Продвинутые методы

Лабораторная работа №2

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

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

Во всех методах (кроме градиентного спуска с константным LR) использовался критерий остановки, относительный по координате:

$$\|x_{k+1} - x_k\| < \varepsilon(\|x_{k+1}\| + 1)$$

Градиент и гессиана функций вычисляется как симметрическая разность функции и градиента соответственно.

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 константного Learning Rate у демпфированного метода Ньютона 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 $q_2 = 0.1x^2 + 3y^2$

Плохо обусловленная квадратичная функция из первой лабораторной.

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

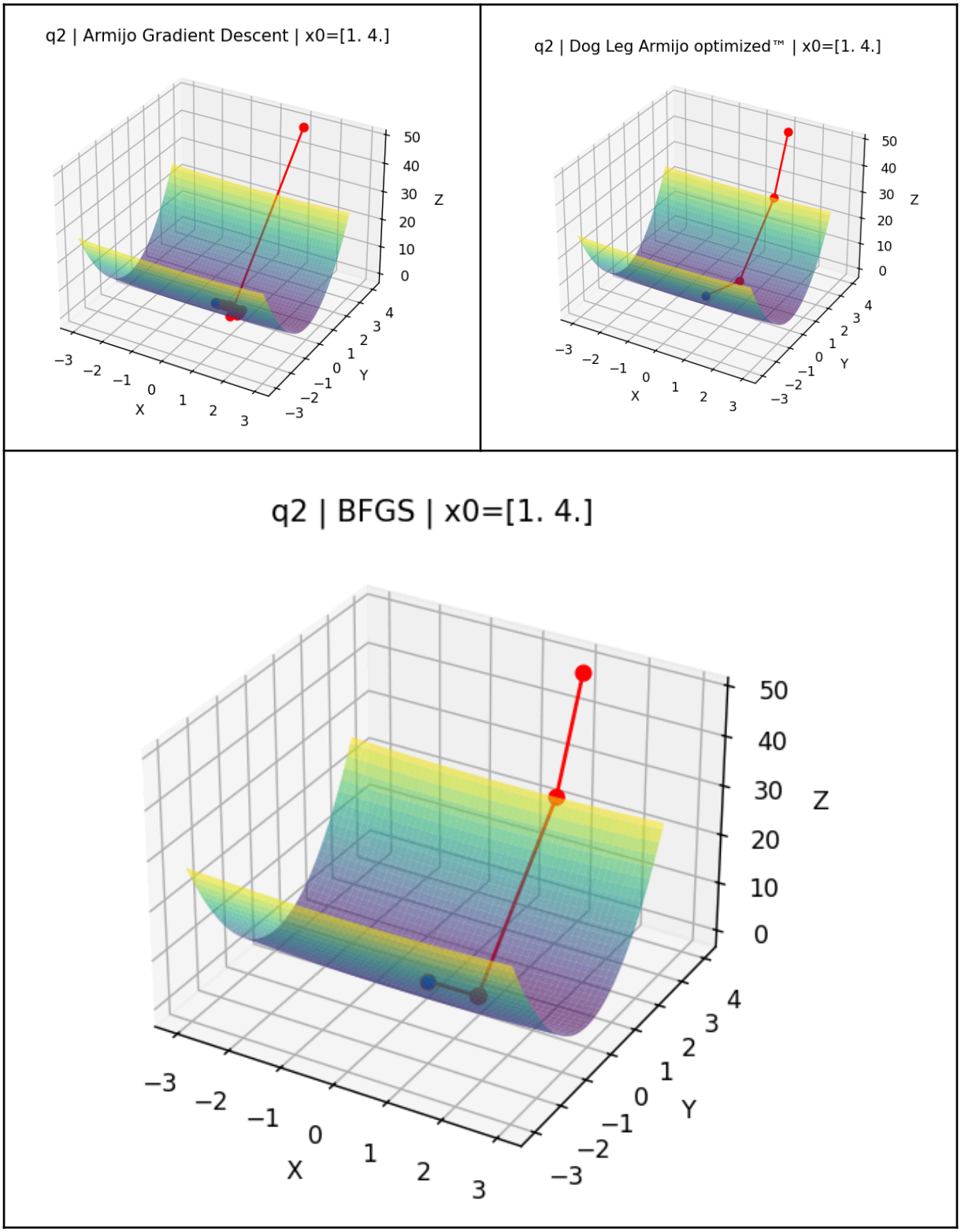
метод	количество итераций	кол-во вычисл. функции	кол-во вычисл. градиента	кол-во вычисл. гессиана	ошибка
GD LR const(0.1)	276	0	828	0	1.43e-6
GD LR exp(0.5)	45	0	45	0	2.39e+0
Armijo GD	1452	3177	1452	0	1.78e-12
Dichotomy GD	21	1169	21	0	1.69e-14
Scipy Wolfe GD	123	345	369	0	1.45e-14
Damped Newton	99	0	297	99	4.19e-8
Damped Newton Opt	5	0	5	5	2.58e-14
Dog Leg Armijo	6	481	6	6	2.44e-14
Dog Leg Armijo Opt	5	72	5	5	2.58e-14
BFGS	4	10	10	0	3.64e-9
Scipy Newton-CG	5	5	5	5	4.71e-39
Scipy BFGS	7	8	8	0	3.96e-12

