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 любой `LearningRateFunc`, т.е любая стратегия выбора α, и так же 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.

Так же мы использовали реализацию BFGS из scipy для сравнения.

3 Оптимизация

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

Так же 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

метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	$[-1, -1]$	10002	0	10002	0	1.52e-15
LR const(0.1)	$[1, 4]$	10002	0	10002	0	9.88e-16

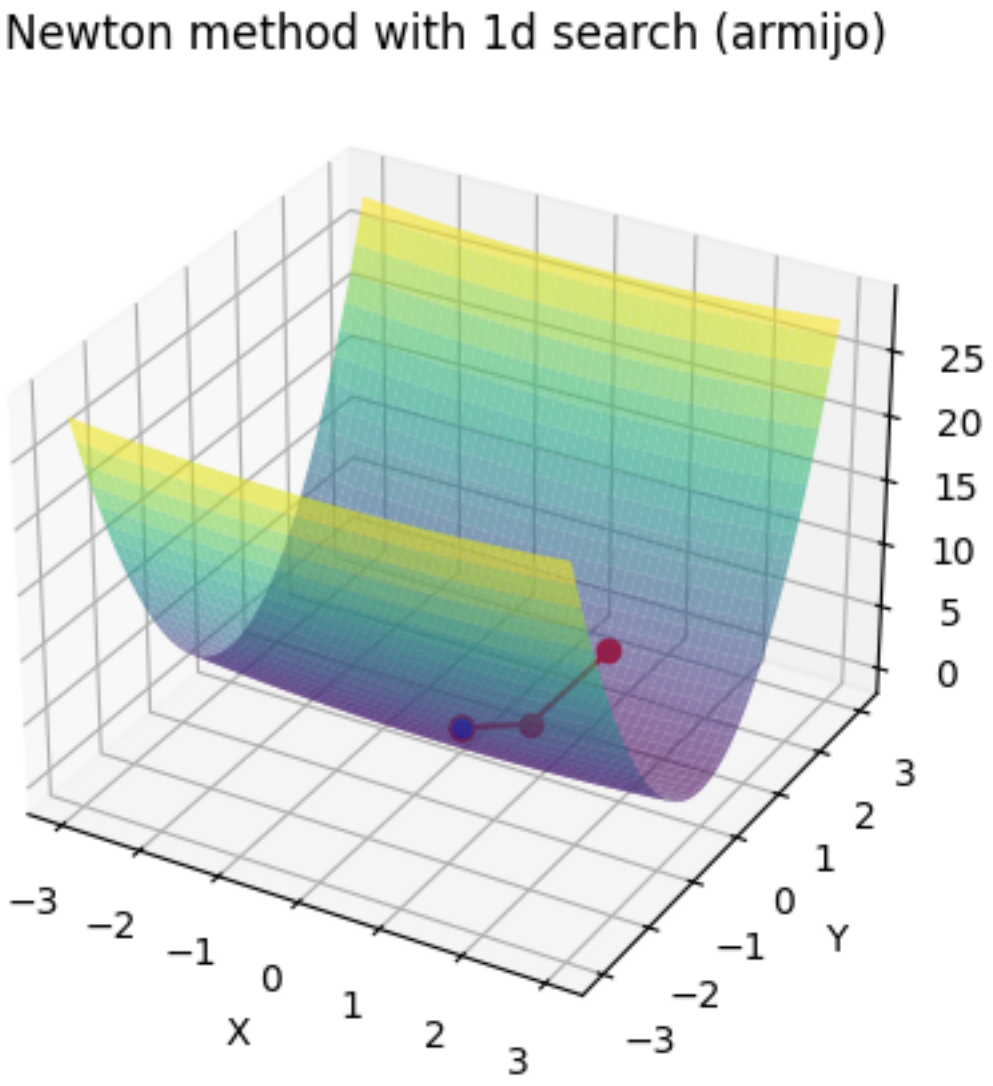
LR const(0.1)	[1, 1]	10002	0	10002	0	5.26e-16
LR exp(0.5)	[−1, −1]	43	0	43	0	1.81e-1
LR exp(0.5)	[1, 4]	45	0	45	0	2.39e+0
LR exp(0.5)	[1, 1]	43	0	43	0	1.81e-1
LR exp(1.27)	[−1, −1]	19	0	19	0	6.66e+0
LR exp(1.27)	[1, 4]	19	0	19	0	1.06e+2
LR exp(1.27)	[1, 1]	19	0	19	0	6.66e+0
Armijo GD	[−1, −1]	1138	3748	1138	0	2.25e-12
Armijo GD	[1, 4]	1098	3794	1098	0	2.37e-12
Armijo GD	[1, 1]	1244	4214	1244	0	2.08e-12
Dichotomy GD	[−1, −1]	21	1193	21	0	3.18e-14
Dichotomy GD	[1, 4]	14	796	14	0	2.51e-14
Dichotomy GD	[1, 1]	14	796	14	0	4.48e-17
Scipy Wolfe GD	[−1, −1]	192	20580	577	0	1.14e-14
Scipy Wolfe GD	[1, 4]	114	20301	341	0	8.14e-16
Scipy Wolfe GD	[1, 1]	188	20566	563	0	2.04e-14
Damped Newton	[−1, −1]	10002	0	10002	10002	1.19e-17
Damped Newton	[1, 4]	10002	0	10002	10002	9.59e-16
Damped Newton	[1, 1]	10002	0	10002	10002	5.37e-17

