**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Разработка и реализация алгоритма роевого интеллекта для решения задач глобальной оптимизации»**

**Вариант – 11**

**Студент гр. 23Б16-пу**

**Крылов А. С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

**Оглавление**

[Цель работы 3](#_Toc183158203)

[Описание задачи 3](#_Toc183158204)

[Теоретическая часть 3](#_Toc183158205)

[Основные шаги работы 4](#_Toc183158206)

[Блок схема программы 5](#_Toc183158207)

[Описание програмы: 6](#_Toc183158208)

[Описание функций: 7](#_Toc183158209)

[Описание переменных: 8](#_Toc183158210)

[Рекомендации пользователя 10](#_Toc183158211)

[Рекомендации программиста 10](#_Toc183158212)

[Исходный код программы 10](#_Toc183158213)

[Контрольный пример 11](#_Toc183158214)

[Вывод 15](#_Toc183158215)

# Цель работы

Целью лабораторной работы является исследование особенностей алгоритмов роевого интеллекта для решения задач глобальной оптимизации и сравнение с генетическим алгоритмом.

# Описание задачи

1. Изучить особенности кодирования роевого интеллекта
2. Написать программу поиска минимума функции, выбрав вариант тестовой функции из таблицы 1, согласна своему номеру в группе.
3. Протестировать программу на выбранной тестовой функции.

# 

# Теоретическая часть

Для выполнения работы использован python с библиотеками tkinter, numpy, matplotlib с расширениями.

Алгоритм роя частиц (Particle Swarm Optimization, PSO) — это метод оптимизации, вдохновленный поведением коллективного интеллекта, например, роем птиц или рыб. Алгоритм используется для поиска оптимума функции, работая с множеством потенциальных решений, называемых частицами.

Таблица 1. Сравнение алгоритмов

|  |  |  |
| --- | --- | --- |
| Критерий | Алгоритм роя частиц (PSO) | Генетический алгоритм (GA) |
| Основная идея | Основан на коллективном поведении роя (например, птиц или рыб). Частицы стремятся к лучшему положению, учитывая собственный и глобальный опыт. | Основан на принципах эволюции (отбор, скрещивание, мутация). Решения эволюционируют через поколения. |
| Представление решений | Решения представлены в виде частиц, каждая из которых имеет положение и скорость. | Решения представлены в виде 'хромосом' (обычно строк или массивов), которые кодируют параметры задачи. |
| Итерационный процесс | Частицы обновляют своё положение и скорость на основе собственной и коллективной информации. | Новое поколение формируется через механизмы отбора, скрещивания и мутации. |
| Обновление решений | Используются уравнения обновления скорости и положения. | Используются генетические операторы: селекция, кроссовер, мутация. |
| Параметры алгоритма | - Инерционный коэффициент w - Коэффициенты локального (c1) и глобального (c2) лучших положений - Максимальная скорость частиц. | - Размер популяции - Вероятность мутации - Вероятность скрещивания - Число поколений. |
| Природа решений | Каждая частица обновляет своё положение, ориентируясь на лучшее положение (глобальное и личное). | Популяция решений обновляется через процесс отбора и генерации новых хромосом. |
| Сходимость | Быстро сходится в начальных итерациях, но может застрять в локальном минимуме. | Сходится медленнее, но лучше исследует пространство поиска благодаря мутациям. |
| Эффективность поиска | Лучше при задачах с гладкими и непрерывными функциями. | Лучше при задачах с дискретными решениями или функциями с множеством локальных минимумов. |
| Характер исследования | Баланс между исследованием (exploration) и эксплуатацией (exploitation) регулируется коэффициентами. | Более мощное исследование (exploration), так как мутации и кроссоверы вводят случайность. |
| Адаптивность | Менее адаптивен к сложным ландшафтам с большими разрывами или дискретными областями. | Более адаптивен за счёт мутации, позволяющей пересекать разрывы в пространстве поиска. |
| Простота реализации | Прост в реализации и настройке. | Более сложен, так как требует реализации генетических операторов и их параметров. |
| Применение | - Оптимизация параметров моделей - Машинное обучение - Управление и планирование. | - Комбинаторные задачи (расписания, маршруты) - Задачи с дискретными параметрами. |

