Нижние оценки для градиентного спуска. Ускоренный градиентный спуск. Момент. Ускорение Нестерова

Даня Меркулов

1 Сходимость градиентного спуска

Градиентный спуск: $\min\limits_{x \in \mathbb{R}^n} f(x)$ $\qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

выпуклая (негладкая)	гладкая (невыпуклая)	гладкая & выпуклая	гладкая & сильно выпуклая
$f(x_k) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\min\limits_{0 \le i \le k} \|\nabla f(x_i)\| = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$f(x_k) - f^* = \mathcal{O}\left(\frac{1}{k}\right)$	$\|x_k - x^*\|^2 = \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon = \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon = \mathcal{O}\left(\varkappa \log \frac{1}{\varepsilon}\right)$

Для гладкой сильно выпуклой функции мы имеем:

$$f(x_k) - f^* \le \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f^*).$$

Обратите внимание, что для любого x, поскольку e^{-x} выпуклая и $1 - x$ является её касательной в точке $x = 0$, мы имеем:

$$1 - x \le e^{-x}$$

Наконец:

$$\begin{aligned} \varepsilon = f(x_{k_\varepsilon}) - f^* &\le \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x_0) - f^*) \\ &\le \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x_0) - f^*) \\ k_\varepsilon &\ge \varkappa \log \frac{f(x_0) - f^*}{\varepsilon} = \mathcal{O}\left(\varkappa \log \frac{1}{\varepsilon}\right) \end{aligned}$$

Вопрос: Можно ли добиться лучшей скорости сходимости, используя только информацию первого порядка? ***Да, можно.***

2 Нижние оценки

Для нижних оценок пишут $\Omega(\cdot)$ *вместо* $\mathcal{O}(\cdot)$.

выпуклая (негладкая)	гладкая (невыпуклая)[1]	гладкая & выпуклая[2]	гладкая & сильно выпуклая
$f(x_k) - f^* = \Omega\left(\frac{1}{\sqrt{k}}\right)$	$\min\limits_{0\le i\le k} \|\nabla f(x_i)\| = \Omega\left(\frac{1}{\sqrt{k}}\right)$	$f(x_k) - f^* = \Omega\left(\frac{1}{k^2}\right)$	$f(x_k) - f^* = \Omega\left(\left(\frac{\sqrt{\varkappa}-1}{\sqrt{\varkappa}+1}\right)^{2k}\right)$
$k_\varepsilon = \Omega\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon = \Omega\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon = \Omega\left(\frac{1}{\sqrt{\varepsilon}}\right)$	$k_\varepsilon = \Omega\left(\sqrt{\varkappa}\log\frac{1}{\varepsilon}\right)$

2.1 Чёрный ящик

Итерация градиентного спуска:

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k \nabla f(x_k) \\ &= x_{k-1} - \alpha_{k-1}\nabla f(x_{k-1}) - \alpha_k \nabla f(x_k) \\ &\vdots \\ &= x_0 - \sum_{i=0}^{k} \alpha_{k-i} \nabla f(x_{k-i}) \end{aligned}$$

Рассмотрим семейство методов первого порядка, где

$$\begin{aligned} x_{k+1} &\in x_0 + \operatorname{Lin}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x_k)\} \qquad f - \text{гладкая} \\ x_{k+1} &\in x_0 + \operatorname{Lin}\{g_0, g_1, \dots, g_k\}, \text{ где } g_i \in \partial f(x_i) \qquad f - \text{негладкая} \end{aligned} \tag{1}$$

Чтобы построить нижнюю оценку, нам нужно найти функцию f из соответствующего класса, такую, что *любой* метод из семейства (1) будет работать не быстрее этой нижней оценки.

2.2 Гладкий случай

Theorem

Существует L-гладкая и выпуклая функция f, такая, что *любой* метод (1) *для всех* k, $1 \le k \le \frac{n-1}{2}$, удовлетворяет:

$$f(x_k) - f^* \ge \frac{3L\|x_0 - x^*\|_2^2}{32(k+1)^2}$$

- Какой бы метод из семейства методов первого порядка вы ни использовали, найдётся функция f, на которой скорость сходимости не лучше $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- Ключом к доказательству является явное построение специальной функции f.

[1] Carmon, Duchi, Hinder, Sidford, 2017
[2] Nemirovski, Yudin, 1979

- Обратите внимание, что эта граница $\mathcal{O}\left(\frac{1}{k^2}\right)$ не соответствует скорости градиентного спуска $\mathcal{O}\left(\frac{1}{k}\right)$. Два возможных варианта:
  a. Нижняя оценка не является точной.
  b. **Метод градиентного спуска не является оптимальным для этой задачи.**

2.3 Наихудшая функция Нестерова

- Пусть $n = 2k + 1$ и $A \in \mathbb{R}^{n \times n}$.

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 \end{bmatrix}$$

- Обратите внимание, что

$$x^T A x = x_1^2 + x_n^2 + \sum_{i=1}^{n-1} (x_i - x_{i+1})^2,$$

  Следовательно, $x^T A x \geq 0$. Также легко увидеть, что $0 \preceq A \preceq 4I$.

Пример, когда $n = 3$:

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Нижняя оценка:

$$\begin{aligned} x^T A x &= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1 x_2 - 2x_2 x_3 \\ &= x_1^2 + x_1^2 - 2x_1 x_2 + x_2^2 + x_2^2 - 2x_2 x_3 + x_3^2 + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 \geq 0 \end{aligned}$$

Верхняя оценка

$$\begin{aligned} x^T A x &= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1 x_2 - 2x_2 x_3 \\ &\leq 4(x_1^2 + x_2^2 + x_3^2) \\ 0 &\leq 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1 x_2 + 2x_2 x_3 \\ 0 &\leq x_1^2 + x_1^2 + 2x_1 x_2 + x_2^2 + x_2^2 + 2x_2 x_3 + x_3^2 + x_3^2 \\ 0 &\leq x_1^2 + (x_1 + x_2)^2 + (x_2 + x_3)^2 + x_3^2 \end{aligned}$$

- Определим следующую L-гладкую выпуклую функцию: $f(x) = \frac{L}{4}\left(\frac{1}{2} x^T A x - e_1^T x\right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$.
- Оптимальное решение x^* удовлетворяет $Ax^* = e_1$, и решение этой системы уравнений дает:

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \\ \vdots \\ x_n^* \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{cases} 2x_1^* - x_2^* = 1 \\ -x_{i-1}^* + 2x_i^* - x_{i+1}^* = 0, \ i = 2, \ldots, n-1 \\ -x_{n-1}^* + 2x_n^* = 0 \end{cases}$$

- Гипотеза: $x_i^* = a + bi$ (вдохновлённая физикой). Проверьте, что выполнено второе уравнение, в то время как a и b вычисляются из первого и последнего уравнений.
- Решение:
$$x_i^* = 1 - \frac{i}{n+1},$$
- И значение функции равно
$$f(x^*) = \frac{L}{8}x^{*T}Ax^* - \frac{L}{4}\langle x^*, e_1\rangle = -\frac{L}{8}\langle x^*, e_1\rangle = -\frac{L}{8}\left(1 - \frac{1}{n+1}\right).$$

2.4 Гладкий случай (доказательство)

- Предположим, что мы начинаем с $x_0 = 0$. Запросив у оракула градиент, мы получаем $g_0 = -\frac{L}{4}e_1$. Тогда, x_1 должен лежать на линии, генерируемой e_1. В этой точке все компоненты x_1 равны нулю, кроме первой, поэтому
$$x_1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$
- На второй итерации оракул возвращает градиент $g_1 = \frac{L}{4}(Ax_1 - e_1)$. Тогда, x_2 должен лежать на линии, генерируемой e_1 и $Ax_1 - e_1$. Все компоненты x_2 равны нулю, кроме первых двух, поэтому
$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow x_2 = \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$
- Из-за структуры матрицы A можно показать, что после k итераций все последние $n-k$ компоненты x_k равны нулю.
$$x_k = \begin{bmatrix} \bullet \\ \bullet \\ \vdots \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ k \\ k+1 \\ \vdots \\ n \end{matrix}$$
- Однако, поскольку каждая итерация x_k, произведенная нашим методом, лежит в $S_k = \text{Lin}\{e_1, e_2, \ldots, e_k\}$ (т.е. имеет нули в координатах $k+1, \ldots, n$), она не может "достичь" полного оптимального вектора x^*. Другими словами, даже если бы мы выбрали лучший возможный вектор из S_k, обозначаемый
$$\tilde{x}_k = \arg\min_{x \in S_k} f(x),$$
значение функции в нём $f(\tilde{x}_k)$ будет выше, чем $f(x^*)$.
- Поскольку $x_k \in S_k = \text{Lin}\{e_1, e_2, \ldots, e_k\}$ и $\tilde{x}_k$ является лучшим возможным приближением к x^* в S_k, мы имеем
$$f(x_k) \geq f(\tilde{x}_k).$$

- Следовательно,
$$f(x_k) - f(x^*) \geq f(\tilde{x}_k) - f(x^*).$$
- Аналогично, для оптимума исходной функции, мы имеем $\tilde{x}_{k_{(i)}} = 1 - \frac{i}{k+1}$ и $f(\tilde{x}_k) = -\frac{L}{8}\left(1 - \frac{1}{k+1}\right)$.
- Теперь мы имеем:

$$\begin{aligned} f(x_k) - f(x^*) &\geq f(\tilde{x}_k) - f(x^*) \\ &= -\frac{L}{8}\left(1 - \frac{1}{k+1}\right) - \left(-\frac{L}{8}\left(1 - \frac{1}{n+1}\right)\right) \\ &= \frac{L}{8}\left(\frac{1}{k+1} - \frac{1}{n+1}\right) = \frac{L}{8}\left(\frac{n-k}{(k+1)(n+1)}\right) \\ &\overset{n=2k+1}{=} \frac{L}{16(k+1)} \end{aligned} \tag{2}$$

- Теперь мы ограничиваем $R = \|x_0 - x^*\|_2$:

$$\begin{aligned} \|x_0 - x^*\|_2^2 = \|0 - x^*\|_2^2 = \|x^*\|_2^2 &= \sum_{i=1}^{n}\left(1 - \frac{i}{n+1}\right)^2 \\ &= n - \frac{2}{n+1}\sum_{i=1}^{n} i + \frac{1}{(n+1)^2}\sum_{i=1}^{n} i^2 \\ &\leq n - \frac{2}{n+1}\cdot\frac{n(n+1)}{2} + \frac{1}{(n+1)^2}\cdot\frac{(n+1)^3}{3} \\ &= \frac{n+1}{3} \overset{n=2k+1}{=} \frac{2(k+1)}{3}. \end{aligned}$$

- Следовательно,
$$k + 1 \geq \frac{3}{2}\|x_0 - x^*\|_2^2 = \frac{3}{2}R^2 \tag{3}$$

Заметим, что

$$\begin{aligned} \sum_{i=1}^{n} i &= \frac{n(n+1)}{2} \\ \sum_{i=1}^{n} i^2 &= \frac{n(n+1)(2n+1)}{6} \\ &\leq \frac{(n+1)^3}{3} \end{aligned}$$

Наконец, используя (2) и (3), мы получаем:

$$\begin{aligned} f(x_k) - f(x^*) &\geq \frac{L}{16(k+1)} = \frac{L(k+1)}{16(k+1)^2} \\ &\geq \frac{L}{16(k+1)^2}\frac{3}{2}R^2 \\ &= \frac{3LR^2}{32(k+1)^2} \end{aligned}$$

Это завершает доказательство с желаемой скоростью $\mathcal{O}\left(\frac{1}{k^2}\right)$.

2.5 Нижние оценки для гладкого случая

Гладкий выпуклый случай

Существует L-гладкая выпуклая функция f, такая, что *любой* метод в форме 1 *для* всех k, $1 \leq k \leq \frac{n-1}{2}$, удовлетворяет:

$$f(x_k) - f^* \geq \frac{3L\|x_0 - x^*\|_2^2}{32(k+1)^2}$$

Гладкий сильно выпуклый случай

Для любого x_0 и *любого* $\mu > 0$, $\varkappa = \frac{L}{\mu} > 1$, существует L-гладкая и μ-сильно выпуклая функция f, такая, что *для любого* метода в форме 1 выполняются неравенства:

$$\|x_k - x^*\|_2 \geq \left(\frac{\sqrt{\varkappa}-1}{\sqrt{\varkappa}+1}\right)^k \|x_0 - x^*\|_2$$

$$f(x_k) - f^* \geq \frac{\mu}{2}\left(\frac{\sqrt{\varkappa}-1}{\sqrt{\varkappa}+1}\right)^{2k} \|x_0 - x^*\|_2^2$$

3 Ускорение для квадратичных функций

3.1 Результат сходимости для квадратичных функций

Предположим, что мы решаем задачу минимизации сильно выпуклой квадратичной функции, с помощью метода градиентного спуска:

$$f(x) = \frac{1}{2}x^T Ax - b^T x \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

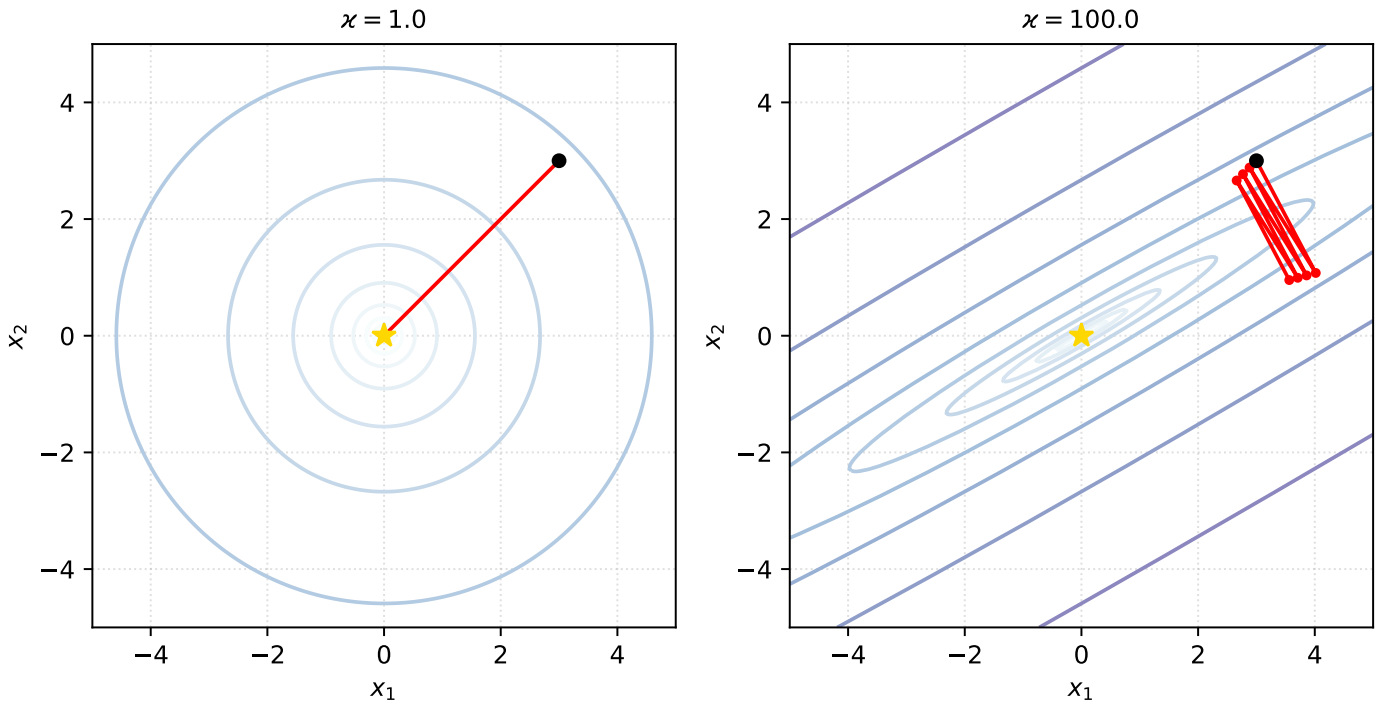
Theorem

Градиентный спуск с шагом $\alpha_k = \frac{2}{\mu+L}$ сходится к оптимальному решению x^* со следующей гарантией:

$$\|x_{k+1} - x^*\|_2 \leq \left(\frac{\varkappa-1}{\varkappa+1}\right)^k \|x_0 - x^*\|_2 \qquad f(x_{k+1}) - f(x^*) \leq \left(\frac{\varkappa-1}{\varkappa+1}\right)^{2k} (f(x_0) - f(x^*))$$

где $\varkappa = \frac{L}{\mu}$ является числом обусловленности A.

