



**Lower bounds for gradient descent.  
Accelerated gradient descent. Momentum.  
Nesterov's acceleration**

**Daniil Merkulov**

Optimization methods. MIPT

# Recap of Gradient Descent convergence

Gradient Descent:

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\ \nabla f(x^k)\ ^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\ x^k - x^*\ ^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$ $k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

# Recap of Gradient Descent convergence

Gradient Descent:

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\ \nabla f(x^k)\ ^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$	$\ x^k - x^*\ ^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any  $x$ , since  $e^{-x}$  is convex and  $1 - x$  is its tangent line at  $x = 0$ , we have:

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

# Recap of Gradient Descent convergence

Gradient Descent:

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\ \nabla f(x^k)\ ^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\ x^k - x^*\ ^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$ $k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any  $x$ , since  $e^{-x}$  is convex and  $1 - x$  is its tangent line at  $x = 0$ , we have:

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

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

## Recap of Gradient Descent convergence

Gradient Descent:

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\ \nabla f(x^k)\ ^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$	$\ x^k - x^*\ ^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any  $x$ , since  $e^{-x}$  is convex and  $1 - x$  is its tangent line at  $x = 0$ , we have:

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

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

**Question:** Can we do faster, than this using the first-order information?

## Recap of Gradient Descent convergence

Gradient Descent:

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\ \nabla f(x^k)\ ^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$	$\ x^k - x^*\ ^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any  $x$ , since  $e^{-x}$  is convex and  $1 - x$  is its tangent line at  $x = 0$ , we have:

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

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

**Question:** Can we do faster, than this using the first-order information? **Yes, we can.**

## Lower bounds

## Lower bounds

convex (non-smooth)	smooth (non-convex) <sup>1</sup>	smooth & convex <sup>2</sup>	smooth & strongly convex (or PL)
$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{k^2}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)$	$\mathcal{O}\left(\frac{1}{k^2}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)$	$\mathcal{O}\left(\left(1 - \sqrt{\frac{\mu}{L}}\right)^k\right)$ $k_\varepsilon \sim \mathcal{O}\left(\sqrt{\kappa} \log \frac{1}{\varepsilon}\right)$

<sup>1</sup>Carmon, Duchi, Hinder, Sidford, 2017

<sup>2</sup>Nemirovski, Yudin, 1979



# Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

# Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

Consider a family of first-order methods, where

$$\begin{aligned}x^{k+1} &\in x^0 + \text{span} \{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \} && f - \text{smooth} \\x^{k+1} &\in x^0 + \text{span} \{ g_0, g_1, \dots, g_k \}, \text{ where } g_i \in \partial f(x^i) && f - \text{non-smooth}\end{aligned} \tag{1}$$

# Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

Consider a family of first-order methods, where

$$\begin{aligned}x^{k+1} &\in x^0 + \text{span} \{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \} && f - \text{smooth} \\x^{k+1} &\in x^0 + \text{span} \{ g_0, g_1, \dots, g_k \}, \text{ where } g_i \in \partial f(x^i) && f - \text{non-smooth}\end{aligned} \tag{1}$$

In order to construct a lower bound, we need to find a function  $f$  from corresponding class such that any method from the family 1 will work at least as slow as the lower bound.

## Smooth case

### Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

## Smooth case

### Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .

## Smooth case

### Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .
- The key to the proof is to explicitly build a special function  $f$ .

## Smooth case

### Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .
- The key to the proof is to explicitly build a special function  $f$ .
- Note, that this bound  $\mathcal{O}\left(\frac{1}{k^2}\right)$  does not match the rate of gradient descent  $\mathcal{O}\left(\frac{1}{k}\right)$ . Two options possible:

## Smooth case

### Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .
- The key to the proof is to explicitly build a special function  $f$ .
- Note, that this bound  $\mathcal{O}\left(\frac{1}{k^2}\right)$  does not match the rate of gradient descent  $\mathcal{O}\left(\frac{1}{k}\right)$ . Two options possible:
  - a. The lower bound is not tight.



## Smooth case

### i Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .
- The key to the proof is to explicitly build a special function  $f$ .
- Note, that this bound  $\mathcal{O}\left(\frac{1}{k^2}\right)$  does not match the rate of gradient descent  $\mathcal{O}\left(\frac{1}{k}\right)$ . Two options possible:
  - a. The lower bound is not tight.
  - b. The gradient method is not optimal for this problem.

## Smooth case

### i Theorem

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function  $f$  that, when you apply your gradient method on minimizing such  $f$ , the convergence rate is lower bounded as  $\mathcal{O}\left(\frac{1}{k^2}\right)$ .
- The key to the proof is to explicitly build a special function  $f$ .
- Note, that this bound  $\mathcal{O}\left(\frac{1}{k^2}\right)$  does not match the rate of gradient descent  $\mathcal{O}\left(\frac{1}{k}\right)$ . Two options possible:
  - a. The lower bound is not tight.
  - b. **The gradient method is not optimal for this problem.**

## Nesterov's worst function

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

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

## Nesterov's worst function

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

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

- Notice, that

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

Therefore,  $x^T A x \geq 0$ . It is also easy to see that  $0 \preceq A \preceq 4I$ .

## Nesterov's worst function

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

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

- Notice, that

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

Therefore,  $x^T A x \geq 0$ . It is also easy to see that  $0 \preceq A \preceq 4I$ .

## Nesterov's worst function

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

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

- Notice, that

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

Therefore,  $x^T A x \geq 0$ . It is also easy to see that  $0 \preceq A \preceq 4I$ .

Example, when  $n = 3$ :

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

## Nesterov's worst function

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

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

- Notice, that

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

Therefore,  $x^T A x \geq 0$ . It is also easy to see that  $0 \preceq A \preceq 4I$ .

Example, when  $n = 3$ :

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

Lower bound:

$$\begin{aligned} x^T A x &= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1x_2 - 2x_2x_3 \\ &= x_1^2 + x_1^2 - 2x_1x_2 + x_2^2 + x_2^2 - 2x_2x_3 + x_3^2 + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 \geq 0 \end{aligned}$$

## Nesterov's worst function

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

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

- Notice, that

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

Therefore,  $x^T A x \geq 0$ . It is also easy to see that  $0 \preceq A \preceq 4I$ .