# 

## Основные шаги работы

* Создаются частицы каждая из которых обладает текущей позицией, скоростью, лучшим положением (локальный минимум), и стремится к глобально лучшему положению.
* В каждом шаге итерации обновляются скорости и положения частиц на основе уравнения:
* Частицы обновляют свои позиции и проверяют, достигли ли они более оптимального значения функции.

## Блок схема программы

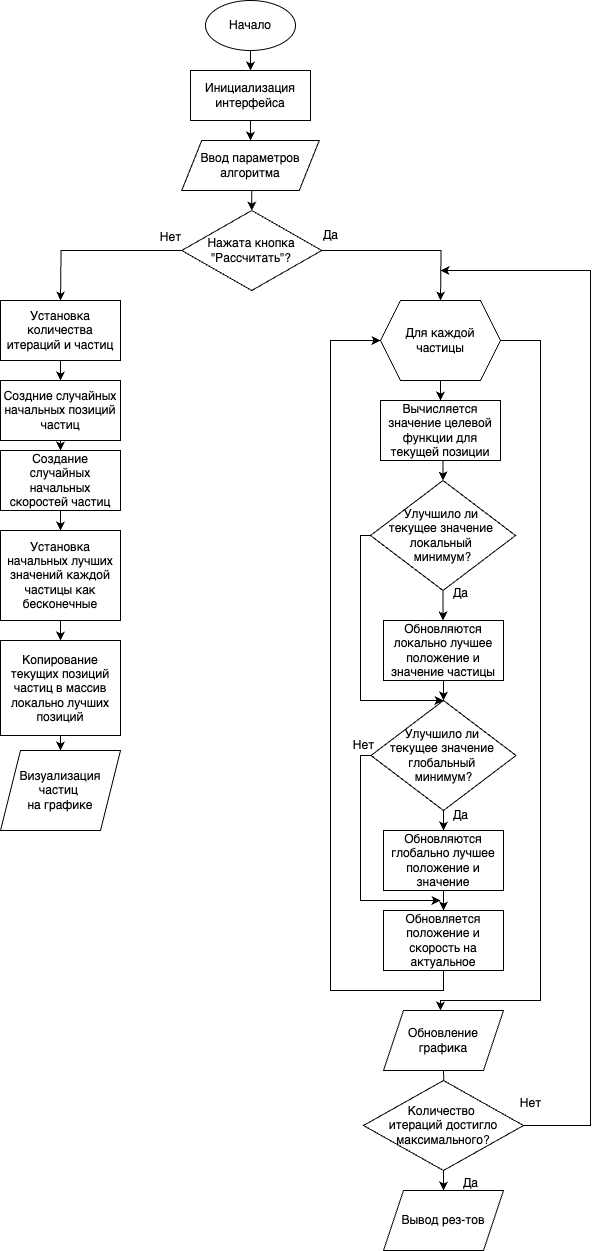


Рис 1. Блок-схема программы.

## Описание програмы:

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек:

* NumPy: Используется для работы с массивами, создания случайных чисел и выполнения векторных вычислений. Это основа для работы с позициями и скоростями частиц.
* Tkinter: Стандартная библиотека Python для создания графических интерфейсов. Обеспечивает удобный пользовательский интерфейс для задания параметров, запуска алгоритма и отображения результатов.
* Matplotlib: Используется для построения графиков и визуализации движения частиц в 3D-пространстве. В сочетании с mpl\_toolkits.mplot3d позволяет отображать частицы в трехмерной координатной системе.
* Matplotlib.backends.backend\_tkagg: Интегрирует графики Matplotlib в интерфейс Tkinter.

