

Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Отчет по лабораторной работе 1
по дисциплине «**Методы оптимизации в машинном обучении**»

Авторы: Багаутдинов Искандер Ильгизович М3237

Пан Артём Олегович М3237

Преподаватель: Андреев Юрий Александрович

Цель работы:

Реализовать и исследовать эффективность метода градиентного спуска с постоянным шагом, метода одномерного поиска и градиентный спуск на его основе и метода Нелдера-Мида реализованного в библиотеке `scipy.optimize`

Задачи работы:

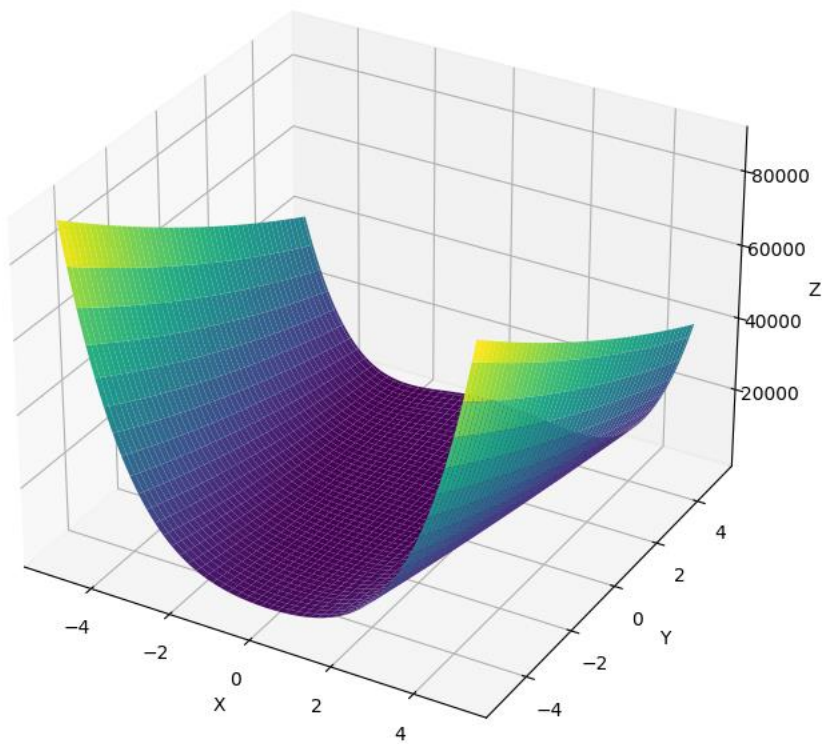
- Выбрать функции, на которых эффективность методов будет явно различаться
- Исследовать работу методов в зависимости от начальной точки
- Проиллюстрировать примеры

Задание 1: Выбор функций

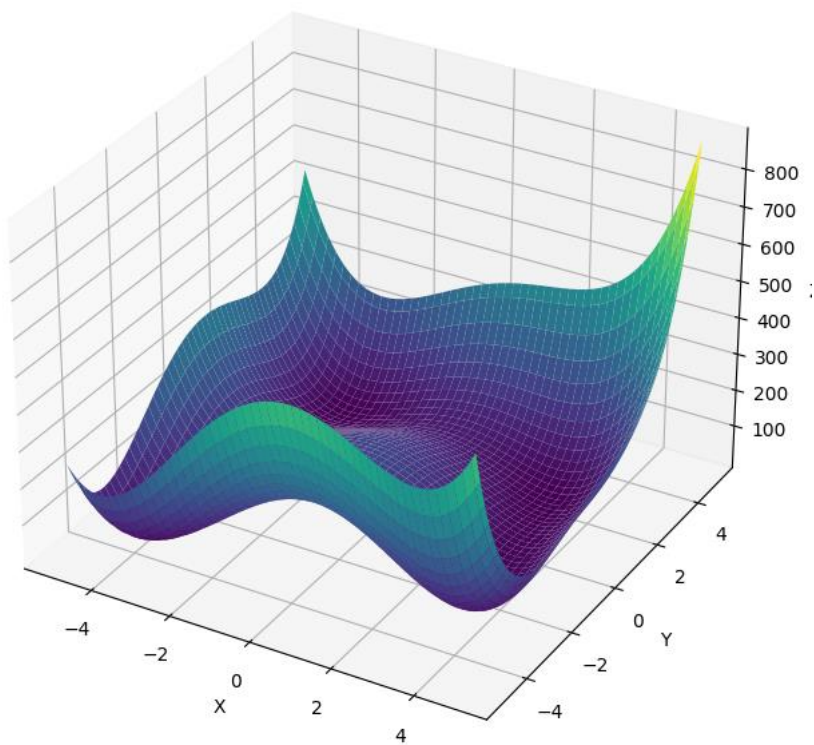
Для проведения исследования эффективности мной были выбраны две стандартные тестовые функции в оптимизации, а также одна простая функция без локальных минимумов. Функция Розенброка- функция вида $(1-x)^2 + 100*(y-x^2)^2$ и функция Химмельблау- $(x^2 + y - 11)^2 + (x + y^2 - 7)^2$. Две данные функции имеют нетривиальную поверхность и достаточное количество точек в которых неэффективные методы могут застрять/не найти точку минимума. Так же, второй функцией была выбрана $x^2 + y^2$, она выпукла (не имеет локального минимума, любой локальный минимум будет глобальным) и гладка, что позволяет неэффективному методу оптимизации все равно найти ее минимум.

Визуализация функций:

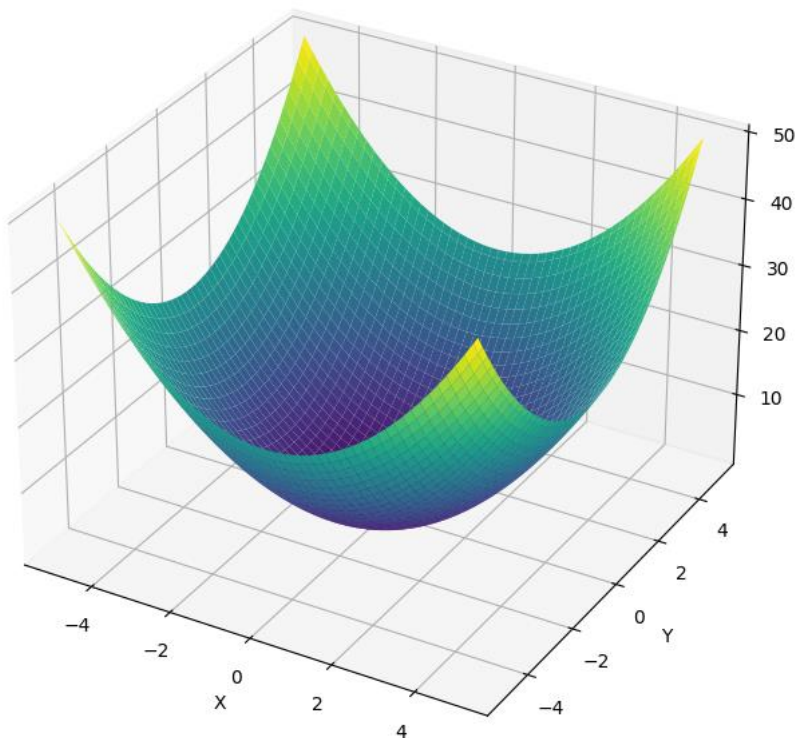
Rosenbrock Function



Himmelblau Function



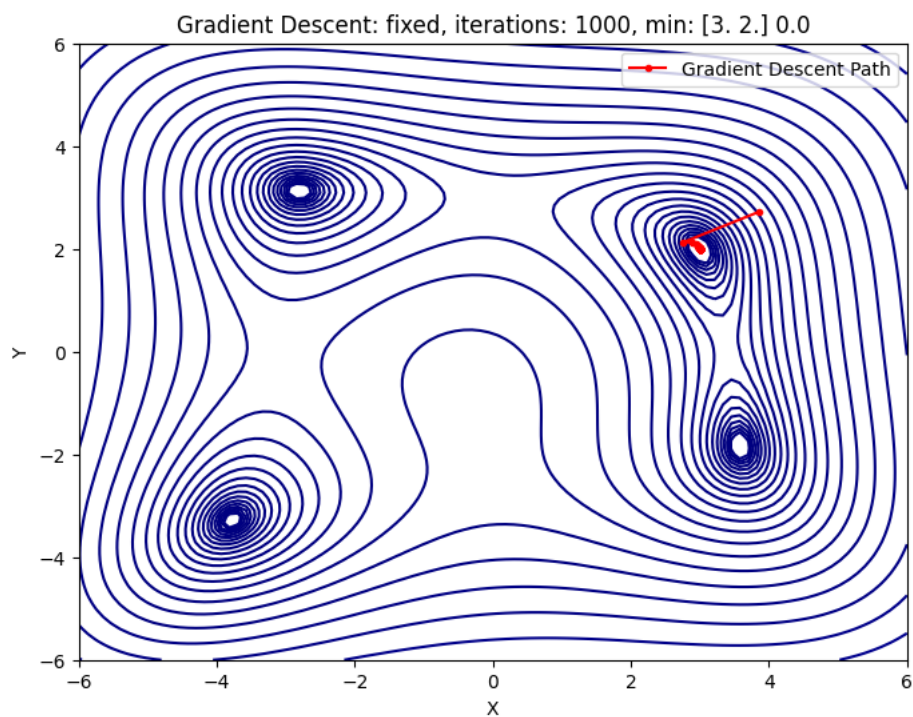
Spherical Function



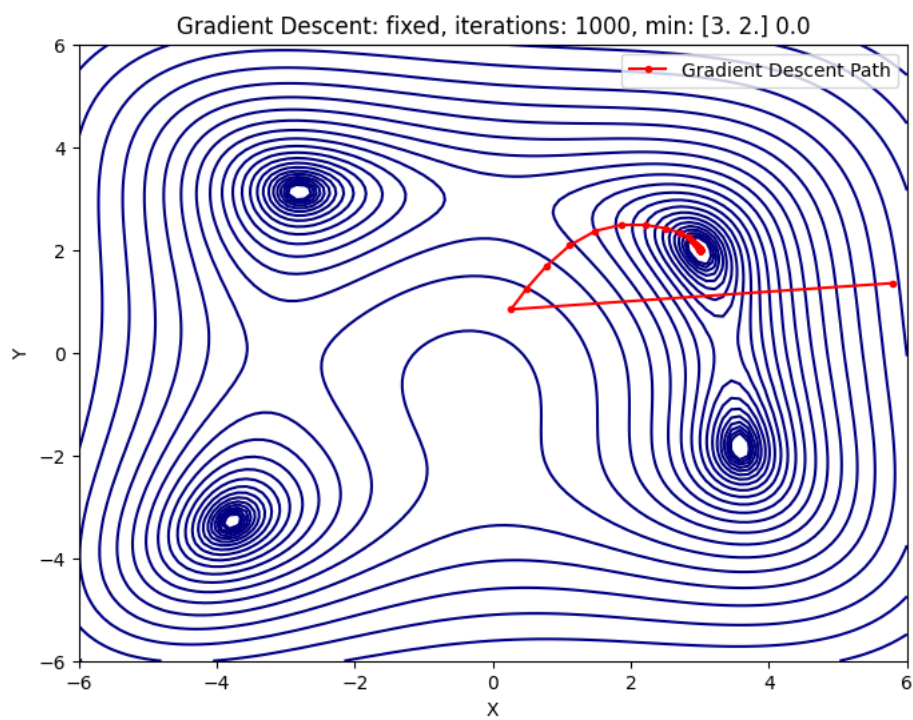
Пункт 1. Метод градиентного спуска с постоянным шагом.

С расчётом на дальнейшее обобщение постоянный шаг был реализован с помощью функции *fixed_search*, для сохранения общей сигнатуры она принимает, как и следующие методы какие-то параметры, но возвращает только фиксированный learning rate представленный в виде константы. Учитывая это обобщение, в функцию градиентного спуска *grad_step* передается сама функция, функция, определяющая метод вычисления *lg*, количество итераций спуска, начальное значение и *tolerance* для прерывания работы алгоритма в случае незначительности изменения значения из-за маленького значения градиента, что позволяет нам понять сколько итераций хватило для прихода к минимуму. Далее просто в цикле считается learning rate, градиент и меняем значения текущего *x*.

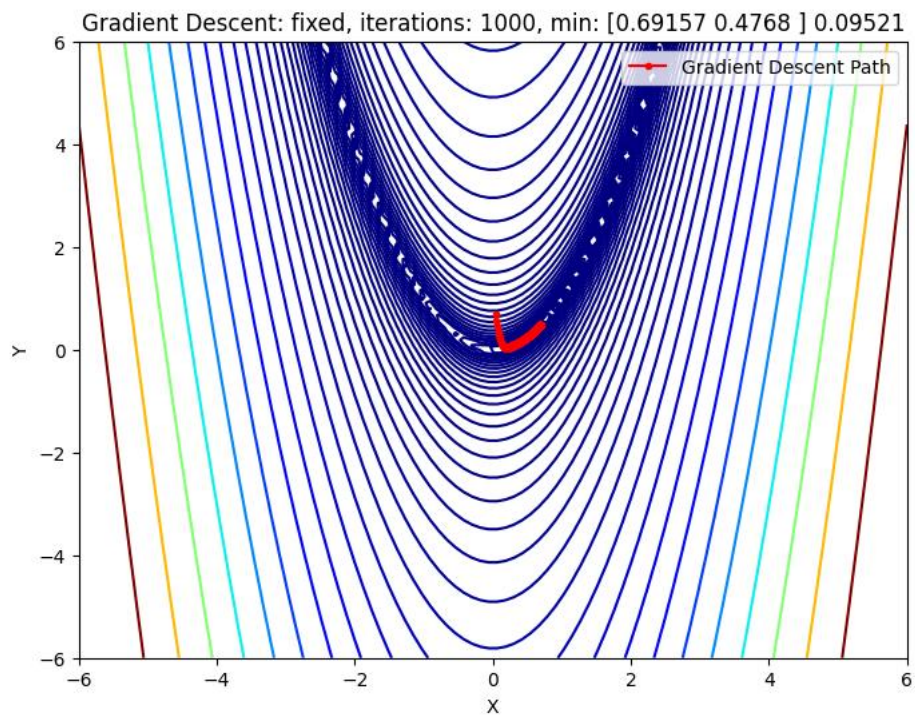
Применение данного метода вычисления learning rate:



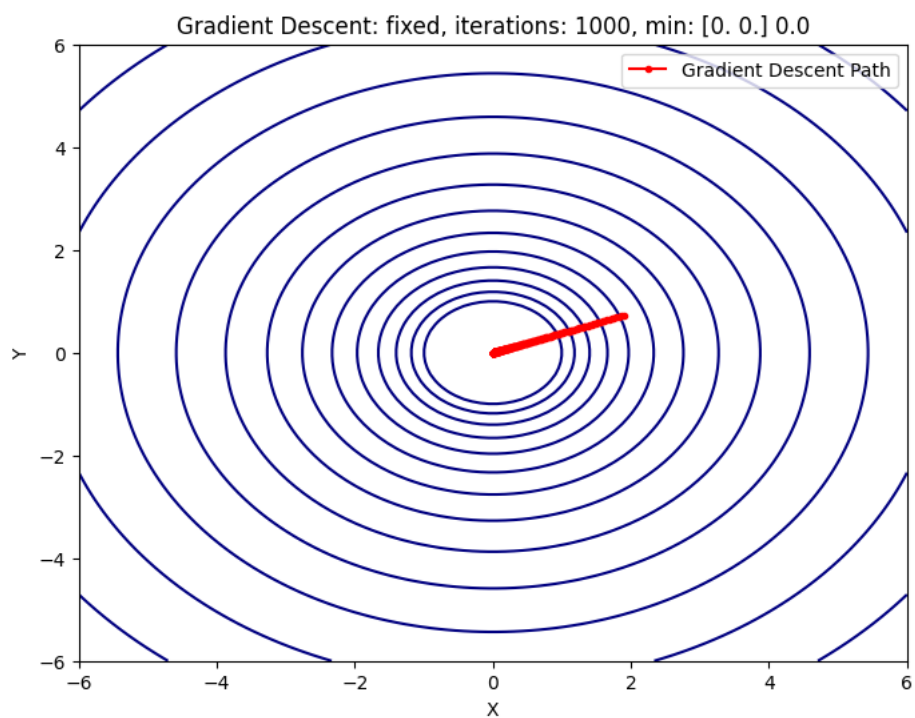
Функция Химмельблау, начальная точка [3.86030667 2.72365164], сошелся к минимуму за выделенное число итераций, предварительно не прервался



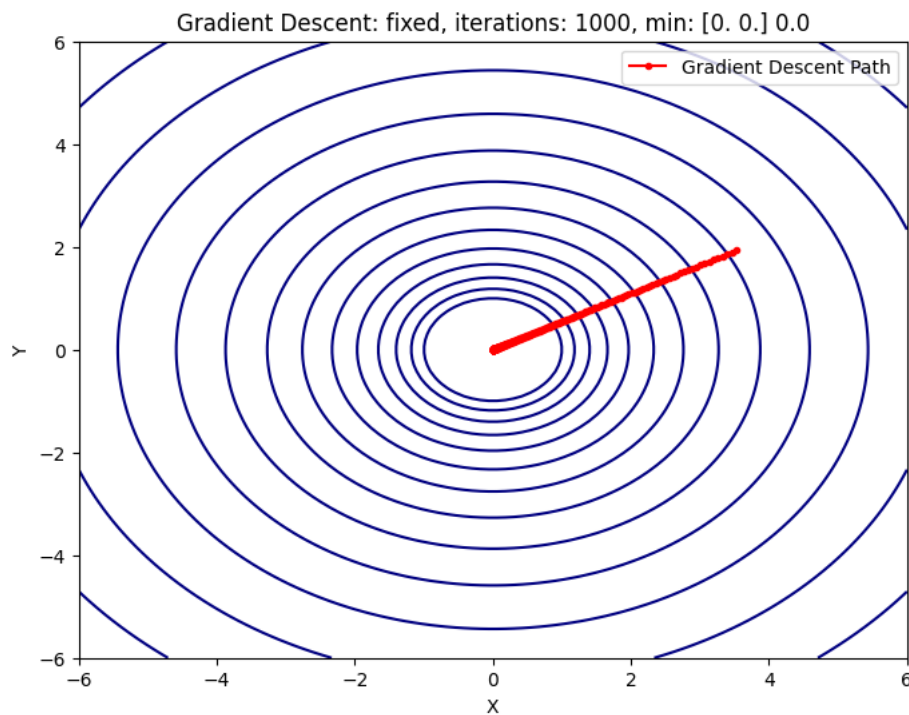
Функция Химмельблау, начальная точка [5.78442868 1.35173453], сошелся к минимуму за выделенное число итераций, предварительно не прервался



Функция Розенброка, начальная точка [0.0372064, 0.69903833], не сошелся к минимуму за выделенное число итераций, предварительно не прервался. Смог посчитаться только при очень маленьком значении learning rate(0.001)



Функция сферы, начальная точка [1.90214151 0.7253265], сошелся к минимуму за выделенное число итераций, предварительно не прервался



