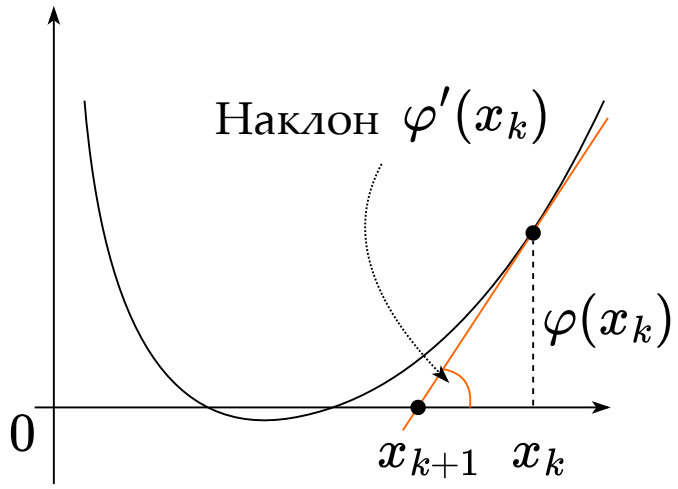
Метод Ньютона и квазиньютоновские методы

Даня Меркулов, Петр Остроухов

1 Метод Ньютона

1.1 Идея метода Ньютона для нахождения корней функции



Рассмотрим функцию $\varphi(x) : \mathbb{R} \to \mathbb{R}$.

Основная идея заключается в том, чтобы построить линейное приближение в точке x_k и найти его корень, который будет новой точкой итерации:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_k - x_{k+1}}$$

Мы получаем итерационную схему:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Этот метод станет методом оптимизации Ньютона в случае $f'(x) = \varphi(x)$[1]:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

[1]Мы фактически решаем задачу нахождения стационарных точек $\nabla f(x) = 0$

1.2 Метод Ньютона как оптимизация локальной квадратичной аппроксимации

Пусть у нас есть функция $f(x)$ и некоторая точка x_k. Рассмотрим квадратичное приближение этой функции в окрестности x_k:

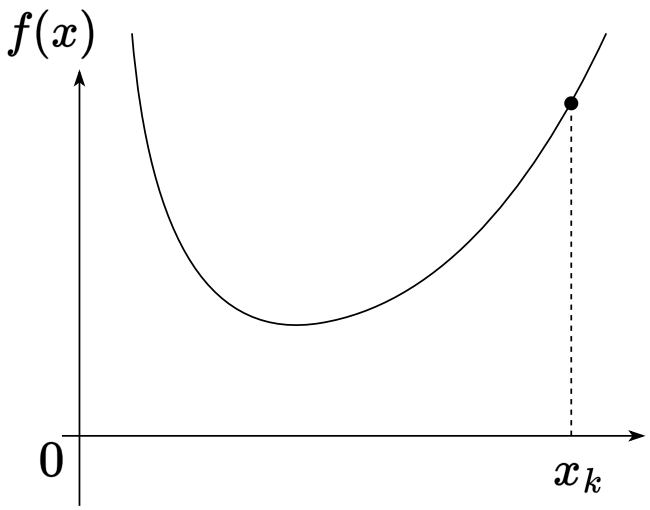
$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2}\langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

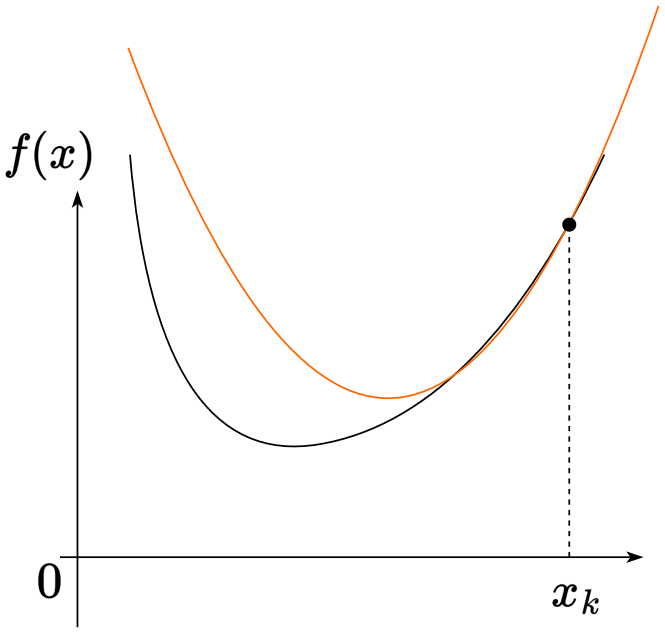
Идея метода заключается в том, чтобы найти точку x_{k+1}, которая минимизирует функцию $f_{x_k}^{II}(x)$, т.е. $\nabla f_{x_k}^{II}(x_{k+1}) = 0$.

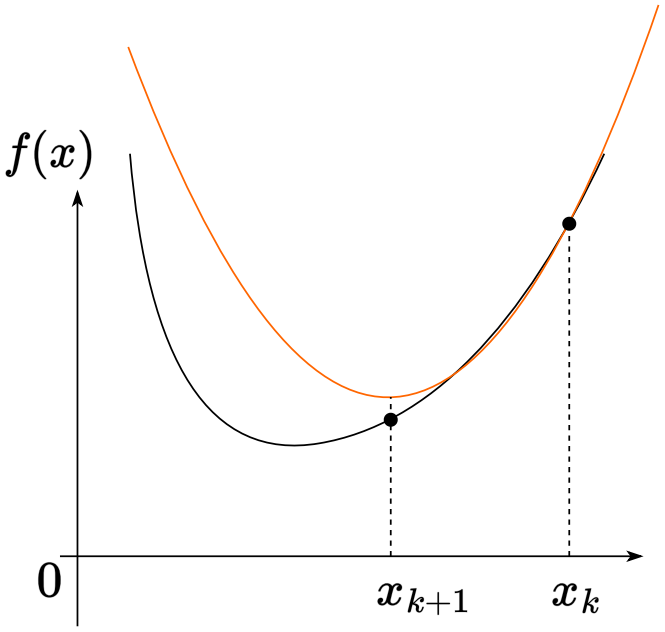
$$\begin{aligned}
\nabla f_{x_k}^{II}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

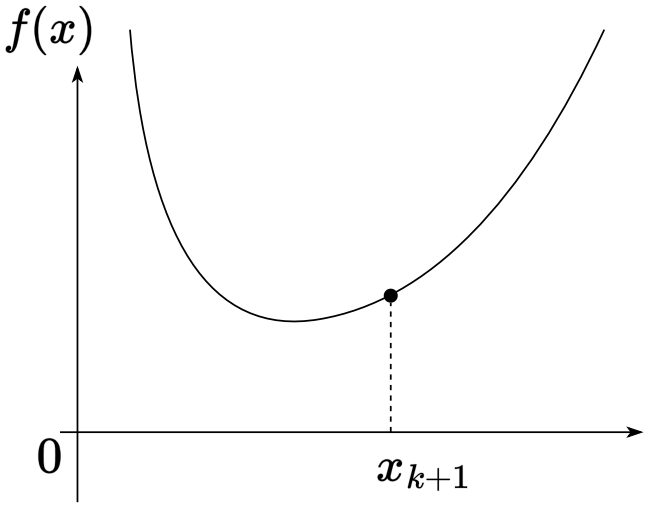
Необходимо отметить ограничения, связанные с необходимостью невырожденности (для существования метода) и положительной определенности (для гарантии сходимости) гессиана.

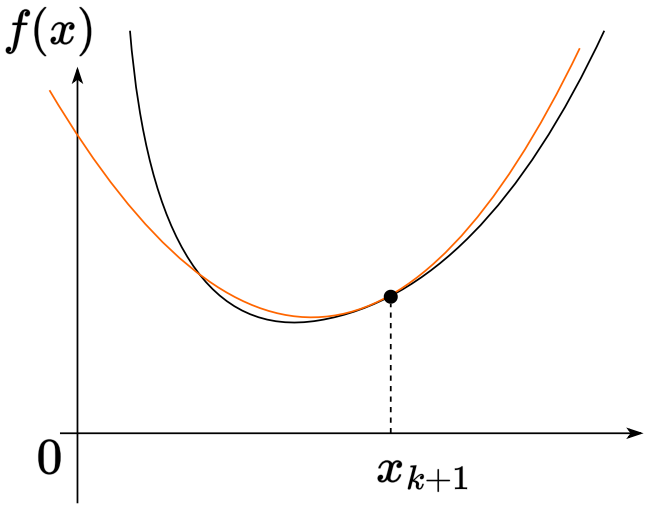
1.3 Метод Ньютона как оптимизация локальной квадратичной аппроксимации

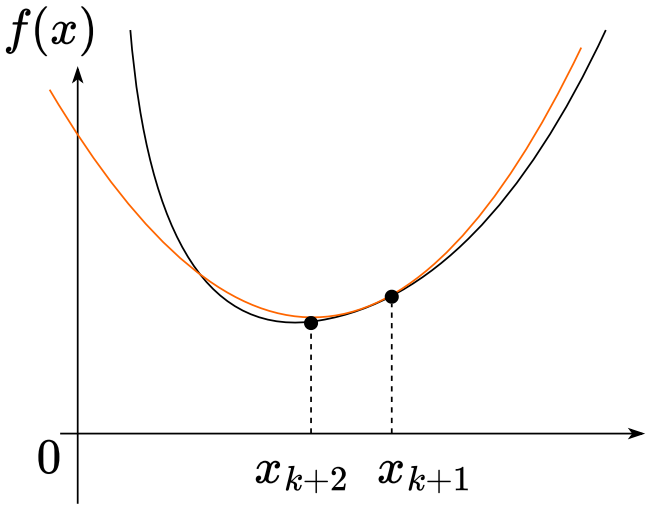



$f(x)$
0
x_k


$f(x)$
0
x_{k+1}
x_k


$f(x)$
0
x_{k+1}


$f(x)$
0
x_{k+1}



2 Свойства метода Ньютона

2.1 Квадратичная сходимость метода Ньютона

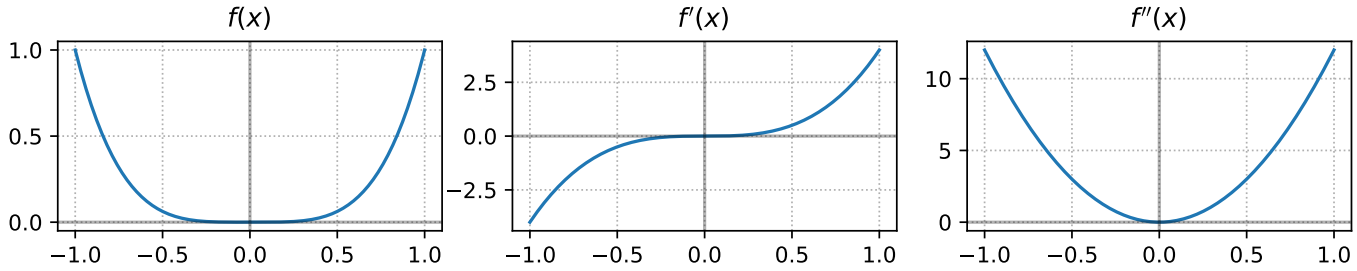
Theorem

Пусть $f(x)$ — сильно выпуклая дважды непрерывно дифференцируемая функция на $\mathbb{R}^n$, для второй производной которой выполняются неравенства: $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$. Пусть также гессиан функции M-липшицев. Тогда метод Ньютона сходится локально к решению с квадратичной скоростью, т.е. при $\|x_0 - x^*\| < \frac{2\mu}{3M}$:

$$\|x_{k+1} - x^*\| \leq \frac{3M}{2\mu}\|x_k - x^*\|^2$$

2.2 Отсутствие квадратичной сходимости, если некоторые предположения нарушаются

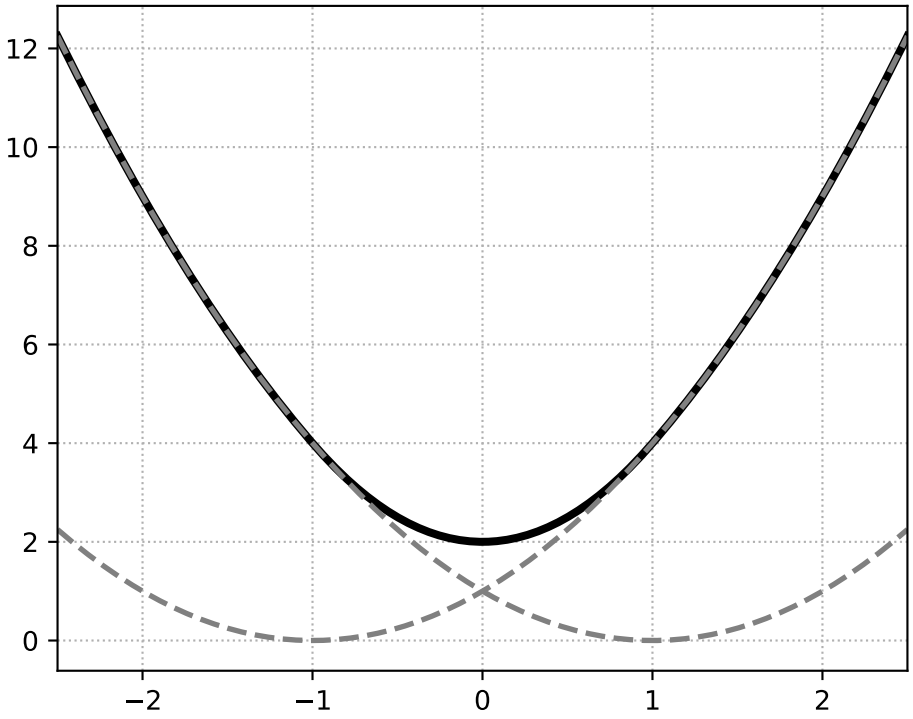
$$f(x) = x^4 \qquad f'(x) = 4x^3 \qquad f''(x) = 12x^2$$



$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} = x_k - \frac{4x_k^3}{12x_k^2} = x_k - \frac{1}{3}x_k = \frac{2}{3}x_k,$$

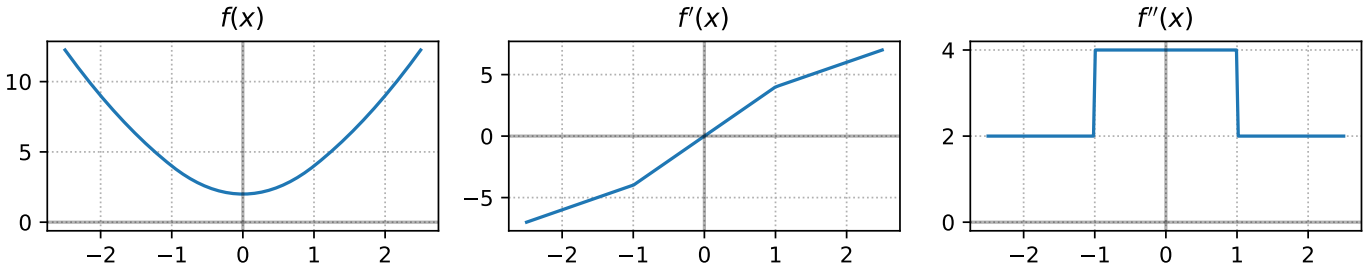
сходится линейно к 0, единственному решению задачи, с линейной скоростью.

2.3 Локальная сходимость метода Ньютона для гладкой сильно выпуклой $f(x)$

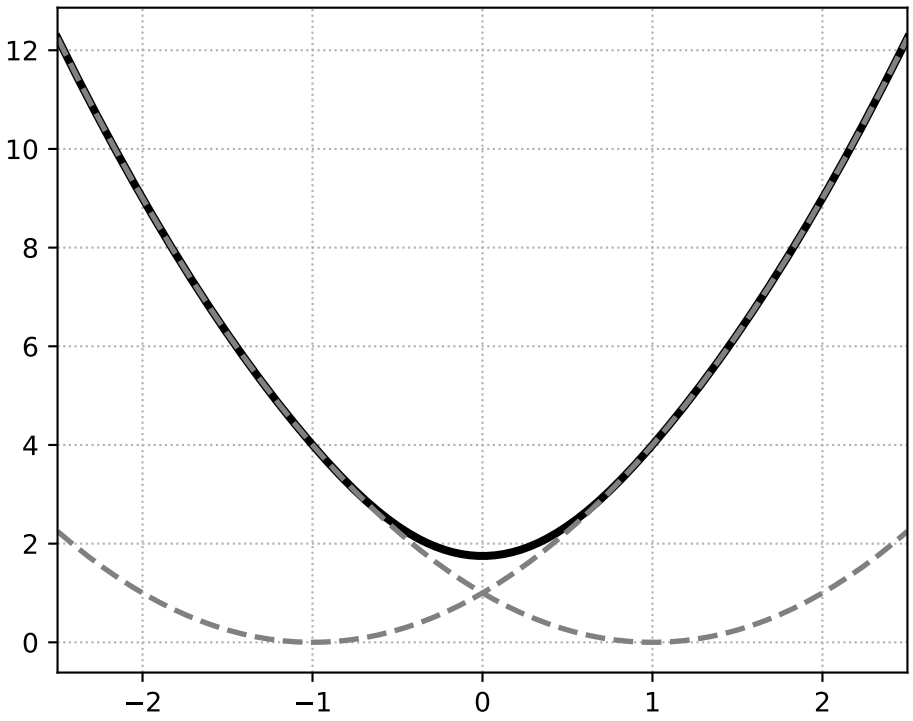


$$f(x) = \begin{cases} (x-1)^2, & x \le -1 \\ 2x^2 + 2, & -1 < x < 1 \\ (x+1)^2, & x \ge 1 \end{cases}$$

Эта функция сильно выпукла, но вторая производная не является липшицевой.

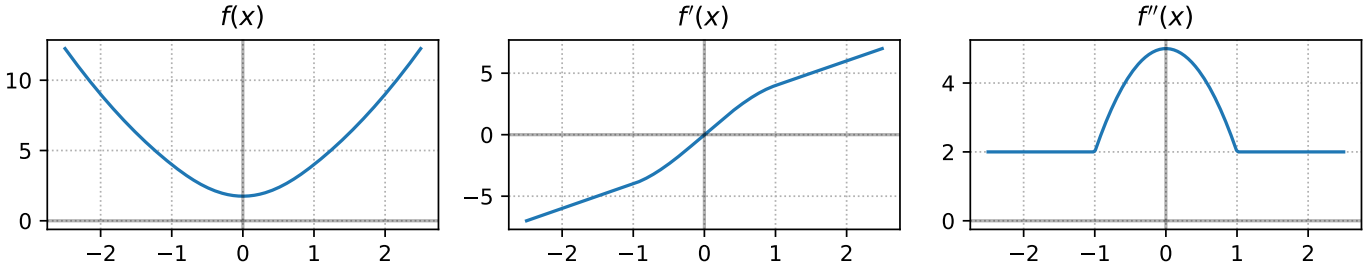


2.4 Локальная сходимость метода Ньютона даже если $\nabla^2 f$ липшицев

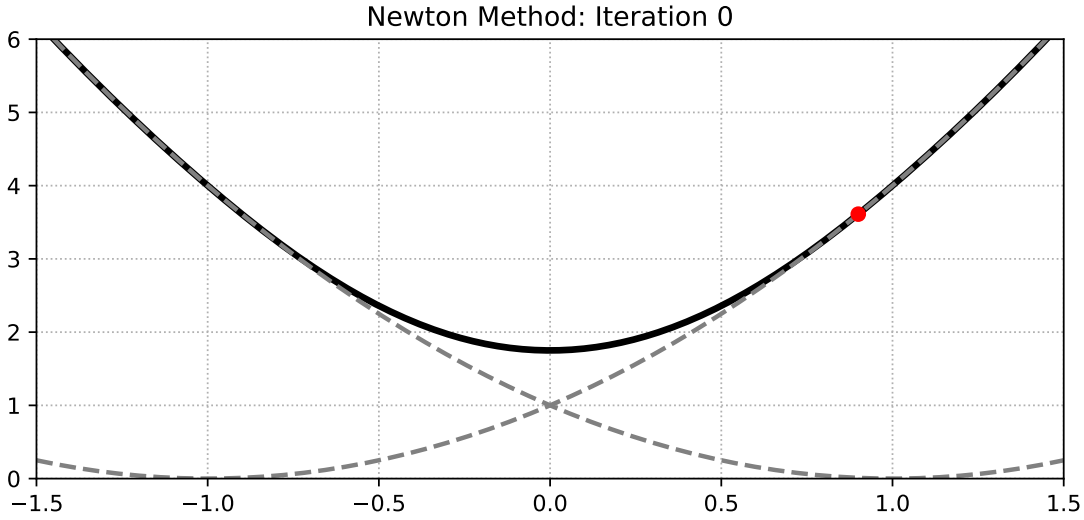


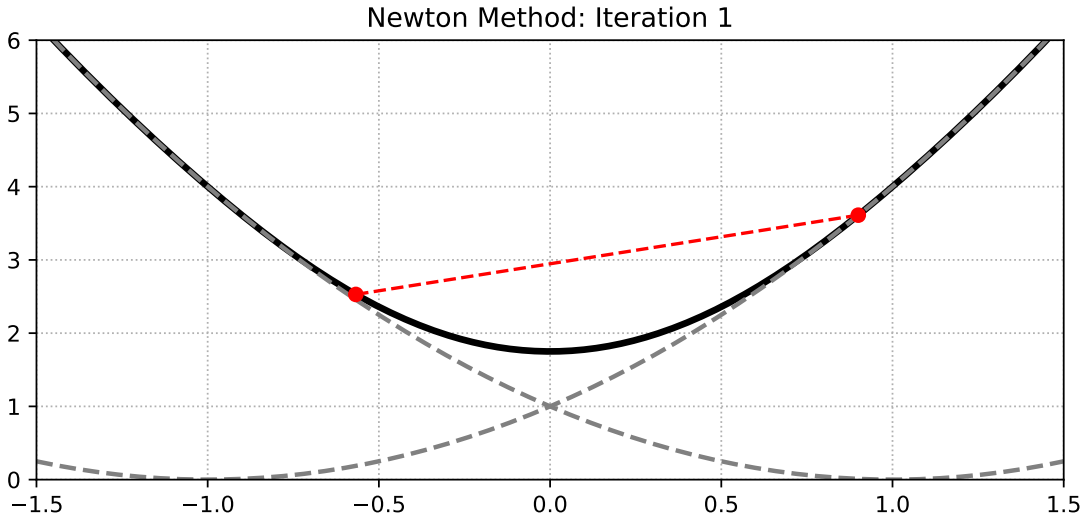
$$f(x) = \begin{cases} (x-1)^2, & x \leq -1 \\ -\frac{1}{4}x^4 + \frac{5}{2}x^2 + \frac{7}{4}, & -1 < x < 1 \\ (x+1)^2, & x \geq 1 \end{cases}$$

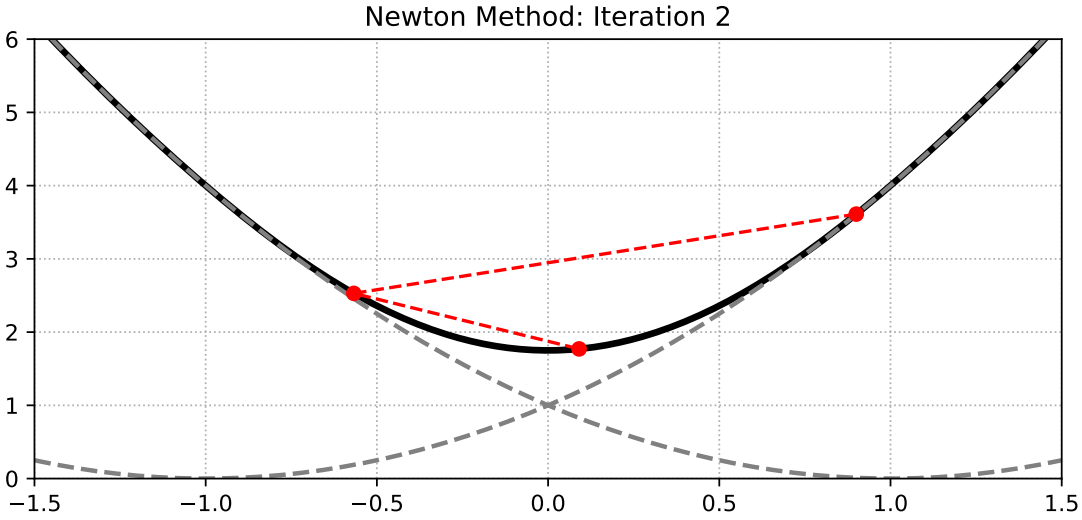
Эта функция сильно выпукла и вторая производная является липшицевой.