## Описание функций:

Таблица 2. Описание функций

|  |  |  |  |
| --- | --- | --- | --- |
| Название функции | Описание | Параметры | Описание параметров |
| function(x) | Целевая функция, вычисляющая значение на основе переданного вектора. | x (numpy.ndarray) | Вектор из трех чисел, представляющий позицию частицы в пространстве. |
| initialize\_swarm() | Инициализирует начальные позиции, скорости частиц и их лучшие значения. | Нет | Нет |
| update\_velocity(position, velocity, best\_position, global\_best\_position) | Обновляет скорость частицы на основе текущей скорости, собственного лучшего и глобального лучшего положения. | position, velocity, best\_position, global\_best\_position | Текущая позиция, скорость частицы, лучшее положение частицы, глобальное лучшее положение. |
| run\_pso() | Запускает процесс оптимизации с начальной инициализацией и анимацией. | Нет | Нет |
| animate\_pso() | Выполняет одну итерацию алгоритма, обновляет позиции, скорости частиц и их лучшее положение, перерисовывает график. | Нет | Нет |
| create\_particles() | Создает частицы и обновляет график их начальных положений. | Нет | Нет |
| create\_gui() | Создает графический интерфейс с настройкой параметров, отображением графика и выводом результатов. | Нет | Нет |

## 

## Описание переменных:

Таблица 3. Описание переменных

|  |  |  |  |
| --- | --- | --- | --- |
| Название переменной | Описание | Тип | Дополнительные пояснения |
| positions | Текущие позиции всех частиц в пространстве. | numpy.ndarray | Матрица размера (num\_particles, 3), где каждая строка — координаты частицы. |
| velocities | Текущие скорости всех частиц. | numpy.ndarray | Матрица размера (num\_particles, 3), где каждая строка — вектор скорости частицы. |
| best\_positions | Лучшие положения, достигнутые каждой частицей. | numpy.ndarray | Матрица размера (num\_particles, 3), сохраняет локально лучшие координаты частиц. |
| best\_values | Лучшие значения функции для каждой частицы. | numpy.ndarray | Вектор размера (num\_particles), содержит минимальные значения функции для частиц. |
| global\_best\_position | Глобально лучшее положение, найденное всеми частицами. | numpy.ndarray | Вектор из трех координат, представляющий глобально лучшее найденное положение. |
| global\_best\_value | Глобально лучшее значение функции, найденное всеми частицами. | float | Минимальное значение целевой функции среди всех частиц. |
| current\_iter | Текущая итерация алгоритма PSO. | int | Счетчик итераций, обновляется на каждом шаге. |
| num\_particles | Количество частиц в рое. | int | Определяется пользователем через интерфейс. |
| max\_iter | Максимальное количество итераций алгоритма. | int | Определяется пользователем через интерфейс. |
| bounds | Границы поиска для всех координат пространства. | numpy.ndarray | Матрица размера (3, 2), где каждая строка — [нижняя граница, верхняя граница] для осей x, y, z. |
| root | Основное окно интерфейса Tkinter. | tk.Tk | Корневой объект для построения GUI. |
| w\_entry, c1\_entry, c2\_entry, max\_velocity\_entry, num\_particles\_entry, max\_iter\_entry | Поля ввода для задания параметров PSO. | tk.Entry | Поля для ввода значений инерционного коэффициента, коэффициентов, числа частиц и итераций. |
| result\_text | Текстовое поле для вывода результатов оптимизации. | tk.Text | Используется для отображения глобального лучшего положения и значения. |
| iteration\_counter | Метка для отображения текущей итерации. | tk.Label | Показывает номер текущей итерации и общее количество итераций. |
| figure | Объект фигуры для графика. | matplotlib.figure.Figure | Используется для построения 3D-визуализации частиц. |
| ax | Ось графика. | Axes3D | Создана через Matplotlib, отвечает за отображение частиц в 3D-пространстве. |
| canvas | Канвас для отображения графика в Tkinter. | FigureCanvasTkAgg | Интегрирует график Matplotlib в окно Tkinter. |

# Рекомендации пользователя

* Для запуска программы, убедитесь, что у вас установлен Python и необходимые библиотеки. Запустите код в среде разработки или командной для его выполнения.
* Запуск производится файлом laba5.py.
* Периодически проверяйте корректность данных перед использованием программы.

