Продвинутые методы

Лабораторная работа №2

Иванов Владимир, M3235
Шепелев Матвей, M3235
Зубарев Денис, M3235
Команда ООО “АДВЧИПСГРУПП”

Содержание

1 Постановка задачи

В данной работе исследуются “продвинутые” методы оптимизации: первого и второго порядка, в частности, метод Ньютона и квазиньютоновский метод.

1.1 Метод Ньютона и квазиньютоновские методы

Метод Ньютона - способ нахождения минимума функции с использованием метода Ньютона для корней с помощью производной.

$$x_{k+1} = x_k + p$$

где $p = -\nabla^2 f(x_k))^{-1} \nabla f(x_k)$

Квазиньютоновские методы - методы оптимизиации, исключающие использование второй производной (то есть гессиана).

Задача лабораторной работы - реализовать метод Ньютона и квазиньютоновские методы (или исследовать их реализацию) и исследовать различия в эффективности между ними и методами градиентного спуска из Лабораторной №1 при работе на Интересных™ функциях двух переменных.

2 Реализованные методы

2.1 Метод Ньютона

Нами были написаны две реализации метода Ньютона: демпфированный (со стратегией выбора шага) и dogleg.

Еще мы использовали реализацию из библиотеки scipy для сравнения.

2.1.1 Демпфированный метод Ньютона

В отличие от стандартного метода Ньютона, x_k здесь вычисляется следующим образом:

$$x_{k+1} = x_k + \alpha p$$

, где $\alpha \in (0, 1)$

Поддерживаются любая `LearningRateFunc`, т.е любая стратегия выбора α, и также реализованы стратегии:

1. Константная (в тестовой реализации $h = 1$)
2. Экспоненциальная ($h = e^{-\lambda k}$)

2.1.2 Dogleg (собачья нога)

Внутри реализации для одномерного поиска использован реализованный в прошлой лабораторной работе поиск по правилу Армихо.

Границы Δ - гиперпараметры, вынесены в сигнатуру метода (в тестовой реализации $\ll 0.05, < 0.25, \geq 0.75$).

2.2 Scipy Newton-CG

В отличие от наших реализаций, эта ищет p решением линейного уравнения, а не через обращение гессиана.

2.3 BFGS

Внутри реализации для одномерного поиска так же используется поиск по правилу Армихо, реализованный в прошлой лабораторной.

Также мы использовали реализацию BFGS из scipy для сравнения.

3 Оптимизация

> Сначала оптимизируем, потом тюним. Давайте оптюнить
>
> — Джек Блэк (возможно)

Также была проведена оптимизация гиперпараметров для выбора шага с экспоненциальной и константной стратегией и для границ дельты для Dogleg.

Оптимизация была проведена с помощью библиотеки optuna, количество итераций оптимизации: до 200 (или до 10 минут, смотря, что займет дольше).