Неоптимизированный Damped Newton потребовал достаточно много итераций, но оптимизация ему помогла: число итераций уменьшилось в 20 раз, при этом он стал точнее.

Dog Leg методы в целом показали хорошие результаты (при этом оптимизация снова помогла, уменьшив число вычислений функций).

BFGS тоже отлично отработал: за малое число итераций он достаточно точно нашел минимум.

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

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

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

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

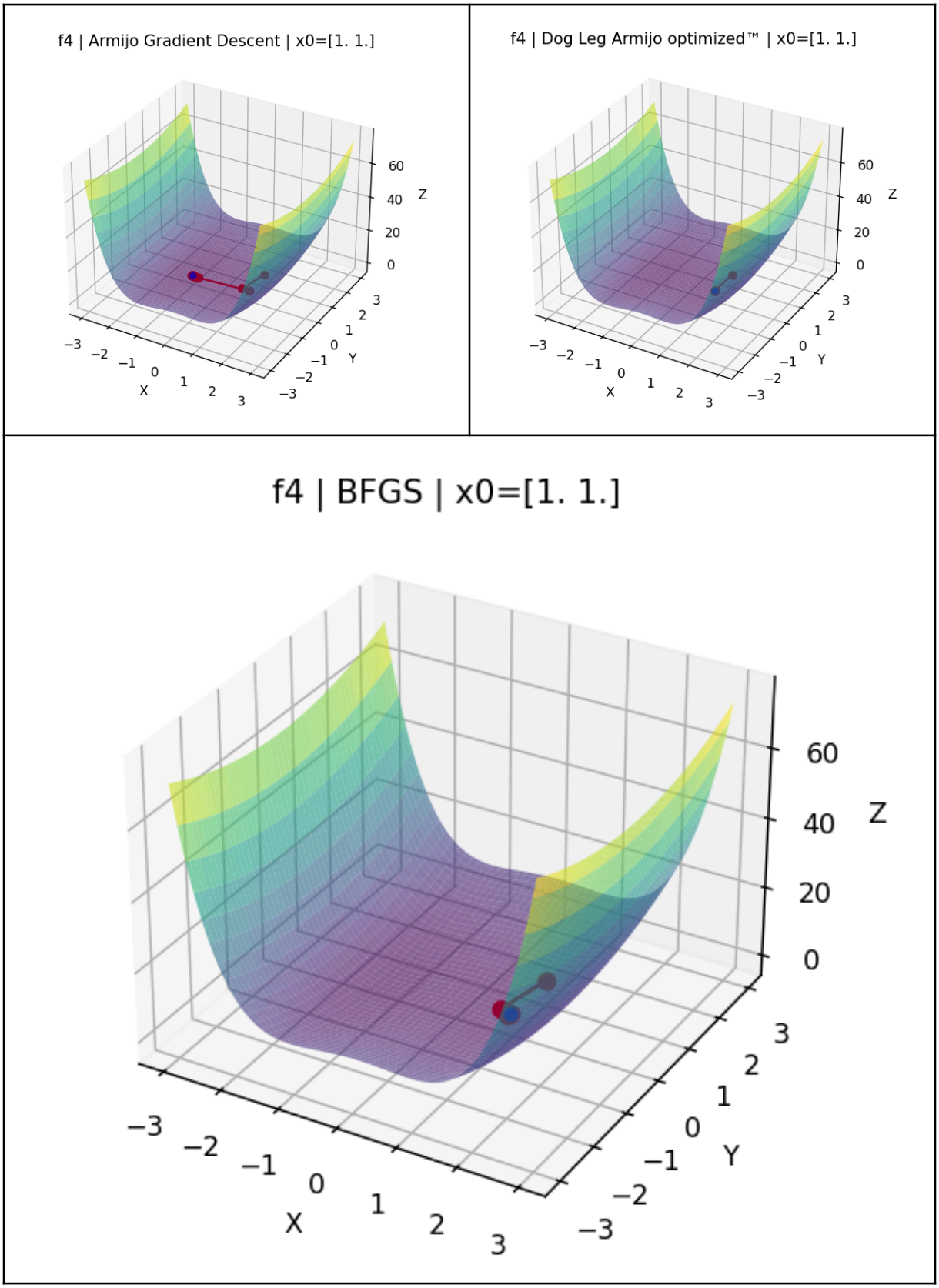
метод	**кол-во итераций**	**кол-во вычисл. функции**	**кол-во вычисл. градиента**	**кол-во вычисл. гессиана**	**ошибка**
GD LR const(0.1)	47	0	141	0	9.98e-1
GD LR exp(0.5)	36	0	36	0	9.99e-1
Armijo GD	32	1019	32	0	3.07e-13
Dichotomy GD	13	746	13	0	9.98e-1
Scipy Wolfe GD	13	75	38	0	9.98e-1
Damped Newton	99	0	297	99	9.98e-1
Damped Newton Opt	10	0	10	10	9.98e-1
Dog Leg Armijo	5	70	5	5	9.98e-1
Dog Leg Armijo Opt	6	807	6	6	9.98e-1
BFGS	4	7	7	0	9.98e-1
Scipy Newton-CG	3	65	53	4	9.98e-1
Scipy BFGS	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, неоптимизированному потребовалось много итераций, чтобы сойтись.