3.2 Число обусловленности $\varkappa$



3.3 Ускорение из первых принципов

$$f(x) = \frac{1}{2}x^T Ax - b^T x \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

Пусть x^* будет единственным решением системы линейных уравнений $Ax = b$ и пусть $e_k = x_k - x^*$, где $x_{k+1} = x_k - \alpha_k(Ax_k - b)$ определяется рекурсивно, начиная с некоторого x_0, а α_k — шаг, который мы определим позже.

$$e_{k+1} = (I - \alpha_k A)e_k.$$

3.3.1 Полиномы

Вышеуказанный расчет дает нам $e_k = p_k(A)e_0$,

где p_k является полиномом

$$p_k(a) = \prod_{i=1}^{k}(1 - \alpha_i a).$$

Мы можем ограничить норму ошибки как

$$\|e_k\| \leq \|p_k(A)\| \cdot \|e_0\|.$$

Поскольку A является симметричной матрицей с собственными значениями в $[\mu, L]$,:

$$\|p_k(A)\| \leq \max_{\mu \leq a \leq L} |p_k(a)| .$$

Это приводит к интересной постановке задачи: среди всех полиномов, удовлетворяющих $p_k(0) = 1$, мы ищем полином, значение которого как можно меньше отклоняется от нуля на интервале $[\mu, L]$.

3.4 Наивное полиномиальное решение

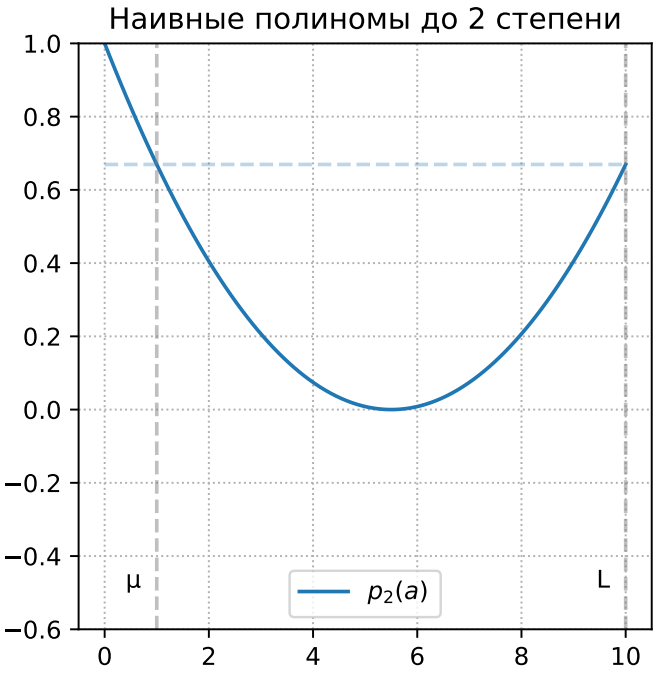
Наивное решение состоит в том, чтобы выбрать равномерный шаг $\alpha_k = \frac{2}{\mu+L}$. Благодаря этому $|p_k(\mu)| = |p_k(L)|$.

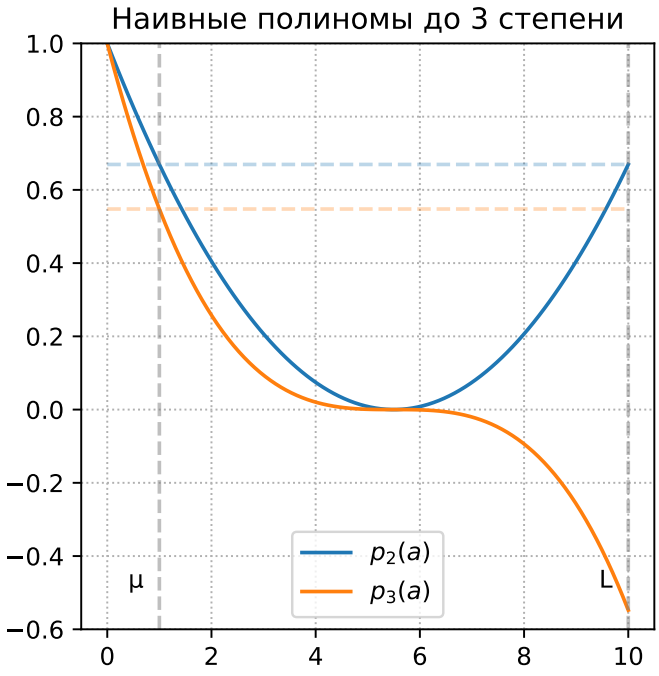
$$\|e_k\| \leq \left(\frac{\varkappa - 1}{\varkappa + 1}\right)^k \|e_0\|$$

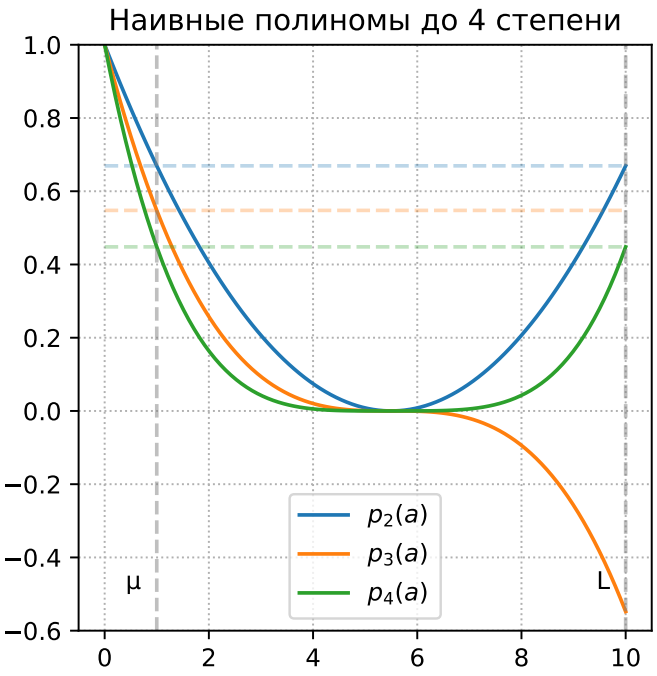
Это точно та же скорость, которую мы доказали в предыдущей лекции для любой гладкой и сильно выпуклой функции.

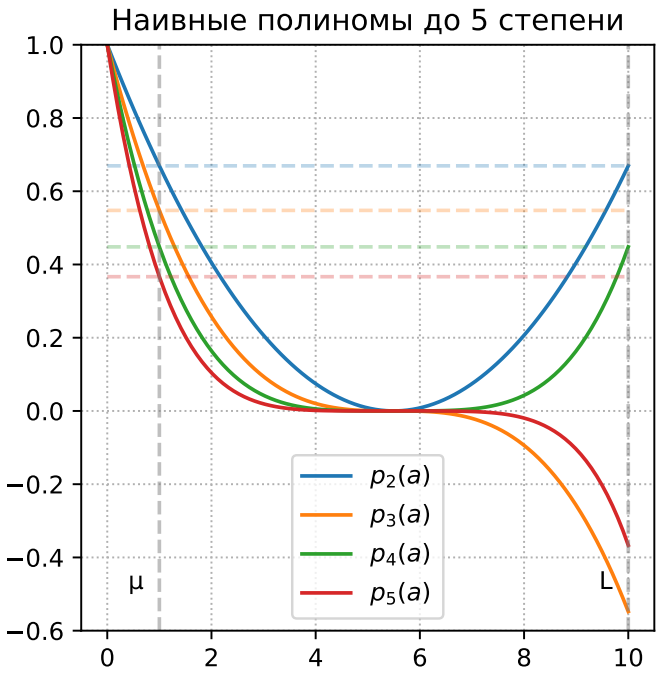
Давайте посмотрим на этот полином поближе. На правом рисунке мы выбираем $\mu = 1$ и $L = 10$ так, что $\varkappa = 10$. Следовательно, соответствующий интервал равен $[1, 10]$.

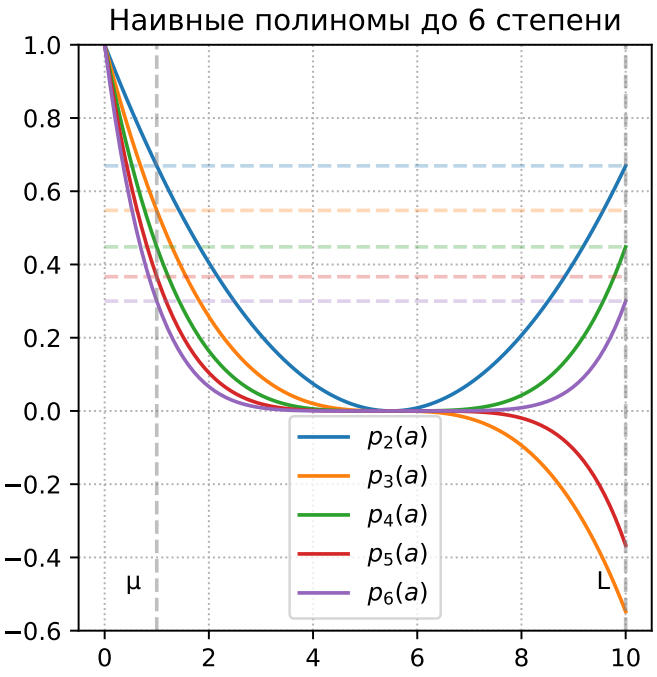
Можем ли мы сделать лучше? Ответ — да.