Example, when  $n = 3$ :

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

Lower bound:

$$\begin{aligned} x^T A x &= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1x_2 - 2x_2x_3 \\ &= x_1^2 + x_1^2 - 2x_1x_2 + x_2^2 + x_2^2 - 2x_2x_3 + x_3^2 + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 \geq 0 \end{aligned}$$

Upper bound

$$\begin{aligned} x^T A x &= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1x_2 - 2x_2x_3 \\ &\leq 4(x_1^2 + x_2^2 + x_3^2) \\ 0 &\leq 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 \\ 0 &\leq x_1^2 + x_1^2 + 2x_1x_2 + x_2^2 + x_2^2 + 2x_2x_3 + x_3^2 + x_3^2 \\ 0 &\leq x_1^2 + (x_1 + x_2)^2 + (x_2 + x_3)^2 + x_3^2 \end{aligned}$$



## Nesterov's worst function

- Define the following  $L$ -smooth convex function:  $f(x) = \frac{L}{4} \left( \frac{1}{2} x^T A x - e_1^T x \right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$ .

## Nesterov's worst function

- Define the following  $L$ -smooth convex function:  $f(x) = \frac{L}{4} \left( \frac{1}{2} x^T A x - e_1^T x \right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$ .
- The optimal solution  $x^*$  satisfies  $Ax^* = e_1$ , and solving this system of equations gives:

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

## Nesterov's worst function

- Define the following  $L$ -smooth convex function:  $f(x) = \frac{L}{4} \left( \frac{1}{2} x^T A x - e_1^T x \right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$ .
- The optimal solution  $x^*$  satisfies  $Ax^* = e_1$ , and solving this system of equations gives:

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

- The hypothesis:  $x_i^* = a + bi$  (inspired by physics). Check, that the second equation is satisfied, while  $a$  and  $b$  are computed from the first and the last equations.

## Nesterov's worst function

- Define the following  $L$ -smooth convex function:  $f(x) = \frac{L}{4} \left( \frac{1}{2} x^T A x - e_1^T x \right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$ .
- The optimal solution  $x^*$  satisfies  $Ax^* = e_1$ , and solving this system of equations gives:

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

- The hypothesis:  $x_i^* = a + bi$  (inspired by physics). Check, that the second equation is satisfied, while  $a$  and  $b$  are computed from the first and the last equations.
- The solution is:

$$x_i^* = 1 - \frac{i}{n+1},$$

## Nesterov's worst function

- Define the following  $L$ -smooth convex function:  $f(x) = \frac{L}{4} \left( \frac{1}{2} x^T A x - e_1^T x \right) = \frac{L}{8} x^T A x - \frac{L}{4} e_1^T x$ .
- The optimal solution  $x^*$  satisfies  $Ax^* = e_1$ , and solving this system of equations gives:

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

- The hypothesis:  $x_i^* = a + bi$  (inspired by physics). Check, that the second equation is satisfied, while  $a$  and  $b$  are computed from the first and the last equations.
- The solution is:

$$x_i^* = 1 - \frac{i}{n+1},$$

- And the objective value is

$$f(x^*) = \frac{L}{8} x^{*T} A x^* - \frac{L}{4} \langle x^*, e_1 \rangle = -\frac{L}{8} \langle x^*, e_1 \rangle = -\frac{L}{8} \left( 1 - \frac{1}{n+1} \right).$$

## Smooth case (proof)

- Suppose, we start from  $x^0 = 0$ . Asking the oracle for the gradient, we get  $g_0 = -e_1$ . Then,  $x^1$  must lie on the line generated by  $e_1$ . At this point all the components of  $x^1$  are zero except the first one, so

$$x^1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

## Smooth case (proof)

- Suppose, we start from  $x^0 = 0$ . Asking the oracle for the gradient, we get  $g_0 = -e_1$ . Then,  $x^1$  must lie on the line generated by  $e_1$ . At this point all the components of  $x^1$  are zero except the first one, so

$$x^1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- At the second iteration we ask the oracle again and get  $g_1 = Ax^1 - e_1$ . Then,  $x^2$  must lie on the line generated by  $e_1$  and  $Ax^1 - e_1$ . All the components of  $x^2$  are zero except the first two, so

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow x^2 = \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

## Smooth case (proof)

- Suppose, we start from  $x^0 = 0$ . Asking the oracle for the gradient, we get  $g_0 = -e_1$ . Then,  $x^1$  must lie on the line generated by  $e_1$ . At this point all the components of  $x^1$  are zero except the first one, so

$$x^1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- At the second iteration we ask the oracle again and get  $g_1 = Ax^1 - e_1$ . Then,  $x^2$  must lie on the line generated by  $e_1$  and  $Ax^1 - e_1$ . All the components of  $x^2$  are zero except the first two, so

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow x^2 = \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$



## Smooth case (proof)

- Suppose, we start from  $x^0 = 0$ . Asking the oracle for the gradient, we get  $g_0 = -e_1$ . Then,  $x^1$  must lie on the line generated by  $e_1$ . At this point all the components of  $x^1$  are zero except the first one, so

$$x^1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- At the second iteration we ask the oracle again and get  $g_1 = Ax^1 - e_1$ . Then,  $x^2$  must lie on the line generated by  $e_1$  and  $Ax^1 - e_1$ . All the components of  $x^2$  are zero except the first two, so

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow x^2 = \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- Due to the structure of the matrix  $A$  one can show using induction that after  $k$  iterations we have all the last  $n - k$  components of  $x^k$  to be zero.

$$x^{(k)} = \begin{bmatrix} \bullet \\ \bullet \\ \vdots \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ k \\ k+1 \\ \vdots \\ n \end{matrix}$$

## Smooth case (proof)

- Suppose, we start from  $x^0 = 0$ . Asking the oracle for the gradient, we get  $g_0 = -e_1$ . Then,  $x^1$  must lie on the line generated by  $e_1$ . At this point all the components of  $x^1$  are zero except the first one, so

$$x^1 = \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- At the second iteration we ask the oracle again and get  $g_1 = Ax^1 - e_1$ . Then,  $x^2$  must lie on the line generated by  $e_1$  and  $Ax^1 - e_1$ . All the components of  $x^2$  are zero except the first two, so

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{bmatrix} \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow x^2 = \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- Due to the structure of the matrix  $A$  one can show using induction that after  $k$  iterations we have all the last  $n - k$  components of  $x^k$  to be zero.

$$x^{(k)} = \begin{bmatrix} \bullet \\ \bullet \\ \vdots \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ k \\ k+1 \\ \vdots \\ n \end{matrix}$$

- However, since every iterate  $x^k$  produced by our method lies in  $S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  (i.e. has zeros in the coordinates  $k+1, \dots, n$ ), it cannot “reach” the full optimal vector  $x^*$ . In other words, even if one were to choose the best possible vector from  $S_k$ , denoted by

$$\tilde{x}^k = \arg \min_{x \in S_k} f(x),$$

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

- Similarly, to the optimum of the original function, we have  $\tilde{x}_i^k = 1 - \frac{i}{k+1}$  and  $f(\tilde{x}^k) = -\frac{L}{8} \left(1 - \frac{1}{k+1}\right)$ .

