**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 – 17**

**Студент гр. 23Б15-пу**
**Сериков К.Г.**

**Преподаватель**
**Дик А.Г.**

**Санкт-Петербург**
**2024 г.**

## Оглавление

## Цель работы

Цель данной лабораторной работы — исследование эффективности алгоритмов роевого интеллекта для решения задач глобальной оптимизации, с акцентом на методе роя частиц (Particle Swarm Optimization). Задачи исследования включают сравнение PSO с генетическими алгоритмами, а также анализ влияния модификации "Коэффициент сжатия" на его производительность. В ходе выполнения работы будет разработана программа, реализующая метод роя частиц с настройками для применения модификации "Коэффициент сжатия". Данная программа будет протестирована на заданной тестовой функции для анализа эффективности PSO и оценки влияния модификации на скорость и точность нахождения оптимального решения.

## Описание задачи (формализация задачи)

В данной лабораторной работе необходимо исследовать эффективность алгоритма роя частиц в решении задачи оптимизации, а именно — в минимизации функции: x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2.

Формализация задачи включает следующие компоненты:

1. **Изучение особенностей алгоритмов роевого интеллекта**: Исследовать принципы работы метода роя частиц, включая его применение для поиска глобальных экстремумов. Рассмотреть влияние параметров на эффективность алгоритма. Изучить особенности модификации "Коэффициента сжатия" и её влияние на сходимость.
2. **Разработка алгоритма для минимизации функции**: Создать программу, реализующую алгоритм роя частиц для поиска минимума указанной функции. Обеспечить возможность выбора модификации "Коэффициент сжатия" для улучшения сходимости.
3. **Анализ эффективности алгоритма**: Провести сравнительный анализ работы алгоритма роя частиц с модификацией и без неё, а также исследовать эффективность различных параметров. Оценить влияние модификации на скорость и точность нахождения минимума функции, на общее количество итераций, требуемое для достижения оптимального результата. Провести сравнительный анализ PSO с генетическим алгоритмом, чтобы определить преимущества и ограничения каждого метода для данной задачи глобальной оптимизации.

## Теоретическая часть

### Алгоритм роя частиц

Алгоритм роя частиц (Particle Swarm Optimization) — это метод оптимизации, основанный на моделировании поведения роя частиц (таких как рой птиц или стая рыб). Алгоритм использует множество частиц, которые перемещаются по пространству поиска, стремясь к наиболее выгодным позициям. Каждый индивид (частица) в популяции имеет свою позицию в пространстве поиска и скорость перемещения. Частицы ищут глобальный минимум, ориентируясь как на свою личную лучшую позицию (личный опыт), так и на лучшую позицию, найденную всей популяцией (глобальный опыт). Алгоритм обновляет скорость и позицию каждой частицы в зависимости от этих двух факторов, что позволяет ускорить процесс поиска оптимума.

### Модификация "Коэффициент сжатия"

Модификация алгоритма роя частиц с использованием коэффициента сжатия направлена на уменьшение размаха движений частиц по мере приближения к оптимуму. Эта модификация позволяет улучшить сходимость, ускоряя процесс нахождения минимума функции, и уменьшить вероятность "прыжков" между локальными минимумами. Она помогает уменьшить чрезмерную скорость, что делает алгоритм более стабильным и точным.

### Оценка эффективности  алгоритма роя частиц

Эффективность алгоритма роя частиц измеряется его способностью находить оптимальные решения за ограниченное время. Для оценки эффективности алгоритма можно использовать следующие параметры:

1. **Скорость сходимости:** Быстрота, с которой алгоритм достигает заданного уровня качества решения.
2. **Качество решения:** Насколько близко полученное решение к истинному оптимуму.

