# ORF522 – Linear and Nonlinear Optimization

## 20. Sequential Convex Programming

**Bartolomeo Stellato — Fall 2020**

# Ed forum

- Theorem of lower bounds (Nesterov'83). The theorem declares the existence of a fuction f, and gives its lower bound for first order methods; but how does it give lower bounds for all convex L-smooth functions?
- Can we include more previous directions instead of just rewinding 1 step in the momentum acceleration scheme? Yes! Anderson Acceleration

# Today's lecture

**[Chapter 4 and 17, Numerical Optimization, Nocedal and Wright]**
**[Stanford ee364b Lecture Notes, Boyd]**

**Convex algorithms to solve nonconvex optimization problems**

- Sequential convex programming
- Trust region methods
- Building convex approximations
- Regularized trust region methods
- Difference of convex programming

# Methods for nonconvex optimization

**Convex optimization algorithms: global** and typically **fast**

**Nonconvex optimization algorithms:** must give up one, global or fast

- **Local methods: fast** but **not global** ⟶ **heuristics**
  Need not find a global (or even feasible) solution.
  They cannot certify global optimality because
  KKT conditions are not sufficient.

- **Global methods: global** but often **slow**
  They find a global solution and certify it.

# Sequential Convex Programming

# Sequential convex programming (SCP)

**Local optimization method that leverages convex optimization**

**Subprograms are convex** $\longrightarrow$ we can solve them efficiently

It is a **heuristic**

- It **can fail** to find an optimal (or even feasible point)
- Results **depend on the starting point.**
  We can run the algorithm from many initial points and take the best result.

**It often works very well**
it finds a feasible point with good objective value (often optimal!)

# Gradient descent as SCP

**Problem**

minimize $\quad f(x)$

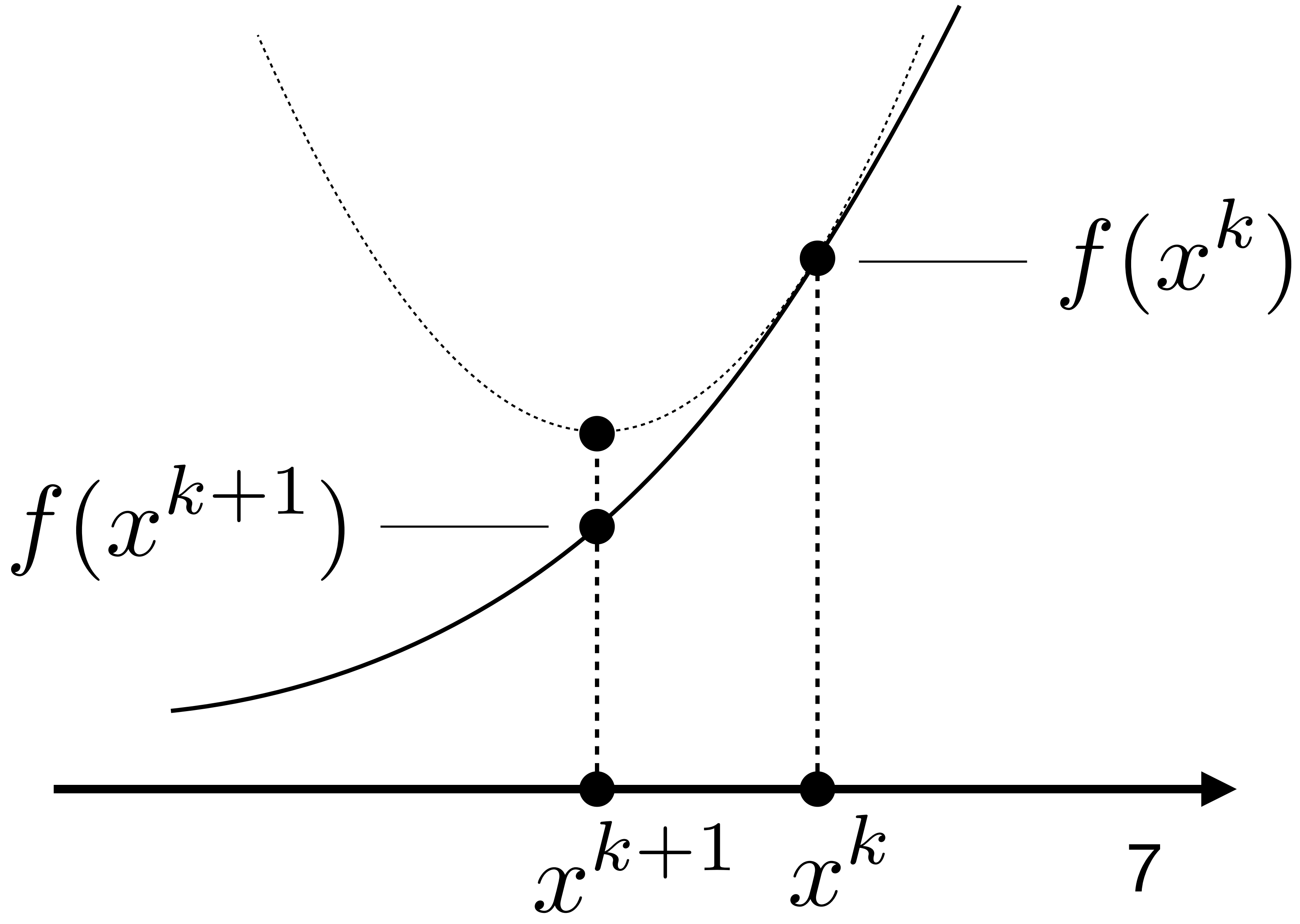
**Iterates**

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

**Quadratic approximation**, replace $\nabla^2 f(x^k)$ with $\dfrac{1}{t_k} I$

$$x^{k+1} = \underset{y}{\text{argmin}}\ f(x^k) + \nabla f(x^k)^T (y - x^k) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

**strongly convex problem**



# The problem

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& h_i(x) = 0, \quad i = 1, \ldots, p
\end{array}
$$

with $x \in \mathbf{R}^n$

- $f$ and $g_i$ can be nonconvex
- $h_i$ can be nonaffine

# Trust region methods

# Main idea

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \le 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

$\big\downarrow$ iterate $x^k$
**trust region** $\mathcal{T}^k$

**approximate convex problem**

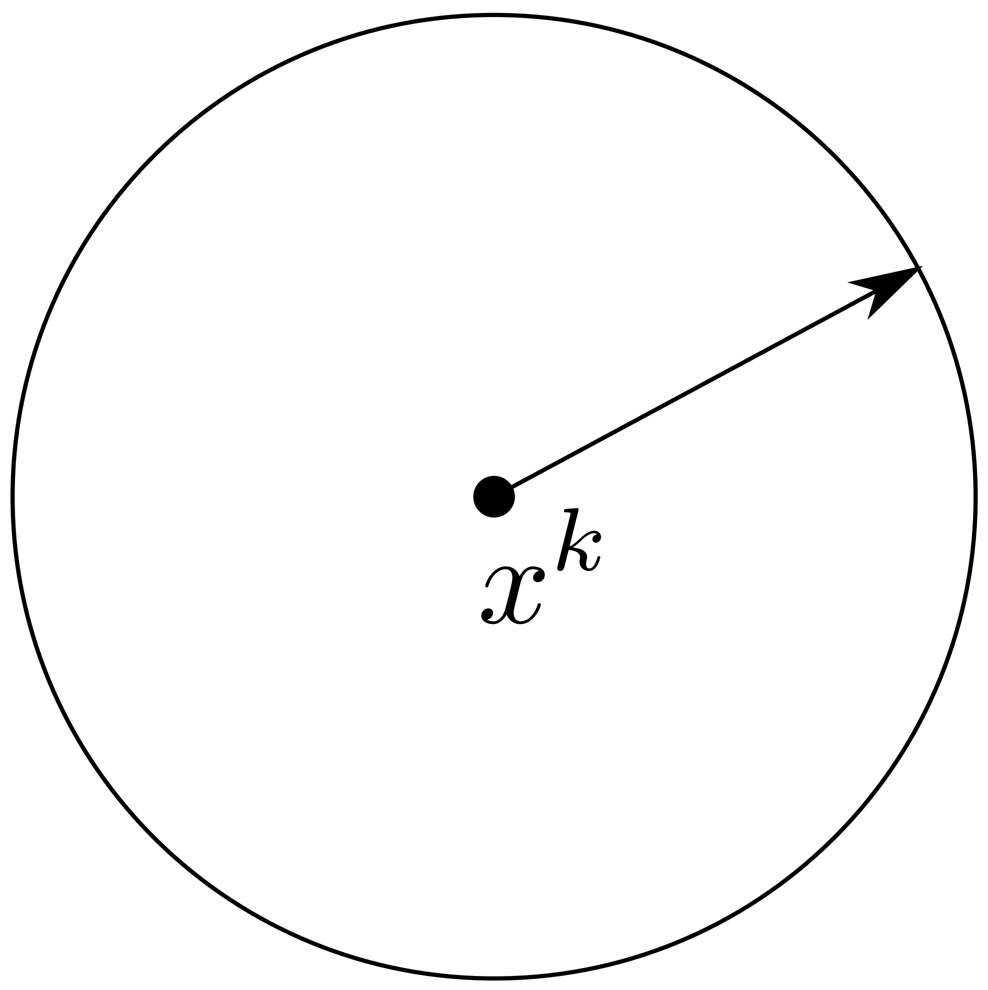
$$\begin{array}{ll} \text{minimize} & \hat{f}(x) \\ \text{subject to} & \hat{g}_i(x) \le 0, \quad i = 1, \ldots, m \\ & \hat{h}_i(x) = 0, \quad i = 1, \ldots, p \\ & x \in \mathcal{T}^k \end{array} \longrightarrow \text{solve to get } x^{k+1}$$

- $\hat{f}$ ($\hat{g}_i$) is a convex approximation of $f$ ($g_i$) over $\mathcal{T}^k$
- $\hat{h}$ is an affine approximation of $h$ over $\mathcal{T}^k$

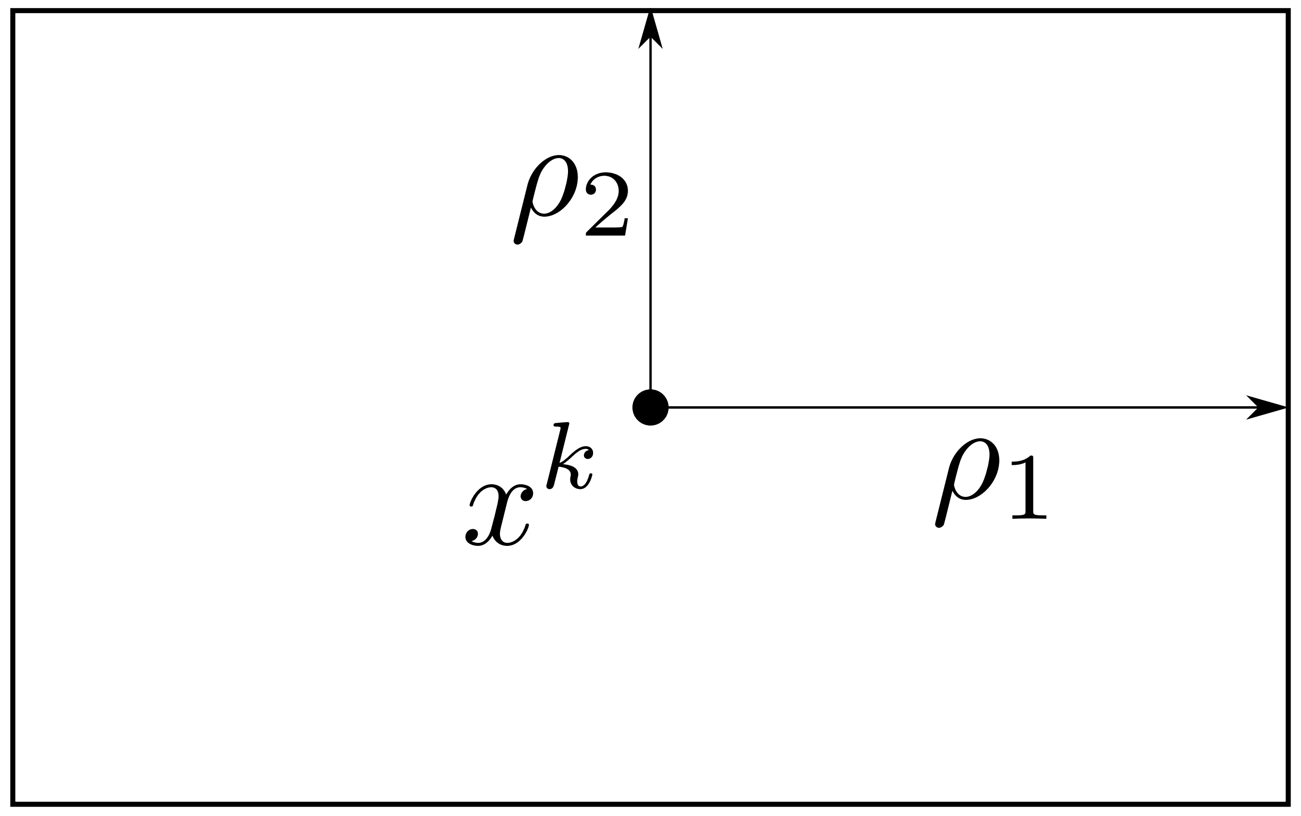
# The trust region

$$\mathcal{T}^k = \{x \mid \|x - x^k\| \leq \rho\}$$

**Ball** $\mathcal{T}^k = \{x \mid \|x - x^k\|_2 \leq \rho\}$




**Box** $\mathcal{T}^k = \{x \mid |x_i - x_i^k| \leq \rho_i\}$




**Note:** if $f$, $g_i$ $h_i$ are convex or affine in $x_i$, then we can take $\rho_i = \infty$

# Proximal operator interpretation

**trust region problem**

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & \|x - x^k\|_2 \le \rho \end{array}$$

**proximal problem**

$$\text{minimize} \quad f(x) + \frac{1}{2\lambda}\|x - x^k\|_2^2$$

**optimality conditions**

$$0 \in \partial f(x^{\mathrm{tr}}) + \mu \frac{x^{\mathrm{tr}} - x^k}{\|x^{\mathrm{tr}} - x^k\|_2},$$

$$\|x^{tr} - x^k\|_2 = \rho$$

$$\lambda = \rho/\mu \longleftrightarrow$$

**optimality conditions**

$$0 \in \partial f(x^{\mathrm{pr}}) + \frac{1}{\lambda}(x^{\mathrm{pr}} - x^k)$$

**Note:** write Lagrangian and use $\partial\|x - v\|_2 = \dfrac{x - v}{\|x - v\|_2}$

# Building convex approximations

# Convex Taylor expansions

Given nonconvex function $f$

**First order** $\hat{f}(x) = f(x^k) + \nabla f(x^k)^T (x - x^k)$

**Second order** $\hat{f}(x) = f(x^k) + \nabla f(x^k)^T (x - x^k) + (1/2)(x - x^k)^T P_+ (x - x^k)$

where $P_+ = \Pi_{\mathbf{S}_+}(\nabla^2 f(x)) = U(\mathbf{diag}(\lambda))_+ U^T$ **positive semidefinite cone projection**

**Local approximation**

it does not depend on trust-region radius $\rho$

# Quasi-linearization

Very easy and cheap method for affine approximation

write $h$ as $h(x) = A(x)x + b(x)$

$\downarrow$

use $\hat{h}(x) = A(x^k)x - b(x^k)$

**Example** $\quad f(x) = (1/2)x^T P x + q^T x = ((1/2)Px + q)^T x + r$

Quasi-linear: $\hat{x} = ((1/2)Px^k + q)^T x + r$

Taylor: $\hat{x} = h(x^k) + (Px^k + q)^T (x - x^k)$

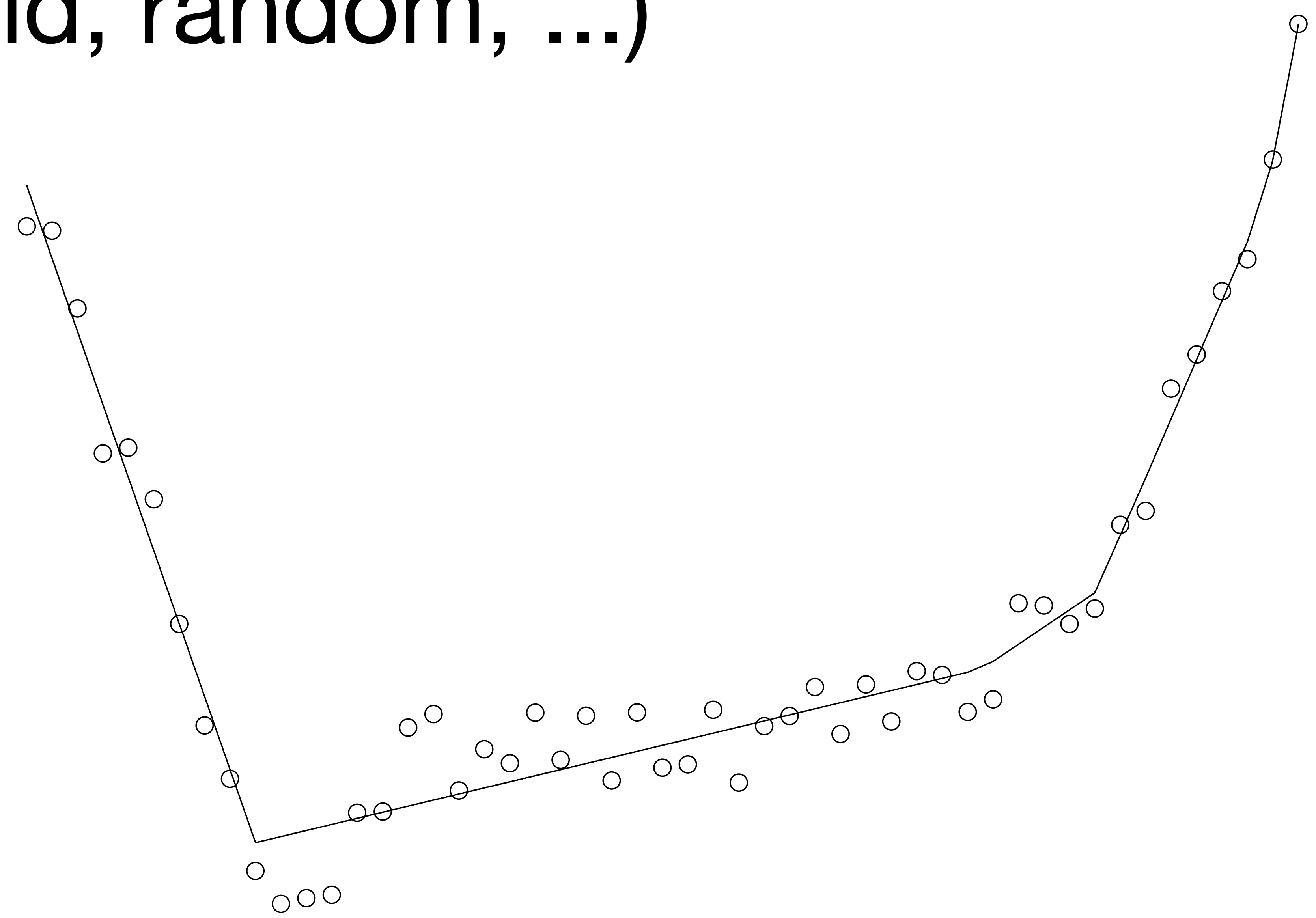
**Local approximation**

it does not depend on trust-region radius $\rho$

# Particle methods

**Idea**

- Choose points $z_1, \ldots, z_K \in \mathcal{T}^k$ (e.g., verticles, grid, random, ...)
- Evaluate function $y_i = f(z_i)$
- Fit data $(z_i, y_i)$ with convex functions (convex optimization)



**Advantages**

- Nondifferentiable functions
- **regional models**: they depend on current $x^k$ and radii $\rho_i$

# Particle methods

## Fit piecewise linear functions to data

$$\hat{f}(x) = \max_i \{\hat{y}_i + g_i^T(x - z_i)\}$$

$\hat{y}_i$ act as function values $\hat{f}(z_i)$

$g_i$ act as subgradients $\partial \hat{f}(z_i)$

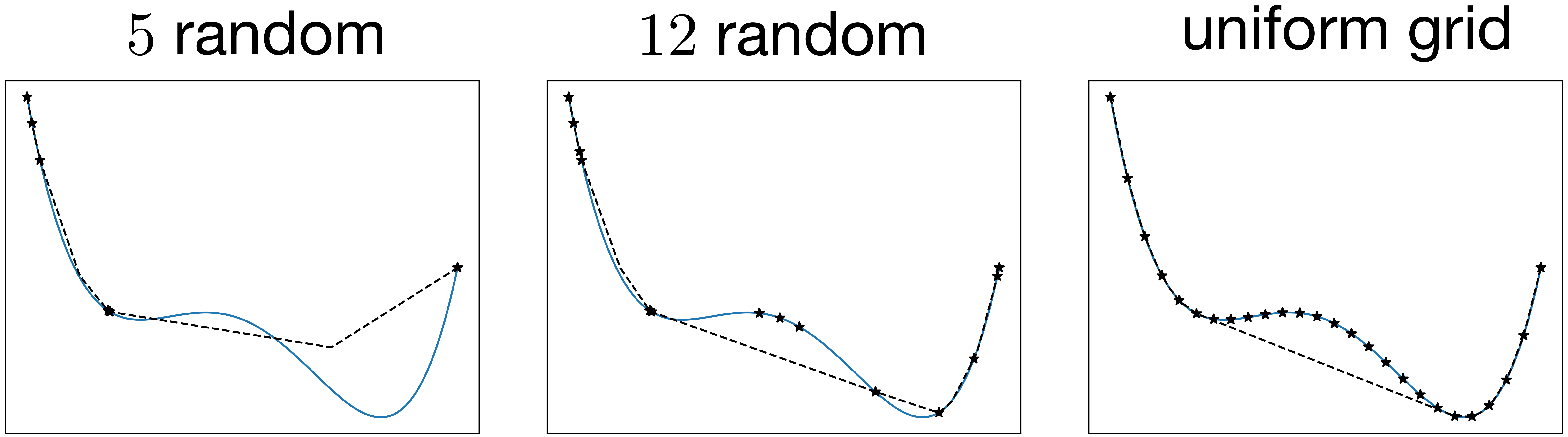
**Fitting problem**

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^K (\hat{y}_i - y_i)^2 \\ \text{subject to} & \hat{y}_j \geq \hat{y}_i + g_i^T(z_j - z_i), \quad i,j = 1,\ldots,K \\ & \hat{y}_i \leq y_i, \quad i = 1,\ldots,K \end{array}$$

**convexity** (first constraint)

**lower bound** (second constraint)

$$f(x) = x^4 - 2x^3 + 0.3x$$



# Particle methods

## Fit quadratic functions to data

$$\hat{f}(x) = (1/2)(x - x^k)^T P(x - x^k) + q^T(x - x^k) + r$$

**Fitting problem**

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{K}((1/2)(z_i - x^k)^T P(z_i - x^k) + q^T(z_i - z^k) + r - y_i)^2 \\ \text{subject to} & P \succeq 0 \end{array}$$

**Remarks**

- No necessarily upper/lower bound
- We can add other objectives, convex constraints and norm penalties
- Can be more sample efficient than piecewise linear
- Need to solve a **convex problem for every function at every SCP iteration**

# Trust region example

# Example: nonconvex quadratic program

$$
\begin{array}{ll}
\text{minimize} & f(x) = (1/2)x^T Px + q^T x \\
\text{subject to} & \|x\|_\infty \leq 1
\end{array}
$$

$P$ is symmetric but not positive semidefinite

**Taylor approximation**

$$\hat{f}(x) = f(x^k) + (Px^k + q)^T (x - x^k) + (1/2)(x - x^k)^T P_+ (x - x^k)$$

# Example: nonconvex quadratic program

## Lower bound via convex duality

$$
\begin{array}{ll}
\text{minimize} & f(x) = (1/2)x^T P x + q^T x \\
\text{subject to} & \|x\|_\infty \leq 1
\end{array}
$$

**Lagrangian**

$$
\begin{aligned}
L(x, \lambda) &= (1/2)x^T P x + q^T x + \textstyle\sum_{i=1}^n \lambda_i (x_i^2 - 1) \\
&= (1/2)x^T (P + 2\mathbf{diag}(\lambda))x + q^T x - \mathbf{1}^T \lambda
\end{aligned}
$$

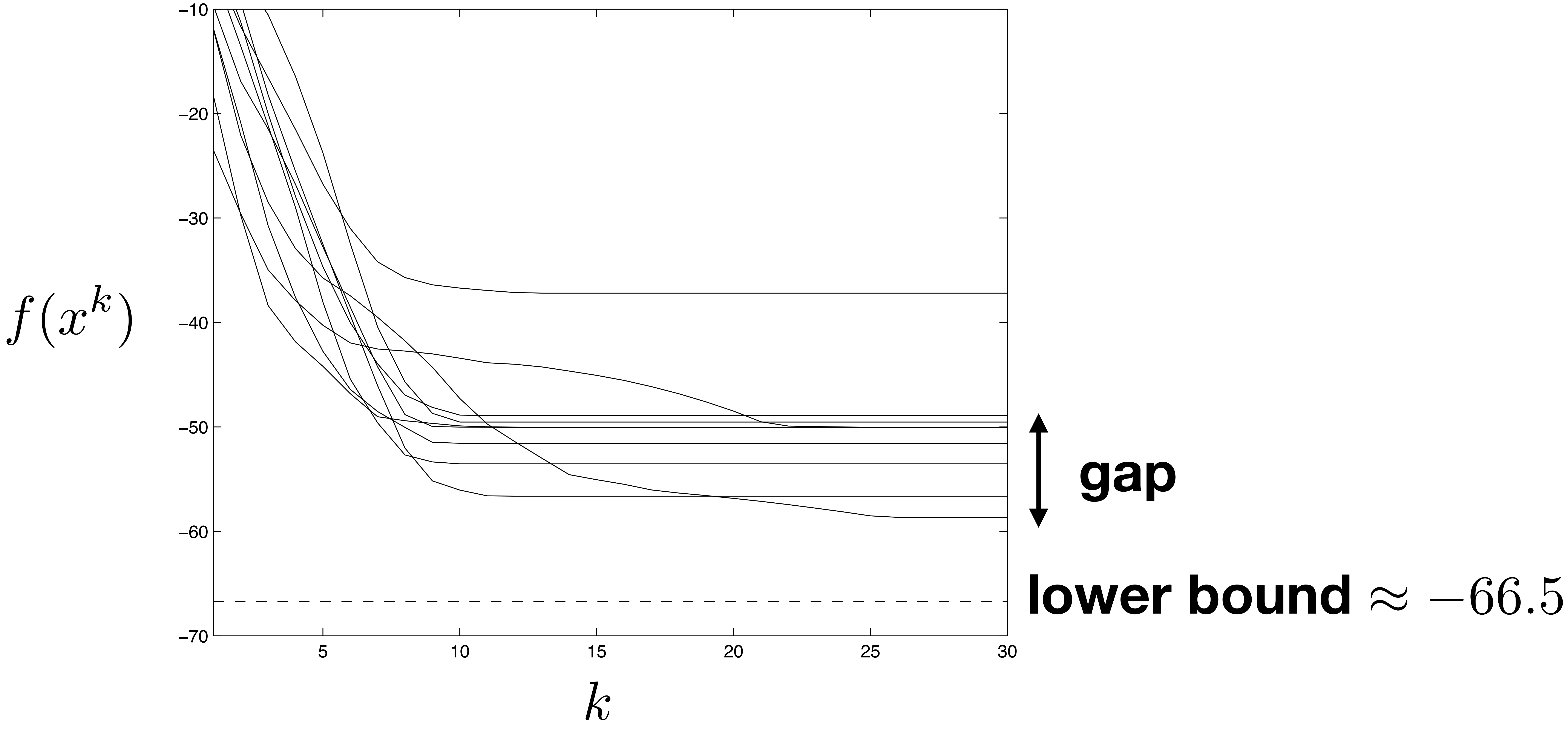
**Dual problem** (always convex)

$$
\begin{array}{ll}
\text{minimize} & -(1/2)q^T (P + 2\mathbf{diag}(\lambda))^{-1} q - \mathbf{1}^T \lambda \\
\text{subject to} & P + 2\mathbf{diag}(\lambda) \succ 0 \\
& \lambda \geq 0
\end{array}
$$

$g(\lambda)$

# Example: nonconvex quadratic program

SCP with $\rho = 0.2$ with 10 different random $x_0 \in \mathbf{R}^n$



# Regularized trust region methods

# Issues with vanilla sequential convex programming

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array} \longrightarrow \begin{array}{ll} \text{minimize} & \hat{f}(x) \\ \text{subject to} & \hat{g}_i(x) \leq 0, \quad i = 1, \ldots, m \\ & \hat{h}_i(x) = 0, \quad i = 1, \ldots, p \\ & x \in \mathcal{T}^k \end{array}$$

**Infeasibility**

Approximate problem can be infeasible (e.g. too small $\rho$)

**Evaluate progress**

when $x^k$ infeasible

- Objective: $f(x^k)$
- Inequality violations: $g_i(x^k)_+$
- Equality violations: $|h_i(x^k)|$

**Controlling trust region size**

- $\rho$ **too large**
  poor approximations $\rightarrow$ bad $x^{k+1}$
- $\rho$ **too small**
  good approximations $\rightarrow$ slow progress

# Exact penalty formulation

Solve unconstrained problem instead of the original problem

$$\text{minimize} \quad \phi(x) = f(x) + \lambda \left( \sum_{i=1}^{m} (g_i(x))_+ + \sum_{i=1}^{p} |h_i(x)| \right), \qquad \lambda > 0$$

For $\lambda$ large enough $\longrightarrow$ $x^\star = \operatorname{argmin} \phi(x)$ solves the original problem

($\lambda > \|y^\star\|_\infty$ where $y^\star$ is the dual variable satisfying the KKT conditions)

SCP solves the convex approximation (always feasible)

$$\hat{\phi}(x) = \hat{f}(x) + \lambda \left( \sum_{i=1}^{m} (\hat{g}_i(x))_+ + \sum_{i=1}^{p} |\hat{h}_i(x)| \right)$$

If $\lambda$ not large enough, we have **sparse violations**

# Trust region update

**Idea** judge progress in $\phi$ using $\hat{x} = \operatorname{argmin} \hat{\phi}(x)$

**Exact decrease**

$$\delta = \phi(x^k) - \phi(\hat{x})$$

**Approximate decrease**

$$\hat{\delta} = \phi(x^k) - \hat{\phi}(\hat{x})$$

**Updates**

$\delta \geq \alpha\hat{\delta} \longrightarrow$

- accept: $x^{k+1} = \hat{x}$
- increase region $\rho = \beta^{\mathrm{acc}}\rho$

$\delta < \alpha\hat{\delta} \longrightarrow$

- reject: $x^{k+1} = x^k$
- decrease region $\rho = \beta^{\mathrm{rej}}\rho$

**Parameters**

tolerance $\alpha$ (e.g., $= 0.1$)
accept multiplier $\beta^{\mathrm{acc}} \geq 1$ (e.g., $= 1.1$)
reject multiplier $\beta^{\mathrm{rej}} \in (0, 1)$ (e.g., 0.5)

**Interpretation**

If actual decrease $\delta$ is more than $\alpha$ fraction of predicted decrease $\hat{\delta}$ then increase trust region size (longer steps). Otherwise decrease it.

# Regularized trust region example

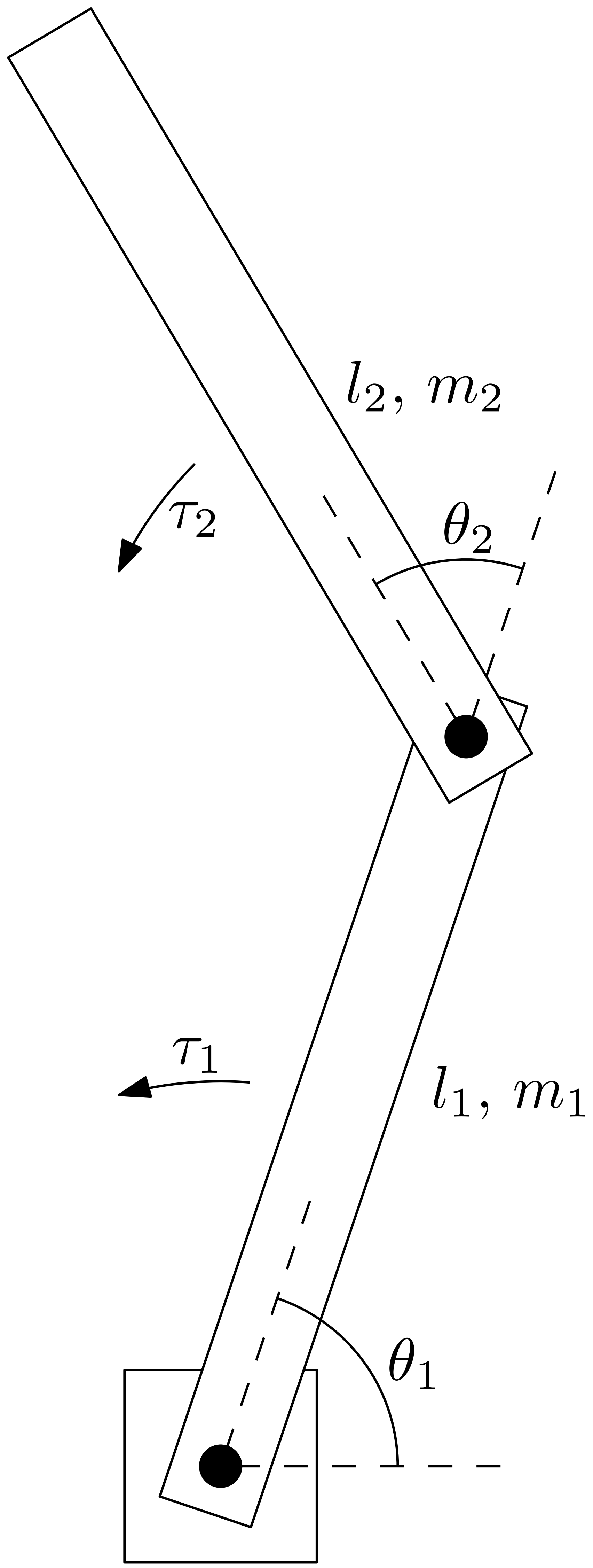
# Nonlinear optimal control

## Robotic arm

2-dimensional system

no gravity (horizontal)

controlled torques $\tau_1, \tau_2$



# Nonlinear optimal control

## The problem

minimize $J = \int_0^T \|\tau(t)\|_2^2 \mathrm{d}t$ **minimum torque**

subject to $\theta(0) = \theta_{\text{init}}, \quad \theta(T) = \theta_{\text{final}}$ **position**

$\dot{\theta}(0) = 0, \quad \dot{\theta}(T) = 0$ **velocity**

$\|\tau(t)\|_\infty \le \tau_{\max}, \quad 0 \le t \le T$

**Dynamics**

$$M(\theta)\ddot{\theta} + W(\theta, \dot{\theta})\dot{\theta} = \tau$$

$$\downarrow$$

**Not convex!**

(Hard to optimize)

**Note**: cheap to simulate

$$M(\theta) = \begin{bmatrix} (m_1 + m_2)l_1^2 & m_2 l_1 l_2 (s_1 s_2 + c_1 c_2) \\ m_2 l_1 l_2 (s_1 s_2 + c_1 c_2) & m_2 l_2^2 \end{bmatrix}$$

$$W(\theta, \dot{\theta}) = \begin{bmatrix} 0 & m_2 l_1 l_2 (s_1 c_2 - c_1 s_2)\dot{\theta}_2 \\ m_2 l_1 l_2 (s_1 c_2 - c_1 s_2)\dot{\theta}_1 & 0 \end{bmatrix}$$

where $s_i = \sin(\theta_i)$ and $c_i = \cos(\theta_i)$

# Nonlinear optimal control

## Discretization

Discretize with **time intervals** $h = T/N$

**Objective** $$J = \int_0^T \|\tau(t)\|_2^2 \mathrm{d}t \approx h \sum_{i=1}^{N} \|\tau_i\|_2^2, \quad \text{with} \quad \tau_i = \tau(ih)$$

**Dynamics**: approximate derivatives

$$M(\theta)\ddot{\theta} + W(\theta, \dot{\theta})\dot{\theta} = \tau$$

$$\dot{\theta}(ih) \approx \frac{\theta_{i+1} - \theta_{i-1}}{2h} \qquad \ddot{\theta}(ih) \approx \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2}$$

$$\theta_0 = \theta_1 = \theta_{\text{init}}$$
$$\theta_N = \theta_{N+1} = \theta_{\text{final}}$$

**nonlinear equality constraints**

$$M(\theta_i)\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} + W\left(\theta_i, \frac{\theta_{i+1} - \theta_{i-1}}{2h}\right)\frac{\theta_{i+1} - \theta_{i-1}}{2h} = \tau_i$$

# Nonlinear optimal control

## Convexification

$$
\begin{array}{ll}
\text{minimize} & h \sum_{i=1}^{N} \|\tau_i\|_2^2 \\
\text{subject to} & \theta_0 = \theta_1 = \theta_{\text{init}}, \quad \theta_N = \theta_{N+1} = \theta_{\text{final}} \\
& \|\tau_i\|_\infty \le \tau_{\max} \\
& M(\theta_i)\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} + W\left(\theta_i, \frac{\theta_{i+1} - \theta_{i-1}}{2h}\right)\frac{\theta_{i+1} - \theta_{i-1}}{2h} = \tau_i
\end{array}
$$

**Quasi-linearization of the dynamics** around previous $x^k$

$$
M(\theta_i^k)\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} + W\left(\theta_i^k, \frac{\theta_{i+1}^k - \theta_{i-1}^k}{2h}\right)\frac{\theta_{i+1} - \theta_{i-1}}{2h} = \tau_i
$$

**Remarks**

- trust region only on $\theta_i$ (cost and constraints convex in $\tau_i$)
- initialize with straight line: $\theta_i = \frac{i-1}{N-1}(\theta_{\text{final}} - \theta_{\text{init}}), \quad i = 1, \dots, N$

# Nonlinear optimal control

## Example

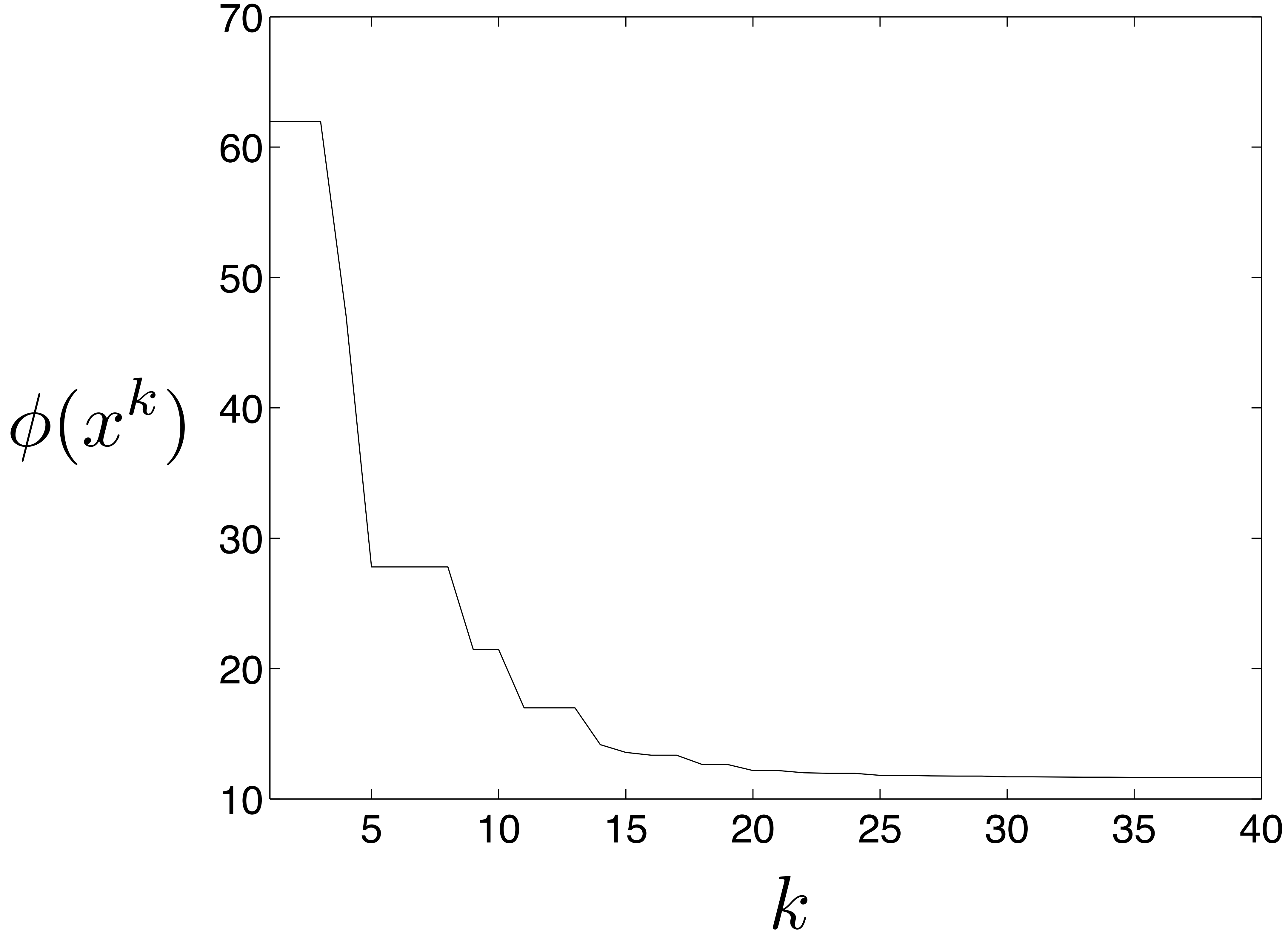
### System

- $m_1 = 1, m_2 = 5, l_1 = l_2 = 1$
- $N = 40$, $T = 10$
- $\theta_{\text{init}} = (0, -2.9), \quad \theta_{\text{final}} = (3, 2.9)$
- $\tau_{\max} = 1.1$

### Algorithm

- $\lambda = 2$
- $\alpha = 0.1$, $\beta^{\text{acc}} = 1.1$, $\beta^{\text{rej}} = 0.5$
- $\rho_1 = 90°$ (very large)

### Progress



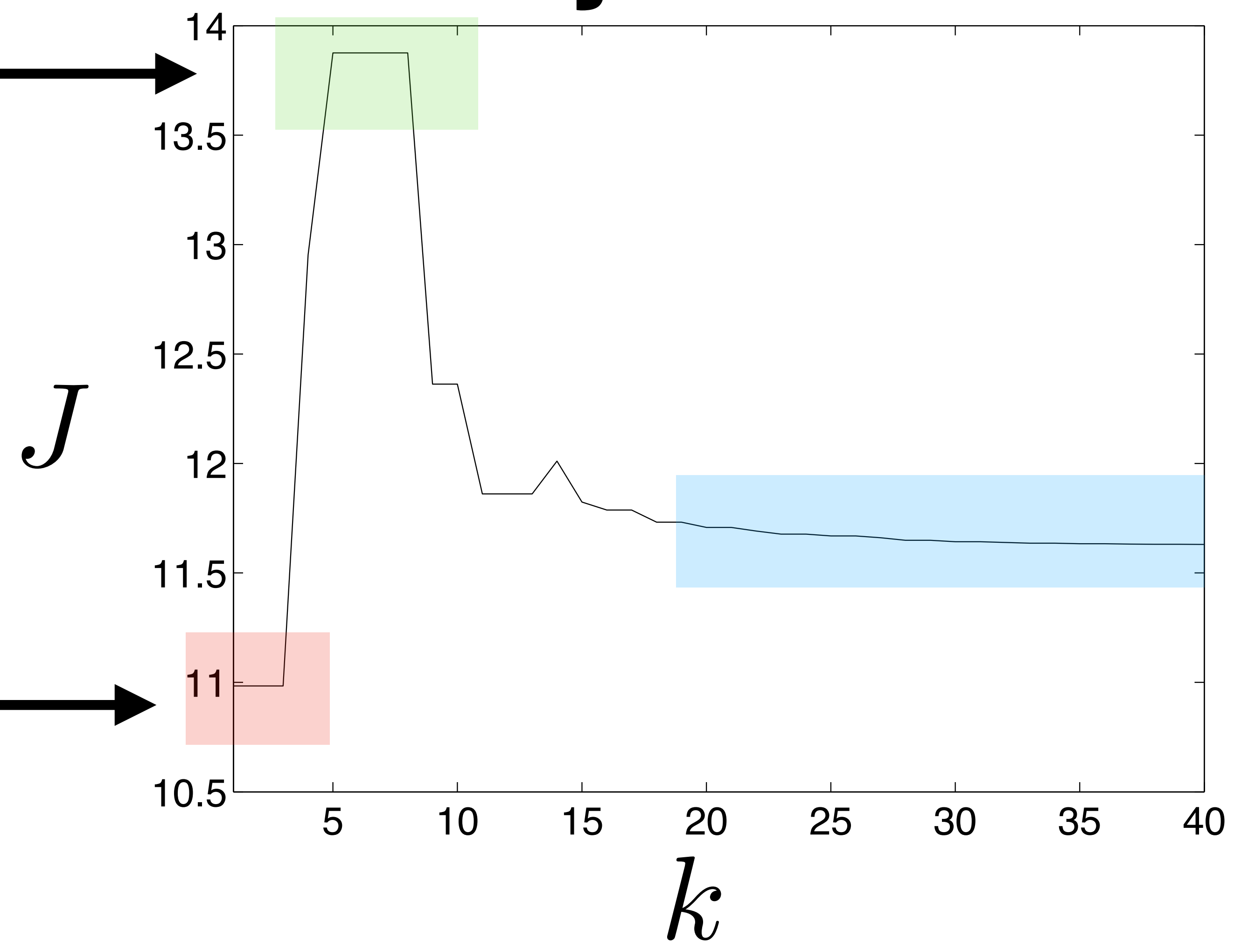

**Note:** does not go to 0

# Nonlinear optimal control

Becomes feasible

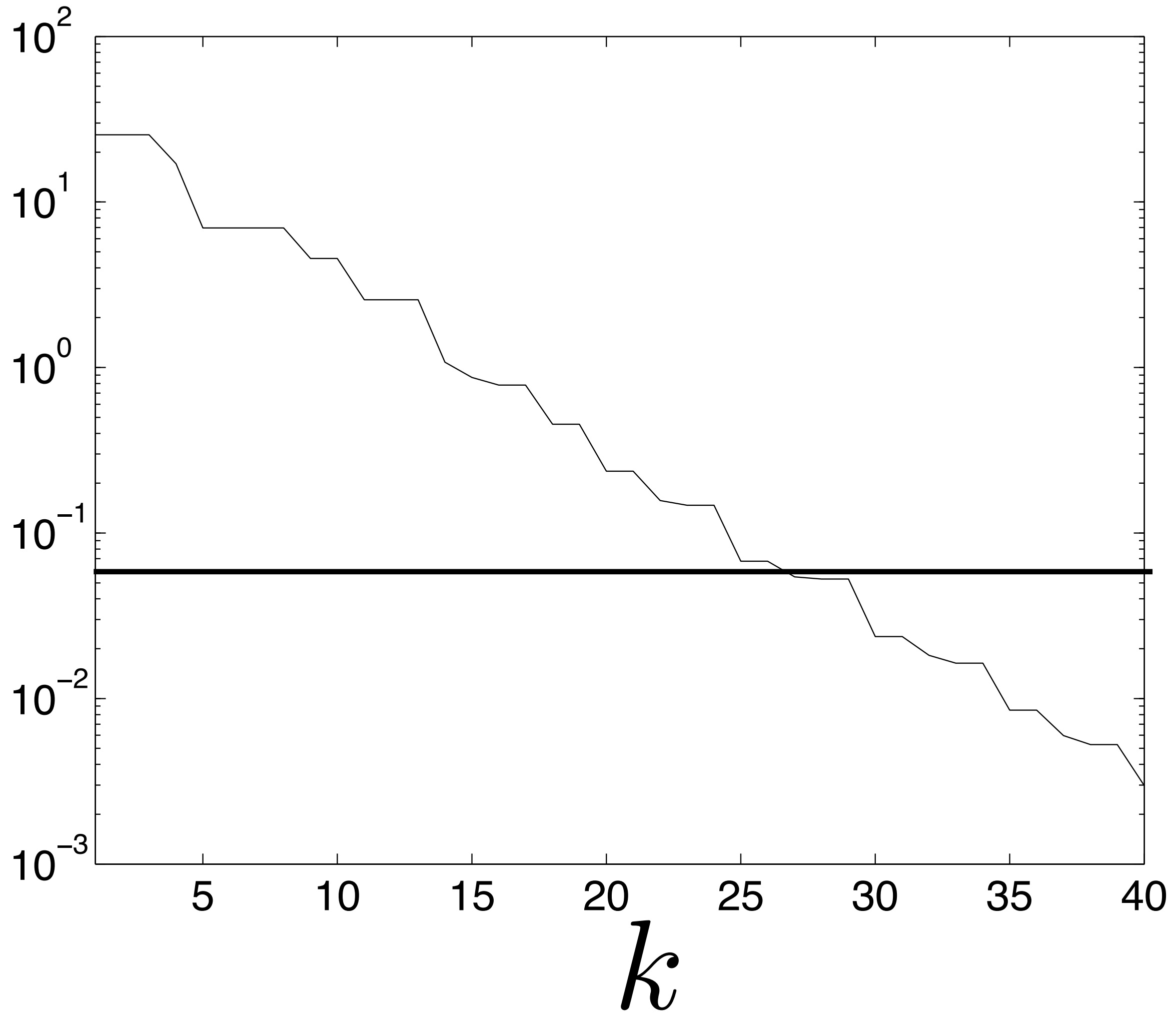
not feasible

**objective**




fine tuning

**torque residuals**

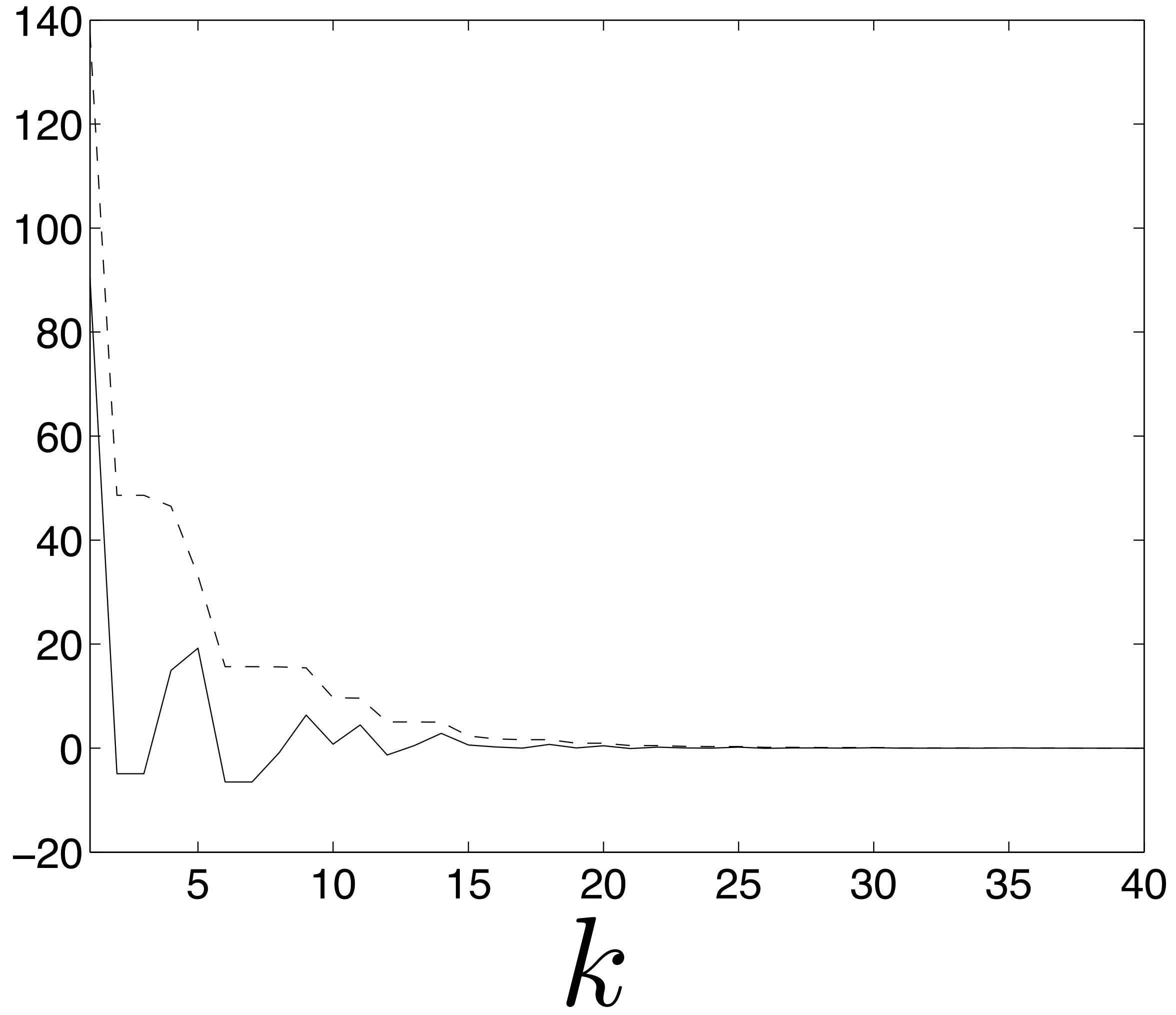



Discretization error

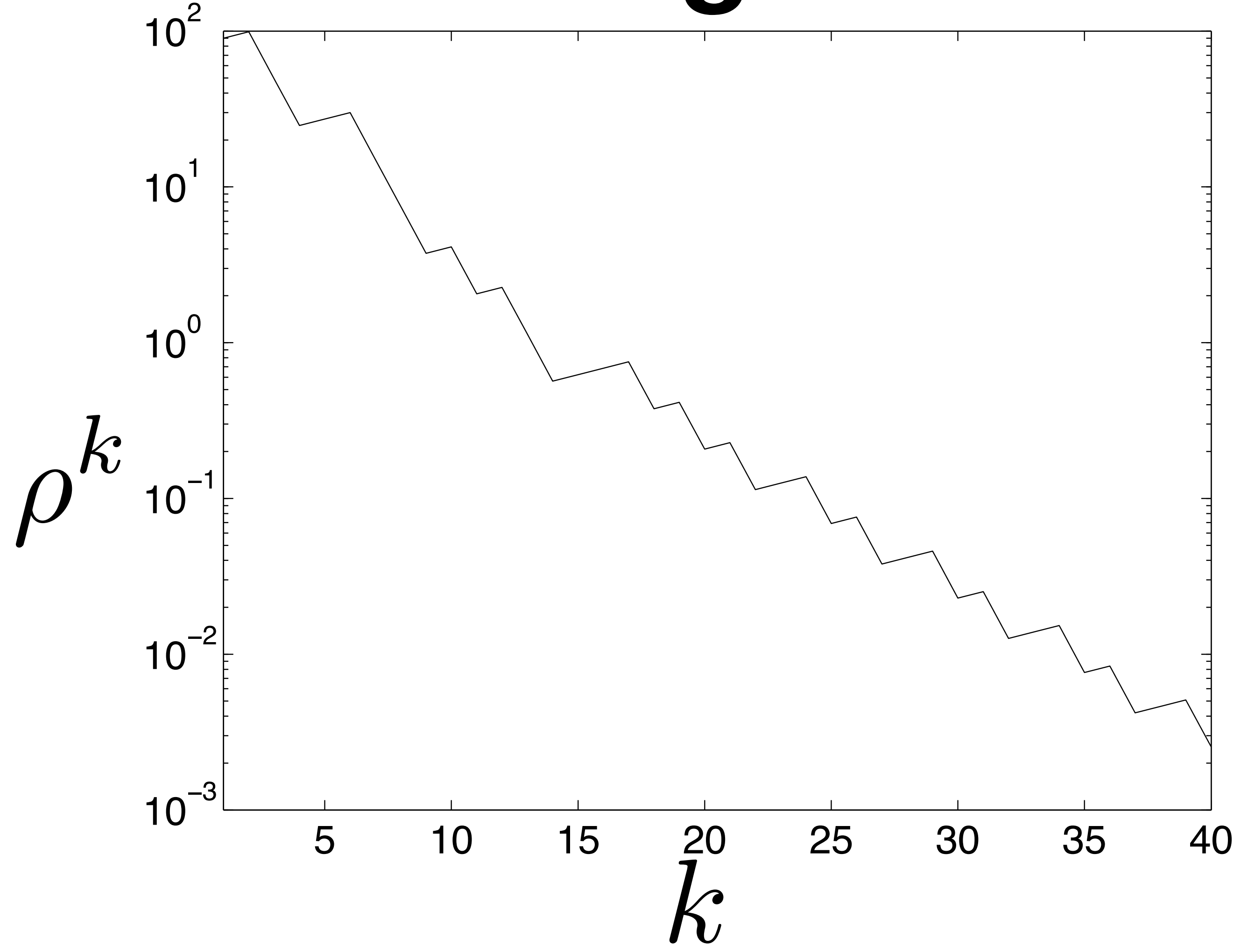
**decrease in $\phi$**

$\hat{\delta}$ : (dashed)

$\delta$ : (solid)

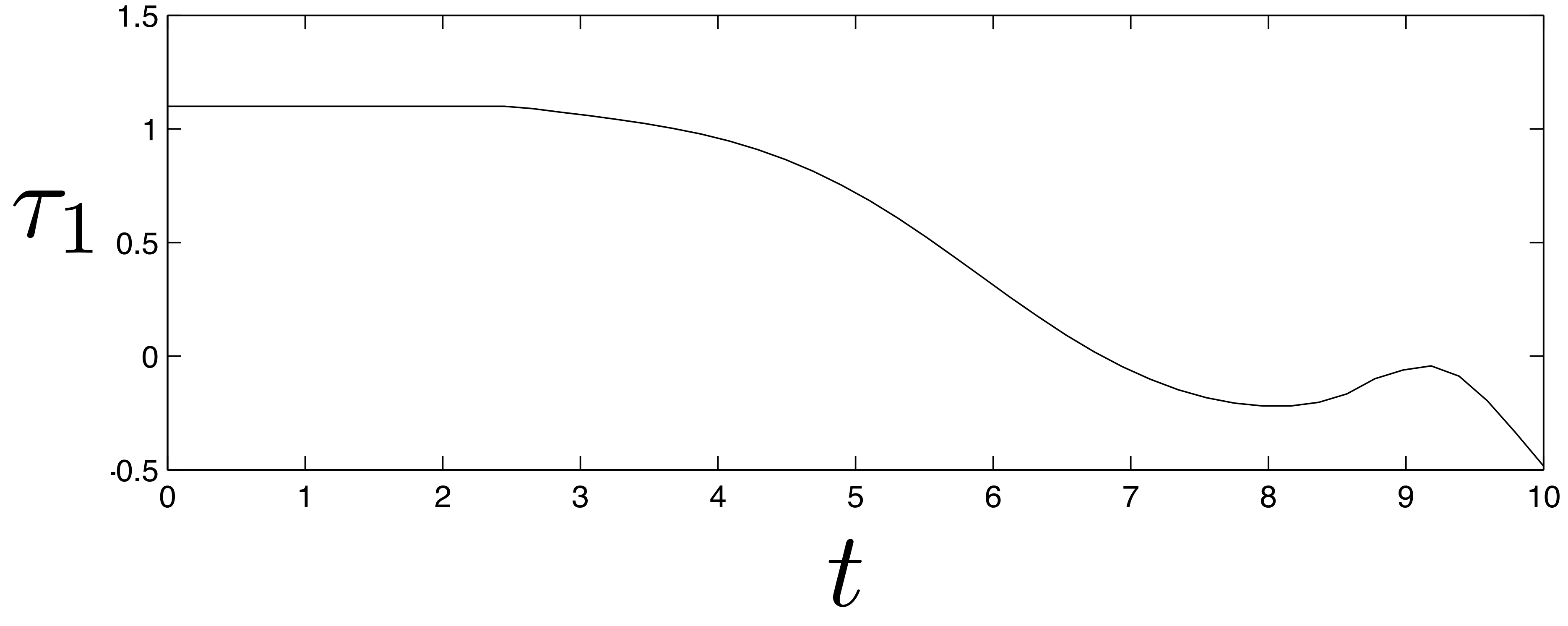


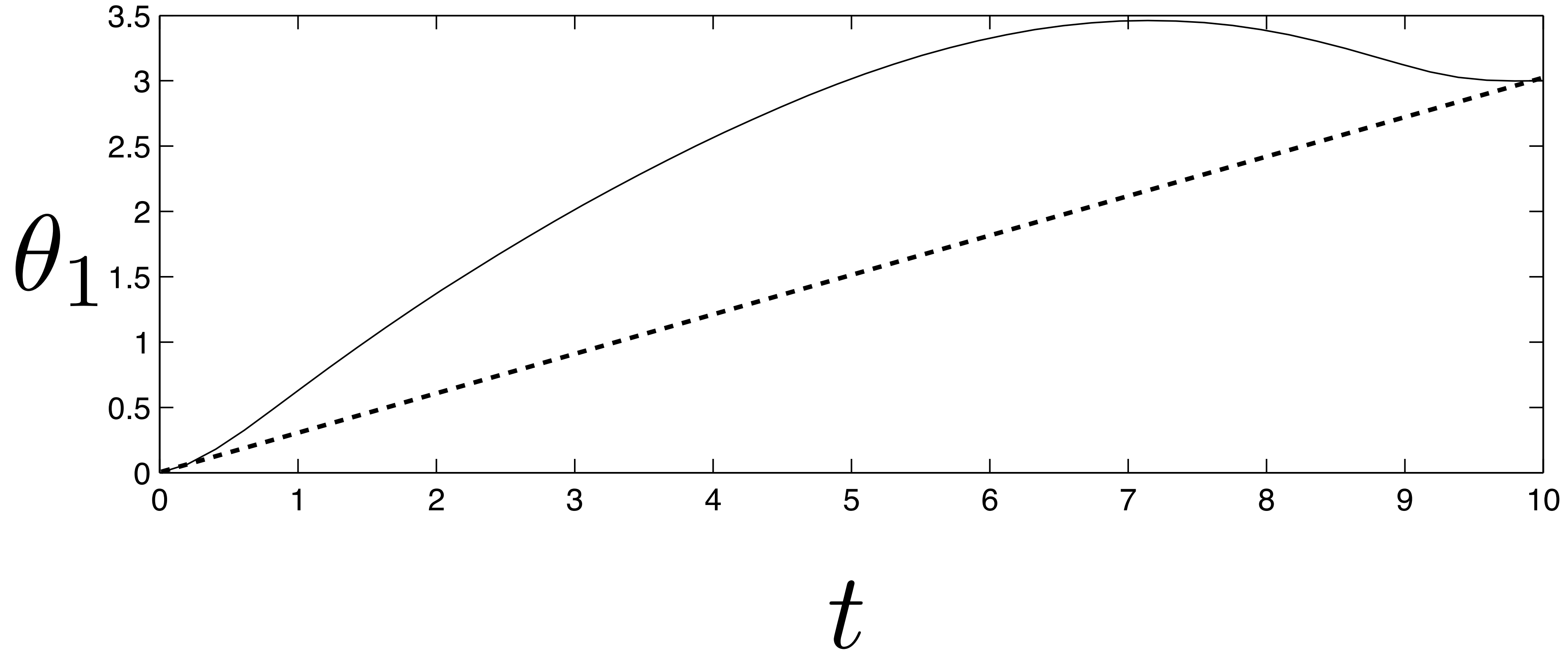

**trust region size**

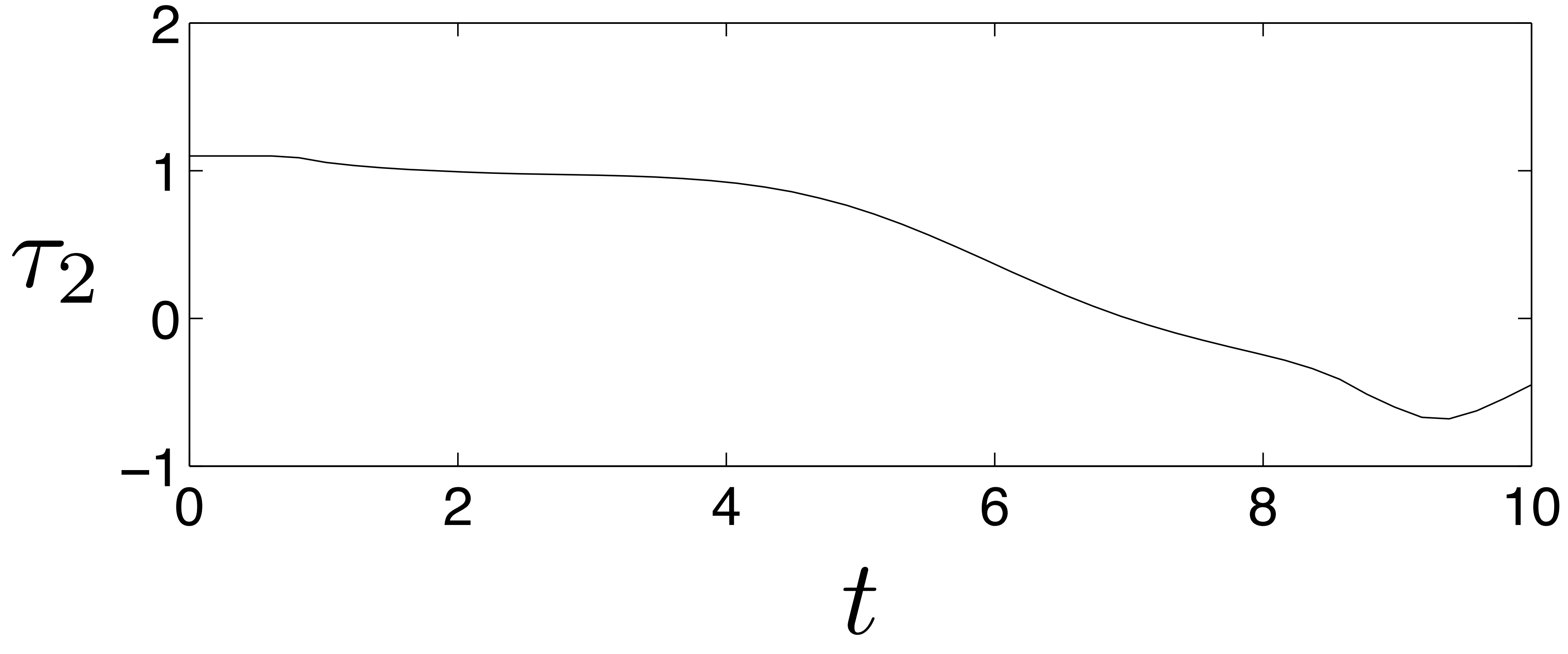


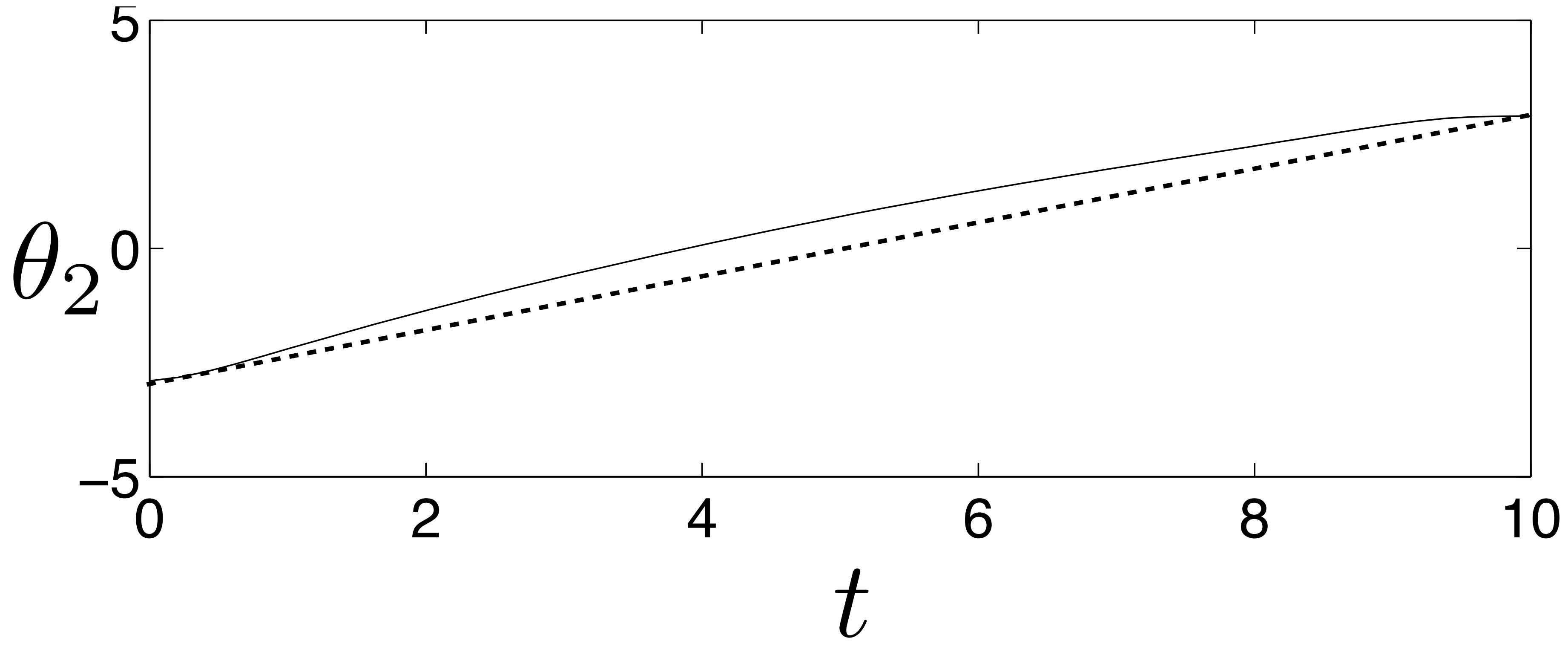
# Nonlinear optimal control

## Trajectories



# Difference of convex programming

# Difference of convex programming

$$\begin{array}{ll} \text{minimize} & \boxed{f_0(x) - g_0(x)} \\ \text{subject to} & \boxed{f_i(x) - g_i(x)} \leq 0, \quad i = 1, \dots, m \end{array}$$

**Difference of convex functions**

where $f_i$ and $g_i$ are convex

**Very powerful**

it can represent any twice differentiable function

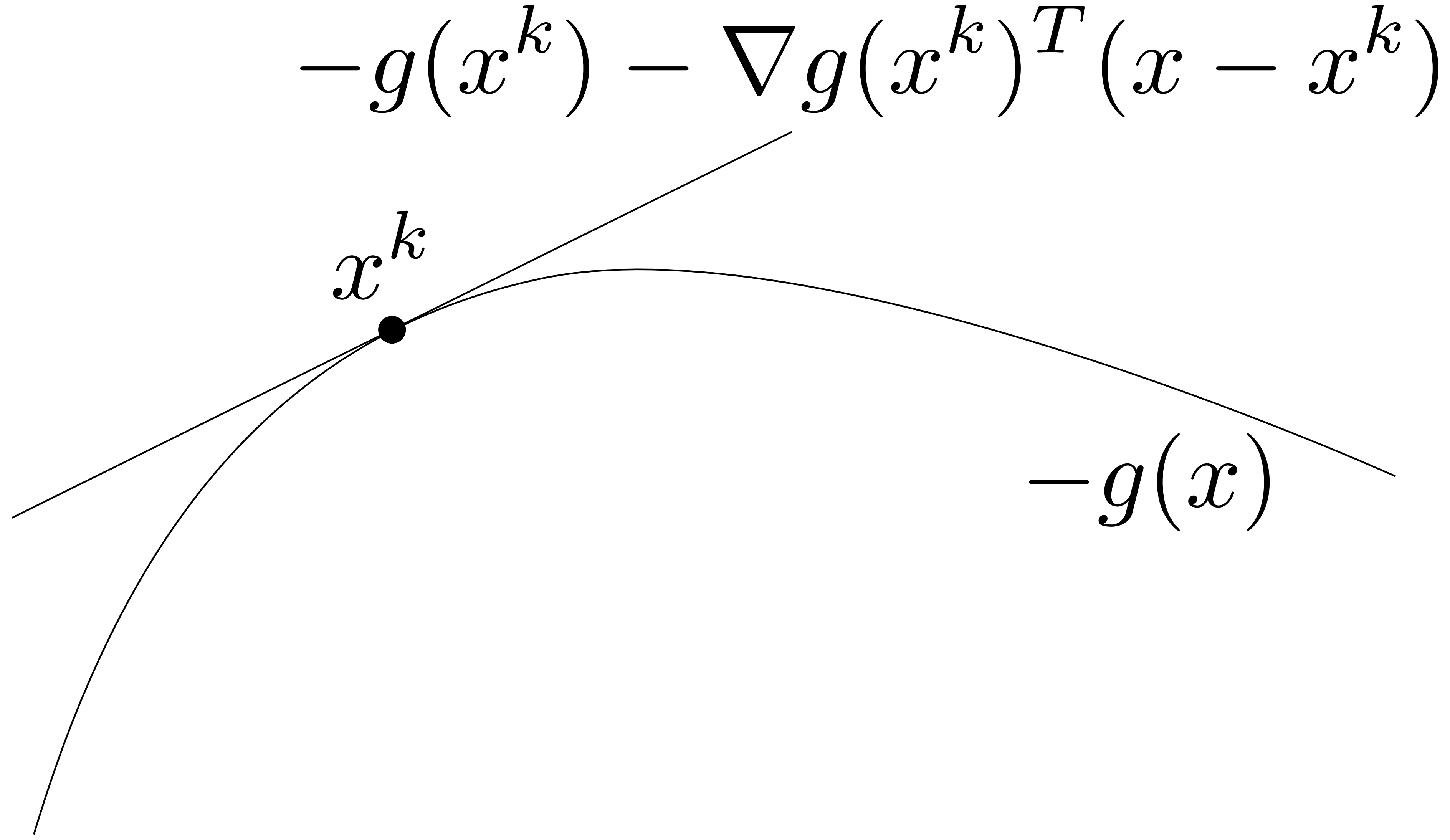
**Hard**

nonconvex problem unless $g_i$ are affine

[On Functions Representable As A Difference Of Convex Functions, Hartman]

# Difference of convex programming

## Convexification




**Convexify** $f(x) - g(x)$

$$f(x) - \hat{g}(x) = f(x) - g(x^k) - \nabla g(x^k)^T(x - x^k)$$

$$\downarrow$$

$$f(x) - g(x) \leq f(x) - \hat{g}(x)$$

**Remarks**

- True objective better than convexified objective
- True feasible set contains convexified feasible set ⟶ **No trust region needed**

# Difference of convex programming

## Iterations

**Convex-concave procedure**

1. Convexify: form $\hat{g}_i(x) = g_i(x^k) + \nabla g_i(x^k)^T(x - x^k)$ for $i = 0, \ldots, m$
2. Solve to obtain $x^{k+1}$

$$\begin{array}{ll} \text{minimize} & f_0(x) - \hat{g}_0(x) \\ \text{subject to} & f_i(x) - \hat{g}_i(x) \leq 0 \end{array}$$

**Remarks**

It always converges to a stationary point (it might be a maximum)

[Variations and extension of the convex–concave procedure, Lipp, Boyd]

# Path planning example

Find shortest path connecting $a$ and $b$ in $\mathbf{R}^d$

Avoid circles centered at $c_j$ with radius $r_j$ with $j = 1, \ldots, m$

$$
\begin{array}{ll}
\text{minimize} & L \\
\text{subject to} & x_0 = a, \quad x_n = b \\
& \|x_i - x_{i-1}\|_2 \leq L/n, \quad i = 1, \ldots, n \\
& \|x_i - c_j\|_2 \geq r_j, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m
\end{array}
$$

**path lengths** —— $\|x_i - x_{i-1}\|_2 \leq L/n$

**obstacle constraints** (not convex) —— $\|x_i - c_j\|_2 \geq r_j$

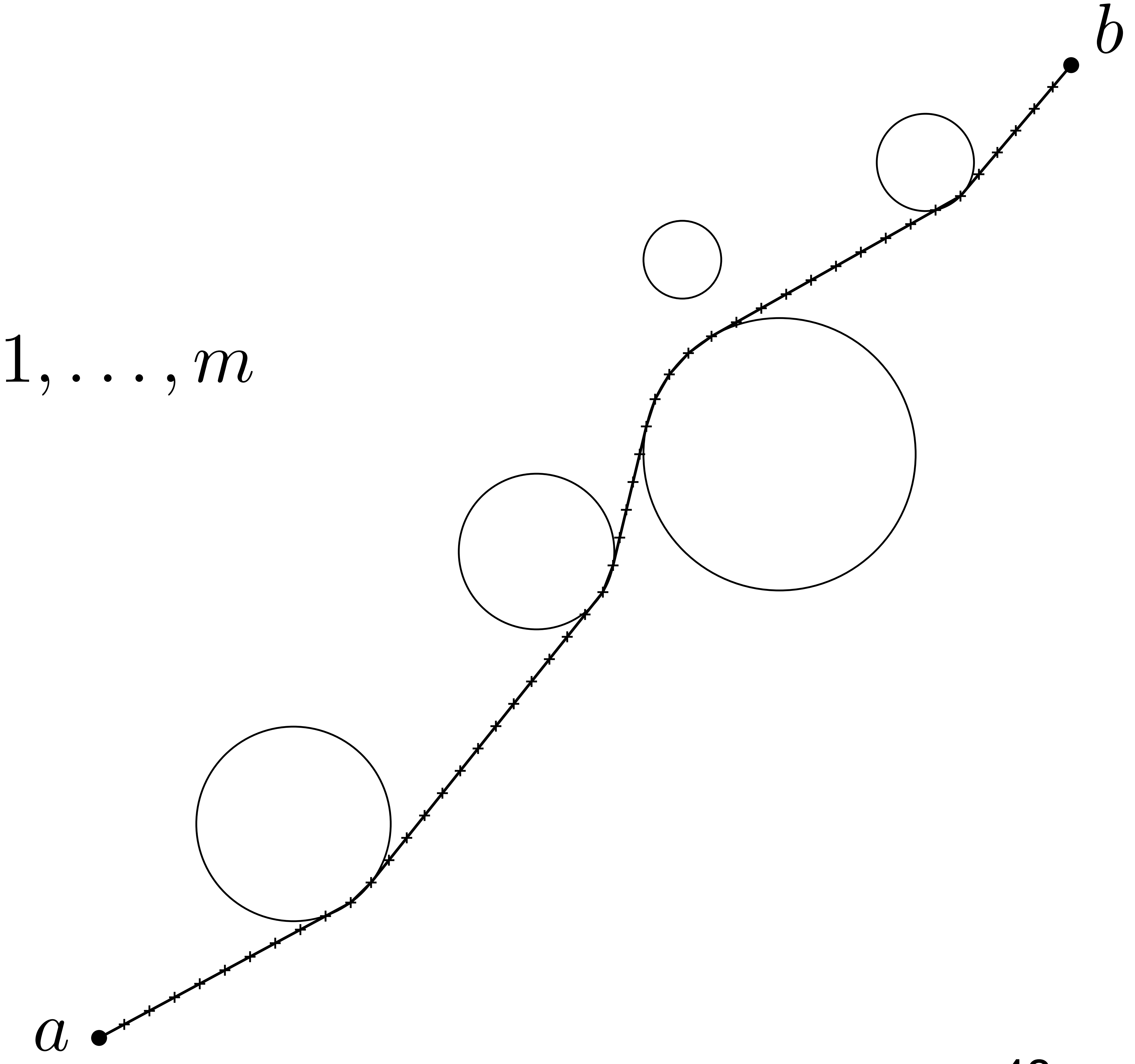
# Path planning example

minimize $\quad L$

subject to $\quad x_0 = a, \quad x_n = b$

$$\|x_i - x_{i-1}\|_2 \leq L/n, \quad i = 1, \ldots, n$$

$$\|x_i - c_j\|_2 \geq r_j, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m$$




Dimension: $d = 2$ Steps: $n = 50$

It converges in $26$ iterations (convex problems)

[Disciplined Convex-Concave Programming, Shen, Diamond, Gu, Boyd]

# Sequential convex programming

Today, we learned to:

- **Familiarize** with concepts of sequential convex programming
- **Develop** trust region algorithms
- **Build** convex approximations of nonlinear/nonsmooth functions
- **Develop** regularized trust region methods to account for infeasibility
- **Recognize** difference-of-convex programs and **apply** convex-concave procedure

# Next lecture

- Branch and bound algorithms