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

- Similarly, to the optimum of the original function, we have  $\tilde{x}_i^k = 1 - \frac{i}{k+1}$  and  $f(\tilde{x}^k) = -\frac{L}{8} \left(1 - \frac{1}{k+1}\right)$ .
- We now have:

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*)$$

(2)

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

- Similarly, to the optimum of the original function, we have  $\tilde{x}_i^k = 1 - \frac{i}{k+1}$  and  $f(\tilde{x}^k) = -\frac{L}{8} \left(1 - \frac{1}{k+1}\right)$ .
- We now have:

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

(2)

## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

- Similarly, to the optimum of the original function, we have  $\tilde{x}_i^k = 1 - \frac{i}{k+1}$  and  $f(\tilde{x}^k) = -\frac{L}{8} \left(1 - \frac{1}{k+1}\right)$ .
- We now have:

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



## Smooth case (proof)

- Because  $x^k \in S_k = \text{span}\{e_1, e_2, \dots, e_k\}$  and  $\tilde{x}^k$  is the best possible approximation to  $x^*$  within  $S_k$ , we have

$$f(x^k) \geq f(\tilde{x}^k).$$

- Thus, the optimality gap obeys

$$f(x^k) - f(x^*) \geq f(\tilde{x}^k) - f(x^*).$$

- Similarly, to the optimum of the original function, we have  $\tilde{x}_i^k = 1 - \frac{i}{k+1}$  and  $f(\tilde{x}^k) = -\frac{L}{8} \left(1 - \frac{1}{k+1}\right)$ .
- We now have:

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

## Smooth case (proof)

- Now we bound  $R = \|x^0 - x^*\|_2$ :

$$\|x^0 - x^*\|_2^2 = \|0 - x^*\|_2^2 = \|x^*\|_2^2 = \sum_{i=1}^n \left(1 - \frac{i}{n+1}\right)^2$$

We observe, that

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\leq \frac{(n+1)^3}{3}$$

## Smooth case (proof)

- Now we bound  $R = \|x^0 - x^*\|_2$ :

$$\begin{aligned}\|x^0 - x^*\|_2^2 &= \|0 - x^*\|_2^2 = \|x^*\|_2^2 = \sum_{i=1}^n \left(1 - \frac{i}{n+1}\right)^2 \\ &= n - \frac{2}{n+1} \sum_{i=1}^n i + \frac{1}{(n+1)^2} \sum_{i=1}^n i^2\end{aligned}$$

We observe, that

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

## Smooth case (proof)

- Now we bound  $R = \|x^0 - x^*\|_2$ :

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

We observe, that

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

## Smooth case (proof)

- Now we bound  $R = \|x^0 - x^*\|_2$ :

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

We observe, that

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

## Smooth case (proof)

- Now we bound  $R = \|x^0 - x^*\|_2$ :

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

- Thus,

$$k+1 \geq \frac{3}{2} \|x^0 - x^*\|_2^2 = \frac{3}{2} R^2 \quad (3)$$

We observe, that

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

## Smooth case (proof)

Finally, using (2) and (3), we get:

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

## Smooth case (proof)

Finally, using (2) and (3), we get:

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

Which concludes the proof with the desired  $\mathcal{O}\left(\frac{1}{k^2}\right)$  rate.



# Smooth case lower bound theorems

## i Smooth convex case

There exists a function  $f$  that is  $L$ -smooth and convex such that any method 1 satisfies for any  $k : 1 \leq k \leq \frac{n-1}{2}$ :

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

## i Smooth strongly convex case

For any  $x^0$  and any  $\mu > 0$ ,  $\kappa = \frac{L}{\mu} > 1$ , there exists a function  $f$  that is  $L$ -smooth and  $\mu$ -strongly convex such that for any method of the form 1 holds:

$$\begin{aligned}\|x^k - x^*\|_2^2 &\geq \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2k} \|x^0 - x^*\|_2^2 \\ f(x^k) - f^* &\geq \frac{\mu}{2} \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2k} \|x^0 - x^*\|_2^2\end{aligned}$$

## Acceleration for quadratics

# Convergence result for quadratics

Suppose, we have a strongly convex quadratic function minimization problem solved by the gradient descent method:

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

## Theorem

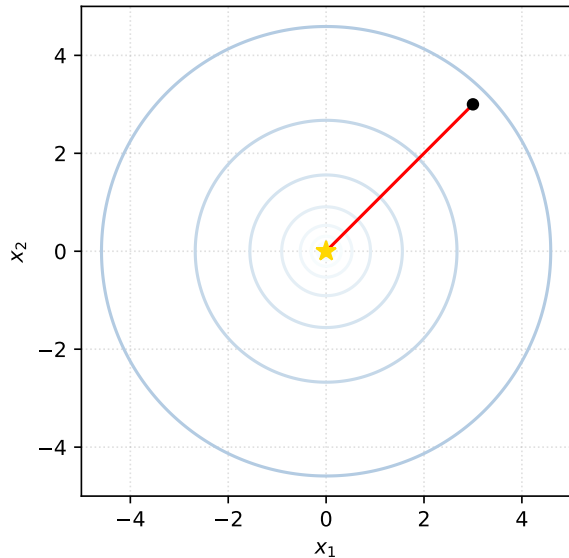
The gradient descent method with the learning rate  $\alpha_k = \frac{2}{\mu+L}$  converges to the optimal solution  $x^*$  with the following guarantee:

$$\|x^{k+1} - x^*\|_2 = \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x^0 - x^*\|_2 \quad f(x^{k+1}) - f(x^*) = \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2k} (f(x^0) - f(x^*))$$

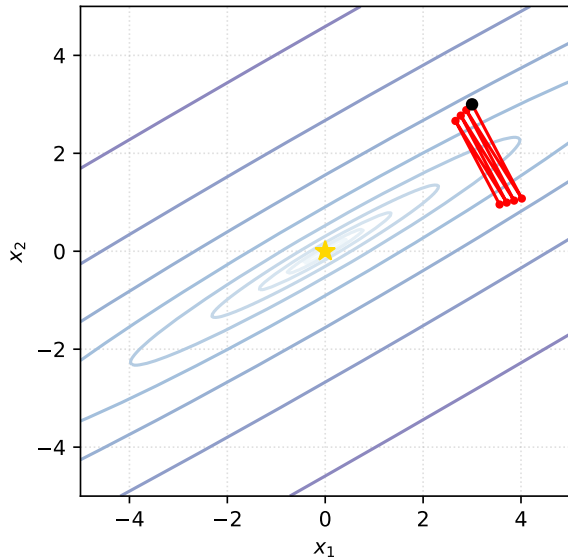
where  $\kappa = \frac{L}{\mu}$  is the condition number of  $A$ .