**Основные шаги программы**

1. **Запуск программы:** Инициализируется графический интерфейс с элементами управления.
2. **Настройка параметров:** Пользователь может задать значения параметров алгоритма в интерфейсе, такие как коэффициент текущей скорости,  коэффициент личного лучшего значения, коэффициент глобального лучшего значения, количество частиц, количество итераций. Также пользователь выбирает между стандартным и модифицированным режимами алгоритма.
3. **Запуск алгоритма:** При нажатии на кнопку "Рассчитать" программа запускает выполнение алгоритма роя частиц, используя заданные параметры.
4. **Инициализация роя:** Создаются частицы с начальным положением и скоростью. Каждая частица инициализируется случайным образом в пределах заданного диапазона значений, и устанавливаются начальные личные и глобальные лучшие позиции.
5. **Оценка приспособленности:** Для каждой частицы рассчитывается значение функции приспособленности с помощью функции fitness. Эта функция измеряет "приспособленность" каждой частицы к решению задачи.
6. **Обновление положения и скорости частиц**: В зависимости от параметров алгоритма, скорости и положения частиц обновляются на каждом шаге итерации, ориентируясь на их личные и глобальные лучшие коэффициенты. Модифицированный режим включает дополнительные корректировки в расчетах скоростей для повышения сходимости.
7. **Итерации алгоритма**: Алгоритм выполняет заданное количество итераций, обновляя скорости и положения частиц на каждом шаге и отслеживая текущее глобальное лучшее решение.
8. **Отображение результатов:** Программа отображает текущий прогресс на графике, где видны частицы, и выводит текущее лучшее решение с его значениями (x1, x2) и значением целевой функции. Количество итераций также отображается для наглядного отслеживания этапов работы алгоритма.

## Блок схема программы

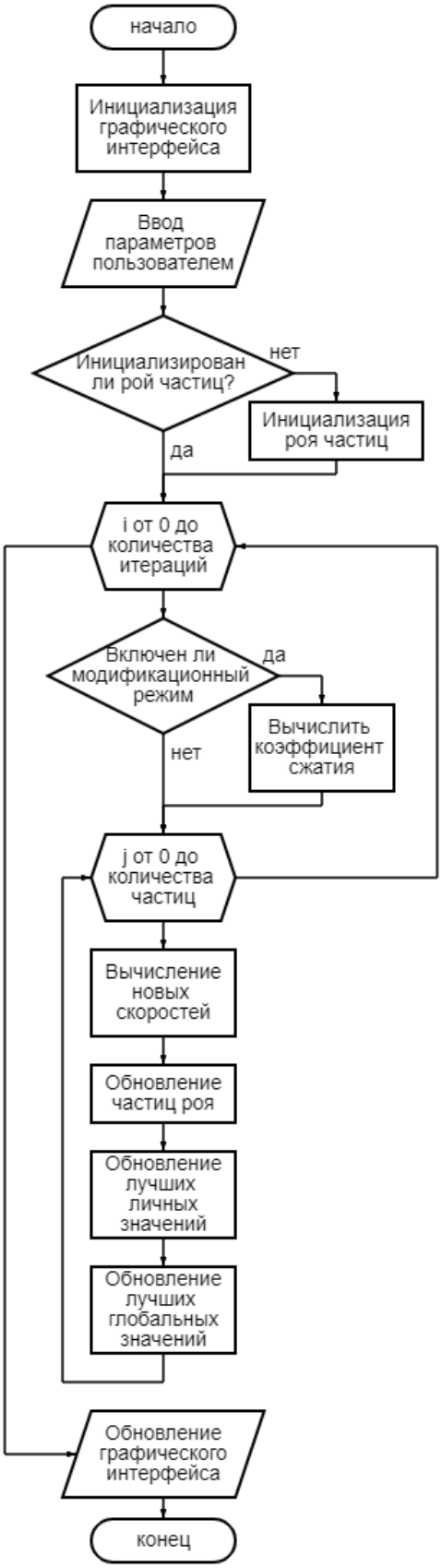


Рис 1. Блок-схема основной программы

## Описание программы

Программная реализация написана на языке Python 3.13.0 с использованием следующих библиотек: random [3], tkinter [4], numpy [5] и matplotlib [6]. Программа организована в виде графического интерфейса для решения задач оптимизации, выполняемых с помощью алгоритма роя частиц. В процессе разработки программы использовался main.py, включающий 7 функций, каждая из которых имеет чётко определённое назначение:

Таблица 1. main.py

| Функция | Описание | Возвращаемое значение |
|---|---|---|
| function | Определение функции. | float |
| fitness | Оценка приспособленности на основе значения функции. | float |
| initialize_particles | Создание роя частиц с параметрами, заданными пользователем. | None |
| update_particles | Обновляет скорости и позиции всех частиц в рое. | None |
| plot_particles | Строит и обновляет график с текущими позициями частиц. | None |
| toggle_mode | Переключение режима работы (обычный/модифицированный) и обновление отображаемого статуса режима. | None |
| run_algorithm | Запуск алгоритма роя частиц и обновление интерфейса для вывода текущих лучших решений. | None |

**Рекомендации пользователя**

Программа позволяет запустить алгоритм роя частиц (PSO) для оптимизации функции x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2 с использованием графического интерфейса.

1. **Запуск программы**: Откройте программу с помощью Python 3.13.0, чтобы инициализировать интерфейс. Интерфейс позволяет вводить параметры алгоритма и выбирать режим: стандартный или модифицированный.
2. **Настройка параметров**: Введите параметры, такие как коэффициент инерции, коэффициент собственного лучшего значения, коэффициент глобального лучшего значения и количество частиц. Вы также можете задать число итераций для работы алгоритма. Убедитесь, что эти параметры подходят для вашего эксперимента.
3. **Запуск алгоритма**: Нажмите кнопку «Рассчитать» для выполнения PSO. При этом будут вычислены текущие наилучшие результаты, и они отобразятся на графике.
4. **Оценка итогов**: В интерфейсе выводится информация о наилучшем найденном значении функции и координатах x1 и x2. График показывает текущие позиции частиц и глобальное лучшее положение, что позволяет оценить результат визуально.

**Рекомендации программиста**

Актуальность версии Python: Используйте обновленную версию Python, random, tkinter, numpy и matplotlib. Уделяйте внимание четкому именованию переменных и функций. Тщательно тестируйте на различных наборах параметров и значениях функций, чтобы удостовериться в корректной работе PSO и визуализации. Убедитесь, что интерфейс программы в Tkinter легко понимается пользователями. Разделите входные данные, управление алгоритмом и отображение результатов по разным секциям.

**Исходный код программы**

https://github.com/romplle/spbu-algorithms-and-data-structures/

## Контрольный пример

1. Запуск программы и ввод параметров

   Для запуска программы откройте main.py. Программа откроет графический интерфейс для настройки и запуска алгоритм роя частиц, предназначенного для минимизации функции x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2.

2. Ввод начальных параметров

   После запуска программы отобразится окно с полями ввода, где можно задать параметры алгоритм роя частиц (Рис. 2).

   Введите значения для следующих параметров: коэффициент инерции, коэффициент собственного лучшего значения, коэффициент глобального лучшего значения и количество частиц. Вы также можете задать число итераций для работы алгоритма и выбрать режим работы (стандартный или модифицированный).