Великолепно показал себя BFGS - ему потребовалось наименьшее число итераций, чтобы найти локальный минимум.

Метод Dog Leg потребовал достаточно много вызовов функции (для внутреннего поиска на основе правила Вульфа), и при этом оптимизация гиперпараметров увеличила их количество.


f4 | Armijo Gradient Descent | x0=[1. 1.]
f4 | Dog Leg Armijo optimized™ | x0=[1. 1.]
f4 | BFGS | x0=[1. 1.]
X
Y
Z

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

Мультимодальная функция с повторяющемися минимумами/максимумами.

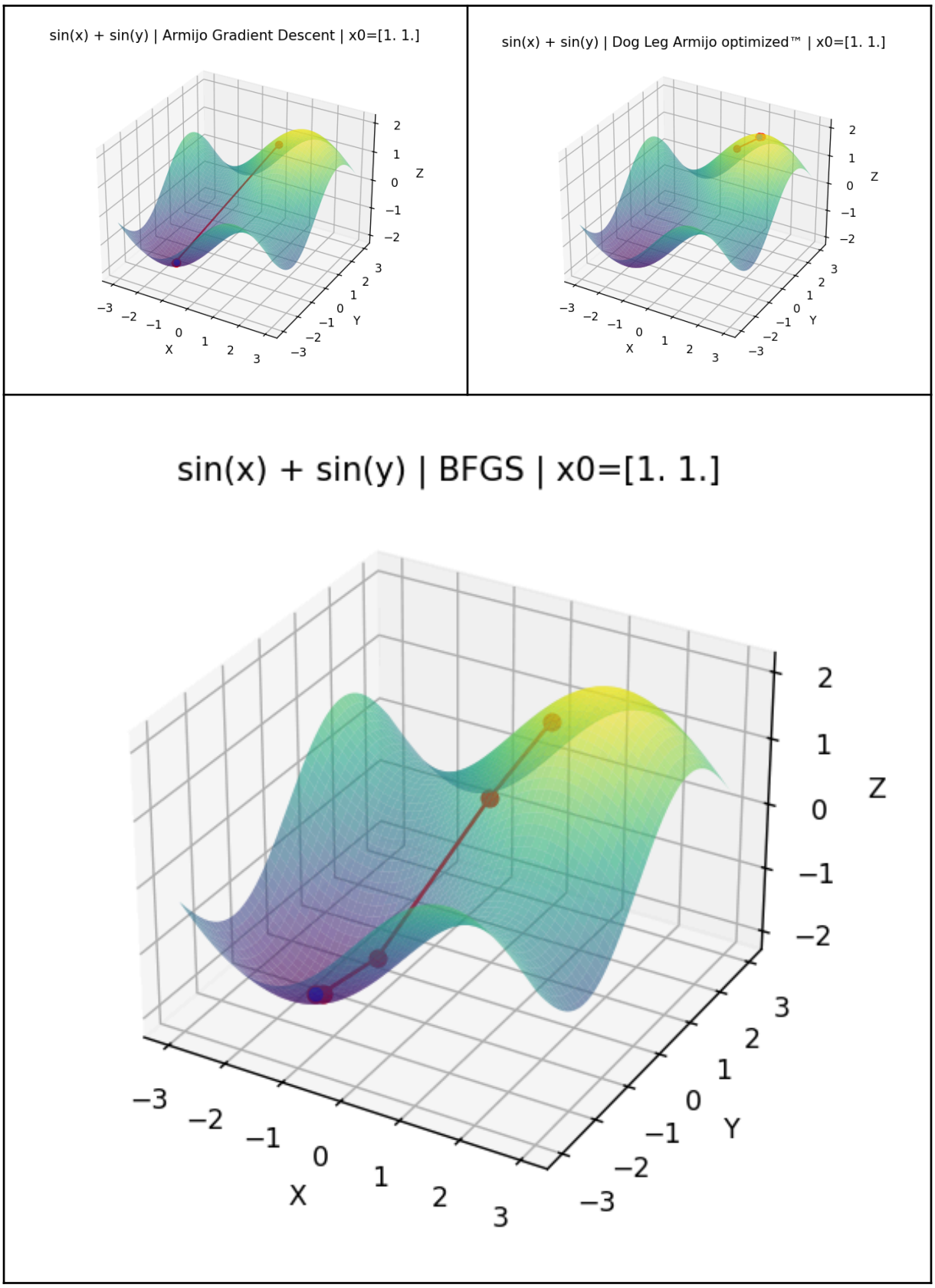
Будем запускать из точки $x_0 = (1, 1)$

метод	кол-во итераций	кол-во вычисл. функции	кол-во вычисл. градиента	кол-во вычисл. гессиана	ошибка
GD LR const(0.1)	123	0	369	0	3.39e-10
GD LR exp(0.5)	41	0	41	0	4.31e-1
Armijo GD	14	654	14	0	6.66e-15
Dichotomy GD	2	157	2	0	0.00e+0
Scipy Wolfe GD	6	195	20	0	4.88e-15
Damped Newton	97	0	291	97	4.00e+0
Damped Newton Opt	9	0	9	9	4.00e+0
Dog Leg Armijo	6	134	6	6	4.00e+0
Dog Leg Armijo Opt	6	134	6	6	4.00e+0
BFGS	4	13	12	0	4.17e-5
Scipy Newton-CG	4	7	7	4	0.00e+0
Scipy BFGS	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 Sciру 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 $x_0 = (2, 2)$

