MetOpt1

Дроздов Олег, Широков Данил, Исламова Камиля

1 апреля 2025 г.

1 Введение

В своей работе мы хотим найти экстремумы различных функций от двух переменных с помощью различных методов поиска минимумов и максимумов. Методы работают на основе градиентного спуска. Также мы сравним полученные результаты с встроенными библиотечными функциями из библиотеки scipy.optimize.

2 Hyperпараметры

- learning rate — дефолтное значение шага.
- max iterations — максимальное количество итераций, которое алгоритм будет выполнять во время поиска минимума функции
- lr method const — константа, используемая в некоторых методах выбора размера шага (например, в методе Армихо, экспоненциального уменьшения и других)
- lr method — метод для выбора размера шага. Подробнее дальше
- tolerance — пороговое значение, используемое для определения, когда алгоритм должен остановиться. Если изменение между итерациями становится меньше этого значения, считается, что алгоритм сошелся. В нашем случае мы выбрали значение = 1e-6

3 Класс градиентного спуска

3.1 Функция compute gradient

Эта функция вычисляет градиент заданной функции в определенной точке: происходит определение переменных, парсинг функции, вычисление частных производных, конвертая частных производных в функции, и в итоге вычисление самого градиента.

3.2 Функция solve

Эта функция выполняет итеративный процесс градиентного спуска для минимизации заданной функции. В основном цикле выполняется следующее (выполняется пока не достигнуто максимальное количество итераций или не выполнено условие градиентного спуска): вычисление градиента и его нормализация, определение размера шага (в зависимости от метода) и обновление точки по формуле:

$$point_k = point - step * grad \tag{1}$$

3.3 Метод plot descent

Этот метод визуализирует путь градиентного спуска в 3D. Он строит график, на котором отображаются все точки, через которые проходил алгоритм во время поиска минимума.

4 Используемые методы

4.1 Фиксированный шаг

Используется постоянное значение на всеъ итерациях

$$step = learning\_rate \tag{2}$$

4.2 Методы для адаптивного выбора шага

4.2.1 Метод Армиджо (armijo)

Вычисляются значения функции в текущей и новой точках (по формуле из solve). Если новое значение функции меньше, чем $f_{current} + const * alpha * (-G)$, то шаг уменьшается вдвое.

Линейная аппроксимация и достаточное убывание:

Разложим $f(x_{k+1})$ в ряд Тейлора вокруг x_k:

$$f(x_k - \alpha \nabla f(x_k)) \approx f(x_k) - \alpha \|\nabla f(x_k)\|^2 + .$$

Если отбросить высшие члены, получаем:

$$f(x_{k+1}) f(x_k) - \alpha \|\nabla f(x_k)\|^2.$$

Но поскольку аппроксимация неточная, вводим коэффициент ослабления c:

$$f(x_{k+1}) \leq f(x_k) - c \cdot \alpha \|\nabla f(x_k)\|^2.$$

Смысл: Мы требуем, чтобы реальное убывание f было хотя бы c-долей от предсказанного линейной моделью. Это гарантирует, что шаг не слишком большой (чтобы не перескочить минимум) и не слишком маленький (чтобы не тормозить сходимость).

4.2.2 Экспоненциальное затухание (exp decay)

Этот метод уменьшает скорость обучения по экспоненциальному закону с каждой итерацией:

$$step = learning\_rate * e^{-const * iteration} \tag{3}$$

4.2.3 Затухание по времени (dec time)

Этот метод также уменьшает скорость обучения с увеличением номера итерации, но по линейному закону:

$$step = learning\_rate / 1 + const * iteration \tag{4}$$

4.3 Одномерные поиски

4.3.1 Золотое сечение (golden ration)

Ищет минимум функции на заданном интервале [a, b] с использованием отношения золотого сечения

1. Шаг золотого сечения

$$\phi = \frac{sqrt(5) + 1}{2} \tag{5}$$

2. Сравниваем значение в двух точках c и d. Выбираем такой интервал, чтобы минимум оставался внутри.

$$c = b - \frac{b - a}{\phi} \tag{6}$$

$$d = a + \frac{b - a}{\phi} \tag{7}$$

3. Процесс повторяется до тех пор, пока длина отрезка не станет меньше заданного порога (epsilon).