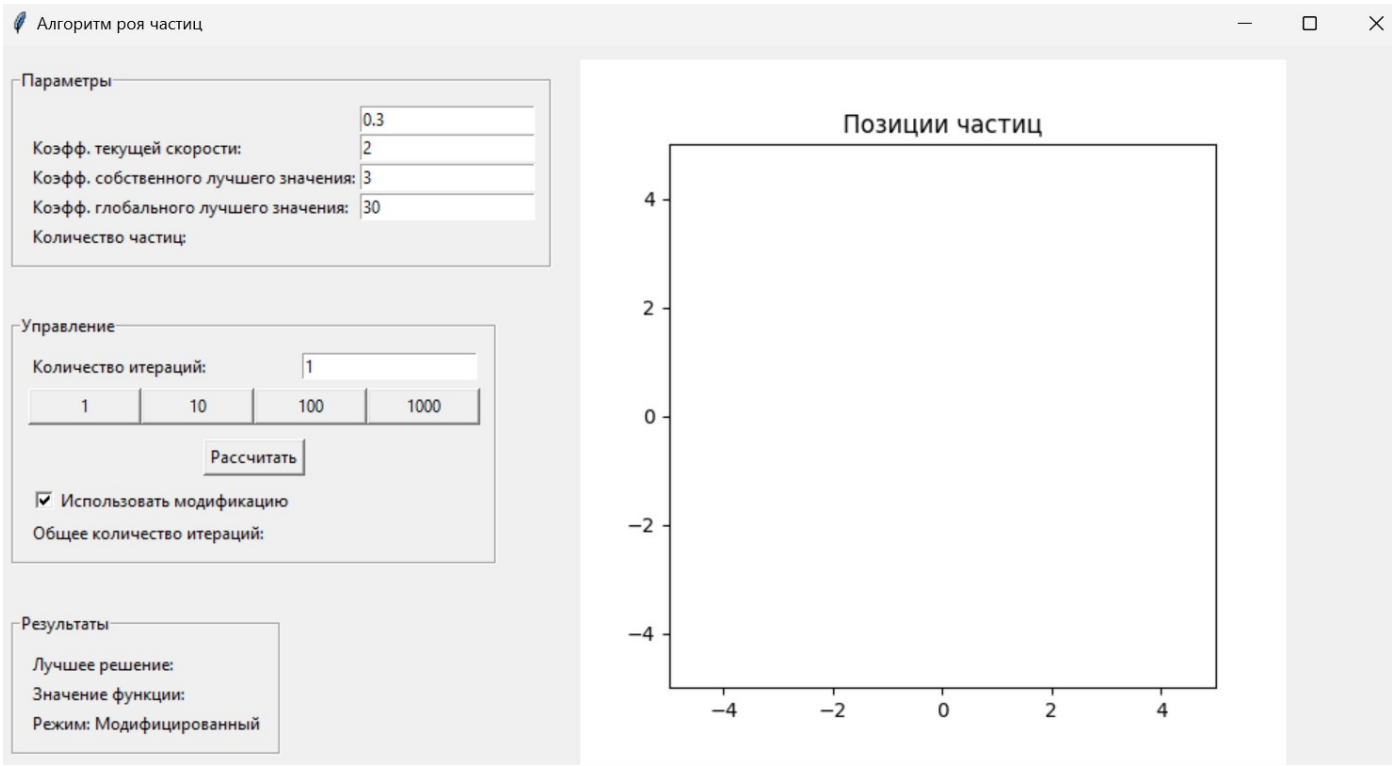


Рис. 2 Пример окна программы

3. Запуск алгоритма

   После задания всех параметров нажмите кнопку «Рассчитать», чтобы запустить алгоритм. Программа выполнит заданное количество итераций, обновляя значения и интерфейс для отображения текущих результатов (Рис. 3).

