

# Subgradients

- subgradients
- strong and weak subgradient calculus
- optimality conditions via subgradients
- directional derivatives

## Basic inequality

recall basic inequality for convex differentiable  $f$ :

$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

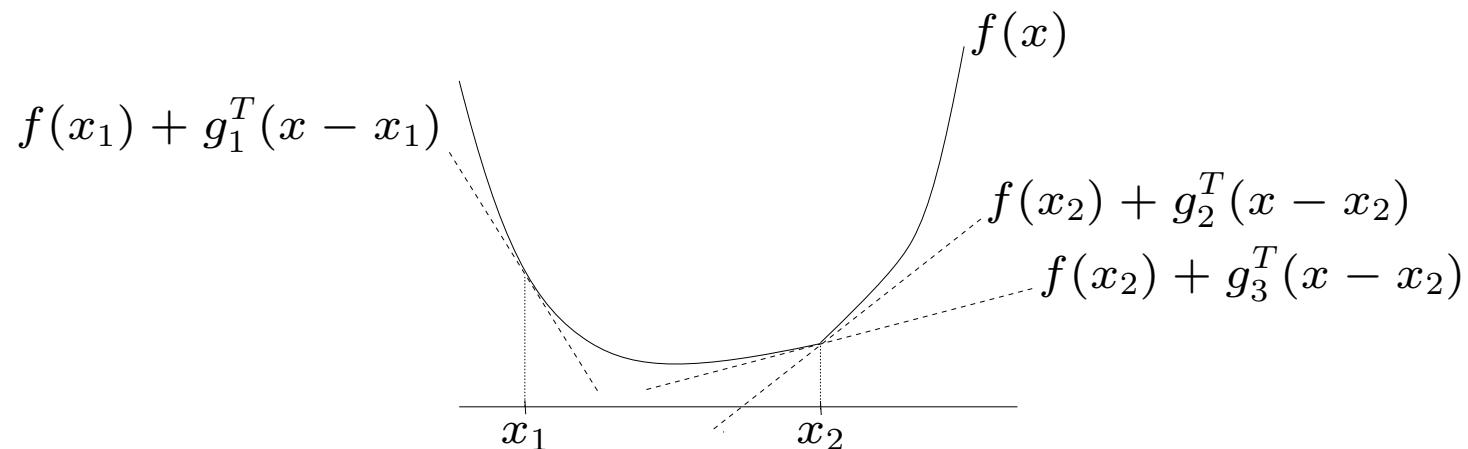
- first-order approximation of  $f$  at  $x$  is global underestimator
- $(\nabla f(x), -1)$  supports **epi**  $f$  at  $(x, f(x))$

what if  $f$  is not differentiable?

# Subgradient of a function

$g$  is a **subgradient** of  $f$  (not necessarily convex) at  $x$  if

$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$



$g_2, g_3$  are subgradients at  $x_2$ ;  $g_1$  is a subgradient at  $x_1$

- $g$  is a subgradient of  $f$  at  $x$  iff  $(g, -1)$  supports  $\text{epi } f$  at  $(x, f(x))$
- $g$  is a subgradient iff  $f(x) + g^T(y - x)$  is a global (affine) underestimator of  $f$
- if  $f$  is convex and differentiable,  $\nabla f(x)$  is a subgradient of  $f$  at  $x$

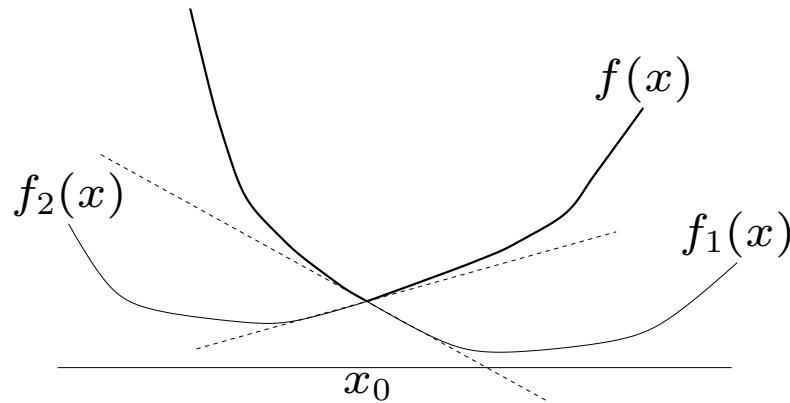
subgradients come up in several contexts:

- algorithms for nondifferentiable convex optimization
- convex analysis, *e.g.*, optimality conditions, duality for nondifferentiable problems

(if  $f(y) \leq f(x) + g^T(y - x)$  for all  $y$ , then  $g$  is a **supergradient**)

## Example

$f = \max\{f_1, f_2\}$ , with  $f_1, f_2$  convex and differentiable



- $f_1(x_0) > f_2(x_0)$ : unique subgradient  $g = \nabla f_1(x_0)$
- $f_2(x_0) > f_1(x_0)$ : unique subgradient  $g = \nabla f_2(x_0)$
- $f_1(x_0) = f_2(x_0)$ : subgradients form a line segment  $[\nabla f_1(x_0), \nabla f_2(x_0)]$

# Subdifferential

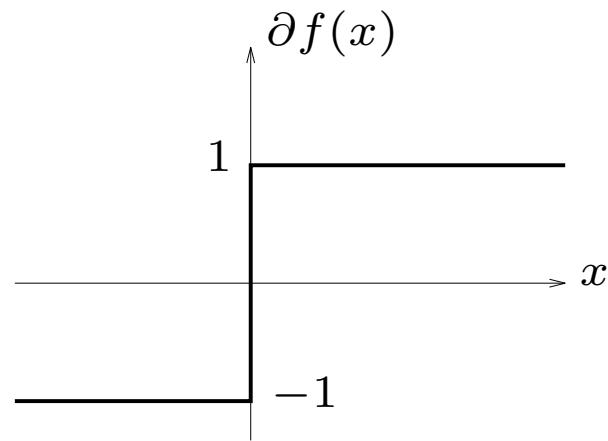
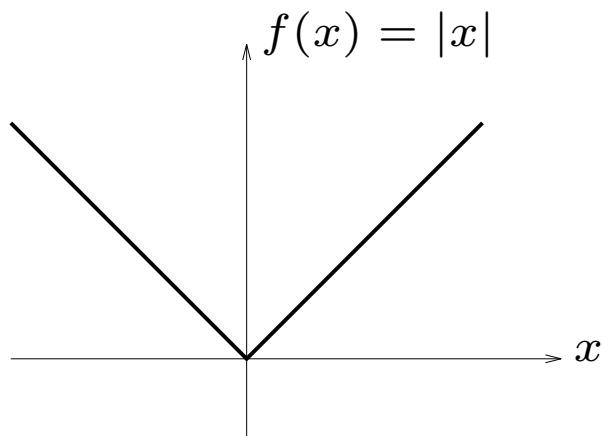
- set of all subgradients of  $f$  at  $x$  is called the **subdifferential** of  $f$  at  $x$ , denoted  $\partial f(x)$
- $\partial f(x)$  is a closed convex set (can be empty)

if  $f$  is convex,

- $\partial f(x)$  is nonempty, for  $x \in \text{relint dom } f$
- $\partial f(x) = \{\nabla f(x)\}$ , if  $f$  is differentiable at  $x$
- if  $\partial f(x) = \{g\}$ , then  $f$  is differentiable at  $x$  and  $g = \nabla f(x)$

## Example

$$f(x) = |x|$$



righthand plot shows  $\bigcup \{(x, g) \mid x \in \mathbb{R}, g \in \partial f(x)\}$

# Subgradient calculus

- **weak subgradient calculus:** formulas for finding *one* subgradient  $g \in \partial f(x)$
- **strong subgradient calculus:** formulas for finding the whole subdifferential  $\partial f(x)$ , *i.e.*, *all* subgradients of  $f$  at  $x$
- many algorithms for nondifferentiable convex optimization require only *one* subgradient at each step, so weak calculus suffices
- some algorithms, optimality conditions, etc., need whole subdifferential
- roughly speaking: if you can compute  $f(x)$ , you can usually compute a  $g \in \partial f(x)$
- we'll assume that  $f$  is convex, and  $x \in \text{relint dom } f$

## Some basic rules

- $\partial f(x) = \{\nabla f(x)\}$  if  $f$  is differentiable at  $x$
- **scaling:**  $\partial(\alpha f) = \alpha \partial f$  (if  $\alpha > 0$ )
- **addition:**  $\partial(f_1 + f_2) = \partial f_1 + \partial f_2$  (RHS is addition of point-to-set mappings)
- **affine transformation of variables:** if  $g(x) = f(Ax + b)$ , then  $\partial g(x) = A^T \partial f(Ax + b)$
- **finite pointwise maximum:** if  $f = \max_{i=1,\dots,m} f_i$ , then

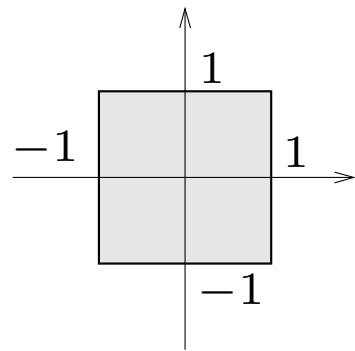
$$\partial f(x) = \text{Co} \bigcup \{\partial f_i(x) \mid f_i(x) = f(x)\},$$

i.e., convex hull of union of subdifferentials of ‘active’ functions at  $x$

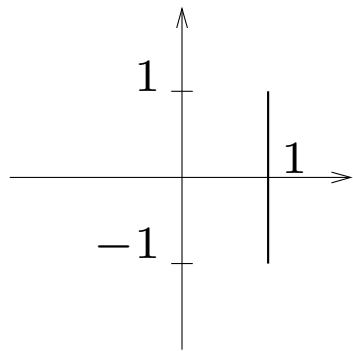
$f(x) = \max\{f_1(x), \dots, f_m(x)\}$ , with  $f_1, \dots, f_m$  differentiable

$$\partial f(x) = \mathbf{Co}\{\nabla f_i(x) \mid f_i(x) = f(x)\}$$

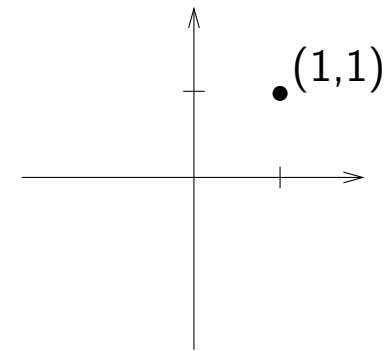
**example:**  $f(x) = \|x\|_1 = \max\{s^T x \mid s_i \in \{-1, 1\}\}$



$\partial f(x)$  at  $x = (0, 0)$



at  $x = (1, 0)$



at  $x = (1, 1)$

## Pointwise supremum

if  $f = \sup_{\alpha \in \mathcal{A}} f_\alpha$ ,

$$\text{cl Co} \bigcup \{\partial f_\beta(x) \mid f_\beta(x) = f(x)\} \subseteq \partial f(x)$$

(usually get equality, but requires some technical conditions to hold, e.g.,  $\mathcal{A}$  compact,  $f_\alpha$  cts in  $x$  and  $\alpha$ )

roughly speaking,  $\partial f(x)$  is closure of convex hull of union of subdifferentials of active functions

## Weak rule for pointwise supremum

$$f = \sup_{\alpha \in \mathcal{A}} f_\alpha$$

- find *any*  $\beta$  for which  $f_\beta(x) = f(x)$  (assuming supremum is achieved)
- choose *any*  $g \in \partial f_\beta(x)$
- then,  $g \in \partial f(x)$

## example

$$f(x) = \lambda_{\max}(A(x)) = \sup_{\|y\|_2=1} y^T A(x) y$$

where  $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$ ,  $A_i \in \mathbf{S}^k$

- $f$  is pointwise supremum of  $g_y(x) = y^T A(x) y$  over  $\|y\|_2 = 1$
- $g_y$  is affine in  $x$ , with  $\nabla g_y(x) = (y^T A_1 y, \dots, y^T A_n y)$
- hence,  $\partial f(x) \supseteq \text{Co} \{ \nabla g_y \mid A(x)y = \lambda_{\max}(A(x))y, \|y\|_2 = 1 \}$   
(in fact equality holds here)

to find **one** subgradient at  $x$ , can choose **any** unit eigenvector  $y$  associated with  $\lambda_{\max}(A(x))$ ; then

$$(y^T A_1 y, \dots, y^T A_n y) \in \partial f(x)$$

# Expectation

- $f(x) = \mathbf{E} f(x, \omega)$ , with  $f$  convex in  $x$  for each  $\omega$ ,  $\omega$  a random variable
- for each  $\omega$ , choose *any*  $g_\omega \in \partial_f(x, \omega)$  (so  $\omega \mapsto g_\omega$  is a function)
- then,  $g = \mathbf{E} g_\omega \in \partial f(x)$

Monte Carlo method for (approximately) computing  $f(x)$  and a  $g \in \partial f(x)$ :

- generate independent samples  $\omega_1, \dots, \omega_K$  from distribution of  $\omega$
- $f(x) \approx (1/K) \sum_{i=1}^K f(x, \omega_i)$
- for each  $i$  choose  $g_i \in \partial_x f(x, \omega_i)$
- $g = (1/K) \sum_{i=1}^K g_i$  is an (approximate) subgradient  
(more on this later)

# Minimization

define  $g(y)$  as the optimal value of

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq y_i, \quad i = 1, \dots, m \end{aligned}$$

( $f_i$  convex; variable  $x$ )

with  $\lambda^*$  an optimal dual variable, we have

$$g(z) \geq g(y) - \sum_{i=1}^m \lambda_i^*(z_i - y_i)$$

i.e.,  $-\lambda^*$  is a subgradient of  $g$  at  $y$

# Composition

- $f(x) = h(f_1(x), \dots, f_k(x))$ , with  $h$  convex nondecreasing,  $f_i$  convex
- find  $q \in \partial h(f_1(x), \dots, f_k(x))$ ,  $g_i \in \partial f_i(x)$
- then,  $g = q_1 g_1 + \dots + q_k g_k \in \partial f(x)$
- reduces to standard formula for differentiable  $h, f_i$

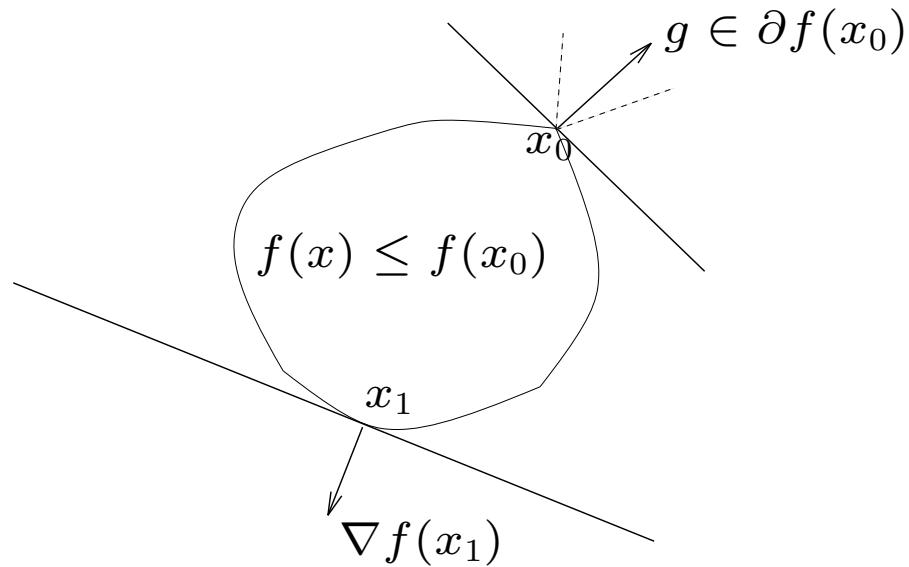
proof:

$$\begin{aligned}f(y) &= h(f_1(y), \dots, f_k(y)) \\&\geq h(f_1(x) + g_1^T(y - x), \dots, f_k(x) + g_k^T(y - x)) \\&\geq h(f_1(x), \dots, f_k(x)) + q^T(g_1^T(y - x), \dots, g_k^T(y - x)) \\&= f(x) + g^T(y - x)\end{aligned}$$

## Subgradients and sublevel sets

$g$  is a subgradient at  $x$  means  $f(y) \geq f(x) + g^T(y - x)$

hence  $f(y) \leq f(x) \implies g^T(y - x) \leq 0$



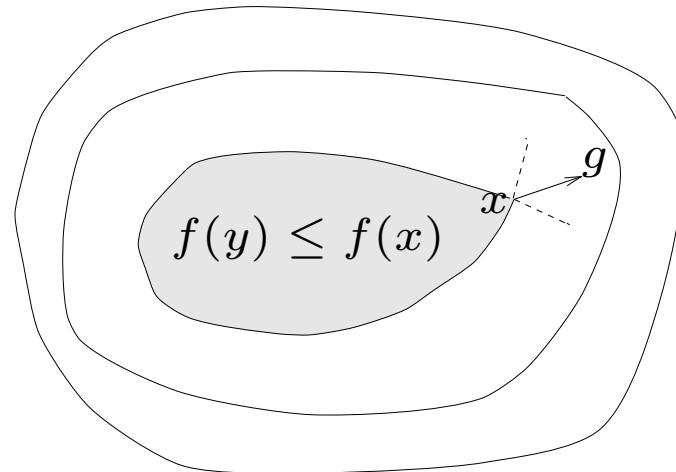
- $f$  differentiable at  $x_0$ :  $\nabla f(x_0)$  is normal to the sublevel set  $\{x \mid f(x) \leq f(x_0)\}$
- $f$  nondifferentiable at  $x_0$ : subgradient defines a supporting hyperplane to sublevel set through  $x_0$

# Quasigradients

$g \neq 0$  is a **quasigradient** of  $f$  at  $x$  if

$$g^T(y - x) \geq 0 \implies f(y) \geq f(x)$$

holds for all  $y$



quasigradients at  $x$  form a cone

**example:**

$$f(x) = \frac{a^T x + b}{c^T x + d}, \quad (\mathbf{dom} f = \{x \mid c^T x + d > 0\})$$

$g = a - f(x_0)c$  is a quasigradient at  $x_0$

proof: for  $c^T x + d > 0$ :

$$a^T(x - x_0) \geq f(x_0)c^T(x - x_0) \implies f(x) \geq f(x_0)$$

**example:** degree of  $a_1 + a_2t + \cdots + a_nt^{n-1}$

$$f(a) = \min\{i \mid a_{i+2} = \cdots = a_n = 0\}$$

$g = \text{sign}(a_{k+1})e_{k+1}$  (with  $k = f(a)$ ) is a quasigradient at  $a \neq 0$

proof:

$$g^T(b - a) = \text{sign}(a_{k+1})b_{k+1} - |a_{k+1}| \geq 0$$

implies  $b_{k+1} \neq 0$

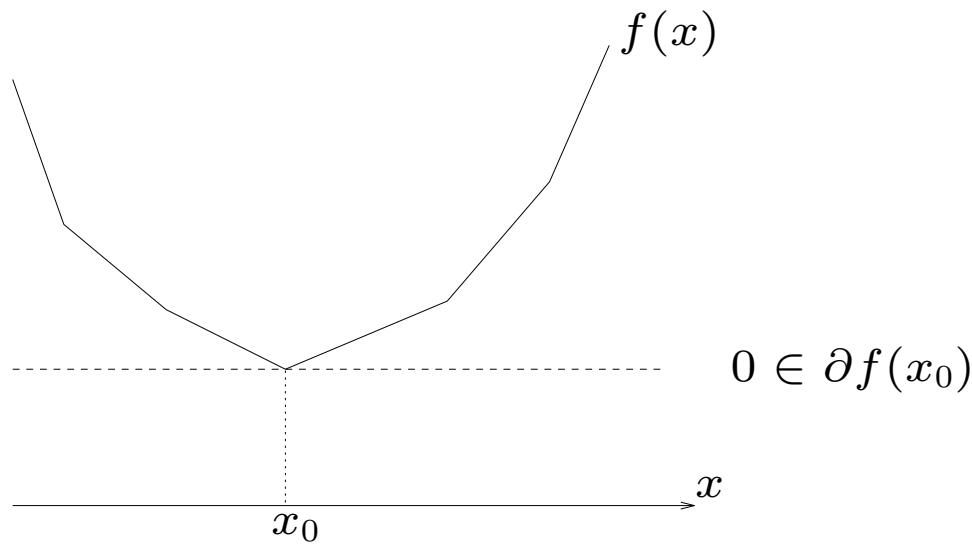
## Optimality conditions — unconstrained

recall for  $f$  convex, differentiable,

$$f(x^*) = \inf_x f(x) \iff 0 = \nabla f(x^*)$$

generalization to nondifferentiable convex  $f$ :

$$f(x^*) = \inf_x f(x) \iff 0 \in \partial f(x^*)$$



**proof.** by definition (!)

$$f(y) \geq f(x^*) + 0^T(y - x^*) \text{ for all } y \iff 0 \in \partial f(x^*)$$

. . . seems trivial but isn't

## Example: piecewise linear minimization

$$f(x) = \max_{i=1,\dots,m} (a_i^T x + b_i)$$

$x^*$  minimizes  $f \iff 0 \in \partial f(x^*) = \text{Co}\{a_i \mid a_i^T x^* + b_i = f(x^*)\}$

$\iff$  there is a  $\lambda$  with

$$\lambda \succeq 0, \quad \mathbf{1}^T \lambda = 1, \quad \sum_{i=1}^m \lambda_i a_i = 0$$

where  $\lambda_i = 0$  if  $a_i^T x^* + b_i < f(x^*)$

. . . but these are the KKT conditions for the epigraph form

$$\begin{aligned} & \text{minimize} && t \\ & \text{subject to} && a_i^T x + b_i \leq t, \quad i = 1, \dots, m \end{aligned}$$

with dual

$$\begin{aligned} & \text{maximize} && b^T \lambda \\ & \text{subject to} && \lambda \succeq 0, \quad A^T \lambda = 0, \quad \mathbf{1}^T \lambda = 1 \end{aligned}$$

## Optimality conditions — constrained

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

we assume

- $f_i$  convex, defined on  $\mathbf{R}^n$  (hence subdifferentiable)
- strict feasibility (Slater's condition)

$x^*$  is primal optimal ( $\lambda^*$  is dual optimal) iff

$$\begin{aligned} & f_i(x^*) \leq 0, \quad \lambda_i^* \geq 0 \\ & 0 \in \partial f_0(x^*) + \sum_{i=1}^m \lambda_i^* \partial f_i(x^*) \\ & \lambda_i^* f_i(x^*) = 0 \end{aligned}$$

. . . generalizes KKT for nondifferentiable  $f_i$

## Directional derivative

**directional derivative** of  $f$  at  $x$  in the direction  $\delta x$  is

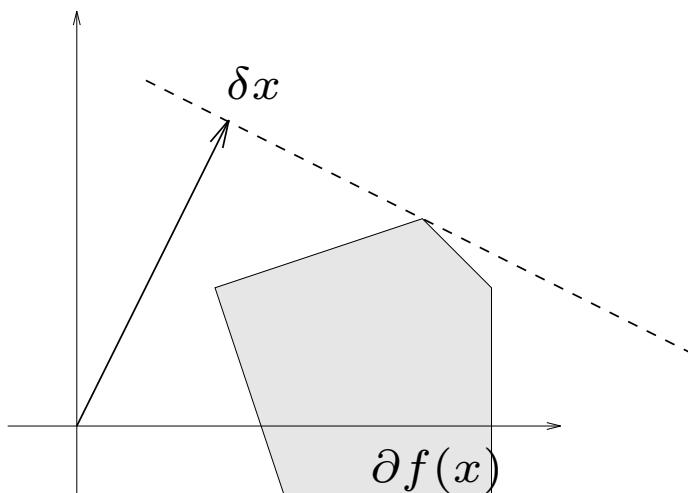
$$f'(x; \delta x) \triangleq \lim_{h \searrow 0} \frac{f(x + h\delta x) - f(x)}{h}$$

can be  $+\infty$  or  $-\infty$

- $f$  convex, finite near  $x \implies f'(x; \delta x)$  exists
- $f$  differentiable at  $x$  if and only if, for some  $g$  ( $= \nabla f(x)$ ) and all  $\delta x$ ,  
 $f'(x; \delta x) = g^T \delta x$  (i.e.,  $f'(x; \delta x)$  is a linear function of  $\delta x$ )

## Directional derivative and subdifferential

general formula for convex  $f$ :  $f'(x; \delta x) = \sup_{g \in \partial f(x)} g^T \delta x$



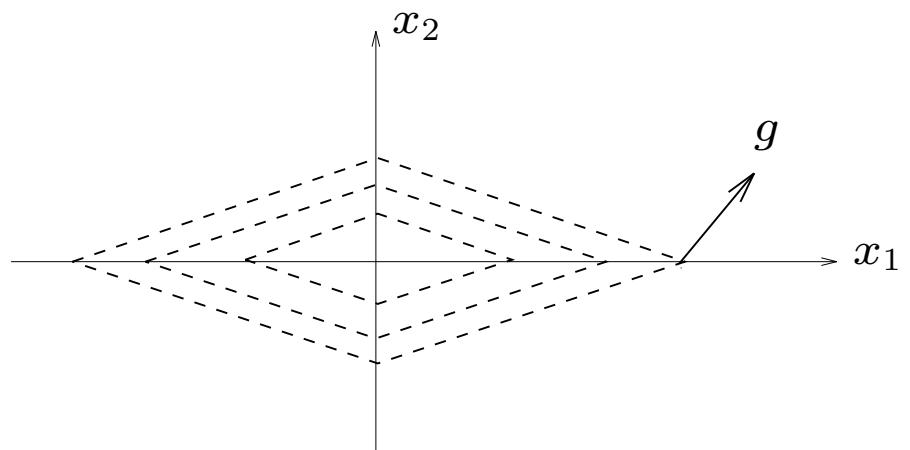
## Descent directions

$\delta x$  is a **descent direction** for  $f$  at  $x$  if  $f'(x; \delta x) < 0$

for differentiable  $f$ ,  $\delta x = -\nabla f(x)$  is always a descent direction (except when it is zero)

**warning:** for nondifferentiable (convex) functions,  $\delta x = -g$ , with  $g \in \partial f(x)$ , need not be descent direction

example:  $f(x) = |x_1| + 2|x_2|$



## Subgradients and distance to sublevel sets

if  $f$  is convex,  $f(z) < f(x)$ ,  $g \in \partial f(x)$ , then for small  $t > 0$ ,

$$\|x - tg - z\|_2 < \|x - z\|_2$$

thus  $-g$  is descent direction for  $\|x - z\|_2$ , for **any**  $z$  with  $f(z) < f(x)$  (e.g.,  $x^*$ )

negative subgradient is descent direction for distance to optimal point

$$\begin{aligned} \text{proof: } \|x - tg - z\|_2^2 &= \|x - z\|_2^2 - 2tg^T(x - z) + t^2\|g\|_2^2 \\ &\leq \|x - z\|_2^2 - 2t(f(x) - f(z)) + t^2\|g\|_2^2 \end{aligned}$$

## Descent directions and optimality

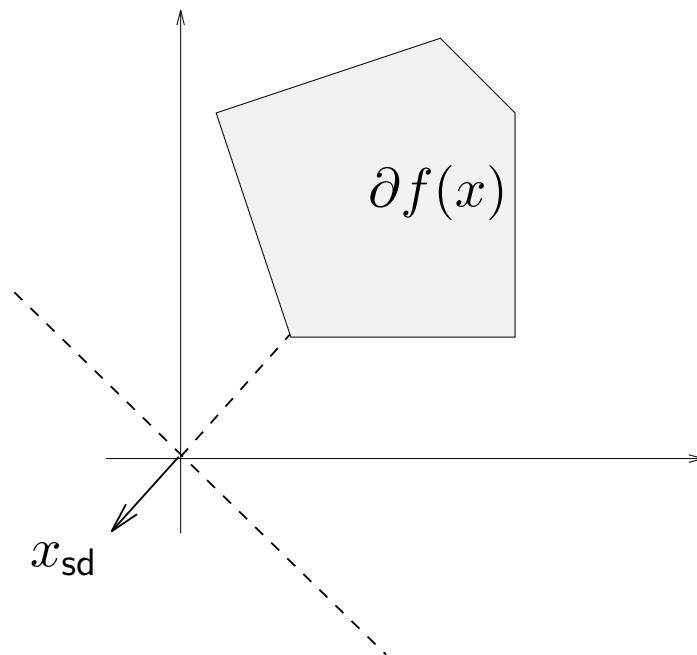
**fact:** for  $f$  convex, finite near  $x$ , either

- $0 \in \partial f(x)$  (in which case  $x$  minimizes  $f$ ), or
- there is a descent direction for  $f$  at  $x$

i.e.,  $x$  is optimal (minimizes  $f$ ) iff there is no descent direction for  $f$  at  $x$

**proof:** define  $\delta x_{\text{sd}} = - \underset{z \in \partial f(x)}{\operatorname{argmin}} \|z\|_2$

if  $\delta x_{\text{sd}} = 0$ , then  $0 \in \partial f(x)$ , so  $x$  is optimal; otherwise  
 $f'(x; \delta x_{\text{sd}}) = - (\inf_{z \in \partial f(x)} \|z\|_2)^2 < 0$ , so  $\delta x_{\text{sd}}$  is a descent direction



idea extends to constrained case (feasible descent direction)

# **Subgradient Methods**

- subgradient method and stepsize rules
- convergence results and proof
- optimal step size and alternating projections
- speeding up subgradient methods

## Subgradient method

**subgradient method** is simple algorithm to minimize nondifferentiable convex function  $f$

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$$

- $x^{(k)}$  is the  $k$ th iterate
- $g^{(k)}$  is **any** subgradient of  $f$  at  $x^{(k)}$
- $\alpha_k > 0$  is the  $k$ th step size

not a descent method, so we keep track of best point so far

$$f_{\text{best}}^{(k)} = \min_{i=1,\dots,k} f(x^{(i)})$$

## Step size rules

step sizes are fixed ahead of time

- *constant step size*:  $\alpha_k = \alpha$  (constant)
- *constant step length*:  $\alpha_k = \gamma/\|g^{(k)}\|_2$  (so  $\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$ )
- *square summable but not summable*: step sizes satisfy

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- *nonsummable diminishing*: step sizes satisfy

$$\lim_{k \rightarrow \infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

## Assumptions

- $f^* = \inf_x f(x) > -\infty$ , with  $f(x^*) = f^*$
- $\|g\|_2 \leq G$  for all  $g \in \partial f$  (equivalent to Lipschitz condition on  $f$ )
- $\|x^{(1)} - x^*\|_2 \leq R$

these assumptions are stronger than needed, just to simplify proofs

## Convergence results

define  $\bar{f} = \lim_{k \rightarrow \infty} f_{\text{best}}^{(k)}$

- *constant step size*:  $\bar{f} - f^* \leq G^2\alpha/2$ , i.e.,  
**converges to  $G^2\alpha/2$ -suboptimal**  
(converges to  $f^*$  if  $f$  differentiable,  $\alpha$  small enough)
- *constant step length*:  $\bar{f} - f^* \leq G\gamma/2$ , i.e.,  
**converges to  $G\gamma/2$ -suboptimal**
- *diminishing step size rule*:  $\bar{f} = f^*$ , i.e., **converges**

## Convergence proof

**key quantity:** *Euclidean distance to the optimal set*, not the function value

let  $x^*$  be any minimizer of  $f$

$$\begin{aligned}\|x^{(k+1)} - x^*\|_2^2 &= \|x^{(k)} - \alpha_k g^{(k)} - x^*\|_2^2 \\ &= \|x^{(k)} - x^*\|_2^2 - 2\alpha_k g^{(k)T} (x^{(k)} - x^*) + \alpha_k^2 \|g^{(k)}\|_2^2 \\ &\leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^*) + \alpha_k^2 \|g^{(k)}\|_2^2\end{aligned}$$

using  $f^* = f(x^*) \geq f(x^{(k)}) + g^{(k)T} (x^* - x^{(k)})$

apply recursively to get

$$\begin{aligned}
\|x^{(k+1)} - x^*\|_2^2 &\leq \|x^{(1)} - x^*\|_2^2 - 2 \sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2 \\
&\leq R^2 - 2 \sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) + G^2 \sum_{i=1}^k \alpha_i^2
\end{aligned}$$

now we use

$$\sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) \geq (f_{\text{best}}^{(k)} - f^*) \left( \sum_{i=1}^k \alpha_i \right)$$

to get

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}.$$

**constant step size:** for  $\alpha_k = \alpha$  we get

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 k \alpha^2}{2k\alpha}$$

righthand side converges to  $G^2\alpha/2$  as  $k \rightarrow \infty$

**constant step length:** for  $\alpha_k = \gamma/\|g^{(k)}\|_2$  we get

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2}{2 \sum_{i=1}^k \alpha_i} \leq \frac{R^2 + \gamma^2 k}{2\gamma k/G},$$

righthand side converges to  $G\gamma/2$  as  $k \rightarrow \infty$

**square summable but not summable step sizes:**

suppose step sizes satisfy

$$\sum_{i=1}^{\infty} \alpha_i^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

then

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}$$

as  $k \rightarrow \infty$ , numerator converges to a finite number, denominator converges to  $\infty$ , so  $f_{\text{best}}^{(k)} \rightarrow f^*$

## Stopping criterion

- terminating when  $\frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i} \leq \epsilon$  is really, really, slow
- optimal choice of  $\alpha_i$  to achieve  $\frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i} \leq \epsilon$  for smallest  $k$ :

$$\alpha_i = (R/G)/\sqrt{k}, \quad i = 1, \dots, k$$

number of steps required:  $k = (RG/\epsilon)^2$

- the truth: there really isn't a good stopping criterion for the subgradient method . . .

## Example: Piecewise linear minimization

$$\text{minimize } f(x) = \max_{i=1,\dots,m} (a_i^T x + b_i)$$

to find a subgradient of  $f$ : find index  $j$  for which

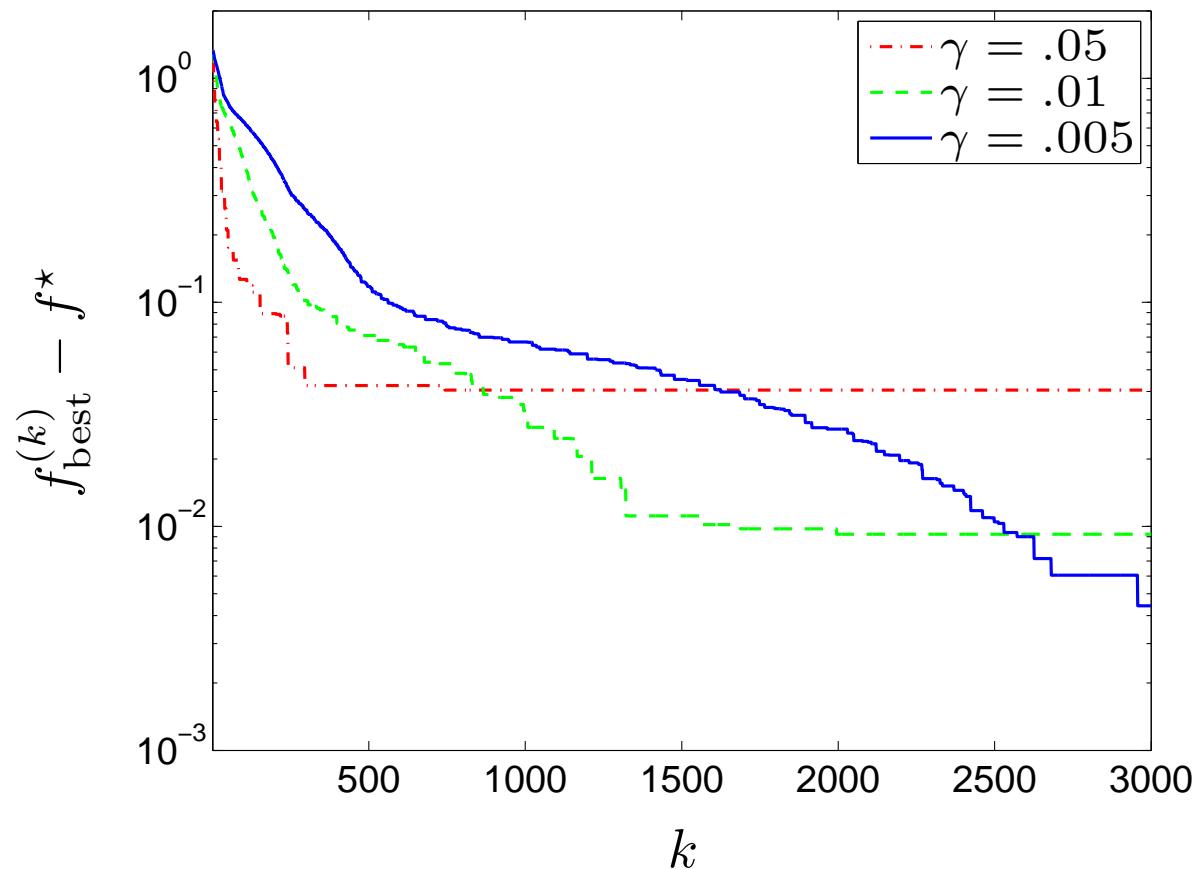
$$a_j^T x + b_j = \max_{i=1,\dots,m} (a_i^T x + b_i)$$

and take  $g = a_j$

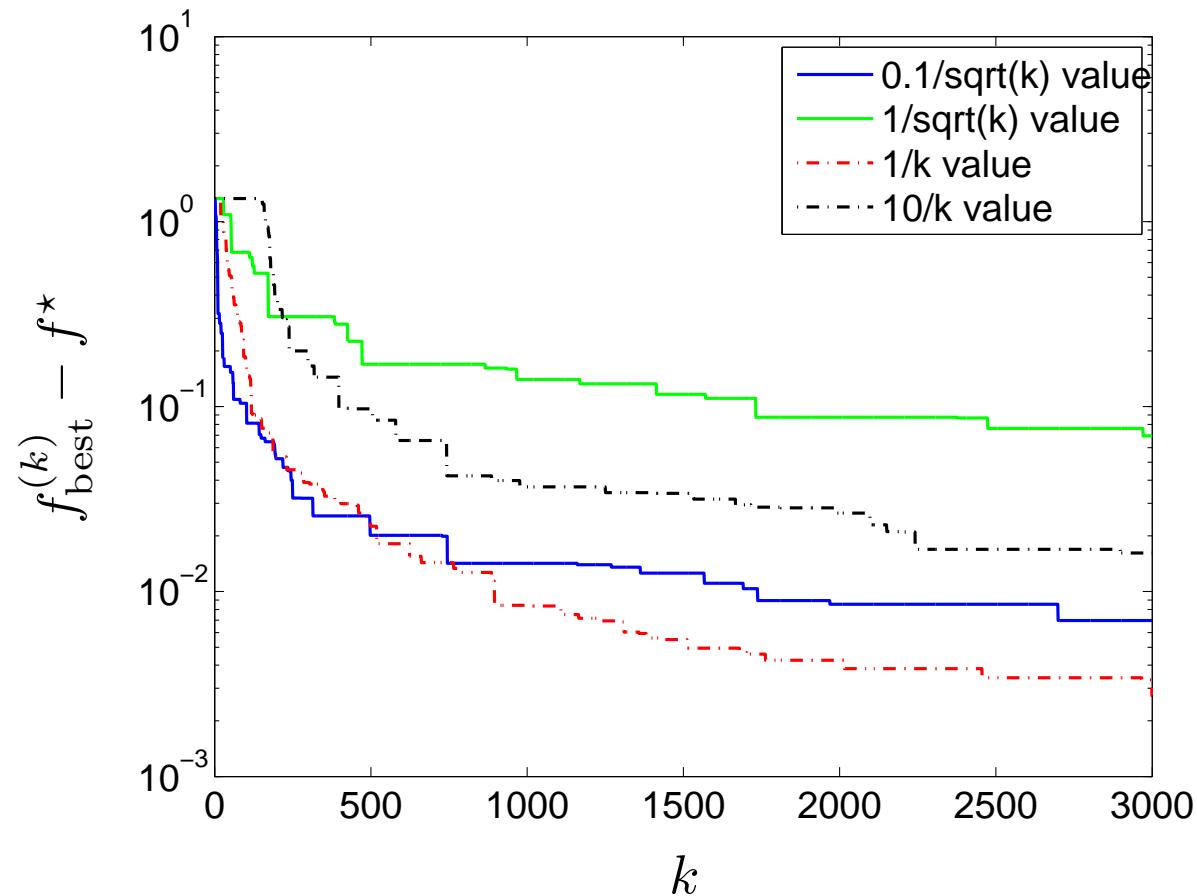
subgradient method:  $x^{(k+1)} = x^{(k)} - \alpha_k a_j$

problem instance with  $n = 20$  variables,  $m = 100$  terms,  $f^* \approx 1.1$

$f_{\text{best}}^{(k)} - f^*$ , constant step length  $\gamma = 0.05, 0.01, 0.005$



diminishing step rules  $\alpha_k = 0.1/\sqrt{k}$  and  $\alpha_k = 1/\sqrt{k}$ , square summable  
step size rules  $\alpha_k = 1/k$  and  $\alpha_k = 10/k$



## Optimal step size when $f^*$ is known

- choice due to Polyak:

$$\alpha_k = \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}$$

(can also use when optimal value is estimated)

- motivation: start with basic inequality

$$\|x^{(k+1)} - x^*\|_2^2 \leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k(f(x^{(k)}) - f^*) + \alpha_k^2\|g^{(k)}\|_2^2$$

and choose  $\alpha_k$  to minimize righthand side

- yields

$$\|x^{(k+1)} - x^*\|_2^2 \leq \|x^{(k)} - x^*\|_2^2 - \frac{(f(x^{(k)}) - f^*)^2}{\|g^{(k)}\|_2^2}$$

(in particular,  $\|x^{(k)} - x^*\|_2$  decreases each step)

- applying recursively,

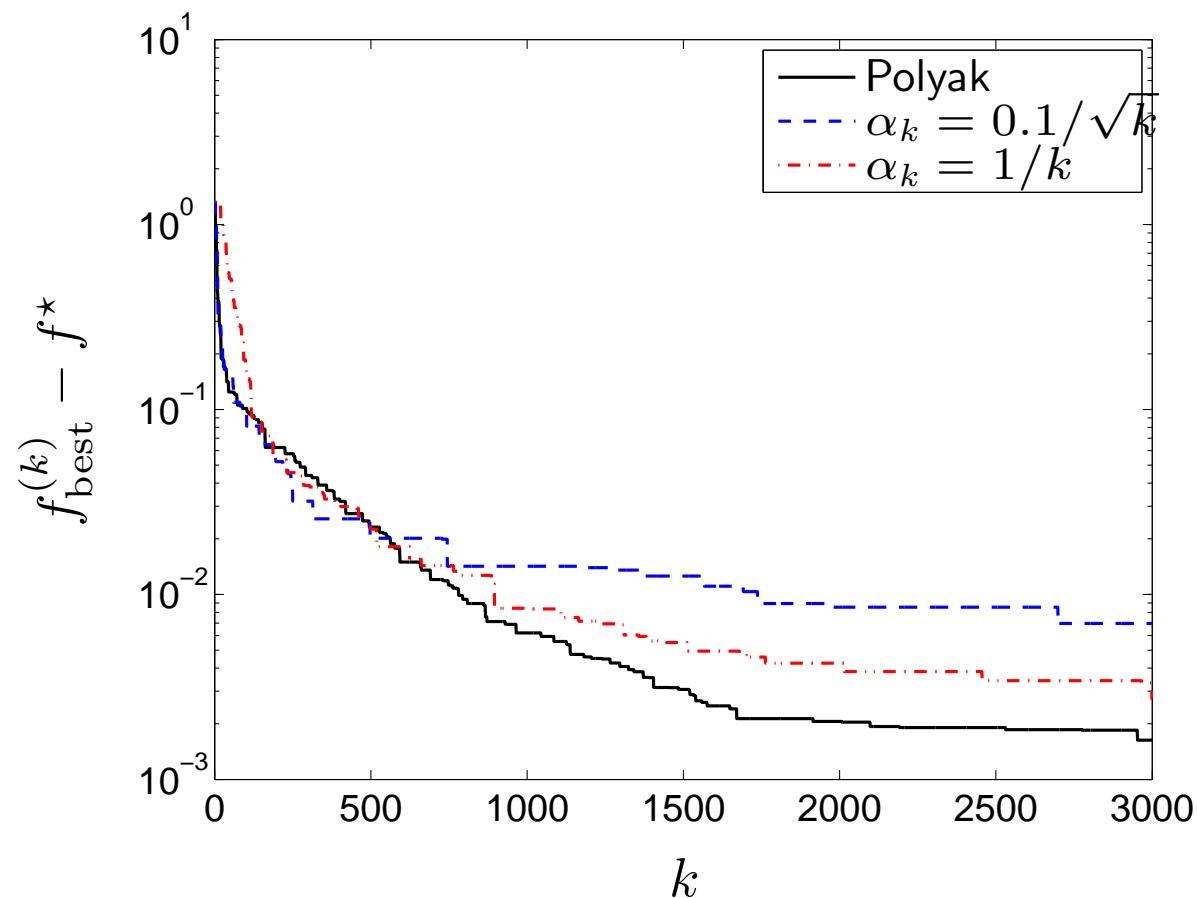
$$\sum_{i=1}^k \frac{(f(x^{(i)}) - f^*)^2}{\|g^{(i)}\|_2^2} \leq R^2$$

and so

$$\sum_{i=1}^k (f(x^{(i)}) - f^*)^2 \leq R^2 G^2$$

which proves  $f(x^{(k)}) \rightarrow f^*$

PWL example with Polyak's step size,  $\alpha_k = 0.1/\sqrt{k}$ ,  $\alpha_k = 1/k$



## Finding a point in the intersection of convex sets

$C = C_1 \cap \cdots C_m$  is nonempty,  $C_1, \dots, C_m \subseteq \mathbf{R}^n$  closed and convex

find a point in  $C$  by minimizing

$$f(x) = \max\{\mathbf{dist}(x, C_1), \dots, \mathbf{dist}(x, C_m)\}$$

with  $\mathbf{dist}(x, C_j) = f(x)$ , a subgradient of  $f$  is

$$g = \nabla \mathbf{dist}(x, C_j) = \frac{x - P_{C_j}(x)}{\|x - P_{C_j}(x)\|_2}$$

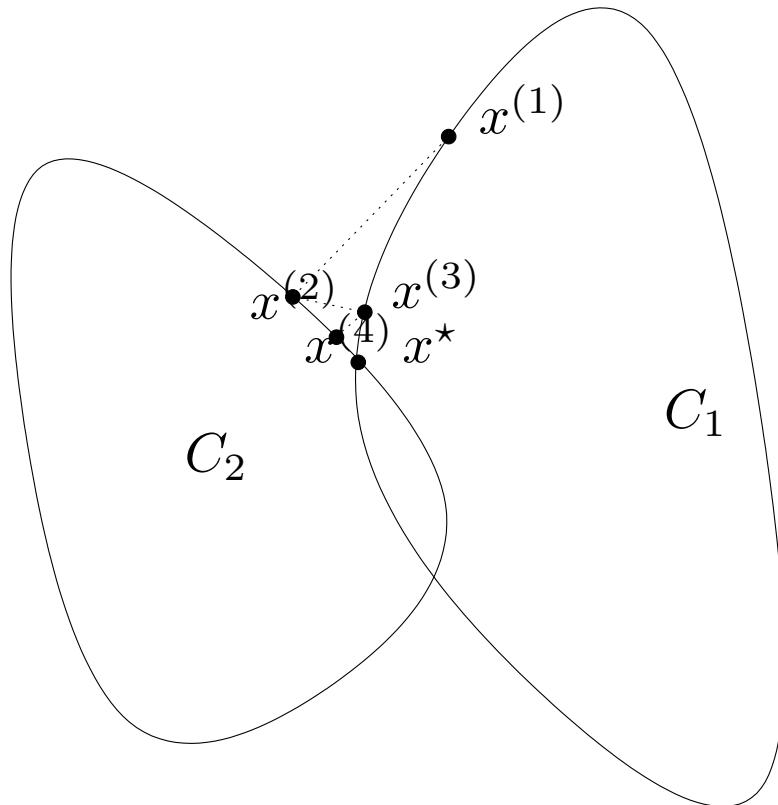
subgradient update with optimal step size:

$$\begin{aligned}x^{(k+1)} &= x^{(k)} - \alpha_k g^{(k)} \\&= x^{(k)} - f(x^{(k)}) \frac{x - P_{C_j}(x)}{\|x - P_{C_j}(x)\|_2} \\&= P_{C_j}(x^{(k)})\end{aligned}$$

- a version of the famous *alternating projections* algorithm
- at each step, project the current point onto the farthest set
- for  $m = 2$  sets, projections alternate onto one set, then the other
- convergence:  $\text{dist}(x^{(k)}, C) \rightarrow 0$  as  $k \rightarrow \infty$

# Alternating projections

first few iterations:



...  $x^{(k)}$  eventually converges to a point  $x^* \in C_1 \cap C_2$

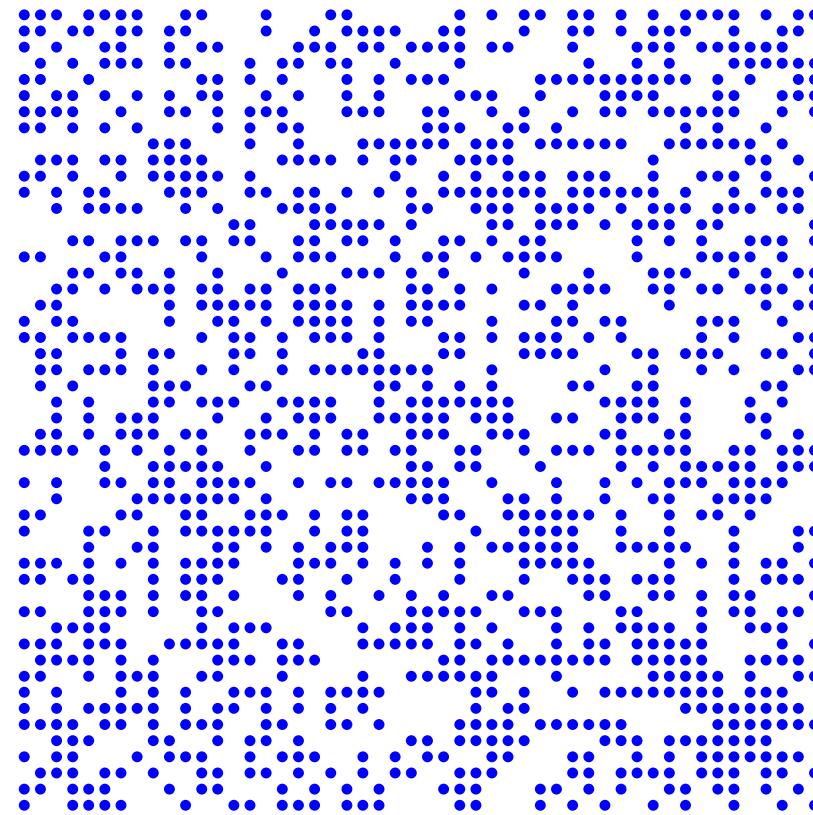
## Example: Positive semidefinite matrix completion

- some entries of matrix in  $\mathbf{S}^n$  fixed; find values for others so completed matrix is PSD
- $C_1 = \mathbf{S}_+^n$ ,  $C_2$  is (affine) set in  $\mathbf{S}^n$  with specified fixed entries
- projection onto  $C_1$  by eigenvalue decomposition, truncation: for  $X = \sum_{i=1}^n \lambda_i q_i q_i^T$ ,

$$P_{C_1}(X) = \sum_{i=1}^n \max\{0, \lambda_i\} q_i q_i^T$$

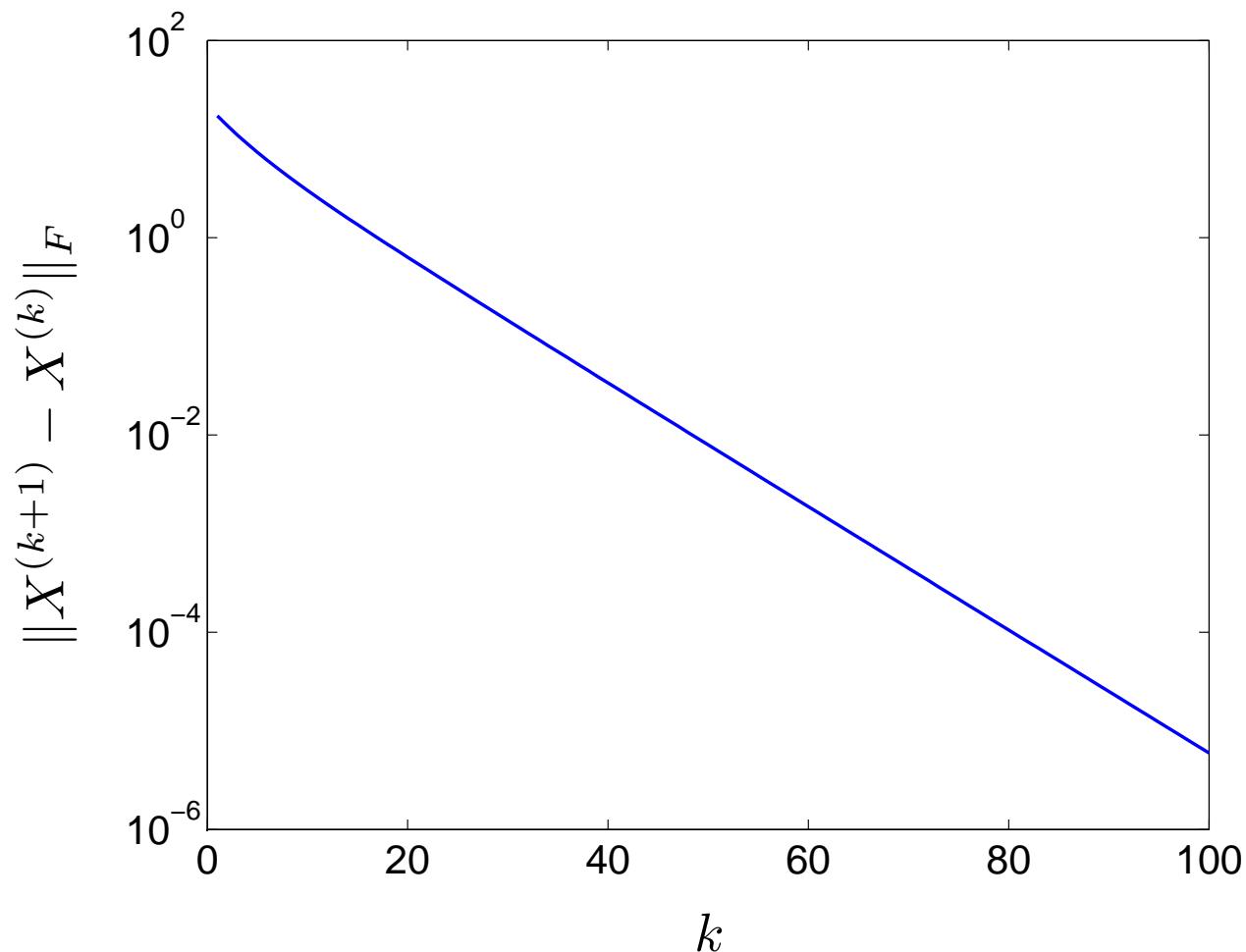
- projection of  $X$  onto  $C_2$  by re-setting specified entries to fixed values

specific example:  $50 \times 50$  matrix missing about half of its entries



- initialize  $X^{(1)}$  with unknown entries set to 0

convergence is linear:



## Polyak step size when $f^*$ isn't known

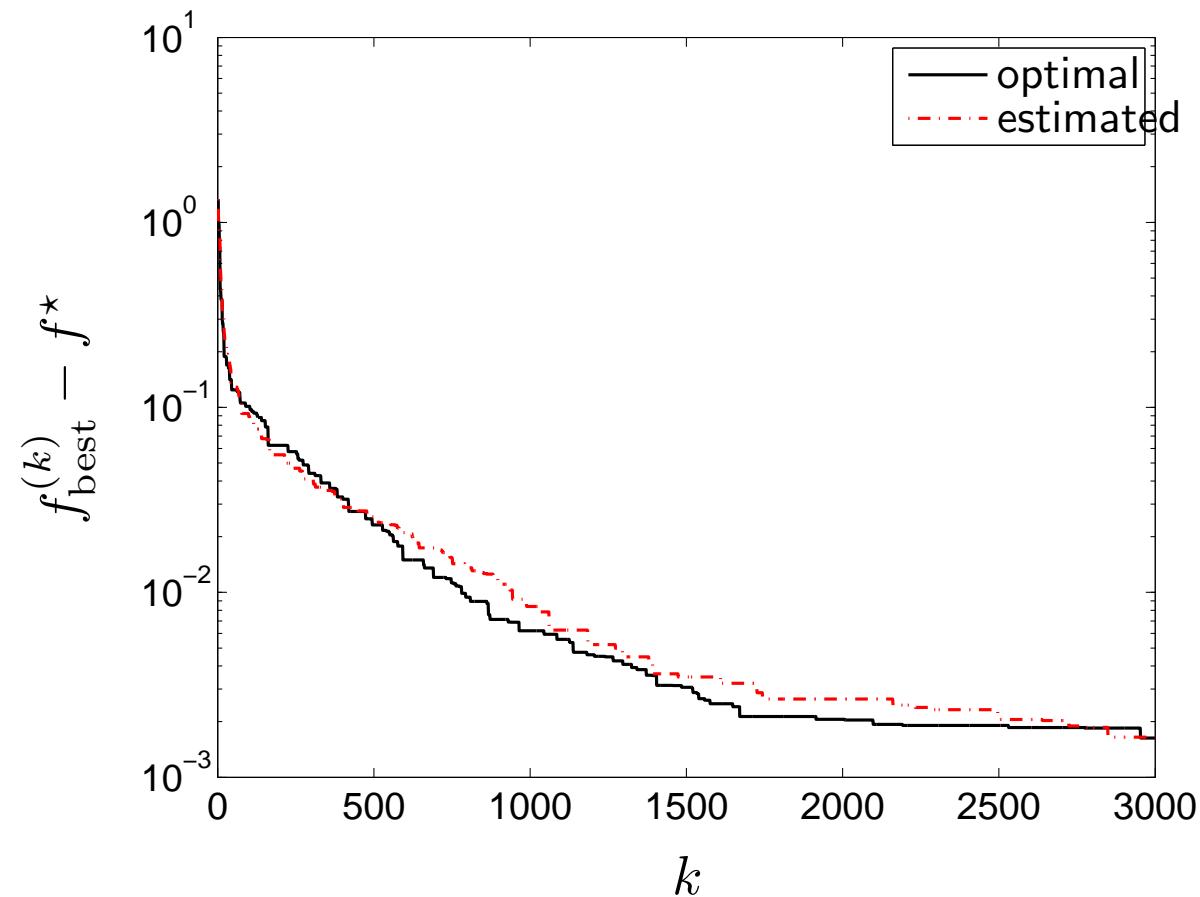
- use step size

$$\alpha_k = \frac{f(x^{(k)}) - f_{\text{best}}^{(k)} + \gamma_k}{\|g^{(k)}\|_2^2}$$

with  $\sum_{k=1}^{\infty} \gamma_k = \infty$ ,  $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$

- $f_{\text{best}}^{(k)} - \gamma_k$  serves as estimate of  $f^*$
- $\gamma_k$  is in scale of objective value
- can show  $f_{\text{best}}^{(k)} \rightarrow f^*$

PWL example with Polyak's step size, using  $f^*$ , and estimated with  $\gamma_k = 10/(10 + k)$



## Speeding up subgradient methods

- subgradient methods are very slow
- often convergence can be improved by keeping memory of past steps

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

(heavy ball method)

**other ideas:** localization methods, conjugate directions, . . .

## A couple of speedup algorithms

$$x^{(k+1)} = x^{(k)} - \alpha_k s^{(k)}, \quad \alpha_k = \frac{f(x^{(k)}) - f^*}{\|s^{(k)}\|_2^2}$$

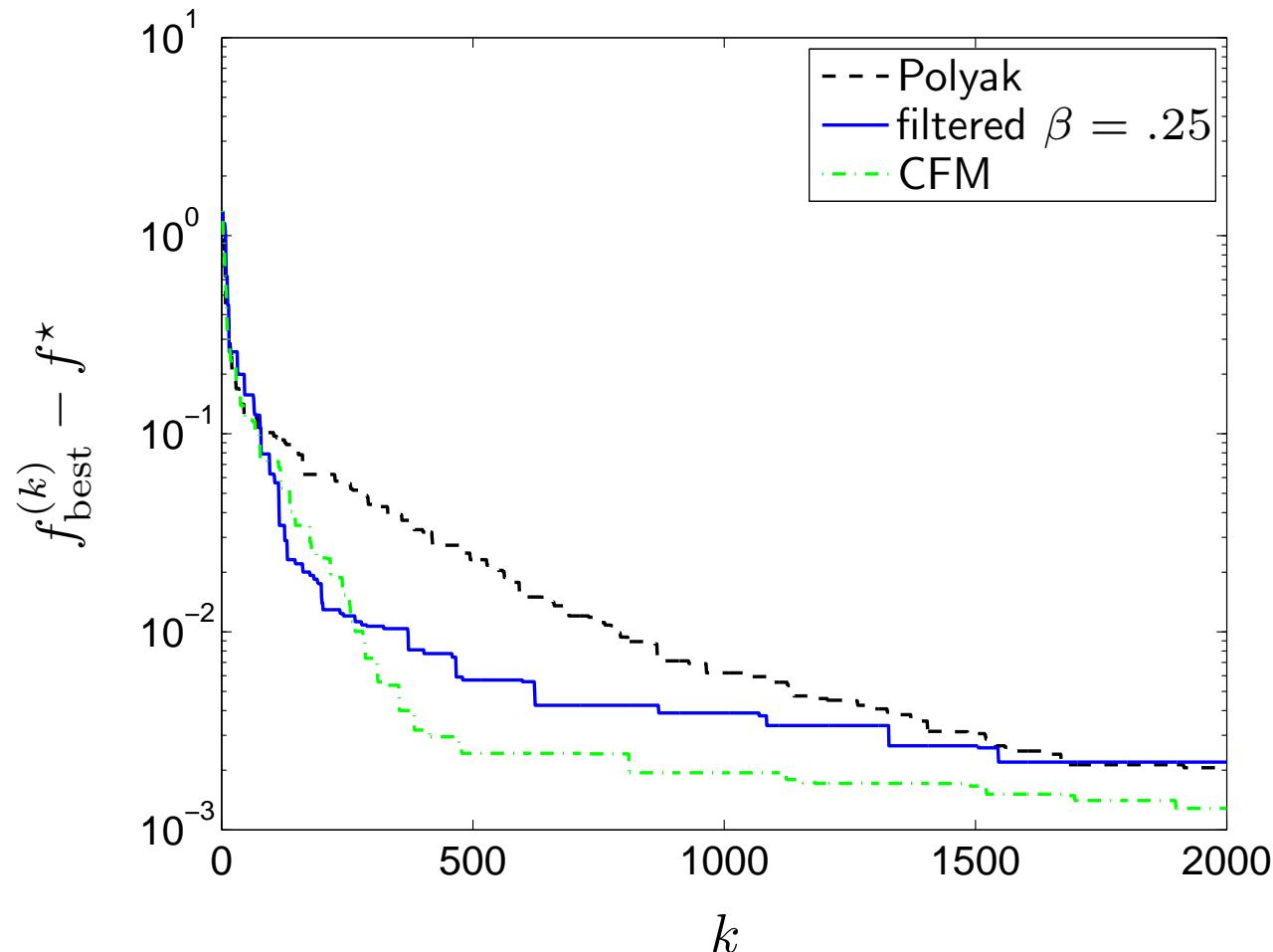
(we assume  $f^*$  is known or can be estimated)

- ‘filtered’ subgradient,  $s^{(k)} = (1 - \beta)g^{(k)} + \beta s^{(k-1)}$ , where  $\beta \in [0, 1]$
- Camerini, Fratta, and Maffioli (1975)

$$s^{(k)} = g^{(k)} + \beta_k s^{(k-1)}, \quad \beta_k = \max\{0, -\gamma_k (s^{(k-1)})^T g^{(k)} / \|s^{(k-1)}\|_2^2\}$$

where  $\gamma_k \in [0, 2)$  ( $\gamma_k = 1.5$  ‘recommended’)

## PWL example, Polyak's step, filtered subgradient, CFM step



# **Subgradient Methods for Constrained Problems**

- projected subgradient method
- projected subgradient for dual
- subgradient method for constrained optimization

# **Projected subgradient method**

solves constrained optimization problem

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C}, \end{aligned}$$

where  $f : \mathbf{R}^n \rightarrow \mathbf{R}$ ,  $\mathcal{C} \subseteq \mathbf{R}^n$  are convex

**projected subgradient method** is given by

$$x^{(k+1)} = \Pi(x^{(k)} - \alpha_k g^{(k)}),$$

$\Pi$  is (Euclidean) projection on  $\mathcal{C}$ , and  $g^{(k)} \in \partial f(x^{(k)})$

same convergence results:

- for constant step size, converges to neighborhood of optimal  
(for  $f$  differentiable and  $h$  small enough, converges)
- for diminishing nonsummable step sizes, converges

**key idea:** projection does not increase distance to  $x^*$

## Linear equality constraints

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && Ax = b \end{aligned}$$

projection of  $z$  onto  $\{x \mid Ax = b\}$  is

$$\begin{aligned} \Pi(z) &= z - A^T(AA^T)^{-1}(Az - b) \\ &= (I - A^T(AA^T)^{-1}A)z + A^T(AA^T)^{-1}b \end{aligned}$$

projected subgradient update is (using  $Ax^{(k)} = b$ )

$$\begin{aligned} x^{(k+1)} &= \Pi(x^{(k)} - \alpha_k g^{(k)}) \\ &= x^{(k)} - \alpha_k (I - A^T(AA^T)^{-1}A)g^{(k)} \\ &= x^{(k)} - \alpha_k \Pi_{\mathcal{N}(A)}(g^{(k)}) \end{aligned}$$

## Example: Least $l_1$ -norm

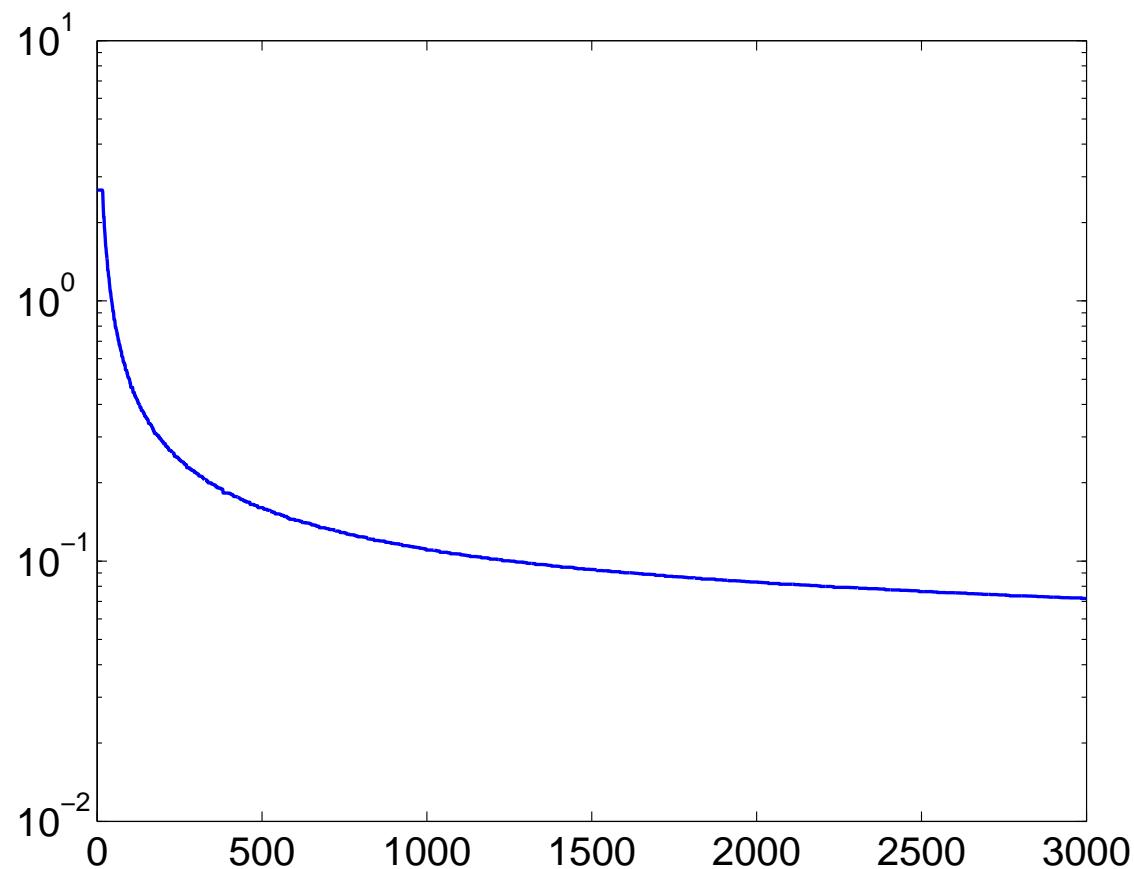
$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && Ax = b \end{aligned}$$

subgradient of objective is  $g = \mathbf{sign}(x)$

projected subgradient update is

$$x^{(k+1)} = x^{(k)} - \alpha_k (I - A^T (A A^T)^{-1} A) \mathbf{sign}(x^{(k)})$$

problem instance with  $n = 1000$ ,  $m = 50$ , step size  $\alpha_k = 0.1/k$ ,  $f^* \approx 3.2$



# Projected subgradient for dual problem

(convex) primal:

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

solve dual problem

$$\begin{aligned} & \text{maximize} && g(\lambda) \\ & \text{subject to} && \lambda \succeq 0 \end{aligned}$$

via projected subgradient method:

$$\lambda^{(k+1)} = \left( \lambda^{(k)} - \alpha_k h \right)_+, \quad h \in \partial(-g)(\lambda^{(k)})$$

## Subgradient of negative dual function

assume  $f_0$  is strictly convex, and denote, for  $\lambda \succeq 0$ ,

$$x^*(\lambda) = \operatorname{argmin}_z (f_0(z) + \lambda_1 f_1(z) + \cdots + \lambda_m f_m(z))$$

$$\text{so } g(\lambda) = f_0(x^*(\lambda)) + \lambda_1 f_1(x^*(\lambda)) + \cdots + \lambda_m f_m(x^*(\lambda))$$

a subgradient of  $-g$  at  $\lambda$  is given by  $h_i = -f_i(x^*(\lambda))$

projected subgradient method for dual:

$$x^{(k)} = x^*(\lambda^{(k)}), \quad \lambda_i^{(k+1)} = \left( \lambda_i^{(k)} + \alpha_k f_i(x^{(k)}) \right)_+$$

- primal iterates  $x^{(k)}$  are not feasible, but become feasible in limit (sometimes can find feasible, suboptimal  $\tilde{x}^{(k)}$  from  $x^{(k)}$ )
- dual function values  $g(\lambda^{(k)})$  converge to  $f^* = f_0(x^*)$

interpretation:

- $\lambda_i$  is price for ‘resource’  $f_i(x)$
- price update  $\lambda_i^{(k+1)} = \left( \lambda_i^{(k)} + \alpha_k f_i(x^{(k)}) \right)_+$ 
  - increase price  $\lambda_i$  if resource  $i$  is over-utilized (*i.e.*,  $f_i(x) > 0$ )
  - decrease price  $\lambda_i$  if resource  $i$  is under-utilized (*i.e.*,  $f_i(x) < 0$ )
  - but never let prices get negative

## Example

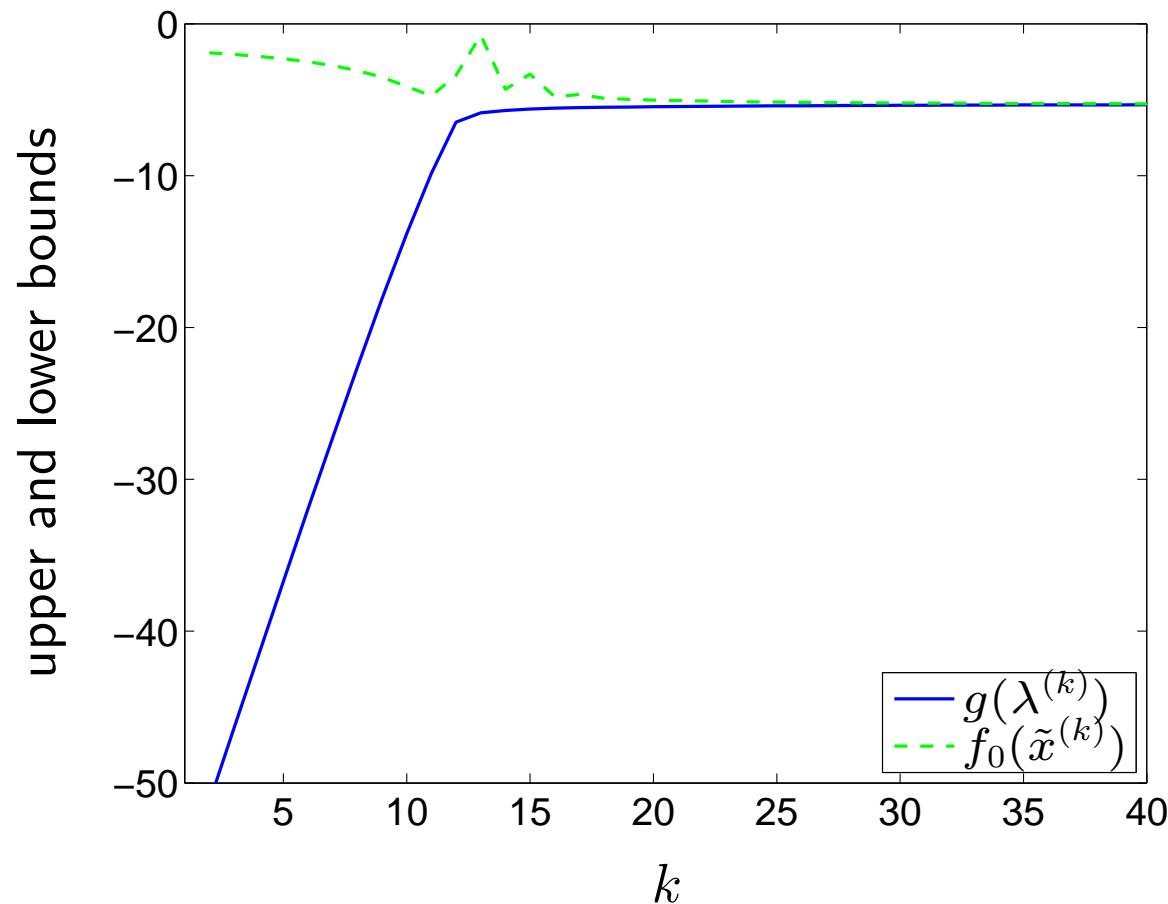
minimize strictly convex quadratic ( $P \succ 0$ ) over unit box:

$$\begin{aligned} & \text{minimize} && (1/2)x^T Px - q^T x \\ & \text{subject to} && x_i^2 \leq 1, \quad i = 1, \dots, n \end{aligned}$$

- $L(x, \lambda) = (1/2)x^T(P + \mathbf{diag}(2\lambda))x - q^T x - \mathbf{1}^T \lambda$
- $x^*(\lambda) = (P + \mathbf{diag}(2\lambda))^{-1}q$
- projected subgradient for dual:

$$x^{(k)} = (P + \mathbf{diag}(2\lambda^{(k)}))^{-1}q, \quad \lambda_i^{(k+1)} = \left( \lambda_i^{(k)} + \alpha_k((x_i^{(k)})^2 - 1) \right)_+$$

problem instance with  $n = 50$ , fixed step size  $\alpha = 0.1$ ,  $f^* \approx -5.3$ ;  
 $\tilde{x}^{(k)}$  is a nearby feasible point for  $x^{(k)}$



## Subgradient method for constrained optimization

solves constrained optimization problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m, \end{aligned}$$

where  $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$  are convex

same update  $x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$ , but we have

$$g^{(k)} \in \begin{cases} \partial f_0(x) & f_i(x) \leq 0, \quad i = 1, \dots, m, \\ \partial f_j(x) & f_j(x) > 0 \end{cases}$$

define  $f_{\text{best}}^{(k)} = \min\{f_0(x^{(i)}) \mid x^{(i)} \text{ feasible}, i = 1, \dots, k\}$

# Convergence

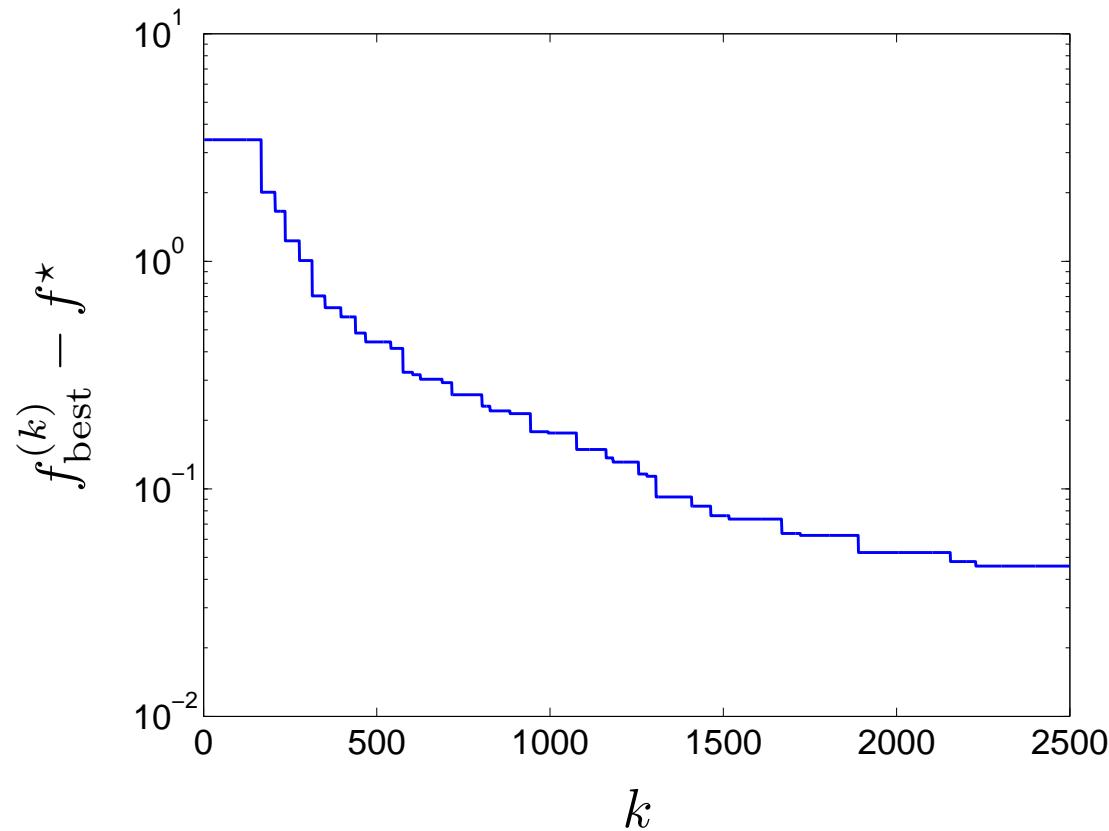
assumptions:

- there exists an optimal  $x^*$ ; Slater's condition holds
- $\|g^{(k)}\|_2 \leq G$ ;  $\|x^{(1)} - x^*\|_2 \leq R$

**typical result:** for  $\alpha_k > 0$ ,  $\alpha_k \rightarrow 0$ ,  $\sum_{i=1}^{\infty} \alpha_i = \infty$ , we have  $f_{\text{best}}^{(k)} \rightarrow f^*$

## Example: Inequality form LP

LP with  $n = 20$  variables,  $m = 200$  inequalities,  $f^* \approx -3.4$ ;  
 $\alpha_k = 1/k$  for optimality step, Polyak's step size for feasibility step



# **Primal-Dual Subgradient Method**

- equality constrained problems
- inequality constrained problems

## Primal-dual subgradient method

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && Ax = b \end{aligned}$$

with variable  $x \in \mathbf{R}^n$ ,  $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$  convex

- primal-dual subgradient method updates both primal and dual variables
- these converge to primal-dual optimal values

## Equality constrained problem

- convex equality constrained problem

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && Ax = b \end{aligned}$$

with variable  $x$  and optimal value  $p^*$

- we will work instead with (equivalent) *augmented problem*

$$\begin{aligned} & \text{minimize} && f(x) + (\rho/2)\|Ax - b\|_2^2 \\ & \text{subject to} && Ax = b \end{aligned}$$

where  $\rho > 0$

# Augmented Lagrangian and optimality conditions

- **augmented Lagrangian** is

$$L(x, \nu) = f(x) + \nu^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2$$

- $(x, \nu)$  primal-dual optimal if and only if

$$0 \in \partial_x L(x, \nu) = \partial f(x) + A^T \nu + \rho A^T (Ax - b)$$

$$0 = -\nabla_\nu L(x, \nu) = b - Ax$$

- same as  $0 \in T(x, \nu)$ , with  $z = (x, \nu)$  and  $T(x, \nu) = \begin{bmatrix} \partial_x L(x, \nu) \\ -\nabla_\nu L(x, \nu) \end{bmatrix}$
- $T$  is a **monotone operator** (much more on this later)

## Primal-dual subgradient method

- primal-dual subgradient method is

$$z^{(k+1)} = z^{(k)} - \alpha_k T^{(k)}$$

where  $T^{(k)} \in T(z^{(k)})$  and  $\alpha_k$  is step length

- more explicitly:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha_k(g^{(k)} + A^T \nu^{(k)} + \rho A^T(Ax^{(k)} - b)) \\ \nu^{(k+1)} &= \nu^{(k)} + \alpha_k(Ax^{(k)} - b) \end{aligned}$$

where  $g^{(k)} \in \partial f(x^{(k)})$

# Convergence

with step size  $\alpha_k = \gamma_k / \|T^{(k)}\|_2$ ,

$$\gamma_k > 0, \quad \sum_k \gamma_k = \infty, \quad \sum_k \gamma_k^2 < \infty$$

we get convergence:

$$f(x^{(k)}) \rightarrow p^*, \quad Ax^{(k)} - b \rightarrow 0$$

## Inequality constrained problem

- convex inequality constrained problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

with variable  $x$ , optimal value  $p^*$

- (equivalent) augmented problem

$$\begin{aligned} & \text{minimize} && f_0(x) + (\rho/2)\|F(x)\|_2^2 \\ & \text{subject to} && F(x) \preceq 0 \end{aligned}$$

where  $F(x) = (f_1(x)_+, \dots, f_m(x)_+)$ ,  $\rho > 0$

## Augmented Lagrangian and optimality conditions

- augmented Lagrangian is

$$L(x, \lambda) = f_0(x) + \lambda^T F(x) + (\rho/2) \|F(x)\|_2^2$$

- $(x, \lambda)$  primal-dual optimal if and only if

$$\begin{aligned} 0 &\in \partial_x L(x, \lambda) = \partial f_0(x) + \sum_{i=1}^m (\lambda_i + \rho f_i(x)_+) \partial f_i(x)_+ \\ 0 &= -\nabla_\lambda L(x, \lambda) = -F(x) \end{aligned}$$

## Primal-dual subgradient method

- define  $z = (x, \nu)$  and

$$T(x, \lambda) = \begin{bmatrix} \partial_x L(x, \lambda) \\ -\nabla_\lambda L(x, \lambda) \end{bmatrix}$$

( $T$  is the KKT operator for the problem, and is monotone)

- primal-dual subgradient method is

$$z^{(k+1)} = z^{(k)} - \alpha_k T^{(k)}$$

where  $T^{(k)} \in T(z^{(k)})$  and  $\alpha_k$  is step length

- more explicitly:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha_k \left( g_0^{(k)} + \sum_{i=1}^m (\lambda_i^{(k)} + \rho f_i(x^{(k)})_+) g_i^{(k)} \right) \\ \lambda_i^{(k+1)} &= \lambda_i^{(k)} + \alpha_k f_i(x^{(k)})_+, \quad i = 1, \dots, m \end{aligned}$$

where  $g_0^{(k)} \in \partial f_0(x^{(k)})$ ,  $g_i^{(k)} \in \partial f_i(x^{(k)})_+$ ,  $i = 1, \dots, m$

- note that  $\lambda_i^{(k)}$  can only increase with  $k$

# Convergence

with step size  $\alpha_k = \gamma_k / \|T^{(k)}\|_2$ ,

$$\gamma_k > 0, \quad \sum_k \gamma_k = \infty, \quad \sum_k \gamma_k^2 < \infty$$

we get convergence:

$$f_0(x^{(k)}) \rightarrow p^*, \quad f_i(x^{(k)})_+ \rightarrow 0, \quad i = 1, \dots, m$$

## Example: Inequality constrained LP

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

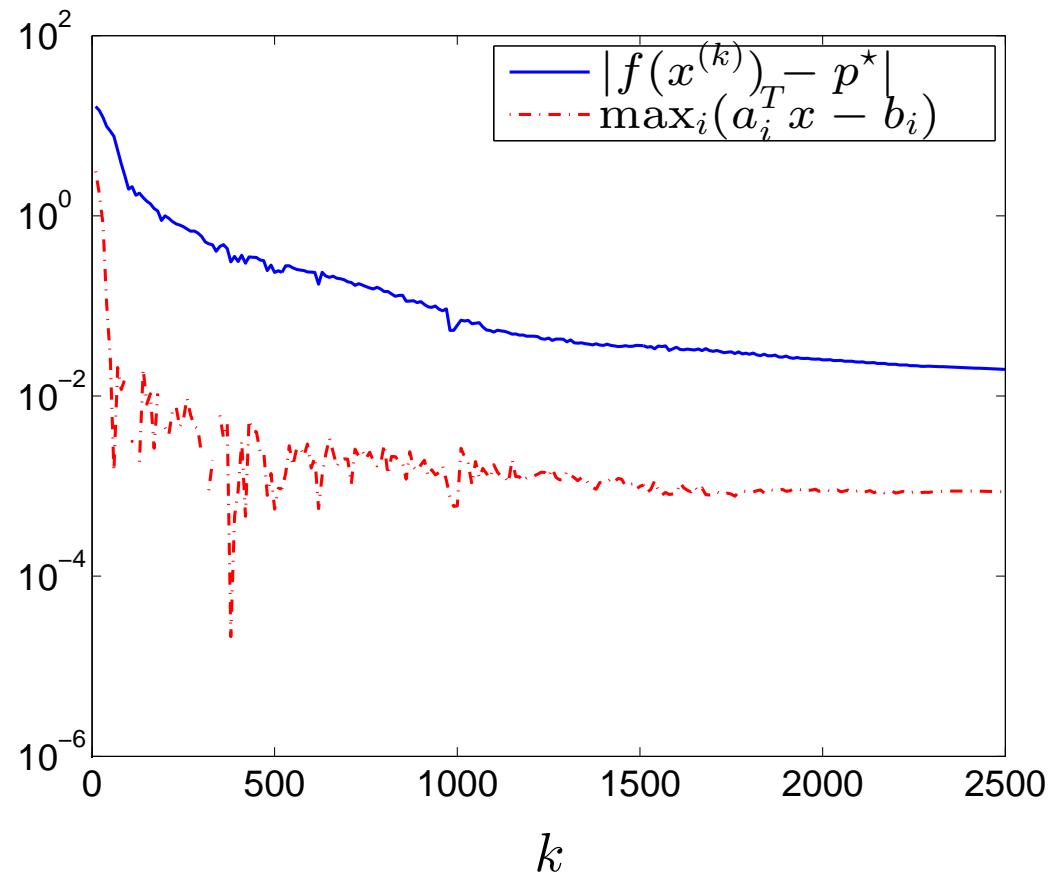
primal-dual subgradient update is

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha_k \left( c + A^T M^{(k)} (\lambda^{(k)} + \rho(Ax^{(k)} - b)_+) \right) \\ \lambda^{(k+1)} &= \lambda^{(k)} + \alpha_k (Ax^{(k)} - b)_+ \end{aligned}$$

where  $M^{(k)}$  is a diagonal matrix

$$M_{ii}^{(k)} = \begin{cases} 1 & a_i^T x^{(k)} > b_i \\ 0 & a_i^T x^{(k)} \leq b_i \end{cases}$$

problem instance with  $n = 20$ ,  $m = 200$ ,  $p^* \approx -3.4$   
step size  $\alpha_k = 1/(k\|T^{(k)}\|_2)$



# Stochastic Subgradient Method

- noisy unbiased subgradient
- stochastic subgradient method
- convergence proof
- stochastic programming
- expected value of convex function
- on-line learning and adaptive signal processing

## Noisy unbiased subgradient

- random vector  $\tilde{g} \in \mathbf{R}^n$  is a **noisy unbiased subgradient** for  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  at  $x$  if for all  $z$

$$f(z) \geq f(x) + (\mathbf{E} \tilde{g})^T (z - x)$$

i.e.,  $g = \mathbf{E} \tilde{g} \in \partial f(x)$

- same as  $\tilde{g} = g + v$ , where  $g \in \partial f(x)$ ,  $\mathbf{E} v = 0$
- $v$  can represent error in computing  $g$ , measurement noise, Monte Carlo sampling error, etc.

- if  $x$  is also random,  $\tilde{g}$  is a noisy unbiased subgradient of  $f$  at  $x$  if

$$\forall z \quad f(z) \geq f(x) + \mathbf{E}(\tilde{g}|x)^T(z - x)$$

holds almost surely

- same as  $\mathbf{E}(\tilde{g}|x) \in \partial f(x)$  (a.s.)