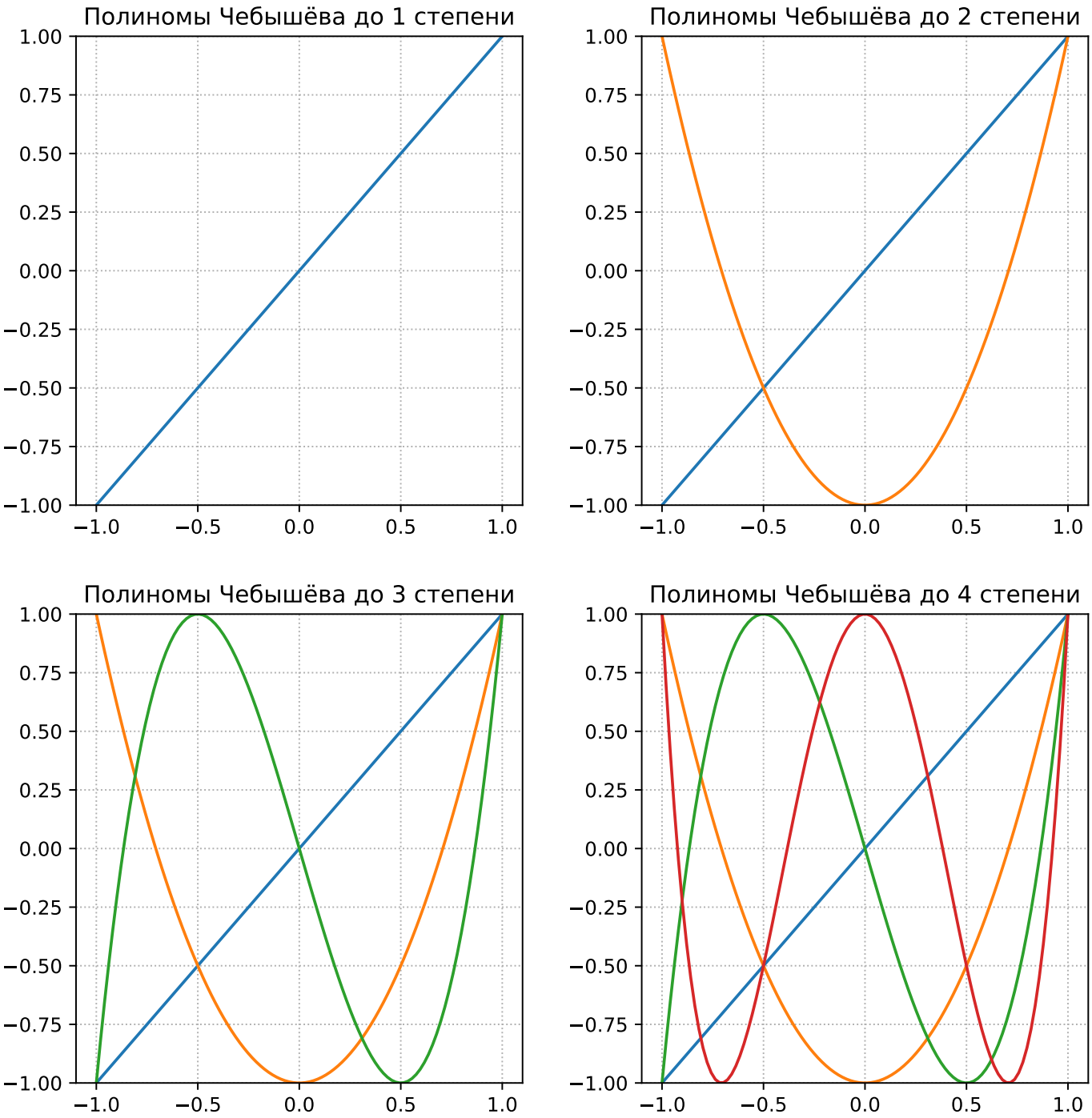
3.5 Полиномы Чебышева

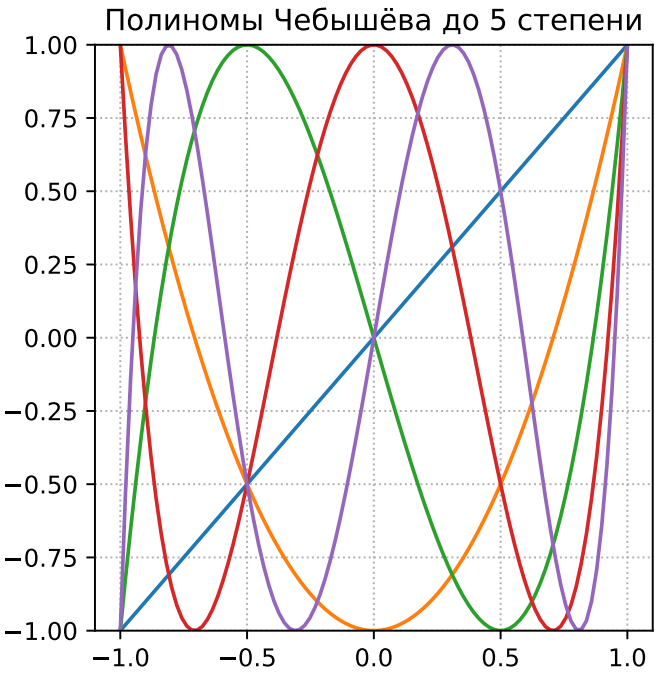
Полиномы Чебышёва дают оптимальный ответ на поставленный вопрос. При соответствующем шкалировании они минимизируют абсолютное значение на заданном интервале $[\mu, L]$, одновременно

удовлетворяя нормировочному условию $p(0) = 1$.

$$
\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x), \qquad k \geq 2.
\end{aligned}
$$

Давайте построим стандартные полиномы Чебышёва (без масштабирования):





3.6 Отшкалированные полиномы Чебышёва

Оригинальные полиномы Чебышёва определены на интервале $[-1, 1]$. Чтобы использовать их для наших целей, мы должны отшкалировать их на интервал $[\mu, L]$.

Мы будем использовать следующее аффинное преобразование:

$$x = \frac{L + \mu - 2a}{L - \mu}, \quad a \in [\mu, L], \quad x \in [-1, 1].$$

Обратите внимание, что $x = 1$ соответствует $a = \mu$, $x = -1$ соответствует $a = L$ и $x = 0$ соответствует $a = \frac{\mu+L}{2}$. Это преобразование гарантирует, что поведение полинома Чебышёва на интервале $[-1, 1]$ транслируется на интервал $[\mu, L]$.

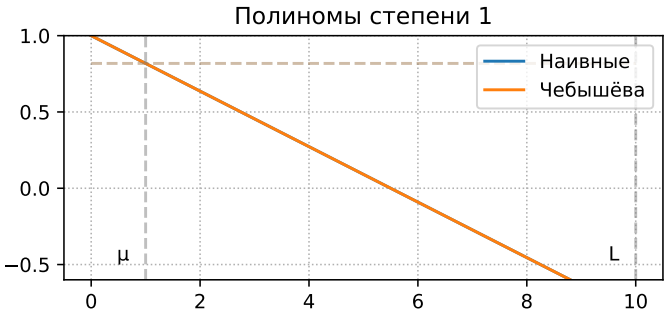
В нашем анализе ошибок мы требуем, чтобы полином был равен 1 в 0 (т.е. $p_k(0) = 1$). После применения преобразования значение T_k в точке, соответствующей $a = 0$, может не быть 1. Следовательно, мы умножаем на обратную величину T_k в точке

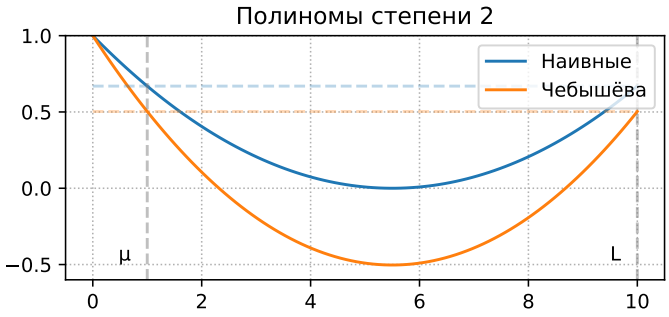
$$\frac{L + \mu}{L - \mu}, \qquad \text{что обеспечивает} \qquad P_k(0) = T_k\left(\frac{L + \mu - 0}{L - \mu}\right) \cdot T_k\left(\frac{L + \mu}{L - \mu}\right)^{-1} = 1.$$

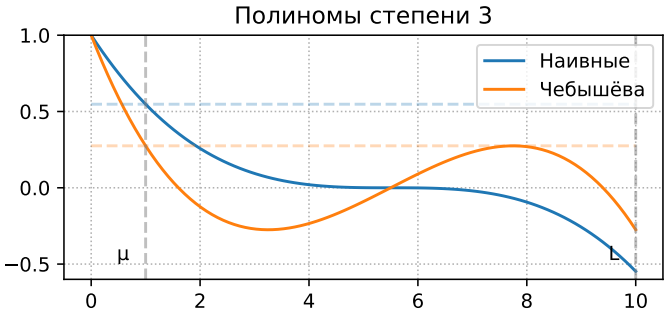
Построим отшкалированные полиномы Чебышёва

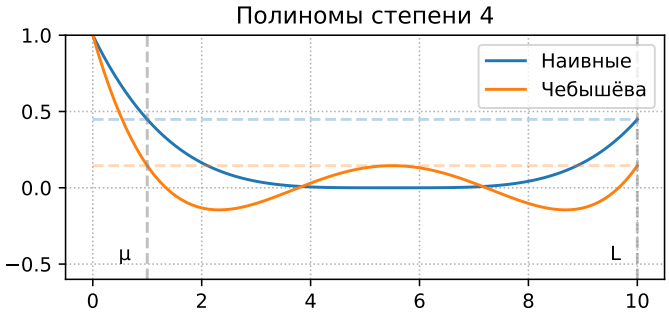
$$P_k(a) = T_k\left(\frac{L + \mu - 2a}{L - \mu}\right) \cdot T_k\left(\frac{L + \mu}{L - \mu}\right)^{-1}$$

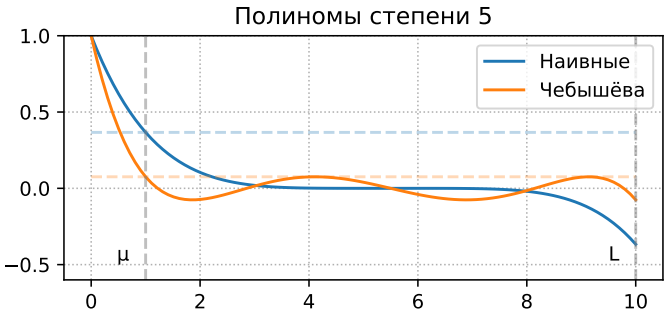
и увидим, что они больше подходят для нашей задачи, чем наивные полиномы на интервале $[\mu, L]$.

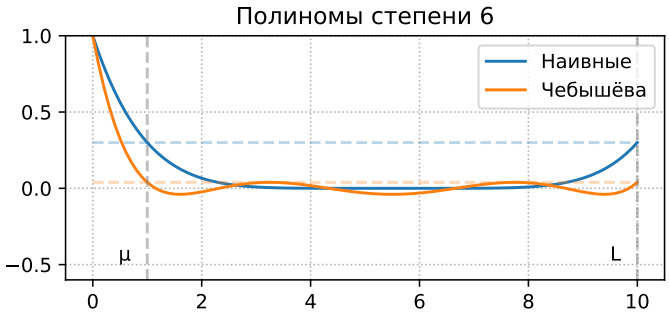

Полиномы степени 1
Наивные
Чебышёва
μ
L

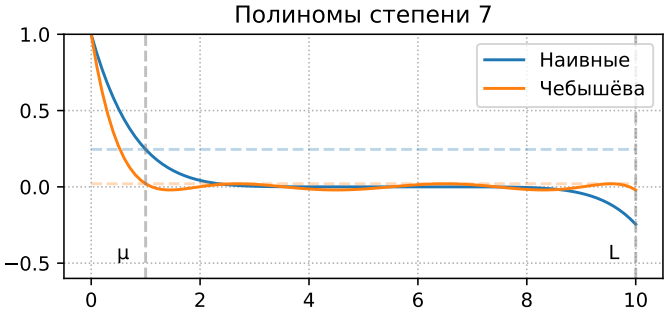

Полиномы степени 2
Наивные
Чебышёва
μ
L

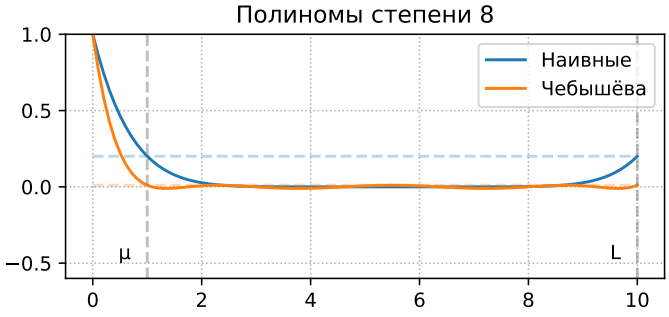

Полиномы степени 3
Наивные
Чебышёва
μ
L

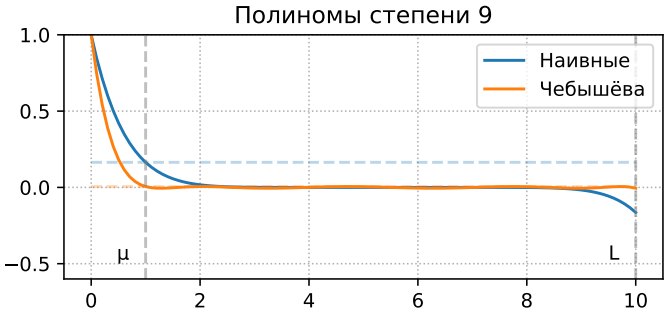

Полиномы степени 4
Наивные
Чебышёва
μ
L

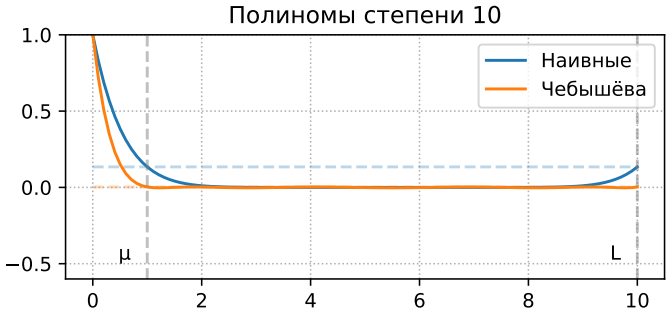

Полиномы степени 5
Наивные
Чебышёва
μ
L


Полиномы степени 6
Наивные
Чебышёва
μ
L


Полиномы степени 7
Наивные
Чебышёва
μ
L


Полиномы степени 8
Наивные
Чебышёва
μ
L


Полиномы степени 9
Наивные
Чебышёва
μ
L


Полиномы степени 10
Наивные
Чебышёва
μ
L

3.7 Верхняя оценка для полиномов Чебышёва

Мы можем видеть, что максимальное значение полинома Чебышёва на интервале $[\mu, L]$ достигается на концах отрезка в точках $a = \mu$ и $a = L$. Следовательно, мы можем использовать следующую верхнюю оценку:

$$\|P_k(A)\|_2 \leq P_k(\mu) = T_k\left(\frac{L+\mu-2\mu}{L-\mu}\right) \cdot T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1} = T_k(1) \cdot T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1} = T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1}$$

Используя определение числа обусловленности $\varkappa = \frac{L}{\mu}$, мы получаем:

$$\|P_k(A)\|_2 \leq T_k\left(\frac{\varkappa+1}{\varkappa-1}\right)^{-1} = T_k\left(1+\frac{2}{\varkappa-1}\right)^{-1} = T_k(1+\epsilon)^{-1}, \quad \epsilon = \frac{2}{\varkappa-1}.$$

Именно в этот момент явно возникнет ускорение. Мы ограничим значение $\|P_k(A)\|_2$ сверху величиной $\left(\frac{1}{1+\sqrt{\epsilon}}\right)^k$. Для этого детально изучим величину $|T_k(1+\epsilon)|$.

Чтобы ограничить $|P_k|$ сверху, мы должны ограничить $|T_k(1+\epsilon)|$ снизу.