# Рекомендации программиста

* Поддерживайте актуальность используемых библиотек и версии Python для сохранения актуальности и работоспособности кода на современных системах.
* Периодически проводите тестирование на различных входных данных для обеспечения надежности и корректности программы.

# Исходный код программы

[**https://github.com/akryloff/spbu-algorithms-and-data-structures**](https://github.com/akryloff/spbu-algorithms-and-data-structures)

# Контрольный пример

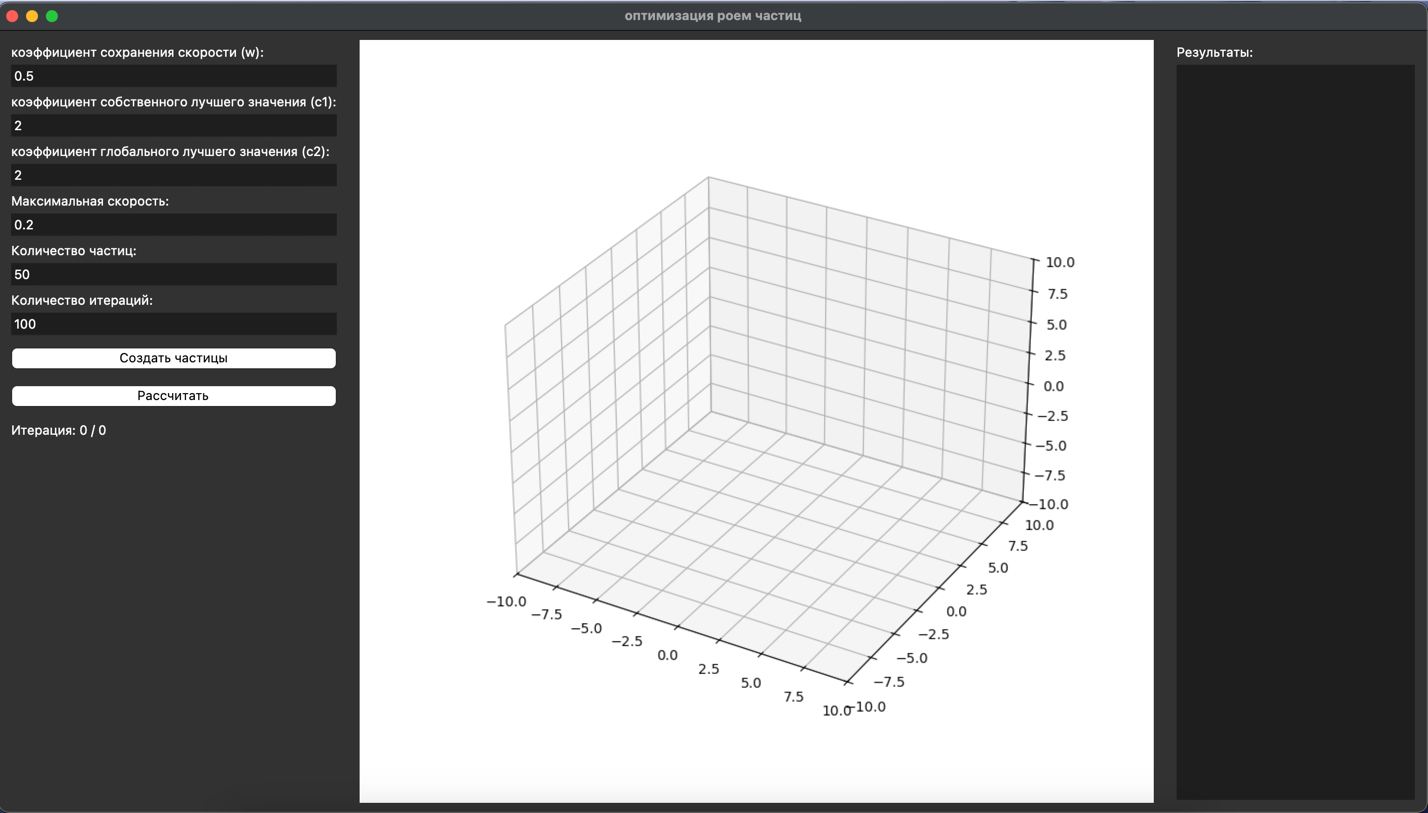


Рис 2. Интерфейс программы

Описание интерфейса:

Левая часть интерфейсa:

1. Параметры алгоритма:

Поля для ввода параметров с пояснительными метками:

* w (коэффициент сохранения скорости):
  + Поле ввода, в котором пользователь задает значение инерционного коэффициента. Этот параметр отвечает за сохранение текущей скорости частиц.
  + Значение по умолчанию: 0.5.
* c1 (коэффициент личного лучшего значения):
  + Определяет степень, с которой частица стремится к своему лучшему положению.
  + Значение по умолчанию: 2.
* c2 (коэффициент глобального лучшего значения):
  + Определяет степень, с которой частица стремится к глобально лучшему положению.
  + Значение по умолчанию: 2.
* Максимальная скорость:
  + Поле для ограничения максимальной скорости частицы.
  + Значение по умолчанию: 0.2.
* Количество частиц:
  + Поле для задания числа частиц в рое.
  + Значение по умолчанию: 50.
* Количество итераций:
  + Поле для задания максимального числа итераций алгоритма.