4 Результаты работы методов

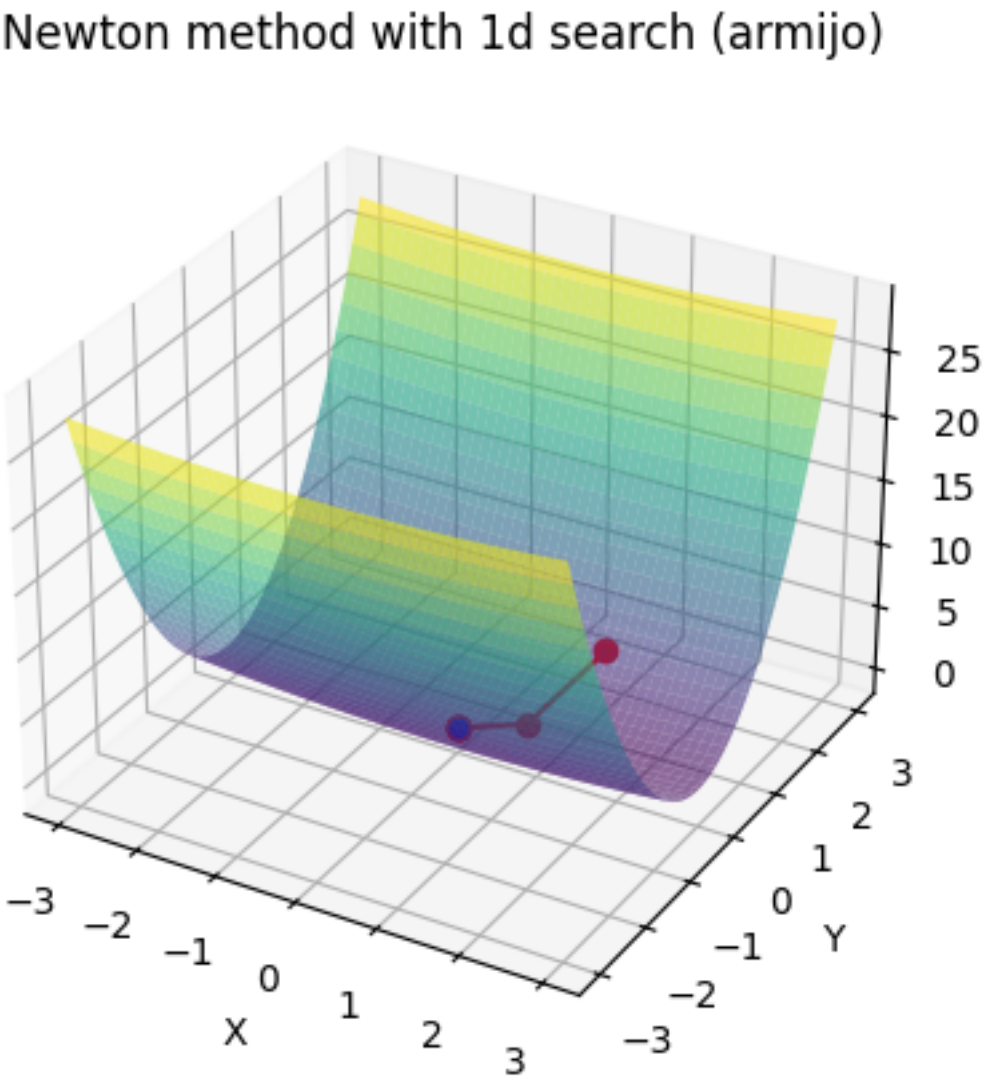
4.1 $q_2 = \left(A = \begin{pmatrix} 0.1 & 0 \\ 0 & 3 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 \end{pmatrix}, C = 0.0\right)$

Плохо обусловленная квадратичная функция из первой лабораторной.

Будем запускать из точки $x_0 = (1, 4)$

метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	[1, 4]	10002	0	10002	0	9.68e-16
LR exp(0.5)	[1, 4]	45	0	45	0	2.39e+0
LR exp(1.27)	[1, 4]	19	0	19	0	1.06e+2
Armijo GD	[1, 4]	1348	4674	1348	0	1.96e-12
Dichotomy GD	[1, 4]	16	900	16	0	2.80e-15
Scipy Wolfe GD	[1, 4]	113	20297	339	0	1.84e-14
Damped Newton	[1, 4]	10002	0	10002	10002	1.76e-15
Damped Newton Opt	[1, 4]	109	0	109	109	2.58e-14
Dog Leg Armijo	[1, 4]	5	76	5	5	2.58e-14
Dog Leg Armijo Opt	[1, 4]	5	76	5	5	2.58e-14
BFGS	[1, 4]	25	100	51	0	1.90e-05
Scipy Newton-CG	[1, 4]	2	5	5	0	6.40e-29
Scipy BFGS	[1, 4]	7	8	8	0	3.96e-12

Dog Leg методы и оптимизированный Damped Newton также показали хорошие результаты. При этом неоптимизированный Damped Newton не смог сойтись и вышел по предохранителю.



BFGS имел относительно большую погрешность и требовал больше итераций, чем реализации метода Ньютона.

При этом Scipy BFGS показал наилучший баланс между точностью и эффективностью, достигая очень высокой точности за минимальное число итераций и вычислений.

По сравнению с методами из предыдущей лабораторной работы “продвинутые” методы показали результаты лучше: им требовалось меньше итераций, при этом они не потеряли в точности.

