University of Pennsylvania, ESE 6190

Model Predictive Control

Chapter 14: Numerical Methods

Prof. Manfred Morari

Spring 2023

Coauthors: Dr. Alexander Domahidi, inspire
Dr. Stefan Richter, Richter Optimization
Prof. Melanie Zeilinger, ETH Zurich
Prof. Colin Jones, EPFL

F. Borrelli, A. Bemporad, and M. Morari, Predictive Control for Linear and Hybrid Systems, Cambridge University Press, 2017. [Ch. 3].

Outline

1. Introduction

2. Unconstrained Optimization

3. Constrained Optimization

4. Software

Outline

Recap: Convex Optimization Problems

The optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{Q} \end{array} \tag{P}$$

is said to be convex, if $f : \mathbb{R}^n \to \mathbb{R}$ and the set $\mathbb{Q}$ are convex.

Most important examples:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \tag{LP}$$

$$\begin{array}{ll} \min & \frac{1}{2} x^T H x + c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \tag{QP}$$

(convex for $H \succeq 0$)

Numerical Optimization Methods

In all but the simplest cases, an analytical solution to (P),

$$x^* \in \arg \quad \min \quad f(x)$$
$$\text{s.t.} \quad x \in \mathbb{Q}$$

cannot be obtained.

$\Rightarrow$ Numerical computation of a solution that is "good enough" by

Iterative optimization methods:

Given an initial guess x_0, produce a sequence of iterates

$$x_{i+1} = \Psi(x_i, f, \mathbb{Q}), \quad i = 0, 1, \ldots, m-1$$

such that

$$|f(x_m) - f(x^*)| \leq \epsilon \quad \text{and} \quad \text{dist}(x_m, \mathbb{Q}) \leq \delta,$$

where ϵ and δ are user defined tolerances.

Numerical Optimization Methods

$$\begin{aligned} &\textit{repeat } x_{i+1} = \Psi(x_i, f, \mathbb{Q}), \quad i = 0, 1, \ldots, m-1 \\ &\textit{until } |f(x_m) - f(x^*)| \leq \epsilon \quad \text{and} \quad \text{dist}(x_m, \mathbb{Q}) \leq \delta \end{aligned}$$

Important aspects of optimization algorithms:

- Convergence: is m finite for every $\epsilon, \delta > 0$?
- Convergence speed: dependence of errors $f(x_m) - f(x^*)$ and $\text{dist}(x_m, \mathbb{Q})$ on iteration counter
- Feasibility: for some methods $\delta = 0$, but in general $\delta \neq 0$
- Numerical robustness in presence of finite precision arithmetics
- Warm-starting: can the method take advantage of x_0 being close to x^*?
- Preconditioning: equivalence transformation of (P) into a similar problem (P') that can be solved in fewer iterations?

Outline

Outline

Unconstrained Optimization Using Gradient Information

Goal: Solve the **unconstrained** (i.e. $\mathbb{Q} = \mathbb{R}^n$) problem

$$\text{minimize} \quad f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable.

Idea: Gradient ∇f gives direction of steepest local ascent
$\Rightarrow$ Make steps of size h_i into anti-gradient direction $-\nabla f$:

$$\boxed{x_{i+1} = x_i - h_i \nabla f(x_i)} \tag{1}$$

Question: How to choose the step sizes h_i?

A Useful Reformulation of the Gradient Update

Gradient update:

$$\boxed{x_{i+1} = x_i - h_i \nabla f(x_i)}$$

Idea: This update rule results from minimizing a quadratic function:

$$x_{i+1} = \arg\min_x f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2h_i} \|x - x_i\|^2$$

$$\Leftrightarrow \nabla_x \left(f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2h_i} \|x - x_i\|^2 \right) \Big|_{x=x_{i+1}} = 0$$

$$\Leftrightarrow \nabla f(x_i) + \frac{1}{h_i}(x_{i+1} - x_i) = 0$$

$$\Leftrightarrow x_{i+1} = x_i - h_i \nabla f(x_i)$$

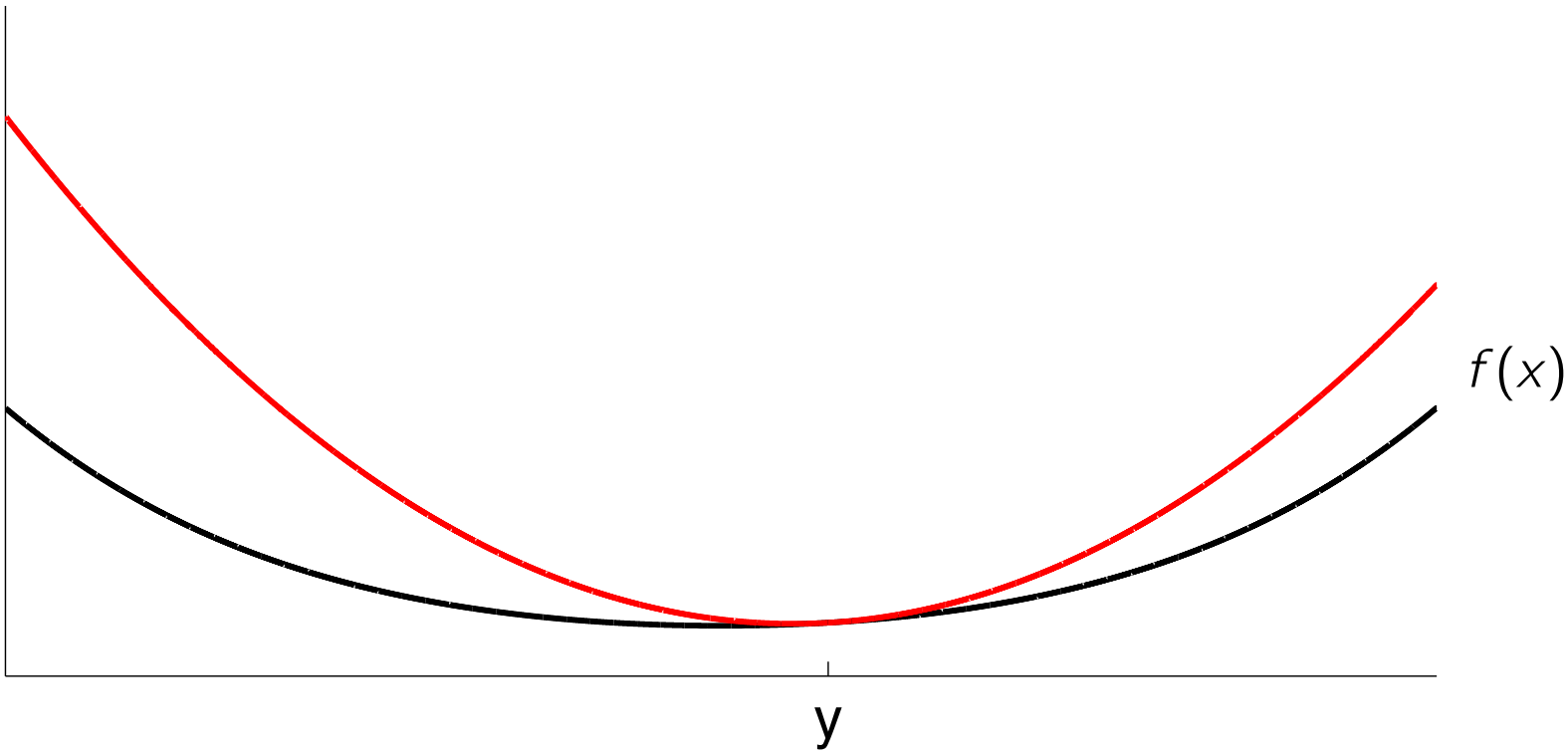
L-smoothness

Assumption: $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \; \forall x, y \in \mathbb{R}^n$

$\Leftrightarrow \nabla f$ is Lipschitz-continuous with constant L

$\Leftrightarrow f$ can be upper bounded by a quadratic function:

$$f(x) \leq f(y) + \nabla f(y)^T(x - y) + \tfrac{L}{2}\|x - y\|^2 \; \forall x, y \in \mathbb{R}^n$$



Example: Quadratic upper bound on quadratic functions

Consider

$$f(x) = \frac{1}{2} x^T H x + c^T x$$

with $H \succeq 0$. By an eigenvalue decomposition $H = V^T \Lambda V$ and $\|V\| = 1$:

$$f(x) = \frac{1}{2} x^T V^T \Lambda V x + c^T x \leq \frac{L}{2} \|x\|_2^2 + c^T x$$

where

$$L = \max \operatorname{diag}(\Lambda) = \max \operatorname{eig}(H)$$

The Gradient Method Converges with a Constant Step Size

Gradient update $\Leftrightarrow$ minimization of quadratic function:

$$\begin{aligned} x_{i+1} &= x_i - h_i \nabla f(x_i) \\ &= \arg\min_x f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2h_i} \|x - x_i\|^2 \end{aligned}$$

L-smoothness $\Leftrightarrow$ f can be upper bounded by quadratic function:

$$f(x) \leq f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{L}{2} \|x - x_i\|^2 \triangleq \bar{f}(x, x_i)$$

Results:

1. Constant step size: $\boxed{h_i = \frac{1}{L}}$
2. Convergence: $f(x_{i+1}) \leq \bar{f}(x_{i+1}, x_i) \leq \bar{f}(x_i, x_i) = f(x_i)$

Gradient Method - Summary

Problem: minimize $f(x)$

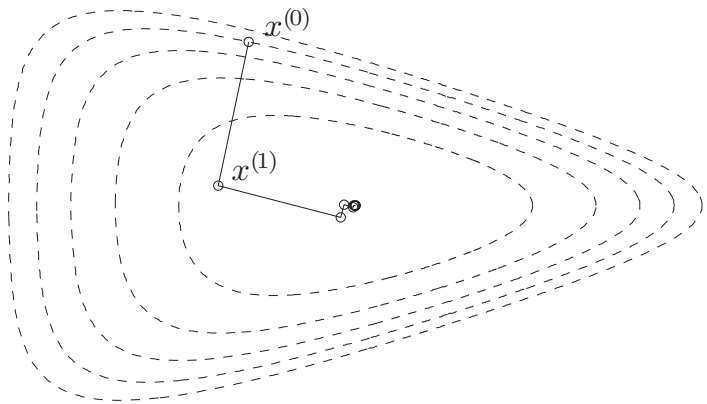
Requirements: f is L-smooth and convex

Gradient method:

$$\begin{aligned} \text{Set} \quad & x_0. \\ \textit{Repeat} \quad & x_{i+1} = x_i - \tfrac{1}{L}\nabla f(x_i) \quad \text{for } i = 0, \ldots, m-1 \\ \textit{until} \quad & f(x^m) - f(x^*) \leq \epsilon_1 \textit{ or } \|x_m - x_{m-1}\| \leq \epsilon_2. \end{aligned}$$

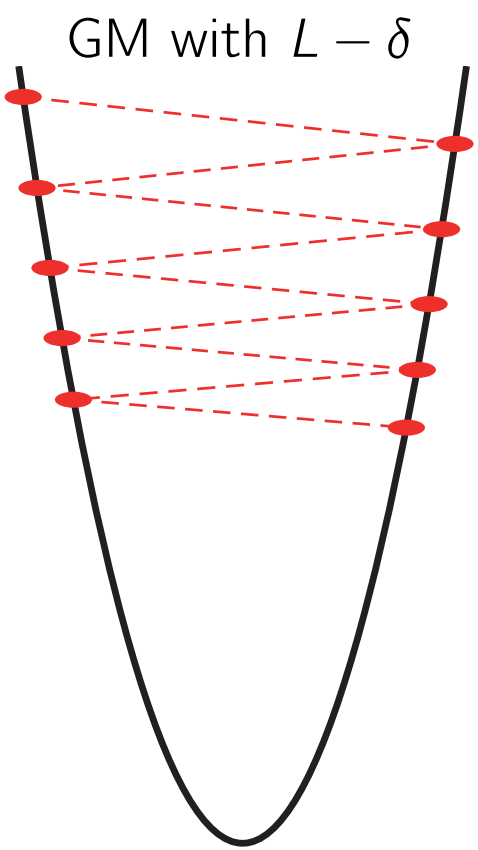
Convergence: The GM converges with $m \sim \mathcal{O}\left(L\|x_0 - x^*\|^2/\epsilon_1\right)$.