1. Для любого $x \geq 1$, полиномы Чебышёва первого рода могут быть записаны как

$$T_k(x) = \cosh\left(k \operatorname{arccosh}(x)\right)$$
$$T_k(1+\epsilon) = \cosh\left(k \operatorname{arccosh}(1+\epsilon)\right).$$

2. Помните, что:

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \quad \operatorname{arccosh}(x) = \ln(x + \sqrt{x^2-1}).$$

3. Теперь, пусть $\phi = \operatorname{arccosh}(1+\epsilon)$,

$$e^\phi = 1 + \epsilon + \sqrt{2\epsilon+\epsilon^2} \geq 1 + \sqrt{\epsilon}.$$

4. Следовательно,

$$\begin{aligned} T_k(1+\epsilon) &= \cosh\left(k \operatorname{arccosh}(1+\epsilon)\right) \\ &= \cosh\left(k\phi\right) \\ &= \frac{e^{k\phi} + e^{-k\phi}}{2} \geq \frac{e^{k\phi}}{2} \\ &= \frac{\left(1+\sqrt{\epsilon}\right)^k}{2}. \end{aligned}$$

5. Наконец, мы получаем:

$$\begin{aligned} \|e_k\| \leq \|P_k(A)\|\|e_0\| &\leq \frac{2}{\left(1+\sqrt{\epsilon}\right)^k}\|e_0\| \\ &\leq 2\left(1+\sqrt{\frac{2}{\varkappa-1}}\right)^{-k}\|e_0\| \\ &\leq 2\exp\left(-\sqrt{\frac{2}{\varkappa-1}}k\right)\|e_0\| \end{aligned}$$

3.8 Ускоренный метод [1/2]

Из-за рекурсивного определения полиномов Чебышёва мы непосредственно получаем итерационную схему ускоренного алгоритма. Переформулируя рекурсию в терминах наших отшкалированных полиномов Чебышёва, мы получаем:

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$

Принимая во внимание, что $x = \frac{L+\mu-2a}{L-\mu}$, и:

$$P_k(a) = T_k\left(\frac{L+\mu-2a}{L-\mu}\right) T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1}$$
$$T_k\left(\frac{L+\mu-2a}{L-\mu}\right) = P_k(a) T_k\left(\frac{L+\mu}{L-\mu}\right)$$

$$T_{k-1}\left(\frac{L+\mu-2a}{L-\mu}\right) = P_{k-1}(a) T_{k-1}\left(\frac{L+\mu}{L-\mu}\right)$$
$$T_{k+1}\left(\frac{L+\mu-2a}{L-\mu}\right) = P_{k+1}(a) T_{k+1}\left(\frac{L+\mu}{L-\mu}\right)$$

$$P_{k+1}(a)t_{k+1} = 2\frac{L+\mu-2a}{L-\mu}P_k(a)t_k - P_{k-1}(a)t_{k-1}, \text{ где } t_k = T_k\left(\frac{L+\mu}{L-\mu}\right)$$
$$P_{k+1}(a) = 2\frac{L+\mu-2a}{L-\mu}P_k(a)\frac{t_k}{t_{k+1}} - P_{k-1}(a)\frac{t_{k-1}}{t_{k+1}}$$

Поскольку мы имеем $P_{k+1}(0) = P_k(0) = P_{k-1}(0) = 1$, получаем рекуррентную формулу вида:

$$P_{k+1}(a) = (1-\alpha_k a)P_k(a) + \beta_k\left(P_k(a) - P_{k-1}(a)\right).$$

3.9 Ускоренный метод [2/2]

Перегруппируя члены, мы получаем:

$$P_{k+1}(a) = (1+\beta_k)P_k(a) - \alpha_k a P_k(a) - \beta_k P_{k-1}(a),$$
$$P_{k+1}(a) = 2\frac{L+\mu}{L-\mu}\frac{t_k}{t_{k+1}}P_k(a) - \frac{4a}{L-\mu}\frac{t_k}{t_{k+1}}P_k(a) - \frac{t_{k-1}}{t_{k+1}}P_{k-1}(a)$$

$$\begin{cases} \beta_k = \dfrac{t_{k-1}}{t_{k+1}}, \\ \alpha_k = \dfrac{4}{L-\mu}\dfrac{t_k}{t_{k+1}}, \\ 1+\beta_k = 2\dfrac{L+\mu}{L-\mu}\dfrac{t_k}{t_{k+1}} \end{cases}$$

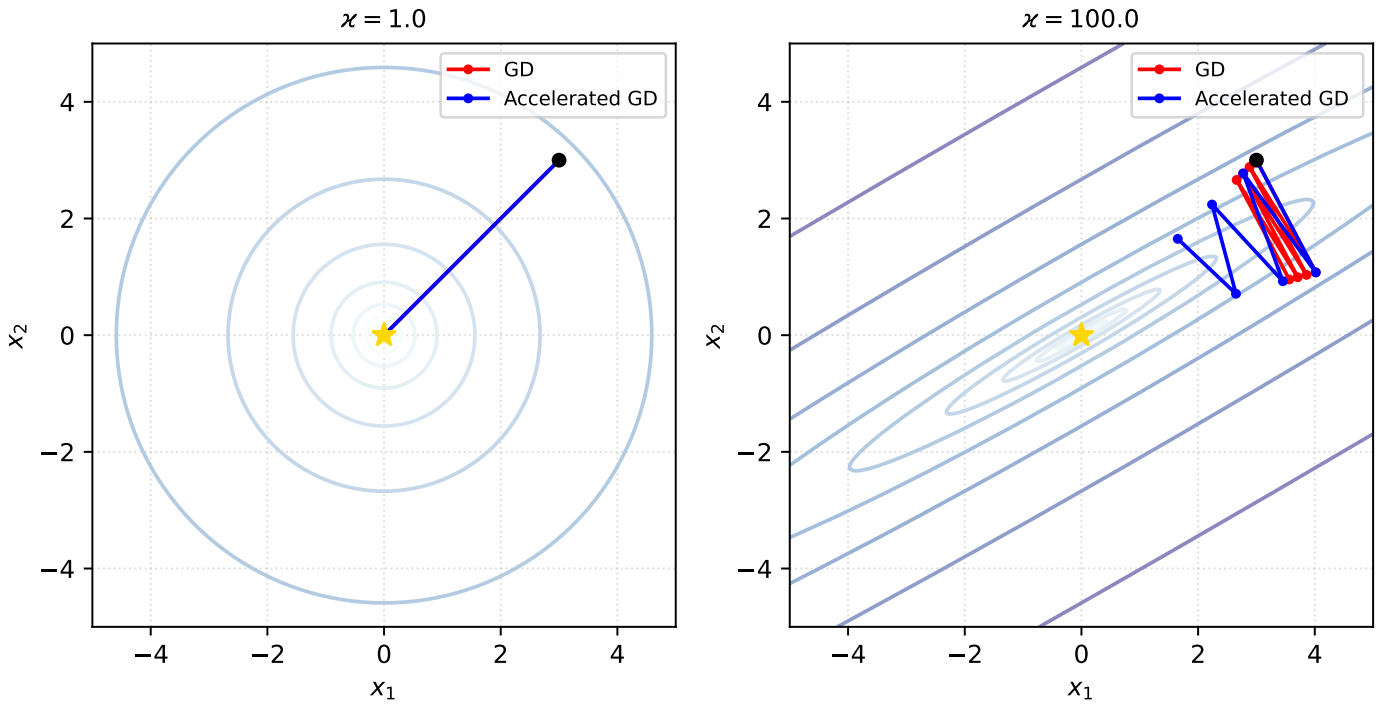
Мы почти закончили :) Помним, что $e_{k+1} = P_{k+1}(A)e_0$. Также обратим внимание, что мы работаем с квадратичной задачей, поэтому мы можем предположить $x^* = 0$ без ограничения общности. В этом случае $e_0 = x_0$ и $e_{k+1} = x_{k+1}$. \

$$\begin{aligned} x_{k+1} &= P_{k+1}(A)x_0 = (I - \alpha_k A)P_k(A)x_0 + \beta_k \left(P_k(A) - P_{k-1}(A)\right) x_0 \\ &= (I - \alpha_k A)x_k + \beta_k (x_k - x_{k-1}) \end{aligned}$$

Для квадратичной задачи мы имеем $\nabla f(x_k) = Ax_k$, поэтому мы можем переписать обновление как:

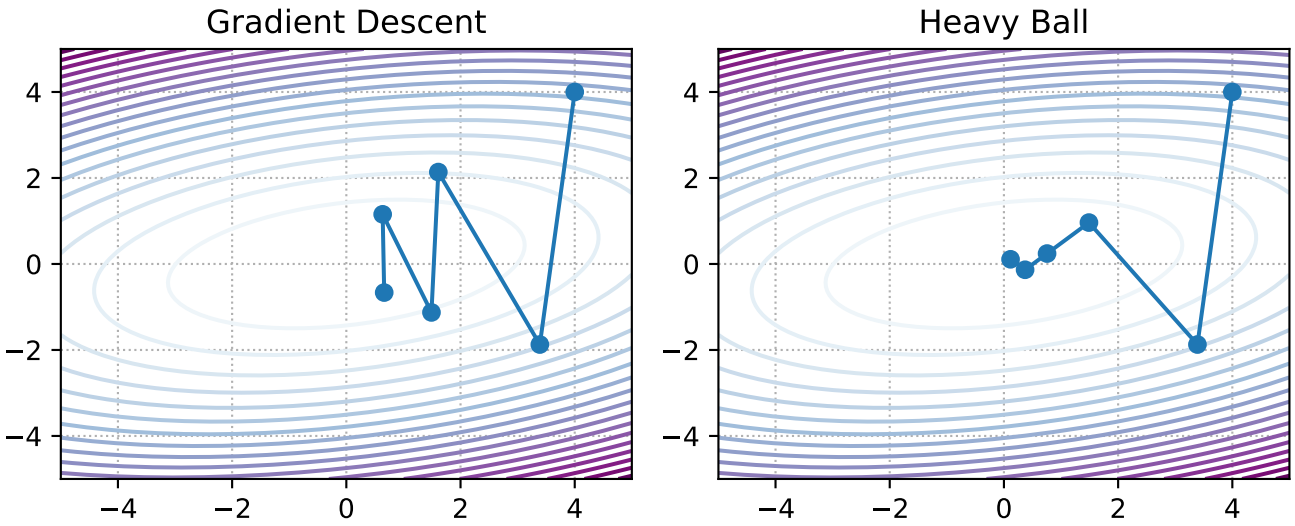
$$\boxed{x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1})}$$

3.10 Ускорение из первых принципов

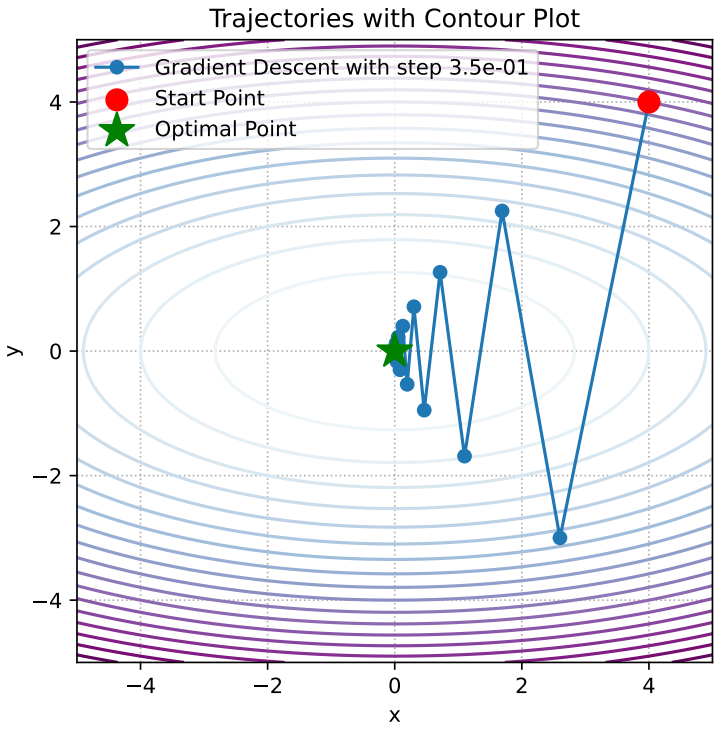


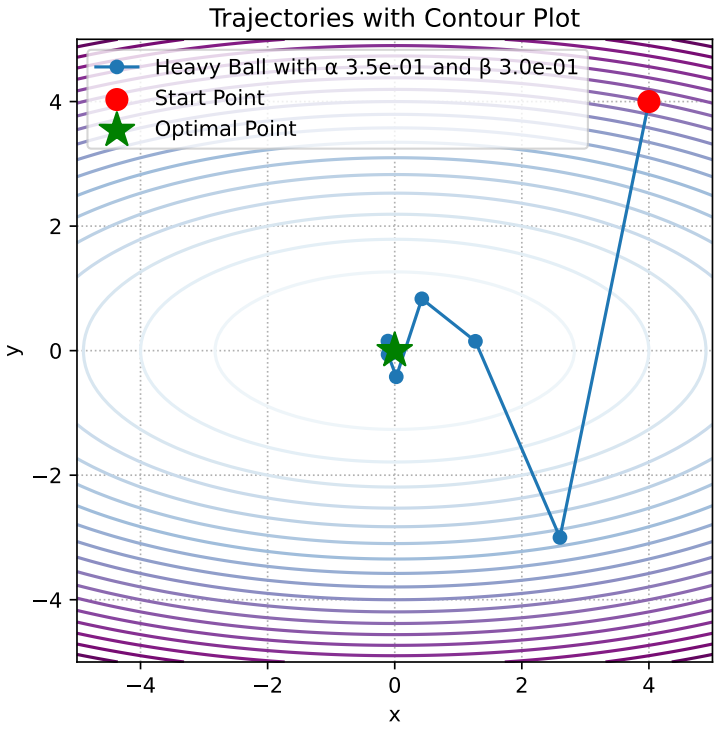
4 Метод тяжёлого шарика

4.1 Колебания и ускорение



4.2 Метод тяжёлого шарика Поляка





Давайте представим идею моментума (импульса, тяжёлого шарика), предложенную Б.Т. Поляком в 1964 году. Обновление метода тяжёлого шарика имеет вид

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}).$$

В нашем (квадратичном) случае это

$$\hat{x}_{k+1} = \hat{x}_k - \alpha\Lambda\hat{x}_k + \beta(\hat{x}_k - \hat{x}_{k-1}) = (I - \alpha\Lambda + \beta I)\hat{x}_k - \beta\hat{x}_{k-1}$$

Это можно переписать как

$$\begin{aligned}\hat{x}_{k+1} &= (I - \alpha\Lambda + \beta I)\hat{x}_k - \beta\hat{x}_{k-1},\\ \hat{x}_k &= \hat{x}_k.\end{aligned}$$

Давайте введем следующее обозначение: $\hat{z}_k = \begin{bmatrix}\hat{x}_{k+1}\\ \hat{x}_k\end{bmatrix}$. Следовательно, $\hat{z}_{k+1} = M\hat{z}_k$, где матрица итерации M имеет вид:

$$M = \begin{bmatrix} I - \alpha\Lambda + \beta I & -\beta I \\ I & 0_d \end{bmatrix}.$$

4.3 Сведение к скалярному случаю

Обратим внимание, что M является матрицей $2d \times 2d$ с четырьмя блочно-диагональными матрицами размера $d \times d$ внутри. Это означает, что мы можем изменить порядок координат, чтобы сделать M блочно-диагональной. Обратите внимание, что в уравнении ниже матрица M обозначает то же самое, что и в обозначении выше, за исключением описанной перестановки строк и столбцов. Мы используем эту небольшую перегрузку обозначений для простоты.