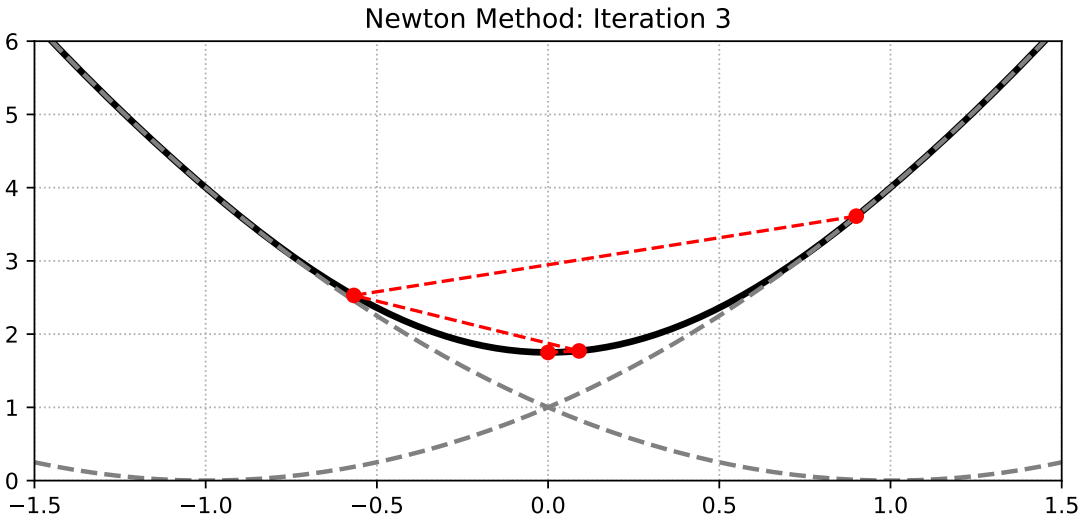


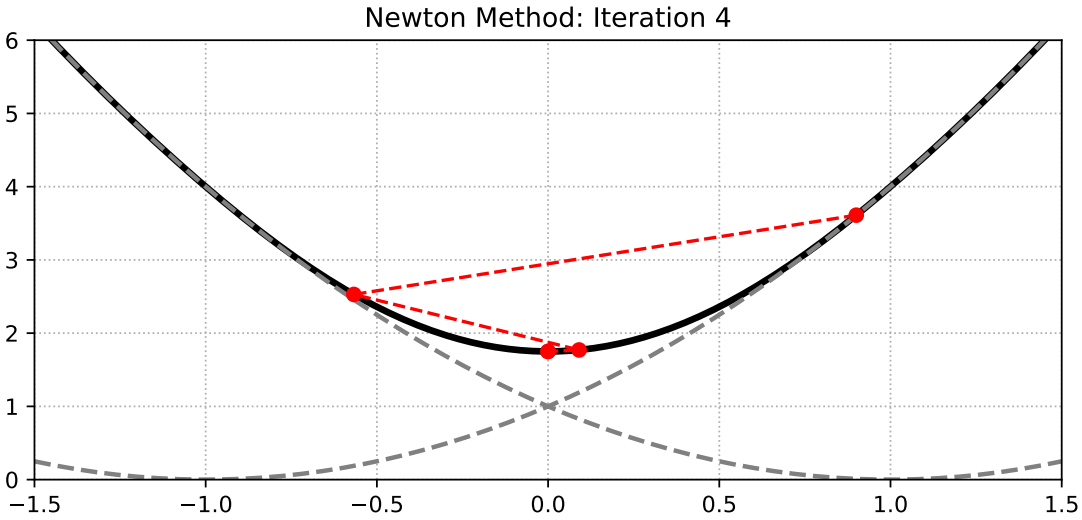
2.5 Локальная сходимость метода Ньютона. Хорошая инициализация

