

Лабораторная работа №1	2024
<i>Методы нулевого порядка</i>	Артемьев Иван Вадимович
	Никитин Михаил Алексеевич
	Павлов Евгений Андреевич

Цель работы: Реализация метода Ньютона и его модификаций для поиска минимумов функций на языке программирования Python и исследование полученных результатов

Инструментарий: Python 3.12 - библиотеки: NumPy, SciPy, Matplotlib

Постановка задачи:

Реализуйте и исследуйте на эффективность следующие методы:

1. Метод Ньютона с постоянным шагом (learning rate);
2. Метод Ньютона с одномерным поиском
3. Метод Newton-CG из Python библиотеки scipy.optimize.
4. Квазиньютоновский метод BFGS из Python библиотеки scipy.optimize
5. Квазиньютоновский метод BFGS реализованный самостоятельно

Содержание исследования:

Для исследования выберите функцию типа Розенброка, а также одну-две не полиномиальные функции (функции можно взять из лаб.1). На выбранных функциях сравните методы:

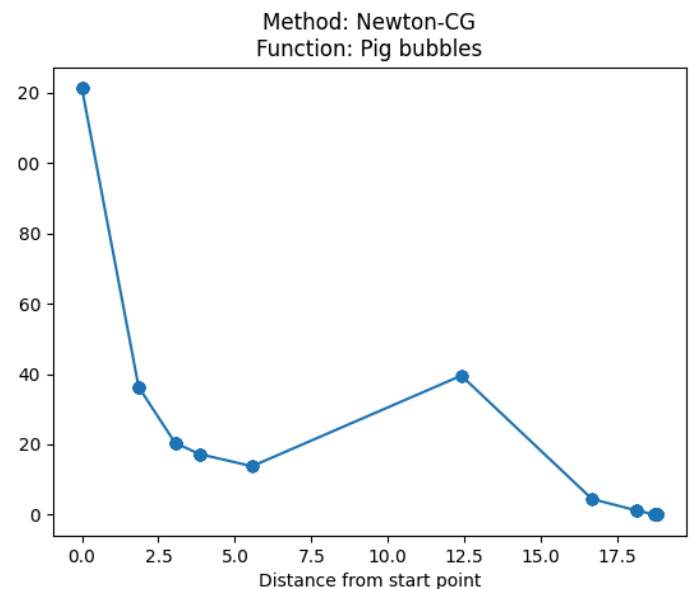
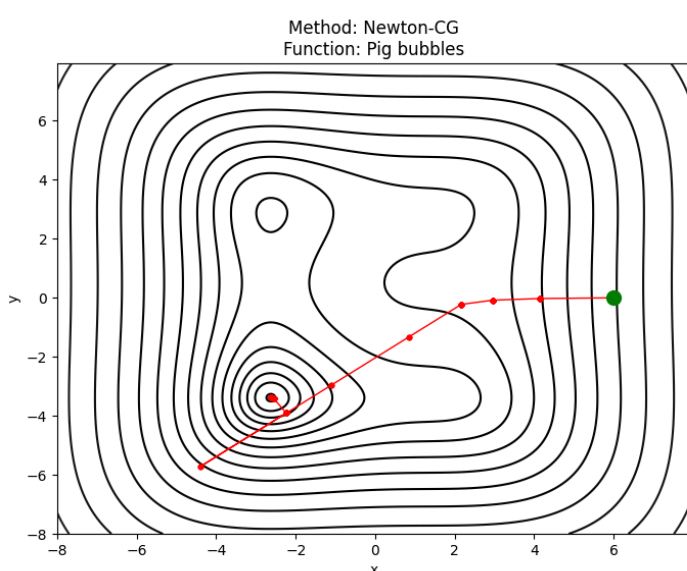
1. Сравните свою реализацию метода Ньютона с методом Newton-CG из библиотеки `scipy.optimize` (по точности и скорости).
2. Сравните эффективность методов нулевого порядка и градиентного спуска из лаб 1., метода Ньютона, квазиньютоновских методов.
3. Сравните эффективность методов нулевого порядка с квазиньютоновскими методами, если в последних производная вычисляется разностным методом (в `scipy.optimize` есть параметр, задающий метод вычисления производной – через аналитические выражения или разностные схемы)
4. Иллюстрируйте примеры, как в лаб.1, чтобы не было скучно.