## Stochastic subgradient method

**stochastic subgradient method** is the subgradient method, using noisy unbiased subgradients

$$x^{(k+1)} = x^{(k)} - \alpha_k \tilde{g}^{(k)}$$

- $x^{(k)}$  is  $k$ th iterate
- $\tilde{g}^{(k)}$  is any noisy unbiased subgradient of (convex)  $f$  at  $x^{(k)}$ , i.e.,

$$\mathbf{E}(\tilde{g}^{(k)}|x^{(k)}) = g^{(k)} \in \partial f(x^{(k)})$$

- $\alpha_k > 0$  is the  $k$ th step size
- define  $f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \dots, f(x^{(k)})\}$

## Assumptions

- $f^* = \inf_x f(x) > -\infty$ , with  $f(x^*) = f^*$
- $\mathbf{E} \|g^{(k)}\|_2^2 \leq G^2$  for all  $k$
- $\mathbf{E} \|x^{(1)} - x^*\|_2^2 \leq R^2$  (can take  $=$  here)
- step sizes are square-summable but not summable

$$\alpha_k \geq 0, \quad \sum_{k=1}^{\infty} \alpha_k^2 = \|\alpha\|_2^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty$$

these assumptions are stronger than needed, just to simplify proofs

## Convergence results

- convergence in expectation:

$$\lim_{k \rightarrow \infty} \mathbf{E} f_{\text{best}}^{(k)} = f^*$$

- convergence in probability: for any  $\epsilon > 0$ ,

$$\lim_{k \rightarrow \infty} \mathbf{Prob}(f_{\text{best}}^{(k)} \geq f^* + \epsilon) = 0$$

- almost sure convergence:

$$\lim_{k \rightarrow \infty} f_{\text{best}}^{(k)} = f^*$$

a.s. (we won't show this)

## Convergence proof

**key quantity:** expected Euclidean distance squared to the optimal set

$$\begin{aligned} \mathbf{E} (\|x^{(k+1)} - x^*\|_2^2 \mid x^{(k)}) &= \mathbf{E} (\|x^{(k)} - \alpha_k \tilde{g}^{(k)} - x^*\|_2^2 \mid x^{(k)}) \\ &= \|x^{(k)} - x^*\|_2^2 - 2\alpha_k \mathbf{E} (\tilde{g}^{(k)T} (x^{(k)} - x^*) \mid x^{(k)}) + \alpha_k^2 \mathbf{E} (\|\tilde{g}^{(k)}\|_2^2 \mid x^{(k)}) \\ &= \|x^{(k)} - x^*\|_2^2 - 2\alpha_k \mathbf{E} (\tilde{g}^{(k)} | x^{(k)})^T (x^{(k)} - x^*) + \alpha_k^2 \mathbf{E} (\|\tilde{g}^{(k)}\|_2^2 \mid x^{(k)}) \\ &\leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^*) + \alpha_k^2 \mathbf{E} (\|\tilde{g}^{(k)}\|_2^2 \mid x^{(k)}) \end{aligned}$$

using  $\mathbf{E}(\tilde{g}^{(k)} | x^{(k)}) \in \partial f(x^{(k)})$

now take expectation:

$$\mathbf{E} \|x^{(k+1)} - x^*\|_2^2 \leq \mathbf{E} \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (\mathbf{E} f(x^{(k)}) - f^*) + \alpha_k^2 \mathbf{E} \|\tilde{g}^{(k)}\|_2^2$$

apply recursively, and use  $\mathbf{E} \|\tilde{g}^{(k)}\|_2^2 \leq G^2$  to get

$$\mathbf{E} \|x^{(k+1)} - x^*\|_2^2 \leq \mathbf{E} \|x^{(1)} - x^*\|_2^2 - 2 \sum_{i=1}^k \alpha_i (\mathbf{E} f(x^{(i)}) - f^*) + G^2 \sum_{i=1}^k \alpha_i^2$$

and so

$$\min_{i=1,\dots,k} (\mathbf{E} f(x^{(i)}) - f^*) \leq \frac{R^2 + G^2 \|\alpha\|_2^2}{2 \sum_{i=1}^k \alpha_i}$$

- we conclude  $\min_{i=1,\dots,k} \mathbf{E} f(x^{(i)}) \rightarrow f^*$
- Jensen's inequality and concavity of minimum yields

$$\mathbf{E} f_{\text{best}}^{(k)} = \mathbf{E} \min_{i=1,\dots,k} f(x^{(i)}) \leq \min_{i=1,\dots,k} \mathbf{E} f(x^{(i)})$$

so  $\mathbf{E} f_{\text{best}}^{(k)} \rightarrow f^*$  (convergence in expectation)

- Markov's inequality: for  $\epsilon > 0$

$$\mathbf{Prob}(f_{\text{best}}^{(k)} - f^* \geq \epsilon) \leq \frac{\mathbf{E}(f_{\text{best}}^{(k)} - f^*)}{\epsilon}$$

righthand side goes to zero, so we get convergence in probability

# Example

piecewise linear minimization

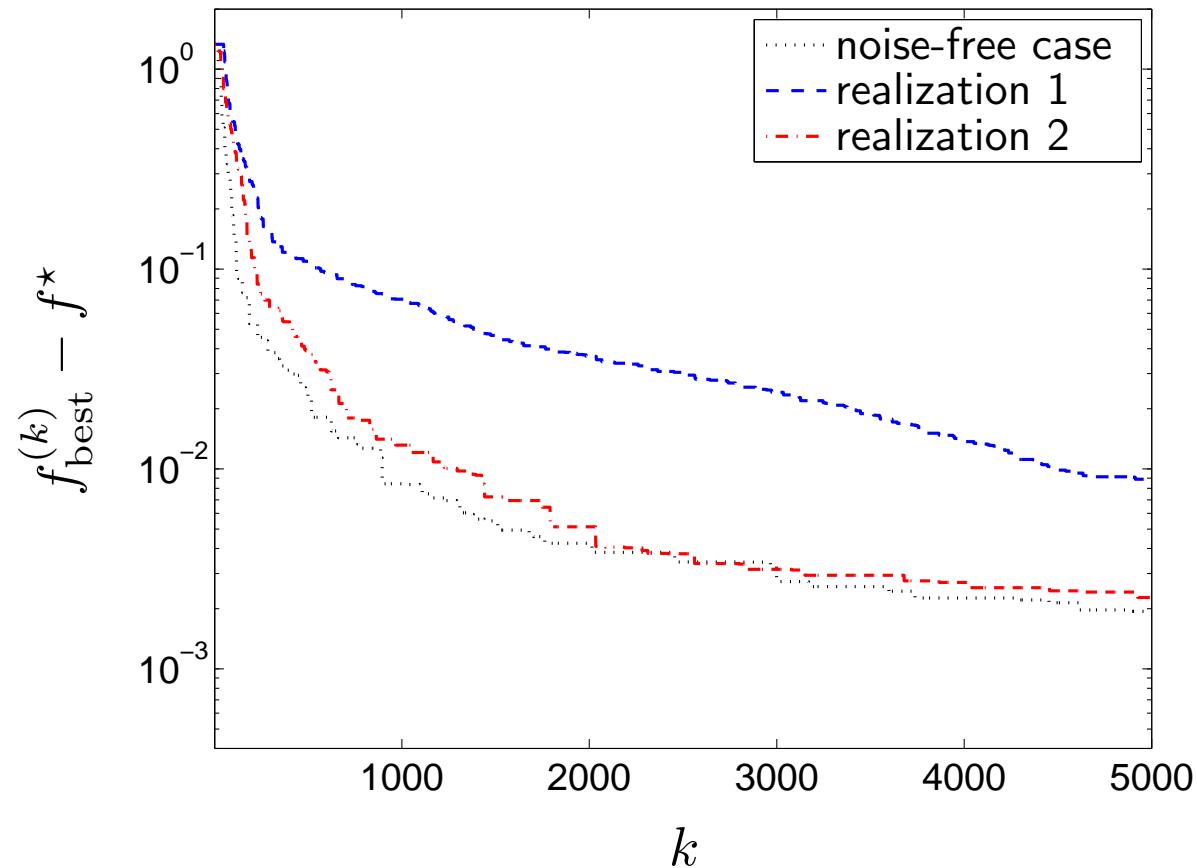
$$\text{minimize } f(x) = \max_{i=1,\dots,m} (a_i^T x + b_i)$$

we use stochastic subgradient algorithm with noisy subgradient

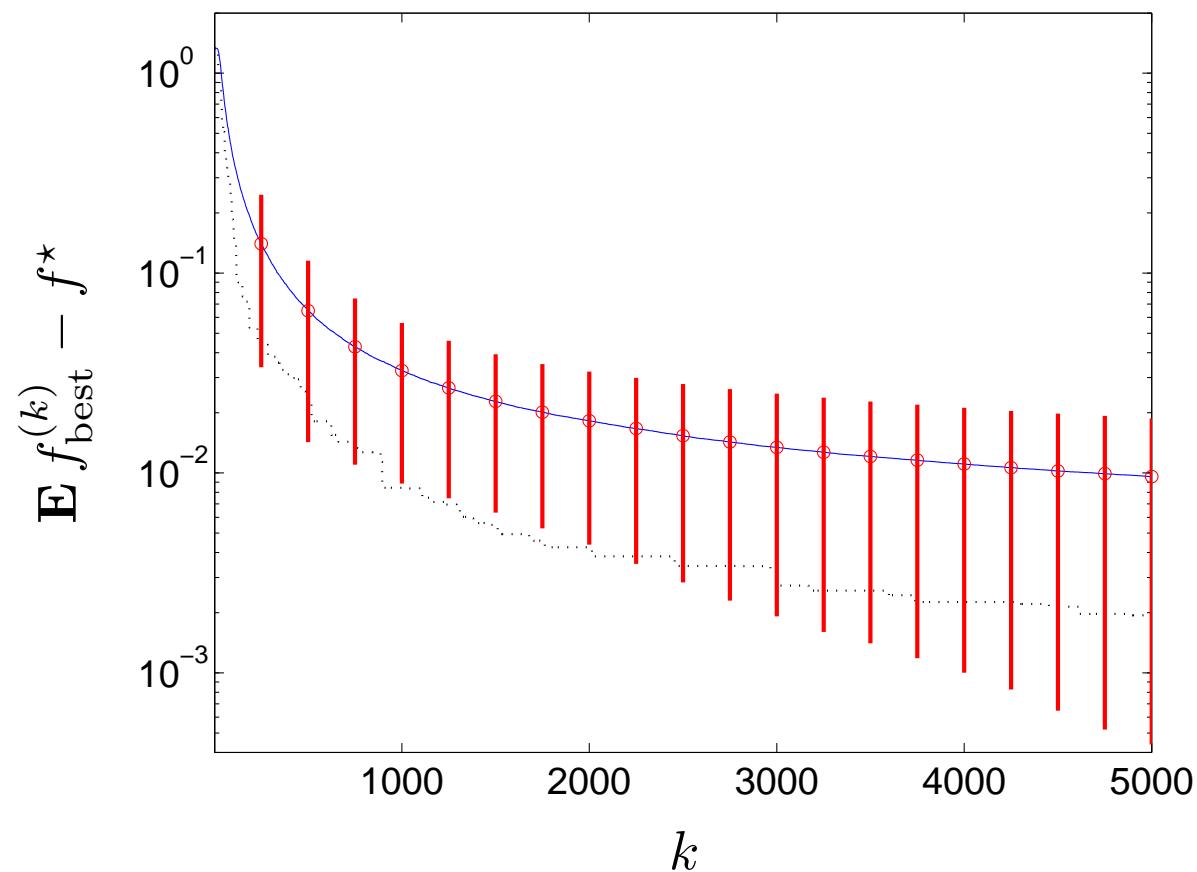
$$\tilde{g}^{(k)} = g^{(k)} + v^{(k)}, \quad g^{(k)} \in \partial f(x^{(k)})$$

$v^{(k)}$  independent zero mean random variables

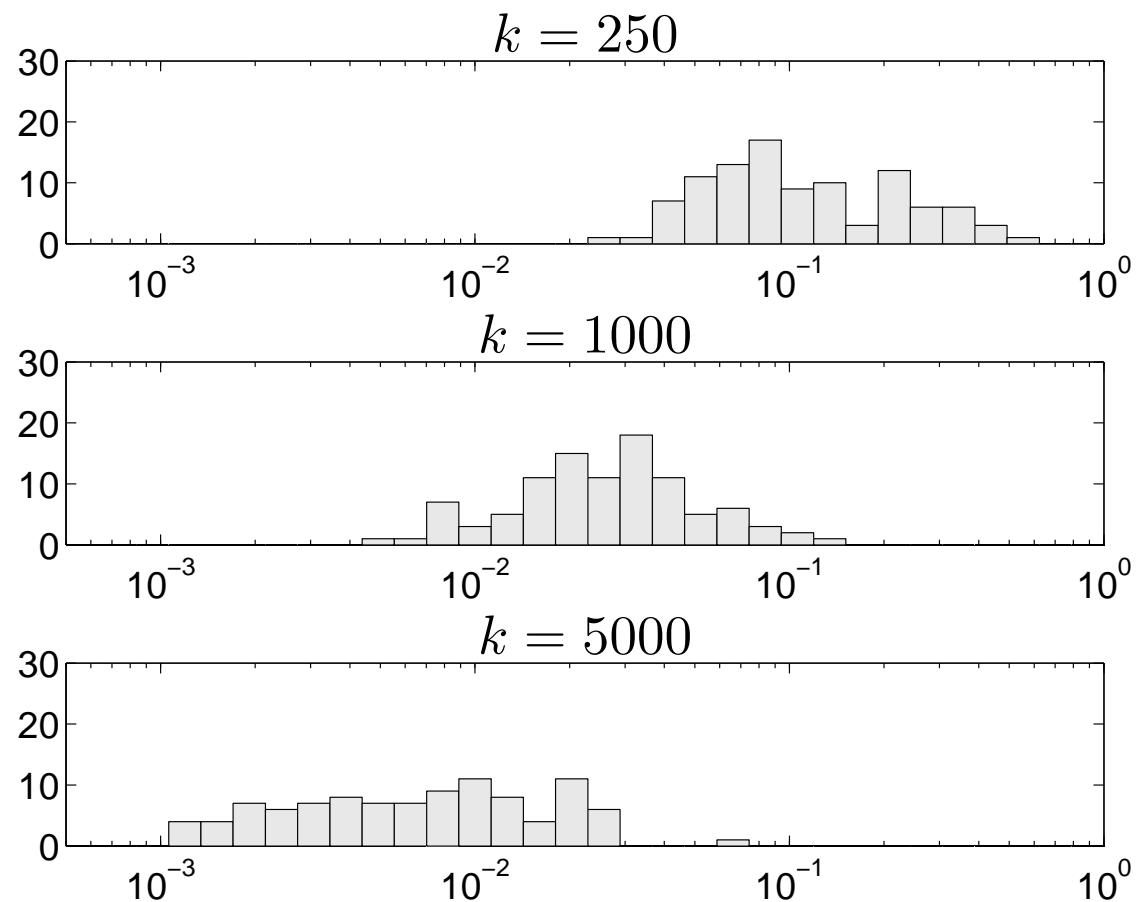
problem instance:  $n = 20$  variables,  $m = 100$  terms,  $f^* \approx 1.1$ ,  $\alpha_k = 1/k$   
 $v^{(k)}$  are IID  $\mathcal{N}(0, 0.5I)$  (25% noise since  $\|g\| \approx 4.5$ )



average and one std. dev. for  $f_{\text{best}}^{(k)} - f^*$  over 100 realizations



empirical distributions of  $f_{\text{best}}^{(k)} - f^*$  at  $k = 250$ ,  $k = 1000$ , and  $k = 5000$



## Stochastic programming

$$\begin{aligned} & \text{minimize} && \mathbf{E} f_0(x, \omega) \\ & \text{subject to} && \mathbf{E} f_i(x, \omega) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

if  $f_i(x, \omega)$  is convex in  $x$  for each  $\omega$ , problem is convex

'certainty-equivalent' problem

$$\begin{aligned} & \text{minimize} && f_0(x, \mathbf{E} \omega) \\ & \text{subject to} && f_i(x, \mathbf{E} \omega) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

(if  $f_i(x, \omega)$  is convex in  $\omega$ , gives a lower bound on optimal value of stochastic problem)

## Variations

- in place of  $\mathbf{E} f_i(x, \omega) \leq 0$  (constraint holds in expectation) can use
  - $\mathbf{E} f_i(x, \omega)_+ \leq \epsilon$  (LHS is expected violation)
  - $\mathbf{E} (\max_i f_i(x, \omega)_+) \leq \epsilon$  (LHS is expected worst violation)
- unfortunately, *chance constraint*  $\mathbf{Prob}(f_i(x, \omega) \leq 0) \geq \eta$  is convex only in a few special cases

## Expected value of convex function

suppose  $F(x, w)$  is convex in  $x$  for each  $w$  and  $G(x, w) \in \partial_x F(x, w)$

- $f(x) = \mathbf{E} F(x, w) = \int F(x, w)p(w) dw$  is convex
- a subgradient of  $f$  at  $x$  is

$$g = \mathbf{E} G(x, w) = \int G(x, w)p(w) dw \in \partial f(x)$$

- a noisy unbiased subgradient of  $f$  at  $x$  is

$$\tilde{g} = \frac{1}{M} \sum_{i=1}^M G(x, w_i)$$

where  $w_1, \dots, w_M$  are  $M$  independent samples (Monte Carlo)

## Example: Expected value of piecewise linear function

$$\text{minimize } f(x) = \mathbf{E} \max_{i=1,\dots,m} (a_i^T x + b_i)$$

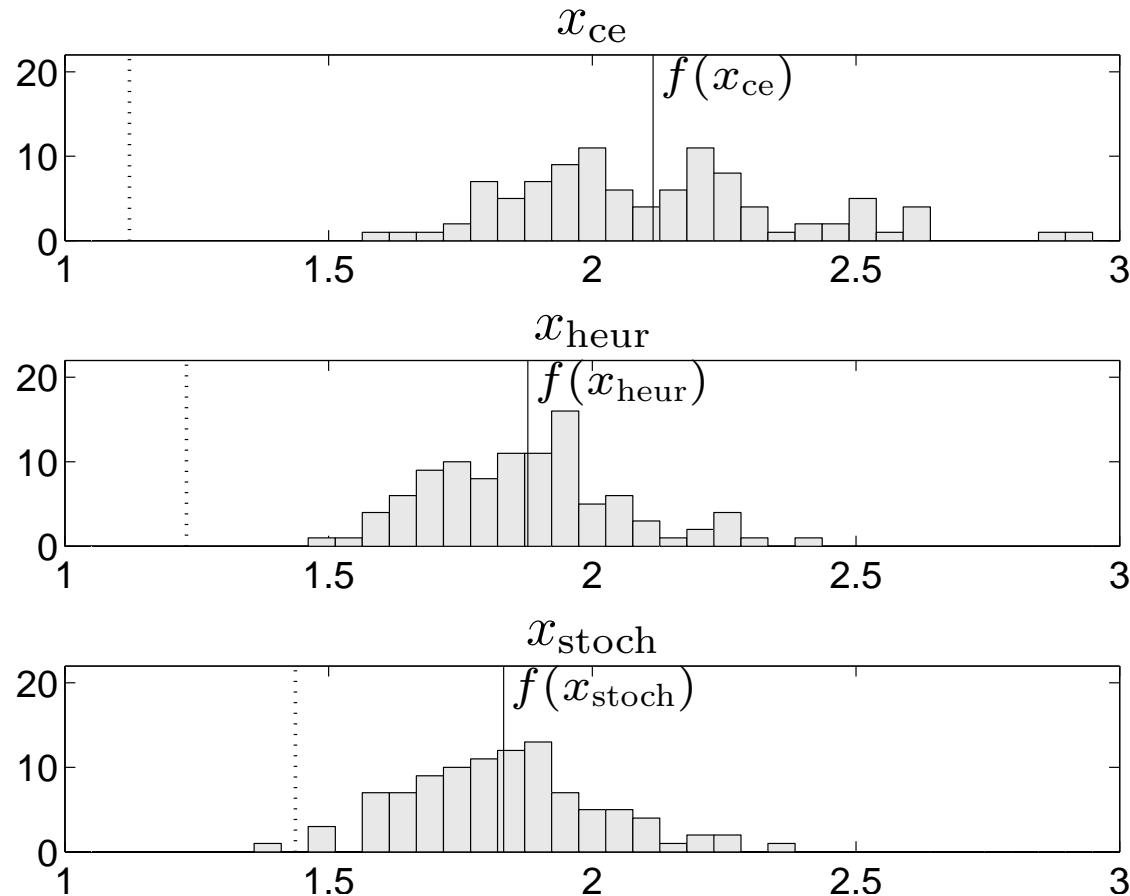
where  $a_i$  and  $b_i$  are random

evaluate noisy subgradient using Monte Carlo method with  $M$  samples,  
and run stochastic subgradient method

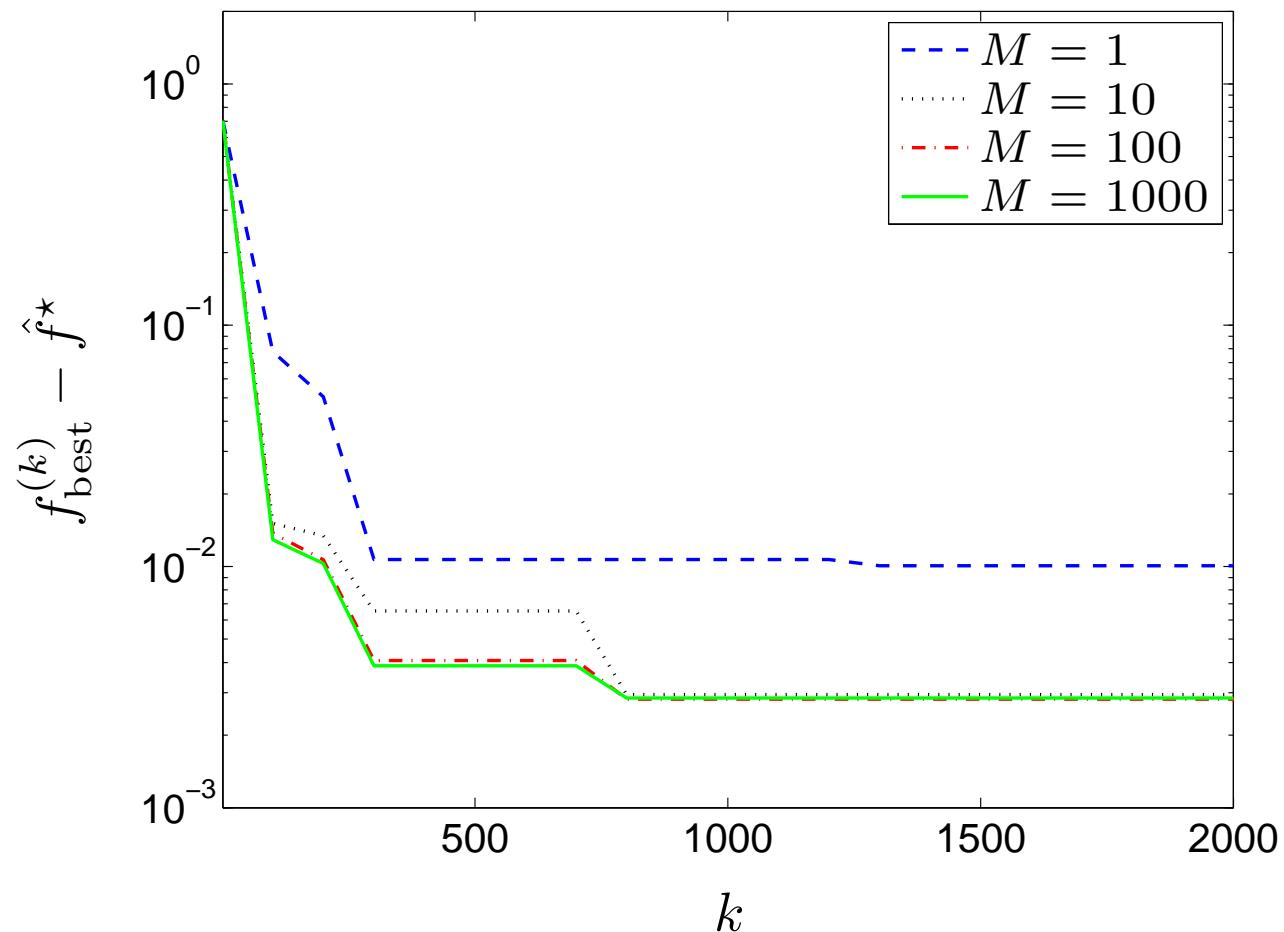
compare to:

- certainty equivalent: minimize  $f_{\text{ce}}(x) = \max_{i=1,\dots,m} (\mathbf{E} a_i^T x + \mathbf{E} b_i)$
- heuristic: minimize  $f_{\text{heur}}(x) = \max_{i=1,\dots,m} (\mathbf{E} a_i^T x + \mathbf{E} b_i + \lambda \|x\|_2)$

problem instance:  $n = 20$ ,  $m = 100$ ,  $a_i \sim \mathcal{N}(\bar{a}_i, 5I)$ ,  $b \sim \mathcal{N}(\bar{b}, 5I)$ ,  
 $\|a_i\|_2 \approx 5$ ,  $\|b\|_2 \approx 10$ ,  $x_{\text{stoch}}$  computed using  $M = 100$



$f^* \approx 1.34$  estimated by running the method with  $M = 1000$  for long time



# On-line learning and adaptive signal processing

- $(x, y) \in \mathbf{R}^n \times \mathbf{R}$  have some joint distribution
- find weight vector  $w \in \mathbf{R}^n$  for which  $w^T x$  is a good estimator of  $y$
- choose  $w$  to minimize expected value of a convex *loss function*  $l$

$$J(w) = \mathbf{E} l(w^T x - y)$$

- $l(u) = u^2$ : mean-square error
- $l(u) = |u|$ : mean-absolute error
- at each step (*e.g.*, time sample), we are given a sample  $(x^{(k)}, y^{(k)})$  from the distribution

noisy unbiased subgradient of  $J$  at  $w^{(k)}$ , based on sample  $x^{(k+1)}, y^{(k+1)}$ :

$$g^{(k)} = l'(w^{(k)T}x^{(k+1)} - y^{(k+1)})x^{(k+1)}$$

where  $l'$  is the derivative (or a subgradient) of  $l$

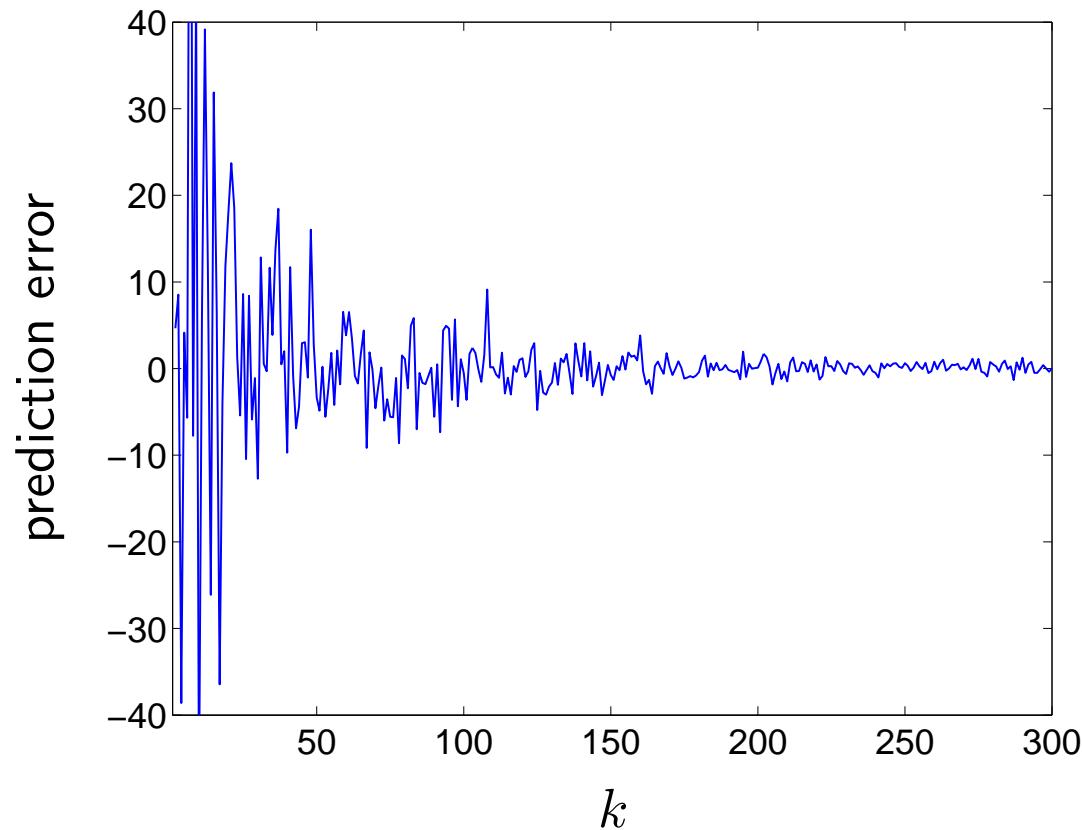
on-line algorithm:

$$w^{(k+1)} = w^{(k)} - \alpha_k l'(w^{(k)T}x^{(k+1)} - y^{(k+1)})x^{(k+1)}.$$

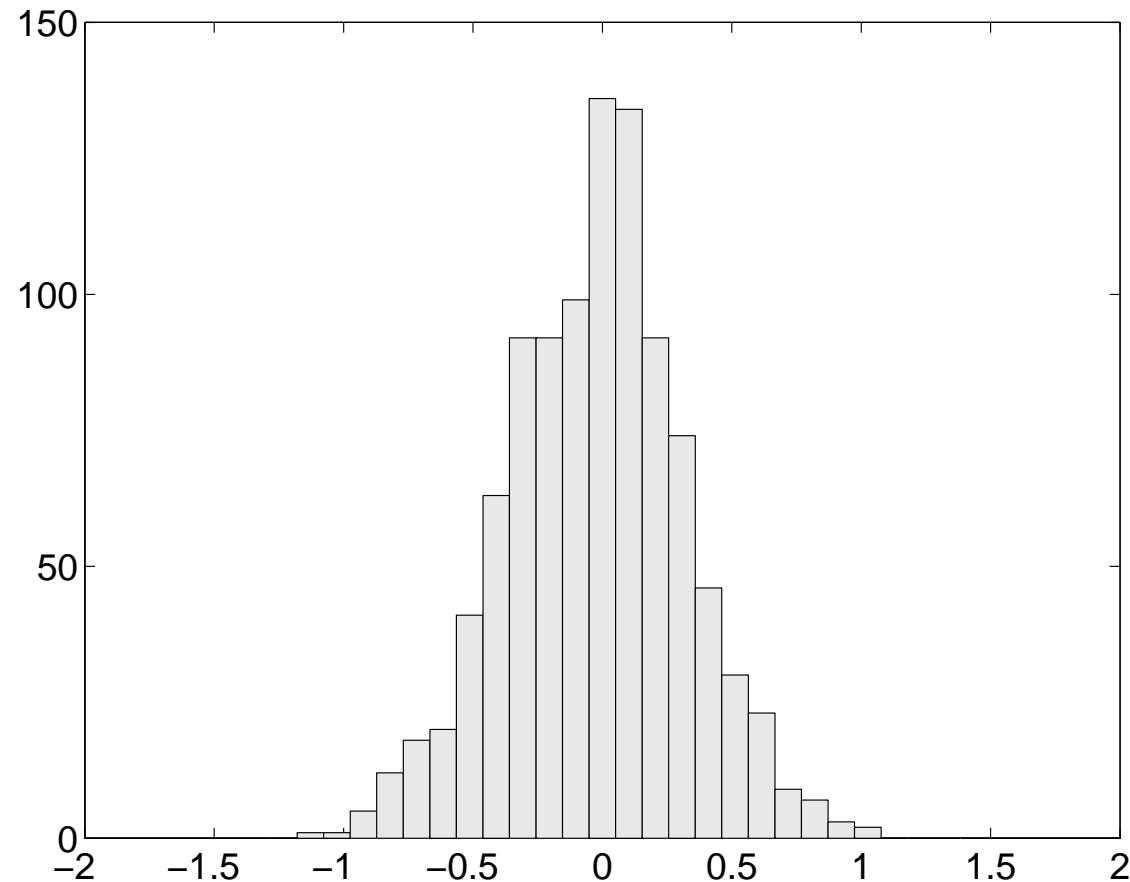
- for  $l(u) = u^2$ , gives the LMS (least mean-square) algorithm
- for  $l(u) = |u|$ , gives the *sign* algorithm
- $w^{(k)T}x^{(k+1)} - y^{(k+1)}$  is the prediction error

## Example: Mean-absolute error minimization

problem instance:  $n = 10$ ,  $(x, y) \sim \mathcal{N}(0, \Sigma)$ ,  $\Sigma$  random with  $\mathbf{E}(y^2) \approx 12$ ,  $\alpha_k = 1/k$



empirical distribution of prediction error for  $w^*$  (over 1000 samples)



# **The choice of metric in subgradient methods**

Stephen Boyd & John Duchi (with help from P. Giselsson)

EE364b, Stanford University

## Mirror descent methods

subgradient method without using Euclidean steps

- let  $h$  be a differentiable convex function, then associated Bregman divergence is

$$D_h(y, x) = h(y) - h(x) - \nabla h(x)^T(y - x)$$

- mirror (or non-linear) subgradient method

- (1) get subgradient  $g^{(k)} \in \partial f(x^{(k)})$
- (2) update

$$x^{(k+1)} = \operatorname{argmin}_{x \in C} \left\{ g^{(k)T} x + \frac{1}{\alpha_k} D_h(x, x^{(k)}) \right\}$$

generalizes projected subgradient decent (take  $h(x) = \frac{1}{2} \|x\|_2^2$ )

## Convergence analysis

properties of  $h$  required: *strong convexity* with respect to norm  $\|\cdot\|$

$$h(y) \geq h(x) + \nabla h(x)^T (y - x) + \frac{1}{2} \|x - y\|^2$$

For any  $x^* \in C$ ,

$$\begin{aligned} f(x^{(k)}) - f(x^*) &\leq g^{(k)T}(x^{(k)} - x^*) \\ &= g^{(k)T}(x^{(k+1)} - x^*) + g^{(k)T}(x^{(k)} - x^{(k+1)}) \end{aligned}$$

Use optimality conditions for  $x^{(k+1)}$ :

$$(\alpha_k g^{(k)} + \nabla h(x^{(k+1)}) - \nabla h(x^{(k)}))^T (y - x^{(k+1)}) \geq 0, \quad \text{all } y \in C$$

so (take  $y = x^*$ )

$$g^{(k)T}(x^{(k+1)} - x^*) \leq \frac{1}{\alpha_k} (\nabla h(x^{(k+1)}) - \nabla h(x^{(k)}))^T (x^* - x^{(k+1)})$$

## Convergence analysis continued

identity for divergences

$$\begin{aligned} & (\nabla h(x^{(k+1)}) - \nabla h(x^{(k)}))^T (x^* - x^{(k+1)}) \\ &= D_h(x^*, x^{(k)}) - D_h(x^*, x^{(k+1)}) - D_h(x^{(k)}, x^{(k+1)}) \end{aligned}$$

for any  $x^* \in C$ ,

$$\begin{aligned} f(x^{(k)}) - f(x^*) &\leq g^{(k)T}(x^{(k+1)} - x^*) + g^{(k)T}(x^{(k)} - x^{(k+1)}) \\ &\leq \frac{1}{\alpha_k} [D_h(x^*, x^{(k)}) - D_h(x^*, x^{(k+1)})] - \frac{1}{\alpha_k} D_h(x^{(k)}, x^{(k+1)}) \\ &\quad + g^{(k)T}(x^{(k)} - x^{(k+1)}) \end{aligned}$$

apply Fenchel-Young inequality ( $x^T y \leq \frac{1}{2\alpha} \|x\|^2 + \frac{\alpha}{2} \|y\|_*^2$ )

$$\begin{aligned} &\leq \frac{1}{\alpha_k} [D_h(x^*, x^{(k)}) - D_h(x^*, x^{(k+1)})] - \frac{1}{\alpha_k} D_h(x^{(k)}, x^{(k+1)}) \\ &\quad + \frac{\alpha_k \|g^{(k)}\|_*^2}{2} + \frac{1}{2\alpha_k} \|x^{(k)} - x^{(k+1)}\|^2 \\ &\leq \frac{1}{\alpha_k} [D_h(x^*, x^{(k)}) - D_h(x^*, x^{(k+1)})] + \frac{\alpha_k}{2} \|g^{(k)}\|_*^2 \end{aligned}$$

## Convergence guarantees

with fixed stepsize  $\alpha_k = \alpha$ ,

$$\frac{1}{k} \sum_{i=1}^k f(x^{(i)}) - f(x^*) \leq \frac{1}{\alpha k} D_h(x^*, x^{(1)}) + \frac{\alpha}{2k} \max_i \|g^{(i)}\|_*^2$$

in general, converges if

- $D_h(x^*, x^{(1)}) < \infty$
- $\sum_k \alpha_k = \infty$  and  $\alpha_k \rightarrow 0$
- for all  $g \in \partial f(x)$  and  $x \in C$ ,  $\|g\|_* \leq G$  for some  $G < \infty$

## Mirror descent examples

- Usual (projected) subgradient descent:  $h(x) = \frac{1}{2} \|x\|_2^2$
- With constraints of simplex,  $C = \{x \in \mathbf{R}_+^n \mid \mathbf{1}^T x = 1\}$ , use negative entropy

$$h(x) = \sum_{i=1}^n x_i \log x_i$$

- (1) Strongly convex with respect to  $\ell_1$ -norm
- (2) With  $x^{(1)} = \mathbf{1}/n$ , have  $D_h(x^*, x^{(1)}) \leq \log n$  for  $x^* \in C$
- (3) If  $G_\infty \geq \|g\|_\infty$  for  $g \in \partial f(x)$  for  $x \in C$ ,

$$f_{\text{best}}^{(k)} - f^* \leq \frac{\log n}{\alpha k} + \frac{\alpha}{2k} G_\infty$$

- (4) Can be much better than regular subgradient decent...

## Example

Robust regression problem (an LP):

$$\begin{aligned} \text{minimize } f(x) &= \|Ax - b\|_1 = \sum_{i=1}^m |a_i^T x - b_i| \\ \text{subject to } x &\in C = \{x \in \mathbf{R}_+^n \mid \mathbf{1}^T x = 1\} \end{aligned}$$

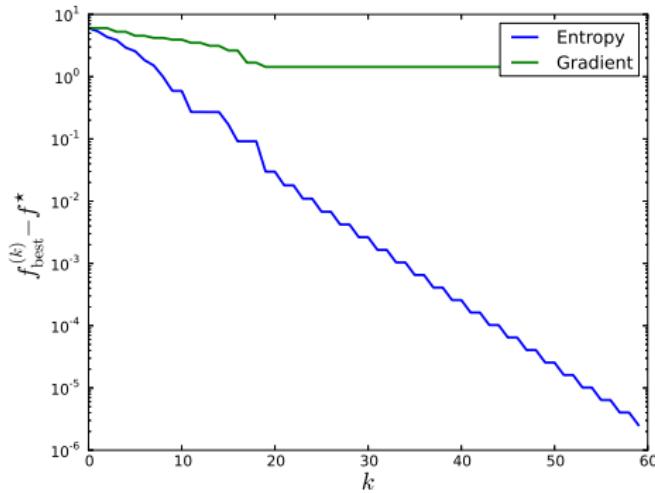
subgradient of objective is  $g = \sum_{i=1}^m \text{sign}(a_i^T x - b_i) a_i$

- Projected subgradient update ( $h(x) = (1/2) \|x\|_2^2$ ): homework
- Mirror descent update ( $h(x) = \sum_{i=1}^n x_i \log x_i$ ):

$$x_i^{(k+1)} = \frac{x_i^{(k)} \exp(-\alpha g_i^{(k)})}{\sum_{j=1}^n x_j^{(k)} \exp(-\alpha g_j^{(k)})}$$

## Example

Robust regression problem with  $a_i \sim N(0, I_{n \times n})$  and  
 $b_i = (a_{i,1} + a_{i,2})/2 + \varepsilon_i$  where  $\varepsilon_i \sim N(0, 10^{-2})$ ,  $m = 20$ ,  $n = 3000$



stepsizes chosen according to best bounds (but still sensitive to stepsize choice)

## Variable metric subgradient methods

subgradient method with variable metric  $H_k \succ 0$ :

- (1) get subgradient  $g^{(k)} \in \partial f(x^{(k)})$
- (2) update (diagonal) metric  $H_k$
- (3) update  $x^{(k+1)} = x^{(k)} - H_k^{-1} g^{(k)}$

- matrix  $H_k$  generalizes step-length  $\alpha_k$

there are many such methods (Ellipsoid method, AdaGrad, . . . )

## Variable metric projected subgradient method

same, with projection carried out in the  $H_k$  metric:

- (1) get subgradient  $g^{(k)} \in \partial f(x^{(k)})$
- (2) update (diagonal) metric  $H_k$
- (3) update  $x^{(k+1)} = P_{\mathcal{X}}^{H_k} (x^{(k)} - H_k^{-1} g^{(k)})$

where

$$\Pi_{\mathcal{X}}^H(y) = \operatorname{argmin}_{x \in \mathcal{X}} \|x - y\|_H^2$$

and  $\|x\|_H = \sqrt{x^T H x}$ .

## Convergence analysis

since  $\Pi_{\mathcal{X}}^{H_k}$  is non-expansive in the  $\|\cdot\|_{H_k}$  norm, we get

$$\begin{aligned}\|x^{(k+1)} - x^*\|_{H_k}^2 &= \left\| P_{\mathcal{X}}^{H_k} \left( x^{(k)} - H_k^{-1} g^{(k)} \right) - P_{\mathcal{X}}^{H_k} (x^*) \right\|_{H_k}^2 \\ &\leq \|x^{(k)} - H_k^{-1} g^{(k)} - x^*\|_{H_k}^2 \\ &= \|x^{(k)} - x^*\|_{H_k}^2 - 2(g^{(k)})^T (x^{(k)} - x^*) + \|g^{(k)}\|_{H_k^{-1}}^2 \\ &\leq \|x^{(k)} - x^*\|_{H_k}^2 - 2(f(x^{(k)}) - f^*) + \|g^{(k)}\|_{H_k^{-1}}^2.\end{aligned}$$

using  $f^* = f(x^*) \geq f(x^{(k)}) + g^{(k)T}(x^* - x^{(k)})$

apply recursively, use

$$\sum_{i=1}^k \left( f(x^{(i)}) - f^\star \right) \geq k \left( f_{\text{best}}^{(k)} - f^\star \right)$$

and rearrange to get

$$\begin{aligned} f_{\text{best}}^{(k)} - f^\star &\leq \frac{\|x^{(1)} - x^\star\|_{H_1}^2 + \sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2}{2k} \\ &+ \frac{\sum_{i=2}^k \left( \|x^{(i)} - x^\star\|_{H_i}^2 - \|x^{(i)} - x^\star\|_{H_{i-1}}^2 \right)}{2k} \end{aligned}$$

numerator of additional term can be bounded to get estimates

- for general  $H_k = \text{diag}(h_k)$

$$f_{\text{best}}^k - f^* \leq \frac{R_\infty^2 \|H_1\|_1 + \sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2}{2k} + \frac{R_\infty^2 \sum_{i=2}^k \|H_i - H_{i-1}\|_1}{2k}$$

- for  $H_k = \text{diag}(h_k)$  with  $h_i \geq h_{i-1}$  for all  $i$

$$f_{\text{best}}^k - f^* \leq \frac{\sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2}{2k} + \frac{R_\infty^2 \|h_k\|_1}{2k}$$

where  $\max_{1 \leq i \leq k} \|x^{(i)} - x^*\|_\infty \leq R_\infty$

converges if

- $R_\infty < \infty$  (e.g. if  $\mathcal{X}$  is compact)
- $\sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2$  grows slower than  $k$
- $\sum_{i=2}^k \|H_i - H_{i-1}\|_1$  grows slower than  $k$  **or**  
 $h_i \geq h_{i-1}$  for all  $i$  and  $\|h_k\|_1$  grows slower than  $k$

## AdaGrad

AdaGrad — adaptive subgradient method

- (1) get subgradient  $g^{(k)} \in \partial f(x^{(k)})$
- (2) choose metric  $H_k$ :

- set  $S_k = \sum_{i=1}^k \text{diag}(g^{(i)})^2$
- set  $H_k = \frac{1}{\alpha} S_k^{\frac{1}{2}}$

- (3) update  $x^{(k+1)} = P_{\mathcal{X}}^{H_k} (x^{(k)} - H_k^{-1} g^{(k)})$

where  $\alpha > 0$  is step-size

## AdaGrad – motivation

- for fixed  $H_k = H$  we have estimate:

$$f_{\text{best}}^{(k)} - f^* \leq \frac{1}{2k} (x^{(1)} - x^*)^T H (x^{(1)} - x^*) + \frac{1}{2k} \sum_{i=1}^k \|g^{(i)}\|_{H^{-1}}^2$$

- **idea:** Choose *diagonal*  $H_k \succ 0$  that minimizes this estimate in hindsight:

$$H_k = \operatorname{argmin}_h \max_{x,y \in C} (x - y)^T \operatorname{diag}(h) (x - y) + \sum_{i=1}^k \|g^{(i)}\|_{\operatorname{diag}(h)^{-1}}^2$$

- optimal  $H_k = \frac{1}{R_\infty} \operatorname{diag} \left( \sqrt{\sum_{i=1}^k (g_1^{(i)})^2}, \dots, \sqrt{\sum_{i=1}^k (g_n^{(i)})^2} \right)$
- **intuition:** adapt step-length based on historical step lengths

## AdaGrad – convergence

by construction,  $H_i = \frac{1}{\alpha} \mathbf{diag}(h_i)$  and  $h_i \geq h_{i-1}$ , so

$$\begin{aligned} f_{\text{best}}^{(k)} - f^* &\leq \frac{1}{2k} \sum_{i=1}^k \|g^{(i)}\|_{H_i^{-1}}^2 + \frac{1}{2k\alpha} R_\infty^2 \|h_k\|_1 \\ &\leq \frac{\alpha}{k} \|h_k\|_1 + \frac{1}{2k\alpha} R_\infty^2 \|h_k\|_1 \end{aligned}$$

(second line is a theorem)

also have (with  $\alpha = R_\infty^2$ ) and for compact sets  $C$

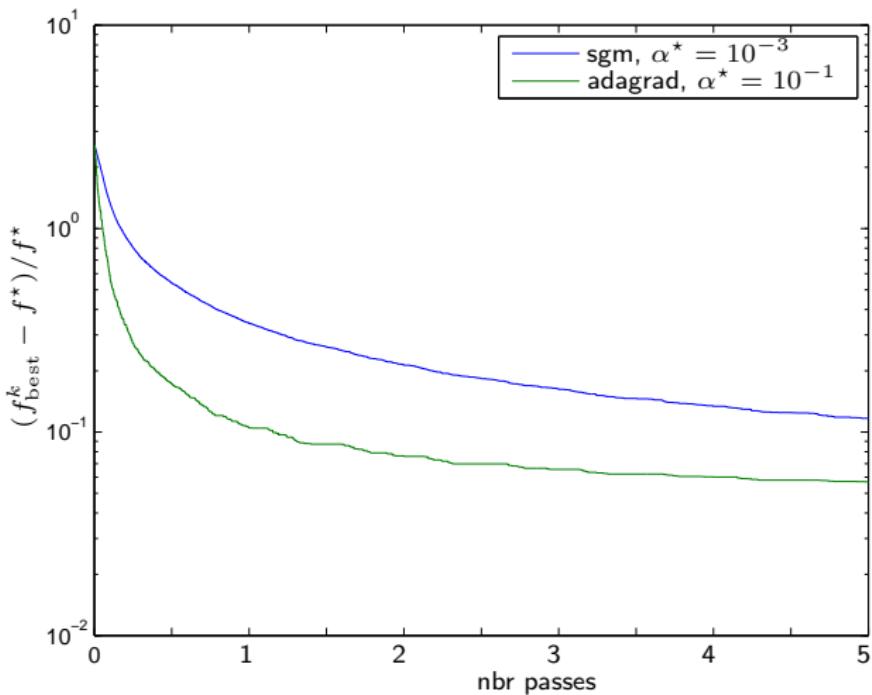
$$f_{\text{best}}^{(k)} - f^* \leq \frac{2}{k} \inf_{h \geq 0} \left\{ \sup_{x,y \in C} (x-y)^T \mathbf{diag}(h)(x-y) + \sum_{i=1}^k \|g^{(i)}\|_{\mathbf{diag}(h)^{-1}}^2 \right\}$$

# Example

Classification problem:

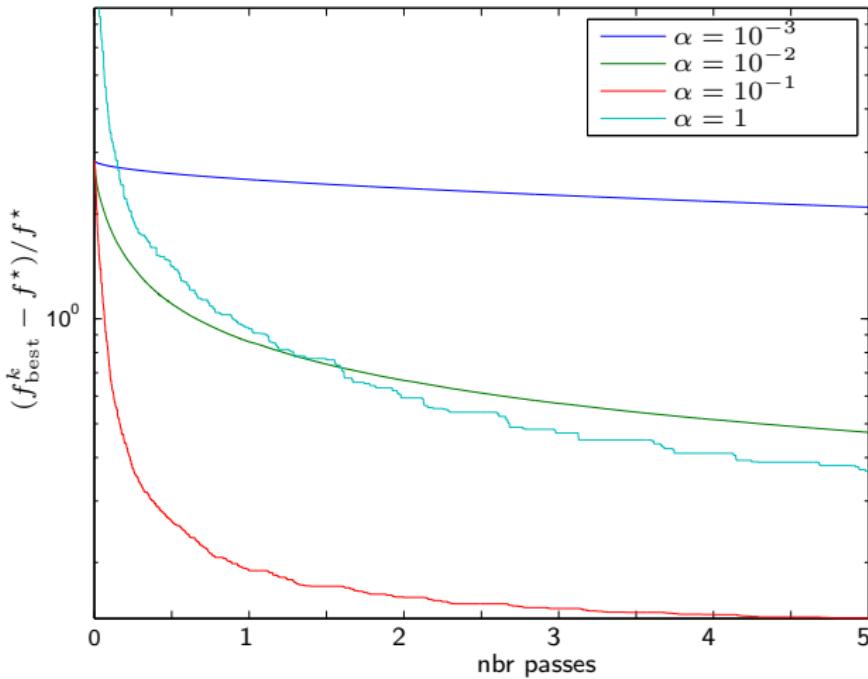
- **Data:**  $\{a_i, b_i\}$ ,  $i = 1, \dots, 50000$ 
  - $a_i \in \mathbf{R}^{1000}$
  - $b \in \{-1, 1\}$
  - Data created with 5% mis-classifications w.r.t.  $w = \mathbf{1}$ ,  $v = 0$
- **Objective:** find classifiers  $w \in \mathbf{R}^{1000}$  and  $v \in \mathbf{R}$  such that
  - $a_i^T w + v > 1$  if  $b = 1$
  - $a_i^T w + v < 1$  if  $b = -1$
- **Optimization method:**
  - Minimize hinge-loss:  $\sum_i \max(0, 1 - b_i(a_i^T w + v))$
  - Choose example uniformly at random, take sub-gradient step w.r.t. that example

## Best subgradient method vs best AdaGrad



Often best AdaGrad performs better than best subgradient method

## AdaGrad with different step-sizes $\alpha$ :



Sensitive to step-size selection (like standard subgradient method)

# Localization and Cutting-Plane Methods

- cutting-plane oracle
- finding cutting-planes
- localization algorithms
- specific cutting-plane methods
- epigraph cutting-plane method
- lower bounds and stopping criteria

## **Localization and cutting-plane methods**

- based on idea of ‘localizing’ desired point in some set, which becomes smaller at each step
- like subgradient methods, require computation of a subgradient of objective or constraint functions at each step
- in particular, directly handle nondifferentiable convex (and quasiconvex) problems
- typically require more memory and computation per step than subgradient methods
- but can be much more efficient (in theory and practice) than subgradient methods

## Cutting-plane oracle

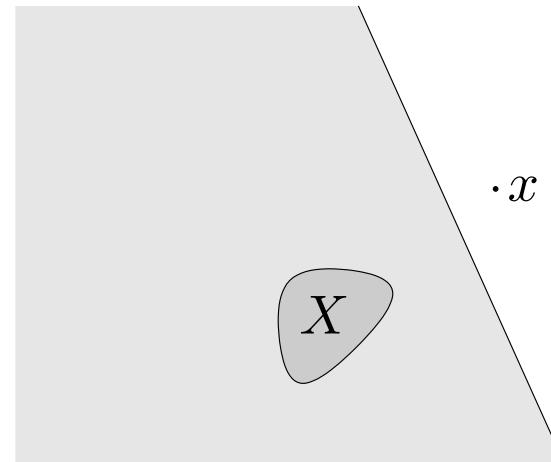
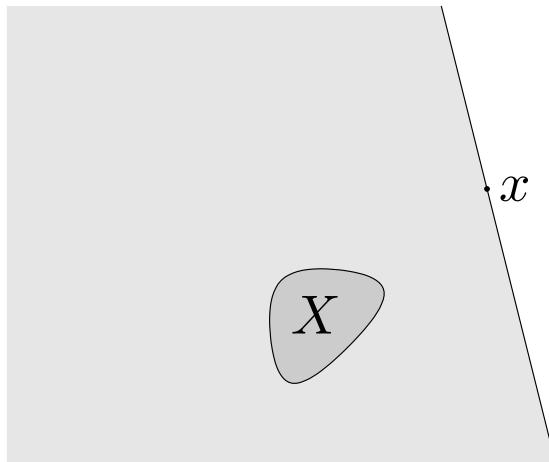
- goal: find a point in convex set  $X \subseteq \mathbf{R}^n$ , or determine that  $X = \emptyset$
- our only access to or description of  $X$  is through a *cutting-plane oracle*
- when cutting-plane oracle is *queried* at  $x \in \mathbf{R}^n$ , it either
  - asserts that  $x \in X$ , or
  - returns a separating hyperplane between  $x$  and  $X$ :  $a \neq 0$ ,

$$a^T z \leq b \text{ for } z \in X, \quad a^T x \geq b$$

- $(a, b)$  called a *cutting-plane*, or *cut*, since it eliminates the halfspace  $\{z \mid a^T z > b\}$  from our search for a point in  $X$

## Neutral and deep cuts

- if  $a^T x = b$  ( $x$  is on boundary of halfspace that is cut) cutting-plane is called *neutral cut*
- if  $a^T x > b$  ( $x$  lies in interior of halfspace that is cut), cutting-plane is called *deep cut*



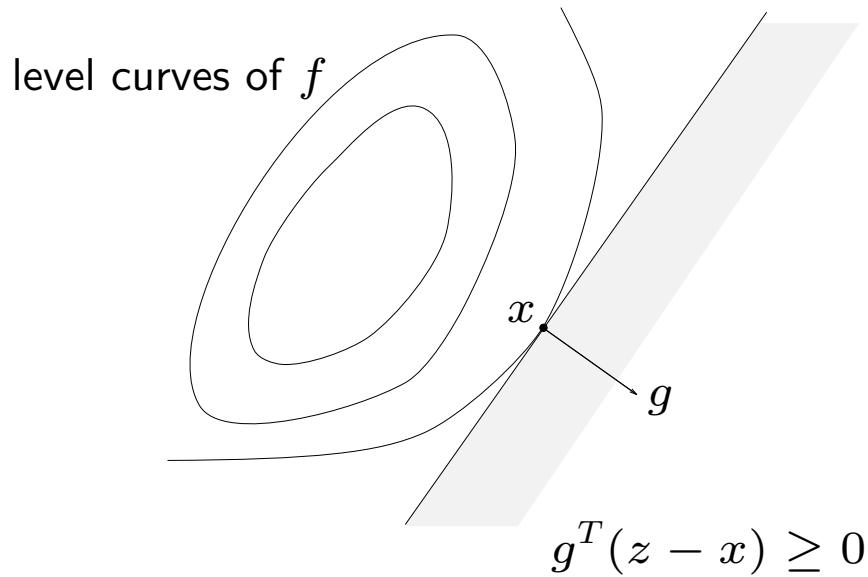
## Unconstrained minimization

- minimize convex  $f : \mathbf{R}^n \rightarrow \mathbf{R}$
- $X$  is set of optimal points (minimizers)
- given  $x$ , find  $g \in \partial f(x)$
- from  $f(z) \geq f(x) + g^T(z - x)$  we conclude

$$g^T(z - x) > 0 \implies f(z) > f(x)$$

i.e., all points in halfspace  $g^T(z - x) \geq 0$  are **worse** than  $x$ , and in particular not optimal

- so  $g^T(z - x) \leq 0$  is (neutral) cutting-plane at  $x$  ( $a = g$ ,  $b = g^T x$ )



- by evaluating  $g \in \partial f(x)$  we rule out a halfspace in our search for  $x^*$
- **idea:** get one bit of info (on location of  $x^*$ ) by evaluating  $g$

## Deep cut for unconstrained minimization

- suppose we know a number  $\bar{f}$  with  $f(x) > \bar{f} \geq f^*$   
(e.g., the smallest value of  $f$  found so far in an algorithm)
- from  $f(z) \geq f(x) + g^T(z - x)$ , we have

$$f(x) + g^T(z - x) > \bar{f} \implies f(z) > \bar{f} \geq f^* \implies z \notin X$$

so we have deep cut

$$g^T(z - x) + f(x) - \bar{f} \leq 0$$

## Feasibility problem

$$\begin{aligned} & \text{find} && x \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

$f_1, \dots, f_m$  convex;  $X$  is set of feasible points

- if  $x$  not feasible, find  $j$  with  $f_j(x) > 0$ , and evaluate  $g_j \in \partial f_j(x)$
- since  $f_j(z) \geq f_j(x) + g_j^T(z - x)$ ,

$$f_j(x) + g_j^T(z - x) > 0 \implies f_j(z) > 0 \implies z \notin X$$

i.e., any feasible  $z$  satisfies the inequality  $f_j(x) + g_j^T(z - x) \leq 0$

- this gives a deep cut

## Inequality constrained problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

$f_0, \dots, f_m : \mathbf{R}^n \rightarrow \mathbf{R}$  convex;  $X$  is set of optimal points;  $p^*$  is optimal value

- if  $x$  is not feasible, say  $f_j(x) > 0$ , we have (deep) *feasibility cut*

$$f_j(x) + g_j^T(z - x) \leq 0, \quad g_j \in \partial f_j(x)$$

- if  $x$  is feasible, we have (neutral) *objective cut*

$$g_0^T(z - x) \leq 0, \quad g_0 \in \partial f_0(x)$$

(or, deep cut  $g_0^T(z - x) + f_0(x) - \bar{f} \leq 0$  if  $\bar{f} \in [p^*, f_0(x)]$  is known)

## Localization algorithm

basic (conceptual) localization (or cutting-plane) algorithm:

**given** initial polyhedron  $\mathcal{P}_0 = \{z \mid Cz \leq d\}$  known to contain  $X$

$k := 0$

**repeat**

    Choose a point  $x^{(k+1)}$  in  $\mathcal{P}_k$

    Query the cutting-plane oracle at  $x^{(k+1)}$

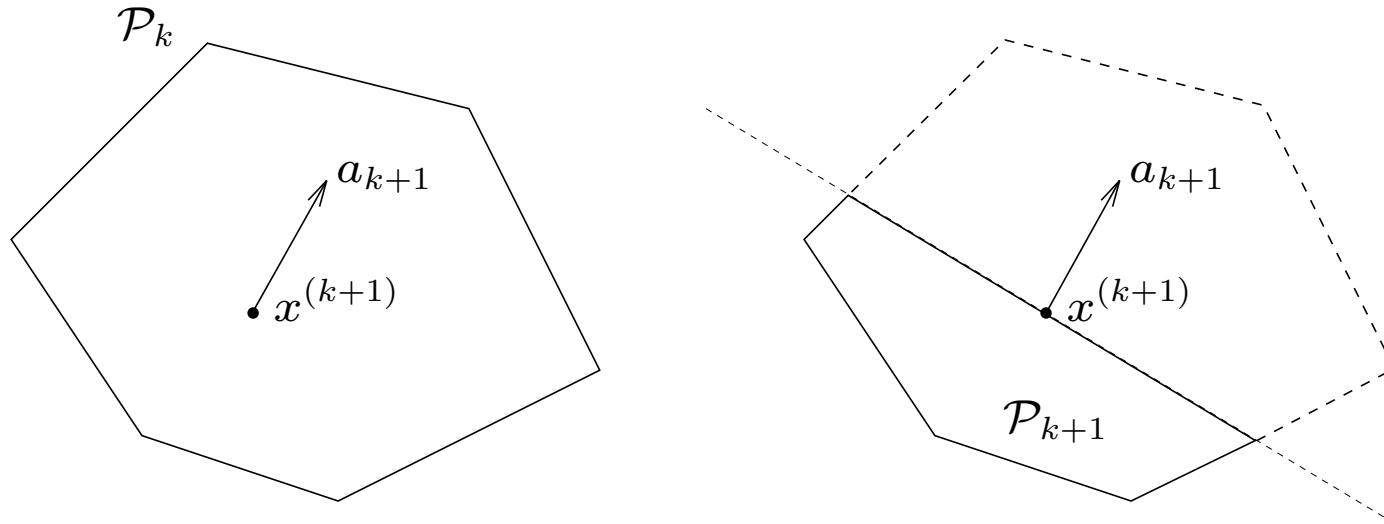
    If  $x^{(k+1)} \in X$ , quit

    Else, add new cutting-plane  $a_{k+1}^T z \leq b_{k+1}$ :

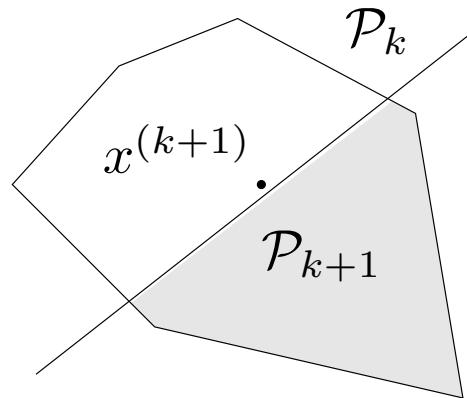
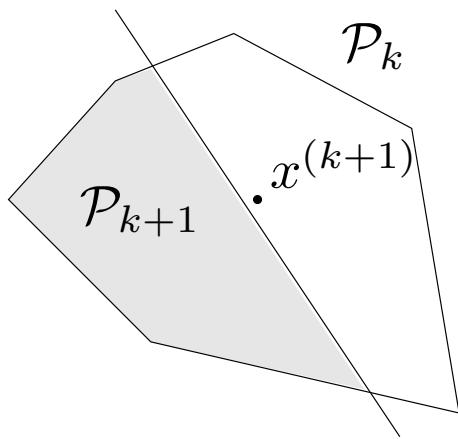
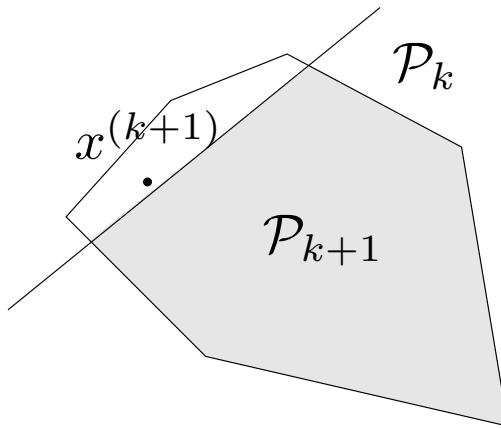
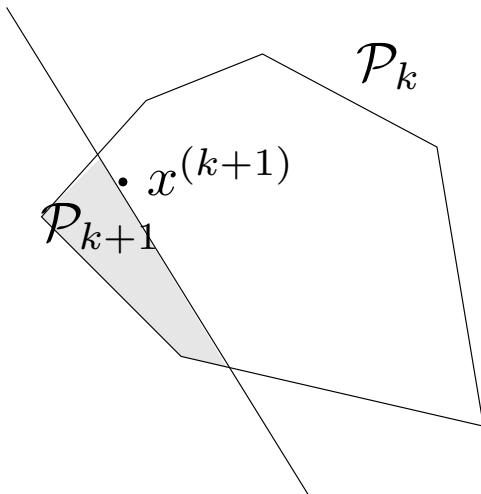
$\mathcal{P}_{k+1} := \mathcal{P}_k \cap \{z \mid a_{k+1}^T z \leq b_{k+1}\}$

        If  $\mathcal{P}_{k+1} = \emptyset$ , quit

$k := k + 1$



- $\mathcal{P}_k$  gives our uncertainty of  $x^*$  at iteration  $k$
- want to pick  $x^{(k+1)}$  so that  $\mathcal{P}_{k+1}$  is as small as possible, no matter what cut is made
- want  $x^{(k+1)}$  near center of  $\mathcal{P}^{(k)}$



## Example: Bisection on $\mathbf{R}$

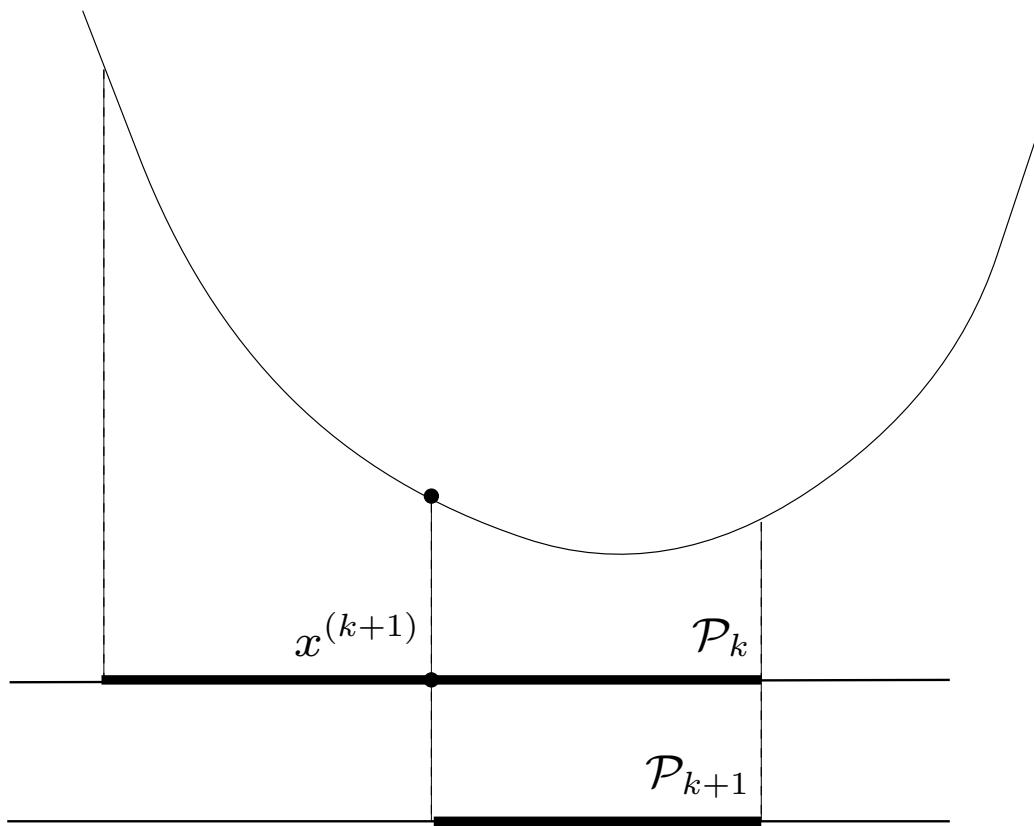
- minimize convex  $f : \mathbf{R} \rightarrow \mathbf{R}$
- $\mathcal{P}_k$  is interval
- obvious choice for query point:  $x^{(k+1)} := \text{midpoint}(\mathcal{P}_k)$

### bisection algorithm

**given** interval  $\mathcal{P}_0 = [l, u]$  containing  $x^*$

repeat

1.  $x := (l + u)/2$
2. evaluate  $f'(x)$
3. if  $f'(x) < 0$ ,  $l := x$ ; else  $u := x$



$$\text{length}(\mathcal{P}_{k+1}) = u_{k+1} - l_{k+1} = \frac{u_k - l_k}{2} = (1/2)\text{length}(\mathcal{P}_k)$$

and so  $\text{length}(\mathcal{P}_k) = 2^{-k}\text{length}(\mathcal{P}_0)$

### interpretation:

- $\text{length}(\mathcal{P}_k)$  measures our uncertainty in  $x^*$
- uncertainty is halved at each iteration; get exactly one bit of info about  $x^*$  per iteration
- # steps required for uncertainty (in  $x^*$ )  $\leq r$ :

$$\log_2 \frac{\text{length}(\mathcal{P}_0)}{r} = \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

## Specific cutting-plane methods

methods vary in choice of query point

- *center of gravity (CG) algorithm:*  
 $x^{(k+1)}$  is center of gravity of  $\mathcal{P}_k$
- *maximum volume ellipsoid (MVE) cutting-plane method:*  
 $x^{(k+1)}$  is center of maximum volume ellipsoid contained in  $\mathcal{P}_k$
- *Chebyshev center cutting-plane method:*  
 $x^{(k+1)}$  is Chebyshev center of  $\mathcal{P}_k$
- *analytic center cutting-plane method (ACCPM):*  
 $x^{(k+1)}$  is analytic center of (inequalities defining)  $\mathcal{P}_k$

## Center of gravity algorithm

take  $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$  (center of gravity)

$$\text{CG}(\mathcal{P}_k) = \int_{\mathcal{P}_k} x \, dx \Bigg/ \int_{\mathcal{P}_k} dx$$

**theorem.** if  $C \subseteq \mathbf{R}^n$  convex,  $x_{\text{cg}} = \text{CG}(C)$ ,  $g \neq 0$ ,

$$\mathbf{vol} (C \cap \{x \mid g^T(x - x_{\text{cg}}) \leq 0\}) \leq (1 - 1/e) \mathbf{vol}(C) \approx 0.63 \mathbf{vol}(C)$$

(independent of dimension  $n$ )

hence in CG algorithm,  $\mathbf{vol}(\mathcal{P}_k) \leq 0.63^k \mathbf{vol}(\mathcal{P}_0)$

## Convergence of CG cutting-plane method

- suppose  $\mathcal{P}_0$  lies in ball of radius  $R$ ,  $X$  includes ball of radius  $r$   
(can take  $X$  as set of  $\epsilon$ -suboptimal points)
- suppose  $x^{(1)}, \dots, x^{(k)} \notin X$ , so  $\mathcal{P}_k \supseteq X$
- we have

$$\alpha_n r^n \leq \mathbf{vol}(\mathcal{P}_k) \leq (0.63)^k \mathbf{vol}(\mathcal{P}_0) \leq (0.63)^k \alpha_n R^n$$

where  $\alpha_n$  is volume of unit ball in  $\mathbf{R}^n$

- so  $k \leq 1.51n \log_2(R/r)$  (cf. bisection on  $\mathbf{R}$ )

## advantages of CG-method

- guaranteed convergence
- affine-invariance
- number of steps proportional to dimension  $n$ , log of uncertainty reduction

## disadvantages

- finding  $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$  is **much harder** than original problem  
(but, can modify CG-method to work with approximate CG computation)

## Maximum volume ellipsoid method

- $x^{(k+1)}$  is center of maximum volume ellipsoid in  $\mathcal{P}_k$   
(can compute as convex problem)
- affine-invariant
- can show  $\text{vol}(\mathcal{P}_{k+1}) \leq (1 - 1/n) \text{vol}(\mathcal{P}_k)$
- hence can bound number of steps:

$$k \leq \frac{n \log(R/r)}{-\log(1 - 1/n)} \approx n^2 \log(R/r)$$

- if cutting-plane oracle cost is not small, MVE is a good practical method

## Chebyshev center method

- $x^{(k+1)}$  is center of largest Euclidean ball in  $\mathcal{P}_k$   
(can compute via LP)
- not affine invariant; sensitive to scaling

## Analytic center cutting-plane method

- $x^{(k+1)}$  is analytic center of  $\mathcal{P}_k = \{z \mid a_i^T z \leq b_i, i = 1, \dots, q\}$

$$x^{(k+1)} = \operatorname{argmin}_x - \sum_{i=1}^q \log(b_i - a_i^T x)$$

- $x^{(k+1)}$  can be computed using infeasible start Newton method
- works quite well in practice (more on this next lecture)

# Extensions

## *Multiple cuts*

- oracle returns set of linear inequalities instead of just one, *e.g.*,
  - all violated inequalities
  - all inequalities (including *shallow cuts*)
  - multiple deep cuts
- at each iteration, append (set of) new inequalities to those defining  $\mathcal{P}_k$

## *Nonlinear cuts*

- use nonlinear convex inequalities instead of linear ones
- localization set no longer a polyhedron
- some methods (*e.g.*, ACCPM) still work

## Dropping constraints

- the problem:
  - number of linear inequalities defining  $\mathcal{P}_k$  increases at each iteration
  - hence, computational effort to compute  $x^{(k+1)}$  increases
- the solution: drop or prune constraints
  - drop redundant constraints
  - keep only a fixed number  $N$  of (the most relevant) constraints  
(can cause localization polyhedron to increase!)

## Epigraph cutting-plane method

apply cutting-plane method to epigraph form problem

$$\begin{aligned} & \text{minimize} && t \\ & \text{subject to} && f_0(x) \leq t \\ & && f_i(x) \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

with variables  $x \in \mathbf{R}^n$  and  $t$

at each  $(x, t)$ , need cutting-plane oracle that separates  $(x, t)$  from  $(x^*, p^*)$

- if  $x^{(k)}$  is infeasible for original problem and violates  $j$ th constraint, add the cutting-plane

$$f_j(x^{(k)}) + g_j^T(x - x^{(k)}) \leq 0, \quad g_j \in \partial f_j(x^{(k)})$$

- if  $x^{(k)}$  is feasible for original problem, add *two* cutting-planes

$$f_0(x^{(k)}) + g_0^T(x - x^{(k)}) \leq t, \quad t \leq f_0(x^{(k)})$$

where  $g_0 \in \partial f_0(x^{(k)})$

## PWL lower bound on convex function

- suppose we have evaluated  $f$  and a subgradient of  $f$  at  $x^{(1)}, \dots, x^{(q)}$
- for all  $z$ ,

$$f(z) \geq f(x^{(i)}) + g^{(i)T}(z - x^{(i)}), \quad i = 1, \dots, q$$

and so

$$f(z) \geq \hat{f}(z) = \max_{i=1, \dots, q} \left( f(x^{(i)}) + g^{(i)T}(z - x^{(i)}) \right).$$

- $\hat{f}$  is a convex piecewise-linear global underestimator of  $f$

## Lower bound

- in solving convex problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && Cx \preceq d \end{aligned}$$

we have evaluated some of the  $f_i$  and subgradients at  $x^{(1)}, \dots, x^{(k)}$

- form piecewise-linear approximations  $\hat{f}_0, \dots, \hat{f}_m$
- form PWL relaxed problem

$$\begin{aligned} & \text{minimize} && \hat{f}_0(x) \\ & \text{subject to} && \hat{f}_i(x) \leq 0, \quad i = 1, \dots, m \\ & && Cx \preceq d \end{aligned}$$

(can be solved via LP)

- optimal value is a lower bound on  $p^*$

# Analytic Center Cutting-Plane Method

- analytic center cutting-plane method
- computing the analytic center
- pruning constraints
- lower bound and stopping criterion

## Analytic center cutting-plane method

**analytic center** of polyhedron  $\mathcal{P} = \{z \mid a_i^T z \leq b_i, i = 1, \dots, m\}$  is

$$\text{AC}(\mathcal{P}) = \operatorname{argmin}_z - \sum_{i=1}^m \log(b_i - a_i^T z)$$

**ACCPM** is localization method with next query point  $x^{(k+1)} = \text{AC}(\mathcal{P}_k)$   
(found by Newton's method)

## ACCPM algorithm

**given** an initial polyhedron  $\mathcal{P}_0$  known to contain  $X$ .

$k := 0$ .

**repeat**

    Compute  $x^{(k+1)} = \text{AC}(\mathcal{P}_k)$ .

    Query cutting-plane oracle at  $x^{(k+1)}$ .

    If  $x^{(k+1)} \in X$ , quit.

    Else, add returned cutting-plane inequality to  $\mathcal{P}$ .

$$\mathcal{P}_{k+1} := \mathcal{P}_k \cap \{z \mid a^T z \leq b\}$$

    If  $\mathcal{P}_{k+1} = \emptyset$ , quit.

$k := k + 1$ .

## Constructing cutting-planes

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

$f_0, \dots, f_m : \mathbf{R}^n \rightarrow \mathbf{R}$  convex;  $X$  is set of optimal points;  $p^*$  is optimal value

- if  $x$  is not feasible, say  $f_j(x) > 0$ , we have (deep) *feasibility cut*

$$f_j(x) + g_j^T(z - x) \leq 0, \quad g_j \in \partial f_j(x)$$

- if  $x$  is feasible, we have (deep) *objective cut*

$$g_0^T(z - x) + f_0(x) - f_{\text{best}}^{(k)} \leq 0, \quad g_0 \in \partial f_0(x)$$

## Computing the analytic center

we must solve the problem

$$\text{minimize } \Phi(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$$

where  $\text{dom } \Phi = \{x \mid a_i^T x < b_i, i = 1, \dots, m\}$

- **challenge:** we are not given a point in  $\text{dom } \Phi$
- some options:
  - use phase I method to find a point in  $\text{dom } \Phi$  (or determine that  $\text{dom } \Phi = \emptyset$ ); then use standard Newton method to compute AC
  - use infeasible start Newton method starting from a point outside  $\text{dom } \Phi$

## Infeasible start Newton method

$$\begin{aligned} & \text{minimize} && -\sum_{i=1}^m \log y_i \\ & \text{subject to} && y = b - Ax \end{aligned}$$

with variables  $x$  and  $y$

- can be started from *any*  $x$  and *any*  $y \succ 0$
- *e.g.*: take initial  $x$  as previous point  $x_{\text{prev}}$ , and choose  $y$  as

$$y_i = \begin{cases} b_i - a_i^T x & b_i - a_i^T x > 0 \\ 1 & \text{otherwise} \end{cases}$$

- define primal and dual residuals as

$$r_p = y + Ax - b, \quad r_d = \begin{bmatrix} A^T \nu \\ g + \nu \end{bmatrix}$$

where  $g = -\text{diag}(1/y_i)\mathbf{1}$  is gradient of objective and  $r = (r_d, r_p)$

- Newton step at  $(x, y, \nu)$  is defined by

$$\begin{bmatrix} 0 & 0 & A^T \\ 0 & H & I \\ A & I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \end{bmatrix},$$

where  $H = \text{diag}(1/y_i^2)$  is Hessian of the objective

- solve this system by block elimination

$$\Delta x = -(A^T H A)^{-1} (A^T g - A^T H r_p)$$

$$\Delta y = -A \Delta x - r_p$$

$$\Delta \nu = -H \Delta y - g - \nu$$

- options for computing  $\Delta x$ :
  - form  $A^T H A$ , then use dense or sparse Cholesky factorization
  - solve (diagonally scaled) least-squares problem

$$\Delta x = \operatorname{argmin}_z \left\| H^{1/2} A z - H^{1/2} r_p + H^{-1/2} g \right\|_2$$

- use iterative method such as conjugate gradients to (approximately) solve for  $\Delta x$

## Infeasible start Newton method algorithm

**given** starting point  $x, y \succ 0$ , tolerance  $\epsilon > 0$ ,  $\alpha \in (0, 1/2)$ ,  $\beta \in (0, 1)$ .

$\nu := 0$ .

**repeat**

1. Compute Newton step  $(\Delta x, \Delta y, \Delta \nu)$  by block elimination.
2. *Backtracking line search on  $\|r\|_2$ .*

$t := 1$ .

**while**  $y + t\Delta y \not\succ 0$ ,  $t := \beta t$ .

**while**  $\|r(x + t\Delta x, y + t\Delta y, \nu + t\Delta \nu)\|_2 > (1 - \alpha t)\|r(x, y, \nu)\|_2$ ,  
 $t := \beta t$ .

3. *Update.*  $x := x + t\Delta x$ ,  $y := y + t\Delta y$ ,  $\nu := \nu + t\Delta \nu$ .

**until**  $y = b - Ax$  and  $\|r(x, y, \nu)\|_2 \leq \epsilon$ .

# Properties

- once any equality constraint is satisfied, it remains satisfied for all future iterates
- once a step size  $t = 1$  is taken, all equality constraints are satisfied
- if  $\text{dom } \Phi \neq \emptyset$ ,  $t = 1$  occurs in finite number of steps
- if  $\text{dom } \Phi = \emptyset$ , algorithm never converges

## Pruning constraints

- let  $x^*$  be analytic center of  $\mathcal{P} = \{z \mid a_i^T z \leq b_i, i = 1, \dots, m\}$
- let  $H^*$  be Hessian of barrier at  $x^*$ ,

$$H^* = -\nabla^2 \sum_{i=1}^m \log(b_i - a_i^T z) \Big|_{z=x^*} = \sum_{i=1}^m \frac{a_i a_i^T}{(b_i - a_i^T x^*)^2}$$

- then,  $\mathcal{P} \subseteq \mathcal{E} = \{z \mid (z - x^*)^T H^* (z - x^*) \leq m^2\}$

define (ir)relevance measure  $\eta_i = \frac{b_i - a_i^T x^*}{\sqrt{a_i^T H^{*-1} a_i}}$

- $\eta_i/m$  is normalized distance from hyperplane  $a_i^T x = b_i$  to outer ellipsoid
- if  $\eta_i \geq m$ , then constraint  $a_i^T x \leq b_i$  is redundant

common ACCPM constraint dropping schemes:

- drop all constraints with  $\eta_i \geq m$  (guaranteed to not change  $\mathcal{P}$ )
- drop constraints in order of irrelevance, keeping constant number, usually  $3n - 5n$

## PWL lower bound on convex function

- suppose  $f$  is convex, and  $g^{(i)} \in \partial f(x^{(i)})$ ,  $i = 1, \dots, m$
- then we have

$$\hat{f}(z) = \max_{i=1,\dots,m} \left( f(x^{(i)}) + g^{(i)T}(z - x^{(i)}) \right) \leq f(z)$$

- $\hat{f}$  is PWL lower bound on  $f$

## Lower bound in ACCPM

- in solving convex problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_1(x) \leq 0, \\ & && Cx \preceq d \end{aligned}$$

(by taking max of constraint functions we can assume there is only one)

- we have evaluated  $f_0$  and subgradient  $g_0$  at  $x^{(1)}, \dots, x^{(q)}$
- we have evaluated  $f_1$  and subgradient  $g_1$  at  $x^{(q+1)}, \dots, x^{(k)}$
- form piecewise-linear approximations  $\hat{f}_0, \hat{f}_1$

- form PWL relaxed problem

$$\begin{aligned} & \text{minimize} && \hat{f}_0(x) \\ & \text{subject to} && \hat{f}_1(x) \leq 0, \\ & && Cx \preceq d \end{aligned}$$

(can be solved via LP)

- optimal value is a lower bound on  $p^*$
- can easily construct a lower bound on the PWL relaxed problem, as a by-product of the analytic centering computation
- this, in turn, gives a lower bound on the original problem

- form dual of PWL relaxed problem

$$\begin{aligned}
 \text{maximize} \quad & \sum_{i=1}^q \lambda_i (f_0(x^{(i)}) - g_0^{(i)T} x^{(i)}) \\
 & + \sum_{i=q+1}^k \lambda_i (f_1(x^{(i)}) - g_1^{(i)T} x^{(i)}) - d^T \mu \\
 \text{subject to} \quad & \sum_{i=1}^q \lambda_i g_0^{(i)} + \sum_{i=q+1}^k \lambda_i g_1^{(i)} + C^T \mu = 0 \\
 & \mu \succeq 0, \quad \lambda \succeq 0, \quad \sum_{i=1}^q \lambda_i = 1,
 \end{aligned}$$

- optimality condition for  $x^{(k+1)}$

$$\begin{aligned}
 & \sum_{i=1}^q \frac{g_0^{(i)}}{f_{\text{best}}^{(i)} - f_0(x^{(i)}) - g_0^{(i)T}(x^{(k+1)} - x^{(i)})} + \\
 & \sum_{i=q+1}^k \frac{g_1^{(i)}}{-f_1(x^{(i)}) - g_1^{(i)T}(x^{(k+1)} - x^{(i)})} + \sum_{i=1}^m \frac{c_i}{d_i - c_i^T x^{(k+1)}} = 0.
 \end{aligned}$$

- take  $\tau_i = 1/(f_{\text{best}}^{(i)} - f_0(x^{(i)}) - g_0^{(i)T}(x^{(k+1)} - x^{(i)}))$  for  $i = 1, \dots, q$ .
- construct a dual feasible point by taking

$$\begin{aligned}\lambda_i &= \begin{cases} \tau_i / \mathbf{1}^T \tau & \text{for } i = 1, \dots, q \\ 1/(-f_1(x^{(i)}) - g_1^{(i)T}(x^{(k+1)} - x^{(i)}))(\mathbf{1}^T \tau) & \text{for } i = q + 1, \dots, k, \end{cases} \\ \mu_i &= 1/(d_i - c_i^T x^{(k+1)})(\mathbf{1}^T \tau) \quad i = 1, \dots, m.\end{aligned}$$

- using these values of  $\lambda$  and  $\mu$ , we conclude that

$$p^* \geq l^{(k+1)},$$

where  $l^{(k+1)} = \sum_{i=1}^q \lambda_i (f_0(x^{(i)}) - g_0^{(i)T} x^{(i)}) + \sum_{i=q+1}^k \lambda_i (f_1(x^{(i)}) - g_1^{(i)T} x^{(i)}) - d^T \mu$ .

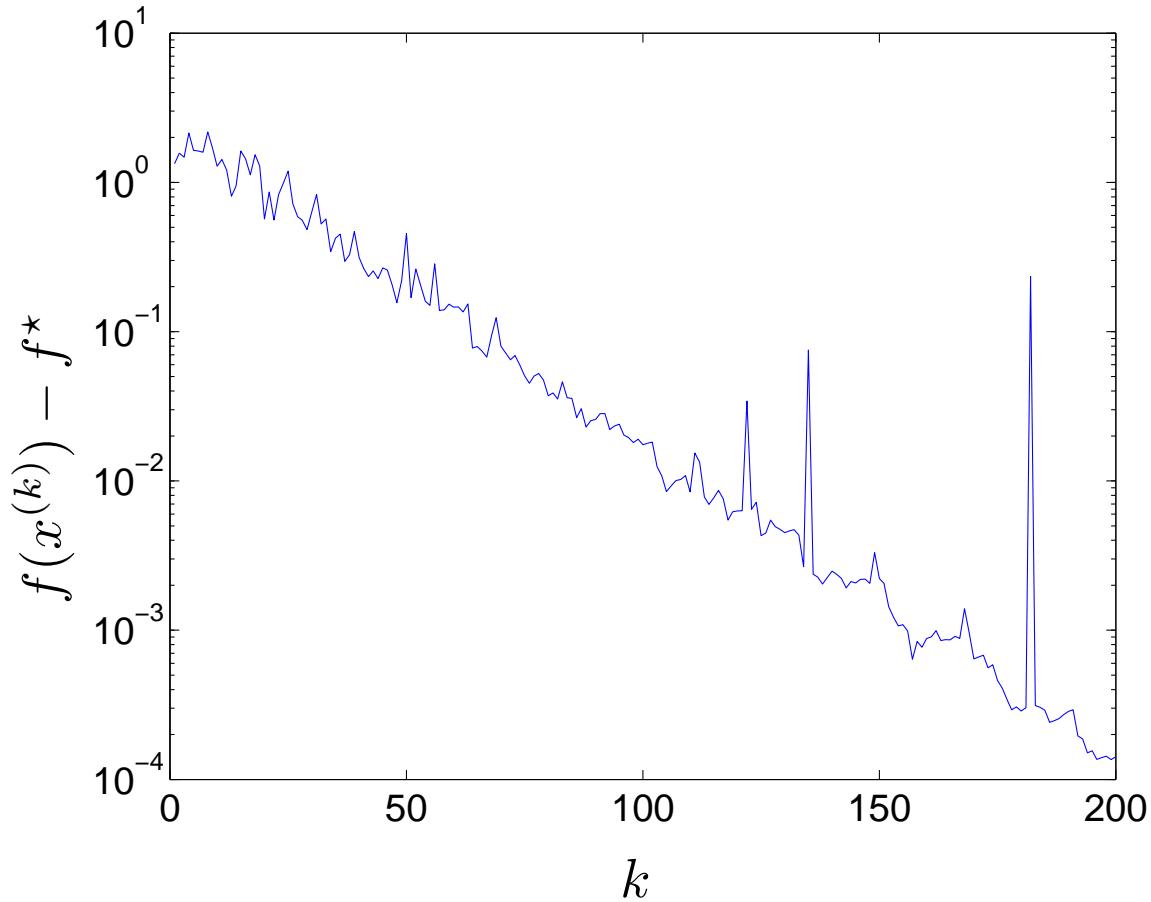
## Stopping criterion

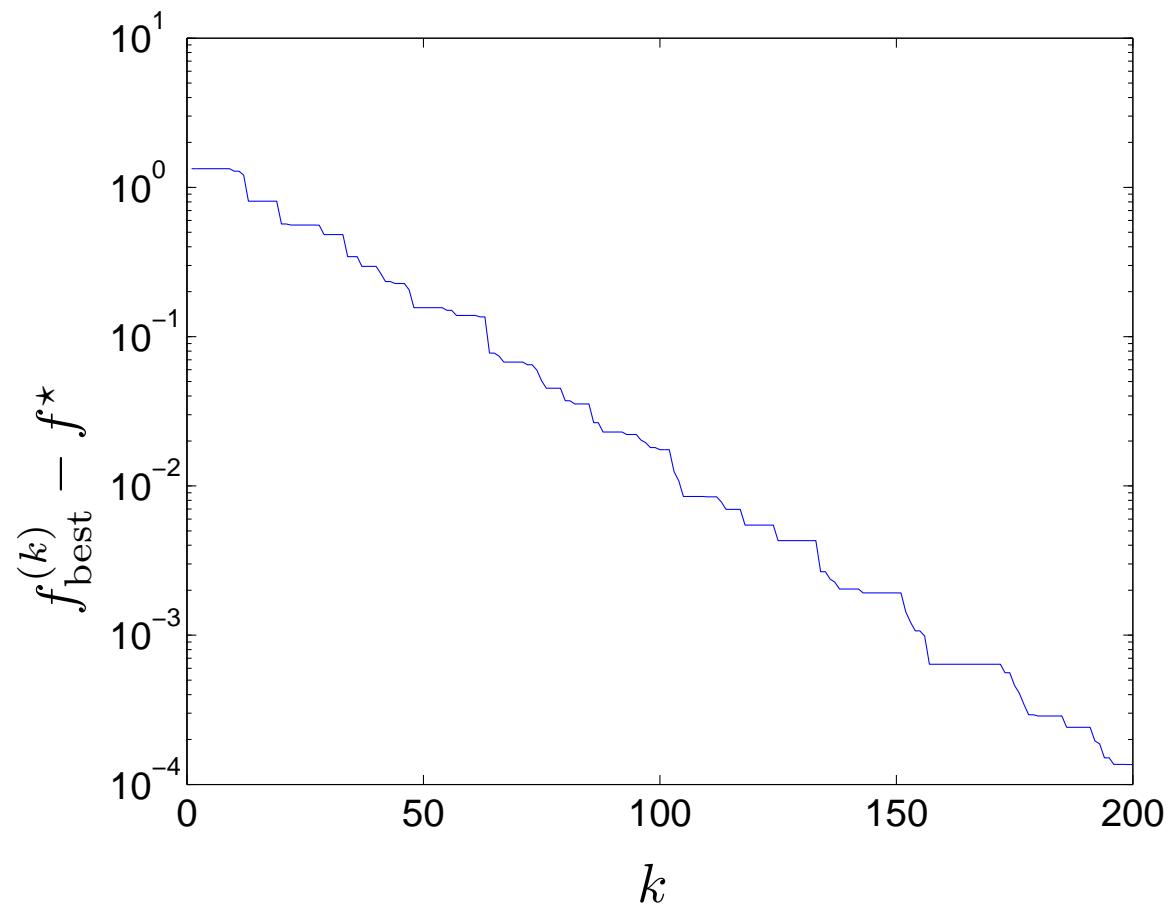
since ACCPM isn't a descent method, we keep track of best point found, and best lower bound

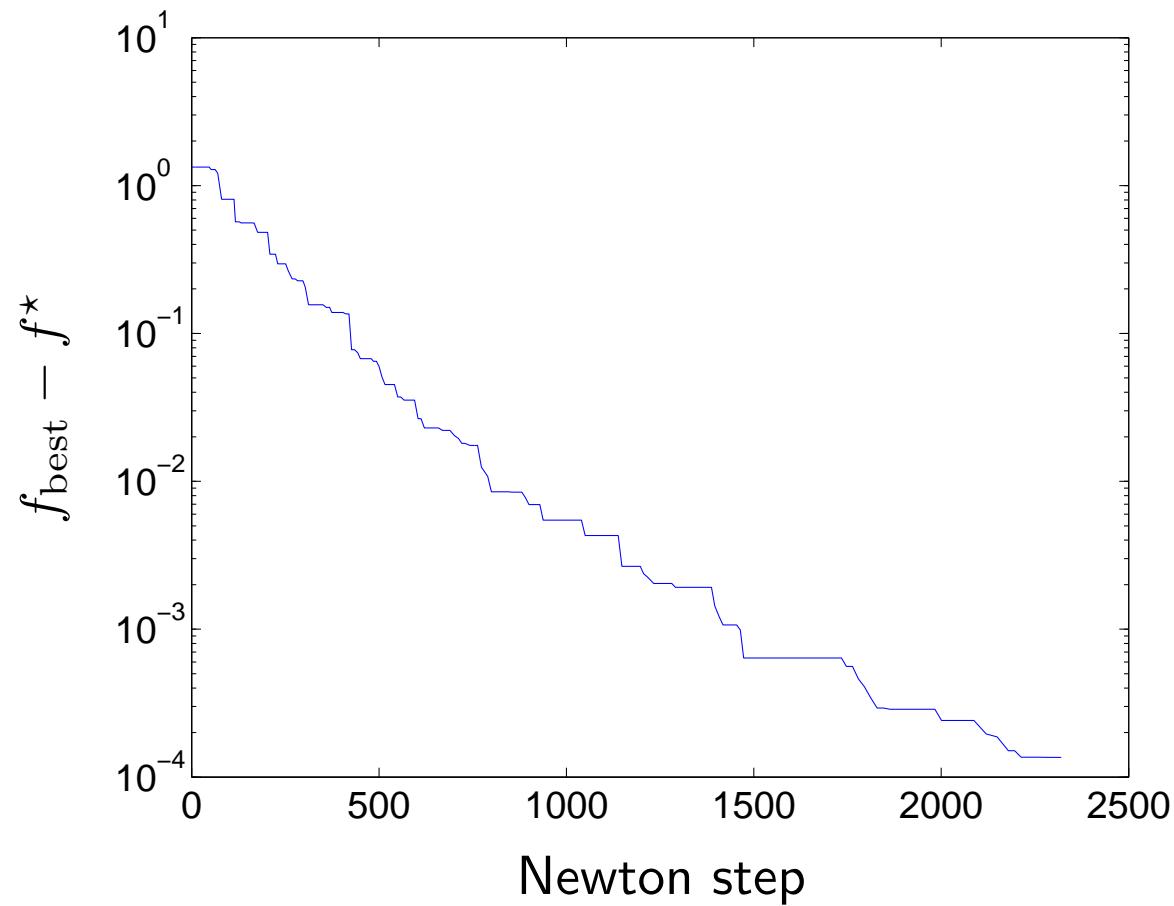
- best function value so far:  $f_{\text{best}}^{(k)} = \min_{i=1,\dots,k} f_0(x^{(k)})$
- best lower bound so far:  $l_{\text{best}}^{(k)} = \max_{i=1,\dots,k} l(x^{(k)})$
- can stop when  $f_{\text{best}}^{(k)} - l_{\text{best}}^{(k)} \leq \epsilon$
- guaranteed to be  $\epsilon$ -suboptimal

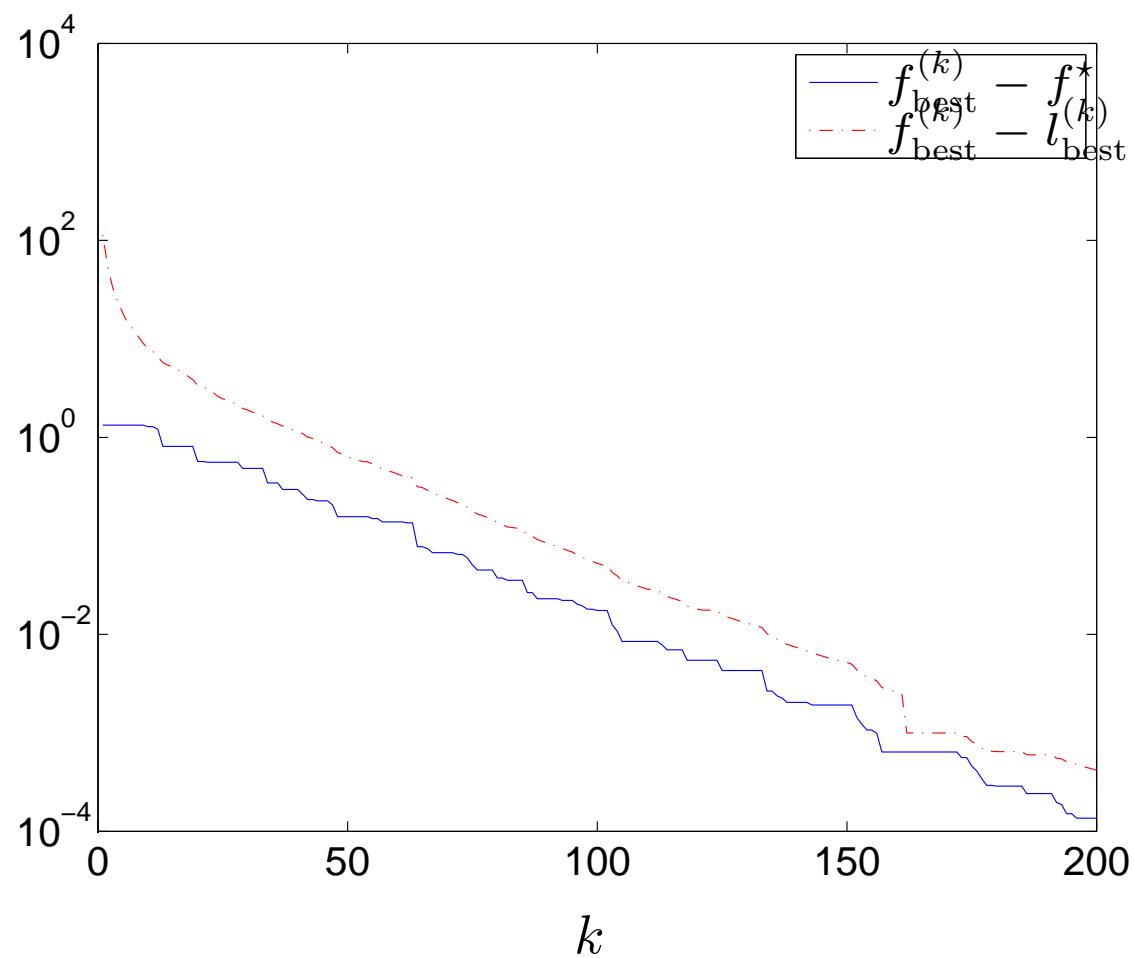
## Example: Piecewise linear minimization

problem instance with  $n = 20$  variables,  $m = 100$  terms,  $f^* \approx 1.1$



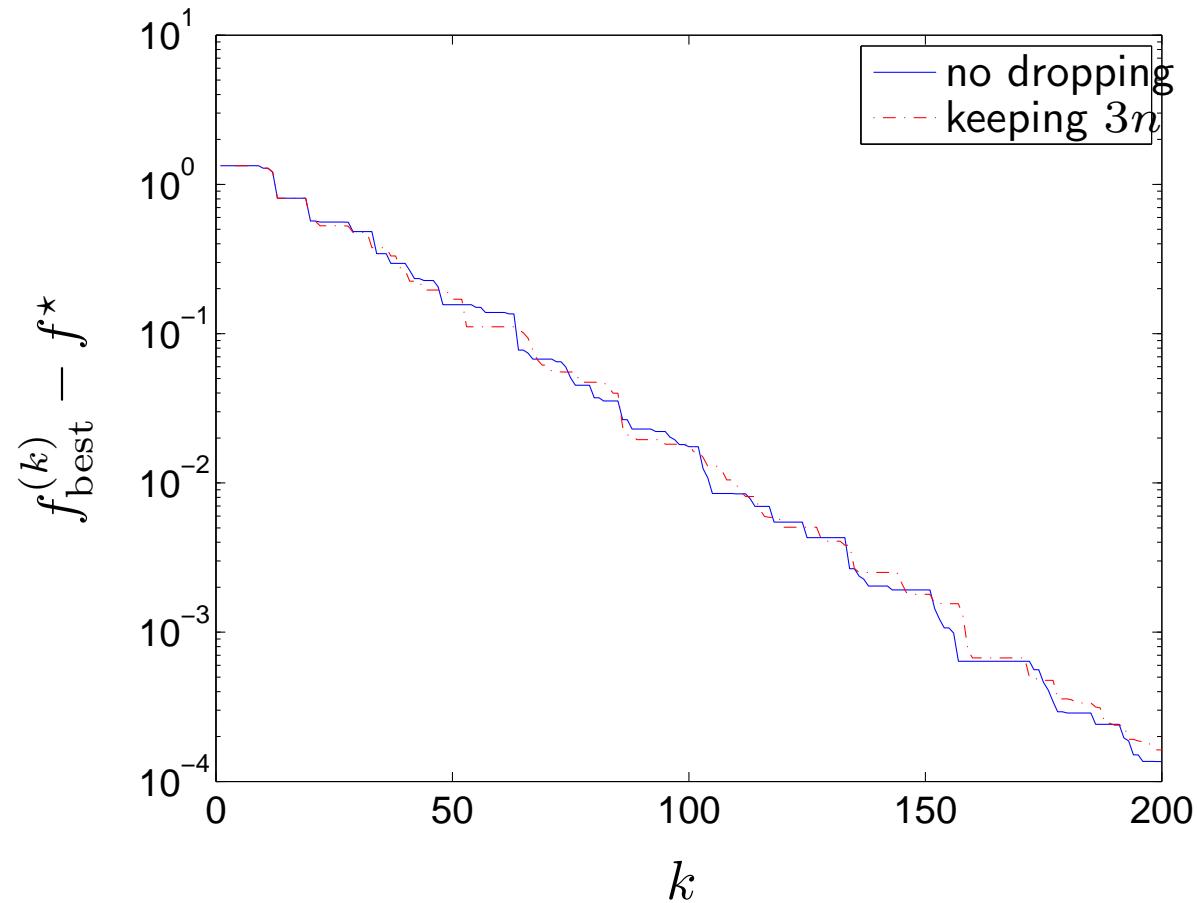




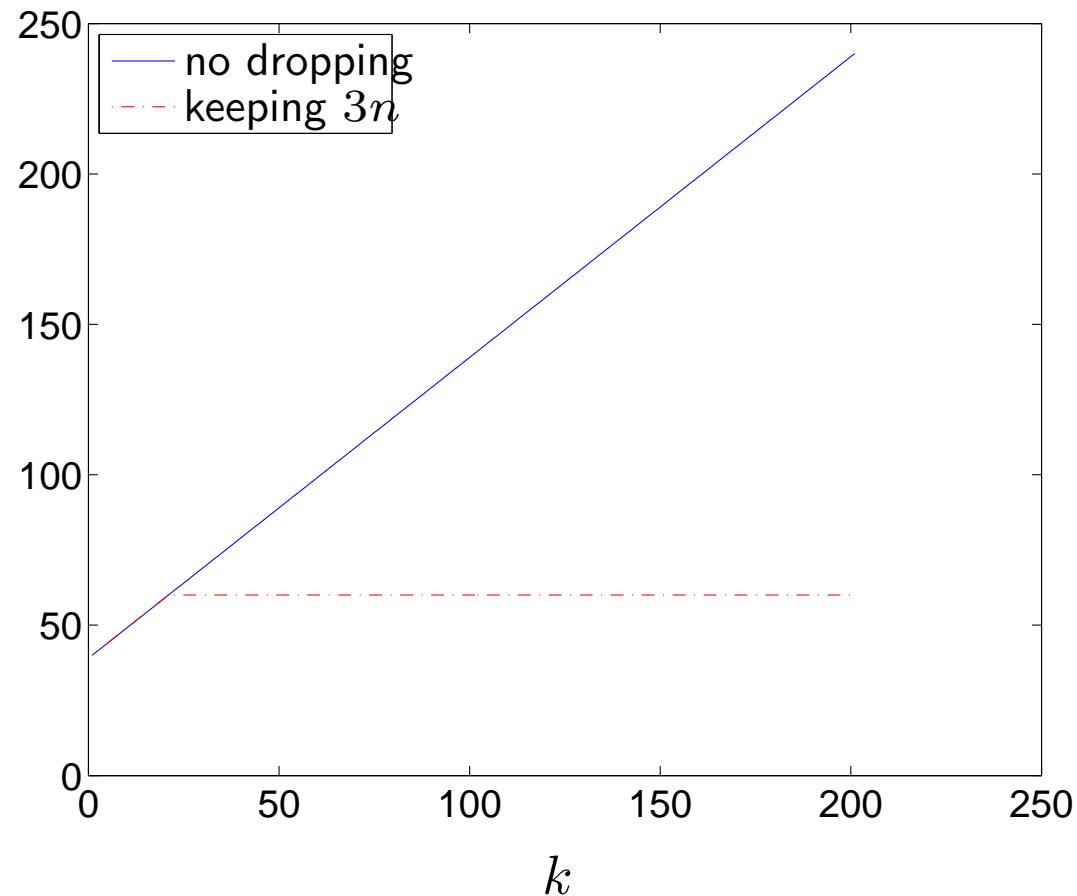


# ACCPM with constraint dropping

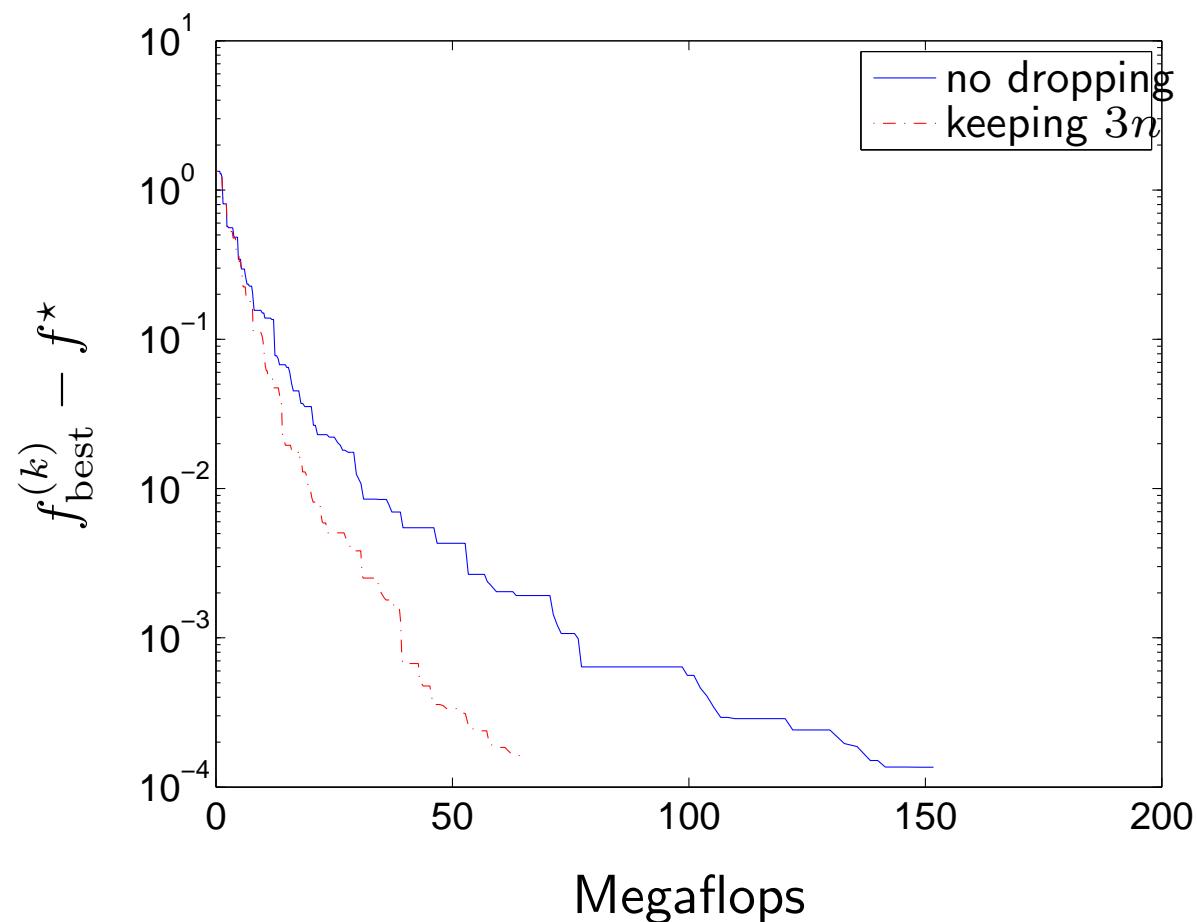
PWL objective,  $n = 20$  variables,  $m = 100$  terms



number of inequalities in  $\mathcal{P}$ :

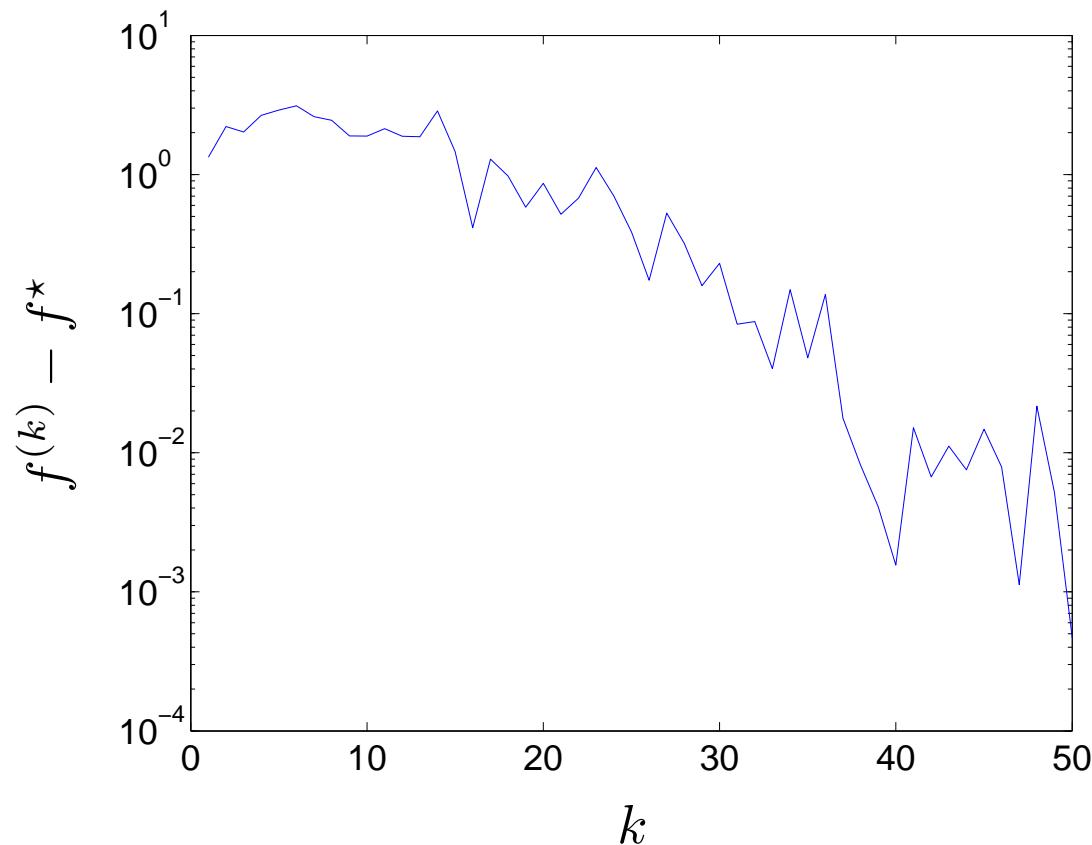


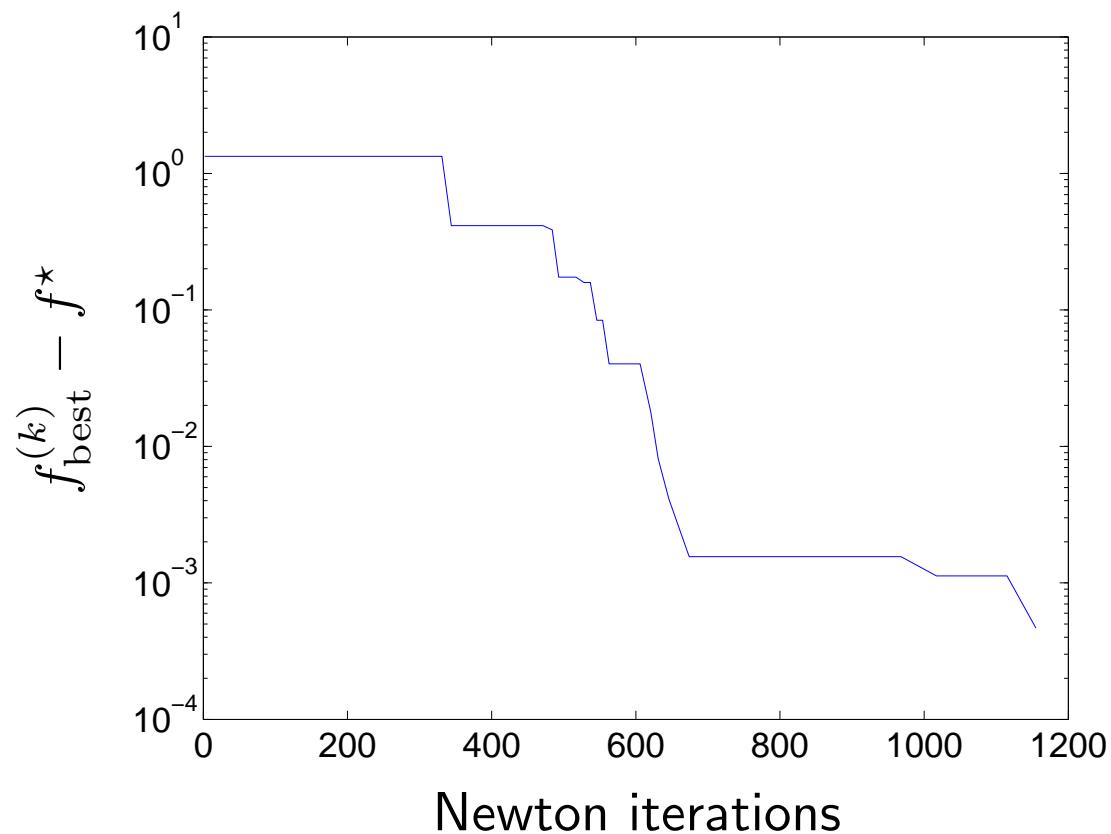
accuracy versus approximate cumulative flop count



# Epigraph ACCPM

PWL objective,  $n = 20$  variables,  $m = 100$  terms





# **Ellipsoid Method**

- ellipsoid method
- convergence proof
- inequality constraints
- feasibility problems

## Ellipsoid method

- developed by Shor, Nemirovsky, Yudin in 1970s
- used in 1979 by Khachian to show polynomial solvability of LPs
- each step requires cutting-plane or subgradient evaluation
- modest storage ( $O(n^2)$ )
- modest computation per step ( $O(n^2)$ ), via analytical formula
- efficient in theory; slow but steady in practice