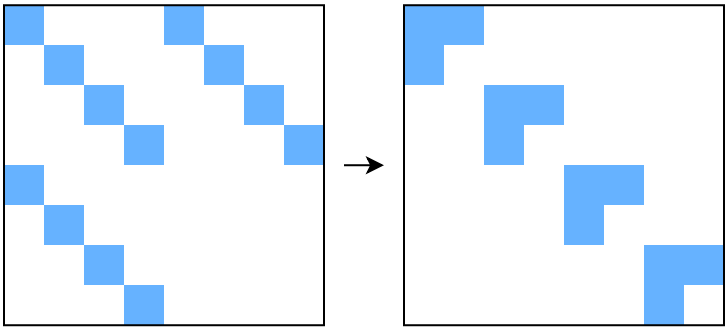


Рисунок 1: Иллюстрация перестановки матрицы M

$$\begin{bmatrix}\hat{x}_k^{(1)}\\ \vdots\\ \hat{x}_k^{(d)}\\ \hat{x}_{k-1}^{(1)}\\ \vdots\\ \hat{x}_{k-1}^{(d)}\end{bmatrix} \to \begin{bmatrix}\hat{x}_k^{(1)}\\ \hat{x}_{k-1}^{(1)}\\ \vdots\\ \hat{x}_k^{(d)}\\ \hat{x}_{k-1}^{(d)}\end{bmatrix} \qquad M = \begin{bmatrix} M_1 & & & \\ & M_2 & & \\ & & \dots & \\ & & & M_d \end{bmatrix}$$

где $\hat{x}_k^{(i)}$ является i-й координатой вектора $\hat{x}_k \in \mathbb{R}^d$ и M_i обозначает 2×2 матрицу. Переупорядочение позволяет нам исследовать динамику метода независимо от размерности. Асимптотическая скорость

сходимости $2d$-мерной последовательности векторов $\hat{z}_k$ определяется наихудшей скоростью сходимости среди его блока координат. Следовательно, достаточно исследовать оптимизацию в одномерном случае.

Для i-й координаты, где λ_i — i-е собственное значение матрицы A, имеем:

$$M_i = \begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}.$$

Метод будет сходиться, если $\rho(M) < 1$, и оптимальные параметры могут быть вычислены путем оптимизации спектрального радиуса

$$\alpha^*, \beta^* = \arg\min_{\alpha,\beta} \max_i \rho(M_i), \quad \alpha^* = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \quad \beta^* = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2.$$

Можно показать, что для таких параметров матрица M имеет комплексные собственные значения, которые образуют комплексно-сопряжённую пару, поэтому расстояние до оптимума (в этом случае $\|z_k\|$) обычно не убывает монотонно.

4.4 Сходимость метода тяжёлого шарика для квадратичной функции

Мы можем явно вычислить собственные значения M_i:

$$\lambda_1^M, \lambda_2^M = \lambda\left(\begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}\right) = \frac{1 + \beta - \alpha\lambda_i \pm \sqrt{(1 + \beta - \alpha\lambda_i)^2 - 4\beta}}{2}.$$

Когда α и β оптимальны (α^*, β^*), собственные значения являются комплексно-сопряженной парой $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$, т.е. $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$.

$$\text{Re}(\lambda^M) = \frac{L + \mu - 2\lambda_i}{(\sqrt{L} + \sqrt{\mu})^2}, \quad \text{Im}(\lambda^M) = \frac{\pm 2\sqrt{(L - \lambda_i)(\lambda_i - \mu)}}{(\sqrt{L} + \sqrt{\mu})^2}, \quad |\lambda^M| = \frac{L - \mu}{(\sqrt{L} + \sqrt{\mu})^2}.$$

И скорость сходимости не зависит от шага и равна $\sqrt{\beta^*}$.

Theorem

Предположим, что f является μ-сильно выпуклой и L-гладкой квадратичной функцией. Тогда метод тяжёлого шарика с параметрами

$$\alpha = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \beta = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2$$

сходится линейно:

$$\|x_k - x^*\|_2 \leq \left(\frac{\sqrt{\varkappa} - 1}{\sqrt{\varkappa} + 1}\right)^k \|x_0 - x^*\|$$

4.5 Глобальная сходимость метода тяжёлого шарика [3]

Theorem

Предположим, что f является гладкой и выпуклой и что

$$\beta \in [0, 1), \quad \alpha \in \left(0, \frac{2(1-\beta)}{L}\right).$$

Тогда последовательность $\{x_k\}$, генерируемая итерациями тяжёлого шарика, удовлетворяет

$$f(\overline{x}_T) - f^\star \leq \begin{cases} \frac{\|x_0 - x^\star\|^2}{2(T+1)}\left(\frac{L\beta}{1-\beta} + \frac{1-\beta}{\alpha}\right), & \text{if } \alpha \in (0, \frac{1-\beta}{L}], \\ \frac{\|x_0 - x^\star\|^2}{2(T+1)(2(1-\beta)-\alpha L)}\left(L\beta + \frac{(1-\beta)^2}{\alpha}\right), & \text{if } \alpha \in [\frac{1-\beta}{L}, \frac{2(1-\beta)}{L}), \end{cases}$$

где $\overline{x}_T$ среднее Чезаро последовательности итераций, т.е.

$$\overline{x}_T = \frac{1}{T+1}\sum_{k=0}^{T} x_k.$$

Theorem

Предположим, что f является гладкой и сильно выпуклой и что

$$\alpha \in \left(0, \frac{2}{L}\right), \quad 0 \leq \beta < \frac{1}{2}\left(\frac{\mu\alpha}{2} + \sqrt{\frac{\mu^2\alpha^2}{4} + 4(1 - \frac{\alpha L}{2})}\right).$$

Тогда последовательность $\{x_k\}$, генерируемая итерациями методатяжёлого шарика, сходится линейно к единственному оптимальному решению $x^\star$. В частности,

$$f(x_k) - f^\star \leq q^k(f(x_0) - f^\star),$$

где $q \in [0, 1)$.

4.6 Итоги по методу тяжёлого шарика

- Обеспечивает ускоренную сходимость для сильно выпуклых квадратичных задач.
- Локально ускоренная сходимость была доказана в оригинальной статье.
- Недавно [4] было доказано, что глобального ускорения сходимости для метода не существует.
- Метод не был чрезвычайно популярен до ML-бума.
- Сейчас он фактически является стандартом для практического ускорения методов градиентного спуска, в том числе для невыпуклых задач (обучение нейронных сетей).

[3] Глобальная сходимость метода тяжёлого шарика для выпуклой оптимизации, Euhanna Ghadimi et al.
[4] Provable non-accelerations of the heavy-ball method

5 Ускоренный градиентный метод Нестерова

5.1 Концепция ускоренного градиентного метода Нестерова

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$

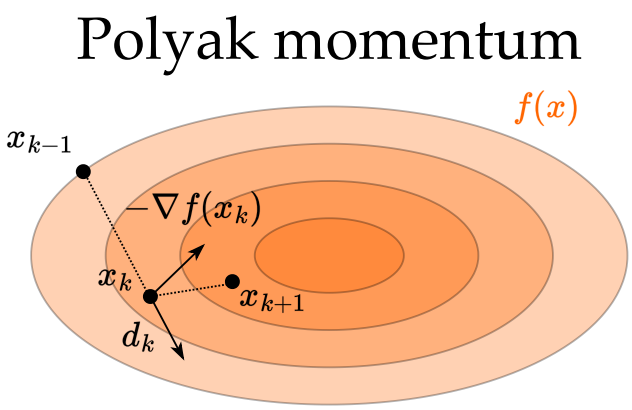
$$\begin{cases} y_{k+1} = x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} = y_{k+1} - \alpha \nabla f(y_{k+1}) \end{cases}$$

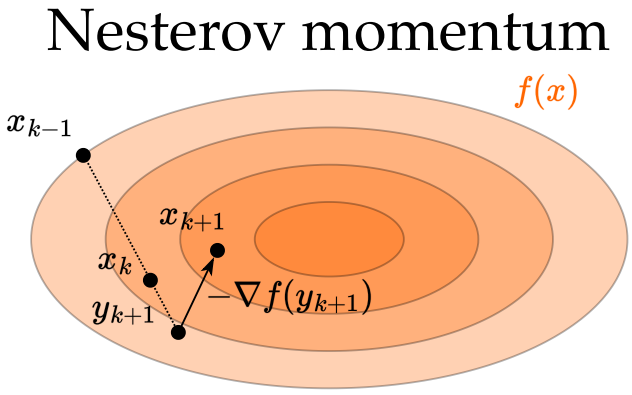
Давайте определим следующие обозначения

$$x^+ = x - \alpha \nabla f(x) \qquad \text{Градиентный шаг}$$
$$d_k = \beta_k (x_k - x_{k-1}) \qquad \text{Импульс}$$

Тогда мы можем записать:

$$x_{k+1} = x_k^+ \qquad \text{Градиентный спуск}$$
$$x_{k+1} = x_k^+ + d_k \qquad \text{Метод тяжёлого шарика}$$
$$x_{k+1} = (x_k + d_k)^+ \qquad \text{Ускоренный градиентный метод Нестерова}$$





5.2 Сходимость для выпуклых функций

Theorem

Предположим, что $f : \mathbb{R}^n \to \mathbb{R}$ является выпуклой и L-гладкой. Ускоренный градиентный метод Нестерова (NAG) предназначен для решения задачи минимизации, начиная с начальной точки $x_0 = y_0 \in \mathbb{R}^n$ и $\lambda_0 = 0$. Алгоритм выполняет следующие шаги:

$$\begin{aligned}
&\textbf{Обновление градиента:} && x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k) \\
&\textbf{Вес экстраполяции:} && \lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2} \\
& && \gamma_k = \frac{\lambda_k - 1}{\lambda_{k+1}} \\
&\textbf{Экстраполяция:} && y_{k+1} = x_{k+1} + \gamma_k \left(x_{k+1} - x_k\right)
\end{aligned}$$

Последовательность $\{f(x_k)\}_{k\in\mathbb{N}}$, генерируемая алгоритмом, сходится к оптимальному значению f^* со скоростью $\mathcal{O}\left(\frac{1}{k^2}\right)$, в частности:

$$f(x_k) - f^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}$$

5.3 Ускоренная сходимость для сильно выпуклых функций

Theorem

Предположим, что $f : \mathbb{R}^n \to \mathbb{R}$ является μ-сильно выпуклой и L-гладкой. Ускоренный градиентный метод Нестерова (NAG) предназначен для решения задачи минимизации, начиная с начальной точки $x_0 = y_0 \in \mathbb{R}^n$ и $\lambda_0 = 0$. Алгоритм выполняет следующие шаги:

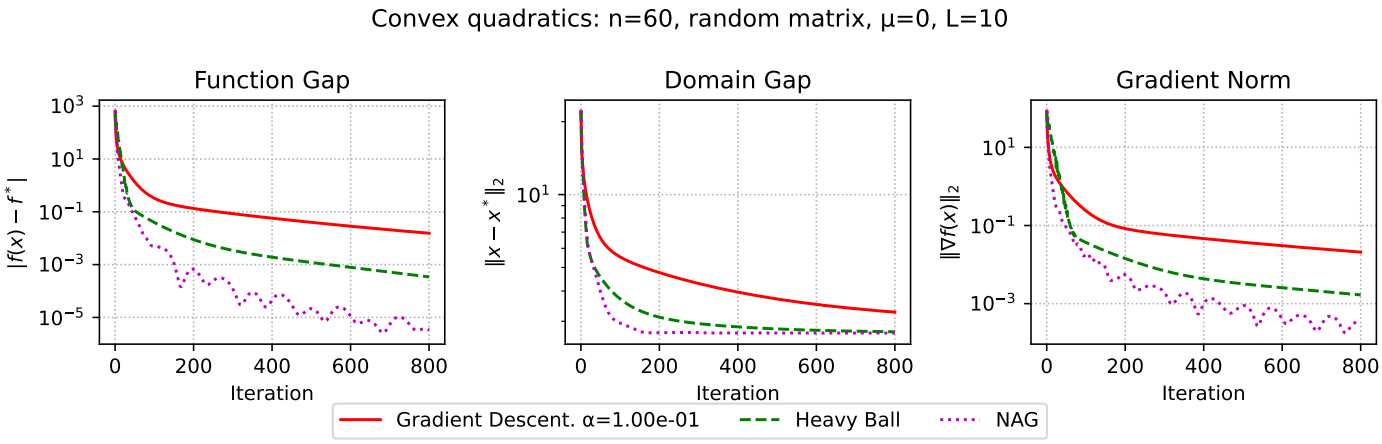
$$\begin{aligned}
&\textbf{Обновление градиента:} && x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k) \\
&\textbf{Экстраполяция:} && y_{k+1} = x_{k+1} - \gamma \left(x_{k+1} - x_k\right) \\
&\textbf{Вес экстраполяции:} && \gamma = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}
\end{aligned}$$

Последовательность $\{f(x_k)\}_{k\in\mathbb{N}}$, генерируемая алгоритмом, сходится к оптимальному значению f^* линейно:

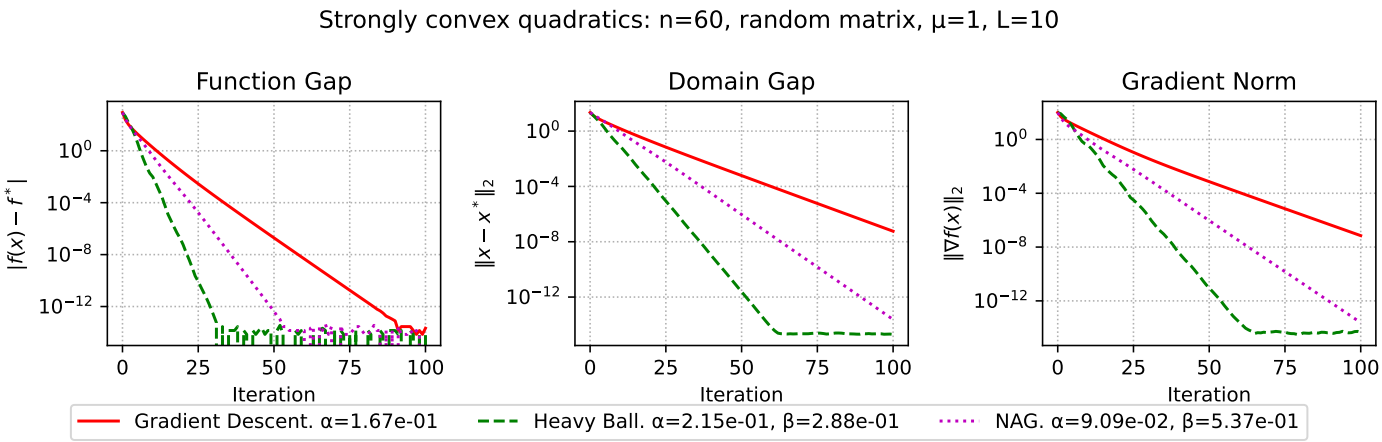
$$f(x_k) - f^* \leq \frac{\mu + L}{2}\|x_0 - x^*\|_2^2 \exp\left(-\frac{k}{\sqrt{\varkappa}}\right)$$

