Национальный исследовательский университет ИТМО

Факультет информационных технологий и программирования

Прикладная математика и информатика

**Методы оптимизации**

**Отчет по лабораторной работе №2**

Работу выполнили:

Булавко Тимофей Евгеньевич M3237

Илляхунов Ансар Адылович M3237

Преподаватель:

Андреев Юрий Александрович

г. Санкт-Петербург

2024 г.

**Постановка задач:**

1. Реализация метода Ньютона с постоянным шагом
2. Реализация любого метода одномерного поиска и метода Ньютона на его основе
3. Реализация метода Newton-CG и один-два квазиньютоновских метода используя готовую реализацию в Python библиотеке scipy.optimize.
4. Провести исследование. Для исследования выбрать функцию типа Розенброка, а также одну-две полиномиальные функции; Сравнить методы на каждой из этих функциях:

a) сравнить свою реализацию метода Ньютона с методом Newton-CG из библиотеки scipy.optimize (по точности и скорости)

b) сравните эффективность методов нулевого порядка и градиентного спуска из лаб 1., метода Ньютона, квазиньютоновских методов.

c) 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)

d) Проиллюстрировать примеры.

**Доп. Задание:**

Одномерный поиск по правилу Вольфе (Wolfe) и метод Ньютона на его

основе; сравните с реализованным методами Ньютона и методом Newton-CG;

**Задача 1: Метод Ньютона с постоянным шагом**

В общем случае процесс нахождения экстремума функции является итерационным:

***,*** *где*

*X – вектор столбец параметров,*

*– градиент*

*– квадратная матрица вторых частных производных (матрица Гессе)*

*– постоянный шаг (если функция квадратична, то при , метод обязательно приведет к минимуму)*

*Итерационный процесс продолжает пока не будет выполнен некоторый критерий останова, например, траектория поиска остается в малой окрестности текущей точки поиска.*

Алгоритм реализуется следующим образом:

1. Выбрать начальную точку ***X0***
2. Вычислить градиент текущей точки и матрицу гессе
3. Перейти в следующую точку по формуле
4. Проверить критерий останова

Реализацию алгоритма можно найти по [ссылке](https://github.com/weldfy/Zero-order-methods/blob/main/gradient_descent_constant_step.py).

Функция ***newton\_method***:

***grad\_func*** — градиент функции.

***x*** — начальная точка нашего приближения.

**tol** - минимальная погрешность.

***h*** — длина шага.

***max\_iter*** — максимальное количество итераций метода.

В реализации использовалась библиотека numpy, а именно модуль linalg (для нахождения шага)

**Задача 2: Метод Ньютона с золотым сечением**

В данном алгоритме реализация остается такой же только теперь мы будем вычислять длину шага с помощью **золотого сечения**.

Для поиска минимума функции (в нашем случае мы будем перебирать коэффициенты на который будет домножен наш градиент) мы будем действовать так:

1. Заведем переменную *phi = (1 + sqrt(5)) / 2*
2. У нас будут границы поиска ***l*** и ***r*** (левая и правая границы соответственно). После мы вычислим значение функции в следующих точках:

*x = l + (r — l) / (phi + 1)*

*y = r — (r — l) / (phi + 1)*

и посчитаем значение в этих точках (***f1*** и ***f2*** соответственно)

1. если *f1 < f2*

*r = y*

*y = x*

*f2 = f1*

*x = l + (r — l) / (phi + 1)*

*f2 = f(y)*

в противном случае:

*l = x*

*x = y*

*f1 = f2*

*y = r — (r — l) / (phi + 1)*

*f2 = f(y)*

1. Остановим итерацию алгоритма когда нас будет устраивать точность то есть ***r-l*** по модулю будет меньше ***eps***

Реализацию можно найти по [ссылке](https://github.com/weldfy/Zero-order-methods/blob/main/gradient_descent_with_golden_section.py).

Функция ***newton\_method\_with\_line\_search***:

**func** – функция.

***grad\_func*** — градиент функции.

***x*** — начальная точка нашего приближения.

**tol** - минимальная погрешность.

***max\_iter*** — максимальное количество итераций метода.

В реализации использовалась библиотека numpy, а именно модуль linalg (для нахождения шага).