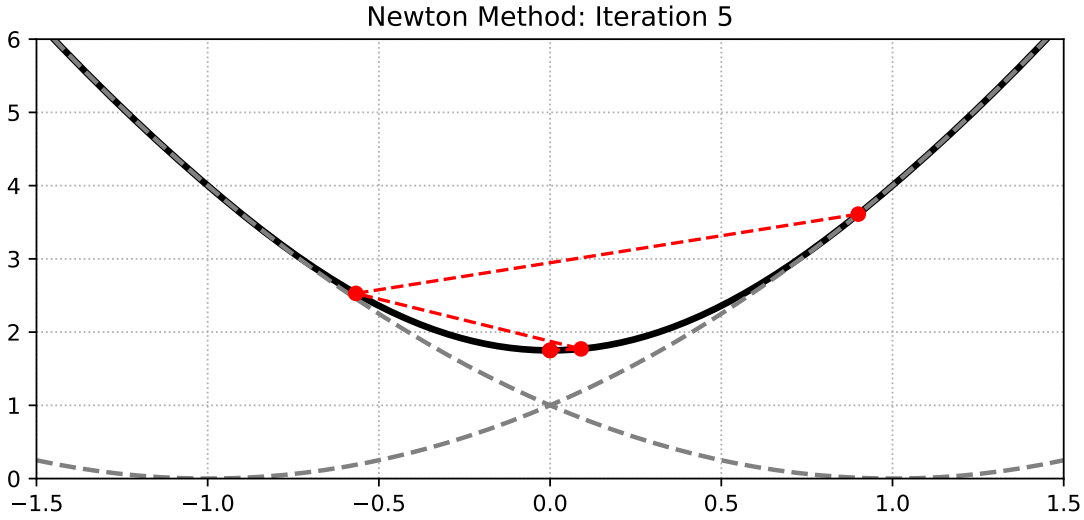




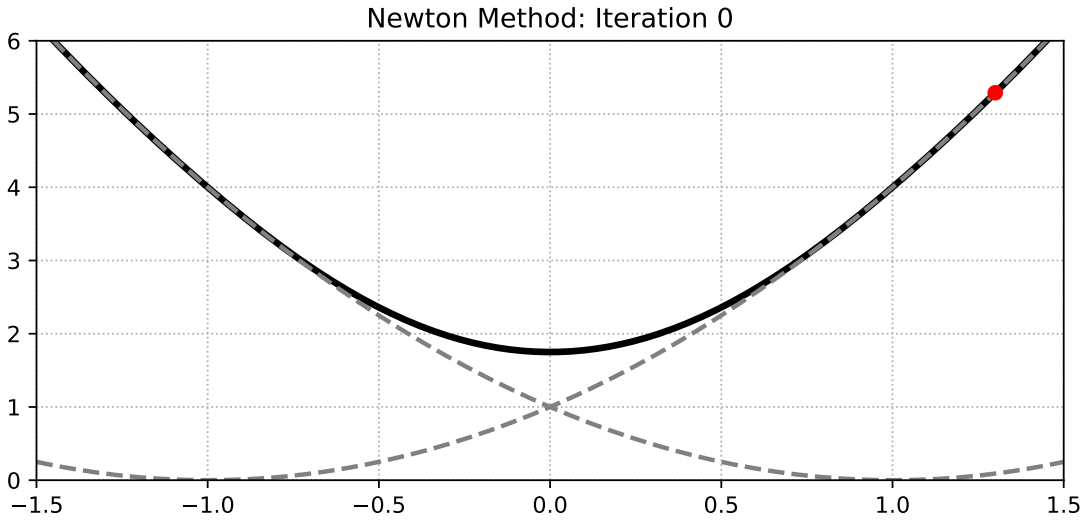

Newton Method: Iteration 2
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

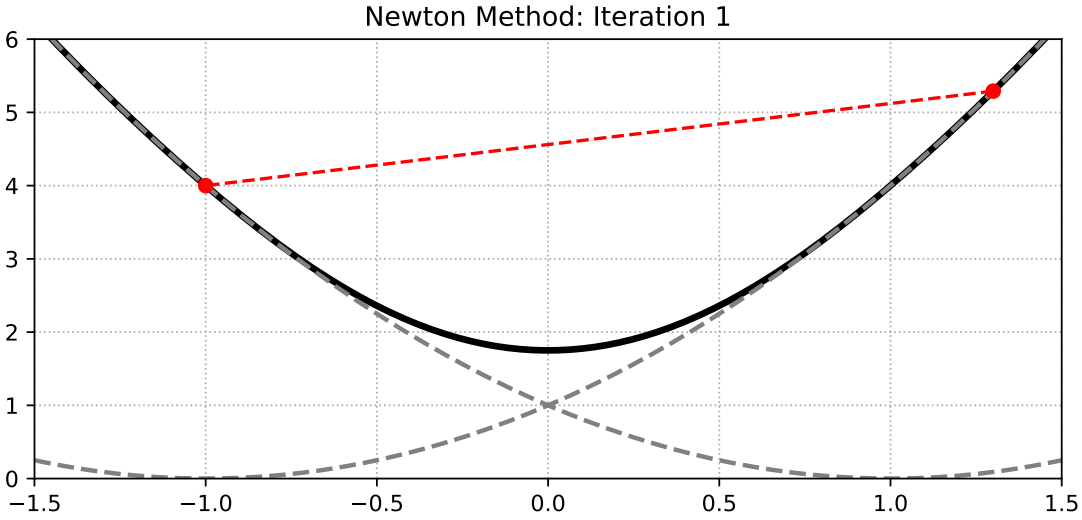

Newton Method: Iteration 3
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

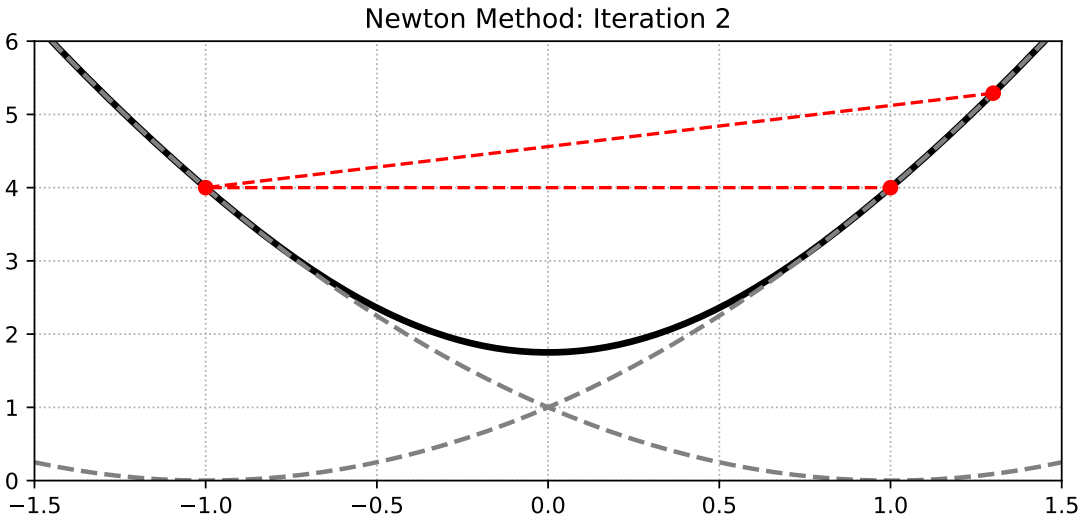

Newton Method: Iteration 4
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

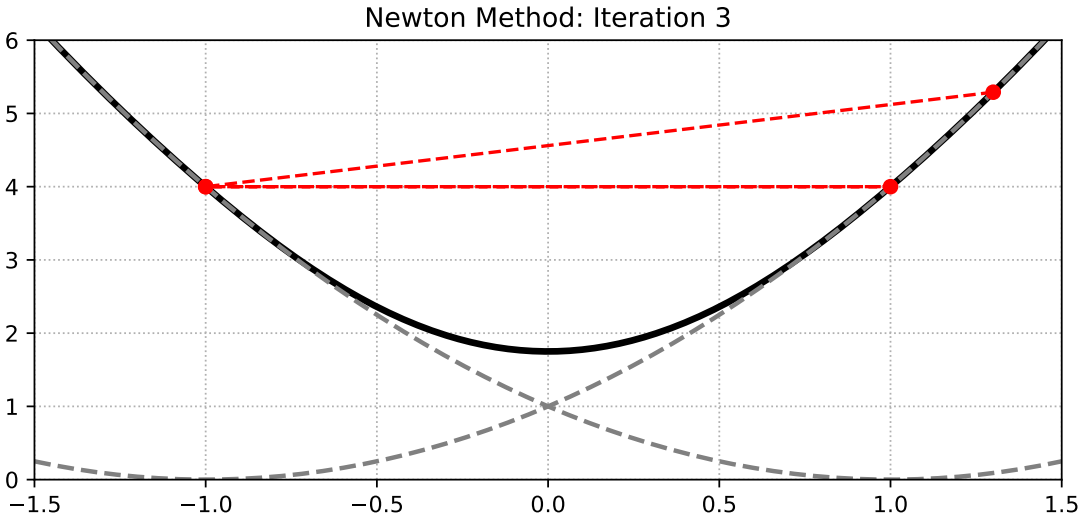


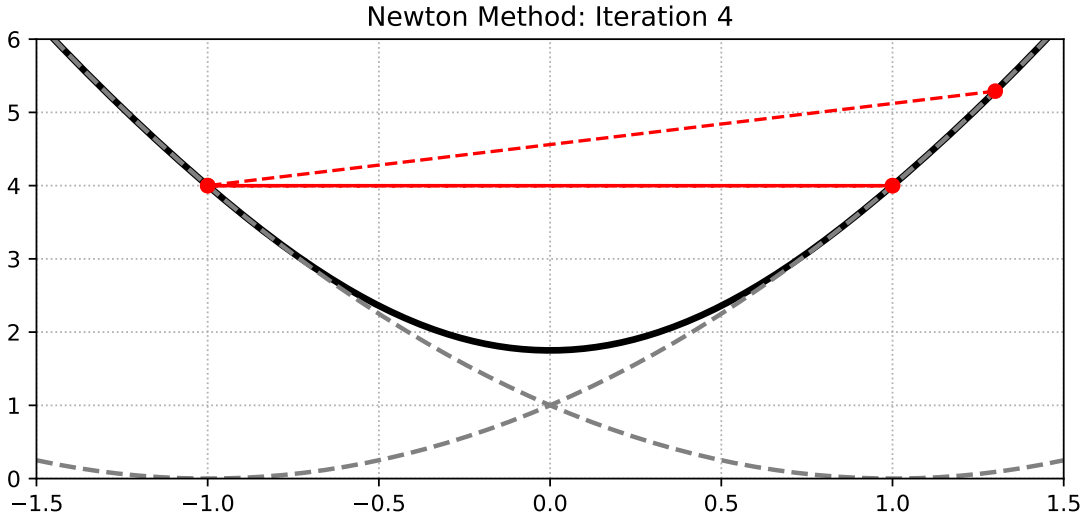
2.6 *Локальная сходимость метода Ньютона. Плохая инициализация*

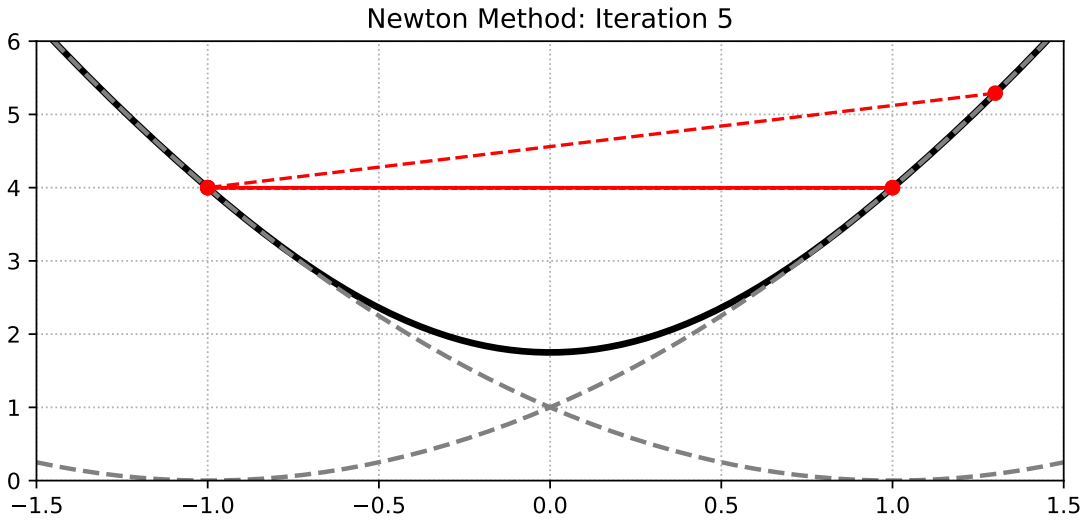



Newton Method: Iteration 1
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Newton Method: Iteration 2
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Newton Method: Iteration 3
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Newton Method: Iteration 4
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Newton Method: Iteration 5
6
5
4
3
2
1
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

2.7 Проблемы метода Ньютона

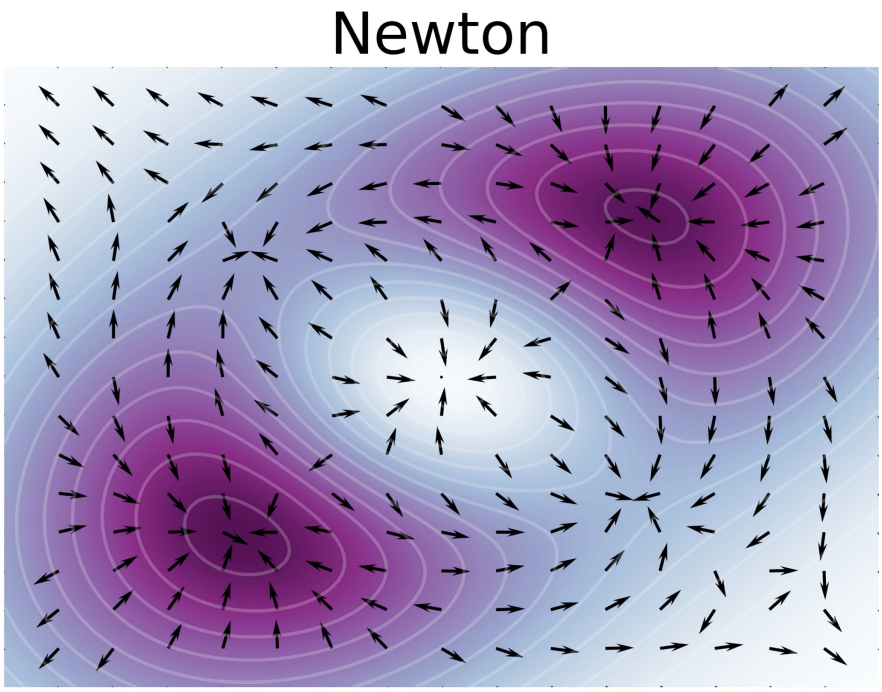


Рисунок 1: Animation

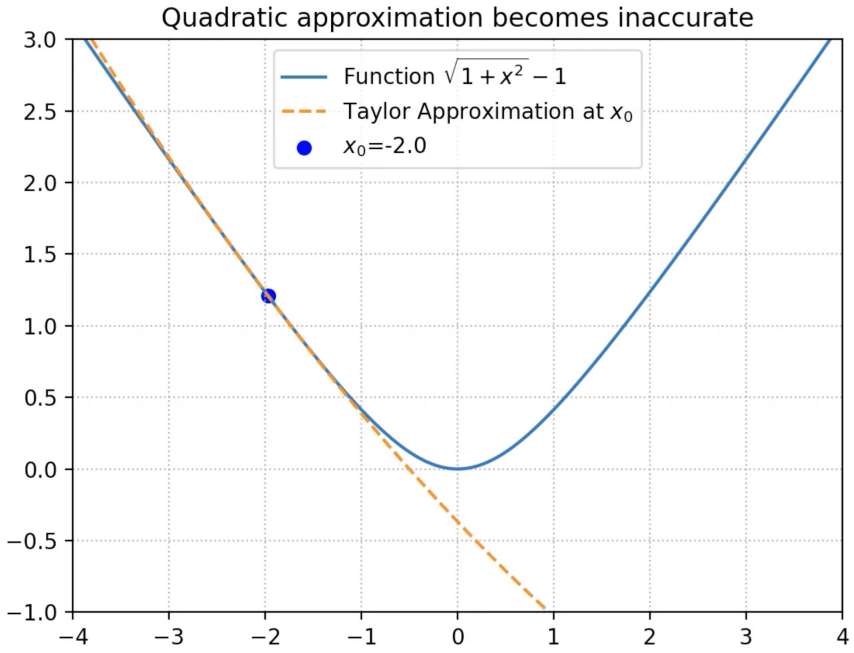


Рисунок 2: Animation

2.8 Метод Ньютона для квадратичной задачи (линейной регрессии)

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^\top A x - b^\top x, \qquad A \in \mathbb{R}^{n \times n}, \quad \lambda(A) \in [\mu; L].$$

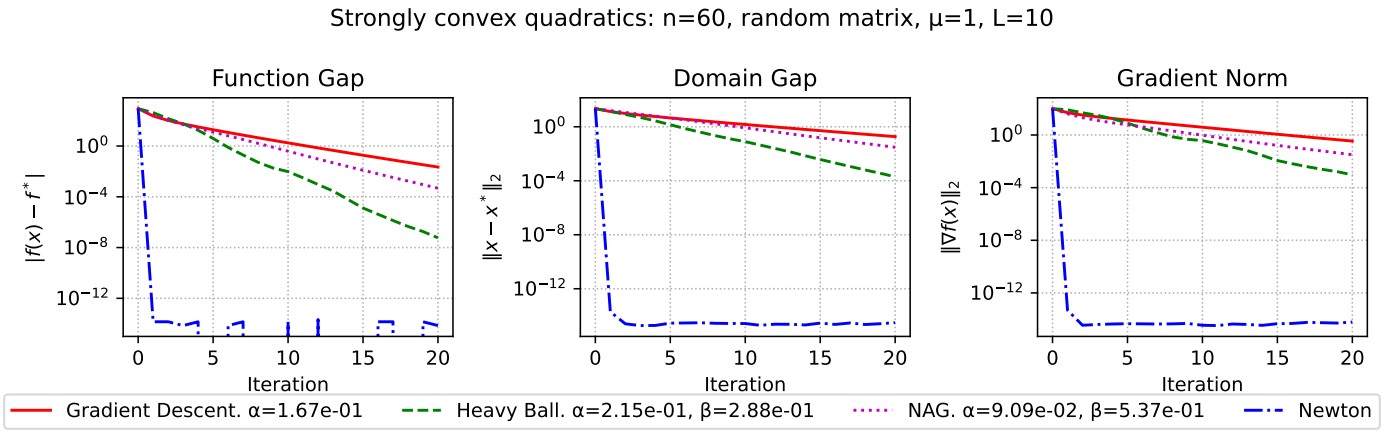


Рисунок 3: Так как задача - квадратичная, то метод Ньютона сходится за один шаг.