Damped Newton Opt	[−1, −1]	18	0	18	18	2.58e-14
Damped Newton Opt	[1, 4]	64	0	64	64	2.58e-14
Damped Newton Opt	[1, 1]	30	0	30	30	2.58e-14
Dog Leg Armijo	[−1, −1]	5	220545	5	5	2.53e-14
Dog Leg Armijo	[1, 4]	5	76	5	5	2.58e-14
Dog Leg Armijo	[1, 1]	5	100636	5	5	2.51e-14
Dog Leg Armijo Opt	[−1, −1]	5	101222	5	5	2.51e-14
Dog Leg Armijo Opt	[1, 4]	6	300137	6	6	2.50e-14
Dog Leg Armijo Opt	[1, 1]	4	69	4	4	2.58e-14
BFGS	[−1, −1]	179	21021	180	0	NaN
BFGS	[1, 4]	175	20975	176	0	NaN
BFGS	[1, 1]	165	21007	166	0	NaN
Scipy Newton-CG	[−1, −1]	3	6	6	0	8.12e-11
Scipy Newton-CG	[1, 4]	2	5	5	0	6.40e-29
Scipy Newton-CG	[1, 1]	3	6	6	0	8.12e-11

Scipy BFGS	[−1, −1]	6	7	7	0	2.90e-24
Scipy BFGS	[1, 4]	7	8	8	0	3.96e-12
Scipy BFGS	[1, 1]	6	7	7	0	2.90e-24

Scipy Newton-CG и Scipy BFGS показали наилучший баланс между точностью и эффективностью, достигая очень высокой точности (до 10^-24 - 10^-29) за минимальное число итераций (2-7) и вычислений.

Dog Leg методы и оптимизированный Damped Newton также показали хорошие результаты, сходясь за 4-30 итераций с точностью порядка 10^-14.



Метод с постоянным шагом гарантированно сходится для квадратичных функций, но требует значительно больше итераций (>10000).

Реализация BFGS оказалась нестабильной, приводя к NaN результатам. Методы с экспоненциальным планированием шага с большими коэффициентами (exp(1.27)) дают большую погрешность.

