МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Поиск глобального минимума на криволинейной поверхности

Студентка гр. 1381 _____ Рымарь М.И.

Студентка гр. 1384 _____ Мухачёва П.Р.

Студент гр. 1384 _____ Белокобыльский И.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2023

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студентка Рымарь М.И. группы 1381

Студентка Мухачёва П.Р. группы 1384

Студент Белокобыльский И.В. группы 1384

Тема практики: Поиск глобального минимума на криволинейной поверхности

Задание на практику:

Задача для алгоритма роя частиц. Поиск глобального минимума на криволинейной поверхности задаваемой уравнением z=f(x,y). Формула должна вводиться через GUI. Для проверки корректности можно использовать функцию Растринга.

Сроки прохождения практики: 30.06.2023 – 13.07.2023

Дата сдачи отчета: 07.07.2023

Дата защиты отчета: 07.07.2023

Студентка	_____	Рымарь М.И.
Студентка	_____	Мухачёва П.Р.
Студент	_____	Белокобыльский И.В.
Руководитель	_____	Жангиров Т.Р.

АННОТАЦИЯ

Данная работа посвящена поиску глобального минимума на криволинейной поверхности, задаваемой уравнением z=f(x,y), с использованием алгоритма роя частиц. Проект реализован на языке программирования Python с использованием библиотек Tkinter, Matplotlib и NumPy для создания графического интерфейса пользователя (GUI) и эффективной работы с численными массивами.

SUMMARY

This project is devoted to finding a global minimum on a curved surface given by the equation z=f(x,y), using particle swarm algorithm. The project is implemented in the Python programming language using the Tkinter, Matplotlib and NumPy libraries to create a graphical user interface (GUI) and work with numerical arrays efficiently.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данного проекта - разработка программы для поиска глобального минимума на криволинейной поверхности.

Задачи проекта:

1. Реализовать GUI с помощью библиотеки Tkinter для удобного ввода формулы криволинейной поверхности, а также необходимых праметров, а также визуализацию криволинейной поверхности с помощью библиотеки
2. Разработать эффективный алгоритм поиска глобального минимума на криволинейной поверхности, основываясь на методе роя частиц, используя библиотеку NumPy
3. Визуализировать результаты на графике для удобного анализа и сравнения полученных минимумов при помощи библиотеки Matplotlib.
4. Протестировать полученный результат используя в качестве примера функцию Растринга для проверки корректности результатов поиска минимума.

1. ПЕРВАЯ ИТЕРАЦИЯ

1.1. Прототип GUI

Прототип GUI разработан с использованием Figma. Он состоит из двух окон. Первое окно является стандартным и предназначено для ввода функции, параметров и количества итераций. В этом окне также присутствуют кнопки для построения конечного графика, пошагового построения графика (перехода на следующий шаг) и сброса данных. Кроме того, в окне есть кнопка, при нажатии на которую открывается дополнительное окно с графиком изменения функции.

Дополнительное окно содержит график изменения функции в зависимости от количества итераций. Этот график позволяет анализировать и сравнивать полученные минимумы.

Прототип GUI разработан с учетом удобства использования и позволяет пользователю легко вводить необходимые параметры, а также анализировать результаты поиска глобального минимума на криволинейной поверхности.

Макеты стандартного и дополнительного окна представлены на рисунках 1 и 2, соответственно.