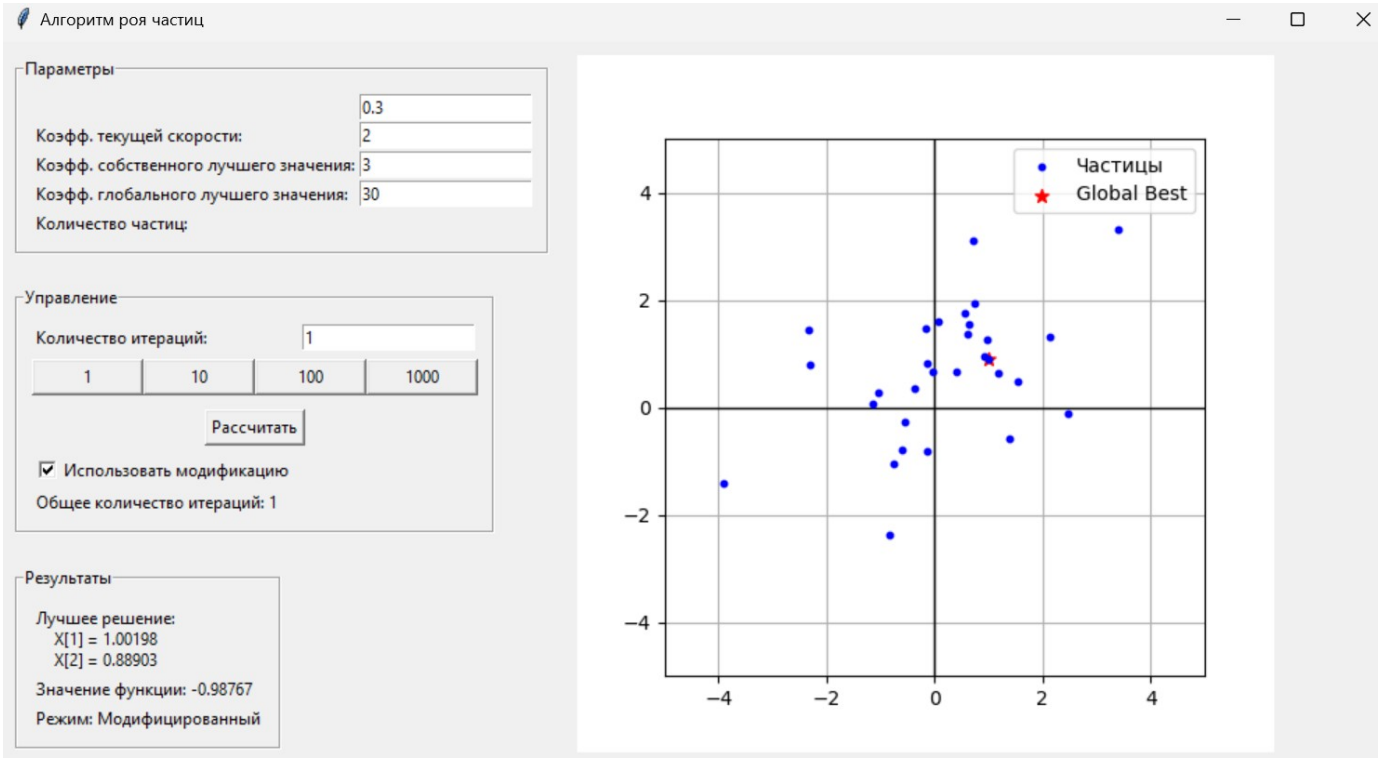


Рис. 3 Пример результатов программы

## Анализ

Тесты проводились при следующих настройках: коэффициент инерции — 0.3, коэффициент собственного лучшего значения — 2, коэффициент глобального лучшего значения — 3 и количество частиц — 30 и 60. Вы также можете задать число итераций для работы алгоритма и выбрать режим работы (стандартный или модифицированный). Ниже приведены результаты для различных сочетаний параметров и режимов, с подробным анализом полученных данных.

## Результаты с модификацией:

Таблица 2: тесты алгоритма роя частиц с модификацией с 30 частицами

| Количество поколений | Количество частиц | Лучшее решение | Значение функции |
|---|---|---|---|
| 1 | 30 | (1.21536, 1.71520) | -0.33936 |
| 5 | 30 | (0.98571, 1.06057) | -0.99572 |
| 50 | 30 | (1.00000, 1.00000) | -1.00000 |

Таблица 3: тесты алгоритма роя частиц с модификацией с 60 частицами

| Количество поколений | Количество частиц | Лучшее решение | Значение функции |
|---|---|---|---|
| 1 | 60 | (1.04219, 1.07057) | -0.98960 |
| 5 | 60 | (1.00224, 0.99826) | -0.99998 |
| 50 | 60 | (1.00000, 1.00000) | -1.00000 |

Таблица 4: тесты генетического алгоритма генома с модификацией

| Количество поколений | Количество хромосом | Лучшее решение | Значение функции |
|---|---|---|---|
| 10 | 32 | (-2.16506, -1.05824.) | 1.58285 |
| 100 | 32 | (-4.21906, -6.83680) | -0.02842 |
| 500 | 32 | (0.99996, 1.00003) | -1.00001 |

## Результаты без модификации:

Таблица 5: тесты алгоритма роя частиц без модификации с 30 частицами

| Количество поколений | Количество частиц | Лучшее решение | Значение функции |
|---|---|---|---|
| 1 | 30 | (-3.45774, -5.0000) | -0.24574 |
| 5 | 30 | (1.03907, 0.91839) | -0.98870 |
| 50 | 30 | (0.99821, 0.99907) | -0.99999 |

Таблица 6: тесты алгоритма роя частиц без модификацией с 60 частицами

| Количество поколений | Количество частиц | Лучшее решение | Значение функции |
|---|---|---|---|
| 1 | 60 | (-2.00942, 0.19517) | -0.43757 |
| 5 | 60 | (-2.01285, 0.38654) | -0.74032 |
| 50 | 60 | (-2.04278, 0.37135) | -1.00087 |