Описание используемых методов и их реализация:

$$f(x, y): \mathbb{R}^2 \rightarrow \mathbb{R}$$

Задача оптимизации: $f(x, y) \rightarrow \min$

1. Метод Ньютона с постоянным шагом:



Визуализация работы метода ньютона

Сперва сформулируем задачу по-другому. Точка минимума - это точка в которой градиент равен нулю. Значит для нахождения заветного минимума, необходимо решить уравнение $\nabla f(x) = 0$.

Для решения данного уравнения достаточно построить следующую последовательность точек вплоть до необходимого приближения.

$$x^{[j+1]} = x^{[j]} - \alpha \cdot H^{-1}(x^{[j]}) \cdot \nabla f(x^{[j]})$$

В данном случае $H^{-1}(x^{[j]}) \cdot \nabla f(x^{[j]})$ - указывает направление наискорейшего спуска, а коэффициент α , который также, как и в градиентном спуске, называется *learning rate*. В данном методе этот коэффициент остается постоянным и выбирается перед запуском - как гиперпараметр.

Условием остановки является $|\nabla f(x^{[j]})| < \varepsilon$.

2. Метод Ньютона с изменяемым шагом(использование метода Дихотомии):

Данный метод является модификацией описанного выше метод, которая заключается в исправлении проблемы с тем, что алгоритм может не сойтись при некоторых значениях α - *learning rate*. На каждом шаге алгоритма выбирается новое значение α с помощью метода дихотомии для функции $g(step) = f(x^{[j]} - step \cdot H^{-1}(x^{[j]}) \cdot \nabla f(x^{[j]}))$ (обратим внимание, что в данном случае $x^{[j]}$, $H^{-1}(x^{[j]})$ и $\nabla f(x^{[j]})$ - фиксированные числа, то есть g - функция одной переменной и к ней действительно применим данный метод):

- А. Задаётся начальный интервал $(0, learning_rate)$;
- В. Убедиться, что на концах функция g имеет разный знак;
- С. Повторять

- a. выбрать внутри интервала точку *mid*;
- b. сравнить знак функции в точке *mid* со знаком функции в одном из концов;
 - i. если совпадает, то переместить этот конец интервала в точку *mid*,
 - ii. иначе переместить в точку *mid* другой конец интервала;