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^\top A x - b^\top x, \qquad A \in \mathbb{R}^{n \times n}, \quad \lambda(A) \in [\mu; L].$$

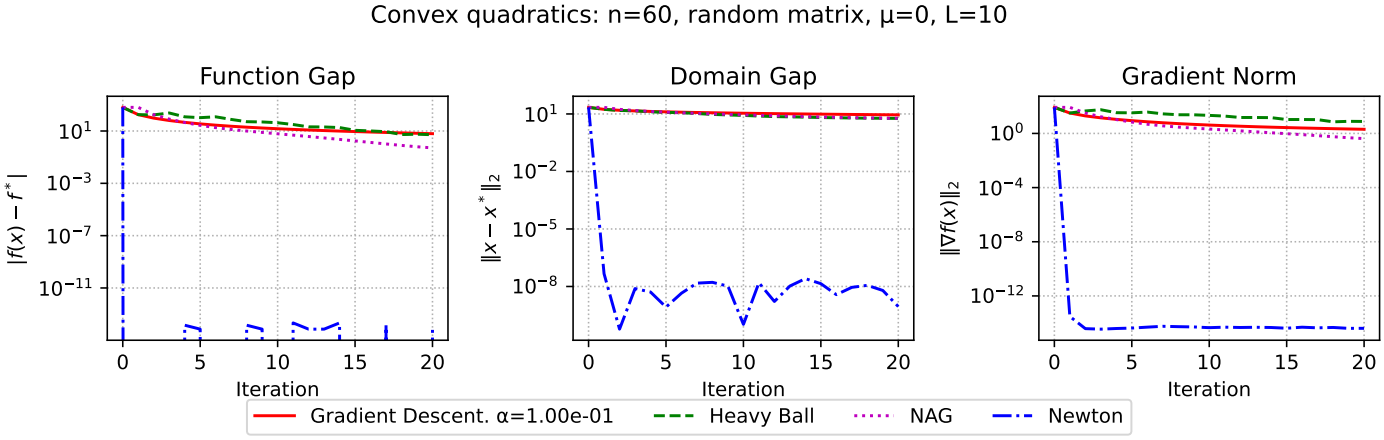


Рисунок 4: В этом случае метод Ньютона тоже крайне быстро сходится, однако, отметим, что это происходит благодаря тому, что минимальное собственное число гессиана не 0, а около 10^{-8}. Если применять метод Ньютона в наивной форме с обращением матрицы, то получится ошибка, так как матрица вырождена. На практике все равно можно использовать метод, если для направления итерации решать линейную систему $\nabla^2 f(x_k) d_k = -\nabla f(x_k)$ методом наименьших квадратов.

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^\top A x - b^\top x, \qquad A \in \mathbb{R}^{n \times n}, \quad \lambda(A) \in [\mu; L].$$

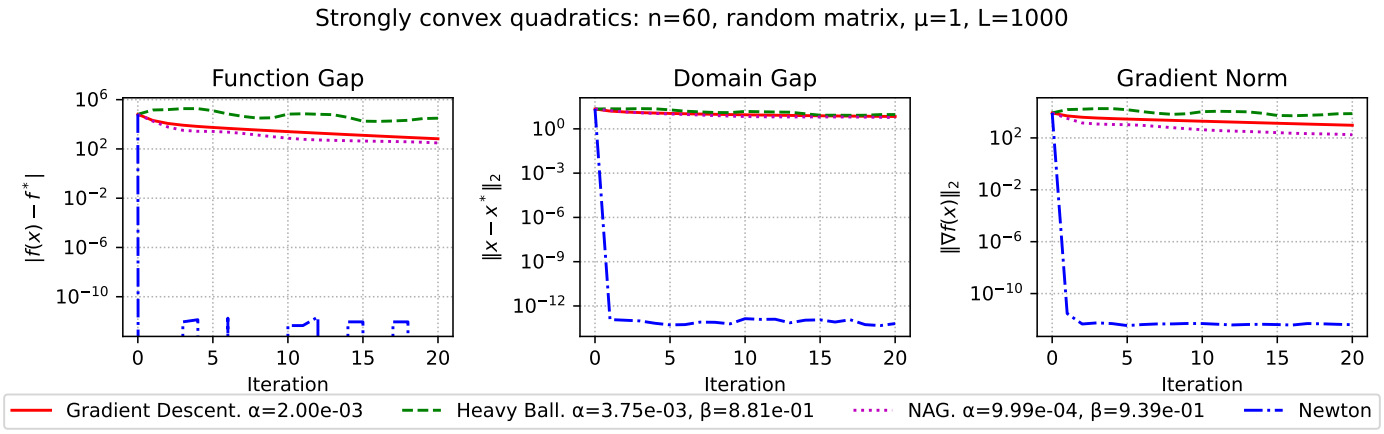


Рисунок 5: Здесь число обусловленности гессиана в 1000 раз больше, чем в предыдущем случае, и метод Ньютона сходится за 1 итерацию.

2.9 Метод Ньютона для задачи бинарной логистической регрессии

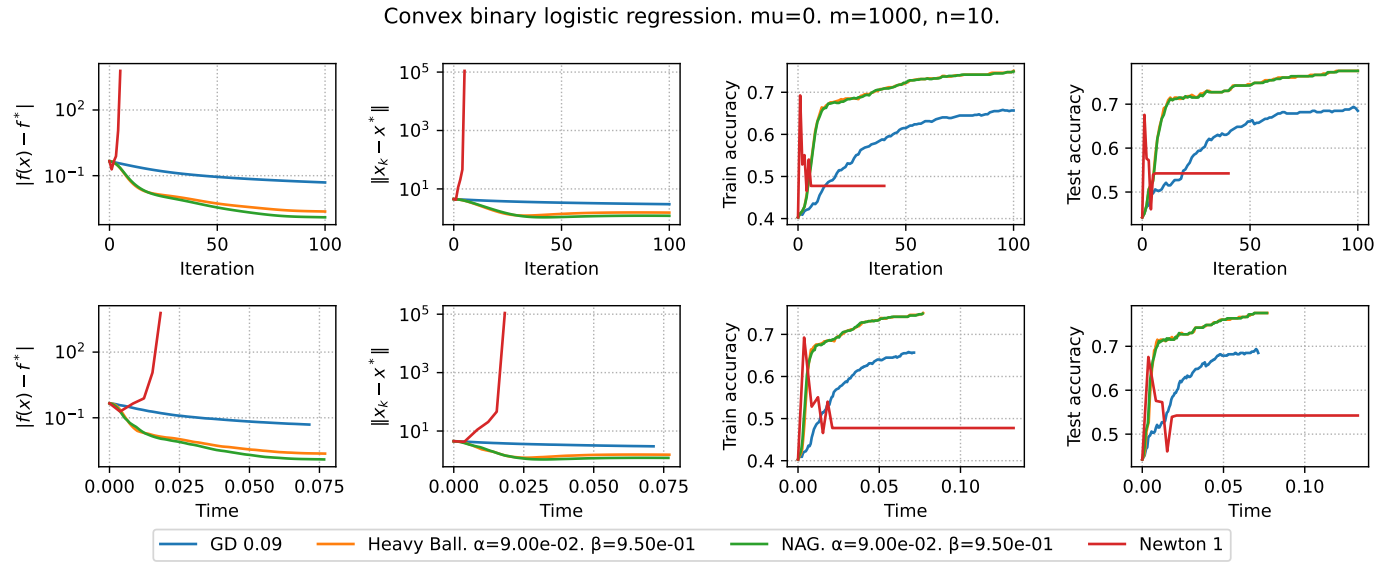


Рисунок 6: Наблюдается расходимость метода Ньютона. Сразу отметим, что в задаче нет регуляризации и гарантии сильной выпуклости. А также нет гарантий того, что мы инициализируем метод в окрестности решения.

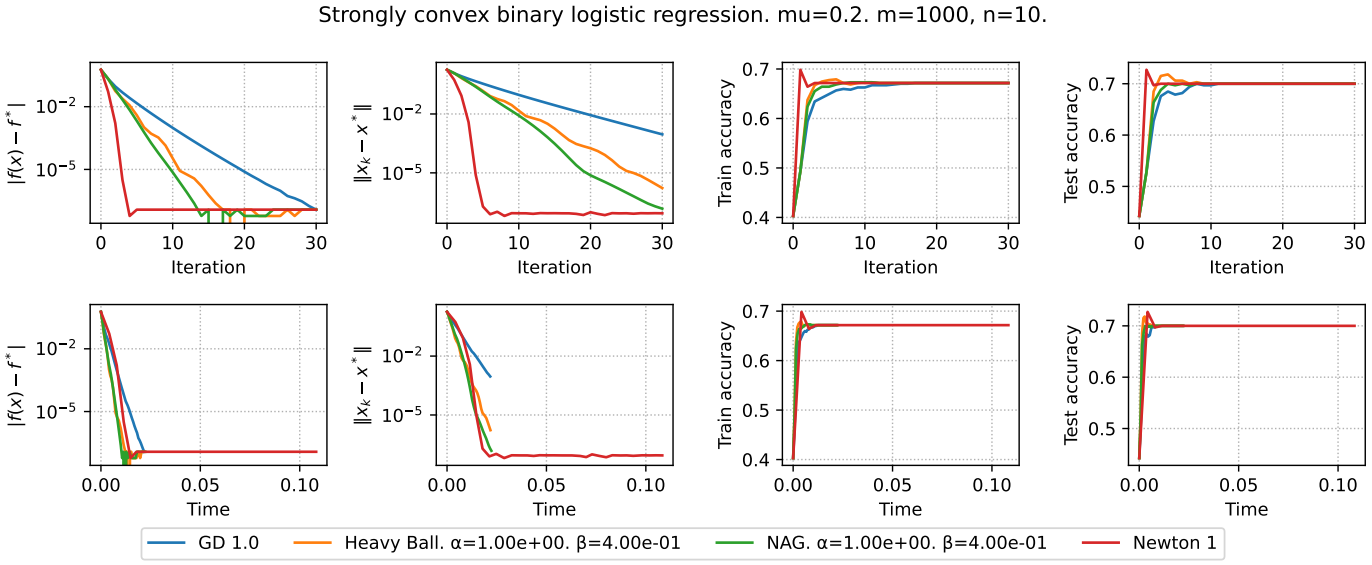


Рисунок 7: Добавление регуляризации гарантирует сильную выпуклость, наблюдается сходимость метода Ньютона.

