Методы оптимизации

Семинар 3: Методы градиентного спуска и Ньютона. Детали реализации и примеры работы

ВМК, осень 2018

Методы спуска. Общая схема

Рассматриваемая задача: $\min_{x \in \mathbb{R}^n} f(x)$, где $f \in C^1(\mathbb{R}^n \to \mathbb{R})$.

Общая схема методов спуска:

1. Выбрать направление спуска $d_k \in \mathbb{R}^n$.
2. *(Линейный поиск)* Выбрать длину шага $\alpha_k \geq 0$.
3. *(Обновление)* $x_{k+1} \leftarrow x_k + \alpha_k d_k$

Направление спуска: $\langle \nabla f(x_k), d_k \rangle < 0$.

Линейный поиск

Функция: $\phi_k(\alpha) := f(x_k + \alpha d_k), \quad \phi_k : [0, +\infty) \to \mathbb{R}.$

Стратегии выбора шага:

- **Постоянный шаг:** $\alpha_k = \text{const}$ для всех $k \geq 0$.
- **Наискорейший спуск:** $\alpha_k := \text{argmin}_{\alpha \geq 0} \phi_k(\alpha)$.
- **Неточный линейный поиск**.

Метод дробления шага (с условием Армихо):

1. Начать с $\alpha := \alpha_k^{(0)}$.
2. Пока не выполнено $\phi_k(\alpha) \leq \phi_k(0) + c_1 \alpha \phi_k'(0)$
   (a) $\alpha \leftarrow \alpha/2$

Альтернативные методы основаны на сильных условиях Вульфа:

$$\phi_k(\alpha) \leq \phi_k(0) + c_1 \alpha \phi_k'(0)$$
$$|\phi_k'(\alpha)| \leq c_2 |\phi_k'(0)|$$

Используют внутри квадратичные/кубические аппроксимации.

Градиентный спуск

Метод градиентного спуска:

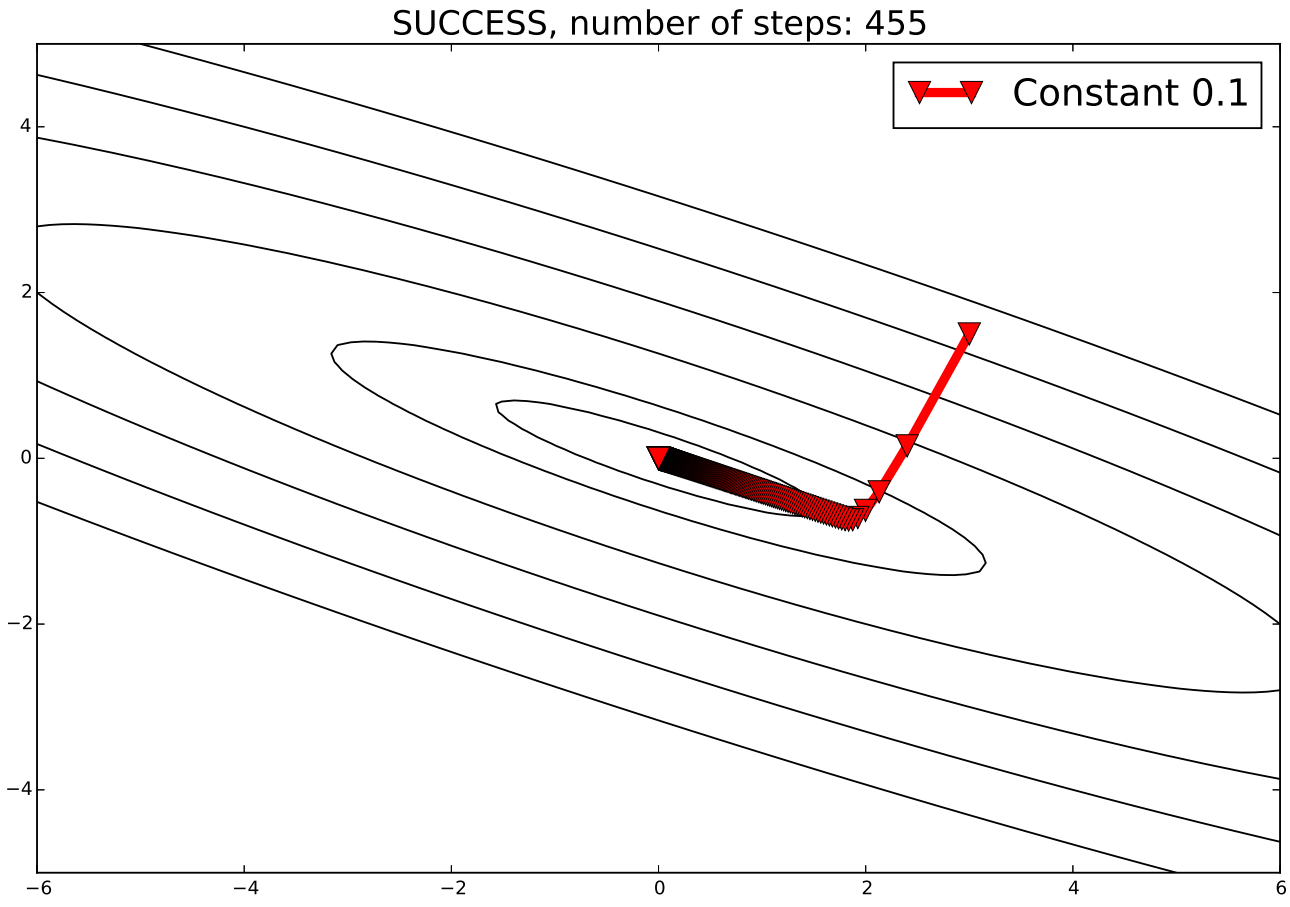
$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