4.2 f_4

$$f_4 = \left(x^2 - 1\right)^2 + y^2 + 0.5x$$

:w

Мультимодальная функция с глобальным минимумом в точке

$$x_0 \approx (-1.058, 0),\ f(x_0) \approx -0.515$$

и еще одним локальным в точке

$$x_1 \approx (0.930, 0),\ f(x_1) \approx 0.483$$

метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	[1, 1]	4487	0	4487	0	9.98e-1
LR exp(0.5)	[1, 1]	36	0	36	0	9.99e-1
LR exp(1.27)	[1, 1]	17	0	17	0	1.14e+0
Armijo GD	[1, 1]	45	22699	45	0	7.72e-13
Dichotomy GD	[1, 1]	13	745	13	0	9.98e-1
Scipy Wolfe GD	[1, 1]	13	20050	38	0	9.98e-1
Damped Newton	[1, 1]	5463	0	5463	5463	9.98e-1
Damped Newton Opt	[1, 1]	31	0	31	31	9.98e-1
Dog Leg Armijo	[1, 1]	6	61603	6	6	9.98e-1
Dog Leg Armijo Opt	[1, 1]	5	2442	5	5	9.98e-1
BFGS	[1, 1]	7	22	15	0	9.98e-1
Scipy Newton-CG	[1, 1]	7	14	14	0	9.98e-1