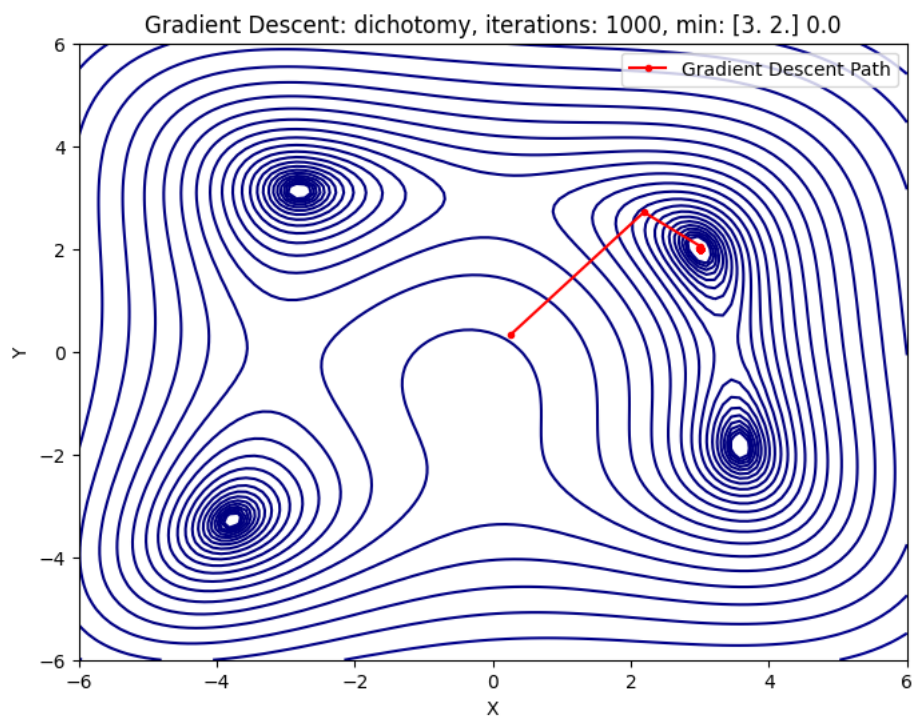
Функция сферы, начальная точка [3.53647661 1.93623233], сошелся к минимуму за выделенное число итераций, предварительно не прервался

Заметим, что алгоритм может при различных параметрах находить минимум функции не перепрыгивая его, однако показывает свою неэффективность на простых функциях очень медленно приближаясь к минимуму. Это можно заметить на последней функции.

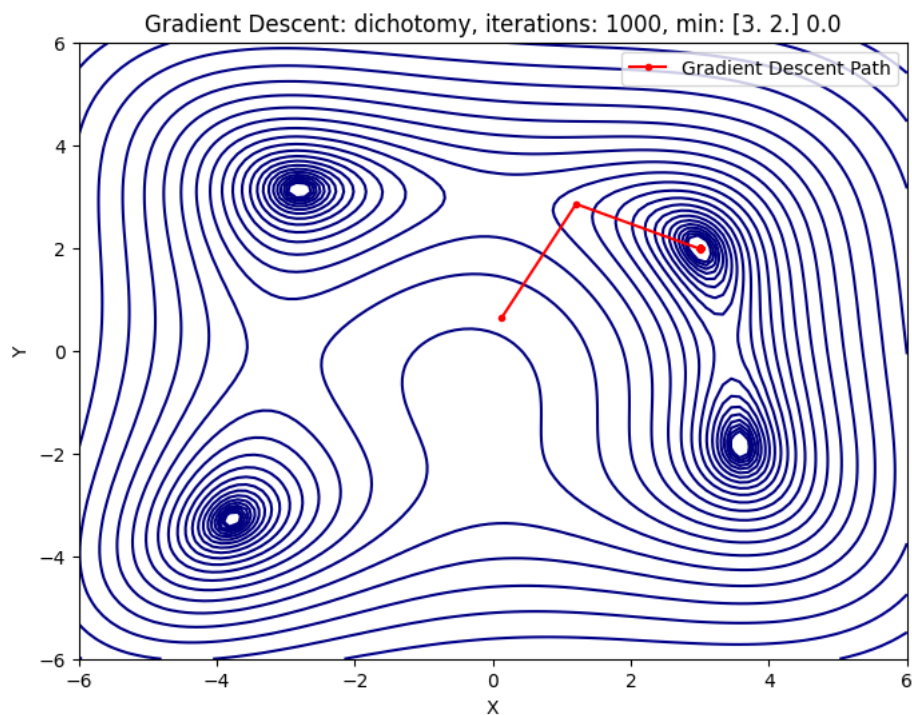
Пункт 2. Метод одномерного поиска и градиентный спуск на его основе.

После обобщения основной части кода реализующий градиентный спуск достаточно рассмотреть функцию вычисляющую learning rate. Рассмотрим метод дихотомии. Он ищет корень на отрезке, где в начальной и конечной точках функция принимает значения разных знаков. При условии непрерывности исходной функции, $[a_{k+1}, b_{k+1}] = \{[a_k, (a_k+b_k)/2]$ при $f((a_k+b_k)/2) > 0$, $[(a_k+b_k)/2, b_k]$ при $f((a_k+b_k)/2) \leq 0$. Так мы находим подходящий отрезок по условиям данного метода и находим на нем локальный минимум.

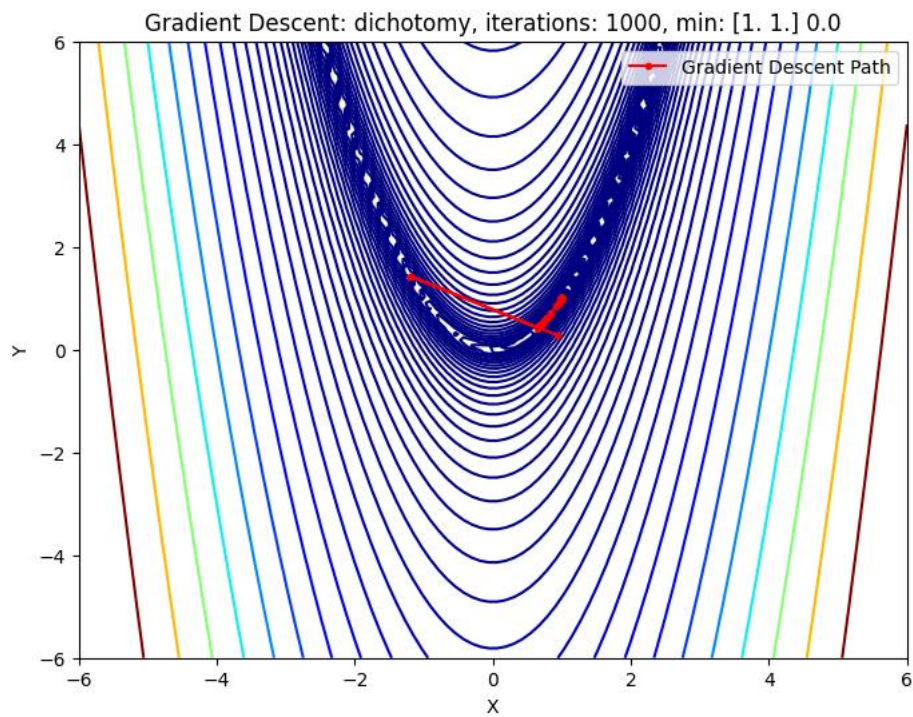
Применение данного метода поиска:



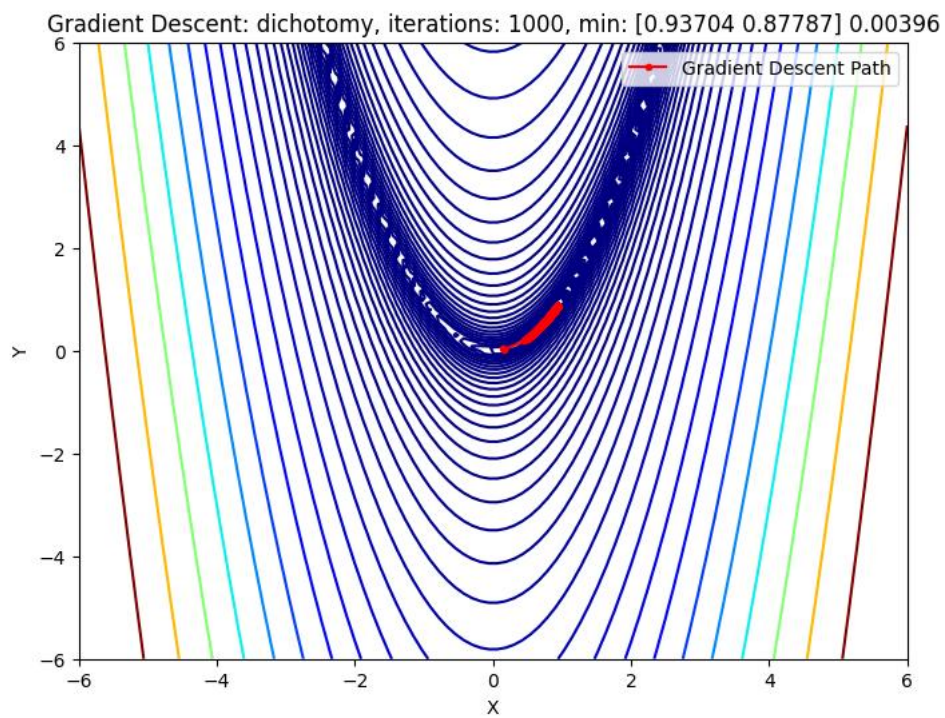
Функция Химмельблау, начальная точка $[0.26324866 \ 0.33777349]$, сошелся к минимуму за выделенное число итераций, предварительно не прервался



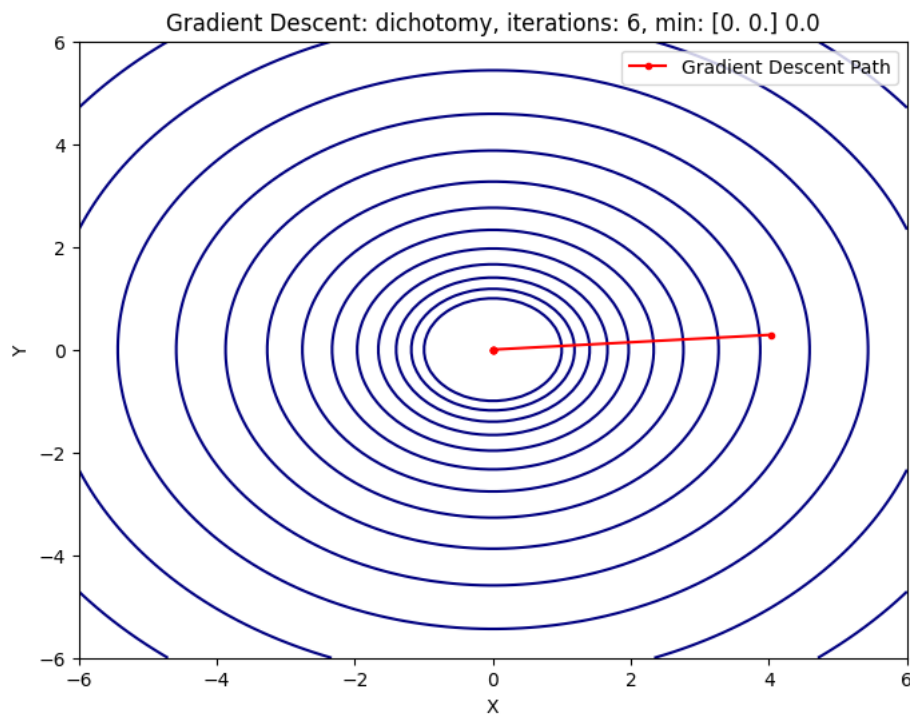
Функция Химмельблау, начальная точка $[0.1228098 \ 0.6310801]$, сошелся к минимуму за выделенное число итераций, предварительно не прервался



Функция Розенброка, начальная точка [0.92657944 0.28404483], сошелся к минимуму за выделенное число итераций, предварительно не прервался



Функция Розенброка, начальная точка [0.14510266 0.0513785], к минимуму не сошелся



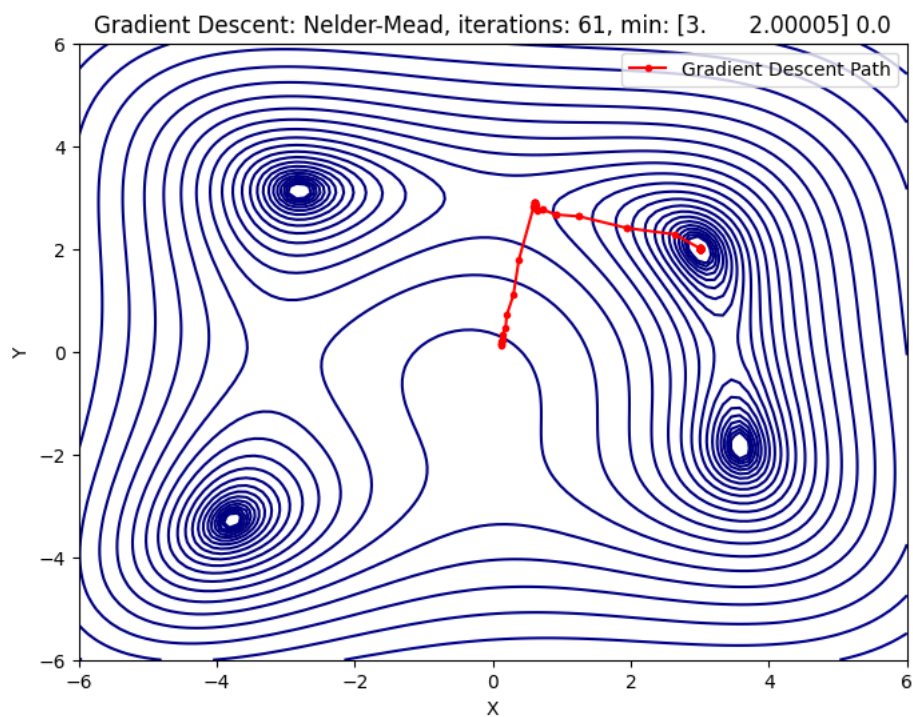
Функция сферы, начальная точка $[4.02796054 \ 0.28822934]$, сошелся к минимуму, прервался.