4.2 f_4

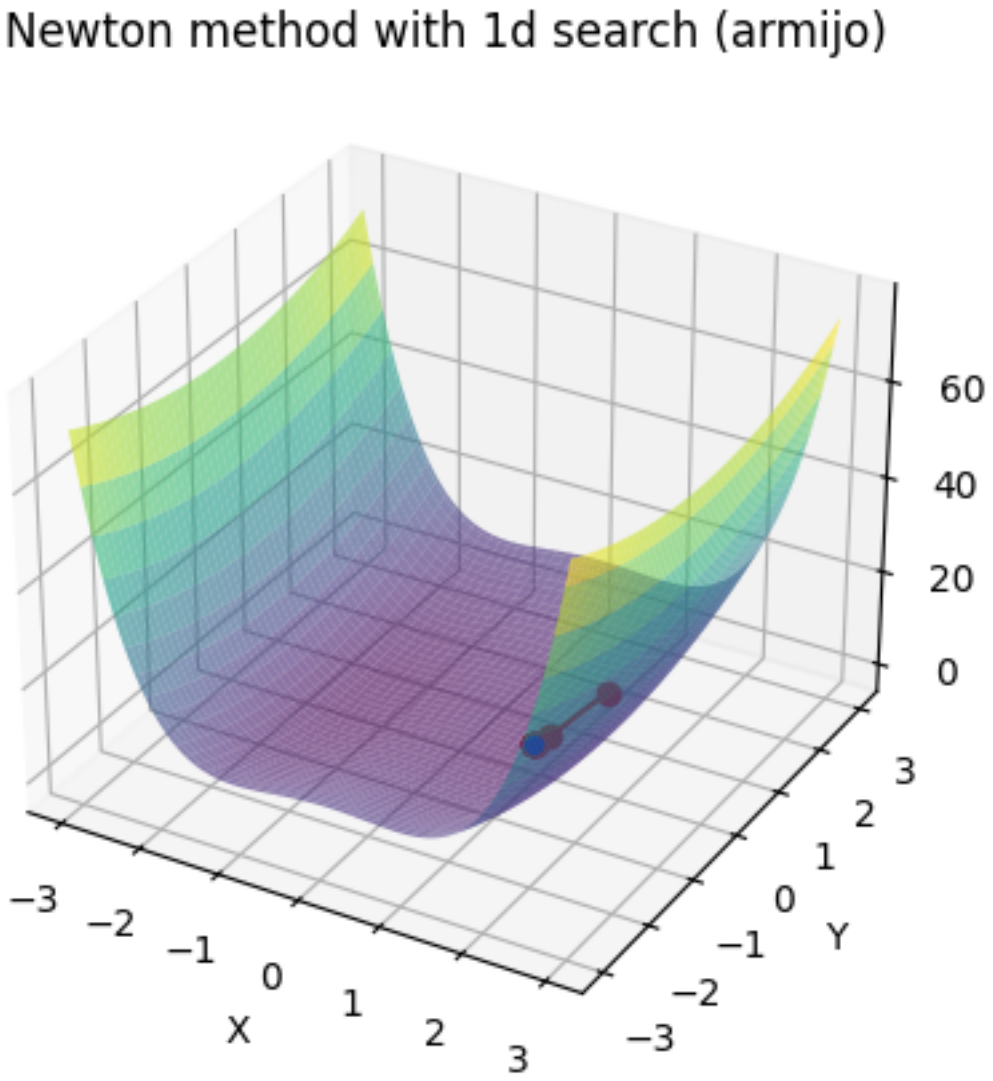
метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	[−1, −1]	99	0	99	0	9.97e-13
LR const(0.1)	[1, 4]	748	0	748	0	9.98e-1
LR const(0.1)	[1, 1]	379	0	379	0	9.98e-1
LR exp(0.5)	[−1, −1]	7	0	7	0	6.12e+141
LR exp(0.5)	[1, 4]	38	0	38	0	1.00e+0
LR exp(0.5)	[1, 1]	36	0	36	0	9.99e-1
LR exp(1.27)	[−1, −1]	18	0	18	0	1.38e+0
LR exp(1.27)	[1, 4]	18	0	18	0	2.95e+0
LR exp(1.27)	[1, 1]	17	0	17	0	1.14e+0
Armijo GD	[−1, −1]	19	1609	19	0	3.64e-12
Armijo GD	[1, 4]	41	2239	41	0	4.19e-14
Armijo GD	[1, 1]	20	1138	20	0	9.98e-1
Dichotomy GD	[−1, −1]	16	920	16	0	2.40e-13
Dichotomy GD	[1, 4]	10	572	10	0	9.98e-1

Dichotomy GD	[1, 1]	13	745	13	0	9.98e-1
Scipy Wolfe GD	[−1, −1]	18	220	53	0	8.95e-13
Scipy Wolfe GD	[1, 4]	41	20176	122	0	3.73e-13
Scipy Wolfe GD	[1, 1]	13	20050	38	0	9.98e-1
Damped Newton	[−1, −1]	157	0	157	157	9.67e-13
Damped Newton	[1, 4]	173	0	173	173	9.98e-1
Damped Newton	[1, 1]	163	0	163	163	9.98e-1
Damped Newton Opt	[−1, −1]	36	0	36	36	9.08e-13
Damped Newton Opt	[1, 4]	14	0	14	14	9.98e-1
Damped Newton Opt	[1, 1]	16	0	16	16	9.98e-1
Dog Leg Armijo	[−1, −1]	6	201475	6	6	9.19e-13
Dog Leg Armijo	[1, 4]	6	320145	6	6	9.98e-1
Dog Leg Armijo	[1, 1]	6	1415	6	6	9.98e-1
Dog Leg Armijo Opt	[−1, −1]	6	82641	6	6	9.52e-13
Dog Leg Armijo Opt	[1, 4]	7	20296	7	7	9.98e-1

