

ORF522 – Linear and Nonlinear Optimization

19. Acceleration schemes

HW 4 → DUE THU 25
HW 5 → OUT THU 25

Ed Forum

$$\begin{array}{ll} \min & f(x) + g(z) \\ \text{st.} & x = z \end{array} \quad ||$$

$$g(x) = I_C(x)$$

- Can Method of multipliers, and ADMM handle inequality constraints, or it only deal with equality constraints?
- On slide 38, it says that "the choice of lambda can greatly change performance". If that's the case, then what are the ways one can go about choosing a proper lambda?
- I am still not sure about how to develop distributed algorithms, as it seemed that the process could be broken down into only two parts (consensus optimization with x and z being separate vector operations). Is there way to distribute the algorithms over more than just two parts?

Recap

Operator splitting

Main idea

We would like to solve

$$0 \in F(x), \quad F \text{ maximal monotone}$$

Split the operator

$$F = A + B, \quad A \text{ and } B \text{ are maximal monotone}$$

Solve by evaluating

$$R_A = (I + A)^{-1}$$

$$R_B = (I + B)^{-1}$$

or

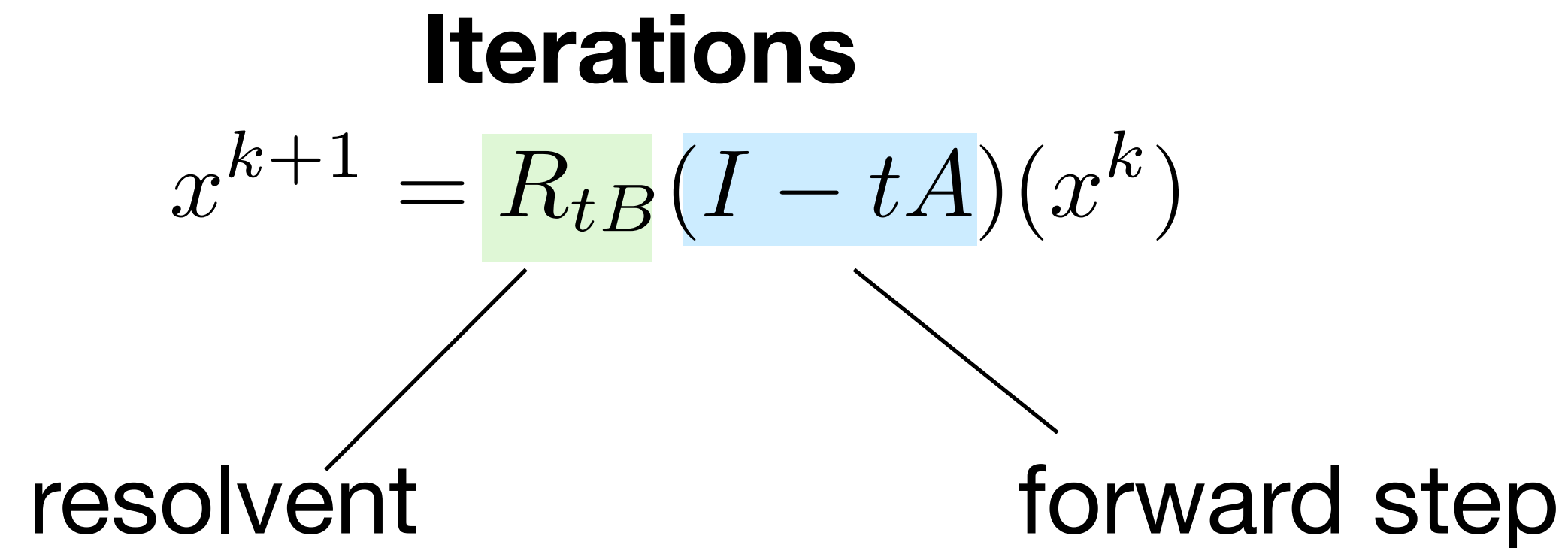
$$C_A = 2R_A - I$$

$$C_B = 2R_B - I$$

Useful when R_A and R_B are cheaper than R_F

Forward-backward splitting

Properties



Properties

- R_{tB} is $1/2$ averaged
- If A is μ -cocoercive then $I - 2\mu A$ is nonexpansive
 $\Rightarrow I - tA$ is averaged for $t \in (0, 2\mu)$
- Therefore forward-backward splitting converges
- If either A or B is strongly monotone, then **linear convergence**

Proximal gradient descent as forward-backward splitting

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \end{array} \quad \begin{array}{l} f \text{ is } L\text{-smooth} \\ g \text{ is nonsmooth but proxable} \end{array}$$

Therefore, ∇f is $(1/L)$ -cocoercive and ∂g maximal monotone

Proximal gradient descent

$$\begin{aligned} x^{k+1} &= R_{t\partial g}(I - t\nabla f)(x^k) \\ &= \text{prox}_{tg}(x^k - t\nabla f(x^k)) \end{aligned}$$

Remarks

- Converges for $t \in (0, 2/L)$
- If either f or g strongly convex **linear convergence**
- If $g = \mathcal{I}_C$, then it's projected gradient descent

Simplified iterations of Douglas-Rachford splitting

DR iterations

$$z^{k+1} = R_B(w^k)$$

$$w^{k+1} = w^k + R_A(2z^{k+1} - w^k) - z^{k+1}$$

1 Swap iterations and counter

$$w^{k+1} = w^k + R_A(2z^k - w^k) - z^k$$

$$z^{k+1} = R_B(w^{k+1})$$

3 Update w^{k+1} at the end

$$x^{k+1} = R_A(2z^k - w^k)$$

$$z^{k+1} = R_B(w^k + x^{k+1} - z^k)$$

$$w^{k+1} = w^k + x^{k+1} - z^k$$

2 Introduce x^{k+1}

$$x^{k+1} = R_A(2z^k - w^k)$$

$$w^{k+1} = w^k + x^{k+1} - z^k$$

$$z^{k+1} = R_B(w^{k+1})$$

4 Define $u^k = w^k - z^k$

$$x^{k+1} = R_A(z^k - u^k)$$

$$z^{k+1} = R_B(x^{k+1} + u^k)$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Douglas-Rachford splitting

Simplified iterations

$$x^{k+1} = R_A(z^k - u^k)$$

$$z^{k+1} = R_B(x^{k+1} + u^k)$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$



Residual: $x^{k+1} - z^{k+1}$

**running sum of
residuals**
 u^k

Interpretation as
integral control

Remarks

- *many* ways to rearrange the D-R algorithm
- Equivalent to many other algorithms (proximal point, Spingarn's partial inverses, Bregman iterative methods, etc.)
- Need very little to converge: A, B maximal monotone
- Splitting A and B , we can uncouple and evaluate R_A and R_B separately

Alternating direction method of multipliers (ADMM)

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

minimize $f(x) + g(x)$

Proximal iterations

$$x^{k+1} = \text{prox}_{\lambda f}(z^k - u^k)$$

$$z^{k+1} = \text{prox}_{\lambda g}(x^{k+1} + u^k) \longrightarrow$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

ADMM iterations

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left(\lambda f(x) + (1/2) \|x - z^k + u^k\|^2 \right)$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \left(\lambda g(z) + (1/2) \|z - x^{k+1} - u^k\|^2 \right)$$

$$u^{k+1} = u^k + z^{k+1} - x^{k+1}$$

Remarks

- It works for any $\lambda > 0$
- The choice of λ can greatly change performance
- It recently gained a **wide popularity** in various fields:
Machine Learning, Imaging, Control, Finance

Today's lecture

[Chapter 3, COAC][Section 2.2, ILCO][Chapter 1, FMO]

First-order methods acceleration

- Lower bounds
- Acceleration
- Interpretation and examples

Recap of nonlinear optimization

Lower bounds

Sublinear convergence rates

For a convex L -smooth function f we have

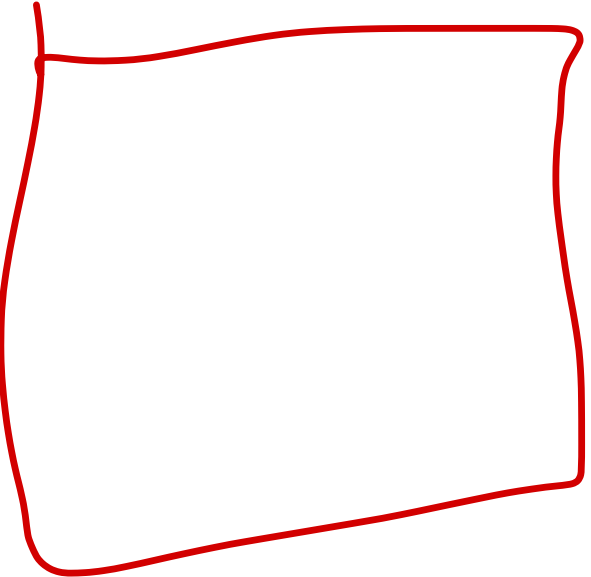
Gradient descent

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

Proximal gradient

$$x^{k+1} = \text{prox}_{tg}(x^k - t \nabla f(x^k))$$

Convergence


$$f(x^k) - f(x^*) \leq \frac{\|x^0 - x^*\|_2^2}{2tk}$$



distance	$O(1/k)$
iterations	$O(1/\epsilon)$

Can we do better?

Lower bounds

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

First-order methods

Any algorithm that selects

$$x^{k+1} \in x_0 + \mathbf{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x^k)\}$$

Lower bounds

First-order methods

Any algorithm that selects

$$x^{k+1} \in x_0 + \mathbf{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x^k)\}$$

Theorem (Nesterov '83)

For every integer $k \leq (n-1)/2$, there exist a convex L -smooth function f such that, for any first-order method

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

Lower bounds

First-order methods

Any algorithm that selects

$$x^{k+1} \in x_0 + \mathbf{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x^k)\}$$

Theorem (Nesterov '83)

For every integer $k \leq (n-1)/2$, there exist a convex L -smooth function f such that, for any first-order method

$$f(x^k) - f(x^*) \geq \frac{3L}{32(k+1)^2} \|x^0 - x^*\|^2 \longrightarrow \begin{array}{ll} \text{distance} & O(1/k^2) \\ \text{iterations} & O(1/\sqrt{\epsilon}) \end{array}$$

Lower bound proof

$$\text{minimize } f(x) = \frac{L}{4} \left(\frac{1}{2} x^T A x - e_1^T x \right) \longrightarrow \nabla f(x) = \frac{L}{4} (A x - e_1)$$

- f is convex and L -smooth
- $x^\star$ is the **optimizer** with $x_i^\star = 1 - \frac{i}{n+1}$ (Solves $\nabla f(x^\star) = 0 \rightarrow A x^\star = e_1$)
- $f(x^\star) = -\frac{L}{8} \frac{n}{n+1}$, $\|x^\star\|^2 \leq \frac{n+1}{3}$

Lower bound proof

$$\text{minimize } f(x) = \frac{L}{4} \left(\frac{1}{2} x^T A x - e_1^T x \right) \longrightarrow \nabla f(x) = \frac{L}{4} (\underline{Ax - e_1})$$

Gilbert Strang (MIT)
“cupcake matrix”

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \quad \underline{e_1 = (1, 0, \dots, 0)}$$

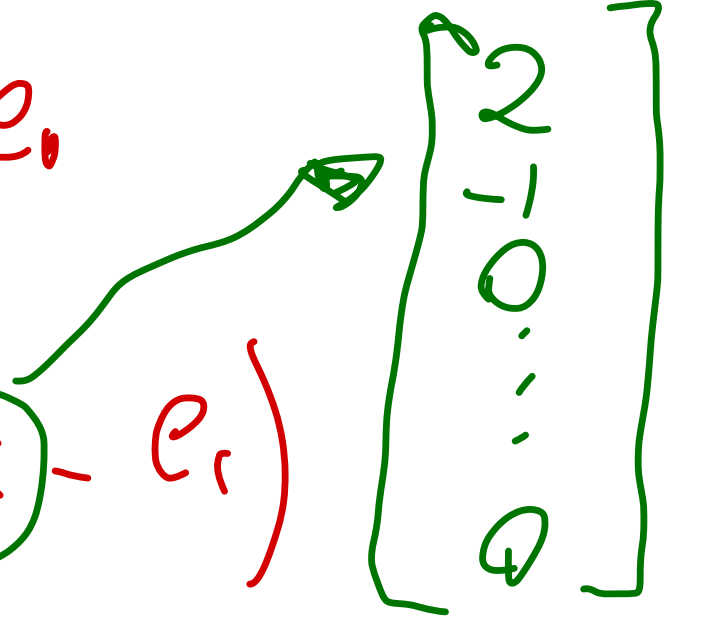
- f is convex and L -smooth
- x^* is the **optimizer** with $x_i^* = 1 - \frac{i}{n+1}$ (Solves $\nabla f(x^*) = 0 \rightarrow Ax^* = e_1$)
- $f(x^*) = -\frac{L}{8} \frac{n}{n+1}$, $\|x^*\|^2 \leq \frac{n+1}{3}$

$x^* = A^{-1}e_1$

Lower bound proof

Iterates

If $x^0 = 0$ then $x^k \in \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{k-1})\} = \text{span}\{e_1, \dots, e_k\}$

$$\begin{aligned}\nabla f(x_0) &= -L/4 e_1 \\ \nabla f(e_i) &= \frac{L}{4} (Ae_i - e_i)\end{aligned}$$


Handwritten notes in red and green ink. The first equation is $\nabla f(x_0) = -L/4 e_1$. The second equation is $\nabla f(e_i) = \frac{L}{4} (Ae_i - e_i)$. A green vector $\begin{bmatrix} 2 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ is written to the right. A green arrow points from the Ae_i term in the second equation to the first element (2) of the vector.

$$e_1 = (1, 0, 0, \dots)$$

$$e_2 = (0, 1, 0, \dots)$$

Lower bound proof

Iterates

If $x^0 = 0$ then $x^k \in \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{k-1})\} = \text{span}\{e_1, \dots, e_k\}$

Upper bound

$$f(x^k) \geq \min_{x \in \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{k-1})\}} f(x) = \min_{x_{k+1} = \dots = x_n = 0} f(x) = -\frac{L}{8} \frac{k}{k+1}$$

Lower bound proof

Iterates

If $x^0 = 0$ then $x^k \in \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{k-1})\} = \text{span}\{e_1, \dots, e_k\}$

Upper bound

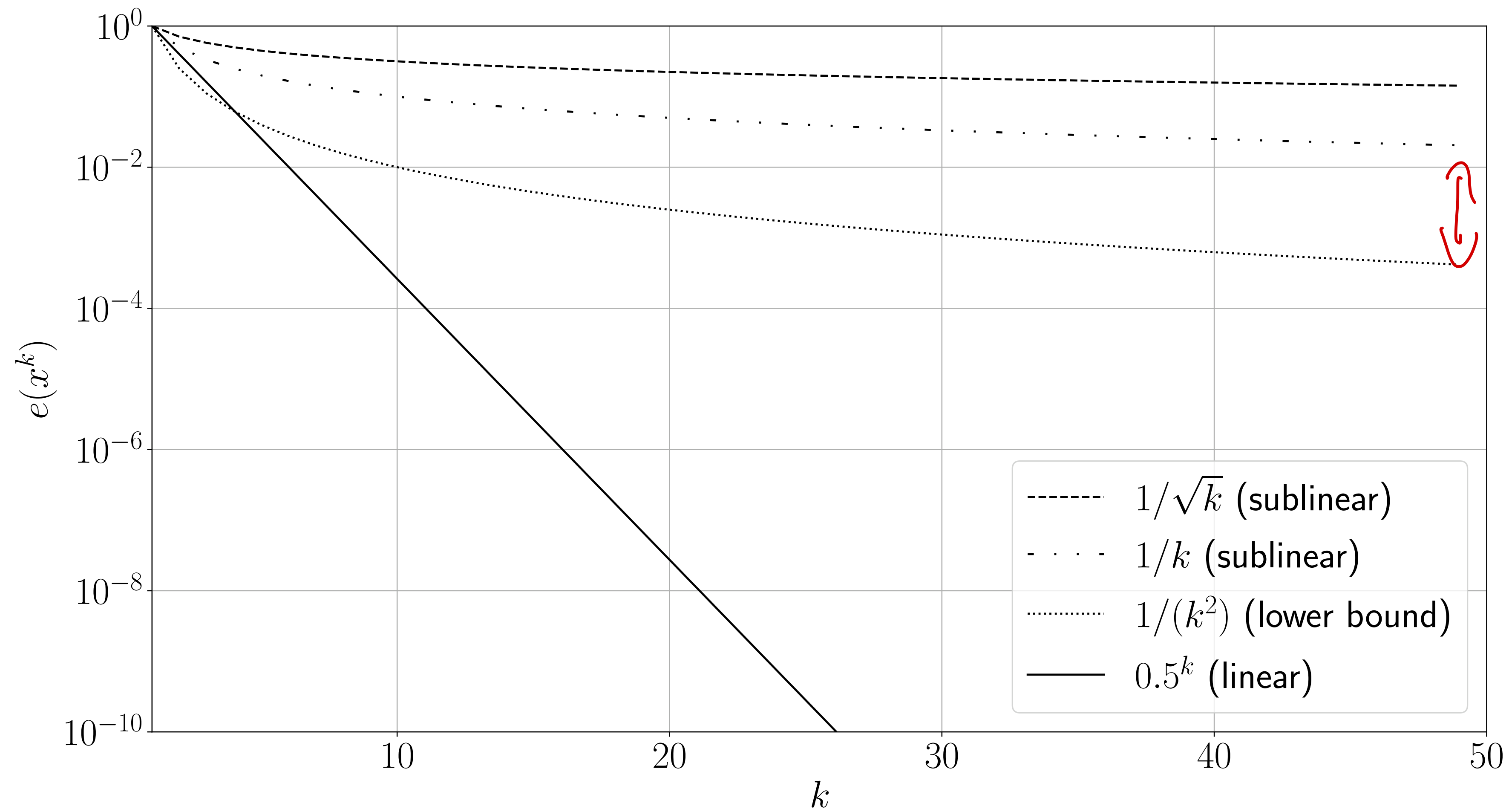
$$f(x^k) \geq \min_{x \in \text{span}\{\nabla f(x^0), \dots, \nabla f(x^{k-1})\}} f(x) = \min_{x_{k+1}=\dots=x_n=0} f(x) = -\frac{L}{8} \frac{k}{k+1}$$

For $k \approx n/2$ or $n = 2k + 1$,

$$\frac{f(x^k) - f(x^*)}{\|x^0 - x^*\|^2} \geq \frac{L}{8} \left(-\frac{k}{k+1} + \frac{2k+1}{2k+2} \right) / \left(\frac{2k+2}{3} \right) = \frac{3L}{32(k+1)^2}$$



Convergence rates



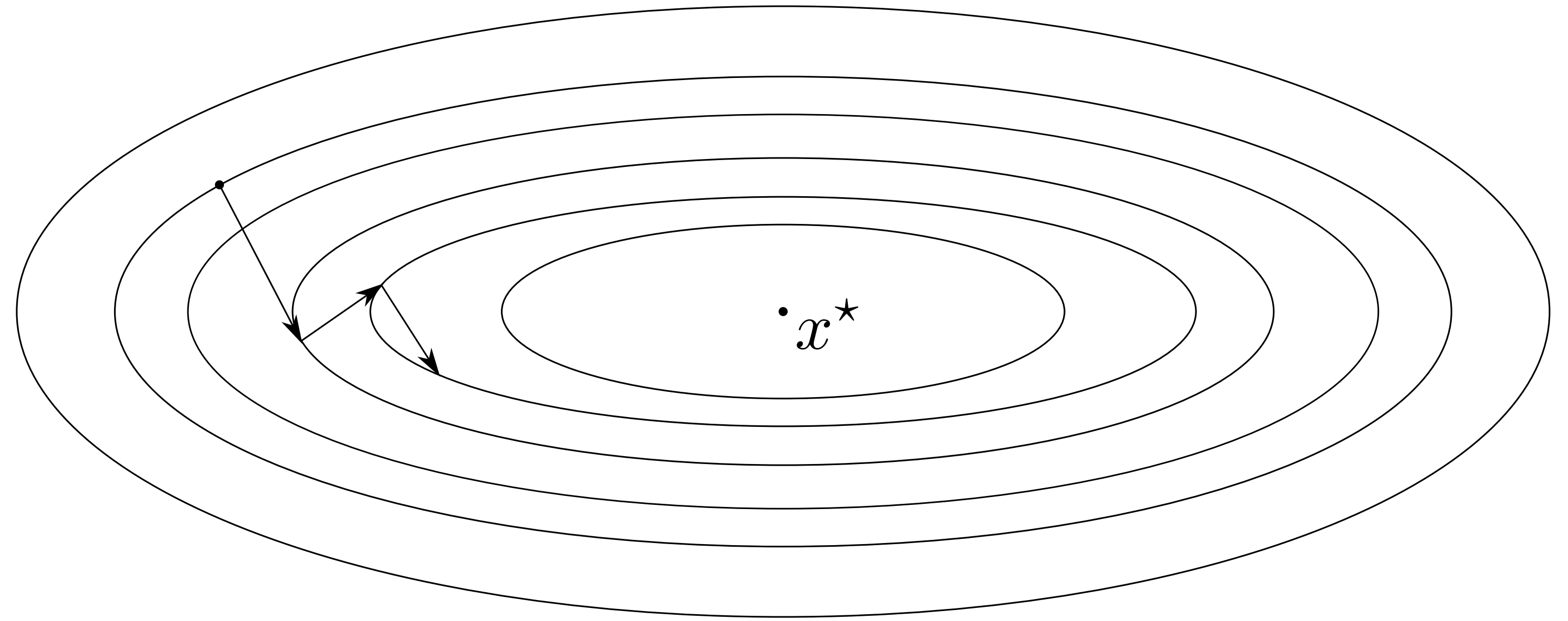
Can we achieve the lower bound?

Acceleration

Momentum

Gradient descent

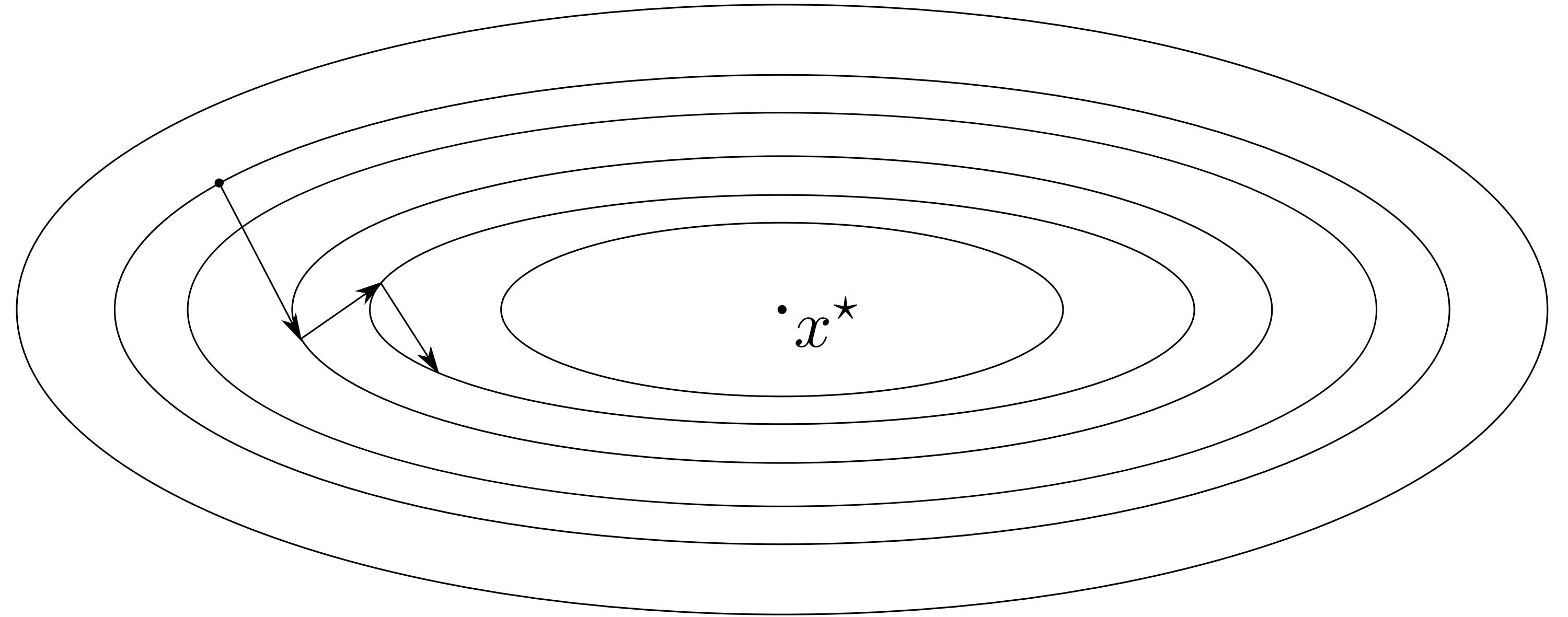
$$x^{k+1} = x^k - t \nabla f(x^k)$$



Momentum

Gradient descent

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

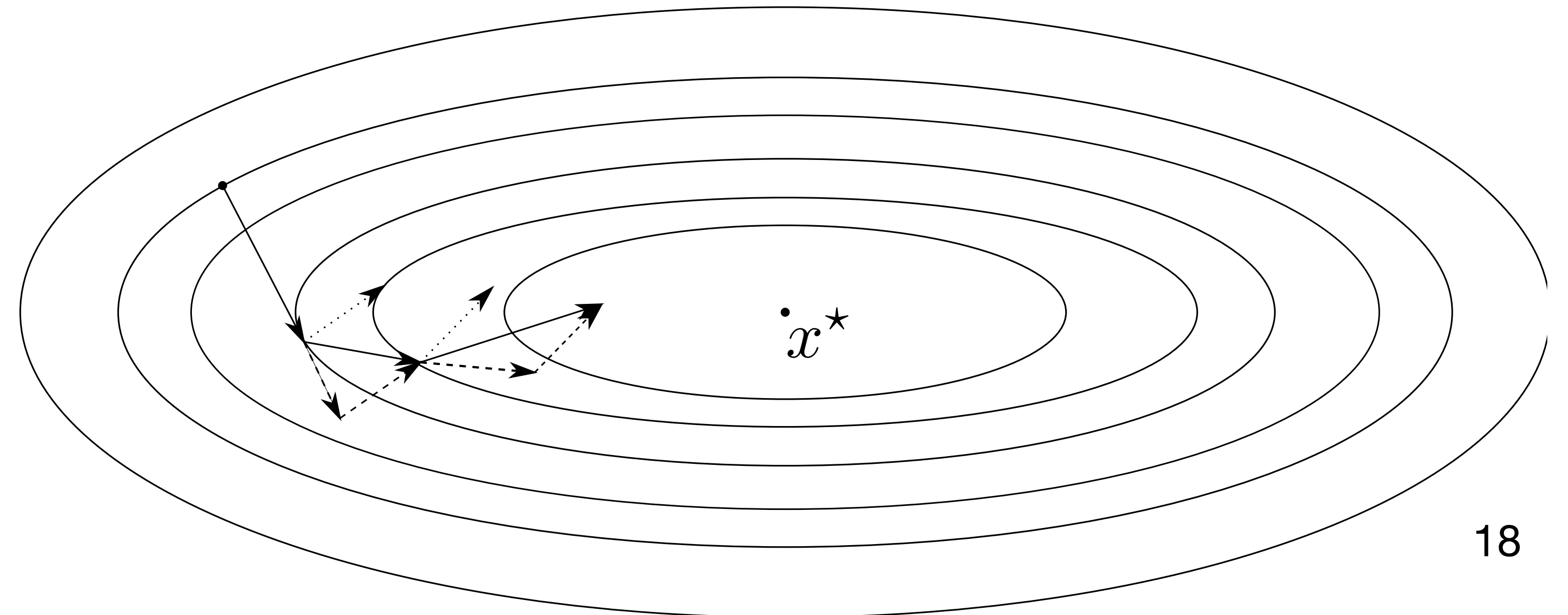


Adding momentum

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

$$y^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$$

momentum



Nesterov momentum

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

$$y^{k+1} = x^{k+1} + \frac{k}{k+3} (x^{k+1} - x^k)$$

Properties

- Original Momentum proposed by Nesterov ('83)
- No longer a descent method (i.e., we can have $f(x^{k+1}) > f(x^k)$)
- Same complexity per iteration as gradient descent

Accelerated proximal gradient method

$$\text{minimize } f(x) + g(x)$$

$f(x)$ convex and smooth

$g(x)$ convex (may be not differentiable)

Iterations

$$x^{k+1} = \text{prox}_{tg} (y^k - t \nabla f(y^k))$$

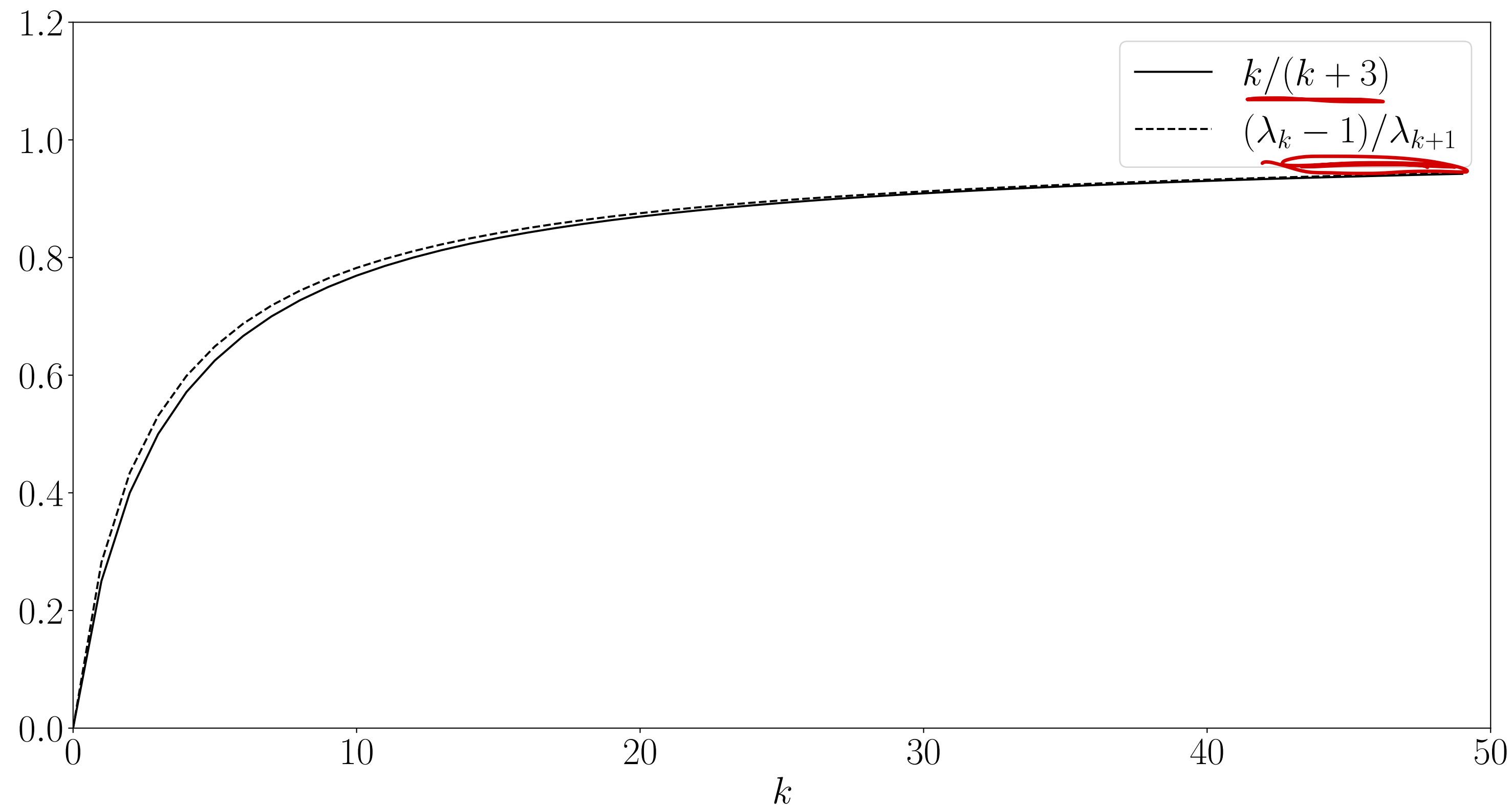
$$y^{k+1} = x^{k+1} + \frac{\lambda_k - 1}{\lambda_{k+1}} (x^{k+1} - x^k)$$

$$\text{where } y_0 = x_0 \text{ and } \lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}$$

Note: $g(x) = 0$ gives accelerated gradient descent

Proximal gradient and Nesterov weights

$$\lambda_0 = 1 \quad \lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2} \longrightarrow \frac{\lambda_k - 1}{\lambda_{k+1}} \approx \frac{k}{k+3} \text{ as } \underline{k \rightarrow \infty}$$



Convergence rate for accelerated proximal gradient method

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \end{array} \quad \begin{array}{l} f(x) \text{ convex and } L\text{-smooth} \\ g(x) \text{ convex (may be not differentiable)} \end{array}$$

Theorem

The accelerated proximal gradient method with step-size $t \leq (1/L)$ satisfies

$$f(x^k) - f(x^*) \leq \frac{2\|x^0 - x^*\|^2}{t(k+1)^2}$$

Convergence rate for accelerated proximal gradient method

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \end{array} \quad \begin{array}{l} f(x) \text{ convex and } L\text{-smooth} \\ g(x) \text{ convex (may be not differentiable)} \end{array}$$

Theorem

The accelerated proximal gradient method with step-size $t \leq (1/L)$ satisfies

$$f(x^k) - f(x^*) \leq \frac{2\|x^0 - x^*\|^2}{t(k+1)^2}$$

Proof

[Thm 4.4, A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems, Beck, Teboulle]

Convergence rate for accelerated proximal gradient method

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \\ & f(x) \text{ convex and } L\text{-smooth} \\ & g(x) \text{ convex (may be not differentiable)} \end{array}$$

Theorem

The accelerated proximal gradient method with step-size $t \leq (1/L)$ satisfies

$$f(x^k) - f(x^*) \leq \frac{2\|x^0 - x^*\|^2}{t(k+1)^2}$$

Proof

[Thm 4.4, A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems, Beck, Teboulle]

Note

It works for any momentum weights $\overbrace{(\lambda_k - 1)/\lambda_{k+1}}^{\beta_k}$ such that

$$\bigvee \lambda_k \geq \frac{k+2}{2} \quad \text{and} \quad \bigvee \lambda_{k+1}^2 \leq \lambda_{k+1} - \lambda_k^2$$

Convergence rate for accelerated proximal gradient method

$$\begin{array}{ll} \text{minimize} & f(x) + g(x) \end{array} \quad \begin{array}{l} f(x) \text{ convex and } L\text{-smooth} \\ g(x) \text{ convex (may be not differentiable)} \end{array}$$

$$f(x^k) - f(x^*) \leq \frac{2\|x^0 - x^*\|^2}{t(k+1)^2}$$

- Better iteration complexity $O(1/k^2)$ (i.e. $O(1/\sqrt{\epsilon})$)
- Fast if prox evaluations are cheap
- Can't do better! (from **lower bound**)

Examples and interpretations

ODE interpretation

Nesterov acceleration

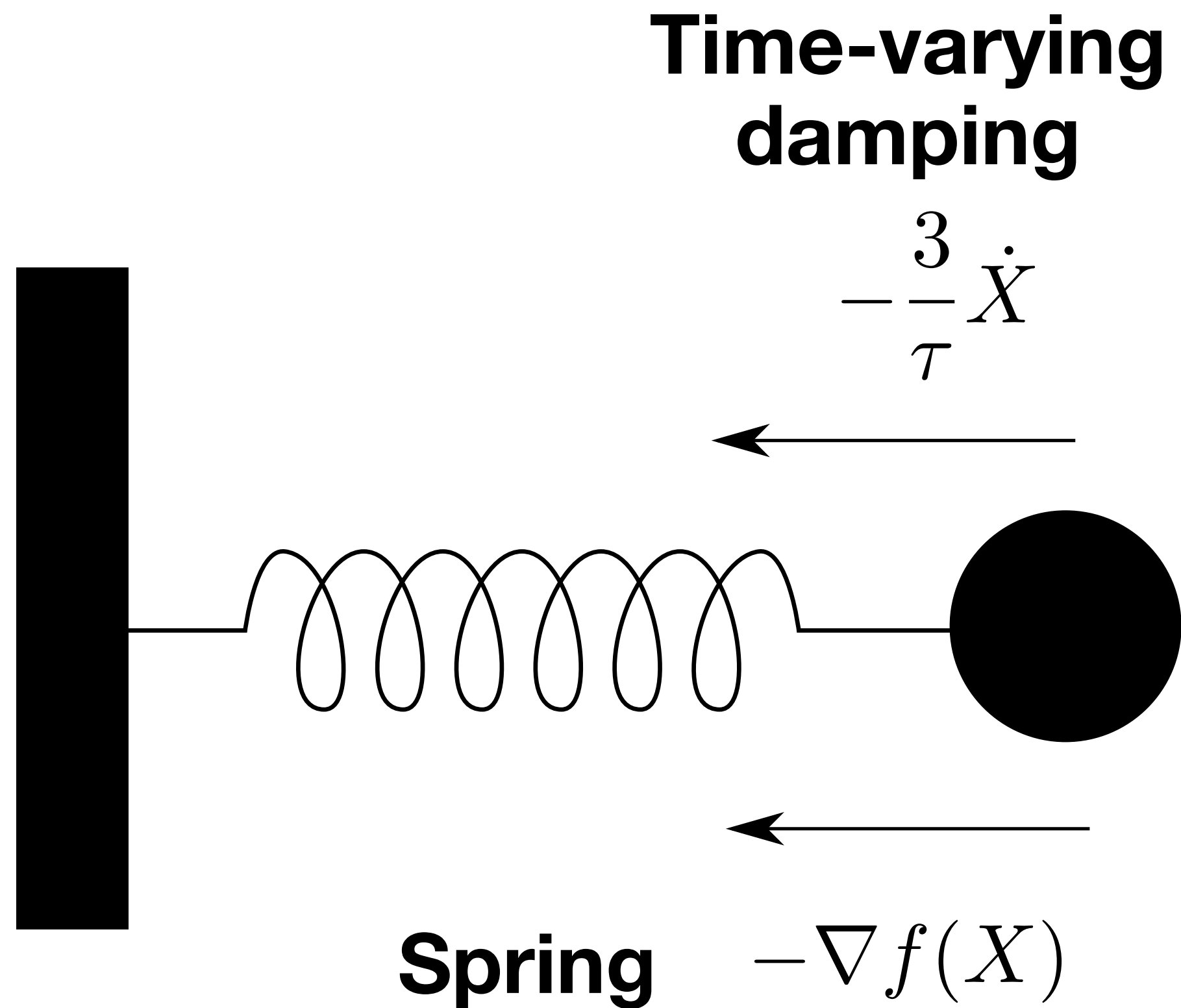
$$x^{k+1} = y^k - t \nabla f(y^k)$$

$$y^{k+1} = x^{k+1} + \frac{k}{k+3} (x^{k+1} - x^k)$$

$$t \rightarrow 0 \quad \downarrow \quad x^k \approx X(k\sqrt{t}) = X(\tau)$$

$$\ddot{X}(\tau) + \frac{3}{\tau} \dot{X}(\tau) + \nabla f(X(\tau)) = 0$$

ODE interpretation



Nesterov acceleration

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

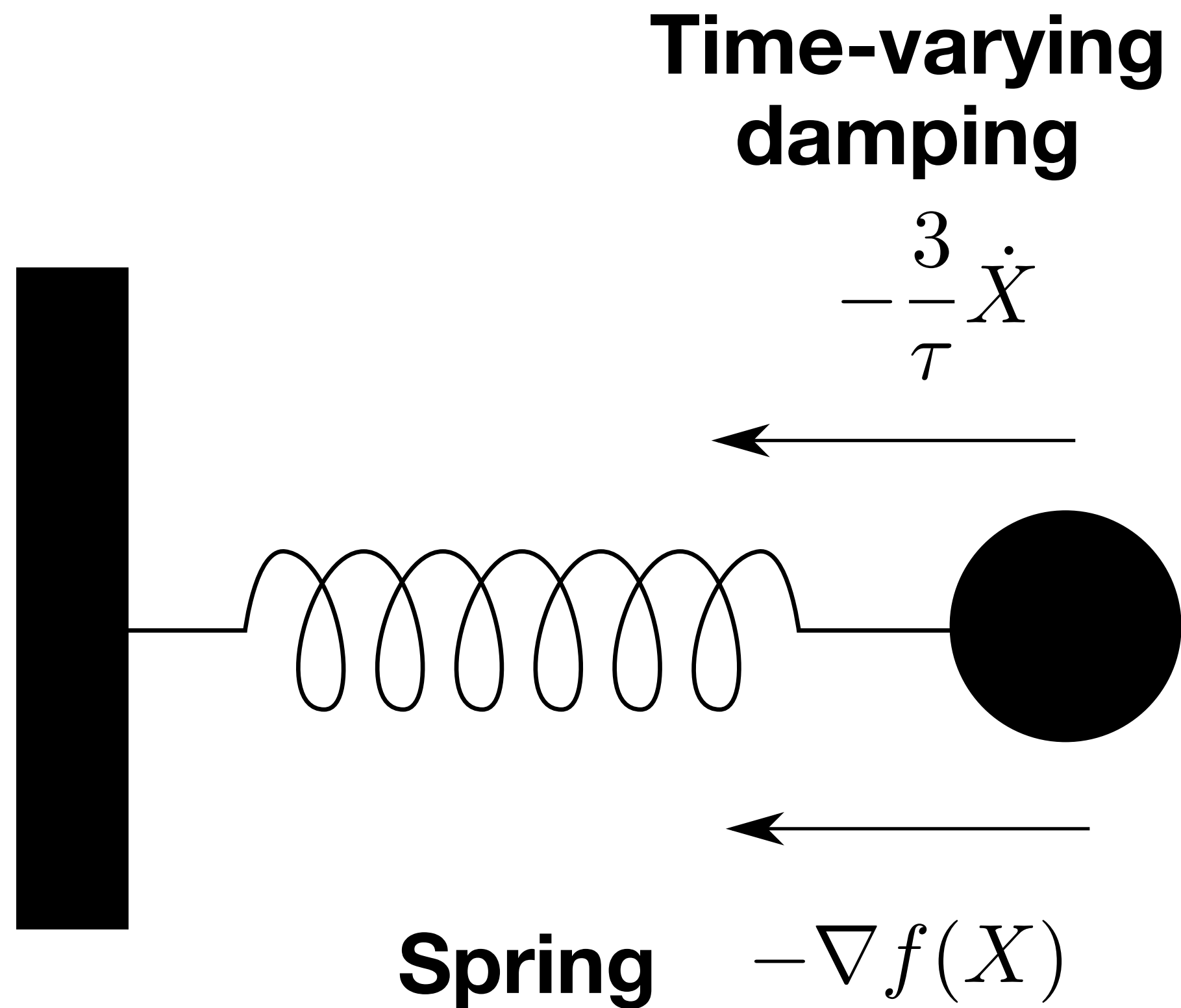
$$y^{k+1} = x^{k+1} + \frac{k}{k+3} (x^{k+1} - x^k)$$

$$t \rightarrow 0 \quad \downarrow \quad x^k \approx X(k\sqrt{t}) = X(\tau)$$

$$\ddot{X}(\tau) + \frac{3}{\tau} \dot{X}(\tau) + \nabla f(X(\tau)) = 0$$

**damping
coefficient**

ODE interpretation



Nesterov acceleration

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

$$y^{k+1} = x^{k+1} + \frac{k}{k+3} (x^{k+1} - x^k)$$

$$t \rightarrow 0 \quad \downarrow \quad x^k \approx X(k\sqrt{t}) = X(\tau)$$

$$\ddot{X}(\tau) + \frac{3}{\tau} \dot{X}(\tau) + \nabla f(X(\tau)) = 0$$

**damping
coefficient**

Note: 3 is the smallest constant that guarantees $O(1/\tau^2)$ convergence

Example: Lasso without linear convergence

$$\text{minimize} \quad \underbrace{(1/2)\|Ax - b\|_2^2}_{f(x)} + \underbrace{\gamma\|x\|_1}_{g(x)}$$

**Proximal gradient descent
(Iterative Shrinkage Thresholding Algorithm)**

$$x^{k+1} = S_{\gamma t} (x^k - tA^T (Ax^k - b))$$

ISTA

Example: Lasso without linear convergence

$$\text{minimize} \quad \underbrace{(1/2)\|Ax - b\|_2^2}_{f(x)} + \underbrace{\gamma\|x\|_1}_{g(x)}$$

**Proximal gradient descent
(Iterative Shrinkage Thresholding Algorithm)**

$$x^{k+1} = S_{\gamma t} (x^k - tA^T (Ax^k - b))$$

ISTA

**Accelerated proximal gradient descent
(Fast Iterative Shrinkage Thresholding Algorithm)**

$$x^{k+1} = S_{\gamma t} (y^k - tA^T (Ay^k - b))$$

$$y^{k+1} = x^{k+1} + \frac{\lambda_k - 1}{\lambda_{k+1}} (x^{k+1} - x^k)$$

FISTA

Example: Lasso without linear convergence

Fast Iterative Soft Thresholding Algorithm (FISTA)

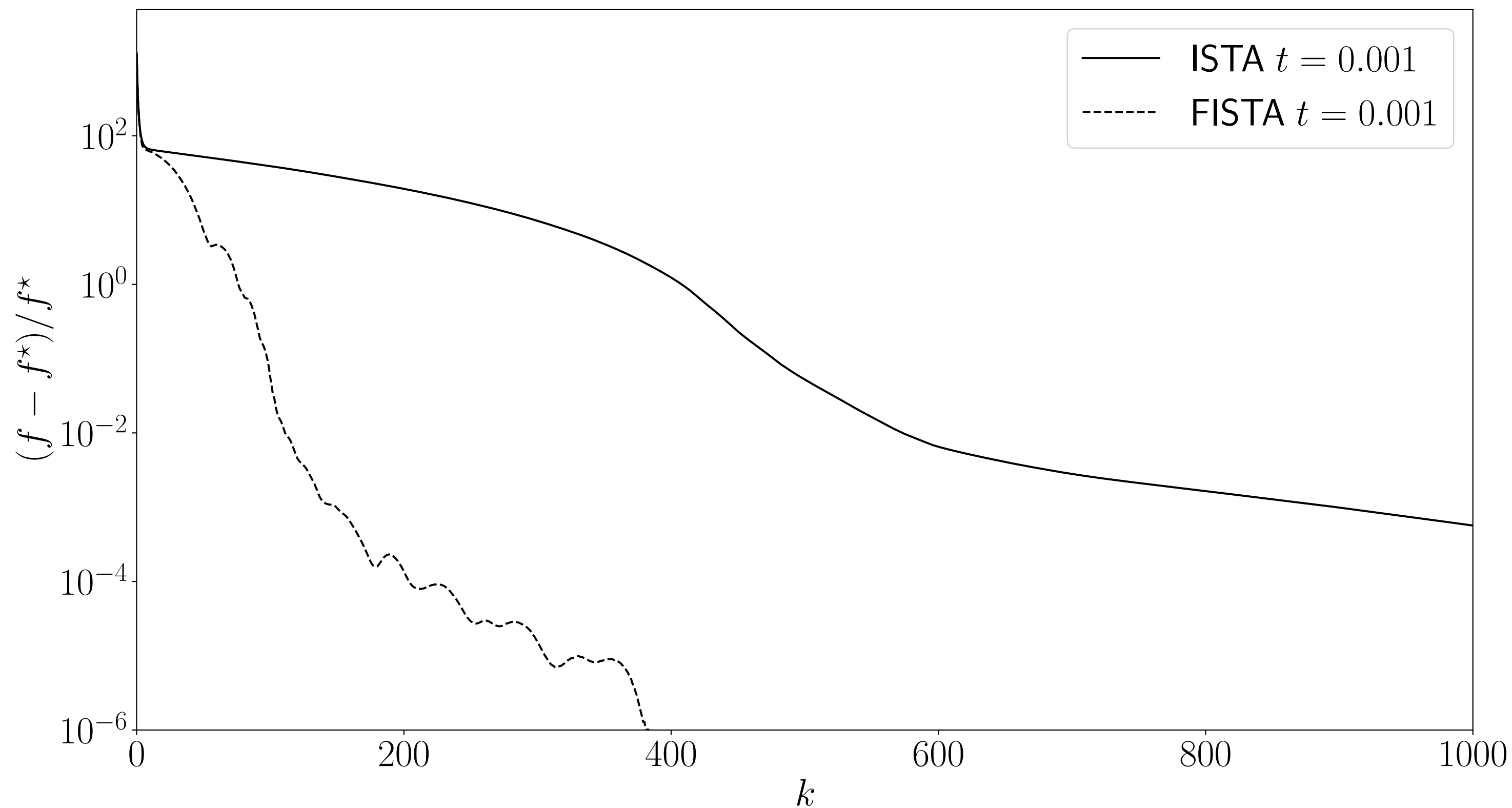
$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$

Example

randomly generated $A \in \mathbf{R}^{300 \times 500}$

$$\Rightarrow \nabla^2 f = A^T A \succeq 0$$

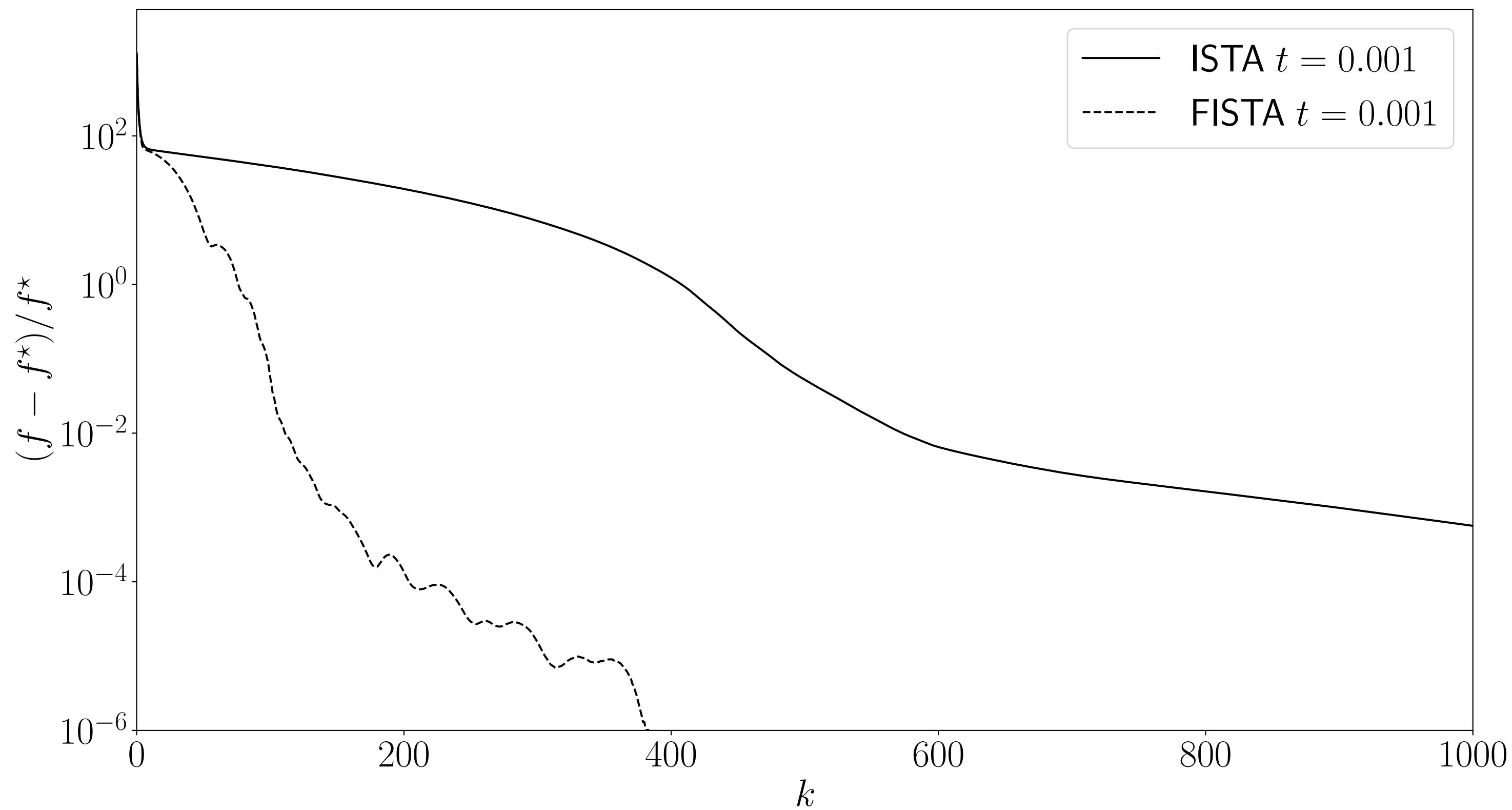
$\Rightarrow f$ not strongly convex



Example: Lasso without linear convergence

Fast Iterative Soft Thresholding Algorithm (FISTA)

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$



Example

randomly generated $A \in \mathbf{R}^{300 \times 500}$

$$\Rightarrow \nabla^2 f = A^T A \succeq 0$$

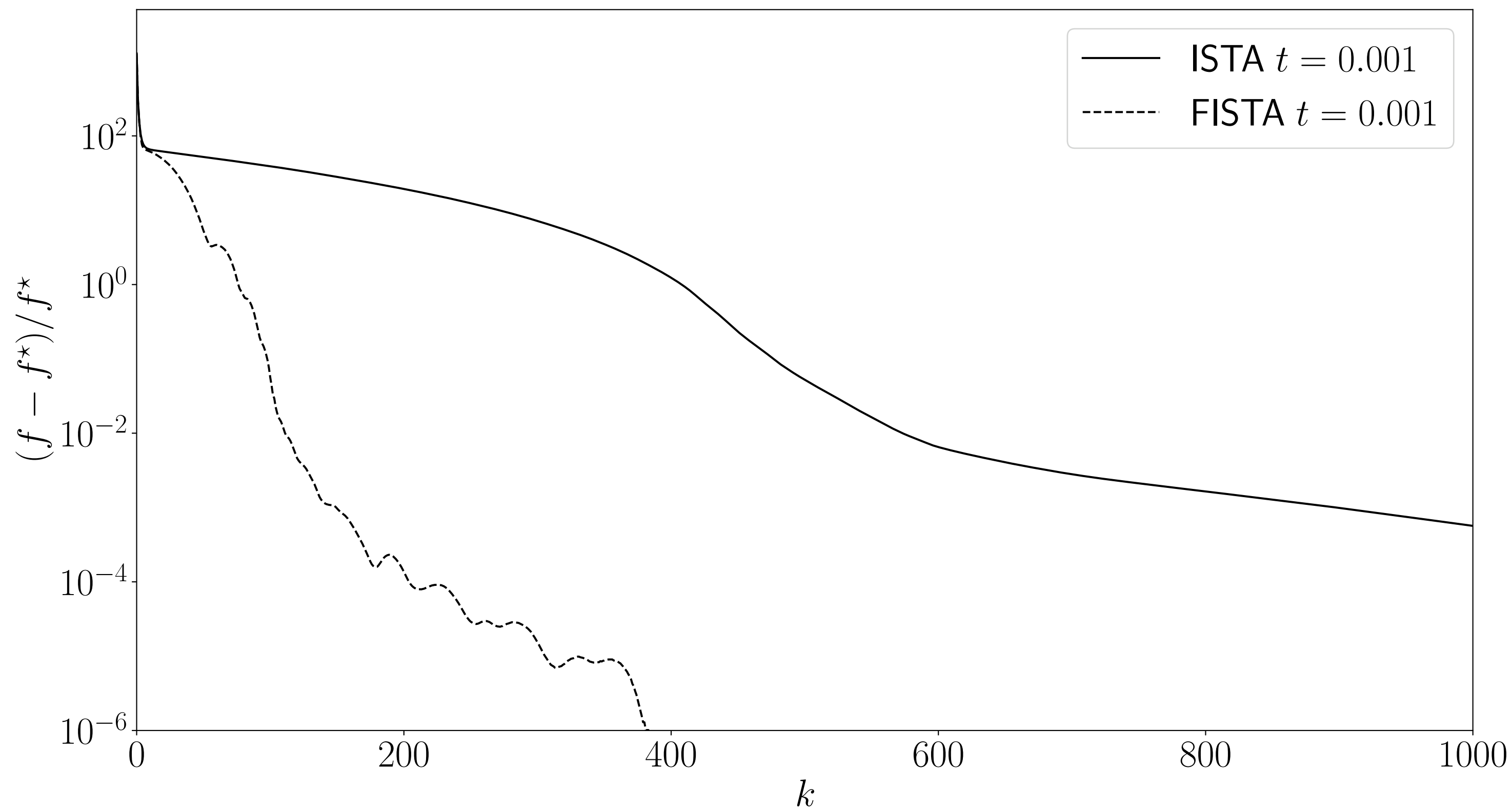
$\Rightarrow f$ not strongly convex

FISTA is much faster

Example: Lasso without linear convergence

Fast Iterative Soft Thresholding Algorithm (FISTA)

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$



Example

randomly generated $A \in \mathbf{R}^{300 \times 500}$

$$\Rightarrow \nabla^2 f = A^T A \succeq 0$$

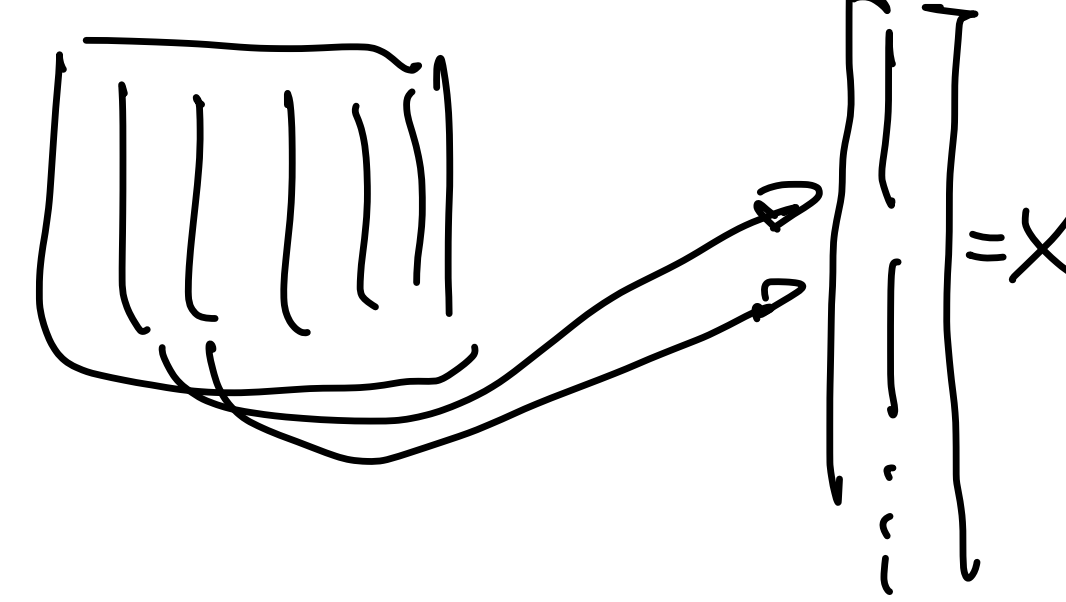
$\Rightarrow f$ not strongly convex

FISTA is much faster

Typical rippling behavior
(not a descent method)

Image deblurring

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$



x : reconstructed image
in wavelet basis (sparse)

original



blurred



Image deblurring

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1$$

x : reconstructed image
in wavelet basis (sparse)

original



blurred



ISTA

$k = 100$



$k = 200$



FISTA



More sophisticated accelerations

Other algorithms

Acceleration can also be applied also to ADMM

[Fast Alternating Direction Optimization Methods, Goldstein, O'Donoghue, Setzer, Baraniuk]

Momentum with restarts
(reset momentum when it makes
small progress)



**Improved
convergence rate**
 $O(1/k^2)$

More sophisticated accelerations

Other algorithms

Acceleration can also be applied also to ADMM

[Fast Alternating Direction Optimization Methods, Goldstein, O'Donoghue, Setzer, Baraniuk]

Momentum with restarts
(reset momentum when it makes
small progress)