# Motivation

in cutting-plane methods

- serious computation is needed to find next query point  
(typically  $O(n^2m)$ , with not small constant)
- localization polyhedron grows in complexity as algorithm progresses  
(we can, however, prune constraints to keep  $m$  proportional to  $n$ , e.g.,  
 $m = 4n$ )

ellipsoid method addresses both issues, but retains theoretical efficiency

# Ellipsoid algorithm for minimizing convex function

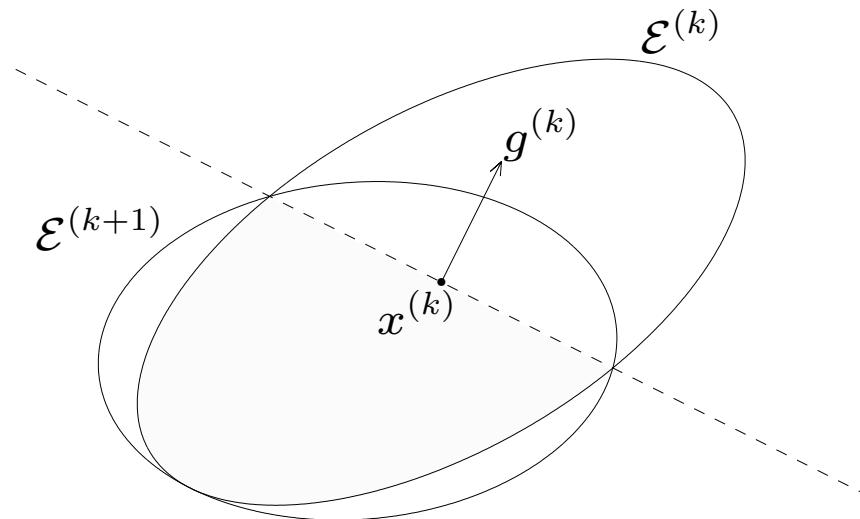
**idea:** localize  $x^*$  in an **ellipsoid** instead of a **polyhedron**

1. at iteration  $k$  we know  $x^* \in \mathcal{E}^{(k)}$
2. set  $x^{(k)} := \text{center}(\mathcal{E}^{(k)})$ ; evaluate  $g^{(k)} \in \partial f(x^{(k)})$   
 $(g^{(k)} = \nabla f(x^{(k)})$  if  $f$  is differentiable)
3. hence we know

$$x^* \in \mathcal{E}^{(k)} \cap \{z \mid g^{(k)T}(z - x^{(k)}) \leq 0\}$$

(a half-ellipsoid)

4. set  $\mathcal{E}^{(k+1)} :=$  minimum volume ellipsoid covering  
 $\mathcal{E}^{(k)} \cap \{z \mid g^{(k)T}(z - x^{(k)}) \leq 0\}$



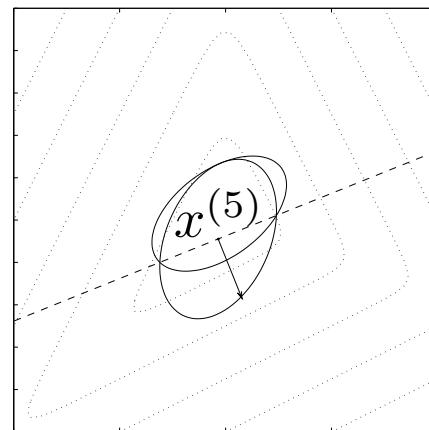
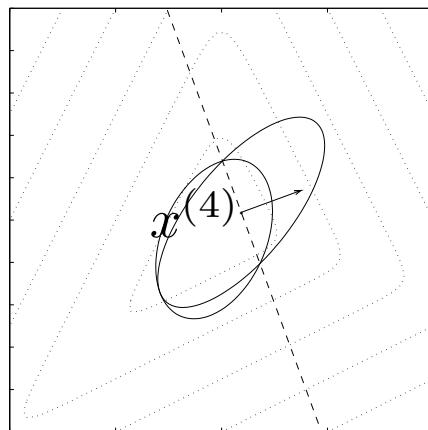
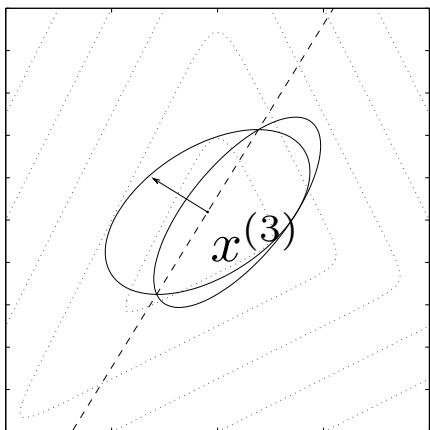
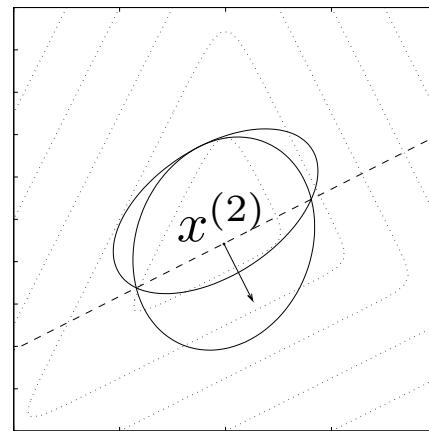
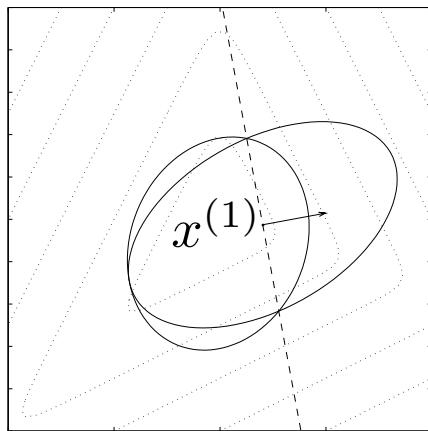
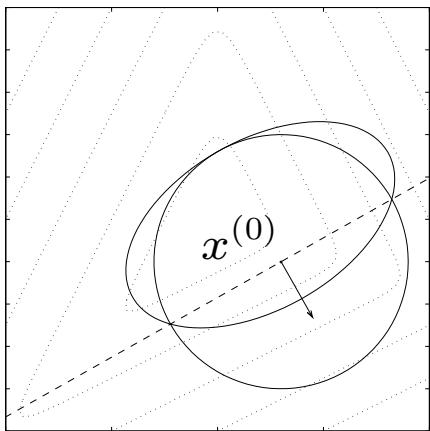
compared to cutting-plane methods:

- localization set doesn't grow more complicated
- easy to compute query point
- but, we add unnecessary points in step 4

## Properties of ellipsoid method

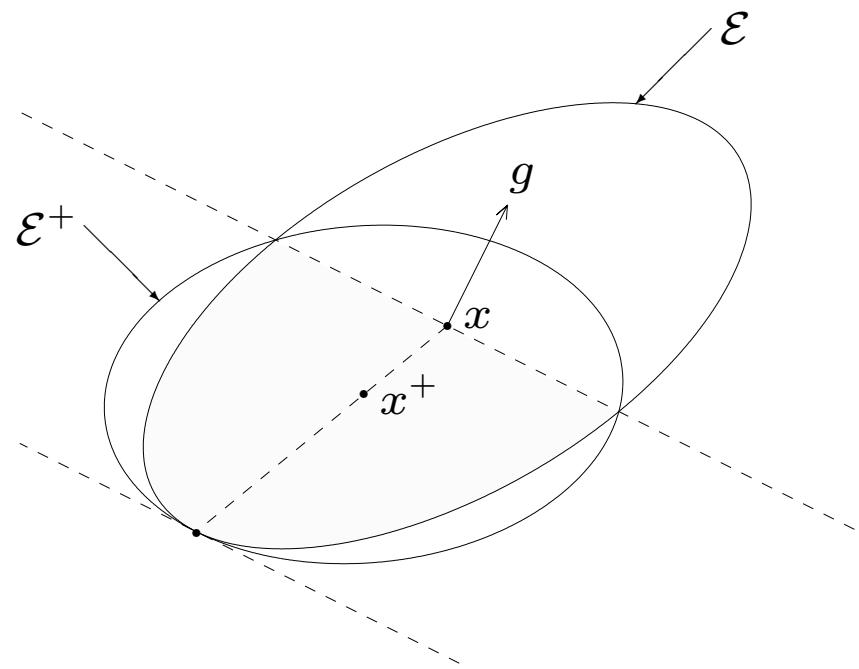
- reduces to bisection for  $n = 1$
- simple formula for  $\mathcal{E}^{(k+1)}$  given  $\mathcal{E}^{(k)}, g^{(k)}$
- $\mathcal{E}^{(k+1)}$  can be larger than  $\mathcal{E}^{(k)}$  in diameter (max semi-axis length), but is always smaller in volume
- $\text{vol}(\mathcal{E}^{(k+1)}) < e^{-\frac{1}{2n}} \text{vol}(\mathcal{E}^{(k)})$   
(volume reduction factor degrades rapidly with  $n$ , compared to CG or MVE cutting-plane methods)
- $\log \text{vol } \mathcal{E}^{(k+1)} \leq \log \text{vol } \mathcal{E}^{(k)} - 1/(2n)$   
(uncertainty in location of  $x^*$  decreases by a fixed number of bits each iteration)

# Example



# Updating the ellipsoid

$$\mathcal{E}(x, P) = \{z \mid (z - x)^T P^{-1} (z - x) \leq 1\}$$



(for  $n > 1$ ) minimum volume ellipsoid containing half-ellipsoid

$$\mathcal{E} \cap \{z \mid g^T(z - x) \leq 0\}$$

is given by

$$\begin{aligned}x^+ &= x - \frac{1}{n+1} P \tilde{g} \\P^+ &= \frac{n^2}{n^2 - 1} \left( P - \frac{2}{n+1} P \tilde{g} \tilde{g}^T P \right)\end{aligned}$$

where  $\tilde{g} = (1/\sqrt{g^T P g})g$

$P \tilde{g}$  is step from  $x$  to boundary of  $\mathcal{E}$

## Ellipsoid update — “Hessian” form

propagate  $H = P^{-1}$  instead of  $P$

$$\begin{aligned}x^+ &= x - \frac{1}{n+1} H^{-1} \tilde{g} \\H^+ &= \left(1 - \frac{1}{n^2}\right) \left(H + \frac{2}{n-1} \tilde{g} \tilde{g}^T\right)\end{aligned}$$

where  $\tilde{g} = (1/\sqrt{g^T H^{-1} g})g$

$H^{-1} \tilde{g}$  is step from  $x$  to boundary of  $\mathcal{E}$

## Simple stopping criterion

$$\begin{aligned} f(x^*) &\geq f(x^{(k)}) + g^{(k)T}(x^* - x^{(k)}) \\ &\geq f(x^{(k)}) + \inf_{z \in \mathcal{E}^{(k)}} g^{(k)T}(z - x^{(k)}) \\ &= f(x^{(k)}) - \sqrt{g^{(k)T} P^{(k)} g^{(k)}} \end{aligned}$$

second inequality holds since  $x^* \in \mathcal{E}_k$

simple stopping criterion:

$$\sqrt{g^{(k)T} P^{(k)} g^{(k)}} \leq \epsilon \implies f(x^{(k)}) - f(x^*) \leq \epsilon$$

## Basic ellipsoid algorithm

ellipsoid described as  $\mathcal{E}(x, P) = \{z \mid (z - x)^T P^{-1}(z - x) \leq 1\}$

**given** ellipsoid  $\mathcal{E}(x, P)$  containing  $x^*$ , accuracy  $\epsilon > 0$

repeat

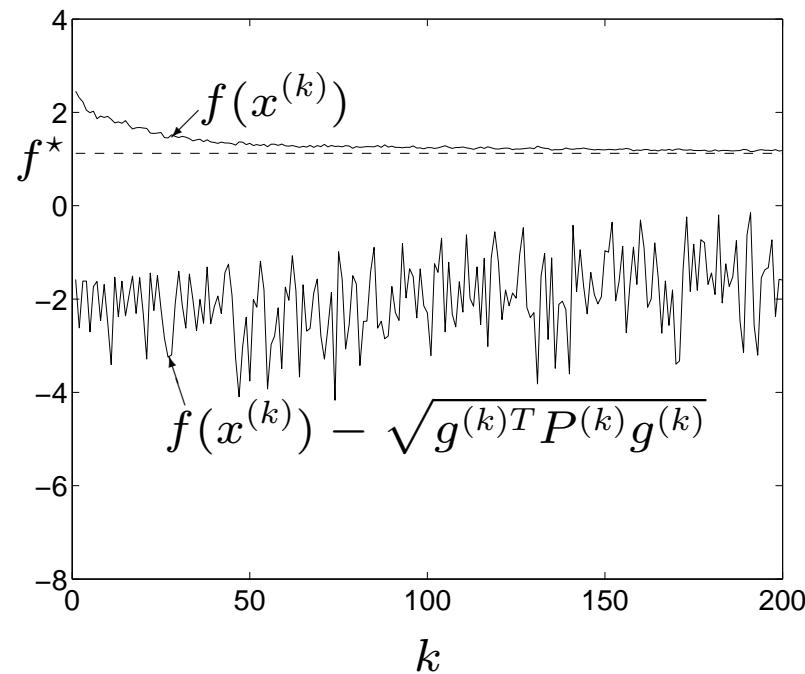
1. evaluate  $g \in \partial f(x)$
2. if  $\sqrt{g^T P g} \leq \epsilon$ , return( $x$ )
3. update ellipsoid
  - 3a.  $\tilde{g} := \frac{1}{\sqrt{g^T P g}} g$
  - 3b.  $x := x - \frac{1}{n+1} P \tilde{g}$
  - 3c.  $P := \frac{n^2}{n^2 - 1} \left( P - \frac{2}{n+1} P \tilde{g} \tilde{g}^T P \right)$

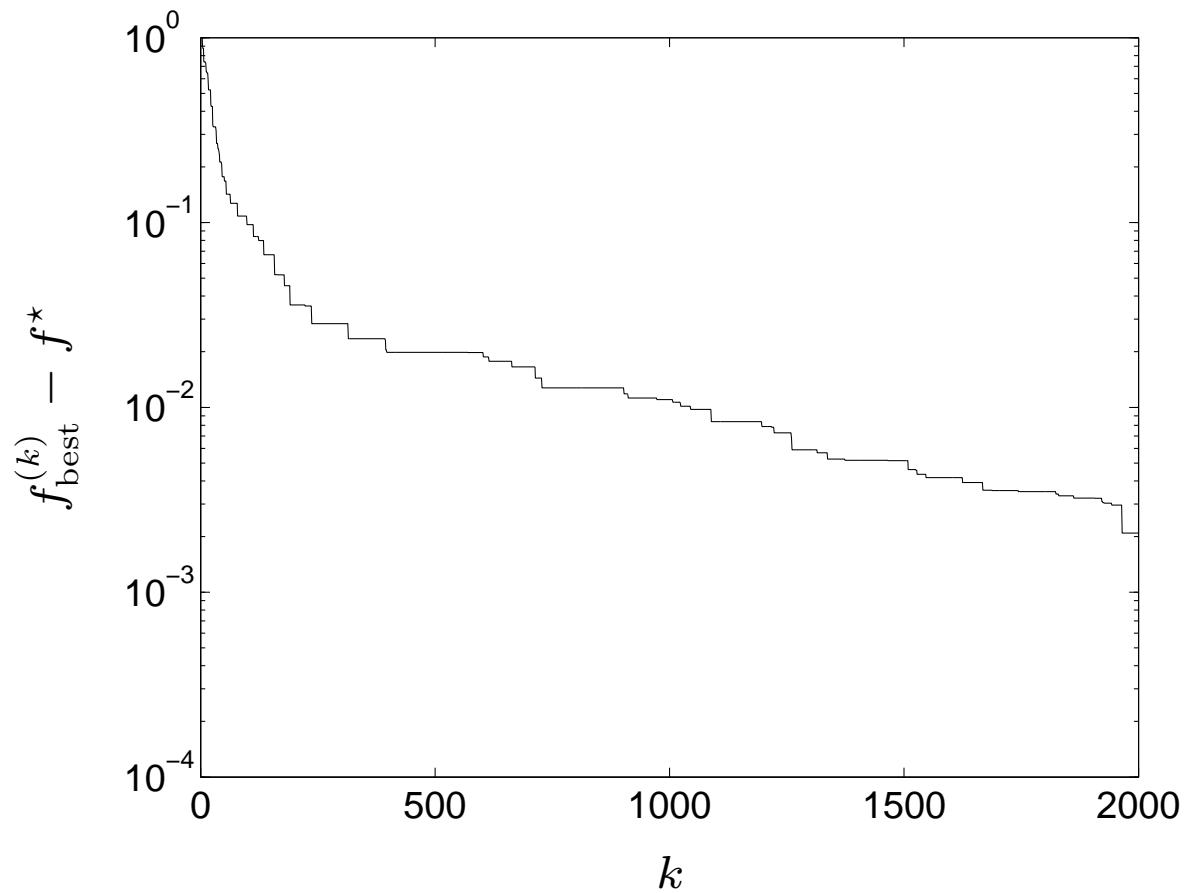
## Interpretation

- change coordinates so uncertainty is isotropic (same in all directions),  
*i.e.*,  $\mathcal{E}$  is unit ball
- take subgradient step with fixed length  $1/(n + 1)$
- Shor calls ellipsoid method ‘gradient method with space dilation in direction of gradient’ (which, strangely enough, didn’t catch on)

# Example

PWL function  $f(x) = \max_{i=1}^m (a_i^T x + b_i)$ , with  $n = 20, m = 100$





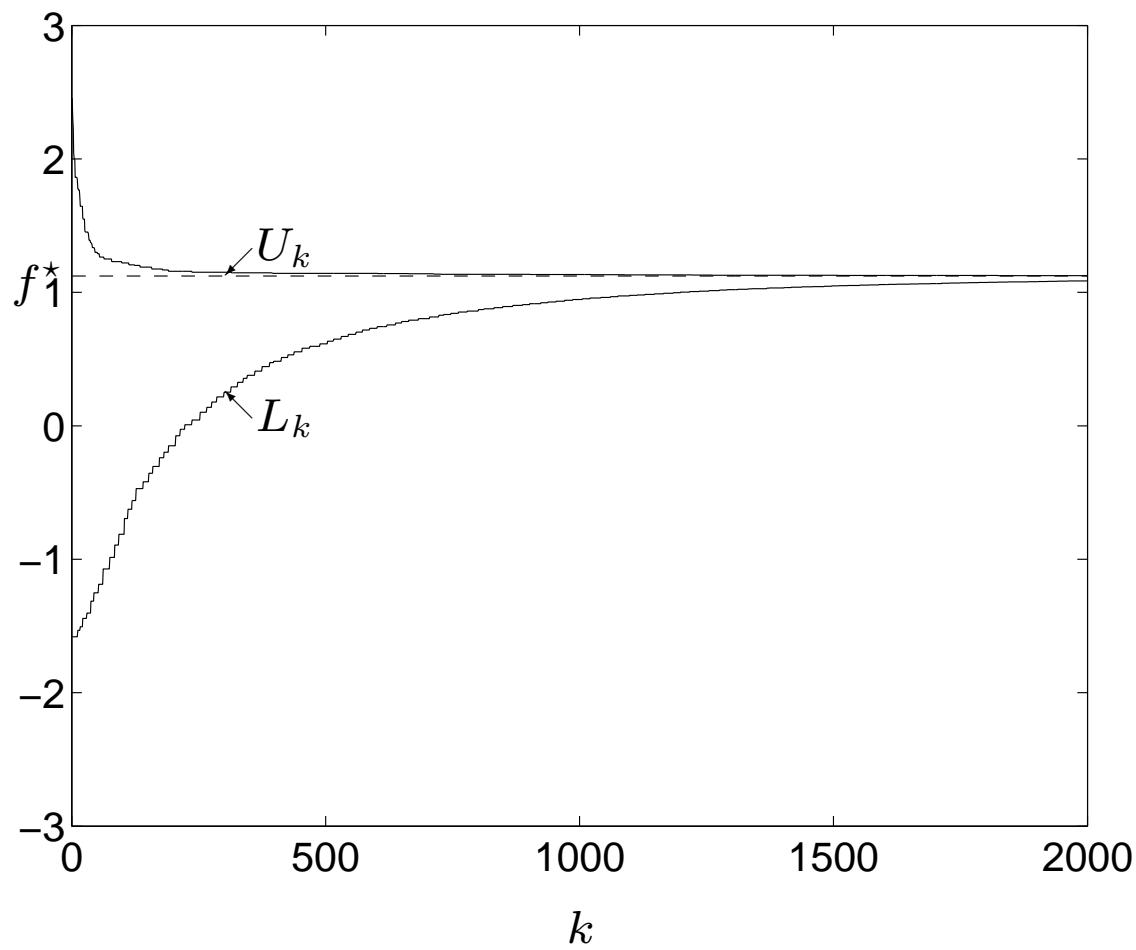
## Improvements

- keep track of best upper and lower bounds:

$$u_k = \min_{i=1,\dots,k} f(x^{(i)}), \quad l_k = \max_{i=1,\dots,k} \left( f(x^{(i)}) - \sqrt{g^{(i)T} P^{(i)} g^{(i)}} \right)$$

stop when  $u_k - l_k \leq \epsilon$

- can propagate Cholesky factor of  $P$   
(avoids problem of  $P \not\succ 0$  due to numerical roundoff)



# Proof of convergence

**assumptions:**

- $f$  is Lipschitz:  $|f(y) - f(x)| \leq G\|y - x\|$
- $\mathcal{E}^{(0)}$  is ball with radius  $R$

suppose  $f(x^{(i)}) > f^* + \epsilon$ ,  $i = 0, \dots, k$

then

$$f(x) \leq f^* + \epsilon \implies x \in \mathcal{E}^{(k)}$$

since at iteration  $i$  we only discard points with  $f \geq f(x^{(i)})$

from Lipschitz condition,

$$\|x - x^*\| \leq \epsilon/G \implies f(x) \leq f^* + \epsilon \implies x \in \mathcal{E}^{(k)}$$

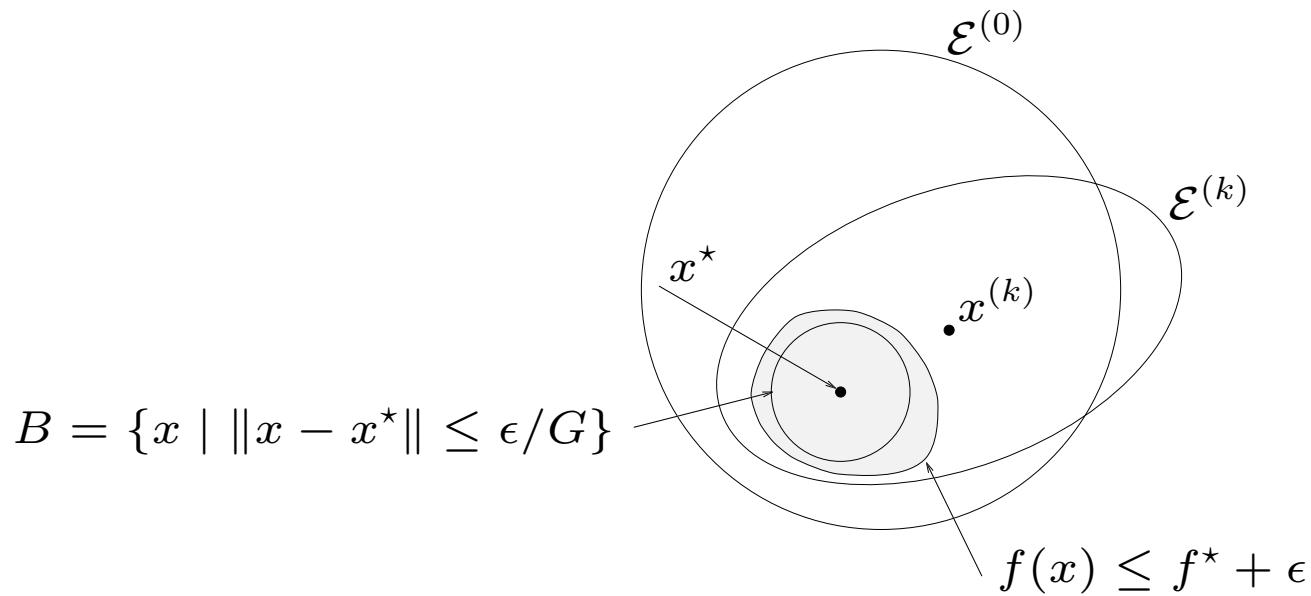
$$\text{so } B = \{x \mid \|x - x^*\| \leq \epsilon/G\} \subseteq \mathcal{E}^{(k)}$$

hence  $\mathbf{vol}(B) \leq \mathbf{vol}(\mathcal{E}^{(k)})$ , so

$$\alpha_n (\epsilon/G)^n \leq e^{-k/2n} \mathbf{vol}(\mathcal{E}^{(0)}) = e^{-k/2n} \alpha_n R^n$$

( $\alpha_n$  is volume of unit ball in  $\mathbf{R}^n$ )

therefore  $k \leq 2n^2 \log(RG/\epsilon)$



**conclusion:** for  $k > 2n^2 \log(RG/\epsilon)$ ,

$$\min_{i=0, \dots, k} f(x^{(i)}) \leq f^* + \epsilon$$

## Interpretation of complexity

since  $x^* \in \mathcal{E}_0 = \{x \mid \|x - x^{(0)}\| \leq R\}$ , our prior knowledge of  $f^*$  is

$$f^* \in [f(x^{(0)}) - GR, f(x^{(0)})]$$

our prior uncertainty in  $f^*$  is  $GR$

after  $k$  iterations our knowledge of  $f^*$  is

$$f^* \in \left[ \min_{i=0,\dots,k} f(x^{(i)}) - \epsilon, \min_{i=0,\dots,k} f(x^{(i)}) \right]$$

posterior uncertainty in  $f^*$  is  $\leq \epsilon$

iterations required:

$$2n^2 \log \frac{RG}{\epsilon} = 2n^2 \log \frac{\text{prior uncertainty}}{\text{posterior uncertainty}}$$

efficiency:  $0.72/n^2$  bits per subgradient evaluation

## Deep cut ellipsoid method

minimum volume ellipsoid containing ellipsoid intersected with halfspace

$$\mathcal{E} \cap \{z \mid g^T(z - x) + h \leq 0\}$$

with  $h \geq 0$ , is given by

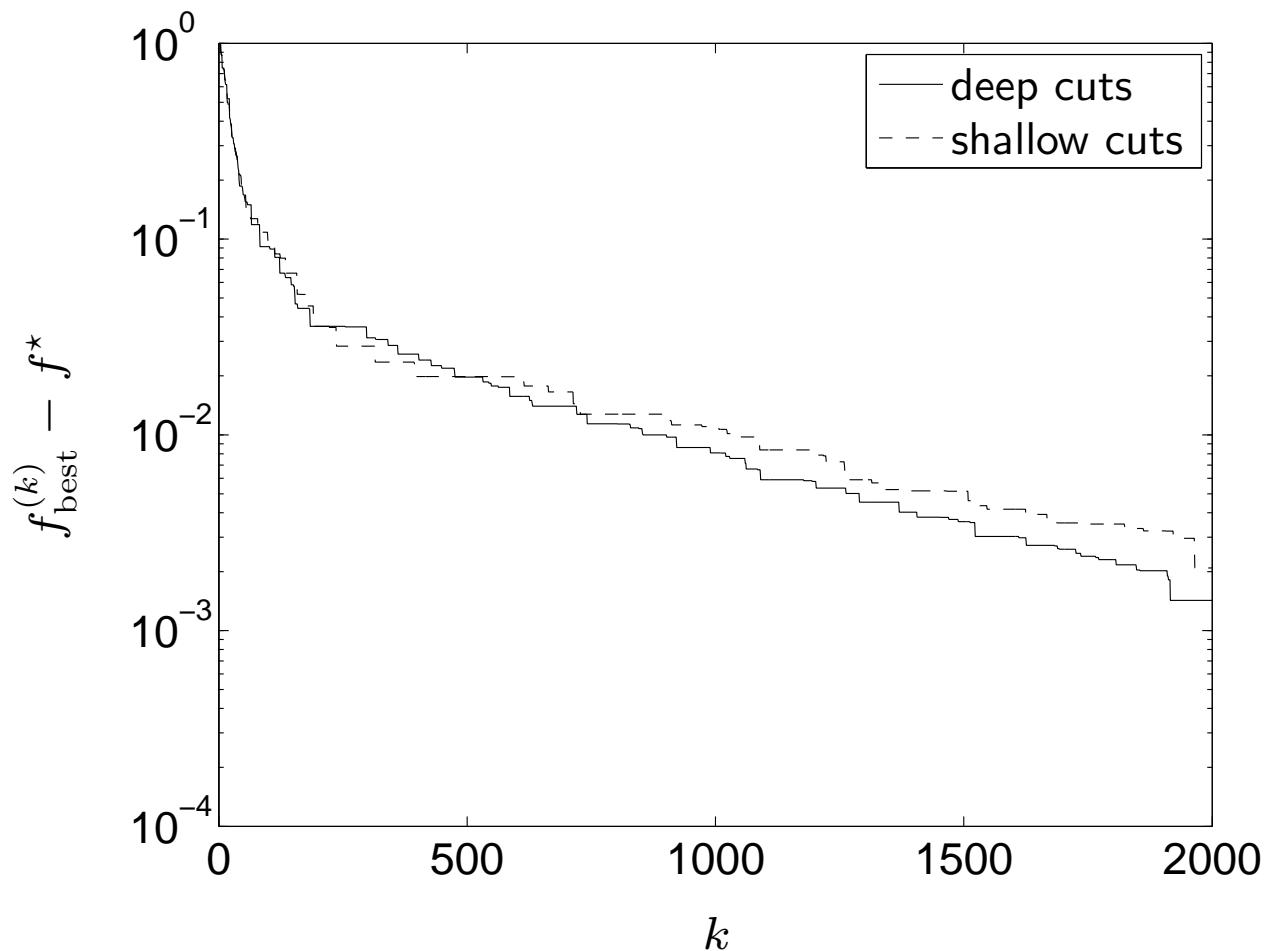
$$\begin{aligned}x^+ &= x - \frac{1 + \alpha n}{n + 1} P \tilde{g} \\P^+ &= \frac{n^2(1 - \alpha^2)}{n^2 - 1} \left( P - \frac{2(1 + \alpha n)}{(n + 1)(1 + \alpha)} P \tilde{g} \tilde{g}^T P \right)\end{aligned}$$

where

$$\tilde{g} = \frac{g}{\sqrt{g^T P g}}, \quad \alpha = \frac{h}{\sqrt{g^T P g}}$$

(if  $\alpha > 1$ , intersection is empty)

## Ellipsoid method with deep objective cuts



## Inequality constrained problems

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

- if  $x^{(k)}$  feasible, update ellipsoid with objective cut

$$g_0^T(z - x^{(k)}) + f_0(x^{(k)}) - f_{\text{best}}^{(k)} \leq 0, \quad g_0 \in \partial f_0(x^{(k)})$$

$f_{\text{best}}^{(k)}$  is best objective value of feasible iterates so far

- if  $x^{(k)}$  infeasible, update ellipsoid with feasibility cut

$$g_j^T(z - x^{(k)}) + f_j(x^{(k)}) \leq 0, \quad g_j \in \partial f_j(x^{(k)})$$

assuming  $f_j(x^{(k)}) > 0$

## Stopping criterion

if  $x^{(k)}$  is feasible, we have lower bound on  $p^*$  as before:

$$p^* \geq f_0(x^{(k)}) - \sqrt{g_0^{(k)T} P^{(k)} g_0^{(k)}}$$

if  $x^{(k)}$  is infeasible, we have for all  $x \in \mathcal{E}^{(k)}$

$$\begin{aligned} f_j(x) &\geq f_j(x^{(k)}) + g_j^{(k)T} (x - x^{(k)}) \\ &\geq f_j(x^{(k)}) + \inf_{z \in \mathcal{E}^{(k)}} g^{(k)T} (z - x^{(k)}) \\ &= f_j(x^{(k)}) - \sqrt{g_j^{(k)T} P^{(k)} g_j^{(k)}} \end{aligned}$$

hence, problem is infeasible if for some  $j$ ,

$$f_j(x^{(k)}) - \sqrt{g_j^{(k)T} P^{(k)} g_j^{(k)}} > 0$$

**stopping criteria:**

- if  $x^{(k)}$  is feasible and  $\sqrt{g_0^{(k)T} P^{(k)} g_0^{(k)}} \leq \epsilon$  ( $x^{(k)}$  is  $\epsilon$ -suboptimal)
- if  $f_j(x^{(k)}) - \sqrt{g_j^{(k)T} P^{(k)} g_j^{(k)}} > 0$  (problem is infeasible)

# Decomposition Methods

- separable problems, complicating variables
- primal decomposition
- dual decomposition
- complicating constraints
- general decomposition structures

## Separable problem

$$\begin{aligned} & \text{minimize} && f_1(x_1) + f_2(x_2) \\ & \text{subject to} && x_1 \in \mathcal{C}_1, \quad x_2 \in \mathcal{C}_2 \end{aligned}$$

- we can solve for  $x_1$  and  $x_2$  separately (in parallel)
- even if they are solved sequentially, this gives advantage if computational effort is superlinear in problem size
- called **separable** or **trivially parallelizable**
- generalizes to any objective of form  $\Psi(f_1, f_2)$  with  $\Psi$  nondecreasing (e.g., max)

# Complicating variable

consider unconstrained problem,

$$\text{minimize } f(x) = f_1(x_1, y) + f_2(x_2, y)$$

$$x = (x_1, x_2, y)$$

- $y$  is the **complicating variable** or **coupling variable**; when it is fixed the problem is separable in  $x_1$  and  $x_2$
- $x_1, x_2$  are **private** or **local** variables;  $y$  is a **public** or **interface** or **boundary** variable between the two subproblems

# Primal decomposition

fix  $y$  and define

$$\begin{aligned} \text{subproblem 1: } & \text{minimize}_{x_1} f_1(x_1, y) \\ \text{subproblem 2: } & \text{minimize}_{x_2} f_2(x_2, y) \end{aligned}$$

with optimal values  $\phi_1(y)$  and  $\phi_2(y)$

original problem is equivalent to **master problem**

$$\text{minimize}_y \quad \phi_1(y) + \phi_2(y)$$

with variable  $y$

called **primal decomposition** since master problem manipulates primal (complicating) variables

- if original problem is convex, so is master problem
- can solve master problem using
  - bisection (if  $y$  is scalar)
  - gradient or Newton method (if  $\phi_i$  differentiable)
  - subgradient, cutting-plane, or ellipsoid method
- each iteration of master problem requires solving the two subproblems (in parallel)
- if master algorithm converges fast enough and subproblems are sufficiently easier to solve than original problem, we get savings

# Primal decomposition algorithm

(using subgradient algorithm for master)

**repeat**

1. Solve the subproblems (in parallel).

Find  $x_1$  that minimizes  $f_1(x_1, y)$ , and a subgradient  $g_1 \in \partial\phi_1(y)$ .

Find  $x_2$  that minimizes  $f_2(x_2, y)$ , and a subgradient  $g_2 \in \partial\phi_2(y)$ .

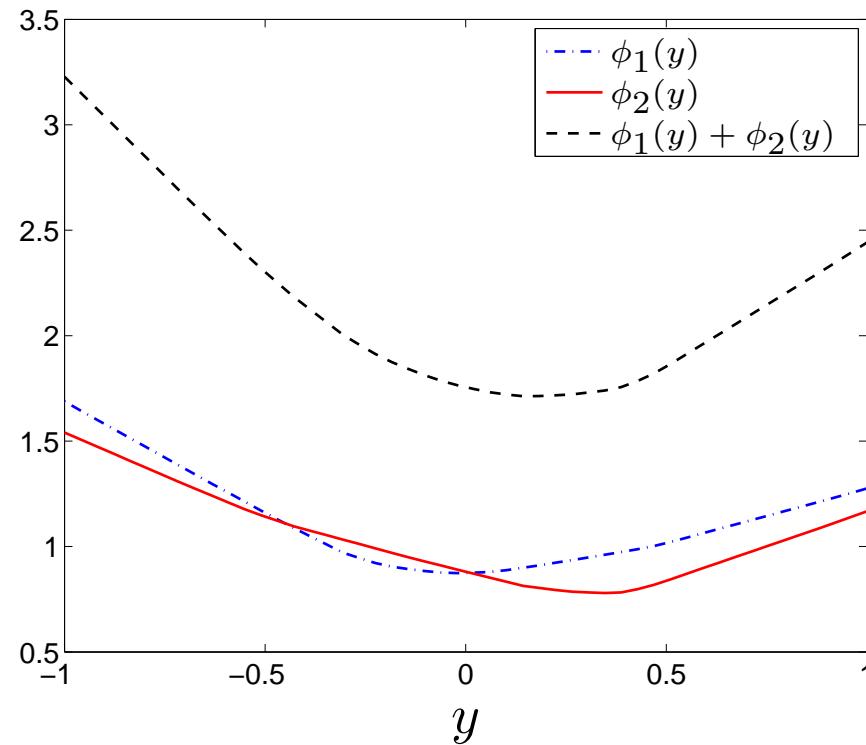
2. Update complicating variable.

$$y := y - \alpha_k(g_1 + g_2).$$

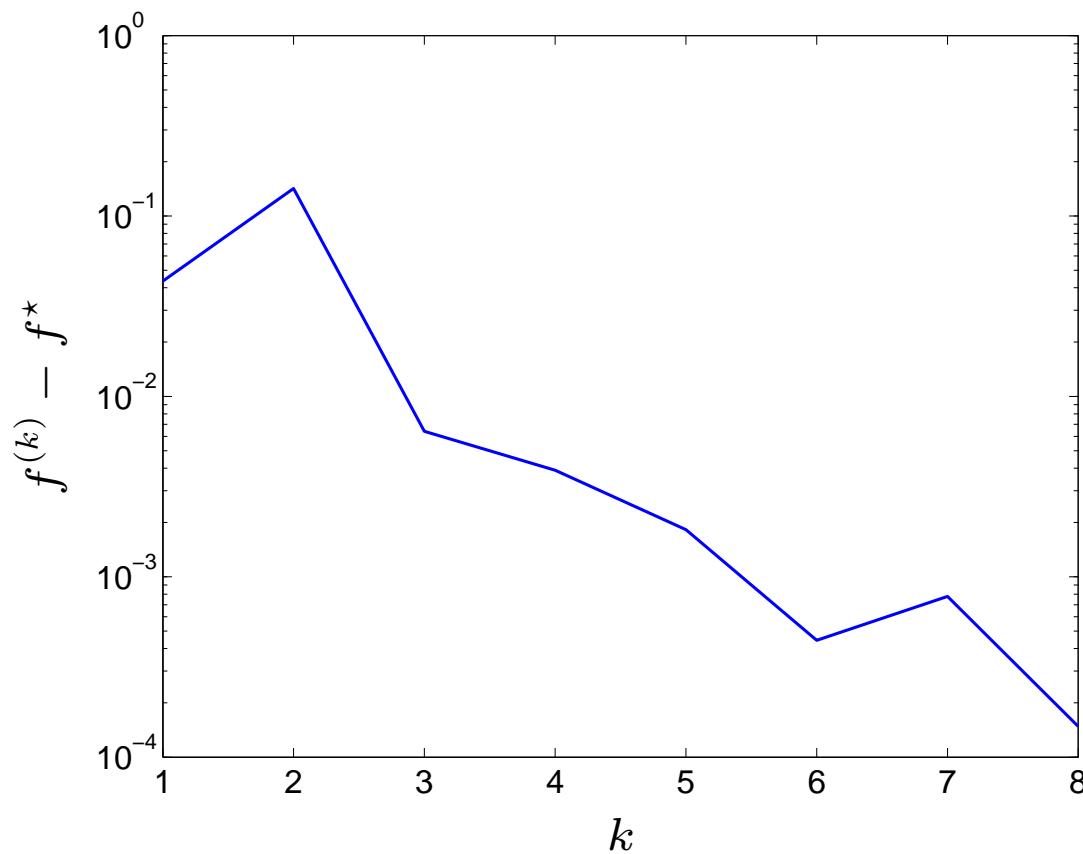
step length  $\alpha_k$  can be chosen in any of the standard ways

# Example

- $x_1, x_2 \in \mathbf{R}^{20}, y \in \mathbf{R}$
- $f_i$  are PWL (max of 100 affine functions each);  $f^* \approx 1.71$



primal decomposition, using bisection on  $y$



## Dual decomposition

Step 1: introduce new variables  $y_1, y_2$

$$\begin{aligned} \text{minimize} \quad & f(x) = f_1(x_1, y_1) + f_2(x_2, y_2) \\ \text{subject to} \quad & y_1 = y_2 \end{aligned}$$

- $y_1, y_2$  are **local** versions of complicating variable  $y$
- $y_1 = y_2$  is **consensus constraint**

Step 2: form dual problem

$$L(x_1, y_1, x_2, y_2) = f_1(x_1, y_1) + f_2(x_2, y_2) + \nu^T(y_1 - y_2)$$

**separable**; can minimize over  $(x_1, y_1)$  and  $(x_2, y_2)$  separately

$$g_1(\nu) = \inf_{x_1, y_1} (f_1(x_1, y_1) + \nu^T y_1) = -f_1^*(0, -\nu)$$

$$g_2(\nu) = \inf_{x_2, y_2} (f_2(x_2, y_2) - \nu^T y_2) = -f_2^*(0, \nu)$$

dual problem is: maximize  $g(\nu) = g_1(\nu) + g_2(\nu)$

- computing  $g_i(\nu)$  are the **dual subproblems**
- can be done in parallel
- a subgradient of  $-g$  is  $y_2 - y_1$  (from solutions of subproblems)

# Dual decomposition algorithm

(using subgradient algorithm for master)

**repeat**

1. Solve the dual subproblems (in parallel).  
Find  $x_1, y_1$  that minimize  $f_1(x_1, y_1) + \nu^T y_1$ .  
Find  $x_2, y_2$  that minimize  $f_2(x_2, y_2) - \nu^T y_2$ .
2. Update dual variables (prices).  
 $\nu := \nu - \alpha_k(y_2 - y_1)$ .

- step length  $\alpha_k$  can be chosen in standard ways
- at each step we have a lower bound  $g(\nu)$  on  $p^*$
- iterates are generally infeasible, *i.e.*,  $y_1 \neq y_2$

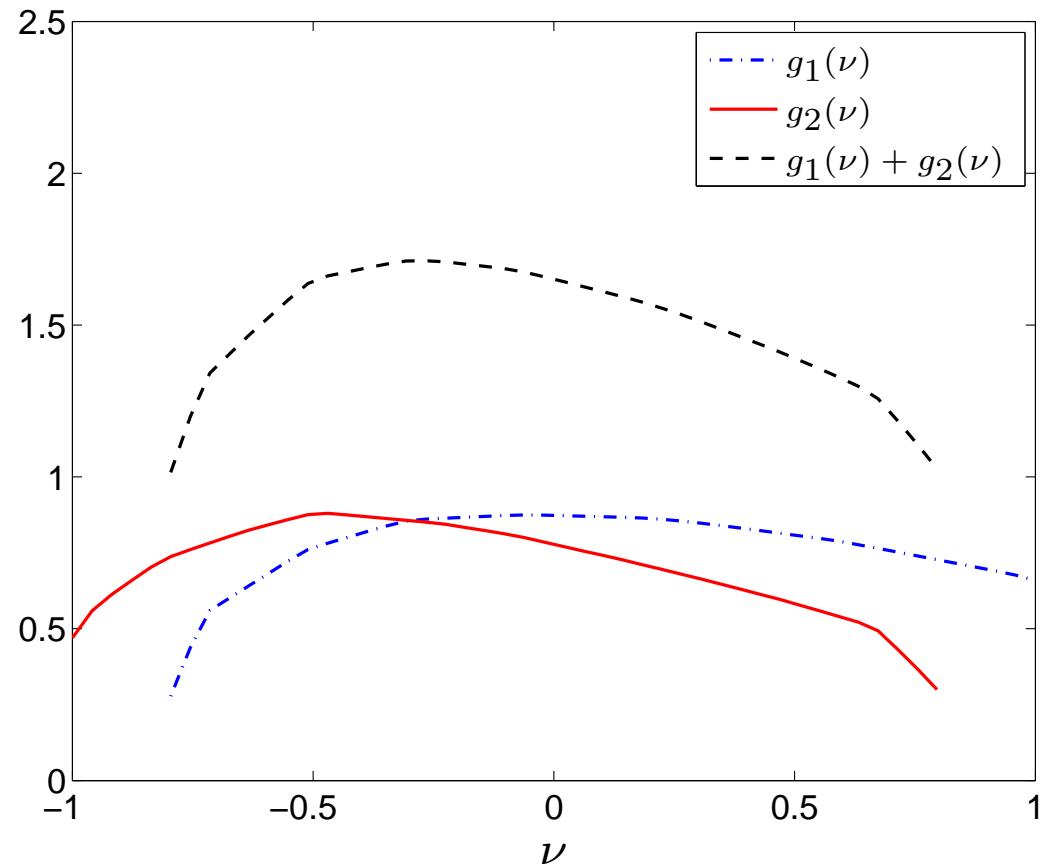
## Finding feasible iterates

- reasonable guess of feasible point from  $(x_1, y_1), (x_2, y_2)$ :

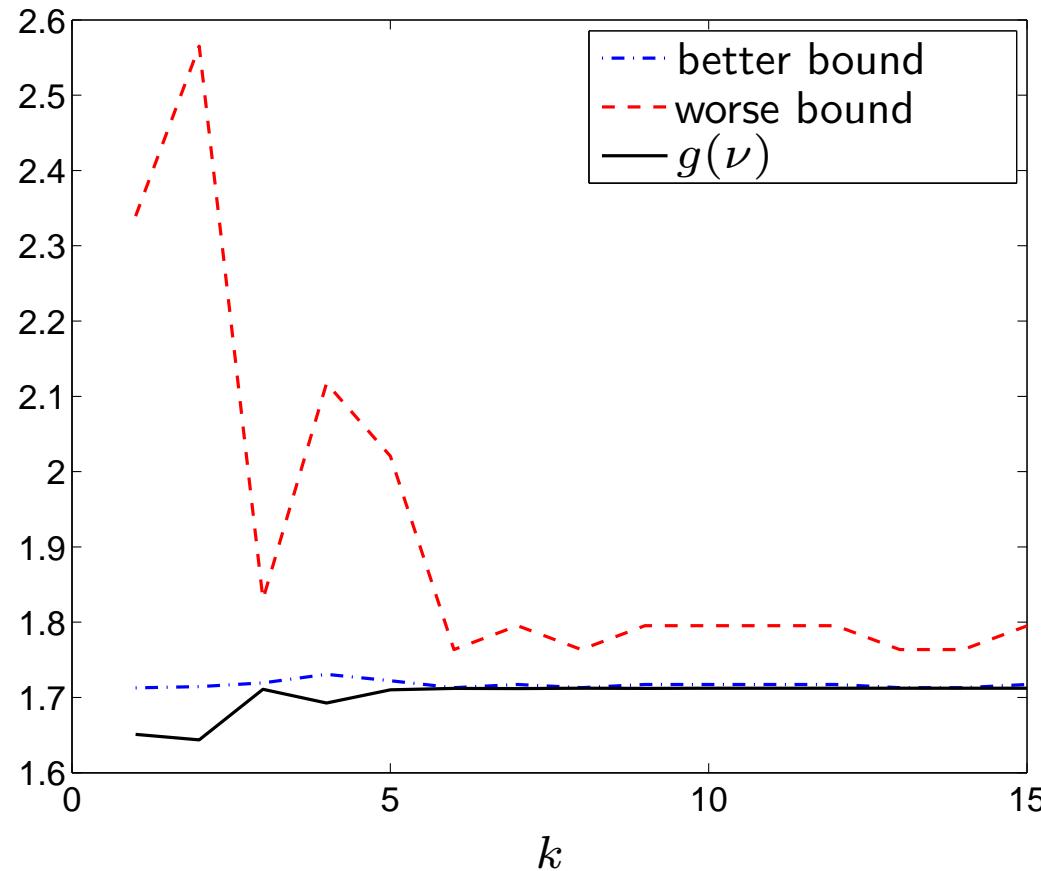
$$(x_1, \bar{y}), \quad (x_2, \bar{y}), \quad \bar{y} = (y_1 + y_2)/2$$

- projection onto feasible set  $y_1 = y_2$
- gives upper bound  $p^* \leq f_1(x_1, \bar{y}) + f_2(x_2, \bar{y})$
- a better feasible point: replace  $y_1, y_2$  with  $\bar{y}$  and solve *primal* subproblems  $\text{minimize}_{x_1} f_1(x_1, \bar{y})$ ,  $\text{minimize}_{x_2} f_2(x_2, \bar{y})$
- gives (better) upper bound  $p^* \leq \phi_1(\bar{y}) + \phi_2(\bar{y})$

## (Same) example



dual decomposition convergence (using bisection on  $\nu$ )



## Interpretation

- $y_1$  is resources consumed by first unit,  $y_2$  is resources generated by second unit
- $y_1 = y_2$  is **consistency** condition: supply equals demand
- $\nu$  is a set of resource prices
- master algorithm adjusts prices at each step, rather than allocating resources directly (primal decomposition)

## Recovering the primal solution from the dual

- iterates in dual decomposition:

$$\nu^{(k)}, \quad (x_1^{(k)}, y_1^{(k)}), \quad (x_2^{(k)}, y_2^{(k)})$$

- $x_1^{(k)}, y_1^{(k)}$  is minimizer of  $f_1(x_1, y_1) + \nu^{(k)T} y_1$  found in subproblem 1
- $x_2^{(k)}, y_2^{(k)}$  is minimizer of  $f_2(x_2, y_2) - \nu^{(k)T} y_2$  found in subproblem 2
- $\nu^{(k)} \rightarrow \nu^*$  (*i.e.*, we have price convergence)
- subtlety: we need not have  $y_1^{(k)} - y_2^{(k)} \rightarrow 0$
- the hammer: if  $f_i$  strictly convex, we have  $y_1^{(k)} - y_2^{(k)} \rightarrow 0$
- can fix allocation (*i.e.*, compute  $\phi_i$ ), or add regularization terms  $\epsilon \|y_i\|^2$

## Decomposition with constraints

can also have **complicating constraints**, as in

$$\begin{aligned} & \text{minimize} && f_1(x_1) + f_2(x_2) \\ & \text{subject to} && x_1 \in \mathcal{C}_1, \quad x_2 \in \mathcal{C}_2 \\ & && h_1(x_1) + h_2(x_2) \preceq 0 \end{aligned}$$

- $f_i, h_i, \mathcal{C}_i$  convex
- $h_1(x_1) + h_2(x_2) \preceq 0$  is a set of  $p$  complicating or coupling constraints, involving both  $x_1$  and  $x_2$
- can interpret coupling constraints as limits on resources shared between two subproblems

## Primal decomposition

fix  $t \in \mathbf{R}^p$  and define

subproblem 1:

$$\begin{aligned} & \text{minimize} && f_1(x_1) \\ & \text{subject to} && x_1 \in \mathcal{C}_1, \quad h_1(x_1) \preceq t \end{aligned}$$

subproblem 2:

$$\begin{aligned} & \text{minimize} && f_2(x_2) \\ & \text{subject to} && x_2 \in \mathcal{C}_2, \quad h_2(x_2) \preceq -t \end{aligned}$$

- $t$  is the quantity of resources allocated to first subproblem ( $-t$  is allocated to second subproblem)
- master problem: minimize  $\phi_1(t) + \phi_2(t)$  (optimal values of subproblems) over  $t$
- subproblems can be solved separately (in parallel) when  $t$  is fixed

# Primal decomposition algorithm

**repeat**

1. Solve the subproblems (in parallel).

Solve subproblem 1, finding  $x_1$  and  $\lambda_1$ .

Solve subproblem 2, finding  $x_2$  and  $\lambda_2$ .

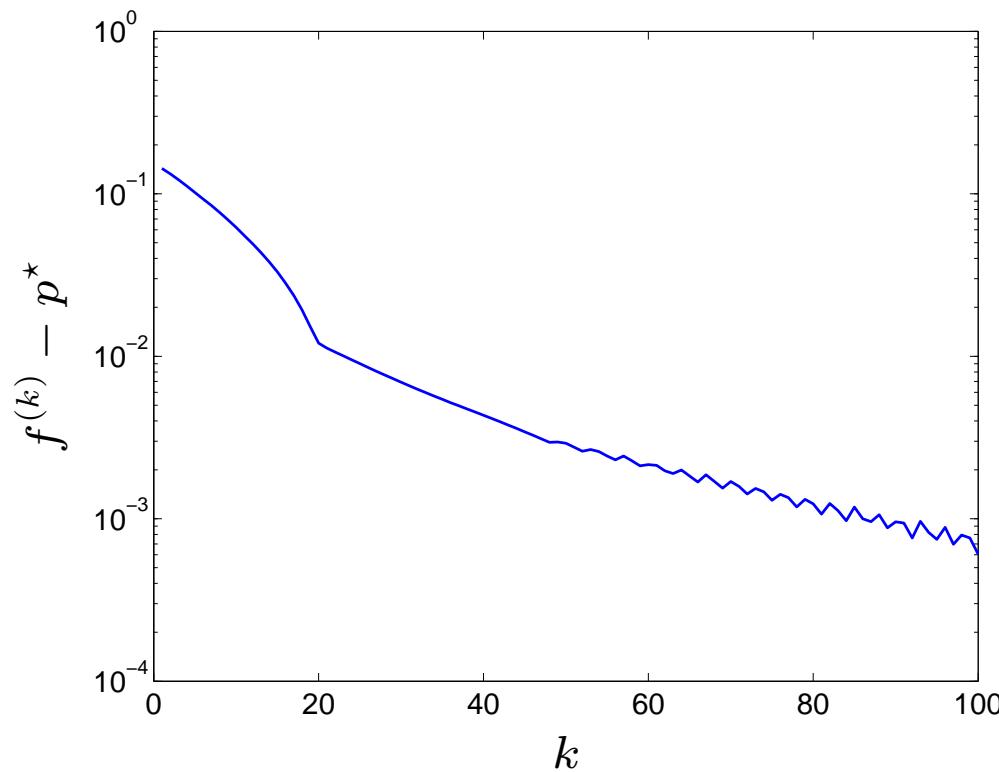
2. Update resource allocation.

$$t := t - \alpha_k(\lambda_2 - \lambda_1).$$

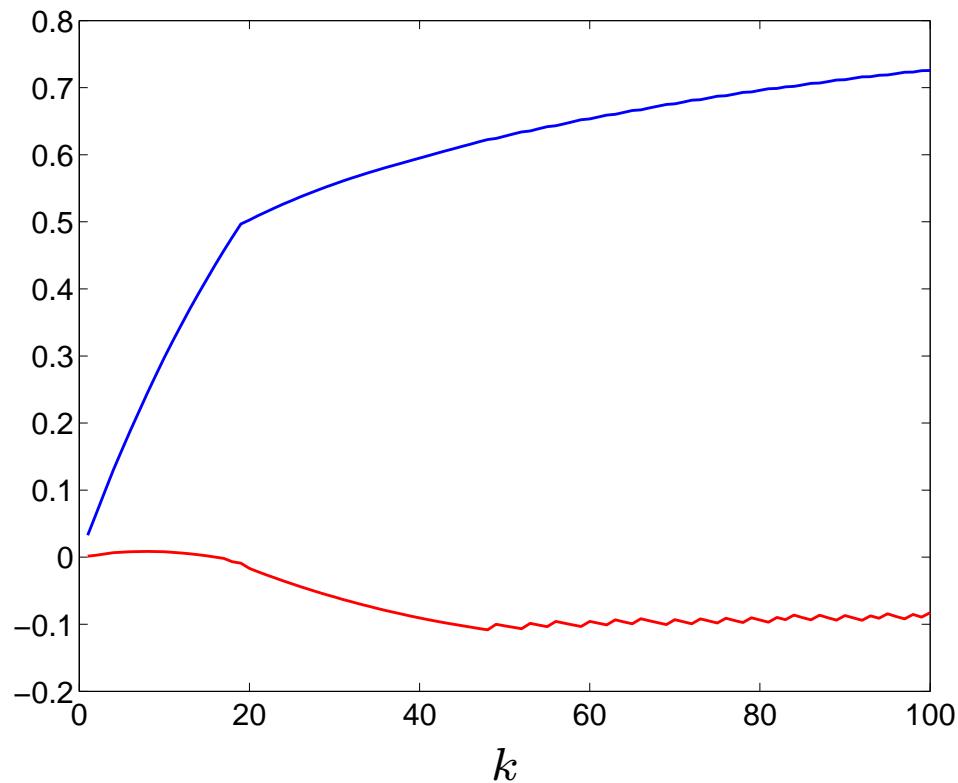
- $\lambda_i$  is an optimal Lagrange multiplier associated with resource constraint in subproblem  $i$
- $\lambda_2 - \lambda_1 \in \partial(\phi_1 + \phi_2)(t)$
- $\alpha_k$  is an appropriate step size
- all iterates are feasible (when subproblems are feasible)

## Example

- $x_1, x_2 \in \mathbf{R}^{20}$ ,  $t \in \mathbf{R}^2$ ;  $f_i$  are quadratic,  $h_i$  are affine,  $\mathcal{C}_i$  are polyhedra defined by 100 inequalities;  $p^* \approx -1.33$ ;  $\alpha_k = 0.5/k$



resource allocation  $t$  to first subsystem (second subsystem gets  $-t$ )



## Dual decomposition

form (separable) partial Lagrangian

$$\begin{aligned} L(x_1, x_2, \lambda) &= f_1(x_1) + f_2(x_2) + \lambda^T(h_1(x_1) + h_2(x_2)) \\ &= (f_1(x_1) + \lambda^T h_1(x_1)) + (f_2(x_2) + \lambda^T h_2(x_2)) \end{aligned}$$

fix dual variable  $\lambda$  and define

subproblem 1:

$$\begin{array}{ll} \text{minimize} & f_1(x_1) + \lambda^T h_1(x_1) \\ \text{subject to} & x_1 \in \mathcal{C}_1 \end{array}$$

subproblem 2:

$$\begin{array}{ll} \text{minimize} & f_2(x_2) + \lambda^T h_2(x_2) \\ \text{subject to} & x_2 \in \mathcal{C}_2 \end{array}$$

with optimal values  $g_1(\lambda)$ ,  $g_2(\lambda)$

- $-h_i(\bar{x}_i) \in \partial(-g_i)(\lambda)$ , where  $\bar{x}_i$  is any solution to subproblem  $i$
- $-h_1(\bar{x}_1) - h_2(\bar{x}_2) \in \partial(-g)(\lambda)$
- the master algorithm updates  $\lambda$  using this subgradient

# Dual decomposition algorithm

(using projected subgradient method)

**repeat**

1. Solve the subproblems (in parallel).

Solve subproblem 1, finding an optimal  $\bar{x}_1$ .

Solve subproblem 2, finding an optimal  $\bar{x}_2$ .

2. Update dual variables (prices).

$$\lambda := (\lambda + \alpha_k(h_1(\bar{x}_1) + h_2(\bar{x}_2)))_+$$

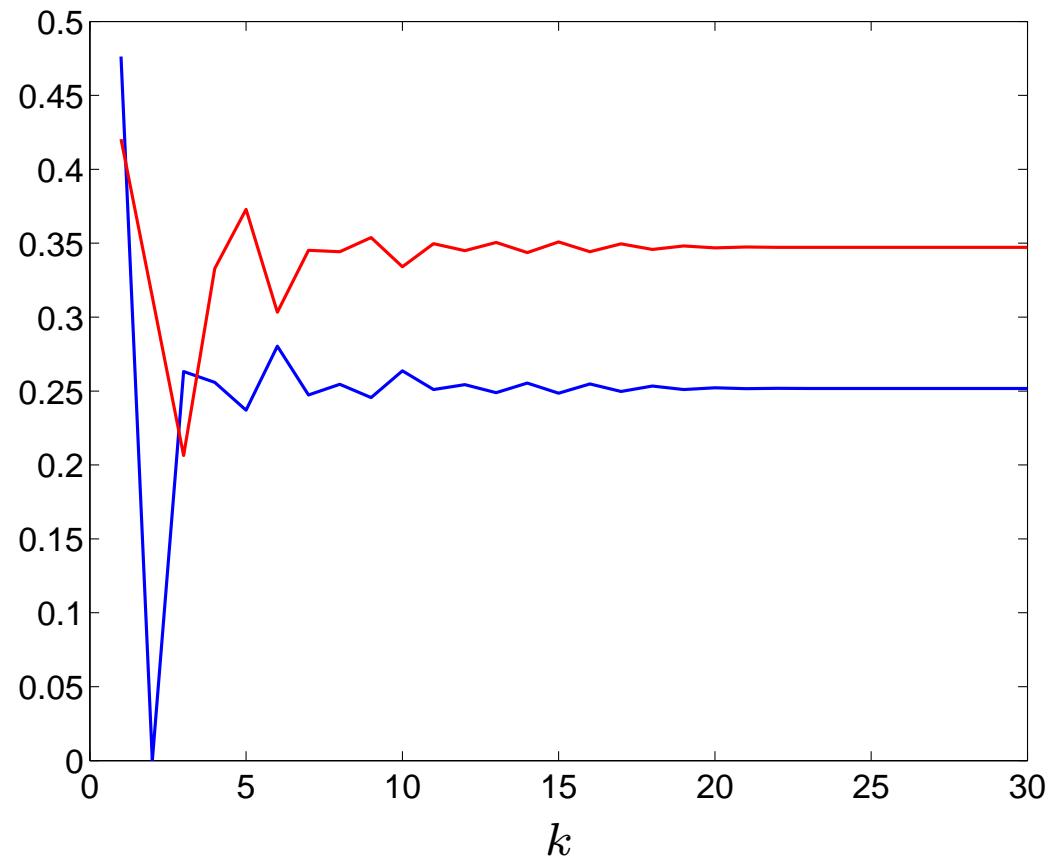
- $\alpha_k$  is an appropriate step size
- iterates need not be feasible
- can again construct feasible primal variables using projection

## Interpretation

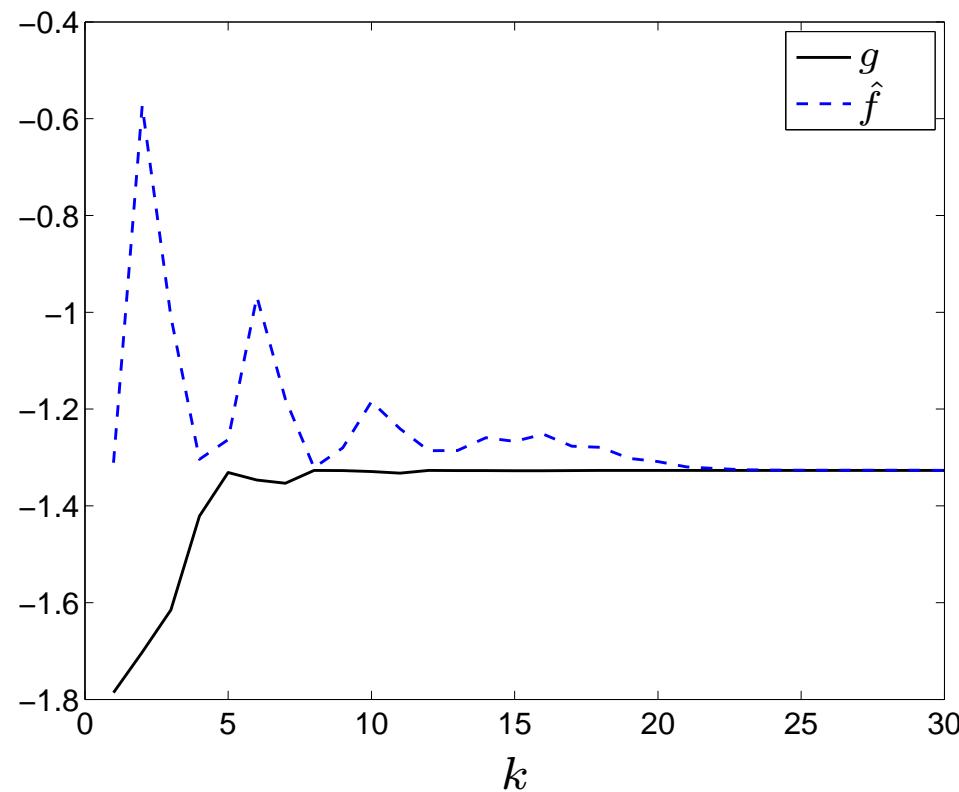
- $\lambda$  gives prices of resources
- subproblems are solved separately, taking income/expense from resource usage into account
- master algorithm adjusts prices
- prices on over-subscribed resources are increased; prices on undersubscribed resources are reduced, but never made negative

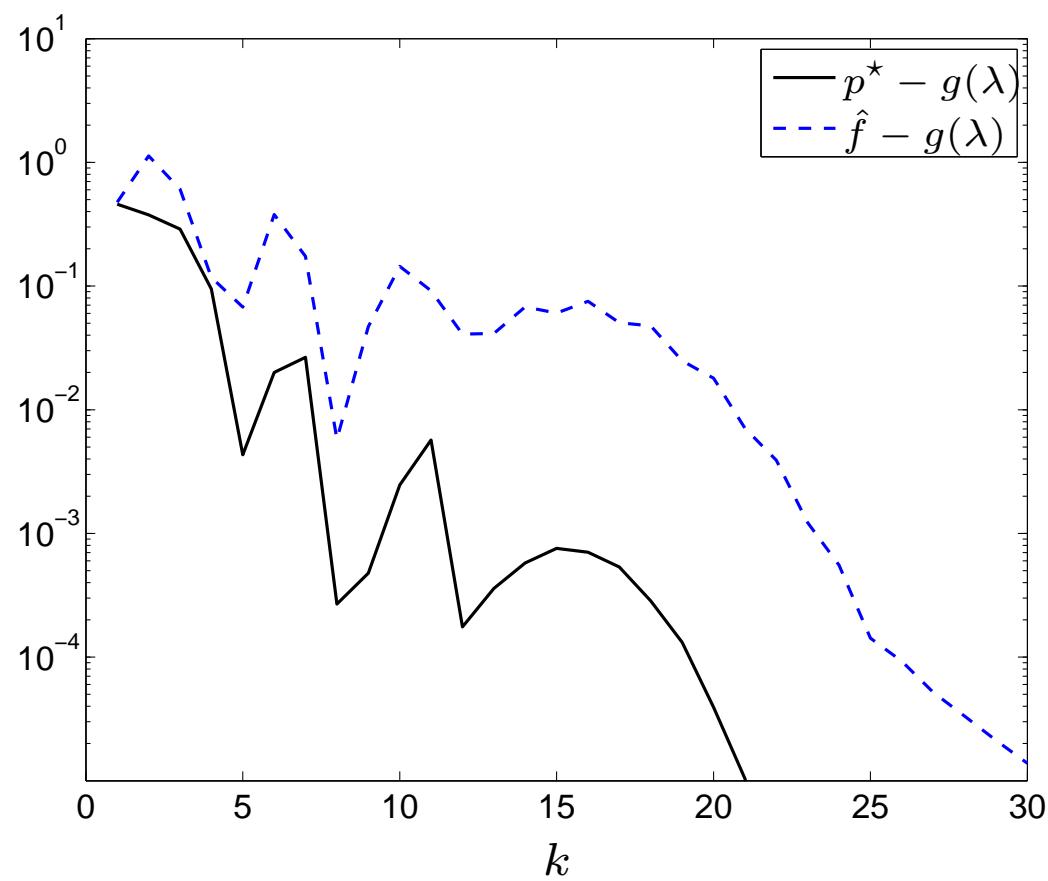
## (Same) example

subgradient method for master; resource prices  $\lambda$



dual decomposition convergence;  $\hat{f}$  is objective of projected feasible allocation

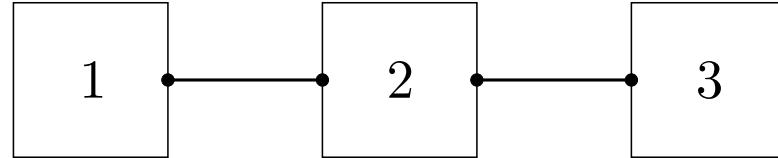




## General decomposition structures

- multiple subsystems
- (variable and/or constraint) coupling constraints between subsets of subsystems
- represent as hypergraph with subsystems as vertices, coupling as hyperedges or nets
- without loss of generality, can assume all coupling is via consistency constraints

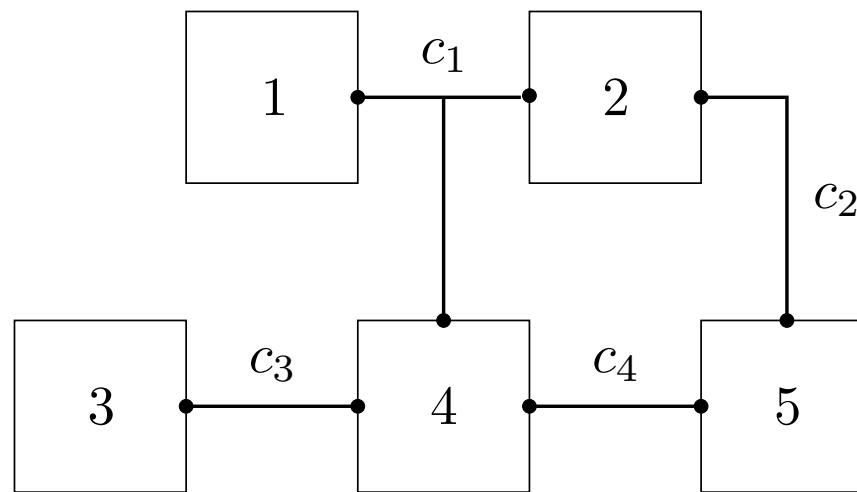
## Simple example



- 3 subsystems, with private variables  $x_1, x_2, x_3$ , and public variables  $y_1, (y_2, y_3)$ , and  $y_4$
- 2 (simple) edges

$$\begin{aligned} \text{minimize} \quad & f_1(x_1, y_1) + f_2(x_2, y_2, y_3) + f_3(x_3, y_4) \\ \text{subject to} \quad & (x_1, y_1) \in \mathcal{C}_1, \quad (x_2, y_2, y_3) \in \mathcal{C}_2, \quad (x_3, y_4) \in \mathcal{C}_3 \\ & y_1 = y_2, \quad y_3 = y_4 \end{aligned}$$

## A more complex example



## General form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^K f_i(x_i, y_i) \\ \text{subject to} \quad & (x_i, y_i) \in \mathcal{C}_i, \quad i = 1, \dots, K \\ & y_i = E_i z, \quad i = 1, \dots, K \end{aligned}$$

- private variables  $x_i$ , public variables  $y_i$
- net (hyperedge) variables  $z \in \mathbf{R}^N$ ;  $z_i$  is common value of public variables in net  $i$
- matrices  $E_i$  give **netlist** or **hypergraph**  
row  $k$  is  $e_p$ , where  $k$ th entry of  $y_i$  is in net  $p$

# Primal decomposition

$\phi_i(y_i)$  is optimal value of subproblem

$$\begin{aligned} & \text{minimize} && f_i(x_i, y_i) \\ & \text{subject to} && (x_i, y_i) \in \mathcal{C}_i \end{aligned}$$

**repeat**

1. Distribute net variables to subsystems.

$$y_i := E_i z, \quad i = 1, \dots, K.$$

2. Optimize subsystems (separately).

Solve subproblems to find optimal  $x_i, g_i \in \partial\phi_i(y_i), \quad i = 1, \dots, K.$

3. Collect and sum subgradients for each net.

$$g := \sum_{i=1}^K E_i^T g_i.$$

4. Update net variables.

$$z := z - \alpha_k g.$$

## Dual decomposition

$g_i(\nu_i)$  is optimal value of subproblem

$$\begin{aligned} & \text{minimize} && f_i(x_i, y_i) + \nu_i^T y_i \\ & \text{subject to} && (x_i, y_i) \in \mathcal{C}_i \end{aligned}$$

**given** initial price vector  $\nu$  that satisfies  $E^T \nu = 0$  (e.g.,  $\nu = 0$ ).

**repeat**

1. Optimize subsystems (separately).  
Solve subproblems to obtain  $x_i, y_i$ .

2. Compute average value of public variables over each net.

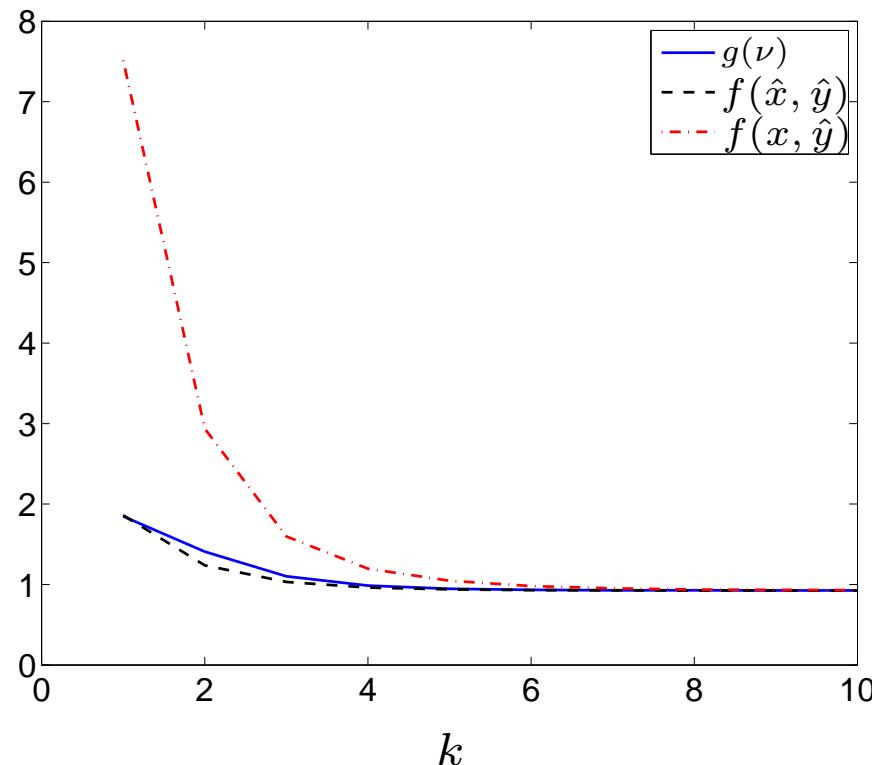
$$\hat{z} := (E^T E)^{-1} E^T y.$$

3. Update prices on public variables.

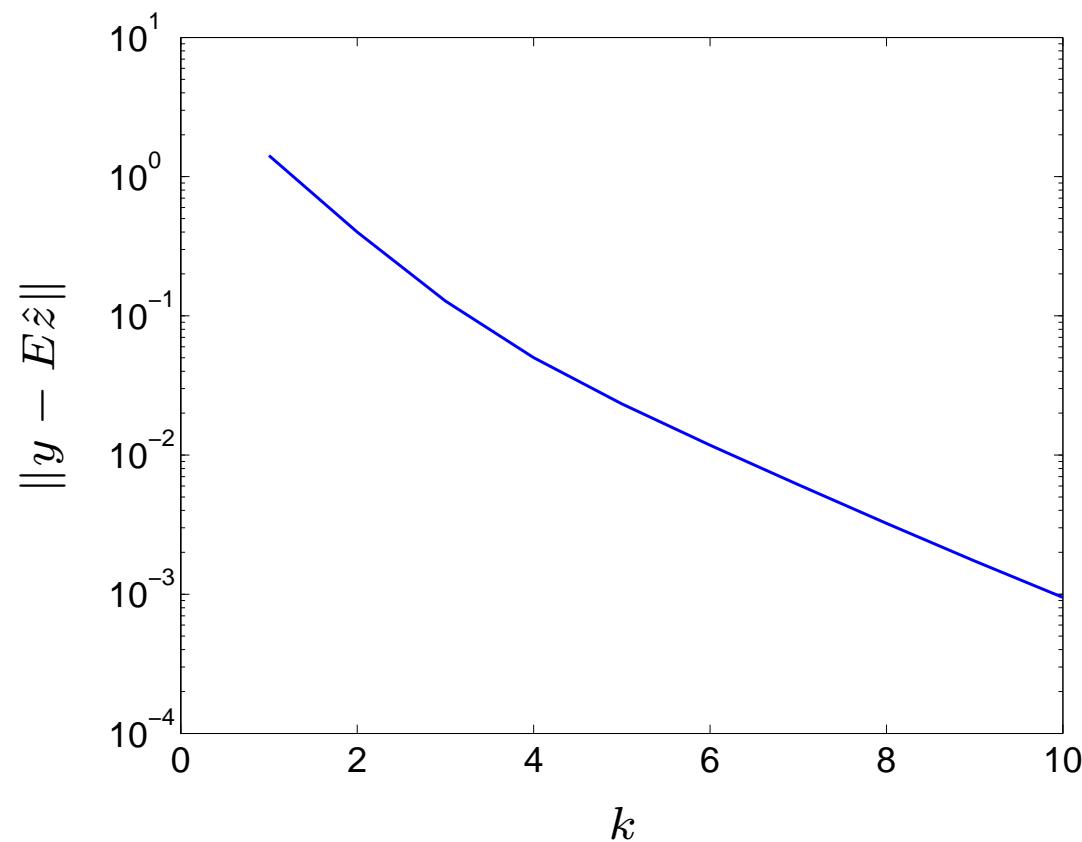
$$\nu := \nu + \alpha_k (y - E\hat{z}).$$

## A more complex example

subsystems: quadratic plus PWL objective with 10 private variables;  
9 public variables and 4 nets;  $p^* \approx 11.1$ ;  $\alpha = 0.5$



consistency constraint residual  $\|y - E\hat{z}\|$  versus iteration number



# Decomposition Applications

- rate control
- single commodity network flow

## Rate control setup

- $n$  flows, with fixed routes, in a network with  $m$  links
- variable  $f_j \geq 0$  denotes the rate of flow  $j$
- flow utility is  $U_j : \mathbf{R} \rightarrow \mathbf{R}$ , strictly concave, increasing
- traffic  $t_i$  on link  $i$  is sum of flows passing through it
- $t = Rf$ , where  $R$  is the routing matrix

$$R_{ij} = \begin{cases} 1 & \text{flow } j \text{ passes over link } i \\ 0 & \text{otherwise} \end{cases}$$

- link capacity constraint:  $t \preceq c$

## Rate control problem

$$\begin{aligned} & \text{maximize} && U(f) = \sum_{j=1}^n U_j(f_j) \\ & \text{subject to} && Rf \preceq c \end{aligned}$$

- convex problem
- dual decomposition gives decentralized method

## Rate control Lagrangian

Lagrangian (for minimizing  $-U$ ) is

$$\begin{aligned} L(f, \lambda) &= -U(f) + \lambda^T(Rf - c) \\ &= -\lambda^T c + \sum_{j=1}^n (-U_j(f_j) + (r_j^T \lambda) f_j) \end{aligned}$$

- $\lambda_i$  is price (per unit flow) for using link  $i$
- $r_j^T \lambda$  is the sum of prices along route  $j$

## Rate control dual

dual function is

$$\begin{aligned} g(\lambda) &= -\lambda^T c + \sum_{j=1}^n \inf_{f_j} (-U_j(f_j) + (r_j^T \lambda) f_j) \\ &= -\lambda^T c - \sum_{j=1}^n (-U_j)^*(-r_j^T \lambda), \end{aligned}$$

dual rate control problem:

$$\begin{array}{ll} \text{maximize} & -\lambda^T c - \sum_{j=1}^n (-U_j)^*(-r_j^T \lambda) \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

subgradient of negative dual:

$$R\bar{f} - c \in \partial(-g)(\lambda)$$

where  $\bar{f}_j = \operatorname{argmax} (U_j(f_j) - (r_j^T \lambda) f_j)$

## Dual decomposition rate control algorithm

**given** initial link price vector  $\lambda \succeq 0$  (e.g.,  $\lambda = 1$ ).

**repeat**

1. Sum link prices along each route.

$$\text{Calculate } \Lambda_j = r_j^T \lambda.$$

2. Optimize flows (separately) using flow prices.

$$f_j := \operatorname{argmax} (U_j(f_j) - \Lambda_j f_j).$$

3. Calculate link capacity margins.

$$s := c - Rf.$$

4. Update link prices.

$$\lambda := (\lambda - \alpha_k s)_+.$$

## Dual decomposition rate control algorithm

- decentralized:
  - links only need to know the flows that pass through them
  - flows only need to know prices on links they pass through
- prices converge to optimal; so do flows (since  $U$  is strictly concave)
- iterates can be (and often are) infeasible, *i.e.*,  $Rf \not\leq c$   
(but we do have  $Rf \preceq c$  in the limit)
- have upper bound  $-g(\lambda)$  on optimal utility  $U^*$

## Generating feasible flows

- define  $\eta_i = t_i/c_i = (Rf)_i/c_i$ 
  - $\eta_i < 1$  means link  $i$  is under capacity
  - $\eta_i > 1$  means link  $i$  is over capacity

- define  $f^{\text{feas}}$  as

$$f_j^{\text{feas}} = \frac{f_j}{\max\{\eta_i \mid \text{flow } j \text{ passes over link } i\}}$$

- $f^{\text{feas}}$  will be feasible, even if  $f$  is not
- finding  $f^{\text{feas}}$  is also decentralized  
(in fact this is a step in primal decomposition)

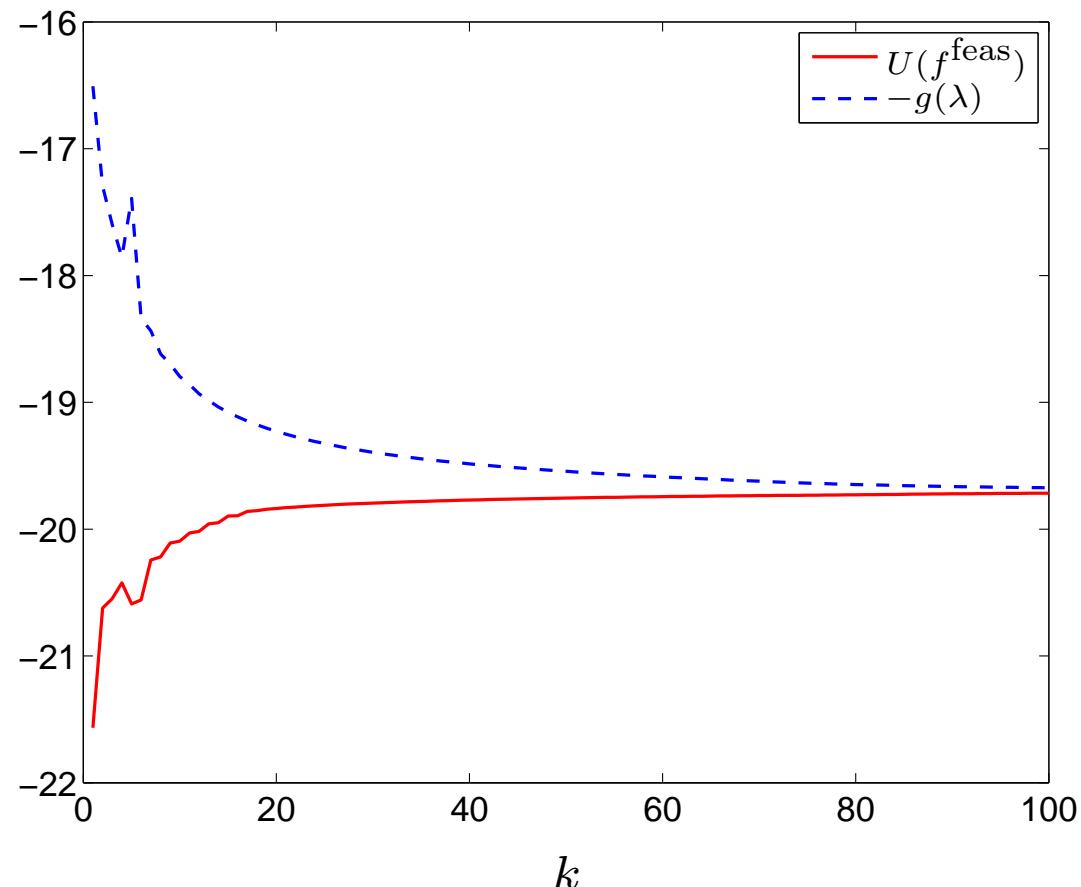
## Example

- $n = 10$  flows,  $m = 12$  links; 3 or 4 links per flow
- link capacities chosen randomly, uniform on  $[0.1, 1]$
- $U_j(f_j) = \log f_j$  (can be argued to give proportionally fair flows)
- optimal flow as a function of price:

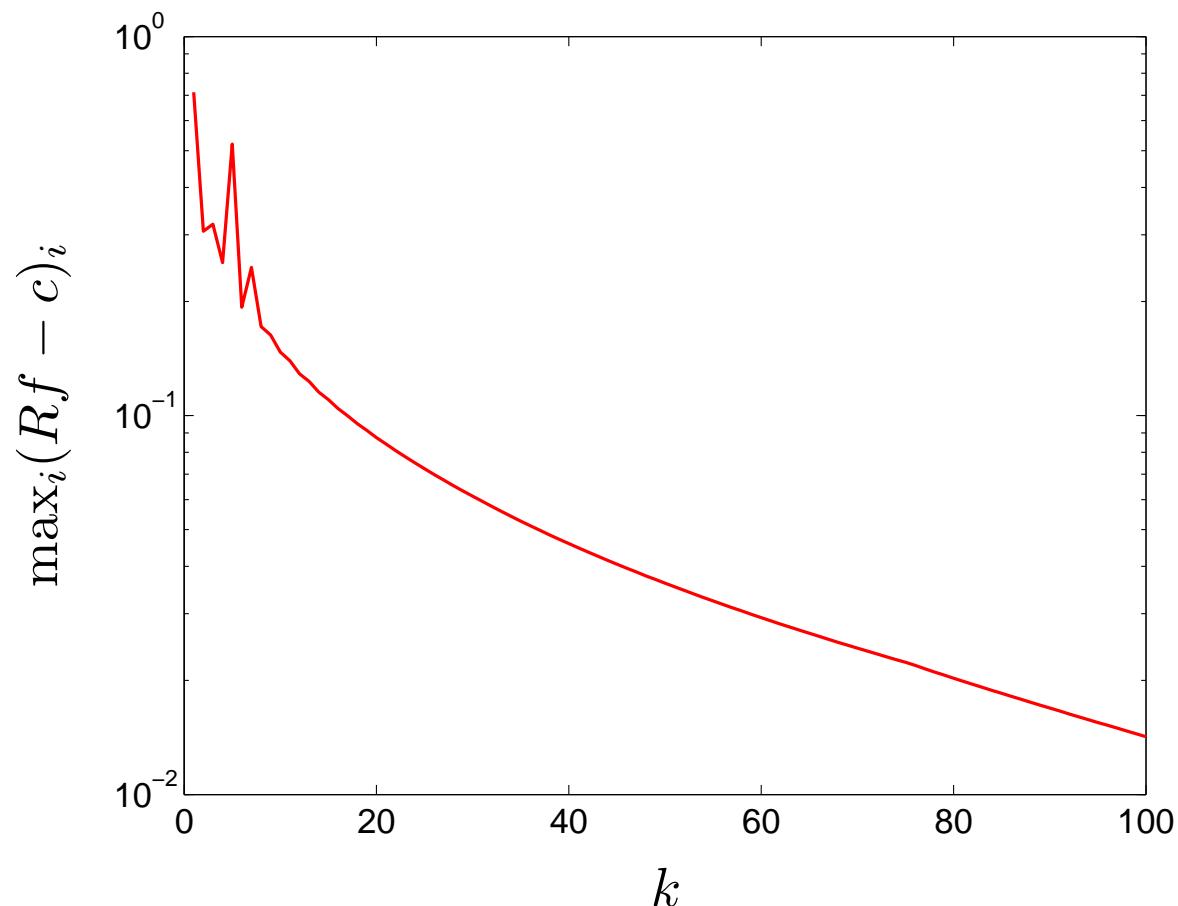
$$\bar{f}_j = \operatorname{argmax}(U_j(f_j) - \Lambda_j f_j) = 1/\Lambda_j$$

- initial prices:  $\lambda = 1$
- constant stepsize  $\alpha_k = 3$

## Convergence of primal and dual objectives



## Maximum capacity violation



## Single commodity network flow setup

- connected, directed graph with  $n$  links,  $p$  nodes
- variable  $x_j$  denotes flow (traffic) on arc  $j$
- given external source (or sink) flow  $s_i$  at node  $i$ ,  $\mathbf{1}^T s = 0$
- node incidence matrix  $A \in \mathbb{R}^{p \times n}$  is

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ enters } i \\ -1 & \text{arc } j \text{ leaves node } i \\ 0 & \text{otherwise} \end{cases}$$

- flow conservation:  $Ax + s = 0$
- $\phi(x) = \sum_{j=1}^n \phi_j(x_j)$  is separable convex flow cost function

## Network flow problem

optimal single commodity network flow problem:

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^n \phi_j(x_j) \\ & \text{subject to} && Ax + s = 0 \end{aligned}$$

- convex, readily solved with standard methods
- dual decomposition yields decentralized solution method

## Network flow Lagrangian

Lagrangian is

$$\begin{aligned} L(x, \nu) &= \phi(x) + \nu^T(Ax + s) \\ &= \nu^T s + \sum_{j=1}^n (\phi_j(x_j) + (a_j^T \nu)x_j) \end{aligned}$$

- $a_j$  is  $j$ th column of  $A$
- we'll interpret  $\nu_i$  as potential at node  $i$
- we use  $\Delta\nu_j$  to denote  $a_j^T \nu$ , which is potential difference across edge  $j$

## Network flow dual

dual function:

$$\begin{aligned} g(\nu) &= \inf_x L(x, \nu) \\ &= \nu^T s + \sum_{j=1}^n \inf_{x_j} (\phi_j(x_j) + (\Delta\nu_j)x_j) \\ &= \nu^T s - \sum_{j=1}^n \phi_j^*(-\Delta\nu_j) \end{aligned}$$

dual problem: maximize  $g(\nu)$

## Recovering primal from dual

- strictly convex  $\phi_j$  means unique minimizer  $x_j^*(y)$  of  $\phi_j(x_j) - yx_j$
- if  $\phi_j$  is differentiable,  $x_j^*(y) = (\phi'_j)^{-1}(y)$  (inverse of derivative function)
- optimal flows, from optimal potentials:  $x_j^* = x_j^*(-\Delta\nu_j^*)$
- subgradient of negative dual function:

$$-(Ax^*(\Delta\nu) + s) \in \partial(-g)(\nu)$$

(negative of flow conservation residual)

## Dual decomposition network flow algorithm

**given** initial potential vector  $\nu$ .

**repeat**

1. Determine link flows from potential differences.

$$x_j := x_j^*(-\Delta\nu_j), \quad j = 1, \dots, n.$$

2. Compute flow surplus at each node.

$$S_i := a_i^T x + s_i, \quad i = 1, \dots, p.$$

3. Update node potentials.

$$\nu_i := \nu_i + \alpha_k S_i, \quad i = 1, \dots, p.$$

$\alpha_k$  is an appropriate step size

## Dual decomposition network flow algorithm

- decentralized:
  - flow calculated from potential difference across edge
  - node potential updated from its own flow surplus
- $g(\nu)$  gives lower bound on  $p^*$
- flow conservation  $Ax + s = 0$  only holds in limit

## Electrical network analogy

- electrical network with node incidence matrix  $A$ , nonlinear resistors in branches
- variable  $x_j$  is the current flow in branch  $j$
- source  $s_i$  is external current injected at node  $i$  (must sum to zero)
- flow conservation equation  $Ax + s = 0$  is Kirhoff Current Law (KCL)
- dual variables are node potentials;  $\Delta\nu_j$  is  $j$ th branch voltage
- branch current-voltage characteristic is  $x_j = x_j^*(-\Delta\nu_j)$

then, current and potentials in circuit are optimal flows and dual variables

## Example: Minimum queueing delay

flow cost function

$$\phi_j(x_j) = \frac{x_j}{c_j - x_j}, \quad \text{dom } \phi_j = [0, c_j)$$

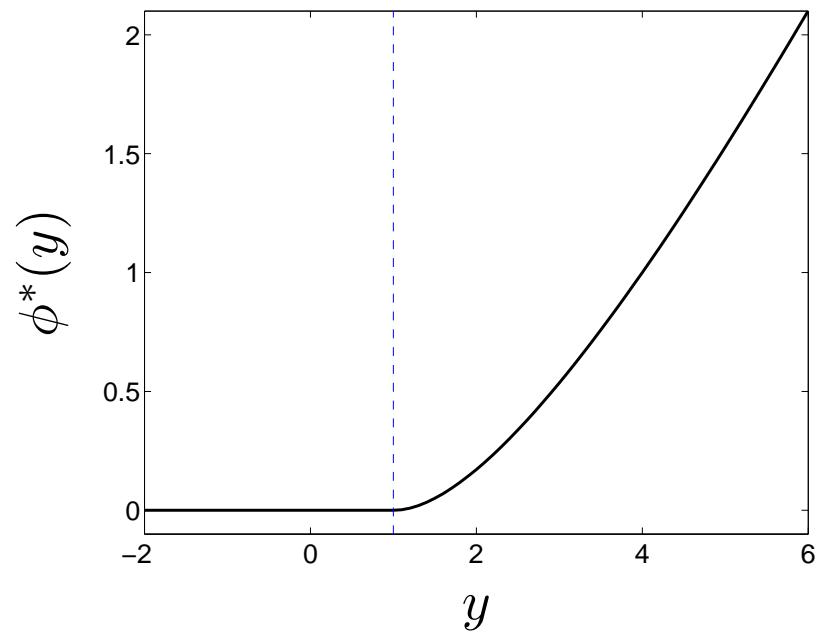
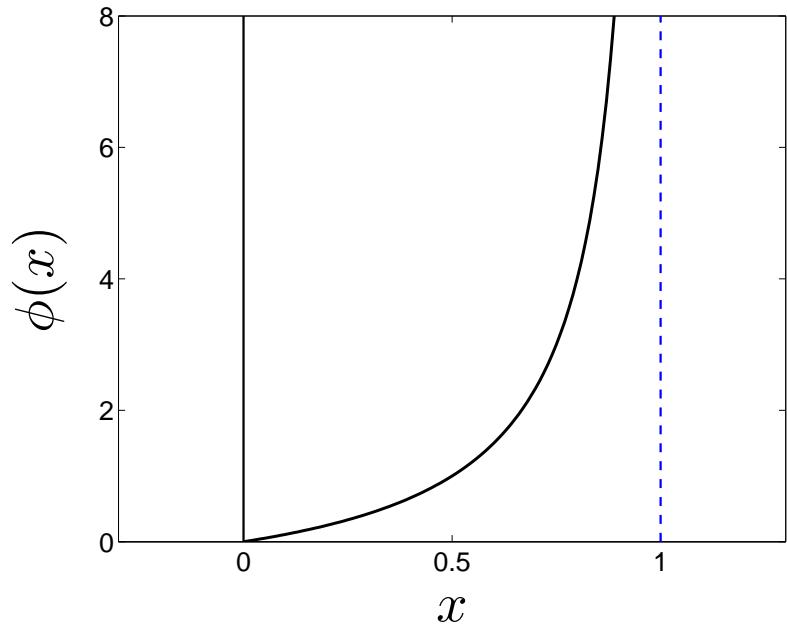
where  $c_j > 0$  are given *link capacities*

( $\phi_j(x_j)$  gives expected waiting time in queue with exponential arrivals at rate  $x_j$ , exponential service at rate  $c_j$ )

conjugate is

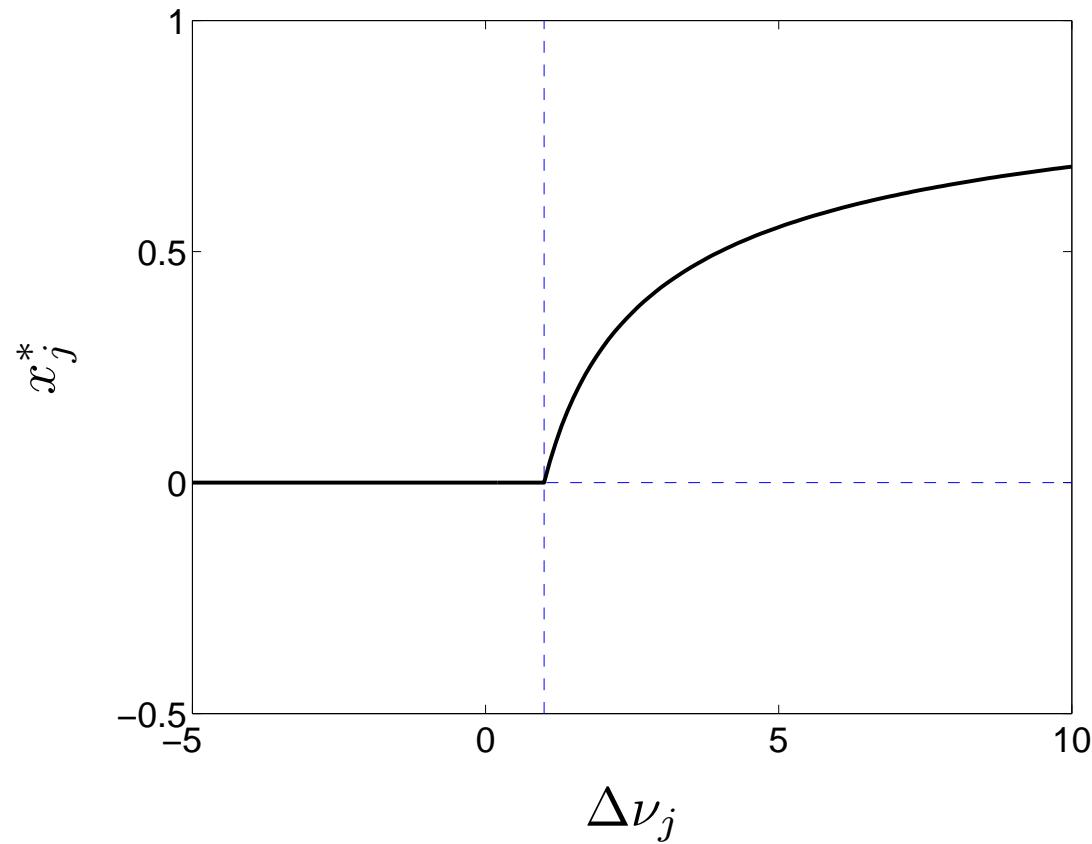
$$\phi_j^*(y) = \begin{cases} (\sqrt{c_j y} - 1)^2 & y > 1/c_j \\ 0 & y \leq 1/c_j \end{cases}$$

cost function  $\phi(x)$  (left) and its conjugate  $\phi^*(y)$  (right),  $c = 1$



(note that conjugate is differentiable)

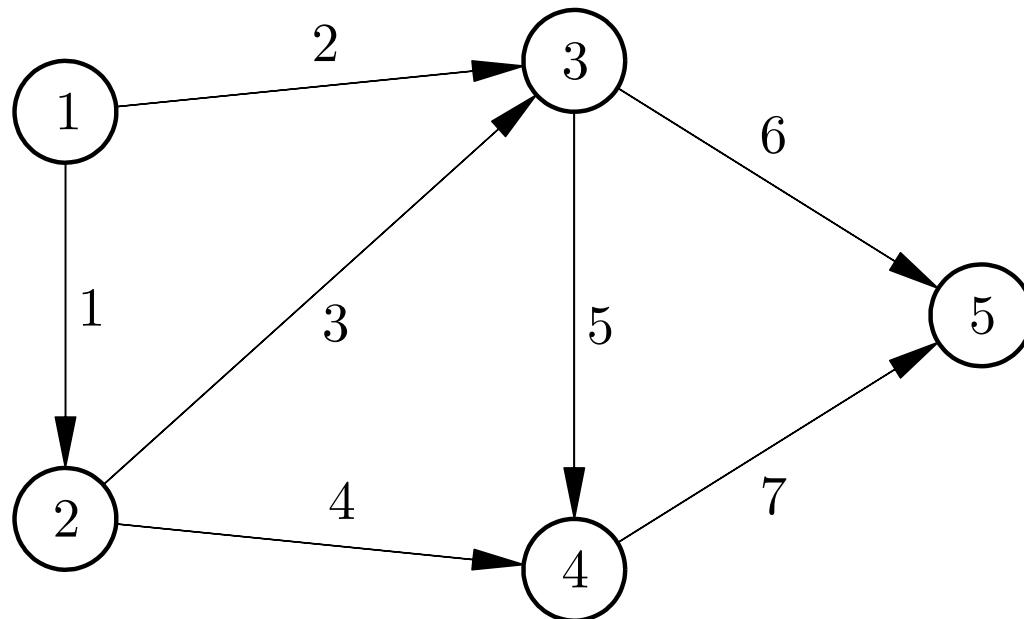
$x_j^*(-\Delta\nu_j)$ , for  $c_j = 1$



gives flow as function of potential difference across link

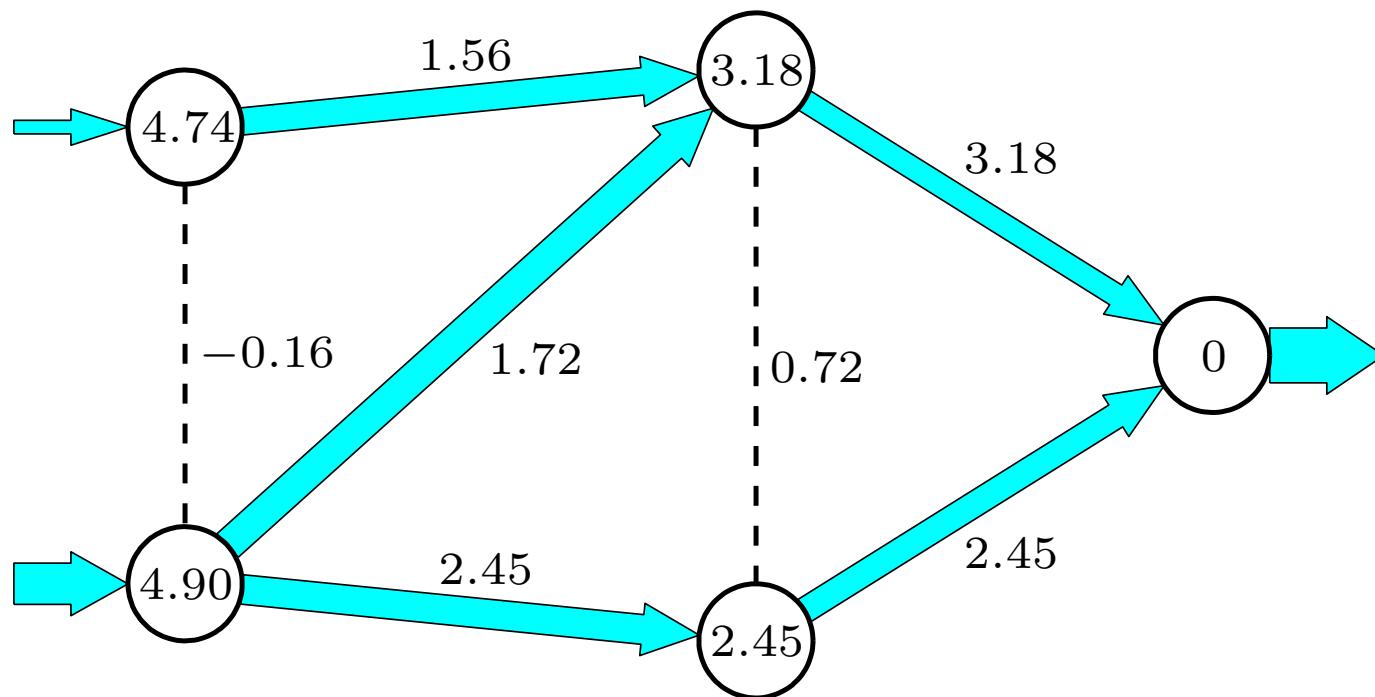
## A specific example

network with 5 nodes, 7 links, capacities  $c_j = 1$



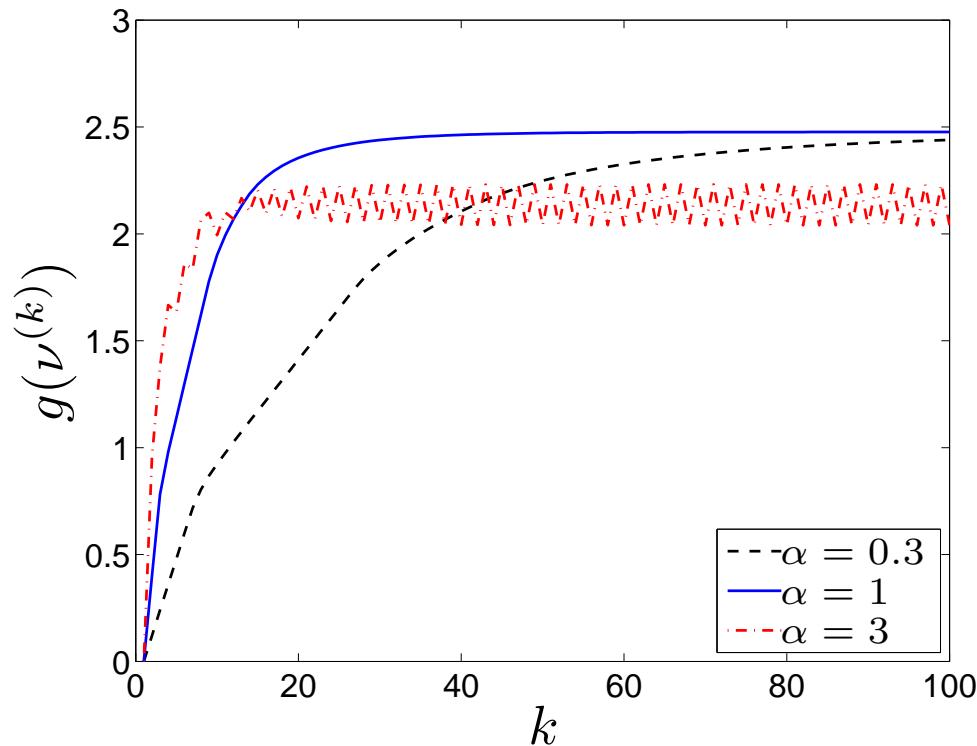
## Optimal flow

optimal flows shown as width of arrows; optimal dual variables shown in nodes; potential differences shown on links