Dog Leg Armijo Opt	[1, 1]	6	20948	6	6	9.98e-1
BFGS	[−1, −1]	237	1577	238	0	3.09e+53
BFGS	[1, 4]	581	3341	582	0	2.50e+54
BFGS	[1, 1]	195	1191	196	0	3.06e+55
Scipy Newton-CG	[−1, −1]	7	14	14	0	3.27e-13
Scipy Newton-CG	[1, 4]	6	13	13	0	9.98e-1
Scipy Newton-CG	[1, 1]	7	14	14	0	9.98e-1
Scipy BFGS	[−1, −1]	6	8	8	0	1.64e-12
Scipy BFGS	[1, 4]	9	10	10	0	9.98e-1
Scipy BFGS	[1, 1]	7	9	9	0	9.98e-1

Scipy Newton-CG, Scipy BFGS и оптимизированные методы с учетом кривизны функции (Damped Newton Opt, Dog Leg) показывают наилучшие результаты сходясь за 6-14 итераций с точностью 10^-13.

Большинство методов достигают глобального минимума при старте из точки [−1, −1], но из точек [1, 4] и [1, 1] многие сходятся к локальному минимуму с погрешностью 0.998.



Простая реализация BFGS показала крайне плохие результаты с огромными ошибками порядка 10^53-10^55.

4.3 sin sin

метод	x_0	итераций	функции	градиента	гессиана	ошибка
LR const(0.1)	$[-1, -1]$	10002	0	10002	0	2.22e-16
LR const(0.1)	$[1, 4]$	10002	0	10002	0	4.44e-16
LR const(0.1)	$[1, 1]$	10002	0	10002	0	6.66e-16
LR exp(0.5)	$[-1, -1]$	30	0	30	0	1.69e-5