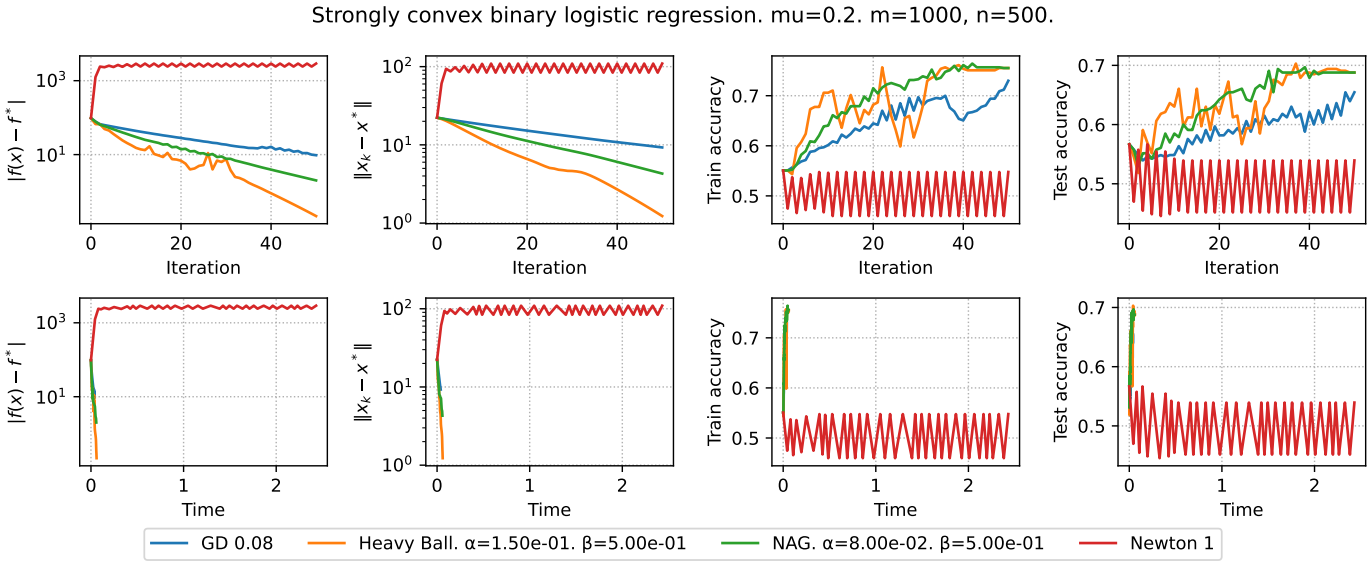


Рисунок 8: Увеличим размерность в 50 раз и наблюдаем расходимость метода Ньютона. Это можно связать с тем, что мы инициализируем метод в точке, далекой от решения

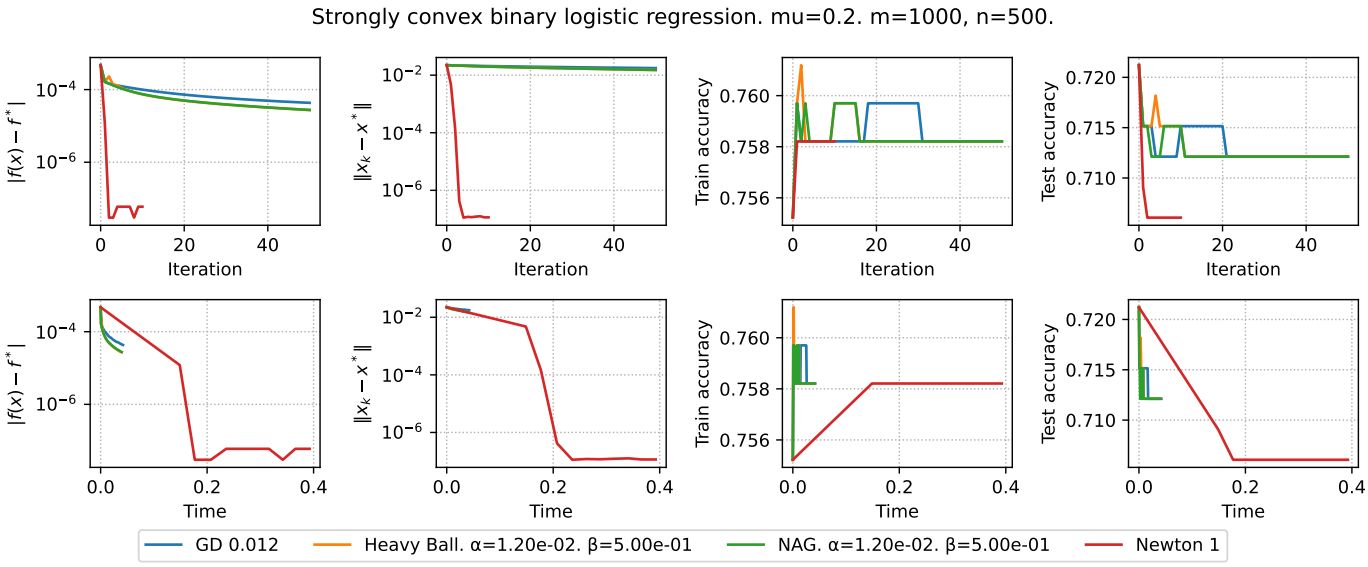


Рисунок 9: Не меняя задачу, изменим начальную точку и наблюдаем квадратичную сходимость метода Ньютона. Однако, обратите внимание на время работы. Уже при небольшой размерности, метод Ньютона работает значительно дольше, чем градиентные методы.

2.10 Задача нахождения аналитического центра многогранника

Найти точку $x \in \mathbb{R}^n$, которая максимизирует сумму логарифмов расстояний до границ многогранника:

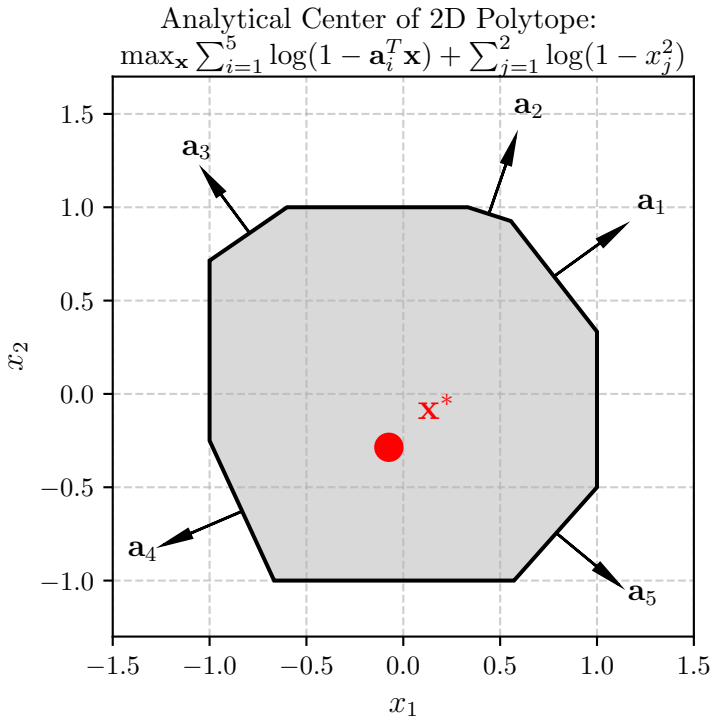
$$\max_x \sum_{i=1}^{m} \log(1 - a_i^T x) + \sum_{j=1}^{n} \log(1 - x_j^2)$$

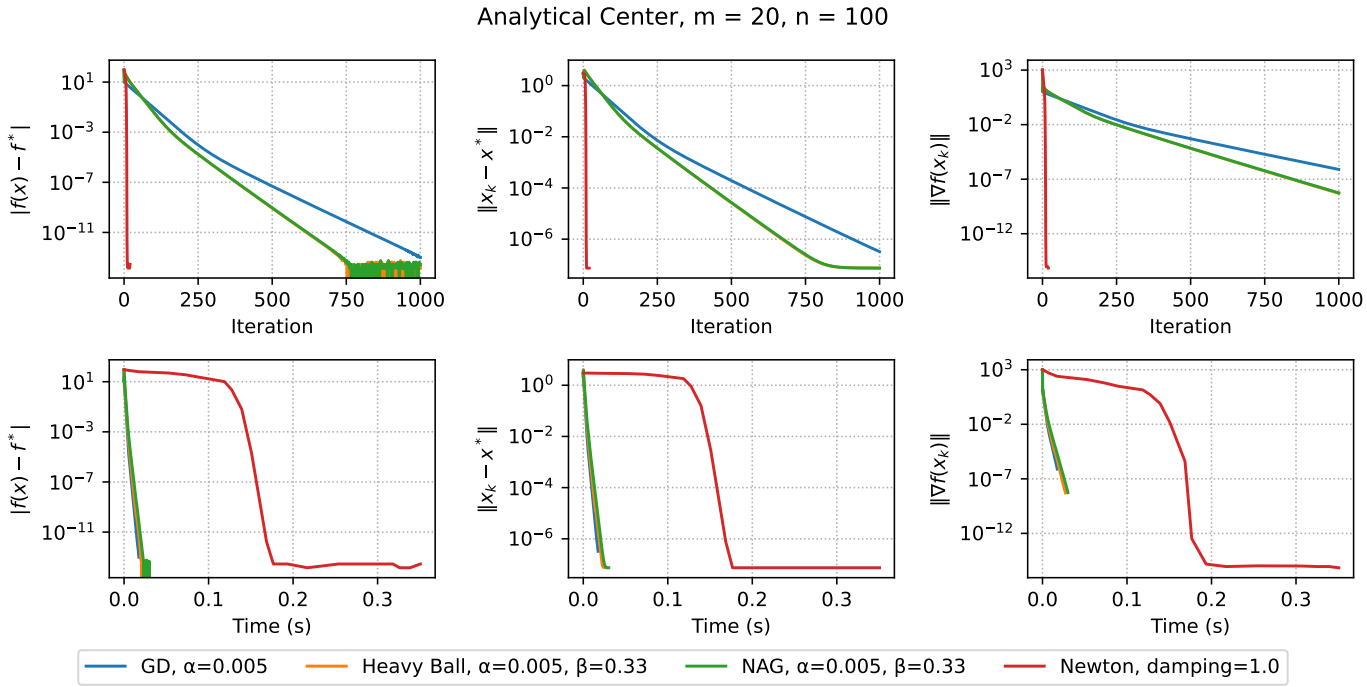
или, эквивалентно, минимизирует:

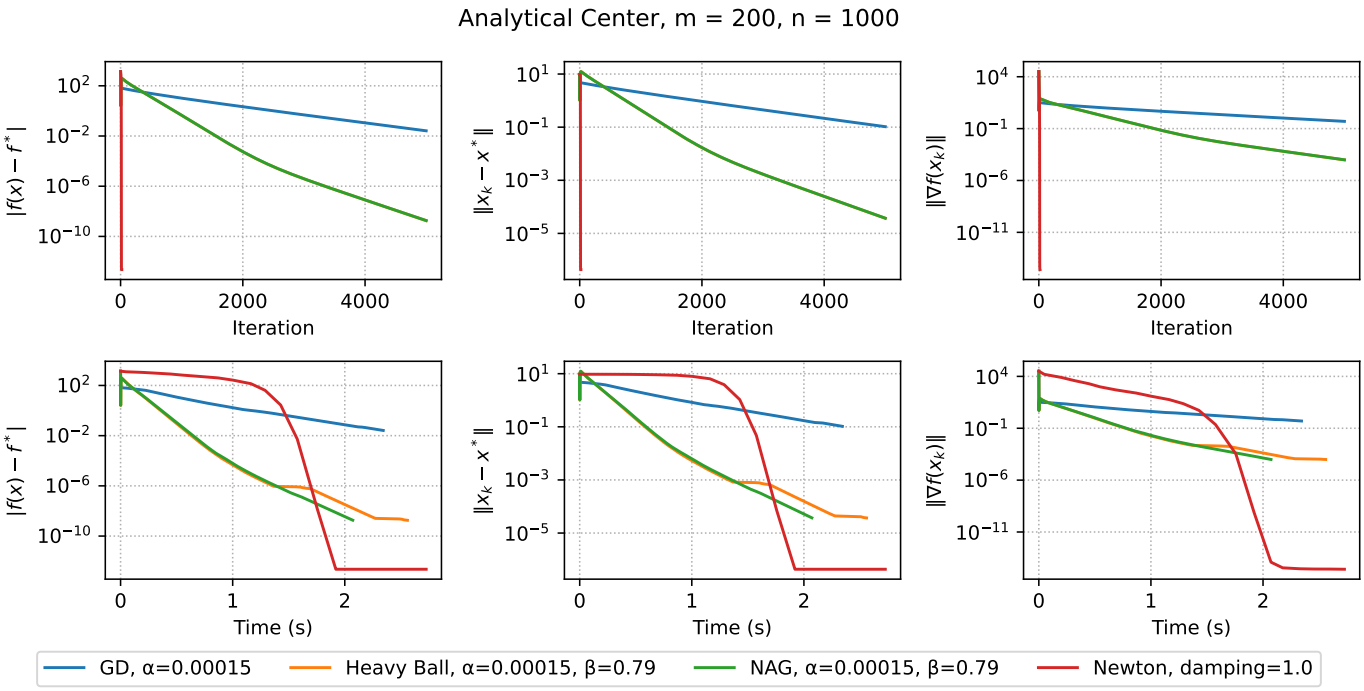
$$\min_x -\sum_{i=1}^{m} \log(1 - a_i^T x) - \sum_{j=1}^{n} \log(1 - x_j^2)$$

при ограничениях: - $a_i^T x < 1$ для всех $i = 1, ..., m$, где a_i - строки матрицы A^T - $|x_j| < 1$ для всех $j = 1, ..., n$

Аналитический центр многогранника - это точка, которая максимально удалена от всех границ многогранника в смысле логарифмического барьера. Эта концепция широко используется в методах внутренней точки для выпуклой оптимизации.