- **Интерпретация:** Метод спуска с направлением

$$d_k = -\nabla f(x_k)$$

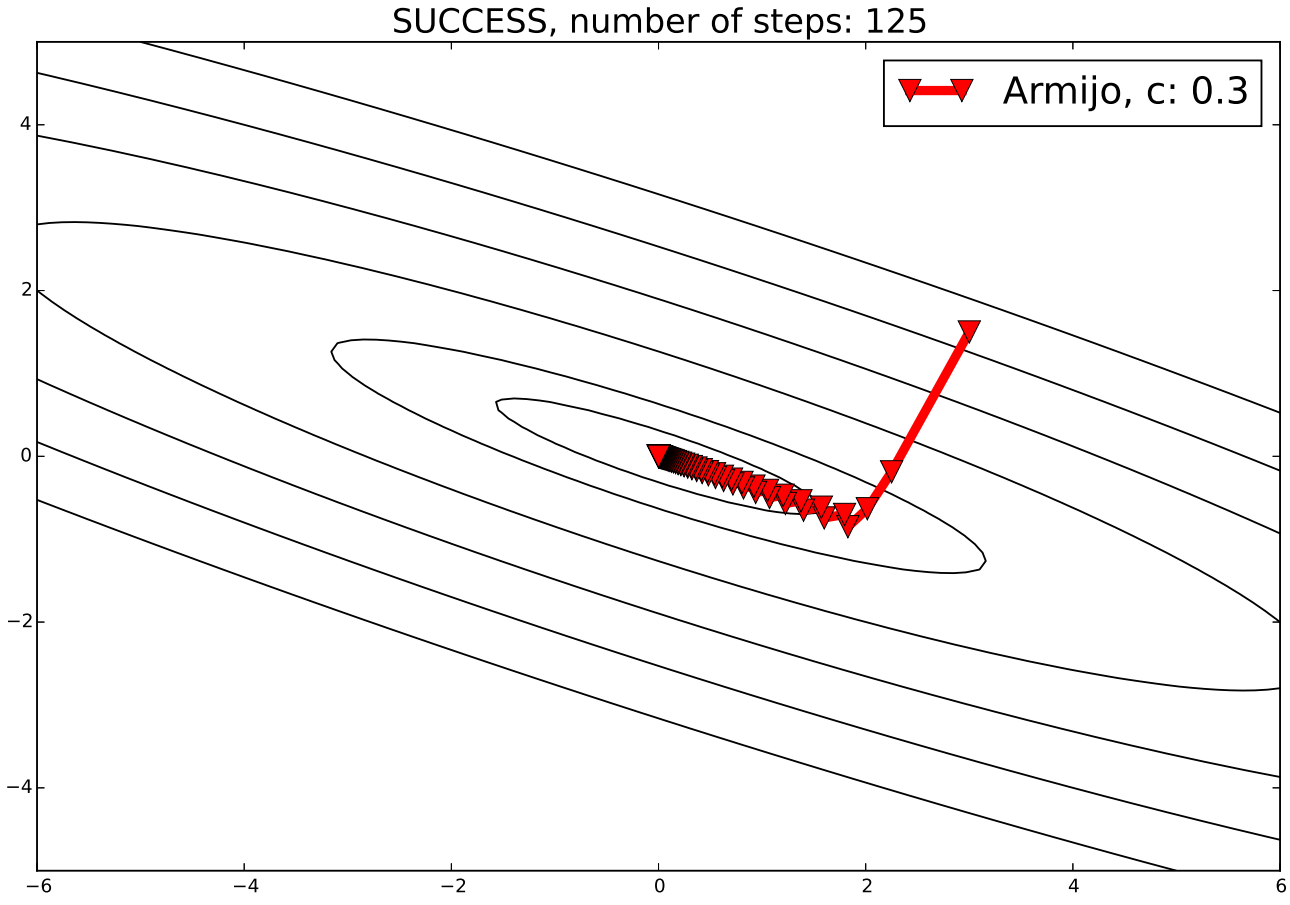
Градиентный спуск: постоянный шаг

Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle, \quad A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}.$



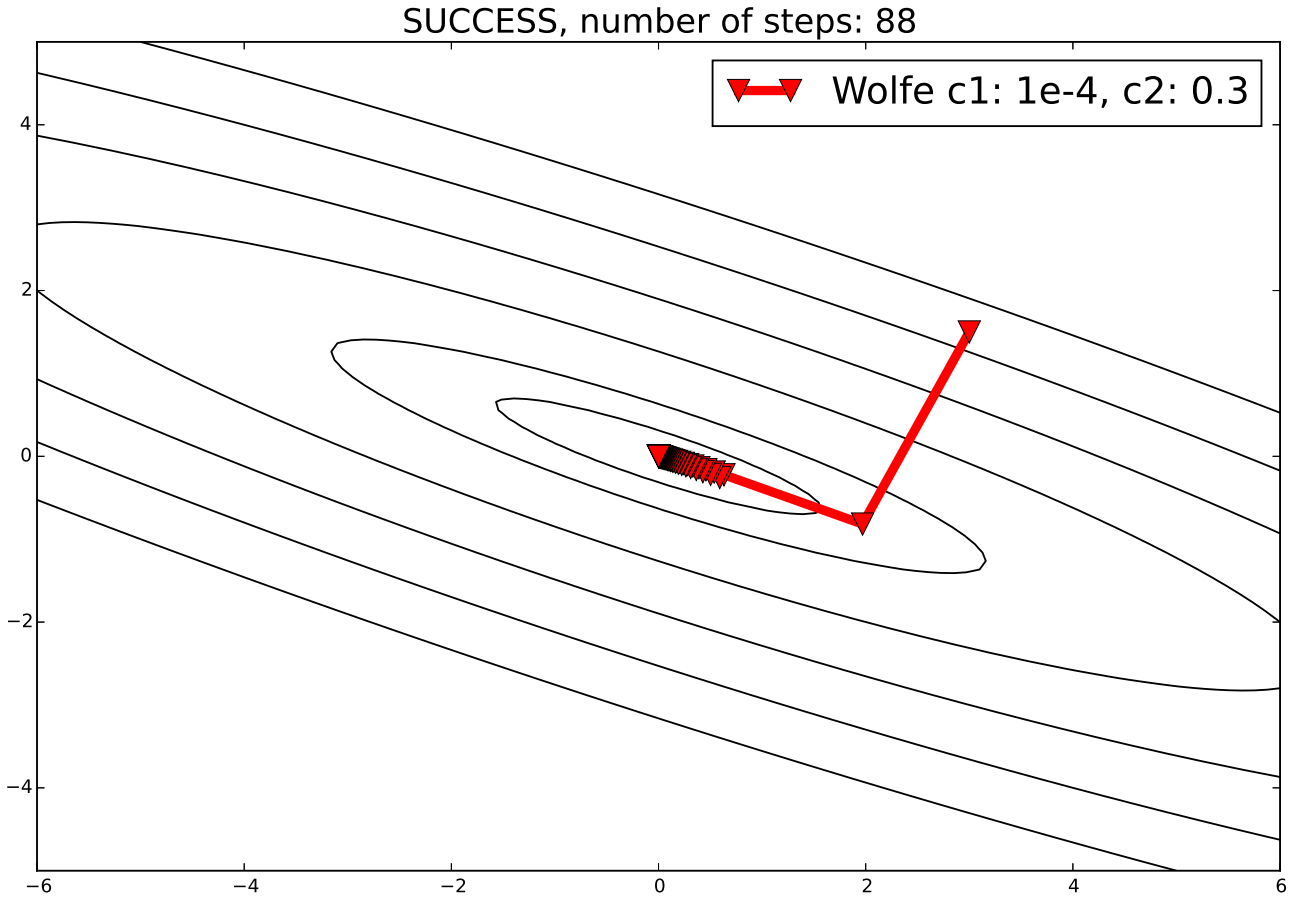
Градиентный спуск: метод дробления шага

Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle, \quad A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}.$