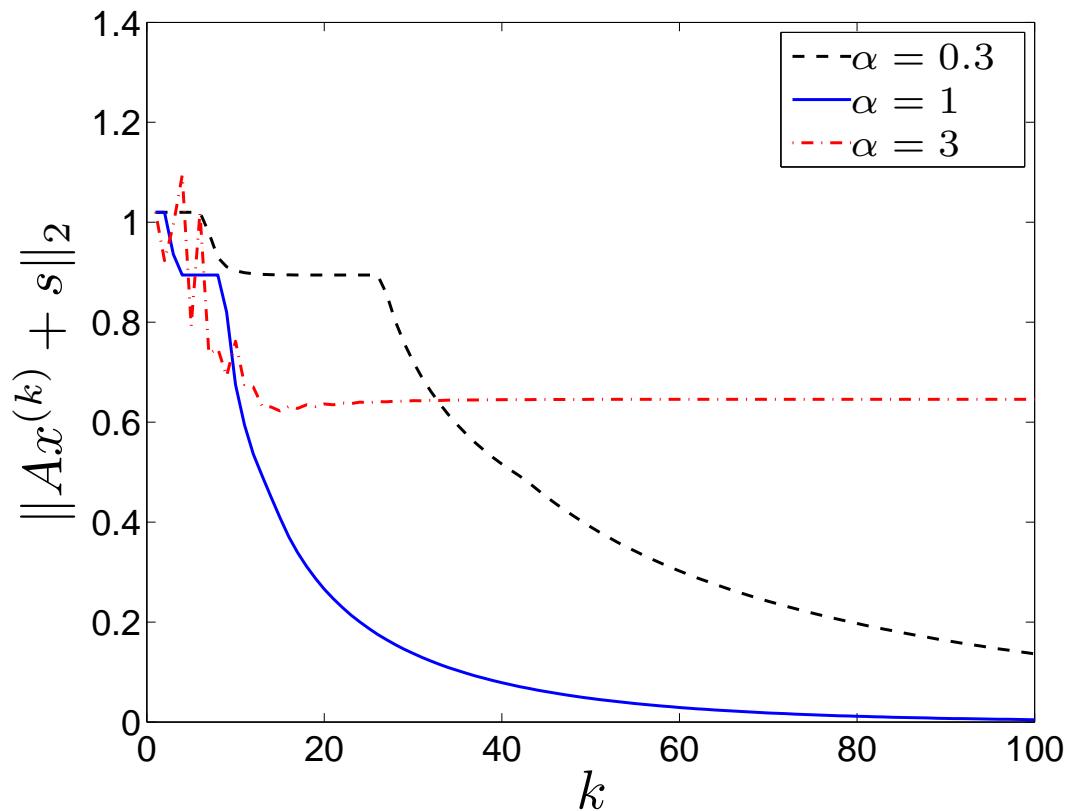
## Convergence of dual function

fixed step size rules,  $\alpha = 0.3, 1, 3$



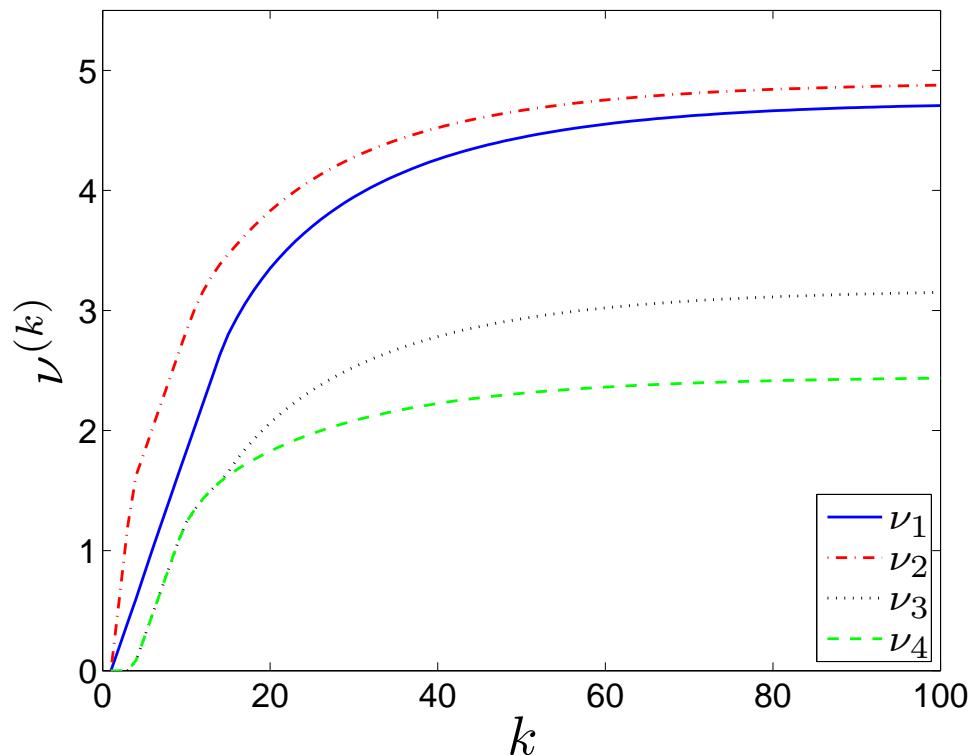
for  $\alpha = 1$ , converges to  $p^* = 2.48$  in around 40 iterations

## Convergence of primal residual



## Convergence of dual variables

$\nu^{(k)}$  versus iteration number  $k$ , fixed step size rule  $\alpha = 1$



( $\nu_5$  is fixed as zero)

# **Distributed Optimization: Analysis and Synthesis via Circuits**

Stephen Boyd

Prof. S. Boyd, EE364b, Stanford University

# Outline

- canonical form for distributed convex optimization
- circuit interpretation
- primal decomposition
- dual decomposition
- prox decomposition
- momentum terms

## Distributed convex optimization problem

- convex optimization problem partitioned into coupled subsystems
- divide variables, constraints, objective terms into two groups
  - **local** variables, constraints, objective terms appear in only one subsystem
  - **complicating** variables, constraints, objective terms appear in more than one subsystem
- describe by hypergraph
  - subsystems are nodes
  - complicating variables, constraints, objective terms are hyperedges

## Conditional separability

- **separable problem:** can solve by solving subsystems separately, *e.g.*,

$$\begin{aligned} & \text{minimize} && f_1(x_1) + f_2(x_2) \\ & \text{subject to} && x_1 \in \mathcal{C}_1, \quad x_2 \in \mathcal{C}_2 \end{aligned}$$

- in distributed problem, two subsystems are **conditionally separable** if they are separable when all other variables are fixed
- two subsystems not connected by a net are **conditionally separable**
- cf. conditional independence in Bayes net: two variables not connected by hyperedge are conditionally independent, given all other variables

## Examples

- minimize  $f_1(z_1, x) + f_2(z_2, x)$ , with variables  $z_1, z_2, x$ 
  - $x$  is the **complicating variable**; when fixed, problem is separable
  - $z_1, z_2$  are **private** or **local** variables
  - $x$  is a **public** or **interface** or **boundary** variable between the two subproblems
  - hypergraph: two nodes connected by an edge
- optimal control problem
  - state is the complicating variable between past and future
  - hypergraph: simple chain

## Transformation to standard form

- introduce slack variables for complicating inequality constraints
- introduce local copies of complicating variables
- implicitly minimize over private variables (preserves convexity)
- represent local constraints in domain of objective term
- we are left with
  - all variables are public, associated with a single node
  - all constraints are **consistency constraints**, *i.e.*, equality of two or more variables

## Example

- minimize  $f_1(z_1, x) + f_2(z_2, x)$ , with variables  $z_1, z_2, x$
- introduce local copies of complicating variable:

$$\begin{aligned} & \text{minimize} && f_1(z_1, x_1) + f_2(z_2, x_2) \\ & \text{subject to} && x_1 = x_2 \end{aligned}$$

- eliminate local variables:

$$\begin{aligned} & \text{minimize} && \tilde{f}_1(x_1) + \tilde{f}_2(x_2) \\ & \text{subject to} && x_1 = x_2 \end{aligned}$$

with  $\tilde{f}_i(x_i) = \inf_{z_i} f_i(z_i, x_i)$

## General form

- $n$  subsystems with variables  $x_1, \dots, x_n$
- $m$  nets with common variable values  $z_1, \dots, z_m$

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n f_i(x_i) \\ & \text{subject to} && x_i = E_i z, \quad i = 1, \dots, n \end{aligned}$$

- matrices  $E_i$  give **netlist** or **hypergraph**  
(row  $k$  is  $e_p$ , where  $k$ th entry of  $x_i$  is in net  $p$ )

## Optimality conditions

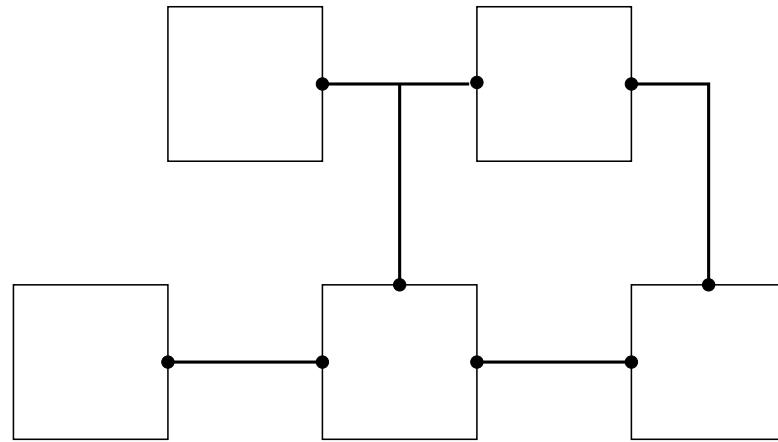
- introduce dual variable  $y_i$  associated with  $x_i = E_i z$
- optimality conditions are

$$\begin{aligned}\nabla f_i(x_i) &= y_i && \text{(subsystem relations)} \\ x_i &= E_i z && \text{(primal feasibility)} \\ \sum_{i=1}^n E_i^T y_i &= 0 && \text{(dual feasibility)}\end{aligned}$$

(for nondifferentiable case, replace  $\nabla f_i(x_i)$  with  $g_i \in \partial f_i(x_i)$ )

- primal condition: (primal) variables on each net are the same
- dual condition: dual variables on each net sum to zero

## Circuit interpretation (primal/voltages)



- subsystems are (grounded) nonlinear resistors
- nets are wires (nets); consistency constraint is KVL
- $z_j$  is voltage on net  $j$
- $x_i$  is vector of pin voltages for resistor  $i$

## Circuit interpretation (dual/currents)

- $y_i$  is vector of currents entering resistor  $i$
- dual feasibility is KCL: sum of currents leaving net  $j$  is zero
- V-I characteristic for resistor  $i$ :  $y_i = \nabla f_i(x_i)$
- $f_i(x)$  is **content function** of resistor  $i$
- convexity of  $f_i$  is **incremental passivity** of resistor  $i$ :

$$(x_i - \tilde{x}_i)^T (y_i - \tilde{y}_i) \geq 0, \quad y_i = \nabla f_i(x_i), \quad \tilde{y}_i = \nabla f_i(\tilde{x}_i)$$

- **optimality conditions are exactly the circuit equations**

## Decomposition methods

- solve distributed problem iteratively
  - algorithm state maintained in nets
- each step consists of
  - (parallel) update of subsystem primal and dual variables, based only on adjacent net states
  - update of the net states, based only on adjacent subsystems
- algorithms differ in
  - interface to subsystems
  - state and update

## Primal decomposition

**repeat**

1. distribute net variables to adjacent subsystems

$$x_i := E_i z$$

2. optimize subsystems (separately)

solve subproblems to evaluate  $y_i = \nabla f_i(x_i)$

3. collect and sum dual variables for each net

$$w := \sum_{i=1}^n E_i^T y_i$$

4. update net variables

$$z := z - \alpha_k w.$$

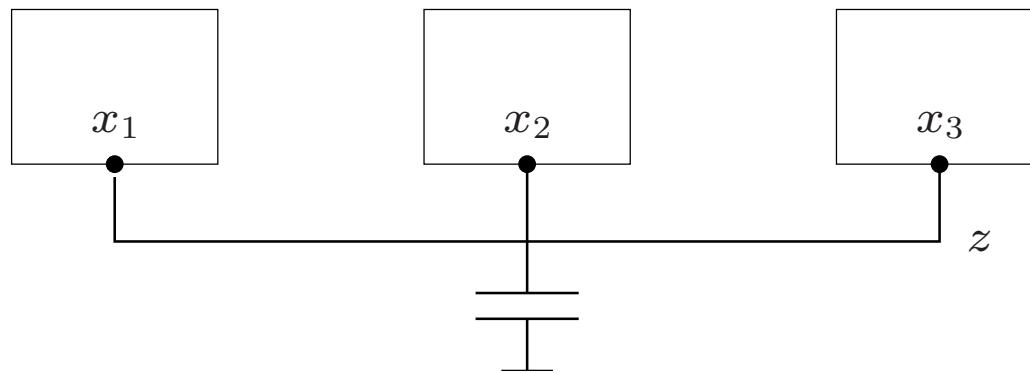
- step factor  $\alpha_k$  chosen by standard gradient or subgradient rules

## Primal decomposition

- algorithm state is net variable  $z$  (net voltages)
- $w = \sum_{i=1}^n E_i^T y_i$  is dual residual (net current residuals)
- primal feasibility maintained; dual feasibility approached in limit
- subsystems are **voltage controlled**:
  - voltage  $x_i$  is asserted at subsystem pins
  - pin currents  $y_i$  are then found

## Circuit interpretation

- connect capacitor to each net; system relaxes to equilibrium
- forward Euler update is primal decomposition
- incremental passivity implies convergence to equilibrium



## Dual decomposition

initialize  $y_i$  so that  $\sum_{i=1}^n E_i^T y_i = 0$   
(dual variables sum to zero on each net)

**repeat**

1. optimize subsystems (separately)

    find  $x_i$  with  $\nabla f_i(x_i) = y_i$ , i.e., minimize  $f_i(x_i) - y_i^T x_i$

2. collect and average primal variables over each net

$$z := (E^T E)^{-1} E^T x$$

3. update dual variables

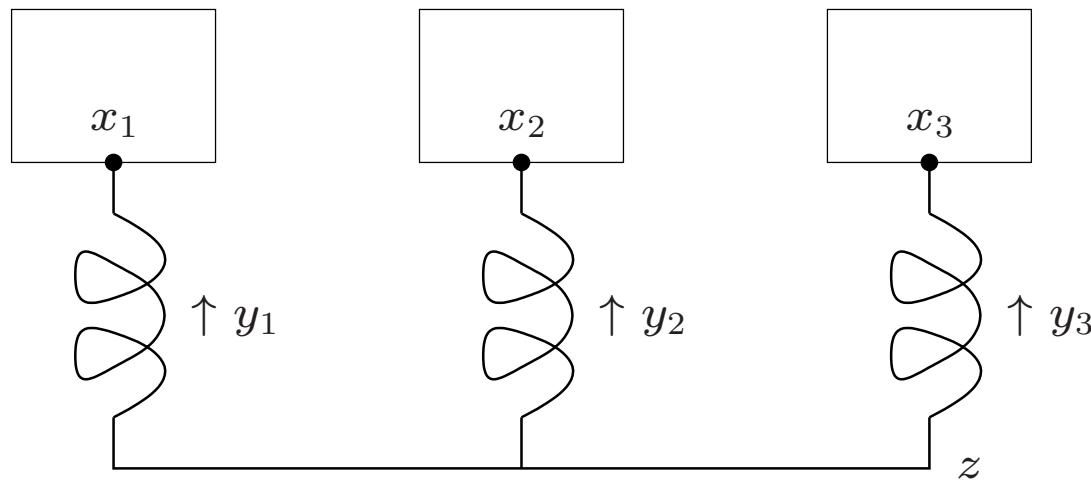
$$y := y - \alpha_k (x - Ez)$$

## Dual decomposition

- algorithm state is dual variable  $y$
- $x - Ez$  is consistency residual
- dual feasibility maintained; primal feasibility approached in limit
- subsystems are **current controlled**:
  - pin currents  $y_i$  are asserted
  - pin voltages  $x_i$  are then found

## Circuit interpretation

- connect inductor to each pin; system relaxes to equilibrium
- forward Euler update is dual decomposition
- incremental passivity implies convergence to equilibrium

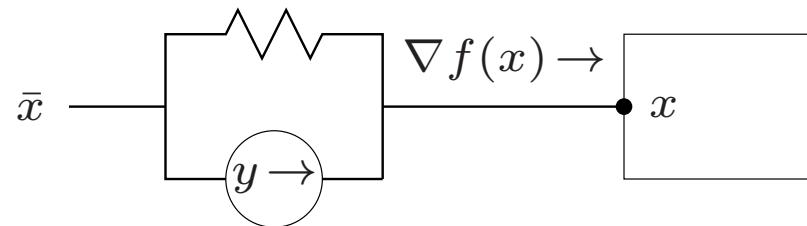


## Prox(imal) interface

- prox operator:

$$P_\rho(y, \bar{x}) = \operatorname{argmin}_x (f(x) - y^T x + (\rho/2)\|x - \bar{x}\|_2^2)$$

- contains usual dual term  $y^T x$  and ‘proximal regularization’ term
- amounts to solving  $\nabla f(x) + \rho(x - \bar{x}) = y$
- circuit interpretation: drive via resistance  $R = 1/\rho$   
cf. voltage (primal) drive or current (dual) drive



## Prox decomposition

initialize  $y_i$  so that  $\sum_{i=1}^n E_i^T y_i = 0$

**repeat**

1. optimize subsystems (separately)

$$\text{minimize } f_i(x_i) - y_i^T x_i + (\rho/2) \|x_i - E_i z\|^2$$

2. collect and average primal variables over each net

$$z := (E^T E)^{-1} E^T x$$

3. update dual variables

$$y := y - \rho(x - Ez)$$

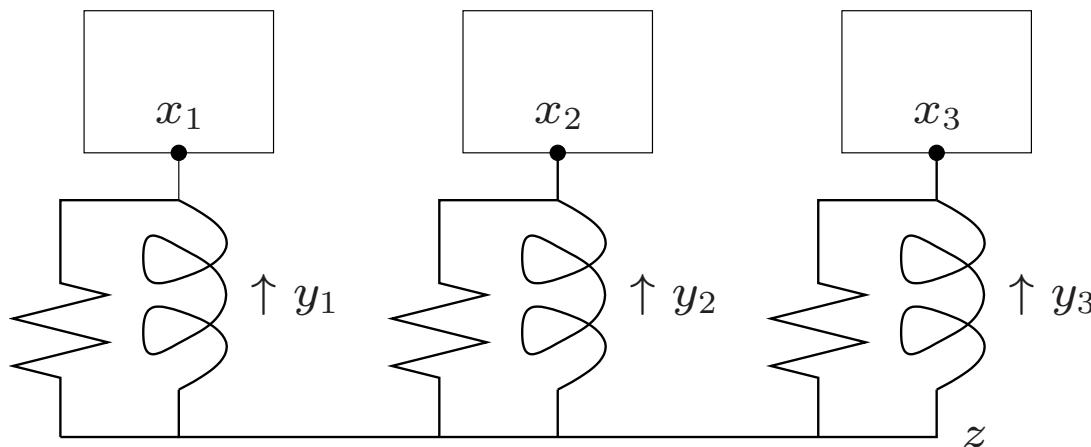
- step size  $\rho$  in dual update guaranteed to work

## Prox decomposition

- has **many** other names . . .
- algorithm state is dual variable  $y$
- $y - \rho(x - \bar{x})$  is dual feasible
- primal and dual feasibility approached in limit
- subsystems are resistor driven; must support prox interface
- interpretations
  - regularized dual decomposition
  - PI feedback (as opposed to I only feedback)

## Circuit interpretation

- connect inductor || resistor to each pin; system relaxes to equilibrium
- forward Euler update is prox decomposition
- incremental passivity implies convergence to equilibrium

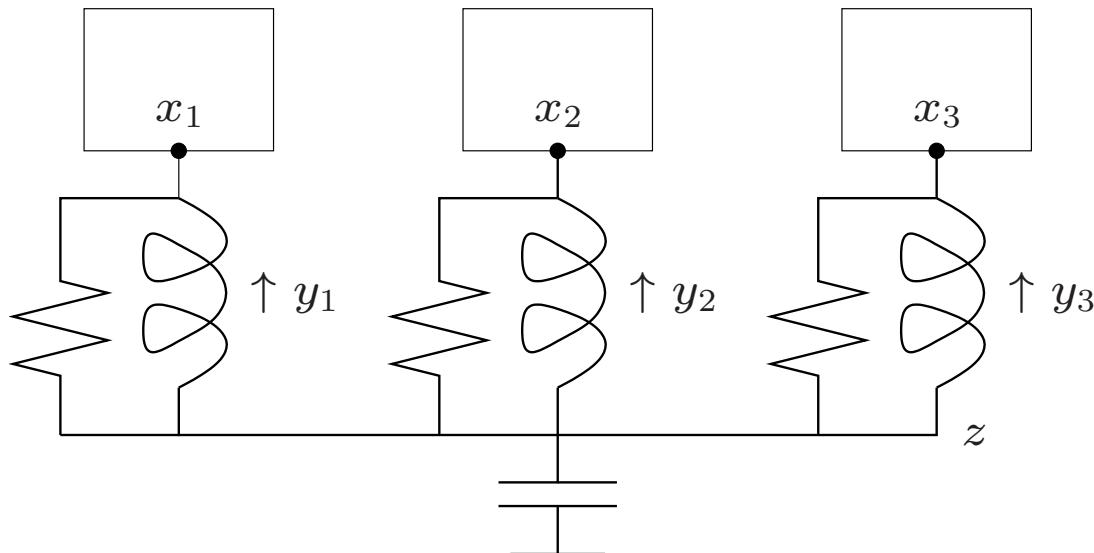


## Momentum terms

- in optimization method, current search direction is
  - standard search direction (gradient, subgradient, prox . . . )
  - plus last search direction, scaled
- interpretations/examples
  - smooth/low pass filter/average search directions
  - add momentum to search algorithm ('heavy-ball method')
  - two term method (CG)
  - Nesterov optimal order method
- often dramatically improves convergence

# You guessed it

- algorithm: prox decomposition with momentum
- just add capacitor to prox LR circuit



## Conclusions

to get a distributed optimization algorithm:

- represent as circuit with interconnecting wires
- replace interconnect wires with passive circuits that reduce to wires at equilibrium
- discretize circuit dynamics
- subsystem interfaces depend on circuit drive  
(current, voltage, via resistor)
- convergence hinges on incremental passivity

# Proximal Algorithms

Neal Parikh and Stephen Boyd  
Stanford University

Based on 'Proximal Algorithms', *Foundations and Trends in Optimization*,  
2014.

# Outline

Proximal operators

Proximal algorithms

Applications

Conclusions

## Proximal operator

- ▶ proximal operator of  $f : \mathbf{R}^n \rightarrow \mathbf{R} \cup \{+\infty\}$  is

$$\mathbf{prox}_{\lambda f}(v) = \operatorname{argmin}_x (f(x) + (1/2\lambda)\|x - v\|_2^2)$$

with parameter  $\lambda > 0$

- ▶  $f$  may be nonsmooth, have embedded constraints, ...
- ▶ evaluating  $\mathbf{prox}_f$  involves solving a convex optimization problem
- ▶ can evaluate via standard methods like BFGS, but very often has an analytical solution or simple specialized linear-time algorithm

## Generalized projection

- ▶ proximal operator of an indicator function of a convex set is projection:

$$\text{prox}_{\lambda I_{\mathcal{C}}}(v) = \Pi_{\mathcal{C}}(v) = \operatorname{argmin}_{x \in \mathcal{C}} \|x - v\|_2$$

- ▶ many properties carry over
- ▶ **example:** projection onto box  $\mathcal{C} = \{x \mid l \preceq x \preceq u\}$  given by

$$(\Pi_{\mathcal{C}}(v))_k = \begin{cases} l_k & v_k \leq l_k \\ v_k & l_k \leq v_k \leq u_k \\ u_k & v_k \geq u_k \end{cases}$$

## Quadratic functions

- ▶ if  $f(x) = (1/2)x^T Px + q^T x + r$ , then

$$\text{prox}_{\lambda f}(v) = (I + \lambda P)^{-1}(v - \lambda q)$$

- ▶ if  $P$  is dense and direct method is used, costs  $O(n^3)$  flops to factor and then  $O(n^2)$  to backsolve on subsequent evaluations of  $\text{prox}_{\lambda f}$
- ▶ still get some discount if  $P$  is sparse or has some structure
- ▶ if using iterative method like CG/LSQR, warm start beginning at  $v$

## Separable sum

- ▶ if  $f$  is block separable, so  $f(x) = \sum_{i=1}^N f_i(x_i)$ , then

$$(\text{prox}_f(v))_i = \text{prox}_{f_i}(v_i), \quad i = 1, \dots, N$$

- ▶ key to parallel/distributed proximal algorithms
- ▶ **example:** if  $f = \|\cdot\|_1$ , then

$$\text{prox}_{\lambda f}(v) = (v - \lambda)_+ - (-v - \lambda)_+ = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases}$$

a simple elementwise operation called *soft thresholding*

## Fixed points

- ▶ the point  $x^*$  minimizes  $f$  if and only if  $x^*$  is a fixed point:

$$x^* = \text{prox}_f(x^*)$$

- ▶ provides a link between proximal operators and fixed point theory
- ▶ many proximal algorithms can be viewed as methods for finding fixed points of appropriate operators

## Moreau-Yosida regularization

- ▶ **infimal convolution** of  $f$  and  $g$ , denoted  $f \square g$ , is defined as

$$(f \square g)(v) = \inf_x (f(x) + g(v - x))$$

- ▶ **Moreau envelope** or **Moreau-Yosida regularization** of  $f$  is

$$M_{\lambda f}(v) = \inf_x (f(x) + (1/2\lambda)\|x - v\|_2^2)$$

- ▶ a smoothed or regularized form of  $f$ :
  - always has full domain
  - always continuously differentiable
  - has the same minimizers as  $f$
- ▶ can minimize  $M_f$  instead of  $f$ , though  $M_f$  could be hard to evaluate

## Moreau-Yosida regularization

- ▶ motivation: can show that

$$M_f = (f^* + (1/2)\|\cdot\|_2^2)^*$$

- ▶ in general,  $\varphi^*$  is smooth when  $\varphi$  is strongly convex
- ▶ Moreau envelope obtains a smooth approximation via:
  1. taking conjugate
  2. regularizing to get a strongly convex function
  3. taking conjugate again
- ▶ **example:** Moreau envelope of  $|\cdot|$  is the Huber function

$$\varphi^{\text{huber}}(x) = \begin{cases} x^2 & |x| \leq 1 \\ 2|x| - 1 & |x| > 1 \end{cases}$$

## Modified gradient step

- ▶ many relationships between proximal operators and gradient steps
- ▶ proximal operator is gradient step for Moreau envelope:

$$\text{prox}_{\lambda f}(x) = x - \lambda \nabla M_{\lambda f}(x)$$

- ▶ for small  $\lambda$ ,  $\text{prox}_{\lambda f}$  converges to gradient step in  $f$ :

$$\text{prox}_{\lambda f}(x) = x - \lambda \nabla f(x) + o(\lambda)$$

- ▶ parameter can be interpreted as a step size, though proximal methods will generally work even for large step sizes, unlike gradient method

## Resolvent of subdifferential operator

- if  $z = \text{prox}_{\lambda f}(x)$ , then

$$z = \operatorname{argmin}_u (f(u) + (1/2\lambda)\|u - x\|_2^2)$$

- can rewrite as

$$\begin{aligned} 0 &\in \partial_z (f(z) + (1/2\lambda)\|z - x\|_2^2) \\ 0 &\in \partial f(z) + (1/\lambda)(z - x) \\ x &\in (I + \lambda\partial f)(z) = z + \lambda\partial f(z) \\ z &\in (I + \lambda\partial f)^{-1}(x) \end{aligned}$$

- must be careful to interpret  $\partial f$  and expressions using it as relations
- mapping  $(I + \lambda\partial f)^{-1}$  known as **resolvent** of operator  $\partial f$

## Moreau decomposition

- ▶ following relation always holds:

$$v = \text{prox}_f(v) + \text{prox}_{f^*}(v)$$

- ▶ main link between proximal operators and duality
- ▶ a generalization of orthogonal decomposition induced by subspace  $L$ :

$$v = \Pi_L(v) + \Pi_{L^\perp}(v)$$

follows from Moreau decomposition and  $(I_L)^* = I_{L^\perp}$

## Norms and norm balls

- ▶ **in general:** if  $f = \|\cdot\|$  and  $\mathcal{B}$  is unit ball of dual norm, then

$$\text{prox}_{\lambda f}(v) = v - \lambda \Pi_{\mathcal{B}}(v/\lambda)$$

by Moreau decomposition

- ▶ if  $f = \|\cdot\|_2$  and  $\mathcal{B}$  is the unit  $\ell_2$  ball, then

$$\Pi_{\mathcal{B}}(v) = \begin{cases} v/\|v\|_2 & \|v\|_2 > 1 \\ v & \|v\|_2 \leq 1 \end{cases}$$

$$\text{prox}_{\lambda f}(v) = \begin{cases} (1 - \lambda/\|v\|_2)v & \|v\|_2 \geq \lambda \\ 0 & \|v\|_2 < \lambda \end{cases}$$

sometimes called ‘block soft thresholding’ operator

## Norms and norm balls

- if  $f = \|\cdot\|_1$  and  $\mathcal{B}$  is the unit  $\ell_\infty$  ball, then

$$(\Pi_{\mathcal{B}}(v))_i = \begin{cases} 1 & v_i > 1 \\ v_i & |v_i| \leq 1 \\ -1 & v_i < -1 \end{cases}$$

lets us derive (elementwise) **soft thresholding**

$$\text{prox}_{\lambda f}(v) = (v - \lambda)_+ - (-v - \lambda)_+ = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases}$$

- if  $f = \|\cdot\|_\infty$  and  $\mathcal{B}$  is unit  $\ell_1$  ball, simple algorithms available

## Matrix functions

- ▶ suppose convex  $F : \mathbf{R}^{m \times n} \rightarrow \mathbf{R}$  is orthogonally invariant:

$$F(QX\tilde{Q}) = F(X)$$

for all orthogonal  $Q, \tilde{Q}$

- ▶ then  $F = f \circ \sigma$  and

$$\text{prox}_{\lambda F}(A) = U \text{diag}(\text{prox}_{\lambda f}(d)) V^T$$

where  $A = U \text{diag}(d) V^T$  is the SVD of  $A$  and  $\sigma(A) = d$

- ▶ e.g.,  $F = \|\cdot\|_*$  has  $f = \|\cdot\|_1$  so  $\text{prox}_{\lambda F}$  is 'singular value thresholding'

# Outline

Proximal operators

Proximal algorithms

Applications

Conclusions

## Proximal minimization

- ▶ the *proximal minimization* or *proximal point algorithm* is

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

- ▶ the simplest proximal method, can be interpreted as
  - gradient method applied to  $M_f$  rather than  $f$
  - simple iteration for finding a fixed point of  $\text{prox}_f$
- ▶ if  $f(x) = (1/2)x^T Ax - b^T x$ , reduces to iterative refinement for  $Ax = b$
- ▶ idea originates with Martinet, Moreau, Rockafellar (1970s)
- ▶ works for any fixed  $\lambda > 0$ , or for non-summable  $\lambda^k$

## Operator splitting

- ▶ the most useful proximal methods use the idea of **operator splitting**
- ▶ these algorithms minimize  $f + g$  only using  $\text{prox}_f$  or  $\text{prox}_g$
- ▶ useful when  $f$  and  $g$  each have useful structure separately
- ▶ very simple historical example: alternating projections to find  $x \in \mathcal{C} \cap \mathcal{D}$
- ▶ literature mostly from 1950s and 1970s

## Proximal gradient method

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

$f$  is smooth

$g : \mathbf{R}^n \rightarrow \mathbf{R} \cup \{+\infty\}$  is closed proper convex

- ▶ method:

$$x^{k+1} := \mathbf{prox}_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$

- ▶ converges with rate  $O(1/k)$  when  $\nabla f$  is Lipschitz continuous with constant  $L$  and step sizes are  $\lambda^k = \lambda \in (0, 1/L]$
- ▶ special case: projected gradient method (take  $g = I_C$ )
- ▶ traces back to Bruck, Lions, Mercier (1970s)

## Proximal gradient method

if  $L$  is not known (usually the case), can use the following line search:

---

**given**  $x^k$ ,  $\lambda^{k-1}$ , and parameter  $\beta \in (0, 1)$ .

Let  $\lambda := \lambda^{k-1}$ .

**repeat**

1. Let  $z := \text{prox}_{\lambda g}(x^k - \lambda \nabla f(x^k))$ .
2. **break if**  $f(z) \leq \hat{f}_\lambda(z, x^k)$ .
3. Update  $\lambda := \beta \lambda$ .

**return**  $\lambda^k := \lambda$ ,  $x^{k+1} := z$ .

---

typical value of  $\beta$  is  $1/2$ , and

$$\hat{f}_\lambda(x, y) = f(y) + \nabla f(y)^T(x - y) + (1/2\lambda)\|x - y\|_2^2$$

## Interpretations

- ▶  $x^+$  is solution to

$$\text{minimize} \quad (1/2) \|x^+ - (x - \lambda \nabla f(x))\|_2^2 + \lambda g(x^+)$$

trade off between minimizing  $g$  and being close to gradient step for  $f$

- ▶ majorization-minimization method for  $f + g$ :
  - keep minimizing convex upper bound to objective tight at previous iterate
  - here, use  $\hat{f}_\lambda(x, x^k) + g(x)$  as upper bound (when  $\lambda \in (0, 1/L]$ )

- ▶  $0 \in \nabla f(x^*) + \partial g(x^*)$  if and only if

$$x^* = (I + \lambda \partial g)^{-1}(I - \lambda \nabla f)(x^*)$$

i.e.,  $x^*$  is a fixed point of *forward-backward operator*

## Accelerated proximal gradient method

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

$f$  is smooth

$g : \mathbf{R}^n \rightarrow \mathbf{R} \cup \{+\infty\}$  is closed proper convex

- ▶ method:

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

$$x^{k+1} := \text{prox}_{\lambda^k g} (y^{k+1} - \lambda^k \nabla f(y^{k+1}))$$

works for  $\omega^k = k/(k + 3)$  and similar line search as before

- ▶ faster  $O(1/k^2)$  convergence rate, originated with Nesterov (1983)

## ADMM

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

$f, g : \mathbf{R}^n \rightarrow \mathbf{R} \cup \{+\infty\}$  are closed proper convex

- ▶ method:

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

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

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

- ▶ basically, always works and has  $O(1/k)$  rate in general
- ▶ if  $f$  and  $g$  are both indicators, get a variation on alternating projections
- ▶ originates from Gabay, Mercier, Glowinski, Marrocco in 1970s

# Outline

Proximal operators

Proximal algorithms

Applications

Conclusions

Applications

24

## Lasso

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

with  $A \in \mathbf{R}^{m \times n}$  and  $\gamma > 0$

- ▶ proximal gradient method (similar for accelerated):

$$x^{k+1} := \text{prox}_{\lambda^k \gamma \|\cdot\|_1}(x^k - \lambda^k A^T(Ax^k - b))$$

- ▶ faster implementations:

- parallel matrix-vector multiplication
- if  $n \ll m$ , precompute  $A^T A$  and  $A^T b$ , then solve smaller lasso problem with  $\tilde{A} = (A^T A)^{1/2}$ ,  $\tilde{b} = A^T b$ ; effort is then mostly computing  $\tilde{A}$ ,  $\tilde{b}$
- compute  $A^T A$  and  $A^T b$  in parallel with one  $a_i$  in memory at a time
- compute entire regularization path with warm starting

- ▶ can easily generalize to group lasso, elastic net, GLMs, ...

## Lasso

- ▶ ADMM:

$$\begin{aligned}x^{k+1} &:= (I + \lambda A^T A)^{-1}(z^k - u^k - \lambda A^T b) \\ z^{k+1} &:= \text{prox}_{\lambda \gamma \|\cdot\|_1}(x^{k+1} + u^k) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1}\end{aligned}$$

- ▶ all the effort is in  $x$ -update (solve linear system)
- ▶ faster implementations:
  - if  $A$  is fat, use matrix inversion lemma in  $x$ -update
  - if using direct method, use factorization caching
  - if using iterative method, warm start
  - if  $n \ll m$ , precompute  $A^T A$  and  $A^T b$  (possibly in parallel)

## Lasso

Method	Iterations	Time (s)	$p^*$	Error (abs)	Error (rel)
CVX	15	26.53	16.5822	—	—
Proximal gradient	127	0.72	16.5835	0.09	0.01
Accelerated	23	0.15	16.6006	0.26	0.04
ADMM	20	0.07	16.6011	0.18	0.03

## Distributed lasso example

- ▶ example with **dense**  $A \in \mathbf{R}^{400000 \times 8000}$  (roughly 30 GB of data)
  - distributed solver written in C using MPI and GSL
  - no optimization or tuned libraries (like ATLAS, MKL)
  - split into 80 subsystems across 10 (8-core) machines on Amazon EC2
- ▶ computation times

loading data	30s
factorization	5m
subsequent ADMM iterations	0.5–2s
lasso solve (about 15 ADMM iterations)	5–6m

## Matrix decomposition

- ▶ decompose matrix  $A$  into sum of ‘simple’ components:

$$\begin{aligned} & \text{minimize} && \varphi_1(X_1) + \gamma_2\varphi_2(X_2) + \cdots + \gamma_N\varphi_N(X_N) \\ & \text{subject to} && A = X_1 + X_2 + \cdots + X_N \end{aligned}$$

- ▶ penalty functions can include
  - squared Frobenius norm (make  $X_i$  small)
  - entrywise  $\ell_1$  norm (make  $X_i$  sparse)
  - sum-{row,column} norm (row/column sparsity)
  - indicator of elementwise constraints (e.g., known values, bounds)
  - indicator of semidefinite cone
  - nuclear norm (make  $X_i$  low rank)
- ▶ common example: decompose  $A = \text{sparse} + \text{low rank}$

## Matrix decomposition via ADMM

- ▶ splitting:

$$f(X) = \sum_{i=1}^N \varphi_i(X_i), \quad g(X) = I_{\mathcal{C}}(X)$$

with

$$X = (X_1, \dots, X_N), \quad \mathcal{C} = \{(X_1, \dots, X_N) \mid X_1 + \dots + X_N = A\}$$

- ▶ ADMM simplifies to:

$$X_i^{k+1} := \text{prox}_{\lambda \varphi_i}(X_i^k - \bar{X}^k + (1/N)A - U^k)$$

$$U^{k+1} := U^k + \bar{X}^{k+1} - (1/N)A$$

## Matrix decomposition results

problem: decompose  $A = \text{rank } 4 + \text{sparse} + \text{small Gaussian noise}$

Method	$m$	$n$	Iterations	Time (s)
CVX	10	30	15	1.11
ADMM	10	30	45	0.02
CVX	20	50	17	2.54
ADMM	20	50	42	0.03
CVX	40	80	20	108.14
ADMM	40	80	36	0.07
ADMM	100	200	38	0.58
ADMM	500	1000	42	35.56

note: last instance has 1.5M variables and 500K constraints

## Multi-period portfolio optimization

$$\text{minimize} \quad \sum_{t=1}^T f_t(x_t) + \sum_{t=1}^T g_t(x_t - x_{t-1})$$

$f_t$  is risk-adjusted negative return in period  $t$  ('stage costs')

$g_t$  is transaction costs

- ▶ splitting: put stage costs in one term and transaction costs in the other
- ▶  $f$  is separable across time steps,  $g$  is separable across assets
- ▶  $\text{prox}_f$  involves solving  $T$  single-period problems in parallel
- ▶  $\text{prox}_g$  involves  $n$  problems in parallel that can be solved in  $O(T)$  flops

## Multi-period portfolio optimization

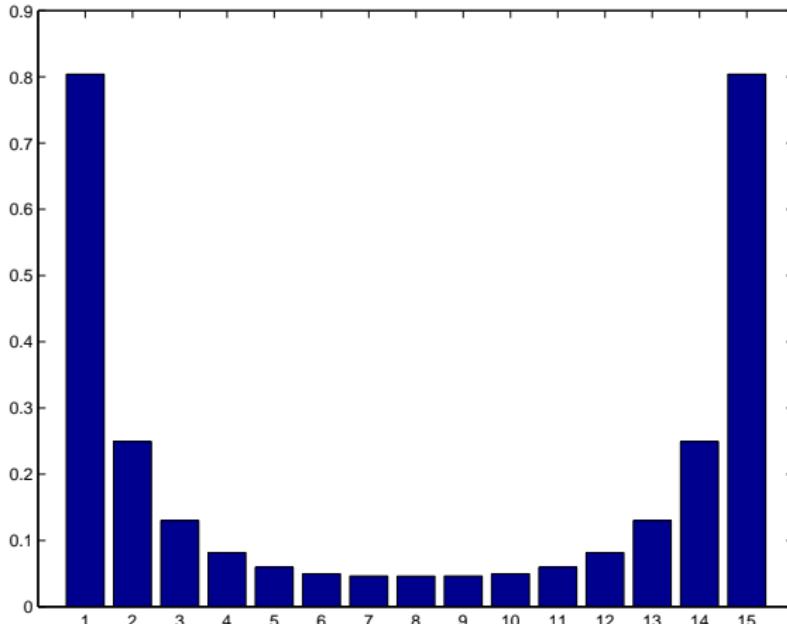


Figure: Time series of  $\ell_1$  deviation from  $x^{\text{static}}$

## Multi-period portfolio optimization

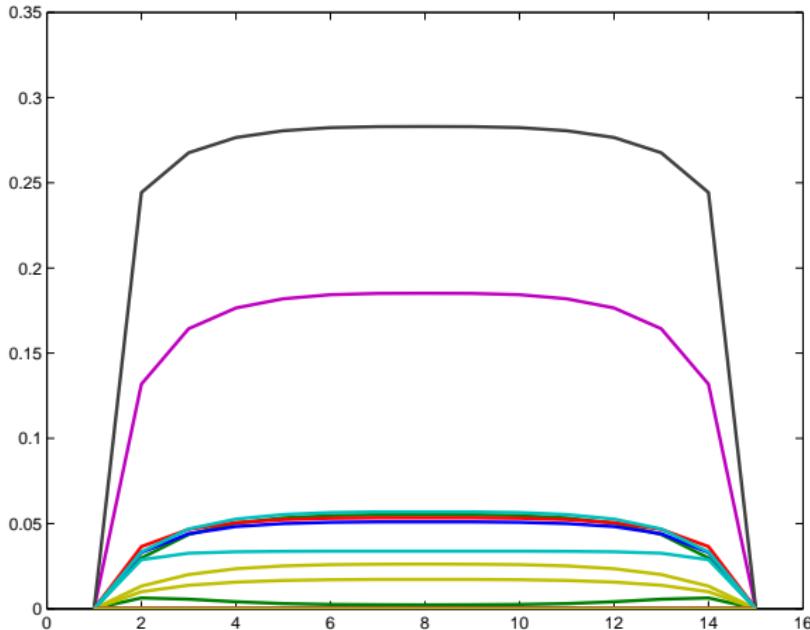


Figure: Time series of asset holdings

# Outline

Proximal operators

Proximal algorithms

Applications

Conclusions

## Conclusions

key ingredients:

1. operator splitting
2. proximal operators

often, use problem transformations so when an operator splitting method is used, each part of the problem has simple, small prox operators

# **Monotone Operators**

Stephen Boyd (with help from Neal Parikh)

EE364b, Stanford University

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

# Relations

- a *relation*  $R$  on a set  $\mathbf{R}^n$  is a subset of  $\mathbf{R}^n \times \mathbf{R}^n$
- $\text{dom } R = \{x \mid \exists y (x, y) \in R\}$
- overload  $R(x)$  to mean the set  $R(x) = \{y \mid (x, y) \in R\}$
- can think of  $R$  as ‘set-valued mapping’, i.e., from  $\text{dom } R$  into  $2^{\mathbf{R}^n}$
- when  $R(x)$  is always empty or a singleton, we say  $R$  is a function
- any function (or operator)  $f : C \rightarrow \mathbf{R}^n$  with  $C \subseteq \mathbf{R}^n$  is a relation ( $f(x)$  is then ambiguous: it can mean  $f(x)$  or  $\{f(x)\}$ )

## Examples

- empty relation:  $\emptyset$
- full relation:  $\mathbf{R}^n \times \mathbf{R}^n$
- identity:  $I = \{(x, x) \mid x \in \mathbf{R}^n\}$
- zero:  $0 = \{(x, 0) \mid x \in \mathbf{R}^n\}$
- $\{x \in \mathbf{R}^2 \mid x_1^2 + x_2^2 = 1\}$
- $\{x \in \mathbf{R}^2 \mid x_1 \leq x_2\}$
- *subdifferential relation:*  $\partial f = \{(x, \partial f(x)) \mid x \in \mathbf{R}^n\}$

## Operations on relations

- *inverse* (relation):  $R^{-1} = \{(y, x) \mid (x, y) \in R\}$ 
  - inverse exists for any relation
  - coincides with inverse function, when inverse function exists
- *composition*:  $RS = \{(x, y) \mid \exists z (x, z) \in S, (z, y) \in R\}$
- *scalar multiplication*:  $\alpha R = \{(x, \alpha y) \mid (x, y) \in R\}$
- *addition*:  $R + S = \{(x, y + z) \mid (x, y) \in R, (x, z) \in S\}$

## Example: Resolvent of operator

for relation  $R$  and  $\lambda \in \mathbf{R}$ , *resolvent* (much more on this later) is relation

$$S = (I + \lambda R)^{-1}$$

- $I + \lambda R = \{(x, x + \lambda y) \mid (x, y) \in R\}$
- $S = (I + \lambda R)^{-1} = \{(x + \lambda y, x) \mid (x, y) \in R\}$
- for  $\lambda \neq 0$ ,  $S = \{(u, v) \mid (u - v)/\lambda \in R(v)\}$

## Generalized equations

- goal: solve *generalized equation*  $0 \in R(x)$
- i.e., find  $x \in \mathbf{R}^n$  with  $(x, 0) \in R$
- *solution set* or *zero set* is  $X = \{x \in \text{dom } R \mid 0 \in R(x)\}$
- if  $R = \partial f$  and  $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$ , then  $0 \in R(x)$  means  $x$  minimizes  $f$

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

## Monotone operators

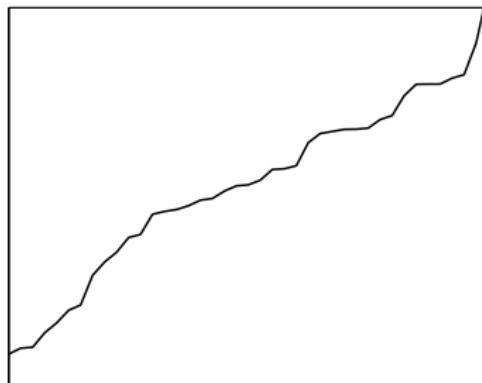
- relation  $F$  on  $\mathbf{R}^n$  is *monotone* if

$$(u - v)^T(x - y) \geq 0 \quad \text{for all } (x, u), (y, v) \in F$$

- $F$  is *maximal monotone* if there is no monotone operator that properly contains it
- we'll be informal (*i.e.*, sloppy) about maximality, other analysis issues
- solving generalized equations with maximal monotone operators  
subsumes many useful problems

## Maximal monotone operators on R

$F$  is maximal monotone iff it is a connected curve with no endpoints, with nonnegative (or infinite) slope



## Some basic properties

suppose  $F$  and  $G$  are monotone

- *sum*:  $F + G$  is monotone
- *nonnegative scaling*: if  $\alpha \geq 0$ , then  $\alpha F$  is monotone
- *inverse*:  $F^{-1}$  is monotone
- *congruence*: for  $T \in \mathbf{R}^{n \times m}$ ,  $T^T F(Tz)$  is monotone (on  $\mathbf{R}^m$ )
- *zero set*:  $\{x \in \mathbf{R}^n \mid 0 \in F(x)\}$  is convex if  $F$  is maximal monotone

affine function  $F(x) = Ax + b$  is monotone iff  $A + A^T \succeq 0$

## Subdifferential

$F(x) = \partial f(x)$  is monotone

- suppose  $u \in \partial f(x)$  and  $v \in \partial f(y)$
- then

$$f(y) \geq f(x) + u^T(y - x), \quad f(x) \geq f(y) + v^T(x - y)$$

- add these and cancel  $f(y) + f(x)$  to get

$$0 \leq (u - v)^T(x - y)$$

if  $f$  is convex closed proper (CCP) then  $F(x) = \partial f(x)$  is maximal monotone

## KKT operator

- equality-constrained convex problem (with  $A \in \mathbf{R}^{m \times n}$ )

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & Ax = b\end{array}$$

with Lagrangian  $L(x, y) = f(x) + y^T(Ax - b)$

- associated *KKT operator* on  $\mathbf{R}^n \times \mathbf{R}^m$ :

$$F(x, y) = \begin{bmatrix} \partial_x L(x, y) \\ -\partial_y L(x, y) \end{bmatrix} = \begin{bmatrix} \partial f(x) + A^T y \\ b - Ax \end{bmatrix} = \begin{bmatrix} r^{\text{dual}} \\ -r^{\text{pri}} \end{bmatrix}$$

- zero set of  $F$  is set of primal-dual optimal points (saddle points of  $L$ )
- KKT operator is monotone: write as sum of monotone operators

$$F(x, y) = \begin{bmatrix} \partial f(x) \\ b \end{bmatrix} + \begin{bmatrix} 0 & A^T \\ -A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

## Multiplier to residual mapping

- same equality-constrained convex problem
- define  $F(y) = b - Ax$  with  $x \in \operatorname{argmin}_z L(z, y)$  (can be set-valued)
- $-F(y)$  is primal residual obtained from dual variable  $y$
- interpretation:  $F(y) = \partial(-g)(y)$ , where  $g$  is dual function
- zero set is set of dual optimal points
- multiplier to residual mapping  $F$  is monotone
- quick proof:  $F(y) = b - A(\partial f)^{-1}(-A^T y)$  (or use  $F(y) = \partial(-g)(y)$ )

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

## Nonexpansive and contractive operators

- $F$  has *Lipschitz constant*  $L$  if

$$\|F(x) - F(y)\|_2 \leq L\|x - y\|_2 \quad \text{for all } x, y \in \text{dom } F$$

- for  $L = 1$ , we say  $F$  is *nonexpansive*
- for  $L < 1$ , we say  $F$  is a *contraction* (with contraction factor  $L$ )

# Properties

- if  $F$  and  $G$  have Lipschitz constant  $L$ , so does

$$\theta F + (1 - \theta)G, \quad \theta \in [0, 1]$$

- composition of nonexpansive operators is nonexpansive
- composition of nonexpansive operator and contraction is contraction
- *fixed point set* of nonexpansive  $F$  (with  $\text{dom } f = \mathbf{R}^n$ )

$$\{x \mid F(x) = x\}$$

is convex (but can be empty)

- a contraction has a single fixed point (more later)

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

## Resolvent and Cayley operator

- for  $\lambda \in \mathbf{R}$ , *resolvent* of relation  $F$  is

$$R = (I + \lambda F)^{-1}$$

- when  $\lambda \geq 0$  and  $F$  monotone,  $R$  is nonexpansive (thus a function)
- when  $\lambda \geq 0$  and  $F$  maximal monotone,  $\text{dom } R = \mathbf{R}^n$
- *Cayley operator* of  $F$  is

$$C = 2R - I = 2(I + \lambda F)^{-1} - I$$

- when  $\lambda \geq 0$  and  $F$  monotone,  $C$  is nonexpansive
- we write  $R_F$  and  $C_F$  to explicitly show dependence on  $F$

## Proof that $C$ is nonexpansive

assume  $\lambda > 0$  and  $F$  monotone

- suppose  $(x, u) \in R$  and  $(y, v) \in R$ , i.e.,

$$u + \lambda F(u) \ni x, \quad v + \lambda F(v) \ni y$$

- subtract to get  $u - v + \lambda(F(u) - F(v)) \ni x - y$
- multiply by  $(u - v)^T$  and use monotonicity of  $F$  to get

$$\|u - v\|_2^2 \leq (x - y)^T (u - v)$$

- so when  $x = y$ , we must have  $u = v$  (i.e.,  $R$  is a function)

## Proof (continued)

- now let's show  $C$  is nonexpansive:

$$\begin{aligned}\|C(x) - C(y)\|_2^2 &= \|(2u - x) - (2v - y)\|_2^2 \\ &= \|2(u - v) - (x - y)\|_2^2 \\ &= 4\|u - v\|_2^2 - 4(u - v)^T(x - y) + \|x - y\|_2^2 \\ &\leq \|x - y\|_2^2\end{aligned}$$

using inequality above

- $R$  is nonexpansive since it is the average of  $I$  and  $C$ :

$$R = (1/2)I + (1/2)(2R - I)$$

## Example: Linear operators

- linear mapping  $F(x) = Ax$  is
  - monotone iff  $A + A^T \succeq 0$
  - nonexpansive iff  $\|A\|_2 \leq 1$
- $\lambda \geq 0$  and  $A + A^T \succeq 0 \implies$ 
  - $I + \lambda A$  nonsingular
  - $\|R_A\|_2 = \|(I + \lambda A)^{-1}\|_2 \leq 1$
  - $\|C_A\|_2 = \|2(I + \lambda A)^{-1} - I\|_2 \leq 1$
- for matrix case, we have alternative formula for Cayley operator:

$$2(I + \lambda A)^{-1} - I = (I + \lambda A)^{-1}(I - \lambda A)$$

cf. bilinear function  $\frac{1 - \lambda a}{1 + \lambda a}$ , which maps

$$\{s \in \mathbf{C} \mid \Re s \geq 0\} \quad \text{into} \quad \{s \in \mathbf{C} \mid |s| \leq 1\}$$

## Resolvent of subdifferential: Proximal mapping

- suppose  $z = (I + \lambda \partial f)^{-1}(x)$ , with  $\lambda > 0$ ,  $f$  convex
- this means  $z + \lambda \partial f(z) \ni x$
- rewrite as

$$0 \in \partial_z (f(z) + (1/2\lambda) \|z - x\|_2^2)$$

which is the same as

$$z = \operatorname{argmin}_u (f(u) + (1/2\lambda) \|u - x\|_2^2)$$

- RHS called *proximal mapping* of  $f$ , denoted  $\text{prox}_{\lambda f}(x)$

## Example: Indicator function

- take  $f = I_C$ , indicator function of convex set  $C$
- $\partial f$  is the *normal cone operator*

$$N_C(x) = \begin{cases} \emptyset & x \notin C \\ \{w \mid w^T(z - x) \leq 0 \quad \forall z \in C\} & x \in C \end{cases}$$

- proximal operator of  $f$  (i.e., resolvent of  $N_C$ ) is

$$(I + \lambda \partial I_C)^{-1}(x) = \operatorname{argmin}_u (I_C(u) + (1/2\lambda) \|u - x\|_2^2) = \Pi_C(x)$$

where  $\Pi_C$  is (Euclidean) projection onto  $C$

## Resolvent of multiplier to residual map

- take  $F$  to be multiplier to residual mapping for convex problem

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & Ax = b\end{array}$$

- $F(y) = b - Ax$  where  $x \in \operatorname{argmin}_w L(w, y)$
- $z = (I + \lambda F)^{-1}(y)$  means  $z + \lambda F(z) \ni y$
- $z + \lambda(b - Ax) = y$  for some  $x \in \operatorname{argmin}_w L(w, z)$
- write as

$$z = y + \lambda(Ax - b), \quad \partial f(x) + A^T z \ni 0$$

## Resolvent of multiplier to residual map

- write second term as  $\partial f(x) + A^T y + \lambda A^T(Ax - b) \ni 0$ , so

$$x \in \operatorname{argmin}_w (f(w) + y^T(Aw - b) + (\lambda/2)\|Aw - b\|_2^2)$$

- function on right side is *augmented Lagrangian* for the problem
- so  $z = R(y)$  can be found as

$$\begin{aligned} x &:= \operatorname{argmin}_w (f(w) + y^T(Aw - b) + (\lambda/2)\|Aw - b\|_2^2) \\ z &:= y + \lambda(Ax - b) \end{aligned}$$

## Fixed points of Cayley and resolvent operators

- assume  $F$  is maximal monotone,  $\lambda > 0$
- solutions of  $0 \in F(x)$  are fixed points of  $R$ :

$$F(x) \ni 0 \iff x + \lambda F(x) \ni x \iff x = (I + \lambda F)^{-1}(x) = R(x)$$

- solutions of  $0 \in F(x)$  are fixed points of  $C$ :

$$x = R(x) \iff x = 2R(x) - x = C(x)$$

- key result: we can solve  $0 \in F(x)$  by finding fixed points of  $C$  or  $R$
- next: how to actually find these fixed points

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

## Contraction mapping theorem

- also known as *Banach fixed point theorem*
- assume  $F$  is contraction, with Lipschitz constant  $L < 1$ ,  $\text{dom } F = \mathbf{R}^n$
- the iteration

$$x^{k+1} := F(x^k)$$

converges to the unique fixed point of  $F$

- proof (sketch):
  - sequence  $x^k$  is Cauchy:  $\|x^{k+m} - x^k\|_2 \leq \|x^{k+1} - x^k\|_2 / (1 - L)$
  - hence converges to a point  $x^*$
  - $x^*$  must be (the) fixed point

## Example: Gradient method

- assume  $f$  is convex,  $mI \preceq \nabla^2 f(x) \preceq LI$   
(i.e.,  $f$  strongly convex,  $\nabla f$  Lipschitz)
- gradient method is

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

(fixed points are exactly solutions of  $F(x) = x$ )

- $DF(x) = I - \alpha \nabla^2 f(x)$
- $F$  is a Lipschitz with parameter  $\max\{|1 - \alpha m|, |1 - \alpha L|\}$
- $F$  is a contraction when  $0 < \alpha < 2/L$
- so gradient method converges (geometrically) when  $0 < \alpha < 2/L$

## Damped iteration of a nonexpansive operator

- suppose  $F$  is nonexpansive,  $\text{dom } F = \mathbf{R}^n$ , with fixed point set  $X = \{x \mid F(x) = x\}$
- can have  $X = \emptyset$  (e.g., translation)
- simple iteration of  $F$  need not converge, even when  $X \neq \emptyset$  (e.g., rotation)
- *damped* iteration:

$$x^{k+1} := (1 - \theta^k)x^k + \theta^k F(x^k)$$

$$\theta^k \in (0, 1)$$

- important special case:  $\theta^k = 1/2$  (more later)
- another special case:  $\theta^k = 1/(k + 1)$ , which gives simple averaging

$$x^k = \frac{1}{k + 1} (x^0 + \cdots + F(x^{k-2}) + F(x^{k-1}))$$

## Convergence results

- assume  $F$  is nonexpansive,  $\text{dom } F = \mathbf{R}^n$ ,  $X \neq \emptyset$ , and

$$\sum_{k=0}^{\infty} \theta^k (1 - \theta^k) = \infty$$

(which holds for special cases above)

- then we have

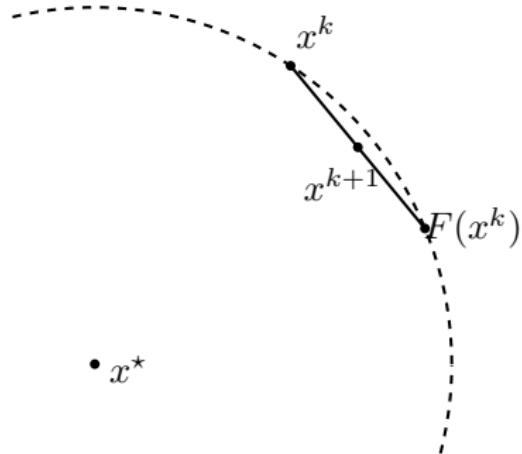
$$\min_{j=0,\dots,k} \mathbf{dist}(x^j, X) \rightarrow 0$$

i.e., (some) iterates get arbitrarily close to fixed point set, and

$$\min_{j=0,\dots,k} \|F(x^j) - x^j\|_2 \rightarrow 0$$

i.e., (some) iterates yield arbitrarily good ‘almost fixed points’

## Idea of proof



- $F(x^k)$  is no farther from  $x^*$  than  $x^k$  is (by nonexpansivity)
- so  $x^{k+1}$  is closer to  $x^*$  than  $x^k$  is

## Proof

- start with identity

$$\|\theta a + (1 - \theta)b\|_2^2 = \theta\|a\|_2^2 + (1 - \theta)\|b\|_2^2 - \theta(1 - \theta)\|b - a\|_2^2$$

- apply to  $x^{k+1} - x^* = (1 - \theta^k)(x^k - x^*) + \theta^k(F(x^k) - x^*)$ :

$$\begin{aligned}\|x^{k+1} - x^*\|_2^2 &= (1 - \theta^k)\|x^k - x^*\|_2^2 + \theta^k\|F(x^k) - x^*\|_2^2 - \theta^k(1 - \theta^k)\|F(x^k) - x^k\|_2^2 \\ &\leq \|x^k - x^*\|_2^2 - \theta^k(1 - \theta^k)\|F(x^k) - x^k\|_2^2\end{aligned}$$

using  $\|F(x^k) - x^*\|_2 \leq \|x^k - x^*\|_2$

## Proof (continued)

- iterate inequality to get

$$\sum_{j=0}^k \theta^j (1 - \theta^j) \|F(x^j) - x^j\|_2^2 \leq \|x^0 - x^*\|_2^2 - \|x^{k+1} - x^*\|_2^2$$

- if  $\|F(x^j) - x^j\|_2 \geq \epsilon$  for  $j = 0, \dots, k$ , then

$$\epsilon^2 \leq \frac{\|x^0 - x^*\|_2^2}{\sum_{j=0}^k \theta^j (1 - \theta^j)}$$

- RHS goes to zero as  $k \rightarrow \infty$

# Outline

- ① Relations
- ② Monotone operators
- ③ Nonexpansive and contractive operators
- ④ Resolvent and Cayley operator
- ⑤ Fixed point iterations
- ⑥ Proximal point algorithm and method of multipliers

## Damped Cayley iteration

- want to solve  $0 \in F(x)$  with  $F$  maximal monotone
- damped Cayley iteration:

$$\begin{aligned}x^{k+1} &:= (1 - \theta^k)x^k + \theta^k C(x^k) \\&= (1 - \theta^k)x^k + \theta^k(2R(x^k) - I(x^k)) \\&= (1 - 2\theta^k)x^k + 2\theta^k R(x^k)\end{aligned}$$

with  $\theta^k \in (0, 1)$  and  $\sum_k \theta^k(1 - \theta^k) = \infty$

- converges (assuming  $X \neq \emptyset$ ) in sense given above
- important: requires ability to evaluate resolvent map of  $F$

## Proximal point algorithm

- take  $\theta^k = 1/2$  in damped Cayley iteration
- gives *resolvent iteration* or *proximal point algorithm*:

$$x^{k+1} := R(x^k) = (I + \lambda F)^{-1}(x^k)$$

- if  $F = \partial f$  with  $f$  convex, yields *proximal minimization algorithm*

$$x^{k+1} := \text{prox}_{f, 1/\lambda}(x^k) = \operatorname{argmin}_x (f(x) + (1/2\lambda)\|x - x^k\|_2^2)$$

can interpret as quadratic regularization that goes away in limit

- many classical algorithms are just proximal point method applied to appropriate maximal monotone operator

## Method of multipliers

- take  $F$  to be multiplier to residual mapping for

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & Ax = b\end{array}$$

- $F(y) = b - Ax$  with  $x \in \operatorname{argmin}_z L(z, y)$
- proximal point algorithm becomes *method of multipliers*:

$$\begin{aligned}x^{k+1} &:= \operatorname{argmin}_w (f(w) + (y^k)^T(Aw - b) + (\lambda/2)\|Aw - b\|_2^2) \\ y^{k+1} &:= y^k + \lambda(Ax^{k+1} - b)\end{aligned}$$

## Method of multipliers

- first step is augmented Lagrangian minimization
- second step is dual variable update
- $y^k$  converges to an optimal dual variable
- primal residual  $Ax^k - b$  converges to zero

## Method of multipliers dual update

- optimality conditions (primal and dual feasibility):

$$Ax - b = 0, \quad \partial f(x) + A^T y \ni 0$$

- from definition of  $x^{k+1}$  we have

$$\begin{aligned} 0 &\in \partial f(x^{k+1}) + A^T y^k + \lambda A^T (Ax^{k+1} - b) \\ &= \partial f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

- so dual update makes  $(x^{k+1}, y^{k+1})$  dual feasible
- primal feasibility occurs in limit as  $k \rightarrow \infty$

## Comparison with dual (sub)gradient method

method of multipliers

- like dual method, but with augmented Lagrangian, specific step size
- converges under *far more general* conditions than dual subgradient
- $f$  need not be strictly convex, or differentiable
- $f$  can take on value  $+\infty$
- but not amenable to decomposition (more later . . . )

# **Monotone Operator Splitting Methods**

Stephen Boyd (with help from Neal Parikh and Eric Chu)

EE364b, Stanford University

# Outline

- ① Operator splitting
- ② Douglas-Rachford splitting
- ③ Consensus optimization

# Operator splitting

- want to solve  $0 \in F(x)$  with  $F$  maximal monotone
- *main idea:* write  $F$  as  $F = A + B$ , with  $A$  and  $B$  maximal monotone
- called *operator splitting*
- solve using methods that require evaluation of resolvents

$$R_A = (I + \lambda A)^{-1}, \quad R_B = (I + \lambda B)^{-1}$$

(or Cayley operators  $C_A = 2R_A - I$  and  $C_B = 2R_B - I$ )

- useful when  $R_A$  and  $R_B$  can be evaluated more easily than  $R_F$

## Main result

- $A, B$  maximal monotone, so Cayley operators  $C_A, C_B$  nonexpansive
- hence  $C_A C_B$  nonexpansive
- key result:

$$0 \in A(x) + B(x) \iff C_A C_B(z) = z, \quad x = R_B(z)$$

- so solutions of  $0 \in A(x) + B(x)$  can be found from fixed points of nonexpansive operator  $C_A C_B$

## Proof of main result

- write  $C_A C_B(z) = z$  and  $x = R_B(z)$  as

$$x = R_B(z), \quad \tilde{z} = 2x - z, \quad \tilde{x} = R_A(\tilde{z}), \quad z = 2\tilde{x} - \tilde{z}$$

- subtract 2nd & 4th equations to conclude  $x = \tilde{x}$
- 4th equation is then  $2x = \tilde{z} + z$
- now add  $x + \lambda B(x) \ni z$  and  $x + \lambda A(x) \ni \tilde{z}$  to get

$$2x + \lambda(A(x) + B(x)) \ni \tilde{z} + z = 2x$$

- hence  $A(x) + B(x) \ni 0$
- argument goes other way (but we don't need it)

# Outline

- ① Operator splitting
- ② Douglas-Rachford splitting
- ③ Consensus optimization

## Peaceman-Rachford and Douglas-Rachford splitting

- *Peaceman-Rachford splitting* is undamped iteration

$$z^{k+1} = C_A C_B(z^k)$$

doesn't converge in general case; need  $C_A$  or  $C_B$  to be contraction

- *Douglas-Rachford splitting* is damped iteration

$$z^{k+1} := (1/2)(I + C_A C_B)(z^k)$$

always converges when  $0 \in A(x) + B(x)$  has solution

- these methods trace back to the mid-1950s (!!)

## Douglas-Rachford splitting

write D-R iteration  $z^{k+1} := (1/2)(I + C_A C_B)(z^k)$  as

$$\begin{aligned}x^{k+1/2} &:= R_B(z^k) \\z^{k+1/2} &:= 2x^{k+1/2} - z^k \\x^{k+1} &:= R_A(z^{k+1/2}) \\z^{k+1} &:= z^k + x^{k+1} - x^{k+1/2}\end{aligned}$$

last update follows from

$$\begin{aligned}z^{k+1} &:= (1/2)(2x^{k+1} - z^{k+1/2}) + (1/2)z^k \\&= x^{k+1} - (1/2)(2x^{k+1/2} - z^k) + (1/2)z^k \\&= z^k + x^{k+1} - x^{k+1/2}\end{aligned}$$

- can consider  $x^{k+1} - x^{k+1/2}$  as a residual
- $z^k$  is running sum of residuals

## Douglas-Rachford algorithm

- *many* ways to rewrite/rearrange D-R algorithm
- equivalent to many other algorithms; often not obvious
- need very little:  $A, B$  maximal monotone; solution exists
- $A$  and  $B$  are handled separately (via  $R_A$  and  $R_B$ ); they are ‘uncoupled’

## Alternating direction method of multipliers

to minimize  $f(x) + g(x)$ , we solve  $0 \in \partial f(x) + \partial g(x)$

with  $A(x) = \partial g(x)$ ,  $B(x) = \partial f(x)$ , D-R is

$$\begin{aligned}x^{k+1/2} &:= \operatorname{argmin}_x \left( f(x) + (1/2\lambda) \|x - z^k\|_2^2 \right) \\z^{k+1/2} &:= 2x^{k+1/2} - z^k \\x^{k+1} &:= \operatorname{argmin}_x \left( g(x) + (1/2\lambda) \|x - z^{k+1/2}\|_2^2 \right) \\z^{k+1} &:= z^k + x^{k+1} - x^{k+1/2}\end{aligned}$$

a special case of the *alternating direction method of multipliers* (ADMM)

# Constrained optimization

- constrained convex problem:

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & x \in C\end{array}$$

- take  $B(x) = \partial f(x)$  and  $A(x) = \partial I_C(x) = N_C(x)$
- so  $R_B(z) = \mathbf{prox}_f(z)$  and  $R_A(z) = \Pi_C(z)$
- D-R is

$$\begin{aligned}x^{k+1/2} &:= \mathbf{prox}_f(z^k) \\z^{k+1/2} &:= 2x^{k+1/2} - z^k \\x^{k+1} &:= \Pi_C(z^{k+1/2}) \\z^{k+1} &:= z^k + x^{k+1} - x^{k+1/2}\end{aligned}$$

## Dykstra's alternating projections

- find a point in the intersection of convex sets  $C, D$
- D-R gives algorithm

$$\begin{aligned}x^{k+1/2} &:= \Pi_C(z^k) \\z^{k+1/2} &:= 2x^{k+1/2} - z^k \\x^{k+1} &:= \Pi_D(z^{k+1/2}) \\z^{k+1} &:= z^k + x^{k+1} - x^{k+1/2}\end{aligned}$$

- this is *Dykstra's alternating projections algorithm*
- much faster than classical alternating projections  
(e.g., for  $C, D$  polyhedral, converges in finite number of steps)

## Positive semidefinite matrix completion

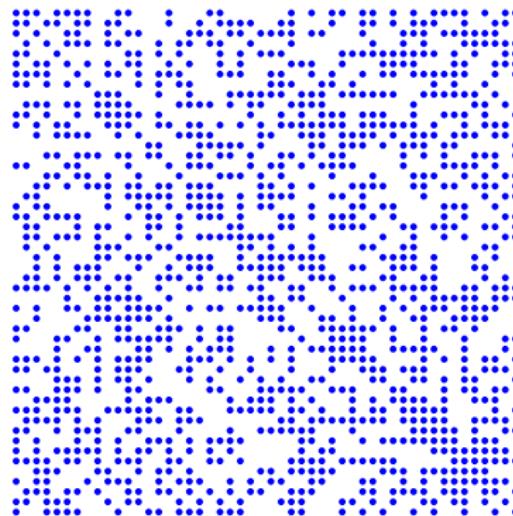
- some entries of matrix in  $\mathbf{S}^n$  known; find values for others so completed matrix is PSD
- $C = \mathbf{S}_+^n$ ,  $D = \{X \mid X_{ij} = X_{ij}^{\text{known}}, (i, j) \in \mathcal{K}\}$
- projection onto  $C$ : find eigendecomposition  $X = \sum_{i=1}^n \lambda_i q_i q_i^T$ ; then

$$\Pi_C(X) = \sum_{i=1}^n \max\{0, \lambda_i\} q_i q_i^T$$

- projection onto  $D$ : set specified entries to known values

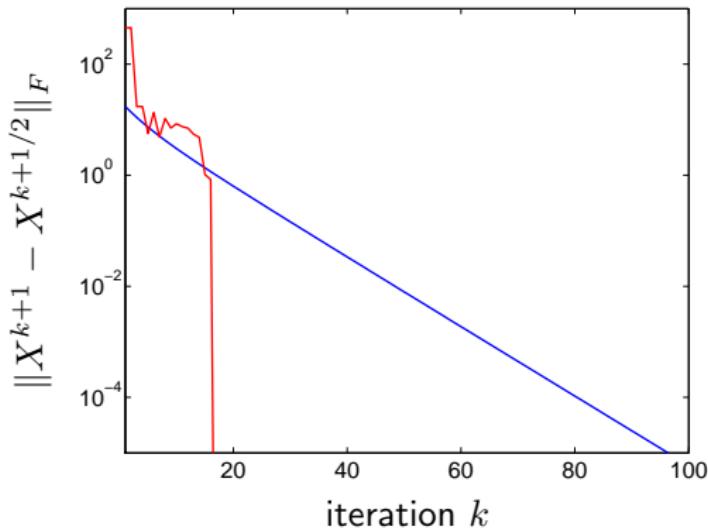
## Positive semidefinite matrix completion

specific example:  $50 \times 50$  matrix missing about half of its entries



- initialize  $Z^0 = 0$

# Positive semidefinite matrix completion



- blue: alternating projections; red: D-R
- $X^{k+1/2} \in C, X^{k+1} \in D$

# Outline

- ① Operator splitting
- ② Douglas-Rachford splitting
- ③ Consensus optimization

## Consensus optimization

- want to minimize  $\sum_{i=1}^N f_i(x)$
- rewrite as *consensus problem*

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^N f_i(x_i) \\ & \text{subject to} && x \in C = \{(x_1, \dots, x_N) \mid x_1 = \dots = x_N\} \end{aligned}$$

- D-R consensus optimization:

$$\begin{aligned} x^{k+1/2} &:= \mathbf{prox}_f(z^k) \\ z^{k+1/2} &:= 2x^{k+1/2} - z^k \\ x^{k+1} &:= \Pi_C(z^{k+1/2}) \\ z^{k+1} &:= z^k + x^{k+1} - x^{k+1/2} \end{aligned}$$

## Douglas-Rachford consensus

- $x^{k+1/2}$ -update splits into  $N$  separate (parallel) problems:

$$x_i^{k+1/2} := \operatorname{argmin}_{z_i} (f_i(z_i) + (1/2\lambda)\|z_i - z_i^k\|_2^2), \quad i = 1, \dots, N$$

- $x^{k+1}$ -update is averaging:

$$x_i^{k+1} := \bar{z}^{k+1/2} = (1/N) \sum_{i=1}^N z_i^{k+1/2}, \quad i = 1, \dots, N$$

- $z^{k+1}$ -update becomes

$$\begin{aligned} z_i^{k+1} &= z_i^k + \bar{z}^{k+1/2} - x_i^{k+1/2} \\ &= z_i^k + 2\bar{x}^{k+1/2} - \bar{z}^k - x_i^{k+1/2} \\ &= z_i^k + (\bar{x}^{k+1/2} - x_i^{k+1/2}) + (\bar{x}^{k+1/2} - \bar{z}^k) \end{aligned}$$

- taking average of last equation, we get  $\bar{z}^{k+1} = \bar{x}^{k+1/2}$

## Douglas-Rachford consensus

- renaming  $x^{k+1/2}$  as  $x^{k+1}$ , D-R consensus becomes

$$x_i^{k+1} := \text{prox}_{f_i}(z_i^k)$$

$$z_i^{k+1} := z_i^k + (\bar{x}^{k+1} - x_i^{k+1}) + (\bar{x}^{k+1} - \bar{x}^k)$$

- subsystem (local) state:  $\bar{x}, z_i, x_i$
- gather  $x_i$ 's to compute  $\bar{x}$ , which is then scattered

## Distributed QP

- we use D-R consensus to solve QP

$$\begin{aligned} \text{minimize} \quad & f(x) = \sum_{i=1}^N (1/2) \|A_i x - b_i\|_2^2 \\ \text{subject to} \quad & F_i x \leq g_i, \quad i = 1, \dots, N \end{aligned}$$

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

- each of  $N$  processors will handle an objective term, block of constraints
- coordinate  $N$  QP solvers to solve big QP

## Distributed QP

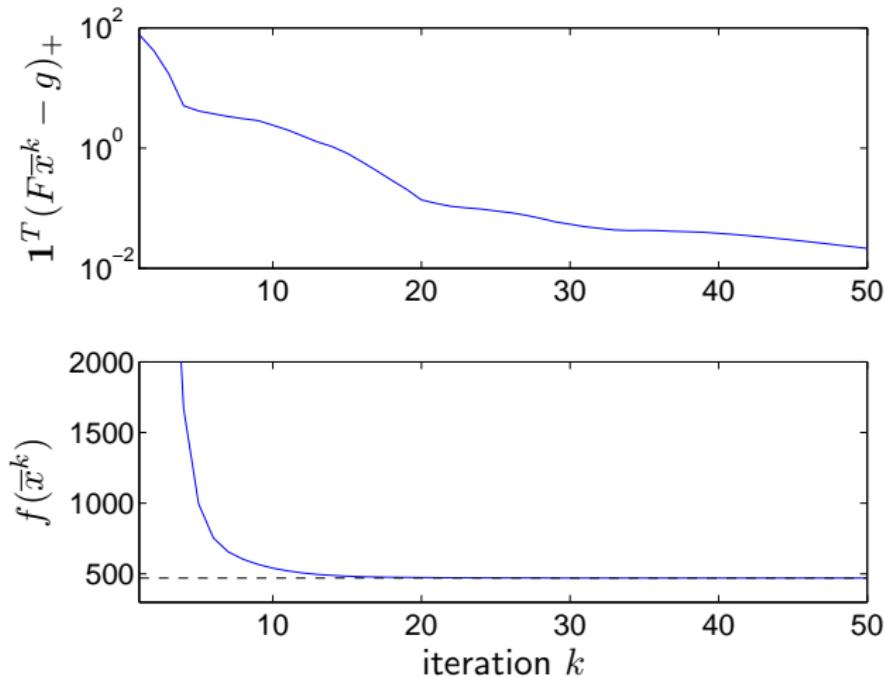
- D-R consensus algorithm is

$$\begin{aligned}x_i^{k+1} &:= \underset{F_i x_i \leq g_i}{\operatorname{argmin}} \left( (1/2) \|A_i x_i - b_i\|_2^2 + (1/2\lambda) \|x_i - z_i^k\|_2^2 \right) \\z_i^{k+1} &:= z_i^k + (\bar{x}^{k+1} - x_i^{k+1}) + (\bar{x}^{k+1} - \bar{x}^k),\end{aligned}$$

- first step is  $N$  parallel QP solves
- second step gives coordination, to solve large problem
- inequality constraint residual is  $\mathbf{1}^T(F\bar{x}^k - g)_+$

## Distributed QP

example with  $n = 100$  variables,  $N = 10$  subsystems



# **Alternating Direction Method of Multipliers**

Prof S. Boyd

EE364b, Stanford University

source:

*Distributed Optimization and Statistical Learning via the Alternating  
Direction Method of Multipliers* (Boyd, Parikh, Chu, Peleato, Eckstein)

# Goals

robust methods for

- ▶ arbitrary-scale optimization
  - machine learning/statistics with huge data-sets
  - dynamic optimization on large-scale network
- ▶ decentralized optimization
  - devices/processors/agents coordinate to solve large problem, by passing relatively small messages

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

Common patterns

Examples

Consensus and exchange

Conclusions

## Dual problem

- ▶ convex equality constrained optimization problem

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & Ax = b\end{array}$$

- ▶ Lagrangian:  $L(x, y) = f(x) + y^T(Ax - b)$

- ▶ dual function:  $g(y) = \inf_x L(x, y)$

- ▶ dual problem: maximize  $g(y)$

- ▶ recover  $x^* = \operatorname{argmin}_x L(x, y^*)$

## Dual ascent

- ▶ gradient method for dual problem:  $y^{k+1} = y^k + \alpha^k \nabla g(y^k)$
- ▶  $\nabla g(y^k) = A\tilde{x} - b$ , where  $\tilde{x} = \operatorname{argmin}_x L(x, y^k)$
- ▶ dual ascent method is

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x L(x, y^k) && // x\text{-minimization} \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b) && // \text{dual update} \end{aligned}$$

- ▶ works, with lots of strong assumptions

## Dual decomposition

- ▶ suppose  $f$  is separable:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \dots, x_N)$$

- ▶ then  $L$  is separable in  $x$ :  $L(x, y) = L_1(x_1, y) + \cdots + L_N(x_N, y) - y^T b$ ,

$$L_i(x_i, y) = f_i(x_i) + y^T A_i x_i$$

- ▶  $x$ -minimization in dual ascent splits into  $N$  separate minimizations

$$x_i^{k+1} := \operatorname{argmin}_{x_i} L_i(x_i, y^k)$$

which can be carried out in parallel

## Dual decomposition

- ▶ dual decomposition (Everett, Dantzig, Wolfe, Benders 1960–65)

$$x_i^{k+1} := \operatorname{argmin}_{x_i} L_i(x_i, y^k), \quad i = 1, \dots, N$$

$$y^{k+1} := y^k + \alpha^k (\sum_{i=1}^N A_i x_i^{k+1} - b)$$

- ▶ scatter  $y^k$ ; update  $x_i$  in parallel; gather  $A_i x_i^{k+1}$
- ▶ solve a large problem
  - by iteratively solving subproblems (in parallel)
  - dual variable update provides coordination
- ▶ works, with lots of assumptions; often slow

# **Outline**

Dual decomposition

**Method of multipliers**

Alternating direction method of multipliers

Common patterns

Examples

Consensus and exchange

Conclusions

## Method of multipliers

- ▶ a method to robustify dual ascent
- ▶ use **augmented Lagrangian** (Hestenes, Powell 1969),  $\rho > 0$

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2$$

- ▶ method of multipliers (Hestenes, Powell; analysis in Bertsekas 1982)

$$x^{k+1} := \operatorname{argmin}_x L_\rho(x, y^k)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} - b)$$

(note specific dual update step length  $\rho$ )

## Method of multipliers dual update step

- optimality conditions (for differentiable  $f$ ):

$$Ax^* - b = 0, \quad \nabla f(x^*) + A^T y^* = 0$$

(primal and dual feasibility)

- since  $x^{k+1}$  minimizes  $L_\rho(x, y^k)$

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T (y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

- dual update  $y^{k+1} = y^k + \rho(x^{k+1} - b)$  makes  $(x^{k+1}, y^{k+1})$  *dual feasible*
- *primal feasibility* achieved in limit:  $Ax^{k+1} - b \rightarrow 0$

## Method of multipliers

(compared to dual decomposition)

- ▶ *good news:* converges under much more relaxed conditions  
( $f$  can be nondifferentiable, take on value  $+\infty, \dots$ )
- ▶ *bad news:* quadratic penalty destroys splitting of the  $x$ -update, so can't do decomposition

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

Common patterns

Examples

Consensus and exchange

Conclusions

## Alternating direction method of multipliers

- ▶ a method
  - with good robustness of method of multipliers
  - which can support decomposition
- ▶ “robust dual decomposition” or “decomposable method of multipliers”
- ▶ proposed by Gabay, Mercier, Glowinski, Marrocco in 1976

## Alternating direction method of multipliers

- ADMM problem form (with  $f, g$  convex)

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

- two sets of variables, with separable objective
- $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$
- ADMM:

$$x^{k+1} := \operatorname{argmin}_x L_\rho(x, z^k, y^k) \quad // x\text{-minimization}$$

$$z^{k+1} := \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) \quad // z\text{-minimization}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \quad // dual\ update$$

## Alternating direction method of multipliers

- ▶ if we minimized over  $x$  and  $z$  jointly, reduces to method of multipliers
- ▶ instead, we do one pass of a Gauss-Seidel method
- ▶ we get splitting since we minimize over  $x$  with  $z$  fixed, and vice versa

## ADMM and optimality conditions

- ▶ optimality conditions (for differentiable case):
  - primal feasibility:  $Ax + Bz - c = 0$
  - dual feasibility:  $\nabla f(x) + A^T y = 0, \quad \nabla g(z) + B^T y = 0$
- ▶ since  $z^{k+1}$  minimizes  $L_\rho(x^{k+1}, z, y^k)$  we have

$$\begin{aligned} 0 &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T y^{k+1} \end{aligned}$$

- ▶ so with ADMM dual variable update,  $(x^{k+1}, z^{k+1}, y^{k+1})$  satisfies second dual feasibility condition
- ▶ primal and first dual feasibility are achieved as  $k \rightarrow \infty$

## ADMM with scaled dual variables

- ▶ combine linear and quadratic terms in augmented Lagrangian

$$\begin{aligned} L_\rho(x, z, y) &= f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2 \\ &= f(x) + g(z) + (\rho/2)\|Ax + Bz - c + u\|_2^2 + \text{const.} \end{aligned}$$

with  $u^k = (1/\rho)y^k$

- ▶ ADMM (scaled dual form):

$$x^{k+1} := \underset{x}{\operatorname{argmin}} \left( f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2 \right)$$

$$z^{k+1} := \underset{z}{\operatorname{argmin}} \left( g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2 \right)$$

$$u^{k+1} := u^k + (Ax^{k+1} + Bz^{k+1} - c)$$

# Convergence

- ▶ assume (very little!)
  - $f, g$  convex, closed, proper
  - $L_0$  has a saddle point
- ▶ then ADMM converges:
  - iterates approach feasibility:  $Ax^k + Bz^k - c \rightarrow 0$
  - objective approaches optimal value:  $f(x^k) + g(z^k) \rightarrow p^*$

## Related algorithms

- ▶ operator splitting methods  
(Douglas, Peaceman, Rachford, Lions, Mercier, ... 1950s, 1979)
- ▶ proximal point algorithm (Rockafellar 1976)
- ▶ Dykstra's alternating projections algorithm (1983)
- ▶ Spingarn's method of partial inverses (1985)
- ▶ Rockafellar-Wets progressive hedging (1991)
- ▶ proximal methods (Rockafellar, many others, 1976–present)
- ▶ Bregman iterative methods (2008–present)
- ▶ most of these are special cases of the proximal point algorithm

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

## **Common patterns**

Examples

Consensus and exchange

Conclusions

## Common patterns

- ▶  $x$ -update step requires minimizing  $f(x) + (\rho/2)\|Ax - v\|_2^2$   
(with  $v = Bz^k - c + u^k$ , which is constant during  $x$ -update)
- ▶ similar for  $z$ -update
- ▶ several special cases come up often
- ▶ can simplify update by exploit structure in these cases

## Decomposition

- ▶ suppose  $f$  is block-separable,

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \dots, x_N)$$

- ▶  $A$  is conformably block separable:  $A^T A$  is block diagonal
- ▶ then  $x$ -update splits into  $N$  parallel updates of  $x_i$

## Proximal operator

- consider  $x$ -update when  $A = I$

$$x^+ = \operatorname{argmin}_x (f(x) + (\rho/2)\|x - v\|_2^2) = \mathbf{prox}_{f,\rho}(v)$$

- some special cases:

$$f = I_C \text{ (indicator fct. of set } C) \quad x^+ := \Pi_C(v) \text{ (projection onto } C)$$

$$f = \lambda \|\cdot\|_1 \text{ (\ell}_1 \text{ norm)} \quad x_i^+ := S_{\lambda/\rho}(v_i) \text{ (soft thresholding)}$$

$$(S_a(v) = (v - a)_+ - (-v - a)_+)$$

## Quadratic objective

- ▶  $f(x) = (1/2)x^T Px + q^T x + r$
- ▶  $x^+ := (P + \rho A^T A)^{-1}(\rho A^T v - q)$
- ▶ use matrix inversion lemma when computationally advantageous

$$(P + \rho A^T A)^{-1} = P^{-1} - \rho P^{-1} A^T (I + \rho A P^{-1} A^T)^{-1} A P^{-1}$$

- ▶ (direct method) cache factorization of  $P + \rho A^T A$  (or  $I + \rho A P^{-1} A^T$ )
- ▶ (iterative method) warm start, early stopping, reducing tolerances

## Smooth objective

- ▶  $f$  smooth
- ▶ can use standard methods for smooth minimization
  - gradient, Newton, or quasi-Newton
  - preconditionned CG, limited-memory BFGS (scale to very large problems)
- ▶ can exploit
  - warm start
  - early stopping, with tolerances decreasing as ADMM proceeds

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

Common patterns

## **Examples**

Consensus and exchange

Conclusions

## **Examples**

## Constrained convex optimization

- ▶ consider ADMM for generic problem

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

- ▶ ADMM form: take  $g$  to be indicator of  $\mathcal{C}$

$$\begin{aligned} & \text{minimize} && f(x) + g(z) \\ & \text{subject to} && x - z = 0 \end{aligned}$$

- ▶ algorithm:

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x (f(x) + (\rho/2)\|x - z^k + u^k\|_2^2) \\ z^{k+1} &:= \Pi_{\mathcal{C}}(x^{k+1} + u^k) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

## Lasso

- lasso problem:

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

- ADMM form:

$$\begin{aligned}\text{minimize} \quad & (1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} \quad & x - z = 0\end{aligned}$$

- ADMM:

$$\begin{aligned}x^{k+1} &:= (A^T A + \rho I)^{-1}(A^T b + \rho z^k - y^k) \\ z^{k+1} &:= S_{\lambda/\rho}(x^{k+1} + y^k / \rho) \\ y^{k+1} &:= y^k + \rho(x^{k+1} - z^{k+1})\end{aligned}$$

## Lasso example

- ▶ example with dense  $A \in \mathbf{R}^{1500 \times 5000}$   
(1500 measurements; 5000 regressors)
- ▶ computation times

factorization (same as ridge regression)	1.3s
subsequent ADMM iterations	0.03s
lasso solve (about 50 ADMM iterations)	2.9s
full regularization path (30 $\lambda$ 's)	4.4s

- ▶ not bad for a *very short* Matlab script

## Sparse inverse covariance selection

- ▶  $S$ : empirical covariance of samples from  $\mathcal{N}(0, \Sigma)$ , with  $\Sigma^{-1}$  sparse (i.e., Gaussian Markov random field)

- ▶ estimate  $\Sigma^{-1}$  via  $\ell_1$  regularized maximum likelihood

$$\text{minimize } \mathbf{Tr}(SX) - \log \det X + \lambda \|X\|_1$$

- ▶ methods: COVSEL (Banerjee et al 2008), graphical lasso (FHT 2008)

## Sparse inverse covariance selection via ADMM

- ADMM form:

$$\begin{aligned} & \text{minimize}_{X} \quad \mathbf{Tr}(SX) - \log \det X + \lambda \|Z\|_1 \\ & \text{subject to} \quad X - Z = 0 \end{aligned}$$

- ADMM:

$$\begin{aligned} X^{k+1} &:= \operatorname{argmin}_X (\mathbf{Tr}(SX) - \log \det X + (\rho/2)\|X - Z^k + U^k\|_F^2) \\ Z^{k+1} &:= S_{\lambda/\rho}(X^{k+1} + U^k) \\ U^{k+1} &:= U^k + (X^{k+1} - Z^{k+1}) \end{aligned}$$

## Analytical solution for $X$ -update

- ▶ compute eigendecomposition  $\rho(Z^k - U^k) - S = Q\Lambda Q^T$
- ▶ form diagonal matrix  $\tilde{X}$  with

$$\tilde{X}_{ii} = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4\rho}}{2\rho}$$

- ▶ let  $X^{k+1} := Q\tilde{X}Q^T$
- ▶ cost of  $X$ -update is an eigendecomposition

## Sparse inverse covariance selection example

- ▶  $\Sigma^{-1}$  is  $1000 \times 1000$  with  $10^4$  nonzeros
  - graphical lasso (Fortran): 20 seconds – 3 minutes
  - ADMM (Matlab): 3 – 10 minutes
  - (depends on choice of  $\lambda$ )
- ▶ very rough experiment, but with no special tuning, ADMM is in ballpark of recent specialized methods
- ▶ (for comparison, COVSEL takes 25+ min when  $\Sigma^{-1}$  is a  $400 \times 400$  tridiagonal matrix)

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

Common patterns

Examples

**Consensus and exchange**

Conclusions

# Consensus optimization

- ▶ want to solve problem with  $N$  objective terms

$$\text{minimize} \quad \sum_{i=1}^N f_i(x)$$

- e.g.,  $f_i$  is the loss function for  $i$ th block of training data
- ▶ ADMM form:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^N f_i(x_i) \\ & \text{subject to} && x_i - z = 0 \end{aligned}$$

- $x_i$  are *local variables*
- $z$  is the *global variable*
- $x_i - z = 0$  are *consistency* or *consensus* constraints
- can add regularization using a  $g(z)$  term

## Consensus optimization via ADMM

- ▶  $L_\rho(x, z, y) = \sum_{i=1}^N (f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2)$

- ▶ ADMM:

$$x_i^{k+1} := \operatorname{argmin}_{x_i} (f_i(x_i) + y_i^{kT}(x_i - z^k) + (\rho/2)\|x_i - z^k\|_2^2)$$

$$z^{k+1} := \frac{1}{N} \sum_{i=1}^N (x_i^{k+1} + (1/\rho)y_i^k)$$

$$y_i^{k+1} := y_i^k + \rho(x_i^{k+1} - z^{k+1})$$

- ▶ with regularization, averaging in  $z$  update is followed by  $\text{prox}_{g,\rho}$

## Consensus optimization via ADMM

- ▶ using  $\sum_{i=1}^N y_i^k = 0$ , algorithm simplifies to

$$\begin{aligned}x_i^{k+1} &:= \operatorname{argmin}_{x_i} \left( f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + (\rho/2)\|x_i - \bar{x}^k\|_2^2 \right) \\y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})\end{aligned}$$

where  $\bar{x}^k = (1/N) \sum_{i=1}^N x_i^k$

- ▶ in each iteration
  - gather  $x_i^k$  and average to get  $\bar{x}^k$
  - scatter the average  $\bar{x}^k$  to processors
  - update  $y_i^k$  locally (in each processor, in parallel)
  - update  $x_i$  locally

## Statistical interpretation

- ▶  $f_i$  is negative log-likelihood for parameter  $x$  given  $i$ th data block
- ▶  $x_i^{k+1}$  is MAP estimate under prior  $\mathcal{N}(\bar{x}^k + (1/\rho)y_i^k, \rho I)$
- ▶ prior mean is previous iteration's consensus shifted by 'price' of processor  $i$  disagreeing with previous consensus
- ▶ processors only need to support a Gaussian MAP method
  - type or number of data in each block not relevant
  - consensus protocol yields global maximum-likelihood estimate

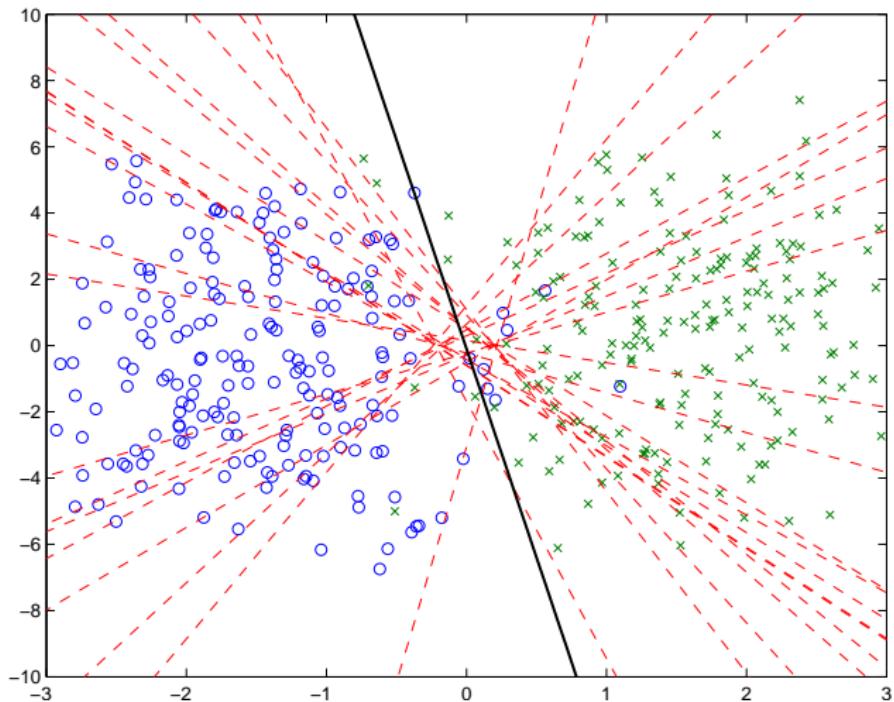
## Consensus classification

- ▶ data (examples)  $(a_i, b_i), i = 1, \dots, N, a_i \in \mathbf{R}^n, b_i \in \{-1, +1\}$
- ▶ linear classifier  $\text{sign}(a^T w + v)$ , with weight  $w$ , offset  $v$
- ▶ margin for  $i$ th example is  $b_i(a_i^T w + v)$ ; want margin to be positive
- ▶ loss for  $i$ th example is  $l(b_i(a_i^T w + v))$ 
  - $l$  is loss function (hinge, logistic, probit, exponential, ...)
- ▶ choose  $w, v$  to minimize  $\frac{1}{N} \sum_{i=1}^N l(b_i(a_i^T w + v)) + r(w)$ 
  - $r(w)$  is regularization term ( $\ell_2, \ell_1, \dots$ )
- ▶ split data and use ADMM consensus to solve

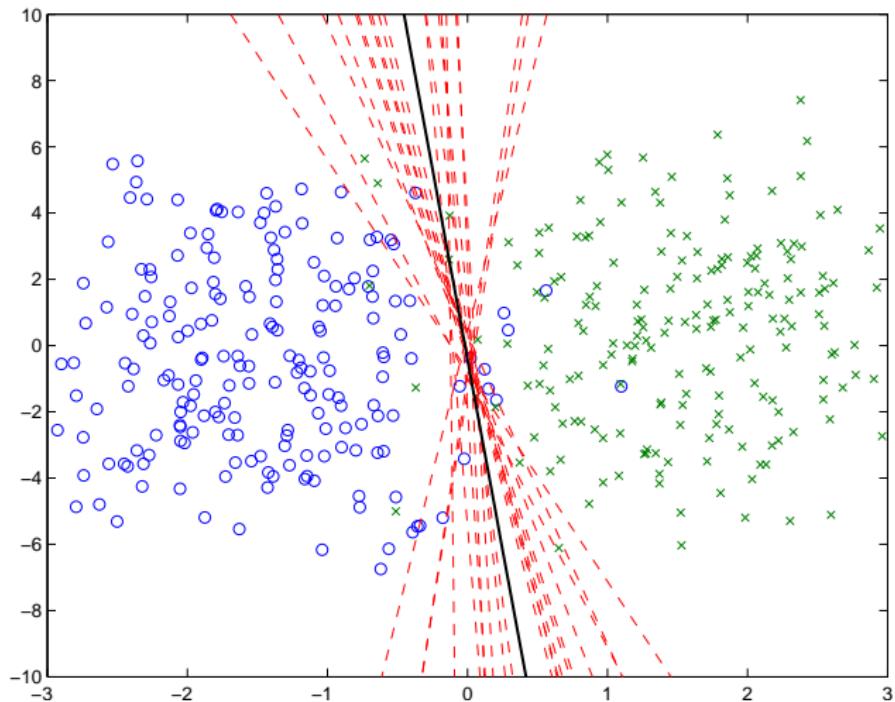
## Consensus SVM example

- ▶ hinge loss  $l(u) = (1 - u)_+$  with  $\ell_2$  regularization
- ▶ baby problem with  $n = 2$ ,  $N = 400$  to illustrate
- ▶ examples split into 20 groups, in worst possible way:  
each group contains only positive or negative examples