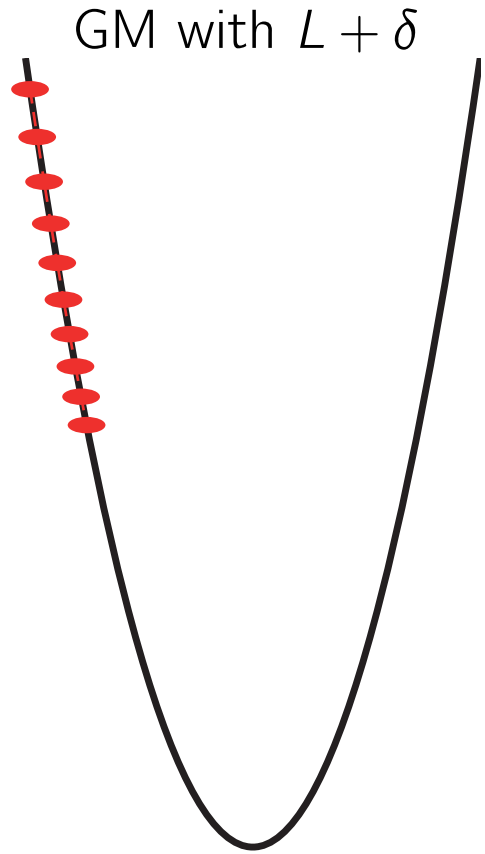
Example: $\min e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$



[Boyd & Vandenberghe, 2004]

The Effect of Inexact L





Inexact information on L yields slower convergence. Convergence can even be lost if $L < L_{\text{true}}/2$.

Figures by courtesy of Dr. Michel Baes, IFOR, ETH Zurich.

Fast Gradient Method

Problem: minimize $f(x)$

Requirements: f is L-smooth and convex

Gradient method:

Set $x_0, y_0 = x_0$ and $\alpha_0 = (\sqrt{5}-1)/2$.

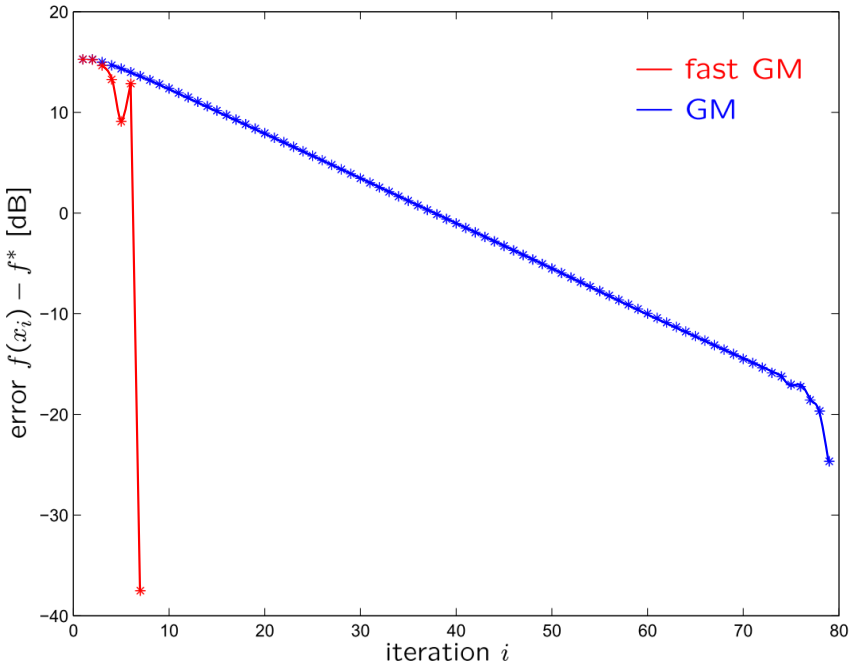
Repeat
$$x_{i+1} = y_i - \frac{1}{L}\nabla f(y_i)$$
$$\alpha_{i+1} = \alpha_i\left(\sqrt{\alpha_i^2+4} - \alpha_i\right)/2$$
$$\beta_i = \frac{\alpha_i(1-\alpha_i)}{\alpha_i^2+\alpha_{i+1}}$$
$$y_{i+1} = x_{i+1} + \beta_i(x_{i+1} - x_i) \quad \text{for } i = 0, \ldots, m$$

until $f(x_m) - f(x^*) \leq \epsilon_1$ *or* $\|x_m - x_{m-1}\| \leq \epsilon_2$.

Convergence: The fast gradient method converges with $m \sim \mathcal{O}\left(\sqrt{L\|x_0 - x^*\|^2/\epsilon_1}\right)$ (best convergence rate for this type of problems and using only first-order information).

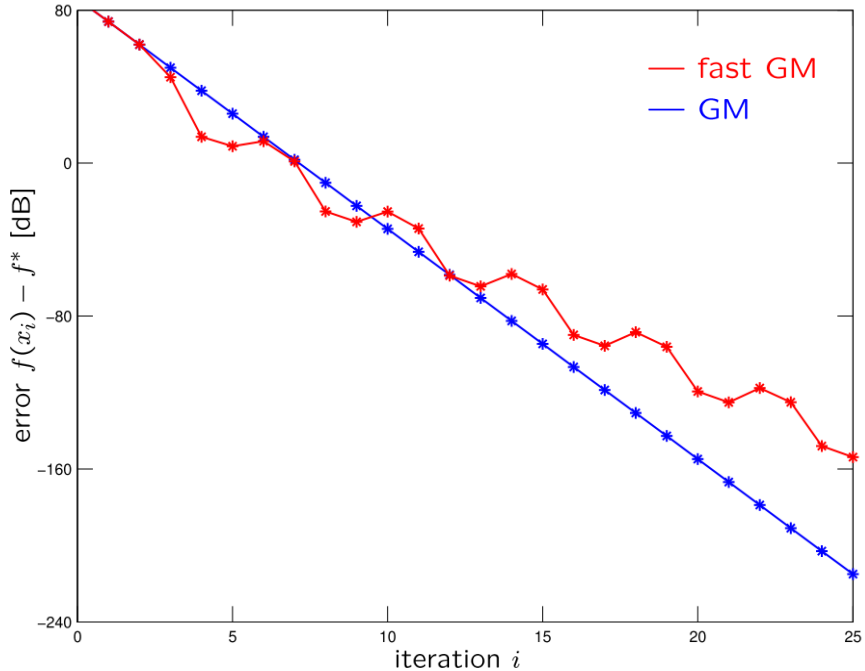
Comparison GM und FGM

FGM **often** faster than GM...



... but not always.

Example: $\min(x_1 - x_2)^2$

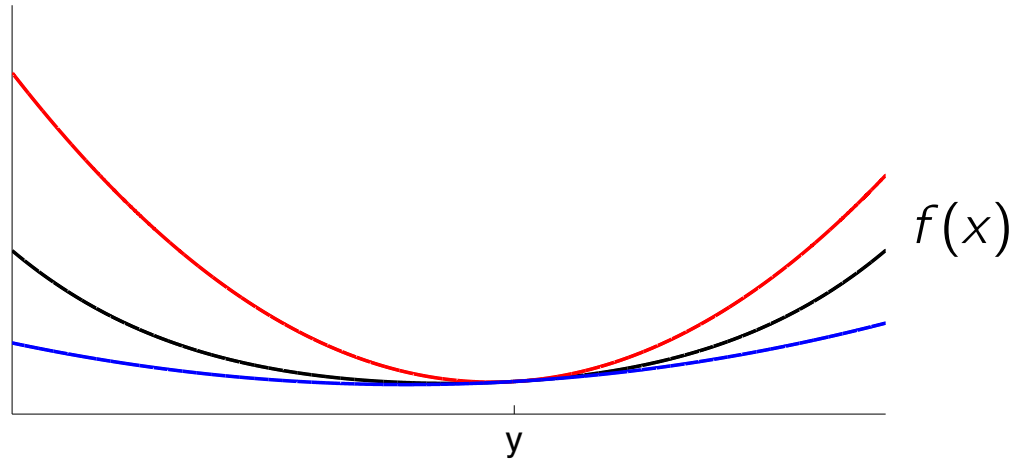


(for step size $h_i = 1/(2L)$)

Figures by courtesy of Dr. Michel Baes, IFOR, ETH Zurich.

Additional property: strong convexity

$$f(x) \leq f(y) + \nabla f(y)^T (x-y) + \tfrac{L}{2}\|x-y\|^2 \;\; \forall x, y \in \mathbb{R}^n$$



$$f(x) \geq f(y) + \nabla f(y)^T (x-y) + \tfrac{\mu}{2}\|x-y\|^2 \;\; \forall x, y \in \mathbb{R}^n$$

Condition number: $\boxed{\kappa \triangleq L/\mu} \geq 1$

Consequences:

- Gradient method: $m \sim \mathcal{O}\left(\kappa \|x_0 - x^*\|^2 \ln \frac{1}{\epsilon_1}\right)$
  (no change in algorithm required)
- Fast gradient method: $m \sim \mathcal{O}\left(\sqrt{\kappa}\|x_0 - x^*\|^2 \ln \frac{1}{\epsilon_1}\right)$
  (change in algorithm required, e.g. by restarting y_i, α_i, β_i)

Example: Quadratic lower bound on quadratic functions

Consider

$$f(x) = \frac{1}{2} x^T H x + c^T x$$

with $H \succ 0$. By an eigenvalue decomposition $H = V^T \Lambda V$ and $\|V\| = 1$:

$$f(x) = \frac{1}{2} x^T V^T \Lambda V x + c^T x \geq \frac{\mu}{2} \|x\|_2^2 + c^T x$$

where

$$\mu = \min \operatorname{diag}(\Lambda) = \min \operatorname{eig}(H) > 0$$

Preconditioning

Observation: Convergence speed depends on condition number κ

Idea: Variable transformation to improve κ:

$$x = Py$$

with P invertible. Then:

$$\min_y h(y) = \min_y f(Py)$$

but, in general, $\kappa_h \neq \kappa_f$ (want P such that $\kappa_h < \kappa_f$).

For some function classes, optimal pre-conditioners (i.e. $\kappa_h = 1$) exist.

Example: $f(x) = 1/2x^T Hx, H \succ 0 \Rightarrow P^* = H^{-1/2}$.