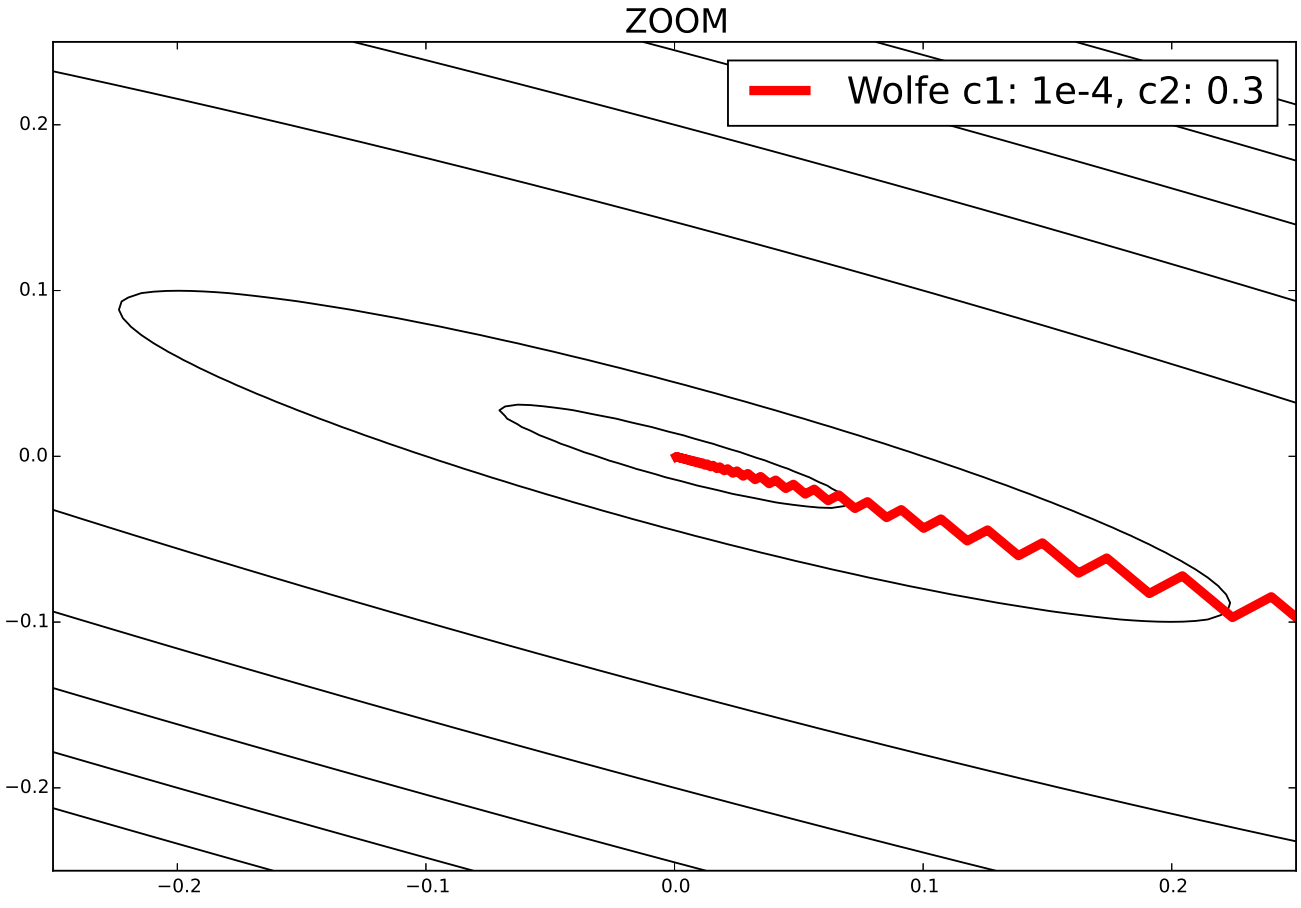
Градиентный спуск: стратегия Вульфа

Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle, \quad A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}.$



Градиентный спуск: стратегия Вульфа

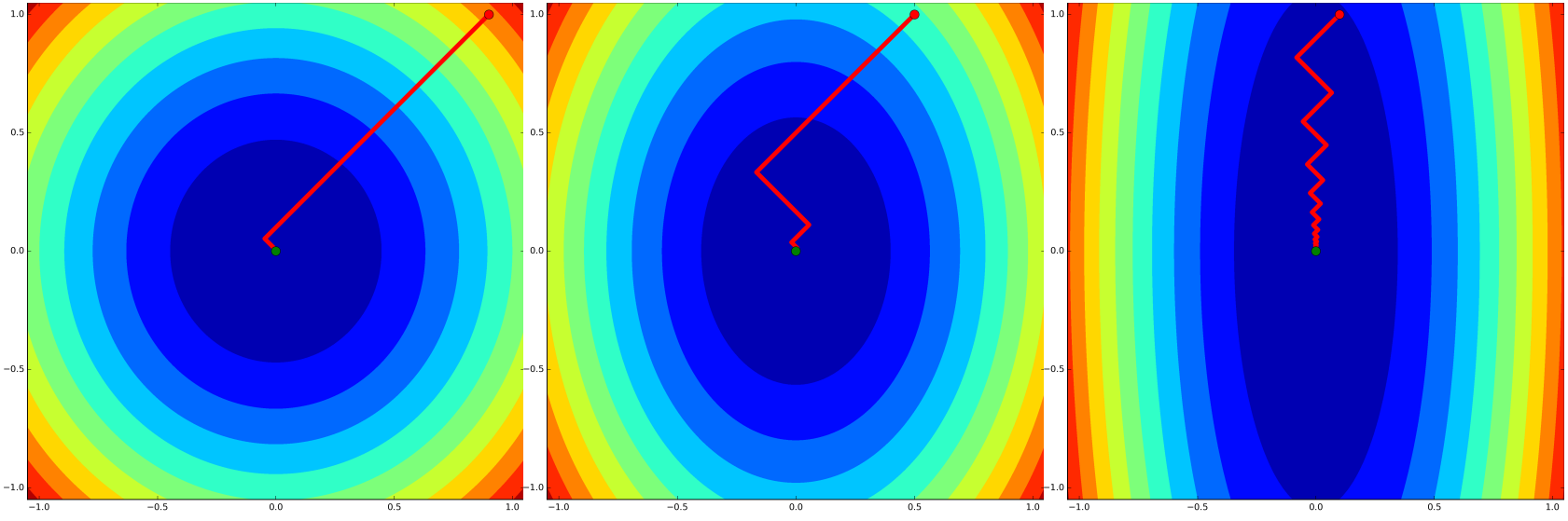
Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle, \quad A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}$.