## Condition number $\kappa$

$\kappa = 1.0$



$\kappa = 100.0$



# Convergence from the first principles

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

Let  $x^*$  be the unique solution of the linear system  $Ax = b$  and put  $e_k = \|x_k - x^*\|$ , where  $x_{k+1} = x_k - \alpha_k(Ax_k - b)$  is defined recursively starting from some  $x_0$ , and  $\alpha_k$  is a step size we'll determine shortly.

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

## Polynomials

The above calculation gives us  $e_k = p_k(A)e_0$ , where  $p_k$  is the polynomial

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

We can upper bound the norm of the error term as

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

# Convergence from the first principles

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

Let  $x^*$  be the unique solution of the linear system  $Ax = b$  and put  $e_k = \|x_k - x^*\|$ , where  $x_{k+1} = x_k - \alpha_k(Ax_k - b)$  is defined recursively starting from some  $x_0$ , and  $\alpha_k$  is a step size we'll determine shortly.

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

## Polynomials

The above calculation gives us  $e_k = p_k(A)e_0$ , where  $p_k$  is the polynomial

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

We can upper bound the norm of the error term as

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

Since  $A$  is a symmetric matrix with eigenvalues in  $[\mu, L]$ ,

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

This leads to an interesting problem: Among all polynomials that satisfy  $p_k(0) = 1$  we're looking for a polynomial whose magnitude is as small as possible in the interval  $[\mu, L]$ .

## Naive polynomial solution

A naive solution is to choose a uniform step size

$\alpha_k = \frac{2}{\mu+L}$  in the expression. This choice makes  $|p_k(\mu)| = |p_k(L)|$ .

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

This is exactly the rate we proved in the previous lecture for any smooth and strongly convex function.

Let's look at this polynomial a bit closer. On the right figure we choose  $\alpha = 1$  and  $\beta = 10$  so that  $\kappa = 10$ . The relevant interval is therefore  $[1, 10]$ .

Can we do better? The answer is yes.



## Naive polynomial solution

A naive solution is to choose a uniform step size

$\alpha_k = \frac{2}{\mu+L}$  in the expression. This choice makes  $|p_k(\mu)| = |p_k(L)|$ .

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

This is exactly the rate we proved in the previous lecture for any smooth and strongly convex function.

Let's look at this polynomial a bit closer. On the right figure we choose  $\alpha = 1$  and  $\beta = 10$  so that  $\kappa = 10$ . The relevant interval is therefore  $[1, 10]$ .

Can we do better? The answer is yes.





## Naive polynomial solution

A naive solution is to choose a uniform step size  $\alpha_k = \frac{2}{\mu+L}$  in the expression. This choice makes  $|p_k(\mu)| = |p_k(L)|$ .

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

This is exactly the rate we proved in the previous lecture for any smooth and strongly convex function.

Let's look at this polynomial a bit closer. On the right figure we choose  $\alpha = 1$  and  $\beta = 10$  so that  $\kappa = 10$ . The relevant interval is therefore  $[1, 10]$ .

Can we do better? The answer is yes.



## Naive polynomial solution

A naive solution is to choose a uniform step size

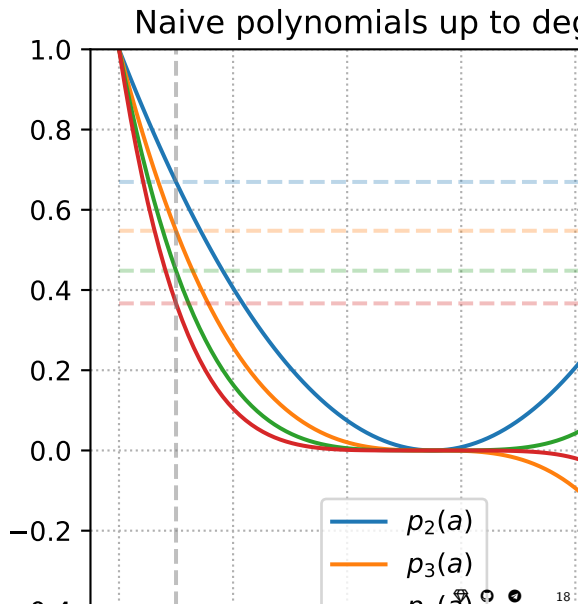
$\alpha_k = \frac{2}{\mu+L}$  in the expression. This choice makes  $|p_k(\mu)| = |p_k(L)|$ .

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

This is exactly the rate we proved in the previous lecture for any smooth and strongly convex function.

Let's look at this polynomial a bit closer. On the right figure we choose  $\alpha = 1$  and  $\beta = 10$  so that  $\kappa = 10$ . The relevant interval is therefore  $[1, 10]$ .

Can we do better? The answer is yes.



## Naive polynomial solution

A naive solution is to choose a uniform step size

$\alpha_k = \frac{2}{\mu+L}$  in the expression. This choice makes  $|p_k(\mu)| = |p_k(L)|$ .

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

This is exactly the rate we proved in the previous lecture for any smooth and strongly convex function.

Let's look at this polynomial a bit closer. On the right figure we choose  $\alpha = 1$  and  $\beta = 10$  so that  $\kappa = 10$ . The relevant interval is therefore  $[1, 10]$ .

Can we do better? The answer is yes.



## Chebyshev polynomials

Chebyshev polynomials turn out to give an optimal answer to the question that we asked. Suitably rescaled, they minimize the absolute value in a desired interval  $[\mu, L]$  while satisfying the normalization constraint of having value 1 at the origin.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2.$$

Let's plot the standard Chebyshev polynomials (without rescaling):



## Chebyshev polynomials

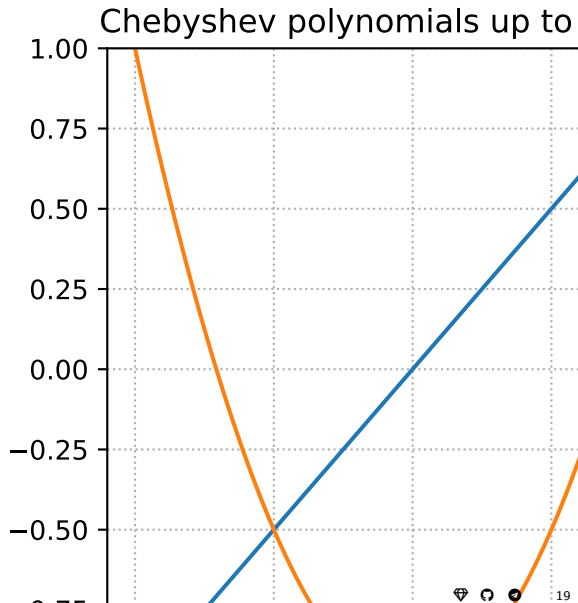
Chebyshev polynomials turn out to give an optimal answer to the question that we asked. Suitably rescaled, they minimize the absolute value in a desired interval  $[\mu, L]$  while satisfying the normalization constraint of having value 1 at the origin.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2.$$