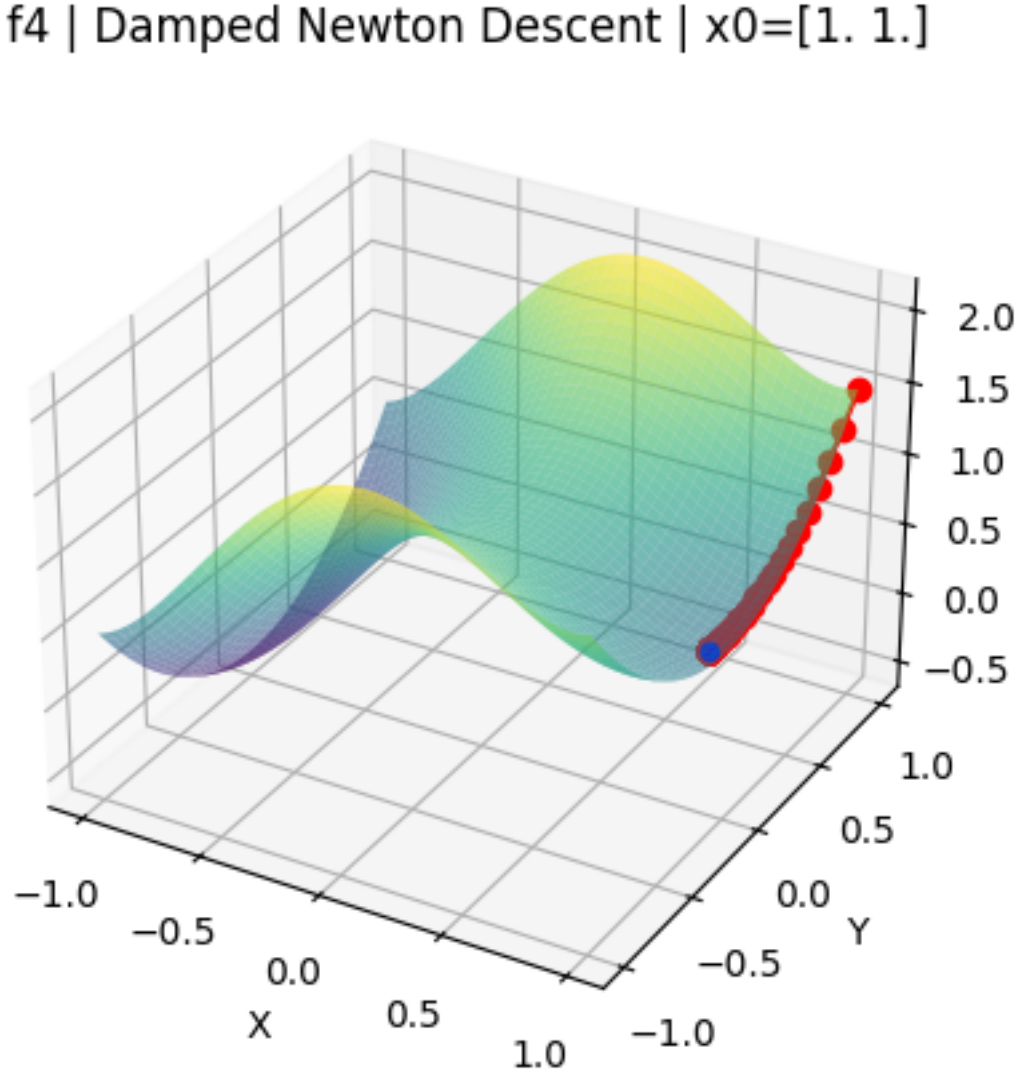
Scipy BFGS	[1, 1]	7	9	9	0	9.98e-1

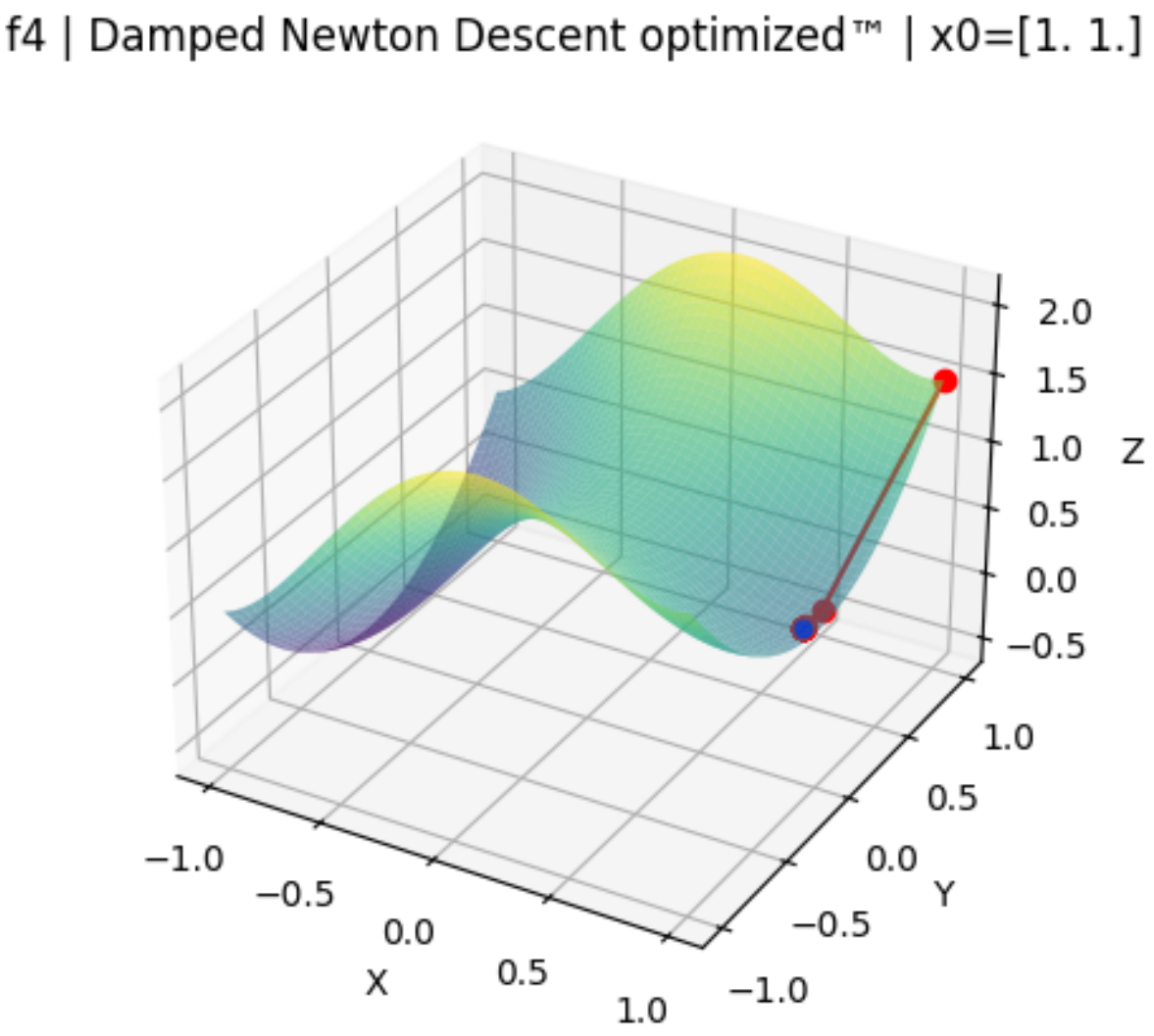
Кроме поиска на основе правила Армихо, ни один метод не осилил найти глобальный минимум функции (что неудивительно, ведь эти методы созданы прежде всего для поиска локального минимума).