Градиентный спуск: типичная траектория

Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$, где $A \in \mathbb{S}^n_{++}$, $b \in \mathbb{R}^n$.

Число обусловленности: $\kappa = \dfrac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \geq 1$.

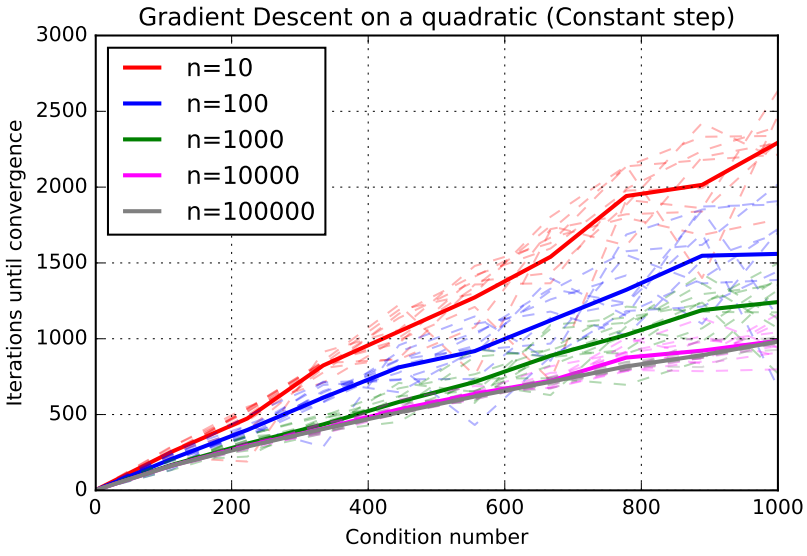


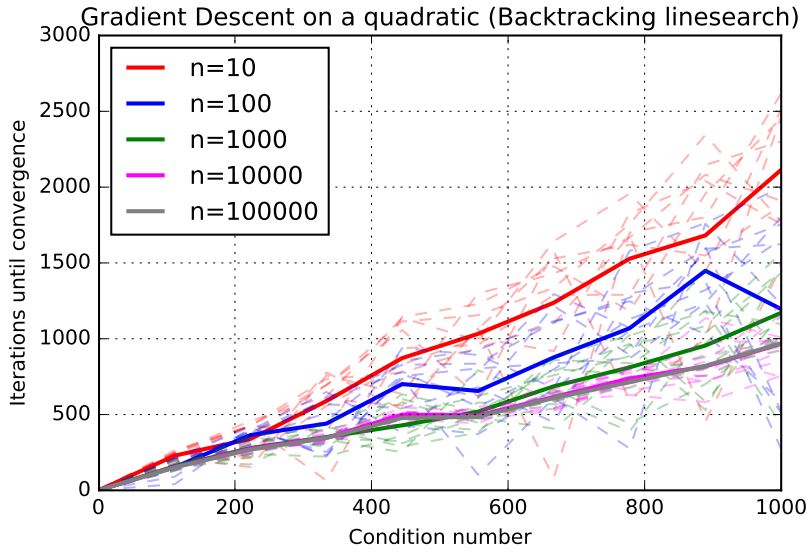
Плохая обусловленность + неудачный старт = **зигзаг**

Зависимость от обусловленности и размерности задачи

Квадратичная функция: $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$.

$$A = \mathrm{Diag}(a), \qquad b \sim \mathcal{N}(0, I_n), \qquad a_i = \begin{cases} 1, & i = 1 \\ \sim \mathrm{Unif}(1, \kappa), & 2 \le i \le n-1 \\ \kappa, & i = n \end{cases}$$



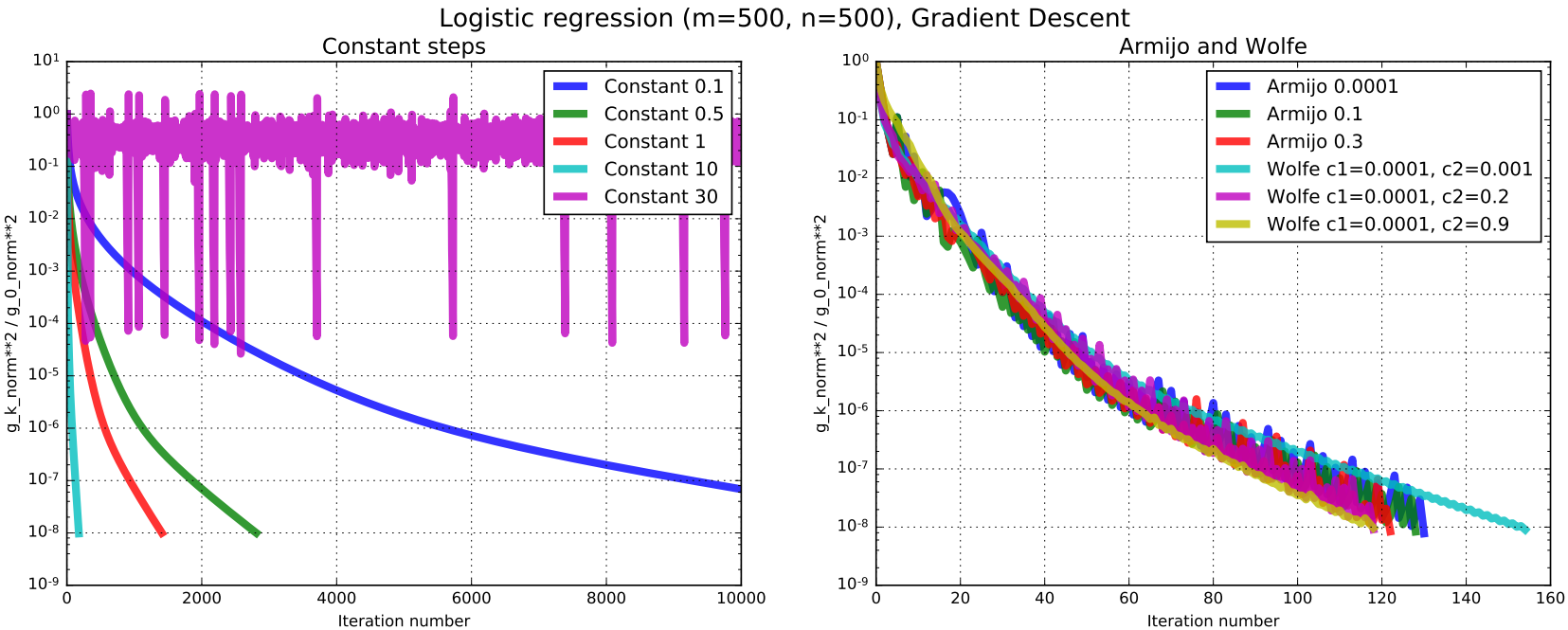


- Линейная зависимость от числа обусловленности.
- С ростом размерности число итераций не увеличивается!

