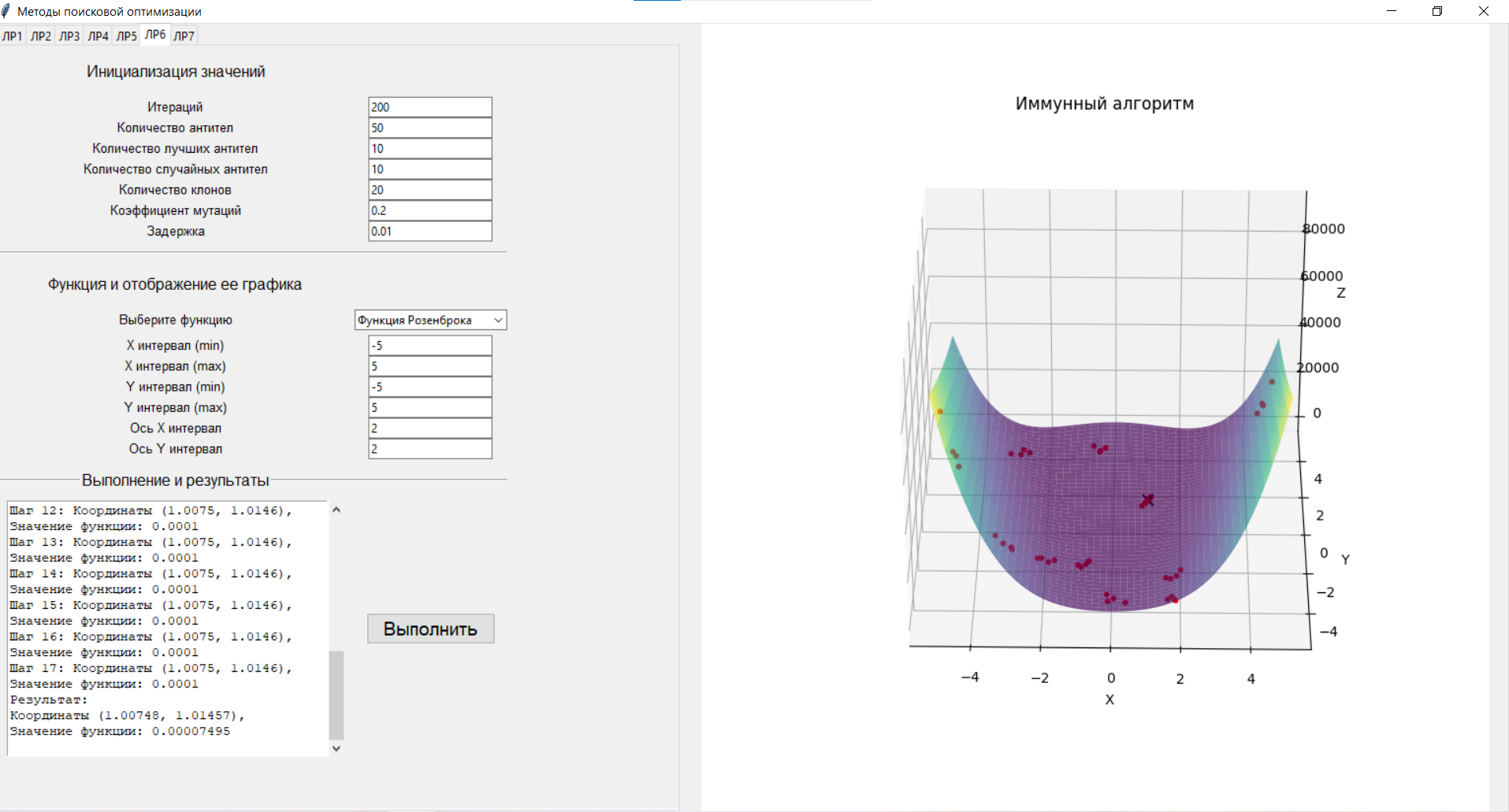
Также, в интерфейсе программы есть панель управления «Функция и отображение её графика», в котором можно выбрать необходимую функцию и задать её параметры:

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 изучен, а затем разработан алгоритм минимизации функции Розенброка искусственной иммунной сетью.

**Листинг программы**

Файл LR6.py

import tkinter as tk

from tkinter import ttk

from tkinter import scrolledtext

import numpy as np

import math

import random

from operator import itemgetter

import time

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

# Функция Розенброка для оптимизации

def rosenbrock\_function(x,y):

return (1.0 - x) \*\* 2 + 100.0 \* (y - x \* x) \*\* 2

class AIS:

def \_\_init\_\_(self, iter\_number, num\_antibodies, num\_best, num\_random, num\_clones, mutation\_rate,

x\_range,

y\_range):

self.iter\_number = iter\_number

self.num\_antibodies = num\_antibodies

self.num\_best = num\_best

self.num\_random = num\_random

self.num\_clones = num\_clones

self.mutation\_rate = mutation\_rate

self.x\_range = x\_range

self.y\_range = y\_range

self.antibodies = [[random.uniform(self.x\_range[0], self.x\_range[1]),

random.uniform(self.y\_range[0], self.y\_range[1]),

0.0] for \_ in range(self.num\_antibodies)]