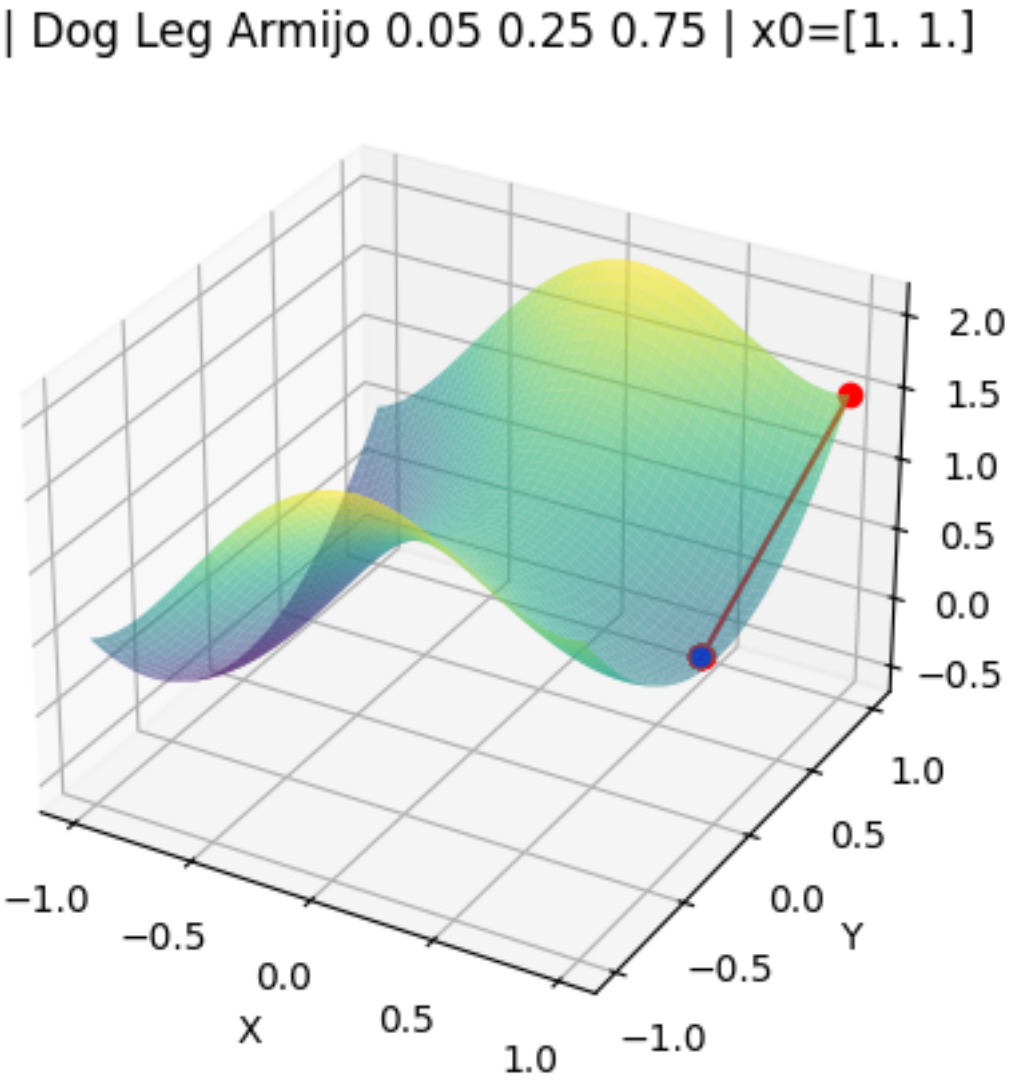
При нахождении локального минимума из собственных методов второго порядка лучше всех справился демпфированный Ньютон с оптимизированными гиперпараметрами (достаточно мало итераций и вычислений градиента/гессиана), неоптимизированному потребовалось много итераций, чтобы сойтись.