Стратегии выбора длины шага в градиентном спуске

Логистическая регрессия с l^2-регуляризатором:

$$f(x) := \frac{1}{m}\sum_{i=1}^{m} \ln(1 + \exp(-b_i\langle a_i, x\rangle)) + \frac{\lambda}{2}\|x\|_2^2 \quad \to \quad \min_{x\in\mathbb{R}^n}.$$



- Сильная чувствительность к значению постоянного шага.
- Адаптивные стратегии сами подбирают «хорошую» длину шага.
- Разница между адаптивными стратегиями незначительная.

Учет структуры функции

Пусть $\boxed{f(x) := \psi(Ax)}$, где $\psi \in C^1(\mathbb{R}^m \to \mathbb{R})$, $A \in \mathbb{R}^{m \times n}$:

- Сложность вычисления ψ в одной точке: $O(m)$.

Примеры:

- Квадратичная функция:

$$f(x) := \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle, \qquad \psi(y) := \frac{1}{2}\langle y, x\rangle - \langle b, x\rangle$$

- Логистическая регрессия:

$$f(x) := \sum_{i=1}^{m} \ln(1 + \exp(-b_i\langle a_i, x\rangle)),$$

$$\psi(y) := \sum_{i=1}^{m} \ln(1 + \exp(-b_i y_i))$$

Учет структуры функции – 2

Функция: $\boxed{f(x) := \psi(Ax)}$, где $\psi \in C^1(\mathbb{R}^m \to \mathbb{R})$, $A \in \mathbb{R}^{m \times n}$.

Линейный поиск: $\boxed{\phi_k(\alpha) := f(x_k + \alpha d_k)}$.

Если считать **«в лоб»**:

- Сложность вычисления для одного α: mn.
- Сложность вычисления для s разных α: smn.

Учтем структуру функции: $\boxed{\phi_k(\alpha) = \psi(Ax_k + \alpha A d_k)}$.

- Предподсчет: один раз вычислим и сохраним Ax_k и Ad_k. Сложность: $2mn$.
- Сложность дальнейшего вычисл. ϕ_k для одного α: $O(m)$.
- Суммарная сложность для s разных α: $2mn$.
- Эффективно, когда $s \geq 2$.

Полностью аналогично для производной $\phi_k'(\alpha) = \langle \nabla \psi(Ax_k + \alpha A d_k), Ad_k \rangle$.

Детали реализации

Итерация метода спуска для $f(x) = \psi(Ax)$:

1. Вызвать оракул в точке x_k:
   (a) $f(x_k) = \psi(Ax_k)$, $\nabla f(x_k) = A^T \nabla \psi(Ax_k)$ и пр.
2. Вычислить d_k (оракул не вызывается).
3. Линейный поиск:
   (a) $\phi(0) = \psi(Ax_k)$, $\phi'(0) = \langle \nabla \psi(Ax_k), Ad_k \rangle$
   (b) $\phi(\bar{\alpha}_1) = \psi(Ax_k + \bar{\alpha}_1 Ad_k)$,
   $\phi'(\bar{\alpha}_1) = \langle \nabla \psi(Ax_k + \bar{\alpha}_1 Ad_k), Ad_k \rangle$
   (c) $\ldots$
   (d) $\phi(\bar{\alpha}_s) = \psi(Ax_k + \bar{\alpha}_s Ad_k)$,
   $\phi'(\bar{\alpha}_s) = \langle \nabla \psi(Ax_k + \bar{\alpha}_s Ad_k), Ad_k \rangle$
4. $x_{k+1} \leftarrow x_k + \bar{\alpha}_s d_k \qquad (Ax_{k+1} = Ax_k + \bar{\alpha}_s Ad_k)$

- Последовательные вызовы используют много одинаковой информации: Ax_k, Ad_k. Имеет смысл запоминать эти величины.

Детали реализации – 2. Пример

$$f(x) := \frac{1}{3}\|Ax\|_2^3, \qquad \nabla f(x) = \|Ax\|_2 A^T Ax.$$

```
class CubicOptimizedOracle(BaseSmoothOracle):
  def __init__(self, A):
    self.A, self.last_x, self.last_x_trial = A, None, None
    self.norm_Ax, self.norm_Ax_trial = None, None

  def func(self, x):
    self._update_Ax(x); return (1/3) * self.norm_Ax**3

  def grad(self, x):
    self._update_Ax(x); return self.norm_Ax**3 * self.A.T.dot(self.Ax)

  def func_directional(self, x, d, alpha):
    self._update_Ax(x); self._update_Ad(d)
    self.last_x_trial = x + alpha * d
    self.Ax_trial = self.Ax + alpha * self.Ad
    return (1/3) * np.linalg.norm(self.Ax_trial)**3

  def _update_Ax(self, x):
    if np.array_equal(x, self.last_x): return
    if np.array_equal(x, self.last_x_trial):
      self.last_x = self.last_x_trial
      self.Ax, self.norm_Ax = self.Ax_trial, self.norm_Ax_trial
      return
    self.last_x = np.copy(x)
    self.Ax = self.A.dot(x)
    self.norm_Ax = np.linalg.norm(self.Ax)
```

Метод Ньютона

Рассматриваемая задача: $\min_{x \in \mathbb{R}^n} f(x)$, где $f \in C^2(\mathbb{R}^n \to \mathbb{R})$.

Чистый метод Ньютона:

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

- **Интерпретация:** Минимизация квадратичной модели:

$$f(x_k + h) \approx f(x_k) + \langle \nabla f(x_k), h \rangle + \frac{1}{2} \langle \nabla^2 f(x_k) h, h \rangle.$$

- **Скорость сходимости:** квадратичная (в невырожденном случае).
- Для плохих начальных приближений x_0 не гарантируется даже сходимости.