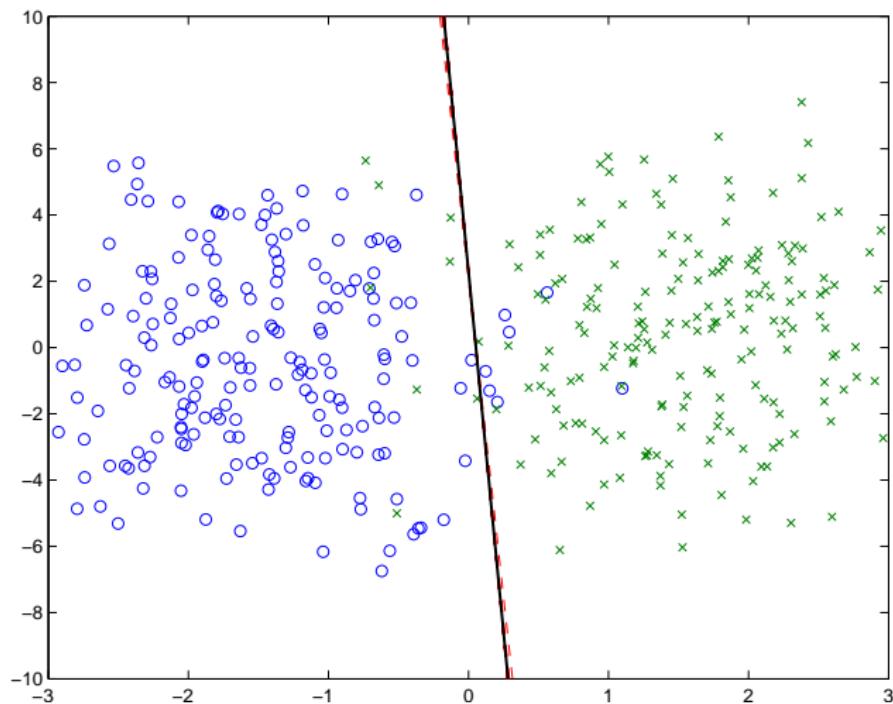
## Iteration 1



## Iteration 5



## Iteration 40



## Distributed lasso example

- ▶ example with **dense**  $A \in \mathbb{R}^{400000 \times 8000}$  (roughly 30 GB of data)
  - distributed solver written in C using MPI and GSL
  - no optimization or tuned libraries (like ATLAS, MKL)
  - split into 80 subsystems across 10 (8-core) machines on Amazon EC2
- ▶ computation times

loading data	30s
factorization	5m
subsequent ADMM iterations	0.5–2s
lasso solve (about 15 ADMM iterations)	5–6m

## Exchange problem

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N x_i = 0\end{array}$$

- ▶ another canonical problem, like consensus
- ▶ in fact, it's the dual of consensus
- ▶ can interpret as  $N$  agents exchanging  $n$  goods to minimize a total cost
- ▶  $(x_i)_j \geq 0$  means agent  $i$  receives  $(x_i)_j$  of good  $j$  from exchange
- ▶  $(x_i)_j < 0$  means agent  $i$  contributes  $|(x_i)_j|$  of good  $j$  to exchange
- ▶ constraint  $\sum_{i=1}^N x_i = 0$  is *equilibrium* or *market clearing* constraint
- ▶ optimal dual variable  $y^*$  is a set of valid prices for the goods
- ▶ suggests real or virtual cash payment  $(y^*)^T x_i$  by agent  $i$

## Exchange ADMM

- ▶ solve as a generic constrained convex problem with constraint set

$$\mathcal{C} = \{x \in \mathbf{R}^{nN} \mid x_1 + x_2 + \cdots + x_N = 0\}$$

- ▶ scaled form:

$$\begin{aligned} x_i^{k+1} &:= \operatorname{argmin}_{x_i} (f_i(x_i) + (\rho/2)\|x_i - x_i^k + \bar{x}^k + u^k\|_2^2) \\ u^{k+1} &:= u^k + \bar{x}^{k+1} \end{aligned}$$

- ▶ unscaled form:

$$\begin{aligned} x_i^{k+1} &:= \operatorname{argmin}_{x_i} (f_i(x_i) + y^{kT} x_i + (\rho/2)\|x_i - (x_i^k - \bar{x}^k)\|_2^2) \\ y^{k+1} &:= y^k + \rho \bar{x}^{k+1} \end{aligned}$$

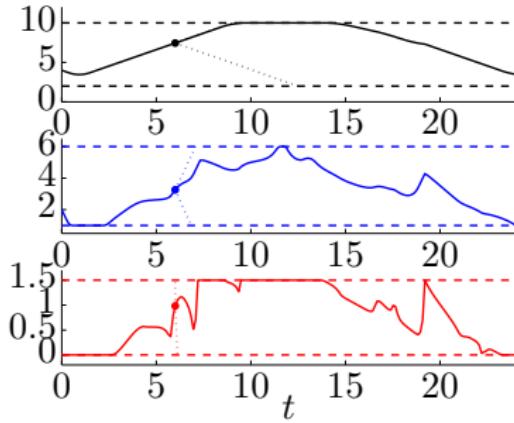
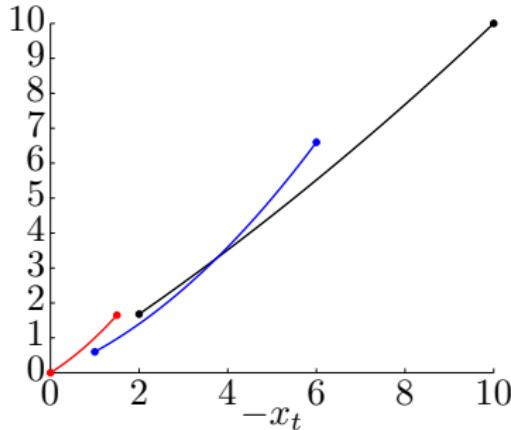
## Interpretation as *tâtonnement* process

- ▶ *tâtonnement process*: iteratively update prices to clear market
- ▶ work towards equilibrium by increasing/decreasing prices of goods based on excess demand/supply
- ▶ dual decomposition is the simplest *tâtonnement* algorithm
- ▶ ADMM adds proximal regularization
  - incorporate agents' prior commitment to help clear market
  - convergence far more robust than dual decomposition

## Distributed dynamic energy management

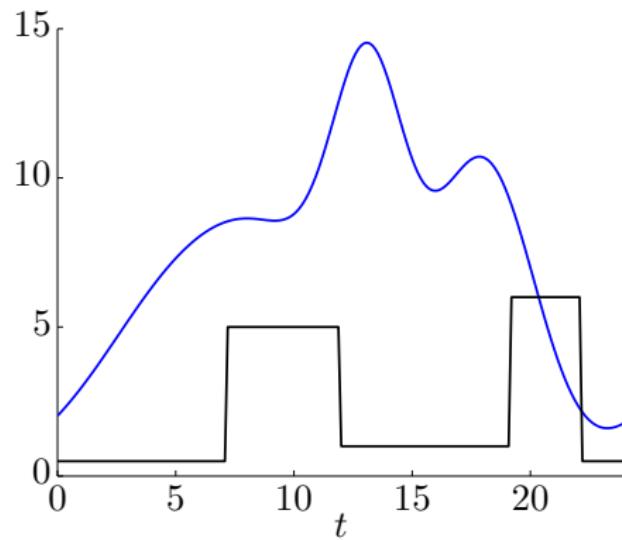
- ▶  $N$  devices exchange power in time periods  $t = 1, \dots, T$
- ▶  $x_i \in \mathbf{R}^T$  is power flow *profile* for device  $i$
- ▶  $f_i(x_i)$  is cost of profile  $x_i$  (and encodes constraints)
- ▶  $x_1 + \dots + x_N = 0$  is energy balance (in each time period)
- ▶ dynamic energy management problem is exchange problem
- ▶ exchange ADMM gives distributed method for dynamic energy management
- ▶ each device optimizes its own profile, with quadratic regularization for coordination
- ▶ residual (energy imbalance) is driven to zero

# Generators



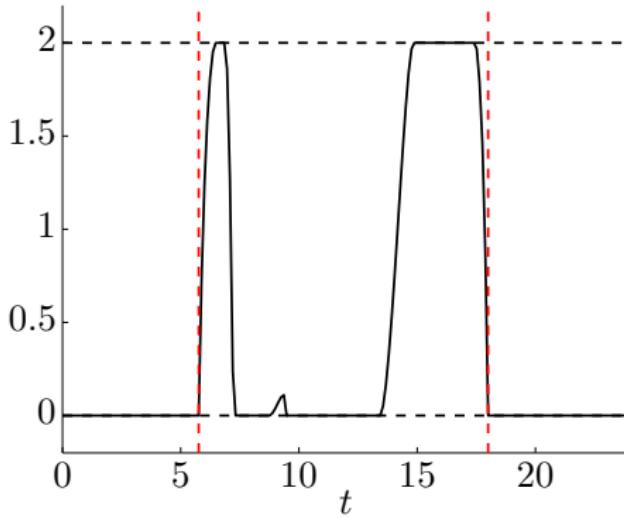
- ▶ 3 example generators
- ▶ left: generator costs/limits; right: ramp constraints
- ▶ can add cost for power changes

## Fixed loads



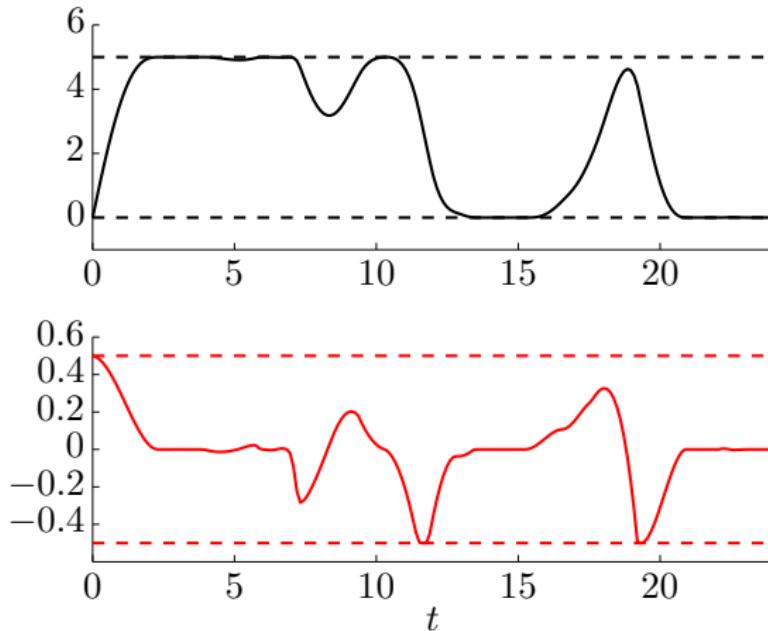
- ▶ 2 example fixed loads
- ▶ cost is  $+\infty$  for not supplying load; zero otherwise

## Shiftable load



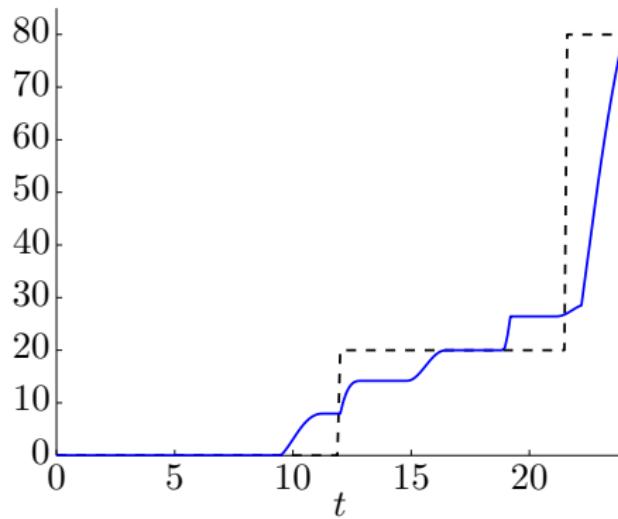
- ▶ total energy consumed over an interval must exceed given minimum level
- ▶ limits on energy consumed in each period
- ▶ cost is  $+\infty$  for violating constraints; zero otherwise

## Battery energy storage system



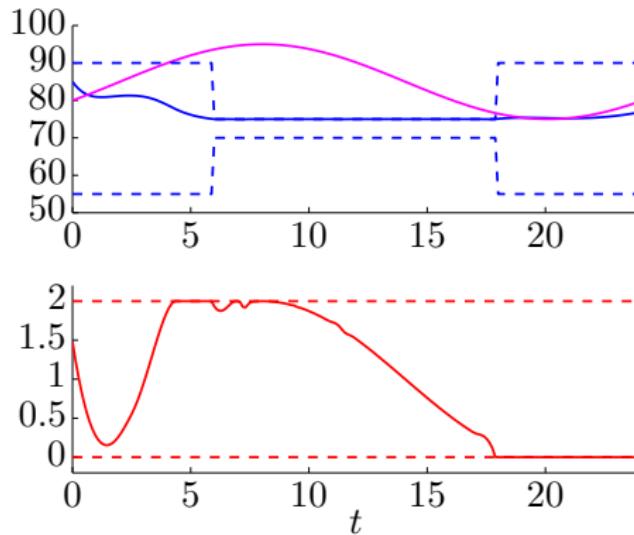
- ▶ energy store with maximum capacity, charge/discharge limits
- ▶ black: battery charge, red: charge/discharge profile
- ▶ cost is  $+\infty$  for violating constraints; zero otherwise

## Electric vehicle charging system



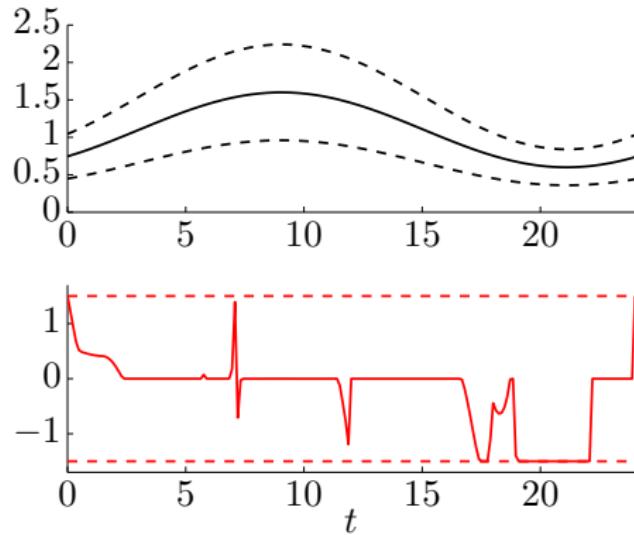
- black: desired charge profile; blue: charge profile
- shortfall cost for not meeting desired charge

# HVAC



- ▶ thermal load (e.g., room, refrigerator) with temperature limits
- ▶ magenta: ambient temperature; blue: load temperature
- ▶ red: cooling energy profile
- ▶ cost is  $+\infty$  for violating constraints; zero otherwise

## External tie



- ▶ buy/sell energy from/to external grid at price  $p^{\text{ext}}(t) \pm \gamma(t)$
- ▶ solid:  $p^{\text{ext}}(t)$ ; dashed:  $p^{\text{ext}}(t) \pm \gamma(t)$

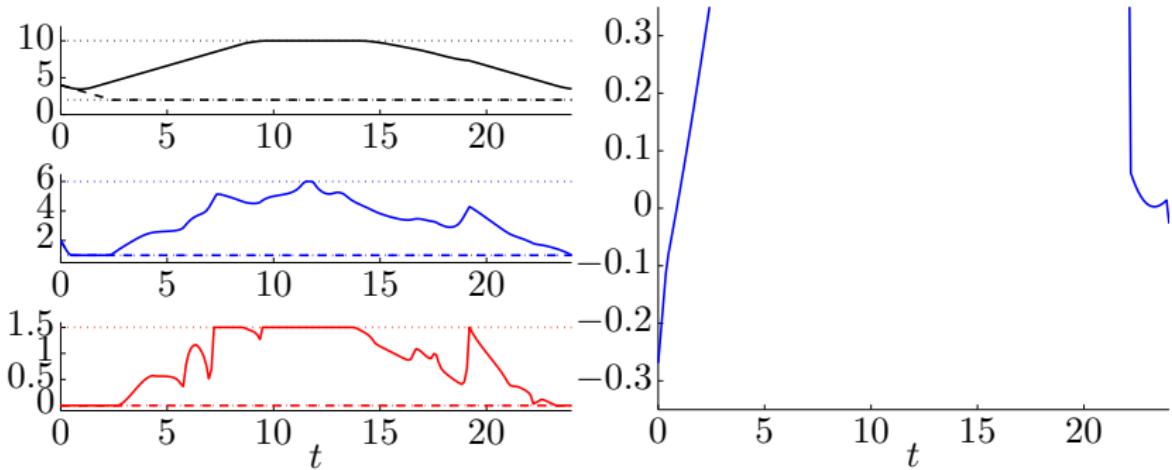
## Smart grid example

10 devices (already described above)

- ▶ 3 generators
- ▶ 2 fixed loads
- ▶ 1 shiftable load
- ▶ 1 EV charging systems
- ▶ 1 battery
- ▶ 1 HVAC system
- ▶ 1 external tie

# Convergence

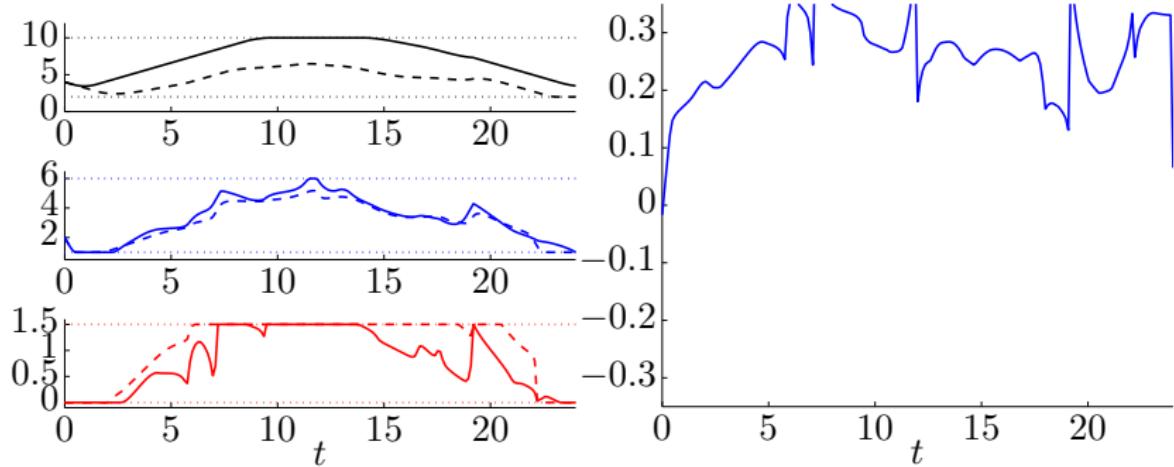
iteration:  $k = 1$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

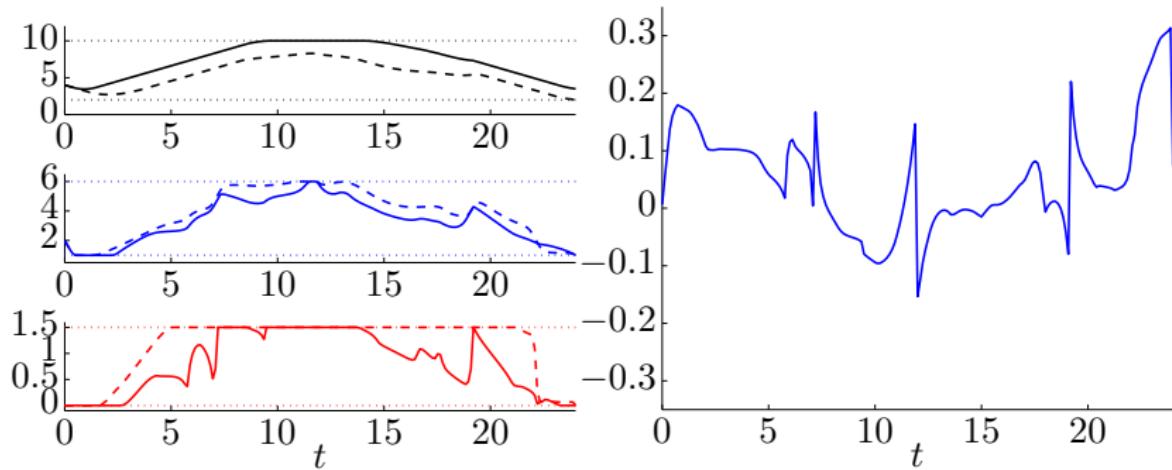
iteration:  $k = 3$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

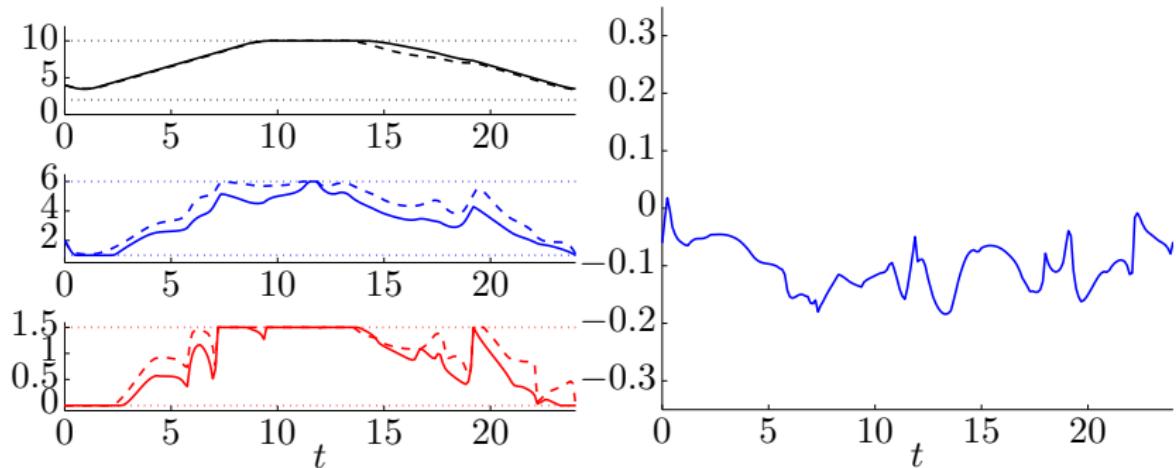
iteration:  $k = 5$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

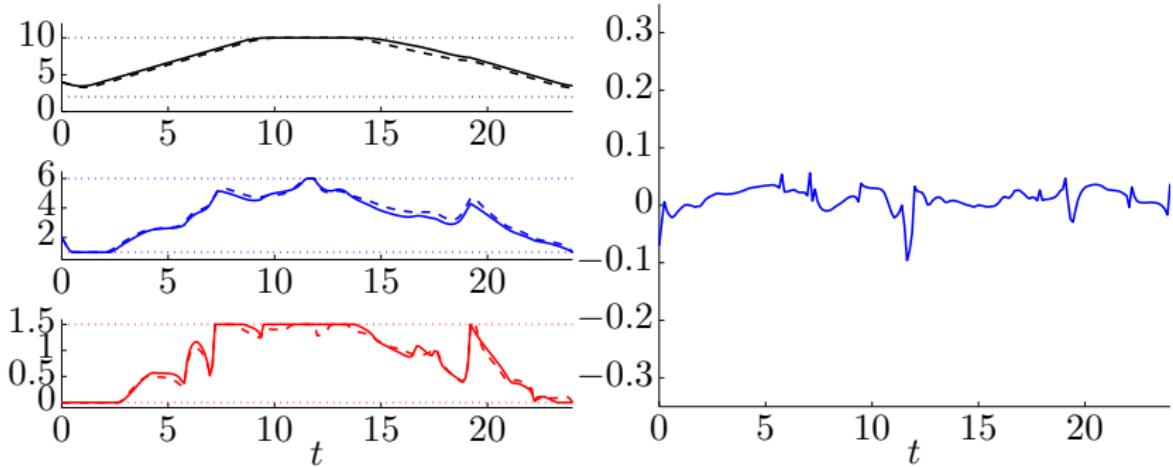
iteration:  $k = 10$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

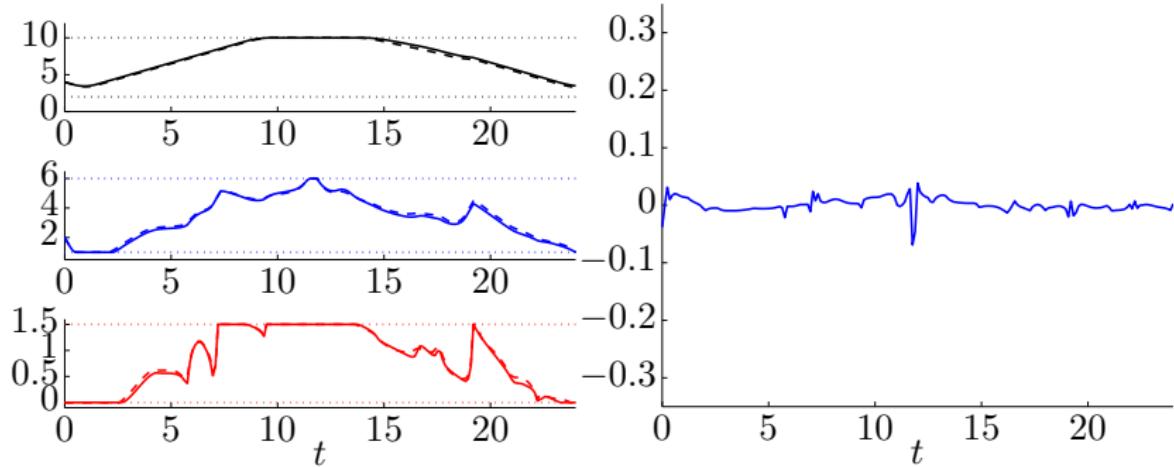
iteration:  $k = 15$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

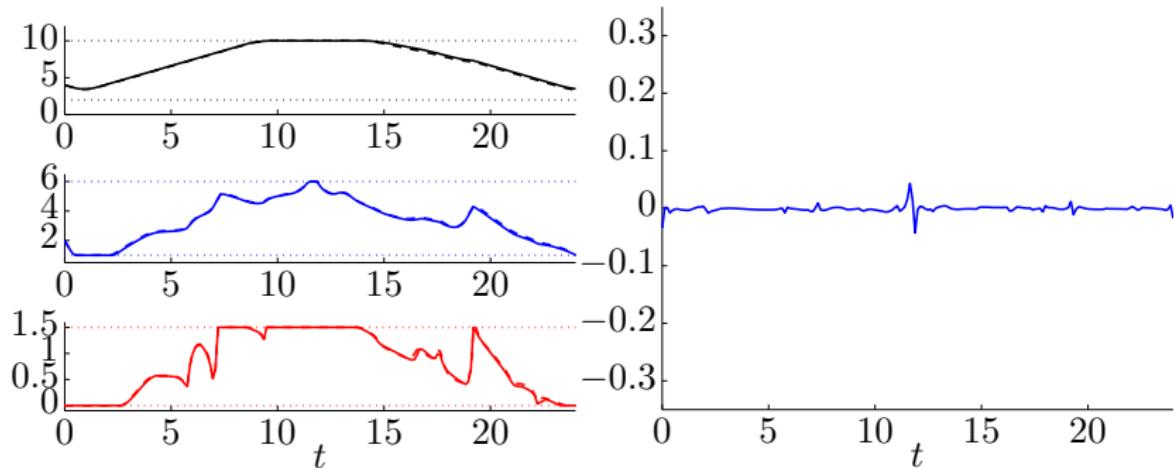
iteration:  $k = 20$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

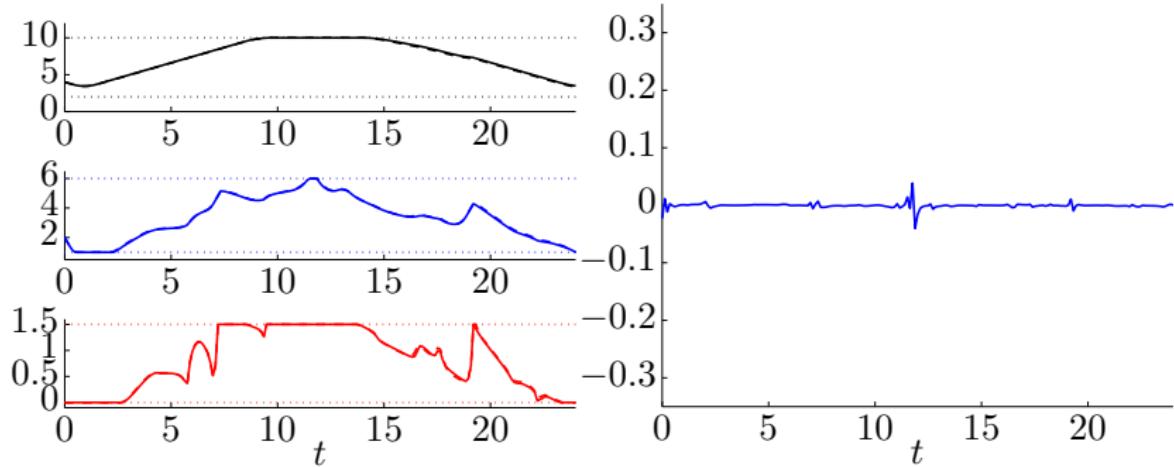
iteration:  $k = 25$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

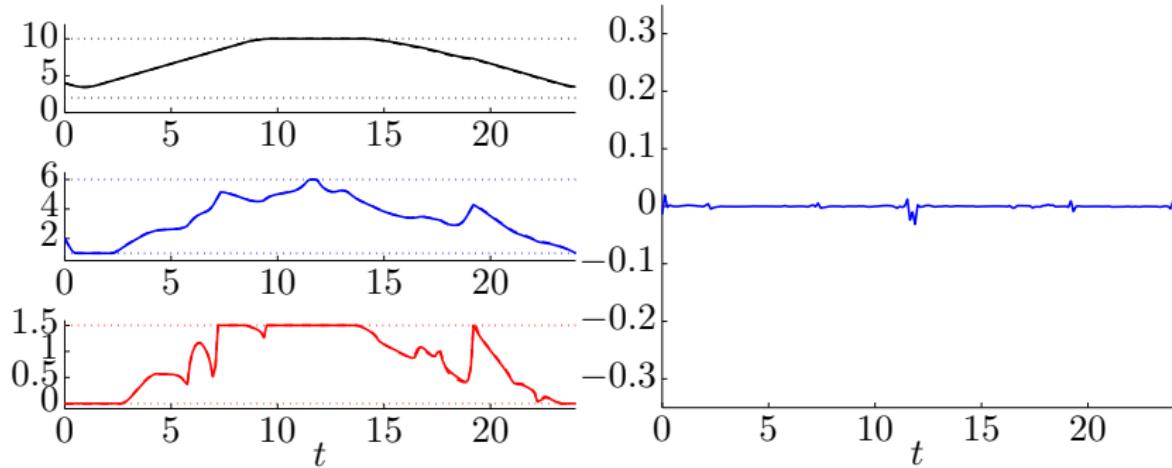
iteration:  $k = 30$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

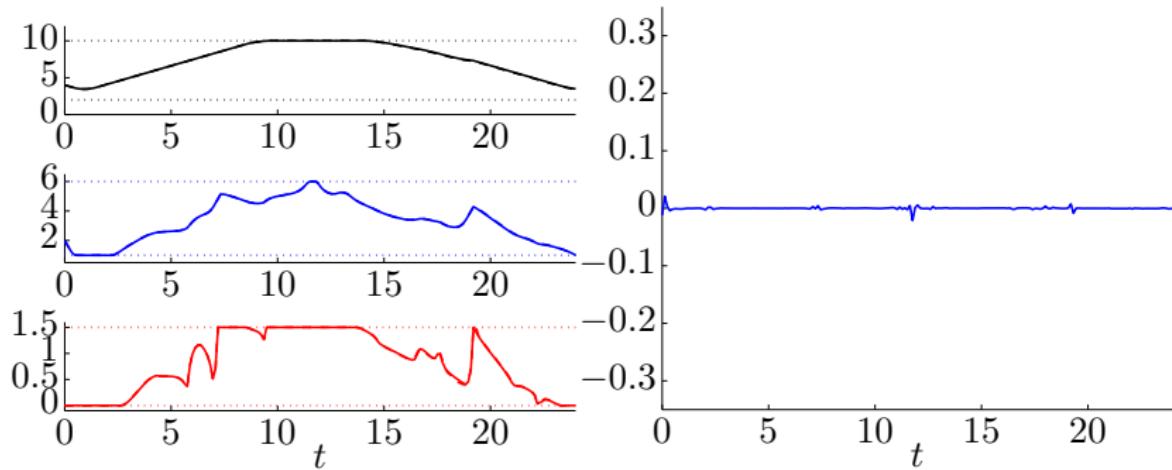
iteration:  $k = 35$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

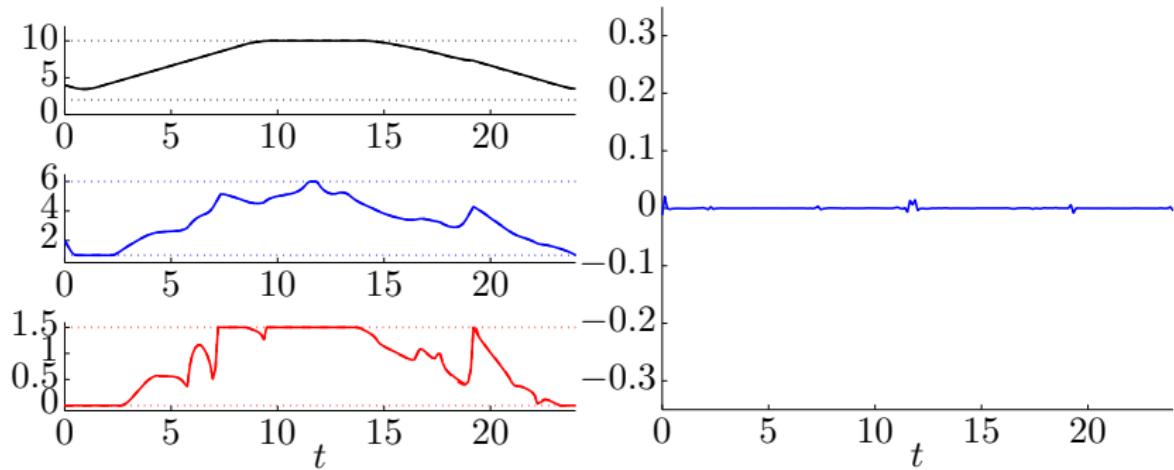
iteration:  $k = 40$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

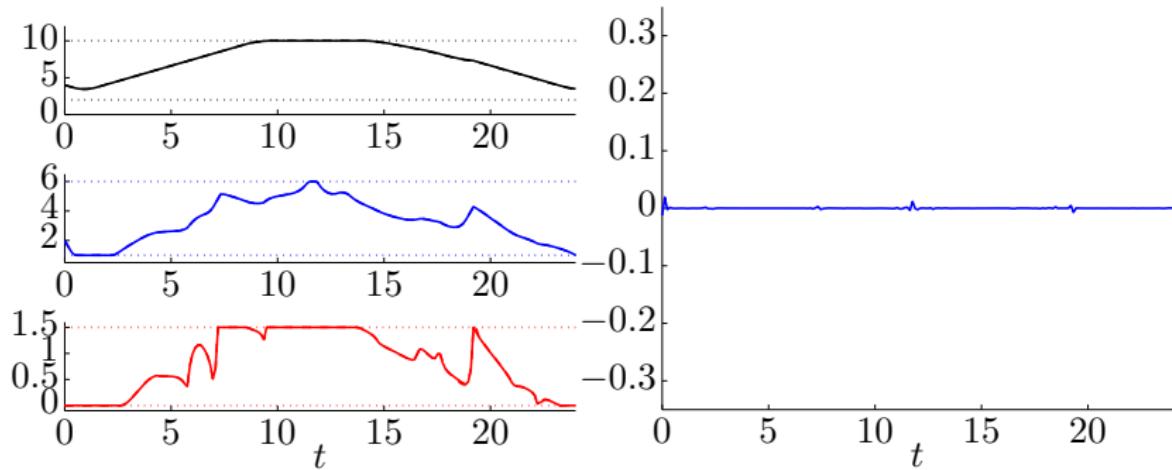
iteration:  $k = 45$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

iteration:  $k = 50$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# **Outline**

Dual decomposition

Method of multipliers

Alternating direction method of multipliers

Common patterns

Examples

Consensus and exchange

**Conclusions**

Conclusions

58

## Summary and conclusions

### ADMM

- ▶ is the same as, or closely related to, many methods with other names
- ▶ has been around since the 1970s
- ▶ gives simple single-processor algorithms that can be competitive with state-of-the-art
- ▶ can be used to coordinate many processors, each solving a substantial problem, to solve a very large problem

# Conjugate Gradient Method

- direct and indirect methods
- positive definite linear systems
- Krylov sequence
- spectral analysis of Krylov sequence
- preconditioning

# Three classes of methods for linear equations

methods to solve linear system  $Ax = b$ ,  $A \in \mathbf{R}^{n \times n}$

- **dense direct** (factor-solve methods)
  - runtime depends only on size; independent of data, structure, or sparsity
  - work well for  $n$  up to a few thousand
- **sparse direct** (factor-solve methods)
  - runtime depends on size, sparsity pattern; (almost) independent of data
  - can work well for  $n$  up to  $10^4$  or  $10^5$  (or more)
  - requires good heuristic for ordering

- **indirect** (iterative methods)
  - runtime depends on data, size, sparsity, required accuracy
  - requires tuning, preconditioning, . . .
  - good choice in many cases; only choice for  $n = 10^6$  or larger

# Symmetric positive definite linear systems

SPD system of equations

$$Ax = b, \quad A \in \mathbf{R}^{n \times n}, \quad A = A^T \succ 0$$

examples

- Newton/interior-point search direction:  $\nabla^2\phi(x)\Delta x = -\nabla\phi(x)$
- least-squares normal equations:  $(A^T A)x = A^T b$
- regularized least-squares:  $(A^T A + \mu I)x = A^T b$
- minimization of convex quadratic function  $(1/2)x^T A x - b^T x$
- solving (discretized) elliptic PDE (*e.g.*, Poisson equation)

- analysis of resistor circuit:  $Gv = i$ 
  - $v$  is node voltage (vector),  $i$  is (given) source current
  - $G$  is circuit conductance matrix

$$G_{ij} = \begin{cases} \text{total conductance incident on node } i & i = j \\ -(\text{conductance between nodes } i \text{ and } j) & i \neq j \end{cases}$$

## CG overview

- proposed by Hestenes and Stiefel in 1952 (as direct method)
- solves SPD system  $Ax = b$ 
  - in theory (*i.e.*, exact arithmetic) in  $n$  iterations
  - each iteration requires a few inner products in  $\mathbf{R}^n$ , and one matrix-vector multiply  $z \rightarrow Az$
- for  $A$  dense, matrix-vector multiply  $z \rightarrow Az$  costs  $n^2$ , so total cost is  $n^3$ , same as direct methods
- get advantage over dense if matrix-vector multiply is cheaper than  $n^2$
- with roundoff error, CG can work poorly (or not at all)
- but for some  $A$  (and  $b$ ), can get good approximate solution in  $\ll n$  iterations

## Solution and error

- $x^* = A^{-1}b$  is solution
- $x^*$  minimizes (convex function)  $f(x) = (1/2)x^T Ax - b^T x$
- $\nabla f(x) = Ax - b$  is gradient of  $f$
- with  $f^* = f(x^*)$ , we have

$$\begin{aligned}f(x) - f^* &= (1/2)x^T Ax - b^T x - (1/2)x^{*T} Ax^* + b^T x^* \\&= (1/2)(x - x^*)^T A(x - x^*) \\&= (1/2)\|x - x^*\|_A^2\end{aligned}$$

i.e.,  $f(x) - f^*$  is half of squared  $A$ -norm of error  $x - x^*$

- a relative measure (comparing  $x$  to 0):

$$\tau = \frac{f(x) - f^*}{f(0) - f^*} = \frac{\|x - x^*\|_A^2}{\|x^*\|_A^2}$$

(fraction of maximum possible reduction in  $f$ , compared to  $x = 0$ )

## Residual

- $r = b - Ax$  is called the **residual** at  $x$
- $r = -\nabla f(x) = A(x^* - x)$
- in terms of  $r$ , we have

$$\begin{aligned}f(x) - f^* &= (1/2)(x - x^*)^T A(x - x^*) \\&= (1/2)r^T A^{-1}r \\&= (1/2)\|r\|_{A^{-1}}^2\end{aligned}$$

- a commonly used measure of relative accuracy:  $\eta = \|r\|/\|b\|$
- $\tau \leq \kappa(A)\eta^2$  ( $\eta$  is easily computable from  $x$ ;  $\tau$  is not)

# Krylov subspace

(a.k.a. controllability subspace)

$$\begin{aligned}\mathcal{K}_k &= \text{span}\{b, Ab, \dots, A^{k-1}b\} \\ &= \{p(A)b \mid p \text{ polynomial}, \deg p < k\}\end{aligned}$$

we define the *Krylov sequence*  $x^{(1)}, x^{(2)}, \dots$  as

$$x^{(k)} = \underset{x \in \mathcal{K}_k}{\operatorname{argmin}} f(x) = \underset{x \in \mathcal{K}_k}{\operatorname{argmin}} \|x - x^*\|_A^2$$

the CG algorithm (among others) generates the Krylov sequence

## Properties of Krylov sequence

- $f(x^{(k+1)}) \leq f(x^{(k)})$  (but  $\|r\|$  can increase)
- $x^{(n)} = x^*$  (i.e.,  $x^* \in \mathcal{K}_n$  even when  $\mathcal{K}_n \neq \mathbf{R}^n$ )
- $x^{(k)} = p_k(A)b$ , where  $p_k$  is a polynomial with  $\deg p_k < k$
- less obvious: there is a *two-term recurrence*

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

for some  $\alpha_k, \beta_k$  (basis of CG algorithm)

## Cayley-Hamilton theorem

characteristic polynomial of  $A$ :

$$\chi(s) = \det(sI - A) = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n$$

by Cayley-Hamilton theorem

$$\chi(A) = A^n + \alpha_1 A^{n-1} + \cdots + \alpha_n I = 0$$

and so

$$A^{-1} = -(1/\alpha_n)A^{n-1} - (\alpha_1/\alpha_n)A^{n-2} - \cdots - (\alpha_{n-1}/\alpha_n)I$$

in particular, we see that  $x^* = A^{-1}b \in \mathcal{K}_n$

## Spectral analysis of Krylov sequence

- $A = Q\Lambda Q^T$ ,  $Q$  orthogonal,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$
- define  $y = Q^T x$ ,  $\bar{b} = Q^T b$ ,  $y^* = Q^T x^*$
- in terms of  $y$ , we have

$$\begin{aligned} f(x) = \bar{f}(y) &= (1/2)x^T Q\Lambda Q^T x - b^T QQ^T x \\ &= (1/2)y^T \Lambda y - \bar{b}^T y \\ &= \sum_{i=1}^n ((1/2)\lambda_i y_i^2 - \bar{b}_i y_i) \end{aligned}$$

so  $y_i^* = \bar{b}_i/\lambda_i$ ,  $f^* = -(1/2) \sum_{i=1}^n \bar{b}_i^2/\lambda_i$

Krylov sequence in terms of  $y$

$$y^{(k)} = \underset{y \in \bar{\mathcal{K}}_k}{\operatorname{argmin}} \bar{f}(y), \quad \bar{\mathcal{K}}_k = \operatorname{span}\{\bar{b}, \Lambda \bar{b}, \dots, \Lambda^{k-1} \bar{b}\}$$

$$y_i^{(k)} = p_k(\lambda_i) \bar{b}_i, \quad \deg p_k < k$$

$$p_k = \underset{\deg p < k}{\operatorname{argmin}} \sum_{i=1}^n \bar{b}_i^2 \left( (1/2)\lambda_i p(\lambda_i)^2 - p(\lambda_i) \right)$$

$$\begin{aligned}
f(x^{(k)}) - f^\star &= \bar{f}(y^{(k)}) - f^\star \\
&= \min_{\deg p < k} (1/2) \sum_{i=1}^n \bar{b}_i^2 \frac{(\lambda_i p(\lambda_i) - 1)^2}{\lambda_i} \\
&= \min_{\deg p < k} (1/2) \sum_{i=1}^n \bar{y}_i^{\star 2} \lambda_i (\lambda_i p(\lambda_i) - 1)^2 \\
&= \min_{\deg q \leq k, q(0)=1} (1/2) \sum_{i=1}^n \bar{y}_i^{\star 2} \lambda_i q(\lambda_i)^2 \\
&= \min_{\deg q \leq k, q(0)=1} (1/2) \sum_{i=1}^n \bar{b}_i^2 \frac{q(\lambda_i)^2}{\lambda_i}
\end{aligned}$$

$$\begin{aligned}\tau_k &= \frac{\min_{\deg q \leq k, q(0)=1} \sum_{i=1}^n \bar{y}_i^{*2} \lambda_i q(\lambda_i)^2}{\sum_{i=1}^n \bar{y}_i^{*2} \lambda_i} \\ &\leq \min_{\deg q \leq k, q(0)=1} \left( \max_{i=1,\dots,n} q(\lambda_i)^2 \right)\end{aligned}$$

- if there is a polynomial  $q$  of degree  $k$ , with  $q(0) = 1$ , that is small on the spectrum of  $A$ , then  $f(x^{(k)}) - f^*$  is small
- if eigenvalues are clustered in  $k$  groups, then  $y^{(k)}$  is a good approximate solution
- if solution  $x^*$  is approximately a linear combination of  $k$  eigenvectors of  $A$ , then  $y^{(k)}$  is a good approximate solution

## A bound on convergence rate

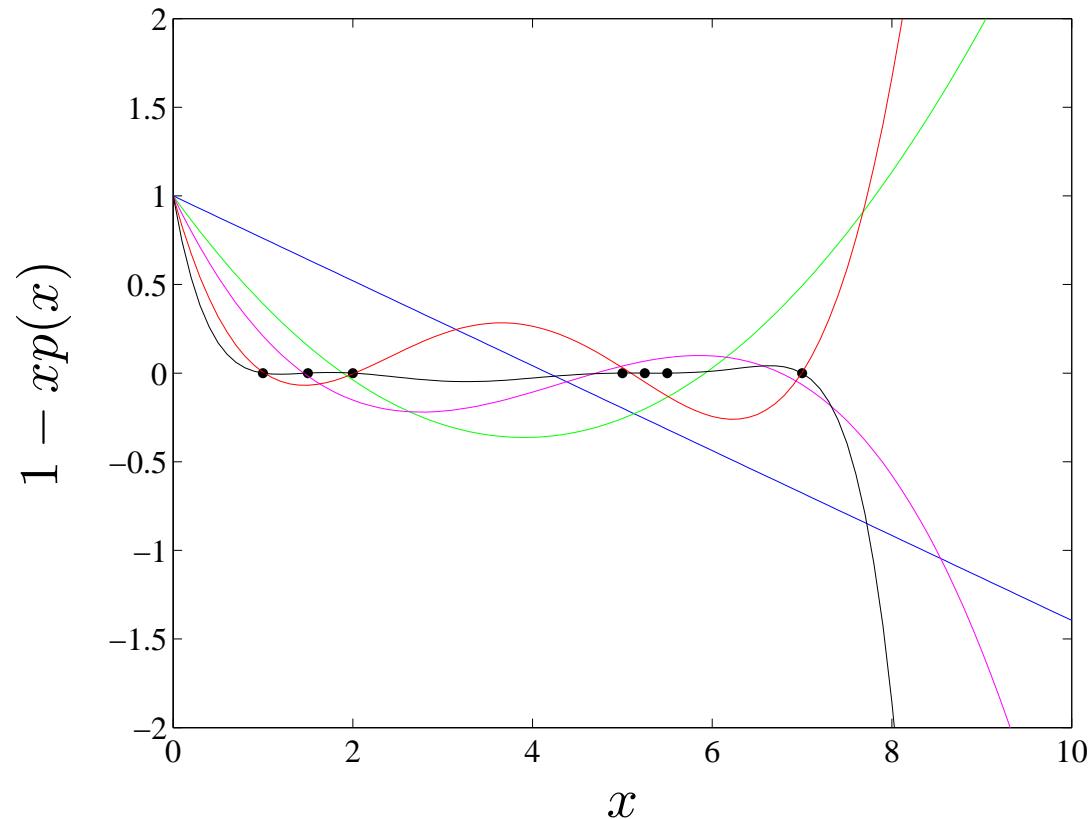
- taking  $q$  as Chebyshev polynomial of degree  $k$ , that is small on interval  $[\lambda_{\min}, \lambda_{\max}]$ , we get

$$\tau_k \leq \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k, \quad \kappa = \lambda_{\max}/\lambda_{\min}$$

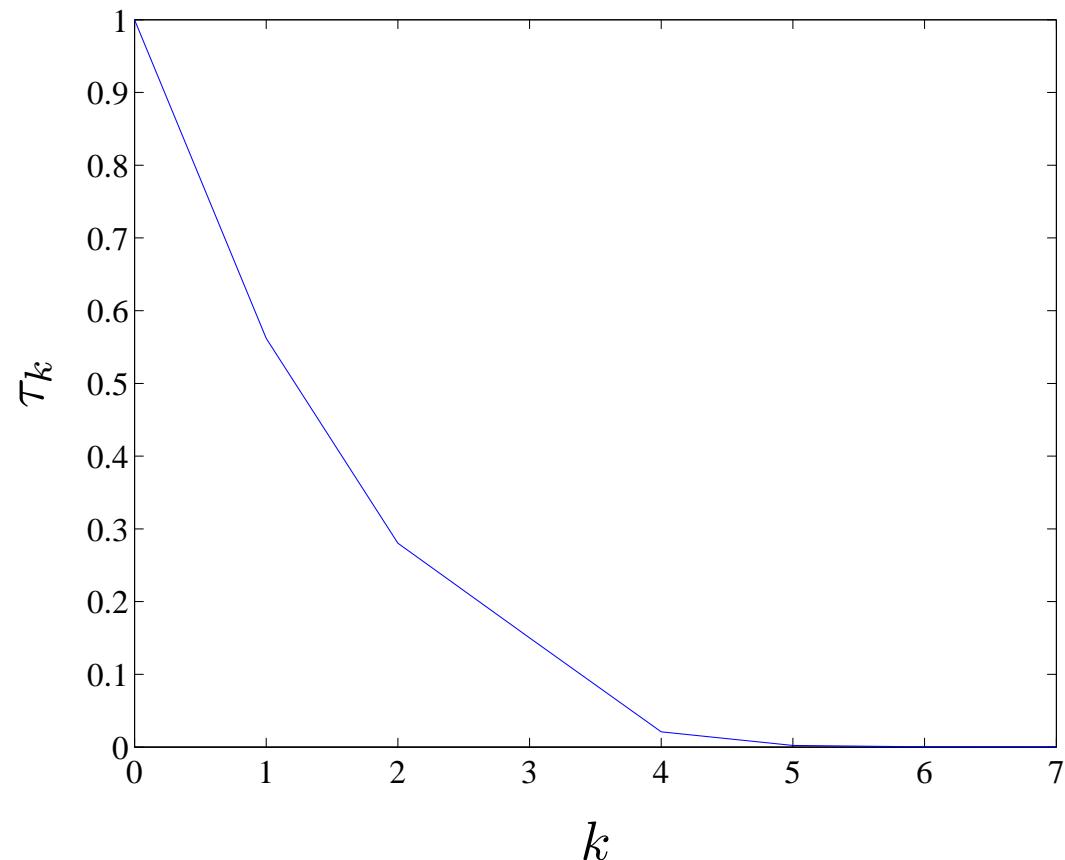
- convergence can be much faster than this, if spectrum of  $A$  is spread but clustered

## Small example

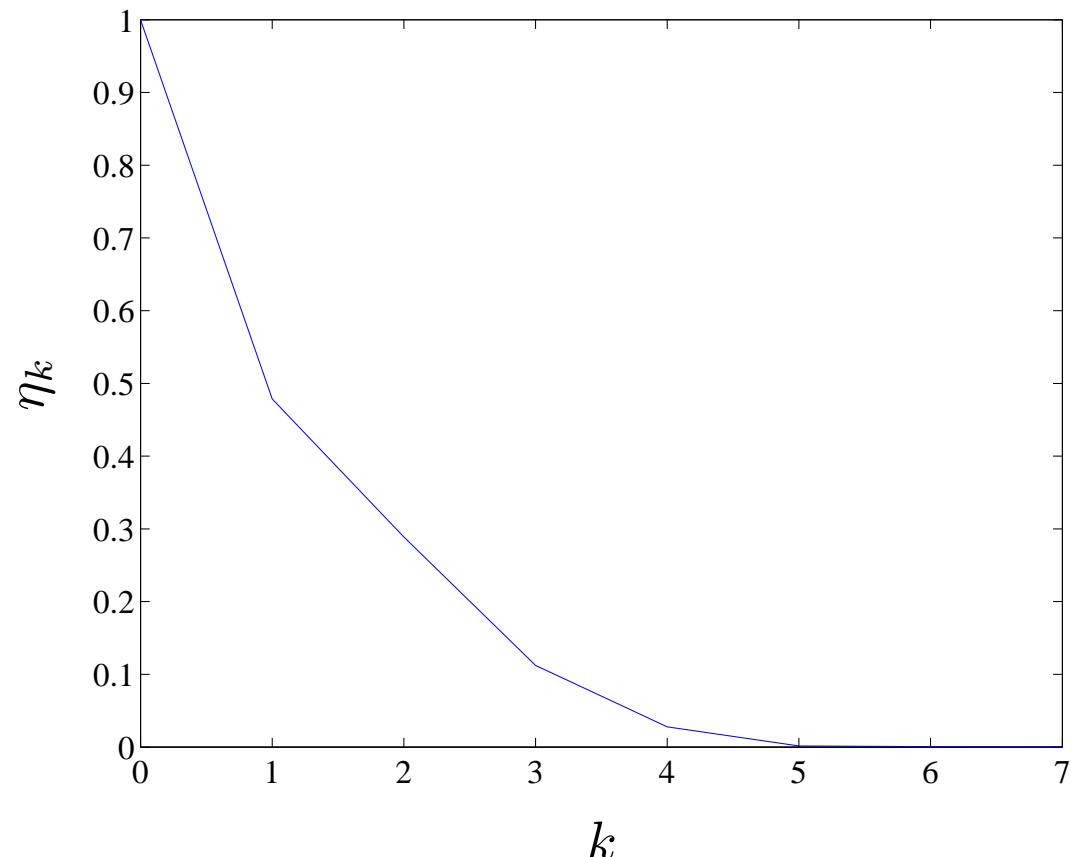
$A \in \mathbb{R}^{7 \times 7}$ , spectrum shown as filled circles;  $p_1, p_2, p_3, p_4$ , and  $p_7$  shown



# Convergence



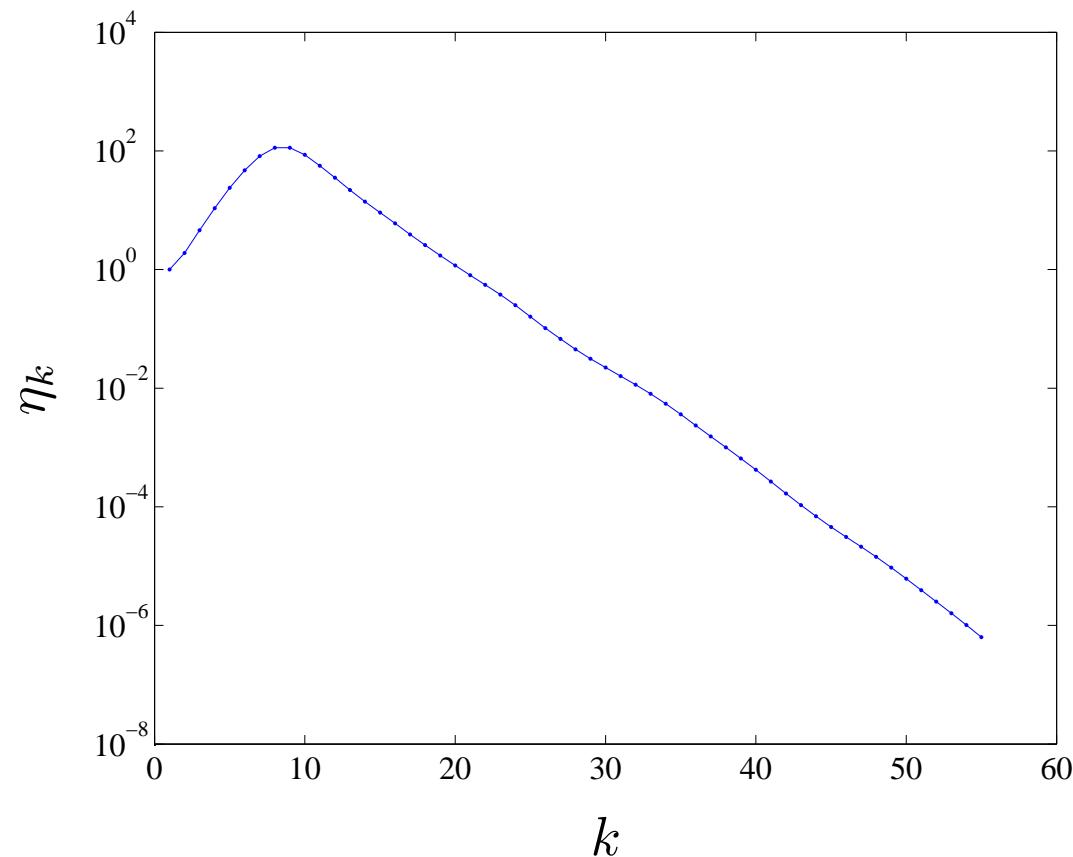
## Residual convergence



## Larger example

- solve  $Gv = i$ , resistor network with  $10^5$  nodes
- average node degree 10; around  $10^6$  nonzeros in  $G$
- random topology with one grounded node
- nonzero branch conductances uniform on  $[0, 1]$
- external current  $i$  uniform on  $[0, 1]$
- sparse Cholesky factorization of  $G$  requires too much memory

## Residual convergence



# CG algorithm

(follows C. T. Kelley)

$$x := 0, \quad r := b, \quad \rho_0 := \|r\|^2$$

for  $k = 1, \dots, N_{\max}$

quit if  $\sqrt{\rho_{k-1}} \leq \epsilon \|b\|$

if  $k = 1$  then  $p := r$ ; else  $p := r + (\rho_{k-1}/\rho_{k-2})p$

$w := Ap$

$\alpha := \rho_{k-1}/p^T w$

$x := x + \alpha p$

$r := r - \alpha w$

$\rho_k := \|r\|^2$

## Efficient matrix-vector multiply

- sparse  $A$
- structured (*e.g.*, sparse) plus low rank
- products of easy-to-multiply matrices
- fast transforms (FFT, wavelet, . . . )
- inverses of lower/upper triangular (by forward/backward substitution)
- fast Gauss transform, for  $A_{ij} = \exp(-\|v_i - v_j\|^2/\sigma^2)$  (via multipole)

## Shifting

- suppose we have guess  $\hat{x}$  of solution  $x^*$
- we can solve  $Az = b - A\hat{x}$  using CG, then get  $x^* = \hat{x} + z$
- in this case  $x^{(k)} = \hat{x} + z^{(k)} = \underset{x \in \hat{x} + \mathcal{K}_k}{\operatorname{argmin}} f(x)$   
 $(\hat{x} + \mathcal{K}_k \text{ is called } \textit{shifted Krylov subspace})$
- same as initializing CG alg with  $x := \hat{x}$ ,  $r := b - Ax$
- good for ‘warm start’, *i.e.*, solving  $Ax = b$  starting from a good initial guess (*e.g.*, the solution of another system  $\tilde{A}x = \tilde{b}$ , with  $A \approx \tilde{A}$ ,  $b \approx \tilde{b}$ )

## Preconditioned conjugate gradient algorithm

- idea: apply CG after linear change of coordinates  $x = Ty$ ,  $\det T \neq 0$
- use CG to solve  $T^T ATy = T^T b$ ; then set  $x^* = T^{-1}y^*$
- $T$  or  $M = TT^T$  is called *preconditioner*
- in naive implementation, each iteration requires multiplies by  $T$  and  $T^T$  (and  $A$ ); also need to compute  $x^* = T^{-1}y^*$  at end
- can re-arrange computation so each iteration requires one multiply by  $M$  (and  $A$ ), and no final solve  $x^* = T^{-1}y^*$
- called *preconditioned conjugate gradient* (PCG) algorithm

## Choice of preconditioner

- if spectrum of  $T^T AT$  (which is the same as the spectrum of  $MA$ ) is clustered, PCG converges fast
- extreme case:  $M = A^{-1}$
- trade-off between enhanced convergence, and extra cost of multiplication by  $M$  at each step
- goal is to find  $M$  that is cheap to multiply, and approximate inverse of  $A$  (or at least has a more clustered spectrum than  $A$ )

## Some generic preconditioners

- diagonal:  $M = \text{diag}(1/A_{11}, \dots, 1/A_{nn})$
- incomplete/approximate Cholesky factorization: use  $M = \hat{A}^{-1}$ , where  $\hat{A} = \hat{L}\hat{L}^T$  is an approximation of  $A$  with cheap Cholesky factorization
  - compute Cholesky factorization of  $\hat{A}$ ,  $\hat{A} = \hat{L}\hat{L}^T$
  - at each iteration, compute  $Mz = \hat{L}^{-T}\hat{L}^{-1}z$  via forward/backward substitution
- examples
  - $\hat{A}$  is central  $k$ -wide band of  $A$
  - $\hat{L}$  obtained by sparse Cholesky factorization of  $A$ , ignoring small elements in  $A$ , or refusing to create excessive fill-in

# Preconditioned conjugate gradient

(with preconditioner  $M \approx A^{-1}$  (hopefully))

$$x := 0, \quad r := b - Ax_0, \quad p := r \quad z := Mr, \quad \rho_1 := r^T z$$

for  $k = 1, \dots, N_{\max}$

quit if  $\sqrt{\rho_k} \leq \epsilon \|b\|_2$  or  $\|r\| \leq \epsilon \|b\|_2$

$$w := Ap$$

$$\alpha := \frac{\rho_k}{w^T p}$$

$$x := x + \alpha p$$

$$r := r - \alpha w$$

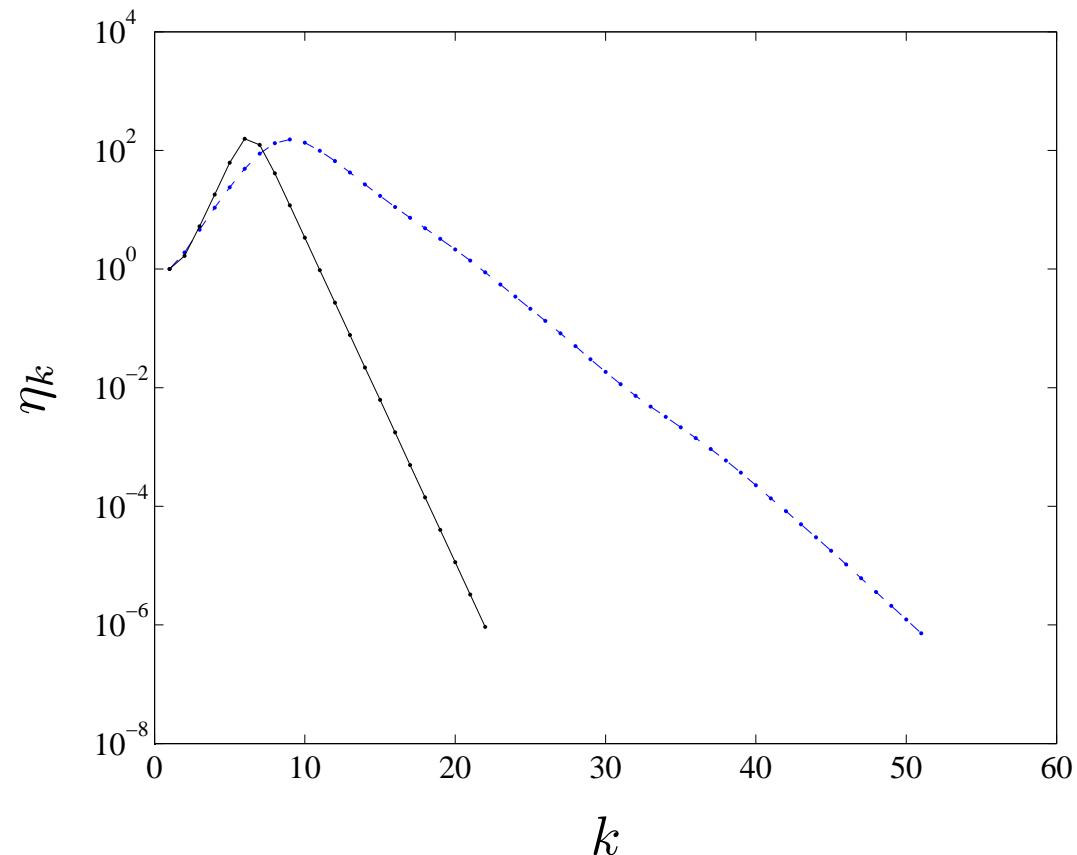
$$z := Mr$$

$$\rho_{k+1} := z^T r$$

$$p := z + \frac{\rho_{k+1}}{\rho_k} p$$

## Larger example

residual convergence with and without diagonal preconditioning



## CG summary

- in theory (with exact arithmetic) converges to solution in  $n$  steps
  - the bad news: due to numerical round-off errors, can take more than  $n$  steps (or fail to converge)
  - the good news: with luck (*i.e.*, good spectrum of  $A$ ), can get good approximate solution in  $\ll n$  steps
- each step requires  $z \rightarrow Az$  multiplication
  - can exploit a variety of structure in  $A$
  - in many cases, never form or store the matrix  $A$
- compared to direct (factor-solve) methods, CG is less reliable, data dependent; often requires good (problem-dependent) preconditioner
- but, when it works, can solve extremely large systems

# Truncated Newton Method

- approximate Newton methods
- truncated Newton methods
- truncated Newton interior-point methods

## Newton's method

- minimize convex  $f : \mathbf{R}^n \rightarrow \mathbf{R}$
- Newton step  $\Delta x_{\text{nt}}$  found from (SPD) Newton system

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

using Cholesky factorization

- backtracking line search on function value  $f(x)$  or norm of gradient  $\|\nabla f(x)\|$
- stopping criterion based on Newton decrement  $\lambda^2/2 = -\nabla f(x)^T \Delta x_{\text{nt}}$  or norm of gradient  $\|\nabla f(x)\|$

## Approximate or inexact Newton methods

- use as search direction an *approximate solution*  $\Delta x$  of Newton system
- idea: no need to compute  $\Delta x_{\text{nt}}$  exactly; only need a good enough search direction
- number of iterations may increase, but if effort per iteration is smaller than for Newton, we win
- examples:
  - solve  $\hat{H}\Delta x = -\nabla f(x)$ , where  $\hat{H}$  is diagonal or band of  $\nabla^2 f(x)$
  - factor  $\nabla^2 f(x)$  every  $k$  iterations and use most recent factorization

## Truncated Newton methods

- approximately solve Newton system using CG or PCG, terminating (sometimes way) early
- also called *Newton-iterative methods*; related to limited memory Newton (or BFGS)
- total effort is measured by cumulative sum of CG steps done
- for good performance, need to tune CG stopping criterion, to use just enough steps to get a good enough search direction
- less reliable than Newton's method, but (with good tuning, good preconditioner, fast  $z \rightarrow \nabla^2 f(x)z$  method, and some luck) can handle very large problems

## Truncated Newton method

- backtracking line search on  $\|\nabla f(x)\|$
- typical CG termination rule: stop after  $N_{\max}$  steps or

$$\eta = \frac{\|\nabla^2 f(x) \Delta x + \nabla f(x)\|}{\|\nabla f(x)\|} \leq \epsilon_{\text{pcg}}$$

- with simple rules,  $N_{\max}$ ,  $\epsilon_{\text{pcg}}$  are constant
- more sophisticated rules adapt  $N_{\max}$  or  $\epsilon_{\text{pcg}}$  as algorithm proceeds (based on, e.g., value of  $\|\nabla f(x)\|$ , or progress in reducing  $\|\nabla f(x)\|$ )  
 $\eta = \min(0.1, \|\nabla f(x)\|^{1/2})$  guarantees (with large  $N_{\max}$ ) superlinear convergence

## CG initialization

- we use CG to approximately solve  $\nabla^2 f(x)\Delta x + \nabla f(x) = 0$
- if we initialize CG with  $\Delta x = 0$ 
  - after one CG step,  $\Delta x$  points in direction of negative gradient (so,  $N_{\max} = 1$  results in gradient method)
  - all CG iterates are descent directions for  $f$
- another choice: initialize with  $\Delta x = \Delta x_{\text{prev}}$ , the previous search step
  - initial CG iterates need not be descent directions
  - but can give advantage when  $N_{\max}$  is small

- simple scheme: if  $\Delta x_{\text{prev}}$  is a descent direction ( $\Delta x_{\text{prev}}^T \nabla f(x) < 0$ )  
start CG from

$$\Delta x = \frac{-\Delta x_{\text{prev}}^T \nabla f(x)}{\Delta x_{\text{prev}}^T \nabla^2 f(x) \Delta x_{\text{prev}}} \Delta x_{\text{prev}}$$

otherwise start CG from  $\Delta x = 0$

## Example

$\ell_2$ -regularized logistic regression

$$\text{minimize } f(w) = (1/m) \sum_{i=1}^m \log(1 + \exp(-b_i x_i^T w)) + \sum_{i=1}^n \lambda_i w_i^2$$

- variable is  $w \in \mathbf{R}^n$
- problem data are  $x_i \in \mathbf{R}^n$ ,  $b_i \in \{-1, 1\}$ ,  $i = 1, \dots, m$ , and regularization parameter  $\lambda \in \mathbf{R}_+^n$
- $n$  is number of features;  $m$  is number of samples/observations

## Hessian and gradient

$$\nabla^2 f(w) = A^T D A + 2\Lambda, \quad \nabla f(w) = A^T g + 2\Lambda w$$

where

$$A = [b_1 x_1 \cdots b_m x_m]^T, \quad D = \text{diag}(h), \quad \Lambda = \text{diag}(\lambda)$$

$$\begin{aligned} g_i &= -(1/m)/(1 + \exp(Aw)_i) \\ h_i &= (1/m) \exp(Aw)_i / (1 + \exp(Aw)_i)^2 \end{aligned}$$

we never form  $\nabla^2 f(w)$ ; we carry out multiplication  $z \rightarrow \nabla^2 f(w)z$  as

$$\nabla^2 f(w)z = (A^T D A + 2\Lambda)z = A^T (D(Az)) + 2\Lambda z$$

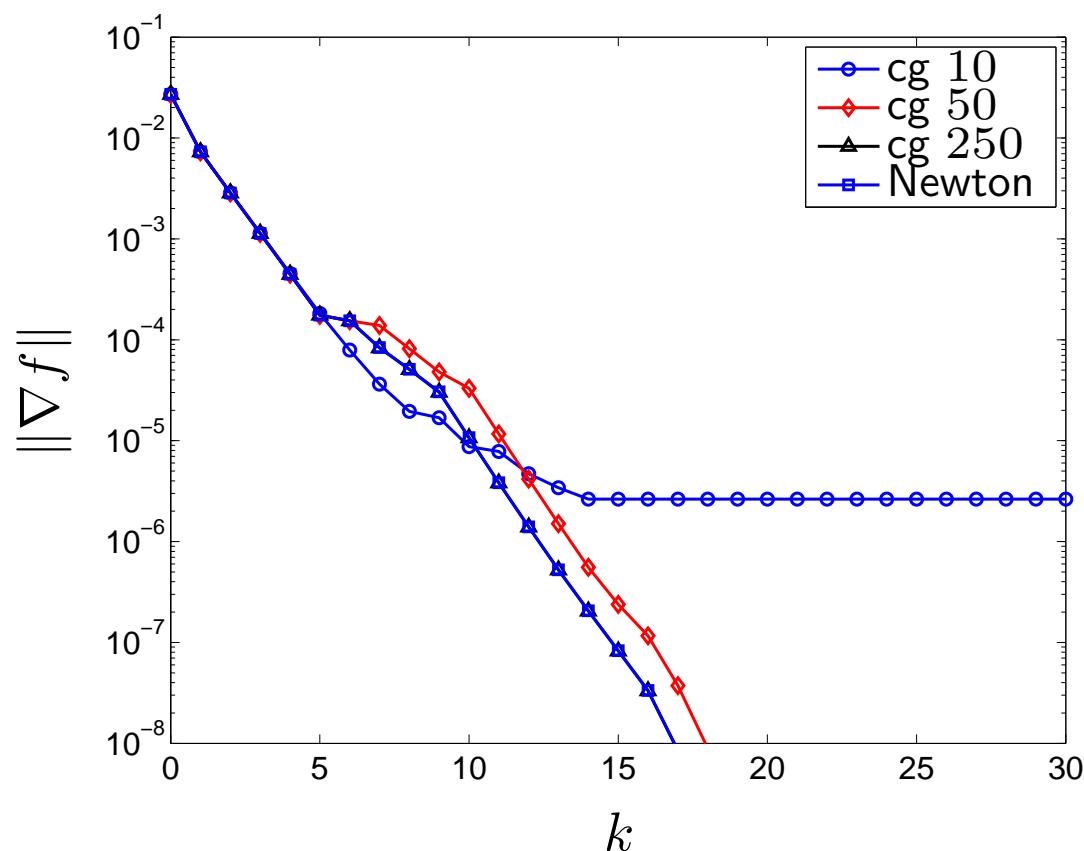
## Problem instance

- $n = 10000$  features,  $m = 20000$  samples (10000 each with  $b_i = \pm 1$ )
- $x_i$  have random sparsity pattern, with around 10 nonzero entries
- nonzero entries in  $x_i$  drawn from  $\mathcal{N}(b_i, 1)$
- $\lambda_i = 10^{-8}$
- around 500000 nonzeros in  $\nabla^2 f$ , and 30M nonzeros in Cholesky factor

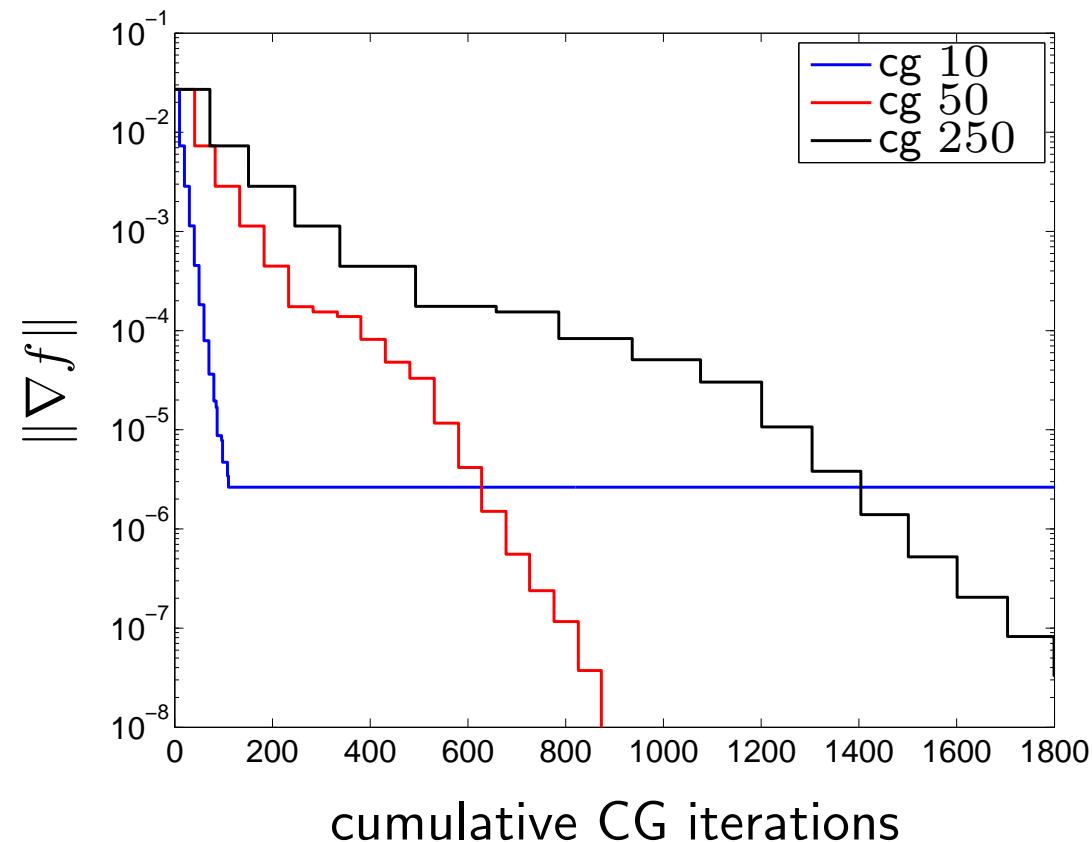
## Methods

- Newton (using Cholesky factorization of  $\nabla^2 f(w)$ )
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\max} = 10$
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\max} = 50$
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\max} = 250$

# Convergence versus iterations



## Convergence versus cumulative CG steps

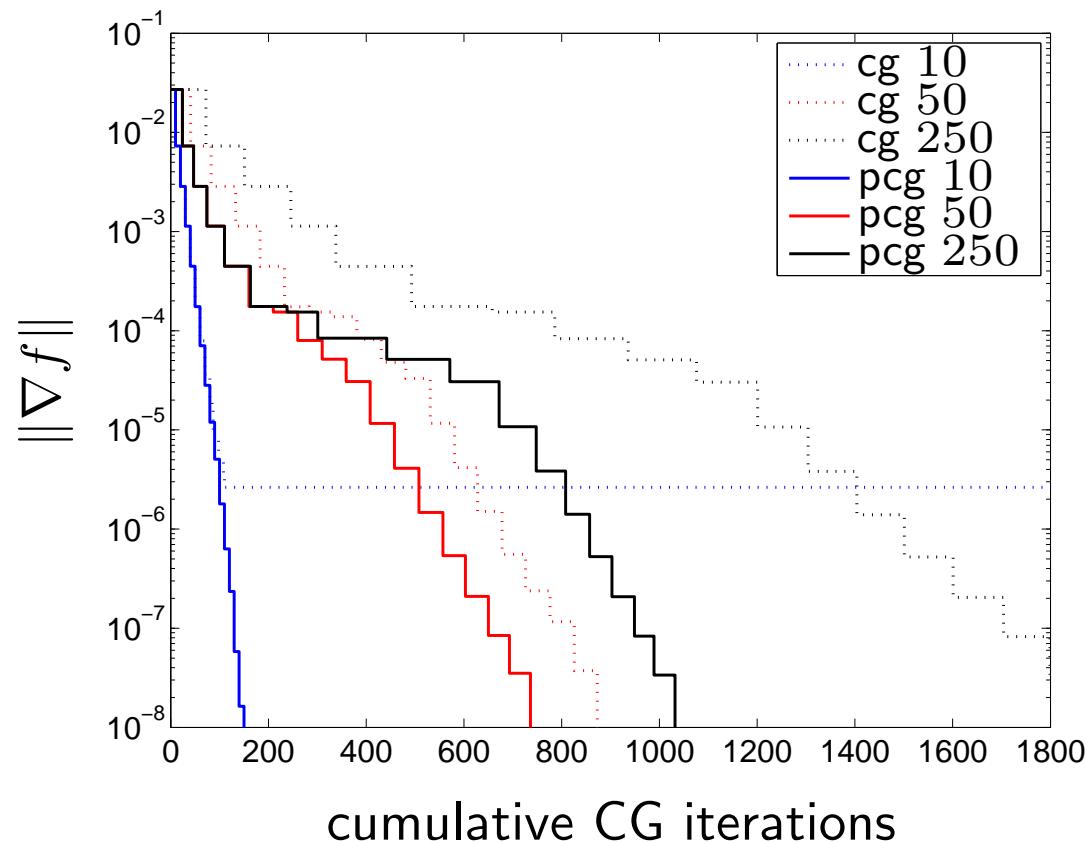


- convergence of exact Newton, and truncated Newton methods with  $N_{\max} = 50$  and 250 essentially the same, in terms of iterations
- in terms elapsed time (and memory!), truncated Newton methods far better than Newton
- truncated Newton with  $N_{\max} = 10$  seems to jam near  $\|\nabla f(w)\| \approx 10^{-6}$
- times (on AMD270 2GHz, 12GB, Linux) in sec:

method	$\ \nabla f(w)\  \leq 10^{-5}$	$\ \nabla f(w)\  \leq 10^{-8}$
Newton	1600	2600
cg 10	4	—
cg 50	17	26
cg 250	35	54

## Truncated PCG Newton method

approximate search direction found via diagonally preconditioned PCG



- diagonal preconditioning allows  $N_{\max} = 10$  to achieve high accuracy; speeds up other truncated Newton methods
- times:

method	$\ \nabla f(w)\  \leq 10^{-5}$	$\ \nabla f(w)\  \leq 10^{-8}$
Newton	1600	2600
cg 10	4	—
cg 50	17	26
cg 250	35	54
pcg 10	3	5
pcg 50	13	24
pcg 250	23	34

- speedups of 1600:3, 2600:5 are not bad  
(and we really didn't do much tuning . . . )

## Extensions

- can extend to (infeasible start) Newton's method with equality constraints
- since we don't use exact Newton step, equality constraints not guaranteed to hold after finite number of steps (but  $\|r_p\| \rightarrow 0$ )
- can use for barrier, primal-dual methods

## Truncated Newton interior-point methods

- use truncated Newton method to compute search direction in interior-point method
- tuning PCG parameters for optimal performance on a given problem class is tricky, since linear systems in interior-point methods often become ill-conditioned as algorithm proceeds
- but can work well (with luck, good preconditioner)

## Network rate control

rate control problem

$$\begin{aligned} \text{minimize} \quad & -U(f) = -\sum_{j=1}^n \log f_j \\ \text{subject to} \quad & Rf \preceq c \end{aligned}$$

with variable  $f$

- $f \in \mathbf{R}_{++}^n$  is vector of flow rates
- $U(f) = \sum_{j=1}^n \log f_j$  is flow utility
- $R \in \mathbf{R}^{m \times n}$  is route matrix ( $R_{ij} \in \{0, 1\}$ )
- $c \in \mathbf{R}^m$  is vector of link capacities

## Dual rate control problem

dual problem

$$\begin{aligned} & \text{maximize} && g(\lambda) = n - c^T \lambda + \sum_{i=1}^m \log(r_i^T \lambda) \\ & \text{subject to} && \lambda \succeq 0 \end{aligned}$$

with variable  $\lambda \in \mathbf{R}^m$

duality gap

$$\begin{aligned} \eta &= -U(f) - g(\lambda) \\ &= -\sum_{j=1}^n \log f_j - n + c^T \lambda - \sum_{i=1}^m \log(r_i^T \lambda) \end{aligned}$$

## Primal-dual search direction (BV §11.7)

primal-dual search direction  $\Delta f$ ,  $\Delta \lambda$  given by

$$(D_1 + R^T D_2 R) \Delta f = g_1 - (1/t) R^T g_2, \quad \Delta \lambda = D_2 R \Delta f - \lambda + (1/t) g_2$$

where  $s = c - Rf$ ,

$$D_1 = \text{diag}(1/f_1^2, \dots, 1/f_n^2), \quad D_2 = \text{diag}(\lambda_1/s_1, \dots, \lambda_m/s_m)$$

$$g_1 = (1/f_1, \dots, 1/f_n), \quad g_2 = (1/s_1, \dots, 1/s_m)$$

## Truncated Newton primal-dual algorithm

primal-dual residual:

$$r = (r_{\text{dual}}, r_{\text{cent}}) = (-g_2 + R^T \lambda, \text{diag}(\lambda)s - (1/t)\mathbf{1})$$

**given**  $f$  with  $Rf \prec c$ ;  $\lambda \succ 0$

**while**  $\eta/g(\lambda) > \epsilon$

$$t := \mu m / \eta$$

compute  $\Delta f$  using PCG as approximate solution of

$$(D_1 + R^T D_2 R) \Delta f = g_1 - (1/t) R^T g_2$$

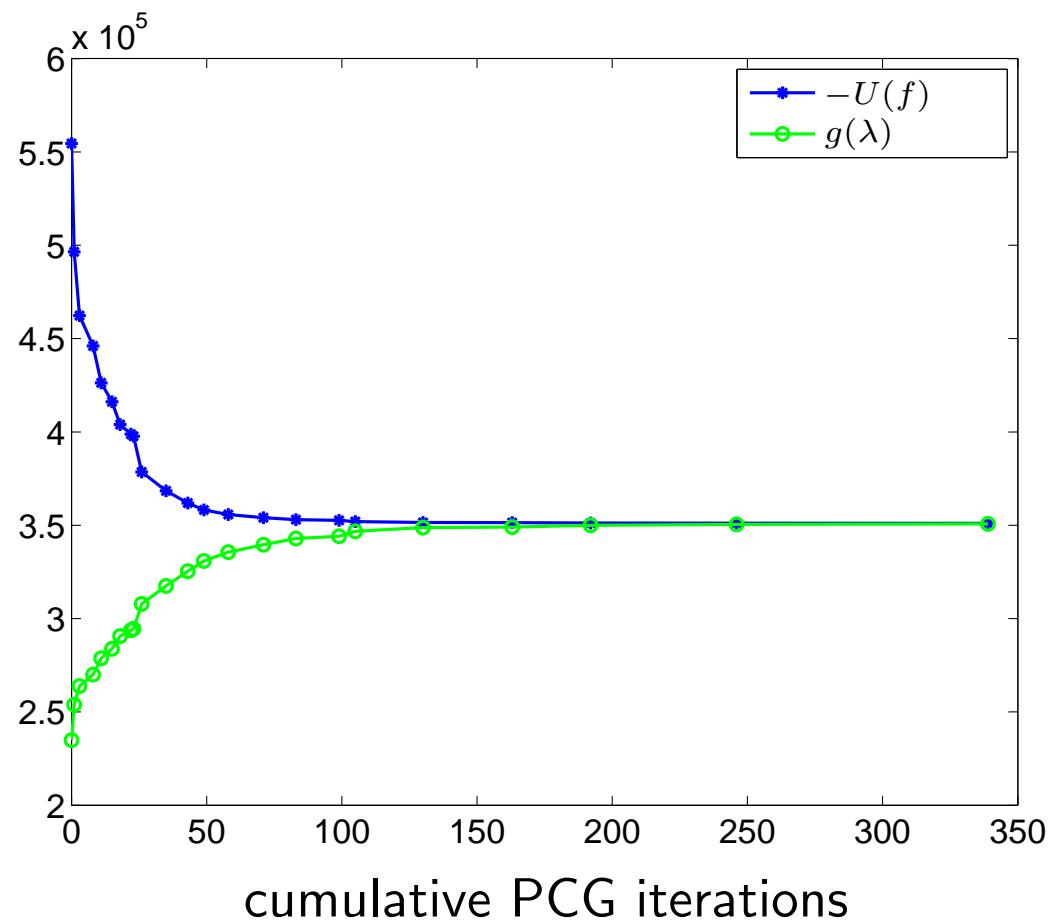
$$\Delta \lambda := D_2 R \Delta f - \lambda + (1/t) g_2$$

carry out line search on  $\|r\|_2$ , and update:

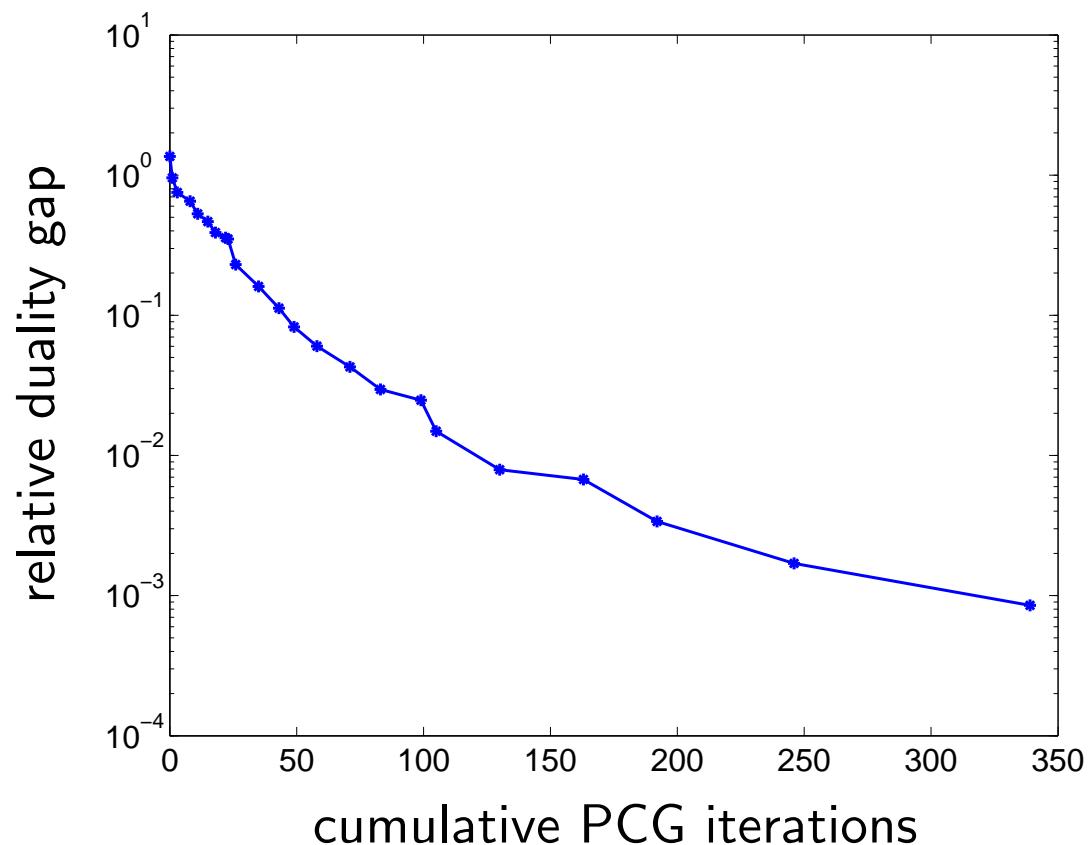
$$f := f + \gamma \Delta f, \lambda := \lambda + \gamma \Delta \lambda$$

- problem instance
  - $m = 200000$  links,  $n = 100000$  flows
  - average of 12 links per flow, 6 flows per link
  - capacities random, uniform on  $[0.1, 1]$
- algorithm parameters
  - truncated Newton with  $\epsilon_{cg} = \min(0.1, \eta/g(\lambda))$ ,  $N_{\max} = 200$  ( $N_{\max}$  never reached)
  - diagonal preconditioner
  - warm start
  - $\mu = 2$
  - $\epsilon = 0.001$  (*i.e.*, solve to guaranteed 0.1% suboptimality)

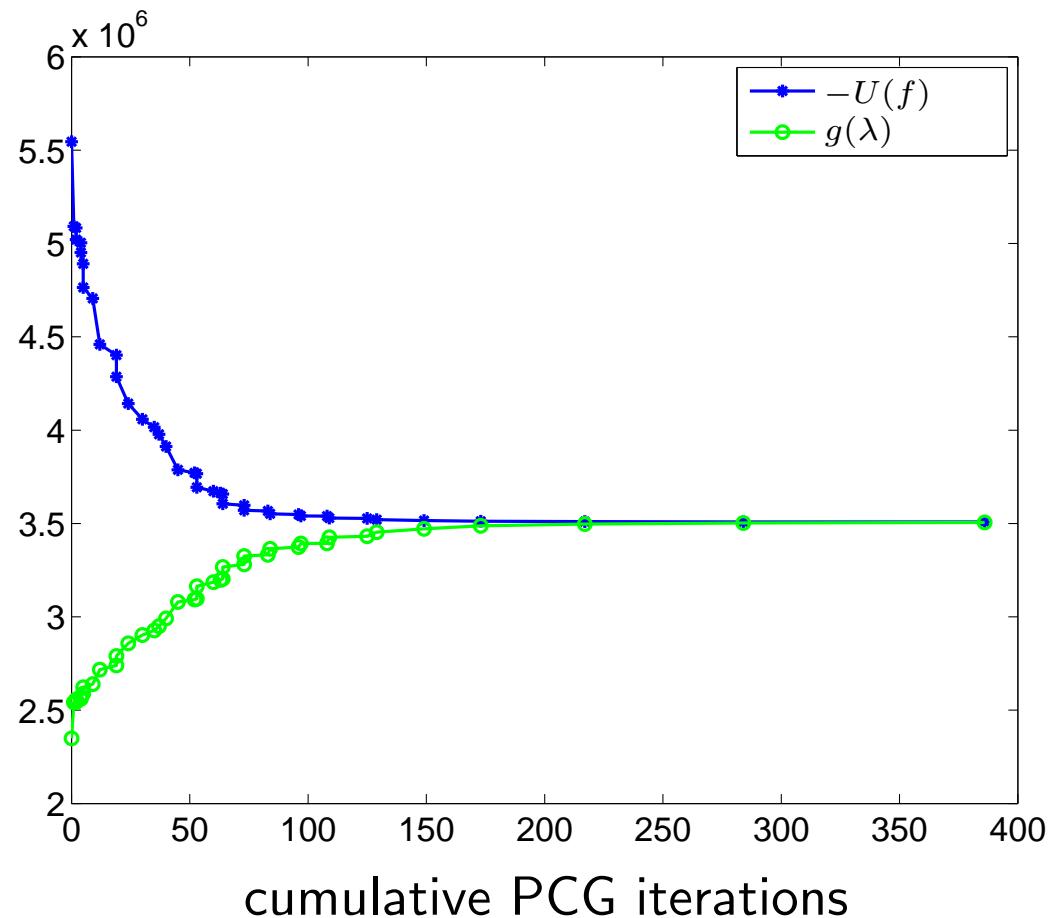
## Primal and dual objective evolution



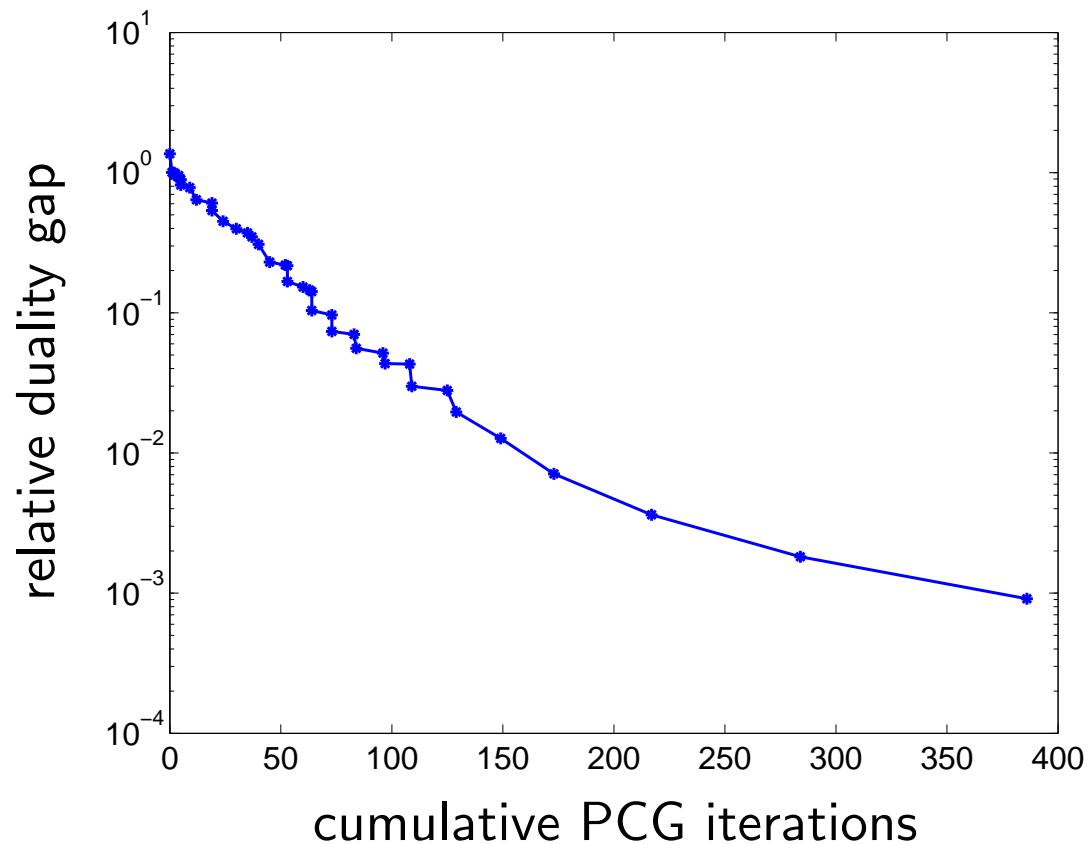
## Relative duality gap evolution



## Primal and dual objective evolution ( $n = 10^6$ )



## Relative duality gap evolution ( $n = 10^6$ )



# $\ell_1$ -norm Methods for Convex-Cardinality Problems

- problems involving cardinality
- the  $\ell_1$ -norm heuristic
- convex relaxation and convex envelope interpretations
- examples
- recent results

## $\ell_1$ -norm heuristics for cardinality problems

- cardinality problems arise often, but are hard to solve exactly
- a simple heuristic, that relies on  $\ell_1$ -norm, seems to work well
- used for many years, in many fields
  - sparse design
  - LASSO, robust estimation in statistics
  - support vector machine (SVM) in machine learning
  - total variation reconstruction in signal processing, geophysics
  - compressed sensing
- new theoretical results guarantee the method works, at least for a few problems

# Cardinality

- the **cardinality** of  $x \in \mathbf{R}^n$ , denoted  $\mathbf{card}(x)$ , is the number of nonzero components of  $x$
- **card** is separable; for scalar  $x$ ,  $\mathbf{card}(x) = \begin{cases} 0 & x = 0 \\ 1 & x \neq 0 \end{cases}$
- **card** is quasiconcave on  $\mathbf{R}_+^n$  (but not  $\mathbf{R}^n$ ) since

$$\mathbf{card}(x + y) \geq \min\{\mathbf{card}(x), \mathbf{card}(y)\}$$

holds for  $x, y \succeq 0$

- but otherwise has no convexity properties
- arises in many problems

## General convex-cardinality problems

a **convex-cardinality problem** is one that would be convex, except for appearance of **card** in objective or constraints

examples (with  $\mathcal{C}$ ,  $f$  convex):

- convex minimum cardinality problem:

$$\begin{aligned} & \text{minimize} && \mathbf{card}(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

- convex problem with cardinality constraint:

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C}, \quad \mathbf{card}(x) \leq k \end{aligned}$$

## Solving convex-cardinality problems

convex-cardinality problem with  $x \in \mathbf{R}^n$

- if we fix the sparsity pattern of  $x$  (*i.e.*, which entries are zero/nonzero) we get a convex problem
- by solving  $2^n$  convex problems associated with all possible sparsity patterns, we can solve convex-cardinality problem (possibly practical for  $n \leq 10$ ; not practical for  $n > 15$  or so . . . )
- general convex-cardinality problem is (NP-) hard
- can solve globally by branch-and-bound
  - can work for particular problem instances (with some luck)
  - in worst case reduces to checking all (or many of)  $2^n$  sparsity patterns

## Boolean LP as convex-cardinality problem

- Boolean LP:

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax \preceq b, \quad x_i \in \{0, 1\} \end{aligned}$$

includes many famous (hard) problems, e.g., 3-SAT, traveling salesman

- can be expressed as

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax \preceq b, \quad \mathbf{card}(x) + \mathbf{card}(1 - x) \leq n \end{aligned}$$

since  $\mathbf{card}(x) + \mathbf{card}(1 - x) \leq n \iff x_i \in \{0, 1\}$

- conclusion: general convex-cardinality problem is hard

## Sparse design

$$\begin{aligned} & \text{minimize} && \mathbf{card}(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

- find sparsest design vector  $x$  that satisfies a set of specifications
- zero values of  $x$  simplify design, or correspond to components that aren't even needed
- examples:
  - FIR filter design (zero coefficients reduce required hardware)
  - antenna array beamforming (zero coefficients correspond to unneeded antenna elements)
  - truss design (zero coefficients correspond to bars that are not needed)
  - wire sizing (zero coefficients correspond to wires that are not needed)

## Sparse modeling / regressor selection

fit vector  $b \in \mathbf{R}^m$  as a linear combination of  $k$  regressors (chosen from  $n$  possible regressors)

$$\begin{aligned} & \text{minimize} && \|Ax - b\|_2 \\ & \text{subject to} && \mathbf{card}(x) \leq k \end{aligned}$$

- gives  $k$ -term model
- chooses subset of  $k$  regressors that (together) best fit or explain  $b$
- can solve (in principle) by trying all  $\binom{n}{k}$  choices
- variations:
  - minimize  $\mathbf{card}(x)$  subject to  $\|Ax - b\|_2 \leq \epsilon$
  - minimize  $\|Ax - b\|_2 + \lambda \mathbf{card}(x)$

# Sparse signal reconstruction

- estimate signal  $x$ , given
  - noisy measurement  $y = Ax + v$ ,  $v \sim \mathcal{N}(0, \sigma^2 I)$  ( $A$  is known;  $v$  is not)
  - prior information  $\mathbf{card}(x) \leq k$
- maximum likelihood estimate  $\hat{x}_{\text{ml}}$  is solution of

$$\begin{aligned} &\text{minimize} && \|Ax - y\|_2 \\ &\text{subject to} && \mathbf{card}(x) \leq k \end{aligned}$$

## Estimation with outliers

- we have measurements  $y_i = a_i^T x + v_i + w_i$ ,  $i = 1, \dots, m$
- noises  $v_i \sim \mathcal{N}(0, \sigma^2)$  are independent
- only assumption on  $w$  is sparsity:  $\text{card}(w) \leq k$
- $\mathcal{B} = \{i \mid w_i \neq 0\}$  is set of bad measurements or *outliers*
- maximum likelihood estimate of  $x$  found by solving

$$\begin{array}{ll}\text{minimize} & \sum_{i \notin \mathcal{B}} (y_i - a_i^T x)^2 \\ \text{subject to} & |\mathcal{B}| \leq k\end{array}$$

with variables  $x$  and  $\mathcal{B} \subseteq \{1, \dots, m\}$

- equivalent to

$$\begin{array}{ll}\text{minimize} & \|y - Ax - w\|_2^2 \\ \text{subject to} & \text{card}(w) \leq k\end{array}$$

## Minimum number of violations

- set of convex inequalities

$$f_1(x) \leq 0, \dots, f_m(x) \leq 0, \quad x \in \mathcal{C}$$

- choose  $x$  to minimize the number of violated inequalities:

$$\begin{aligned} & \text{minimize} && \mathbf{card}(t) \\ & \text{subject to} && f_i(x) \leq t_i, \quad i = 1, \dots, m \\ & && x \in \mathcal{C}, \quad t \geq 0 \end{aligned}$$