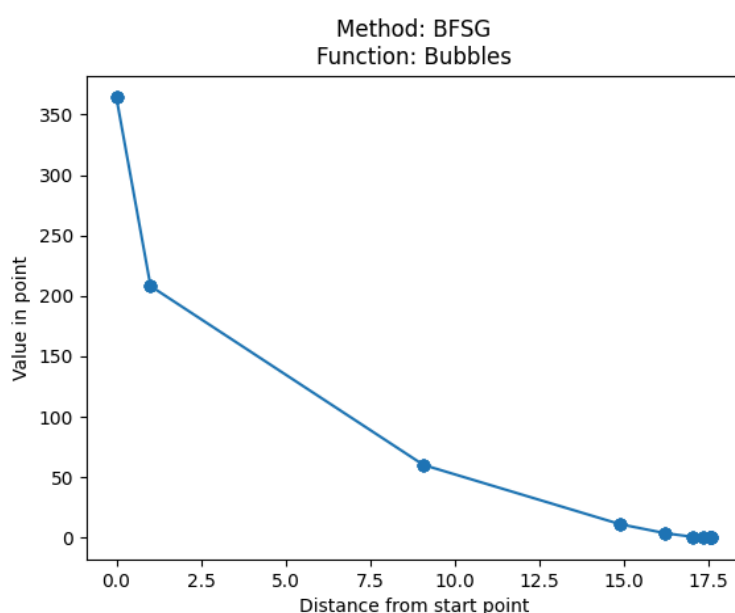
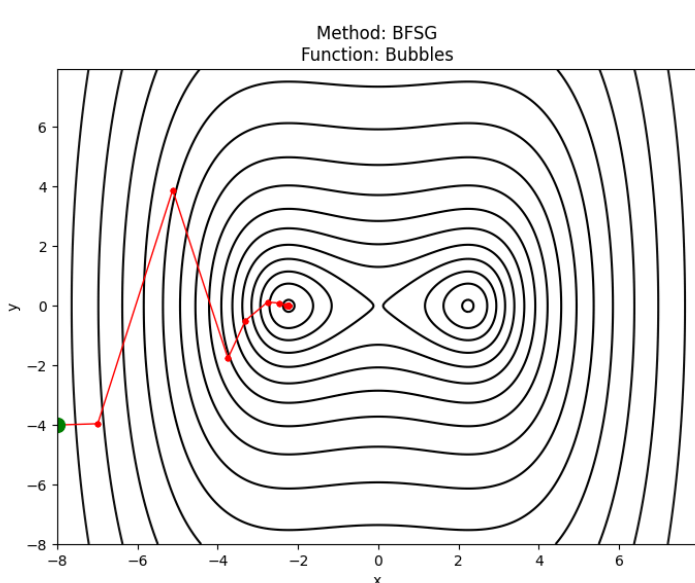
пока не будет достигнута нужная точность.

Благодаря данной модификации - метод градиентного спуска больше не будет расходиться, так как значение α будет уменьшаться.

Реализация метода дихотомии представлена в файле *d1_methods.py*

3. Методы из библиотеки *scipy*

- a) Метод Ньютона (Newton-CG) запускаем с параметрами *jac* (функция, вычисляющая якобиан), *hess* (функция, вычисляющая гессиан)
- b) Квазиньютоновский метод (BFGS) запускаем без параметра *jac*, для вычисления производной разностным методом



Визуализация работы метода BFGS

4. Самостоятельная реализация метода (BFGS) доп. задание 1:

Задаем начальную точку x_0 и точность(epsilon).

Определяем начальное приближение $H_0 = B_0^{-1}$, где B_0 - обратный гессиан функции.

Находим точку, в направлении которой будем производить поиск, она определяется следующим образом: $p_k = -H_k \cdot (\text{grad})f_k$

Находим коэффициент k используя линейный поиск (line search из библиотеки scipy), где k удовлетворяет условиям Вольфе:

$$f(x_k + k \cdot p_k) \leq f(x_k) + c_1 \cdot k \cdot \nabla f_k^T \cdot p_k$$

$$f(x_k + k \cdot p_k)^T \cdot p_k \geq c_2 \cdot \nabla f_k^T \cdot p_k$$

$$\text{Вычисляем } x_{k+1} = x_k + k \cdot p_k$$

Фактически мы находим такое k при котором значение функции $f(x_k + k \cdot p_k)$ минимально

Определяем вектора:

$$s_k = x_{k+1} - x_k$$

$$y_k = \nabla f_{k+1} - \nabla f_k$$

Обновляем гессиан функции, согласно следующей формуле:

$$H_{k+1} = (I - k \cdot s_k \cdot y_k^T) \cdot H_k \cdot (I - k \cdot y_k \cdot s_k^T) + s_k \cdot s_k^T$$