Демпфированный метод Ньютона

Демпфированный метод Ньютона:

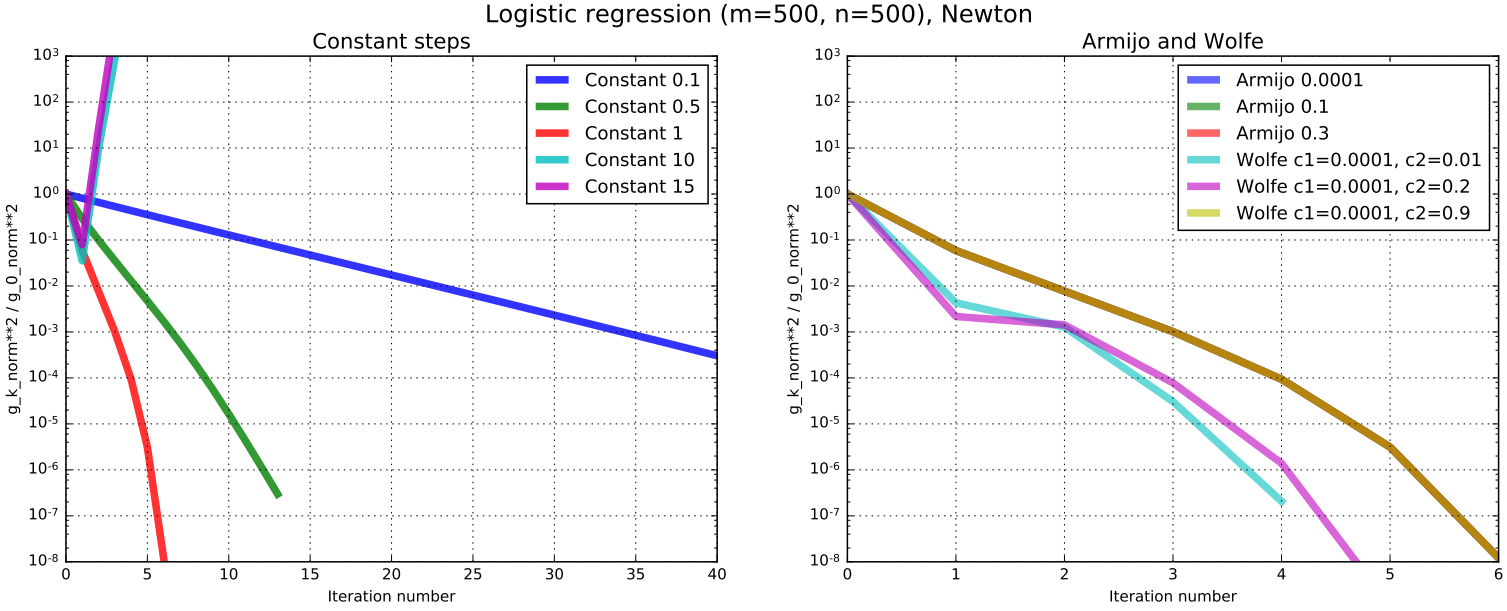
$$x_{k+1} = x_k - \alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k).$$

- Добавляется «демпфирующая» длина шага $\alpha_k \in [0, 1]$.
- **Интерпретация:** Метод спуска с $d_k := -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$:
  - Если $\nabla^2 f(x_k) \succ 0$, то
    $$\langle \nabla f(x_k), d_k \rangle = -\langle [\nabla^2 f(x_k)]^{-1} \nabla f(x_k), \nabla f(x_k) \rangle < 0.$$
  - Если $\nabla^2 f(x_k) \not\succ 0$, то применяют модификацию гессиана.
- Длина шага α_k настраивается с помощью линейного поиска.
- Важный момент: линейный поиск всегда нужно начинать с $\alpha_k^{(0)} = 1$. Иначе не будет квадратичной сходимости.

Сравнение стратегий выбора шага в методе Ньютона

Логистическая регрессия с l^2-регуляризатором:

$$f(x) := \frac{1}{m}\sum_{i=1}^{m} \ln(1 + \exp(-b_i\langle a_i, x\rangle)) + \frac{\lambda}{2}\|x\|_2^2 \quad \to \quad \min_{x\in\mathbb{R}^n}.$$

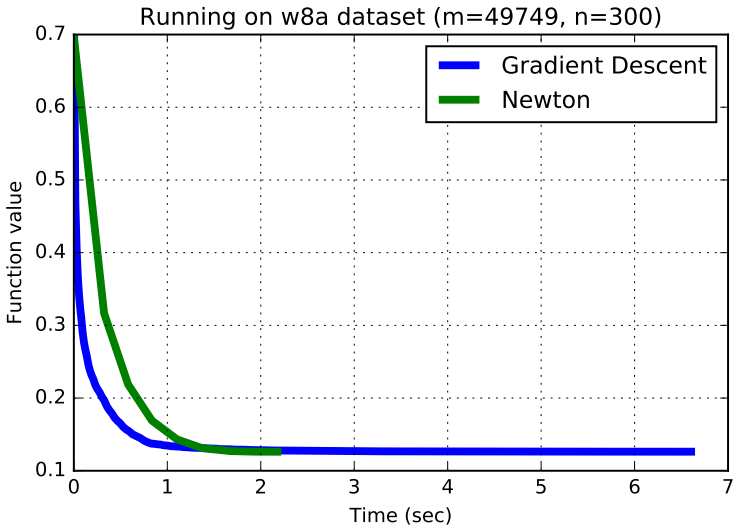


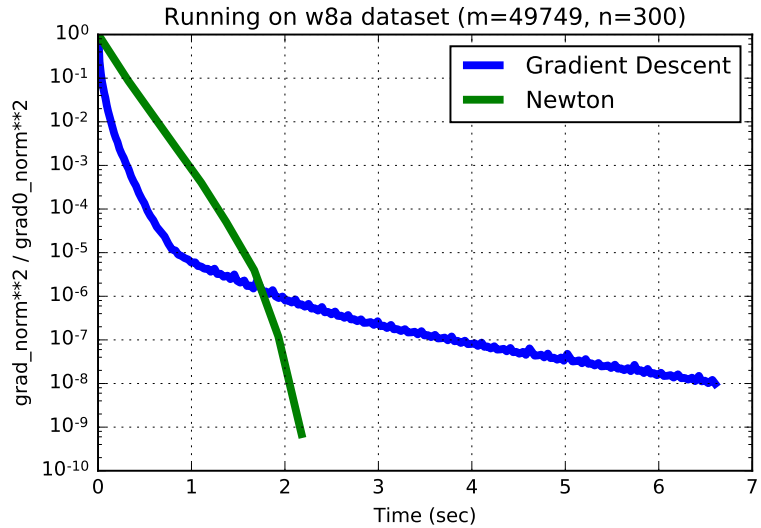
- Постоянный шаг:
  - При $\alpha > 1$ наблюдается расходимость.
  - При $\alpha < 1$ сходимость может быть линейной.
  - Наилучший выбор: $\alpha = 1$ (квадратичная сходимость).
- Адаптивные стратегии сами выбирают «хорошую» длину шага.
- Разница между адаптивными стратегиями незначительная.