for antibody in self.antibodies:

antibody[2] = rosenbrock\_function(antibody[0], antibody[1])

self.antibody\_best = min(self.antibodies, key=itemgetter(2))

def sort\_antibodies(self):

self.antibodies.sort(key=lambda x: x[2])

def mutate(self, antibody):

new\_x\_val = np.clip(antibody[0] + self.mutation\_rate \* np.random.randn(), self.x\_range[0],

self.x\_range[1])

new\_y\_val = np.clip(antibody[1] + self.mutation\_rate \* np.random.randn(), self.y\_range[0],

self.y\_range[1])

return [new\_x\_val, new\_y\_val, rosenbrock\_function(new\_x\_val, new\_y\_val)]

def next\_iteration(self):

for iteration in range(self.iter\_number):

self.sort\_antibodies()

for i in range(self.num\_best, self.num\_antibodies):

if i < self.num\_best + self.num\_random:

self.antibodies[i] = [random.uniform(self.x\_range[0], self.x\_range[1]),

random.uniform(self.y\_range[0], self.y\_range[1]),

0.0]

self.antibodies[i][2] = rosenbrock\_function(self.antibodies[i][0], self.antibodies[i][1])

else:

self.antibodies[i] = self.mutate(self.antibodies[i - self.num\_random])

self.antibodies[i][2] = rosenbrock\_function(self.antibodies[i][0], self.antibodies[i][1])

self.antibody\_best = min(self.antibodies, key=itemgetter(2))

def run\_optimization():

iter\_number = iterations\_var.get()

antibodies\_num = antibodies\_number\_var.get()

best\_num = best\_number\_var.get()

random\_num = random\_number\_var.get()

clones\_num = clones\_number\_var.get()

mutation\_coef = mutation\_rate\_var.get()

delay = delay\_var.get()

# 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 = rosenbrock\_function(X, Y)

ax.cla()

# 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\_title("Иммунный алгоритм")

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()))

ais = AIS(iter\_number, antibodies\_num, best\_num, random\_num, clones\_num, mutation\_coef,

[x\_interval\_min.get(), x\_interval\_max.get()], [y\_interval\_min.get(), y\_interval\_max.get()])

# отрисовка стартовой популяции

for antibody in ais.antibodies:

ax.scatter(antibody[0], antibody[1], antibody[2], c="red", s=10)

# ax.scatter(ais.antibody\_best[0], ais.antibody\_best[1], ais.antibody\_best[2], c="blue")

canvas.draw()

root.update()

# очистка графика

ax.cla()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_title("Иммунный алгоритм")

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

canvas.draw()

cnt = 0

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

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

# отрисовка промежуточной популяции и эволюция

for i in range(iter\_number):

prev\_antibody\_best = ais.antibody\_best

ais.next\_iteration()

# подсчет продолжительности стагнации

if abs(prev\_antibody\_best[2] - ais.antibody\_best[2]) < 0.0001:

cnt += 1

else:

cnt = 0

if cnt == 15:

break

# отрисовка промежуточной популяции

for antibody in ais.antibodies:

ax.scatter(antibody[0], antibody[1], antibody[2], c="red", s=10)

# ax.scatter(ais.antibody\_best[0], ais.antibody\_best[1], ais.antibody\_best[2], c="blue")

results\_text.insert(tk.END,

f"Шаг {i}: Координаты ({ais.antibody\_best[0]:.4f}, "

f"{ais.antibody\_best[1]:.4f}),"

f" Значение функции: {ais.antibody\_best[2]:.4f}\n")

results\_text.yview\_moveto(1)

canvas.draw()

root.update()

time.sleep(delay)

# очистка графика

ax.cla()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_title("Иммунный алгоритм")

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

canvas.draw()

# отрисовка результирующей популяции

for antibody in ais.antibodies:

ax.scatter(antibody[0], antibody[1], antibody[2], c="red", s=10)

ax.scatter(ais.antibody\_best[0], ais.antibody\_best[1], ais.antibody\_best[2], c='black', marker='x', s=60)

canvas.draw()

root.update()

results\_text.insert(tk.END,

f"Результат:\nКоординаты ({ais.antibody\_best[0]:.5f}, "

f"{ais.antibody\_best[1]:.5f}),\nЗначение функции: {ais.antibody\_best[2]:.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="Итераций", font=("Helvetica", 10)).grid(row=1, column=0)

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

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

ttk.Label(param\_frame2, text="Количество случайных антител", font=("Helvetica", 10)).grid(row=4, column=0)

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

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

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

iterations\_var = tk.IntVar(value=200)

antibodies\_number\_var = tk.IntVar(value=50)

best\_number\_var = tk.IntVar(value=10)

random\_number\_var = tk.IntVar(value=10)

clones\_number\_var = tk.IntVar(value=20)

mutation\_rate\_var = tk.DoubleVar(value=0.2)

delay\_var = tk.DoubleVar(value=0.01)

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

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

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

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

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

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

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

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

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

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

random\_number\_entry.grid(row=4, column=1)

clones\_number\_entry.grid(row=5, column=1)

mutation\_rate\_entry.grid(row=6, column=1)

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

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

separator.grid(row=8, 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 = ["Функция Розенброка"]

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=18, 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=21, column=0, padx=10)