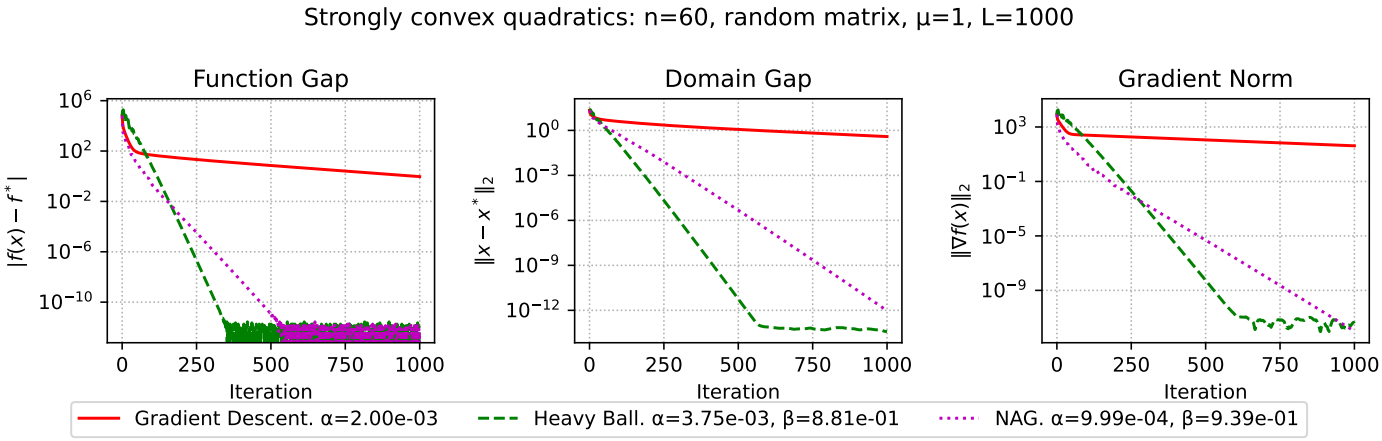
6 Численные эксперименты

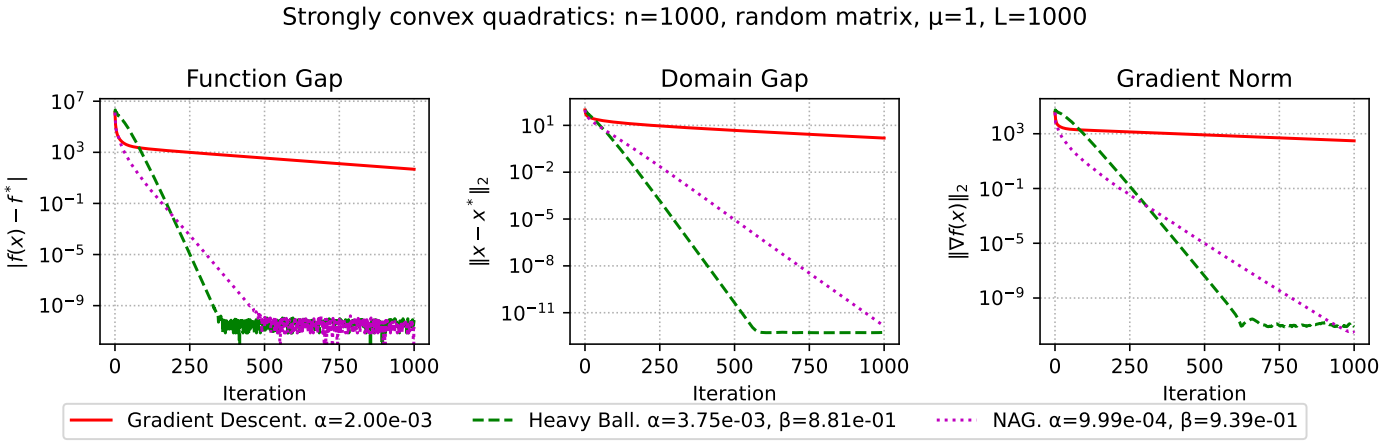
6.1 Выпуклая квадратичная задача (линейная регрессия)



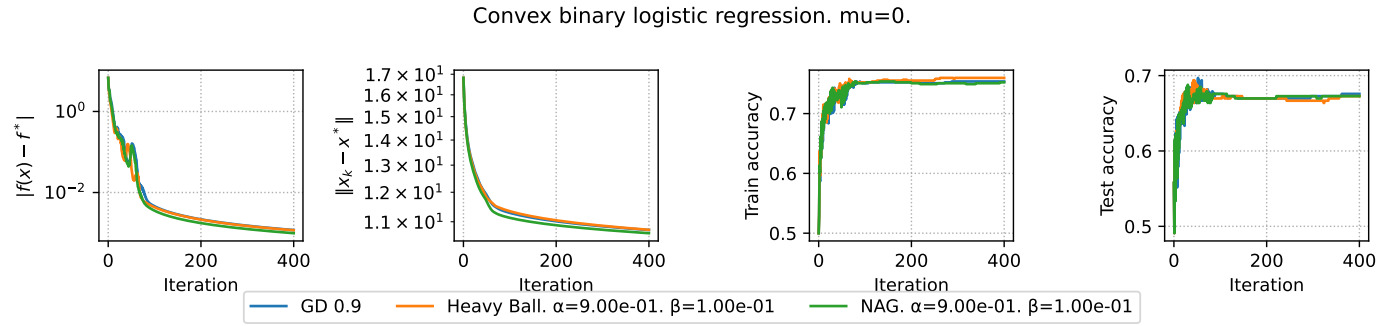
6.2 Сильно выпуклая квадратичная задача (регуляризованная линейная регрессия)

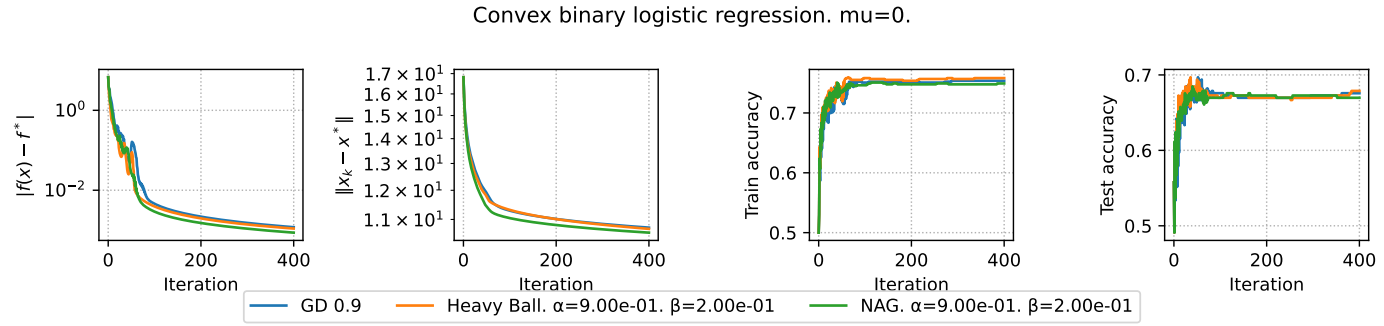


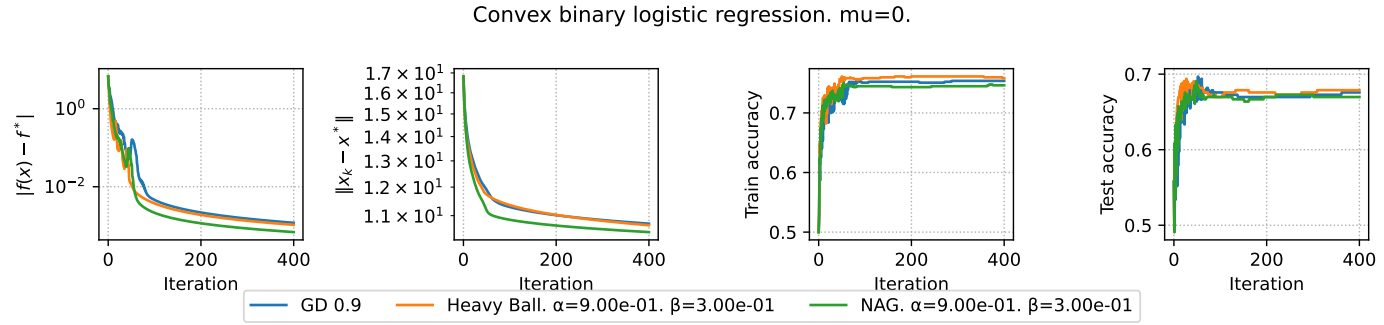


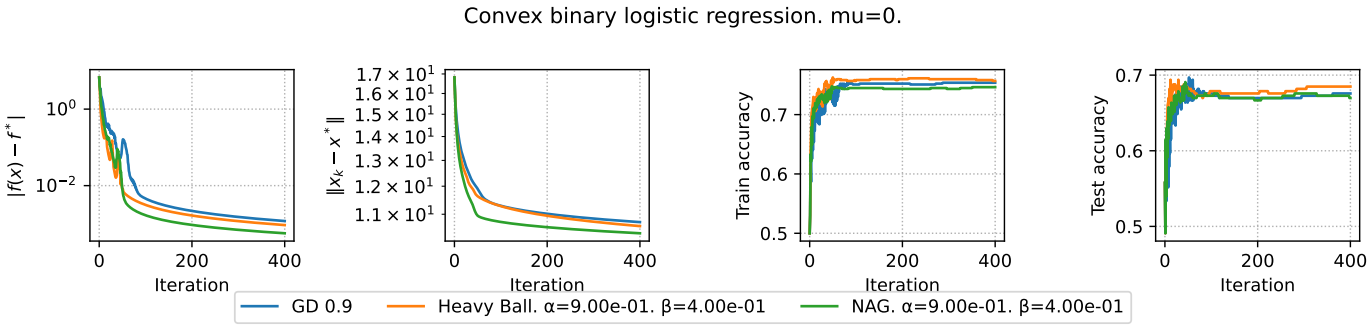


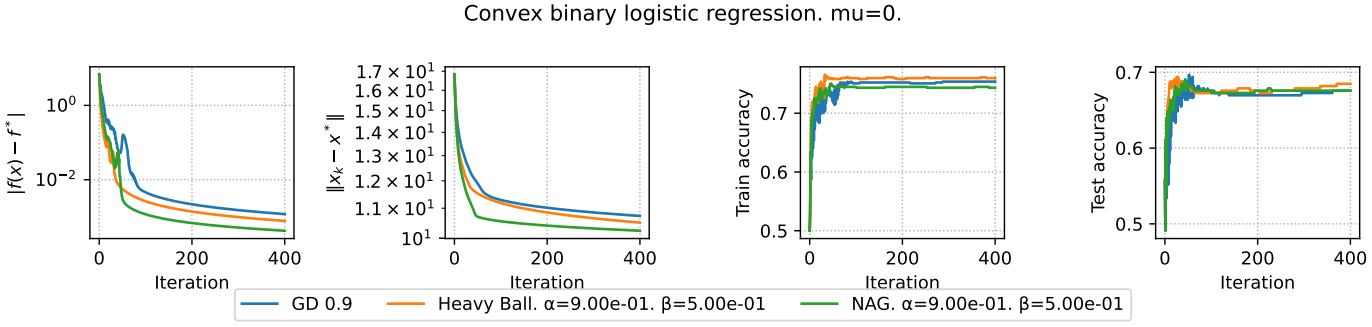
6.3 Выпуклая бинарная логистическая регрессия