  + Значение по умолчанию: 100.

1. Кнопки управления:

* Создать частицы:
  + Инициализирует начальные положения и скорости частиц.
  + Отображает созданные частицы на графике.
* Рассчитать:
  + Запускает процесс оптимизации.
  + Выполняет анимацию движения частиц и поиск оптимального решения.

1. Индикация итераций:

* Текстовая метка, показывающая текущую итерацию и общее количество итераций.

Центральная часть интерфейса предназначена для визуализации процесса оптимизации в трехмерном пространстве.

##### Особенности графика:

1. Трехмерное отображение:

* График создается с использованием Matplotlib с поддержкой 3D-графиков через модуль mpl\_toolkits.mplot3d.
* Оси графика (x, y, z) ограничены значениями, заданными в матрице bounds (по умолчанию [−10,10][-10, 10][−10,10]).

1. Отображение частиц:

* Все частицы роя отображаются как точки в трехмерном пространстве:
* Глобально лучшее положение выделено красной точкой.

1. Динамическое обновление:

* На каждом шаге итерации график очищается и перерисовывается, показывая актуальные положения частиц.
* Обновление осуществляется через встроенный метод Tkinter after, обеспечивая плавную анимацию.

1. Интеграция с Tkinter:

* График встраивается в окно интерфейса с помощью FigureCanvasTkAgg.

#### Правая панель вывода результатов:

На этой панели отображаются результаты выполнения алгоритма после завершения всех итераций.

* Лучшее положение (трехмерные координаты x,y,zx, y, zx,y,z).
* Лучшее значение функции (минимум функции в найденной точке).