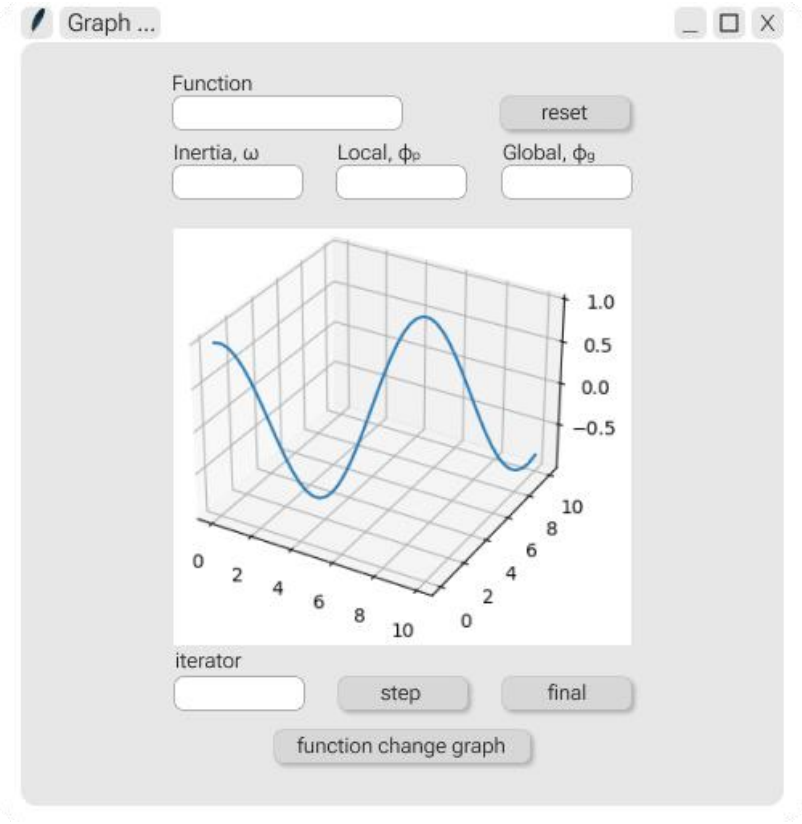

Рисунок 1 – Макет стандартного окна

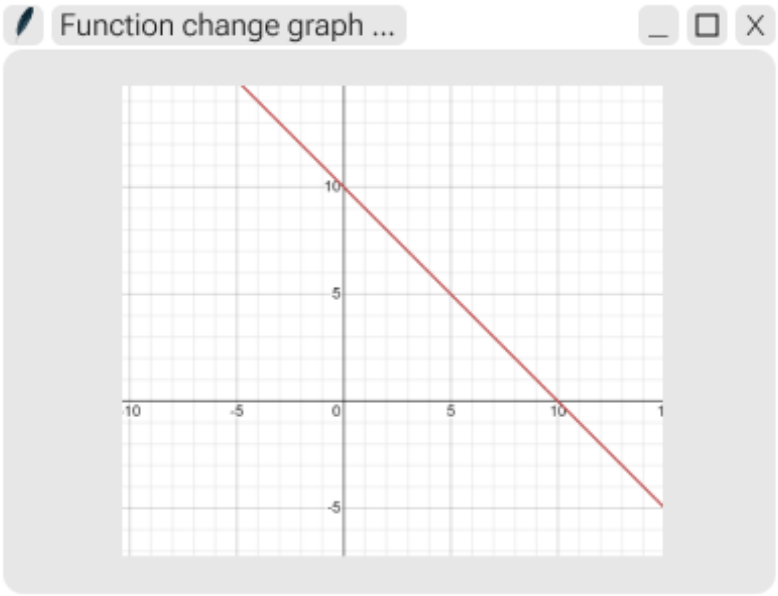

Рисунок 2 – Макет дополнительного окна

1.2. Выбор модификаций, метрик, настраиваемых параметров

Суть алгоритма заключается в изменении положения частиц в пространстве, задаваемой функцией f(x, y). Сама функция выглядит следующим образом:

$$v_i \leftarrow \omega v_i + \phi_p r_p \cdot (p_i - x_i) + \phi_g r_g \cdot (g - x_i)$$

Где ω – инерция, ϕ_p – коэффициент локального оптимума, ϕ_g – коэффициент глобального оптимума, v_i – скорость частицы, r_p, r_g – случайные векторы из равномерного распределения на отрезке [0, 1], p_i – локальный оптимум для i-й частицы, g – глобальный оптимум, x_i – положение частицы в пространстве. В качестве операции умножения имеется в виду покомпонентное умножение векторов.

Первые три из описанных выше параметров доступны для ввода пользователем. Также, помимо этого, пользователь может повлиять на количество итераций алгоритма, так как сам метод роя частиц является стохастическим, следовательно, не имеет единого для всех алгоритмов критерия останова.

Было проведено сравнение двух модификаций алгоритма – с инерцией и без нее. На тестовых данных вариант с инерцией показал себя лучше, поэтому

было принято решение оставить его. К тому же, чтобы убрать влияние инерции на вычисления, достаточно приравнять ее к 1.

1.3. Стек технологий и сторонние библиотеки

В качестве языка программирования был выбран Python, так как на нем присутствует наиболее удобные на взгляд участников бригады решения для визуализации – библиотеки Matplotlib для составления графиков и Tkinter для создания GUI.

Так как язык Python предоставляет не самый эффективный способ работы с числовыми массивами, в дополнение к представленному выше списку была взята библиотека NumPy.

1.4. Распределение ролей в бригаде

Мухачёва Полина – Написание самого алгоритма роя частиц;

Рымарь Мария – Разработка пользовательского интерфейса, анализ проделанной всей бригадой работы;

Белокобыльский Илья – Проектирование архитектуры приложения, выбор стека технологий, тестирование.