Let's plot the standard Chebyshev polynomials (without rescaling):



## Chebyshev polynomials

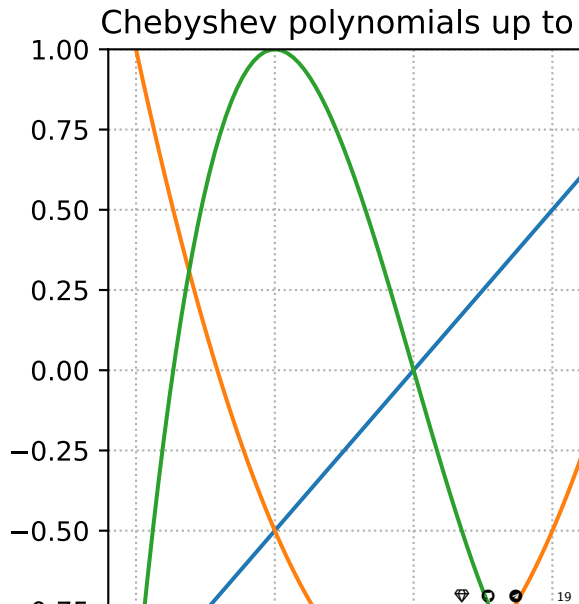
Chebyshev polynomials turn out to give an optimal answer to the question that we asked. Suitably rescaled, they minimize the absolute value in a desired interval  $[\mu, L]$  while satisfying the normalization constraint of having value 1 at the origin.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2.$$

Let's plot the standard Chebyshev polynomials (without rescaling):



## Chebyshev polynomials

Chebyshev polynomials turn out to give an optimal answer to the question that we asked. Suitably rescaled, they minimize the absolute value in a desired interval  $[\mu, L]$  while satisfying the normalization constraint of having value 1 at the origin.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2.$$

Let's plot the standard Chebyshev polynomials (without rescaling):



## Chebyshev polynomials

Chebyshev polynomials turn out to give an optimal answer to the question that we asked. Suitably rescaled, they minimize the absolute value in a desired interval  $[\mu, L]$  while satisfying the normalization constraint of having value 1 at the origin.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2.$$

Let's plot the standard Chebyshev polynomials (without rescaling):





## Rescaled Chebyshev polynomials

Original Chebyshev polynomials are defined on the interval  $[-1, 1]$ . To use them for our purposes, we need to rescale them to the interval  $[\mu, L]$ .

## Rescaled Chebyshev polynomials

Original Chebyshev polynomials are defined on the interval  $[-1, 1]$ . To use them for our purposes, we need to rescale them to the interval  $[\mu, L]$ .

We will use the following affine transformation:

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

Note, that  $x = 1$  corresponds to  $a = \mu$ ,  $x = -1$  corresponds to  $a = L$  and  $x = 0$  corresponds to  $a = \frac{\mu+L}{2}$ . This transformation ensures that the behavior of the Chebyshev polynomial on  $[-1, 1]$  is reflected on the interval  $[\mu, L]$

## Rescaled Chebyshev polynomials

Original Chebyshev polynomials are defined on the interval  $[-1, 1]$ . To use them for our purposes, we need to rescale them to the interval  $[\mu, L]$ .

We will use the following affine transformation:

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

Note, that  $x = 1$  corresponds to  $a = \mu$ ,  $x = -1$  corresponds to  $a = L$  and  $x = 0$  corresponds to  $a = \frac{\mu+L}{2}$ . This transformation ensures that the behavior of the Chebyshev polynomial on  $[-1, 1]$  is reflected on the interval  $[\mu, L]$

In our error analysis, we require that the polynomial equals 1 at 0 (i.e.,  $p_k(0) = 1$ ). After applying the transformation, the value  $T_k$  takes at the point corresponding to  $a = 0$  might not be 1. Thus, we multiply by the inverse of  $T_k$  evaluated at

$$\frac{L + \mu}{L - \mu}, \quad \text{ensuring that} \quad P_k(0) = T_k\left(\frac{L + \mu - 0}{L - \mu}\right) \cdot T_k\left(\frac{L + \mu}{L - \mu}\right)^{-1} = 1.$$

## Rescaled Chebyshev polynomials

Original Chebyshev polynomials are defined on the interval  $[-1, 1]$ . To use them for our purposes, we need to rescale them to the interval  $[\mu, L]$ .

We will use the following affine transformation:

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

Note, that  $x = 1$  corresponds to  $a = \mu$ ,  $x = -1$  corresponds to  $a = L$  and  $x = 0$  corresponds to  $a = \frac{\mu+L}{2}$ . This transformation ensures that the behavior of the Chebyshev polynomial on  $[-1, 1]$  is reflected on the interval  $[\mu, L]$

In our error analysis, we require that the polynomial equals 1 at 0 (i.e.,  $p_k(0) = 1$ ). After applying the transformation, the value  $T_k$  takes at the point corresponding to  $a = 0$  might not be 1. Thus, we multiply by the inverse of  $T_k$  evaluated at

$$\frac{L + \mu}{L - \mu}, \quad \text{ensuring that} \quad P_k(0) = T_k\left(\frac{L + \mu - 0}{L - \mu}\right) \cdot T_k\left(\frac{L + \mu}{L - \mu}\right)^{-1} = 1.$$

Let's plot the rescaled Chebyshev polynomials

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

and observe, that they are much better behaved than the naive polynomials in terms of the magnitude in the interval  $[\mu, L]$ .

## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials





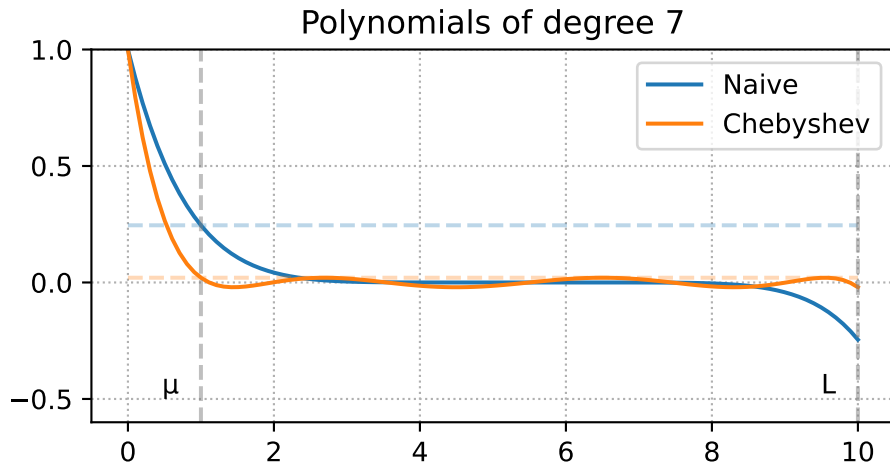
## Rescaled Chebyshev polynomials



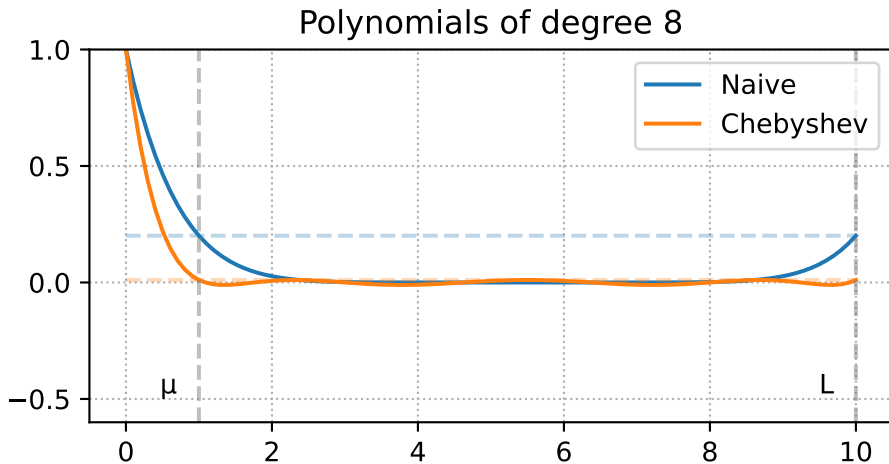
## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Rescaled Chebyshev polynomials



## Chebyshev polynomials upper bound

We can see, that the maximum value of the Chebyshev polynomial on the interval  $[\mu, L]$  is achieved at the point  $a = \mu$ . Therefore, we can use the following upper bound:

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

## Chebyshev polynomials upper bound

We can see, that the maximum value of the Chebyshev polynomial on the interval  $[\mu, L]$  is achieved at the point  $a = \mu$ . Therefore, we can use the following upper bound:

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

Using the definition of condition number  $\kappa = \frac{L}{\mu}$ , we get:

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



## Chebyshev polynomials upper bound

We can see, that the maximum value of the Chebyshev polynomial on the interval  $[\mu, L]$  is achieved at the point  $a = \mu$ . Therefore, we can use the following upper bound:

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

Using the definition of condition number  $\kappa = \frac{L}{\mu}$ , we get:

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

Therefore, we only need to understand the value of  $T_k$  at  $1 + \epsilon$ . This is where the acceleration comes from. We will bound this value with  $\mathcal{O}\left(\frac{1}{\sqrt{\epsilon}}\right)$ .

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

1. For any  $x \geq 1$ , the Chebyshev polynomial of the first kind can be written as

$$T_k(x) = \cosh(k \operatorname{arccosh}(x))$$

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

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

1. For any  $x \geq 1$ , the Chebyshev polynomial of the first kind can be written as

$$T_k(x) = \cosh(k \operatorname{arccosh}(x))$$

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

2. Recall that:

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

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

1. For any  $x \geq 1$ , the Chebyshev polynomial of the first kind can be written as

$$T_k(x) = \cosh(k \operatorname{arccosh}(x))$$

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

2. Recall that:

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

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

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

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

1. For any  $x \geq 1$ , the Chebyshev polynomial of the first kind can be written as
4. Therefore,

$$T_k(x) = \cosh(k \operatorname{arccosh}(x))$$

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

2. Recall that:

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

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

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

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

## Chebyshev polynomials upper bound

To upper bound  $|P_k|$ , we need to lower bound  $|T_k(1 + \epsilon)|$ .

1. For any  $x \geq 1$ , the Chebyshev polynomial of the first kind can be written as
4. Therefore,

$$T_k(x) = \cosh(k \operatorname{arccosh}(x))$$

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

2. Recall that:

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

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

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

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

5. Finally, we get:

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

## Accelerated method [1/2]

Due to the recursive definition of the Chebyshev polynomials, we directly obtain an iterative acceleration scheme. Reformulating the recurrence in terms of our rescaled Chebyshev polynomials, we obtain:

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

Given the fact, that  $x = \frac{L+\mu-2a}{L-\mu}$ , and:

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

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



## Accelerated method [1/2]

Due to the recursive definition of the Chebyshev polynomials, we directly obtain an iterative acceleration scheme. Reformulating the recurrence in terms of our rescaled Chebyshev polynomials, we obtain:

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

Given the fact, that  $x = \frac{L+\mu-2a}{L-\mu}$ , and:

$$\begin{aligned} P_k(a) &= T_k\left(\frac{L+\mu-2a}{L-\mu}\right) T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1} & T_{k-1}\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_{k-1}(a) T_{k-1}\left(\frac{L+\mu}{L-\mu}\right) \\ T_k\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_k(a) T_k\left(\frac{L+\mu}{L-\mu}\right) & T_{k+1}\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_{k+1}(a) T_{k+1}\left(\frac{L+\mu}{L-\mu}\right) \end{aligned}$$

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

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

## Accelerated method [1/2]

Due to the recursive definition of the Chebyshev polynomials, we directly obtain an iterative acceleration scheme. Reformulating the recurrence in terms of our rescaled Chebyshev polynomials, we obtain:

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

Given the fact, that  $x = \frac{L+\mu-2a}{L-\mu}$ , and:

$$\begin{aligned} P_k(a) &= T_k\left(\frac{L+\mu-2a}{L-\mu}\right) T_k\left(\frac{L+\mu}{L-\mu}\right)^{-1} & T_{k-1}\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_{k-1}(a) T_{k-1}\left(\frac{L+\mu}{L-\mu}\right) \\ T_k\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_k(a) T_k\left(\frac{L+\mu}{L-\mu}\right) & T_{k+1}\left(\frac{L+\mu-2a}{L-\mu}\right) &= P_{k+1}(a) T_{k+1}\left(\frac{L+\mu}{L-\mu}\right) \end{aligned}$$

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

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

Since we have  $P_{k+1}(0) = P_k(0) = P_{k-1}(0) = 1$ , we can find the method in the following form:

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

## Accelerated method [2/2]

Rearranging the terms, we get:

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

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

## Accelerated method [2/2]

Rearranging the terms, we get:

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

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

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

## Accelerated method [2/2]

Rearranging the terms, we get:

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

We are almost done :) We remember, that  $e_{k+1} = P_{k+1}(A)e_0$ . Note also, that we work with the quadratic problem, so we can assume  $x^* = 0$  without loss of generality. In this case,  $e_0 = x_0$  and  $e_{k+1} = x_{k+1}$ .