Заметим, что данный метод одномерного поиска так же не справляется с функцией Розенброка как и спуск с фиксированным шагом, однако показывает большое преимущество на простых функциях, где из-за способности менять шаг алгоритм завершился всего через 7 итераций.

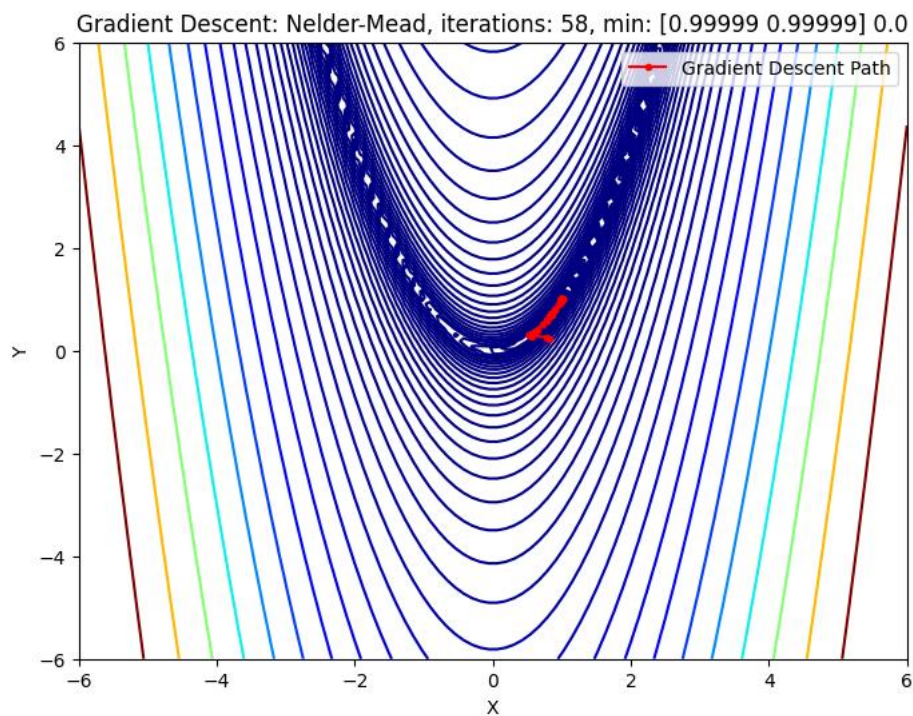
Пункт 3. Метод Нелдера-Мида

Метод оптимизации, не основанный на производной, а на разбиениях на треугольники, использую его реализацию из модуля `scipy`.

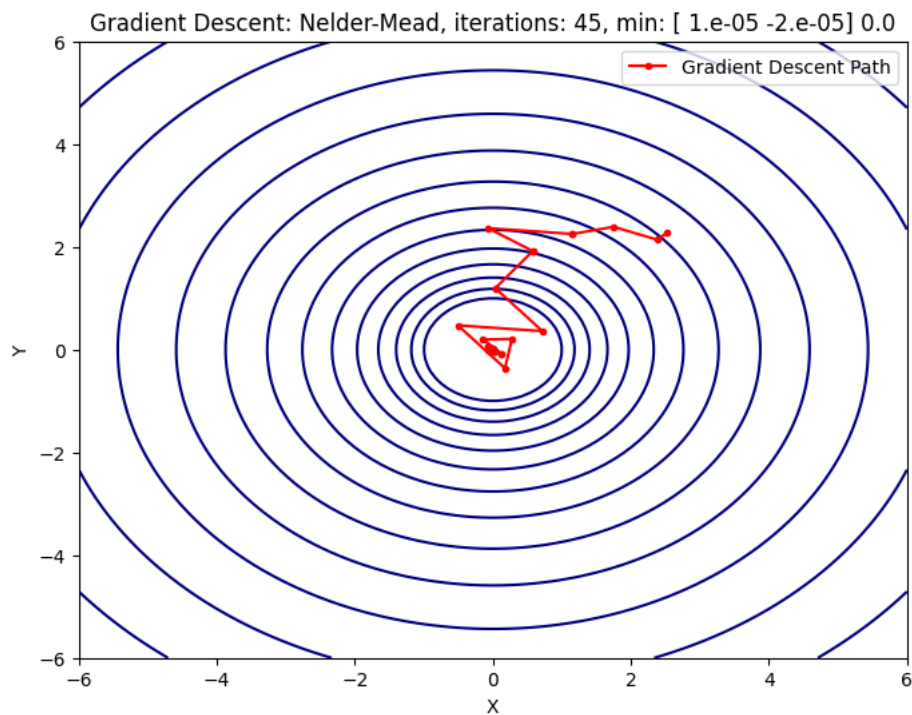
Сравним на функциях:



Функция Химмельблау, начальная точка [0.11243966 0.12395764], сошелся к минимуму, прервался.



Функция Розенброка, начальная точка [0.91649243 0.2197488], сошелся к минимуму, прервался.



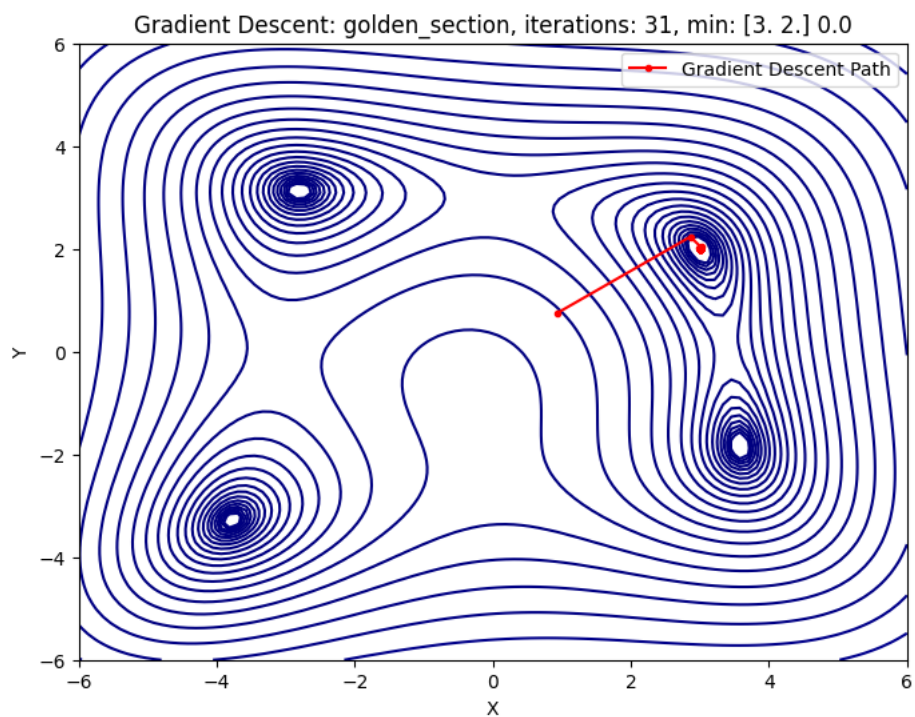
Функция сферы, начальная точка [2.79926617 2.11253293], сошелся к минимуму, прервался.