- determining whether zero inequalities can be violated is (easy) convex feasibility problem

## Linear classifier with fewest errors

- given data  $(x_1, y_1), \dots, (x_m, y_m) \in \mathbf{R}^n \times \{-1, 1\}$
- we seek linear (affine) classifier  $y \approx \text{sign}(w^T x + v)$
- classification error corresponds to  $y_i(w^T x + v) \leq 0$
- to find  $w, v$  that give fewest classification errors:

$$\begin{aligned} & \text{minimize} && \text{card}(t) \\ & \text{subject to} && y_i(w^T x_i + v) + t_i \geq 1, \quad i = 1, \dots, m \end{aligned}$$

with variables  $w, v, t$  (we use homogeneity in  $w, v$  here)

## Smallest set of mutually infeasible inequalities

- given a set of mutually infeasible convex inequalities  
 $f_1(x) \leq 0, \dots, f_m(x) \leq 0$
- find smallest (cardinality) subset of these that is infeasible
- certificate of infeasibility is  $g(\lambda) = \inf_x (\sum_{i=1}^m \lambda_i f_i(x)) \geq 1, \lambda \succeq 0$
- to find smallest cardinality infeasible subset, we solve

$$\begin{aligned} & \text{minimize} && \mathbf{card}(\lambda) \\ & \text{subject to} && g(\lambda) \geq 1, \quad \lambda \succeq 0 \end{aligned}$$

(assuming some constraint qualifications)

## Portfolio investment with linear and fixed costs

- we use budget  $B$  to purchase (dollar) amount  $x_i \geq 0$  of stock  $i$
- trading fee is fixed cost plus linear cost:  $\beta \mathbf{card}(x) + \alpha^T x$
- budget constraint is  $\mathbf{1}^T x + \beta \mathbf{card}(x) + \alpha^T x \leq B$
- mean return on investment is  $\mu^T x$ ; variance is  $x^T \Sigma x$
- minimize investment variance (risk) with mean return  $\geq R_{\min}$ :

$$\begin{aligned} & \text{minimize} && x^T \Sigma x \\ & \text{subject to} && \mu^T x \geq R_{\min}, \quad x \succeq 0 \\ & && \mathbf{1}^T x + \beta \mathbf{card}(x) + \alpha^T x \leq B \end{aligned}$$

## Piecewise constant fitting

- fit corrupted  $x_{\text{cor}}$  by a piecewise constant signal  $\hat{x}$  with  $k$  or fewer jumps
- problem is convex once location (indices) of jumps are fixed
- $\hat{x}$  is piecewise constant with  $\leq k$  jumps  $\iff \text{card}(D\hat{x}) \leq k$ , where

$$D = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix} \in \mathbf{R}^{(n-1) \times n}$$

- as convex-cardinality problem:

$$\begin{aligned} & \text{minimize} && \|\hat{x} - x_{\text{cor}}\|_2 \\ & \text{subject to} && \text{card}(D\hat{x}) \leq k \end{aligned}$$

## Piecewise linear fitting

- fit  $x_{\text{cor}}$  by a piecewise linear signal  $\hat{x}$  with  $k$  or fewer kinks
- as convex-cardinality problem:

$$\begin{aligned} & \text{minimize} && \|\hat{x} - x_{\text{cor}}\|_2 \\ & \text{subject to} && \mathbf{card}(\nabla \hat{x}) \leq k \end{aligned}$$

where

$$\nabla = \begin{bmatrix} -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \end{bmatrix}$$

## $\ell_1$ -norm heuristic

- replace **card**( $z$ ) with  $\gamma\|z\|_1$ , or add regularization term  $\gamma\|z\|_1$  to objective
- $\gamma > 0$  is parameter used to achieve desired sparsity  
(when **card** appears in constraint, or as term in objective)
- more sophisticated versions use  $\sum_i w_i|z_i|$  or  $\sum_i w_i(z_i)_+ + \sum_i v_i(z_i)_-$ , where  $w, v$  are positive weights

## Example: Minimum cardinality problem

- start with (hard) minimum cardinality problem

$$\begin{aligned} & \text{minimize} && \mathbf{card}(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

( $\mathcal{C}$  convex)

- apply heuristic to get (easy)  $\ell_1$ -norm minimization problem

$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

## Example: Cardinality constrained problem

- start with (hard) cardinality constrained problem ( $f, \mathcal{C}$  convex)

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C}, \quad \mathbf{card}(x) \leq k \end{aligned}$$

- apply heuristic to get (easy)  $\ell_1$ -constrained problem

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C}, \quad \|x\|_1 \leq \beta \end{aligned}$$

or  $\ell_1$ -regularized problem

$$\begin{aligned} & \text{minimize} && f(x) + \gamma \|x\|_1 \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

$\beta, \gamma$  adjusted so that  $\mathbf{card}(x) \leq k$

## Polishing

- use  $\ell_1$  heuristic to find  $\hat{x}$  with required sparsity
- fix the sparsity pattern of  $\hat{x}$
- re-solve the (convex) optimization problem with this sparsity pattern to obtain final (heuristic) solution

## Interpretation as convex relaxation

- start with

$$\begin{aligned} & \text{minimize} && \text{card}(x) \\ & \text{subject to} && x \in \mathcal{C}, \quad \|x\|_\infty \leq R \end{aligned}$$

- equivalent to mixed Boolean convex problem

$$\begin{aligned} & \text{minimize} && \mathbf{1}^T z \\ & \text{subject to} && |x_i| \leq Rz_i, \quad i = 1, \dots, n \\ & && x \in \mathcal{C}, \quad z_i \in \{0, 1\}, \quad i = 1, \dots, n \end{aligned}$$

with variables  $x, z$

- now relax  $z_i \in \{0, 1\}$  to  $z_i \in [0, 1]$  to obtain

$$\begin{aligned} & \text{minimize} && \mathbf{1}^T z \\ & \text{subject to} && |x_i| \leq Rz_i, \quad i = 1, \dots, n \\ & && x \in \mathcal{C} \\ & && 0 \leq z_i \leq 1, \quad i = 1, \dots, n \end{aligned}$$

which is equivalent to

$$\begin{aligned} & \text{minimize} && (1/R)\|x\|_1 \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

the  $\ell_1$  heuristic

- optimal value of this problem is lower bound on original problem

## Interpretation via convex envelope

- convex envelope  $f^{\text{env}}$  of a function  $f$  on set  $\mathcal{C}$  is the largest convex function that is an underestimator of  $f$  on  $\mathcal{C}$
- $\text{epi}(f^{\text{env}}) = \text{Co}(\text{epi}(f))$
- $f^{\text{env}} = (f^*)^*$  (with some technical conditions)
- for  $x$  scalar,  $|x|$  is the convex envelope of  $\text{card}(x)$  on  $[-1, 1]$
- for  $x \in \mathbf{R}^n$  scalar,  $(1/R)\|x\|_1$  is convex envelope of  $\text{card}(x)$  on  $\{z \mid \|z\|_\infty \leq R\}$

## Weighted and asymmetric $\ell_1$ heuristics

- minimize  $\text{card}(x)$  over convex set  $\mathcal{C}$
- suppose we know lower and upper bounds on  $x_i$  over  $\mathcal{C}$

$$x \in \mathcal{C} \implies l_i \leq x_i \leq u_i$$

(best values for these can be found by solving  $2n$  convex problems)

- if  $u_i < 0$  or  $l_i > 0$ , then  $\text{card}(x_i) = 1$  (*i.e.*,  $x_i \neq 0$ ) for all  $x \in \mathcal{C}$
- assuming  $l_i < 0$ ,  $u_i > 0$ , convex relaxation and convex envelope interpretations suggest using

$$\sum_{i=1}^n \left( \frac{(x_i)_+}{u_i} + \frac{(x_i)_-}{-l_i} \right)$$

as surrogate (and also lower bound) for  $\text{card}(x)$

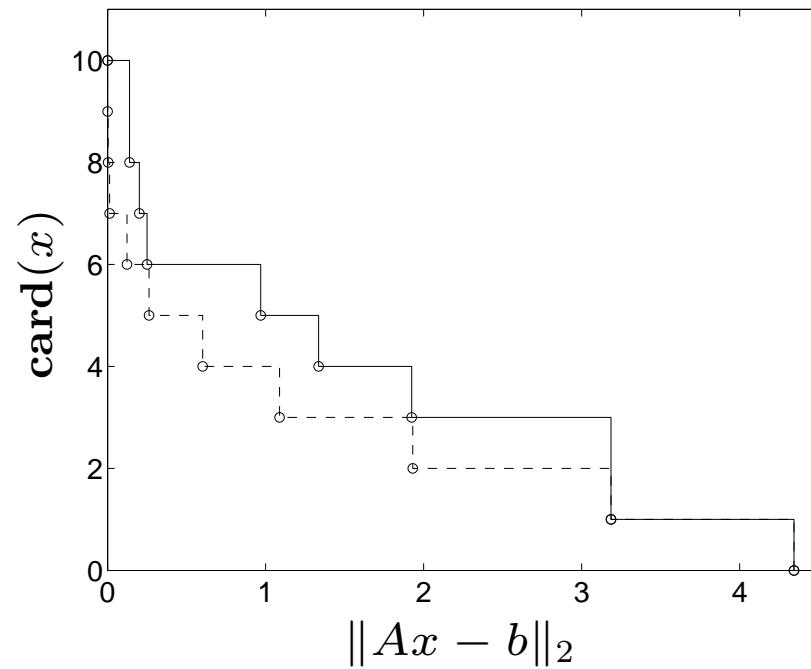
# Regressor selection

$$\begin{aligned} & \text{minimize} && \|Ax - b\|_2 \\ & \text{subject to} && \mathbf{card}(x) \leq k \end{aligned}$$

- heuristic:
  - minimize  $\|Ax - b\|_2 + \gamma \|x\|_1$
  - find smallest value of  $\gamma$  that gives  $\mathbf{card}(x) \leq k$
  - fix associated sparsity pattern (*i.e.*, subset of selected regressors) and find  $x$  that minimizes  $\|Ax - b\|_2$

## Example (6.4 in BV book)

- $A \in \mathbf{R}^{10 \times 20}$ ,  $x \in \mathbf{R}^{20}$ ,  $b \in \mathbf{R}^{10}$
- dashed curve: exact optimal (via enumeration)
- solid curve:  $\ell_1$  heuristic with polishing



# Sparse signal reconstruction

- convex-cardinality problem:

$$\begin{aligned} & \text{minimize} && \|Ax - y\|_2 \\ & \text{subject to} && \text{card}(x) \leq k \end{aligned}$$

- $\ell_1$  heuristic:

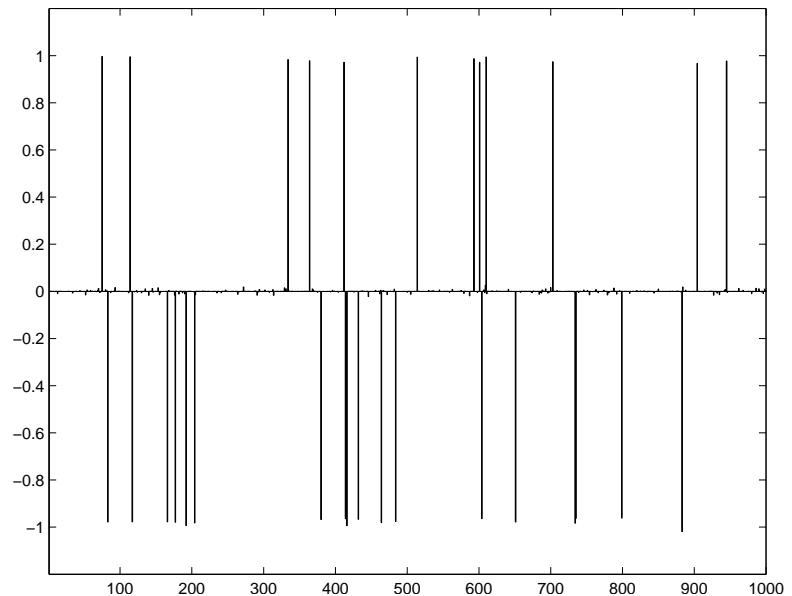
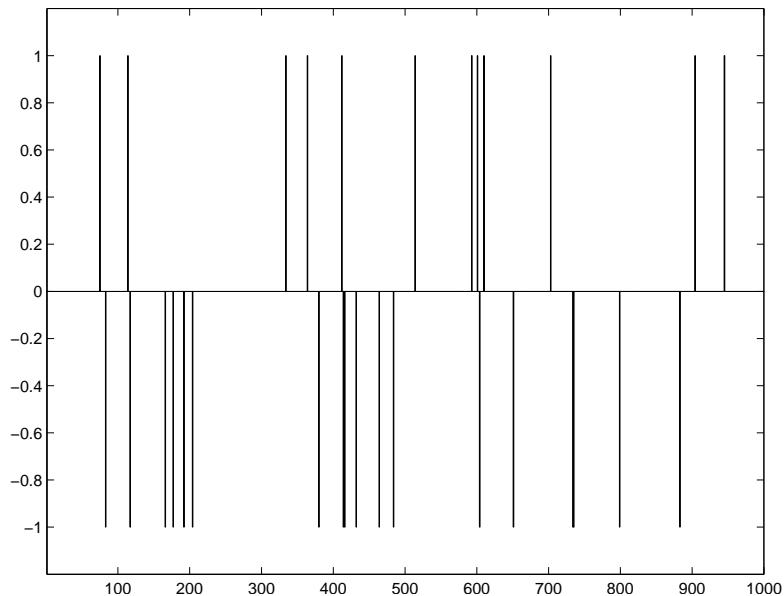
$$\begin{aligned} & \text{minimize} && \|Ax - y\|_2 \\ & \text{subject to} && \|x\|_1 \leq \beta \end{aligned}$$

(called LASSO)

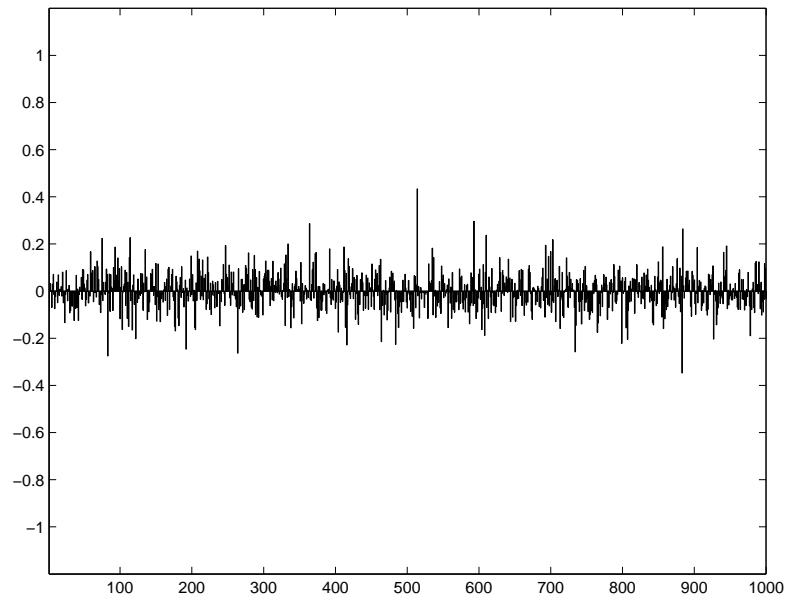
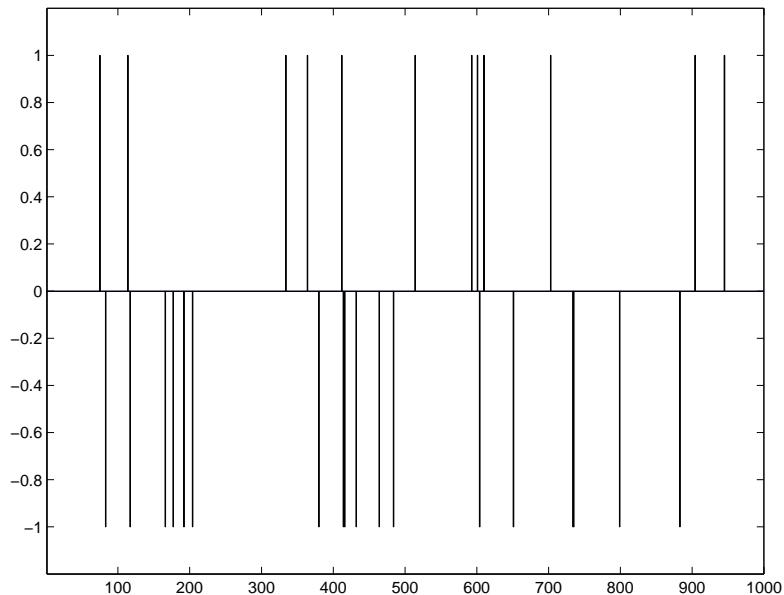
- another form: minimize  $\|Ax - y\|_2 + \gamma\|x\|_1$   
(called basis pursuit denoising)

## Example

- signal  $x \in \mathbb{R}^n$  with  $n = 1000$ ,  $\text{card}(x) = 30$
- $m = 200$  (random) noisy measurements:  $y = Ax + v$ ,  $v \sim \mathcal{N}(0, \sigma^2 \mathbf{1})$ ,  $A_{ij} \sim \mathcal{N}(0, 1)$
- *left*: original; *right*:  $\ell_1$  reconstruction with  $\gamma = 10^{-3}$



- $\ell_2$  reconstruction; minimizes  $\|Ax - y\|_2 + \gamma\|x\|_2$ , where  $\gamma = 10^{-3}$
- *left*: original; *right*:  $\ell_2$  reconstruction



## Some recent theoretical results

- suppose  $y = Ax$ ,  $A \in \mathbf{R}^{m \times n}$ ,  $\text{card}(x) \leq k$
- to reconstruct  $x$ , clearly need  $m \geq k$
- if  $m \geq n$  and  $A$  is full rank, we can reconstruct  $x$  without cardinality assumption
- when does the  $\ell_1$  heuristic (minimizing  $\|x\|_1$  subject to  $Ax = y$ ) reconstruct  $x$  (exactly)?

recent results by Candès, Donoho, Romberg, Tao, . . .

- (for some choices of  $A$ ) if  $m \geq (C \log n)k$ ,  $\ell_1$  heuristic reconstructs  $x$  exactly, with overwhelming probability
- $C$  is absolute constant; valid  $A$ 's include
  - $A_{ij} \sim \mathcal{N}(0, \sigma^2)$
  - $Ax$  gives Fourier transform of  $x$  at  $m$  frequencies, chosen from uniform distribution

# $\ell_1$ -norm Methods for Convex-Cardinality Problems

## Part II

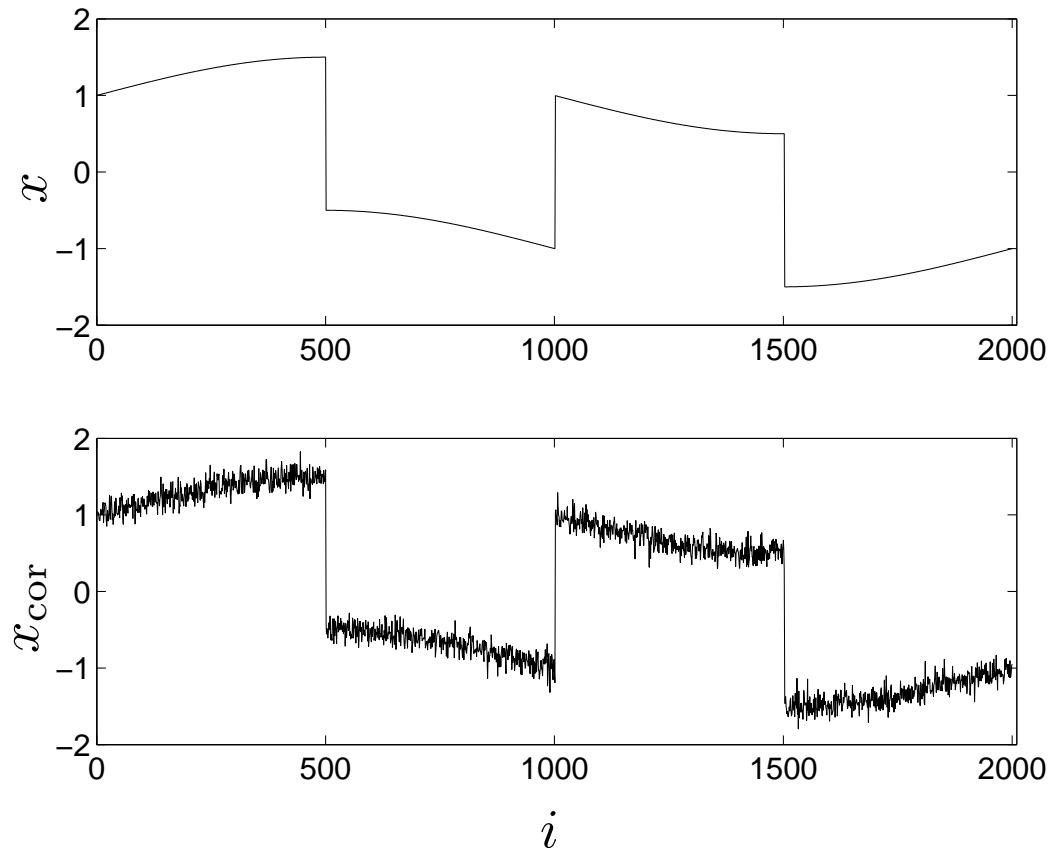
- total variation
- iterated weighted  $\ell_1$  heuristic
- matrix rank constraints

## Total variation reconstruction

- fit  $x_{\text{cor}}$  with piecewise constant  $\hat{x}$ , no more than  $k$  jumps
- convex-cardinality problem: minimize  $\|\hat{x} - x_{\text{cor}}\|_2$  subject to  $\mathbf{card}(Dx) \leq k$  ( $D$  is first order difference matrix)
- heuristic: minimize  $\|\hat{x} - x_{\text{cor}}\|_2 + \gamma \|Dx\|_1$ ; vary  $\gamma$  to adjust number of jumps
- $\|Dx\|_1$  is *total variation* of signal  $\hat{x}$
- method is called *total variation reconstruction*
- unlike  $\ell_2$  based reconstruction, TVR filters high frequency noise out while preserving sharp jumps

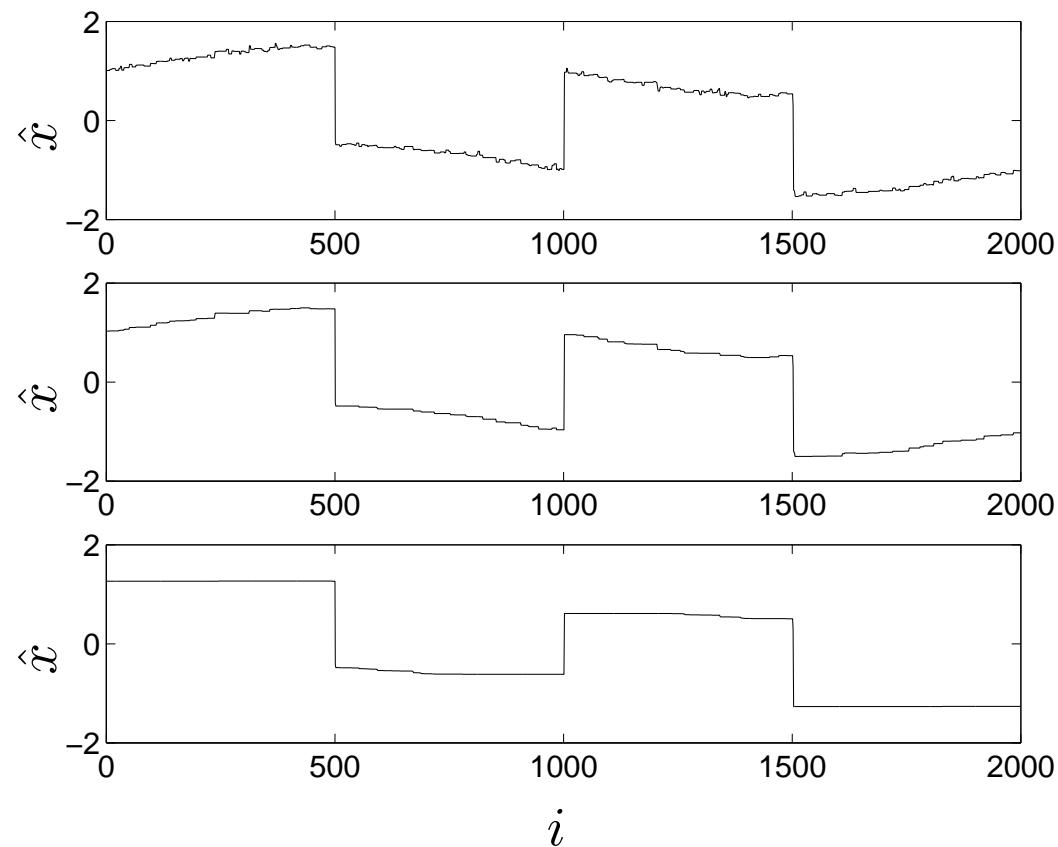
## Example (§6.3.3 in BV book)

signal  $x \in \mathbb{R}^{2000}$  and corrupted signal  $x_{\text{cor}} \in \mathbb{R}^{2000}$



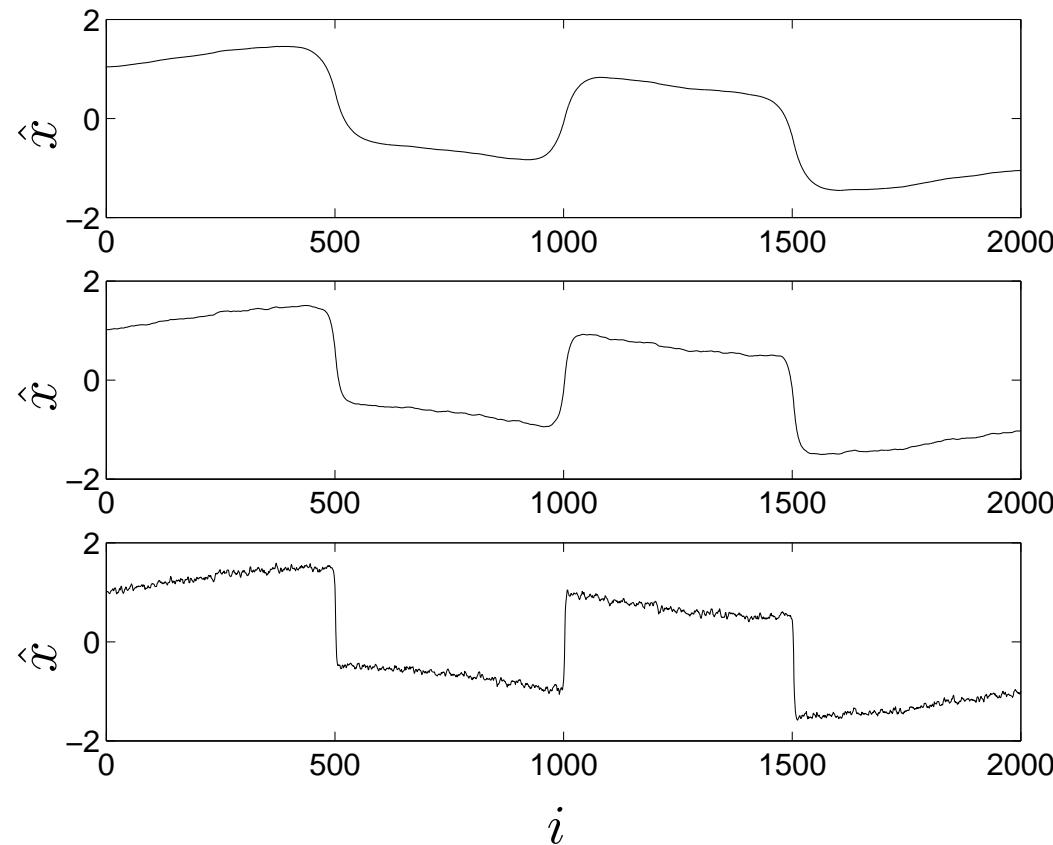
# Total variation reconstruction

for three values of  $\gamma$



## $\ell_2$ reconstruction

for three values of  $\gamma$



## Example: 2D total variation reconstruction

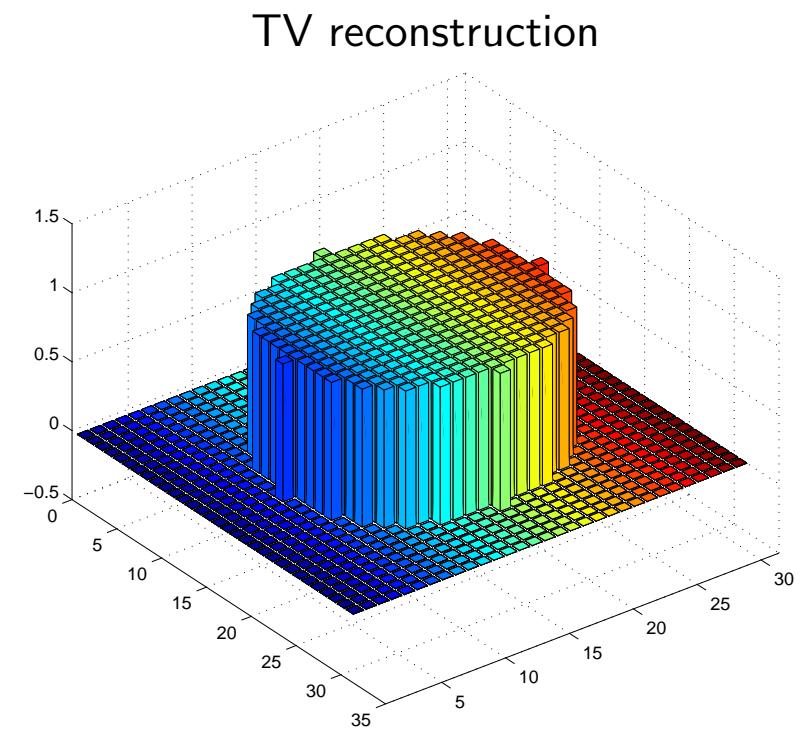
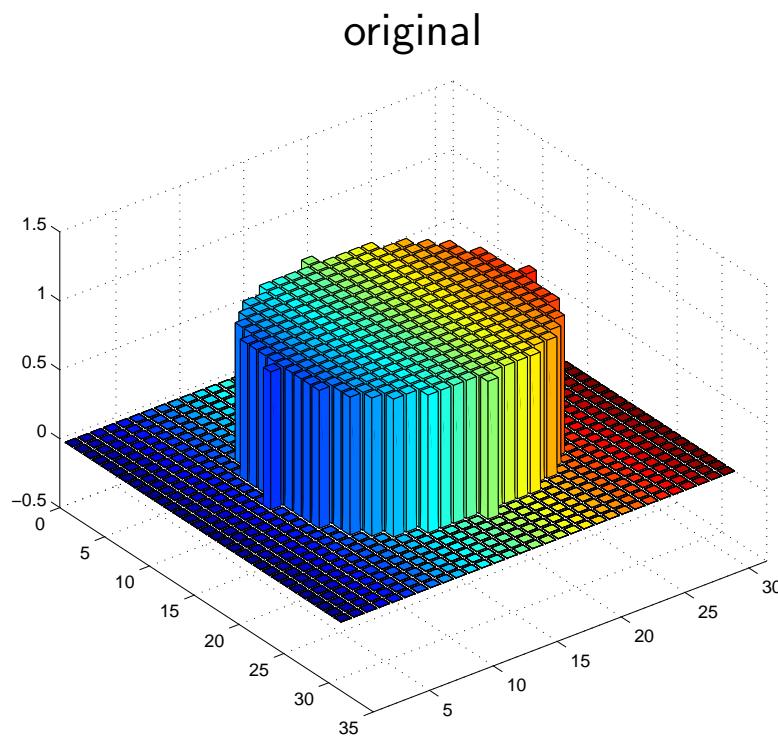
- $x \in \mathbf{R}^n$  are values of pixels on  $N \times N$  grid ( $N = 31$ , so  $n = 961$ )
- assumption:  $x$  has relatively few big changes in value (*i.e.*, boundaries)
- we have  $m = 120$  linear measurements,  $y = Fx$  ( $F_{ij} \sim \mathcal{N}(0, 1)$ )
- as convex-cardinality problem:

$$\begin{aligned} & \text{minimize} && \mathbf{card}(x_{i,j} - x_{i+1,j}) + \mathbf{card}(x_{i,j} - x_{i,j+1}) \\ & \text{subject to} && y = Fx \end{aligned}$$

- $\ell_1$  heuristic (objective is a 2D version of total variation)

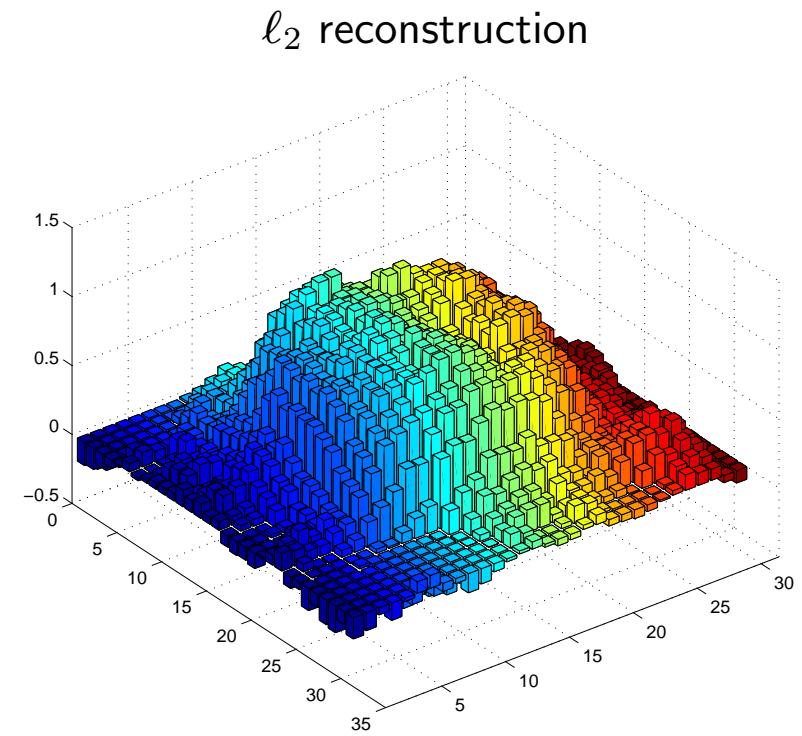
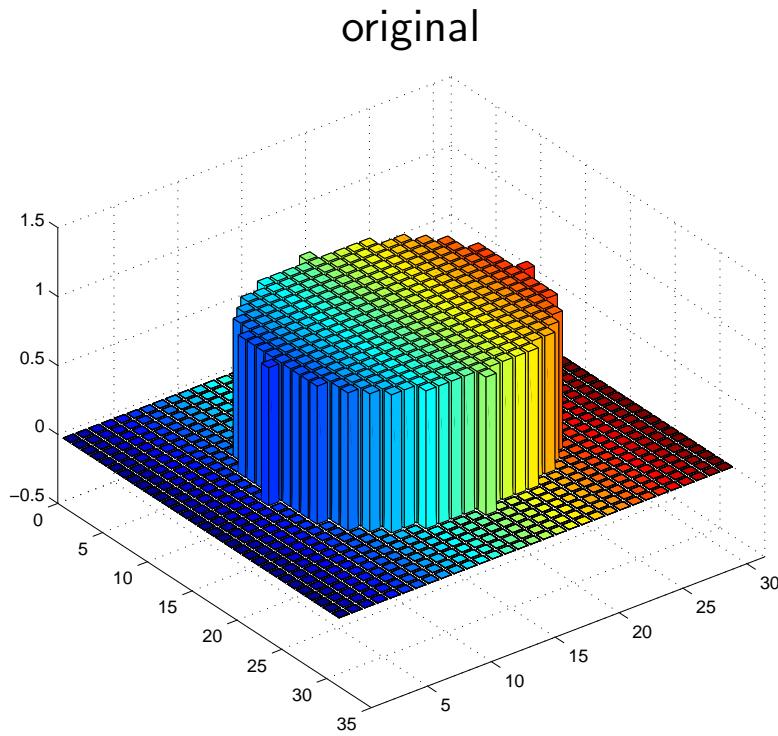
$$\begin{aligned} & \text{minimize} && \sum |x_{i,j} - x_{i+1,j}| + \sum |x_{i,j} - x_{i,j+1}| \\ & \text{subject to} && y = Fx \end{aligned}$$

# TV reconstruction



. . . not bad for 8 $\times$  more variables than measurements!

## $\ell_2$ reconstruction



. . . this is what you'd expect with  $8 \times$  more variables than measurements

## Iterated weighted $\ell_1$ heuristic

- to minimize  $\text{card}(x)$  over  $x \in \mathcal{C}$

$w := \mathbf{1}$

repeat

minimize  $\| \text{diag}(w)x \|_1$  over  $x \in \mathcal{C}$

$w_i := 1/(\epsilon + |x_i|)$

- first iteration is basic  $\ell_1$  heuristic
- increases relative weight on small  $x_i$
- typically converges in 5 or fewer steps
- often gives a modest improvement (*i.e.*, reduction in  $\text{card}(x)$ ) over basic  $\ell_1$  heuristic

## Interpretation

- wlog we can take  $x \succeq 0$  (by writing  $x = x_+ - x_-$ ,  $x_+, x_- \succeq 0$ , and replacing  $\text{card}(x)$  with  $\text{card}(x_+) + \text{card}(x_-)$ )
- we'll use approximation  $\text{card}(z) \approx \log(1 + z/\epsilon)$ , where  $\epsilon > 0$ ,  $z \in \mathbf{R}_+$
- using this approximation, we get (nonconvex) problem

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^n \log(1 + x_i/\epsilon) \\ &\text{subject to} && x \in \mathcal{C}, \quad x \succeq 0 \end{aligned}$$

- we'll find a local solution by linearizing objective at current point,

$$\sum_{i=1}^n \log(1 + x_i/\epsilon) \approx \sum_{i=1}^n \log(1 + x_i^{(k)}/\epsilon) + \sum_{i=1}^n \frac{x_i - x_i^{(k)}}{\epsilon + x_i^{(k)}}$$

and solving resulting convex problem

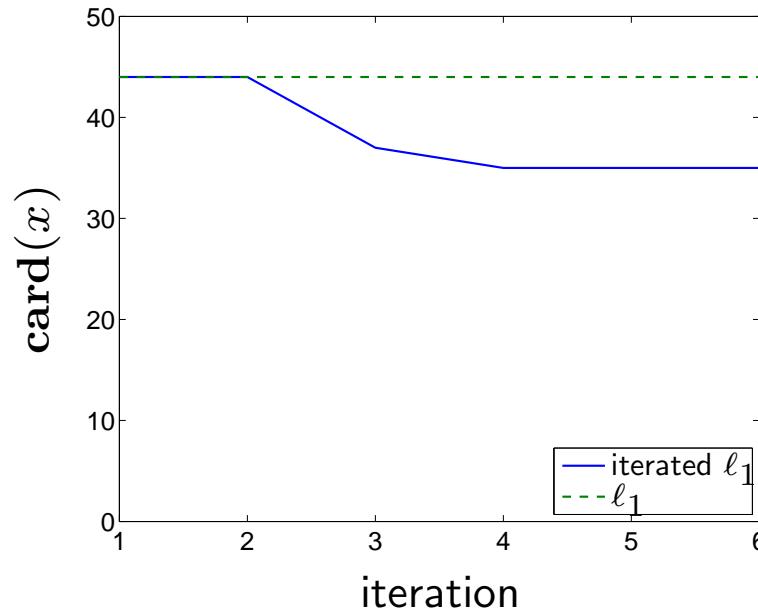
$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n w_i x_i \\ & \text{subject to} && x \in \mathcal{C}, \quad x \succeq 0 \end{aligned}$$

with  $w_i = 1/(\epsilon + x_i)$ , to get next iterate

- repeat until convergence to get a local solution

## Sparse solution of linear inequalities

- minimize  $\text{card}(x)$  over polyhedron  $\{x \mid Ax \preceq b\}$ ,  $A \in \mathbf{R}^{100 \times 50}$
- $\ell_1$  heuristic finds  $x \in \mathbf{R}^{50}$  with  $\text{card}(x) = 44$
- iterated weighted  $\ell_1$  heuristic finds  $x$  with  $\text{card}(x) = 36$   
(global solution, via branch & bound, is  $\text{card}(x) = 32$ )



## Detecting changes in time series model

- AR(2) scalar time-series model

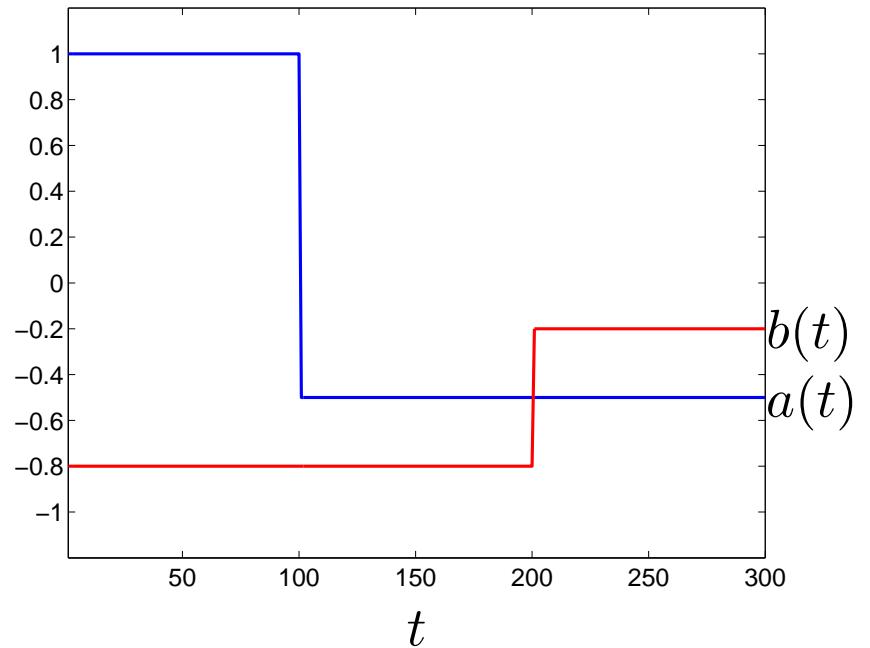
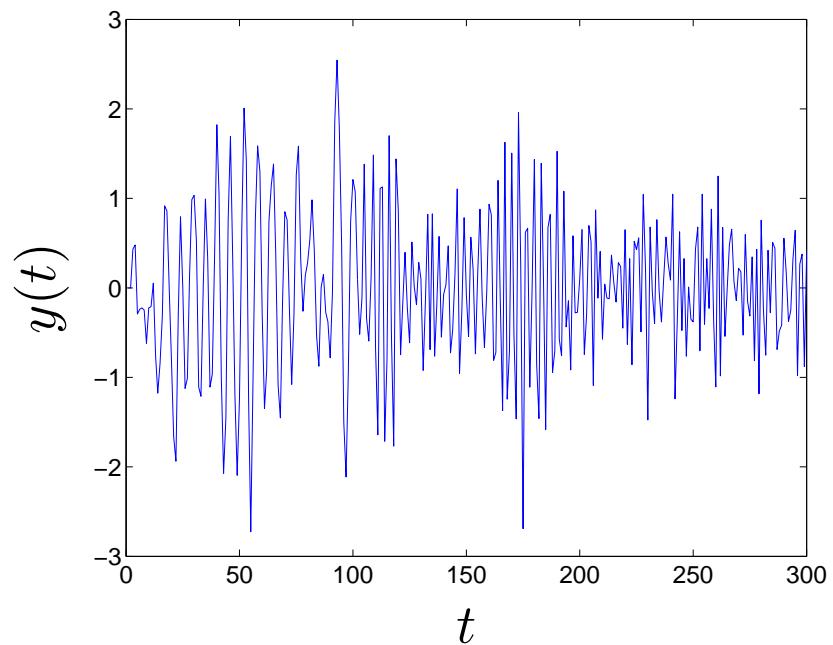
$$y(t+2) = a(t)y(t+1) + b(t)y(t) + v(t), \quad v(t) \text{ IID } \mathcal{N}(0, 0.5^2)$$

- assumption:  $a(t)$  and  $b(t)$  are piecewise constant, change infrequently
- given  $y(t)$ ,  $t = 1, \dots, T$ , estimate  $a(t)$ ,  $b(t)$ ,  $t = 1, \dots, T - 2$
- heuristic: minimize over variables  $a(t)$ ,  $b(t)$ ,  $t = 1, \dots, T - 1$

$$\begin{aligned} & \sum_{t=1}^{T-2} (y(t+2) - a(t)y(t+1) - b(t)y(t))^2 \\ & + \gamma \sum_{t=1}^{T-2} (|a(t+1) - a(t)| + |b(t+1) - b(t)|) \end{aligned}$$

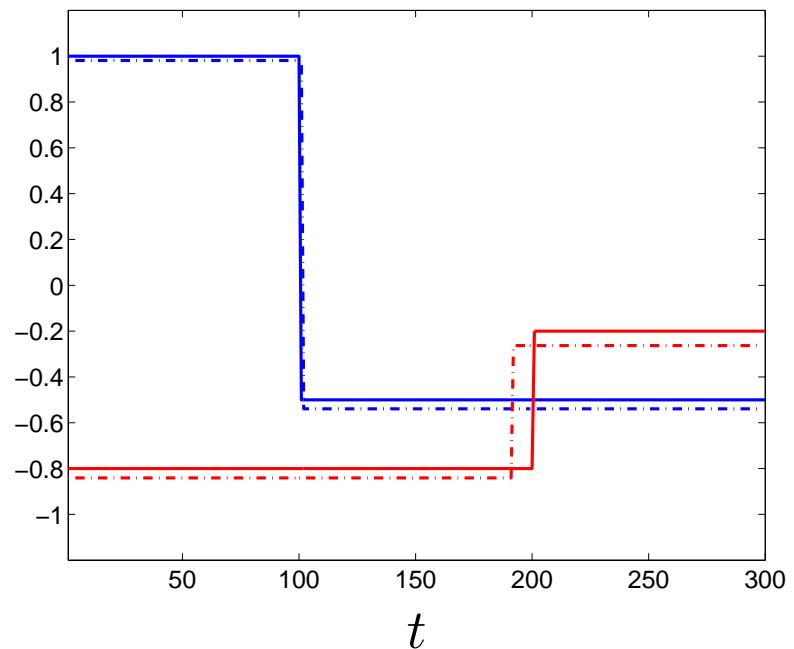
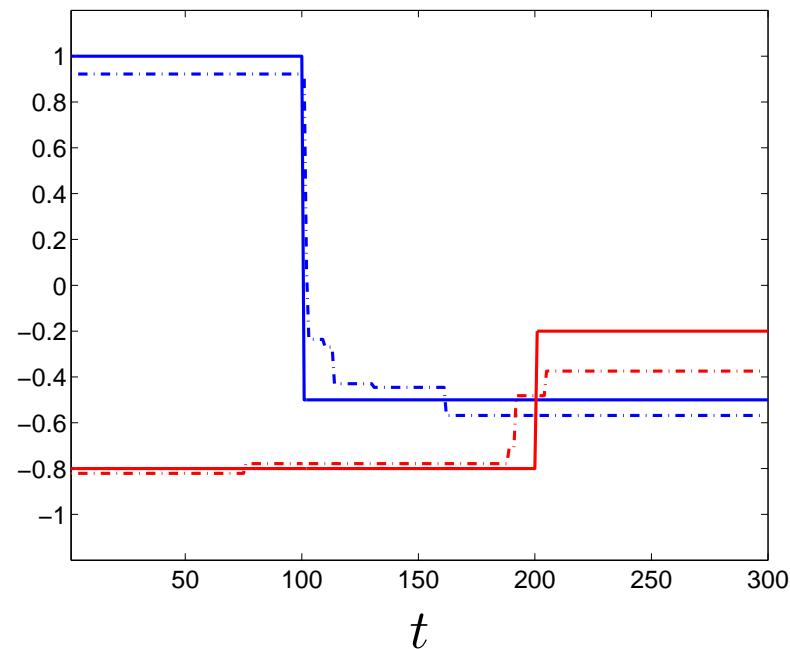
- vary  $\gamma$  to trade off fit versus number of changes in  $a$ ,  $b$

## Time series and true coefficients



## TV heuristic and iterated TV heuristic

*left:* TV with  $\gamma = 10$ ;    *right:* iterated TV, 5 iterations,  $\epsilon = 0.005$



## Extension to matrices

- **Rank** is natural analog of **card** for matrices
- convex-rank problem: convex, except for **Rank** in objective or constraints
- rank problem reduces to card problem when matrices are diagonal:  
 $\text{Rank}(\text{diag}(x)) = \text{card}(x)$
- analog of  $\ell_1$  heuristic: use *nuclear norm*,  $\|X\|_* = \sum_i \sigma_i(X)$   
(sum of singular values; dual of spectral norm)
- for  $X \succeq 0$ , reduces to **Tr**  $X$  (for  $x \succeq 0$ ,  $\|x\|_1$  reduces to  $\mathbf{1}^T x$ )

## Factor modeling

- given matrix  $\Sigma \in \mathbf{S}_+^n$ , find approximation of form  $\hat{\Sigma} = FF^T + D$ , where  $F \in \mathbf{R}^{n \times r}$ ,  $D$  is diagonal nonnegative
- gives underlying factor model (with  $r$  factors)

$$x = Fz + v, \quad v \sim \mathcal{N}(0, D), \quad z \sim \mathcal{N}(0, I)$$

- model with fewest factors:

$$\begin{aligned} & \text{minimize} && \mathbf{Rank } X \\ & \text{subject to} && X \succeq 0, \quad D \succeq 0 \text{ diagonal} \\ & && X + D \in \mathcal{C} \end{aligned}$$

with variables  $D, X \in \mathbf{S}^n$

$\mathcal{C}$  is convex set of acceptable approximations to  $\Sigma$

- e.g., via KL divergence

$$\mathcal{C} = \{\hat{\Sigma} \mid -\log \det(\Sigma^{-1/2} \hat{\Sigma} \Sigma^{-1/2}) + \mathbf{Tr}(\Sigma^{-1/2} \hat{\Sigma} \Sigma^{-1/2}) - n \leq \epsilon\}$$

- trace heuristic:

$$\begin{aligned} & \text{minimize} && \mathbf{Tr} X \\ & \text{subject to} && X \succeq 0, \quad D \succeq 0 \text{ diagonal} \\ & && X + D \in \mathcal{C} \end{aligned}$$

with variables  $d \in \mathbf{R}^n$ ,  $X \in \mathbf{S}^n$

## Example

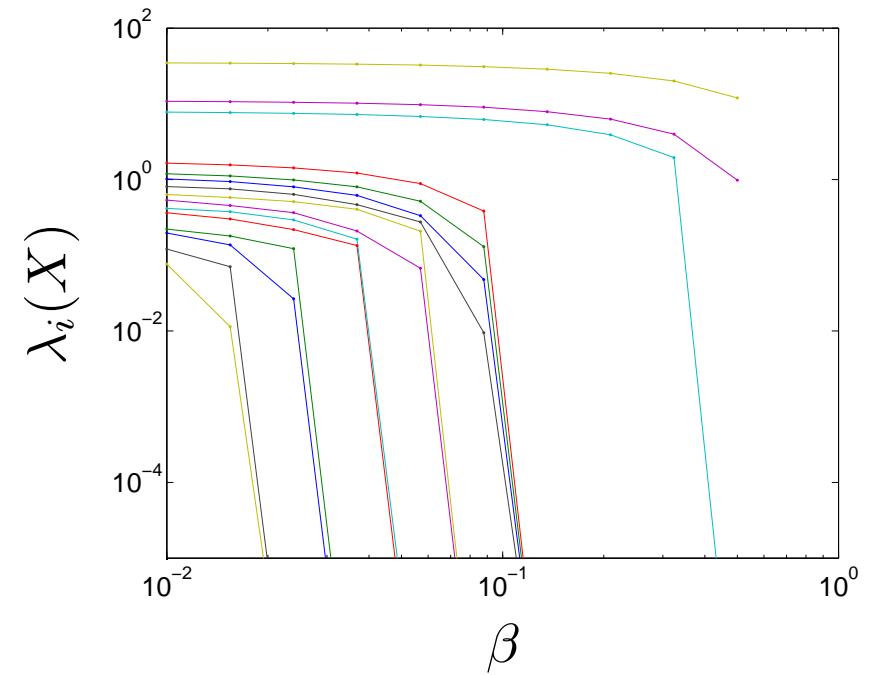
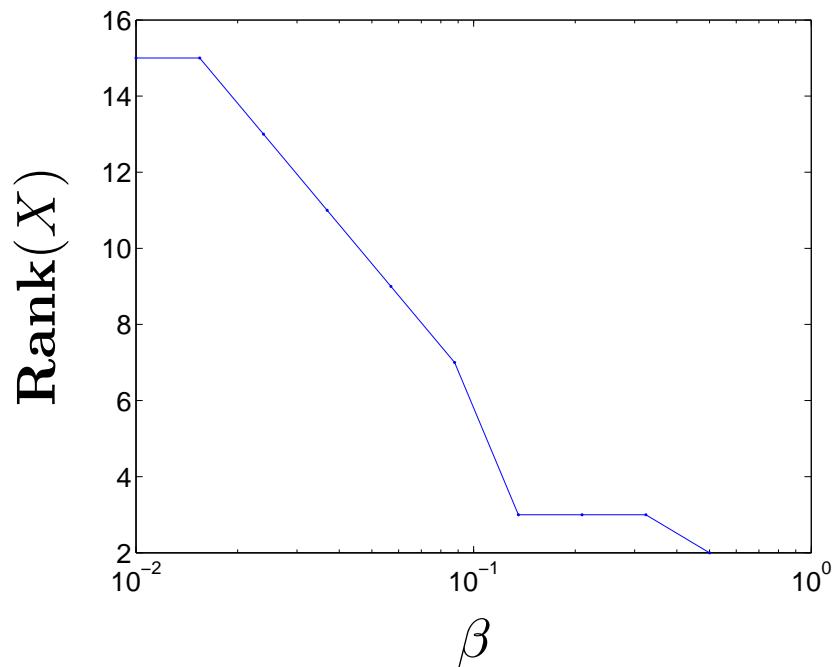
- $x = Fz + v$ ,  $z \sim \mathcal{N}(0, I)$ ,  $v \sim \mathcal{N}(0, D)$ ,  $D$  diagonal;  $F \in \mathbf{R}^{20 \times 3}$
- $\Sigma$  is empirical covariance matrix from  $N = 3000$  samples
- set of acceptable approximations

$$\mathcal{C} = \{\hat{\Sigma} \mid \|\Sigma^{-1/2}(\hat{\Sigma} - \Sigma)\Sigma^{-1/2}\| \leq \beta\}$$

- trace heuristic

$$\begin{aligned} & \text{minimize} && \mathbf{Tr} X \\ & \text{subject to} && X \succeq 0, \quad d \succeq 0 \\ & && \|\Sigma^{-1/2}(X + \mathbf{diag}(d) - \Sigma)\Sigma^{-1/2}\| \leq \beta \end{aligned}$$

# Trace approximation results



- for  $\beta = 0.1357$  (knee of the tradeoff curve) we find
  - $\angle(\text{range}(X), \text{range}(FF^T)) = 6.8^\circ$
  - $\|d - \text{diag}(D)\|/\|\text{diag}(D)\| = 0.07$
- *i.e.*, we have recovered the factor model from the empirical covariance

# Sequential Convex Programming

- sequential convex programming
- alternating convex optimization
- convex-concave procedure

# Methods for nonconvex optimization problems

- **convex optimization methods** are (roughly) always global, always fast
- for general nonconvex problems, we have to give up one
  - **local optimization methods** are fast, but need not find global solution (and even when they do, cannot certify it)
  - **global optimization methods** find global solution (and certify it), but are not always fast (indeed, are often slow)
- **this lecture:** local optimization methods that are based on solving a sequence of convex problems

# Sequential convex programming (SCP)

- a local optimization method for nonconvex problems that leverages convex optimization
  - convex portions of a problem are handled ‘exactly’ and efficiently
- SCP is a **heuristic**
  - it can fail to find optimal (or even feasible) point
  - results can (and often do) depend on starting point  
(can run algorithm from many initial points and take best result)
- SCP often works well, *i.e.*, finds a feasible point with good, if not optimal, objective value

# Problem

we consider nonconvex problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad j = 1, \dots, p \end{aligned}$$

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

- $f_0$  and  $f_i$  (possibly) nonconvex
- $h_i$  (possibly) non-affine

## Basic idea of SCP

- maintain estimate of solution  $x^{(k)}$ , and convex **trust region**  $\mathcal{T}^{(k)} \subset \mathbf{R}^n$
- form convex approximation  $\hat{f}_i$  of  $f_i$  over trust region  $\mathcal{T}^{(k)}$
- form affine approximation  $\hat{h}_i$  of  $h_i$  over trust region  $\mathcal{T}^{(k)}$
- $x^{(k+1)}$  is optimal point for approximate convex problem

$$\begin{aligned} & \text{minimize} && \hat{f}_0(x) \\ & \text{subject to} && \hat{f}_i(x) \leq 0, \quad i = 1, \dots, m \\ & && \hat{h}_i(x) = 0, \quad i = 1, \dots, p \\ & && x \in \mathcal{T}^{(k)} \end{aligned}$$

## Trust region

- typical trust region is box around current point:

$$\mathcal{T}^{(k)} = \{x \mid |x_i - x_i^{(k)}| \leq \rho_i, i = 1, \dots, n\}$$

- if  $x_i$  appears only in convex inequalities and affine equalities, can take  $\rho_i = \infty$

## Affine and convex approximations via Taylor expansions

- (affine) first order Taylor expansion:

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

- (convex part of) second order Taylor expansion:

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

$$P = (\nabla^2 f(x^{(k)}))_+, \text{ PSD part of Hessian}$$

- give local approximations, which don't depend on trust region radii  $\rho_i$

## Particle method

- particle method:
  - choose points  $z_1, \dots, z_K \in \mathcal{T}^{(k)}$   
(e.g., all vertices, some vertices, grid, random, . . . )
  - evaluate  $y_i = f(z_i)$
  - fit data  $(z_i, y_i)$  with convex (affine) function  
(using convex optimization)
- advantages:
  - handles nondifferentiable functions, or functions for which evaluating derivatives is difficult
  - gives **regional models**, which depend on current point and trust region radii  $\rho_i$

## Fitting affine or quadratic functions to data

fit convex quadratic function to data  $(z_i, y_i)$

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

with variables  $P \in \mathbf{S}^n$ ,  $q \in \mathbf{R}^n$ ,  $r \in \mathbf{R}$

- can use other objectives, add other convex constraints
- no need to solve exactly
- this problem is solved for each nonconvex constraint, each SCP step

## Quasi-linearization

- a cheap and simple method for affine approximation
- write  $h(x)$  as  $A(x)x + b(x)$  (many ways to do this)
- use  $\hat{h}(x) = A(x^{(k)})x + b(x^{(k)})$
- example:

$$h(x) = (1/2)x^T Px + q^T x + r = ((1/2)Px + q)^T x + r$$

- $\hat{h}_{\text{ql}}(x) = ((1/2)Px^{(k)} + q)^T x + r$
- $\hat{h}_{\text{tay}}(x) = (Px^{(k)} + q)^T (x - x^{(k)}) + h(x^{(k)})$

## Example

- nonconvex QP

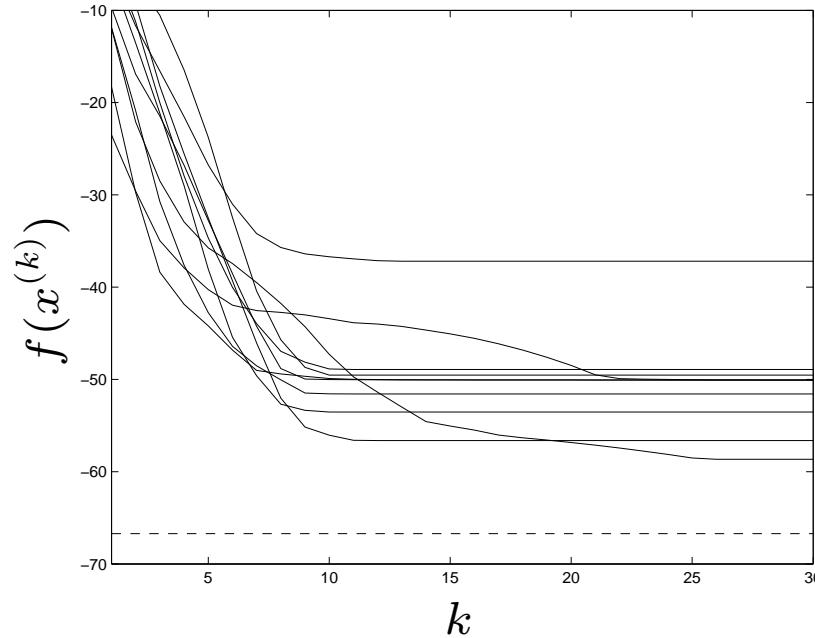
$$\begin{aligned} & \text{minimize} && f(x) = (1/2)x^T Px + q^T x \\ & \text{subject to} && \|x\|_\infty \leq 1 \end{aligned}$$

with  $P$  symmetric but not PSD

- use approximation

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

- example with  $x \in \mathbb{R}^{20}$
- SCP with  $\rho = 0.2$ , started from 10 different points



- runs typically converge to points between  $-60$  and  $-50$
- dashed line shows lower bound on optimal value  $\approx -66.5$

## Lower bound via Lagrange dual

- write constraints as  $x_i^2 \leq 1$  and form Lagrangian

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

- $g(\lambda) = -(1/2)q^T (P + 2 \mathbf{diag}(\lambda))^{-1} q - \mathbf{1}^T \lambda$ ; need  $P + 2 \mathbf{diag}(\lambda) \succ 0$
- solve dual problem to get best lower bound:

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

## Some (related) issues

- approximate convex problem can be infeasible
- how do we evaluate progress when  $x^{(k)}$  isn't feasible?  
need to take into account
  - objective  $f_0(x^{(k)})$
  - inequality constraint violations  $f_i(x^{(k)})_+$
  - equality constraint violations  $|h_i(x^{(k)})|$
- controlling the trust region size
  - $\rho$  too large: approximations are poor, leading to bad choice of  $x^{(k+1)}$
  - $\rho$  too small: approximations are good, but progress is slow

## Exact penalty formulation

- instead of original problem, we solve unconstrained problem

$$\text{minimize } \phi(x) = f_0(x) + \lambda (\sum_{i=1}^m f_i(x)_+ + \sum_{i=1}^p |h_i(x)|)$$

where  $\lambda > 0$

- for  $\lambda$  large enough, minimizer of  $\phi$  is solution of original problem
- for SCP, use convex approximation

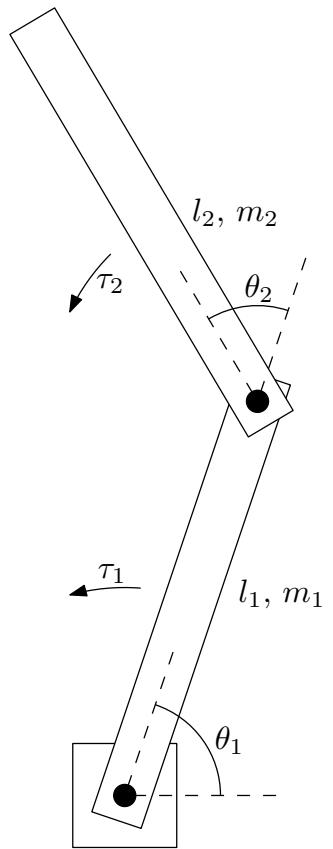
$$\hat{\phi}(x) = \hat{f}_0(x) + \lambda \left( \sum_{i=1}^m \hat{f}_i(x)_+ + \sum_{i=1}^p |\hat{h}_i(x)| \right)$$

- approximate problem always feasible

## Trust region update

- judge algorithm progress by decrease in  $\phi$ , using solution  $\tilde{x}$  of approximate problem
- decrease with approximate objective:  $\hat{\delta} = \phi(x^{(k)}) - \hat{\phi}(\tilde{x})$  (called *predicted decrease*)
- decrease with exact objective:  $\delta = \phi(x^{(k)}) - \phi(\tilde{x})$
- if  $\delta \geq \alpha\hat{\delta}$ ,  $\rho^{(k+1)} = \beta^{\text{succ}}\rho^{(k)}$ ,  $x^{(k+1)} = \tilde{x}$   
( $\alpha \in (0, 1)$ ,  $\beta^{\text{succ}} \geq 1$ ; typical values  $\alpha = 0.1$ ,  $\beta^{\text{succ}} = 1.1$ )
- if  $\delta < \alpha\hat{\delta}$ ,  $\rho^{(k+1)} = \beta^{\text{fail}}\rho^{(k)}$ ,  $x^{(k+1)} = x^{(k)}$   
( $\beta^{\text{fail}} \in (0, 1)$ ; typical value  $\beta^{\text{fail}} = 0.5$ )
- interpretation: if actual decrease is more (less) than fraction  $\alpha$  of predicted decrease then increase (decrease) trust region size

# Nonlinear optimal control



- 2-link system, controlled by torques  $\tau_1$  and  $\tau_2$  (no gravity)

- dynamics given by  $M(\theta)\ddot{\theta} + W(\theta, \dot{\theta})\dot{\theta} = \tau$ , with

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

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

$$s_i = \sin \theta_i, \quad c_i = \cos \theta_i$$

- nonlinear optimal control problem:

$$\begin{aligned} \text{minimize} \quad J &= \int_0^T \|\tau(t)\|_2^2 dt \\ \text{subject to} \quad \theta(0) &= \theta_{\text{init}}, \quad \dot{\theta}(0) = 0, \quad \theta(T) = \theta_{\text{final}}, \quad \dot{\theta}(T) = 0 \\ \|\tau(t)\|_\infty &\leq \tau_{\max}, \quad 0 \leq t \leq T \end{aligned}$$

# Discretization

- discretize with time interval  $h = T/N$
- $J \approx h \sum_{i=1}^N \|\tau_i\|_2^2$ , with  $\tau_i = \tau(ih)$
- approximate derivatives as

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

- approximate dynamics as set of nonlinear equality constraints:

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

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

- discretized nonlinear optimal control problem:

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

- replace equality constraints with quasilinearized versions

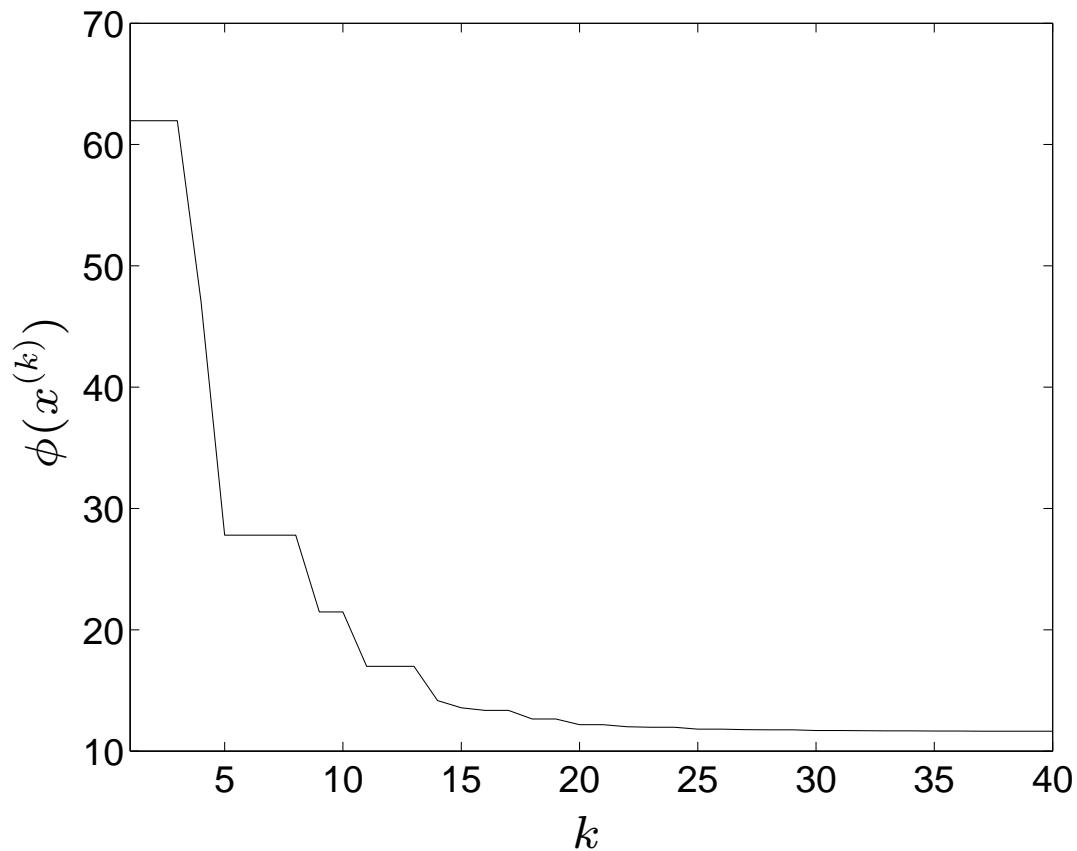
$$M(\theta_i^{(k)}) \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} + W\left(\theta_i^{(k)}, \frac{\theta_{i+1}^{(k)} - \theta_{i-1}^{(k)}}{2h}\right) \frac{\theta_{i+1} - \theta_{i-1}}{2h} = \tau_i$$

- trust region: only on  $\theta_i$
- initialize with  $\theta_i = ((i-1)/(N-1))(\theta_{\text{final}} - \theta_{\text{init}})$ ,  $i = 1, \dots, N$

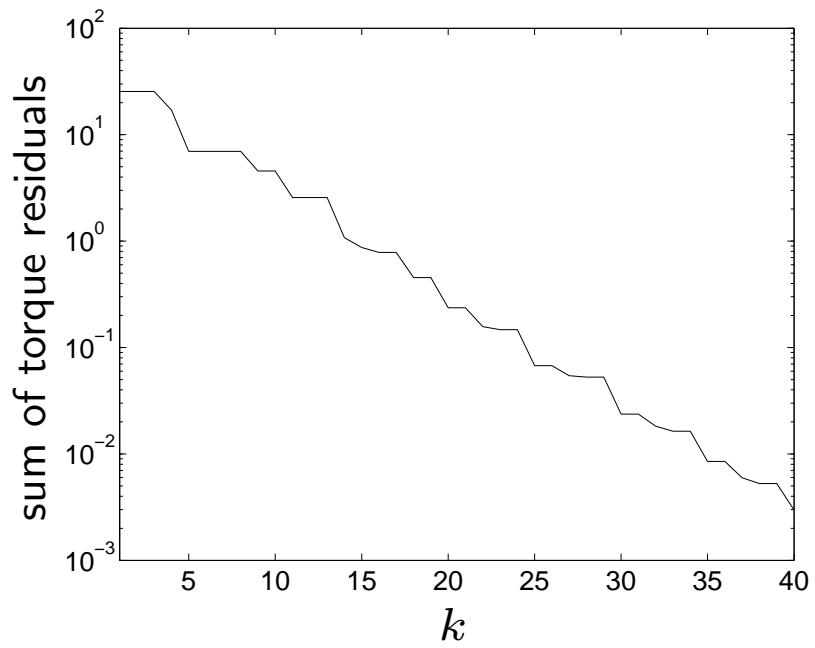
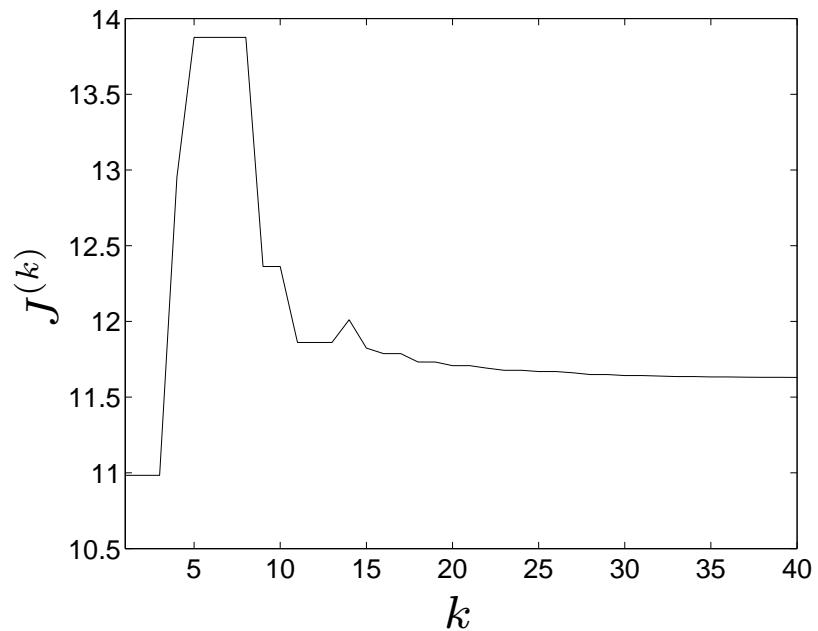
## Numerical example

- $m_1 = 1, m_2 = 5, l_1 = 1, l_2 = 1$
- $N = 40, T = 10$
- $\theta_{\text{init}} = (0, -2.9), \theta_{\text{final}} = (3, 2.9)$
- $\tau_{\max} = 1.1$
- $\alpha = 0.1, \beta^{\text{succ}} = 1.1, \beta^{\text{fail}} = 0.5, \rho^{(1)} = 90^\circ$
- $\lambda = 2$

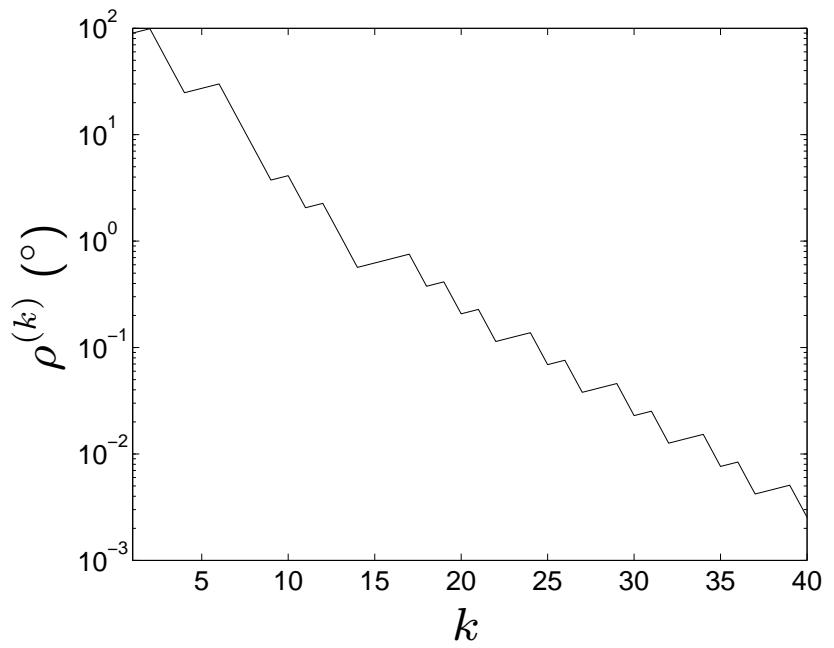
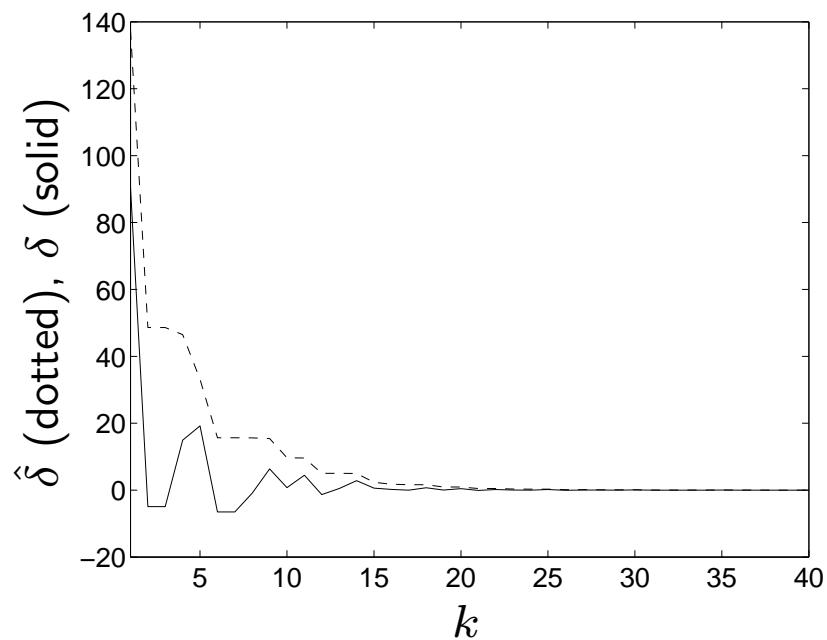
## SCP progress



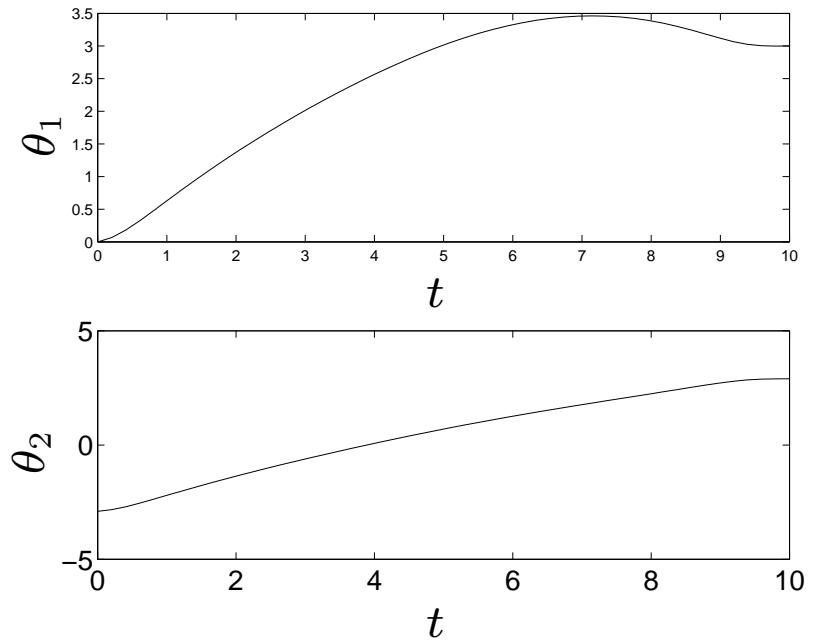
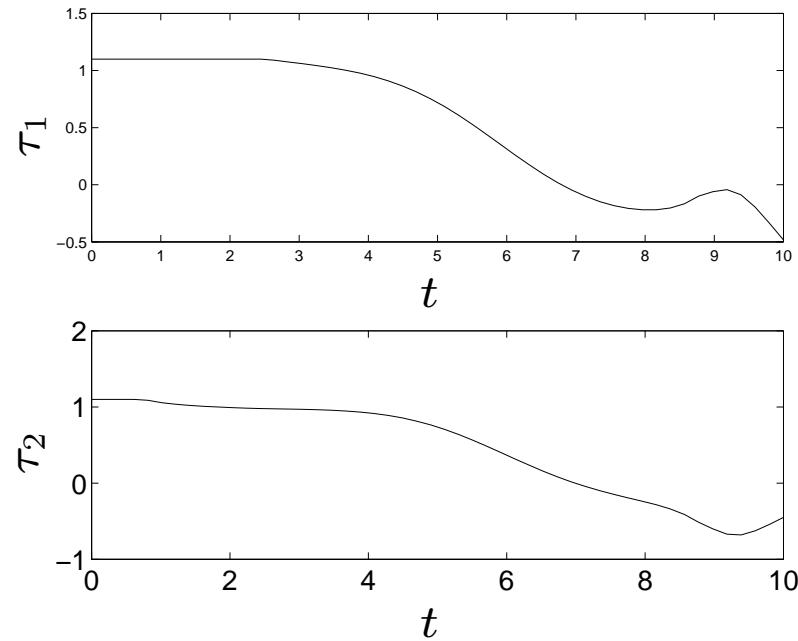
## Convergence of $J$ and torque residuals



## Predicted and actual decreases in $\phi$



# Trajectory plan



## ‘Difference of convex’ programming

- express problem as

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

where  $f_i$  and  $g_i$  are convex

- $f_i - g_i$  are called ‘difference of convex’ functions
- problem is sometimes called ‘difference of convex programming’

## Convex-concave procedure

- obvious convexification at  $x^{(k)}$ : replace  $f(x) - g(x)$  with

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

- since  $\hat{f}(x) \geq f(x)$  for all  $x$ , no trust region is needed
  - true objective at  $\tilde{x}$  is better than convexified objective
  - true feasible set contains feasible set for convexified problem
- SCP sometimes called ‘convex-concave procedure’

## Example (BV §7.1)

- given samples  $y_1, \dots, y_N \in \mathbf{R}^n$  from  $\mathcal{N}(0, \Sigma^{\text{true}})$
- negative log-likelihood function is

$$f(\Sigma) = \log \det \Sigma + \mathbf{Tr}(\Sigma^{-1} Y), \quad Y = (1/N) \sum_{i=1}^N y_i y_i^T$$

(dropping a constant and positive scale factor)

- ML estimate of  $\Sigma$ , with prior knowledge  $\Sigma_{ij} \geq 0$ :

$$\begin{aligned} & \text{minimize} && f(\Sigma) = \log \det \Sigma + \mathbf{Tr}(\Sigma^{-1} Y) \\ & \text{subject to} && \Sigma_{ij} \geq 0, \quad i, j = 1, \dots, n \end{aligned}$$

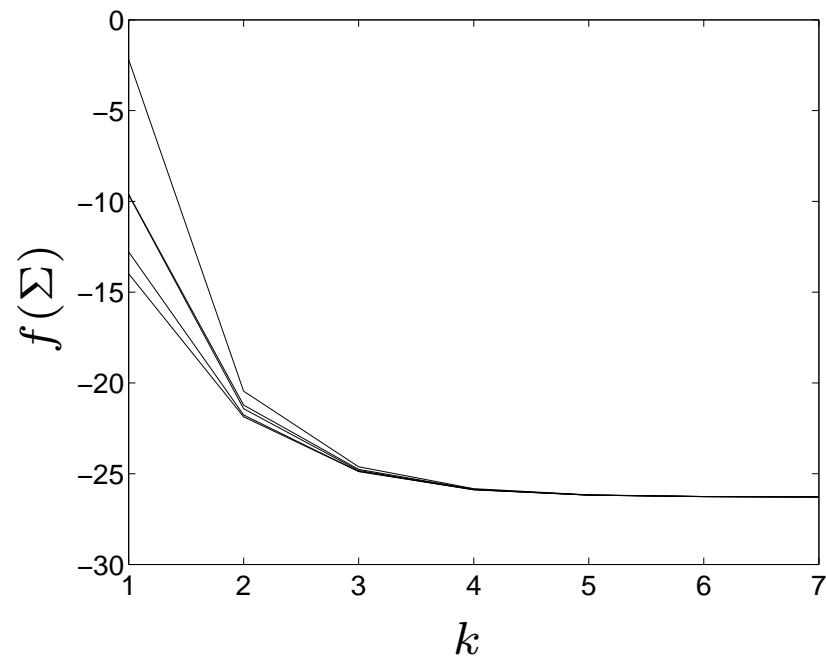
with variable  $\Sigma$  (constraint  $\Sigma \succ 0$  is implicit)

- first term in  $f$  is concave; second term is convex
- linearize first term in objective to get

$$\hat{f}(\Sigma) = \log \det \Sigma^{(k)} + \mathbf{Tr} \left( (\Sigma^{(k)})^{-1} (\Sigma - \Sigma^{(k)}) \right) + \mathbf{Tr}(\Sigma^{-1} Y)$$

## Numerical example

convergence of problem instance with  $n = 10, N = 15$



## Alternating convex optimization

- given nonconvex problem with variable  $(x_1, \dots, x_n) \in \mathbf{R}^n$
- $\mathcal{I}_1, \dots, \mathcal{I}_k \subset \{1, \dots, n\}$  are index subsets with  $\bigcup_j \mathcal{I}_j = \{1, \dots, n\}$
- suppose problem is convex in subset of variables  $x_i, i \in \mathcal{I}_j$ ,  
when  $x_i, i \notin \mathcal{I}_j$  are fixed
- alternating convex optimization method: cycle through  $j$ , in each step  
optimizing over variables  $x_i, i \in \mathcal{I}_j$
- special case: bi-convex problem
  - $x = (u, v)$ ; problem is convex in  $u$  ( $v$ ) with  $v$  ( $u$ ) fixed
  - alternate optimizing over  $u$  and  $v$

## Nonnegative matrix factorization

- NMF problem:

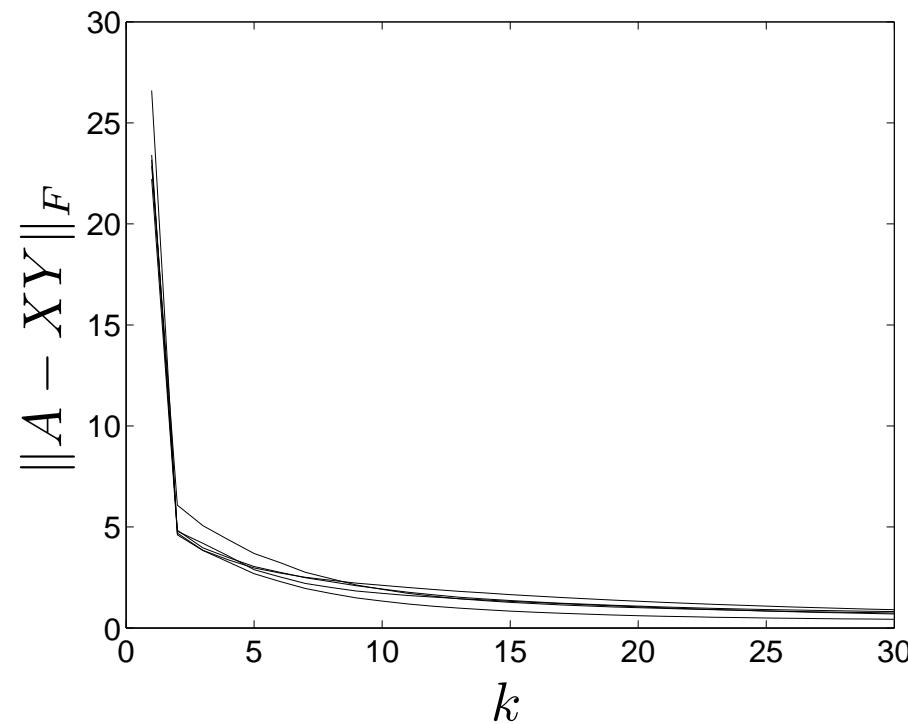
$$\begin{aligned} & \text{minimize} && \|A - XY\|_F \\ & \text{subject to} && X_{ij}, Y_{ij} \geq 0 \end{aligned}$$

variables  $X \in \mathbf{R}^{m \times k}$ ,  $Y \in \mathbf{R}^{k \times n}$ , data  $A \in \mathbf{R}^{m \times n}$

- difficult problem, except for a few special cases (*e.g.*,  $k = 1$ )
- alternating convex optimization: solve QPs to optimize over  $X$ , then  $Y$ , then  $X \dots$

# Example

- convergence for example with  $m = n = 50, k = 5$   
(five starting points)



# Branch and Bound Methods

- basic ideas and attributes
- unconstrained nonconvex optimization
- mixed convex-Boolean optimization

# Methods for nonconvex optimization problems

- **convex optimization methods** are (roughly) always global, always fast
- for general nonconvex problems, we have to give up one
- **local optimization methods** are fast, but need not find global solution (and even when they do, cannot certify it)
- **global optimization methods** find global solution (and certify it), but are not always fast (indeed, are often slow)

## Branch and bound algorithms

- methods for **global** optimization for nonconvex problems
- nonheuristic
  - maintain provable lower and upper bounds on global objective value
  - terminate with certificate proving  $\epsilon$ -suboptimality
- often slow; exponential worst case performance
- but (with luck) can (sometimes) work well

## Basic idea

- rely on two subroutines that (efficiently) compute a lower and an upper bound on the optimal value over a given region
  - upper bound can be found by choosing any point in the region, or by a local optimization method
  - lower bound can be found from convex relaxation, duality, Lipschitz or other bounds, . . .
- basic idea:
  - partition feasible set into convex sets, and find lower/upper bounds for each
  - form global lower and upper bounds; quit if close enough
  - else, refine partition and repeat

## Unconstrained nonconvex minimization

**goal:** find global minimum of function  $f : \mathbf{R}^m \rightarrow \mathbf{R}$ , over an  $m$ -dimensional rectangle  $\mathcal{Q}_{\text{init}}$ , to within some prescribed accuracy  $\epsilon$

- for any rectangle  $\mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}$ , we define  $\Phi_{\min}(\mathcal{Q}) = \inf_{x \in \mathcal{Q}} f(x)$
- global optimal value is  $f^\star = \Phi_{\min}(\mathcal{Q}_{\text{init}})$

## Lower and upper bound functions

- we'll use lower and upper bound functions  $\Phi_{lb}$  and  $\Phi_{ub}$ , that satisfy, for any rectangle  $\mathcal{Q} \subseteq \mathcal{Q}_{init}$ ,

$$\Phi_{lb}(\mathcal{Q}) \leq \Phi_{min}(\mathcal{Q}) \leq \Phi_{ub}(\mathcal{Q})$$

- bounds must become tight as rectangles shrink:

$$\forall \epsilon > 0 \ \exists \delta > 0 \ \forall \mathcal{Q} \subseteq \mathcal{Q}_{init}, \text{ size}(\mathcal{Q}) \leq \delta \implies \Phi_{ub}(\mathcal{Q}) - \Phi_{lb}(\mathcal{Q}) \leq \epsilon$$

where  $\text{size}(\mathcal{Q})$  is diameter (length of longest edge of  $\mathcal{Q}$ )

- to be practical,  $\Phi_{ub}(\mathcal{Q})$  and  $\Phi_{lb}(\mathcal{Q})$  should be cheap to compute

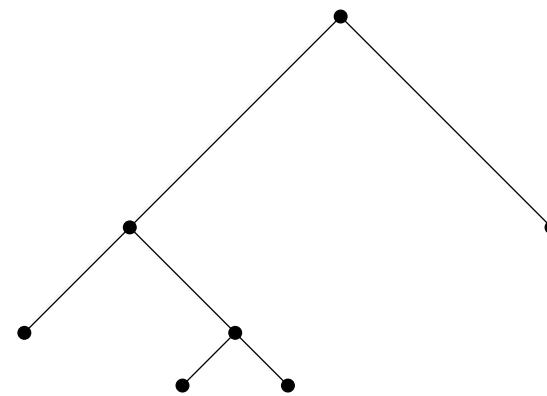
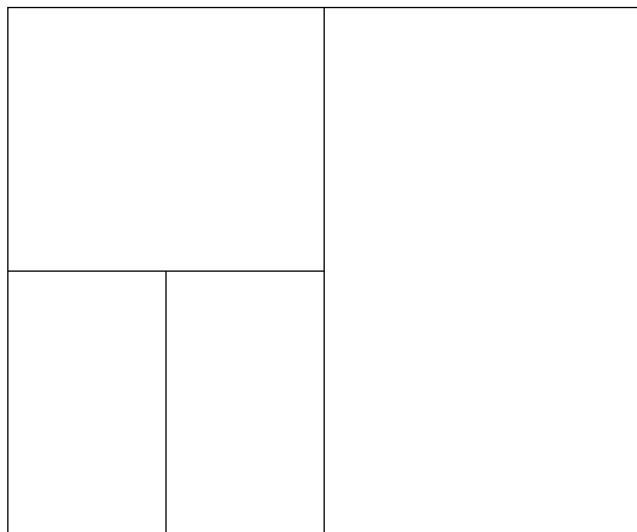
## Branch and bound algorithm

1. compute lower and upper bounds on  $f^*$ 
  - set  $L_1 = \Phi_{lb}(\mathcal{Q}_{\text{init}})$  and  $U_1 = \Phi_{ub}(\mathcal{Q}_{\text{init}})$
  - terminate if  $U_1 - L_1 \leq \epsilon$
2. partition (split)  $\mathcal{Q}_{\text{init}}$  into two rectangles  $\mathcal{Q}_{\text{init}} = \mathcal{Q}_1 \cup \mathcal{Q}_2$
3. compute  $\Phi_{lb}(\mathcal{Q}_i)$  and  $\Phi_{ub}(\mathcal{Q}_i)$ ,  $i = 1, 2$
4. update lower and upper bounds on  $f^*$ 
  - update lower bound:  $L_2 = \min\{\Phi_{lb}(\mathcal{Q}_1), \Phi_{lb}(\mathcal{Q}_2)\}$
  - update upper bound:  $U_2 = \max\{\Phi_{ub}(\mathcal{Q}_1), \Phi_{ub}(\mathcal{Q}_2)\}$
  - terminate if  $U_2 - L_2 \leq \epsilon$
5. refine partition, by splitting  $\mathcal{Q}_1$  or  $\mathcal{Q}_2$ , and repeat steps 3 and 4

- can assume w.l.o.g.  $U_i$  is nonincreasing,  $L_i$  is nondecreasing
- at each step we have a partially developed binary tree; children correspond to the subrectangles formed by splitting the parent rectangle
- leaves give the current partition of  $\mathcal{Q}_{\text{init}}$
- need rules for choosing, at each step
  - which rectangle to split
  - which edge (variable) to split
  - where to split (what value of variable)
- some good rules: split rectangle with smallest lower bound, along longest edge, in half

# Example

partitioned rectangle in  $\mathbb{R}^2$ , and associated binary tree, after 3 iterations



# Pruning

- can eliminate or **prune** any rectangle  $\mathcal{Q}$  in tree with  $\Phi_{lb}(\mathcal{Q}) > U_k$ 
  - every point in rectangle is worse than current upper bound
  - hence cannot be optimal
- does not affect algorithm, but does reduce storage requirements
- can track progress of algorithm via
  - total pruned (or unpruned) volume
  - number of pruned (or unpruned) leaves in partition

## Convergence analysis

- number of rectangles in partition  $\mathcal{L}_k$  is  $k$  (without pruning)
- total volume of these rectangles is  $\text{vol}(\mathcal{Q}_{\text{init}})$ , so

$$\min_{\mathcal{Q} \in \mathcal{L}_k} \text{vol}(\mathcal{Q}) \leq \frac{\text{vol}(\mathcal{Q}_{\text{init}})}{k}$$

- so for  $k$  large, at least one rectangle has small volume
- need to show that small volume implies small size
- this will imply that one rectangle has  $U - L$  small
- hence  $U_k - L_k$  is small

## Bounding condition number

- **condition number** of rectangle  $\mathcal{Q} = [l_1, u_1] \times \cdots \times [l_n, u_n]$  is

$$\text{cond}(\mathcal{Q}) = \frac{\max_i(u_i - l_i)}{\min_i(u_i - l_i)}$$

- if we split rectangle along longest edge, we have

$$\text{cond}(\mathcal{Q}) \leq \max\{\text{cond}(\mathcal{Q}_{\text{init}}), 2\}$$

for any rectangle in partition

- other rules (*e.g.*, cycling over variables) also guarantee bound on  $\text{cond}(\mathcal{Q})$

## **Small volume implies small size**

$$\begin{aligned}\text{vol}(\mathcal{Q}) &= \prod_i (u_i - l_i) \geq \max_i (u_i - l_i) \left( \min_i (u_i - l_i) \right)^{m-1} \\ &= \frac{(2 \text{ size}(\mathcal{Q}))^m}{\text{cond}(\mathcal{Q})^{m-1}} \geq \left( \frac{2 \text{ size}(\mathcal{Q})}{\text{cond}(\mathcal{Q})} \right)^m\end{aligned}$$

and so  $\text{size}(\mathcal{Q}) \leq (1/2)\text{vol}(\mathcal{Q})^{1/m}\text{cond}(\mathcal{Q})$

therefore if  $\text{cond}(\mathcal{Q})$  is bounded and  $\text{vol}(\mathcal{Q})$  is small,  $\text{size}(\mathcal{Q})$  is small

## Mixed Boolean-convex problem

$$\begin{aligned} & \text{minimize} && f_0(x, z) \\ & \text{subject to} && f_i(x, z) \leq 0, \quad i = 1, \dots, m \\ & && z_j \in \{0, 1\}, \quad j = 1, \dots, n \end{aligned}$$

- $x \in \mathbf{R}^p$  is called *continuous variable*
- $z \in \{0, 1\}^n$  is called *Boolean variable*
- $f_0, \dots, f_n$  are convex in  $x$  and  $z$
- optimal value denoted  $p^*$
- for each fixed  $z \in \{0, 1\}^n$ , reduced problem (with variable  $x$ ) is convex

## Solution methods

- *brute force*: solve convex problem for each of the  $2^n$  possible values of  $z \in \{0, 1\}^n$ 
  - possible for  $n \leq 15$  or so, but not  $n \geq 20$
- *branch and bound*
  - in worst case, we end up solving all  $2^n$  convex problems
  - hope that branch and bound will actually work much better

## Lower bound via convex relaxation

### convex relaxation

$$\begin{aligned} & \text{minimize} && f_0(x, z) \\ & \text{subject to} && f_i(x, z) \leq 0, \quad i = 1, \dots, m \\ & && 0 \leq z_j \leq 1, \quad j = 1, \dots, n \end{aligned}$$

- convex with (continuous) variables  $x$  and  $z$ , so easily solved
- optimal value (denoted  $L_1$ ) is lower bound on  $p^*$ , optimal value of original problem
- $L_1$  can be  $+\infty$  (which implies original problem infeasible)

## Upper bounds

- can find an upper bound (denoted  $U_1$ ) on  $p^*$  several ways
- simplest method: round each relaxed Boolean variable  $z_i^*$  to 0 or 1
- more sophisticated method: round each Boolean variable, then solve the resulting convex problem in  $x$
- randomized method:
  - generate random  $z_i \in \{0, 1\}$ , with  $\text{Prob}(z_i = 1) = z_i^*$
  - (optionally, solve for  $x$  again)
  - take best result out of some number of samples
- upper bound can be  $+\infty$  (method failed to produce a feasible point)
- if  $U_1 - L_1 \leq \epsilon$  we can quit

# Branching

- pick any index  $k$ , and form two subproblems
- first problem:

$$\begin{aligned} & \text{minimize} && f_0(x, z) \\ & \text{subject to} && f_i(x, z) \leq 0, \quad i = 1, \dots, m \\ & && z_j \in \{0, 1\}, \quad j = 1, \dots, n \\ & && z_k = 0 \end{aligned}$$

- second problem:

$$\begin{aligned} & \text{minimize} && f_0(x, z) \\ & \text{subject to} && f_i(x, z) \leq 0, \quad i = 1, \dots, m \\ & && z_j \in \{0, 1\}, \quad j = 1, \dots, n \\ & && z_k = 1 \end{aligned}$$

- each of these is a Boolean-convex problem, with  $n - 1$  Boolean variables
- optimal value of original problem is the smaller of the optimal values of the two subproblems
- can solve convex relaxations of subproblems to obtain lower and upper bounds on optimal values

## New bounds from subproblems

- let  $\tilde{L}, \tilde{U}$  be lower, upper bounds for  $z_k = 0$
- let  $\bar{L}, \bar{U}$  be lower, upper bounds for  $z_k = 1$
- $\min\{\tilde{L}, \bar{L}\} \geq L_1$
- can assume w.l.o.g. that  $\min\{\tilde{U}, \bar{U}\} \leq U_1$
- thus, we have new bounds on  $p^*$ :

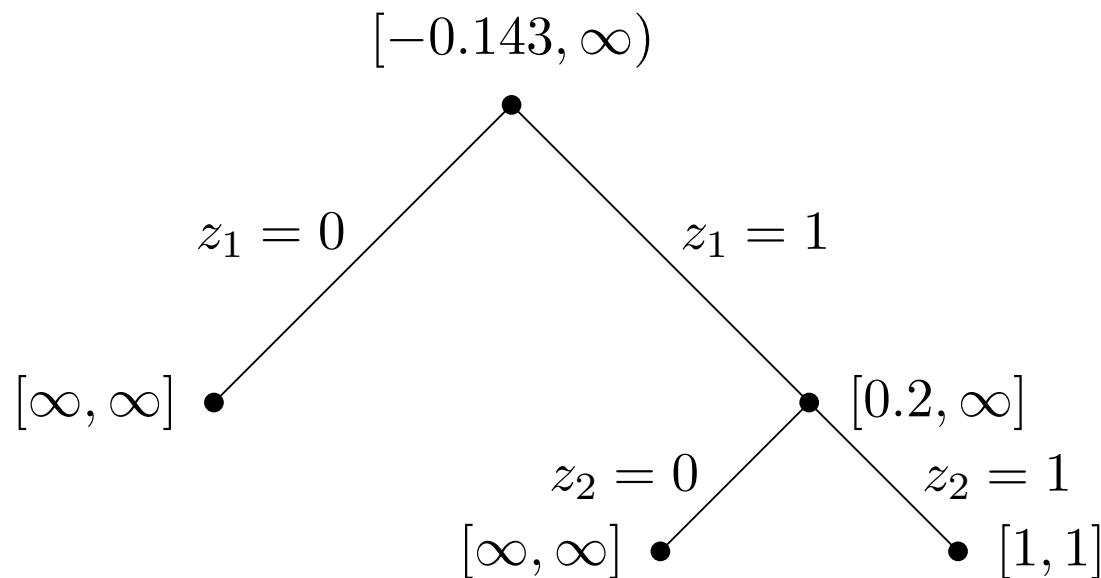
$$L_2 = \min\{\tilde{L}, \bar{L}\} \leq p^* \leq U_2 = \min\{\tilde{U}, \bar{U}\}$$

## Branch and bound algorithm

- continue to form binary tree by splitting, relaxing, calculating bounds on subproblems
- convergence proof is trivial: cannot go more than  $2^n$  steps before  $U = L$
- can prune nodes with  $L$  exceeding current  $U_k$
- common strategy is to pick a node with smallest  $L$
- can pick variable to split several ways
  - ‘least ambivalent’: choose  $k$  for which  $z^* = 0$  or  $1$ , with largest Lagrange multiplier
  - ‘most ambivalent’: choose  $k$  for which  $|z_k^* - 1/2|$  is minimum

## Small example

nodes show lower and upper bounds for three-variable Boolean LP



## Minimum cardinality example

find sparsest  $x$  satisfying linear inequalities

$$\begin{aligned} & \text{minimize} && \mathbf{card}(x) \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

equivalent to mixed Boolean-LP:

$$\begin{aligned} & \text{minimize} && \mathbf{1}^T z \\ & \text{subject to} && L_i z_i \leq x_i \leq U_i z_i, \quad i = 1, \dots, n \\ & && Ax \preceq b \\ & && z_i \in \{0, 1\}, \quad i = 1, \dots, n \end{aligned}$$

with variables  $x$  and  $z$  and lower and upper bounds on  $x$ ,  $L$  and  $U$