Aspects of Gradient Methods

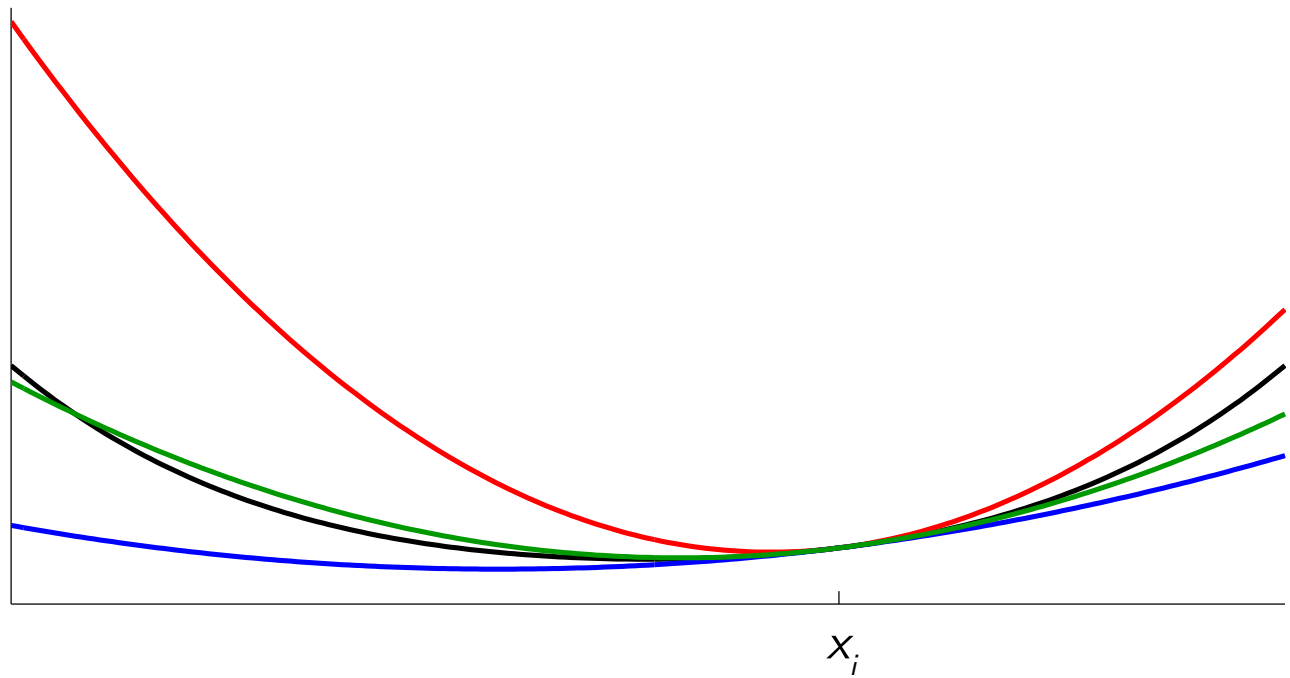
- Convergence: is m finite for every $\epsilon, \delta > 0$? ✓ (globally)
- Convergence speed: dependence of errors $f(x_m) - f(x^*)$ and $\text{dist}(x_m, \mathbb{Q})$ on iteration counter ✓ (globally)
- Numerical robustness in presence of finite precision arithmetics
  - Gradient method: ✓
  - Fast gradient method: can diverge due to numerical errors
- Warm-starting: can the method take advantage of x_0 being close to x^*? ✓
- Preconditioning: equivalence transformation of (P) into a similar problem (P') that can be solved in fewer iterations? ✓
- Each iteration computationally cheap (matrix-vector multiplication for QPs)

Outline

Newton's Method

Idea: Minimize second-order approximation of f at current iterate x_i:

$$x_{i+1} = \arg\min_x \underbrace{f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2}(x - x_i)^T \nabla^2 f(x_i)(x - x_i)}_{\triangleq \tilde{f}(x, x_i)}$$



Note: $\tilde{f}$ is not necessarily an upper bound on f

Newton's Method

Idea: Minimize second-order approximation of f at current iterate x_i:

$$x_{i+1} = \arg\min_x f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2}(x - x_i)^T \nabla^2 f(x_i)(x - x_i)$$

$$\nabla_x \left(f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2}(x - x_i)^T \nabla^2 f(x_i)(x - x_i) \right) \Big|_{x = x_{i+1}} = 0$$

$$\Leftrightarrow \nabla f(x_i) + \nabla^2 f(x_i)(x_{i+1} - x_i) = 0$$

$$\Leftrightarrow x_{i+1} = x_i \underbrace{- \left(\nabla^2 f(x_i)\right)^{-1} \nabla f(x_i)}_{\text{Newton direction } \Delta x_{nt}}$$

Since $\tilde{f}$ is not an upper bound on f, full Newton step does not necessarily yield descent (i.e. $f(x_{i+1}) > f(x_i)$ might occur)

Idea: Use step size $h_i > 0$ such that Newton step yields descent

$$\text{Newton step: } \boxed{x_{i+1} = x_i - h_i \left(\nabla^2 f(x_i)\right)^{-1} \nabla f(x_i)} \tag{2}$$

Line Search

Newton step: $\boxed{x_{i+1} = x_i + h_i \Delta x_{nt}}$

Problem: Find $h_i > 0$ s.t. $f(x_i + h_i \Delta x_{nt}) < f(x_i)$

Line search (LS) methods:

- **Exact**: Compute **best** h_i:

$$h_i^* = \arg\min_{h>0} f(x_i + h_i \Delta x_{nt})$$

  Optimization in 1 variable $\rightarrow$ solve by bisection
  Time consuming (requires many evaluations of f)

- **Inexact**: Find h_i that decreases f by some per cent. Example: **Backtracking**[1] line search. For $\alpha \in (0,\, 0.5)$ and $\beta \in (0,\, 1)$:
  Initialize $h_i = 1$.
  while $f(x_i + h_i \Delta x_{nt}) > f(x_i) + \alpha h_i \nabla f(x_i)^T \Delta x_{nt}$ **do** $h_i \leftarrow \beta h_i$

[1] More details in e.g. *[Boyd & Vandenberghe, Convex Optimization, 2004]*

Details on Backtracking Line Search

Backtracking line search terminates when *Armijo's condition* is satisfied:

$$f(x_i + h_i \Delta x_{nt}) \leq f(x_i) + \alpha h_i \nabla f(x_i)^T \Delta x_{nt}$$

This is required for convergence of the optimization algorithm.
From *[Boyd & Vandenberghe, Convex Optimization, 2004]* with $t \equiv h, x \equiv x_i$:

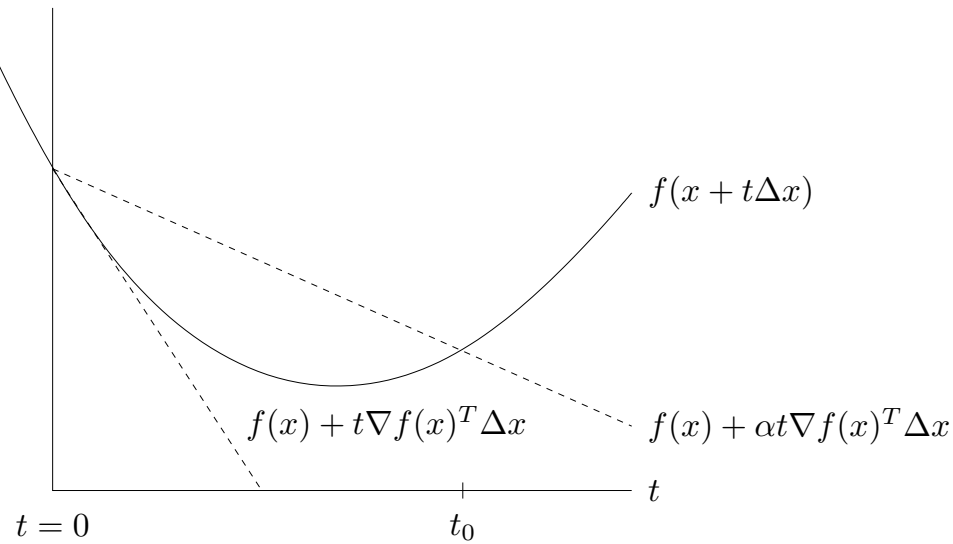


Figure 9.1 *Backtracking line search.* The curve shows f, restricted to the line over which we search. The lower dashed line shows the linear extrapolation of f, and the upper dashed line has a slope a factor of α smaller. The backtracking condition is that f lies below the upper dashed line, *i.e.*, $0 \leq t \leq t_0$.

Equality constraints in Newton's method

Consider the equality constrained problem (with matrix $A \in \mathbb{R}^{m \times n}$)

$$\text{minimize } f(x)$$
$$\text{subject to } Ax = b$$

Newton: Minimize quadratic model of f around x_i to obtain descent direction:

$$\Delta x_{nt}(x_i) \in \arg\min_{\Delta x} \frac{1}{2}\Delta x^T \nabla^2 f(x_i)\Delta x + \nabla f(x_i)\Delta x \quad (**)$$
$$\text{s.t. } A\Delta x = 0$$

Equality constraint demands $\Delta x \in \text{null}(A)$, which yields the induction

$$Ax_i = b \Rightarrow Ax_{i+1} = Ax_i + h_i A\Delta x_{nt}(x_i) = b + 0 \quad \forall h_i$$

Hence if initial iterate satisfies $Ax_0 = b$, then $Ax_i = b\ \forall i$

Computation: Amounts to solving a linear system.

Optimality conditions of (**) (with multipliers $\lambda \in \mathbb{R}^m$):

$$\begin{aligned} \nabla^2 f(x_i)\Delta x + \nabla f(x_i) + A^T\lambda &= 0 \\ A\Delta x &= 0 \end{aligned} \quad \Leftrightarrow \quad \boxed{\begin{bmatrix} \nabla^2 f(x_i) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_i) \\ 0 \end{bmatrix}}$$

Convergence of Newton's Method

Assumptions:

- f strongly convex with constant μ, i.e. $\nabla^2 f(x^*) \succeq \mu I$
- $\nabla^2 f$ is Lipschitz continuous with constant $M > 0$,
  i.e., $\|\nabla^2 f(x_1) - \nabla^2 f(x_2)\| \leq M\|x_1 - x_2\|_2$
  (M measures how well f can be approximated by a quadratic function)

Two phases: There exist constants $\eta \in (0, \mu^2/M), \gamma > 0$ such that

Damped Newton phase: $\|\nabla f(x)\|_2 \geq \eta$

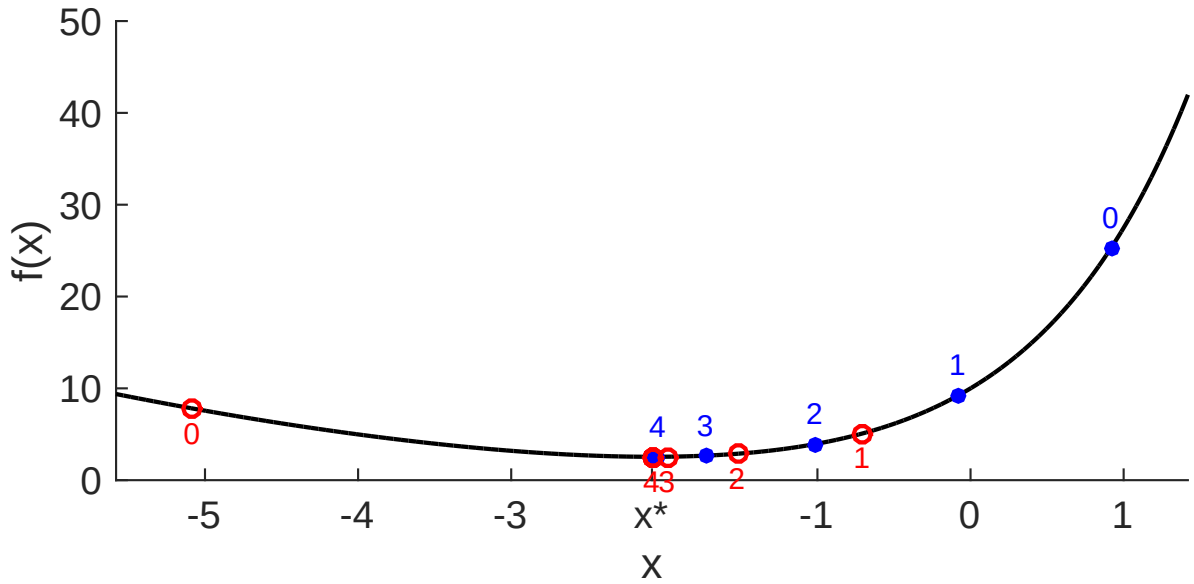
- Most iterations require backtracking steps
- Function value decreases by at least γ at each iteration

Quadratically convergent phase: $\|\nabla f(x)\|_2 < \eta$

- All iterations use step size $h_i = 1$
- $\|\nabla f(x)\|_2$ converges to zero **quadratically**