4.3.2 Дихотомия (dichotomy)

Метод дихотомии также является методом одномерной оптимизации. Он использует подход "разделяй и властвуй"для нахождения оптимального шага на изначально заданном отрезке [a, b]

1. Находится средняя точка (m) и две точки:

$$c = m - delta; d = m + delta \tag{8}$$

2. Значения функции в этих точках сравниваются, и границы обновляются в зависимости от того, где находится минимум.

4.4 Библиотечный метод

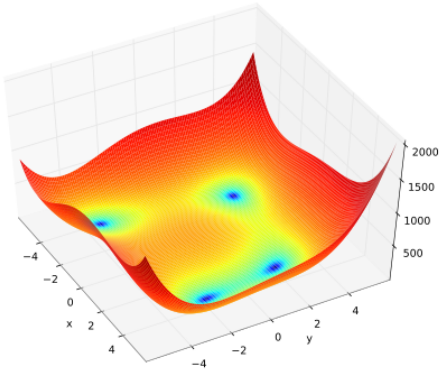
Мы решили использовать функцию minimize из библиотеки scipy.optimize (method='BFGS')

5 Исследумые функции

5.1 Функция Химмельблау

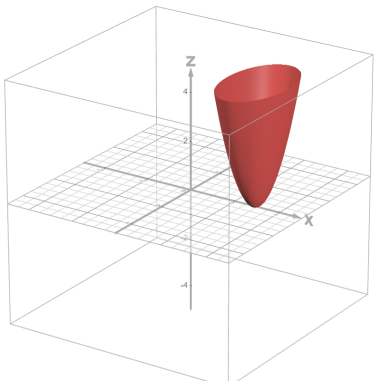
Пример мультимодальной функции. Имеет четыре равнозначных локальных минимума в точках: $(3; 2); (\approx -2,8; \approx 3,13); (\approx -3,77; \approx -3,28); (\approx 3,58; \approx -1,84)$

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{9}$$



5.2 Парабалоид

$$f(x, y) = 3(x - 3)^2 + y^2 \tag{10}$$

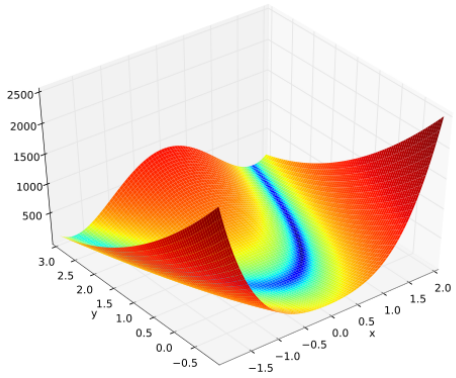


5.3 Функция Розенброка

Невыпуклая функция, используемая для оценки производительности алгоритмов оптимизации, предложенная Ховардом Розенброком в 1960 го-

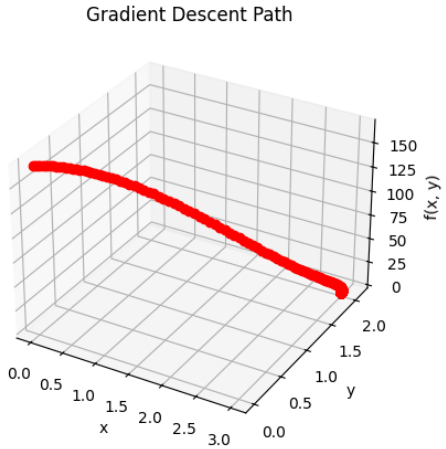
ду. Считается, что поиск глобального минимума для данной функции является нетривиальной задачей. Имеет минимум 0 в точке (1,1)