Сравнение градиентного спуска и Ньютона

Логистическая регрессия с l^2-регуляризатором:

$$f(x) := \frac{1}{m}\sum_{i=1}^{m} \ln(1 + \exp(-b_i\langle a_i, x\rangle)) + \frac{\lambda}{2}\|x\|_2^2 \quad \to \quad \min_{x\in\mathbb{R}^n}.$$



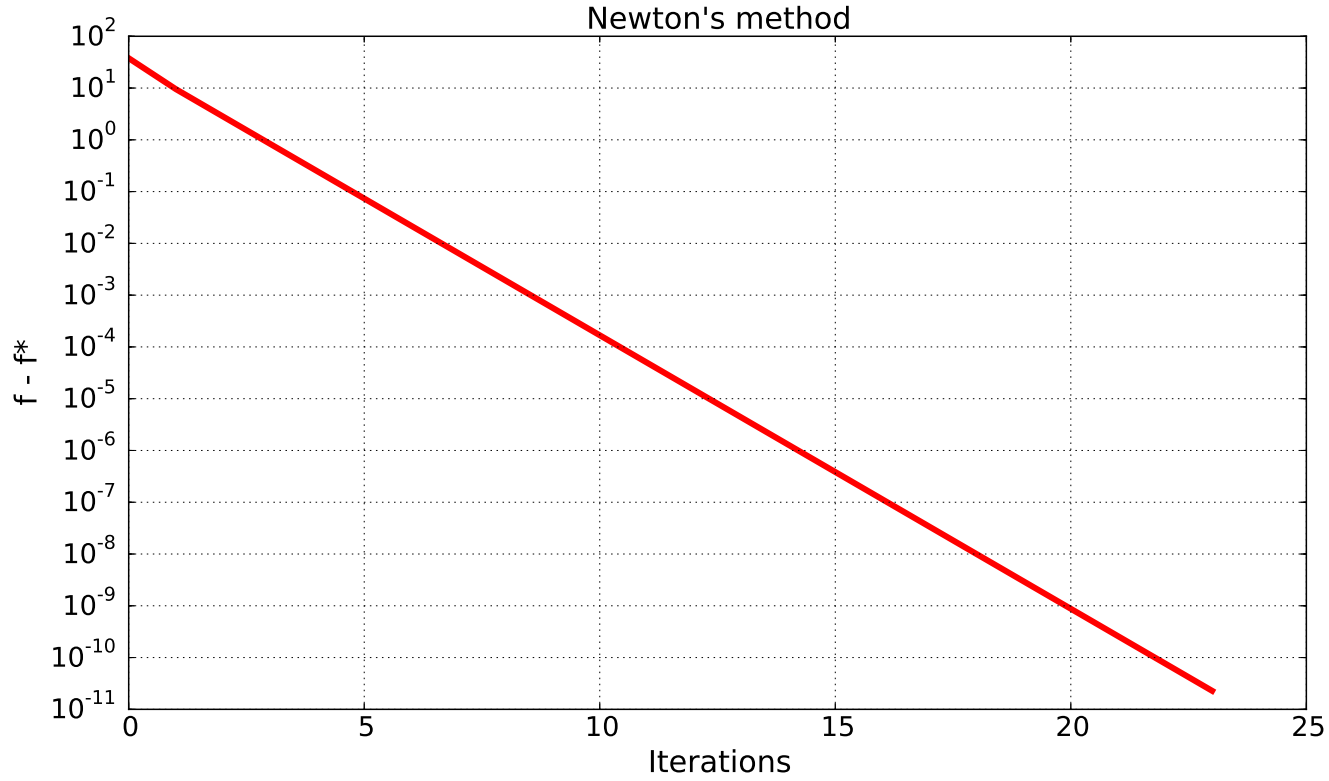


- Типичная ситуация: более простой метод быстрее работает в начале (для небольшой точности), но медленнее в конце (для большой точности).

Линейная сходимость метода Ньютона

Функция: $f(x) := \|x\|_2^3$.

Чистый метод Ньютона:



Почему нет квадратичной сходимости?

Модификация гессиана

Модификация Левенберга–Марквардта:

1. Начать с $\tau := 0$.
2. Пока $\nabla^2 f(x_k) + \tau I_n \not\succ 0$:
   (a) $\tau \leftarrow \max\{1, 2\tau\}$.