2. ВТОРАЯ ИТЕРАЦИЯ

2.1. Частичная реализация GUI

Графический интерфейс пользователя частично реализован с помощью библиотеки Tkinter.

Создается главное окно с заданными размерами и заголовком. Окно нерасширяемое по размерам, то есть параметры метода resizable установлены в False.

Затем создаются различные элементы интерфейса (виджеты), такие как надписи (Label), поля ввода (Entry), кнопки (Button), ползунки (Scale) и холст (Canvas). Элементы размещаются на окне с помощью методов place() или pack().

Также определены функции, которые будут вызываться при нажатии на кнопки. Например, функция graph_window() открывает новое окно, в котором появляется график изменения функции, при нажатии на кнопку *function change graph*.

В конце вызывается метод mainloop(), который запускает цикл обработки событий окна и позволяет взаимодействовать с пользователем.

Взаимодействие с моделью происходит при помощи обработчиков событий. Например, как было указано выше, функция graph_window() является одним из обработчиков событий для кнопки *function change graph*. Для остальных кнопок тоже были сделаны обработчики событий (для кнопки *reset* – функция reset(), *step* – step(), *final* – final()). Также значения полей ввода и ползунков можно использовать для передачи параметров в модель.

2.2. Хранение данных

Классы Particle, Swarm и System.

Класс Частицы является классом, необходимым только для хранения информации о каждой частице (текущие положение и скорость, лучшие положение и значение).

Класс Роя хранит в своих полях данные: количество частиц, параметры, границы исследуемой области и текущую функцию. Так же отдельным полем

хранится список объектов Типа частица, который генерируется внутри класса роя методом объекта класса роя. Кроме хранения информации класс роя частиц содержит методы для работы алгоритма на одной итерации.

Класс Системы содержит в себе все те же поля, что и класс роя, но им заданы параметры по умолчанию. Кроме этих полей, в классе Системы существует поле, хранящее информацию, необходимую для критерия останова — долю от общего числа частиц, чья скорость близка к нулю. Класс Системы, кроме полей с данными, содержит в себе методы, необходимые для выполнения всех заданных итераций и выполнения критерия останова.

2.3. Модификация алгоритма, критерий останова

Для решения задачи был выбран алгоритм роя с параметром инерции. Это помогает точнее определять минимум текущей функции, т. к. при умножении на параметр инерции, значение которого лежит в интервале (0;1], шаги, совершаемые частицами за 1 итерацию, становятся меньше.

Критерии останова:

1. Работа алгоритма считается завершённой, когда скорость некоторого заданного количества частиц близка к нулю. За значение «близкое к нулю» выбрано число равное длине самой короткой границы / 1000. Если значение скорости заданного количества частиц <= этого значения, то срабатывает критерий останова.

2. Выход из алгоритма по истечению количества итераций, введённых пользователем.

2.4. Алгоритм роя частиц

Для работы алгоритма было создано три класса: Particle, Swarm и System. Классы Swarm и System как предназначены для хранения данных, так и содержат методы, в которых реализуется сам алгоритм. Класс Particle служит только для хранения данных.

Алгоритм выполняется по следующим шагам:

1. Значения координат частиц и их скоростей генерируются как векторы, принадлежащие равномерному распределению на заданных промежутках. Значение лучшей координаты и лучшего значения каждой частицы инициализируется от единственного сгенерированного значения. Лучшее значение среди всех частиц выбирается из лучших положений частиц и тоже инициализируется.

2. Для каждой частицы пересчитывается значение скорости при помощи значений скорости на предыдущем шаге, лучших локального и глобального значений и заданных параметров.

3. Для каждой частицы пересчитывается координата её положения и считается значение функции в этой точке.

4. Для каждой частицы проверяется, является ли данное значение лучшим (наименьшим) относительно прошлого лучшего значения данной частицы. Так же обновляется значение глобального минимума, если появится значение, меньшее прошлого лучшего значения.

5. Проверяется критерий останова. Если он выполнен, флаг, сигнализирующий это, выставляется в 1, производится выход из функции по условию флаг = 1.

Если критерий останова не выполнен и данная итерация не является последней, производится возврат к п.2.

Если критерий останова не выполнен и данная итерация последняя, переход к п.6.

6. Выход из функции по критерию количества итераций, если ранее не был произведён выход по критерию близости скорости к нулю.

2.5. Тестирование

Тестирование алгоритма можно произвести на объектах классов Plane и Rastring. Первые являются плоскостями с задаваемыми коэффициентами. Вторые являются функцией Растринга.