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{11}$$

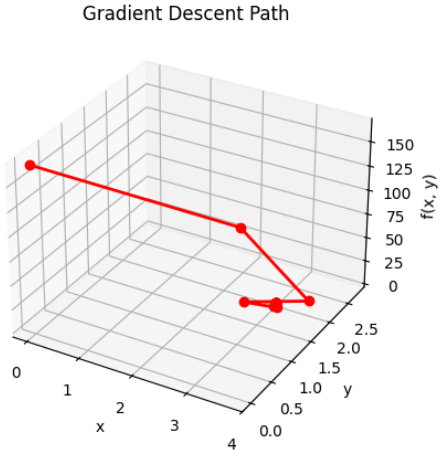


6 Графики

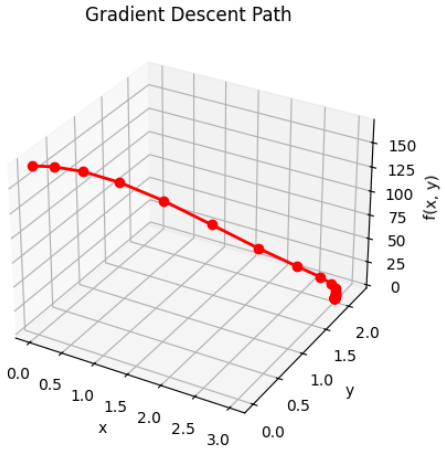
6.1 Функция Химмельблау



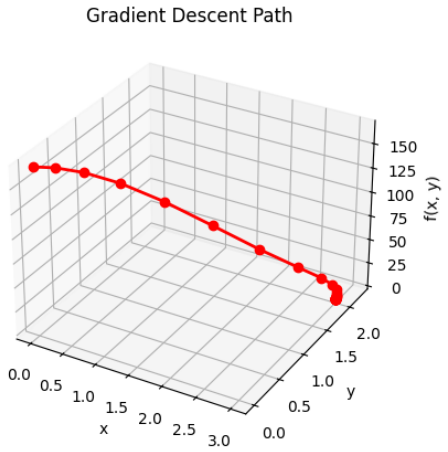
- fixed



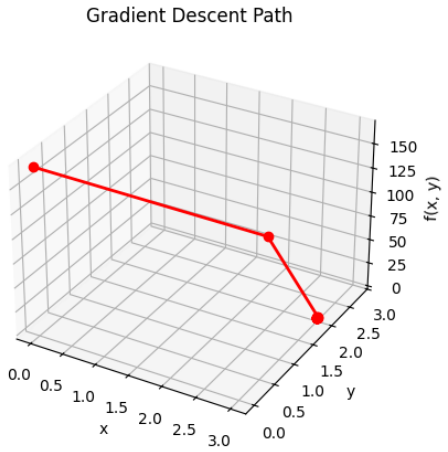
- armijo



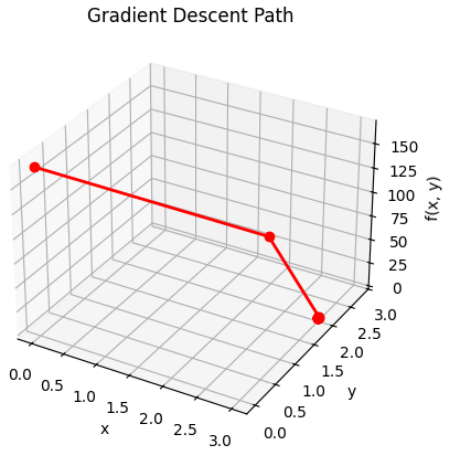
- exp decay



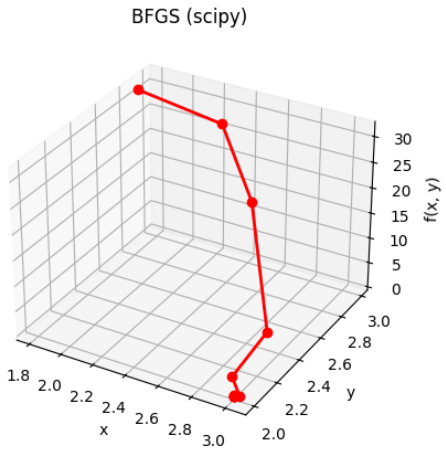
- dec time



- golden ratio

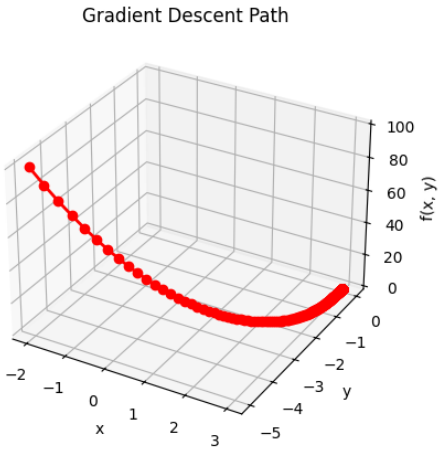


- dichotomy

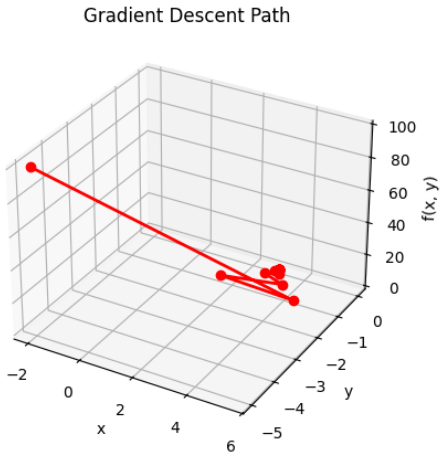


- BFGS

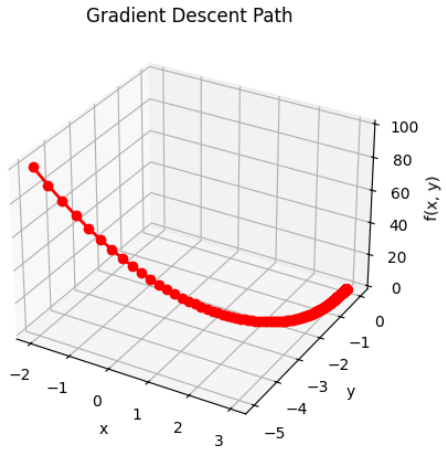
6.2 Парабалоид



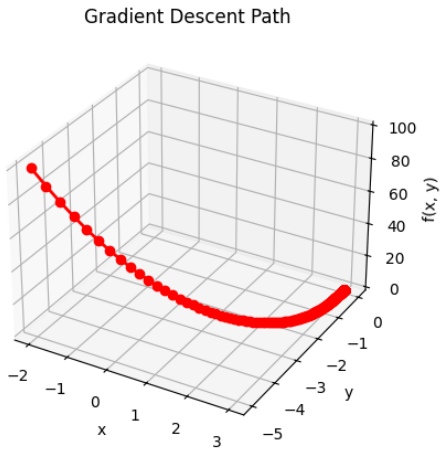
- fixed



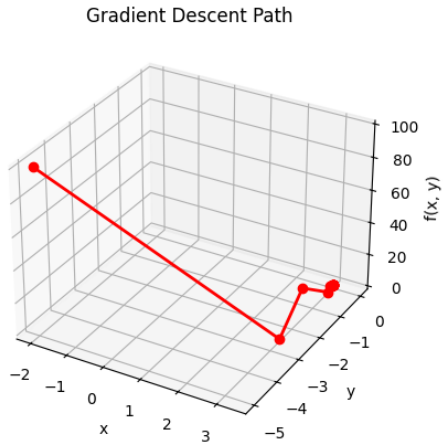
- armijo



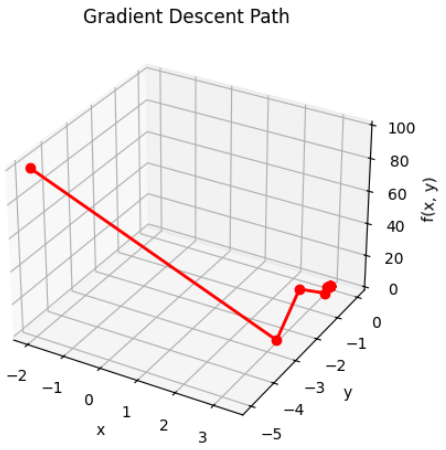
- exp decay



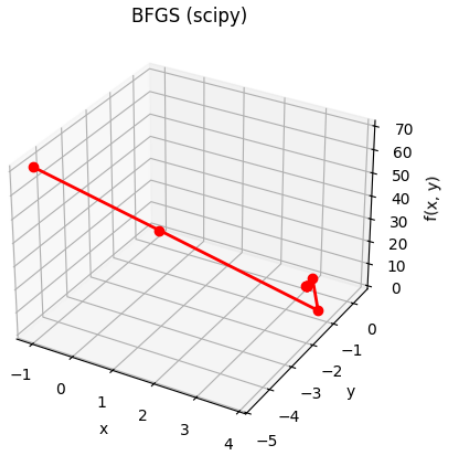
- dec time



- golden ratio

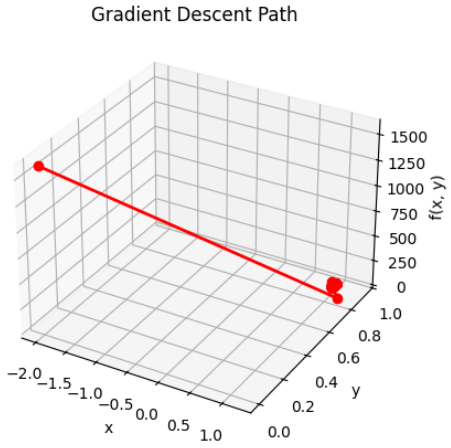


- dichotomy

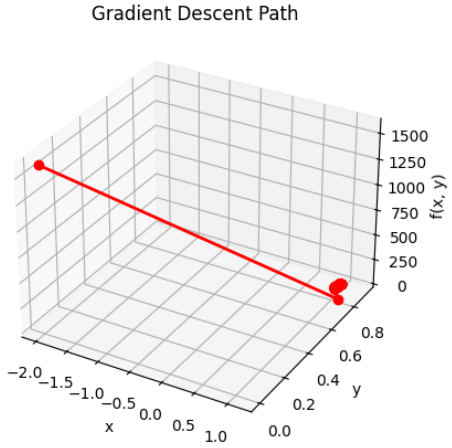


- BFGS

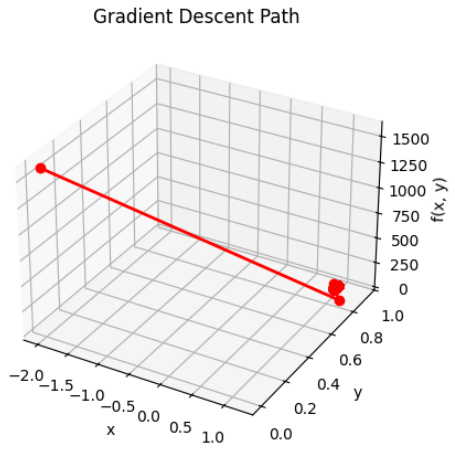
6.3 Функция Розенброка



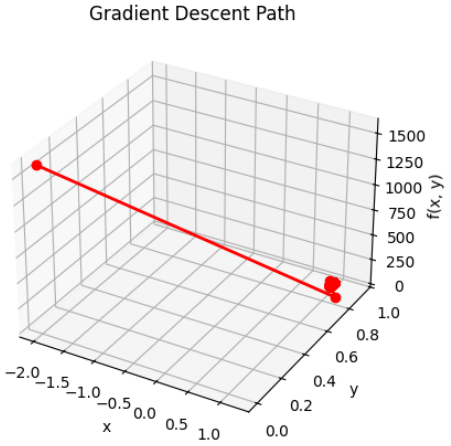
- fixed



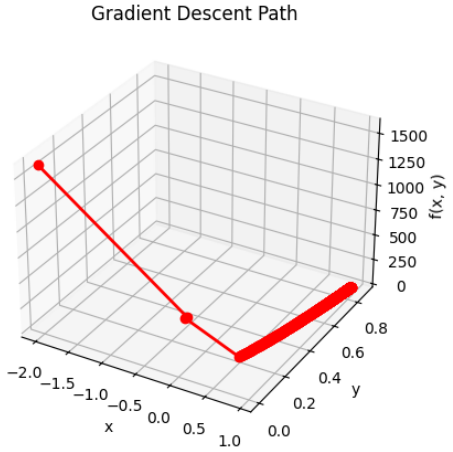
- armijo



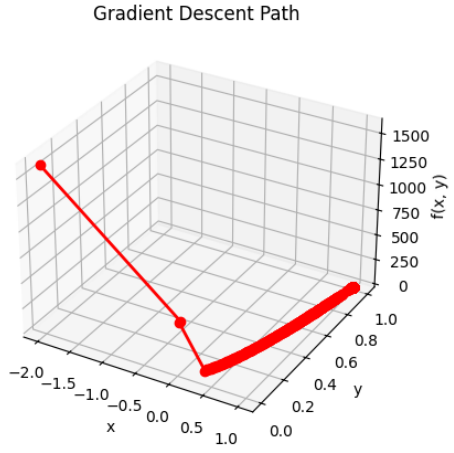