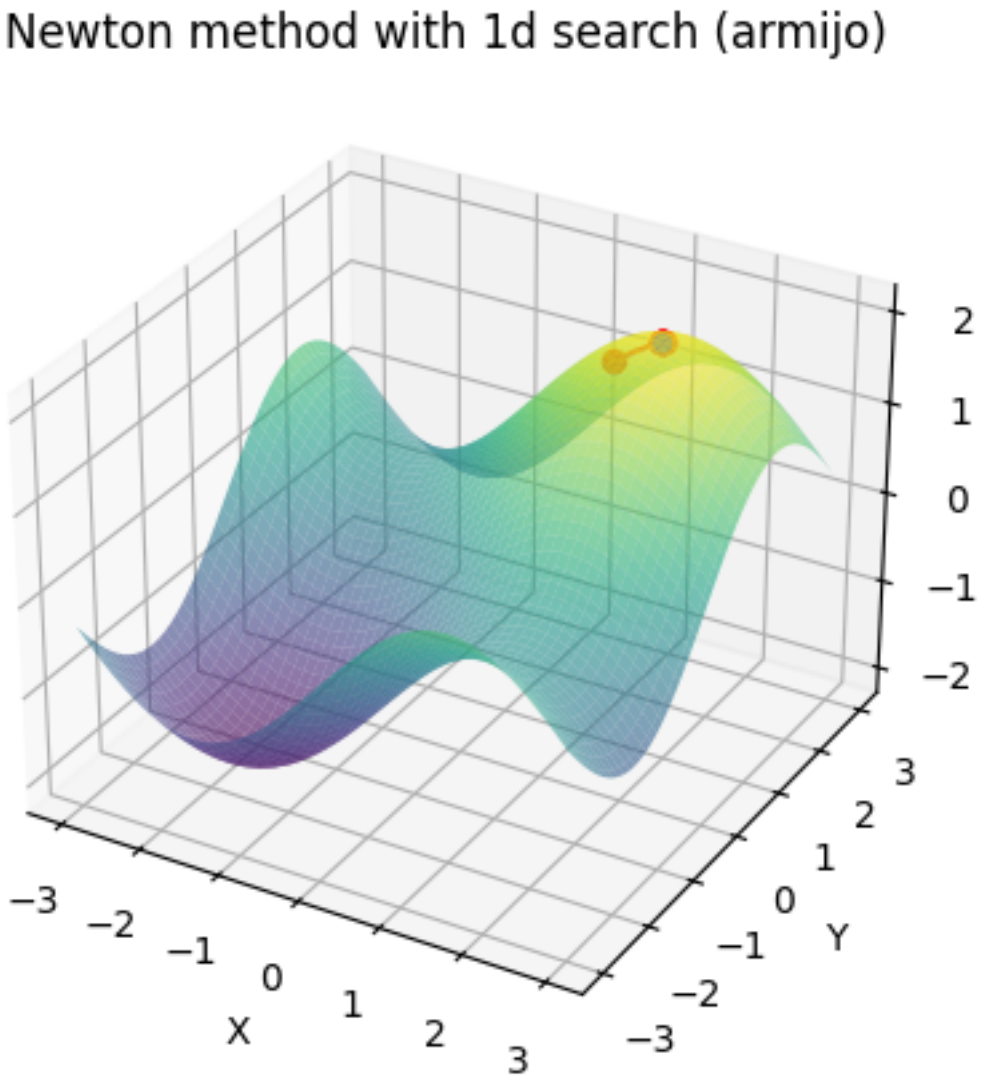
LR exp(0.5)	[1, 4]	38	0	38	0	2.16e-1
LR exp(0.5)	[1, 1]	41	0	41	0	4.31e-1
LR exp(1.27)	[−1, −1]	14	0	14	0	3.84e-4
LR exp(1.27)	[1, 4]	16	0	16	0	1.10e+0
LR exp(1.27)	[1, 1]	18	0	18	0	2.20e+0
Armijo GD	[−1, −1]	12	774	12	0	2.00e-15
Armijo GD	[1, 4]	19	20809	19	0	1.78e-15
Armijo GD	[1, 1]	16	20916	16	0	5.33e-15
Dichotomy GD	[−1, −1]	143	8650	143	0	4.44e-16
Dichotomy GD	[1, 4]	5	316	5	0	9.77e-15
Dichotomy GD	[1, 1]	2	157	2	0	0.00e+0
Scipy Wolfe GD	[−1, −1]	4	20021	12	0	9.99e-15
Scipy Wolfe GD	[1, 4]	5	20023	15	0	0.00e+0
Scipy Wolfe GD	[1, 1]	6	20027	20	0	0.00e+0
Damped Newton	[−1, −1]	10002	0	10002	10002	4.44e-16
Damped Newton	[1, 4]	6917	0	6917	6917	2.00e+0
Damped Newton	[1, 1]	147	0	147	147	4.00e+0
Damped Newton Opt	[−1, −1]	14	0	14	14	9.99e-15
Damped Newton Opt	[1, 4]	43	0	43	43	2.00e+0

Damped Newton Opt	[1, 1]	25	0	25	25	4.00e+0
Dog Leg Armijo	[−1, −1]	6	300443	6	6	8.66e-15
Dog Leg Armijo	[1, 4]	6	420120	6	6	2.00e+0
Dog Leg Armijo	[1, 1]	6	153	6	6	4.00e+0
Dog Leg Armijo Opt	[−1, −1]	6	280658	6	6	7.99e-15
Dog Leg Armijo Opt	[1, 4]	5	69	5	5	2.00e+0
Dog Leg Armijo Opt	[1, 1]	6	153	6	6	4.00e+0
BFGS	[−1, −1]	31	277	32	0	9.44e-1
BFGS	[1, 4]	27	167	28	0	6.41e-4
BFGS	[1, 1]	20	172	21	0	2.65e+0
Scipy Newton-CG	[−1, −1]	2	3	3	0	2.24e-11
Scipy Newton-CG	[1, 4]	5	9	9	0	1.47e-14
Scipy Newton-CG	[1, 1]	2	10	9	0	0.00e+0
Scipy BFGS	[−1, −1]	3	4	4	0	7.02e-14
Scipy BFGS	[1, 4]	8	9	9	0	4.10e-13
Scipy BFGS	[1, 1]	3	6	6	0	1.04e-11

Для тригонометрической функции sin(x) + sin(y) наблюдаются следующие закономерности:

Scipy Newton-CG и Scipy BFGS демонстрируют высокую эффективность, достигая точного результата за 2-8 итераций с минимальным числом вычислений функции и градиента.