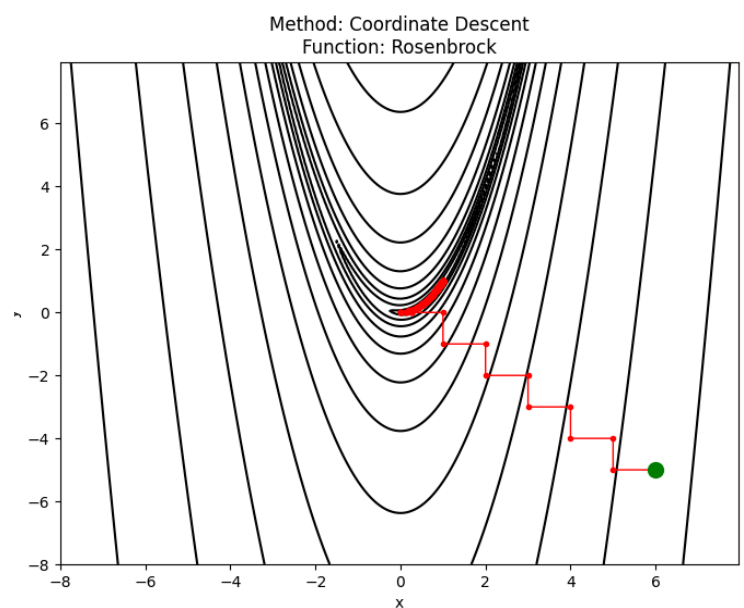
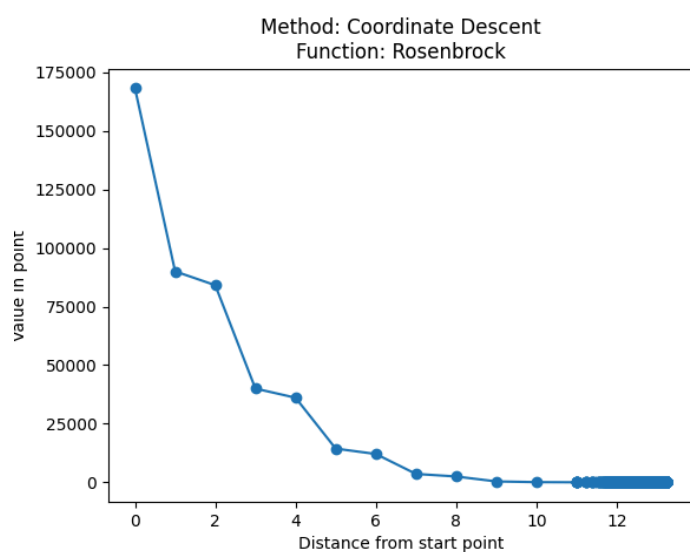
$$\text{Где } k = \frac{1}{y_k^T \cdot s_k}$$

Алгоритм продолжает выполняться до тех пор пока истинно неравенство:

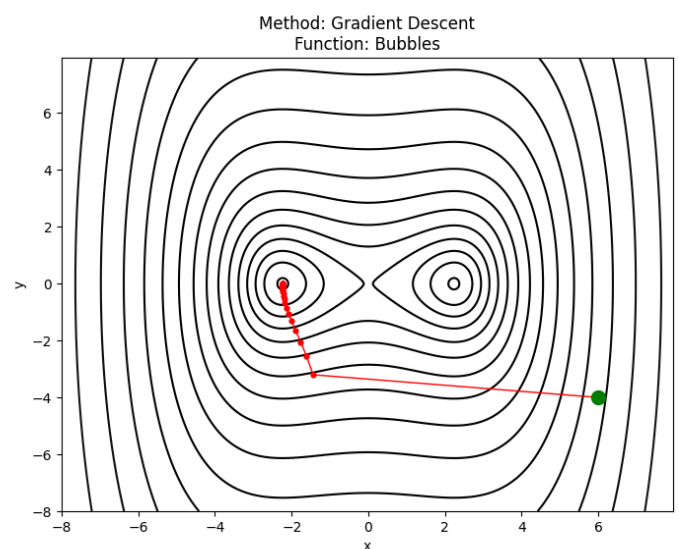
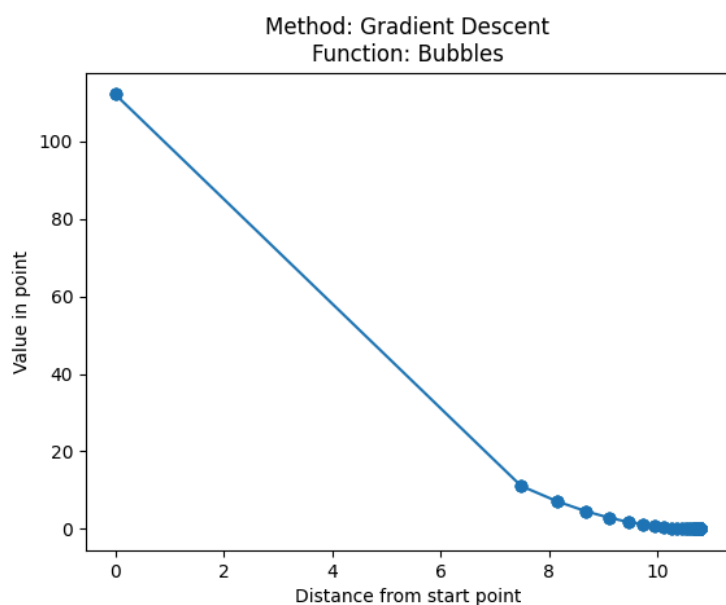
$$|\nabla f_k| > \epsilon$$