- exp decay



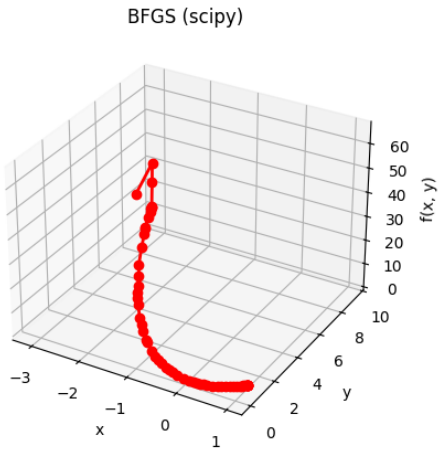
- dec time



- golden ratio



- dichotomy



- BFGS

7 Таблица

(0, 0) - Начальная точка	Функция Химмельблау	(x^2 + y -11)^2 + (x + y^2 - 7)		
Метод	Результат (x, y)	Итерации	Расчеты градиента	Расчеты функции
fixed	(2.999985305742581, 2.0000354735403634)	408	408	0
armijo	(3.0000006206981995, 1.9999985015000346)	27	54	33
exp_decay	(2.9999988887987845, 2.0000026826696637)	48	48	0
dec_time	(2.9999988887988702, 2.0000026826694564)	48	48	0
golden_ratio	(3.0000000459797103, 2.0000015213307583)	17	17	442
dichotomy	(2.9999994959822485, 2.000000723312923)	18	18	648
BFGS	(3.00000000000092, 1.9999999999969098)	12	18	18
(-2, 0) - Начальная точка	Функция Розенброка	(1-x)^2 + 100*(y-x^2)^2		
Метод	Результат (x, y)	Итерации	Расчеты градиента	Расчеты функции
fixed	(0.9660887307077805, 0.933188943723161)	1000	1000	0
armijo	(0.9551985017511018, 0.9122200132595994)	1000	2000	1010
exp_decay	(0.9660471500814515, 0.9331084310332342)	1000	1000	0
dec_time	(0.9660471521613984, 0.9331084350605552)	1000	1000	0
golden_ratio	(0.9323962744864359, 0.8693040693545461)	1000	1000	26000
dichotomy	(0.9994570845488654, 0.9989151533461795)	1000	1000	36000
BFGS	(1.0000000000176987, 1.0000000000341227)	57	71	71
(-2, -5) - Начальная точка	Параболоид	3(x-3)^2 + y^2		
Метод	Результат (x, y)	Итерации	Расчеты градиента	Расчеты функции
fixed	(3.0000000000000000, -4.9869e-05)	571	571	0
armijo	(3.0000005960464478, -5.9605e-07)	24	48	26
exp_decay	(3.0000000000000000, -4.9927e-05)	571	571	0
dec_time	(3.0000000000000000, -4.9927e-05)	571	571	0
golden_ratio	(2.9999991889819375, -8.1108e-07)	15	15	390
dichotomy	(2.9999991889319086, -8.1115e-07)	15	15	540
BFGS	(3.0000000004807, 7.877500640285836e-10)	9	10	10

8 Выводы по методам

- Фиксированный шаг. Это самый простой подход, но он может не всегда работать эффективно, особенно если функция имеет сильно наклонные участки или "плоские"области.
- Метод Армиджо. Может делать слишком маленькие шаги, замедляя сходимость. Требует дополнительных вычислений функции (на каждой проверке условия). Но при этом показывает стабильную сходимость, шаг всегда выбирается так, чтобы функция уменьшилась.
- Экспоненциальное затухание: начальная скорость обучения уменьшается с увеличением номера итерации. Это позволяет алгоритму

делать большие шаги в начале, когда он еще далеко от минимума, и меньшие шаги по мере приближения к минимуму.

- Затухание по времени идейно то же самое, что и экспоненциальное, но шаг уменьшается медленнее.

- Метод золотого сечения работает для любых унимодальных функций, всегда сходится к минимуму с заданной точностью. На более сложных функциях может вести себя нестабильно, его нужно дополнительно ограничивать либо подбирать начальные условия. Вычисления дороже чем в предыдущих методах

- Метод дихотомии устойчив к шумам и малым колебаниям функции. Интервал неопределённости строго уменьшается в 2 раза каждые 2 вычисления функции. Также имеет быструю скорость сходимости

- Методы фиксированного шага, экспоненциального затухания и затухания по времени неудобны, так как требуют точного подбора гиперпараметров. Так же этому подвержен метод Армихо, хоть и в меньшей степени

- bfgs в среднем показывает более лучший результат, так как является более сложным квази-Ньютоновским методом (а также написан не нами)

9 Конец

Спасибо за внимание!