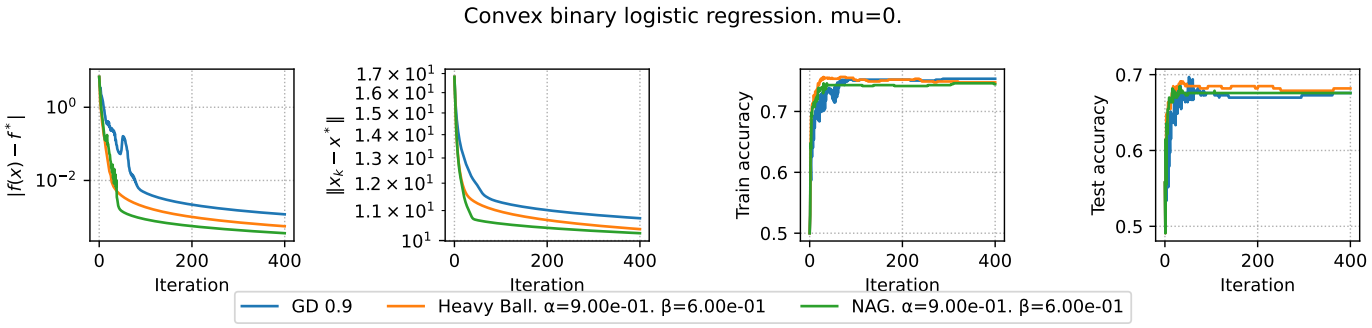


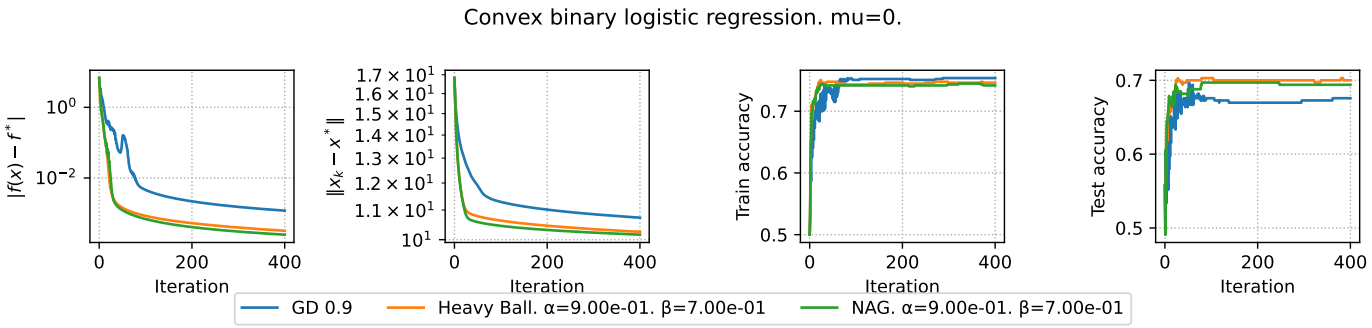


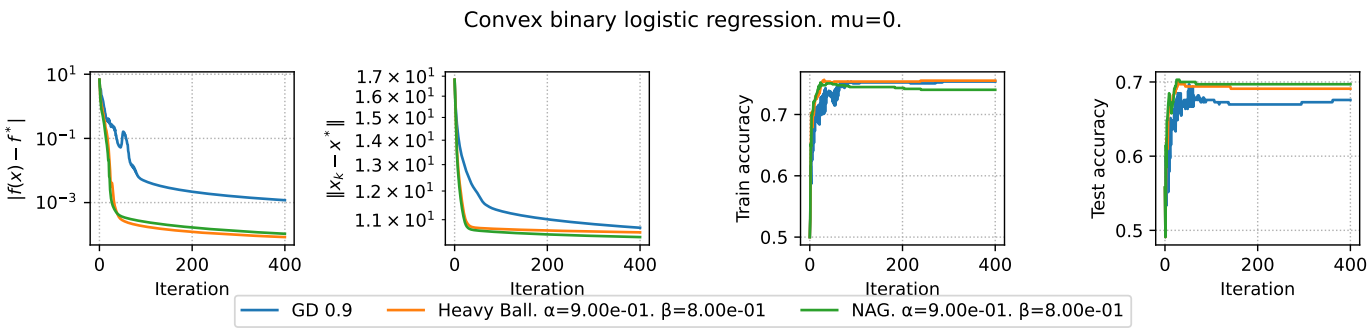


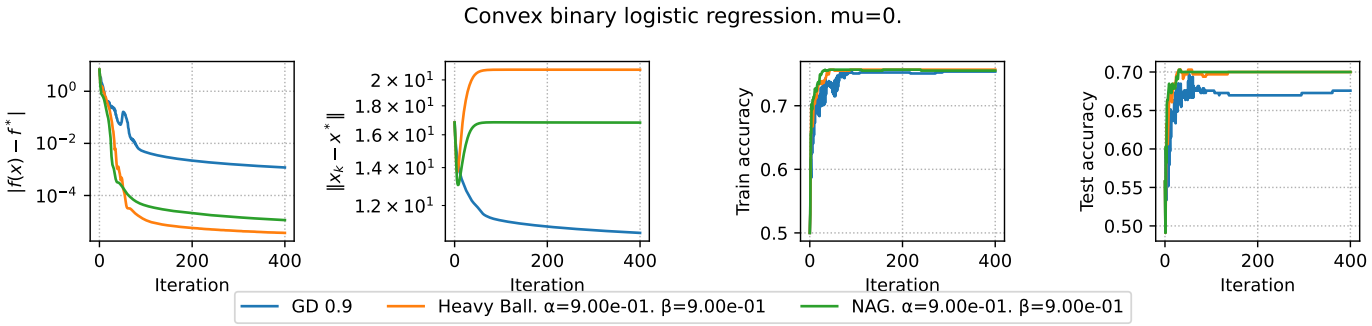

Convex binary logistic regression. mu=0.
$|f(x) - f^*|$
$\|x_k - x^*\|$
Train accuracy
Test accuracy
Iteration
GD 0.9
Heavy Ball. α=9.00e-01. β=4.00e-01
NAG. α=9.00e-01. β=4.00e-01

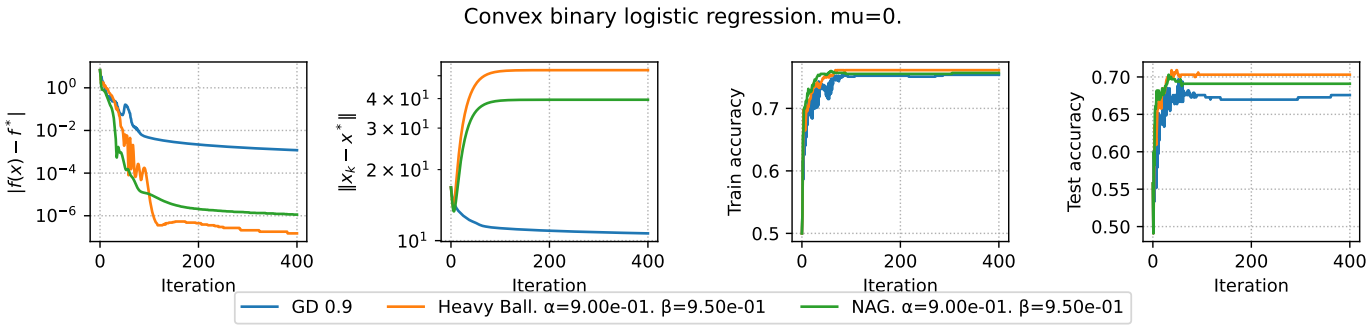

Convex binary logistic regression. mu=0.
$|f(x) - f^*|$
$\|x_k - x^*\|$
Train accuracy
Test accuracy
Iteration
GD 0.9
Heavy Ball. α=9.00e-01. β=5.00e-01
NAG. α=9.00e-01. β=5.00e-01

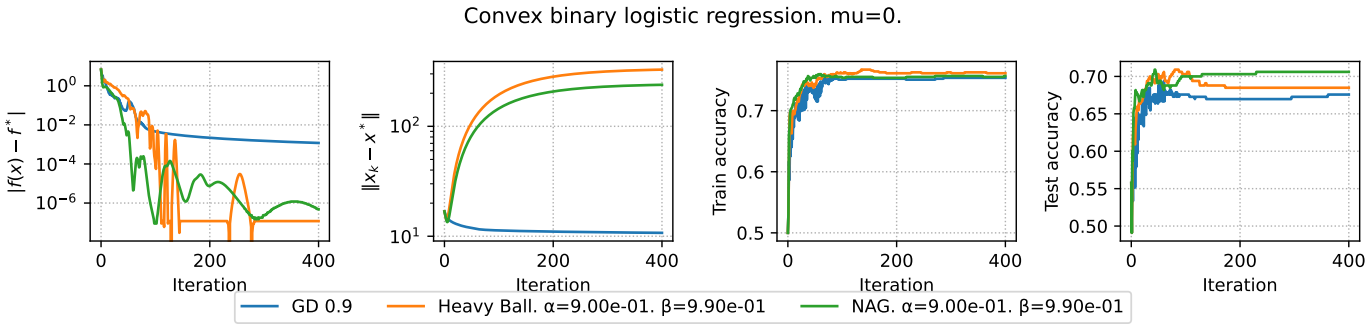

Convex binary logistic regression. mu=0.
$|f(x) - f^*|$
$\|x_k - x^*\|$
Train accuracy
Test accuracy
Iteration
GD 0.9
Heavy Ball. α=9.00e-01. β=6.00e-01
NAG. α=9.00e-01. β=6.00e-01


Convex binary logistic regression. mu=0.
$|f(x) - f^*|$
$\|x_k - x^*\|$
Train accuracy
Test accuracy
Iteration
GD 0.9
Heavy Ball. α=9.00e-01. β=7.00e-01
NAG. α=9.00e-01. β=7.00e-01

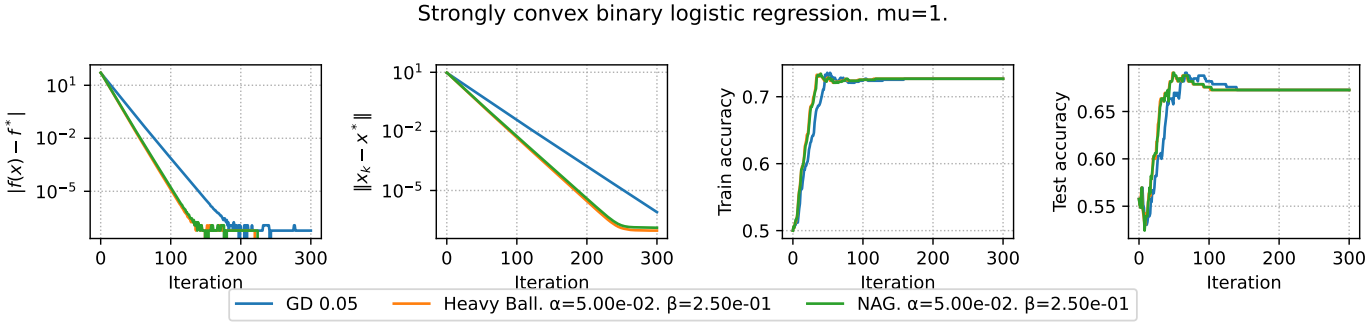

Convex binary logistic regression. mu=0.
$|f(x) - f^*|$
$\|x_k - x^*\|$
Train accuracy
Test accuracy
Iteration
GD 0.9
Heavy Ball. α=9.00e-01. β=8.00e-01
NAG. α=9.00e-01. β=8.00e-01

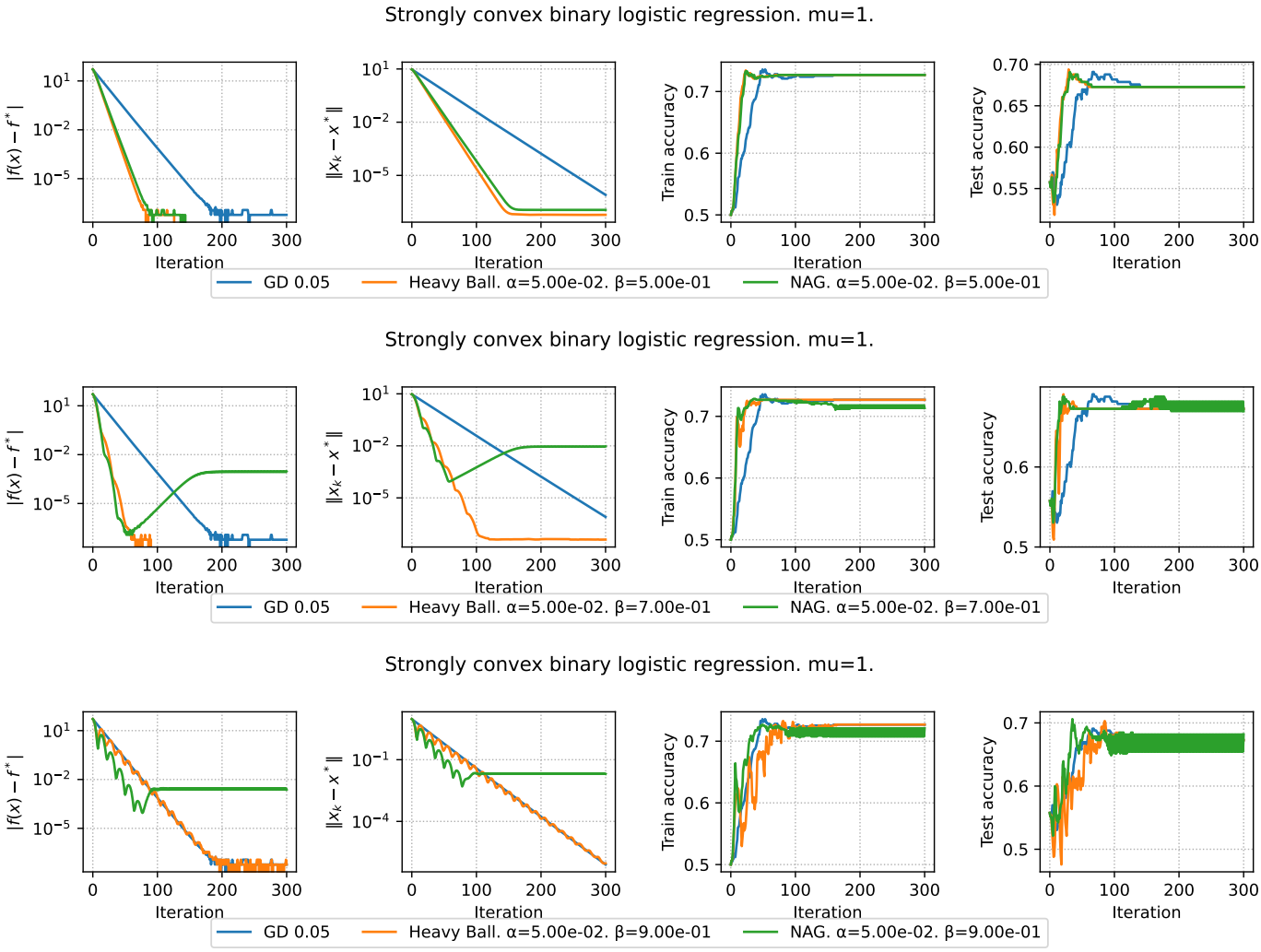






6.4 Сильно выпуклая бинарная логистическая регрессия





6.5 Нижние оценки для методов I порядка (источник)

Тип задачи	Критерий	Нижняя оценка	Верхняя оценка	Ссылка (Ниж.)	Ссылка (Верх.)
L-гладкая выпуклая	Зазор оптимальности	$\Omega\left(\sqrt{L}\,\varepsilon^{-1}\right)$	✔(точное совпадение)	[1], Теорема 2.1.7	[1], Теорема 2.2.2
L-гладкая μ-сильно выпуклая	Зазор оптимальности	$\Omega\left(\sqrt{\varkappa}\,\log\frac{1}{\varepsilon}\right)$	✔	[1], Теорема 2.1.13	[1], Теорема 2.2.2
Негладкая G-липшицева выпуклая	Зазор оптимальности	$\Omega\left(G^2\,\varepsilon^{-2}\right)$	✔(точное совпадение)	[1], Теорема 3.2.1	[1], Теорема 3.2.2
Негладкая G-липшицева μ-сильно выпуклая	Зазор оптимальности	$\Omega\left(G^2\,(\mu\varepsilon)^{-1}\right)$	✔	[1], Теорема 3.2.5	[3], Теорема 3.9
L-гладкая выпуклая (сходимость по функции)	Стационарность	$\Omega\left(\sqrt{\Delta L}\,\varepsilon^{-1}\right)$	✔(с точностью до логарифмического множителя)	[2], Теорема 1	[2], Приложение A.1
L-гладкая выпуклая (сходимость по аргументу)	Стационарность	$\Omega\left(\sqrt{DL}\,\varepsilon^{-1/2}\right)$	✔	[2], Теорема 1	[6], Раздел 6.5
L-гладкая невыпуклая	Стационарность	$\Omega\left(\Delta L\,\varepsilon^{-2}\right)$	✔	[5], Теорема 1	[7], Теорема 10.15
Негладкая G-липшицева ρ-слабо выпуклая (WC)	Квази-стационарность	Неизвестно	$\mathcal{O}\left(\varepsilon^{-4}\right)$	/	[8], Следствие 2.2
L-гладкая μ-PL	Зазор оптимальности	$\Omega\left(\varkappa\log\frac{1}{\varepsilon}\right)$	✔	[9], Теорема 3	[10], Теорема 1

Источники:

- [1] - Lectures on Convex Optimization, Y. Nesterov.
- [2] - Lower bounds for finding stationary points II: first-order methods, Y. Carmon, J.C. Duchi, O. Hinder, A. Sidford.
- [3] - Convex optimization: Algorithms and complexity, S. Bubeck, others.