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

## Accelerated method [2/2]

Rearranging the terms, we get:

$$\begin{aligned} P_{k+1}(a) &= (1 + \beta_k)P_k(a) - \alpha_k a P_k(a) - \beta_k P_{k-1}(a), \\ P_{k+1}(a) &= 2 \frac{L + \mu}{L - \mu} \frac{t_k}{t_{k+1}} P_k(a) - \frac{4a}{L - \mu} \frac{t_k}{t_{k+1}} P_k(a) - \frac{t_{k-1}}{t_{k+1}} P_{k-1}(a) \end{aligned} \quad \left\{ \begin{aligned} \beta_k &= \frac{t_{k-1}}{t_{k+1}}, \\ \alpha_k &= \frac{4}{L - \mu} \frac{t_k}{t_{k+1}}, \\ 1 + \beta_k &= 2 \frac{L + \mu}{L - \mu} \frac{t_k}{t_{k+1}} \end{aligned} \right.$$

We are almost done :) We remember, that  $e_{k+1} = P_{k+1}(A)e_0$ . Note also, that we work with the quadratic problem, so we can assume  $x^* = 0$  without loss of generality. In this case,  $e_0 = x_0$  and  $e_{k+1} = x_{k+1}$ .

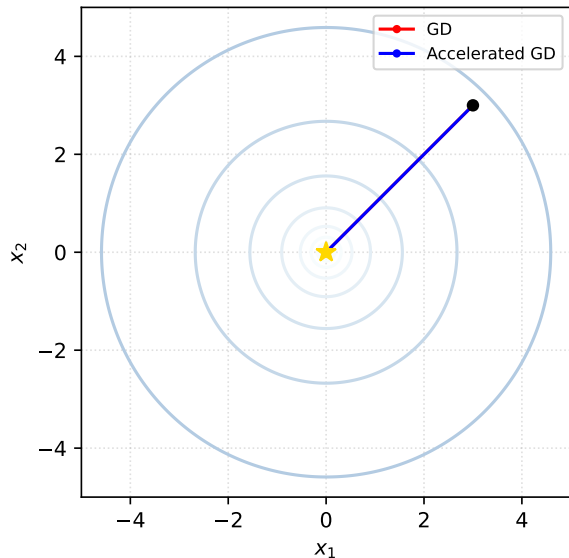
$$\begin{aligned} x_{k+1} &= P_{k+1}(A)x_0 = (I - \alpha_k A)P_k(A)x_0 + \beta_k (P_k(A) - P_{k-1}(A))x_0 \\ &= (I - \alpha_k A)x_k + \beta_k (x_k - x_{k-1}) \end{aligned}$$

For quadratic problem, we have  $\nabla f(x_k) = Ax_k$ , so we can rewrite the update as:

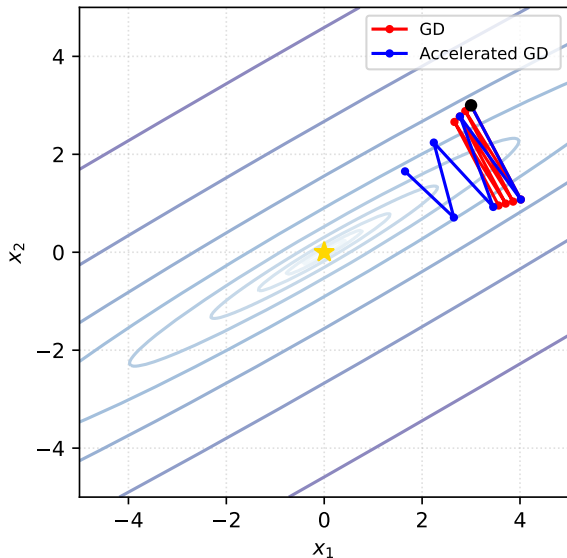
$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1})$$

# Acceleration from the first principles

$\kappa = 1.0$



$\kappa = 100.0$



## Heavy ball



# Oscillations and acceleration

## Gradient Descent



## Heavy Ball



# Polyak Heavy ball method

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

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



# Polyak Heavy ball method

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

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

Which is in our (quadratics) case is

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



# Polyak Heavy ball method

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

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

Which is in our (quadratics) case is

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

This can be rewritten as follows

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

$$\hat{x}_k = \hat{x}_k.$$



# Polyak Heavy ball method



Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

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

Which is in our (quadratics) case is

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

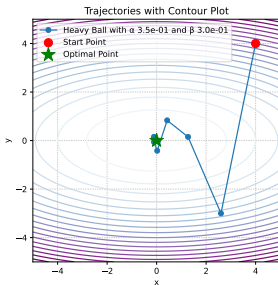
This can be rewritten as follows

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

$$\hat{x}_k = \hat{x}_k.$$

Let's use the following notation  $\hat{z}_k = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_k \end{bmatrix}$ . Therefore  $\hat{z}_{k+1} = M \hat{z}_k$ , where the iteration matrix  $M$  is:

# Polyak Heavy ball method



Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

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

Which is in our (quadratics) case is

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

This can be rewritten as follows

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

$$\hat{x}_k = \hat{x}_k.$$

Let's use the following notation  $\hat{z}_k = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_k \end{bmatrix}$ . Therefore  $\hat{z}_{k+1} = M \hat{z}_k$ , where the iteration matrix  $M$  is:

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

## Reduction to a scalar case

Note, that  $M$  is  $2d \times 2d$  matrix with 4 block-diagonal matrices of size  $d \times d$  inside. It means, that we can rearrange the order of coordinates to make  $M$  block-diagonal in the following form. Note that in the equation below, the matrix  $M$  denotes the same as in the notation above, except for the described permutation of rows and columns. We use this slight abuse of notation for the sake of clarity.

## Reduction to a scalar case

Note, that  $M$  is  $2d \times 2d$  matrix with 4 block-diagonal matrices of size  $d \times d$  inside. It means, that we can rearrange the order of coordinates to make  $M$  block-diagonal in the following form. Note that in the equation below, the matrix  $M$  denotes the same as in the notation above, except for the described permutation of rows and columns. We use this slight abuse of notation for the sake of clarity.