Заметим, что данный метод показывает превосходную точность, одновременно показывая достаточность малого количества итераций.

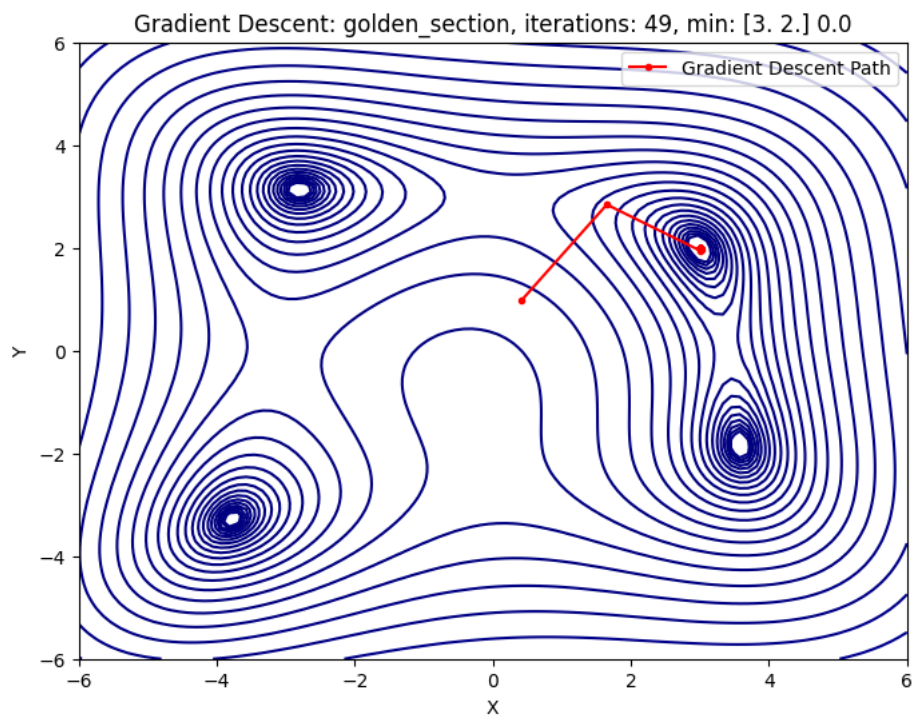
Дополнительное задание 1. Сравнение еще одного одномерного метода поиска

Рассмотрим метод поиска с помощью золотого сечения. Этот метод так же итеративно сужает границы отрезка пока его точность нас не устроит. Делается разбиения с помощью прибавления и вычитания $(r-l)/(\phi+1)$ (отрезки разбиваются симметрично), дальше рассматривается два случая в зависимости от того какое значение функции будет больше, благодаря чему на каждой итерации отрезок сужается, когда он сузился достаточно возвращается его середина.

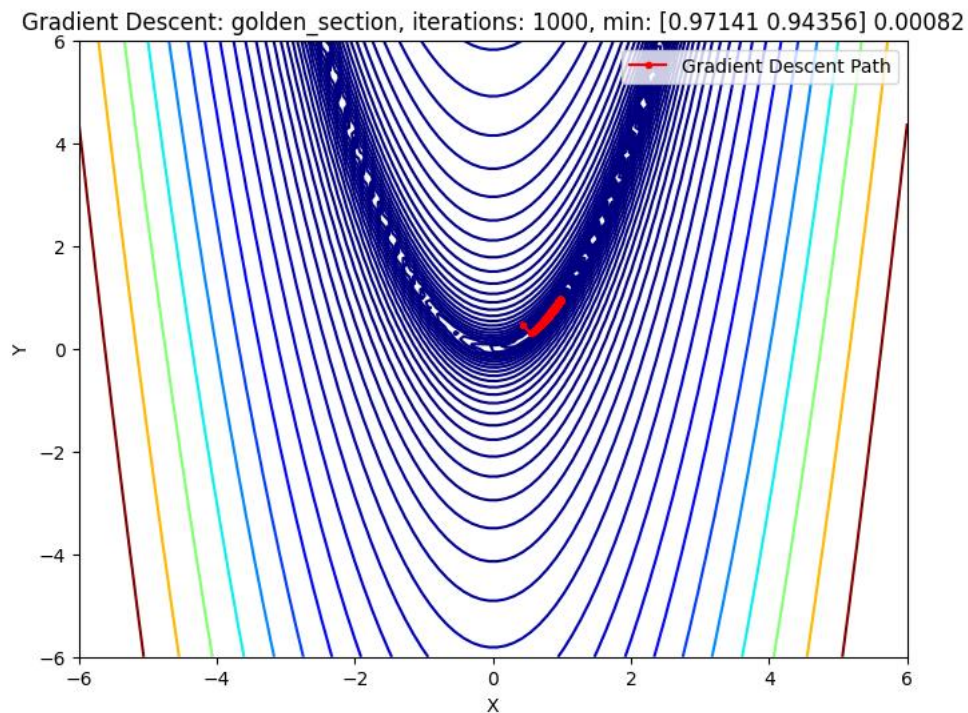
Сравним на функциях:



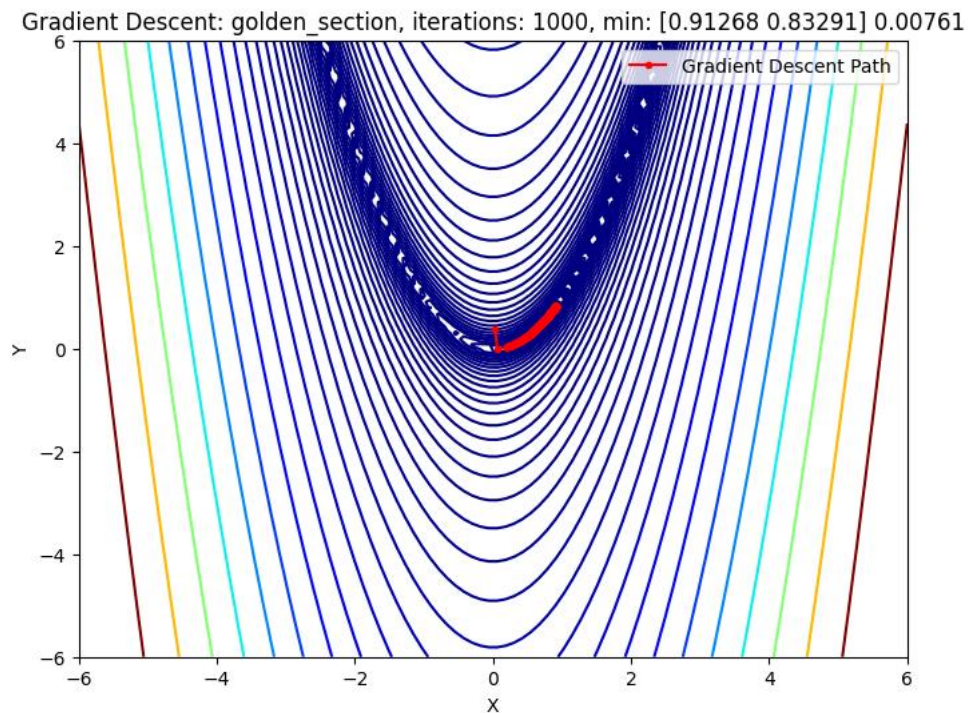
Функция Химмельблау, начальная точка $[0.92417768 \ 0.74481266]$, сошелся к минимуму за выделенное число итераций, предварительно не прервался