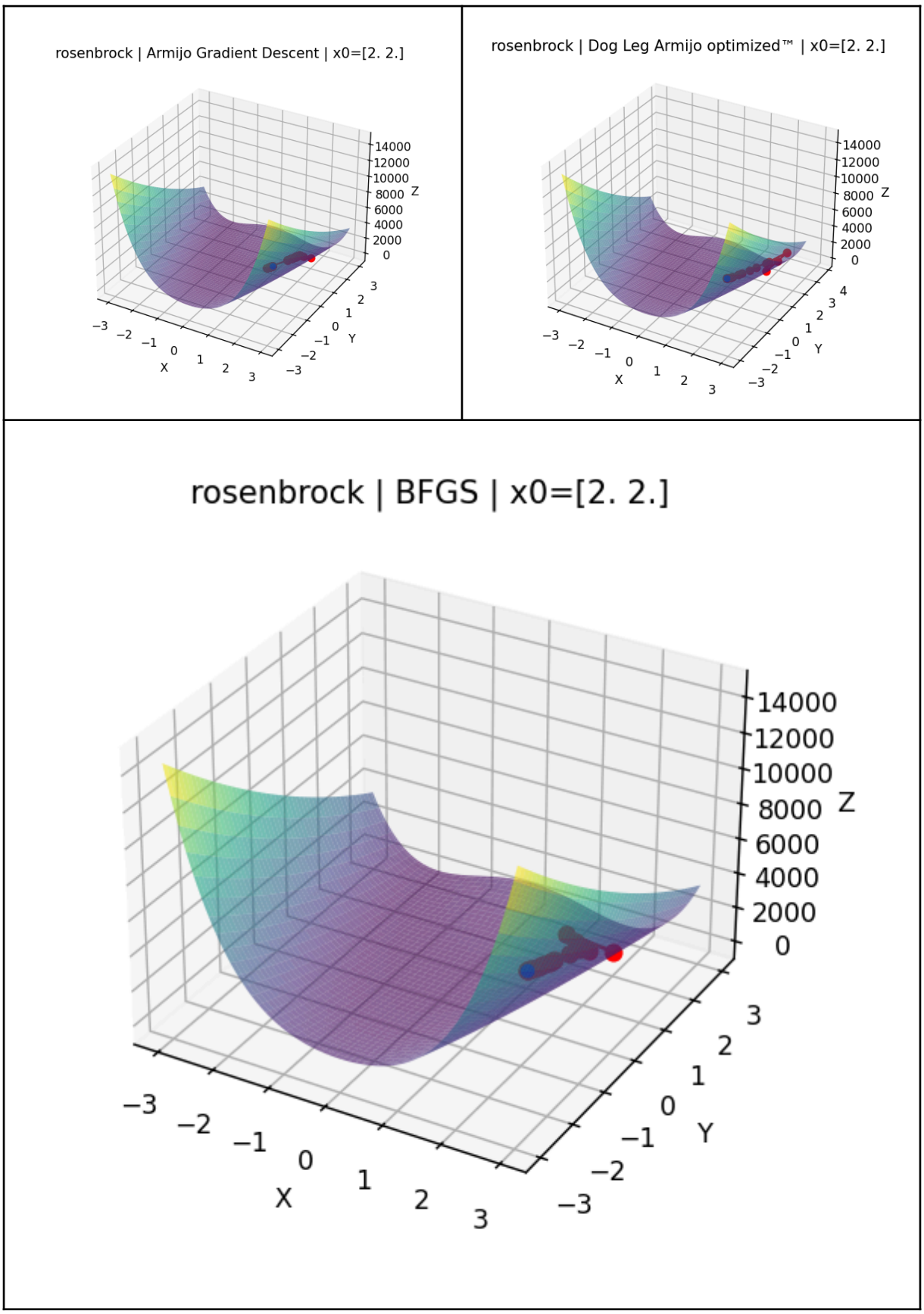
метод	кол-во итераций	кол-во вычисл. функции	кол-во вычисл. градиента	кол-во вычисл. гессиана	ошибка
GD LR const(0.1)	3	0	9	0	6.25e+106
GD LR exp(0.5)	4	0	4	0	3.99e+154
Armijo GD	601	9082	601	0	4.94e-6
Dichotomy GD	220	11112	220	0	7.46e-9
Scipy Wolfe GD	5675	54521	17025	0	1.87e-8
Damped Newton	120	0	360	120	2.28e-5
Damped Newton Opt	19	0	19	19	4.00e-8
Dog Leg Armijo	14	966	14	14	4.00e-8
Dog Leg Armijo Opt	17	1026	17	17	4.00e-8
BFGS	22	37	35	0	4.28e-7
Scipy Newton-CG	27	64	52	28	3.14e-10
Scipy BFGS	30	35	35	0	4.00e-8

Наиболее эффективным оказался демпфированный метод Ньютона с оптимизированными гиперпараметрами.

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

BFGS вновь показал отличные результаты. Он нашел минимум с достаточно малой погрешностью за низкое число итераций.

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


rosenbrock | Armijo Gradient Descent | x0=[2. 2.]
rosenbrock | Dog Leg Armijo optimized™ | x0=[2. 2.]
rosenbrock | BFGS | x0=[2. 2.]
X
Y
Z

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 отличной альтернативой методу Ньютона, когда вычисление гессиана затратно.

При этом на некоторых функциях (например, функции Розенброка) эти методы особенно эффективны по сравнению с методами оптимизаций первого порядка.

Оптимизация гиперпараметров может дать значительный прирост к эффективности некоторых методов (например, демпфированный метод Ньютона), но для некоторых может даже ее ухудшить (например, увеличение числа итераций/вычислений фукнции для Dogleg).