Newton's Method: $\min f(x) = 10e^x + 0.3x^2$

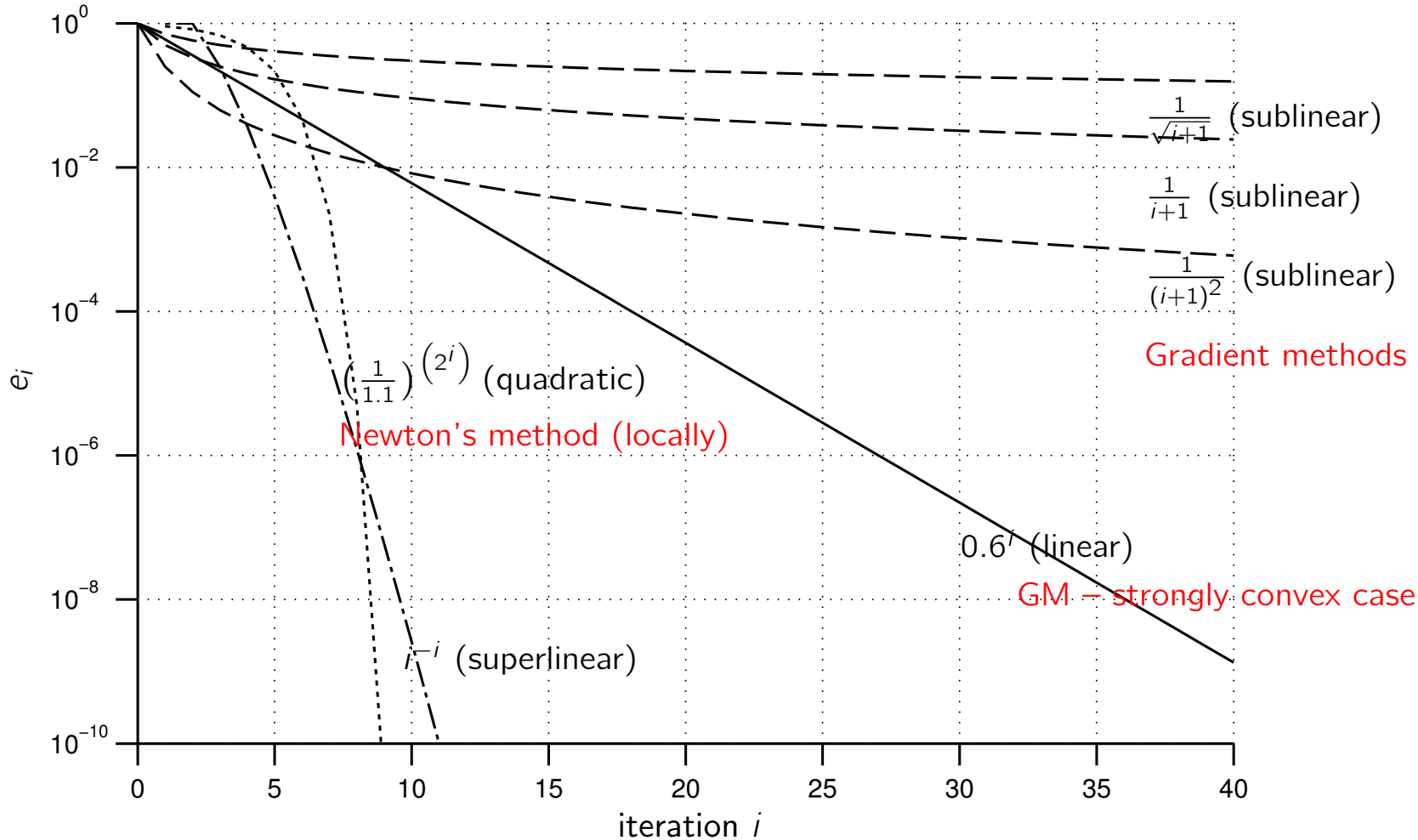
Initial values: $x_0^{(1)} = x^* + 3$ and $x_0^{(2)} = x^* - 3$ (backtracking step in iteration 1)



i	$\lvert x_i^{(1)} - x^* \rvert$	$h_i^{(1)}$	$\lvert x_i^{(2)} - x^* \rvert$	$h_i^{(2)}$
0	3.0e+00	—	3.0e+00	—
1	2.0e+00	1.00	1.4e+00	0.97
2	1.1e+00	1.00	5.6e-01	1.00
3	3.5e-01	1.00	1.0e-01	1.00
4	4.2e-02	1.00	3.6e-03	1.00
5	5.8e-04	1.00	4.3e-06	1.00
6	1.1e-07	1.00	6.4e-12	1.00
7	4.4e-15	1.00	0.0e+00	1.00

Convergence Rates

Examples of different classes of converging sequences $\{e_i\}$, with corresponding methods for **unconstrained** optimization:



Aspects of Newton's Method

- Convergence: is m finite for every $\epsilon, \delta > 0$? ✓
  (globally with line search)
- Convergence speed: dependence of errors $f(x_m) - f(x^*)$ and $\text{dist}(x_m, \mathbb{Q})$ on iteration counter ✓ (locally quadratically converging)
- Numerical robustness in presence of finite precision arithmetics ✓ (with line search)
- Warm-starting: can the method take advantage of x_0 being close to x^*? ✓
- Preconditioning: equivalence transformation of (P) into a similar problem (P') that can be solved in fewer iterations? no
- Each iteration computationally expensive
  (requires solving a system of linear equations)
- Equality constraints $Ax = b$ can be naturally handled by Newton step

Outline

Outline

Constrained Minimization Using Gradient Methods

Consider the following constrained convex optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{Q} \end{array} \tag{P}$$

where $\mathbb{Q}$ is convex and f is convex and L-smooth.

Recap: We can solve the **unconstrained** problem $\mathbb{Q} \equiv \mathbb{R}^n$ efficiently by the gradient method:

$$\begin{array}{rl} \text{Set} & x_0. \\ \textit{Repeat} & x_{i+1} = x_i - \frac{1}{L}\nabla f(x_i) \quad \text{for } i = 0, \ldots, m-1 \\ \textit{until} & f(x^m) - f(x^*) \leq \epsilon_1 \textit{ or } \|x_m - x_{m-1}\| \leq \epsilon_2. \end{array}$$

Question: How to handle constraints?

A Useful Reformulation of the Gradient Update

- **Unconstrained case:** Gradient update results from minimizing a quadratic function:

$$x_{i+1} = \arg\min_x f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2h_i}\|x - x_i\|^2$$

$$\Leftrightarrow \quad x_{i+1} = x_i - h_i \nabla f(x_i)$$

- **Constrained case:** Incorporate constraints in minimization:

$$x_{i+1} = \arg\min_{x \in \mathbb{Q}} f(x_i) + \nabla f(x_i)^T (x - x_i) + \frac{1}{2h_i}\|x - x_i\|^2$$

$$\Leftrightarrow \quad \boxed{x_{i+1} = \pi_{\mathbb{Q}}\left(x_i - h_i \nabla f(x_i)\right)}$$

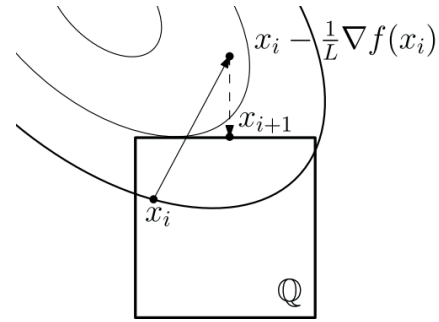
where $\pi_{\mathbb{Q}}$ is a **projection**:

$$\pi_{\mathbb{Q}}(y) \triangleq \arg\min_x \ \frac{1}{2}\|x - y\|_2^2$$
$$\text{s.t. } x \in \mathbb{Q}$$

(Proof by equivalence of optimality conditions and strong convexity.)

Gradient Methods for Constrained Optimization

Gradient method

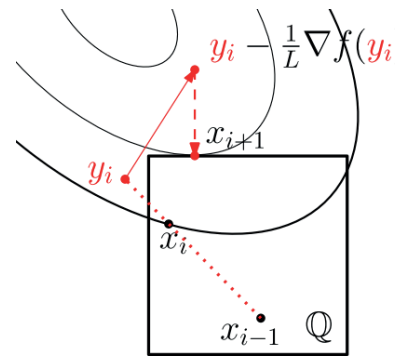


Set x_0.

Repeat $x_{i+1} = \pi_{\mathbb{Q}}\left(x_i - \frac{1}{L}\nabla f(x_i)\right)\ i = 0, \ldots, m-1$

until $f(x^m) - f(x^*) \leq \epsilon_1$ *or* $\|x_m - x_{m-1}\| \leq \epsilon_2$.

Fast gradient method



Set $x_0, y_0 = x_0$ and $\alpha_0 = (\sqrt{5}-1)/2$.

Repeat $x_{i+1} = \pi_{\mathbb{Q}}\left(y_i - \frac{1}{L}\nabla f(y_i)\right)$

$\alpha_{i+1} = \alpha_i/2\left(\sqrt{\alpha_i^2+4} - \alpha_i\right)$

$\beta_i = \frac{\alpha_i(1-\alpha_i)}{\alpha_i^2+\alpha_{i+1}}$

$y_{i+1} = x_{i+1} + \beta_i(x_{i+1} - x_i)$ for $i = 0, \ldots, m$

until $f(x_m) - f(x^*) \leq \epsilon_1$ *or* $\|x_m - x_{m-1}\| \leq \epsilon_2$.

Convergence rates are as in the unconstrained case.

Implications of Projection Operator in GMs

Problem: $\min_x \{f(x) \mid x \in \mathbb{Q}\}$

Recursion of gradient method: $x_{i+1} = \pi_{\mathbb{Q}}\left(x_i - h_i \nabla f(x_i)\right)$

If the projection

$$\pi_{\mathbb{Q}}(y) \triangleq \arg\min_x \left\{1/2\|x-y\|_2^2 \mid x \in \mathbb{Q}\right\} \qquad (*)$$

- is "easy" to compute: extremely efficient algorithm (same as for unconstrained optimization)
- can only be computed by solving (*) numerically: result not useful for primal problem, solve dual instead (potentially slow).

Sets that Allow Inexpensive Projections

Euclidean norm projection operators of closed convex sets in $\mathbb{R}^n$:

Set Name	Set Definition	Projection
hyperplane	$\mathbb{Q} = \left\{ x \in \mathbb{R}^n \mid a^T x = b \right\}$ with $(a, b) \in \mathbb{R}^n \times \mathbb{R}$, $a \neq 0$	$\pi_{\mathbb{Q}}(x) = x + \dfrac{b - a^T x}{\|a\|^2} a$
affine set	$\mathbb{Q} = \{ x \in \mathbb{R}^n \mid Ax = b \}$ with $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$, $A \neq 0$	$\pi_{\mathbb{Q}}(x) = \begin{cases} x + A^T (AA^T)^{-1} (b - Ax) & \text{if } \operatorname{rank} A = m, \\ x + A^T A^{T\dagger} \left(A^\dagger b - x \right) & \text{otherwise} \end{cases}$
halfspace	$\mathbb{Q} = \left\{ x \in \mathbb{R}^n \mid a^T x \leq b \right\}$ with $(a, b) \in \mathbb{R}^n \times \mathbb{R}$, $a \neq 0$	$\pi_{\mathbb{Q}}(x) = \begin{cases} x + \dfrac{b - a^T x}{\|a\|^2} a & \text{if } a^T x > b, \\ x & \text{otherwise} \end{cases}$
nonnegative orthant	$\mathbb{Q} = \{ x \in \mathbb{R}^n \mid x \geq 0 \}$	$\left(\pi_{\mathbb{Q}}(x) \right)_k = \begin{cases} x_k & \text{if } x_k \geq 0, \\ 0 & \text{otherwise} \end{cases} \quad k = 1, \ldots, n$
rectangle (e.g. ∞-norm ball)	$\mathbb{Q} = \{ x \in \mathbb{R}^n \mid l \leq x \leq u \}$ with $(l, u) \in \mathbb{R}^n \times \mathbb{R}^n$	$\left(\pi_{\mathbb{Q}}(x) \right)_k = \begin{cases} l_k & \text{if } x_k < l_k, \\ x_k & \text{if } l_k \leq x_k \leq u_k, \\ u_k & \text{if } x_k > u_k \end{cases} \quad k = 1, \ldots, n$

Sets that Allow Inexpensive Projections

Euclidean norm projection operators of closed convex sets in $\mathbb{R}^n$:

Set Name	Set Definition	Projection
2-norm ball	$\mathbb{Q} = \{x \in \mathbb{R}^n \mid \|x\| \leq r\}$ with $r \geq 0$	$\pi_{\mathbb{Q}}(x) = \begin{cases} r\frac{x}{\|x\|} & \text{if } \|x\| > r, \\ x & \text{otherwise} \end{cases}$
simplex	$\mathbb{Q} = \left\{ x \in \mathbb{R}^n \mid x \geq 0,\ \sum_{k=1}^{n} x_k = r \right\}$ with $r \geq 0$	Easy to implement $\mathcal{O}\left(n^2\right)$ algorithm or a $\mathcal{O}(n \log n)$ algorithm based on sorting. Both algorithms compute the projection exactly.
1-norm ball	$\mathbb{Q} = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq r\}$ with $r \geq 0$	The projection on the 1-norm ball can be reduced to a projection on a simplex, so that the complexity is $\mathcal{O}\left(n^2\right)$ or $\mathcal{O}(n \log n)$, see above.
second-order cone	$\mathbb{Q} = \left\{(x, s) \in \mathbb{R}^n \times \mathbb{R}_+ \mid \|x\| \leq s\right\}$	$\pi_{\mathbb{Q}}((x,s)) = \begin{cases} (x,s) & \text{if } \|x\| \leq s, \\ \frac{s+\|x\|}{2\|x\|} \begin{bmatrix} x \\ \|x\| \end{bmatrix} & \text{if } -\|x\| < s < \|x\|, \\ (0,0) & \text{if } \|x\| \leq -s \end{cases}$

Implications for MPC

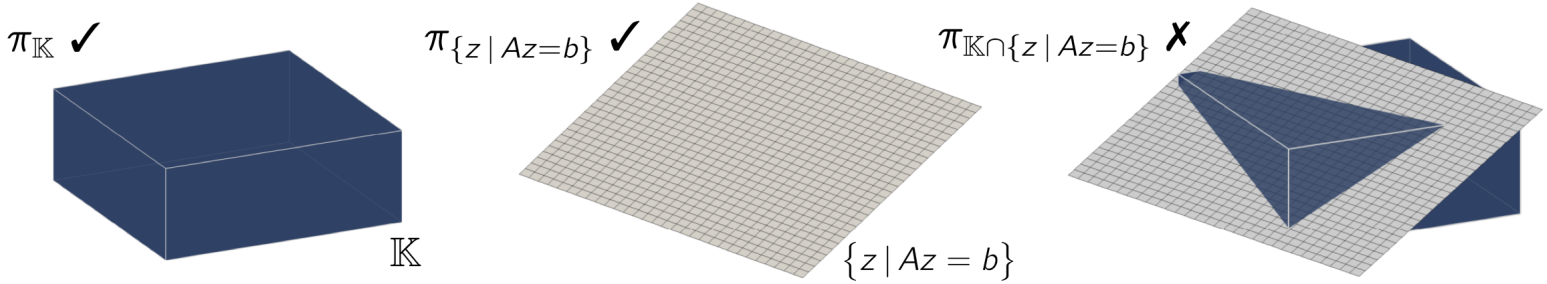
- MPC with simple input constraint set $\mathbb{U}$ (using "condensed" formulation, i.e. eliminate states):

$$\begin{aligned} \min \ & \frac{1}{2} u^T H u + x_0^T F u \\ \text{s.t. } & u \in \mathbb{U} \end{aligned}$$

- MPC with state constraints: Individual projections are easy, projection on intersection is not $\rightarrow$ dualize

$$\begin{aligned} \min \ & 1/2 \left(\sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N \right) \\ \text{s.t. } & x_{i+1} = A x_i + B u_i \\ & u_i \in \mathbb{U}_i, \quad x_i \in \mathbb{X}_i \end{aligned}$$

Illustration: $z \triangleq (x, u)$, $\mathbb{K} \triangleq \mathbb{X} \times \mathbb{U}$



If projection not simple: Dual Gradient Methods

Idea: Gradient ascent method on dual problem

Example:

- Primal problem ($A \in \mathbb{R}^{m \times n}$, f strongly convex):

$$f^\star \triangleq \min\ f(x)$$
$$\text{s.t. } Ax = b, x \in \mathbb{K}$$

- (Partial) Lagrangian with multiplier $\lambda \in \mathbb{R}^m$:

$$L(x, \lambda) = f(x) + \lambda^T (Ax - b), x \in \mathbb{K}$$

- Dual function:

$$d(\lambda) = \min_{x \in \mathbb{K}} L(x, \lambda)$$

  with gradient

$$\nabla d(\lambda) = Ax^\star(\lambda) - b \quad \text{where} \quad x^\star(\lambda) \triangleq \arg\min_{x \in \mathbb{K}} L(x, \lambda)$$

  Due to strong duality, $\max_\lambda d(\lambda) \equiv f^\star$, and $x^\star(\lambda^\star) = \arg\min_{x \in \mathbb{K}} L(x, \lambda^\star)$

- Use gradient ascent to solve dual problem: $\lambda_{i+1} = \lambda_i + \frac{1}{L_d} \nabla d(\lambda_i)$

Dual Gradient Methods: Caveats

Issues with dual gradient methods:

- Dual function $d(\lambda)$ is not strongly concave $\Rightarrow$ sub-linear convergence
- If f is not strongly convex, $d(\lambda)$ is not differentiable $\Rightarrow$ sub-gradient method
- Each gradient ascent step requires solution to "inner" optimization problem:

$$\nabla d(\lambda) = Ax^\star(\lambda) - b \quad \text{where} \quad x^\star(\lambda) \triangleq \arg\min_{x \in \mathbb{K}} f(x) + \lambda^T(Ax - b)$$

- Lipschitz constant L_d hard to compute in general. Can be upper bounded by

$$L_d \leq \frac{\sigma^2_{\max}(A)}{\mu}$$

$\sigma_{\max}(A)$: largest singular value of A, μ: strong convexity parameter of f. For quadratic functions $f(x) = \frac{1}{2}x^T Hx + c^T x$, L_d is known exactly:

$$L_d = \|AH^{-1/2}\|^2$$

Consequently, other methods are usually used in practice.

Recap: Gradient Methods for Constrained Optimization

- Constrained gradient methods = gradient scheme + projection
- Very efficient if projection is inexpensive ($\rightarrow$ simple sets)
  Linear convergence rate for strongly convex problems
- Otherwise problem needs to be dualized (slower)
- MPC with
  - simple input constraints: extremely fast ✓
  - state constraints: solved in dual domain (potentially slower)

Outline

Composite Programs

Consider the problem

$$\begin{aligned} &\text{minimize } f(x) + g(y) \\ &\text{subject to } Ax + By = c \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ are convex functions, and the matrices $A \in \mathbb{R}^{p \times n}$ and $B \in \mathbb{R}^{p \times m}$ have full rank.

Remarks:

- Encoding inequalities: $g(y)$ can be indicator function for set $\mathbb{Q}$: $g(y) \equiv I_{\mathbb{Q}}(y)$
- If $A = -B = I$ and $c = 0$, equality constraint reduces to $x = y$
- Can therefore easily transform the problem into the form above:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in \mathbb{Q} \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} &\text{minimize } f(x) + I_{\mathbb{Q}}(y) \\ &\text{subject to } x = y \end{aligned}$$

Augmented Lagrangian and Dual Function

Consider the problem

$$\begin{aligned} &\text{minimize } f(x) + g(y) \\ &\text{subject to } Ax + By = c \end{aligned} \qquad (*)$$

Augmented Lagrangian:

$$L_\rho(x, y, \lambda) \triangleq f(x) + g(y) + \lambda^T(Ax + By - c) + \frac{\rho}{2}\|Ax + By - c\|_2^2$$

for some $\rho > 0$. $L_\rho(x, y, \lambda)$ is strongly convex in both x and y, irrespective of strong convexity of f or g. It is concave in λ.

Dual function:

$$d(\lambda) \triangleq \min_{x,y} L_\rho(x, y, \lambda)$$

Gradient of dual function (usually expensive to compute):

$$\nabla d(\lambda) = Ax^\star + By^\star - c \quad \text{where} \quad (x^\star(\lambda),\, y^\star(\lambda)) = \arg\min_{x,y} L_\rho(x, y, \lambda)$$

Alternating Minimization Method

Consider dual problem

$$d^\star = \max_\lambda d(\lambda)$$

Iteration of standard gradient ascent method with step size h_i:

$$\lambda_{i+1} = \lambda_i + h_i \nabla d(\lambda_i) \quad \Leftrightarrow \quad \begin{aligned} (x_{i+1}, y_{i+1}) &= \arg\min_{x,y} L_\rho(x, y, \lambda_i) \\ \lambda_{i+1} &= \lambda_i + h_i \underbrace{(Ax_{i+1} + By_{i+1} - c)}_{=\nabla d(\lambda_i)} \end{aligned}$$

Idea of alternating minimization: Approximate $\nabla d(\lambda_i)$ by two subsequent minimizations that are easy to compute, one in x and one in y:

$$\boxed{\begin{aligned} x_{i+1} &= \arg\min_x L_\rho(x, y_i, \lambda_i) \\ y_{i+1} &= \arg\min_y L_\rho(x_{i+1}, y, \lambda_i) \\ \lambda_{i+1} &= \lambda_i + h_i(Ax_{i+1} + By_{i+1} - c) \end{aligned}}$$

Alternating Direction Method of Multipliers (ADMM)

With constant step size $h_i \equiv \rho \; \forall i$, we obtain the standard ADMM iterations:

$$
\begin{aligned}
x_{i+1} &= \arg\min_x f(x) + \lambda_i^T A x + \frac{\rho}{2}\|Ax + By_i - c\|_2^2 \\
y_{i+1} &= \arg\min_y g(y) + \lambda_i^T B y + \frac{\rho}{2}\|Ax_{i+1} + By - c\|_2^2 \\
\lambda_{i+1} &= \lambda_i + \rho(Ax_{i+1} + By_{i+1} - c)
\end{aligned}
$$

Can be very efficient if single minimization much cheaper than $\min_{x,y} L(x, y, \lambda_i)$

Remarks:

- ADMM is rather a framework than a method, many variants (variable splittings, solving primal or dual) exist. Dates back to 1960s
- Used also for large-scale distributed optimization (e.g. machine learning)
- Can be shown to converge for all $\rho > 0$ (proof via Lyapunov fcn, beyond this lecture), but careful selection of ρ crucial for convergence speed
- Convergence rate known in special cases, but generally unknown
- Can be accelerated using Nesterov's scheme

Example: ADMM applied to QPs

Consider the quadratic program

$$\begin{aligned} &\text{minimize } \frac{1}{2}x^T Hx + c^T x \\ &\text{subject to } Ax = b, x \in \mathbb{K} \end{aligned}$$

Assumption: projection on $\mathbb{K}$ can be efficiently computed

Re-formulate the problem (choose variable **splitting**) into

$$\begin{aligned} &\text{minimize } f(x) + g(y) \\ &\text{subject to } x = y \end{aligned} \quad \text{using} \quad \begin{aligned} f(x) &= \frac{1}{2}x^T Hx + c^T x + I_{Ax=b}(x) \\ g(y) &= I_{\mathbb{K}}(y) \end{aligned}$$

ADMM iterations:

$$x_{i+1} = \arg\min \frac{1}{2}x^T Hx + c^T x + \lambda_i^T x + \frac{\rho}{2}\|x - y_i\|_2^2 \quad \rightarrow \text{eq. constr. QP}$$

$$\text{s.t. } Ax = b \qquad = 1 \text{ Newton step}$$

$$y_{i+1} = \arg\min_{y \in \mathbb{K}} -\lambda_i^T y + \frac{\rho}{2}\|x_{i+1} - y\|_2^2 \quad \rightarrow \text{Projection on } \mathbb{K}$$

$$\lambda_{i+1} = \lambda_i + \rho(x_{i+1} - y_{i+1})$$

Outline

Constrained Minimization Problem

Consider the following problem with inequality constraints

$$\begin{aligned} \min \;& f(x) \\ \text{s.t. } & g_i(x) \leq 0,\ i = 1, \ldots, m \end{aligned}$$

Assumptions:

- f, g_i convex, twice continuously differentiable
- $f(x^*)$ is finite and attained
- strict feasibility: there exists a $\tilde{x}$ with

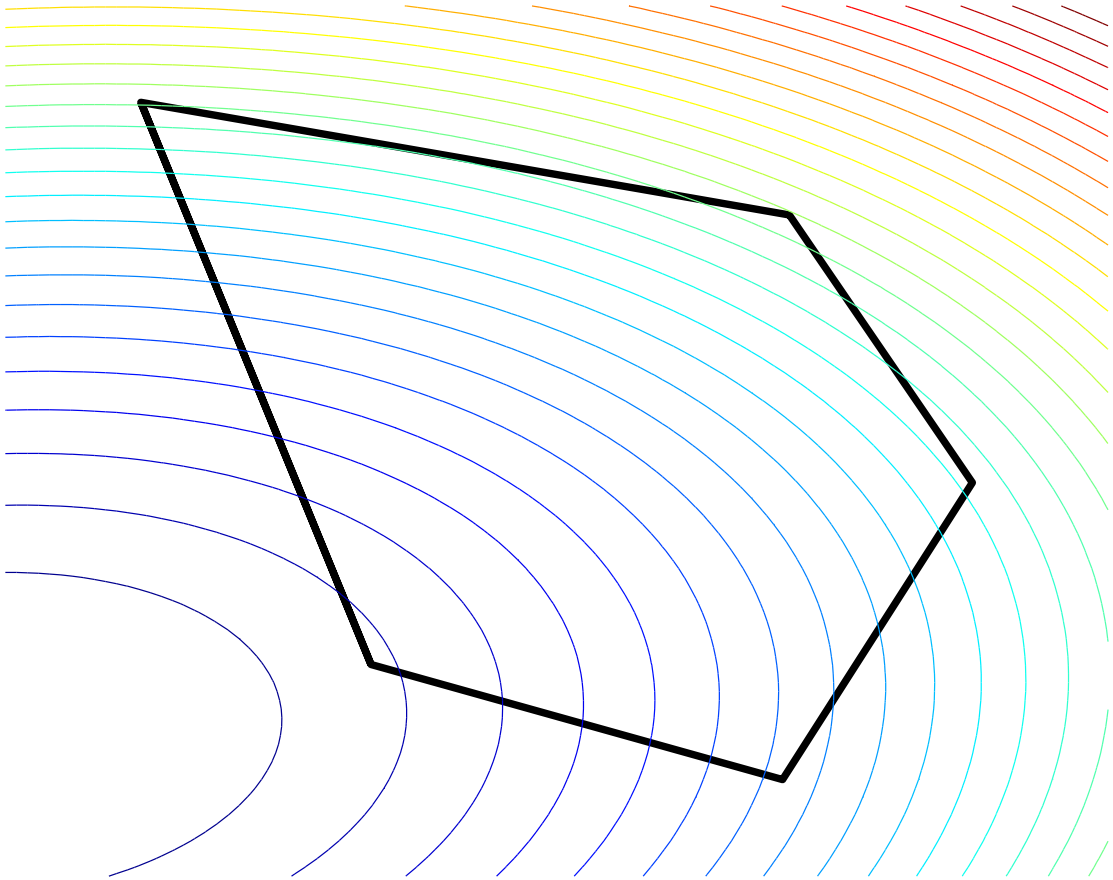
$$\tilde{x} \in \text{ dom } f, \quad g_i(\tilde{x}) < 0, i = 1, \ldots, m$$

- feasible set is closed and compact

Idea: There exist many methods for unconstrained minimization

$\Rightarrow$ Reformulate problem as an unconstrained problem

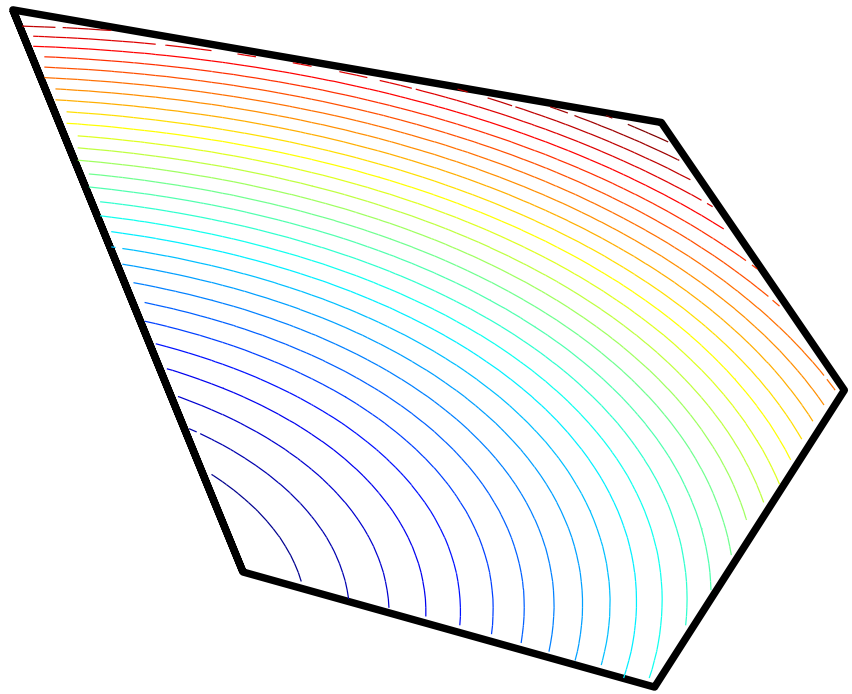
Graphical Illustration



Minimize a function over a set

Graphical Illustration

Define function as ∞ if constraints violated.



Minimize this function over $\mathbb{R}^n$

Barrier Method

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \leq 0,\ i = 1, \ldots, m \end{array} \quad \Leftrightarrow \quad \min \quad f(x) + \kappa\phi(x)$$

Constraints have been moved to objective via **indicator function:**

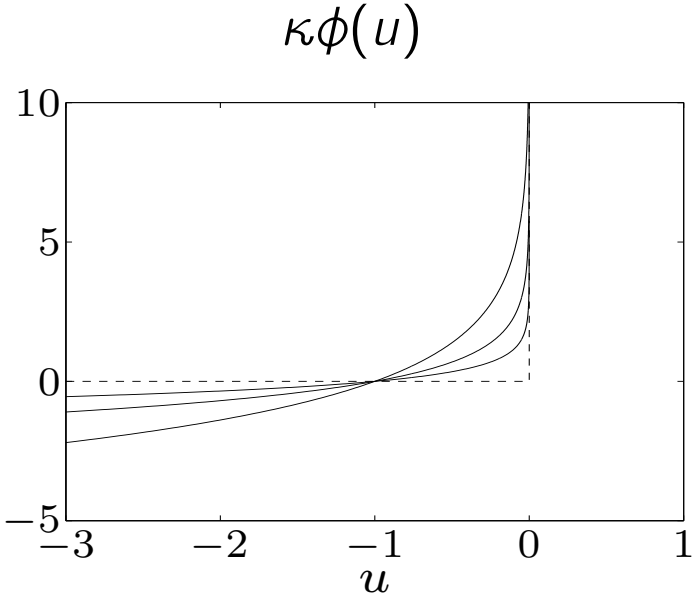
$$\phi(x) = \sum_{i=1}^{m} I_-(g_i(x)), \quad \kappa = 1$$

where $I_-(u) = 0$ if $u \leq 0$ and $I_- = \infty$ otherwise

- Augmented cost is not differentiable
  $\rightarrow$ approximation by **logarithmic barrier:**

$$\phi(x) = -\sum_{i=1}^{m} \log(-g_i(x))$$

- For $\kappa > 0$ smooth approximation of indicator function
- Approximation improves as $\kappa \rightarrow 0$



Logarithmic Barrier Function

$$\phi(x) = -\sum_{i=1}^{m} log(-g_i(x)), \qquad \text{dom } \phi = \{x \mid g_1(x) < 0, \ldots, g_m(x) < 0\}$$

- Convex, smooth on its domain
- $\phi(x) \to \infty$ as x approaches boundary of domain
- $\arg\min_x \phi(x)$ is called **analytic center** of the set defined by inequalities $g_1 < 0, \ldots, g_m < 0$
- Twice continuously differentiable with derivatives

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-g_i(x)} \nabla g_i(x)$$

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x) \nabla g_i(x)^T + \frac{1}{-g_i(x)} \nabla^2 g_i(x)$$

Central Path

- Define $x^*(\kappa)$ as the solution of

$$\min_x f(x) + \kappa\phi(x)$$

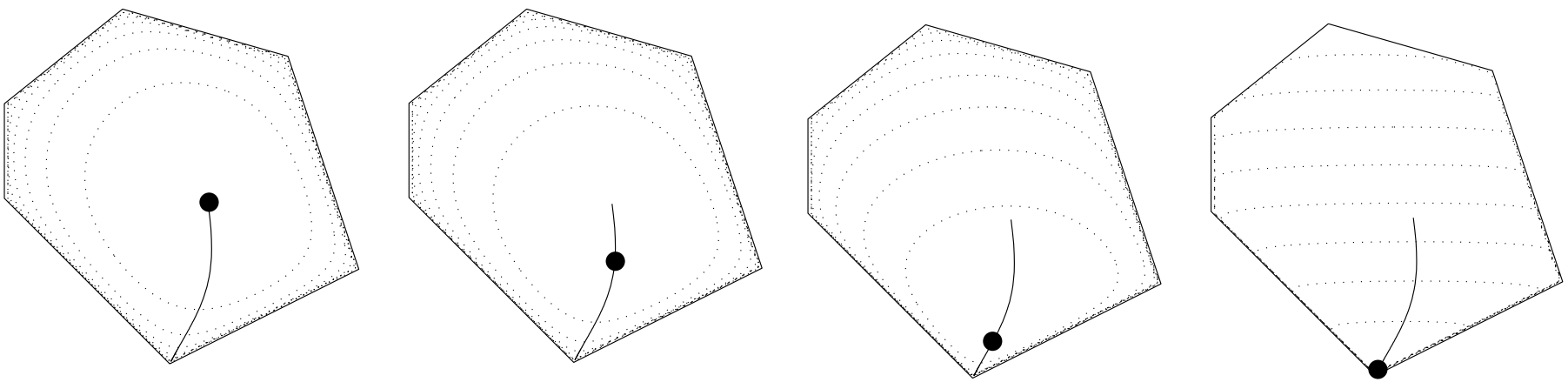
  (assume minimizer exists and is unique for each $\kappa > 0$)
- Barrier parameter κ determines relative weight between objective and barrier
- Barrier 'traps' $x(\kappa)$ in strictly feasible set
- **Central path** is defined as $\{x^*(\kappa) \mid \kappa > 0\}$
- For given κ can compute $x^*(\kappa)$ by solving smooth unconstrained minimization problem
- Intuitively $x^*(\kappa)$ converges to optimal solution as $\kappa \to 0$
  (easy to prove under mild conditions)

Example: Central Path for an LP

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & a_i^T x \leq b_i, i = 1, \ldots, 6 \end{aligned}$$