- [4] - Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions D. Kim, J.A. Fessler.
- [5] - Lower bounds for finding stationary points I, Y. Carmon, J.C. Duchi, O. Hinder, A. Sidford.
- [6] - Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions, D. Kim, J.A. Fessler.
- [7] - First-order methods in optimization, A. Beck. SIAM. 2017.
- [8] - Stochastic subgradient method converges at the rate $ O (k^{-1/4}) $ on weakly convex functions, D. Davis, D. Drusvyatskiy.
- [9] - On the lower bound of minimizing Polyak-Lojasiewicz functions, P. Yue, C. Fang, Z. Lin.
- [10] - Linear convergence of gradient and proximal-gradient methods under the Polyak-Lojasiewicz condition, H. Karimi, J. Nutini, M. Schmidt.

Обозначения:

- Зазор оптимальности: $f(x_k) - f^* \leq \varepsilon$
- Стационарность: $\|\nabla f(x_k)\| \leq \varepsilon$
- Квази-стационарность: $\|\nabla f_\lambda(x_k)\| \leq \varepsilon$, где $f_\lambda(x) = \inf\limits_{y \in \mathbb{R}^n} \left(f(y) + \frac{1}{2\lambda}\|y - x\|^2\right)$
- *Липшицевость функции*: $|f(x) - f(y)| \leq G\|x - y\| \forall x, y \in \mathbb{R}^n$
- *Липшицевость градиента* (L-гладкость): $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \forall x, y \in \mathbb{R}^n$
- μ-сильная выпуклость: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{\mu}{2}\lambda(1-\lambda)\|x - y\|^2$
- ρ-слабо выпуклая функция: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) + \rho\lambda(1-\lambda)\|x - y\|^2 \forall x, y \in \mathbb{R}^n$
- Число обусловленности: $\varkappa = \frac{L}{\mu}$
- Зазор в начальной точке: $f(x_0) - f^* \leq \Delta$
- Зазор по аргументу: $D = \|x_0 - x^*\|$

7 Задачи на дом

1. ***Локальная сходимость метода тяжелого шарика.*** [20 баллов] Мы будем работать с методом тяжелого шарика

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \tag{HB}$$

Известно, что для квадратичных функций оптимальный выбор гиперпараметров равен $\alpha^* = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \beta^* = \frac{(\sqrt{L} - \sqrt{\mu})^2}{(\sqrt{L} + \sqrt{\mu})^2}$, который обеспечивает ускоренную линейную сходимость для сильно выпуклой квадратичной функции.

Рассмотрим следующую непрерывно дифференцируемую, сильно выпуклую с параметром μ, и гладкую с параметром L функцию:

$$f(x) = \begin{cases} \frac{25}{2}x^2, & \text{if } x < 1 \\ \frac{1}{2}x^2 + 24x - 12, & \text{if } 1 \leq x < 2 \\ \frac{25}{2}x^2 - 24x + 36, & \text{if } x \geq 2 \end{cases} \quad \nabla f(x) = \begin{cases} 25x, & \text{if } x < 1 \\ x + 24, & \text{if } 1 \leq x < 2 \\ 25x - 24, & \text{if } x \geq 2 \end{cases}$$

1. Как доказать, что данная функция является выпуклой? Сильно выпуклой? Гладкой?
2. Найдите константы μ и L для данной функции.

3. Постройте график функции для $x \in [-4, 4]$.

4. Запустите метод тяжелого шарика для функции с оптимальными гиперпараметрами $\alpha^* = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \beta^* = \frac{(\sqrt{L}-\sqrt{\mu})^2}{(\sqrt{L}+\sqrt{\mu})^2}$ для квадратичной функции, начиная с $x_0 = 3.5$. Если вы все сделали правильно, вы должны получить что-то вроде

   (heavy_ball_conv.mp4)

   Вы можете использовать следующий код для построения:

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML

# Gradient of the function
def grad_f(x):
    ...

# Heavy Ball method implementation
def heavy_ball_method(alpha, beta, x0, num_iterations):
    x = np.zeros(num_iterations + 1)
    x_prev = x0
    x_curr = x0  # Initialize x[1] same as x[0] to start the algorithm
    for i in range(num_iterations):
        x[i] = x_curr
        x_new = x_curr - alpha * grad_f(x_curr) + beta * (x_curr - x_prev)
        x_prev = x_curr
        x_curr = x_new
    x[num_iterations] = x_curr
    return x

# Parameters
L = ...
mu = ...
alpha_star = ...
beta_star = ...
x0 = ...
num_iterations = 30

# Generate the trajectory of the method
trajectory = heavy_ball_method(alpha_star, beta_star, x0, num_iterations)

# Setup the figure and axes for the animation
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(7, 3.5))
fig.suptitle("Heavy ball method with optimal hyperparameters  *  *")

# Function for updating the animation
def update(i):
    ax1.clear()
    ax2.clear()
```

```
    # Plot f(x) and trajectory
    x_vals = np.linspace(-4, 4, 100)
    f_vals = np.piecewise(x_vals, [x_vals < 1, (x_vals >= 1) & (x_vals < 2), x_vals >= 2],
                          [lambda x: 12.5 * x**2, lambda x: .5 * x**2 + 24 * x - 12, lambda x
    ax1.plot(x_vals, f_vals, 'b-')
    ax1.plot(trajectory[:i], [12.5 * x**2 if x < 1 else .5 * x**2 + 24 * x - 12 if x < 2 el
    # Add vertical dashed lines at x=1 and x=2 on the left subplot
    ax1.axvline(x=1, color='navy', linestyle='--')
    ax1.axvline(x=2, color='navy', linestyle='--')

    # Plot function value from iteration
    f_trajectory = [None for x in trajectory]
    f_trajectory[:i] = [12.5 * x**2 if x < 1 else .5 * x**2 + 24 * x - 12 if x < 2 else 12.
    ax2.plot(range(len(trajectory)), f_trajectory, 'ro-')
    ax2.set_xlim(0, len(trajectory))
    ax2.set_ylim(min(f_vals), max(f_vals))
    # Add horizontal dashed lines at f(1) and f(2) on the right subplot
    f_1 = 12.5 * 1.0**2
    f_2 = .5 * 2.**2 + 24 * 2. - 12
    ax2.axhline(y=f_1, color='navy', linestyle='--')
    ax2.axhline(y=f_2, color='navy', linestyle='--')

    # ax1.set_title("Function f(x) and Trajectory")
    ax1.set_xlabel("x")
    ax1.set_ylabel("f(x)")
    ax1.grid(linestyle=":")

    # ax2.set_title("Function Value from Iteration")
    ax2.set_xlabel("Iteration")
    ax2.set_ylabel("f(x)")
    ax2.grid(linestyle=":")

    plt.tight_layout()

# Create the animation
ani = animation.FuncAnimation(fig, update, frames=num_iterations, repeat=False, interval=10
HTML(ani.to_jshtml())
```

5. Измените начальную точку на $x_0 = 3.4$. Что вы видите? Как можно назвать такое поведение метода?
6. Измените гиперпараметры $\alpha^{\text{Global}} = \frac{2}{L}, \beta^{\text{Global}} = \frac{\mu}{L}$ и запустите метод снова с $x_0 = 3.4$. Проверьте, что вы получили ускоренную сходимость.

Контекст: этот контрпример был предоставлен в статье, в то время как глобальная сходимость метода тяжелого шарика для общей гладкой сильно выпуклой функции была введена в другой статье. Недавно было предложено, что метод тяжелого шарика (HB) доказуемо не достигает ускоренной сходимости на гладких сильно выпуклых задачах.

2. [40 points] В этой задаче мы будем работать с ускоренными методами, применяемыми к задаче логистической регрессии. Хорошее визуальное введение по теме доступно здесь.

Логистическая регрессия является стандартной моделью в задачах классификации. Для простоты рассмотрим только случай бинарной классификации. Интуитивно, задача формулируется следующим образом: Есть обучающая выборка $\{(a_i, b_i)\}_{i=1}^m$, состоящая из m векторов $a_i \in \mathbb{R}^n$ (относящихся к признакам) и соответствующих чисел $b_i \in \{-1, 1\}$ (относящихся к классам или меткам). Цель состоит в том, чтобы построить алгоритм $b(\cdot)$, который для любого нового вектора признаков a автоматически определяет его класс $b(a) \in \{-1, 1\}$.

В модели логистической регрессии класс определяется на основе знака линейной комбинации компонентов вектора a с некоторыми фиксированными коэффициентами $x \in \mathbb{R}^n$:

$$b(a) := \text{sign}(\langle a, x \rangle).$$

Коэффициенты x являются параметрами модели и подбираются путем решения следующей оптимизационной задачи:

$$\min_{x \in \mathbb{R}^n} \left(\frac{1}{m} \sum_{i=1}^m \ln(1 + \exp(-b_i \langle a_i, x \rangle)) + \frac{\lambda}{2} \|x\|^2 \right), \tag{LogReg}$$

где $\lambda \geq 0$ является коэффициентом регуляризации (параметром модели).

1. Будет ли задача LogReg выпуклой для $\lambda = 0$? Каков градиент целевой функции? Будет ли она сильно выпуклой? Что будет, если вы добавите регуляризацию с $\lambda > 0$?
2. Мы будем работать с реальными данными для A и b: возьмите датасет mushrooms. Будьте осторожны, вам нужно будет предсказать, является ли гриб ядовитым или съедобным. Плохая модель может привести к смерти в этом упражнении.

```python
import requests
from sklearn.datasets import load_svmlight_file

# URL of the file to download
url = 'https://cu25.fmin.xyz/files/mushrooms.txt'

# Download the file and save it locally
response = requests.get(url)
dataset = 'mushrooms.txt'

# Ensure the request was successful
if response.status_code == 200:
    with open(dataset, 'wb') as f:
        f.write(response.content)

    # Load the dataset from the downloaded file
    data = load_svmlight_file(dataset)
    A, b = data[0].toarray(), data[1]
    n, d = A.shape

    print("Data loaded successfully.")
    print(f"Number of samples: {n}, Number of features: {d}")
else:
    print(f"Failed to download the file. Status code: {response.status_code}")
```

3. Разделите данные на две части: обучение и тест. Мы будем обучать модель на A_{train}, b_{train} и измерять точность модели на A_{test}, b_{test}.

```
from sklearn.model_selection import train_test_split
# Split the data into training and test sets
A_train, A_test, b_train, b_test = train_test_split(A, b, test_size=0.2, random_state=214)
```

4. Для обучения A_{train}, b_{train}, оцените константы μ, L задачи оптимизации. Используйте одно и то же маленькое значение λ для всех экспериментов

5. Используя градиентный спуск с шагом $\frac{1}{L}$, обучите модель. Постройте график: точность в зависимости от номера итерации.

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \tag{HB}$$

Зафиксируйте шаг $\alpha = \frac{1}{L}$ и найдите различные значения импульса β от -1 до 1. Выберите свой собственный критерий сходимости и постройте сходимость для нескольких значений импульса на одном графике. Сходится ли она всегда монотонно?

6. Для лучшего значения импульса β, постройте зависимость точности модели на тестовой выборке от времени работы метода. Добавьте на тот же график сходимость градиентного спуска с шагом $\frac{1}{L}$. Сделайте вывод. Убедитесь, что вы используете одно и то же значение λ для обоих методов.

7. Решите задачу логистической регрессии с использованием метода Нэстерова.

$$x_{k+1} = x_k - \alpha \nabla f(x_k + \beta(x_k - x_{k-1})) + \beta(x_k - x_{k-1}) \tag{NAG}$$

Зафиксируйте шаг $\frac{1}{L}$ и найдите различные значения импульса β от -1 до 1. Проверьте также значения импульса равные $\frac{k}{k+3}$, $\frac{k}{k+2}$, $\frac{k}{k+1}$ (k - число итераций), и если вы решаете сильно выпуклую задачу, также $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$. Постройте сходимость метода в зависимости от числа итераций (выберите свой собственный критерий сходимости) для различных значений импульса. Сходится ли она всегда монотонно?

8. Для лучшего значения импульса β, постройте зависимость точности модели на тестовой выборке от времени работы метода. Добавьте этот график к графикам для тяжелого шарика и градиентного спуска из предыдущих шагов. Сделайте вывод.

9. Теперь мы отбросим оценку значения L и будем пытаться сделать его адаптивным. Давайте сделаем выбор константы L адаптивным.

$$f(y) \leq f(x^k) + \langle \nabla f(x^k), y - x^k \rangle + \frac{L}{2}\|x^k - y\|_2^2$$

В частности, процедура может работать так:

```
def backtracking_L(f, grad, x, h, L0, rho, maxiter=100):
    L = L0
    fx = f(x)
    gradx = grad(x)
    iter = 0
    while iter < maxiter :
```

```
        y = x - 1 / L * h
        if f(y) <= fx - 1 / L gradx.dot(h) + 1 / (2 * L) h.dot(h):
            break
        else:
            L = L * rho

        iter += 1
    return L
```

Что должно быть взято как h? Должно ли ρ быть больше или меньше 1? Должно ли L_0 быть больше или меньше? Постройте график, аналогичный тому, что был в предыдущем шаге для L, вычисленного адаптивно (6 строк - GD, HB, NAG, GD adaptive L, HB adaptive L, NAG adaptive L)