Большинство методов успешно находят глобальный минимум из точки [−1, −1], но многие методы высшего порядка (Dog Leg, Damped Newton) сходятся к локальным максимумам из точек [1, 4] и [1, 1].



BFGS показывает неудовлетворительные результаты с большими ошибками, особенно из точки [1, 1] (ошибка 2.65).

4.4 fopp3

метод	x_0	итераций	функции	градиента	гессиана	ошибка

LR const(0.1)	[−1, −1]	10002	0	10002	0	1.28e-1
LR const(0.1)	[1, 4]	10002	0	10002	0	9.99e-2
LR const(0.1)	[1, 1]	10002	0	10002	0	3.26e-2
LR exp(0.5)	[−1, −1]	37	0	37	0	1.22e+0
LR exp(0.5)	[1, 4]	42	0	42	0	3.72e-1
LR exp(0.5)	[1, 1]	32	0	32	0	3.16e-6
LR exp(1.27)	[−1, −1]	18	0	18	0	7.34e-1
LR exp(1.27)	[1, 4]	18	0	18	0	3.97e+0
LR exp(1.27)	[1, 1]	17	0	17	0	1.92e-1
Armijo GD	[−1, −1]	58	2506	58	0	6.32e-2
Armijo GD	[1, 4]	51	21741	51	0	6.32e-2
Armijo GD	[1, 1]	22	1088	22	0	5.64e-8
Dichotomy GD	[−1, −1]	9	508	9	0	5.64e-8
Dichotomy GD	[1, 4]	13	731	13	0	5.64e-8
Dichotomy GD	[1, 1]	28	1560	28	0	6.32e-2
Scipy Wolfe GD	[−1, −1]	33	226	99	0	5.64e-8
Scipy Wolfe GD	[1, 4]	33	483	99	0	6.32e-2
Scipy Wolfe GD	[1, 1]	9	20036	28	0	5.64e-8

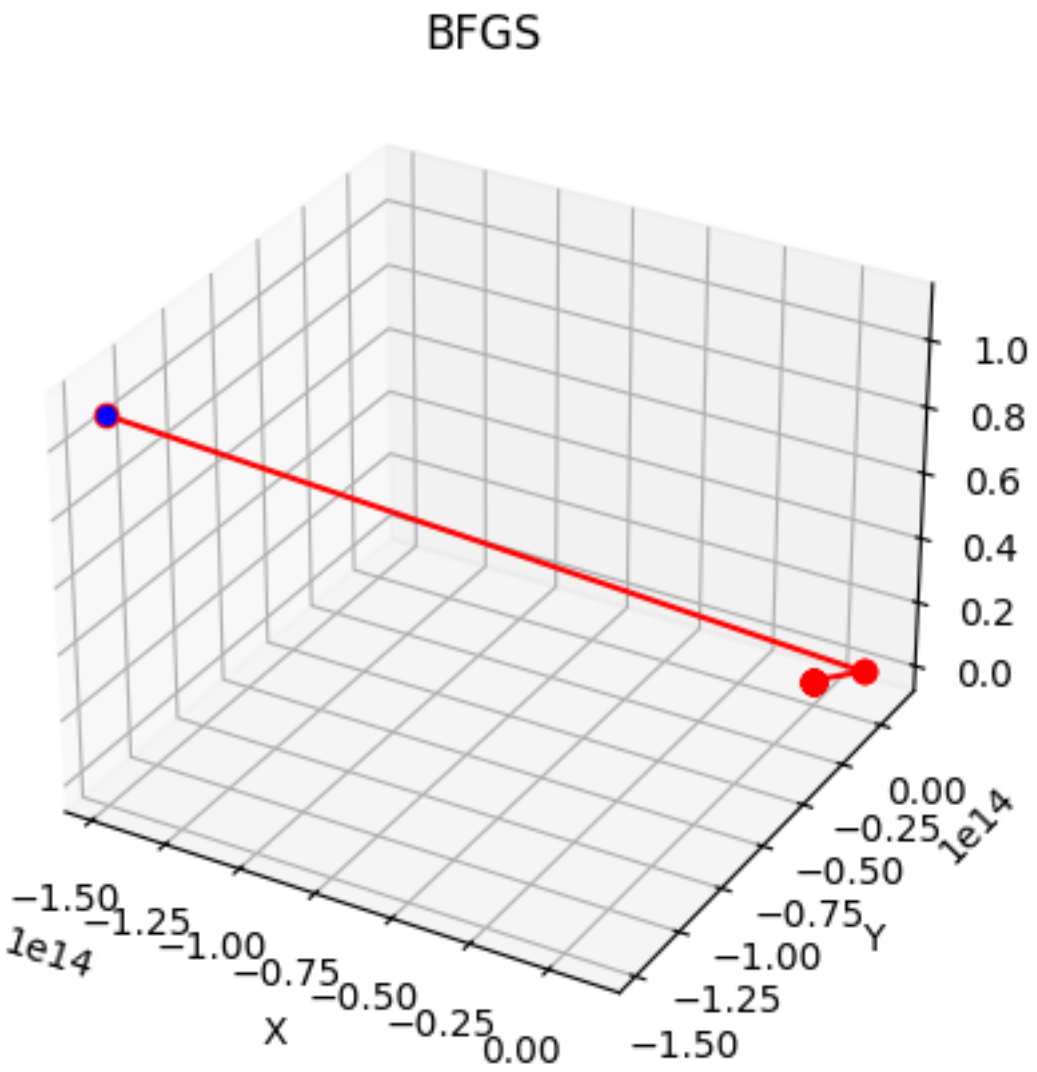
Damped Newton	[−1, −1]	146	0	146	146	6.93e-1
Damped Newton	[1, 4]	139	0	139	139	4.18e+0
Damped Newton	[1, 1]	145	0	145	145	5.77e-1
Damped Newton Opt	[−1, −1]	21	0	21	21	7.41e-1
Damped Newton Opt	[1, 4]	18	0	18	18	1.16e+1
Damped Newton Opt	[1, 1]	15	0	15	15	8.45e-1
Dog Leg Armijo	[−1, −1]	8	1072	8	8	7.41e-1
Dog Leg Armijo	[1, 4]	9	521	9	9	4.30e+0
Dog Leg Armijo	[1, 1]	17	569	17	17	6.48e-1
Dog Leg Armijo Opt	[−1, −1]	8	850	8	8	7.41e-1
Dog Leg Armijo Opt	[1, 4]	10	865	10	10	3.45e+0
Dog Leg Armijo Opt	[1, 1]	18	922	18	18	6.48e-1
BFGS	[−1, −1]	31	259	32	0	4.61e+26
BFGS	[1, 4]	52	40952	53	0	4.08e+28
BFGS	[1, 1]	24	192	25	0	1.09e+28
Scipy Newton-CG	[−1, −1]	9	20	20	0	5.64e-8