Метод Dog Leg потребовал очень много вызовов функции (для внутреннего поиска на основе правила Вульфа), однако оптимизация гиперпараметров смогла уменьшить количество вызовов.

При этом реализации Scipy оказались заметно более эффективными по количеству вызовов функций.


f4 | Damped Newton Descent | x0=[1. 1.]
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0
X
1.0
0.5
0.0
−0.5
−1.0
Y


f4 | Damped Newton Descent optimized™ | x0=[1. 1.]
2.0
1.5
1.0
Z
0.5
0.0
−0.5
1.0
0.5
0.0
Y
−0.5
−1.0
−1.0
−0.5
0.0
X
0.5
1.0



4.3 $f(x) = \sin(x) + \sin(y)$

Мультимодальная функция с повторяющемися минимумами/максимумами.

Будем запускать из точки $x_0 = (1, 1)$

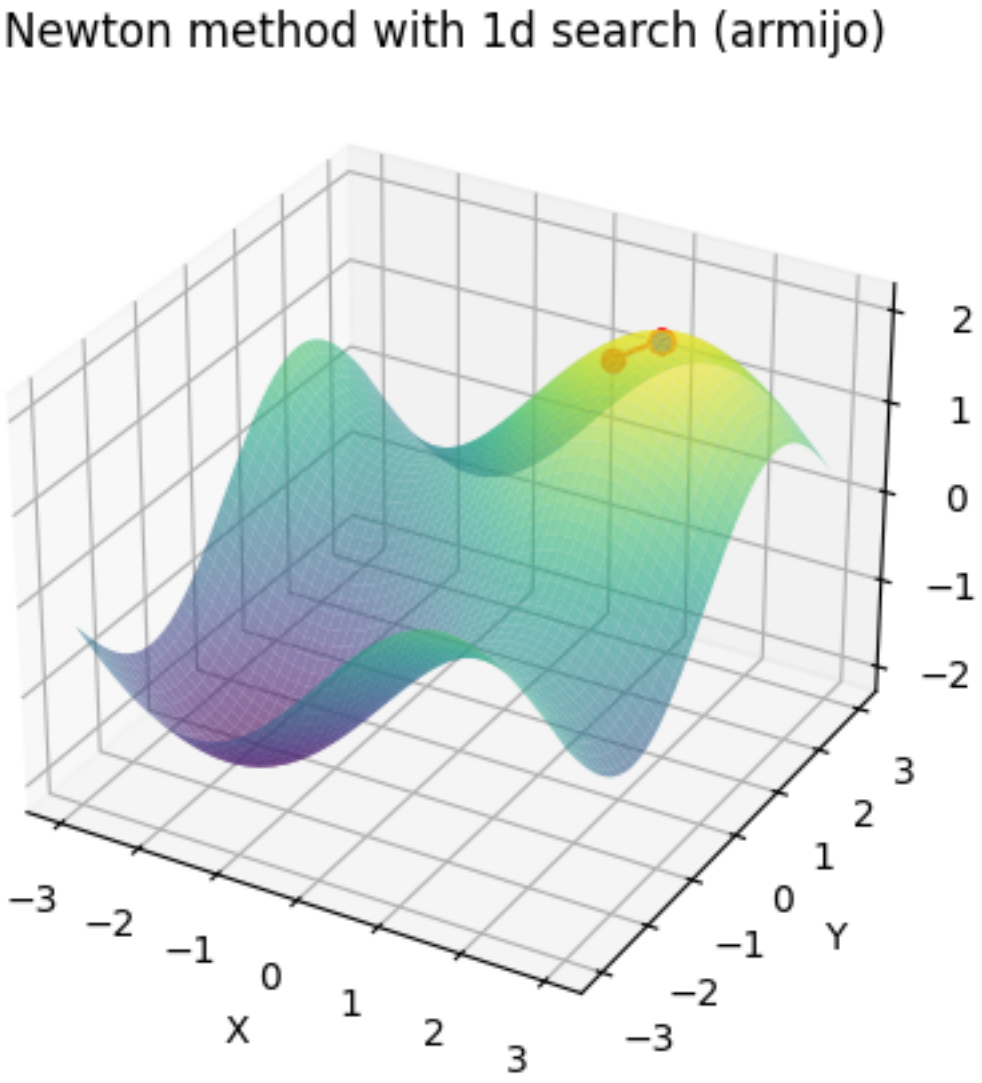
метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	[1, 1]	10002	0	10002	0	2.22e-16
LR exp(0.5)	[1, 1]	41	0	41	0	4.31e-1
LR exp(1.27)	[1, 1]	18	0	18	0	2.20e+0
Armijo GD	[1, 1]	16	20686	16	0	3.15e-14