## Bounding $x$

- $L_i$  is optimal value of LP

$$\begin{aligned} & \text{minimize} && x_i \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

- $U_i$  is optimal value of LP

$$\begin{aligned} & \text{maximize} && x_i \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

- solve  $2n$  LPs to get all bounds
- if  $L_i > 0$  or  $U_i < 0$ , we can just set  $z_i = 1$

## Relaxation problem

- relaxed problem is

$$\begin{aligned} & \text{minimize} && \mathbf{1}^T z \\ & \text{subject to} && L_i z_i \leq x_i \leq U_i z_i, \quad i = 1, \dots, n \\ & && Ax \preceq b \\ & && 0 \leq z_i \leq 1, \quad i = 1, \dots, n \end{aligned}$$

- (assuming  $L_i < 0$ ,  $U_i > 0$ ) equivalent to

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n ((1/U_i)(x_i)_+ + (-1/L_i)(x_i)_-) \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

- objective is asymmetric weighted  $\ell_1$ -norm

## A few more details

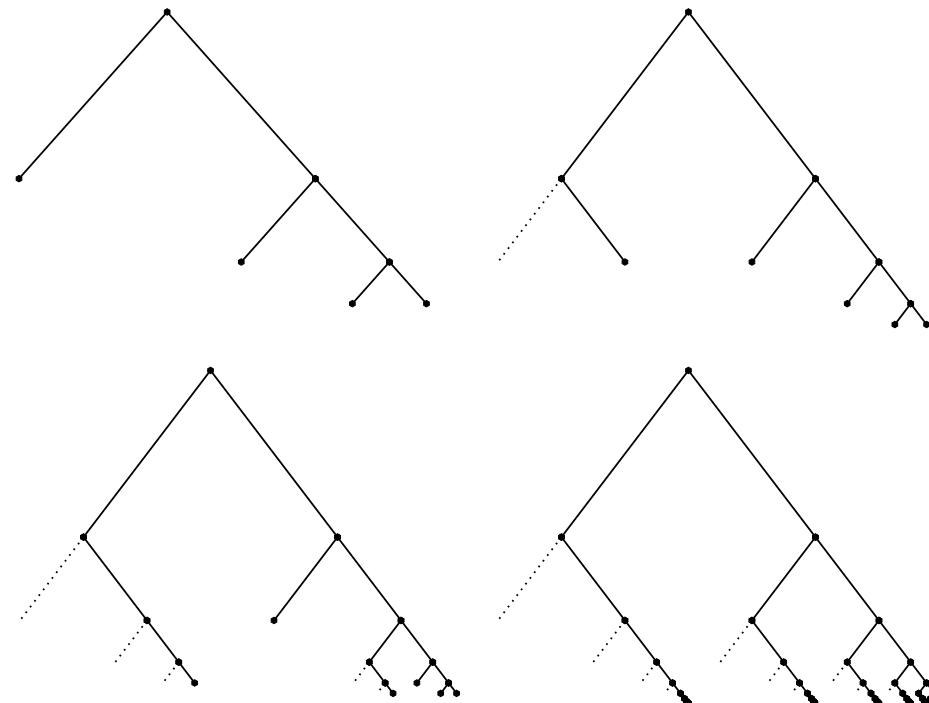
- relaxed problem is well known heuristic for finding a sparse solution, so we take  $\text{card}(x^*)$  as our upper bound
- for lower bound, we can replace  $L$  from LP with  $\lceil L \rceil$ , since  $\text{card}(x)$  is integer valued
- at each iteration, split node with lowest lower bound
- split most ambivalent variable

## Small example

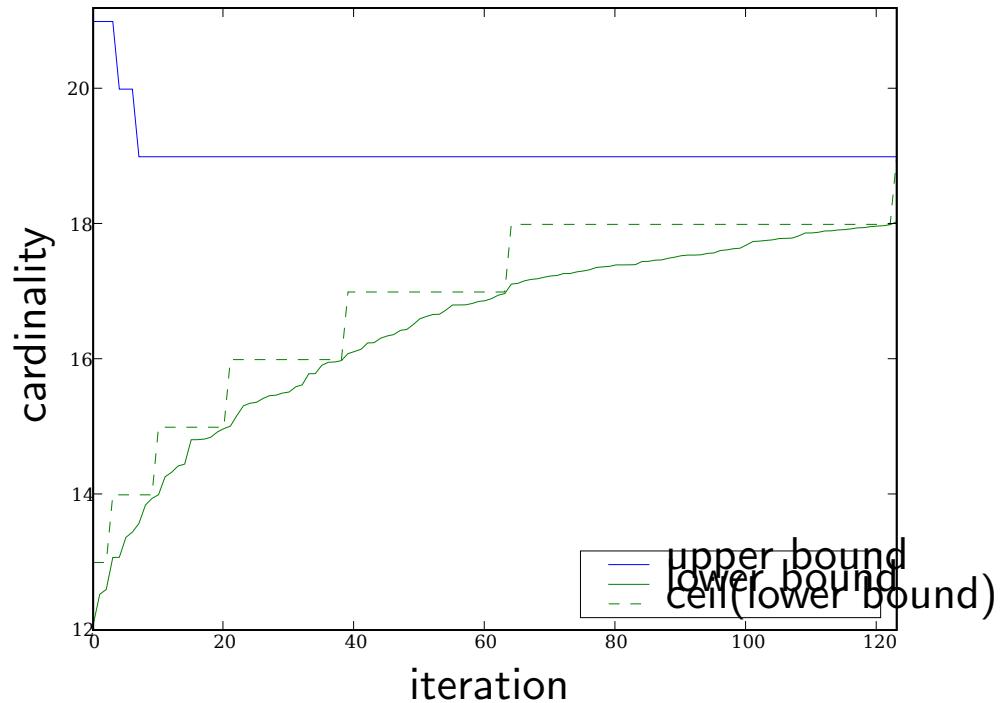
- random problem with 30 variables, 100 constraints
- $2^{30} \approx 10^9$
- takes 8 iterations to find a point with globally minimum cardinality (19)
- but, takes 124 iterations to **prove** minimum cardinality is 19
- requires 309 LP solves (including 60 to calculate lower and upper bounds on each variable)

## Algorithm progress

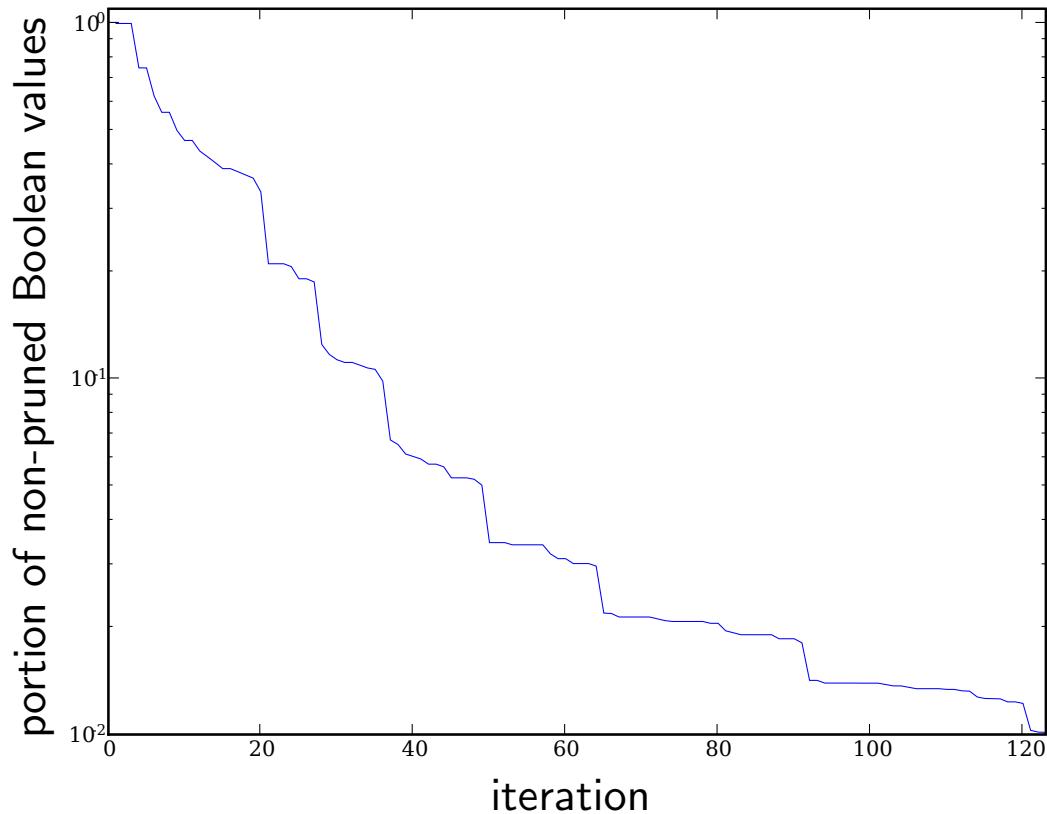
tree after 3 iterations (top left), 5 iterations (top right), 10 iterations (bottom left), and 124 iterations (bottom right)



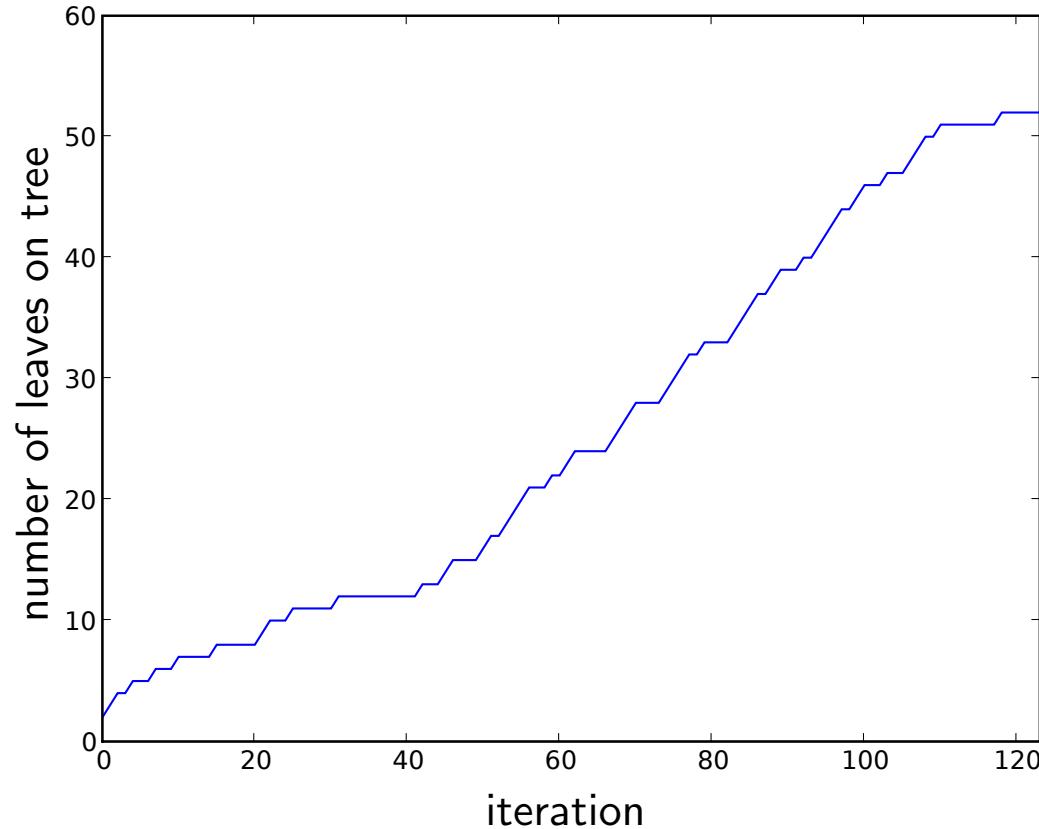
## Global lower and upper bounds



## Portion of non-pruned sparsity patterns



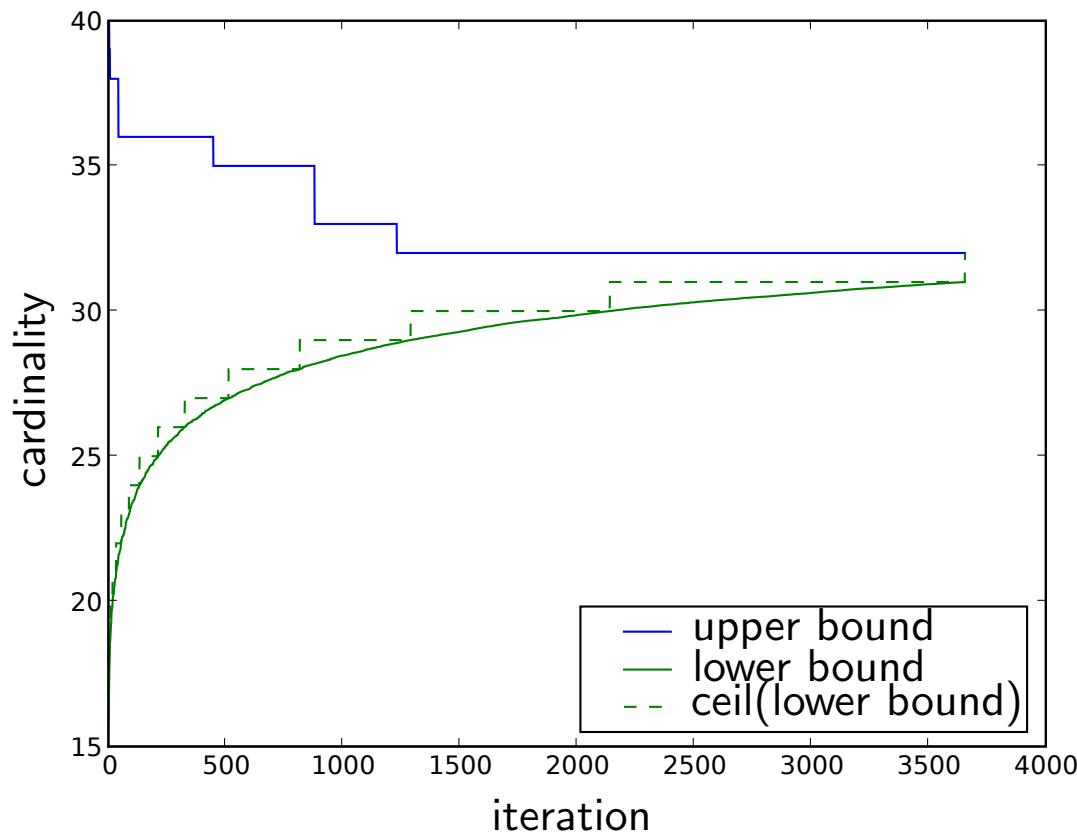
## Number of active leaves in tree



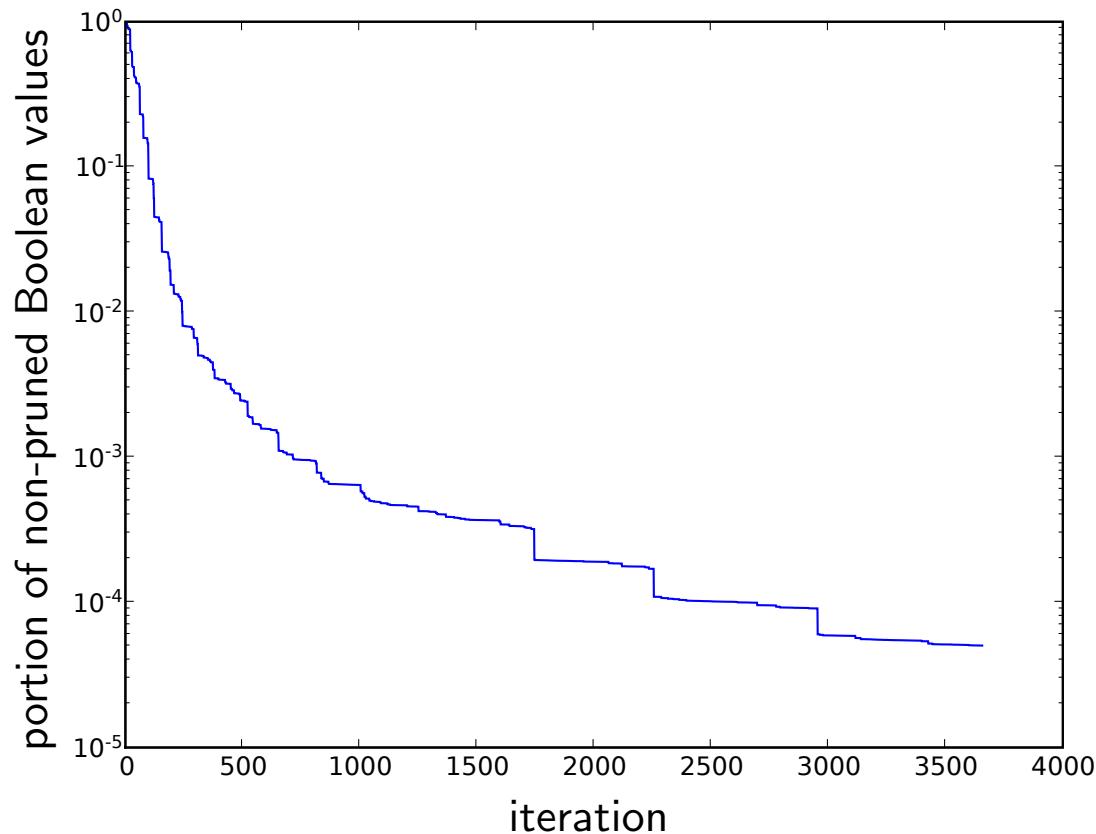
## Larger example

- random problem with 50 variables, 100 constraints
- $2^{50} \approx 10^{15}$
- took 3665 iterations (1300 to find an optimal point)
- minimum cardinality 31
- same example as used in  $\ell_1$ -norm methods lecture
  - basic  $\ell_1$ -norm relaxation (1 LP) gives  $x$  with  $\text{card}(x) = 44$
  - iterated weighted  $\ell_1$ -norm heuristic (4 LPs) gives  $x$  with  $\text{card}(x) = 36$

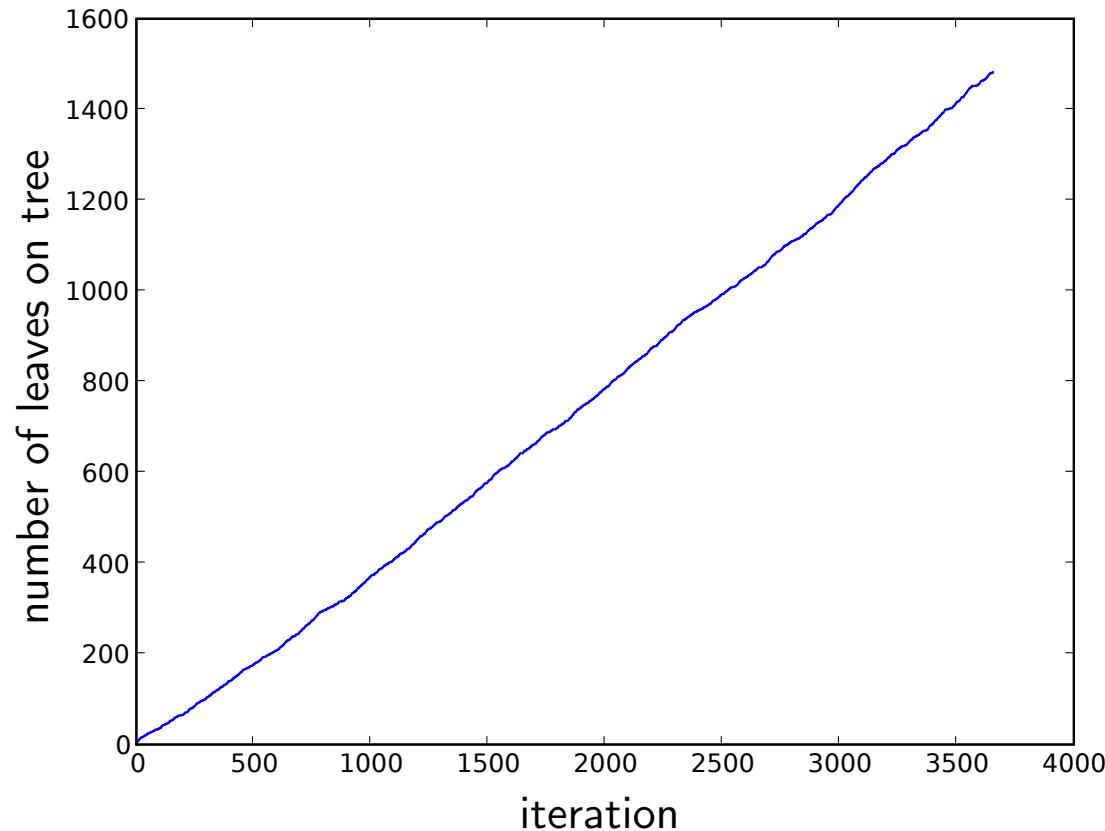
## Global lower and upper bounds



## Portion of non-pruned sparsity patterns



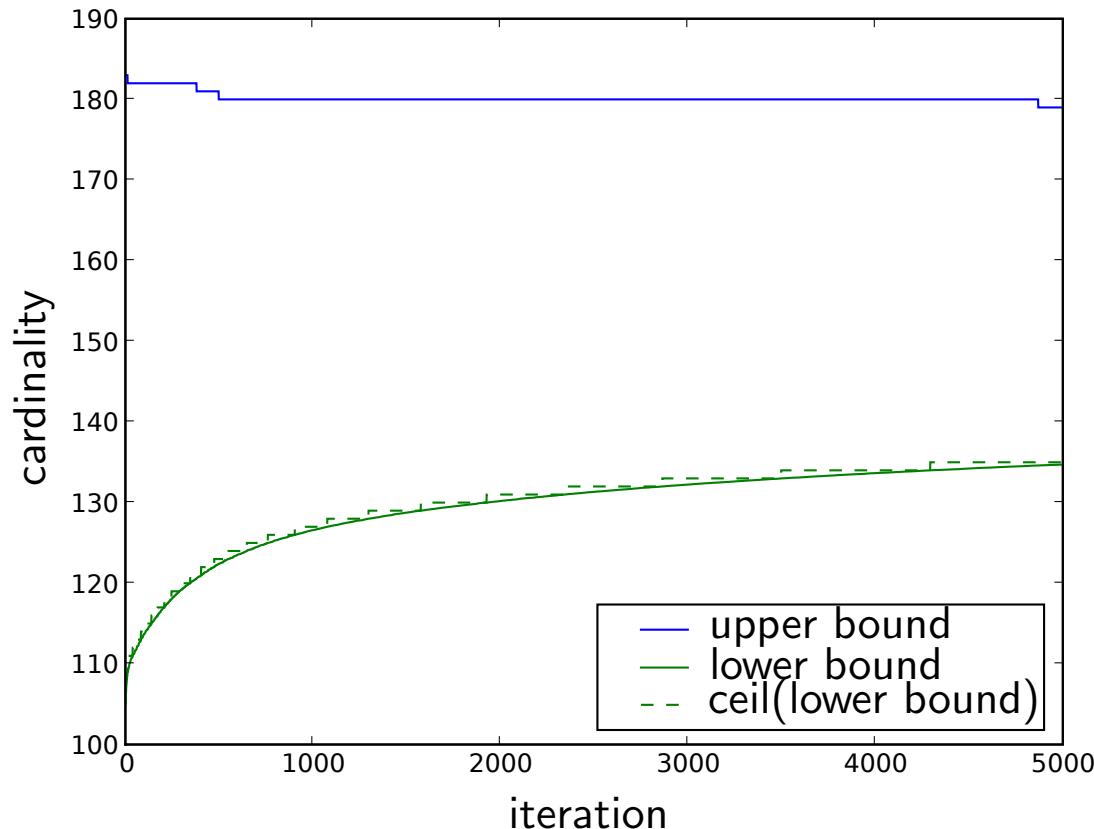
## Number of active leaves in tree



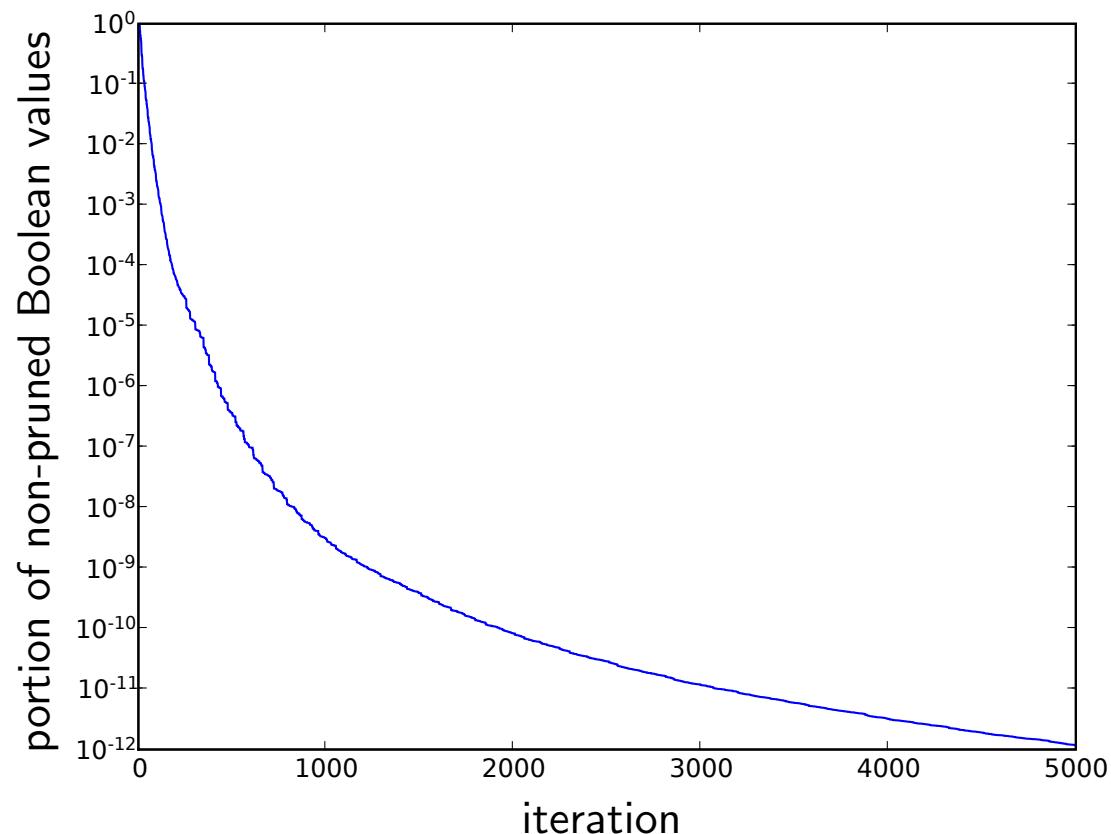
## Even larger example

- random problem with 200 variables, 400 constraints
- $2^{200} \approx 1.6 \cdot 10^{60}$
- we quit after 10000 iterations (50 hours on a single processor machine with 1 GB of RAM)
- only know that optimal cardinality is between 135 and 179
- but have reduced number of possible sparsity patterns by factor of  $10^{12}$

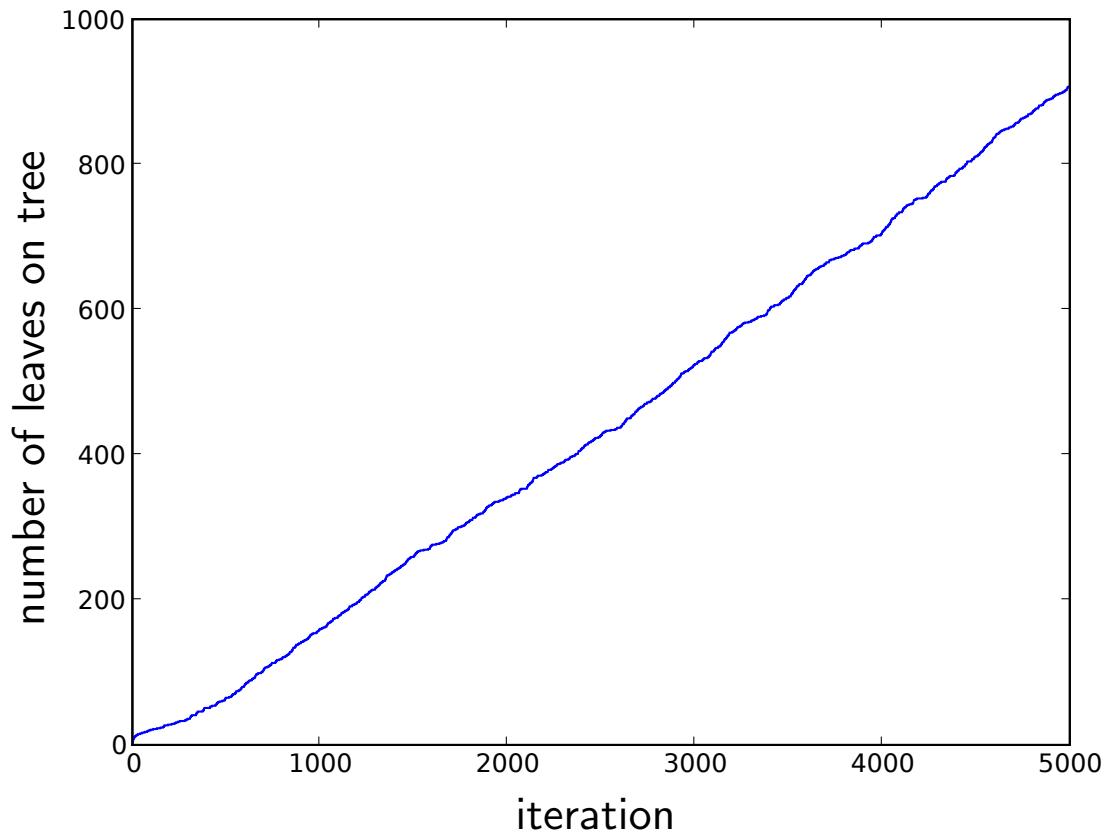
## Global lower and upper bounds



## Portion of non-pruned sparsity patterns



## Number of active leaves in tree



# Semidefinite Relaxations and Applications

- Semidefinite relaxations
- Lagrangian relaxations for QCQPs
- Randomization
- Bounds on suboptimality
- Applications

# Nonconvex problems

ee364 (more or less correct) view:

- **convex** is easy
- **nonconvex** is hard(er)

we will use convex optimization to

- find bounds on optimal value by **relaxation**
- get “good enough” feasible points by **randomization**

## Basic problem: QCQPs

$$\text{minimize } x^T A_0 x + b_0^T x + c_0$$

$$\text{subject to } x^T A_i x + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m.$$

- if all  $A_i$  are PSD, convex problem, use ee364
- here, we suppose at least one  $A_i$  is not PSD

## Example: Boolean Least Squares

Boolean least-squares problem is to

$$\text{minimize } \|Ax - b\|_2^2 \text{ subject to } x_i^2 = 1, \quad i = 1, \dots, n$$

- basic problem in digital communications (noisy channel)
- could check all  $2^n$  possible values of  $x$  . . .
- an NP-hard problem, and very hard in practice
- many heuristics for approximate solution

## Example: Partitioning Problem

two-way partitioning problem (§5.1.5 in [BV04]):

$$\begin{aligned} & \text{minimize } x^T W x \\ & \text{subject to } x_i^2 = 1, \quad i = 1, \dots, n \end{aligned}$$

where  $W = W^T$ ,  $W_{ii} = 0$

- feasible  $x \in \{-1, 1\}$  corresponds to partitioning
- coefficients  $W_{ij}$  interpreted as the cost of having the elements  $i$  and  $j$  in the same partition.
- the objective is to find the partition with least total cost
- classic particular instance: MAXCUT ( $W_{ij} \geq 0$ )

## Example: cardinality problems

minimize  $\text{card}(x)$

subject to  $x \in \mathcal{C}$

introduce  $z_i \in \{0, 1\}$ , i.e.  $z_i(1 - z_i) = 0$ ,

minimize  $\mathbf{1}^T z$

subject to  $z_i - z_i^2 = 0$ ,  $x_i(1 - z_i) = 0 \quad i = 1, \dots, n$

$x \in \mathcal{C}$

## Semidefinite relaxation

original QCQP

$$\begin{aligned} & \text{minimize } x^T A_0 x + b_0^T x + c_0 \\ & \text{subject to } x^T A_i x + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

is equivalent to

$$\begin{aligned} & \text{minimize } \mathbf{Tr}(A_0 X) + b_0^T x + c_0 \\ & \text{subject to } \mathbf{Tr}(A_i X) + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m \\ & X = xx^T \end{aligned}$$

change  $X = xx^T$  into  $X \succeq xx^T$

## Lagrangian relaxation

original QCQP

$$\begin{aligned} & \text{minimize } x^T A_0 x + b_0^T x + c_0 \\ & \text{subject to } x^T A_i x + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

forming Lagrangian

$$L(x, \lambda) = x^T \left( A_0 + \sum_{i=1}^m \lambda_i A_i \right) x + \left( b_0 + \sum_{i=1}^m \lambda_i b_i \right)^T x + c_0 + \lambda^T c$$

recall that

$$\inf_x \{x^T P x + q^T x + r\} = \begin{cases} r - \frac{1}{4} q^T P^\dagger q & \text{if } P \succeq 0, \quad q \in \mathcal{R}(P) \\ -\infty & \text{otherwise} \end{cases}$$

## Lagrangian relaxation: dual

$$L(x, \lambda) = x^T \left( A_0 + \sum_{i=1}^m \lambda_i A_i \right) x + \left( b_0 + \sum_{i=1}^m \lambda_i b_i \right)^T x + c_0 + \lambda^T c$$

has (for  $B = [b_1 \ \cdots \ b_m]^T \in \mathbf{R}^{m \times n}$ )

$$\begin{aligned} g(\lambda) &= \inf_x L(x, \lambda) \\ &= -\frac{1}{4} (b_0 + B^T \lambda)^T \left( A_0 + \sum_i \lambda_i A_i \right)^\dagger (b_0 + B^T \lambda) + \lambda^T c + c_0 \end{aligned}$$

## Lagrangian relaxation: dual

Taking Schur complements gives dual problem

$$\begin{aligned} & \text{maximize} \quad \frac{1}{4}\gamma + c^T\lambda + c_0 \\ \text{subject to} \quad & \begin{bmatrix} (A_0 + \sum_{i=1}^m \lambda_i A_i) & (b_0 + B^T\lambda) \\ (b_0 + B^T\lambda)^T & -\gamma \end{bmatrix} \succeq 0, \\ & \lambda \succeq 0 \end{aligned}$$

semidefinite program in variable  $\lambda \in \mathbf{R}_+^m$  and can be solved “efficiently”

## Lagrangian relaxation: Bidual

Taking dual again gives SDP

$$\text{minimize } \mathbf{Tr}(A_0 X) + b_0^T x + c_0$$

$$\text{subject to } \mathbf{Tr}(A_i X) + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m$$

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$$

in variables  $X \in \mathbf{S}^n$ ,  $x \in \mathbf{R}^n$

- have recovered original SDP relaxation “automatically”
- convexification of original problem!

## Example: Partitioning

minimize  $x^T W x$

subject to  $x_i^2 = 1, \quad i = 1, \dots, n$

no need to maintain variable  $x$ , gives relaxation (via  $X = xx^T$ )

minimize  $\text{Tr}(WX)$

subject to  $X \succeq 0, \quad \text{diag}(X) = \mathbf{1}$

## **Feasible points?**

- have lower bounds on optimal value of problem
- big question: how do we compute good feasible points?
- can we measure if our lower bound is suboptimal?

## Simplest idea: randomization

original problem

$$\text{minimize } x^T A_0 x + b_0^T x + c_0$$

$$\text{subject to } x^T A_i x + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m.$$

and relaxation

$$\text{minimize } \text{Tr}(A_0 X) + b_0^T x + c_0$$

$$\text{subject to } \text{Tr}(A_i X) + b_i^T x + c_i \leq 0, \quad i = 1, \dots, m$$

$$X - x x^T \succeq 0$$

- if  $X, x$  solve relaxed problem, then  $X - x x^T \succeq 0$  can be a covariance matrix.

## Gaussian randomization

- pick  $z$  as a Gaussian variable with  $z \sim \mathcal{N}(x, X - xx^T)$
- $z$  will solve the QCQP “on average” over this distribution

in other words:

$$\text{minimize } \mathbf{E}[z^T A_0 z + b_0^T z + r_0]$$

$$\text{subject to } \mathbf{E}[z^T A_i z + b_i^T z + c_i] \leq 0, \quad i = 1, \dots, m$$

a good feasible point obtained by sampling enough  $z$  (often more sophisticated strategies)

## Gaussian randomization

- possible to get sharper guarantees and exactly feasible points, e.g. for MAXCUT or other boolean problems
- constraint

$$x_i^2 = 1$$

so just take  $x_i = \text{sign}(z_i)$

- for  $\hat{x} = \text{sign}(z_i)$ ,  $z_i \sim \mathcal{N}(0, X)$ , have

$$\mathbf{E}[\hat{x}_i \hat{x}_j] = \frac{2}{\pi} \arcsin(X_{ij})$$

## Approximation guarantees

MAXCUT relaxation

$$\begin{aligned} & \text{maximize } \mathbf{Tr}(WX) \\ & \text{subject to } \mathbf{diag}(X) = \mathbf{1}, X \succeq 0 \end{aligned}$$

gives

$$\mathbf{E}[\hat{x}^T W \hat{x}] = \frac{2}{\pi} \mathbf{E}[W \arcsin(X)]$$

- draw a few samples  $\hat{x}$ , get at least that good with high probability
- optimal value of MAXCUT is between  $\frac{2}{\pi} \mathbf{Tr}(W \arcsin(X))$  and  $\mathbf{Tr}(WX)$ .

## Better rounding (Goemans & Williamson)

suppose  $W_{ij} \geq 0$ , maximize

$$\sum_{ij} W_{ij}(1 - X_{ij}) \text{ subject to } \mathbf{diag}(X) = \mathbf{1}, X \succeq 0$$

- sample coordinates  $\hat{x}_i$  at random, get  $\mathbf{Tr}(W) - \mathbf{E}[\hat{x}^T W \hat{x}] = \mathbf{Tr}(W)$ , at least 50% optimal
- sample directions:

$$X_{ij} = v_i^T v_j \text{ with } \|v_i\| = 1$$

i.e.  $X = V^T V$  by Cholesky

- draw  $Z$  uniformly at random on unit sphere, set

$$\hat{x}_i = \mathbf{sign}(Z^T v_i)$$

## Better rounding (Goemans & Williamson)

expected value of cut is

$$\begin{aligned}\mathbf{E}[W_{ij}(1 - \hat{x}_i \hat{x}_j)] &= 2W_{ij} \Pr(Z \text{ separates } v_i, v_j) \\ &= 2W_{ij} \Pr(\mathbf{sign}(v_i^T Z) \neq \mathbf{sign}(v_j^T Z)) \\ &= 2W_{ij} \frac{2\theta(v_i, v_j)}{2\pi} \\ &= \frac{2}{\pi} W_{ij} \cos^{-1}(v_i^T v_j)\end{aligned}$$

so

$$\sum_{ij} \mathbf{E}[W_{ij}(1 - \hat{x}_i \hat{x}_j)] = \frac{2}{\pi} \sum_{ij} W_{ij} \cos^{-1}(X_{ij})$$

- Fact:  $\cos^{-1}(t) \geq \frac{\pi}{2}\alpha(1 - t)$ ,  $\alpha \approx .87856$

## Better rounding: final bound

- expected weight from random cut generated by optimal  $X$  is at least

$$\frac{2}{\pi} \sum_{ij} W_{ij} \cos^{-1}(X_{ij}) \geq \alpha \sum_{ij} W_{ij}(1 - X_{ij}) = \alpha \text{SDP}^*.$$

- alternatives: if  $W \succeq 0$ , then (Nesterov 98)

$$\mathbf{Tr}(W \arcsin(X)) \geq \mathbf{Tr}(WX)$$

so (using earlier bound)

$$\text{SDP}^* \geq \text{OPT} \geq \frac{2}{\pi} \text{SDP}^*$$

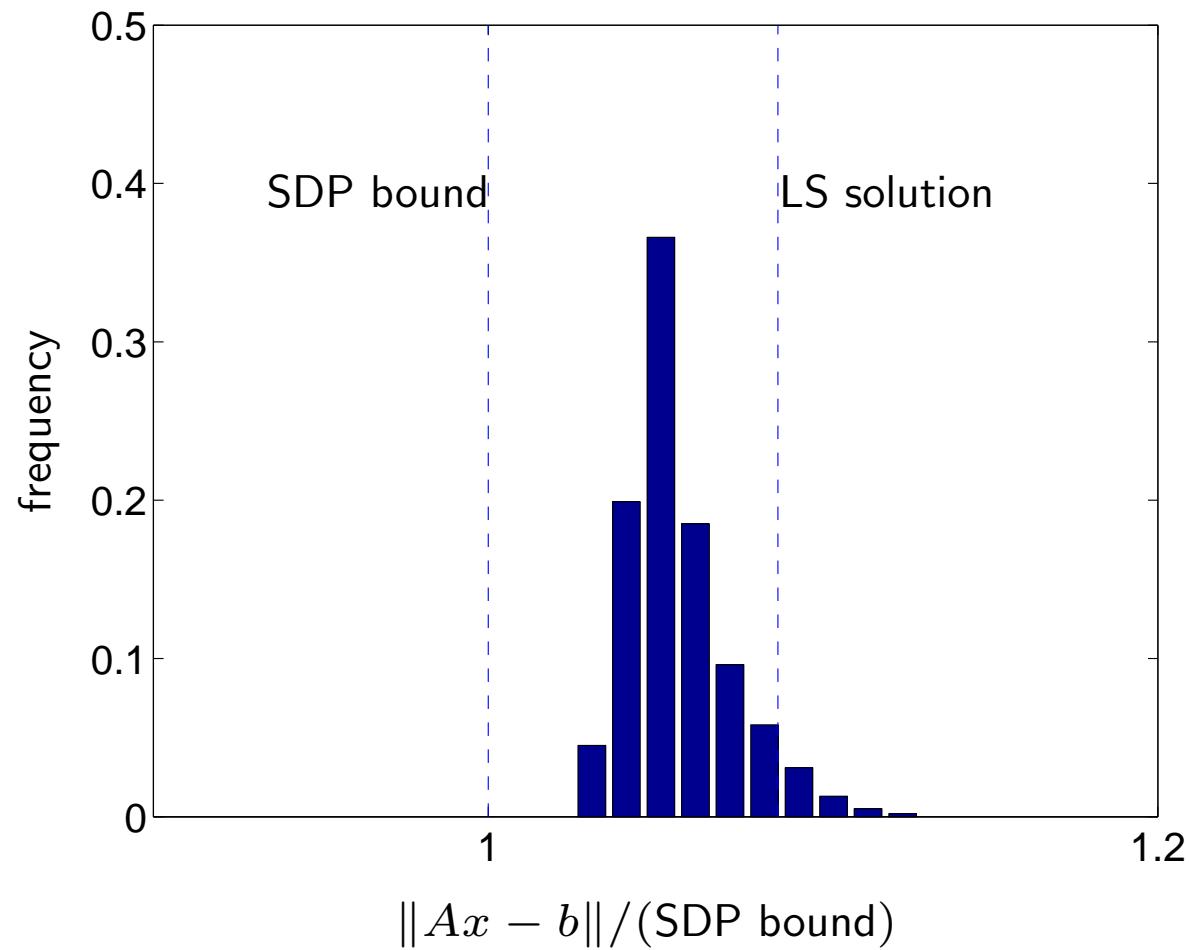
## Example: boolean least squares

- (randomly chosen) parameters  $A \in \mathbf{R}^{150 \times 100}$ ,  $b \in \mathbf{R}^{150}$
- $x \in \mathbf{R}^{100}$ , so feasible set has  $2^{100} \approx 10^{30}$  points

**LS approximate solution:** minimize  $\|Ax - b\|$  s.t.  $\|x\|_2^2 \leq n$ , then round yields objective 8.7% over SDP relaxation bound

**randomized method:** (using SDP optimal distribution)

- best of 20 samples: 3.1% over SDP bound
- best of 1000 samples: 2.6% over SDP bound



## Example: partitioning problem

$$\text{minimize } x^T W x$$

$$\text{subject to } x_i^2 = 1, \quad i = 1, \dots, n$$

with SDP relaxation

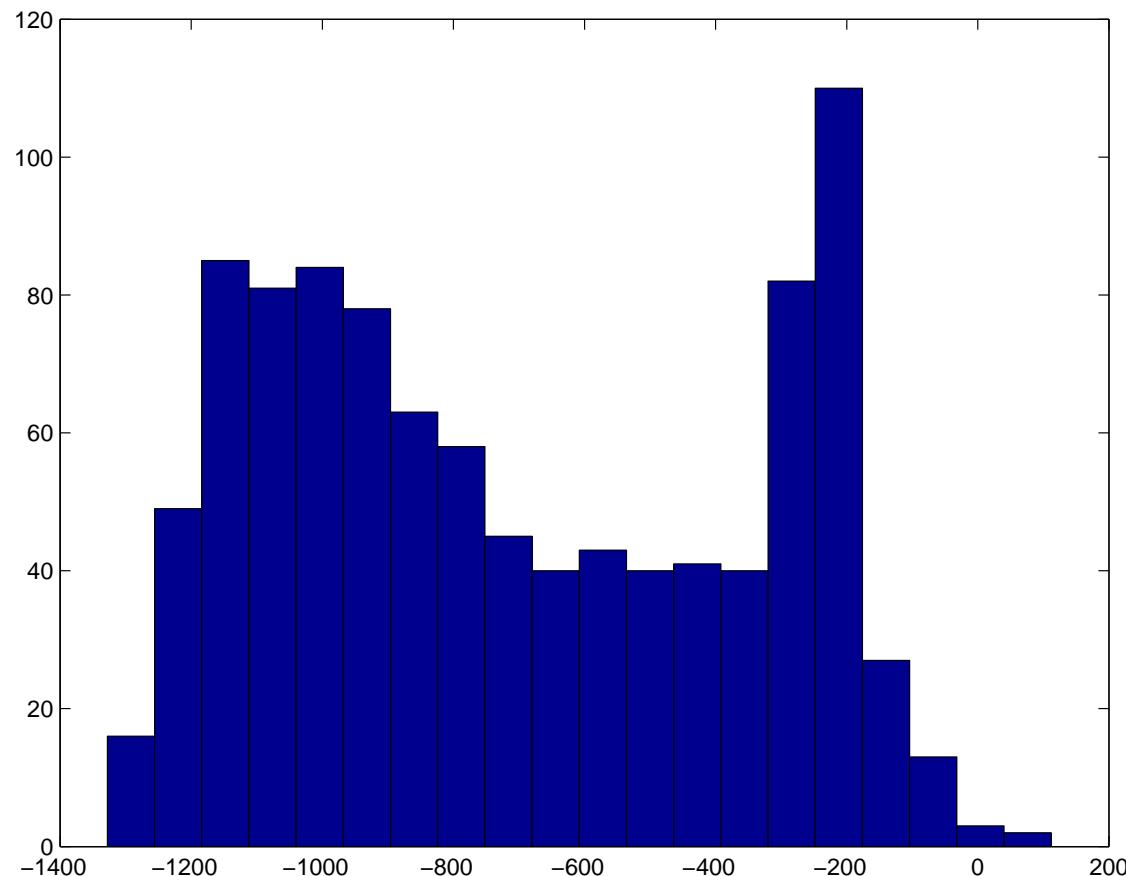
$$\text{minimize } \text{Tr}(W X)$$

$$\text{subject to } \text{diag}(X) = \mathbf{1}, X \succeq 0$$

and solution  $X^{\text{opt}}$

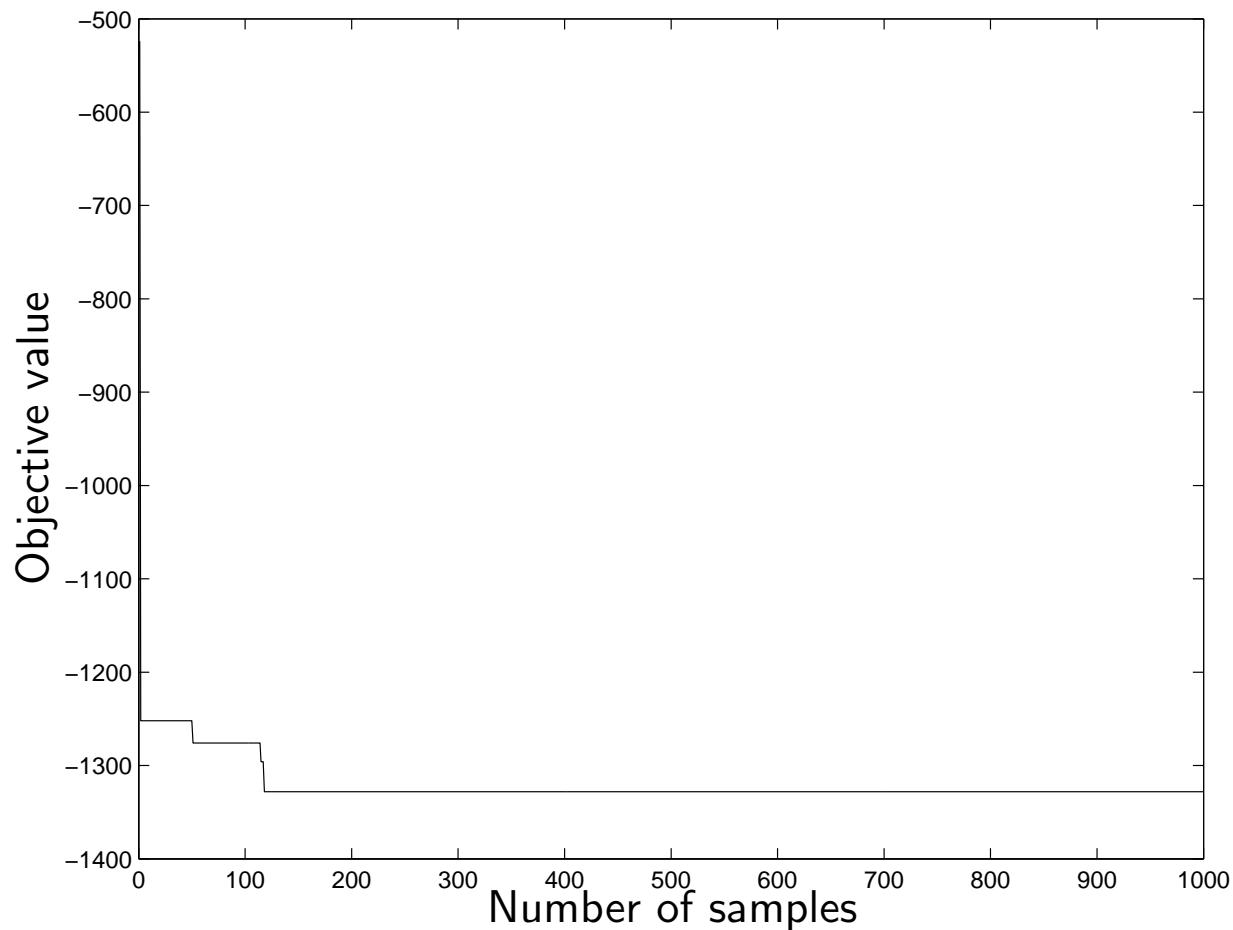
- generate samples  $x^{(i)} \sim \mathcal{N}(0, X^{\text{opt}})$ ,  $\hat{x}^{(i)} = \text{sign}(x^{(i)})$
- take one with lowest cost (SDP<sup>opt</sup> is  $-1641$ )

## Histogram of partitions



heuristic on 1000 samples: minimum value attained is  $-1328$

## Objective progress in partitioning



know optimal cost is between  $-1641$  and  $-1328$

# Robust Optimization

- definitions of robust optimization
- robust linear programs
- robust cone programs
- chance constraints

## Robust optimization

convex objective  $f_0 : \mathbf{R}^n \rightarrow \mathbf{R}$ , uncertainty set  $\mathcal{U}$ , and  $f_i : \mathbf{R}^n \times \mathcal{U} \rightarrow \mathbf{R}$ ,

$x \mapsto f_i(x, u)$  convex for all  $u \in \mathcal{U}$

general form

minimize  $f_0(x)$

subject to  $f_i(x, u) \leq 0$  for all  $u \in \mathcal{U}, i = 1, \dots, m$ .

equivalent to

minimize  $f_0(x)$

subject to  $\sup_{u \in \mathcal{U}} f_i(x, u) \leq 0, i = 1, \dots, m$ .

- Bertsimas, Ben-Tal, El-Ghaoui, Nemirovski (1990s–now)

## Setting up robust problem

- can always replace objective  $f_0$  with  $\sup_{u \in \mathcal{U}} f_0(x, u)$ , rewrite in epigraph form to

minimize  $t$

subject to  $\sup_u f_0(x, u) \leq t, \sup_u f_i(x, u) \leq 0, i = 1, \dots, m$

- equality constraints make no sense: a robust equality  $a^T(x + u) = b$  for all  $u \in \mathcal{U}$ ?

**three questions:**

- is robust formulation useful?
- is robust formulation computable?
- how should we choose  $\mathcal{U}$ ?

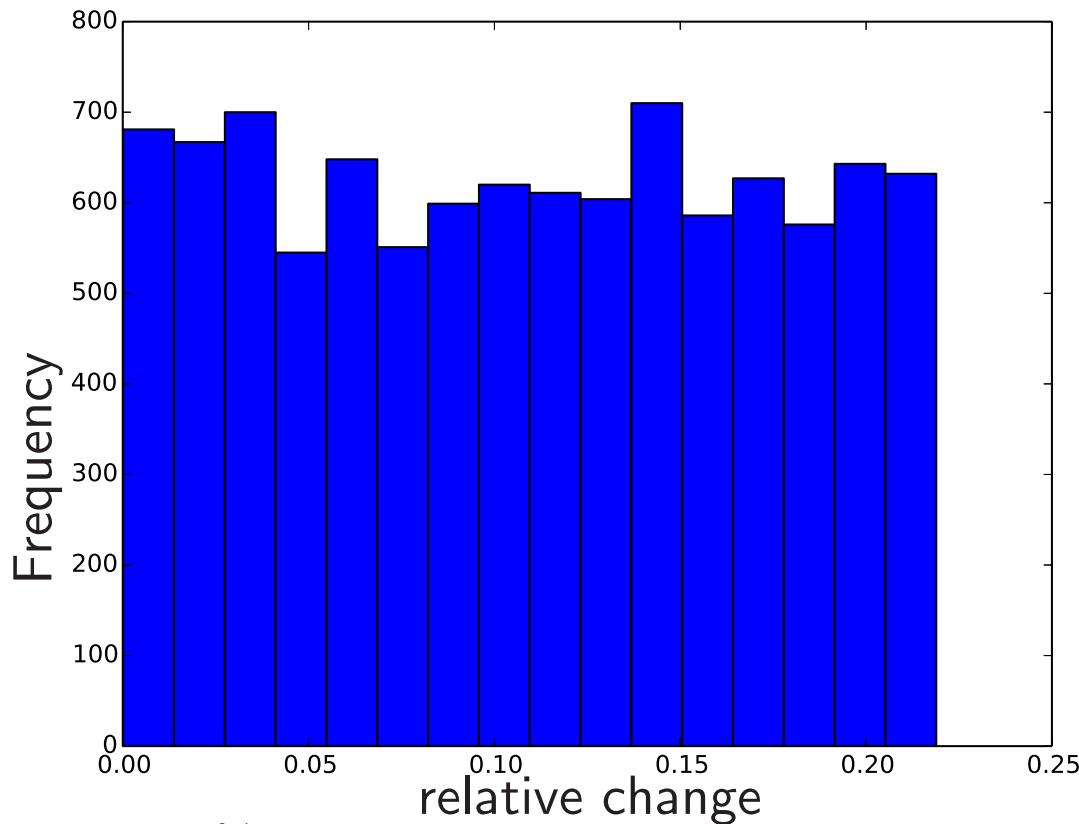
## Example failure for linear programming

$$c = \begin{bmatrix} 100 \\ 199.9 \\ -5500 \\ -6100 \end{bmatrix} \quad A = \begin{bmatrix} -.01 & -.02 & .5 & .6 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 90 & 100 \\ 0 & 0 & 40 & 50 \\ 100 & 199.9 & 700 & 800 \\ & & -I_4 & \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 1000 \\ 2000 \\ 800 \\ 100000 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

$c$  vector of costs/profits for two drugs, constraints  $Ax \leq b$  on production

- what happens if we vary percentages .01, .02 (chemical composition of raw materials) by .5% and 2%, i.e.  $.01 \pm .00005$  and  $.02 \pm .0004$ ?

## Example failure for linear programming



Frequently lose 15–20% of profits

## Alternative robust LP

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } (A + \Delta)x \preceq b, \quad \text{all } \Delta \in \mathcal{U} \end{aligned}$$

where  $|\Delta_{11}| \leq .00005$ ,  $|\Delta_{12}| \leq .0004$ ,  $\Delta_{ij} = 0$  otherwise

- solution  $x_{\text{robust}}$  has degradation *provably* no worse than 6%

## How to choose uncertainty sets

- uncertainty set  $\mathcal{U}$  a modeling choice
- common idea: let  $U$  be random variable, want constraints that

$$\mathbf{Prob}(f_i(x, U) \geq 0) \leq \epsilon \quad (1)$$

- typically hard (non-convex except in special cases)
- find set  $\mathcal{U}$  such that  $\mathbf{Prob}(U \in \mathcal{U}) \geq 1 - \epsilon$ , then sufficient condition for (1)

$$f_i(x, u) \leq 0 \text{ for all } u \in \mathcal{U}$$

## Uncertainty set with Gaussian data

minimize  $c^T x$

subject to  $\mathbf{Prob}(a_i^T x > b_i) \leq \epsilon, i = 1, \dots, m$

coefficient vectors  $a_i$  i.i.d.  $\mathcal{N}(\bar{a}, \Sigma)$  and failure probability  $\epsilon$

- marginally  $a_i^T x \sim \mathcal{N}(\bar{a}_i^T x, x^T \Sigma x)$

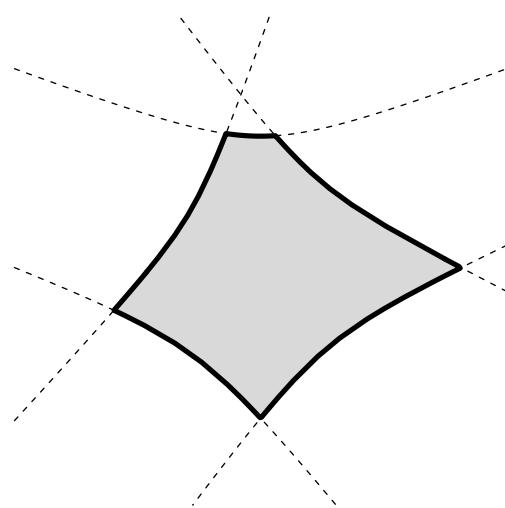
- for  $\epsilon = .5$ , just LP

minimize  $c^T x$  subject to  $a_i^T x \leq b_i, i = 1, \dots, m$

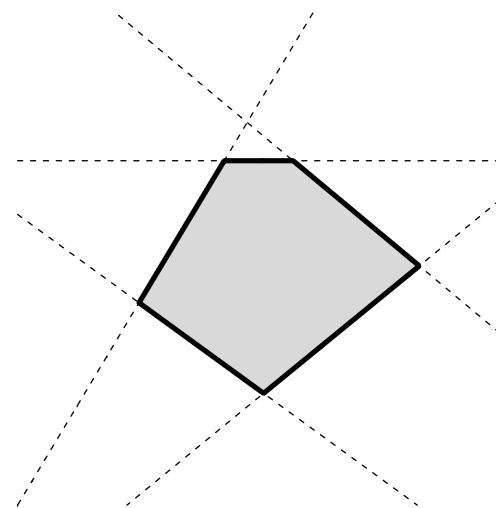
- what about  $\epsilon = .1, .9?$

## Gaussian uncertainty sets

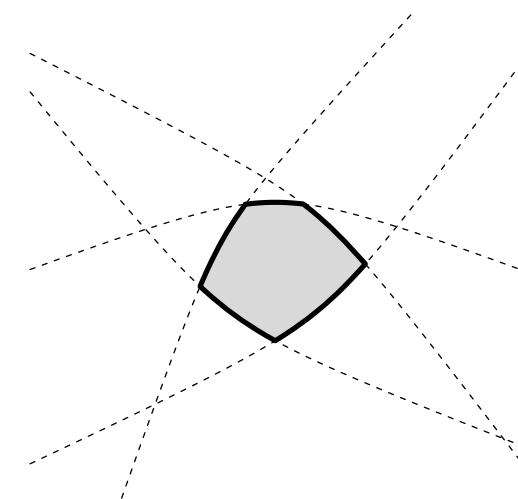
$$\{x \mid \mathbf{Prob}(a_i^T x > b_i) \leq \epsilon\} = \{x \mid \bar{a}_i^T x - b_i - \Phi^{-1}(\epsilon)\sqrt{x^T \Sigma x} \leq 0\}$$



$$\epsilon = .9$$



$$\epsilon = .5$$



$$\epsilon = .1$$

## Problem is convex, so no problem?

not quite...

consider quadratic constraint

$$\|Ax + Bu\|_2 \leq 1 \text{ for all } \|u\|_\infty \leq 1$$

- convex quadratic *maximization* in  $u$
- solutions on extreme points  $u \in \{-1, 1\}^n$
- and NP-hard to maximize (even approximately [Håstad]) convex quadratics over hypercube

## Robust LPs

Important question: when is a robust LP still an LP (robust SOCP an SOCP, robust SDP an SDP)

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } (A + U)x \preceq b \text{ for } U \in \mathcal{U}. \end{aligned}$$

can always represent formulation constraint-wise, consider only one inequality

$$(a + u)^T x \leq b \text{ for all } u \in \mathcal{U}.$$

- Simple example:  $\mathcal{U} = \{u \in \mathbf{R}^n \mid \|u\|_\infty \leq \delta\}$ , then

$$a^T x + \delta \|x\|_1 \leq b$$

## Polyhedral uncertainty

for matrix  $F \in \mathbf{R}^{m \times n}$ ,  $g \in \mathbf{R}^m$ ,

$$(a + u)^T x \leq b \quad \text{for } u \in \mathcal{U} = \{u \in \mathbf{R}^n \mid Fu + g \succeq 0\}.$$

**duality** essential for transforming (semi-)infinite inequality into tractable problem

- Lagrangian for maximizing  $u^T x$ :

$$L(u, \lambda) = x^T u + \lambda^T (Fu + g), \quad \sup_u L(u, \lambda) = \begin{cases} +\infty & \text{if } F^T \lambda + x \neq 0 \\ \lambda^T g & \text{if } F^T \lambda + x = 0. \end{cases}$$

- gives equivalent inequality constraints

$$a^T x + \lambda^T g \leq b, \quad F^T \lambda + x = 0, \quad \lambda \succeq 0.$$

## Portfolio optimization (with robust LPs)

- $n$  assets  $i = 1, \dots, n$ , random multiplicative return  $R_i$  with  $\mathbf{E}[R_i] = \mu_i \geq 1$ ,  $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$
- “certain” problem has solution  $x_{\text{nom}} = e_1$ ,

$$\text{maximize } \mu^T x \text{ subject to } x^T \mathbf{1} = 1, x \succeq 0$$

- if asset  $i$  varies in range  $\mu_i \pm u_i$ , robust problem

$$\text{maximize } \sum_{i=1}^n \inf_{u \in [-u_i, u_i]} (\mu_i + u)x_i \text{ subject to } \mathbf{1}^T x = 1, x \succeq 0$$

and equivalent

$$\text{maximize } \mu^T x - u^T x \text{ subject to } \mathbf{1}^T x = 1, x \succeq 0$$

## Robust LPs as SOCPs

norm-based uncertainty on data vectors  $a$ ,

$$(a + Pu)^T x \leq b \quad \text{for } u \in \mathcal{U} = \{u \in \mathbf{R}^m \mid \|u\| \leq 1\},$$

gives dual-norm constraint

$$a^T x + \|P^T x\|_* \leq b$$

## Portfolio optimization (tighter control)

- Returns  $R_i \in [\mu_i - u_i, \mu_i + u_i]$  with  $\mathbf{E} R_i = \mu_i$
- guarantee return with probability  $1 - \epsilon$

$$\underset{\mu, t}{\text{maximize}} \quad t \quad \text{subject to} \quad \mathbf{Prob} \left( \sum_{i=1}^n R_i x_i \geq t \right) \geq 1 - \epsilon$$

- *value at risk* is non-convex in  $x$ , approximate it?
- approximate with high-probability bounds
- less conservative than LP (certain returns) approach

## Portfolio optimization: probability approximation

- Hoeffding's inequality

$$\mathbf{Prob} \left( \sum_{i=1}^n (R_i - \mu_i)x_i \leq -t \right) \leq \exp \left( -\frac{t^2}{2 \sum_{i=1}^n x_i^2 u_i^2} \right).$$

- written differently

$$\mathbf{Prob} \left[ \sum_{i=1}^n R_i x_i \leq \mu^T x - t \left( \sum_{i=1}^n u_i^2 x_i^2 \right)^{\frac{1}{2}} \right] \leq \exp \left( -\frac{t^2}{2} \right)$$

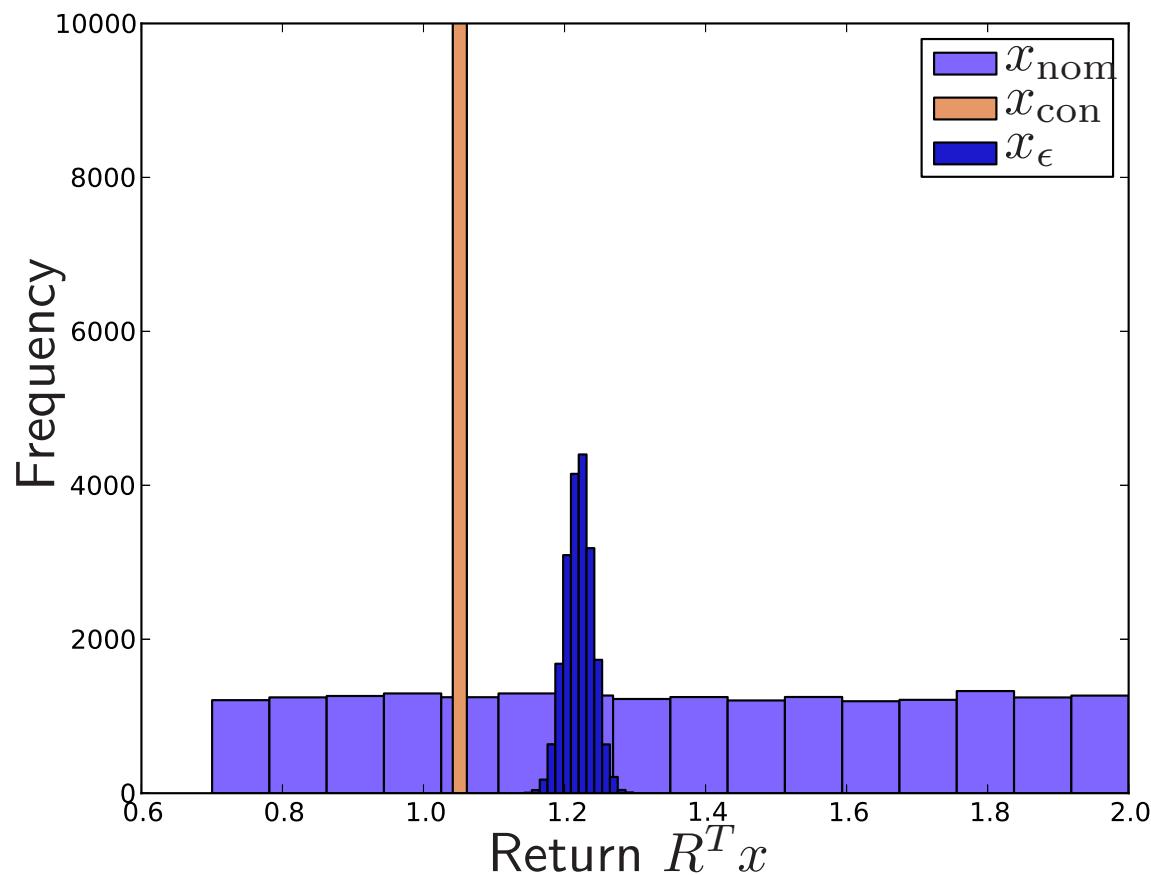
- set  $t = \sqrt{2 \log(1/\epsilon)}$ , gives robust problem

$$\text{maximize } \mu^T x - \sqrt{2 \log \frac{1}{\epsilon}} \|\text{diag}(u)x\|_2 \quad \text{subject to } \mathbf{1}^T x = 1, \quad x \succeq 0.$$

## Portfolio optimization comparison

- data  $\mu_i = 1.05 + \frac{3(n-i)}{10n}$ , uncertainty  $|u_i| \leq u_i = .05 + \frac{n-i}{2n}$  and  $u_n = 0$
- nominal minimizer  $x_{\text{nom}} = e_1$
- conservative (LP) minimizer  $x_{\text{con}} = e_n$  (guaranteed 5% return),
- robust (SOCP) minimizer  $x_\epsilon$  for value-at risk  $\epsilon = 2 \times 10^{-4}$

## Portfolio optimization comparison



Returns chosen randomly in  $\mu_i \pm u_i$ , 10,000 experiments

## LPs with conic uncertainty

- convex cone  $K$ , dual cone  $K^* = \{v \in \mathbf{R}^m \mid v^T x \geq 0, \text{ all } x \in K\}$
- recall  $x \succeq_K y$  iff  $x - y \in K$
- robust inequality

$$(a + u)^T x \leq b \text{ for all } u \in \mathcal{U} = \{u \in \mathbf{R}^n \mid Fu + g \succeq_K 0\}$$

- under constraint qualification, equivalent to

$$a^T x + \lambda^T g \leq b, \quad \lambda \succeq_{K^*} 0, \quad x + F^T \lambda = 0$$

## Example calculation: LP with semidefinite uncertainty

- symmetric matrices  $A_0, A_1, \dots, A_m \in \mathbf{S}^k$ , robust counterpart to  $a^T x \leq b$

$$(a + Pu)^T x \leq b \quad \text{for all } u \text{ s.t. } A_0 + \sum_{i=1}^m u_i A_i \succeq 0$$

- cones  $K = \mathbf{S}_+^k$ ,  $K^* = \mathbf{S}_+^k$
- Slater condition:  $\bar{u}$  such that  $A_0 + \sum_i A_i \bar{u}_i \succ 0$
- duality gives equivalent representation

$$a^T x + \mathbf{Tr}(\Lambda A_0) \leq b, \quad P^T x + \begin{bmatrix} \mathbf{Tr}(\Lambda A_1) \\ \vdots \\ \mathbf{Tr}(\Lambda A_m) \end{bmatrix} = 0, \quad \Lambda \succeq 0.$$

## Robust second-order cone problems

- Lorentz/SOCP cone, nominal inequality

$$\|Ax + b\|_2 \leq c^T x + d$$

- $A = [a_1 \ \cdots \ a_n]^T \in \mathbf{R}^{m \times n}$ , allow  $A, c$  to vary
- interval uncertainty
- ellipsoidal uncertainty
- matrix uncertainty

## SOCPs with interval uncertainty

entries  $A_{ij}$  perturbed by  $\Delta_{ij}$  with  $|\Delta_{ij}| \leq \delta$ ,  $c$  by cone:

$$\|(A + \Delta)x + b\|_2 \leq (c + u)^T x + d \quad \text{all } \|\Delta\|_\infty \leq \delta, \quad u \in \mathcal{U}$$

- split into two inequalities (first is robust LP)

$$\|(A + \Delta)x + b\|_2 \leq t, \quad t \leq (c + u)^T x + d$$

second

$$\begin{aligned} \sup_{\Delta: |\Delta_{ij}| \leq \delta} \|(A + \Delta)x + b\|_2 &= \sup_{\Delta: |\Delta_{ij}| \leq \delta} \left( \sum_{i=1}^m [(a_i + \Delta_i)^T x + b_i]^2 \right)^{1/2} \\ &= \sup_{\Delta \in \mathbf{R}^{m \times n}} \left\{ \|z\|_2 \mid z_i = a_i^T x + \Delta_i^T x + b_i, \|\Delta_i\|_\infty \leq \delta \right\} \\ &= \inf \left\{ \|z\|_2 \mid z_i \geq |a_i^T x + b| + \delta \|x\|_1 \right\}. \end{aligned}$$

## SOCPs with ellipse-like uncertainty

- matrices  $P_1, \dots, P_m \in \mathbf{R}^{n \times n}$ ,  $u \in \mathbf{R}^m$  with  $\|u\| \leq 1$
- robust/uncertain inequality

$$\left( \sum_{i=1}^m [(a_i + P_i u)^T x + b_i]^2 \right)^{1/2} \leq t \text{ for all } u \text{ s.t. } \|u\|_2 \leq 1.$$

- rewrite  $z_i \geq \sup_{\|u\| \leq 1} |a_i^T x + b_i + u^T P_i^T x|$ , equivalent

$$\|z\|_2 \leq t, \quad z_i \geq |a_i^T x + b_i| + \|P_i^T x\|_*, \quad i = 1, \dots, m.$$

## SOCPs with matrix uncertainty

- Matrix  $P \in \mathbf{R}^{m \times n}$  and radius  $\delta$ , uncertain inequality

$$\|(A + P\Delta)x + b\|_2 \leq t, \quad \text{for } \Delta \in \mathbf{R}^{n \times n} \text{ s.t. } \|\Delta\| \leq \delta,$$

- tool one: Schur complements gives equivalence of

$$\|x\|_2 \leq t \quad \text{and} \quad \begin{bmatrix} t & x^T \\ x & tI_n \end{bmatrix} \succeq 0.$$

- tool two: homogeneous *S*-lemma

$$x^T Ax \geq 0 \text{ implies } x^T Bx \geq 0 \quad \text{if and only if} \quad \exists \lambda \geq 0 \text{ s.t. } B \succeq \lambda A.$$

## SOCPs with matrix uncertainty

$$\|(A + P\Delta)x + b\|_2 \leq t, \quad \text{for } \Delta \in \mathbf{R}^{n \times n} \text{ s.t. } \|\Delta\| \leq \delta,$$

equivalent to

$$\begin{bmatrix} t & ((A + P\Delta)x + b)^T \\ (A + P\Delta)x + b & tI_m \end{bmatrix} \succeq 0 \quad \text{for } \|\Delta\| \leq 1.$$

or

$$ts^2 + 2s((A + P\Delta)x + b)^T v + t \|v\|_2^2 \geq 0 \quad \text{for all } s \in \mathbf{R}, v \in \mathbf{R}^m, \quad \|\Delta\| \leq 1.$$

## SOCPs with matrix uncertainty: final result

$$\|(A + P\Delta)x + b\|_2 \leq t, \quad \text{for } \Delta \in \mathbf{R}^{n \times n} \text{ s.t. } \|\Delta\| \leq \delta,$$

equivalent to

$$\begin{bmatrix} t & (Ax + b)^T & x^T \\ Ax + b & t - \lambda PP^T & 0 \\ x & 0 & \lambda I_n \end{bmatrix} \succeq 0.$$

## Example: robust regression

$$\text{minimize} \|Ax - b\|_2$$

where  $A$  corrupted by Gaussian noise,

$$A = A_\star + \Delta \quad \text{for } \Delta_{ij} \sim \mathcal{N}(0, 1)$$

decide to be robust to  $\Delta$  by

- bounding individual entries  $\Delta_{ij}$
- bounding norms of rows  $\Delta_i$
- bounding ( $\ell_2$ -operator) norm of  $\Delta$

## Choice of uncertainty in robust regression

**Theorem** [e.g. Vershynin 2012] Let  $\Delta \in \mathbf{R}^{m \times n}$  have i.i.d.  $\mathcal{N}(0, 1)$  entries. For all  $t \geq 0$ , the following hold:

- For each pair  $i, j$

$$\mathbf{Prob}(|\Delta_{ij}| \geq t) \leq 2 \exp\left(-\frac{t^2}{2}\right).$$

- For each  $i$

$$\mathbf{Prob}(\|\Delta_i\|_2 \geq \sqrt{n} + t) \leq \exp\left(-\frac{t^2}{2}\right).$$

- For the entire matrix  $\Delta$ ,

$$\mathbf{Prob}(\|\Delta\| \geq \sqrt{m} + \sqrt{n} + t) \leq \exp\left(-\frac{t^2}{2}\right).$$

## Choice of uncertainty in robust regression

**idea:** choose bounds  $t(\delta)$  to guarantee  $\mathbf{Prob}(\text{deviation} \geq t(\delta)) \leq \delta$

- coordinate-wise:  $t_\infty(\delta)^2 = 2 \log \frac{2mn}{\delta}$ ,

$$\mathbf{Prob}(\max_{i,j} |\Delta_{ij}| \geq t_\infty(\delta)) \leq 2mn \exp\left(-\frac{t_\infty(\delta)^2}{2}\right) = \delta$$

- row-wise:  $t_2(\delta)^2 = 2 \log \frac{m}{\delta}$ ,

$$\mathbf{Prob}(\max_i \|\Delta_i\|_2 \geq t_2(\delta)) \leq m \exp\left(-\frac{t_2(\delta)^2}{2}\right) = \delta$$

- matrix-norm:  $t_{\text{op}}(\delta)^2 = 2 \log \frac{1}{\delta}$ ,

$$\mathbf{Prob}(\|\Delta\| \geq \sqrt{n} + \sqrt{m} + t_{\text{op}}(\delta)) \leq \exp\left(-\frac{t_{\text{op}}(\delta)^2}{2}\right) = \delta.$$

## Robust regression results

$$\underset{x}{\text{minimize}} \quad \sup_{\Delta \in \mathcal{U}} \|(A + \Delta)x - b\|_2$$

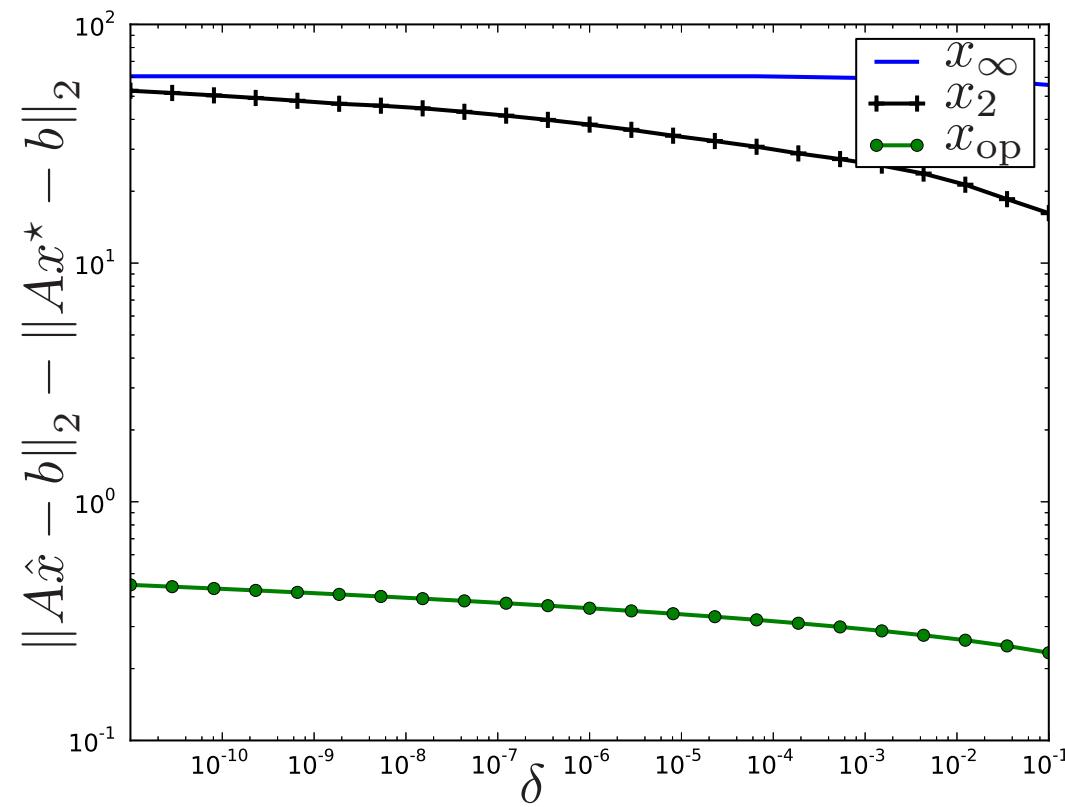
where  $\mathcal{U}$  is one of the three uncertainty sets

$$\mathcal{U}_\infty = \{\Delta \mid \|\Delta\|_\infty \leq t_\infty(\delta)\},$$

$$\mathcal{U}_2 = \{\Delta \mid \|\Delta_i\|_2 \leq \sqrt{n} + t_2(\delta) \text{ for } i = 1, \dots, m\},$$

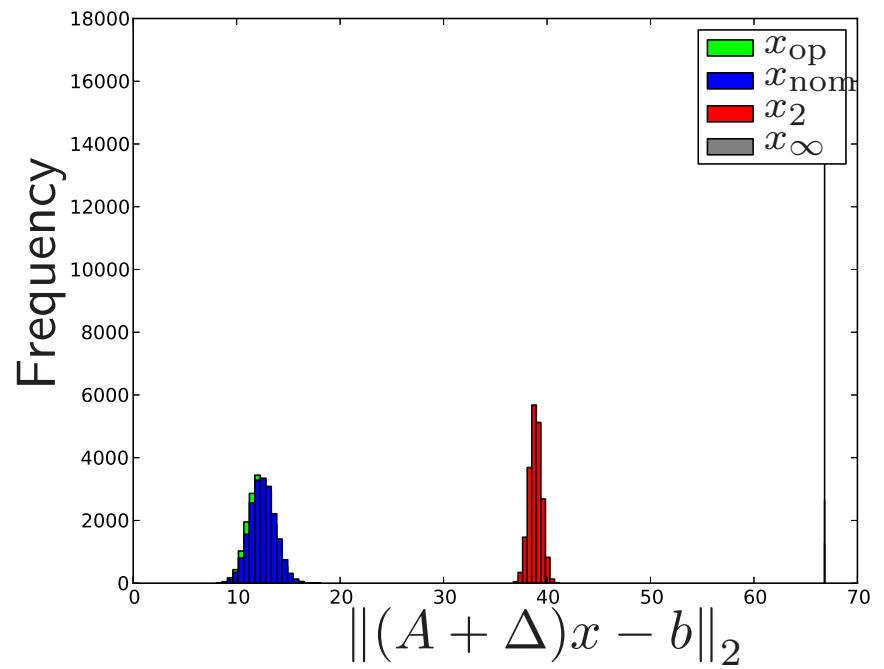
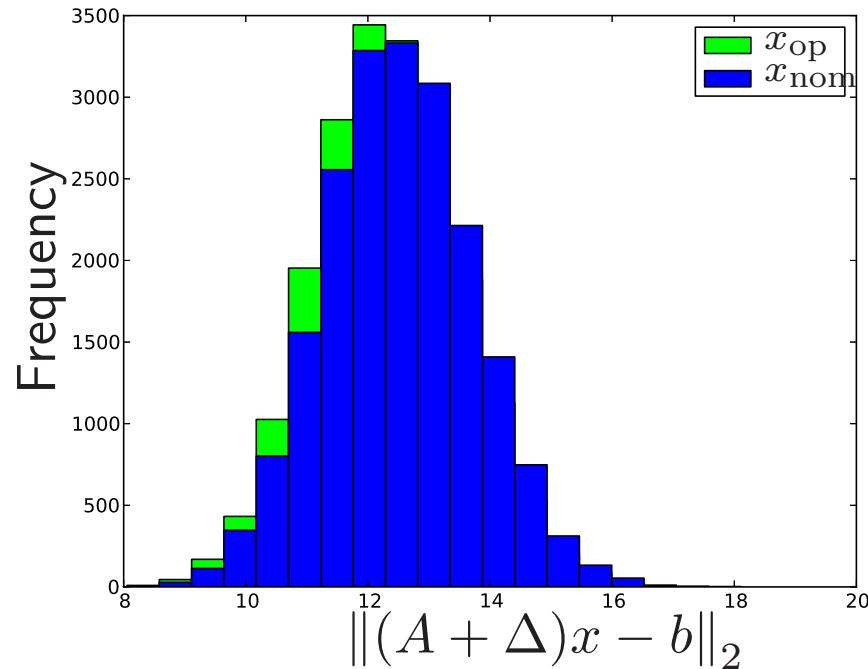
$$\mathcal{U}_{\text{op}} = \{\Delta \mid \|\Delta\| \leq \sqrt{n} + \sqrt{m} + t_{\text{op}}(\delta)\}.$$

## Robust regression results



Objective value  $\|\hat{A}\hat{x} - b\|_2 - \|Ax^* - b\|_2$  versus  $\delta$ , where  $x^*$  minimizes nominal objective and  $\hat{x}$  denotes robust solution

# Robust regression results



- residuals for the robust least squares problem  $\|(A + \Delta)x - b\|_2$
- uncertainty sets  $\mathcal{U}_{\text{nom}} = \{0\}$  vs.  $\mathcal{U}_\infty, \mathcal{U}_2, \mathcal{U}_{\text{op}}$
- experiment with  $N = 10^5$  random Gaussian matrices

# **Chance constraints and distributionally robust optimization**

- chance constraints
- approximations to chance constraints
- distributional robustness

# Chance constraints

non-convex problem

$$\text{minimize } f_0(x)$$

$$\text{subject to } \mathbf{Prob}(f_i(x, U) > 0) \leq \epsilon, \quad i = 1, \dots, m$$

**safe approximation:** find convex  $g_i : \mathbf{R}^n \rightarrow \mathbf{R}$  such that

$$g_i(x) \leq 0 \text{ implies } \mathbf{Prob}(f_i(x, U) > 0) \leq \epsilon$$

- sufficient condition: find set  $\mathcal{U}$  such that

$$\mathbf{Prob}(U \in \mathcal{U}) \geq 1 - \epsilon \text{ set } g_i(x) = \sup_{u \in \mathcal{U}} f_i(x, u)$$

## Bounds on probability of error

- sometimes useful to directly bound  $\mathbf{Prob}(f_i(x, U) > 0)$  instead of  $\mathbf{Prob}(U \in \mathcal{U})$
- **Value at risk** of random  $Z$  is

$$\mathbf{VaR}(Z; \epsilon) = \inf \{\gamma \mid \mathbf{Prob}(Z \leq \gamma) \geq 1 - \epsilon\} = \inf \{\gamma \mid \mathbf{Prob}(Z > \gamma) \leq \epsilon\}$$

- equivalence of **VaR** and deviation:

$$\mathbf{VaR}(Z; \epsilon) \leq 0 \text{ if and only if } \mathbf{Prob}(Z > 0) \leq \epsilon$$

- Gaussians: if  $U \sim \mathcal{N}(\mu, \Sigma)$  then  $x^T U - \gamma \sim \mathcal{N}(\mu^T x - \gamma, x^T \Sigma x)$  and

$$\mathbf{Prob}(x^T U \leq \gamma) = \Phi\left(\frac{\gamma - x^T U}{\sqrt{x^T \Sigma x}}\right)$$

so

$$\mathbf{VaR}(U^T x - \gamma; \epsilon) \leq 0 \quad \text{iff} \quad \gamma \geq \mu^T x + \Phi^{-1}(1 - \epsilon) \left\| \Sigma^{1/2} x \right\|_2$$

convex iff  $\epsilon \leq 1/2$

## Convex bounds on probability of error

- sometimes more usable idea: convex upper bounds on probability of error
- simple observation: if  $\phi : \mathbf{R} \rightarrow \mathbf{R}$  is non-negative, non-decreasing

$$1(z \geq 0) \leq \phi(z)$$

- consequence: for all  $\alpha > 0$ ,

$$\mathbf{Prob}(Z \geq 0) \leq \mathbf{E}\phi(\alpha^{-1}Z),$$

- so if

$$\mathbf{E}\phi(\alpha^{-1}Z) \leq \epsilon, \text{ then } \mathbf{Prob}(Z \geq 0) \leq \epsilon$$

## Perspective transforms and convex bounds

- perspective transform of function  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  is

$$f_{\text{per}}(x, \lambda) = \lambda f\left(\frac{x}{\lambda}\right)$$

- jointly convex in  $(x, \lambda) \in \mathbf{R}^n \times \mathbf{R}_+$  when  $f$  convex
- so for  $\phi(\cdot) \geq 1(\cdot)$ , better convex constraint (valid for all  $\alpha > 0$ )

$$\alpha \mathbf{E} \phi\left(\frac{f(x, U)}{\alpha}\right) \leq \alpha \epsilon$$

is convex in  $x, \alpha$  if  $f$  is

- optimize this bound,

$$\inf_{\alpha \geq 0} \left\{ \alpha \mathbf{E} \phi \left( \frac{f(x, U)}{\alpha} \right) - \alpha \epsilon \right\} \leq 0$$

- convex constraint satisfied implies that

$$\mathbf{Prob}(f(x, U) > 0) \leq \epsilon$$

## Tightest convex relaxation

- set  $\phi(z) = [1 + z]_+$ , where  $[x]_+ = \max\{x, 0\}$

$$\inf_{\alpha \geq 0} \left\{ \alpha \mathbf{E} \left[ \frac{f(x, U)}{\alpha} + 1 \right]_+ - \alpha \epsilon \right\} = \inf_{\alpha \geq 0} \left\{ \mathbf{E} [f(x, U) + \alpha]_+ - \alpha \epsilon \right\}$$

- **conditional value at risk** is

$$\text{CVaR}(Z; \epsilon) = \inf_{\alpha} \left\{ \frac{1}{\epsilon} \mathbf{E} [Z - \alpha]_+ + \alpha \right\},$$

- key inequalities:

$$\mathbf{Prob}(Z \geq 0) - \epsilon \leq \epsilon \text{CVaR}(Z; \epsilon)$$

## Interpretation of conditional value at risk

- minimize out  $\alpha$  and find

$$0 = \frac{\partial}{\partial \alpha} \left\{ \alpha + \frac{1}{\epsilon} \mathbf{E} [Z - \alpha]_+ \right\} = 1 - \frac{1}{\epsilon} \mathbf{E} 1(Z \geq \alpha) = 1 - \frac{1}{\epsilon} \mathbf{Prob}(Z \geq \alpha).$$

- value at risk plus upward deviations: set  $\alpha^*$  s.t.  $\epsilon = \mathbf{Prob}(Z \geq \alpha^*)$ ,

$$\mathbf{CVaR}(Z; \epsilon) = \frac{1}{\epsilon} \mathbf{E} [Z - \alpha^*]_+ + \alpha^* = \frac{1}{\epsilon} \mathbf{E} [Z - \alpha^*]_+ + \mathbf{VaR}(Z; \epsilon)$$

- conditional expectation version:

$$\begin{aligned} \mathbf{E}[Z \mid Z \geq \alpha^*] &= \mathbf{E}[\alpha^* + (Z - \alpha^*) \mid Z \geq \alpha^*] \\ &= \alpha^* + \frac{\mathbf{E} [Z - \alpha^*]_+}{\mathbf{Prob}(Z \geq \alpha^*)} = \alpha^* + \frac{\mathbf{E} [Z - \alpha^*]_+}{\epsilon} = \mathbf{CVaR}(Z; \epsilon). \end{aligned}$$

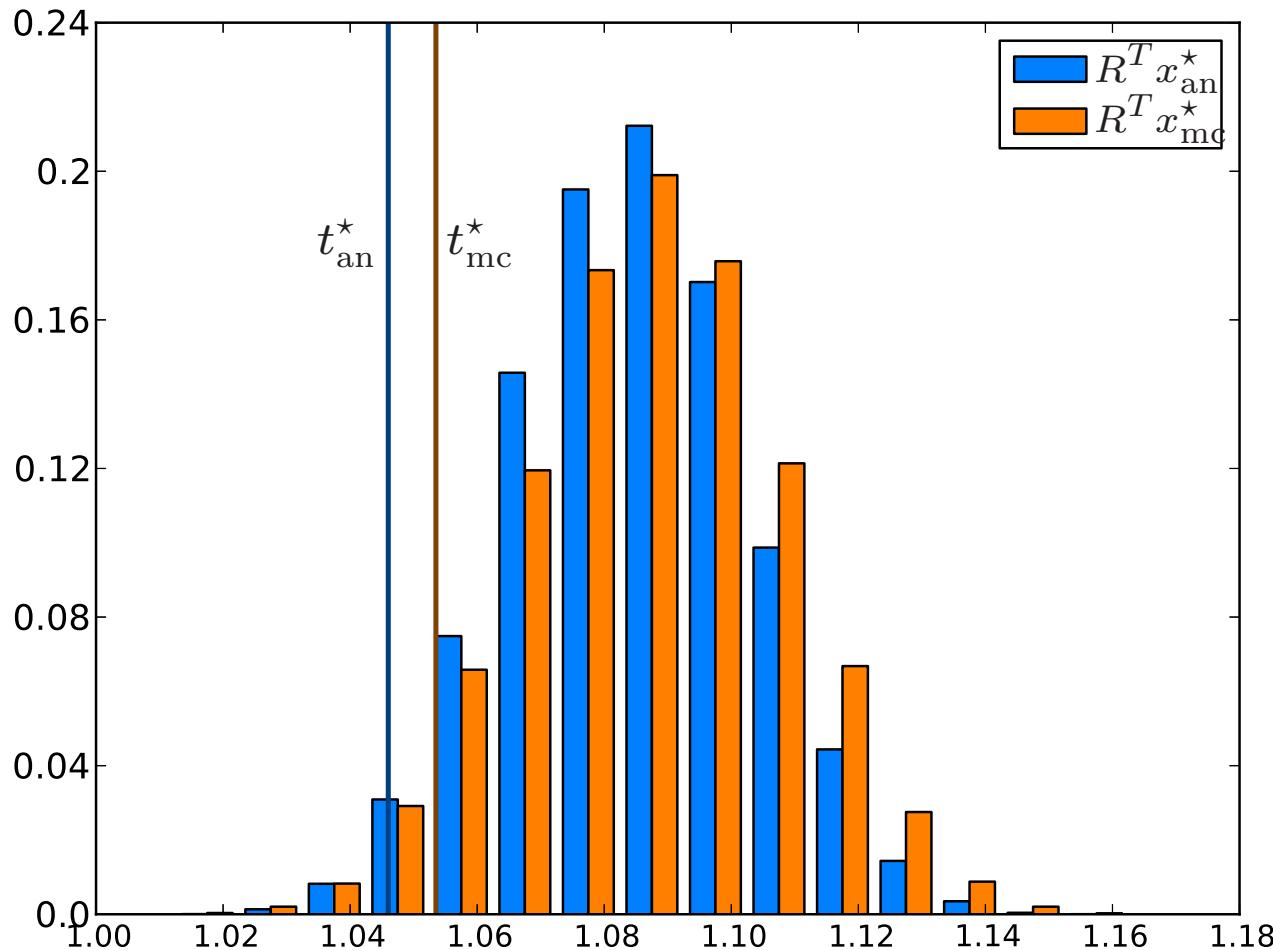
## Benefits and drawbacks of CVaR

- easy to simulate, approximate well
- typically hard to evaluate exactly; not very tractable bounds
- e.g.  $f(x, U) = U^T x$  and  $U \sim \text{Uniform}\{-1, 1\}^n$  gives combinatorial sum
- if available, analytic approximations using moment generating function (MGF) can give better behavior (notes)

## Portfolio optimization example

- $n$  assets  $i = 1, \dots, n$ , random multiplicative return  $R_i$  with  $\mathbf{E}[R_i] = \mu_i \geq 1$ ,  $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$
- asset  $i$  return varies in range  $R_i \in [\mu_i - u_i, \mu_i + u_i]$
- data  $\mu_i = 1.05 + \frac{3(n-i)}{10n}$ , uncertainty  $|u_i| \leq u_i = .05 + \frac{n-i}{2n}$  and  $u_n = 0$
- moment generating function approximation to  $\mathbf{VaR}(R^T x - t; \epsilon) \leq 0$  is

$$t - \mu^T x + \sqrt{\frac{1}{2} \log \frac{1}{\epsilon}} \|\mathbf{diag}(u)x\|_2 \leq 0$$



Monte-Carlo CVaR solution  $x_{mc}^*$  vs. analytic (MGF) approximation  $x_{an}^*$

## Distributionally robust optimization

stochastic optimization problems:

$$\underset{x}{\text{minimize}} \quad \mathbf{E}_P f(x, S) = \int f(x, s) dP(s)$$

**distributionally robust formulation:**

$$\underset{x}{\text{minimize}} \quad \sup_{P \in \mathcal{P}} \mathbf{E}_P f(x, S) = \sup_{P \in \mathcal{P}} \int f(x, s) dP(s)$$

new question: how should we choose  $\mathcal{P}$ ?