На следующих методах подробно останавливаться не будем, они были разобраны в прошлой лабораторной. Ниже представлены визуализации.

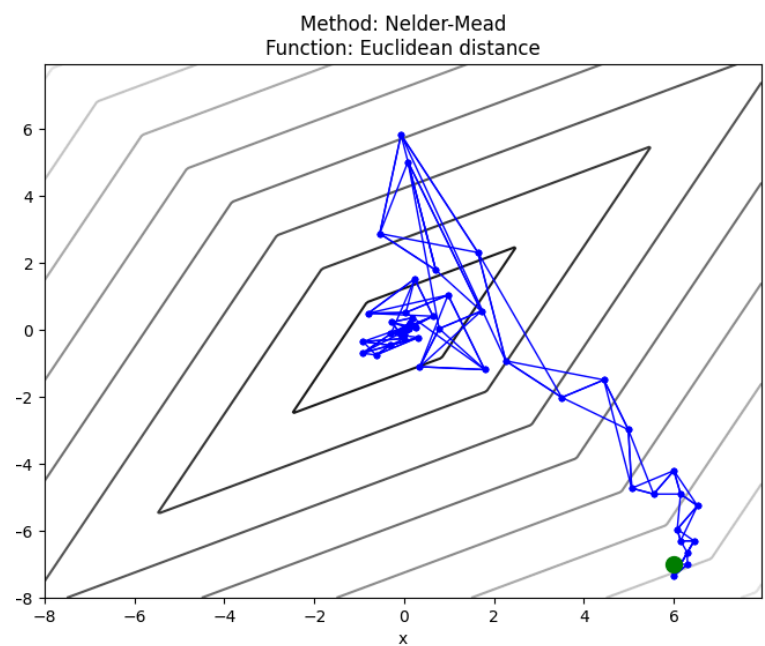
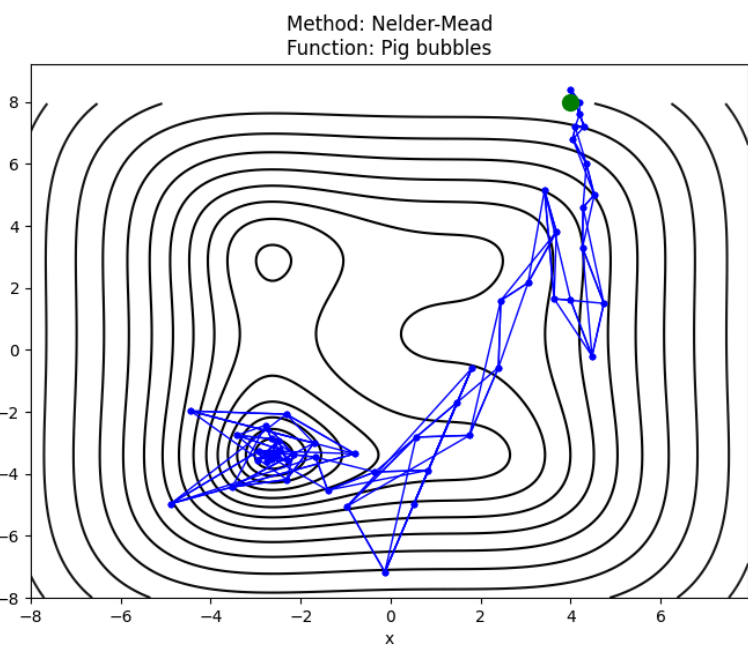
1. Координатный спуск