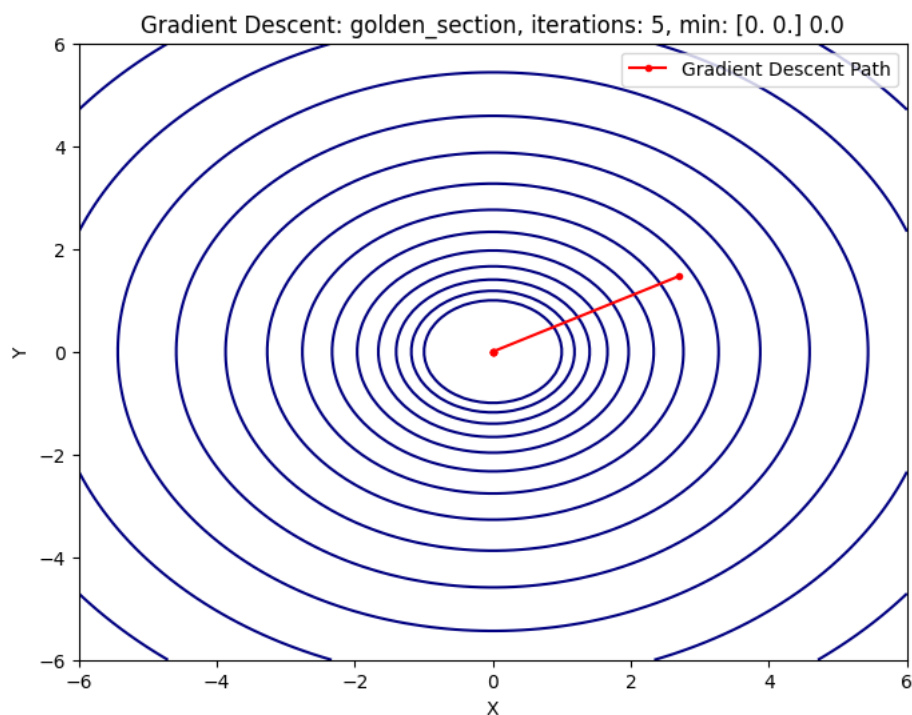
Функция Химмельблау, начальная точка [0.41808488 0.98353343], сошелся к минимуму за выделенное число итераций, предварительно не прервался



Функция Розенброка, начальная точка [0.4228748 0.45119854], не сошелся.



Функция Розенброка, начальная точка [0.02915941 0.3806357], не сошелся.



Функция сферы, начальная точка [2.69403073 1.46730378], сошелся к минимуму, прервался.

Заметим, что количество итераций для простой функции так же значительно сократилось относительно спуска с фиксированным learning rate, однако для функции Розенброка данный метод, так же как и дихотомия не сходится даже при достаточно большом количестве итераций.

Таблица для функции Розенброка

Начальная точка	Точность	Постоянный шаг. Результат	Дихотомия. Результат	Метод Нелдера-Мида. Результат	Метод золотого сечения. Результат	Количество итераций. Постоянный шаг	Количество итераций. Дихотомия	Количество итераций. Метод Нелдера-Мида	Количество итераций. Метод золотого сечения
[0.25992047, 0.22968035]	1e-3	0.39	0.05	0	0.05	50	176	65	178

[0.0507 1181, 0.89095 865]	1e-8	0.1033	0.0085	0	0.085	1000	1000	80	1000
[0.1930 7619, 0.06130 238]	1e-18	0.09	0.008	0	0.001	1000	1000	72	1000

Таблица для функции Химельблау

Началь ная точка	Точн ость	Постоя нный шаг. Результ ат	Дихото мия. Результ ат	Мето д Нелд ера- Мида . Р езул ьтат	Мето д золот ого сечен ия. Резул ьтат	Количе ство итерац ий. Постоя нный шаг	Колич ество итерац ий. Дихото мия	Колич ество итерац ий. Метод Нелде ра- Мида	Количеств о итераций. Метод золотого сечения
[0.2599 2047, 0.22968 035]	1e-3	0.01	0.0000 2	0.0	0.000 02	103	7	55	7
[0.0329 4912, 0.11885 609]	1e-8	0	0	0	0	593	22	69	22
[0.1930 7619, 0.06130 238]	1e-18	0	0	0	0	1000	1000	64	39

Таблица для функции сферы

Начальная точка	Точность	Постоянный шаг. Результат	Дихотомия. Результат	Метод Нелдера-Мида. Результат	Метод золотого сечения. Результат	Количество итераций. Постоянный шаг	Количество итераций. Дихотомия	Количество итераций. Метод Нелдера-Мида	Количество итераций. Метод золотого сечения
[0.25992047, 0.22968035]	1e-3	0.12	0.0	0.0	0.0	0	1	37	1
[0.03294912, 0.11885609]	1e-8	0.0028	0	0	0	1000	2	35	2
[0.19307619, 0.06130238]	1e-18	0.0008	0	0	0	1000	3	36	3

Вывод:

Методы золотого сечения и дихотомии делают значительно меньше итераций чем метод фиксированного спуска так как их шаги длиннее. Точность так же сильно влияет на количество итераций, результат при меньшей точности не сильно отличается чем от при большей, но при этом количество итераций становится кратно меньше.

Лучше всего себя показал метод Нелдера-Мида. Его ломаная напоминает ломанную при стохастическом градиентном спуске, потому что двигается все время в разные стороны. У этого метода значительно меньше итераций, при этом точность лучше, потому что почти всегда алгоритм сходился.

На простых функциях метод с фиксированным шагом сильно уступает из-за большого количества итераций, однако для сложных количество итераций примерно совпадает.