Dichotomy GD	[1, 1]	2	157	2	0	0.00e+0
Scipy Wolfe GD	[1, 1]	6	20027	20	0	0.00e+0
Damped Newton	[1, 1]	150	0	150	150	4.00e+0
Damped Newton Opt	[1, 1]	20	0	20	20	4.00e+0
Dog Leg Armijo	[1, 1]	6	156	6	6	4.00e+0
Dog Leg Armijo Opt	[1, 1]	5	66	5	5	4.00e+0
BFGS	[1, 1]	4	39	10	0	3.53e-05
Scipy Newton-CG	[1, 1]	2	10	9	0	0.00e+0
Scipy BFGS	[1, 1]	3	6	6	0	1.04e-11

Dog Leg и Damped Newton сходятся к локальным максимумам из стартовой точки. Это связано с особенностью методов оптимизации на основе метода Ньютона.

BFGS нашел точку с не очень большой погрешностью за достаточно малое число итераций/вызовов функций.

Методы из Scipy смогли найти минимум с минимальной погрешностью за минимальное число итераций.



4.4 Функция Розенброка

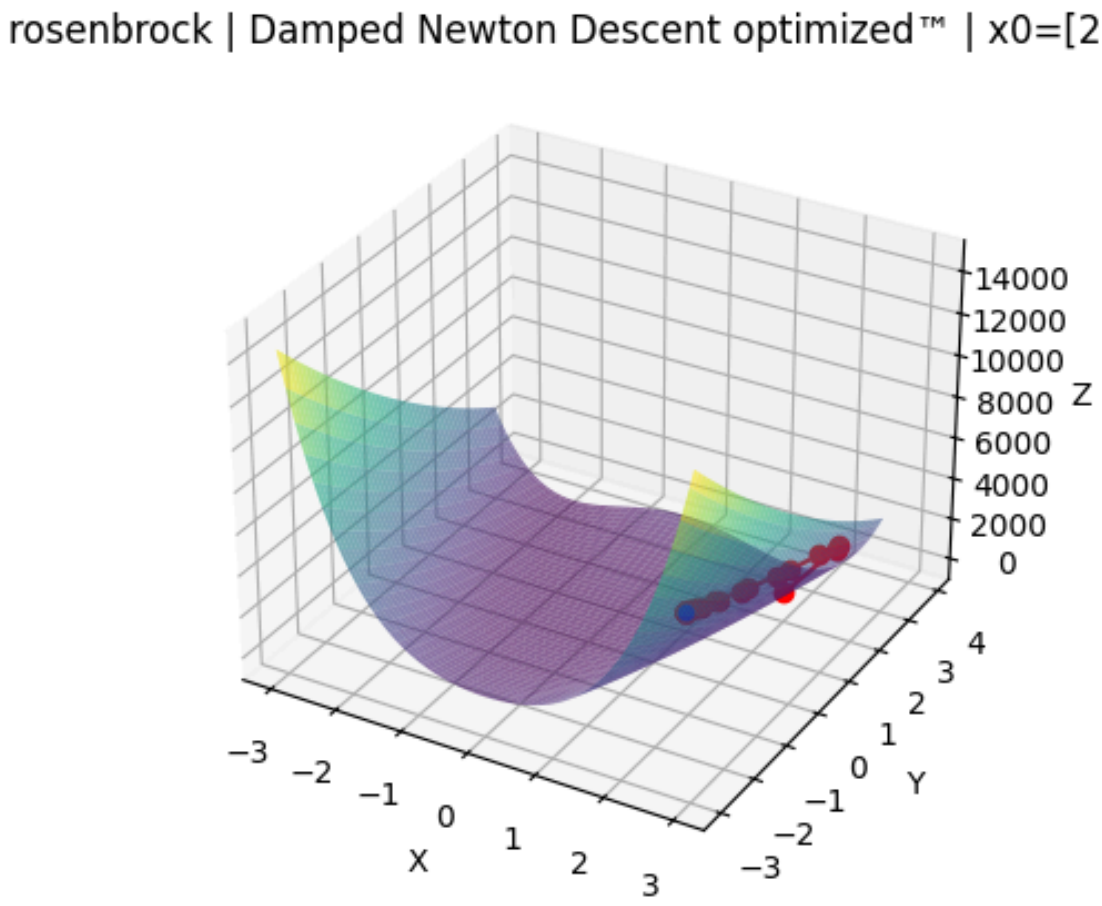
$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$