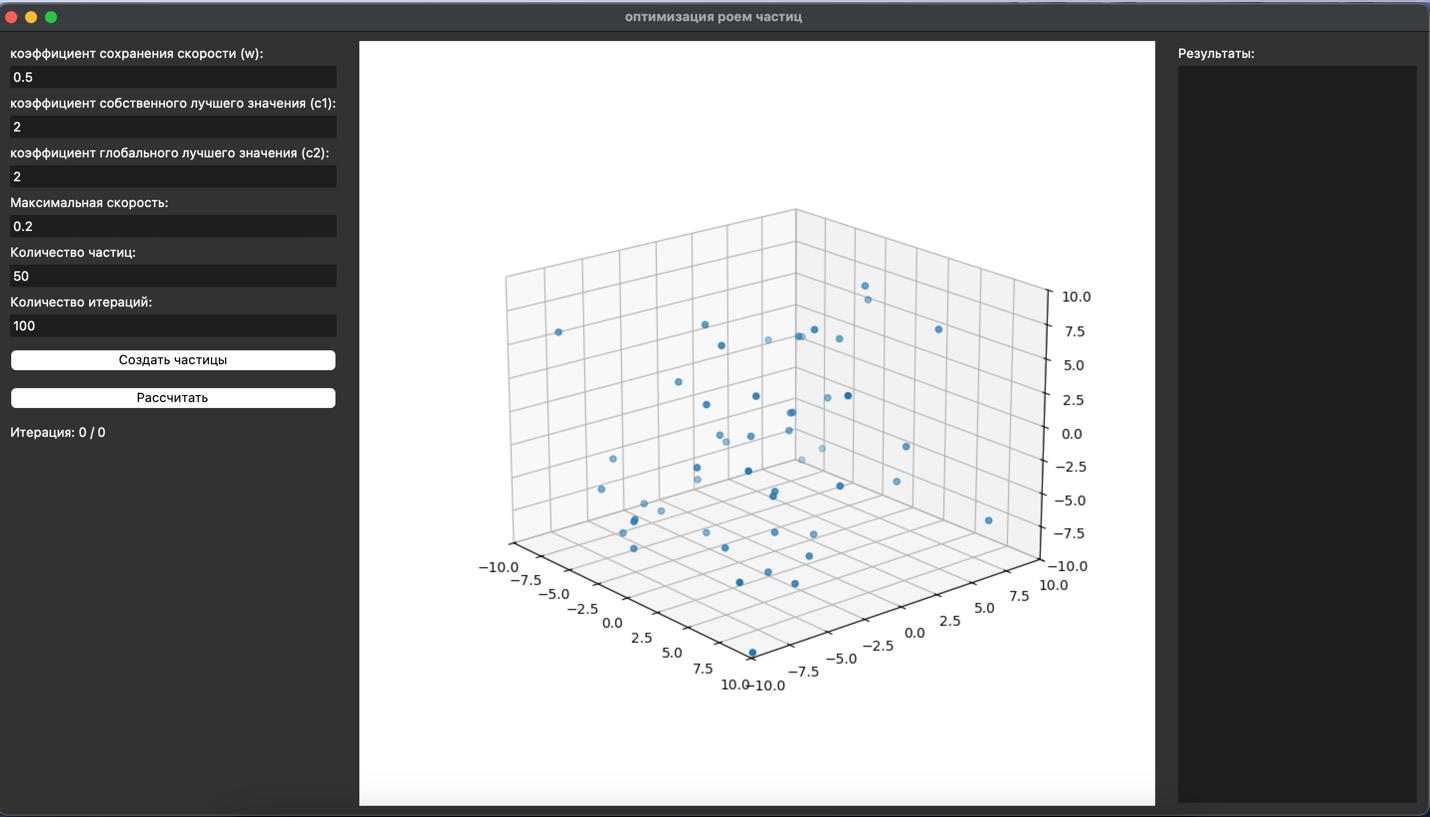


Рис 3. Результат создания частиц

При нажатии кнопки “Создать частицы”:

* Создаются случайные начальные позиции и скорости всех частиц.
* Обнуляются параметры роя (например, лучшие значения).
* Отображаются частицы на графике в центральной области интерфейса.
* Пользователь визуально видит начальное состояние роя, готовое для выполнения алгоритма.