Figure 1: Illustration of matrix  $M$  rearrangement

where  $\hat{x}_k^{(i)}$  is  $i$ -th coordinate of vector  $\hat{x}_k \in \mathbb{R}^d$  and  $M_i$  stands for  $2 \times 2$  matrix. This rearrangement allows us to study the dynamics of the method independently for each dimension. One may observe, that the asymptotic convergence rate of the  $2d$ -dimensional vector sequence of  $\hat{z}_k$  is defined by the worst convergence rate among its block of coordinates. Thus, it is enough to study the optimization in a one-dimensional case.



## Reduction to a scalar case

For  $i$ -th coordinate with  $\lambda_i$  as an  $i$ -th eigenvalue of matrix  $W$  we have:

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

## Reduction to a scalar case

For  $i$ -th coordinate with  $\lambda_i$  as an  $i$ -th eigenvalue of matrix  $W$  we have:

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

The method will be convergent if  $\rho(M) < 1$ , and the optimal parameters can be computed by optimizing the spectral radius

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

## Reduction to a scalar case

For  $i$ -th coordinate with  $\lambda_i$  as an  $i$ -th eigenvalue of matrix  $W$  we have:

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

The method will be convergent if  $\rho(M) < 1$ , and the optimal parameters can be computed by optimizing the spectral radius

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

It can be shown, that for such parameters the matrix  $M$  has complex eigenvalues, which forms a conjugate pair, so the distance to the optimum (in this case,  $\|z_k\|$ ), generally, will not go to zero monotonically.

# Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of  $M_i$ :

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

## Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of  $M_i$ :

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

When  $\alpha$  and  $\beta$  are optimal  $(\alpha^*, \beta^*)$ , the eigenvalues are complex-conjugated pair  $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$ , i.e.  $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$ .

## Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of  $M_i$ :

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

When  $\alpha$  and  $\beta$  are optimal  $(\alpha^*, \beta^*)$ , the eigenvalues are complex-conjugated pair  $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$ , i.e.  $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$ .

$$\operatorname{Re}(\lambda_1^M) = \frac{L + \mu - 2\lambda_i}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \operatorname{Im}(\lambda_1^M) = \frac{\pm 2\sqrt{(L - \lambda_i)(\lambda_i - \mu)}}{(\sqrt{L} + \sqrt{\mu})^2}; \quad |\lambda_1^M| = \frac{L - \mu}{(\sqrt{L} + \sqrt{\mu})^2}.$$

## Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of  $M_i$ :

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

When  $\alpha$  and  $\beta$  are optimal  $(\alpha^*, \beta^*)$ , the eigenvalues are complex-conjugated pair  $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$ , i.e.  $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$ .

$$\operatorname{Re}(\lambda_1^M) = \frac{L + \mu - 2\lambda_i}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \operatorname{Im}(\lambda_1^M) = \frac{\pm 2\sqrt{(L - \lambda_i)(\lambda_i - \mu)}}{(\sqrt{L} + \sqrt{\mu})^2}; \quad |\lambda_1^M| = \frac{L - \mu}{(\sqrt{L} + \sqrt{\mu})^2}.$$

And the convergence rate does not depend on the stepsize and equals to  $\sqrt{\beta^*}$ .

# Heavy Ball quadratics convergence

## **i** Theorem

Assume that  $f$  is quadratic  $\mu$ -strongly convex  $L$ -smooth quadratics, then Heavy Ball method with parameters

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

converges linearly:

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



# Heavy Ball Global Convergence<sup>3</sup>

## i Theorem

Assume that  $f$  is smooth and convex and that

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

Then, the sequence  $\{x_k\}$  generated by Heavy-ball iteration satisfies

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

where  $\bar{x}_T$  is the Cesaro average of the iterates, i.e.,

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

<sup>3</sup>Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

# Heavy Ball Global Convergence <sup>4</sup>

## i Theorem

Assume that  $f$  is smooth and strongly convex and that

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

where  $\alpha_0 \in (0, 1/L]$ . Then, the sequence  $\{x_k\}$  generated by Heavy-ball iteration converges linearly to a unique optimizer  $x^*$ . In particular,

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

where  $q \in [0, 1)$ .

<sup>4</sup>Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

# Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems

# Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.

# Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.

# Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.
- Method was not extremely popular until the ML boom

# Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.
- Method was not extremely popular until the ML boom
- Nowadays, it is de-facto standard for practical acceleration of gradient methods, even for the non-convex problems (neural network training)

## Nesterov accelerated gradient



# The concept of Nesterov Accelerated Gradient method

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

# The concept of Nesterov Accelerated Gradient method

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

Let's define the following notation

$$x^+ = x - \alpha \nabla f(x) \quad \text{Gradient step}$$

$$d_k = \beta_k(x_k - x_{k-1}) \quad \text{Momentum term}$$

Then we can write down:

$$x_{k+1} = x_k^+ \quad \text{Gradient Descent}$$

$$x_{k+1} = x_k^+ + d_k \quad \text{Heavy Ball}$$

$$x_{k+1} = (x_k + d_k)^+ \quad \text{Nesterov accelerated gradient}$$

# General case convergence

## i Theorem

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is convex and  $L$ -smooth. The Nesterov Accelerated Gradient (NAG) algorithm is designed to solve the minimization problem starting with an initial point  $x_0 = y_0 \in \mathbb{R}^n$  and  $\lambda_0 = 0$ . The algorithm iterates the following steps:

**Gradient update:**  $y_{k+1} = x_k - \frac{1}{L} \nabla f(x_k)$

**Extrapolation:**  $x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$

**Extrapolation weight:**  $\lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}$

**Extrapolation weight:**  $\gamma_k = \frac{1 - \lambda_k}{\lambda_{k+1}}$

The sequences  $\{f(y_k)\}_{k \in \mathbb{N}}$  produced by the algorithm will converge to the optimal value  $f^*$  at the rate of  $\mathcal{O}\left(\frac{1}{k^2}\right)$ , specifically:

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

# General case convergence

## i Theorem

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $\mu$ -strongly convex and  $L$ -smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point  $x_0 = y_0 \in \mathbb{R}^n$  and  $\lambda_0 = 0$ . The algorithm iterates the following steps:

**Gradient update:**  $y_{k+1} = x_k - \frac{1}{L} \nabla f(x_k)$

**Extrapolation:**  $x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$

**Extrapolation weight:**  $\gamma_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$

The sequences  $\{f(y_k)\}_{k \in \mathbb{N}}$  produced by the algorithm will converge to the optimal value  $f^*$  linearly:

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