2. Градиентный спуск



3. Метод Нелдера-Мида (очень красиво)



См. след. стр.

Результаты исследования:

Мы провели исследование всех методов на 6 функциях:

$$1. f_1(x, y) = y^2 - x^2 + \frac{x^4}{10}$$

Данная функция весьма проста - в ней всего два локальных минимума.

$$2. f_2(x, y) = -y^2 - x^2 + \frac{x^4}{10} + \frac{y^4}{20} + y + 2x$$

В данной функции - целых четыре локальных минимума, причем три из них - это достаточно маленькие “впадины”, а четвертая - самая глубокая - глобальный максимум.

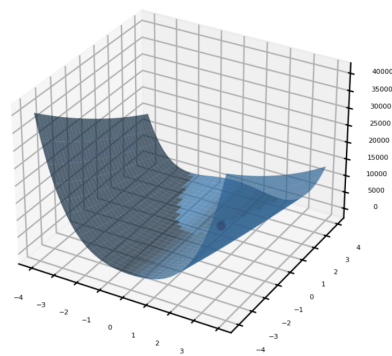
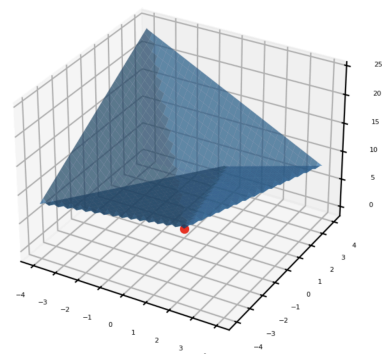
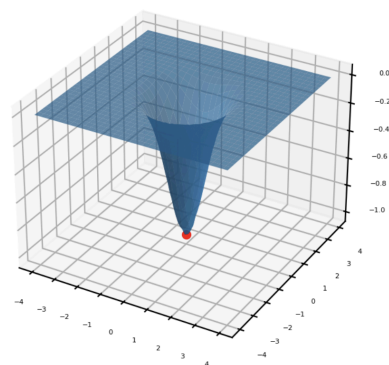
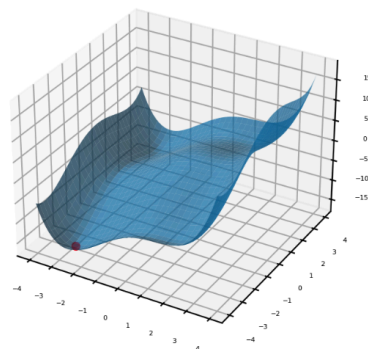
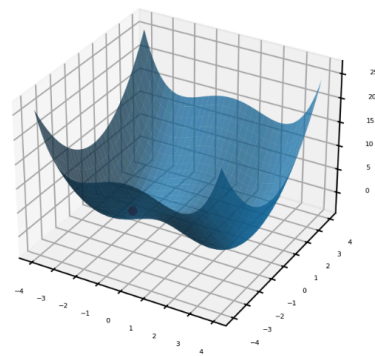
$$3. f_3(x, y) = -e^{-x^2-y^2}$$

В данной функции всего один локальный - он же глобальный минимум. Однако интерес этой функции заключается в том, что на большинстве значений (x, y) - те, что находятся за окружностью с центром в $(0, 0)$ и радиусом 10 - функция имеет очень маленький(по длине) градиент.

$$4. f_4(x, y) = |x + y| + 3|y - x|$$

У данной функции - всего один минимум, но ее особенность заключается в том, что она не гладкая.

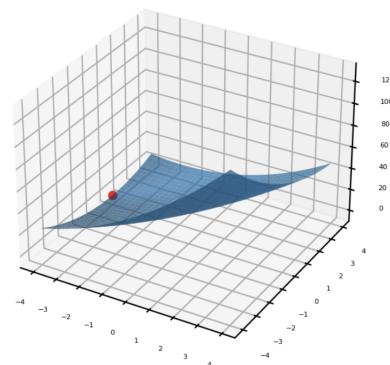
$$5. f_5(x, y) = (1 - x)^2 + 100(y - x^2)^2$$



Функция Розенброка

$$6. f_6(x, y) = x^2 - xy + y^2 + 9x - 6y + 20$$

Простая полиномиальная функция



Сравнение нашей реализации метода

Ньютона с библиотечной Newton-CG:

Если сравнивать результаты, которые показывают данные методы. То можно отметить следующие факты:

- Наша реализация работает по времени примерно столько же, как и библиотечная на функциях, на которых они хорошо работают.
- Наша реализация показывает меньшую точность на сложных функциях, где важна стартовая точка.
- Среднее число вызовов функции в нашей реализации в 4 раза больше, чем в библиотечной.
- Что интересно, наша реализация на функции Розенброка, показывает лучший результат, чем библиотечная.

Из представленных фактов, можно сделать вывод, что библиотечная реализация задумывается о выборе стартовой точки - в то время, как наша стартует с заданной. При этом время работы обоих методов примерно одинаковое - что хорошо. Но в нашей реализации необходимо большее количество раз считать саму функцию. Скорее всего в библиотечной реализации есть оптимизация на расчет Гессиана, чего у нас нет.

При этом сравнивая наши реализации метода Ньютона (без и с одномерной оптимизацией), можно отметить что по работоспособности они в целом показывают одинаковые результаты. Однако в варианте с оптимизацией необходимо большее количество вызовов функции, но меньшее количество

итераций. За счет чего данный метод выигрывает на сложных функциях, которые быстро считаются.

В целом мы приблизились к библиотечной реализации, и даже местами обогнали её.

Сравнение эффективности методов нулевого порядка, градиентного спуска с методом Ньютона, квазиньютоновским методом BFGS:

Исходя из полученных результатов, выделим основные факты:

Ньютон vs градиентный спуск:

- Количество успехов на всех функциях больше
- Время работы в среднем в 3 раза меньше
- На хорошо работающих функциях среднее число вызовов меньше
- Большее количество попаданий в локальные минимумы, а не глобальные

Как итог сходимость двух методов примерно одинаковая, при этом Ньютон выигрывает по производительности.

Ньютон vs Нелдер-Мид:

- Количество успехов на работающих функциях - не меньше
- Меньшее количество функций, на которых метод работает
- Время работы на работающих функциях в 6 раз меньше
- Большее количество попаданий в локальные минимумы

На хороших функциях(где адекватная вторая производная) Ньютон показывает лучший результат по производительности, но при этом Нелдер-Мид работает почти на всех функциях.

BFGS vs градиентный спуск:

- По всем параметрам, кроме времени работы, BFGS выигрывает у градиентного спуска

BFGS - показался нам лучше чем градиентный спуск, так как выигрывает по основным параметрам.

BFGS vs Нелдер-Мида:

- Количество вызовов функции меньше
- Время работы меньше у Нелдера-Мида
- На большинстве функций оба метода хорошо показывают результат
- На функциях, где важна точка старта или плохая производная результат работы хуже

На хороших функциях(где адекватная вторая производная) BFGS показывает лучший результат по вызовам функции, но при этом Нелдер-Мид работает почти на всех функциях.

Подытожив все факты и выводы из них, можно сказать, что новые методы имеет более высокую производительность, но они более привередливы к функциям, которые подаются им. Поэтому если функция имеет хороший Гессиан(ненулевой), то можно смело выбирать новые методы, но если нет, то лучше обратиться к классике.

Доп. задание 2:

1.

$$\text{Функция: } f_2(x, y) = -y^2 - x^2 + \frac{x^4}{10} + \frac{y^4}{20} + y + 2x$$

Начальная точка: (5, 0)

В данной точке при запуске метода Ньютона с дихотомией происходит ошибка из-за того, что определитель Гессеана равен нулю. Однако градиентный спуск справляется с данной задачей.

При этом для большинства других точек оба метода хорошо работают (видно по статистике в табличке ниже)

2.

$$\text{Функция: } f_2(x, y) = -y^2 - x^2 + \frac{x^4}{10} + \frac{y^4}{20} + y + 2x$$

Два метода: градиентный спуск с дихотомией и метод Ньютона с дихотомией

Начальная точка: (2, -8)

Результат метода Ньютона: -2.6273651453286857 -3.387619107967227

Результат градиентного спуска: -2.627365132299499 2.874075287440471

В данном случае градиентный спуск попал в локальный минимум, а метод Ньютона в глобальный. Это обусловлено тем, что данные методы, хоть и двигаются в направлении убывания функции, но они имеют разную длину шага, из-за чего в некоторых стартовых точках могут перескочить через какую-то “выпуклость” и попасть в другую и остаться в ней.

См. след. стр.

Анализ сходимости:

	f_1	f_2	f_3	f_4	f_5	f_6
Градиентный спуск	90%	88%	7%	0%	0%	100%
Градиентный спуск с дихотомией	89%	87%	7%	0%	0%	100%
Нелдер-Мид	100%	100%	100%	91%	100%	99.7%
Покоординатный спуск	100%	100%	100%	10%	100%	100%
Метод Ньютона	100%	80%	0%	0%	100%	100%
Метод Ньютона с дихотомией	100%	80%	0%	0%	100%	100%
Newton-CG	100%	100%	28%	2%	84%	100%
scipy BFGS	100%	100%	16%	22%	100%	100%
JBFGS	82%	57%	0%	0%	25%	88%

Количества вызовов:

	f_1	f_2	f_3	f_4	f_5	f_6
Градиентный спуск	263.55	150.7	314.3	-	-	563.568
Градиентный спуск с дихотомией	429.46	268.8	526.6	-	-	870.552
Нелдер-Мид	93.431	95.06	88.37	98.55	178.4	98.798
Покоординатный спуск	182.53	196.1	147.1	133.9	44626	235.699
Метод Ньютона	330.76	780.4	61.29	-	906.6	310.032
Метод Ньютона с дихотомией	427.48	1152	65.58	-	1001	408.052
Newton-CG	149.37	191.7	102.3	474.8	1924	102.694
scipy BFGS	30.048	50.78	8.364	257.1	249.4	17.7
JBFGS	121.80	174.6	21.62	-	723.3	50.585

Время работы:

	f_1	f_2	f_3	f_4	f_5	f_6
Градиентный спуск	0.33	0.25	0.06	-	-	0.67
Градиентный спуск с дихотомией	0.56	0.43	0.1	-	-	1.17
Нелдер-Мид	1.37	0.74	0.69	0.74	1.21	0.84
Покоординатный спуск	0.04	0.06	0.08	0.13	10.93	0.06
Метод Ньютона	0.28	0.91	0.07	-	0.67	0.27
Метод Ньютона с дихотомией	0.37	1.28	0.08	-	0.82	0.4
Newton-CG	0.38	0.5	0.39	1.12	4.76	0.28
scipy BFGS	1.03	1.7	0.25	8.22	9.51	0.61
JBFGS	0.21	0.42	0.02	-	1.52	0.08

Ссылка на репозиторий с кодом:

<https://github.com/Sedromun/lab2-MetOpt>