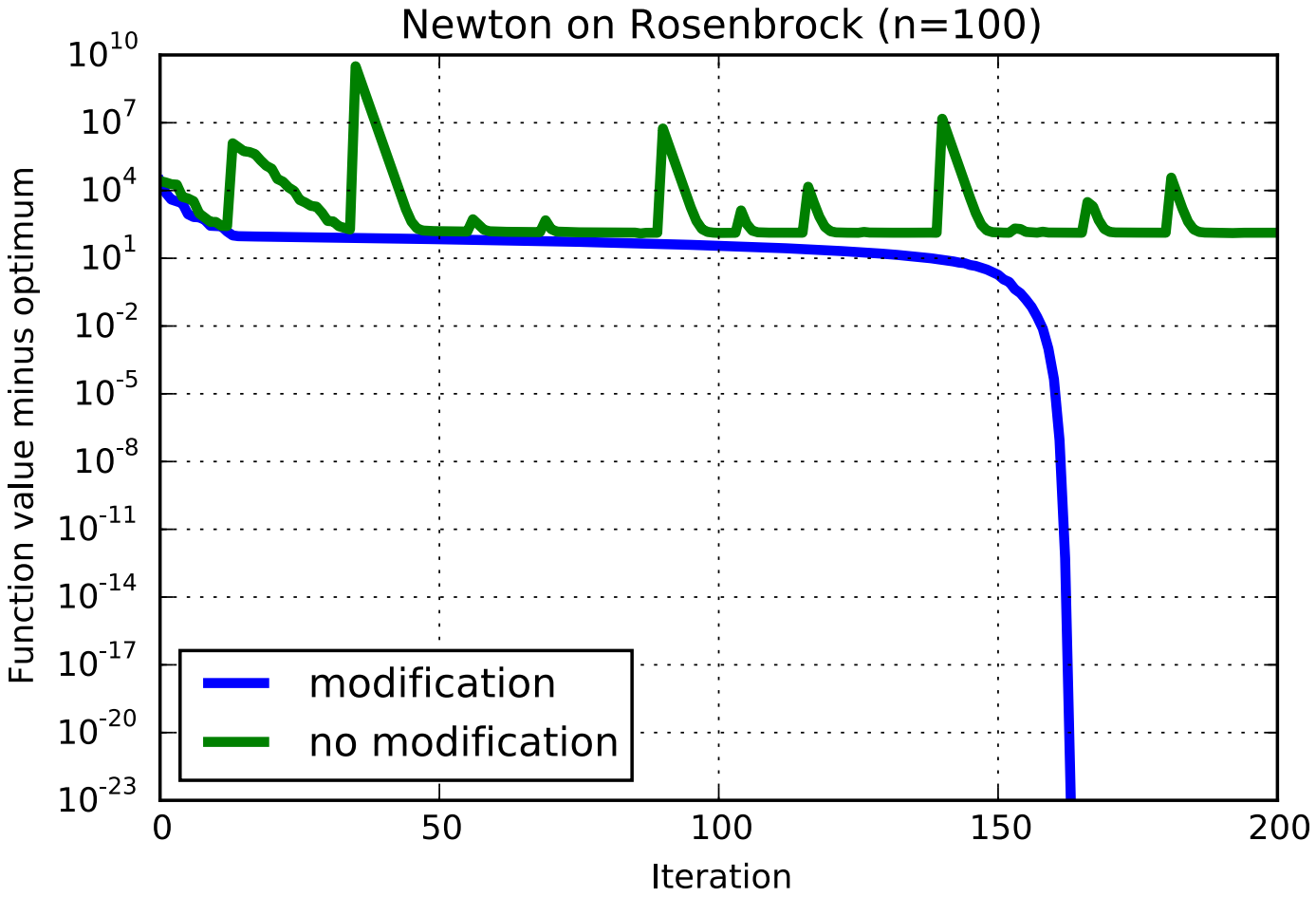
- Условие $\nabla^2 f(x_k) + \tau I_n \succ 0$ проверяется с помощью **алгоритма Холецкого** (сложность: $n^3/3$).
- В случае успешного завершения получаем факторизацию
$$\nabla^2 f(x_k) + \tau I_n = LL^T,$$
где $L \in \mathbb{R}^{n \times n}$ — нижнетреугольная матрица.
- Дальнейшее вычисление $d_k := -[\nabla^2 f(x_k) + \tau I_n]^{-1} \nabla f(x_k)$ имеет сложность $O(n^2)$.

Сложность = число запусков алгоритма Холецкого.

Модификация Левенберга–Маркварта: пример

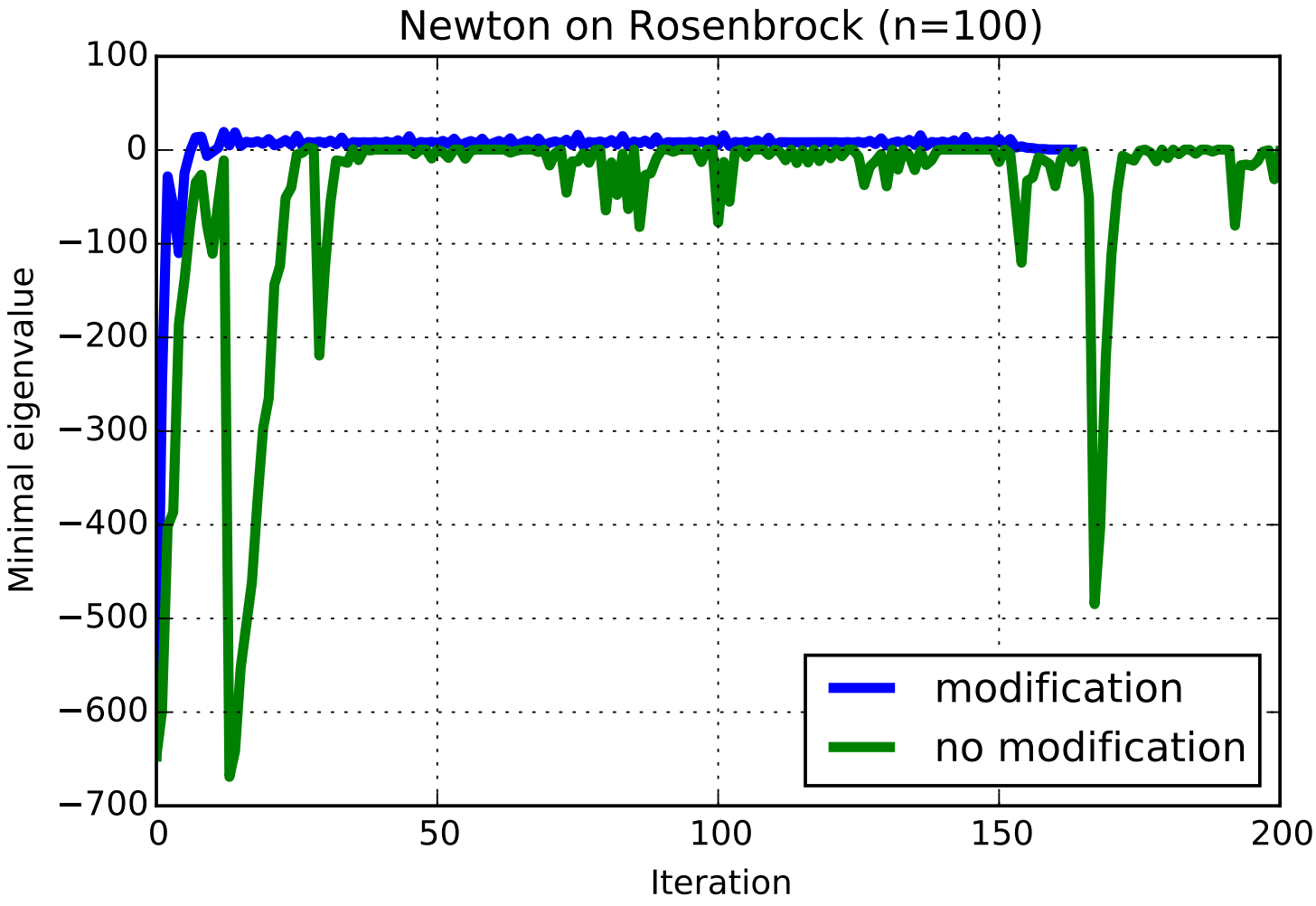
Многомерная функция Розенброка:

$$f(x) := \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$



Модификация Левенберга–Маркварта: пример – 2

Собственные значения $\nabla^2 f(x_k)$ в итерациях:



Модификация Левенберга–Маркварта: пример – 3

Число запусков алгоритма Холецкого в итерациях: