

Кафедра КТ

**Отчет по лабораторной работе №1:
“Методы нулевого и первого порядка”
по курсу: Методы Оптимизации**

Работу выполнили:

Атаев А. Г.
Беспалов А. М.
Тиунов И. В.

Преподаватель:

Андреев Юрий Александрович

Основное задание

Задача: реализовать и исследовать методы:

- Метод градиентного спуска с различными стратегиями выбора шага
- Любой метод одномерного поиска и градиентный спуск на его основе
- Изучение возможностей библиотеки `scipy.optimize` и сравнение ее эффективности и эффективности реализованных самостоятельно методов

Ссылка на реализованные методы (код)

1. Исследование метода градиентного спуска с различными стратегиями выбора шага

Описание методов:

1. Метод градиентного спуска с различными стратегиями выбора шага

На вход методу градиентного спуска дается функция f , которая является непрерывной и дифференцируемой, а также начальная точка x_0 - первое приближение минимума

В качестве результата работы метода ожидается значение x , соответствующее некому локальному минимуму. Предполагается, что такой минимум действительно существует и может быть достигнут

Градиентный спуск имеет вид итеративного процесса:

$$\bullet \quad x_{k+1} = x_k - h(k)\nabla f(x_k)$$

, который на основании текущей точки x_k и градиента в ней $\nabla f(x_k)$ строит точку x_{k+1} . Таким образом, на каждой итерации совершается шаг в направлении антиградиента, а значит в сторону быстрейшего убывания. Для этого у него есть функция $h(k)$, которая по номеру итерации возвращает число. Эта функция называется стратегией выбора шага.

Алгоритм заканчивает работу по достижении критерия остановки. Возможны разные критерии остановки, например, оценки на разность норм аргументов глобального минимума и найденной точки, оценки на разность норм значений, а также оценок на разности точек, рассмотренных на двух последних итерациях. В данной работе, если не сказано иного (иногда интересно

рассмотреть поведение при отличном критерии остановки), в качестве критерия остановки был выбран критерий:

- $\|x_{k+1} - x_k\| \leq \varepsilon, \varepsilon = 0.001,$

который прекращает работу, если разность норм аргументов соседних точек не превосходит ε .

Также реализован предохранитель, который позволяет ограничить выполнение в случае, если критерий остановки никогда не будет выполнен. В данном случае предохранитель - это ограничение на максимальное число итераций процесса, равное 500 - после 500 итераций исполнение принудительно завершается, и возвращается точка на последней итерации.

Для анализа поведения метода были выбраны следующие функции:

- Квадратичная функция с хорошей обусловленностью

$$f(x, y) = x^2 + 3y^2$$

- Квадратичная функция с плохой обусловленностью

$$f(x, y) = 0.1x^2 + 3y^2$$

Эти функции позволяют рассмотреть два фундаментально различных вида функций от двух переменных, которые имеют минимум, а все прочие имеющие минимум функции можно правильным выбором координат свести к похожему виду.

В рамках задания были рассмотрены следующие стратегии выбора шага:

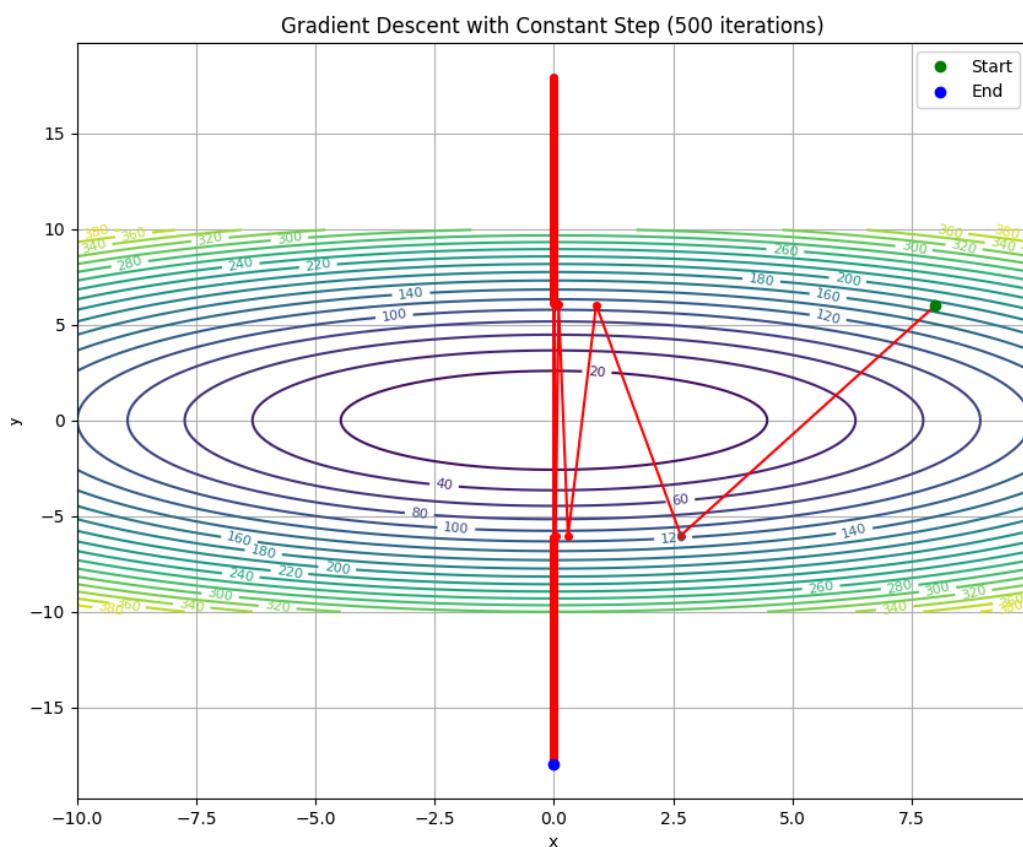
- постоянный шаг
 - $h(k) = h_0$
- функциональная зависимость шага: в 2 вариантах
 - $h(k) = \frac{h_0}{\sqrt{k+1}}$
 - $h(k) = h_0 e^{-\lambda k}$
- постоянный с добавлением “инерции”
 - $h(k) = h_0 + h(k - 1)a$

Для постоянного шага важно выбрать оптимальное значение h_0 . Слишком маленькое значение может вызвать медленное сходжение, а слишком большое - осциллирование или расхождение.

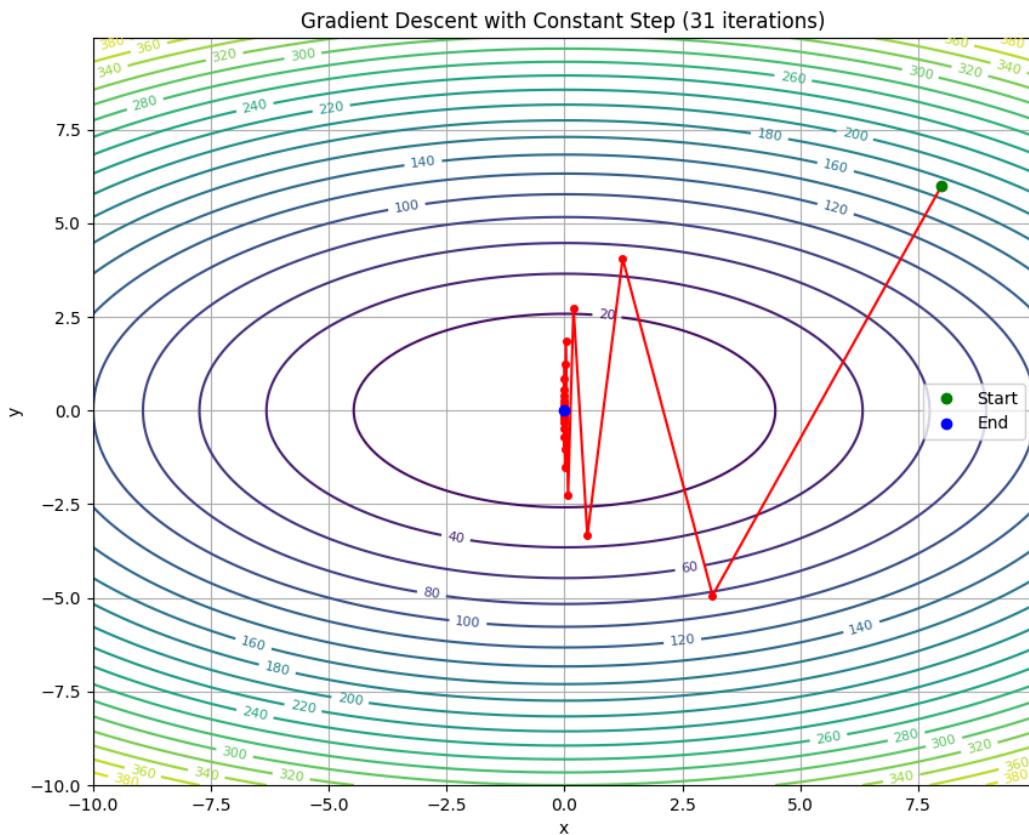
На практике хорошим является такое h , что величина $h * \lambda_{max}$ близка к 1 и обязательно меньше 2

(тут λ_{max} - наибольшее из собственных чисел матрицы квадратичной функции)

Если это значение больше 2, то возникает осцилляция с расхождением:
 (тут для наглядности взято такое h , что $h * \lambda_{max}$ чуть-чуть больше двойки - видно, что по одной координате сходжение удалось, по другой, соответствующей λ_{max} - нет)



Если это значение от 1 до 2, то сходимость есть, но осциллирующая:



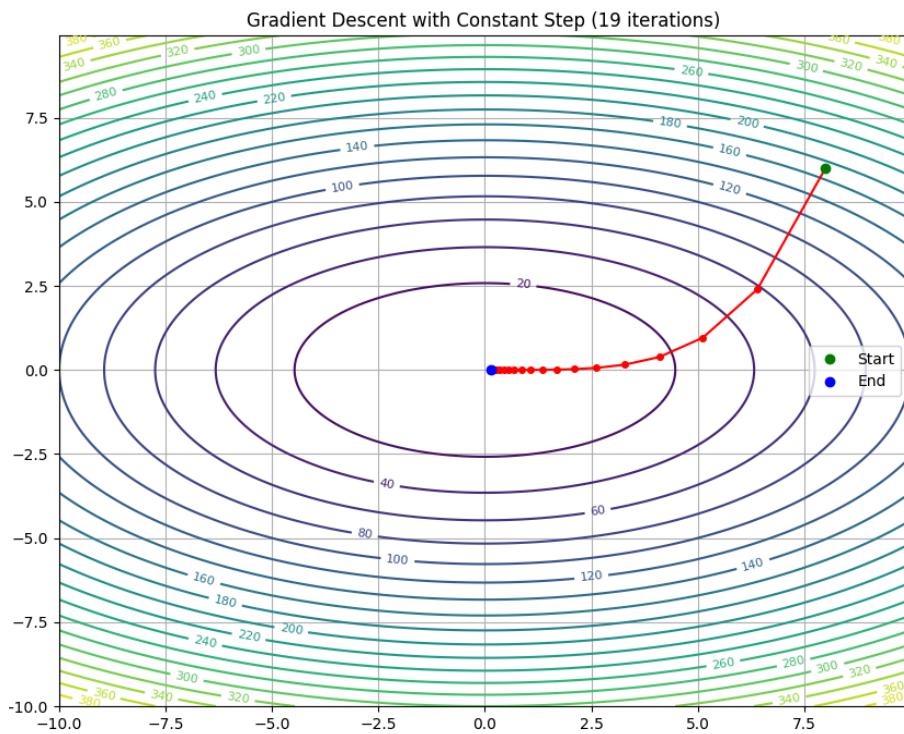
Для функциональной зависимости также важно соблюсти баланс, так как слишком большое стартовое значение h может спровоцировать расхождение куда-то далеко на первых шагах, а слишком маленькое - мелкие шаги, которые могут вообще не успеть дойти до минимума в ряде случаев

Для квадратичных функций была выбрана стартовая точка $(8; 6)$, как достаточно удаленная от оптимума $(0; 0)$.

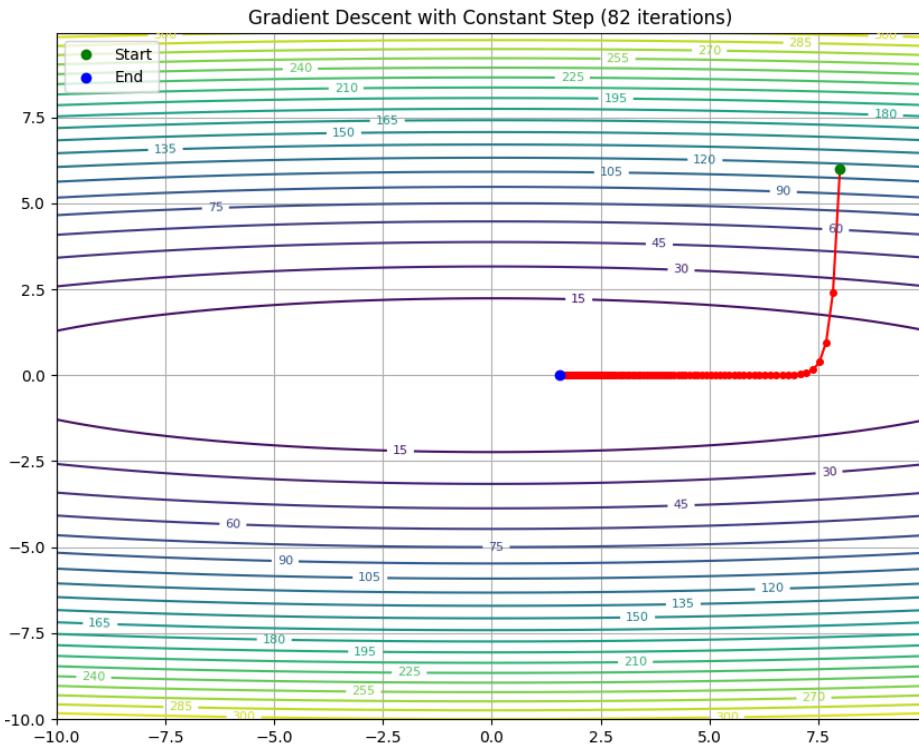
Результаты. Исследование эффективности методов и сравнительный анализ:

- В случае хорошей обусловленности:

Шаг взят немножко меньше оптимального постоянного для наглядности
покоординатного сходжения



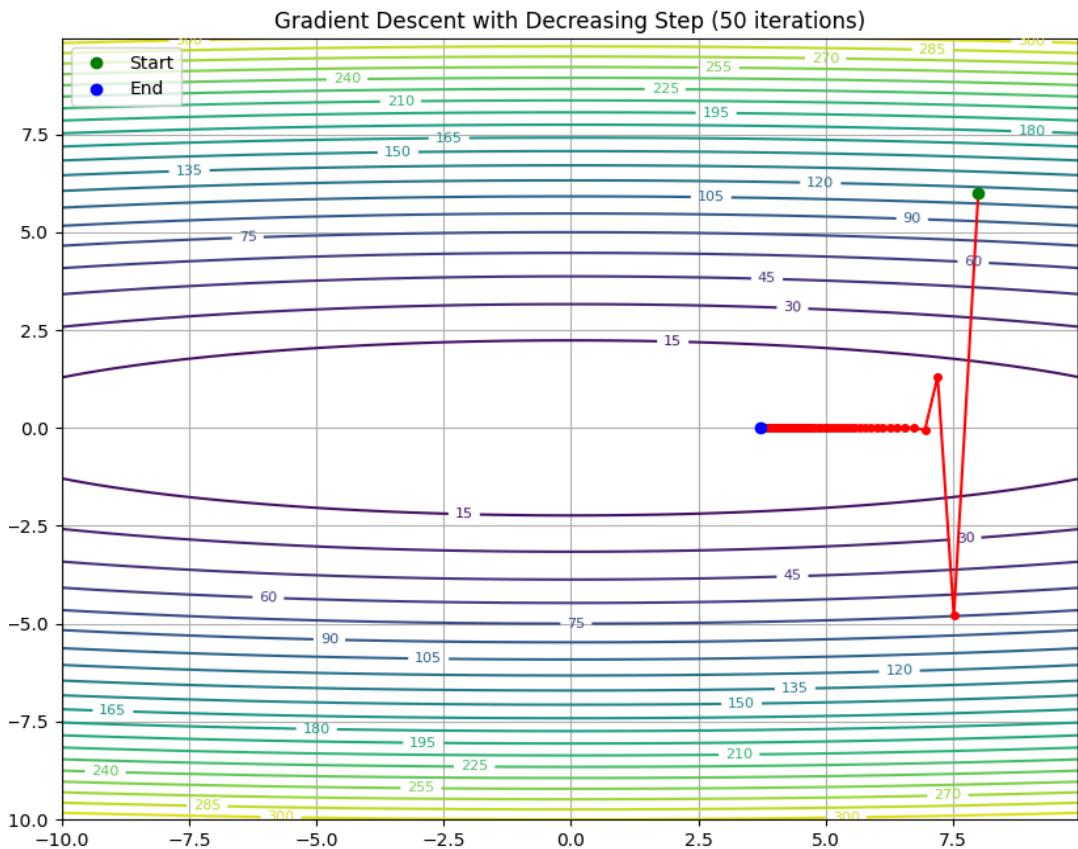
- В случае плохой обусловленности:



В обоих случаях метод сходится к глобальному минимуму, однако в случае плохой обусловленности скорость сходимости по x-координате заметно медленнее и нужно больше итераций. Это связано с тем, что, так как x-координата является базисной при переводе матрицы функции в диагональный вид (она сразу диагональна), скорость сходимости по ней

экспоненциальна с основанием $1 - h\lambda$ что в случае маленького собственного числа, как у координаты x , близко к 1.

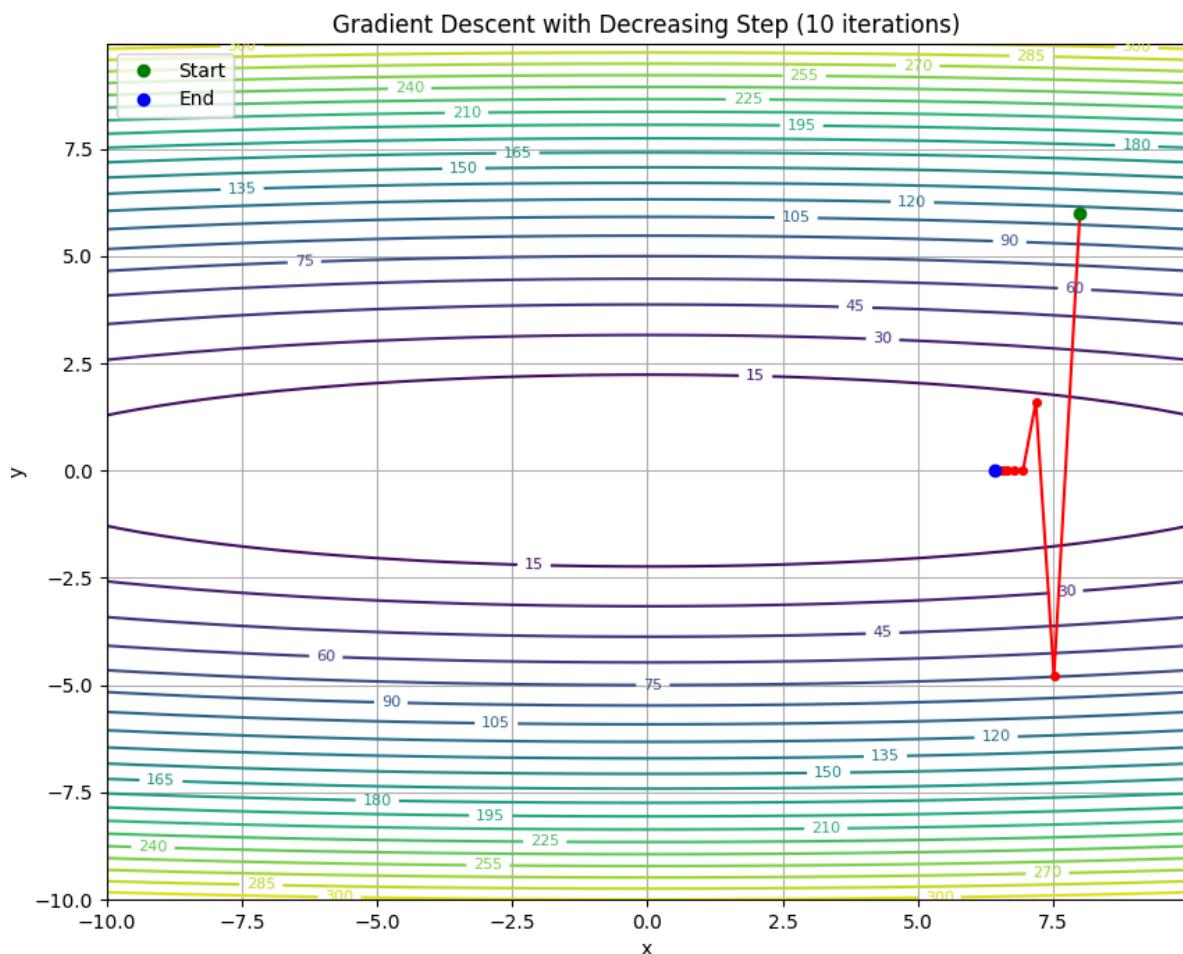
- Для функциональной зависимости шага, изначальный размер которого взят несколько больше оптимального постоянного:
 - Вариант с корнем



Здесь для случая плохой обусловленности заметно, что функция не успела сойтись достаточно близко к оптимуму, прекратив спуск, так как условие остановки - близость x_k и x_{k+1} , а скорость сходимости для плохой обусловленности очень плохая.

Так происходит потому, что функциональная зависимость шага может помочь при поиске узких "ям" большой глубины, не проскочив их, но ямы в случае квадратичных функций достаточно пологие, и в ней нет смысла

- Вариант с экспонентой:

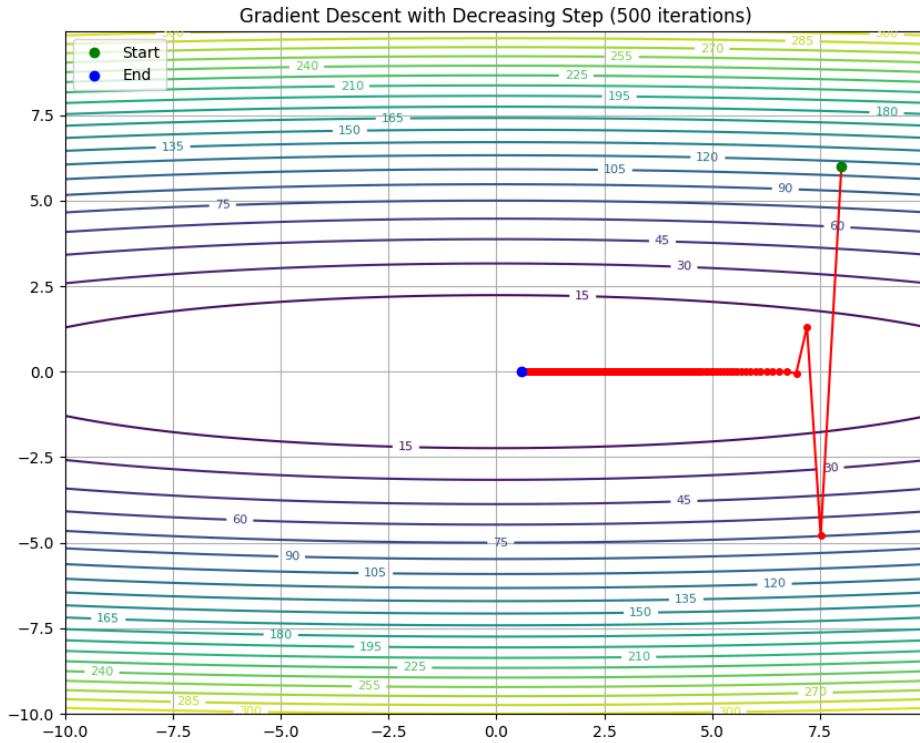


Результаты в целом аналогичные, за тем исключением, что размер шага уменьшается быстрее, а значит и прекращение спуска случается раньше.

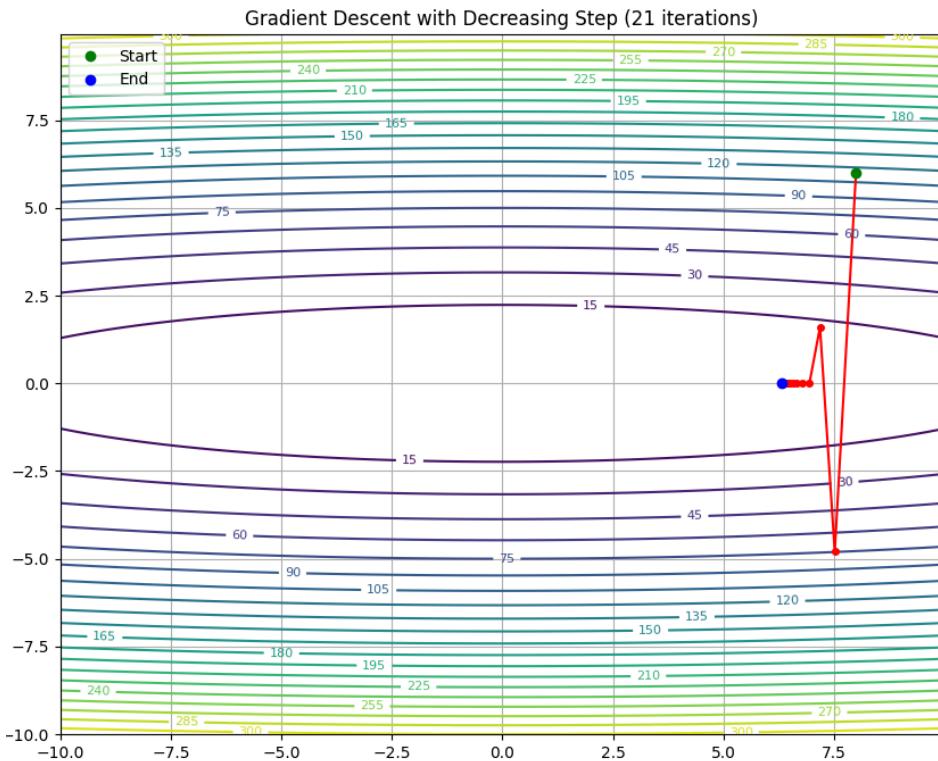
Есть основания полагать, что в этом случае может случится так, что в $(0; 0)$ не выйдет попасть даже через бесконечное число итераций без прерываний, так как можно оценить норму градиента на приведенной области как некое M , тогда длина i -го шага будет не больше $M * h_0 * e^{-\lambda k}$, а ряд по k сходится - значит, если сумма этого ряда слишком мала (например, при больших λ или небольшом h_0), не получится уйти от исходной точки дальше некоторой константы.

- Также исследуем различия между двумя функциональными зависимостями внимательнее, уменьшив ϵ до 0.000001

- Вариант с корнем:

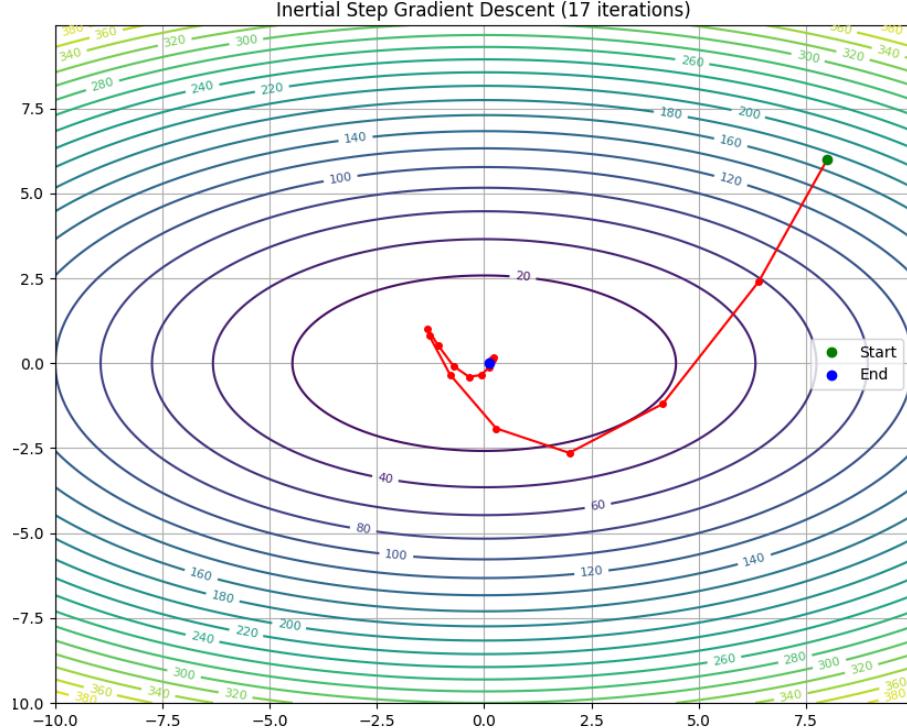


- Вариант с экспонентой:

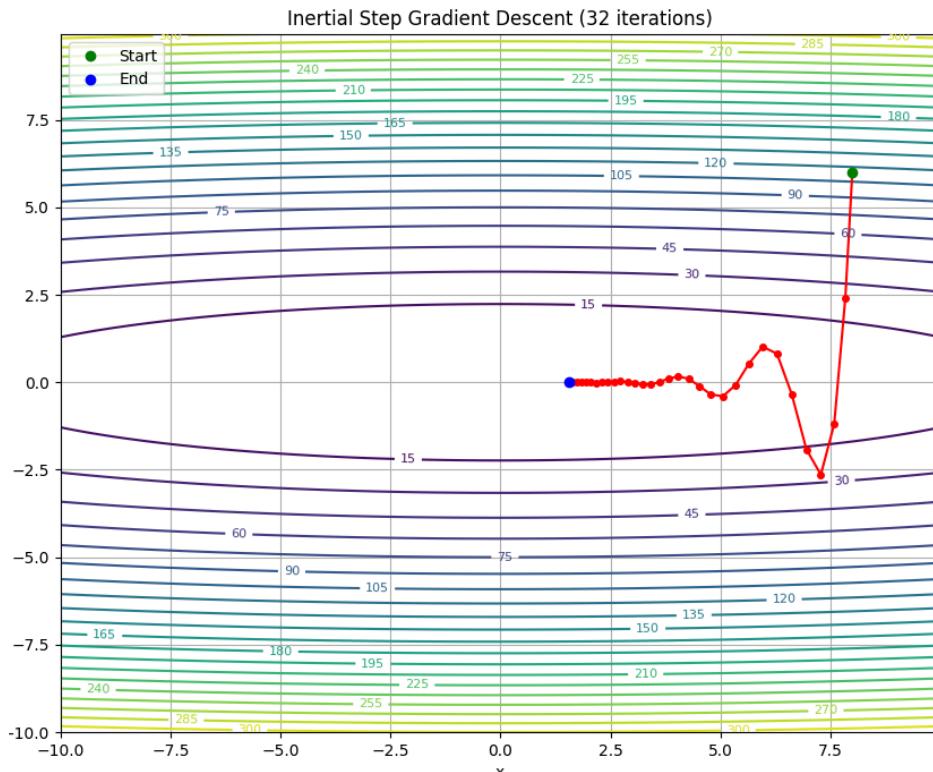


Видим, что обратный корень в целом сходится к $(0; 0)$, если предоставить ему большую точность и много итераций, а экспонента быстро зависает далеко от оптимума и недвигается дальше.

- Шаги с инерцией не показали свою эффективность, они могут быть полезны на мультимодальных функциях, но на унимодальных инерция только мешает:
 - В случае хорошей обусловленности:



- В случае плохой обусловленности:



Вывод по различным стратегиям выбора шага:

Для простых функций лучше всего показал себя метод с постоянным шагом, где постоянный шаг выбран удачно из теоретических соображений. Функциональная зависимость и шаг с инерцией также показали сходимость, однако менее эффективную - это связано с тем, что они более гибкие в случае функций более сложного поведения с различным гессианом в разных точках, но на простых постоянный шаг действует лучше.

Для плохой обусловленности все методы сходятся заметно хуже.

2. Реализация метода одномерного поиска и градиентного спуска на его основе + доп. задание 1 (реализация дополнительного метода одномерного поиска и градиентного спуска)

Описание методов:

Сначала опишем градиентный спуск вида `steepest_descent` и как он работает при заданном методе одномерного поиска

Так же, как и градиентный спуск, `steepest_descent` - итеративный процесс, на основании точки x_k строящий точку x_{k+1} , которому дается метод одномерного поиска, функция и стартовое приближение x_0 , а целью является отыскание минимума целевой функции $f(x)$, алгоритм завершает работу по достижению критерия остановки или предохранителя. Но алгоритм перехода к новой точке принципиально отличается:

Чтобы по x_k получить x_{k+1} , необходимо сделать следующее:

1. Определить градиент в точке x_k . Определить направление антиградиента
2. Найти минимум на прямой, коллинеарной найденному направлению. Для нахождения этого минимума используется метод оптимизации для одномерного случая - находится минимум функции
$$g(t) = f(x_k + t * \nu),$$
 где t - выбранное направление. Затем по найденному оптимальному t восстанавливается новая точка x_{k+1} .

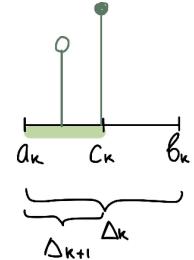
Классический вариант одномерных методов оптимизации - метод, который исходно имеет промежуток $[a_0; b_0]$, затем на каждой итерации сужает его до нового отрезка $[a_{k+1}; b_{k+1}]$, пока не достигнет своего критерия остановки

В качестве методов одномерного поиска были рассмотрены метод золотого сечения и метод дихотомии

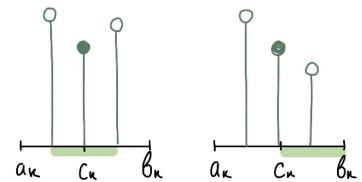
Краткий экскурс в детали методов дихотомии в методы дихотомии и золотого сечения

1. Метод дихотомии

Возьмём точку c_k посередине отрезка $[a_k, b_k]$. Если слева от него (в $\frac{a_k + c_k}{2}$ посередине между ней и a_k) функция меньше, то отбрасываем правую половину. Иначе смотрим справа (в $\frac{c_k + b_k}{2}$). Если там меньше, то отбрасываем левую половину.



Если ни там, ни там не меньше, то новый отрезок — $\left[\frac{a_k + c_k}{2}, \frac{c_k + b_k}{2}\right]$, то есть между теми двумя, которые мы только что измерили.



На каждой итерации отрезок уменьшается в 2 раза, а измерений мы делаем каждый раз не более 2.

Применяем метод дихотомии итерационно, до тех пор, пока $|a_k - b_k| > \varepsilon$

$$|a_k - b_k| < \varepsilon \rightarrow \text{остановка} \rightarrow \text{ответ} = c_k$$

Погрешность ответа не более $\frac{1}{2^k}$

Оценим, сколько раз нам придётся вызывать функцию, для k итераций — в худшем случае $k \cdot 2$

То есть, погрешность ответа не более $\frac{1}{2^{\frac{n}{2}}} = 0.707^N$

2. Метод золотого сечения

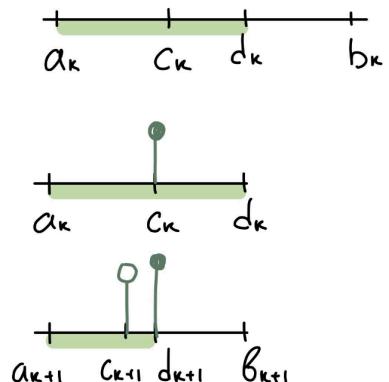
$$\frac{|a_k - b_k|}{|c_k - b_k|} = \frac{|c_k - b_k|}{|a_k - c_k|}$$

Давайте попробуем задать коэффициенты уменьшения отрезков $c_k = 0.382\Delta_k$, $d_k = 0.618\Delta_k = \frac{\Delta_k}{\tau}$, $\tau = 1.618\dots$ — положительная константа золотого сечения.

На каждом шаге, начиная со второго, происходит лишь одно вычисление — одной из границ. В первом нам всё-таки нужно посчитать два значения.

Так же в данном методе $\frac{\Delta_k}{\Delta_{k+1}} = \tau \rightarrow \Delta_k = \frac{1}{\tau^k}$

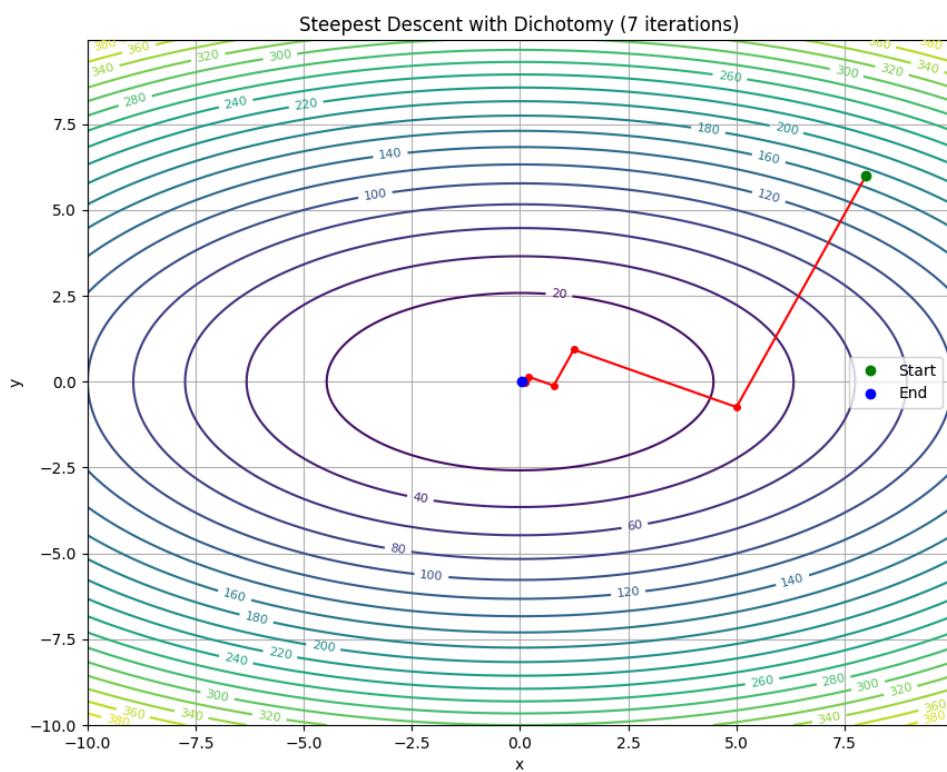
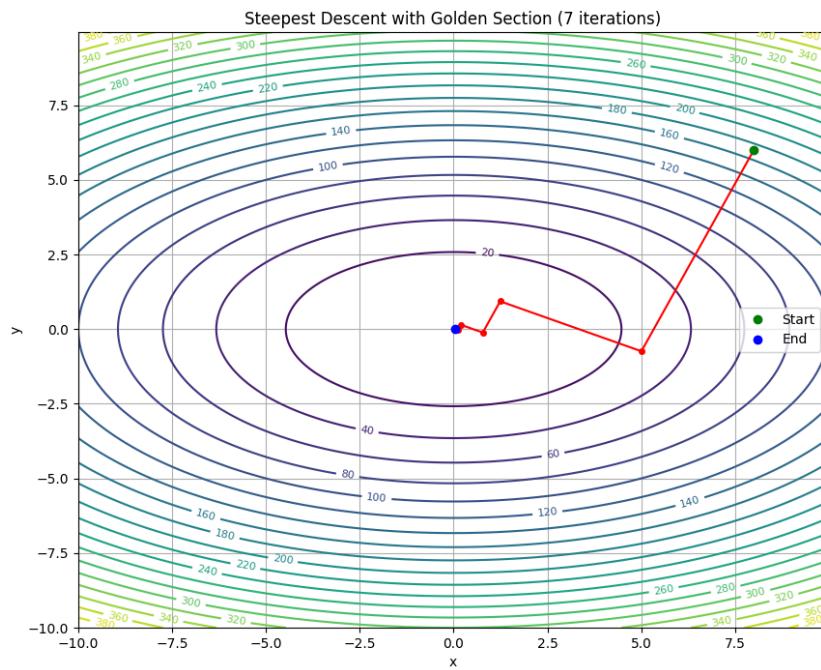
Погрешность ответа не более $\left(\frac{1}{\tau}\right)^k = \left(\frac{1}{\tau}\right)^N = 0.618^N$



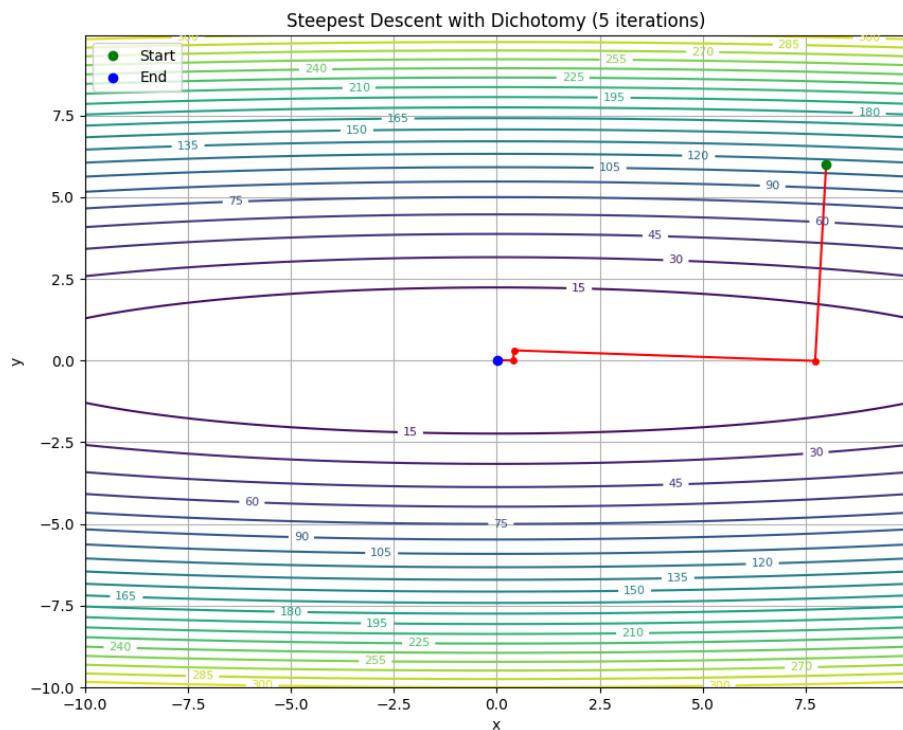
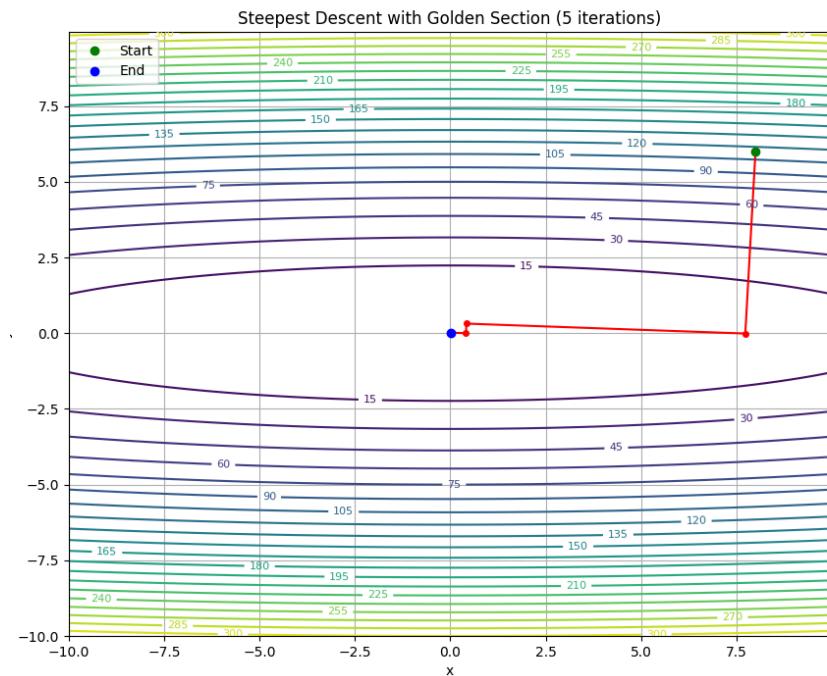
Метод золотого сечения медленнее сужает функцию, чем метод дихотомии, но делает меньше вычислений.

Результаты. Сравнительный анализ

Визуально методы выглядят примерно одинаково, и требуют одинаковое число одномерных поисков для сходжения (что понятно, учитывая, что оба метода, в сущности, возвращают лучшую точку на направлении)
Однако количество вычислений значения функции у них разное:



На хорошей квадратичной функции метод дихотомии потребовал 293 вычисления функции, а метод золотого сечения - только 245



Здесь число требуемых вычислений составило 205 и 175 соответственно

Вывод

Видно, что методы `steepest_descent` сходятся намного быстрее (по количеству точек), чем методы с различным выбором шага, хотя и могут проигрывать по числу вычислений целевой функции

Особенную эффективность методы продемонстрировали при исследовании плохо обусловленных функций - там они фактически сначала могут очень хорошо приближать значение по одной из осей, затем по другой , в итоге процесс сходится всего за несколько итераций

5) Исследование методов на мультимодальной и зашумленной функции

Задача: подобрать мультимодальную и зашумленную функцию и исследовать реализованные методы на эффективность

Описание выбранных функций:

В качестве мультимодальной функции была использована

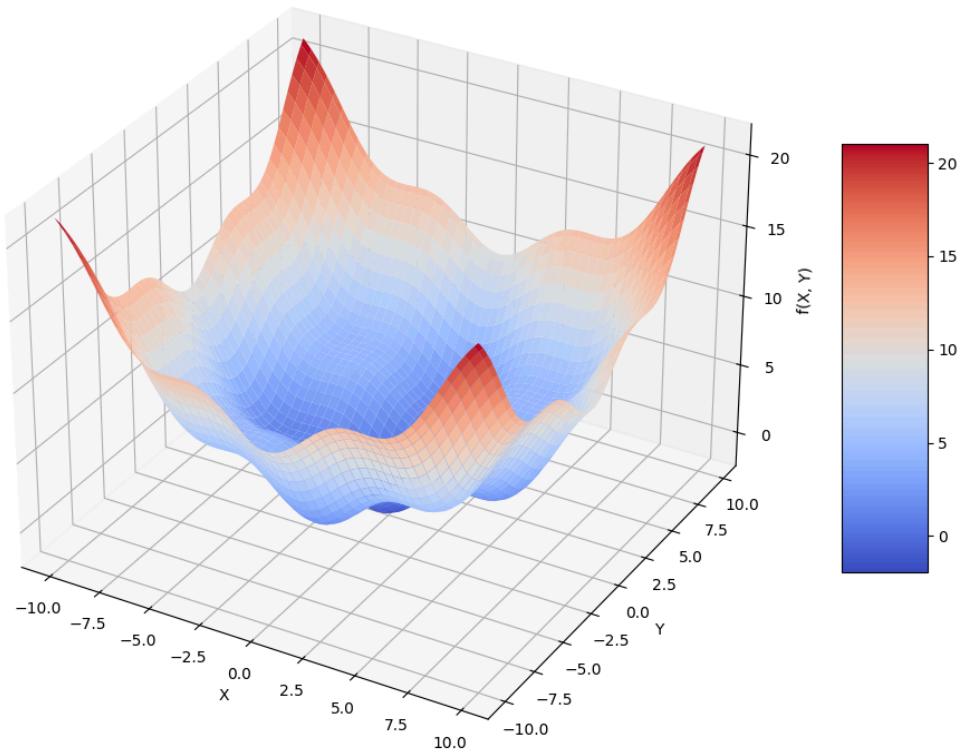
$0.1 * x^2 + 0.1 * y^2 + M * (\sin(x - \pi/2) + \sin(y - \pi/2))$, где параметр M обозначает силу потенциальных ям

В точке $(0, 0)$ у функции единственный глобальный минимум, равный $-2M$, однако синусоидальное слагаемое дает дополнительные локальные минимумы, тем более глубокие (и тем более многочисленные), чем выше параметр M - синусоидальная составляющая начнет играть больший вклад

Так выглядит трехмерный график функции для параметра $M = 1$

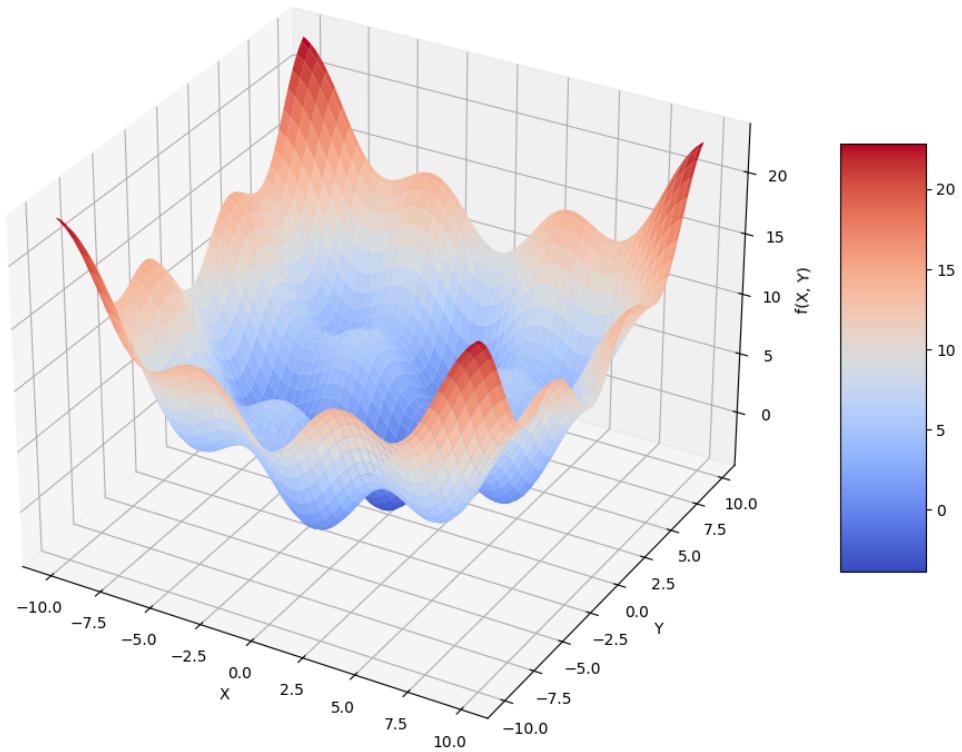
При $M = 1$ видно, что локальные минимумы - едва заметные ямки

Multimodal Function Surface



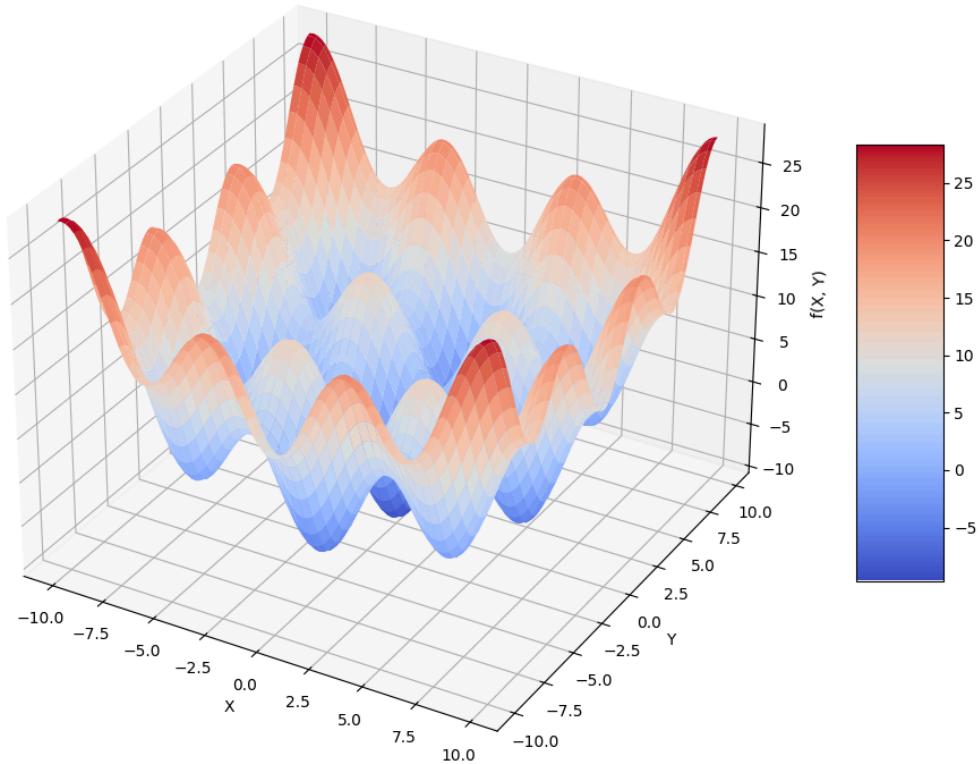
Так выглядит трехмерный график функции для параметра $M = 2$

Multimodal Function Surface



Так выглядит график функции для $M = 5$. Видно обилие локальных минимумов, почти столь же глубоких, что и глобальный

Multimodal Function Surface



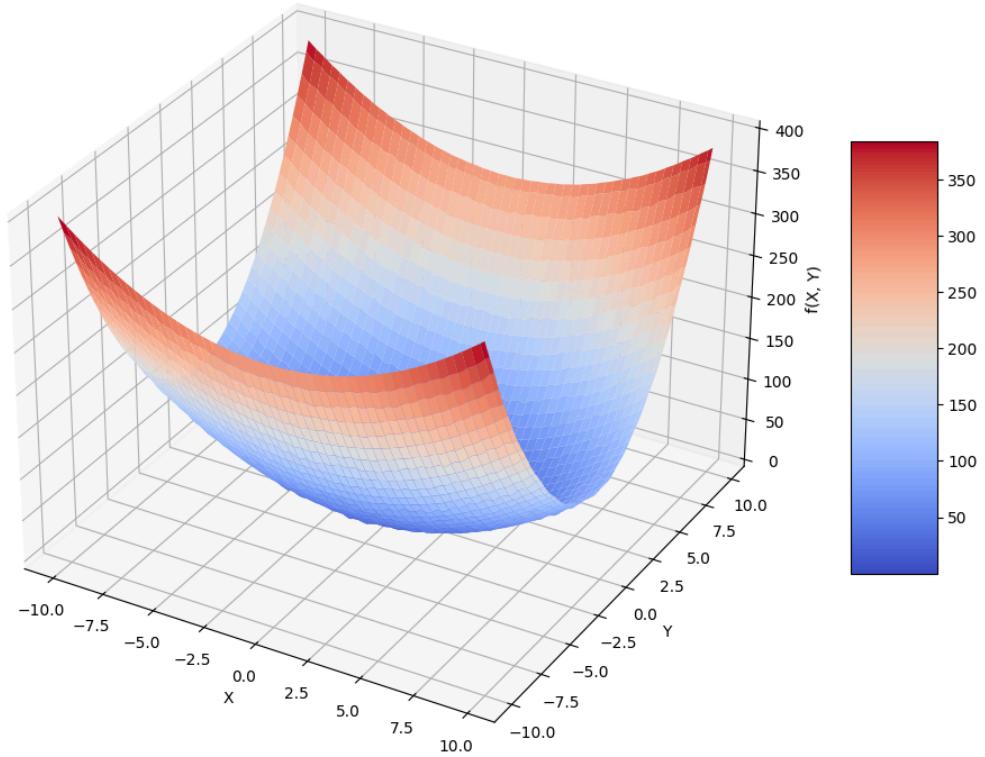
В качестве зашумленной функции была взята уже знакомая $x^2 + 3 * y^2$, к которой добавили слагаемое, равное $\sin(5x)*\sin(3y)*\sin(\pi*x)*(sin(\pi * y/2)*M)$, где M - мощность шума

Видно, что шум ограничен по модулю значением M, и имеет вычислимую производную, что приятно, однако он апериодичен и имеет практически непредсказуемое поведение, что нам и нужно

Также он имеет достаточно крупную производную, особенно при больших значениях M

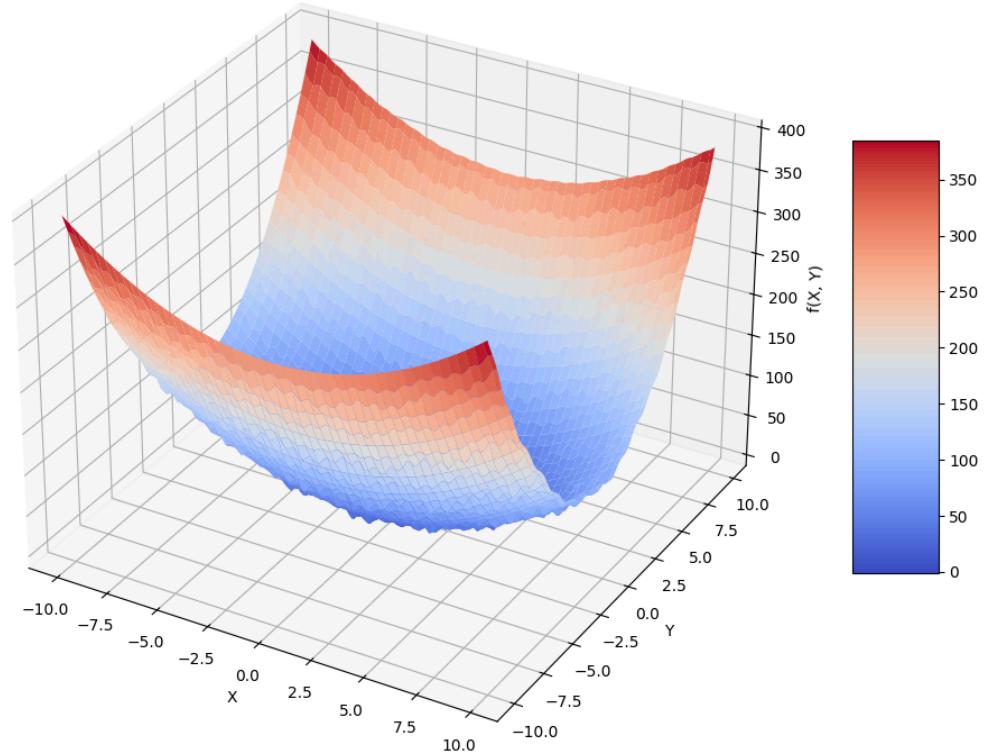
Поведение при силе шума M = 3, видны небольшие колебания:

Noisy Quadratic Function Surface

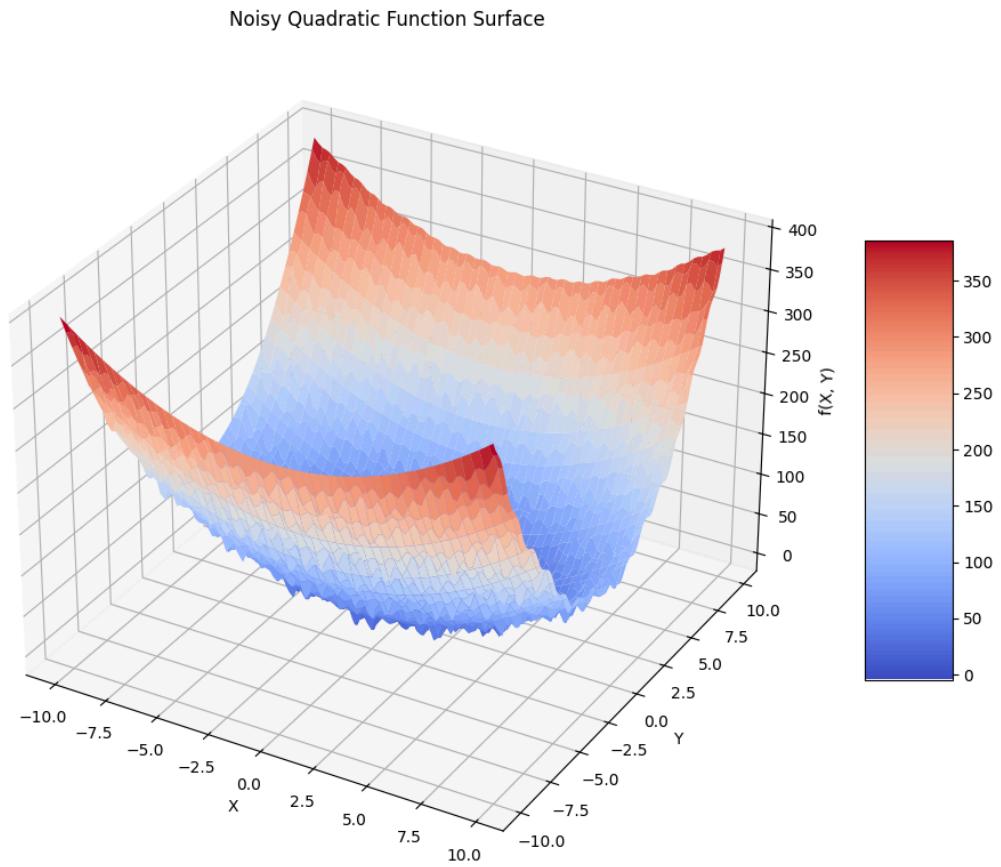


При $M = 10$:

Noisy Quadratic Function Surface



При $M = 20$ колебания очень сильны:

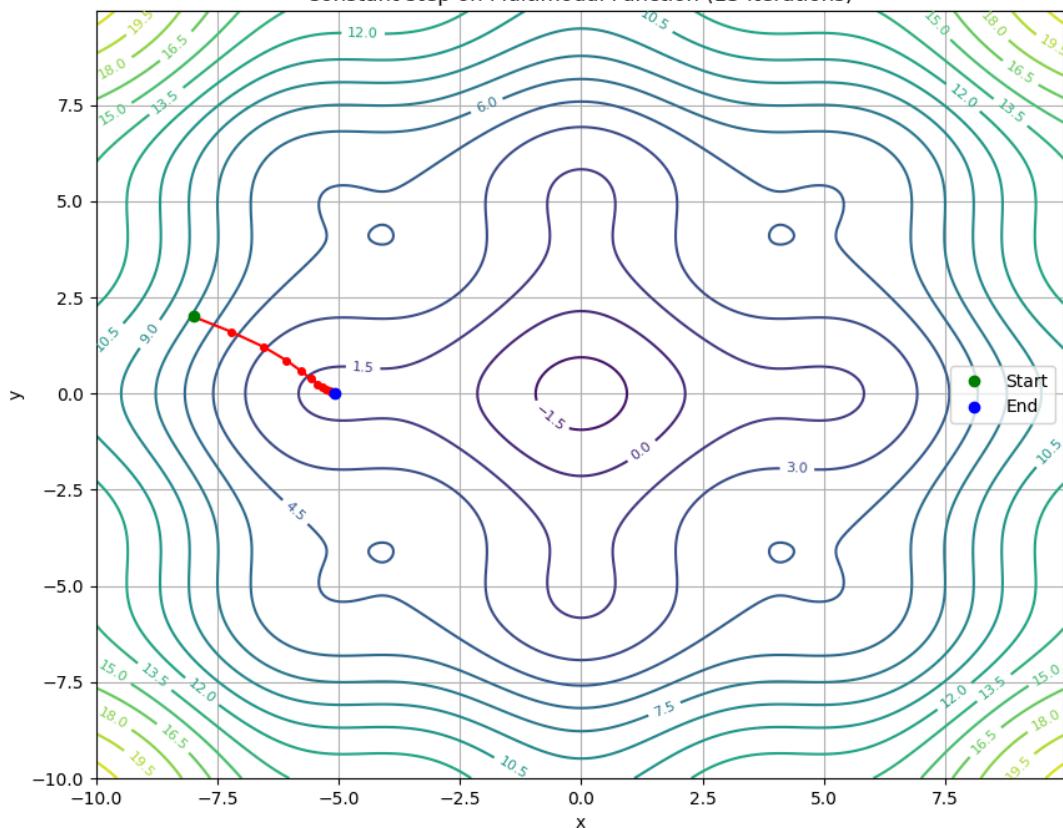


Результаты. Исследование эффективности методов на новых функциях.

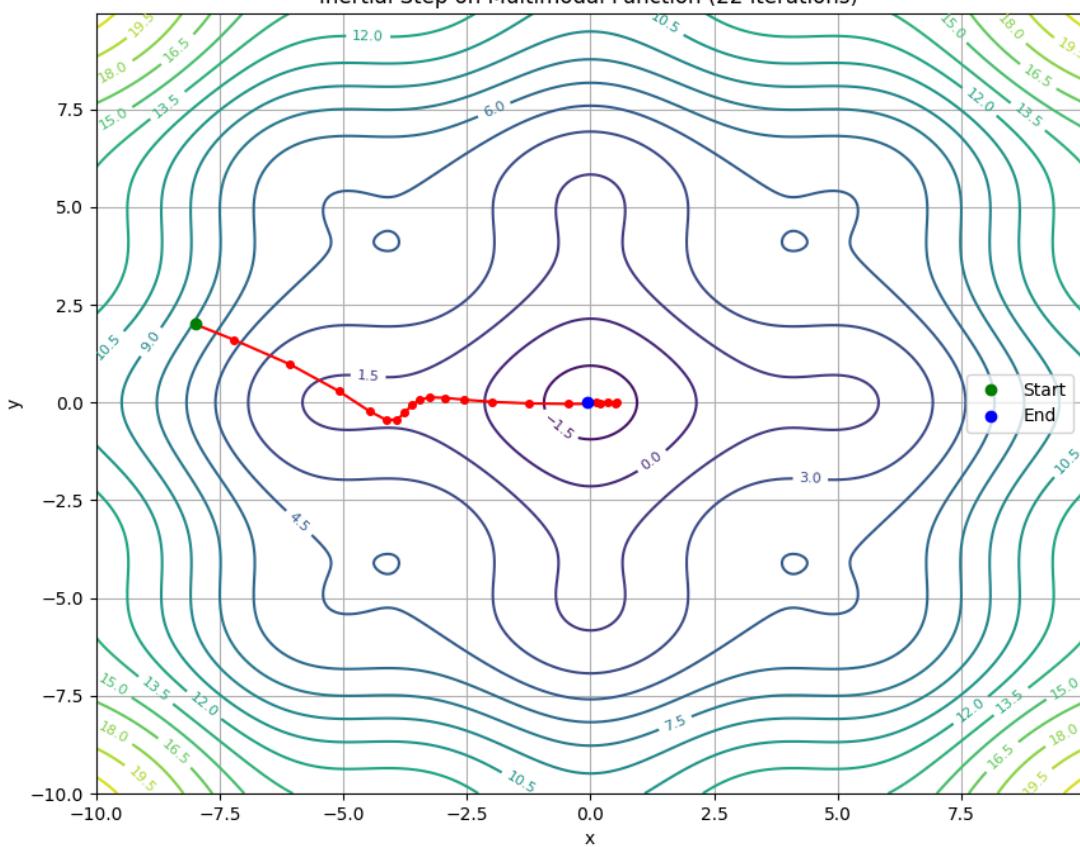
Результаты для параметра $M = 1$. Видно, что метод с константным выбором шага очень уязвим к неудачно попавшимся локальным минимумам на пути, а `steepest_descent` более надежны

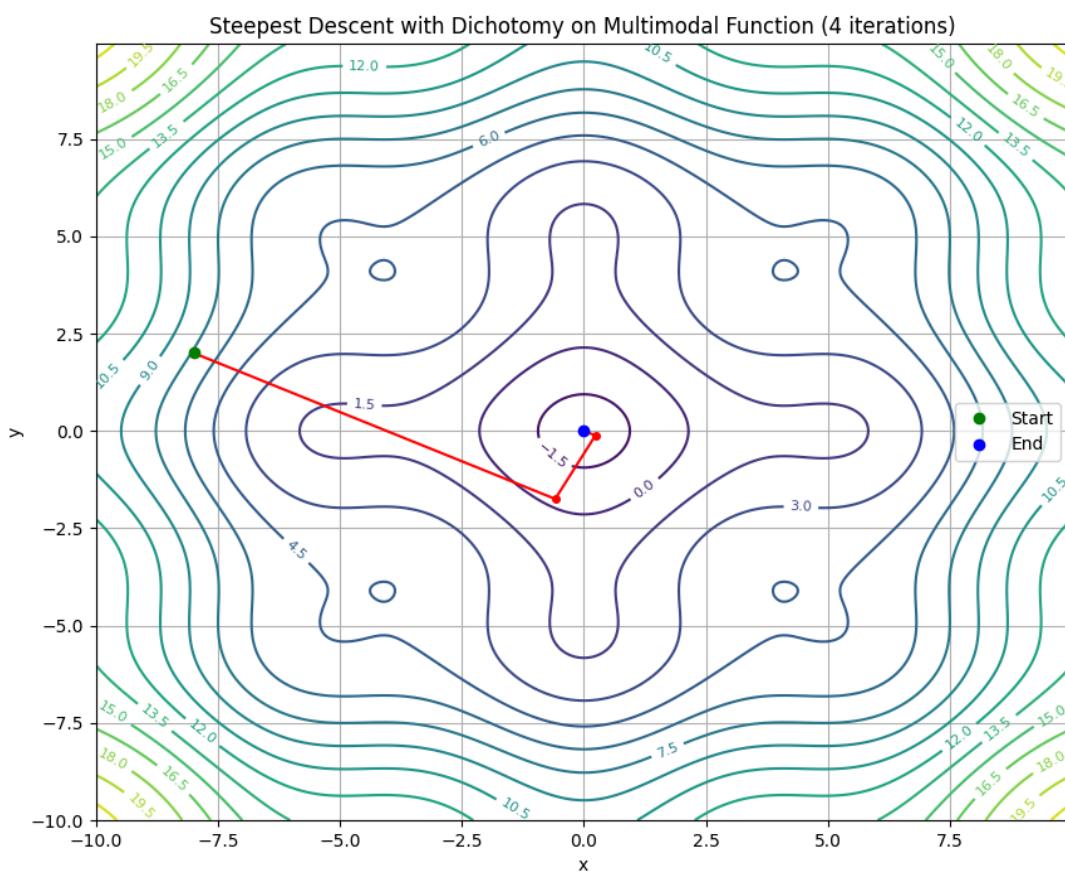
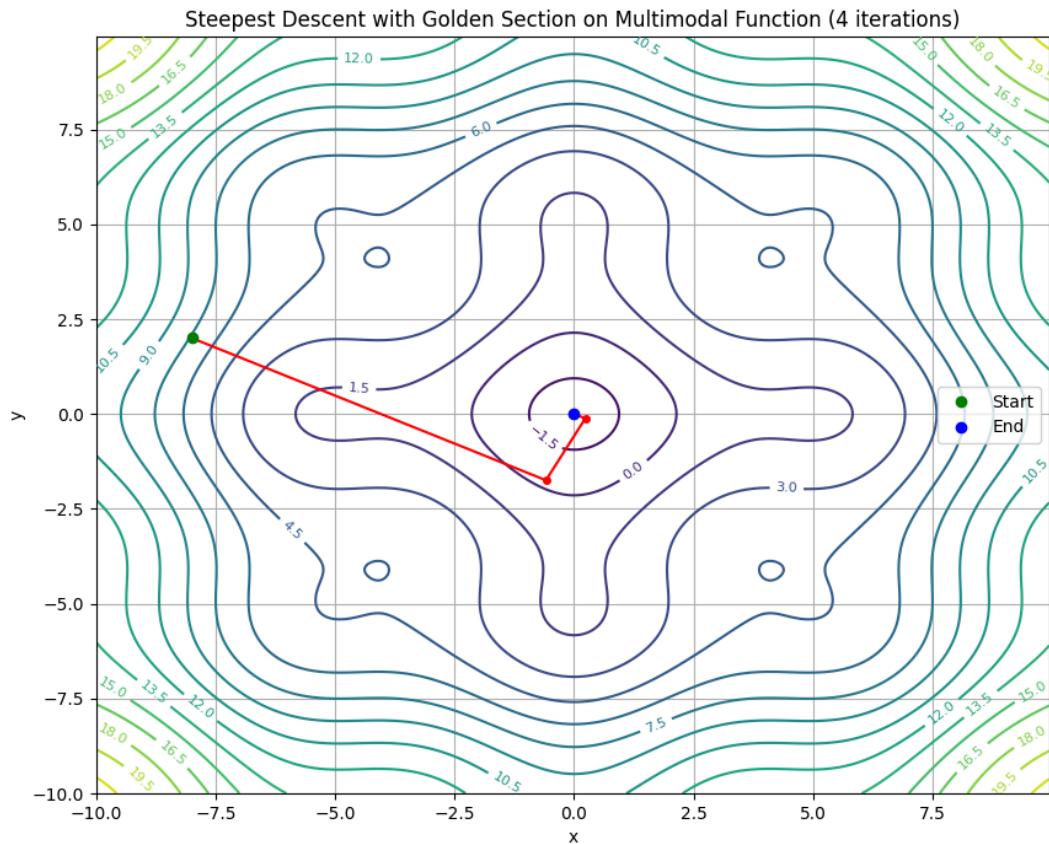
Причем инерция иногда может спасти от мелких ямок:

Constant step on Multimodal Function (13 iterations)

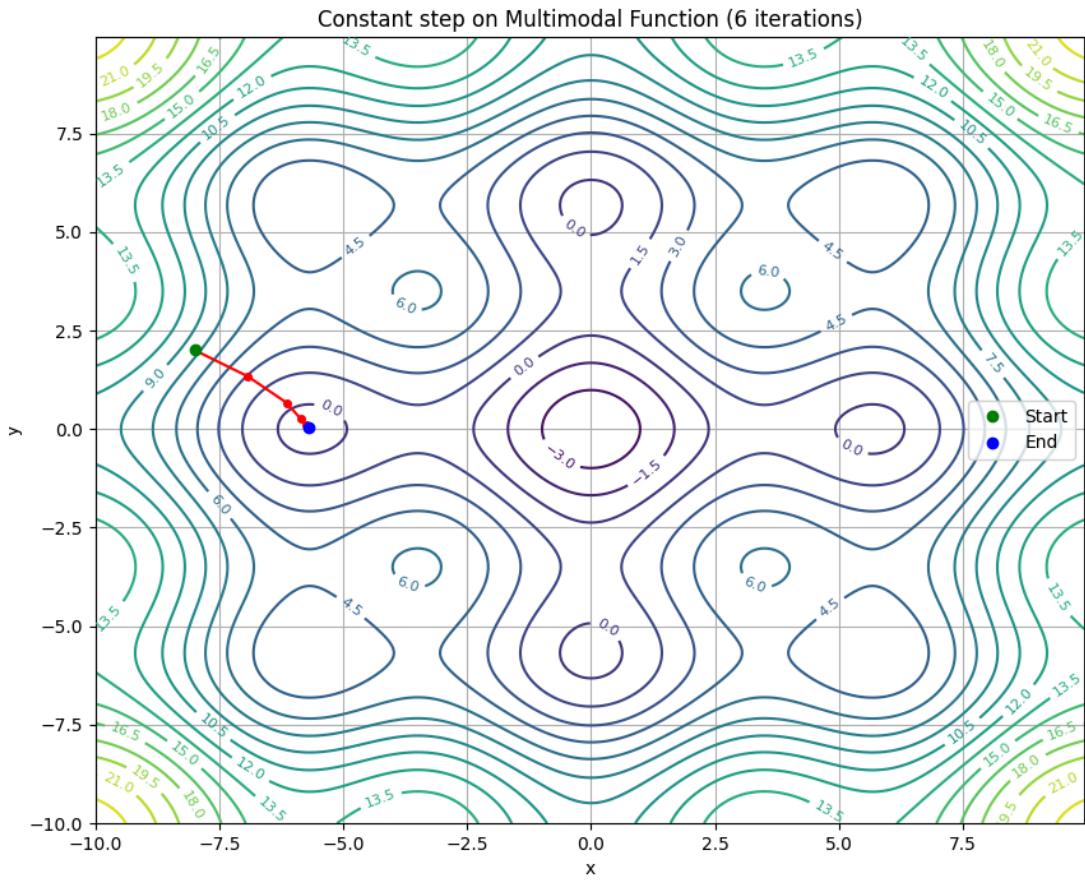


Inertial Step on Multimodal Function (22 iterations)

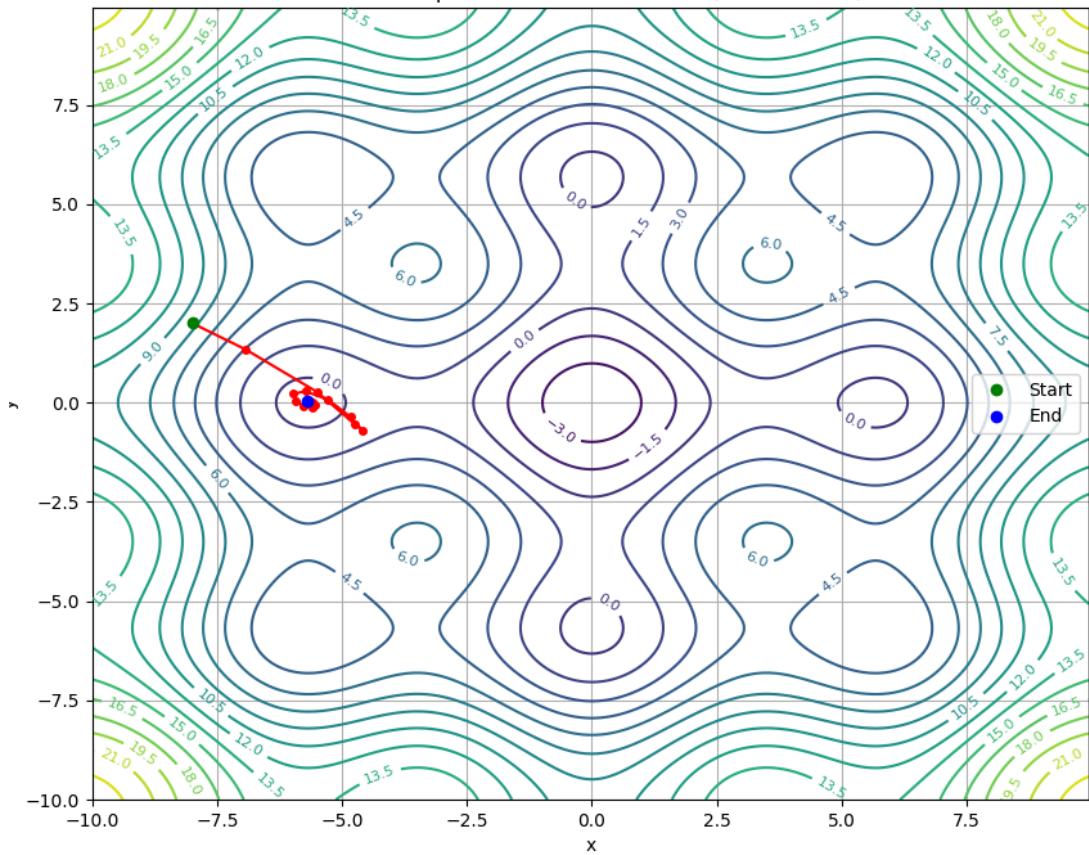




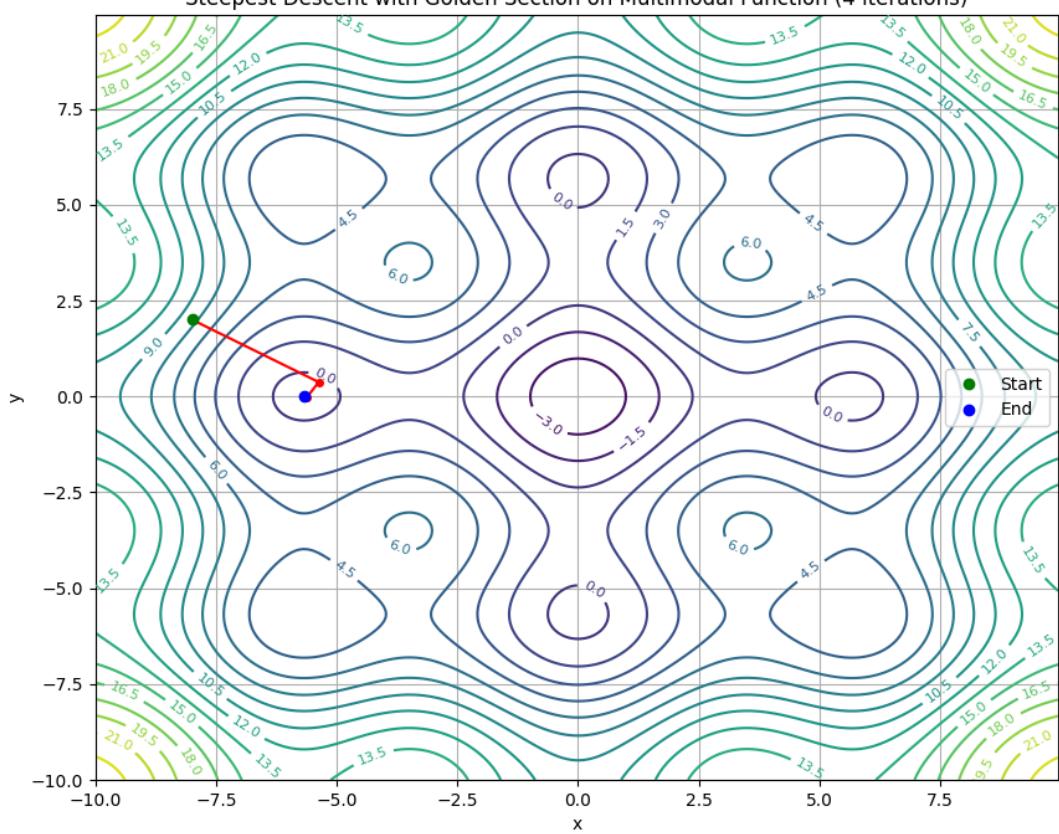
При значении $M = 2$ сила локальных минимумов уже может спровоцировать `steepest_descent` на неверный выбор при неудачном подборе точки, и инерция слабо спасает:



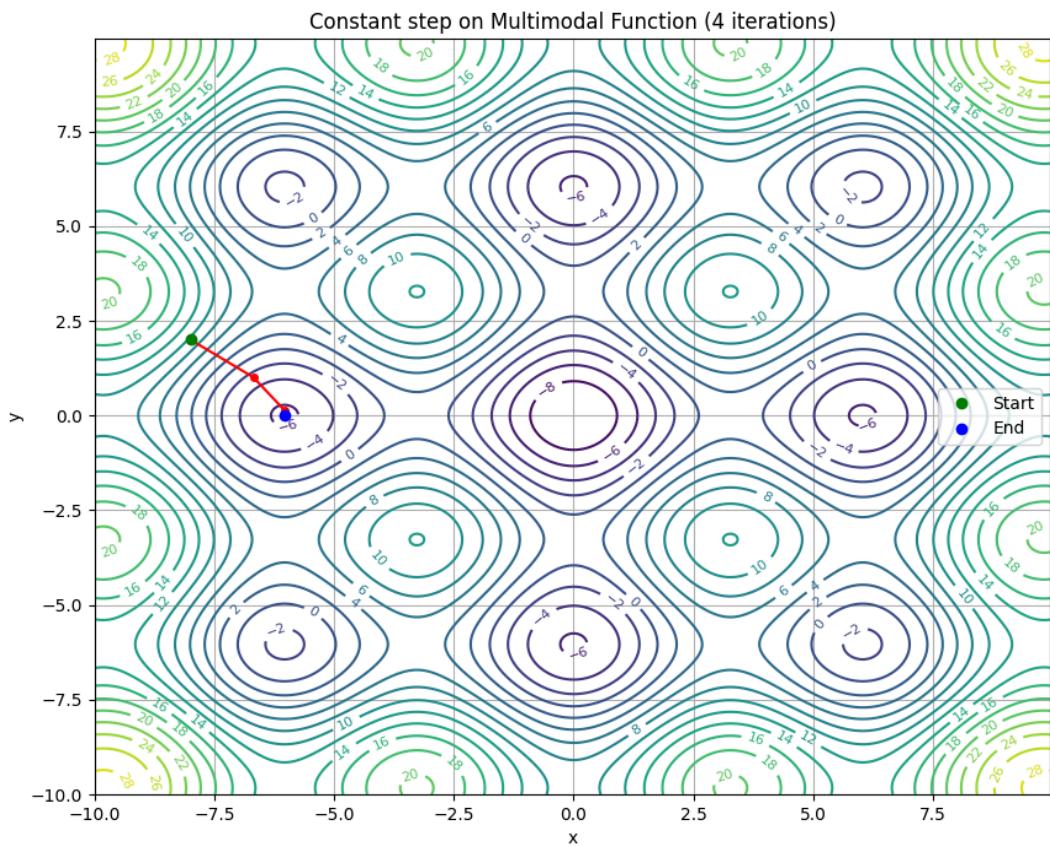
Inertial Step on Multimodal Function (16 iterations)



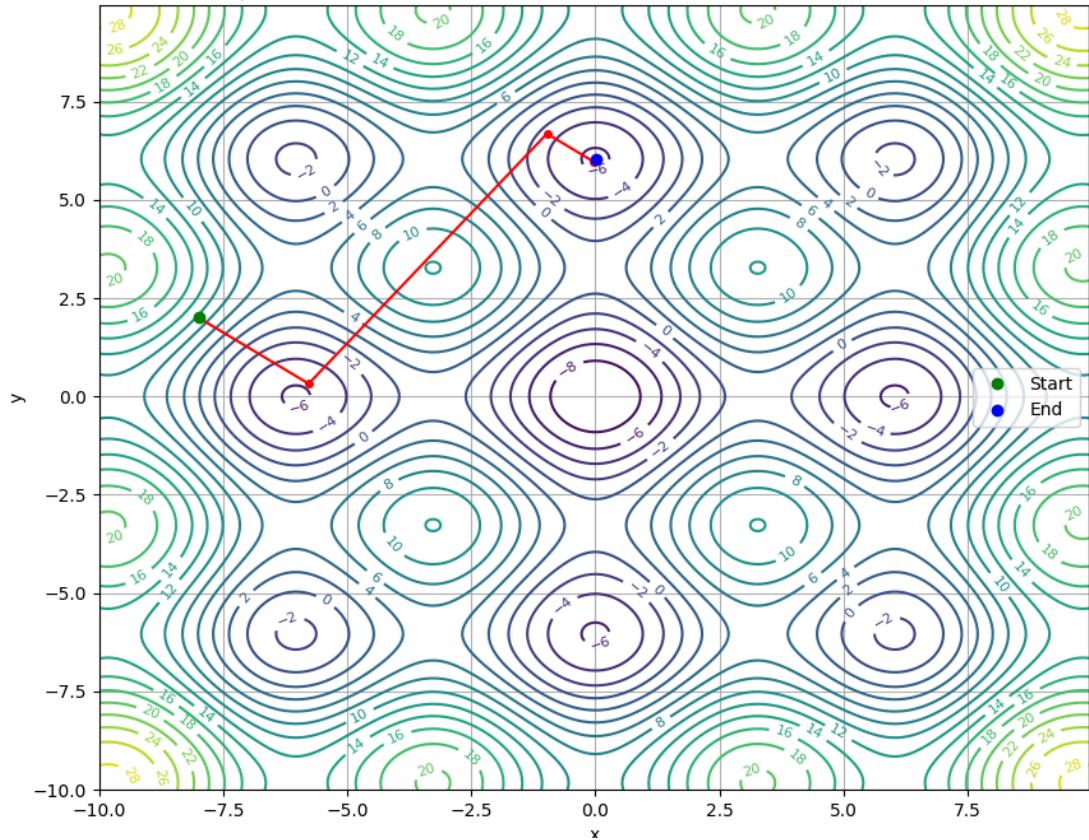
Steepest Descent with Golden Section on Multimodal Function (4 iterations)



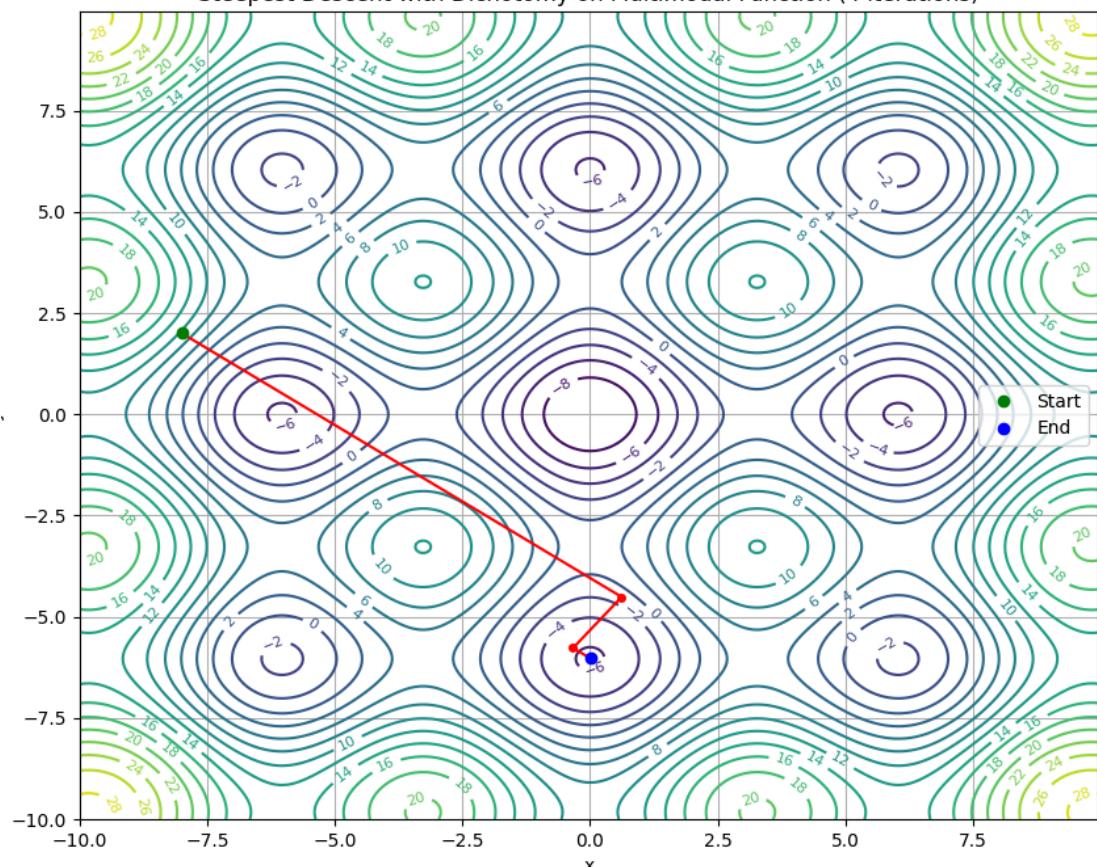
При значении $M = 5$ уже ничто не спасает от сильных локальных минимумов

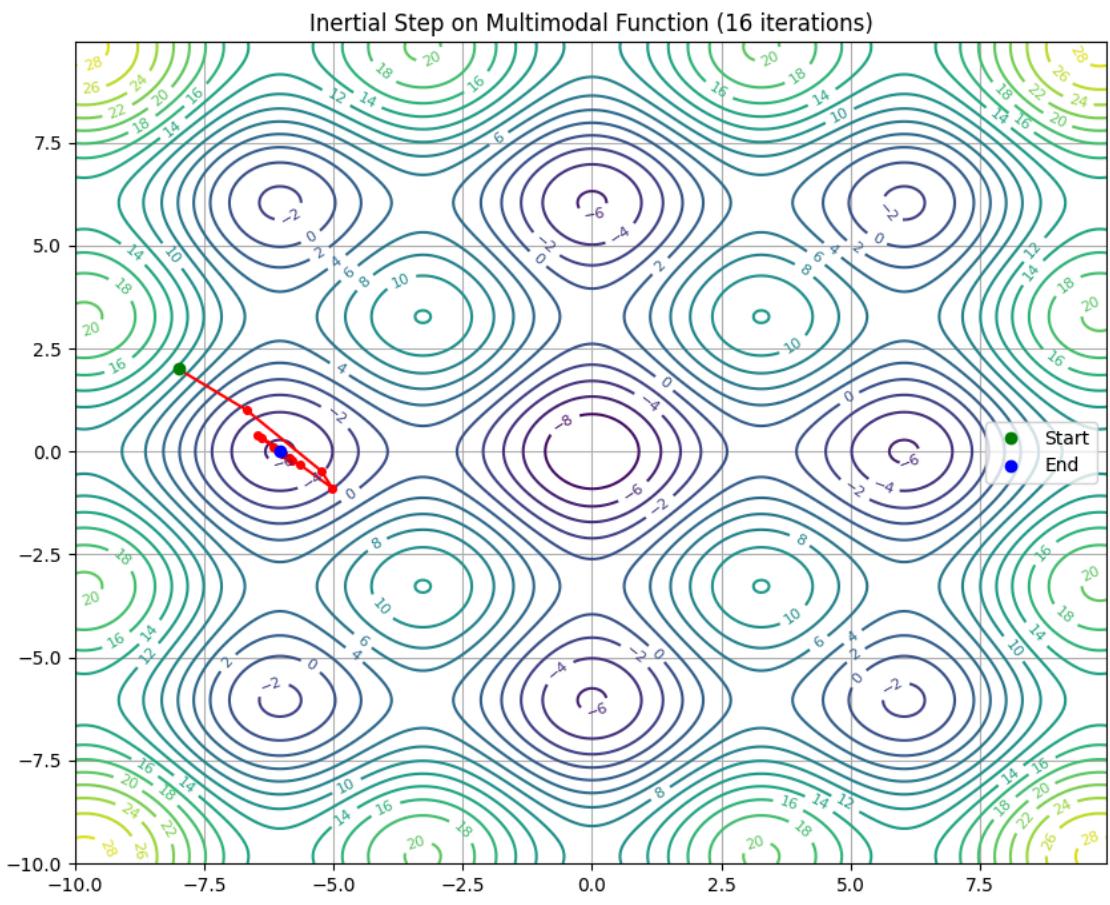


Steepest Descent with Golden Section on Multimodal Function (5 iterations)



Steepest Descent with Dichotomy on Multimodal Function (4 iterations)

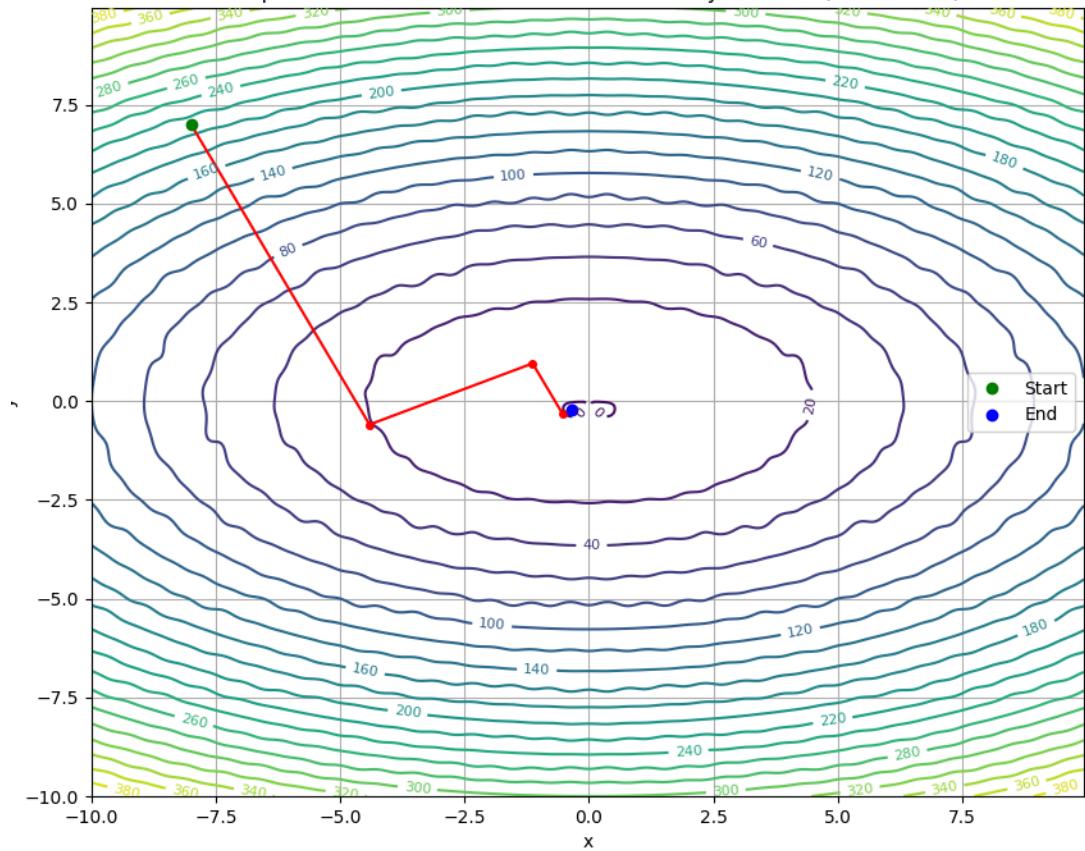




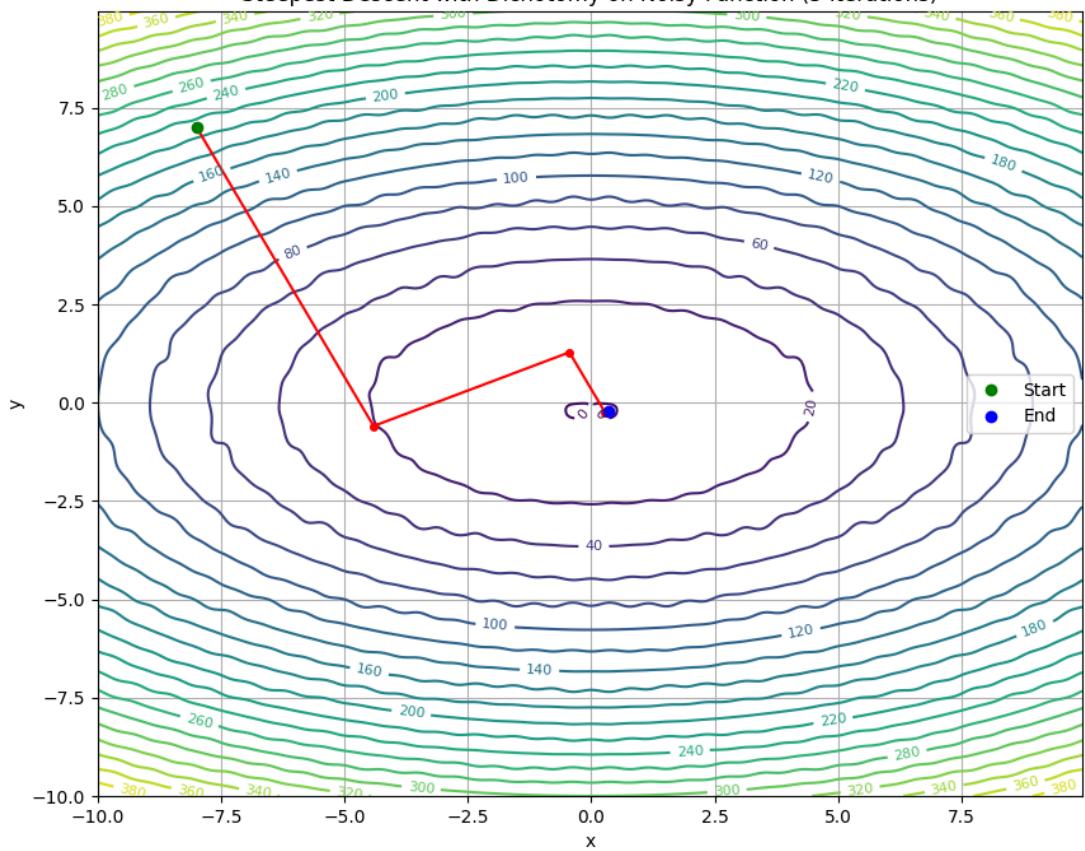
Для исследования поведения методов были выбраны метод дихотомии, метод золотого сечения, метод с постоянным шагом и метод с функциональным шагом (обратный корень)

Для $M = 3$ все методы справляются достаточно хорошо:

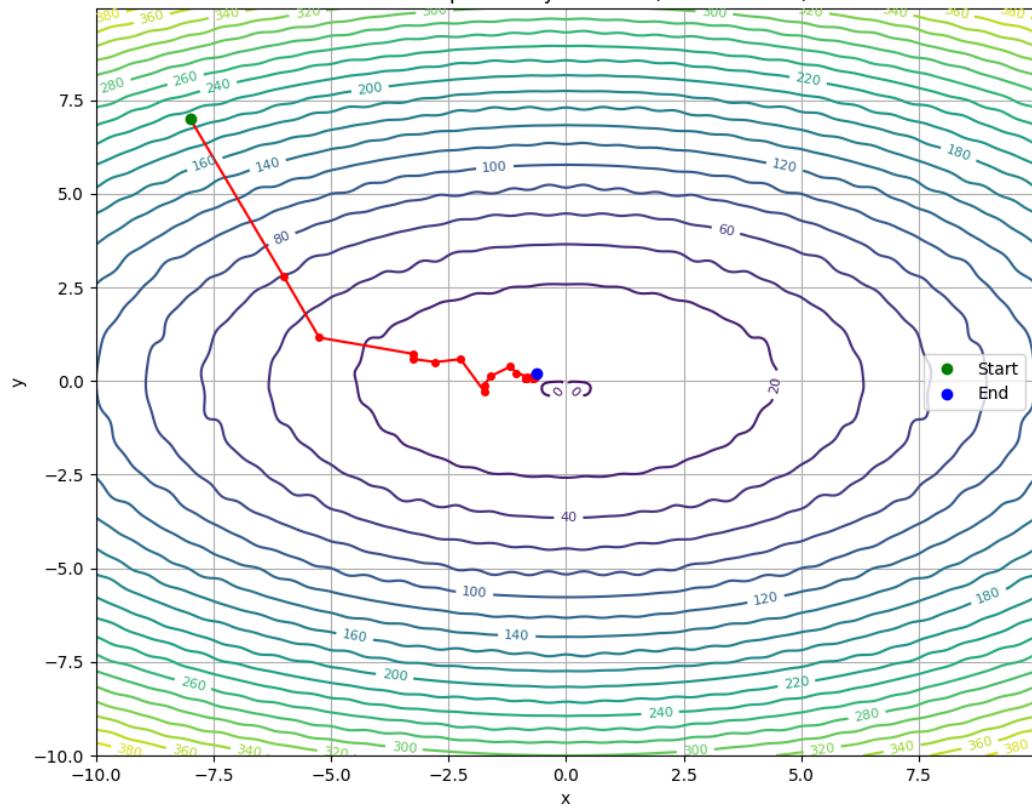
Steepest Descent with Golden Section on Noisy Function (5 iterations)



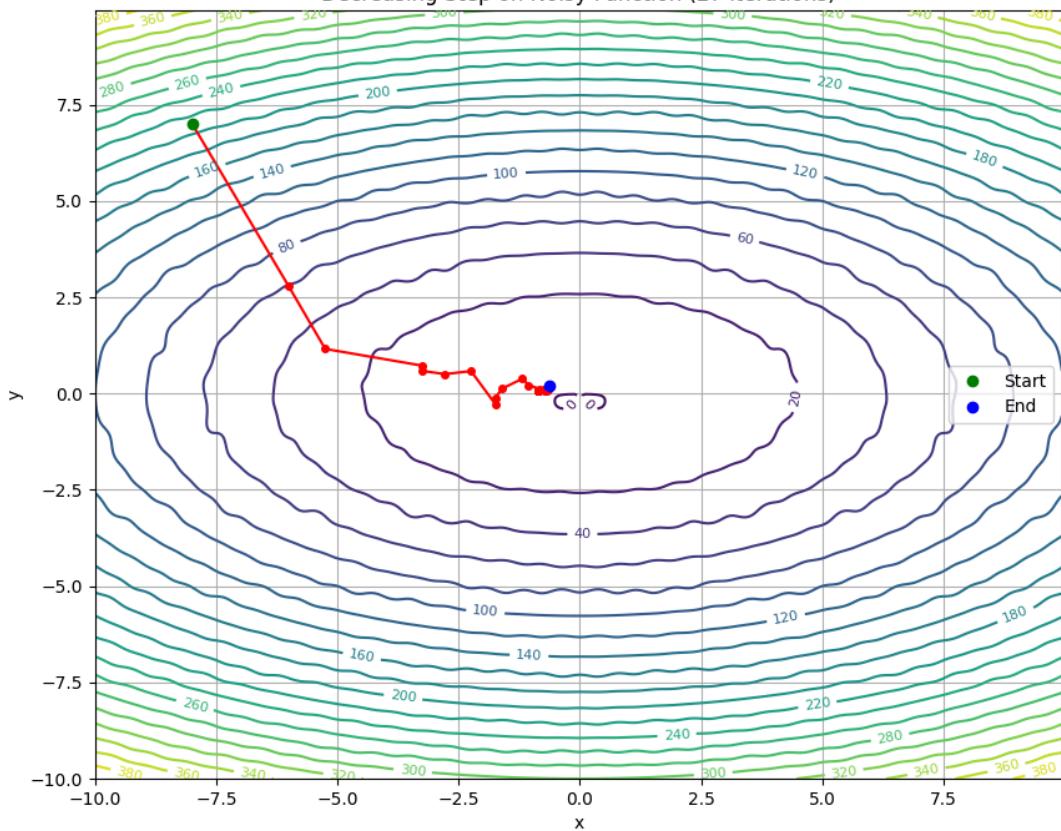
Steepest Descent with Dichotomy on Noisy Function (5 iterations)

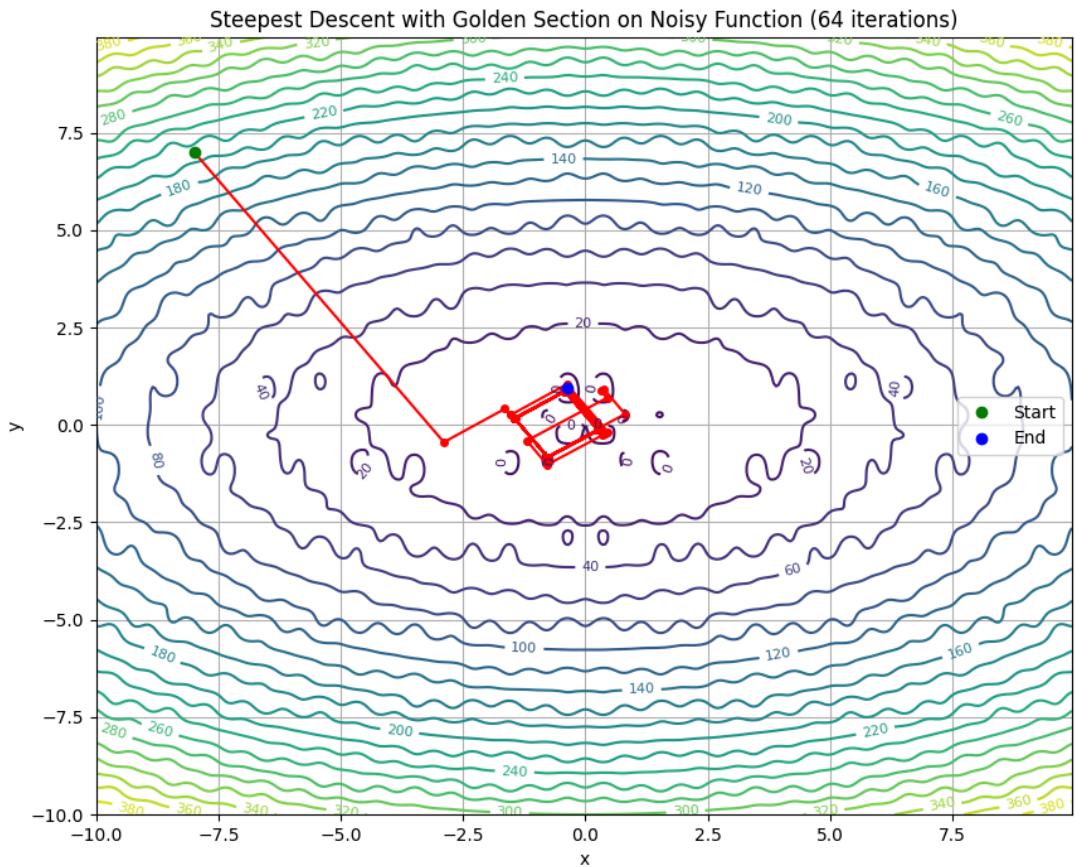


Constant Step on Noisy Function (500 iterations)



Decreasing Step on Noisy Function (27 iterations)



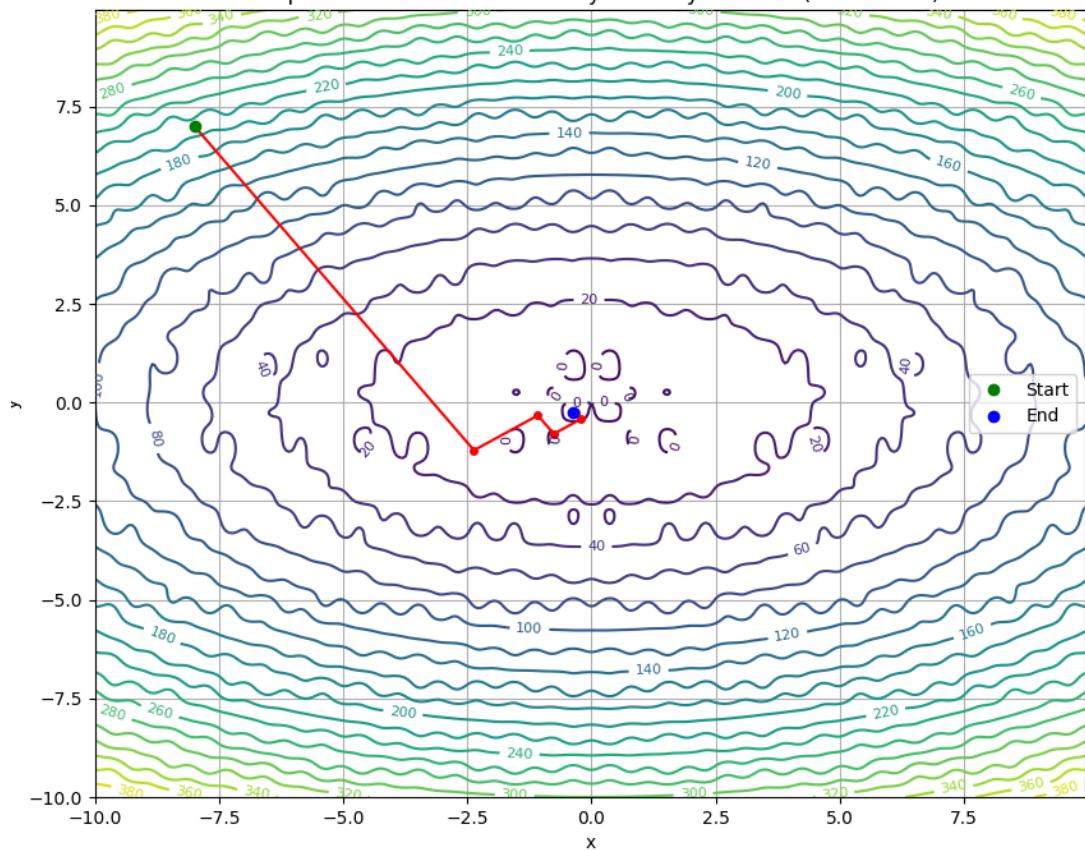


Для $M = 10$:

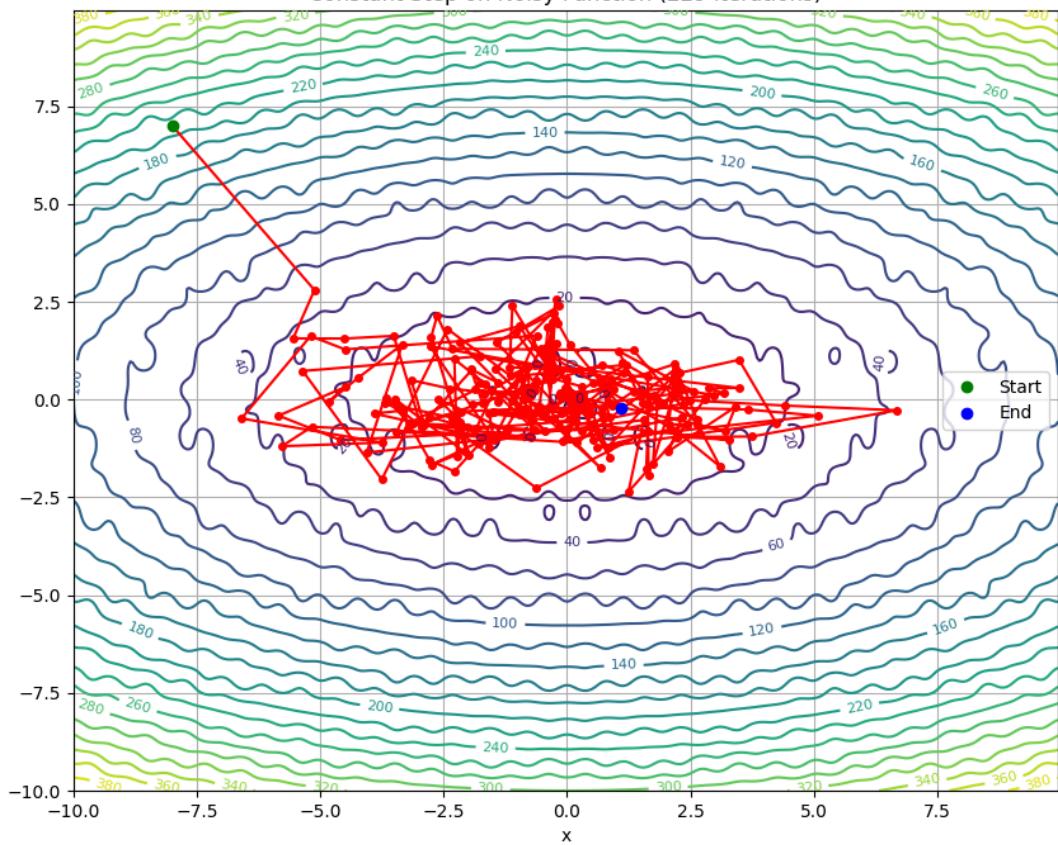
Видно, что steepest descent справляется все еще неплохо, но постоянный шаг страдает, уменьшающийся шаг также страдает, но заметно меньше и сходится в правильную сторону лучше

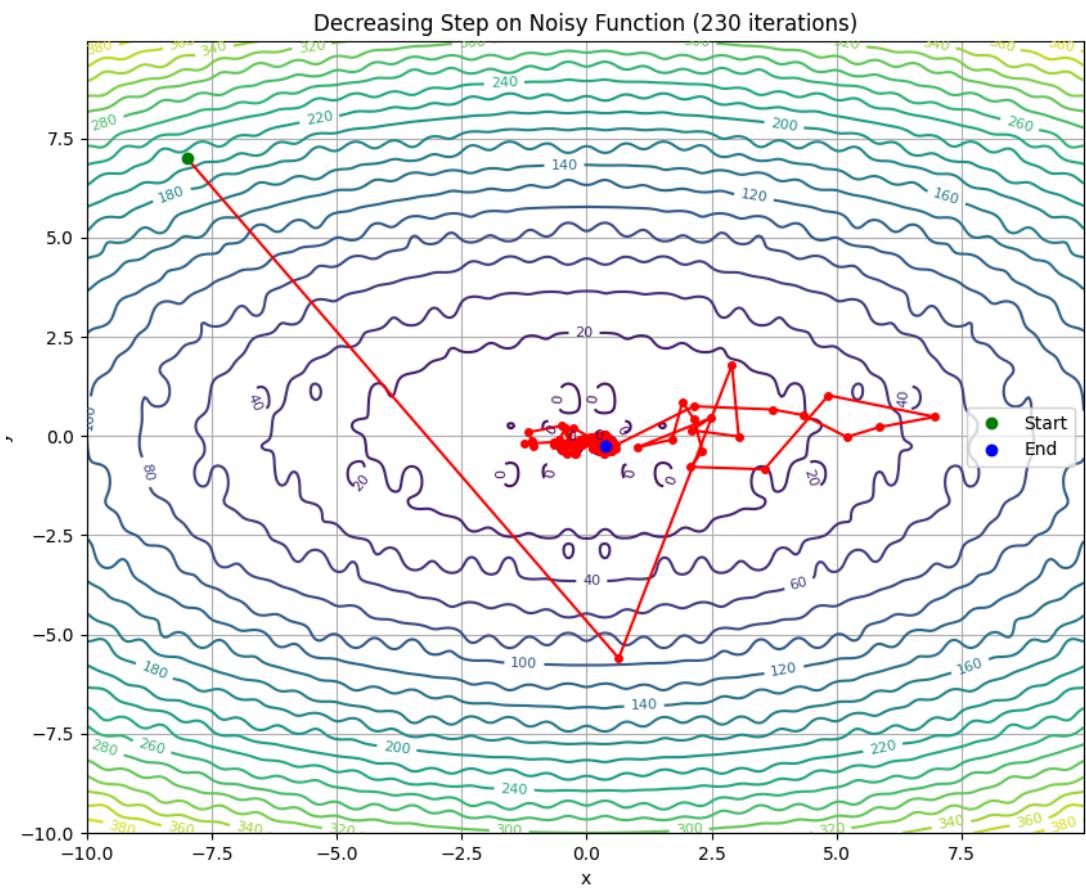
Это связано с тем, что градиент шума может быть очень большим, и поэтому если делать шаг константным, можно двигаться хаотично на большие расстояния, не обязательно в верную сторону

Steepest Descent with Dichotomy on Noisy Function (7 iterations)

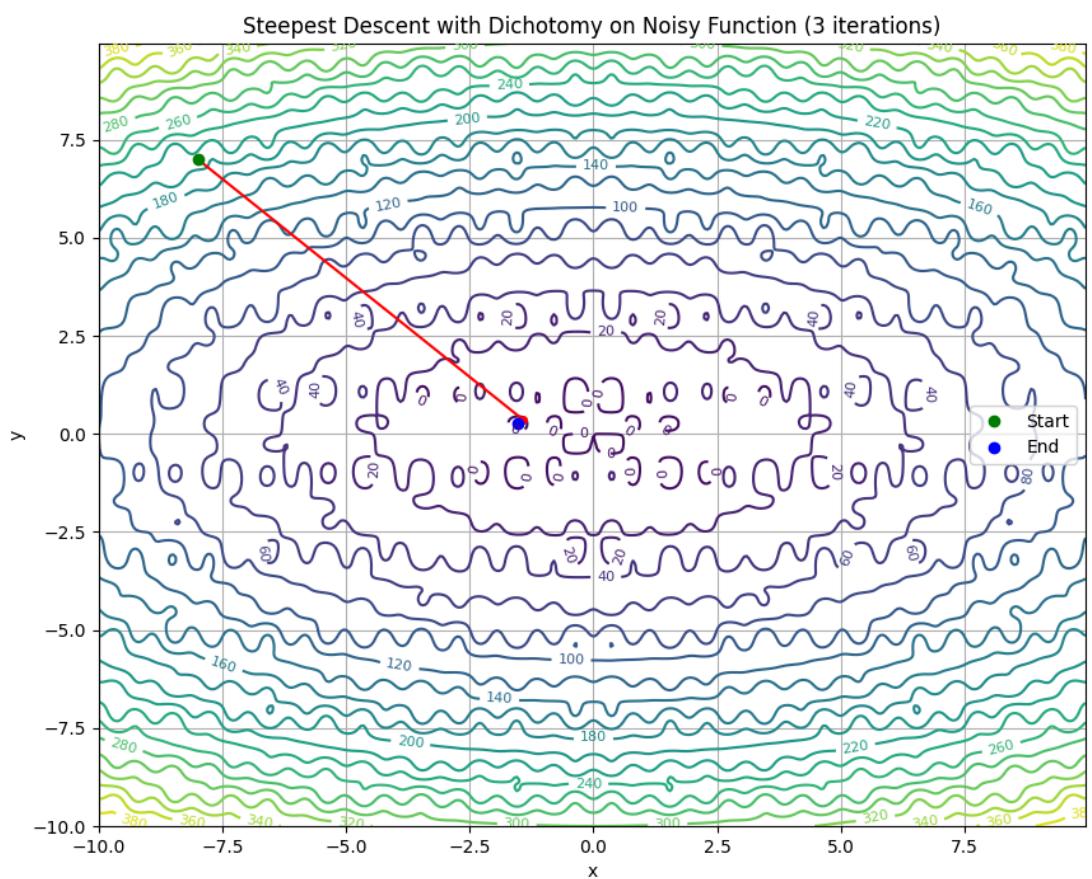
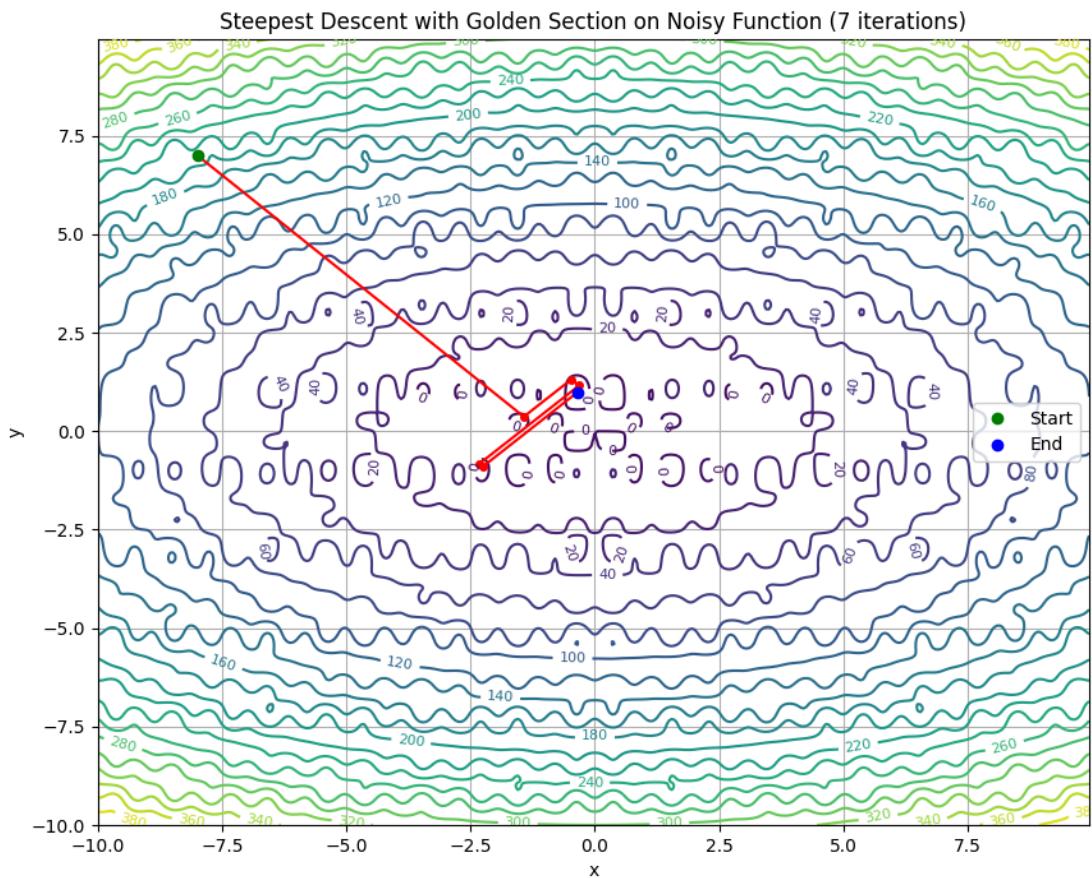


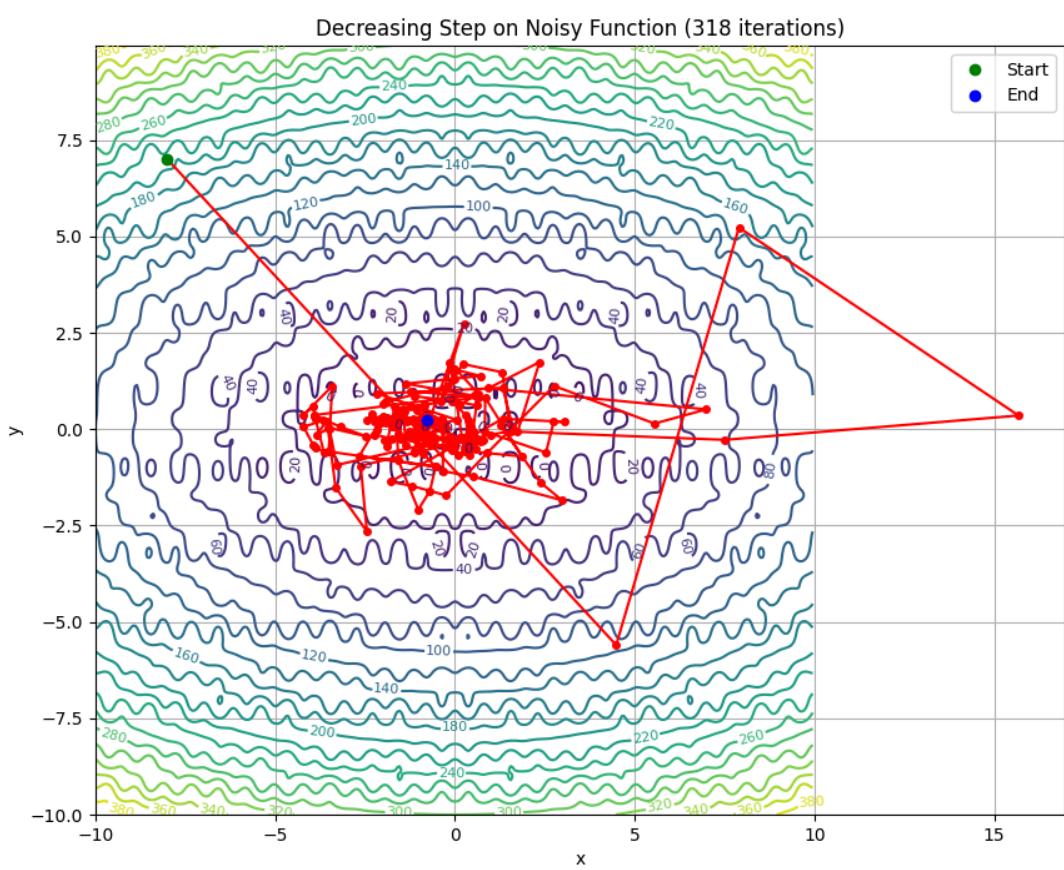
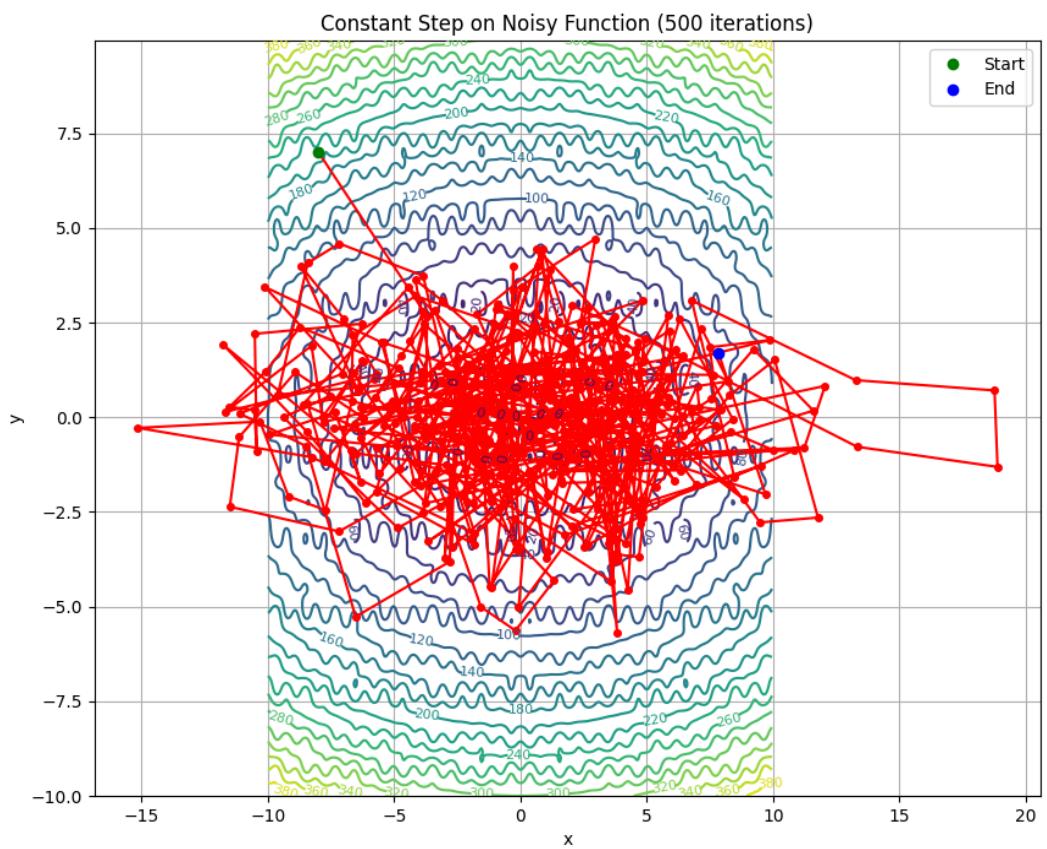
Constant Step on Noisy Function (229 iterations)





для $M = 20$ вышенназванные тенденции усиливаются:





Интересно, что на мультимодальной и зашумленной функции методы дихотомии и золотого сечения начинают давать разные результаты - они не рассчитаны на мультимодальность (а шум - это эффективно также мультимодальность), и их вызов похож на тернарный поиск в хаотичном массиве - он не обязан вернуть правильное значение

Тем не менее, они оба ведут себя в этих обстоятельствах не так уж плохо

Вывод.

Приведенные методы поиска рассчитаны на унимодальные функции с разумным значением градиента, поэтому на новых функциях они сработали плохо, особенно в случае, когда параметр, отвечающий за шум и мультимодальность, принимал большие значения.

При малых значениях параметра мультимодальности локальные минимумы были достаточно слабы, чтобы некоторые методы смогли избежать попадания туда, но при увеличении параметра все методы стали стабильно сходиться к близкому к стартовой точке локальному минимуму.

При малых значениях шума методы с различным выбором шага сходились достаточно близко к глобальному минимуму, но с увеличением шума (и как следствие, градиента) стали терять эффективность, лучше остальных себя показала функциональная зависимость, так как она уменьшает шаг, что в случаях резко скачущего градиента может быть полезно

Общая сравнительная таблица.

Сравнительная таблица всех приведенных методов на 3 типах функций - квадратичной, квадратичной зашумленной (для силы шума 20) и мультимодальной (для параметра $M = 2$):

===== Quadratic Function Results =====			
Method	Iterations	Function Calls	Gradient Calls
<hr/>			
Constant Step	19	19	19
Decreasing Step	12	12	12
Fast Decreasing Step	9	9	9
Golden Section	7	245	7
Dichotomy	7	293	7
SciPy	7	49	7
Analytical Gradient	7	245	7
Inertial Step	17	17	17

===== Noisy Function Results =====			
Method	Iterations	Function Calls	Gradient Calls
<hr/>			
Golden Section	7	245	7
Dichotomy	3	128	3
Inertial Step	500	500	500
Constant step	500	1000	1000
Decreasing step	318	1318	1318

===== Multimodal Function Results =====			
Method	Iterations	Function Calls	Gradient Calls
<hr/>			
Constant step	6	6	6
Golden Section	4	140	4
Dichotomy	4	164	4
Inertial Step	16	16	16

3. Изучение возможностей библиотеки `scipy.optimize`. Сравнение библиотечных методов и аналогов, реализованных нами

3.1 Введение

Цель данного этапа работы – провести всестороннее исследование возможностей библиотеки `scipy.optimize` для решения задач оптимизации, используя методы первого и нулевого порядка, а затем сравнить их с собственными реализациями методов оптимизации (градиентного спуска с различными стратегиями выбора шага и методов одномерного поиска). Рассматриваются не только числовые показатели (число итераций, вызовы целевой функции, вызовы градиента, время работы, остаточная ошибка), но и качество сходимости на различных типах функций: квадратичных (с хорошей и плохой обусловленностью), зашумленных и мультимодальных.

3.2 Обзор функционала `scipy.optimize`

Библиотека `scipy.optimize` предоставляет широкий спектр методов для решения задач оптимизации, что позволяет решать задачи как с доступной информацией о градиенте, так и без неё. Основной функцией является:

`scipy.optimize.minimize(fun, x0, method, jac, options)`

- **fun:** целевая функция для минимизации.
- **x0:** начальное приближение.
- **method:** метод оптимизации (например, "BFGS", "CG", "L-BFGS-B", "Powell", "Nelder-Mead").
- **jac:** функция для вычисления градиента (при использовании методов, требующих информацию о градиенте).
- **options:** параметры, задающие точность, максимальное число итераций и т.д.

3.3 Выбор аналогов для сравнения

Методы, использующие информацию о градиенте (первого порядка):

- **BFGS (Broyden–Fletcher–Goldfarb–Shanno):**

Квазиньютоновский метод, который аппроксимирует матрицу Гессе, что позволяет ускорить сходимость по сравнению с классическим градиентным спуском.

- **CG (Conjugate Gradient):**

Метод сопряженных градиентов, который не требует вычисления полной матрицы Гессе.

- **L-BFGS-B (Limited-memory BFGS with Box Constraints):**

Модификация BFGS с ограниченным использованием памяти, позволяющая работать с ограничениями переменных.

Методы, не использующие градиент (нулевого порядка):

- **Powell:**

Метод, основанный на линейном поиске вдоль системы сопряжённых направлений аппроксимируя заданную функцию квадратичным полиномом

- **Nelder-Mead:**

Метод, который не использует информацию о градиенте и ищет минимум, опираясь на изменение формы симплекса..

Дополнительный метод для одномерного поиска:

- **line_search:**

Функция `scipy.optimize.line_search`, которая может быть использована для нахождения оптимального шага вдоль заданного направления.

3.4 Постановка эксперимента

Для экспериментов были выбраны следующие тестовые функции:

1. Квадратичная функция с хорошей обусловленностью:

$$f(x, y) = x^2 + 3y^2$$

2. Квадратичная функция с плохой обусловленностью:

$$f(x, y) = 0.1x^2 + 3y^2$$

3. Зашумленная функция:

$$f(x, y) = 0.1x^2 + 3y^2 + \text{noise}(x, y)$$

Где к квадратичной функции добавляется синусоидальный шум..

4. Мультимодальная функция:

$$f(x, y) = 0.1x^2 + 0.1y^2 + M * (\sin(x - \frac{\pi}{2}) + \sin(x - \frac{\pi}{2}))$$

Здесь параметр M регулирует количество и глубину локальных минимумов.

Ключевые показатели

Для каждого метода будут собраны следующие метрики:

- Число итераций до сходимости.
- Количество вызовов целевой функции.
- Количество вызовов градиента.
- Затраченное время.
- Итоговое значение функции и остаточная ошибка.

3.5 Реализация экспериментов и сбор метрик

В качестве основы для экспериментов был разработан автоматизированный скрипт (см. код), который:

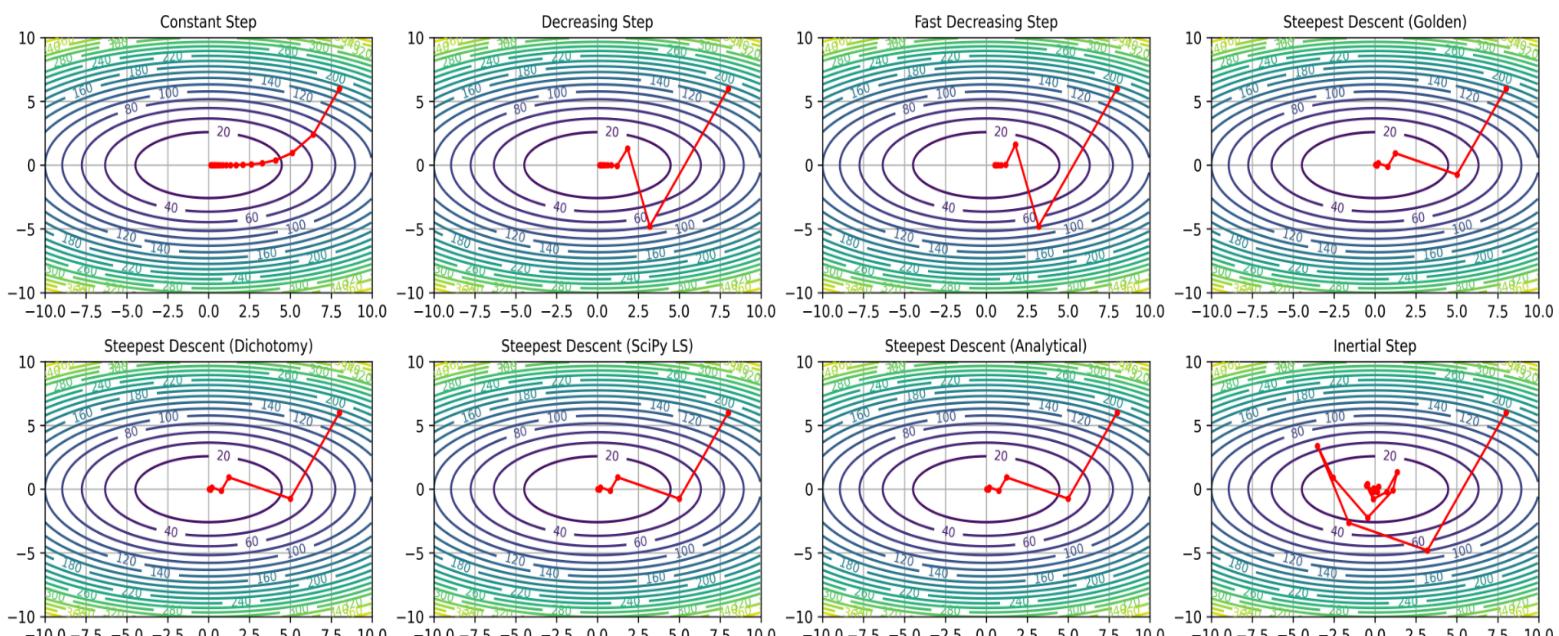
- Запускает эксперименты для каждой тестовой функции по всем методам (как собственным, так и библиотечным).
- Собирает метрики в виде словаря.
- Для каждого теста формирует «коллаж» с контурными графиками, на которых отображаются траектории оптимизации всех методов.

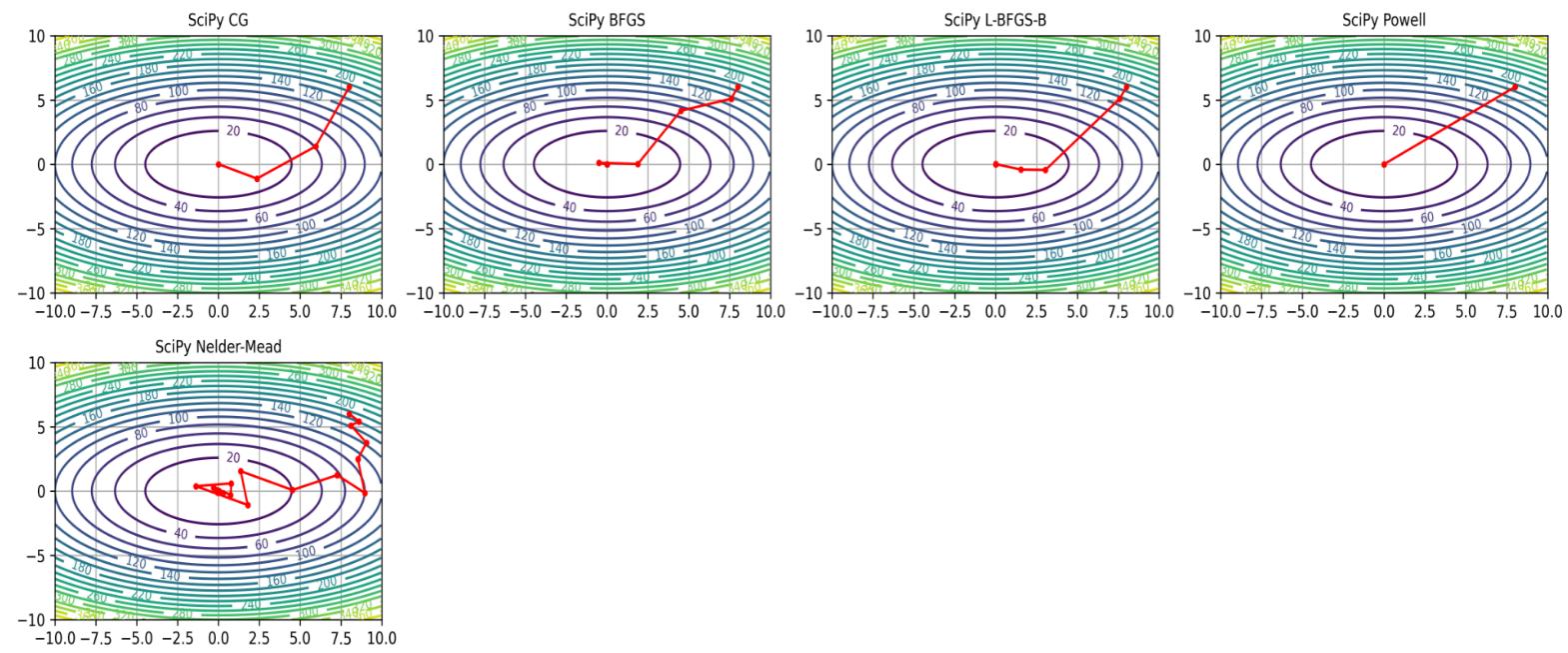
3.6 Сравнение и анализ результатов

3.6.1 Хорошо обусловленная функция

Method	Iter	FuncCalls	GradCalls	Time (sec)	Final Value
Constant Step	19	19	19	0.0000	2.0769e-02
Decreasing Step	12	12	12	0.0000	2.2686e-02
Fast Decreasing Step	9	9	9	0.0000	2.8503e-01
Steepest Descent (Golden)	7	245	7	0.0005	2.3992e-03
Steepest Descent (Dichotomy)	7	293	7	0.0012	2.3998e-03
Steepest Descent (SciPy LS)	7	63	7	0.0041	2.3998e-03
Steepest Descent (Analytical)	7	245	7	0.0010	2.3992e-03
Inertial Step	20	20	20	0.0000	1.3108e-03
SciPy CG	3	21	21	0.0040	1.2068e-14
SciPy BFGS	7	24	24	0.0062	1.7553e-15
SciPy L-BFGS-B	6	21	21	0.0051	2.1900e-12
SciPy Powell	2	34	0	0.0000	4.9698e-29
SciPy Nelder-Mead	50	94	0	0.0037	1.3876e-09

Good Quadratic Function - Methods Comparison





- **Собственные методы.**

При оптимизации хорошей квадратичной функции собственные методы продемонстрировали достаточно быструю сходимость. Методы на основе оптимального выбора шага достигли значения порядка 10^{-3} за 7 итераций, что свидетельствует о корректной реализации адаптивного поиска шага. Метод *Inertial Step* показал наименьшее итоговое значение, однако потребовал больше итераций, но меньше раз вызывал функцию. Постоянный шаг и методы с убывающим шагом потребовали немного большего числа итераций, а *Fast Decreasing Step* оказался менее эффективен.

- **Библиотечные методы.**

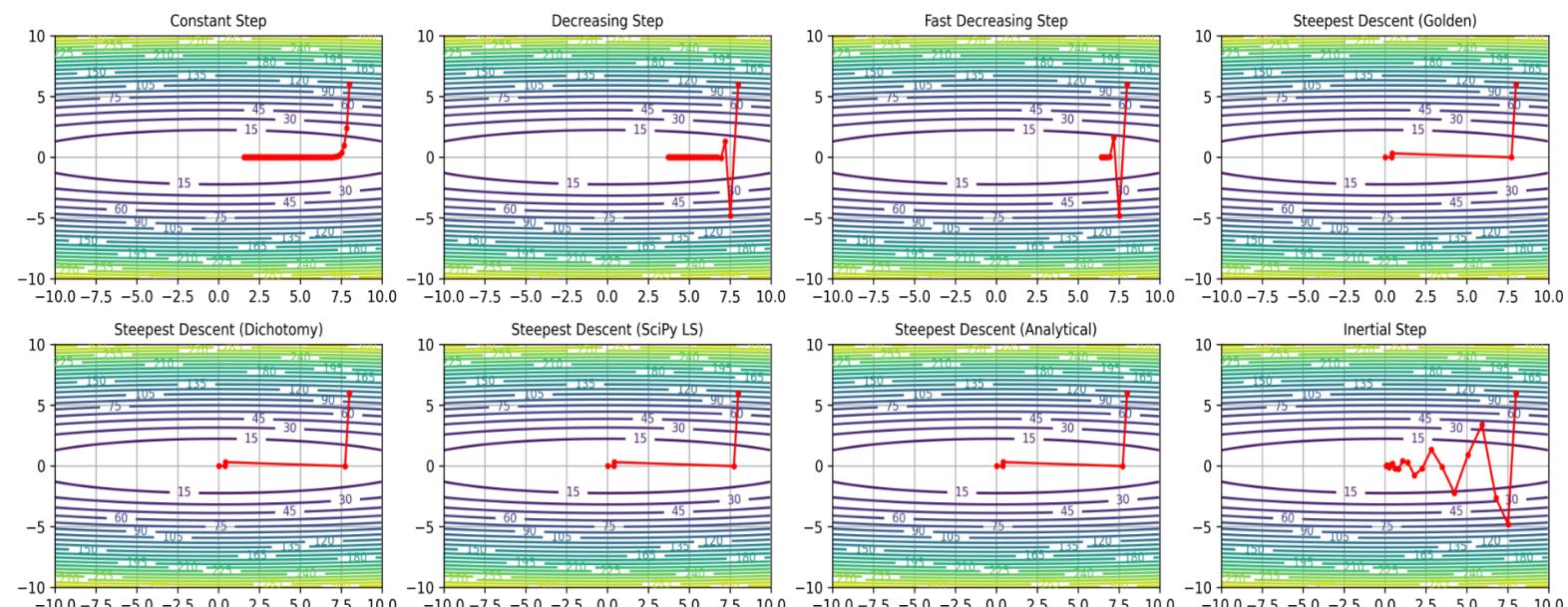
Методы CG, BFGS и L-BFGS-B показали превосходные результаты, минимальные вызовы функции и градиента, а итоговые значения функции почти нулевые. Powell продемонстрировал невероятно высокую точность при очень малом числе итераций, что подтверждает его эффективность для гладких задач, хотя его стратегия не использует градиент.

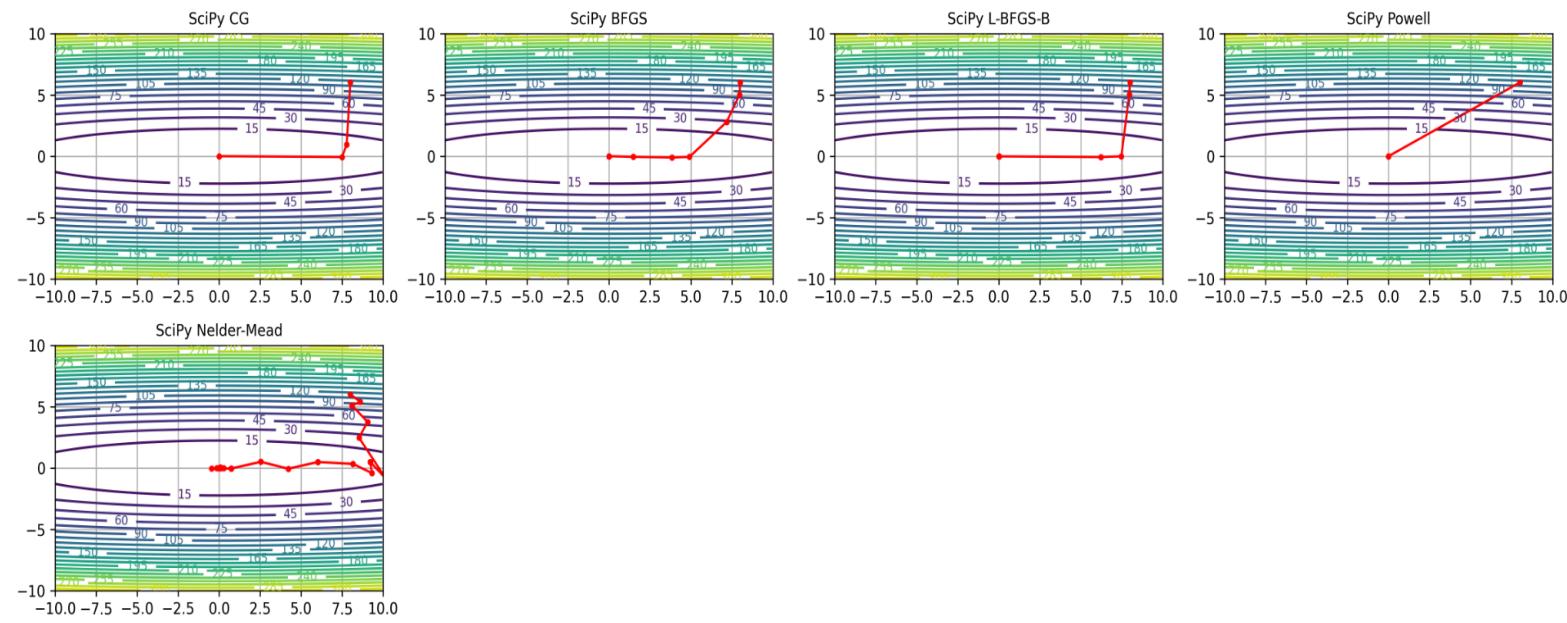
При оптимизации хорошо обусловленной квадратичной функции методы, использующие информацию о градиенте, однозначно превосходят собственные реализации по скорости и точности. Однако, адаптивные стратегии одномерного поиска (*Steepest Descent*) показывают достойные результаты, что подтверждает корректность реализации.

3.6.2 Плохо обусловленная функция

Method	Iter	FuncCalls	GradCalls	Time (sec)	Final Value
Constant Step	82	82	82	0.0000	2.4255e-01
Decreasing Step	50	50	50	0.0020	1.3861e+00
Fast Decreasing Step	10	10	10	0.0000	4.1215e+00
Steepest Descent (Golden)	5	175	5	0.0000	8.9259e-04
Steepest Descent (Dichotomy)	5	205	5	0.0000	8.4120e-04
Steepest Descent (SciPy LS)	5	47	5	0.0010	8.5416e-04
Steepest Descent (Analytical)	5	175	5	0.0000	8.9259e-04
Inertial Step	20	20	20	0.0000	6.8922e-04
SciPy CG	3	21	21	0.0028	4.4495e-15
SciPy BFGS	8	27	27	0.0031	6.9314e-16
SciPy L-BFGS-B	6	24	24	0.0030	4.2507e-14
SciPy Powell	2	34	0	0.0010	1.8654e-26
SciPy Nelder-Mead	50	95	0	0.0025	1.3913e-09

Bad Quadratic Function - Methods Comparison





- **Собственные методы.**

Для плохо обусловленной функции собственные методы показывают ухудшение результатов по сравнению с хорошей квадратичной функцией.

Метод Constant Step требует значительно больше итераций, в то время как методы, использующие адаптивный поиск шага, стабильно сходятся за 5 итераций, достигая меньших значений. Это говорит о том, что адаптивный выбор шага особенно важен для плохо обусловленных задач. Метод Inertial Step также сходится, но не даёт значительного улучшения по итоговому значению.

- **Библиотечные методы.**

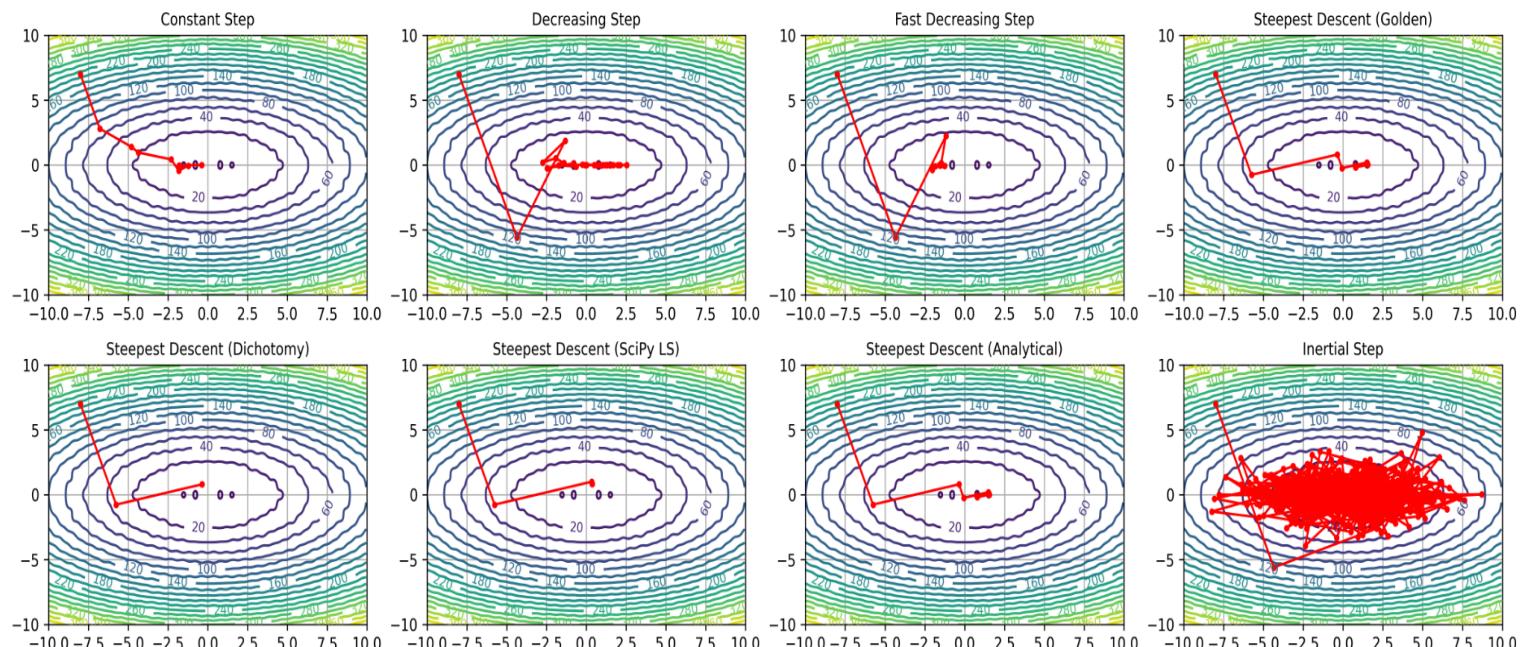
Библиотечные алгоритмы, показывают очень высокую точность и быстро сходятся (3–8 итераций), независимо от плохой обусловленности. Powell снова демонстрирует наивысшую точность, а Nelder-Mead уступает по эффективности.

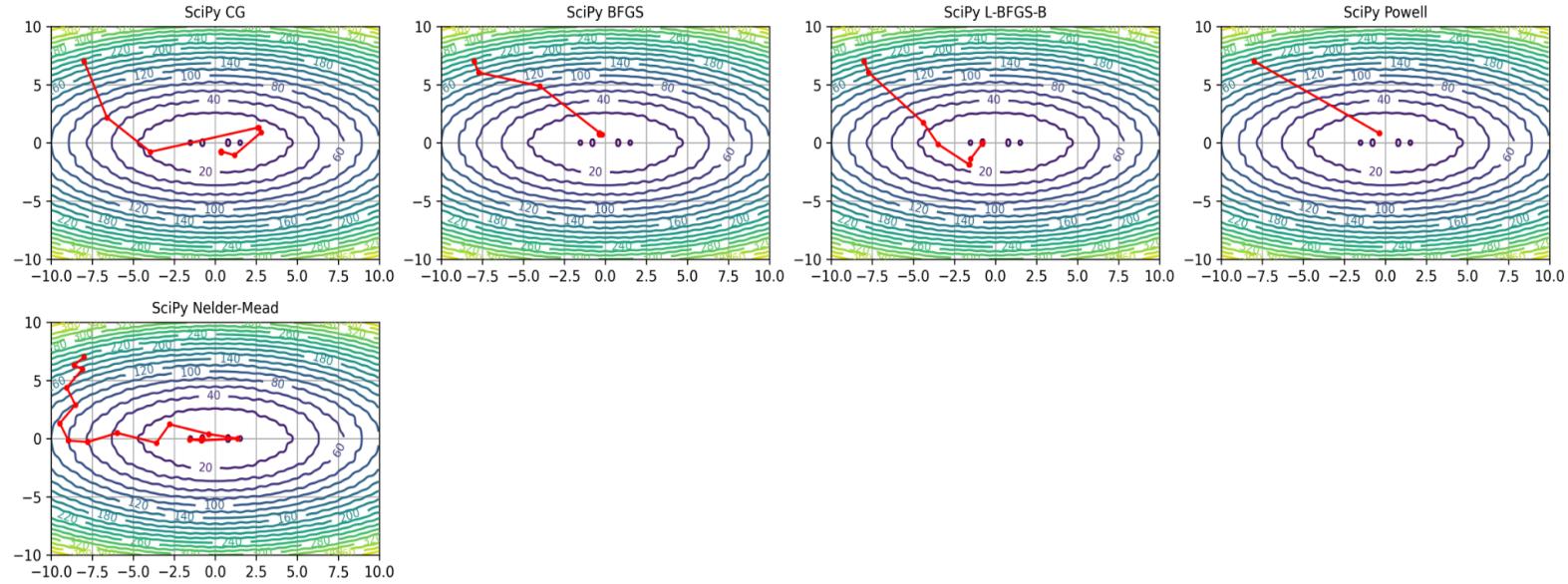
При плохой обусловленности критически важен адаптивный выбор шага. Собственные методы на основе Steepest Descent обеспечивают быстрый и стабильный результат, но по числовым метрикам SciPy остается лидером. Это подтверждает, что для плохо обусловленных задач лучше применять методы, способные динамически корректировать шаг.

3.6.3 Зашумленная функция

Method	Iter	FuncCalls	GradCalls	Time (sec)	Final Value
<i>Constant Step</i>	9	9	9	0.0010	2.7847e+00
<i>Decreasing Step</i>	27	27	27	0.0014	4.0075e+00
<i>Fast Decreasing Step</i>	13	13	13	0.0000	-5.8677e-01
<i>Steepest Descent (Golden)</i>	10	350	10	0.0030	-5.5828e-01
<i>Steepest Descent (Dichotomy)</i>	3	125	3	0.0010	9.6689e-02
<i>Steepest Descent (SciPy LS)</i>	5	82	5	0.0027	9.9172e-02
<i>Steepest Descent (Analytical)</i>	10	350	10	0.0030	-5.5828e-01
<i>Inertial Step</i>	500	500	500	0.0188	1.1926e+01
<i>SciPy CG</i>	14	99	99	0.0078	9.3790e-02
<i>SciPy BFGS</i>	10	39	39	0.0026	9.3790e-02
<i>SciPy L-BFGS-B</i>	11	60	60	0.0105	-7.0754e-01
<i>SciPy Powell</i>	3	73	0	0.0025	9.3790e-02
<i>SciPy Nelder-Mead</i>	55	107	0	0.0031	-7.0754e-01

Noisy Function - Methods Comparison





- **Собственные методы.**

На зашумленной функции собственные реализации демонстрируют заметное увеличение числа итераций и вызовов. Значения итоговой функции варьируются от положительных до отрицательных, что свидетельствует о влиянии шума на сходимость, ухудшающего стабильность градиентных вычислений. Метод *Inertial Step* потребовал 500 итераций и дал значительно неверный результат, что указывает на его низкую устойчивость при высоком уровне шума.

- **Библиотечные методы.**

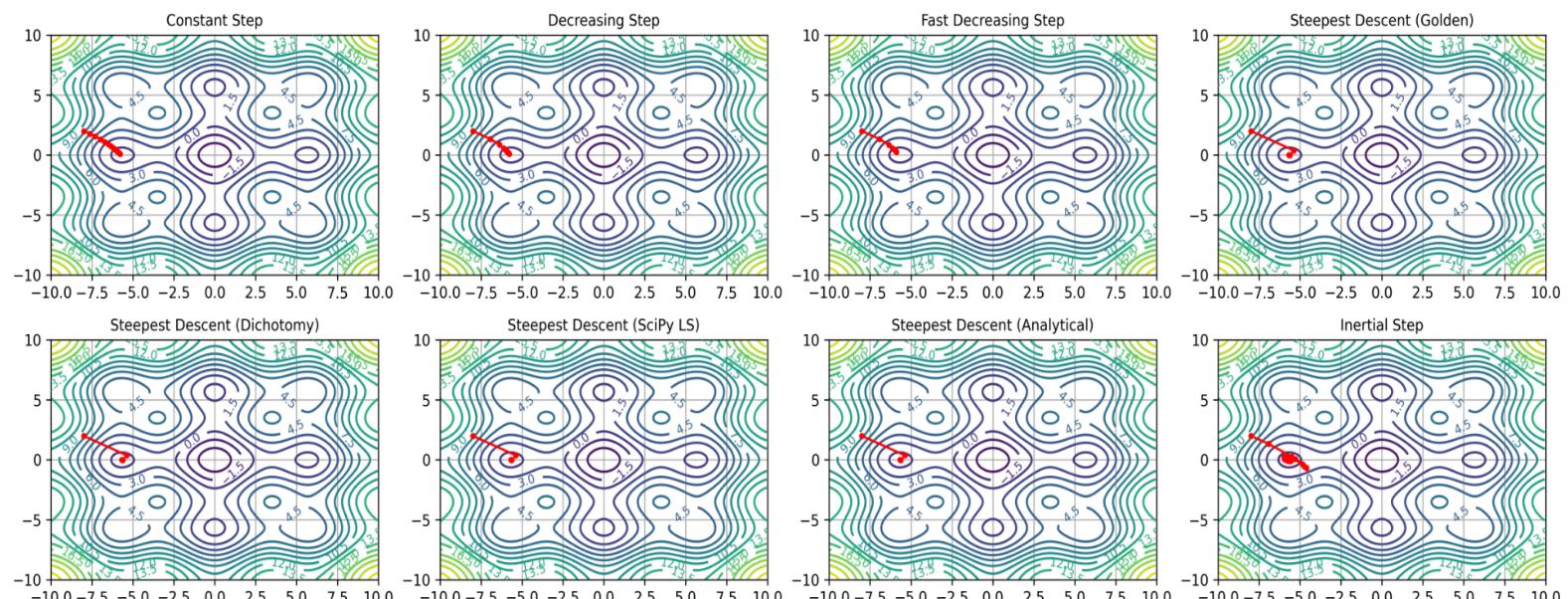
Среди библиотечных методов на зашумленной функции CG и BFGS показали относительную стабильность, в то время как L-BFGS-B и Nelder-Mead дали отрицательные значения, свидетельствуя о возможном смещении. Метод Powell (без градиента) также продемонстрировал хорошую устойчивость.

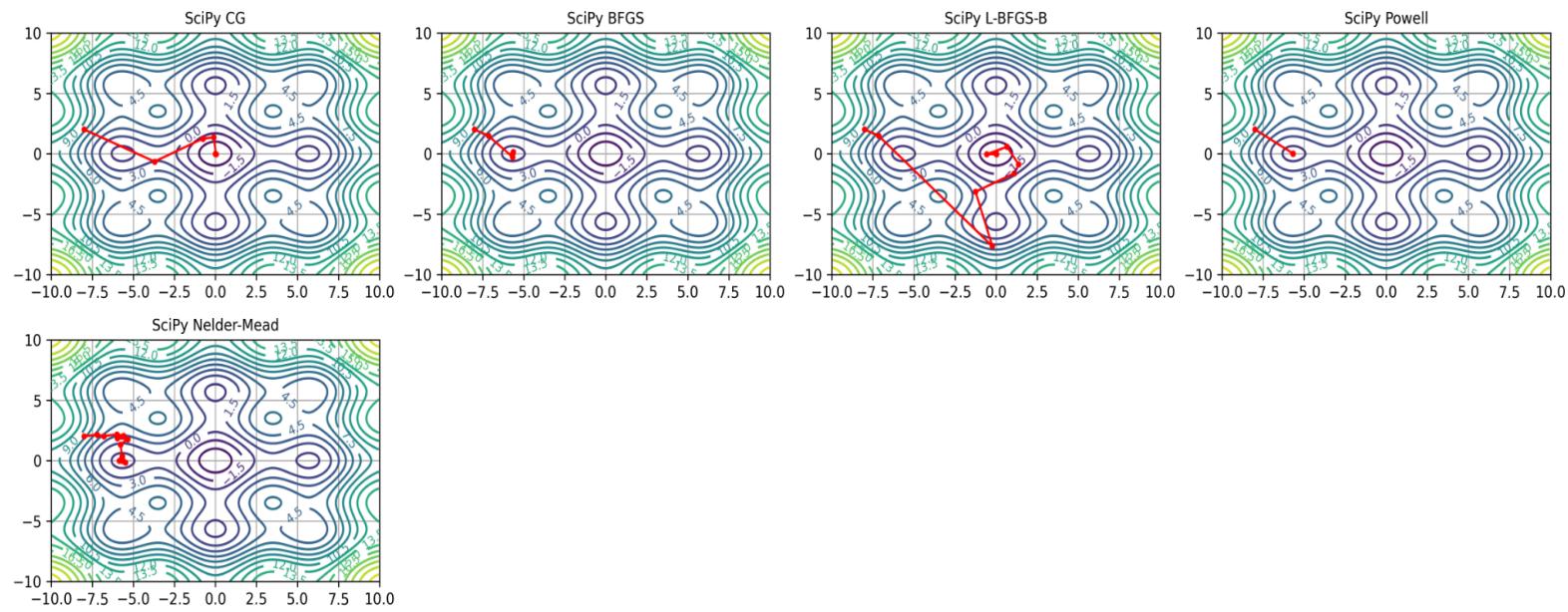
Шум существенно ухудшает эффективность большинства методов, особенно тех, что полагаются на точные вычисления градиента. В таких условиях библиотечные методы (особенно CG, BFGS и Powell) показывают лучшие результаты по стабильности, тогда как собственные методы требуют дополнительной настройки для работы с шумом.

3.6.4 Мультимодальная функция, $M = 2$

Method	Iter	FuncCalls	GradCalls	Time (sec)	Final Value
<i>Constant Step</i>	15	15	15	0.0000	-3.9767e-01
<i>Decreasing Step</i>	10	10	10	0.0000	-4.0444e-01
<i>Fast Decreasing Step</i>	8	8	8	0.0000	-3.0295e-01
<i>Steepest Descent (Golden)</i>	4	140	4	0.0011	-4.2080e-01
<i>Steepest Descent (Dichotomy)</i>	4	164	4	0.0005	-4.2080e-01
<i>Steepest Descent (SciPy LS)</i>	4	47	4	0.0020	-4.2080e-01
<i>Steepest Descent (Analytical)</i>	4	140	4	0.0000	-4.2080e-01
<i>Inertial Step</i>	16	16	16	0.0014	-4.1861e-01
SciPy CG	8	69	69	0.0060	-4.0000e+00
SciPy BFGS	6	21	21	0.0027	-4.2083e-01
SciPy L-BFGS-B	12	42	42	0.0124	-4.0000e+00
SciPy Powell	2	39	0	0.0020	-4.2083e-01
SciPy Nelder-Mead	43	83	0	0.0037	-4.2083e-01

Multimodal Function - Methods Comparison





- **Собственные методы.**

На мультимодальной функции собственные реализации, особенно варианты *Steepest Descent* (все три варианта), сходятся очень быстро – всего за 4 итерации, достигая итогового значения около $-4.2080e-01$. Это свидетельствует о том, что адаптивные методы одномерного поиска эффективно находят оптимальный шаг даже в присутствии нескольких локальных минимумов. Постоянный шаг и методы с убывающим шагом дали чуть менее точные результаты, а метод *Inertial Step* требует большего числа итераций, но всё равно сходится к значению, близкому к оптимальному.

- **Библиотечные методы.**

Среди библиотечных методов наиболее стабильными оказались *BFGS*, *L-BFGS-B* и *Powell*. Методы *CG* и *L-BFGS-B* показали смещение (, что может свидетельствовать о том, что алгоритм попал в локальный минимум, не совпадающий с ожидаемым оптимумом.

В задачах с множественными минимумами собственные методы на основе *Steepest Descent* демонстрируют высокую эффективность и быструю сходимость, в то время как некоторые библиотечные методы (*CG*, *L-BFGS-B*) могут давать смещенные результаты. *Powell* и *BFGS* показали хорошую точность, однако, устойчивость решения зависит от начального приближения и характера функции.

3. 7 Анализ результатов и выводы

В проведённом исследовании сравнительный анализ методов оптимизации, реализованных с использованием библиотеки `scipy.optimize` и собственных реализаций, показал следующие ключевые моменты:

- 1. Методы, использующие информацию о градиенте (CG, BFGS, L-BFGS-B):**

Эти алгоритмы демонстрируют высокую скорость сходимости и выдающуюся точность на гладких функциях, как для хорошо, так и для плохо обусловленных задач. Благодаря использованию градиентной информации, они достигают минимальных значений функции за несколько итераций, что подтверждается малыми итоговыми ошибками.

- 2. Адаптивные стратегии одномерного поиска (Steepest Descent с золотым сечением, дихотомией, line search):**

Собственные реализации, использующие адаптивное определение оптимального шага, демонстрируют стабильную сходимость и высокую точность, особенно при оптимизации квадратичных функций. Однако такие подходы требуют большего числа вызовов целевой функции из-за итеративного поиска оптимального шага, что является компромиссом между гибкостью метода и вычислительными затратами.

- 3. Методы нулевого порядка (Powell, Nelder-Mead):**

Данные алгоритмы, не использующие информацию о градиенте, особенно полезны в задачах, где градиентная информация недоступна или трудна для вычисления. Несмотря на более высокое число вызовов функции и относительно медленную сходимость по сравнению с градиентными методами, они показывают хорошую точность на гладких задачах и могут быть эффективны при решении сложных задач, где градиент не вычисляется.

- 4. Влияние характеристик задачи:**

- **Гладкие функции с хорошей обусловленностью:** Библиотечные методы (CG, BFGS, L-BFGS-B) являются оптимальными, а собственные адаптивные реализации также демонстрируют хорошие результаты.

- **Плохо обусловленные функции:** Адаптивные стратегии одномерного поиска существенно улучшают сходимость по сравнению с методами с постоянным шагом, что критически важно для таких задач.
- **Зашумленные задачи:** Наличие шума существенно влияет на стабильность вычислений. Здесь библиотечные методы (особенно CG, BFGS и Powell) показывают более высокую устойчивость, в то время как собственные реализации требуют дополнительной настройки для компенсации влияния шума.
- **Мультимодальные функции:** В задачах с множественными локальными минимумами особую роль играет корректный выбор шага. Собственные методы на основе Steepest Descent быстро сходятся, тогда как некоторые библиотечные методы могут застревать в неподходящих локальных минимумах.

Практические рекомендации:

Выбор метода оптимизации должен опираться на особенности решаемой задачи. Для гладких задач с доступной информацией о градиенте предпочтительны методы CG, BFGS и L-BFGS-B. В задачах, где градиентная информация недоступна или вычислительно затратна, следует использовать методы нулевого порядка, такие как Powell или Nelder-Mead. Для сложных функций с шумом и множественными минимумами рекомендуется экспериментировать с адаптивными стратегиями одномерного поиска для достижения наилучшей устойчивости и точности.

Заключение

Проведённое исследование подтверждает, что библиотечные методы из `scipy.optimize`, особенно алгоритмы, использующие информацию о градиенте, существенно превосходят собственные реализации по скорости и точности при решении гладких оптимизационных задач. Однако собственные методы, основанные на адаптивном выборе шага, демонстрируют высокую гибкость и могут успешно применяться в задачах с плохой обусловленностью и мультимодальностью. Оптимальный выбор метода зависит от характеристик задачи, и для сложных условий (например, при наличии шума) целесообразно проводить дополнительную настройку гиперпараметров.