## Choices of uncertainty sets

- moment-based conditions, e.g.

$$\mathcal{P} = \{P \mid \mathbf{E}_P[h_i(S)] \preceq b_i\}$$

for function  $h_i : \mathcal{S} \rightarrow \mathbf{R}^{n_i}$

- “nonparametric” sets, using divergence-like quantities, e.g.

$$\mathcal{P} = \{P \mid \mathbf{D}_{\text{kl}}(P \| P_0) \leq \rho\}$$

- often estimate these based on sample  $S_1, \dots, S_m$

## Moment-based uncertainty

- general moment constraints: let  $K_i \subset \mathbf{R}^{n_i}$  be convex cones,  $i = 1, \dots, m$ ,  $h_i : \mathcal{S} \rightarrow \mathbf{R}^{n_i}$ , and

$$\mathcal{P} = \left\{ P \mid \int h_i(s) dP(s) \preceq_{K_i} b_i \right\}$$

- under constraint qualification (Rockafellar 1970, Isii 1963, Shapiro 2001, Delage & Ye 2010)

$$\sup_{P \in \mathcal{P}} \int f(s) dP(s) = \inf_{r, t, z} \left\{ \begin{array}{ll} r + \sum_i t_i \text{ s.t. } & r \geq f(s), \quad z_i \in K_i^* \\ & t_i \geq z_i^T b_i - z_i^T h_i(s), \quad \text{all } s \in \mathcal{S} \end{array} \right\}$$

## Uncertainty from central limit theorems

- typical idea: start with some probabilistic understanding, work to get uncertainty set
- central limit theorem: for  $S_i \in \mathbf{R}$ , drawn i.i.d.,  $\Phi$  Gaussian CDF

$$\mathbf{Prob} \left( \frac{1}{m} \sum_{i=1}^m S_i \geq \mathbf{E}S + \frac{1}{\sqrt{m}} \sqrt{\mathbf{Var}(S)}t \right) \rightarrow \Phi(-t)$$

- empirical confidence set: let  $\mathcal{U}_\rho = \{u \in \mathbf{R}^m \mid \mathbf{1}^T u = 0, \|u\|_2 \leq \rho\}$

$$\left\{ \frac{1}{m} \sum_{i=1}^m S_i + \frac{1}{m} \sum_{i=1}^m u_i S_i \mid u \in \mathcal{U}_\rho \right\} = \left\{ \bar{S}_m \pm \frac{\rho}{\sqrt{m}} \sqrt{\frac{1}{m} \sum_{i=1}^m (S_i - \bar{S}_m)^2} \right\}$$

- slight extension of CLT implies

$$\begin{aligned}
& \mathbf{Prob} \left( \bar{S}_m - \frac{\rho}{\sqrt{m}} \sqrt{\mathbf{Var}_m(S)} \leq \mathbf{E} S \leq \bar{S}_m + \frac{\rho}{\sqrt{m}} \sqrt{\mathbf{Var}_m(S)} \right) \\
&= \mathbf{Prob} (\mathbf{E} S \in \{\bar{S}_m + u^T S/m \mid u \in \mathcal{U}_\rho\}) \\
&\rightarrow \mathbf{Prob}(-\rho \leq \mathcal{N}(0, 1) \leq \rho) = \Phi(\rho) - \Phi(-\rho)
\end{aligned}$$

- natural confidence set for distributionally robust optimization:

$$\mathcal{P}_{m,\rho} = \{p \in \mathbf{R}^n \mid \mathbf{1}^T p = 1, \|p - \mathbf{1}/m\|_2 \leq \rho/m\}$$

and

$$\sup_{p \in \mathcal{P}_{m,\rho}} \sum_{i=1}^m p_i f(x, S_i)$$

## Divergence-based confidence sets

- general form of robustness set:  $\phi$ -divergences

$$\mathbf{D}_\phi(P\|Q) = \int \phi\left(\frac{p(s)}{q(s)}\right) q(s)$$

where  $\phi : \mathbf{R}_+ \rightarrow \mathbf{R}$  is convex,  $\phi(1) = 0$

- uncertainty set for  $P_m = \frac{1}{m} \sum_{i=1}^m \delta_{S_i}$  (empirical distribution)

$$\mathcal{P}_m = \{P : \mathbf{D}_\phi(P\|P_m) \leq \rho/m\}$$

- examples:  $\phi(t) = (t - 1)^2$ ,  $\phi(t) = t \log t$ ,  $\phi(t) = -\log t$

- can show (Duchi, Glynn, Namkoong) that for  $\phi$  with  $\phi''(0) = 2$  that for  $S_1, \dots, S_m \sim P_0$ ,

$$\mathbf{Prob} \left( \sup_{P \in \mathcal{P}_m} \mathbf{E}_P f(x, S) \leq \mathbf{E}_{P_0} f(x, S) \right) \rightarrow \Phi(-\rho)$$

(and versions uniform in  $x$  too)

## Dual representation of divergence-based confidence sets

if

$$\mathcal{P} = \{P \mid \mathbf{D}_\phi(P \| P_0) \leq \rho\}$$

then

$$\sup_{P \in \mathcal{P}} \mathbf{E}_P Z = \inf_{\alpha \geq 0, \eta} \left\{ \alpha \mathbf{E} \phi^* \left( \frac{Z - \eta}{\alpha} \right) + \rho \alpha + \eta \right\}$$

- example:  $\phi(t) = \frac{1}{2}t^2 - 1$  has

$$\phi^*(u) = \frac{1}{2}(u)_+^2 + 1$$

so

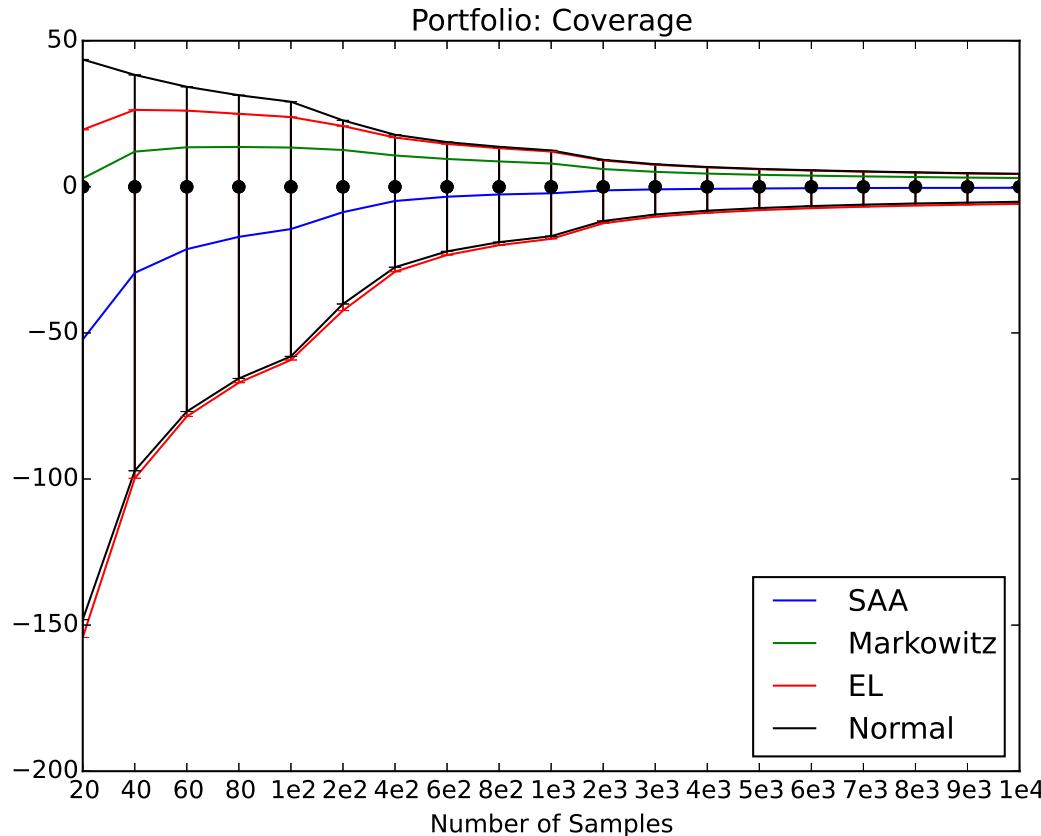
$$\sup_{P \in \mathcal{P}} \mathbf{E}_P f(x, S) = \inf_{\eta} \left\{ \sqrt{1 + \rho} (\mathbf{E}_{P_0}(f(x, S) - \eta)_+^2)^{1/2} + \eta \right\}$$

## Portfolio optimization revisited

- returns  $R \in \mathbf{R}^n$ ,  $n = 20$ , domain  
 $X = \{x \in \mathbf{R}^n \mid \mathbf{1}^T x = 1, x \in [-10, 10]\}$  (leveraging allowed)
- returns  $R \sim \mathcal{N}(\mu, \Sigma)$
- within simulation,  $\mu, \Sigma$  chosen randomly
- recall Markowitz portfolio problem to

$$\text{maximize } \frac{1}{m} \sum_{i=1}^m R_i^T x - \sqrt{\rho/m} \sqrt{x^T \Sigma_m x}$$

where  $\Sigma_m = \frac{1}{m} \sum_i (R_i - \bar{R}_m)(R_i - \bar{R}_m)^T$  is empirical covariance



compare returns of robust/empirical likelihood method, Markowitz portfolio, true gaussian results, sample average, 95% confidence

# **Dikin's Method**

Stephen Boyd (with help from AJ Friend)

EE364b, Stanford University

# **Outline**

Dikin's original method for LP

Generalized Dikin's method

## Dikin's method

- a simple proto interior-point method, originally for solving LPs
- invented by Dikin in 1967, but ignored/unknown for decades
- also called affine scaling method
- has very simple interpretation

## Standard form LP

primal problem:

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b \\ & && x \succeq 0 \end{aligned}$$

dual:

$$\begin{aligned} & \text{maximize} && -b^T \nu \\ & \text{subject to} && c + A^T \nu \succeq 0 \end{aligned}$$

optimality conditions:

- primal feasibility:  $Ax = b, x \succeq 0$
- dual feasibility:  $c + A^T \nu \succeq 0$
- zero gap/ complementarity:  $x^T(c + A^T \nu) = c^T x + b^T \nu = 0$

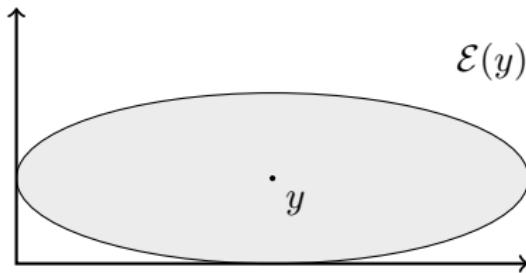
## Dikin ellipsoid

- for  $y \succ 0$ , *Dikin ellipsoid* is

$$\mathcal{E}(y) = \{x \mid (x - y)^T H(x - y) \leq 1\}, \quad H = \text{diag}(y)^{-2}$$

- $\mathcal{E}(y) \subset \mathbf{R}_+^n$ ; follows from

$$\sum_i \frac{(x_i - y_i)^2}{y_i^2} \leq 1 \Rightarrow \frac{(x_i - y_i)^2}{y_i^2} \leq 1 \Rightarrow |x_i - y_i| \leq y_i \Rightarrow x_i \geq 0$$

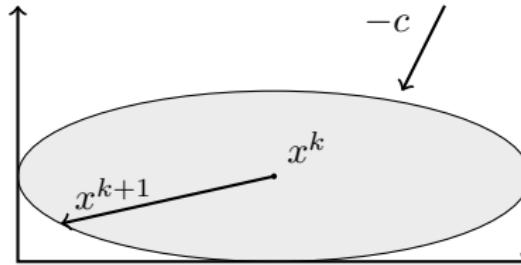


## Dikin's method

- start with (strictly feasible)  $x^0 \succ 0, Ax^0 = b$
- $x^{k+1}$  is solution of

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \mathcal{E}(x^k)\end{array}$$

- there's a simple formula for  $x^{k+1}$
- maintains feasibility
- converges to solution



## Dikin update

- with  $H = \text{diag}(x^k)^{-2}$ ,  $x^{k+1}$  is solution of

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & (x - x^k)^T H(x - x^k) \leq 1\end{array}$$

- find  $\Delta x^k = x^{k+1} - x^k$  via KKT system

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \mu\nu \end{bmatrix} = \begin{bmatrix} -\mu c \\ 0 \end{bmatrix},$$

where  $\mu$  is chosen after the solve to enforce the ellipsoid constraint

- more explicitly:

$$\nu^k = -(AH^{-1}A^T)^{-1}AH^{-1}c$$

$$s^k = -H^{-1}(c + A^T\nu^k)$$

$$\mu^k = 1/\sqrt{s^{kT} H s^k}$$

$$\Delta x^k = \mu^k s^k$$

## Comparison with barrier method

- barrier method centering problem:

$$\begin{aligned} & \text{minimize} && tc^T x - \sum_{i=1}^n \log x_i \\ & \text{subject to} && Ax = b \end{aligned}$$

- Newton step  $\Delta x^k$  given by

$$\begin{bmatrix} H^k & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ t\nu \end{bmatrix} = \begin{bmatrix} -tc + \text{diag}(x^k)^{-1}\mathbf{1} \\ 0 \end{bmatrix}$$

where  $H^k = \text{diag}(x^k)^{-2}$

- with  $t = \mu$ , same as Dikin update except for centering term  
 $\text{diag}(x^k)^{-1}\mathbf{1}$

## Stopping criteria

- Dikin iterates always primal feasible:  $Ax^k = b, x^k \succeq 0$
- dual feasibility and zero duality gap only satisfied in the limit
- reasonable stopping criteria are

$$\min(c + A^T \nu^k) \geq -\epsilon_{\text{df}}, \quad |c^T x^k + b^T \nu^k| \leq \epsilon_{\text{gap}}$$

(second term is a *pseudo-gap*; it is only the true gap when  $c + A^T \nu^k \succeq 0$ )

- $\mu(c + A^T \nu^k) = -H^k \Delta x^k = -(\Delta x_1^k/(x_1^k)^2, \dots, \Delta x_n^k/(x_n^k)^2)$
- stopping criteria can be written

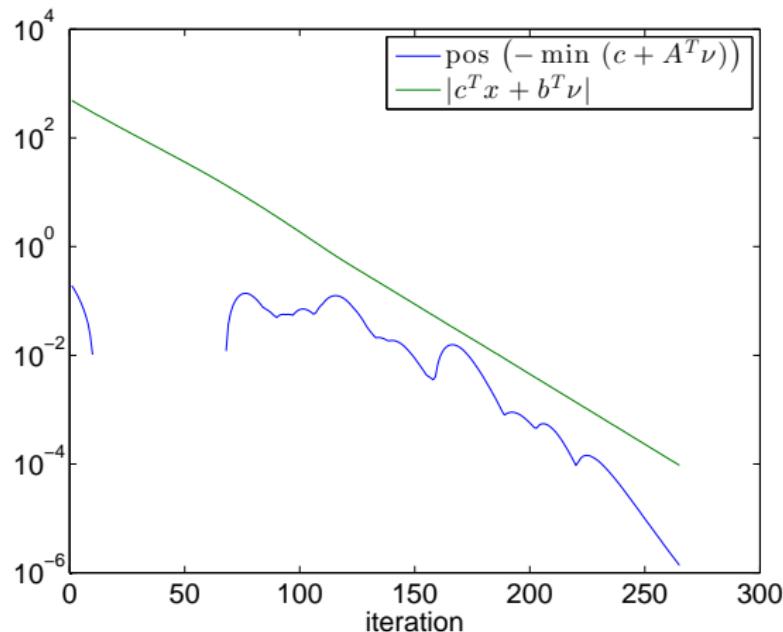
$$\max_i \frac{1}{\mu} \frac{\Delta x_i^k}{(x_i^k)^2} \leq \epsilon_{\text{df}}, \quad \frac{1}{\mu} \left| \sum_i \frac{\Delta x_i^k}{x_i^k} \right| \leq \epsilon_{\text{gap}}$$

## Long step Dikin's method

- long step given by  $x^{k+1} = x^k + \tau \Delta x^k$
- $\tau = 0.95 \max\{t \mid x + t\Delta x \succeq 0\}$
- i.e., step in Dikin direction, 95% of the way to boundary

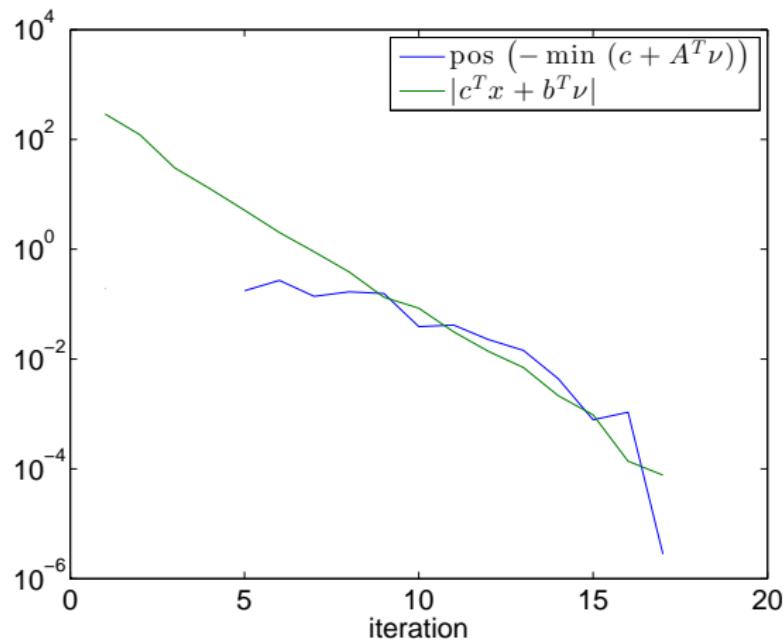
## Short step example

$m = 100, n = 400$



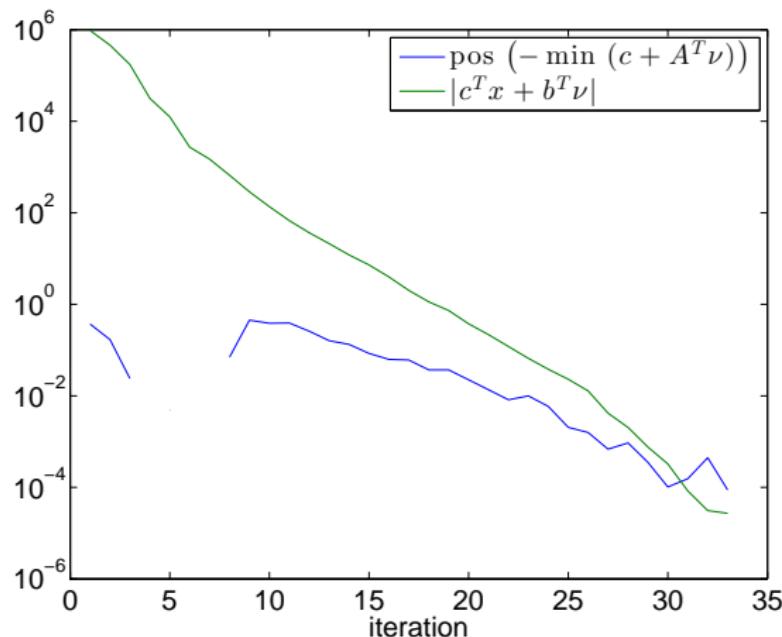
## Long step example

$m = 100, n = 400$



## Larger example

$m = 4000, n = 16000, A$  0.8% dense (500000 nonzeros), long step



# **Outline**

Dikin's original method for LP

Generalized Dikin's method

## Dikin ellipsoid for general constraints

- $\phi$  is self-concordant barrier for set  $\mathcal{C}$
- for  $y \in \text{int } \mathcal{C}$ ,  $(x - y)^T \nabla^2 \phi(y)(x - y) \leq 1 \implies x \in \mathcal{C}$
- for  $\mathcal{C} = \mathbf{R}_+^n$ ,

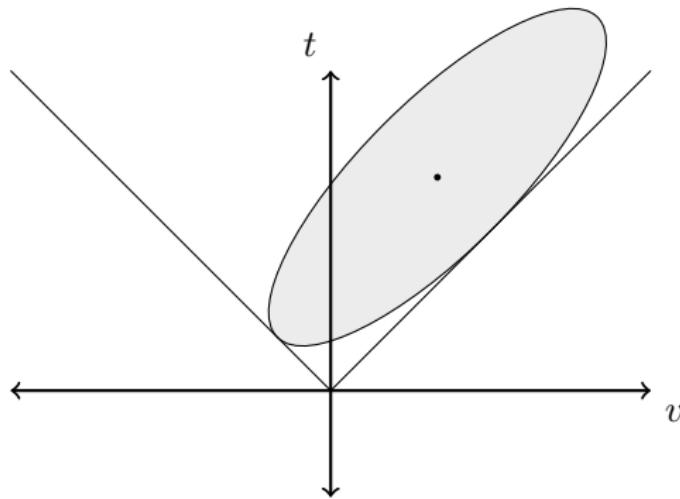
$$\phi(x) = - \sum_{i=1}^n \log(x_i), \quad \nabla^2 \phi(y) = \text{diag}(y)^{-2}$$

- for  $\mathcal{C} = \mathbf{S}_{++}^n$ ,
- $$\phi(X) = -\log \det X, \quad \nabla^2 \phi(Y)(\Delta X) = Y^{-1}(\Delta X)Y^{-1}$$
- for direct product  $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_k$ , use barrier

$$\phi(x) = \sum_{i=1}^k \phi_i(x_i)$$

## SOCP Dikin ellipsoid

- $\mathcal{C} = \{x = (v, t) \in \mathbf{R}^n \times \mathbf{R} \mid \|v\|_2 \leq t\}$
- $\phi(x) = -\log(t^2 - v^T v)$
- $\nabla^2 \phi(x) = \frac{2}{t^2 - v^T v} \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} + \frac{4}{(t^2 - v^T v)^2} \begin{bmatrix} -v \\ t \end{bmatrix} \begin{bmatrix} -v^T & t \end{bmatrix}$



## Dikin's method with general constraints

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b \\ & && x \in \mathcal{C} \end{aligned}$$

- $\phi$  is self-concordant barrier for  $\mathcal{C}$
- $x^{k+1}$  is solution of

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b \\ & && (x - x^k)^T H^k (x - x^k) \leq 1, \end{aligned}$$

where  $H^k = \nabla^2 \phi(x^k)$

- long step update: move 95% towards boundary

## Inequality form LP

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax \preceq b \end{aligned}$$

- $\phi(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$
- $H = \nabla^2 \phi(x) = A^T \mathbf{diag}(b - Ax)^{-2} A$
- Dikin step is

$$x^{k+1} = x^k - \frac{H^{-1}c}{\sqrt{c^T H^{-1} c}}$$

## Inequality form LP cont.

- dual

$$\begin{array}{ll}\text{maximize} & -b^T \lambda \\ \text{subject to} & c + A^T \lambda = 0 \\ & \lambda \succeq 0\end{array}$$

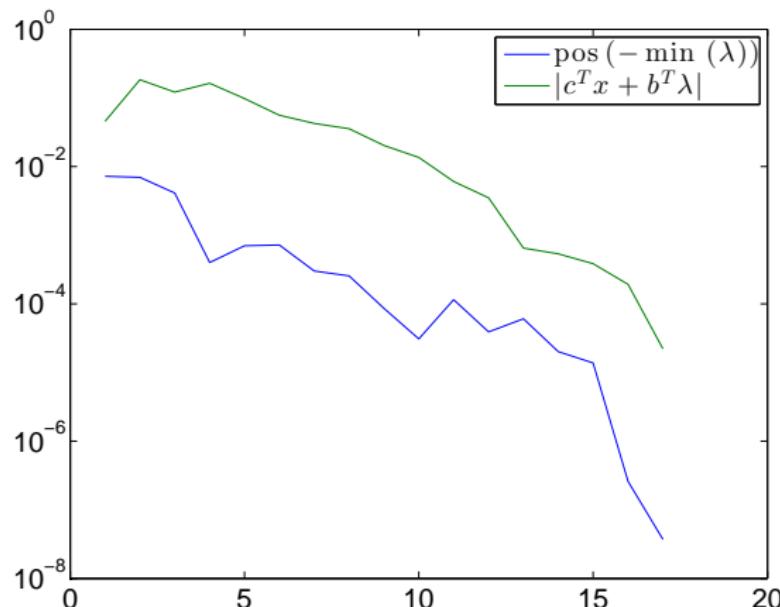
- stopping criteria

$$\lambda = \text{diag}(b - Ax)^2 As \succeq -\epsilon_{\text{df}}, \quad |c^T x + b^T \lambda| \leq \epsilon_{\text{gap}},$$

where  $s = -H^{-1}c$

## Inequality form LP example (long step)

$m = 1500, n = 500$



# SDP

- SDP in inequality form

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && \sum_{i=1}^n x_i A_i \preceq B \end{aligned}$$

$$B, A_i \in \mathbf{S}^m$$

- $\phi(x) = -\log \det(B - \sum_{i=1}^n x_i A_i)$
- Hessian given by

$$H_{ij} = \mathbf{tr}(S^{-1} A_i S^{-1} A_j), \quad S = B - \sum_{i=1}^n x_i A_i$$

- step identical to inequality form LP

$$x^{k+1} = x^k - \frac{H^{-1}c}{\sqrt{c^T H^{-1} c}}$$

## SDP cont.

- dual

$$\begin{aligned} & \text{maximize} && -\mathbf{tr}(BZ) \\ & \text{subject to} && c_i + \mathbf{tr}(A_i Z) = 0, \quad i = 1, \dots, n \\ & && Z \succeq 0 \end{aligned}$$

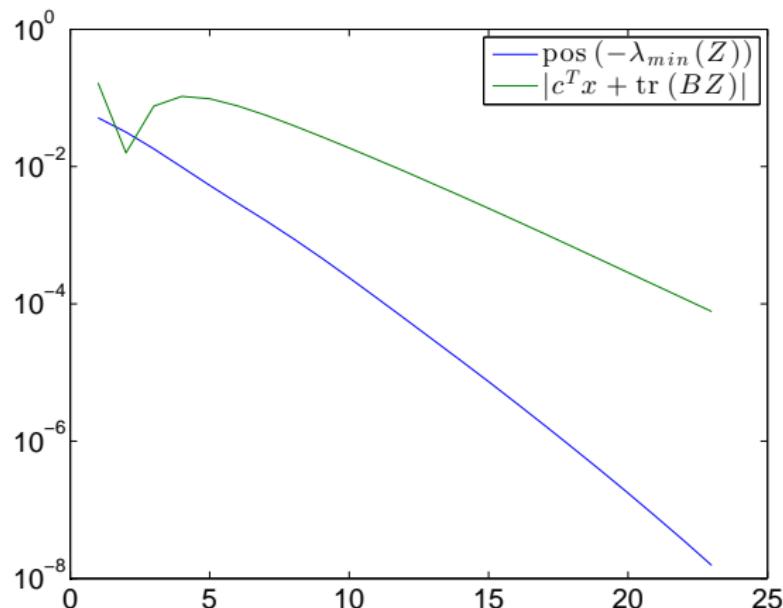
- stopping criteria

$$Z \succeq -\epsilon_{\text{df}} I, \quad |c^T x + \mathbf{tr}(BZ)| \leq \epsilon_{\text{gap}},$$

where  $Z = \sum_{i=1}^n S^{-1} A_j S^{-1} s_j$ ,  $s = -H^{-1}c$

## SDP example (short step)

$m = 100, n = 100$



# Model Predictive Control

- linear convex optimal control
- finite horizon approximation
- model predictive control
- fast MPC implementations
- supply chain management

# Linear time-invariant convex optimal control

$$\begin{aligned} \text{minimize} \quad & J = \sum_{t=0}^{\infty} \ell(x(t), u(t)) \\ \text{subject to} \quad & u(t) \in \mathcal{U}, \quad x(t) \in \mathcal{X}, \quad t = 0, 1, \dots \\ & x(t+1) = Ax(t) + Bu(t), \quad t = 0, 1, \dots \\ & x(0) = z. \end{aligned}$$

- variables: state and input trajectories  $x(0), x(1), \dots \in \mathbf{R}^n$ ,  $u(0), u(1), \dots \in \mathbf{R}^m$
- problem data:
  - dynamics and input matrices  $A \in \mathbf{R}^{n \times n}$ ,  $B \in \mathbf{R}^{n \times m}$
  - convex stage cost function  $\ell : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}$ ,  $\ell(0, 0) = 0$
  - convex state and input constraint sets  $\mathcal{X}, \mathcal{U}$ , with  $0 \in \mathcal{X}$ ,  $0 \in \mathcal{U}$
  - initial state  $z \in \mathcal{X}$

## Greedy control

- use  $u(t) = \operatorname{argmin}_w \{\ell(x(t), w) \mid w \in \mathcal{U}, Ax(t) + Bw \in \mathcal{X}\}$
- minimizes current stage cost only, ignoring effect of  $u(t)$  on future, except for  $x(t+1) \in \mathcal{X}$
- typically works very poorly; can lead to  $J = \infty$  (when optimal  $u$  gives finite  $J$ )

## ‘Solution’ via dynamic programming

- (Bellman) **value function**  $V(z)$  is optimal value of control problem as a function of initial state  $z$
- can show  $V$  is convex
- $V$  satisfies Bellman or dynamic programming equation

$$V(z) = \inf \{ \ell(z, w) + V(Az + Bw) \mid w \in \mathcal{U}, Az + Bw \in \mathcal{X} \}$$

- optimal  $u$  given by

$$u^*(t) = \operatorname{argmin}_{w \in \mathcal{U}, Ax(t) + Bw \in \mathcal{X}} (\ell(x(t), w) + V(Ax(t) + Bw))$$

- interpretation: term  $V(Ax(t) + Bw)$  properly accounts for future costs due to current action  $w$
- optimal input has ‘state feedback form’  $u^*(t) = \phi(x(t))$

# Linear quadratic regulator

- special case of linear convex optimal control with
  - $\mathcal{U} = \mathbf{R}^m$ ,  $\mathcal{X} = \mathbf{R}^n$
  - $\ell(x(t), u(t)) = x(t)^T Q x(t) + u(t)^T R u(t)$ ,  $Q \succeq 0$ ,  $R \succ 0$
- can be solved using DP
  - value function is quadratic:  $V(z) = z^T P z$
  - $P$  can be found by solving an algebraic Riccati equation (ARE)

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A$$

- optimal policy is linear state feedback:  $u^*(t) = Kx(t)$ , with  
 $K = -(R + B^T P B)^{-1} B^T P A$

## Finite horizon approximation

- use finite horizon  $T$ , impose terminal constraint  $x(T) = 0$ :

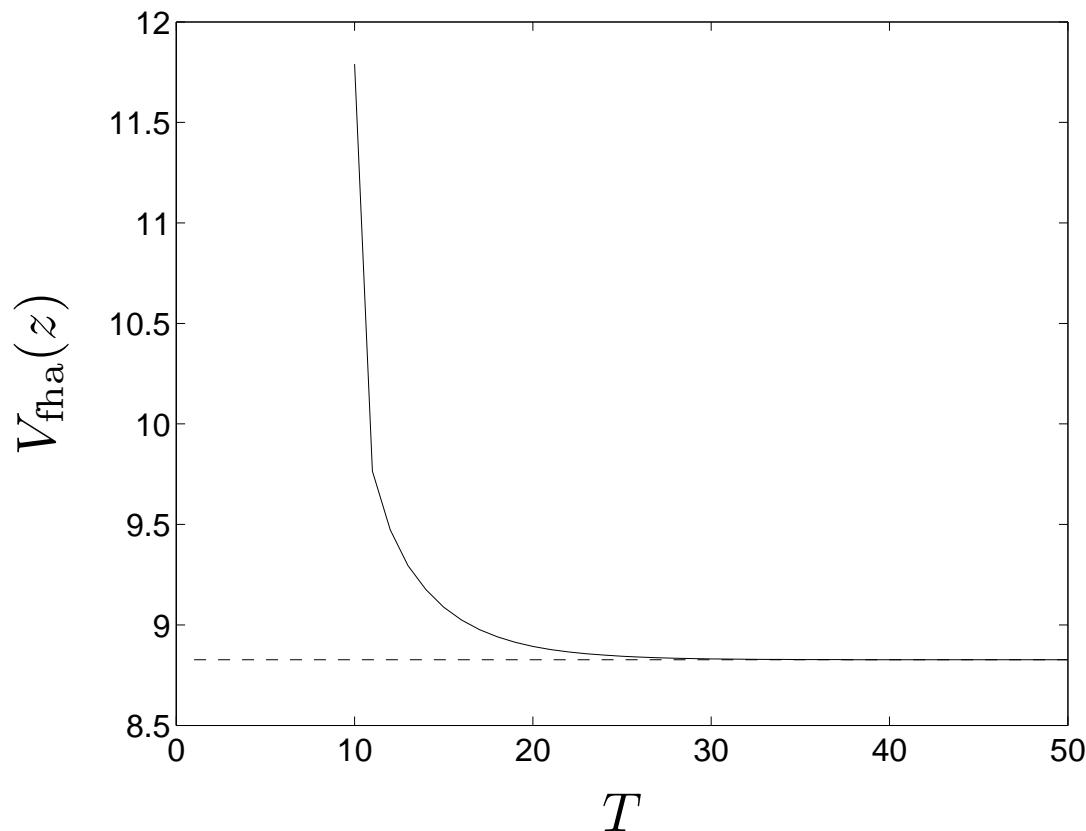
$$\begin{aligned} \text{minimize} \quad & \sum_{\tau=0}^{T-1} \ell(x(\tau), u(\tau)) \\ \text{subject to} \quad & u(\tau) \in \mathcal{U}, \quad x(\tau) \in \mathcal{X} \quad \tau = 0, \dots, T \\ & x(\tau+1) = Ax(\tau) + Bu(\tau), \quad \tau = 0, \dots, T-1 \\ & x(0) = z, \quad x(T) = 0. \end{aligned}$$

- apply the input sequence  $u(0), \dots, u(T-1), 0, 0, \dots$
- a finite dimensional convex problem
- gives suboptimal input for original optimal control problem

## Example

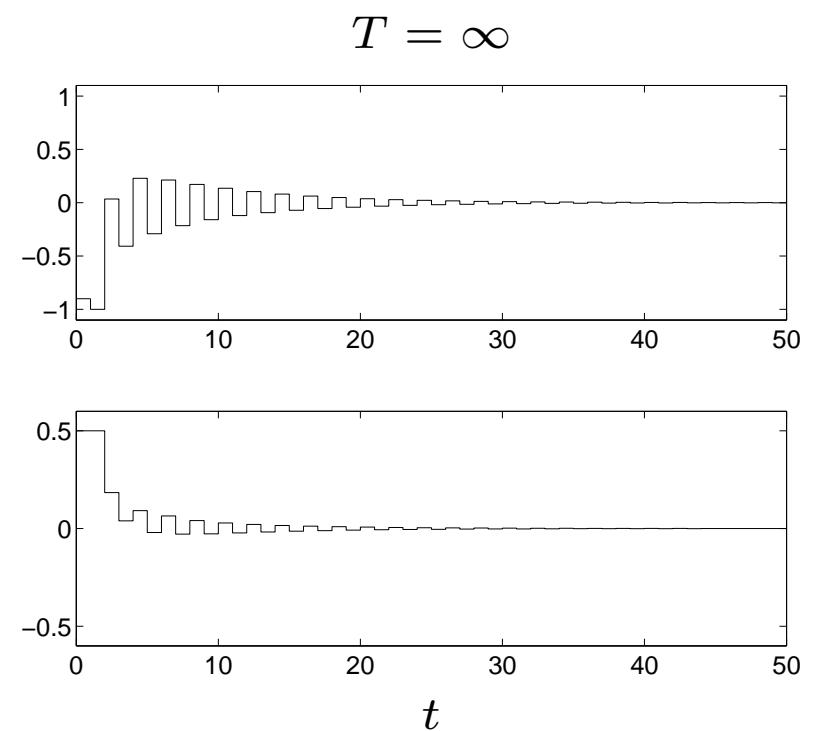
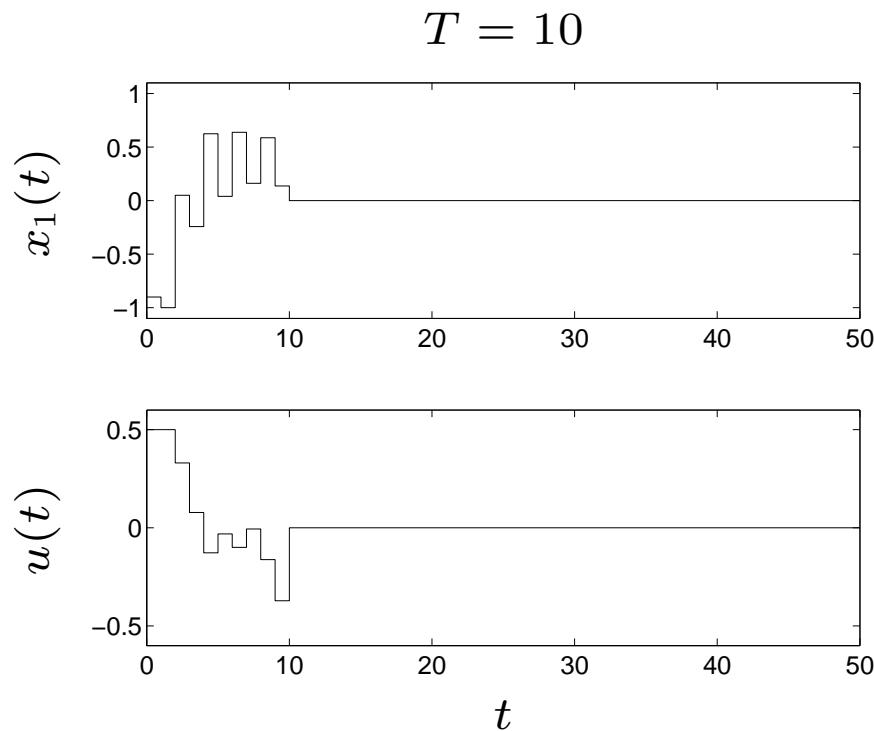
- system with  $n = 3$  states,  $m = 2$  inputs;  $A, B$  chosen randomly
- quadratic stage cost:  $\ell(v, w) = \|v\|^2 + \|w\|^2$
- $\mathcal{X} = \{v \mid \|v\|_\infty \leq 1\}$ ,  $\mathcal{U} = \{w \mid \|w\|_\infty \leq 0.5\}$
- initial point:  $z = (0.9, -0.9, 0.9)$
- optimal cost is  $V(z) = 8.83$

## Cost versus horizon



dashed line shows  $V(z)$ ; finite horizon approximation infeasible for  $T \leq 9$

# Trajectories



## Model predictive control (MPC)

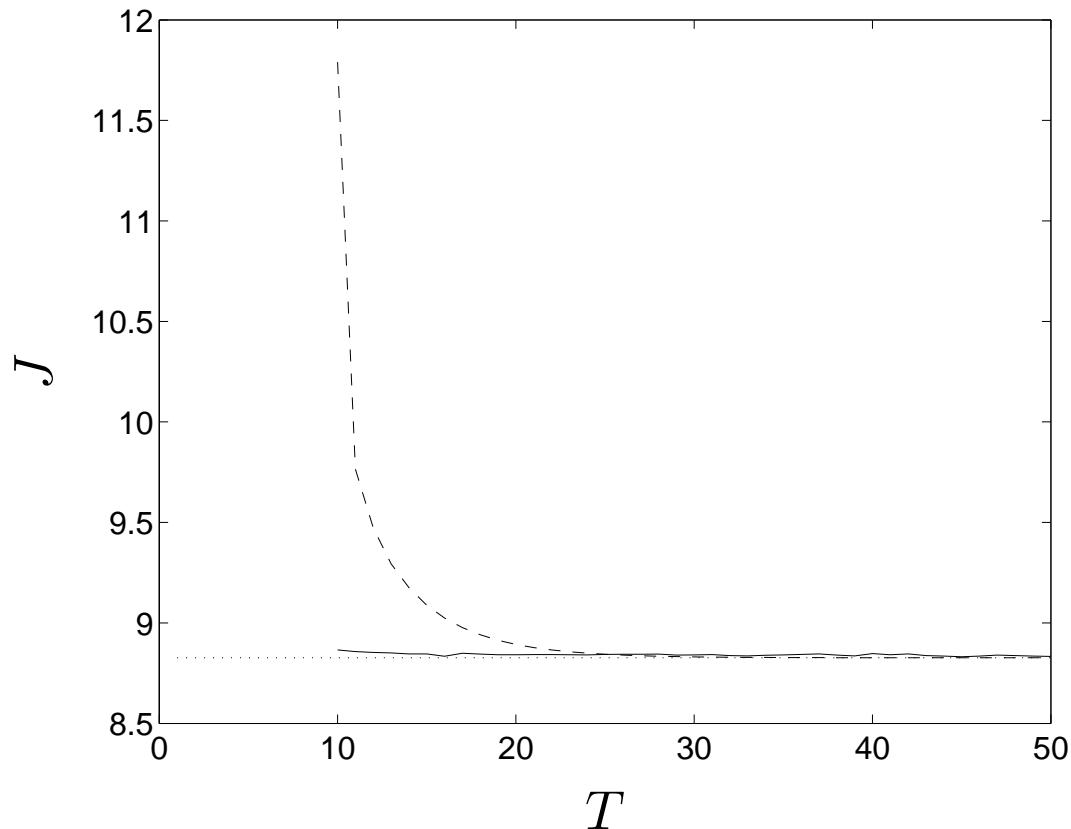
- at each time  $t$  solve the (planning) problem

$$\begin{array}{ll}\text{minimize} & \sum_{\tau=t}^{t+T} \ell(x(\tau), u(\tau)) \\ \text{subject to} & u(\tau) \in \mathcal{U}, \quad x(\tau) \in \mathcal{X}, \quad \tau = t, \dots, t + T \\ & x(\tau + 1) = Ax(\tau) + Bu(\tau), \quad \tau = t, \dots, t + T - 1 \\ & x(t + T) = 0\end{array}$$

with variables  $x(t + 1), \dots, x(t + T)$ ,  $u(t), \dots, u(t + T - 1)$   
and data  $x(t)$ ,  $A$ ,  $B$ ,  $\ell$ ,  $\mathcal{X}$ ,  $\mathcal{U}$

- call solution  $\tilde{x}(t + 1), \dots, \tilde{x}(t + T)$ ,  $\tilde{u}(t), \dots, \tilde{u}(t + T - 1)$
- we interpret these as *plan of action* for next  $T$  steps
- we take  $u(t) = \tilde{u}(t)$
- this gives a complicated state feedback control  $u(t) = \phi_{\text{mpc}}(x(t))$

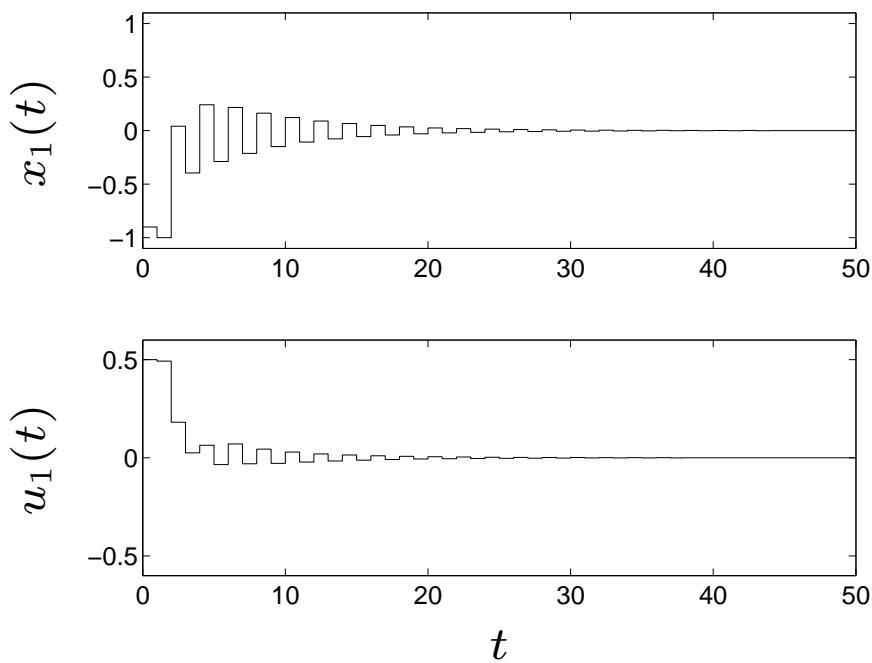
## MPC performance versus horizon



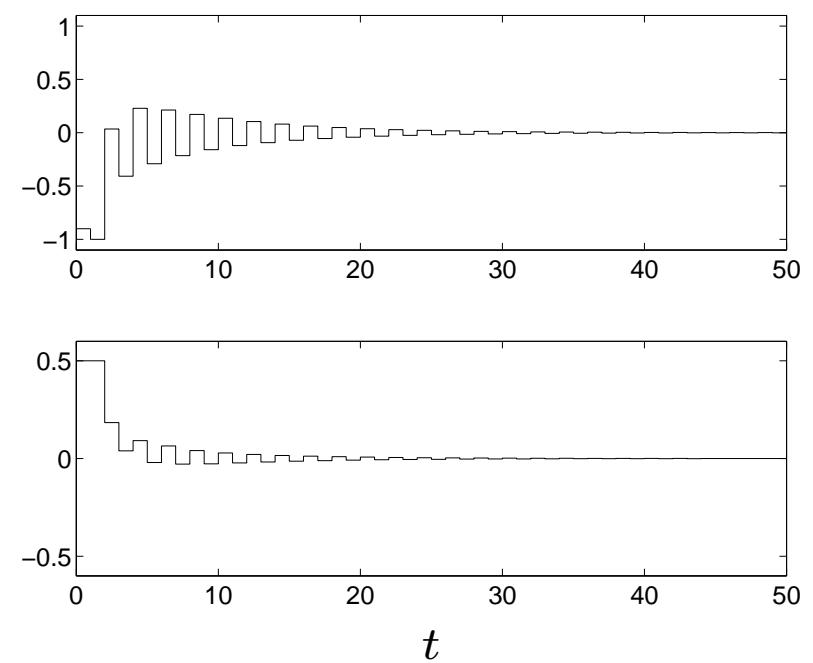
solid: MPC, dashed: finite horizon approximation, dotted:  $V(z)$

# MPC trajectories

MPC,  $T = 10$



$T = \infty$



# MPC

- goes by many other names, *e.g.*, dynamic matrix control, receding horizon control, dynamic linear programming, rolling horizon planning
- widely used in (some) industries, typically for systems with slow dynamics (chemical process plants, supply chain)
- MPC typically works very well in practice, even with short  $T$
- under some conditions, can give performance guarantees for MPC

## Variations on MPC

- add final state cost  $\hat{V}(x(t + T))$  instead of insisting on  $x(t + T) = 0$ 
  - if  $\hat{V} = V$ , MPC gives optimal input
- convert hard constraints to violation penalties
  - avoids problem of planning problem infeasibility
- solve MPC problem every  $K$  steps,  $K > 1$ 
  - use current plan for  $K$  steps; then re-plan

## Explicit MPC

- MPC with  $\ell$  quadratic,  $\mathcal{X}$  and  $\mathcal{U}$  polyhedral
- can show  $\phi_{\text{mpc}}$  is piecewise affine

$$\phi_{\text{mpc}}(z) = K_j z + g_j, \quad z \in \mathcal{R}_j$$

$\mathcal{R}_1, \dots, \mathcal{R}_N$  is polyhedral partition of  $\mathcal{X}$

(solution of *any* QP is PWA in righthand sides of constraints)

- $\phi_{\text{mpc}}$  (*i.e.*,  $K_j$ ,  $g_j$ ,  $\mathcal{R}_j$ ) can be computed explicitly, off-line
- on-line controller simply evaluates  $\phi_{\text{mpc}}(x(t))$   
(effort is dominated by determining which region  $x(t)$  lies in)

- can work well for (very) small  $n$ ,  $m$ , and  $T$
- number of regions  $N$  grows exponentially in  $n$ ,  $m$ ,  $T$ 
  - needs lots of storage
  - evaluating  $\phi_{\text{mpc}}$  can be slow
- simplification methods can be used to reduce the number of regions, while still getting good control

## MPC problem structure

- MPC problem is highly structured (see *Convex Optimization*, §10.3.4)
  - Hessian is block diagonal
  - equality constraint matrix is block banded
- use block elimination to compute Newton step
  - Schur complement is block tridiagonal with  $n \times n$  blocks
- can solve in order  $T(n + m)^3$  flops using an interior point method

## Fast MPC

- can obtain further speedup by solving planning problem approximately
  - fix barrier parameter; use warm-start
  - (sharply) limit the total number of Newton steps
- results for simple C implementation

problem size			QP size		run time (ms)	
$n$	$m$	$T$	vars	constr	fast mpc	SDPT3
4	2	10	50	160	0.3	150
10	3	30	360	1080	4.0	1400
16	4	30	570	1680	7.7	2600
30	8	30	1110	3180	23.4	3400

- can run MPC at **kilohertz** rates

## Supply chain management

- $n$  nodes (warehouses/buffers)
- $m$  unidirectional links between nodes, external world
- $x_i(t)$  is amount of commodity at node  $i$ , in period  $t$
- $u_j(t)$  is amount of commodity transported along link  $j$
- incoming and outgoing node incidence matrices:

$$A_{ij}^{\text{in}(\text{out})} = \begin{cases} 1 & \text{link } j \text{ enters (exits) node } i \\ 0 & \text{otherwise} \end{cases}$$

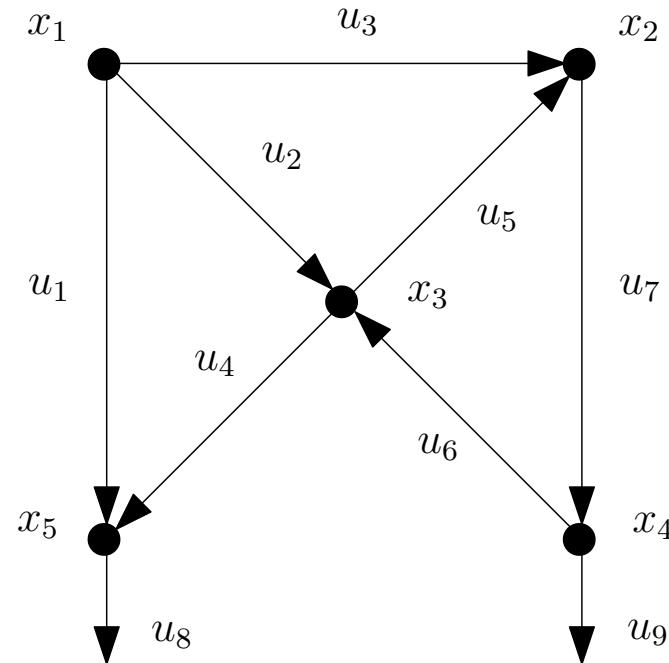
- dynamics:  $x(t+1) = x(t) + A^{\text{in}}u(t) - A^{\text{out}}u(t)$

## Constraints and objective

- buffer limits:  $0 \leq x_i(t) \leq x_{\max}$   
(could allow  $x_i(t) < 0$ , to represent back-order)
- link capacities:  $0 \leq u_i(t) \leq u_{\max}$
- $A^{\text{out}}u(t) \preceq x(t)$  (can't ship out what's not on hand)
- shipping/transportation cost:  $S(u(t))$   
(can also include sales revenue or manufacturing cost)
- warehousing/storage cost:  $W(x(t))$
- objective:  $\sum_{t=0}^{\infty} (S(u(t)) + W(x(t)))$

# Example

- $n = 5$  nodes,  $m = 9$  links (links 8, 9 are external links)



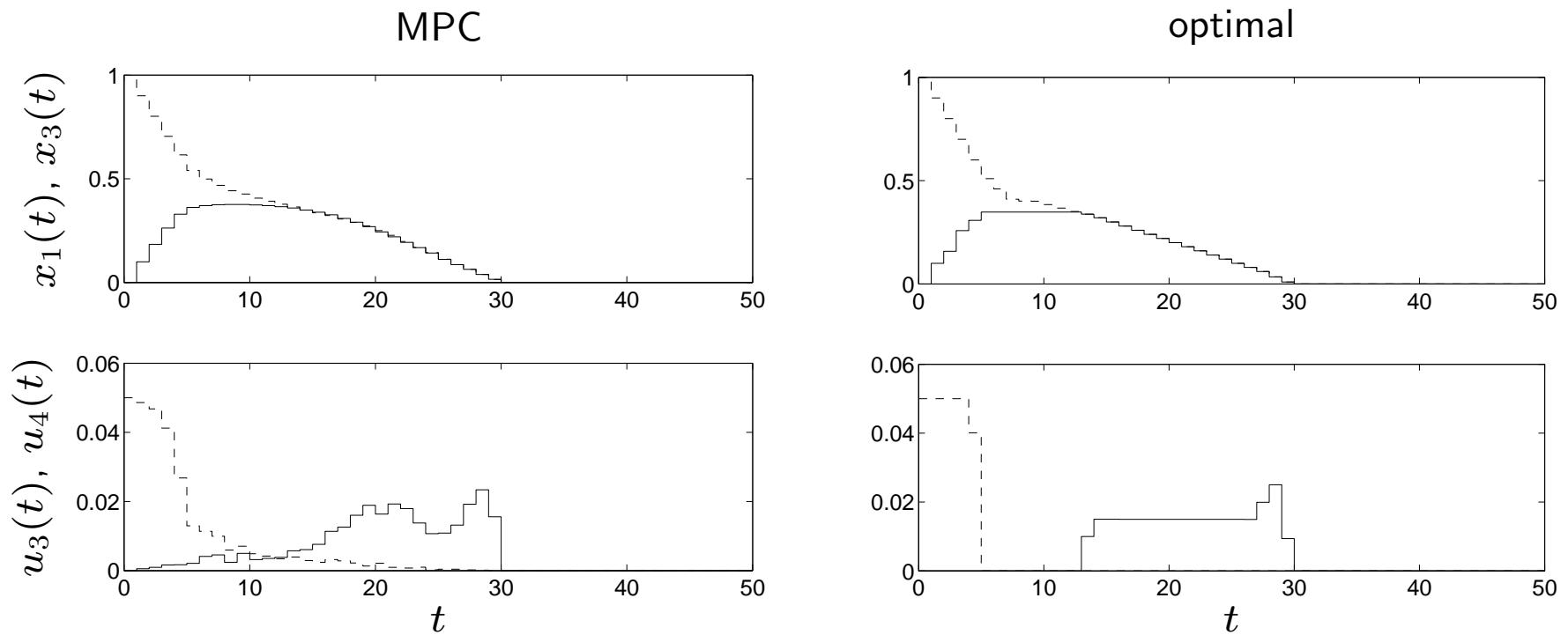
## Example

- $x_{\max} = 1, u_{\max} = 0.05$
- storage cost:  $W(x(t)) = \sum_{i=0}^n (x_i(t) + x_i(t)^2)$
- shipping cost:

$$S(u(t)) = \underbrace{u_1(t) + \cdots + u_7(t)}_{\text{transportation cost}} - \underbrace{(u_8(t) + u_9(t))}_{\text{revenue}}$$

- initial stock:  $x(0) = (1, 0, 0, 1, 1)$
- we run MPC with  $T = 5$ , final cost  $\hat{V}(x(t+T)) = 10(\mathbf{1}^T x(t+T))$
- optimal cost:  $V(z) = 68.2$ ; MPC cost 69.5

## MPC and optimal trajectories



solid:  $x_3(t)$ ,  $u_4(t)$ ; dashed:  $x_1(t)$ ,  $u_3(t)$

## Variations on optimal control problem

- time varying costs, dynamics, constraints
  - discounted cost
  - convergence to nonzero desired state
  - tracking time-varying desired trajectory
- coupled state and input constraints, *e.g.*,  $(x(t), u(t)) \in \mathcal{P}$  (as in supply chain management)
- slew rate constraints, *e.g.*,  $\|u(t+1) - u(t)\|_\infty \leq \Delta u_{\max}$
- stochastic control: future costs, dynamics, disturbances not known (next lecture)

# Stochastic Model Predictive Control

- stochastic finite horizon control
- stochastic dynamic programming
- certainty equivalent model predictive control

## Causal state-feedback control

- linear dynamical system, over finite time horizon:

$$x_{t+1} = Ax_t + Bu_t + w_t, \quad t = 0, \dots, T-1$$

- $x_t \in \mathbf{R}^n$  is state,  $u_t \in \mathbf{R}^m$  is the input at time  $t$
- $w_t$  is the process noise (or exogeneous input) at time  $t$

- $X_t = (x_0, \dots, x_t)$  is the state history up to time  $t$
- causal state-feedback control:

$$u_t = \phi_t(X_t) = \psi_t(x_0, w_0, \dots, w_{t-1}), \quad t = 0, \dots, T-1$$

- $\phi_t : \mathbf{R}^{(t+1)n} \rightarrow \mathbf{R}^m$  called the control **policy** at time  $t$

## Stochastic finite horizon control

- $(x_0, w_0, \dots, w_{T-1})$  is a random variable
- objective:  $J = \mathbf{E} \left( \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \right)$ 
  - convex stage cost functions  $\ell_t : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}$ ,  $t = 0, \dots, T - 1$
  - convex terminal cost function  $\ell_T : \mathbf{R}^n \rightarrow \mathbf{R}$
- $J$  depends on control policies  $\phi_0, \dots, \phi_{T-1}$
- constraints:  $u_t \in \mathcal{U}_t$ ,  $t = 0, \dots, T - 1$ 
  - convex input constraint sets  $\mathcal{U}_0, \dots, \mathcal{U}_{T-1}$
- **stochastic control problem:** choose control policies  $\phi_0, \dots, \phi_{T-1}$  to minimize  $J$ , subject to constraints

## Stochastic finite horizon control

- an infinite dimensional problem: variables are *functions*  $\phi_0, \dots, \phi_{T-1}$ 
  - can restrict policies to finite dimensional subspace, *e.g.*,  $\phi_t$  all affine
- key idea: we have **recourse** (a.k.a. feedback, closed-loop control)
  - we can change  $u_t$  based on the observed state history  $x_0, \dots, x_t$
  - cf standard ('open loop') optimal control problem, where we commit to  $u_0, \dots, u_{T-1}$  ahead of time
- in general case, need to evaluate  $J$  (for given control policies) via Monte Carlo simulation

## ‘Solution’ via dynamic programming

- let  $V_t(X_t)$  be optimal value of objective, from  $t$  on, starting from initial state history  $X_t$
- $V_T(X_T) = \ell_T(x_T); J^* = \mathbf{E} V_0(x_0)$
- $V_t$  can be found by backward recursion: for  $t = T - 1, \dots, 0$

$$V_t(X_t) = \inf_{v \in \mathcal{U}} \{\ell_t(x_t, v) + \mathbf{E}(V_{t+1}((X_t, Ax_t + Bv + w_t))|X_t)\}$$

- $V_t, t = 0, \dots, T$  are convex functions
- optimal policy is causal state feedback

$$\phi_t^*(X_t) = \operatorname{argmin}_{v \in \mathcal{U}} \{\ell_t(x_t, v) + \mathbf{E}(V_{t+1}((X_t, Ax_t + Bv + w_t))|X_t)\}$$

## Independent process noise

- assume  $x_0, w_0, \dots, w_{T-1}$  are independent
- $V_t$  depends only on the current state  $x_t$  (and not the state history  $X_t$ )
- Bellman equations:  $V_T(x_T) = \ell_T(x_T)$ ; for  $t = T-1, \dots, 0$ ,

$$V_t(x_t) = \inf_{v \in \mathcal{U}} \{\ell_t(x_t, v) + \mathbf{E} V_{t+1}(Ax_t + Bv + w_t)\}$$

- optimal policy is a function of current state  $x_t$

$$\phi^*(x_t) = \operatorname{argmin}_{v \in \mathcal{U}} \{\ell_t(x_t, v) + \mathbf{E} V_{t+1}(Ax_t + Bv + w_t)\}$$

# Linear quadratic stochastic control

- special case of linear stochastic control

- $\mathcal{U}_t = \mathbf{R}^m$

- $x_0, w_0, \dots, w_{T-1}$  are independent, with

$$\mathbf{E} x_0 = 0, \quad \mathbf{E} w_t = 0, \quad \mathbf{E} x_0 x_0^T = \Sigma, \quad \mathbf{E} w_t w_t^T = W_t$$

- $\ell_t(x_t, u_t) = x_t^T Q_t x_t + u_t^T R_t u_t$ , with  $Q_t \succeq 0$ ,  $R_t \succ 0$

- $\ell_T(x_T) = x_T^T Q_T x_T$ , with  $Q_T \succeq 0$

- can show value functions are quadratic, *i.e.*,

$$V_t(x_t) = x_t^T P_t x_t + q_t, \quad t = 0, \dots, T$$

- Bellman recursion:  $P_T = Q_T$ ,  $q_T = 0$ ; for  $t = T - 1, \dots, 0$ ,

$$\begin{aligned} V_t(z) &= \inf_v \{ z^T Q_t z + v^T R_t v \\ &\quad + \mathbf{E}((Az + Bv + w_t)^T P_{t+1}(Az + Bv + w_t) + q_{t+1}) \} \end{aligned}$$

- works out to

$$\begin{aligned} P_t &= A^T P_{t+1} A - A^T P_{t+1} B (B^T P_{t+1} B + R_t)^{-1} B^T P_{t+1} A + Q_t \\ q_t &= q_{t+1} + \mathbf{Tr}(W_t P_{t+1}) \end{aligned}$$

- optimal policy is linear state feedback:  $\phi_t^\star(x_t) = K_t x_t$ ,

$$K_t = -(B^T P_{t+1} B + R_t)^{-1} B^T P_{t+1} A$$

(which, strangely, does not depend on  $\Sigma, W_0, \dots, W_{T-1}$ )

- optimal cost

$$\begin{aligned} J^\star &= \mathbf{E} V_0(x_0) \\ &= \mathbf{Tr}(\Sigma P_0) + q_0 \\ &= \mathbf{Tr}(\Sigma P_0) + \sum_{t=0}^{T-1} \mathbf{Tr}(W_t P_{t+1}) \end{aligned}$$

## Certainty equivalent model predictive control

- at every time  $t$  we solve the certainty equivalent problem

$$\begin{aligned} \text{minimize} \quad & \sum_{\tau=t}^{T-1} \ell_t(x_\tau, u_\tau) + \ell_T(x_T) \\ \text{subject to} \quad & u_\tau \in \mathcal{U}_\tau, \quad \tau = t, \dots, T-1 \\ & x_{\tau+1} = Ax_\tau + Bu_\tau + \hat{w}_{\tau|t}, \quad \tau = t, \dots, T-1 \end{aligned}$$

with variables  $x_{t+1}, \dots, x_T, u_t, \dots, u_{T-1}$  and data  $x_t, \hat{w}_{t|t}, \dots, \hat{w}_{T-1|t}$

- $\hat{w}_{t|t}, \dots, \hat{w}_{T-1|t}$  are predicted values of  $w_t, \dots, w_{T-1}$  based on  $X_t$  (e.g., conditional expectations)
- call solution  $\tilde{x}_{t+1}, \dots, \tilde{x}_T, \tilde{u}_t, \dots, \tilde{u}_{T-1}$
- we take  $\phi^{\text{mpc}}(X_t) = \tilde{u}_t$ 
  - $\phi^{\text{mpc}}$  is a function of  $X_t$  since  $\hat{w}_{t|t}, \dots, \hat{w}_{T-1|t}$  are functions of  $X_t$

## **Certainty equivalent model predictive control**

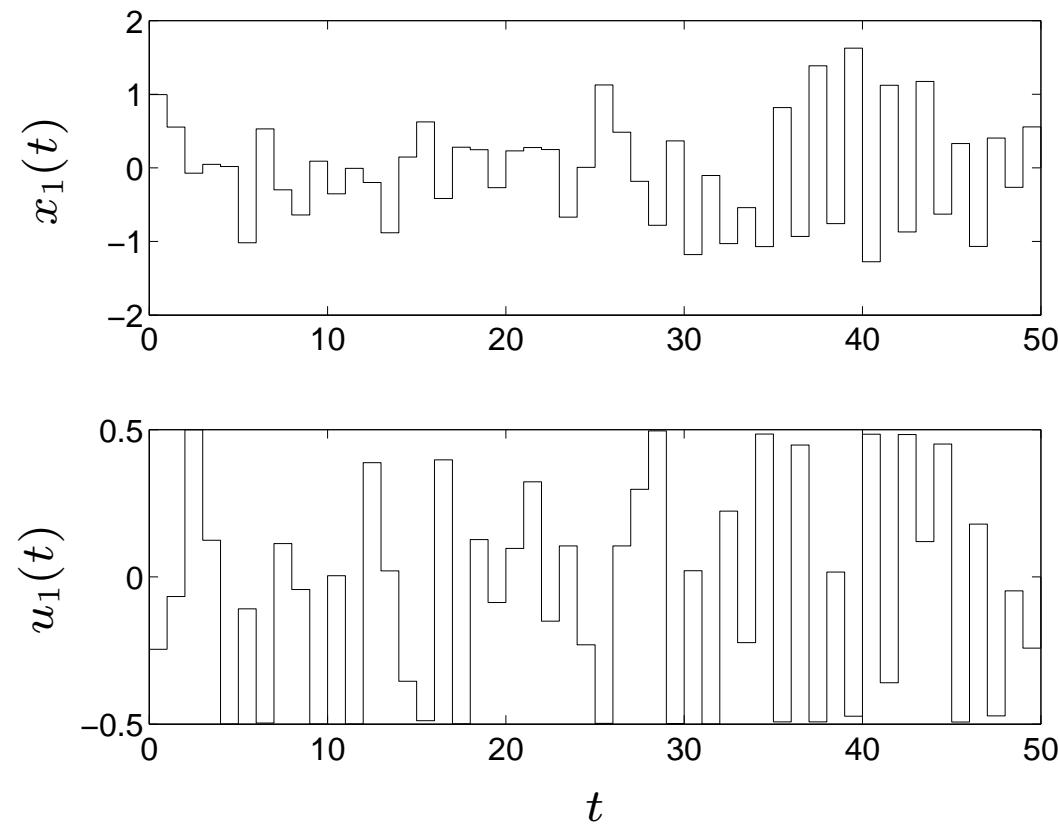
- widely used, *e.g.*, in ‘revenue management’
- based on (bad) approximations:
  - future values of disturbance are exactly as predicted; there is no future uncertainty
  - in future, no recourse is available
- yet, often works very well

## Example

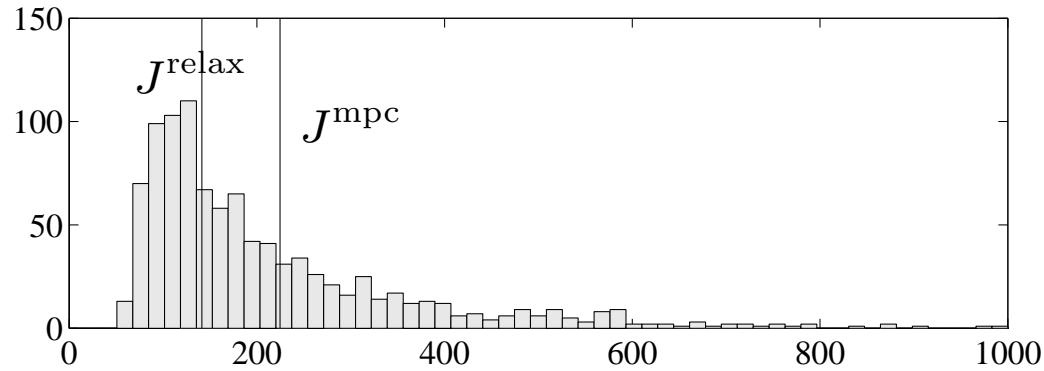
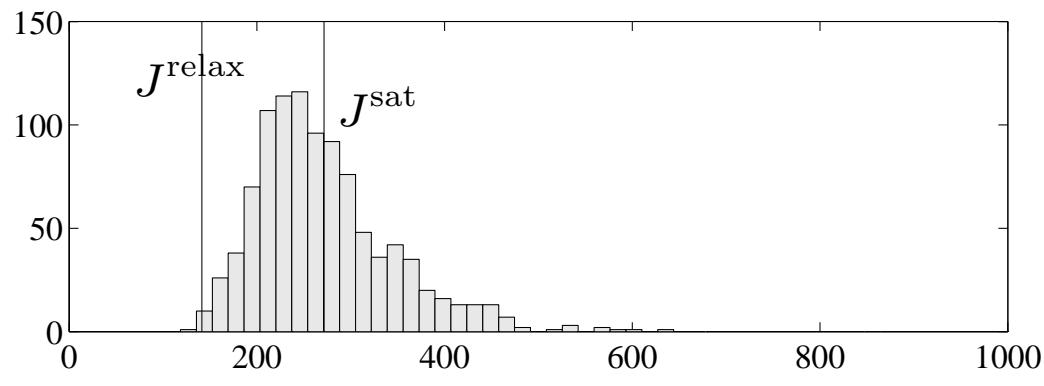
- system with  $n = 3$  states,  $m = 2$  inputs; horizon  $T = 50$
- $A, B$  chosen randomly
- quadratic stage cost:  $\ell_t(x, u) = \|x\|_2^2 + \|u\|_2^2$
- quadratic final cost:  $\ell_T(x) = \|x\|_2^2$
- constraint set:  $\mathcal{U} = \{u \mid \|u\|_\infty \leq 0.5\}$
- $x_0, w_0, \dots, w_{T-1}$  iid  $\mathcal{N}(0, 0.25I)$

## Stochastic MPC: Sample trajectory

sample trace of  $x_1$  and  $u_1$



# Cost histogram



## Simple lower bound for quadratic stochastic control

- $x_0, w_0, \dots, w_{T-1}$  independent
- quadratic stage and final cost
- relaxation:
  - ignore  $\mathcal{U}_t$ ; yields linear quadratic stochastic control problem
  - solve relaxed problem exactly; optimal cost is  $J^{\text{relax}}$
- $J^* \geq J^{\text{relax}}$
- for our numerical example,
  - $J^{\text{mpc}} = 224.7$  (via Monte Carlo)
  - $J^{\text{sat}} = 271.5$  (linear quadratic stochastic control with saturation)
  - $J^{\text{relax}} = 141.3$

# Sums of Squares

Sanjay Lall  
Stanford University

EE364b  
April 19, 2011

# polynomial programming

A familiar problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \quad \text{for all } i = 1, \dots, m \\ & h_i(x) = 0 \quad \text{for all } i = 1, \dots, p \end{array}$$

in this section, objective, inequality and equality constraint functions are all *polynomials*

## polynomial nonnegativity

does there exist  $x \in \mathbb{R}^n$  such that  $f(x) < 0$

- if not,  $f$  is called *positive semidefinite* or *PSD*

$f(x) \geq 0$  for all  $x \in \mathbb{R}^n$

- the problem is *NP-hard*, but decidable

## certificates

does there exist  $x \in \mathbb{R}^n$  such that  $f(x) < 0$

- answer yes is easy to verify; exhibit  $x$  such that  $f(x) < 0$
- answer no is hard; we need a *certificate* or a *witness*  
i.e, a proof that there is no feasible point

## Sum of Squares Decomposition

$f$  is nonnegative if there are polynomials  $g_1, \dots, g_s$  such that

$$f = \sum_{i=1}^s g_i^2$$

a checkable certificate, called a *sum-of-squares (SOS)* decomposition

- how do we find the  $g_i$
- when does such a certificate exist?

## example

we can write any polynomial as a *quadratic function of monomials*

$$\begin{aligned}
 f &= 4x^4 + 4x^3y - 7x^2y^2 - 2xy^3 + 10y^4 \\
 &= \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix}^T \begin{bmatrix} 4 & 2 & -\lambda \\ 2 & -7 + 2\lambda & -1 \\ -\lambda & -1 & 10 \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix} \\
 &= z^T Q(\lambda) z
 \end{aligned}$$

- above equation holds for all  $\lambda \in \mathbb{R}$
- if for some  $\lambda$  we have  $Q(\lambda) \succeq 0$ , then we can factorize  $Q(\lambda)$

## example, continued

e.g., with  $\lambda = 6$ , we have

$$Q(\lambda) = \begin{bmatrix} 4 & 2 & -6 \\ 2 & 5 & -1 \\ -6 & -1 & 10 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 1 \\ 2 & 1 & -3 \end{bmatrix}$$

so we have an SOS decomposition

$$\begin{aligned} f &= \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix}^T \begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 1 \\ 2 & 1 & -3 \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix} \\ &= \left\| \begin{bmatrix} 2xy + y^2 \\ 2x^2 + xy - 3y^2 \end{bmatrix} \right\|^2 \\ &= (2xy + y^2)^2 + (2x^2 + xy - 3y^2)^2 \end{aligned}$$

## sum of squares and semidefinite programming

suppose  $f \in \mathbb{R}[x_1, \dots, x_n]$ , of degree  $2d$

let  $z$  be a vector of all monomials of degree less than or equal to  $d$

$f$  is SOS if and only if there exists  $Q$  such that

$$\boxed{\begin{aligned} Q &\succeq 0 \\ f &= z^T Q z \end{aligned}}$$

- this is an SDP in standard primal form
- the number of components of  $z$  is  $\binom{n+d}{d}$
- comparing terms gives affine constraints on the elements of  $Q$

## sum of squares and semidefinite programming

if  $Q$  is a feasible point of the SDP, then to construct the SOS representation

factorize  $Q = VV^T$ , and write  $V = [v_1 \dots v_r]$ , so that

$$\begin{aligned} f &= z^T VV^T z \\ &= \|V^T z\|^2 \\ &= \sum_{i=1}^r (v_i^T z)^2 \end{aligned}$$

- one can factorize using e.g., Cholesky or eigenvalue decomposition
- the number of squares  $r$  equals the rank of  $Q$

## example

$$f = 2x^4 + 2x^3y - x^2y^2 + 5y^4$$

$$= \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix}$$

$$= q_{11}x^4 + 2q_{12}x^3y + (q_{22} + 2q_{13})x^2y^2 + 2q_{23}xy^3 + q_{33}y^4$$

so  $f$  is SOS if and only if there exists  $Q$  satisfying the SDP

$$Q \succeq 0 \quad q_{11} = 2 \quad 2q_{12} = 2$$

$$2q_{12} + q_{22} = -1 \quad 2q_{23} = 0$$

$$q_{33} = 5$$

## convexity

the sets of PSD and SOS polynomials are a *convex cones*; i.e.,

$$f, g \text{ PSD} \implies \lambda f + \mu g \text{ is PSD for all } \lambda, \mu \geq 0$$

let  $P_{n,d}$  be the set of PSD polynomials of degree  $\leq d$

let  $\Sigma_{n,d}$  be the set of SOS polynomials of degree  $\leq d$

- both  $P_{n,d}$  and  $\Sigma_{n,d}$  are *convex cones* in  $\mathbb{R}^N$  where  $N = \binom{n+d}{d}$
- we know  $\Sigma_{n,d} \subset P_{n,d}$ , and testing if  $f \in P_{n,d}$  is NP-hard
- but testing if  $f \in \Sigma_{n,d}$  is an SDP (but a large one)

## polynomials in one variable

if  $f \in \mathbb{R}[x]$ , then  $f$  is SOS if and only if  $f$  is PSD

### example

all real roots must have even multiplicity, and highest coeff. is positive

$$\begin{aligned} f &= x^6 - 10x^5 + 51x^4 - 166x^3 + 342x^2 - 400x + 200 \\ &= (x - 2)^2(x - (2 + i))(x - (2 - i))(x - (1 + 3i))(x - (1 - 3i)) \end{aligned}$$

now reorder complex conjugate roots

$$\begin{aligned} &= (x - 2)^2(x - (2 + i))(x - (1 + 3i))(x - (2 - i))(x - (1 - 3i)) \\ &= (x - 2)^2((x^2 - 3x - 1) - i(4x - 7))((x^2 - 3x - 1) + i(4x - 7)) \\ &= (x - 2)^2((x^2 - 3x - 1)^2 + (4x - 7)^2) \end{aligned}$$

so every PSD scalar polynomial is the sum of *one or two* squares

## quadratic polynomials

a quadratic polynomial in  $n$  variables is PSD if and only if it is SOS

because it is PSD if and only if

$$f = x^T Q x$$

where  $Q \geq 0$

and it is SOS if and only if

$$\begin{aligned} f &= \sum_i (v_i^T x)^2 \\ &= x^T \left( \sum_i v_i v_i^T \right) x \end{aligned}$$

## some background

In 1888, Hilbert showed that PSD=SOS if and only if

- $d = 2$ , i.e., quadratic polynomials
- $n = 1$ , i.e., univariate polynomials
- $d = 4, n = 2$ , i.e., quartic polynomials in two variables

$n \setminus d$	2	4	6	8
1	yes	yes	yes	yes
2	yes	yes	no	no
3	yes	no	no	no
4	yes	no	no	no

- in general  $f$  is PSD does not imply  $f$  is SOS

## some background

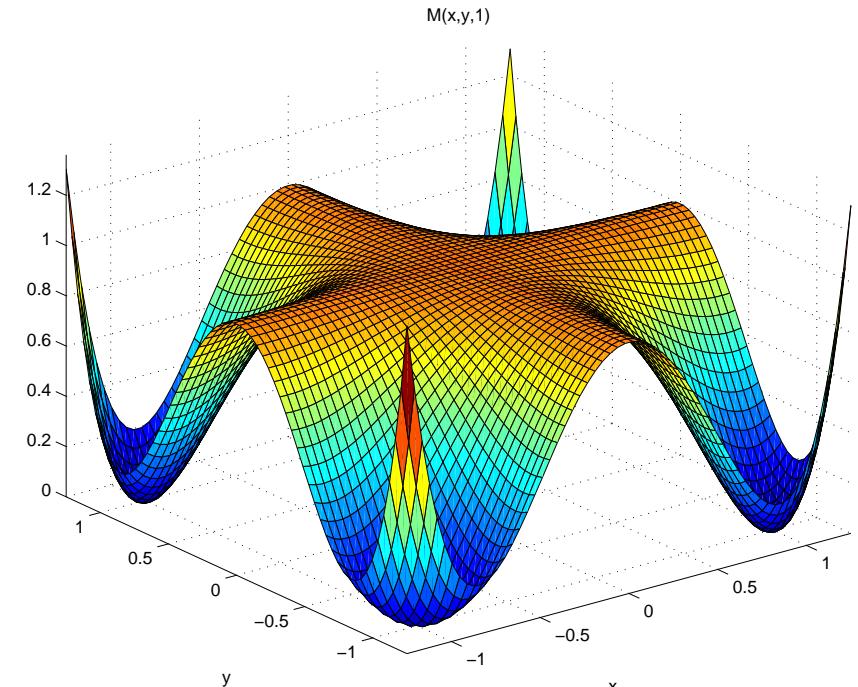
- Connections with Hilbert's 17th problem, solved by Artin: every PSD polynomial is a SOS of *rational functions*.
- If  $f$  is not SOS, then can try with  $gf$ , for some  $g$ .
  - For fixed  $f$ , can optimize over  $g$  too
  - Otherwise, can use a “universal” construction of Pólya-Reznick.

More about this later.

# The Motzkin Polynomial

A positive semidefinite polynomial,  
that is *not* a sum of squares.

$$M(x, y) = x^2y^4 + x^4y^2 + 1 - 3x^2y^2$$



- Nonnegativity follows from the arithmetic-geometric inequality applied to  $(x^2y^4, x^4y^2, 1)$
- Introduce a nonnegative factor  $x^2 + y^2 + 1$
- Solving the SDPs we obtain the decomposition:

$$(x^2 + y^2 + 1) M(x, y) = (x^2y - y)^2 + (xy^2 - x)^2 + (x^2y^2 - 1)^2 + \frac{1}{4}(xy^3 - x^3y)^2 + \frac{3}{4}(xy^3 + x^3y - 2xy)^2$$

## The Univariate Case:

$$\begin{aligned}
 f(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_{2d}x^{2d} \\
 &= \begin{bmatrix} 1 \\ x \\ \vdots \\ x^d \end{bmatrix}^T \begin{bmatrix} q_{00} & q_{01} & \cdots & q_{0d} \\ q_{01} & q_{11} & \cdots & q_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ q_{0d} & q_{1d} & \cdots & q_{dd} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^d \end{bmatrix} \\
 &= \sum_{i=0}^d \left( \sum_{j+k=i} q_{jk} \right) x^i
 \end{aligned}$$

- In the univariate case, the SOS condition is exactly equivalent to non-negativity.
- The matrices  $A_i$  in the SDP have a Hankel structure. This can be exploited for efficient computation.

## About SOS/SDP

- The resulting SDP problem is polynomially sized (in  $n$ , for fixed  $d$ ).
- By properly choosing the monomials, we can exploit structure (sparsity, symmetries, ideal structure).
- An important feature: the problem is still a SDP *if the coefficients of  $F$  are variable*, and the dependence is affine.
- Can optimize over SOS polynomials in affinely described families.  
For instance, if we have  $p(x) = p_0(x) + \alpha p_1(x) + \beta p_2(x)$ , we can “easily” find values of  $\alpha, \beta$  for which  $p(x)$  is SOS.

# Global Optimization

Consider the problem

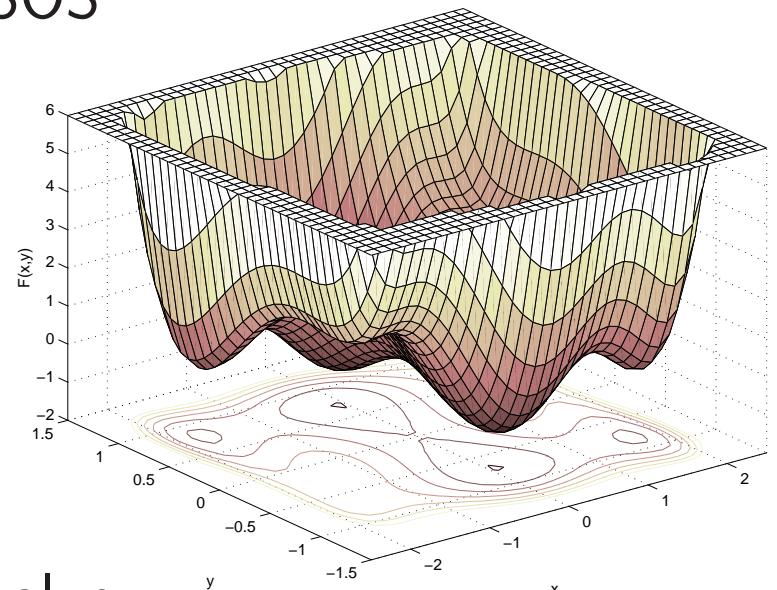
$$\min_{x,y} f(x, y)$$

with

$$f(x, y) := 4x^2 - \frac{21}{10}x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4$$

- Not convex. Many local minima. NP-hard.
- Find the largest  $\gamma$  s.t.  $f(x, y) - \gamma$  is SOS
- Essentially due to Shor (1987).
- A semidefinite program (convex!).
- If exact, can recover optimal solution.
- *Surprisingly* effective.

Solving, the maximum  $\gamma$  is -1.0316. Exact value.



## Lyapunov Example

A jet engine model

$$\begin{aligned}\dot{x} &= -y - \frac{3}{2}x^2 - \frac{1}{2}x^3 \\ \dot{y} &= 3x - y\end{aligned}$$

Try a generic 4th order polynomial Lyapunov function.

$$V(x, y) = \sum_{0 \leq j+k \leq 4} c_{jk} x^j y^k$$

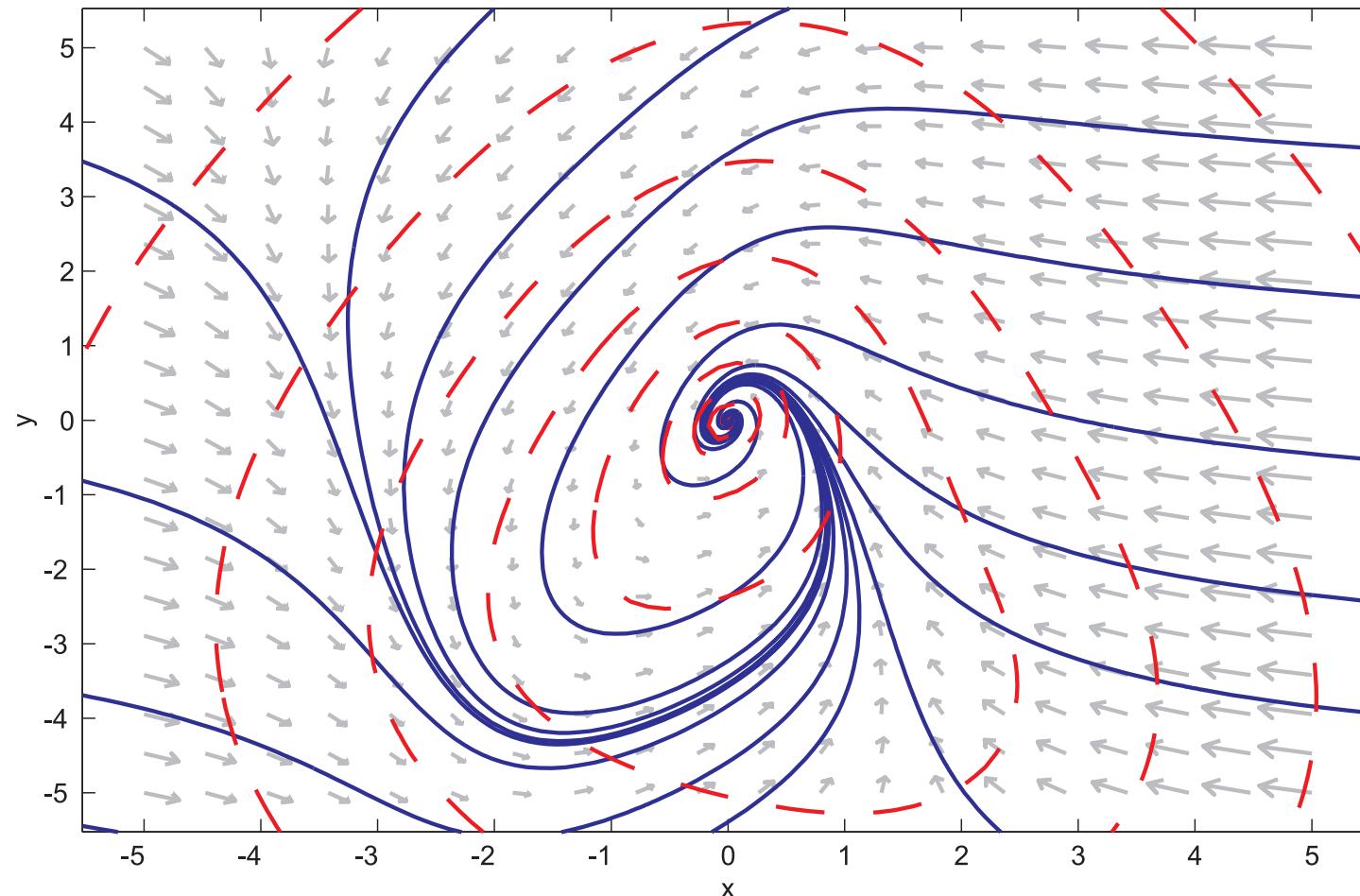
Find a  $V(x, y)$  that satisfies the conditions:

- $V(x, y)$  is SOS.
- $-\dot{V}(x, y)$  is SOS.

Both conditions are affine in the  $c_{jk}$ . Can do this directly using SOS/SDP!

# Lyapunov Example

After solving the SDPs, we obtain a Lyapunov function.



$$\begin{aligned}
 V = & 4.5819x^2 - 1.5786xy + 1.7834y^2 - 0.12739x^3 + 2.5189x^2y - 0.34069xy^2 \\
 & + 0.61188y^3 + 0.47537x^4 - 0.052424x^3y + 0.44289x^2y^2 + 0.0000018868xy^3 + 0.090723y^4
 \end{aligned}$$

## Extensions

- Other linear differential inequalities (e.g. Hamilton-Jacobi).
- Many possible variations: nonlinear optimal control, parameter dependent Lyapunov functions, etc.
- Can also do local results (for instance, on compact domains).
- Polynomial and rational vector fields, or functions with an underlying algebraic structure.
- Natural extension of the SDPs for the linear case.

## Automated Inference and Algebra

Automated inference is a well-known approach for formal proof systems.

Suppose  $f_1(x) \geq 0$  and  $f_2(x) \geq 0$ , then  $h(x) \geq 0$  if any of the following hold:

- (i)  $h(x) = f_1(x) + f_2(x)$
- (ii)  $h(x) = f_1(x)f_2(x)$
- (iii) For any  $f$ , the function  $h(x) = f(x)^2$

- We can use *algebra* to generate such *valid inequalities*
- Closure under these inference rules gives the cone of polynomials *generated* by the  $f_i$ , written  $\text{cone}\{f_1, f_2, \dots, f_m\}$

## The Sum-of-Squares Cone

A polynomial  $f \in \mathbb{R}[x_1, \dots, x_n]$  is called a *sum-of-squares* (SOS) if

$$f(x) = \sum_{i=1}^r s_i(x)^2$$

for some polynomials  $s_1, \dots, s_r$  and some  $r \geq 0$

- Denote by  $\Sigma$  the set of SOS polynomials
- $\Sigma$  is the *smallest* cone.
- This cone can be computationally characterized using semidefinite programming.
- The SOS decomposition is a simple certificate of nonnegativity of  $f$ .

## The Cone

We can explicitly parameterize the cone generated by the  $f_i$ .

For example,  $h \in \text{cone}\{f_1, f_2, f_3\}$  if and only if

$$h = s_1g_1 + \cdots + s_rg_r$$

where

$$s_i \in \Sigma \quad \text{and} \quad g_i \in \left\{ 1, f_1, f_2, f_3, f_1f_2, f_2f_3, f_3f_1, f_1f_2f_3 \right\}$$

In general, every  $h$  is a linear combination of *squarefree products* of the  $f_i$ , with *SOS coefficients*

## An Algebraic Dual Problem

Suppose  $f_1, \dots, f_m$  are polynomials. The primal feasibility problem is

does there exist  $x \in \mathbb{R}^n$  such that  
 $f_i(x) \geq 0 \quad \text{for all } i = 1, \dots, m$

The *dual feasibility problem* is

Is it true that  $-1 \in \text{cone}\{f_1, \dots, f_m\}$

If the dual problem is feasible, then the primal problem is infeasible.

In fact, a result called the *Positivstellensatz* (Stengle 1974) implies the *converse*; i.e., this is a *strong duality* result.

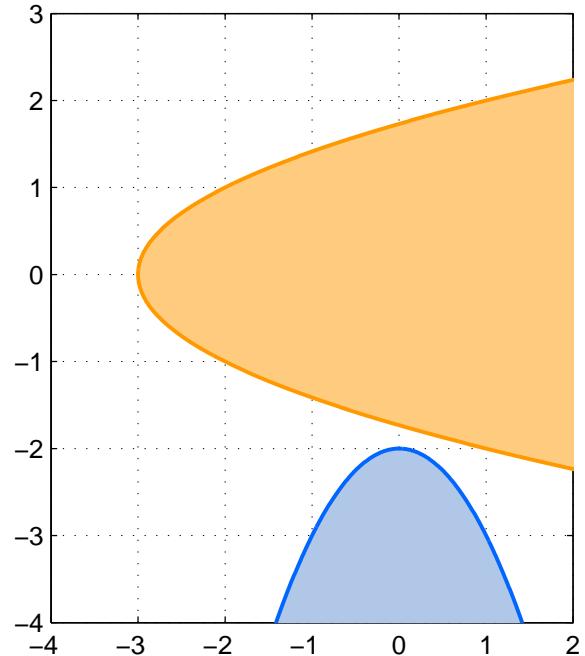
## Example

Consider the feasibility problem

$$S = \{ (x, y) \in \mathbb{R}^2 \mid f(x, y) \geq 0, g(x, y) \geq 0 \}$$

where

$$f = x - y^2 + 3 \quad g = -y - x^2 - 2$$



By the P-satz, the primal is infeasible if and only if there exist polynomials  $s_0, s_1, s_2, s_3 \in \Sigma$  such that

$$-1 = s_0 + s_1 f + s_2 g + s_3 f g$$

A certificate is given by

$$s_0 = \frac{1}{3} + 2\left(y + \frac{3}{2}\right)^2 + 6\left(x - \frac{1}{6}\right)^2, \quad s_1 = 2, \quad s_2 = 6, \quad s_3 = 0$$

Suppose we have SOS polynomials  $s_0, \dots, s_3$  such that

$$-1 = s_0 + s_1 f_1 + s_2 f_2 + s_3 f_1 f_2$$

Then this is a *certificate* that there is no  $x \in \mathbb{R}^n$  such that

$$f_1(x) \geq 0 \quad \text{and} \quad f_2(x) \geq 0$$

## Positivstellensatz

The polynomials  $s_i$  give a *certificate of infeasibility* of the primal problem.

Given them, one may immediately computationally *verify* that

$$-1 = s_1g_1 + \cdots + s_rg_r$$

and this is a *proof* of infeasibility

## Finding Refutations

- Geometrically,  $\text{cone}\{f_1, \dots, f_m\}$  is a *convex cone*, so testing if it contains  $-1$  is a *convex program*.
- There is a *correspondence* between the geometric object (the *feasible set*) and the algebraic object (the *cone*).

# Numerical Linear Algebra Software

(based on slides written by Michael Grant)

- BLAS, ATLAS
- LAPACK
- sparse matrices

# Numerical linear algebra in optimization

most *memory usage* and *computation time* in optimization methods is spent on numerical linear algebra, *e.g.*,

- constructing sets of linear equations (*e.g.*, Newton or KKT systems)
    - matrix-matrix products, matrix-vector products, . . .
  - and solving them
    - factoring, forward and backward substitution, . . .
- . . . so knowing about numerical linear algebra is a good thing

## Why not just use Matlab?

- Matlab (Octave, . . . ) is OK for prototyping an algorithm
- but you'll need to use a real language (*e.g.*, C, C++, Python) when
  - your problem is very large, or has special structure
  - speed is critical (*e.g.*, real-time)
  - your algorithm is embedded in a larger system or tool
  - you want to avoid proprietary software
- in any case, the numerical linear algebra in Matlab is done using standard free libraries

# How to write numerical linear algebra software

## DON'T!

whenever possible, rely on *existing, mature* software libraries

- you can focus on the higher-level algorithm
- your code will be more portable, less buggy, and will run faster—sometimes *much* faster

# **Netlib**

the grandfather of *all* numerical linear algebra web sites

<http://www.netlib.org>

- maintained by University of Tennessee, Oak Ridge National Laboratory, and colleagues worldwide
- most of the code is public domain or freely licensed
- much written in FORTRAN 77 (gasp!)

# Basic Linear Algebra Subroutines (BLAS)

written by people who had the foresight to understand the future benefits of a standard suite of “kernel” routines for linear algebra.

created and organized in three *levels*:

- *Level 1*, 1973-1977:  $O(n)$  vector operations: addition, scaling, dot products, norms
- *Level 2*, 1984-1986:  $O(n^2)$  matrix-vector operations: matrix-vector products, triangular matrix-vector solves, rank-1 and symmetric rank-2 updates
- *Level 3*, 1987-1990:  $O(n^3)$  matrix-matrix operations: matrix-matrix products, triangular matrix solves, low-rank updates

## BLAS operations

Level 1	addition/scaling dot products, norms	$\alpha x, \quad \alpha x + y$ $x^T y, \quad \ x\ _2, \quad \ x\ _1$
Level 2	matrix/vector products rank 1 updates rank 2 updates triangular solves	$\alpha Ax + \beta y, \quad \alpha A^T x + \beta y$ $A + \alpha xy^T, \quad A + \alpha xx^T$ $A + \alpha xy^T + \alpha yx^T$ $\alpha T^{-1}x, \quad \alpha T^{-T}x$
Level 3	matrix/matrix products  rank- $k$ updates rank- $2k$ updates triangular solves	$\alpha AB + \beta C, \quad \alpha AB^T + \beta C$ $\alpha A^T B + \beta C, \quad \alpha A^T B^T + \beta C$ $\alpha AA^T + \beta C, \quad \alpha A^T A + \beta C$ $\alpha A^T B + \alpha B^T A + \beta C$ $\alpha T^{-1}C, \quad \alpha T^{-T}C$

## Level 1 BLAS naming convention

BLAS routines have a Fortran-inspired naming convention:

prefix	X	XXXX
	data type	operation

data types:

s	single precision real	d	double precision real
c	single precision complex	z	double precision complex

operations:

axpy	$y \leftarrow \alpha x + y$	dot	$r \leftarrow x^T y$
nrm2	$r \leftarrow \ x\ _2 = \sqrt{x^T x}$	asum	$r \leftarrow \ x\ _1 = \sum_i  x_i $

example:

**cbLAS\_ddot**    double precision real dot product

## BLAS naming convention: Level 2/3

cblas_	X	XX	XXX
prefix	data type	structure	operation

matrix structure:

tr	triangular	tp	packed triangular	tb	banded triangular
sy	symmetric	sp	packed symmetric	sb	banded symmetric
hy	Hermitian	hp	packed Hermitian	hn	banded Hermitian
ge	general			gb	banded general

operations:

mv	$y \leftarrow \alpha Ax + \beta y$	sv	$x \leftarrow A^{-1}x$ (triangular only)
r	$A \leftarrow A + xx^T$	r2	$A \leftarrow A + xy^T + yx^T$
mm	$C \leftarrow \alpha AB + \beta C$	r2k	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$

examples:

  cblas\_dtrmv   double precision real triangular matrix-vector product  
  cblas\_dsyrr2k   double precision real symmetric rank-2k update

## Using BLAS efficiently

always choose a higher-level BLAS routine over multiple calls to a lower-level BLAS routine

$$A \leftarrow A + \sum_{i=1}^k x_i y_i^T, \quad A \in \mathbf{R}^{m \times n}, \quad x_i \in \mathbf{R}^m, \quad y_i \in \mathbf{R}^n$$

two choices:  $k$  separate calls to the Level 2 routine `cblas_dger`

$$A \leftarrow A + x_1 y_1^T, \quad \dots \quad A \leftarrow A + x_k y_k^T$$

or a single call to the Level 3 routine `cblas_dgemm`

$$A \leftarrow A + XY^T, \quad X = [x_1 \cdots x_k], \quad Y = [y_1 \cdots y_k]$$

the Level 3 choice will perform much better

## Is BLAS necessary?

why use BLAS when writing your own routines is so easy?

$$A \leftarrow A + XY^T, \quad A \in \mathbf{R}^{m \times n}, X \in \mathbf{R}^{m \times p}, Y \in \mathbf{R}^{n \times p}$$

$$A_{ij} \leftarrow A_{ij} + \sum_{k=1}^p X_{ik}Y_{jk}$$

```
void matmultadd( int m, int n, int p, double* A,
                  const double* X, const double* Y ) {
    int i, j, k;
    for ( i = 0 ; i < m ; ++i )
        for ( j = 0 ; j < n ; ++j )
            for ( k = 0 ; k < p ; ++k )
                A[ i + j * n ] += X[ i + k * p ] * Y[ j + k * p ];
}
```

## Is BLAS necessary?

- tuned/optimized BLAS will run faster than your home-brew version — often 10× or more
- BLAS is tuned by selecting block sizes that fit well with your processor, cache sizes
- ATLAS (automatically tuned linear algebra software)

<http://math-atlas.sourceforge.net>

uses automated code generation and testing methods to *generate* an optimized BLAS library for a specific computer

## Improving performance through blocking

*blocking* is used to improve the performance of matrix/vector and matrix/matrix multiplications, Cholesky factorizations, etc.

$$A + XY^T \leftarrow \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} X_{11} \\ X_{21} \end{bmatrix} + \begin{bmatrix} Y_{11}^T & Y_{21}^T \end{bmatrix}$$

$$\begin{aligned} A_{11} &\leftarrow A_{11} + X_{11}Y_{11}^T, & A_{12} &\leftarrow A_{12} + X_{11}Y_{21}^T, \\ A_{21} &\leftarrow A_{21} + X_{21}Y_{11}^T, & A_{22} &\leftarrow A_{22} + X_{21}Y_{21}^T \end{aligned}$$

optimal block size, and order of computations, depends on details of processor architecture, cache, memory

## Linear Algebra PACKage (LAPACK)

LAPACK contains subroutines for solving linear systems and performing common matrix decompositions and factorizations

- first release: February 1992; latest version (3.0): May 2000
- supercedes predecessors EISPACK and LINPACK
- supports same data types (single/double precision, real/complex) and matrix structure types (symmetric, banded, . . . ) as BLAS
- uses BLAS for internal computations
- routines divided into three categories: *auxiliary* routines, *computational* routines, and *driver* routines

## LAPACK computational routines

computational routines perform single, specific tasks

- factorizations:  $LU$ ,  $LL^T/LL^H$ ,  $LDL^T/LDL^H$ ,  $QR$ ,  $LQ$ ,  $QRZ$ , generalized  $QR$  and  $RQ$
- symmetric/Hermitian and nonsymmetric eigenvalue decompositions
- singular value decompositions
- generalized eigenvalue and singular value decompositions

## LAPACK driver routines

driver routines call a sequence of computational routines to solve standard linear algebra problems, such as

- linear equations:  $AX = B$
- linear least squares:  $\text{minimize}_x \|b - Ax\|_2$
- linear least-norm:

$$\begin{aligned} & \text{minimize}_y \|y\|_2 \\ & \text{subject to } d = By \end{aligned}$$

- generalized linear least squares problems:

$$\begin{aligned} & \text{minimize}_x \|c - Ax\|_2 \\ & \text{subject to } Bx = d \end{aligned}$$

$$\begin{aligned} & \text{minimize}_y \|y\|_2 \\ & \text{subject to } d = Ax + By \end{aligned}$$

## LAPACK example

solve KKT system

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$x \in \mathbf{R}^n$ ,  $v \in \mathbf{R}^m$ ,  $H = H^T \succ 0$ ,  $m < n$

*option 1:* driver routine `dssv` uses computational routine `dSYTRF` to compute permuted  $LDL^T$  factorization

$$\begin{bmatrix} H & A \\ A & 0 \end{bmatrix} \rightarrow PLDL^TP^T$$

and performs remaining computations to compute solution

$$\begin{bmatrix} x \\ y \end{bmatrix} = P^T L^{-1} D^{-1} L^{-T} P \begin{bmatrix} a \\ b \end{bmatrix}$$

*option 2: block elimination*

$$y = (AH^{-1}A^T)^{-1}(AH^{-1}a - b), \quad x = H^{-1}a - H^{-1}A^Ty$$

- first we solve the system  $H[Z \ w] = [A^T \ a]$  using driver routine `dspsv`
- then we construct and solve  $(AZ)y = Aw - b$  using `dspsv` again
- $x = w - Zy$

using this approach we could exploit structure in  $H$ , e.g., banded

## What about other languages?

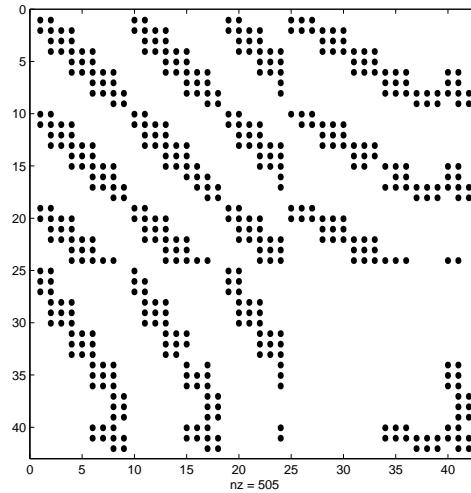
BLAS and LAPACK routines can be called from C, C++, Java, Python,

...

an alternative is to use a “native” library, such as

- C++: Boost uBlas, Matrix Template Library
- Python: NumPy/SciPy, CVXOPT
- Java: JAMA

# Sparse matrices



- $A \in \mathbb{R}^{m \times n}$  is *sparse* if it has “enough zeros that it pays to take advantage of them” (J. Wilkinson)
- usually this means  $n_{\text{nz}}$ , number of elements known to be nonzero, is small:  $n_{\text{nz}} \ll mn$

## Sparse matrices

sparse matrices can save memory and time

- storing  $A \in \mathbf{R}^{m \times n}$  using double precision numbers
  - dense:  $8mn$  bytes
  - sparse:  $\approx 16n_{\text{NZ}}$  bytes or less, depending on storage format
- operation  $y \leftarrow y + Ax$ :
  - dense:  $mn$  flops
  - sparse:  $n_{\text{NZ}}$  flops
- operation  $x \leftarrow T^{-1}x$ ,  $T \in \mathbf{R}^{n \times n}$  triangular, nonsingular:
  - dense:  $n^2/2$  flops
  - sparse:  $n_{\text{NZ}}$  flops

## Representing sparse matrices

- several methods used
- simplest (but typically not used) is to store the data as list of  $(i, j, A_{ij})$  triples
- column compressed format: an array of pairs  $(A_{ij}, i)$ , and an array of pointers into this array that indicate the start of a new column
- for high end work, exotic data structures are used
- sadly, no universal standard (yet)

## Sparse BLAS?

sadly there is not (yet) a standard sparse matrix BLAS library

- the “official” *sparse BLAS*

<http://www.netlib.org/blas/blast-forum>  
<http://math.nist.gov/spblas>

- C++: Boost uBlas, Matrix Template Library, SparseLib++
- Python: SciPy, PySparse, CVXOPT

## Sparse factorizations

libraries for factoring/solving systems with sparse matrices

- most comprehensive: SuiteSparse (Tim Davis)

<http://www.cise.ufl.edu/research/sparse/SuiteSparse>

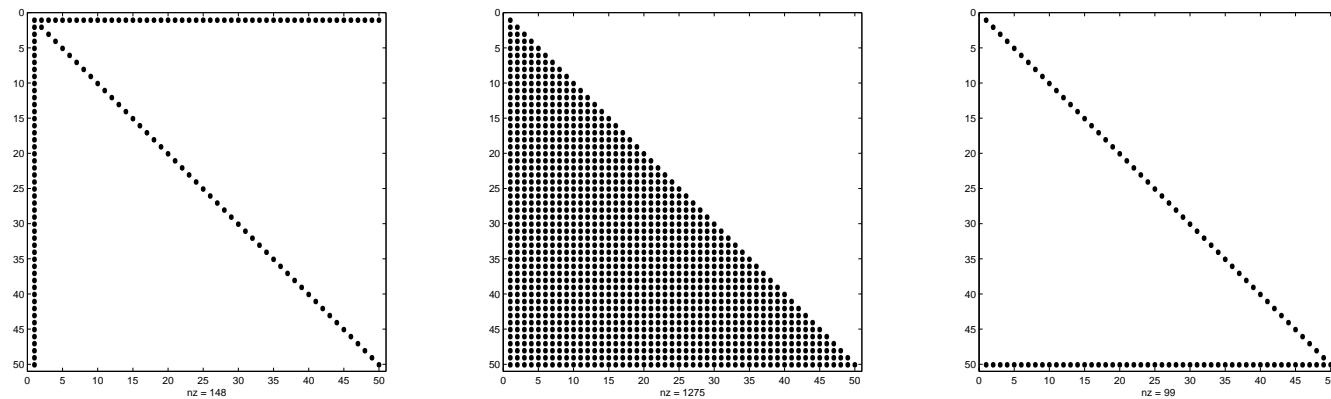
- others include SuperLU, TAUCS, SPOOLES
- typically include

- $A = PLL^T P^T$  Cholesky
- $A = PLDL^T P^T$  for symmetric indefinite systems
- $A = P_1 L U P_2^T$  for general (nonsymmetric) matrices

$P$ ,  $P_1$ ,  $P_2$  are permutations or *orderings*

# Sparse orderings

sparse orderings can have a *dramatic* effect on the sparsity of a factorization



- left: spy diagram of original NW arrow matrix
- center: spy diagram of Cholesky factor with no permutation ( $P = I$ )
- right: spy diagram of Cholesky factor with the best permutation (permute  $1 \rightarrow n$ )

## Sparse orderings

- general problem of choosing the ordering that produces the sparsest factorization is hard
- but, several simple heuristics are very effective
- more exotic ordering methods, *e.g.*, nested dissection, can work very well

## Symbolic factorization

- for Cholesky factorization, the ordering can be chosen based only on the sparsity pattern of  $A$ , and *not* its numerical values
- factorization can be divided into two stages: *symbolic* factorization and *numerical* factorization
  - when solving *multiple* linear systems with identical sparsity patterns, symbolic factorization can be computed just once
  - more effort can go into selecting an ordering, since it will be amortized across multiple numerical factorizations
- ordering for  $LDL^T$  factorization usually has to be done on the fly, *i.e.*, based on the data

## Other methods

we list some other areas in numerical linear algebra that have received significant attention:

- *iterative* methods for sparse and structured linear systems
- parallel and distributed methods (MPI)
- fast linear operators: fast Fourier transforms (FFTs), convolutions, state-space linear system simulations

there is considerable existing research, and accompanying public domain (or freely licensed) code