Стартуем из точки $x_0 = (2, 2)$

метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	[2, 2]	4	0	4	0	6.25e+106
LR exp(0.5)	[2, 2]	4	0	4	0	3.99e+154

LR exp(1.27)	[2, 2]	4	0	4	0	7.67e+147
Armijo GD	[2, 2]	462	14494	462	0	2.22e-06
Dichotomy GD	[2, 2]	220	11112	220	0	7.46e-09
Scipy Wolfe GD	[2, 2]	5647	74224	16941	0	1.78e-08
Damped Newton	[2, 2]	225	0	225	225	4.00e-08
Damped Newton Opt	[2, 2]	22	0	22	22	3.99e-08
Dog Leg Armijo	[2, 2]	18	360348	18	18	3.99e-08
Dog Leg Armijo Opt	[2, 2]	17	300276	17	17	4.00e-08
BFGS	[2, 2]	10000	114906	20001	0	1.13e-05
Scipy Newton-CG	[2, 2]	13	86	74	0	1.04e-08
Scipy BFGS	[2, 2]	30	35	35	0	3.99e-08

Наиболее эффективным оказался демпфированный метод Ньютона с оптимизированными гиперпараметрами (он нашел минимум с приемлемой точностью за малое число итераций/вычислений градиента/гессиана).



С другой стороны, методу Dog Leg потребовалось очень много вычислений функций из-за использования одномерного поиска по условию Вульфа.

BFGS не смог сойтись и вышел по предохранителю, при этом Sciру BFGS показал себя достаточно хорошо.

На удивление, реализации метода Ньютона из Sciру потребоваллось больше вычислений функций/градиента, чем собственной (но у нее и погрешность меньше).

По сравнению с методами из первой лабораторной “продвинутые” методы оказались намного эффективнее для этой функции (learning rate scheduling совсем не смог найти результат, а градиентный спуск с линейным поиском потребовал больше итераций/вычислений функций/градиента).

5 Вывод

В результате лабораторной работы мы реализовали несколько “продвинутых” методов оптимизации и проанализировали их эффективность на нескольких функциях.

Метод Ньютона в целом показывает высокую эффективность (небольшое число итераций при малой погрешности).

BFGS является хорошей альтернативой методу Ньютона, когда вычисление гессиана затратно.

При этом на некоторых функциях эти методы особенно эффективны по сравнению с методами оптимизаций первого порядка.

Оптимизация гиперпараметров дает значительный прирост к эффективности некоторых методов (например, демпфированный метод Ньютона), но для некоторых может даже ее ухудшить (например, потеря точности у learning rate scheduling с экспоненциальным шагом).