Таблица 7: тесты генетического алгоритма генома без модификации

| Количество поколений | Количество хромосом | Лучшее решение | Значение функции |
|---|---|---|---|
| 10 | 32 | (-1.86453, -10.59788) | 134.62237 |
| 100 | 32 | (-6.44511, -16.17656) | 47.64342 |
| 500 | 32 | (-6.05397, -13.32659) | 2.53160 |

## Эффективность модифицированного алгоритм роя частиц

Модифицированный алгоритм роя частиц показал более высокую сходимость к оптимальному значению целевой функции даже при малом числе итераций. Использование модификаций увеличивает точность и ускоряет процесс поиска глобального минимума, делая алгоритм роя частиц устойчивым и более подходящим для задач, где требуется высокая точность.

### Сравнение с алгоритм роя частиц без модификации

Без модификации алгоритм роя частиц демонстрирует медленное сближение с минимальным значением функции и показывает большую разбросанность значений на малых итерациях. Для достижения результатов, близких к глобальному минимуму, алгоритм роя частиц без модификации требует большего числа итераций, что подтверждает важность добавления модификаций для повышения стабильности и сходимости алгоритма.

### Сравнение с генетическим алгоритмом

Модифицированный генетический алгоритм также способен найти решения, приближенные к глобальному минимуму, но требует больше итераций для сходимости. Сравнение с алгоритм роя частиц показывает, что он с модификацией эффективнее для задач, где необходимы высокоскоростная сходимость и более точные результаты, так как генетическому алгоритму требуется больше поколений для аналогичной точности.

### Рекомендации по параметрам и модификациям

Коэффициент инерции 0.3, коэффициент личного лучшего значения 2 и глобального лучшего значения 3 показали хорошую сходимость для данной задачи, особенно при выборе модифицированного режима алгоритм роя частиц. Добавление модификаций для ускорения сходимости делает алгоритм роя частиц мощным инструментом для оптимизации сложных функций, позволяя получать стабильные результаты с высокой скоростью.

### Общие наблюдения

Модифицированный алгоритм роя частиц оказался более быстрым и стабильным, достигая глобального минимума с меньшим количеством итераций по сравнению с генетическим алгоритмом. Результаты показывают, что модифицированные алгоритмы более устойчивы к случайным отклонениям и имеют преимущества в сложных задачах оптимизации.

Благодаря улучшенной производительности, алгоритм роя частиц становится предпочтительным выбором для задач, требующих высокой точности и быстрого нахождения глобального минимума.

## Вывод

В ходе работы были исследованы особенности применения алгоритма роя частиц для оптимизации функции, а также влияние модификаций алгоритма на его эффективность. Модификация позволила ускорить процесс нахождения глобального минимума функции, увеличив скорость сходимости и точность результата.

Программа с графическим интерфейсом демонстрирует влияние различных параметров, таких как количество частиц, итераций и режим работы, на результаты алгоритма. Визуализация позволяет удобно наблюдать за процессом сходимости и оценивать производительность алгоритма, что делает приложение полезным для дальнейших исследований и тестирования оптимизационных задач.

## Источники


1. Particle swarm optimization // URL: https://en.wikipedia.org/wiki/Particle_swarm_optimization (дата обращения: 12.11.2024).
2. Implementing the Particle Swarm Optimization (PSO) Algorithm in Python // URL: https://medium.com/analytics-vidhya/implementing-particle-swarm-optimization-pso-algorithm-in-python-9efc2eb179a6 (дата обращения: 12.11.2024).
3. random— Generate pseudo-random numbers // URL: https://docs.python.org/3/library/random.html (дата обращения: 13.11.2024).
4. tkinter — Python interface to Tcl/Tk // URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 13.11.2024).
5. NumPy documentation // URL: https://numpy.org/doc/stable/index.html (дата обращения: 14.11.2024).
6. Matplotlib 3.9.2 documentation // URL: https://matplotlib.org/stable/index.html (дата обращения: 14.11.2024).