**Задача 3: scipy.optimize**

**Метод Newton-CG** является гибридом метода Ньютона и метода сопряженных градиентов, поэтому он более универсален и подходит для решения задач без ограничений. Основная идея заключается в ускорении вычисления шага

путем **приближенного** решения линейного уравнения

с помощью метода **сопряженных градиентов**.

**Гиперпараметры:**

**maxiter** – максимальное кол-во итераций

**c1** – параметр для правила Армихо (по дефолту 1e-4)

**c2** – параметр для условия кривизны (по дефолту 0.9)

**xtol** – параметр для останова по средней относительной ошибки решения

**Метод BFGS ­**является одним из наиболее широко применяемым **квазиньютоновским методом**. Основная идея, как и во всех квазиньютоновских методах, вычислить гессиан приближенно, исходя из сделанных уже шагов.

**Гиперпараметры:**

**maxiter** – максимальное кол-во итераций

**c1** – параметр для правила Армихо (по дефолту 1e-4)

**c2** – параметр для условия кривизны (по дефолту 0.9)

**hess\_inv0** – начальная матрица для подсчета гессиана (по дефолту Identity)

**gtol** – параметр для останова по значению градиента

**Доп. задание: Метод Ньютона по правилу Вольфе**

Алгоритм основанный на методе Ньютона, в котором шаг вычисляется по определенному правилу.

Идея состоит в подборе такого коэффициента

чтобы выполнялись два условия:

(Армихо)

(кривизна)

Где c1 и c2 подбираются так, что 0 < c1 < c2 < 1. Обычно c1 берут маленьким, чтобы функция после шага уменьшилась, но побыстрее (здесь надо соблюсти баланс), а c2 берут около 1, чтобы градиент функции при новом шаге существенно изменился (поменял направление или уменьшился).

Реализацию можно найти по [ссылке](https://github.com/weldfy/Zero-order-methods/blob/main/gradient_descent_with_golden_section.py).

Функция ***newton\_method\_wolfe***:

**func** – функция.

***grad\_func*** — градиент функции.

**hess\_func** ­– гессиан функции

***x*** — начальная точка нашего приближения.

**tol** - минимальная погрешность.

***max\_iter*** — максимальное количество итераций метода.

В реализации использовалась библиотека numpy, а именно модуль linalg (для нахождения шага).

**Исследование функции:**

Для исследования были выбраны две полиномиальные функции и функция розенброка:

***Метод Ньютона с фиксированным шагом:***

Метод заслуживает внимания, отлично справляется с выпуклыми функциями, довольно быстрый, но сильно зависит от выбранного шага. Слишком маленький шаг повысит потенциальную точность алгоритма, однаком может сильно сказаться на его скорости, а так же устойчивости, ведь алгоритм будет застревать в локальных экстремумах. В свою очередь большой шаг может существенно ускорить время алгоритма, потенциально будет способен выйти из точек локальных экстремумов, но во вред точности.

**Метод Ньютона с золотым сечением:**

Данный метод будет лучше показывать себя при высокой точности, так как количество приближений будет меньше. При этом будет много вызовов функции, поэтому если получение значения функции - тяжелая операция, этот метод будет не эффективен. При этом метод Ньютона с золотым сечением часто предпочтительнее метода Ньютона с фиксированным шагом, так как он обеспечивает более точную и быструю сходимость к минимуму функции

***Метод Newton-CG***

Очень быстро находит значение близкое к экстремуму, но там же начинает крутиться в этой каше и производить много лишних операций около найденного локального экстремума, так метод глабольный экстремум находит плохо и не слишком устойчив. Из плюсов можно отметить, что не нужно вычислять гессиан.

**Метод BFGS**

Показал себя лучше своего собрата, быстро находит значение экстремума, но дальше долго крутиться, чтобы добиться нужной точности. Справляется наура в большинстве случаев.

***Метод Ньютона по правилу Вольфе***

*Данный подход хорошо показал себя для гладких функций, но встретил неприятности на выпуклостях. Метод гарантирует сходимость при выборе оптимального шага, это позволяет избежать лишних расходов и повышает устойчивость алгоритма. Он превзошел метод Ньютона с фиксированным шагом, но в среднем при выборе рандомных начальных значений проиграл по эвристике золотому сечению.*

**Все графики по которым происходили исследования можно найти по** [**ссылке**](https://github.com/weldfy/Zero-order-methods)**.**