$x \in \mathbb{R}^2$, c points upward

$\kappa = 1000$ $\kappa = 1$ $\kappa = 1/5$ $\kappa = 1/100$



Path-following Method

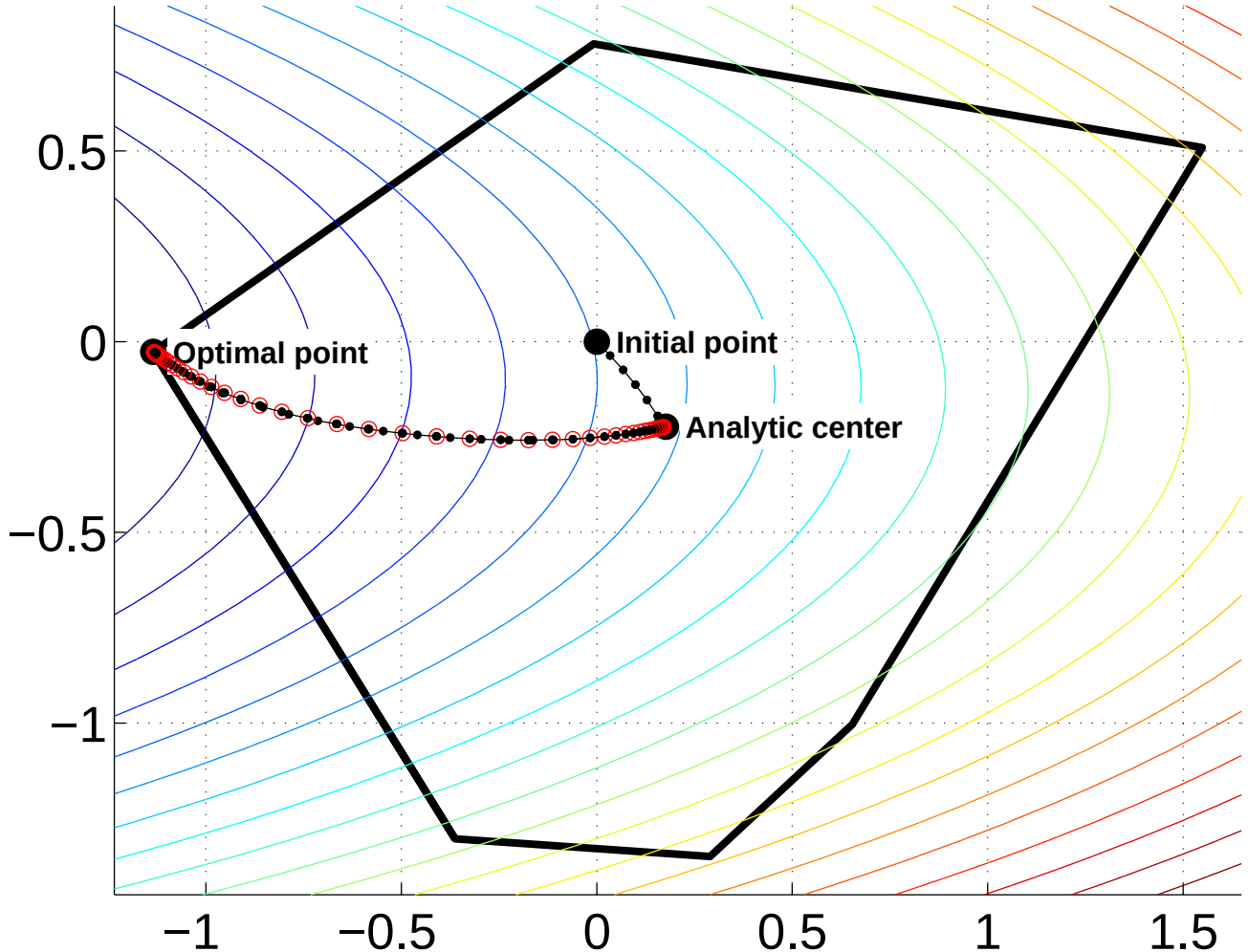
Idea: Follow central path to the optimal solution

Solve sequence of smooth unconstrained problems:

$$x^*(\kappa) = \arg\min_x f(x) + \kappa\phi(x)$$

- Assume current solution is on the central path $x_i = x^*(\kappa_i)$
- Obtain κ_{i+1} by decreasing κ_i by some amount: $\kappa_{i+1} = \kappa_i/\mu, \mu > 1$
- Solve for $x^*(\kappa_{i+1})$ starting from $x^*(\kappa_i)$ (unconstrained optimization). Called "centering step" because it computes a point on (or close to) the central path
- Method converges to the optimal solution, i.e., $x_i \to x^*$ for $\kappa \to 0$

Example - Quadratic Program



Centering Step Using Newton's Method

Idea: Exploit fast local convergence of Newton's method starting at point close to optimum

- **Newton direction**: Δx_{nt} minimizes second order approximation

$$\begin{aligned} f(x+v) + \kappa\phi(x+v) \approx & f(x) + \kappa\phi(x) + \nabla f(x)^T v + \kappa\nabla\phi(x)^T v \\ & + \frac{1}{2} v^T \nabla^2 f(x) v + \frac{1}{2}\kappa v^T \nabla^2 \phi(x) v \end{aligned}$$

- Newton direction for barrier method is given by solution of

$$(\nabla^2 f(x) + \kappa\nabla^2\phi(x))\Delta x_{nt} = -\nabla f(x) - \kappa\nabla\phi(x)$$

Line search consists of two steps:

1. For retaining feasibility, find
   $h_{max} = \mathbf{arg\,max}_{h>0}\{h \mid g_1(x + h\Delta x) < 0, \ldots, g_m(z + h\Delta x) < 0\}$
2. Find $h^* = \mathbf{arg\,min}_{h\in[0,\, h_{max}]}\{f(x + h\Delta x) + \kappa\phi(x + h\Delta x)\}$

both either solved exactly or through backtracking

Centering Step with Equality Constraints

Centering Step: Compute $x^*(\kappa)$ by solving

$$\begin{array}{ll} \text{min} & f(x) + \kappa\phi(x) \\ \text{s.t.} & Cx = d \end{array}$$

- Newton step Δx_{nt} for minimization with equality constraints is given by solution of

$$\begin{bmatrix} \nabla^2 f(x) + \kappa\nabla^2\phi(x) & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \nu \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \kappa\nabla\phi(x) \\ 0 \end{bmatrix}$$

- Same interpretation as Newton step for unconstrained problem: $x + \Delta x_{nt}$ minimizes second order approximation

$$\begin{array}{ll} \text{min} & \nabla f(x)^T v + \kappa\nabla\phi(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v + \frac{1}{2}\kappa v^T \nabla^2\phi(x) v \\ \text{s.t.} & Cv = 0 \end{array}$$

Barrier Interior-point Method

$$\min_x \{f(x) \mid g(x) \leq 0,\ Cx = d\}$$

Input: strictly feasible x_0 w.r.t. $g(x)$, κ_0, $\mu > 1$, tol. $\epsilon > 0$

repeat

1. **Centering step: Compute** $x^*(\kappa_i)$ **by minimizing** $f(x) + \kappa_i \phi(x)$ **subject to** $Cx = d$ **starting from** x_{i-1}
2. *Update* $x_i = x^*(\kappa_i)$
3. *Stopping criterion:* Stop if $m\kappa_i < \epsilon$
4. *Decrease barrier parameter:* $\kappa_{i+1} = \kappa_i / \mu$

- Several heuristics for choice of κ_0 and other parameters[2]
- Terminates with $f(x_i) - f(x^*) \leq \epsilon$
- Steps 1-4 represent one outer iteration
- Step 1: Solve equality constrained minimization problem (via Newton steps)

[2] More details in e.g. *[Boyd & Vandenberghe, Convex Optimization, 2004]*

Example: Newton Step for Quadratic Program

$$\min_x \left\{ 1/2 x^T H x \mid Gx \leq d \right\}$$

- Barrier method:

$$\min_x f(x) + \kappa\phi(x) = \min_x \frac{1}{2} x^T H x - \kappa \sum_{i=1}^{m} \log(d_i - g_i x)$$

  where $g_1, \ldots, g_m$ are the rows of G.

- The gradient and Hessian of the barrier function are:

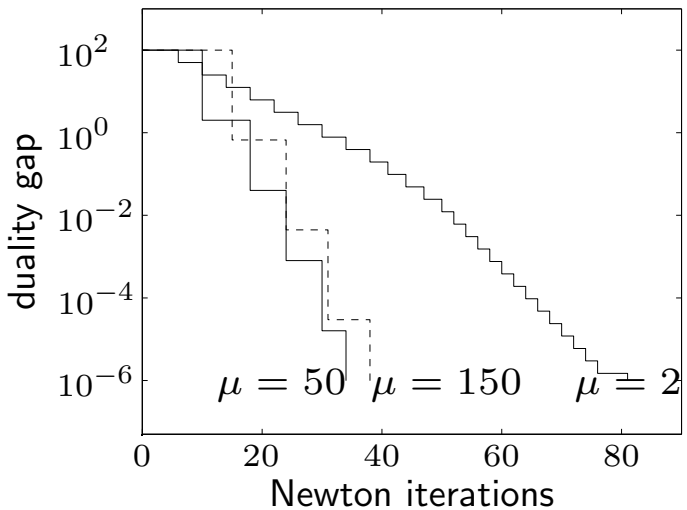
$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T, \nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i$$

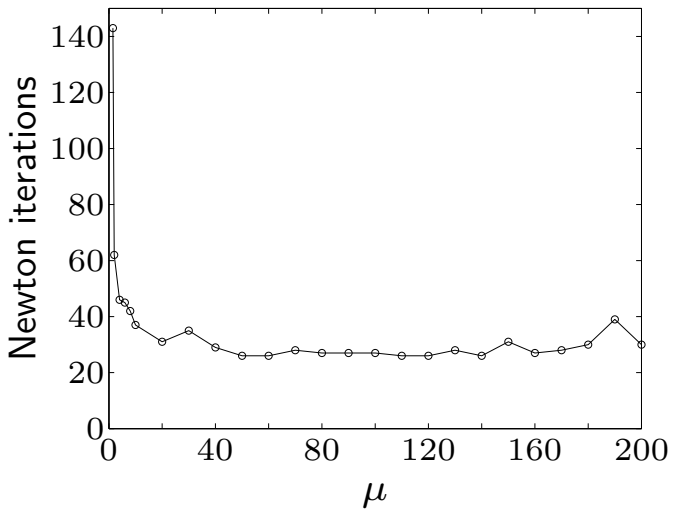
- Newton step: $(\nabla^2 f(x) + \kappa\nabla^2\phi(x))\Delta x_{nt} = -\nabla f(x) - \kappa\nabla\phi(x)$

$$(H + \kappa \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i)\Delta x_{nt} = -Hx - \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T$$

Remarks on Barrier Method

- Choice of μ involves trade-off: large $\mu \Rightarrow$ few outer iterations, but more inner iterations to compute x_{i+1} from x_i (typical values $\mu = 10 - 20$)
- Good convergence properties for a wide range of parameters μ
  Example: LP with 100 inequalities, 50 variables





- Barrier method requires strictly feasible initial point
  $\rightarrow$ Phase I method, e.g., $\mathsf{min}_{x,s}\{s \mid g(x) \leq s \cdot \mathbf{1}\}$

Interpretation of Barrier IPM via KKT conditions

KKT conditions of the barrier problem:

$$Cx^* = d$$

$$g_i(x^*) \leq 0, i = 1, \ldots, m$$

$$\nabla f(x^*) + \kappa \sum_{i=1}^{m} \frac{1}{-g_i(x^*)} \nabla g_i(x^*) + C^T \nu^* = 0$$

Defining

$$\lambda_i^* = \kappa \frac{1}{-g_i(x^*)} \geq 0$$

results in the standard KKT conditions where complementarity slackness is replaced by the relaxed condition

$$\lambda_i^* g_i(x^*) = -\kappa$$

Primal-Dual Interior-point Methods

Using the result from above, the **relaxed** KKT system can be written as

$$\begin{aligned} Cx^* &= d \\ g_i(x^*) + s_i^* &= 0 \; i = 1, \ldots, m \\ \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + C^T \nu^* &= 0 \\ \lambda_i^* g_i(x^*) &= -\kappa \\ \lambda_i^*, s_i^* &\geq 0, \; i = 1, \ldots, m \end{aligned} \qquad (**)$$

Idea: leave dual multipliers λ_i^* as variables (before, they were implicitly defined by primal log barrier)[3]:

- Solve the primal and dual problem simultaneously via (**)
- Primal-dual central path $\triangleq \{(x, s, \lambda, \nu) \mid (**) \text{ holds}\}$
- Follow central path to solution by reducing κ to zero
- Solve (**) by Newton method (with additional "safeguards" & line search)

[3] See e.g. *[Stephen Wright, Primal-dual Interior-point Methods, SIAM 1997]*

Primal-Dual Search Direction Computation

At each iteration, linearize (**) and solve

$$\begin{bmatrix} H(x,\lambda) & C^T & G(x)^T & 0 \\ C & 0 & 0 & 0 \\ G(x) & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + C^T y + G(x)^T \lambda \\ Cx - d \\ g(x) + s \\ S\lambda - \nu \end{bmatrix}$$

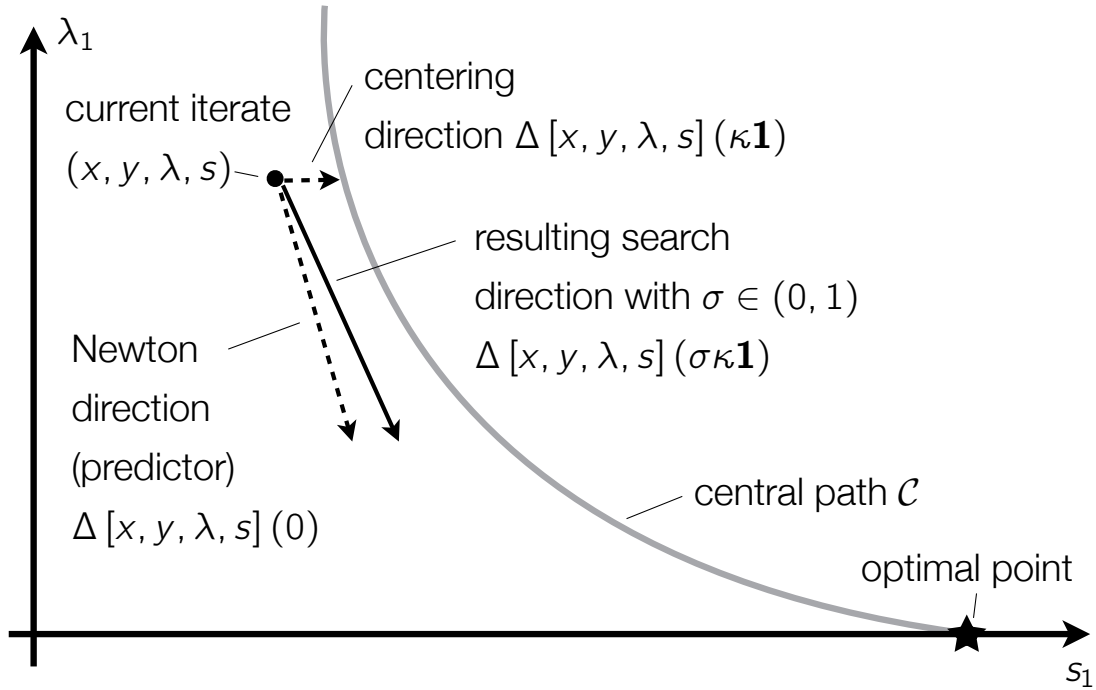
where $S \triangleq \text{diag}(s_1, \ldots, s_m)$ and $\Lambda \triangleq \text{diag}(\lambda_1, \ldots, \lambda_m)$, the (1,1) block in the coefficient matrix is

$$H(x,\lambda) \triangleq \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x)$$

and the vector $\nu \in \mathbb{R}^m$ allows for a modification of the right-hand side. Call resulting direction $\Delta\,[x, y, \lambda, s]\,(\nu)$.

Search Directions in Primal-Dual Methods

Can generate different directions $\Delta[x, y, \lambda, s](\nu)$ depending on ν:



- $\nu = 0$: standard Newton method for solving nonlinear equations
- $\nu = \kappa\mathbf{1}$: centering direction, next iterate is on central path

$\Rightarrow$ Using linear combination via **centering parameter** $\sigma \in (0, 1)$ ensures fast convergence in theory & practice by tracking central path to solution

Recap: Interior-point Methods

Barrier method
(also called **Sequential Unconstrained Minimization Technique, SUMT)**

- Intuition: Follow central path to the optimal solution
- Log barrier function ensures satisfaction of inequality constraints
- Centering problems can be solved efficiently using Newton's method
- Requires strictly feasible initial point

'Modern' methods: Primal-dual methods

- Often more efficient than barrier method (superlinear convergence)
- Cost per iteration roughly the same as barrier method
- Allow for **infeasible start** (w.r.t. both equality and inequality constraints)
- Can provide **certificates of infeasibility** (using self-dual embedding)
- Most efficient in practice: Mehrotra's predictor-corrector method[4]: < 30 iterations in practice

Interior-point methods are very efficient for range of optimization problems, e.g. LPs, QPs, second-order cone and semidefinite programs.

[4] See e.g. J. Nocedal and S. Wright: *Numerical Optimization*, 2006, Springer

Active Set Idea

Consider

$$\min_x \ f(x)$$
$$\text{subj. to } x \in S,$$

where the feasible set $S \subset \mathbb{R}^s$ is a polyhedron, i.e. a set defined by linear equalities and inequalities, and the objective f is a linear function (linear programming (LP)) or a convex quadratic function (quadratic programming (QP)).

- Active set methods aim to identify the set of active constraints at the solution. Once this set is known, a solution to the problem can be easily identified.
- Since the number of potentially visited active sets depends combinatorially on the number of decision variables and constraints, these methods have a worst case complexity that is exponential in the problem size (as opposed to first-order and interior point methods). However, active set methods have proved to work quite well in practice.

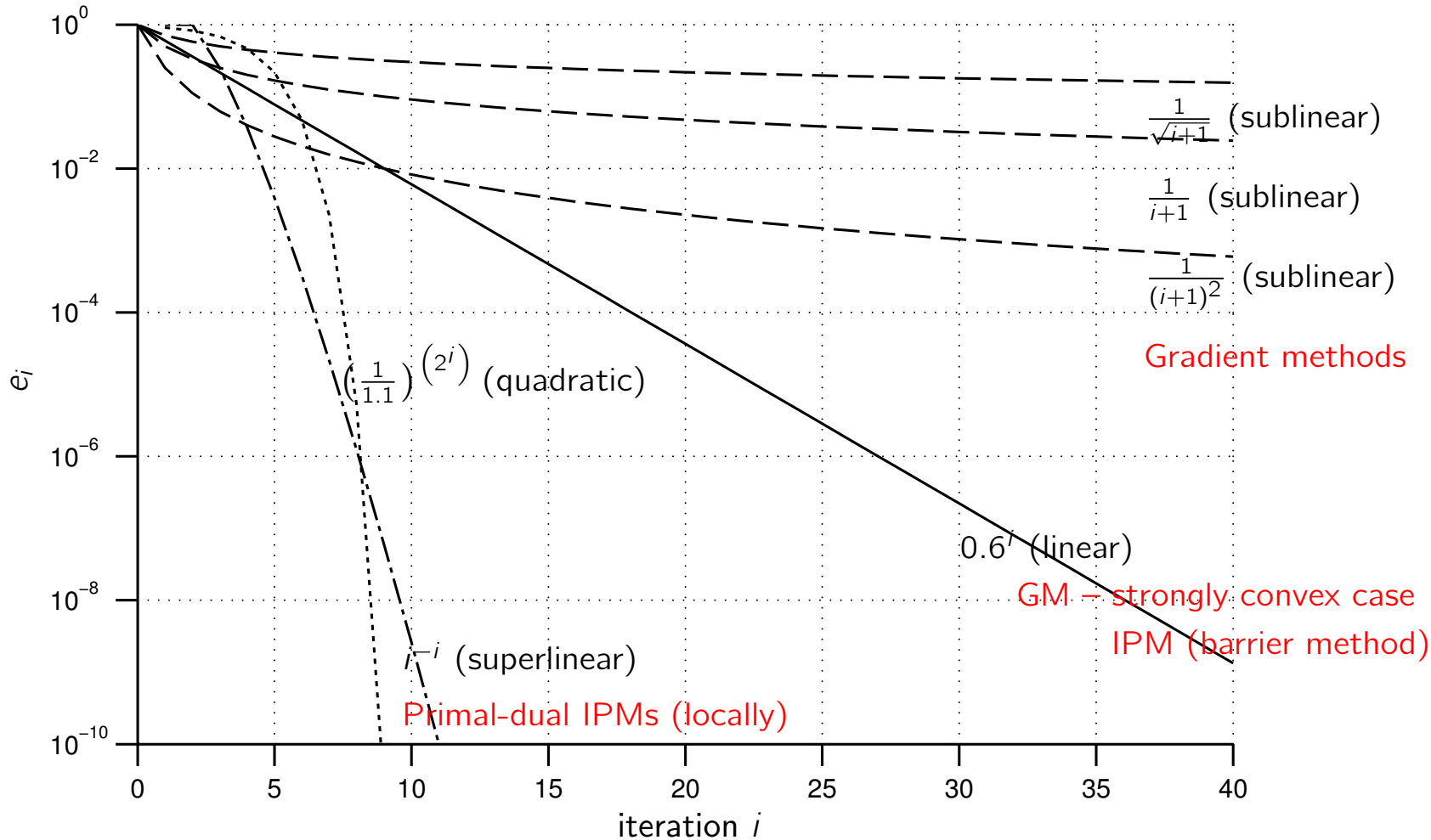
Active Set for LPs

$$\min_x \; c^\top x$$
$$\text{Subj. to } Gx \leq w,$$

- Central to active set methods for LP is the observation that a solution is always attained at a vertex of the polyhedra feasible set.
- A simple strategy would be to enumerate all vertices of the polyhedron and declare the vertex with the smallest cost as the solution.
- However, one can do better by only visiting those vertices that improve the cost at the previous one.
- This is the main idea behind active set methods for LP.

Convergence Speeds

Examples of different classes of converging sequences $\{e_i\}$, with corresponding methods for **constrained** optimization:



Outline

Outline

Software for Modeling Optimization Problems

Formulate optimization problem in mathematical language and pass to solver:

- **CVX** (cvxr.com/cvx): Matlab software for modeling convex problems
- **YALMIP** (users.isy.liu.se/johanl/yalmip): Matlab software for modeling convex and some non-convex optimization problems. MPC-Example:

```
u = sdpvar(repmat(nu,1,N),repmat(1,1,N));
constraints = []; objective = 0; x = x0;
for k = 1:N
    x = A*x + B*u{k};
    objective = objective + norm(Q*x,1) + norm(R*u{k},1);
    constraints = [constraints, -1 <= u{k}<= 1, -5<=x<=5];
end
```

- **AMPL** (www.ampl.com): industry standard, proprietary software. Supports basically all solvers.
- **GAMS** (www.gams.com): commercial high-level modeling system for large-scale optimization. Supports many different types of problems (LPs, QCQPs, MILPs, MINLPs, ...) and solvers

Outline

4. Software
    - Modeling
    - Solvers for Desktop PCs
    - Solvers for Embedded Platforms

Software for Solving Convex Problems on Desktop PCs

Standalone solvers (tedious to use without modeling language)

- **SeDuMi** (`sedumi.ie.lehigh.edu`): widely used free solver, with Matlab interface
- **SDPT3** (`www.math.nus.edu.sg/~mattohkc/sdpt3`): Matlab software, free (GPL)
- **CVXOPT** (`abel.ee.ucla.edu/cvxopt`): free Python solver, allows customization of linear system solvers
- **IBM CPLEX**: industry standard for (MI)LPs and (MI)QCQPs (commercial)
- **Gurobi** (`www.gurobi.com`): commercial (MI)SOCP solver, by creators of CPLEX, strong Python support
- **MOSEK** (`www.mosek.com`): fastest commercial solver for second-order cone programs

All listed solvers are based on interior-point methods (5 out of 6 primal-dual)

Outline

Convex Optimization Solvers for Embedded Platforms

Solvers: – open source:

- **qpOASES** (`www.kuleuven.be/optec/software/qpOASES`): active set solver, free (LGPL)
- **OOQP** (`pages.cs.wisc.edu/~swright/ooqp`): object-oriented QP solver (needs LAPACK/BLAS)
- **ECOS** (`github.com/embotech/ecos`): Sparse SOCP solver, 800 lines of library free C code, Python & Matlab interface

Code generation: – generate problem-specific C-code:

- **CVXGEN** (`cvxgen.com`): code generation for small QPs, extremely fast, code can get large
- **FiOrdOs** (`fiordos.ethz.ch`): code generation for gradient methods. Supports certification, automatic preconditioning, etc.
- **FORCES PRO** (`www.embotech.com`): Code generation for interior-point, gradient methods, ADMM, non-convex solvers (NLPs), problems with binary variables

How Fast?

Time per iteration for MPC problem on desktop PC.
Total time will be $\sim 10\times$ slower.