Scipy Newton-CG	[1, 4]	10	78	65	0	6.32e-2
Scipy Newton-CG	[1, 1]	9	22	22	0	6.32e-2
Scipy BFGS	[−1, −1]	11	15	15	0	5.64e-8
Scipy BFGS	[1, 4]	7	64	52	0	2.42e+0
Scipy BFGS	[1, 1]	8	14	14	0	6.32e-2

Методы Dichotomy GD, Scipy Wolfe GD и Scipy Newton-CG из определенных начальных точек достигают наивысшей точности (погрешность 5.64e-8) за относительно небольшое число итераций (9-13).

Многие методы, включая Armijo GD и Dog Leg, сходятся к другому стационарному решению с погрешностью 0.063 или 0.74, что может указывать на локальный минимум или седловую точку.

BFGS (не Scipy) показал экстремально плохие результаты с погрешностями порядка 10^26-10^28, а некоторые методы с экспоненциальным шагом приводят к б ольшим ошибкам (>1.0).



LR const(0.1) показывает относительно стабильные результаты (ошибка 0.033-0.128), но требует большого числа итераций (10002).

5 Вывод

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

Библиотечные реализации (Scipy Newton-CG, Scipy BFGS) значительно превосходят наши “ручные” как по эффективности, так и по точности для большинства функций и начальных точек. Это связано с тем, что они используют более сложные стратегии для обновления направлений.

“Простые” методы медленнее, но часто надежнее для сложных функций. Метод с экспоненциальным уменьшением шага совсем не точен, и даже оптимизация гиперпараметра не смогла сильно ему помочь.

Нестабилизированные методы второго порядка и квази-ньютоновские методы (BFGS без правильной имплементации) могут быть катастрофически неэффективными для некоторых функций.

Для квадратичных функций методы второго порядка работают особенно хорошо, в то время как для функций с множеством локальных экстремумов более простые методы могут быть надежнее.