Analytical Center of 2D Polytope:
$\max_{\mathbf{x}} \sum_{i=1}^{5} \log(1 - \mathbf{a}_i^T \mathbf{x}) + \sum_{j=1}^{2} \log(1 - x_j^2)$
$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{a}_3$
$\mathbf{a}_4$
$\mathbf{a}_5$
$\mathbf{x}^*$
x_1
x_2


Analytical Center, m = 20, n = 100
$|f(x) - f^*|$
$\|x_k - x^*\|$
$\|\nabla f(x_k)\|$
Iteration
Time (s)
GD, α=0.005
Heavy Ball, α=0.005, β=0.33
NAG, α=0.005, β=0.33
Newton, damping=1.0



2.11 Аффинная инвариантность метода Ньютона

Важным свойством метода Ньютона является **аффинная инвариантность**. Пусть дана функция f и невырожденная матрица $A \in \mathbb{R}^{n \times n}$, пусть $x = Ay$, и пусть $g(y) = f(Ay)$. Заметим, что $\nabla g(y) = A^T \nabla f(x)$ и $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. Шаги Ньютона на g выражаются как:

$$y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)$$

Раскрывая это, мы получаем:

$$y_{k+1} = y_k - \left(A^T \nabla^2 f(Ay_k) A\right)^{-1} A^T \nabla f(Ay_k)$$

Используя свойство обратной матрицы $(AB)^{-1} = B^{-1} A^{-1}$, это упрощается до:

$$y_{k+1} = y_k - A^{-1} \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$
$$Ay_{k+1} = Ay_k - \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$

Таким образом, правило обновления для x выглядит так:

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Это показывает, что итерация метода Ньютона, не зависит от масштаба задачи. У градиентного спуска такого свойства нет!

2.12 Резюме

Плюсы:

- Квадратичная сходимость вблизи решения x^*
- Аффинная инвариантность
- Отсутствие параметров у метода
- Сходимость можно сделать глобальной, если использовать демпфированный метод Ньютона (добавить процедуру линейного поиска и шага метода)

Минусы:

- Необходимо хранить (обратный) гессиан на каждой итерации: $\mathcal{O}(n^2)$ памяти
- Необходимо решать линейные системы: $\mathcal{O}(n^3)$ операций
- Гессиан может быть вырожденным в x^*
- Гессиан может не быть положительно определенным $\to$ направление $-(f''(x))^{-1}f'(x)$ может не быть направлением спуска

3 Квазиньютоновские методы

3.1 Интуиция квазиньютоновских методов

Для классической задачи безусловной оптимизации $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ общий алгоритм итерационного метода записывается как:

$$x_{k+1} = x_k + \alpha_k d_k$$

В методе Ньютона направление d_k (направление Ньютона) устанавливается решением линейной системы на каждом шаге:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

т.е. на каждой итерации необходимо **вычислить** гессиан и градиент и **решить** линейную систему. Обратите внимание, что если мы возьмем единичную матрицу $B_k = I_n$ в качестве B_k на каждом шаге, мы получим точно метод градиентного спуска.

Общий алгоритм квазиньютоновских методов основан на выборе матрицы B_k так, чтобы она в некотором смысле стремилась к истинному значению гессиана $\nabla^2 f(x_k)$ при $k \to \infty$. Конечно, важно при этом сделать так, чтобы вычисления были не слишком дорогими.

3.2 Схема построения квазиньютоновских методов

Пусть $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. Для $k = 1, 2, 3, ...$, повторяем:

1. Решить $B_k d_k = -\nabla f(x_k)$
2. Обновить $x_{k+1} = x_k + \alpha_k d_k$
3. Вычислить B_{k+1} из B_k

Разные квазиньютоновские методы реализуют шаг 3 по-разному. Мы скоро увидим, что часто можно вычислить $(B_{k+1})^{-1}$ из $(B_k)^{-1}$.

Основная идея: Поскольку B_k уже содержит информацию о гессиане, используем подходящее обновление матрицы для формирования B_{k+1}.

Разумное требование для B_{k+1} (вдохновленное методом секущих):

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1}d_k$$
$$\Delta y_k = B_{k+1}\Delta x_k$$

Помимо уравнения секущей, мы хотим наложить следующие ограничения на B_{k+1}:

- B_{k+1} должна быть симметричной
- B_{k+1} близка к B_k
- $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$

3.3 Симметричное одноранговое обновление

Попробуем обновление вида:

$$B_{k+1} = B_k + auu^T$$

Уравнение секущей $B_{k+1}d_k = \Delta y_k$ дает:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

Это верно только если u кратно $\Delta y_k - B_k d_k$. Положив $u = \Delta y_k - B_k d_k$, мы решаем уравнение,

$$a = \frac{1}{(\Delta y_k - B_k d_k)^T d_k},$$

что приводит к

$$B_{k+1} = B_k + \frac{(\Delta y_k - B_k d_k)(\Delta y_k - B_k d_k)^T}{(\Delta y_k - B_k d_k)^T d_k}$$

Это называется симметричным одноранговым (SR1) обновлением или методом Бройдена.

3.4 Симметричное одноранговое обновление для обратной матрицы

Как мы можем решить

$$B_{k+1}d_{k+1} = -\nabla f(x_{k+1}),$$

чтобы сделать следующий шаг? Помимо вывода B_{k+1} из B_k, давайте получим выражения для вывода обратной матрицы, т.е. $C_{k+1} = (B_{k+1})^{-1}$ из $C_k = B_k^{-1}$.

3.4.1 Формула Шермана-Моррисона:

Формула Шермана-Моррисона утверждает:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

Таким образом, для SR1 обновления, обратная матрица также легко обновляется:

$$C_{k+1} = C_k + \frac{(d_k - C_k\Delta y_k)(d_k - C_k\Delta y_k)^T}{(d_k - C_k\Delta y_k)^T\Delta y_k}$$

SR1 прост и дешев, но у него есть ключевой недостаток: он не сохраняет положительную определенность.

3.5 Обновление Давидона-Флетчера-Пауэлла

Мы могли бы продолжить ту же идею для обновления обратной матрицы C:

$$C_{k+1} = C_k + auu^T + bvv^T.$$

Умножая на Δy_k, используя уравнение секущей $d_k = C_k\Delta y_k$ и решая для a, b, получаем:

$$C_{k+1} = C_k - \frac{C_k\Delta y_k\Delta y_k^TC_k}{\Delta y_k^TC_k\Delta y_k} + \frac{d_kd_k^T}{\Delta y_k^Td_k}$$

3.5.1 Применение формулы Вудбери

Формула Вудбери для обновления обратной матрицы:

$$B_{k+1} = \left(I - \frac{\Delta y_kd_k^T}{\Delta y_k^Td_k}\right)B_k\left(I - \frac{d_k\Delta y_k^T}{\Delta y_k^Td_k}\right) + \frac{\Delta y_k\Delta y_k^T}{\Delta y_k^Td_k}$$

Это обновление Давидона-Флетчера-Пауэлла (DFP). Также дешево: $O(n^2)$, но сохраняет положительную определенность. Не так популярно, как BFGS.

3.6 Обновление Бройдена — Флетчера — Гольдфарба — Шанно

Попробуем теперь двухранговое обновление:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

Уравнение секущей $\Delta y_k = B_{k+1}d_k$ дает:

$$\Delta y_k - B_kd_k = (au^Td_k)u + (bv^Td_k)v$$

Положив $u = \Delta y_k$, $v = B_kd_k$ и решая для a, b, получаем:

$$B_{k+1} = B_k - \frac{B_kd_kd_k^TB_k}{d_k^TB_kd_k} + \frac{\Delta y_k\Delta y_k^T}{d_k^T\Delta y_k}$$

Это обновление Бройдена — Флетчера — Гольдфарба — Шанно (BFGS).

3.7 Обновление Бройдена — Флетчера — Гольдфарба — Шанно с инверсией

3.7.1 Формула Вудбери

Формула Вудбери, обобщение формулы Шермана-Моррисона, дается как:

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$$

Примененная к нашему случаю, мы получаем двухранговое обновление на обратной матрице C:

$$C_{k+1} = C_k + \frac{(d_k - C_k\Delta y_k)d_k^T}{\Delta y_k^T d_k} + \frac{d_k(d_k - C_k\Delta y_k)^T}{\Delta y_k^T d_k} - \frac{(d_k - C_k\Delta y_k)^T\Delta y_k}{(\Delta y_k^T d_k)^2}d_k d_k^T$$

$$C_{k+1} = \left(I - \frac{d_k\Delta y_k^T}{\Delta y_k^T d_k}\right)C_k\left(I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k}\right) + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

Эта формулировка обеспечивает, что обновление BFGS, оставаясь достаточно общим, сохраняет вычислительную эффективность и требует $O(n^2)$ операций. Важно, что обновление BFGS сохраняет положительную определенность: $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$. Эквивалентно, $C_k \succ 0 \Rightarrow C_{k+1} \succ 0$

3.8 Код

- Открыть в Colab
- Сравнение квазиньютоновских методов
- Некоторые практические замечания о методе Ньютона

4 Бонус: доказательства

4.1 Квадратичная сходимость метода Ньютона

> **Theorem**
>
> Пусть $f(x)$ — сильно выпуклая дважды непрерывно дифференцируемая функция на $\mathbb{R}^n$, для второй производной которой выполняются неравенства: $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$. Пусть также гессиан функции M-липшицев. Тогда метод Ньютона сходится локально к решению с квадратичной скоростью, т.е. при $\|x_0 - x^*\| < \frac{2\mu}{3M}$:
>
> $$\|x_{k+1} - x^*\| \leq \frac{3M}{2\mu}\|x_k - x^*\|^2$$

Доказательство

1. Мы будем использовать формулу Ньютона-Лейбница

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*)d\tau$$

2. Мы будем отслеживать расстояние до решения

$$x_{k+1} - x^* = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) - x^* = x_k - x^* - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) =$$
$$= x_k - x^* - \left[\nabla^2 f(x_k)\right]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

4.2 Сходимость

3.

$$= \left(I - \left[\nabla^2 f(x_k)\right]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau\right)(x_k - x^*) =$$
$$= \left[\nabla^2 f(x_k)\right]^{-1} \left(\nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau\right)(x_k - x^*) =$$
$$= \left[\nabla^2 f(x_k)\right]^{-1} \left(\int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau)\right)(x_k - x^*) =$$
$$= \left[\nabla^2 f(x_k)\right]^{-1} G_k (x_k - x^*)$$