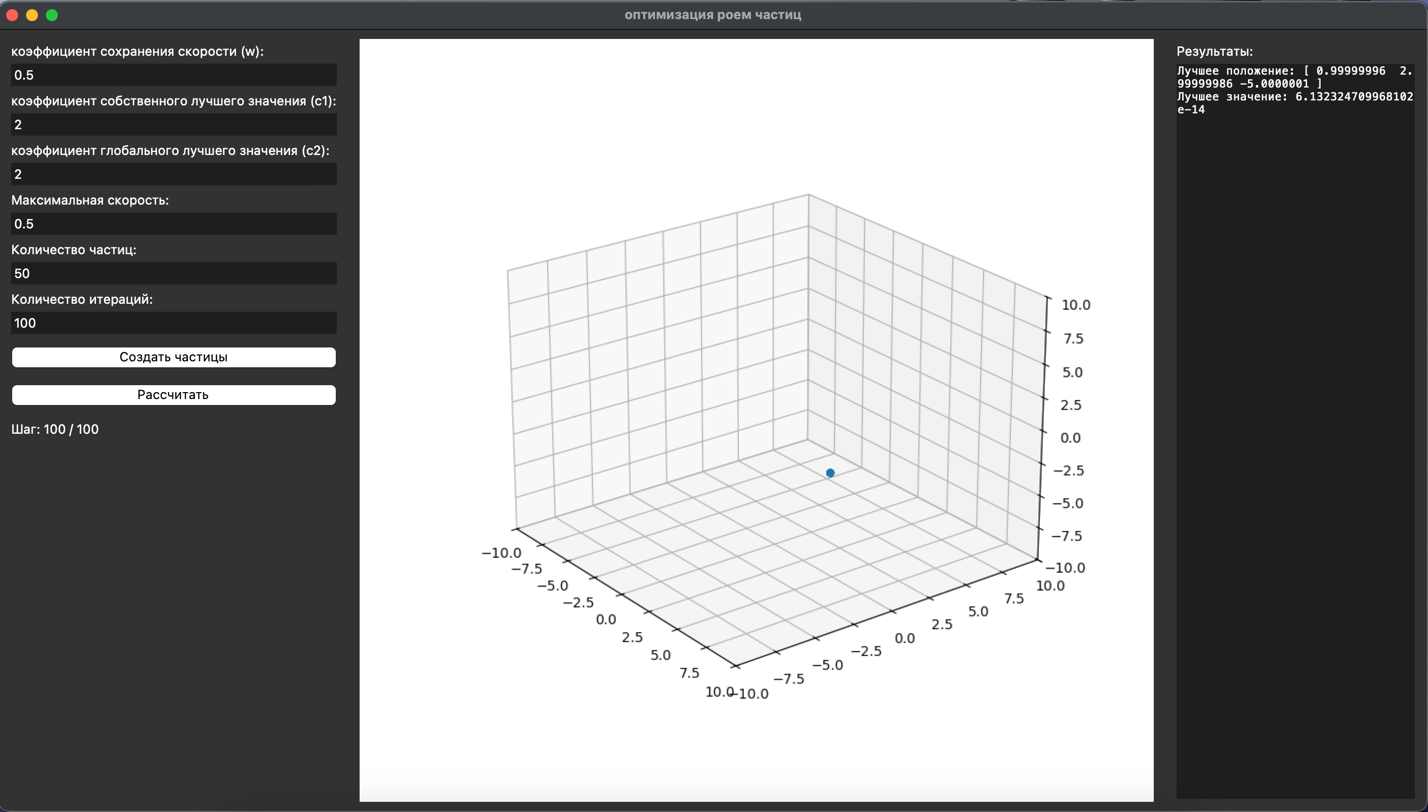


Рис 4. Результат выполнения рассчета

При нажатии кнопки "Рассчитать":

* Запускается алгоритм PSO.
* Частицы итеративно обновляют свои положения и скорости.
* Глобально лучшее значение минимизируемой функции отображается в конце.
* Пользователь видит процесс поиска оптимального решения через 3D-визуализацию.

# Вывод

В рамках данной работы были исследован алгоритм роя частиц, его применение, разработан алгоритм для конкретной функции и удобный интерфейс взаимодействия с программой. Было изучено отличие алгоритма роя частиц от генетического алгоритма.