3. ФИНАЛЬНАЯ ИТЕРАЦИЯ

На заключительном этапе проекта были внесены следующие корректировки (все они связаны с пользовательским интерфейсом):

- Графики представлены в одном окне, убрали кнопку, которая открывала новое окно с графиком изменения функции;
- Добавлены подписи к графикам;
- Изменён ввод функции, то есть в поле ввода *function* вводится только уравнение поверхности, для интервалов по осям абсцисс и ординат добавлены по два новых поля;
- В связи с изменениями, указанными выше, увеличены размеры окна.

Новая версия GUI представлена на рисунке 3.

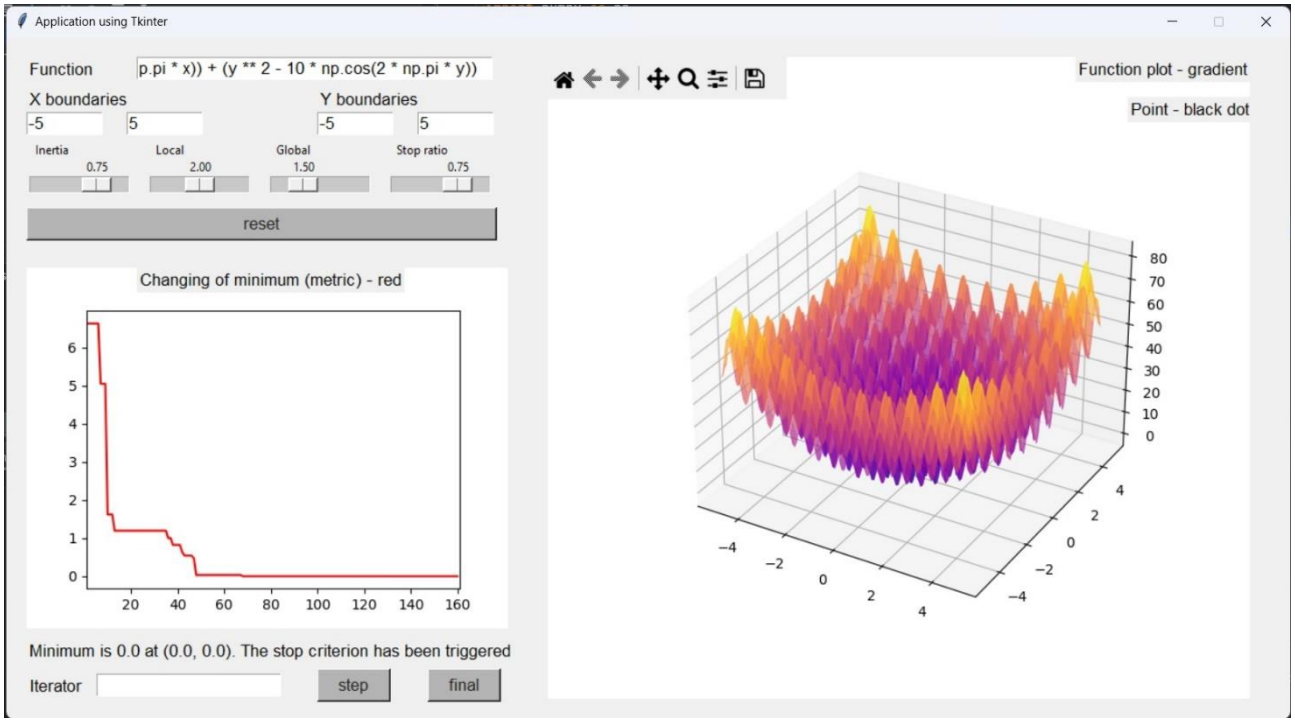

Рисунок 3 – Финальная версия GUI

ЗАКЛЮЧЕНИЕ

В финальной итерации проекта был доработан пользовательский интерфейс, внесены корректировки в графики, исправлены недочёты.

В ходе выполнения задания учебной практики команда изучила алгоритм роя частиц; познакомилась и поработала со средой для дизайна Figma, графическим интерфейсом пользователя и библиотекой Tkinter, построением графиков с использованием Matplotlib, хранением данных с помощью NumPy. Все технологии были применены для реализации поиска глобального минимума на криволинейной поверхности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Jenyay.net. (2021). Particle Swarm Optimization (Метод роя частиц) [Online]. Available: https://jenyay.net/Programming/ParticleSwarm

2. Wikipedia.org. (2021). Метод роя частиц [Online]. Available: https://ru.wikipedia.org/wiki/Метод_роя_частиц

3. Metanit.com. (2023). Руководство по Tkinter [Online]. Available: https://metanit.com/python/tkinter/