4. Введём:

$$G_k = \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau)\,.$$

4.3 Сходимость

5. Попробуем оценить размер G_k с помощью $r_k = \|x_k - x^*\|$:

$$\|G_k\| = \left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau) \right\| \leq$$
$$\leq \int_0^1 \left\| \nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) \right\| d\tau \leq \qquad \text{(Липшицевость гессиана)}$$
$$\leq \int_0^1 M \|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M \|x_k - x^*\| (1 - \tau) d\tau = \frac{r_k}{2} M,$$

6. Получаем:

$$r_{k+1} \leq \left\| \left[\nabla^2 f(x_k)\right]^{-1} \right\| \cdot \frac{r_k}{2} M \cdot r_k$$

и нам нужно оценить норму обратного гессиана

4.4 Сходимость

7. Из липшицевости и симметричности гессиана:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -M r_k I_n$$
$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - M r_k I_n$$
$$\nabla^2 f(x_k) \succeq \mu I_n - M r_k I_n$$
$$\nabla^2 f(x_k) \succeq (\mu - M r_k) I_n$$

8. Из сильной выпуклости следует, что $\nabla^2 f(x_k) \succ 0$, т.е. $r_k < \frac{\mu}{M}$.

$$\left\|[\nabla^2 f(x_k)]^{-1}\right\| \leq (\mu - Mr_k)^{-1}$$
$$r_{k+1} \leq \frac{r_k^2 M}{2(\mu - Mr_k)}$$

9. Потребуем, чтобы верхняя оценка на r_{k+1} была меньше r_k, учитывая, что $0 < r_k < \frac{\mu}{M}$:

$$\frac{r_k^2 M}{2(\mu - Mr_k)} < r_k$$
$$\frac{M}{2(\mu - Mr_k)} r_k < 1$$
$$Mr_k < 2(\mu - Mr_k)$$
$$3Mr_k < 2\mu$$
$$r_k < \frac{2\mu}{3M}$$

10. Возвращаясь к оценке невязки на $k+1$-ой итерации, получаем:

$$r_{k+1} \leq \frac{r_k^2 M}{2(\mu - Mr_k)} < \frac{3Mr_k^2}{2\mu}$$

Таким образом, мы получили важный результат: метод Ньютона для функции с липшицевым положительно определённым гессианом сходится **квадратично** вблизи решения.

4.5 Бонус: Идея методов адаптивной метрики

Пусть дана функция $f(x)$ и точка x_0. Определим $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$ как множество точек с расстоянием ε до x_0. Здесь мы предполагаем существование функции расстояния $d(x, x_0)$.

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Далее, мы можем определить другое направление *наискорейшего спуска* в терминах минимизатора функции на сфере:

$$s = \lim_{\varepsilon \to 0} \frac{x^* - x_0}{\varepsilon}$$

Предположим, что расстояние локально определяется некоторой метрикой A:

$$d(x, x_0) = (x - x_0)^\top A (x - x_0)$$

Далее рассмотрим первый порядок аппроксимации функции $f(x)$ в окрестности точки x_0:

$$f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \tag{1}$$

Теперь мы можем сформулировать задачу нахождения s, как это было сказано выше.

$$\min_{\delta x \in \mathbb{R}^n} f(x_0 + \delta x)$$
$$\text{s.t. } \delta x^\top A \delta x = \varepsilon^2$$

Используя уравнение 1, получаем:

$$\min_{\delta x \in \mathbb{R}^n} \nabla f(x_0)^\top \delta x$$
$$\text{s.t. } \delta x^\top A \delta x = \varepsilon^2$$

Используя метод множителей Лагранжа:

$$\delta x = -\frac{2\varepsilon^2}{\nabla f(x_0)^\top A^{-1} \nabla f(x_0)} A^{-1} \nabla f$$

Новое направление наискорейшего спуска : $-A^{-1}\nabla f(x_0)$. Действительно, если пространство изотропно и $A = I$, мы сразу получаем формулу градиентного спуска, в то время как метод Ньютона использует локальный гессиан как матрицу метрик.

<!– # Задачи на дом

1. ✙ **Сходимость метода Ньютона** (7 баллов)

   Рассмотрим следующую функцию:

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

   и возьмем начальную точку $x_0 = (0, 2)^\top$. Как ведет себя метод Ньютона при старте из этой точки? Как это объяснить? Как ведет себя градиентный спуск с фиксированным шагом $\alpha = 0.01$ и метод наискорейшего спуска при тех же условиях? (Не обязательно показывать численные симуляции в этой задаче).

2. **Метод Ньютона без вычисления гессиана** (16 баллов) В этой задаче мы рассмотрим оптимизацию задачи бинарной логистической регрессии с использованием различных методов. Не беспокойтесь о размере описания задачи, первые 5 пунктов из 7 можно выполнить довольно быстро. В этой задаче вы должны начать с этого colab notebook

   Дана выборка с n наблюдениями, где каждое наблюдение состоит из вектора признаков x_i и связанной с ним бинарной целевой переменной $y_i \in \{0, 1\}$. Логистическая регрессия моделирует вероятность того, что $y_i = 1$ при данном x_i с использованием логистической функции. Функция потерь, которая минимизируется, это отрицательный логарифм правдоподобия наблюдаемых результатов под этой моделью, суммированный по всем наблюдениям. Она имеет высокое значение, когда выходные данные модели значительно отличаются от данных y.

   Функция потерь бинарной кросс-энтропии для одного наблюдения (x_i, y_i) задается как:

$$\text{Loss}(w; x_i, y_i) = -\left[y_i \log(p(y_i = 1|x_i; w)) + (1 - y_i) \log(1 - p(y_i = 1|x_i; w))\right]$$

   Здесь $p(y = 1|x; w)$ определяется как:

$$p(y = 1|x; w) = \frac{1}{1 + e^{-w^T x}}$$

   Чтобы определить общую потерю над набором данных, мы суммируем индивидуальные потери:

$$f(w) = -\sum_{i=1}^{n} \left[y_i \log(p(y_i = 1|x_i; w)) + (1 - y_i) \log(1 - p(y_i = 1|x_i; w))\right]$$

   Таким образом, оптимизационная задача в логистической регрессии задается как:

$$\min_w f(w) = \min_w -\sum_{i=1}^{n} \left[y_i \log\left(p\left(y_i = 1|x_i; w\right)\right) + (1 - y_i) \log\left(1 - p(y_i = 1|x_i; w)\right)\right]$$

   Это задача выпуклой оптимизации и может быть решена с использованием градиентных методов, таких как градиентный спуск, метод Ньютона, или более сложных оптимизационных алгоритмов, часто доступных в библиотеках машинного обучения. Однако, эта задача часто сопровождается регуляризацией l_2:

$$\min_w f(w) = \min_w -\sum_{i=1}^{n} \left[y_i \log\left(p\left(y_i = 1|x_i; w\right)\right) + (1 - y_i) \log\left(1 - p(y_i = 1|x_i; w)\right)\right] + \frac{\mu}{2}\|w\|_2^2$$

1. (1,6 баллов) Во-первых, рассмотрим оптимизацию с помощью метода градиентного спуска (Gradient Descent, GD) в сильно выпуклой постановке при $\mu = 1$. Используйте постоянный шаг обучения α. Запустите алгоритм градиентного спуска. Укажите наибольший шаг обучения, при котором алгоритм гарантированно сходится. Постройте график сходимости в терминах как области (значения параметров), так и значения функции (потери). Опишите тип наблюдаемой сходимости.

```
params = {
    "mu": 1,
    "m": 1000,
    "n": 100,
    "methods": [
        {
            "method": "GD",
            "learning_rate": 3e-2,
            "iterations": 550,
        },
    ]
}

results, params = run_experiments(params)
```

2. (1,6 баллов) Запустите метод Ньютона при тех же условиях, используя вторые производные для направления оптимизации. Опишите и проанализируйте наблюдаемые свойства сходимости.

```
params = {
    "mu": 1,
    "m": 1000,
    "n": 100,
    "methods": [
        {
            "method": "GD",
            "learning_rate": 3e-2,
            "iterations": 550,
        },
        {
            "method": "Newton",
            "iterations": 20,
        },
    ]
}

results, params = run_experiments(params)
```

3. (1,6 баллов) В случаях, когда метод Ньютона может сходиться слишком быстро или «перескакивать» через минимум, заторможенная версия метода может быть более устойчивой. Запустите заторможенный метод Ньютона. Отрегулируйте фактор затухания как шаг обучения. Укажите наибольший шаг обучения, обеспечивающий стабильность и сходимость. Постройте график сходимости.

```
params = {
    "mu": 1,
```

```
    "m": 1000,
    "n": 100,
    "methods": [
        {
            "method": "GD",
            "learning_rate": 3e-2,
            "iterations": 550,
        },
        {
            "method": "Newton",
            "iterations": 20,
        },
        {
            "method": "Newton",
            "learning_rate": 5e-1,
            "iterations": 50,
        },
    ]
}

results, params = run_experiments(params)
```

4. (1,6 баллов) Теперь отключим регуляризацию, установив $\mu = 0$. Попробуйте найти наибольший шаг обучения, который обеспечивает сходимость градиентного спуска. Используйте постоянный шаг обучения α. Запустите алгоритм градиентного спуска. Укажите наибольший шаг обучения, при котором алгоритм гарантированно сходится. Постройте график сходимости в терминах как области (значения параметров), так и значения функции (потери). Опишите тип наблюдаемой сходимости. Как вы можете описать идею применения этого метода для достижения строгого первичного разрыва $f(x_k) - f^* \approx 10^{-2}$ или $10^{-3}, 10^{-4}$?

```
params = {
    "mu": 0,
    "m": 1000,
    "n": 100,
    "methods": [
        {
            "method": "GD",
            "learning_rate": 3e-2,
            "iterations": 200,
        },
        {
            "method": "GD",
            "learning_rate": 7e-2,
            "iterations": 200,
        },
    ]
}

results, params = run_experiments(params)
```

5. (1,6 баллов) Что вы можете сказать о сходимости метода Ньютона в той же постановке $\mu = 0$?

Попробуйте использовать несколько значений скорости обучения, меньших единицы, для заторможенного метода Ньютона. Работает ли это? Напишите ваши выводы о сходимости второго порядка для задачи бинарной логистической регрессии.

6. (4 балла) Теперь вернемся к сильно выпуклой постановке $\mu = 1$. Чтобы избежать прямого вычисления матрицы гессиана в методе Ньютона, используйте метод сопряженных градиентов (Conjugate Gradient, CG) для решения линейной системы в шаге Ньютона. Разработайте функцию `newton_method_cg`, которая вычисляет шаг Ньютона, используя CG для решения системы $\nabla^2 f(x_k) d_k = -\nabla f(x_k),\ x_{k+1} = x_k + \alpha d_k$ определенной гессианом. Вы должны использовать функцию `jax.scipy.sparse.linalg.cg`. Итак, сначала вычислите гессиан, как это было сделано в коде, затем поместите его в этот решатель линейных систем. Сравните его производительность в терминах вычислительной эффективности и скорости сходимости с обычным методом Ньютона.

7. (4 балла) Теперь реализуйте версию метода Ньютона без вычисления гессиана (HFN), которая использует произведение гессиана на вектор, получаемые с помощью автоматического дифференцирования. Обратите внимание, что функция `jax.scipy.sparse.linalg.cg` может принимать функцию matvec, которая напрямую вычисляет произведение любого входного вектора x на гессиан. Реализуйте метод HFN без явного формирования или хранения матрицы гессиана в функции `newton_method_hfn`. Используйте autograd для вычисления произведений гессиана на вектор, как это описано здесь. Сравните вычислительную сложность по времени и затраты памяти этого метода с предыдущими реализациями.