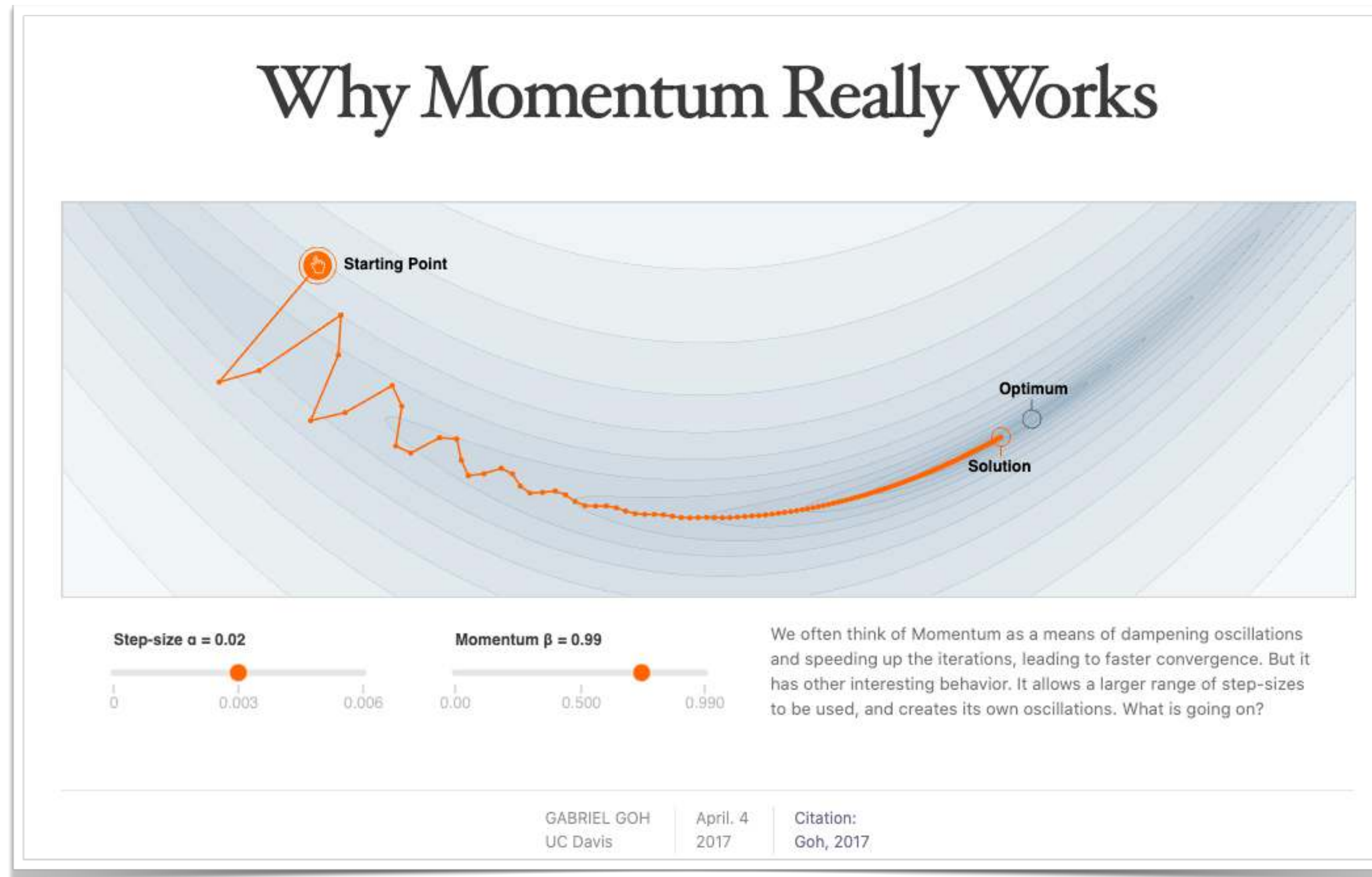
**Improved
convergence rate**
 $O(1/k^2)$

Nonlinear acceleration
(e.g., Anderson Acceleration)

Adaptively pick weights by solving
a small optimization problem
(usually least-squares)

[Acceleration Methods, d'Aspremont, Scieur, Taylor]

Momentum intuition and much more



All deep learning optimization algorithms are based on Momentum/Acceleration: RMSprop, AdaGrad, Adam, etc.

<https://distill.pub/2017/momentum/>

Summary of nonlinear optimization

Nonlinear optimization

Optimality conditions

- KKT optimality conditions
- Subgradient optimality conditions $0 \in \partial f(x^*)$

General
Necessary

Convex
Necessary and sufficient

Nonlinear optimization

Optimality conditions

- KKT optimality conditions
- Subgradient optimality conditions $0 \in \partial f(x^*)$

General
Necessary

Convex
Necessary and sufficient

First order methods: Moderate accuracy on Large-scale data

- Gradient descent
- Subgradient methods
- Proximal algorithms (e.g., ISTA)
- Operator splitting algorithms (e.g., ADMM)

Convergence rates

Typical rates

(gradient descent, proximal gradient, ADMM, etc.)

- L -smoothness: $O(1/k)$, accelerated $O(1/k^2)$
- μ -strong convexity: $O(\log(1/k))$
- We can always combine **line search**
- Convergence bounds usually in terms of cost function distance

Convergence rates

Typical rates

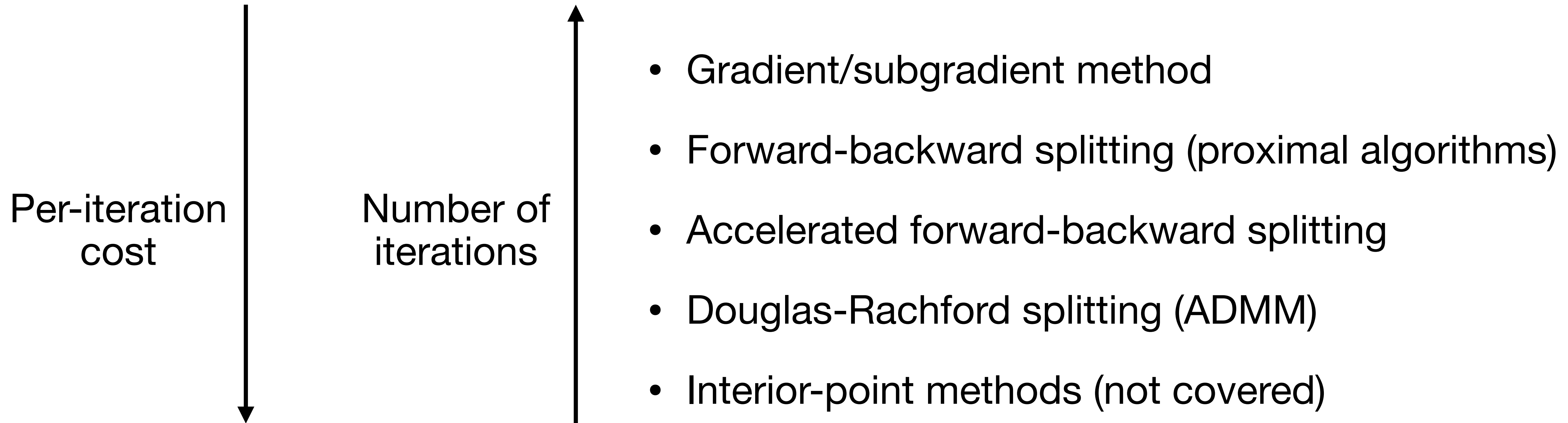
(gradient descent, proximal gradient, ADMM, etc.)

- L -smoothness: $O(1/k)$, accelerated $O(1/k^2)$
- μ -strong convexity: $O(\log(1/k))$
- We can always combine **line search**
- Convergence bounds usually in terms of cost function distance

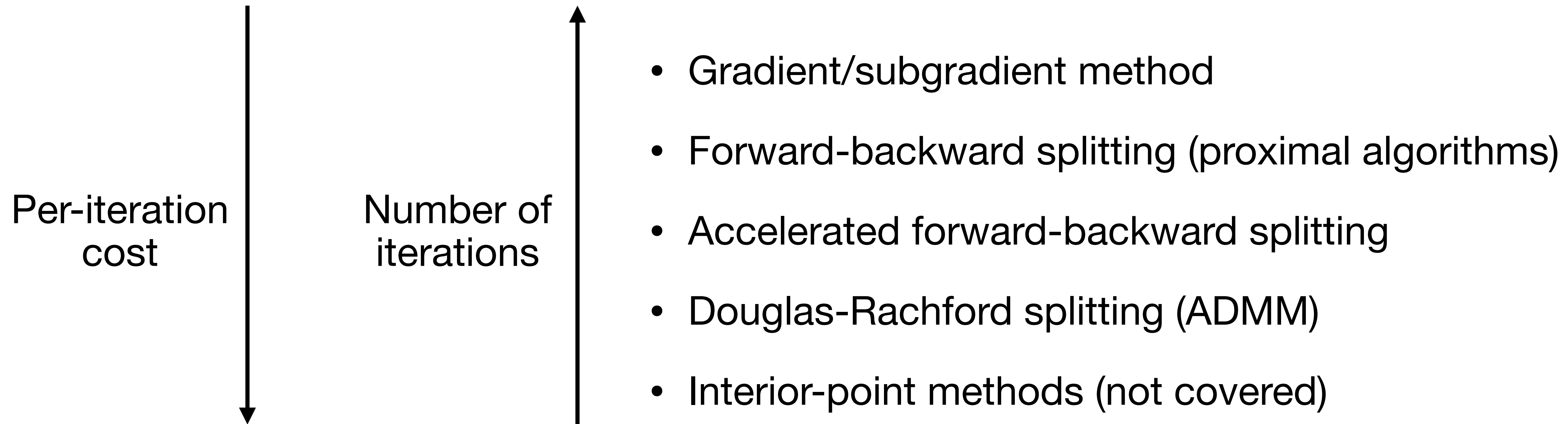
Operator theory

- Helps developing and analyzing serial and distributed algorithms
- Algorithms **always** converge for convex problems (independently from step size)
- Convergence bounds usually in terms of iterates distance

First-order methods



First-order methods



Large-scale systems

- start with feasible method with cheapest per-iteration cost
- if too many iterations, transverse down the list

Acceleration in nonlinear optimization

Today, we learned to:

- **Derive** lower bounds on cost optimality for first-order methods
- **Accelerate** first-order algorithms by adding momentum term
- **Apply** acceleration schemes to get the best possible convergence
- **Select** the appropriate algorithms to apply in large-scale optimization

Next lecture

- Extensions and nonconvex optimization