

# Alternating Direction Method of Multipliers

Prof S. Boyd

NIPS Workshop on Optimization for Machine Learning, 12/16/11

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

# 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

$$x^{k+1} := \operatorname{argmin}_x L(x, y^k) \quad // \textit{x-minimization}$$

$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b) \quad // \textit{dual update}$$

- ▶ 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)

$$\begin{aligned}x^{k+1} &:= \underset{x}{\operatorname{argmin}} L_\rho(x, y^k) \\y^{k+1} &:= y^k + \rho(Ax^{k+1} - b)\end{aligned}$$

(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  $y^{k+1} = y^k + \rho(Ax^{k+1} - b)$  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  $+\infty$ , ...)
- ▶ *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:

$$x^{k+1} := \operatorname{argmin}_x L_\rho(x, z^k, y^k) \quad // \textit{x-minimization}$$

$$z^{k+1} := \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) \quad // \textit{z-minimization}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \quad // \textit{dual update}$$

## 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):

$$\begin{aligned}x^{k+1} &:= \underset{x}{\operatorname{argmin}} (f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2) \\ z^{k+1} &:= \underset{z}{\operatorname{argmin}} (g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2) \\ u^{k+1} &:= u^k + (Ax^{k+1} + Bz^{k+1} - c)\end{aligned}$$

# 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 exploiting 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  $A = I$

$$x^+ = \underset{x}{\operatorname{argmin}} (f(x) + (\rho/2)\|x - v\|_2^2) = \mathbf{prox}_{f,\rho}(v)$$

- ▶ 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

▶  $f(x) = (1/2)x^T P x + q^T x + r$

▶  $x^+ := (P + \rho A^T A)^{-1}(\rho A^T v - q)$

- ▶ 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
  - preconditioned 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

# Constrained convex optimization

- ▶ consider ADMM for generic problem

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

- ▶ ADMM form: take  $g$  to be indicator of  $\mathcal{C}$

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

- ▶ algorithm:

$$x^{k+1} := \underset{x}{\operatorname{argmin}} (f(x) + (\rho/2)\|x - z^k + u^k\|_2^2)$$

$$z^{k+1} := \Pi_{\mathcal{C}}(x^{k+1} + u^k)$$

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

# 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:

$$\begin{aligned} & \text{minimize} && \mathbf{Tr}(SX) - \log \det X + \lambda \|Z\|_1 \\ & \text{subject to} && X - Z = 0 \end{aligned}$$

► ADMM:

$$\begin{aligned} X^{k+1} &:= \underset{X}{\operatorname{argmin}} \left( \mathbf{Tr}(SX) - \log \det X + (\rho/2) \|X - Z^k + U^k\|_F^2 \right) \\ Z^{k+1} &:= S_{\lambda/\rho}(X^{k+1} + U^k) \\ U^{k+1} &:= U^k + (X^{k+1} - Z^{k+1}) \end{aligned}$$



## 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

$$\text{minimize } \sum_{i=1}^N f_i(x)$$

- 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  $\operatorname{prox}_{g,\rho}$

## Consensus optimization via ADMM

- ▶ using  $\sum_{i=1}^N y_i^k = 0$ , algorithm simplifies to

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}} (f_i(x_i) + y_i^{kT} (x_i - \bar{x}^k) + (\rho/2) \|x_i - \bar{x}^k\|_2^2)$$

$$y_i^{k+1} := y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})$$

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

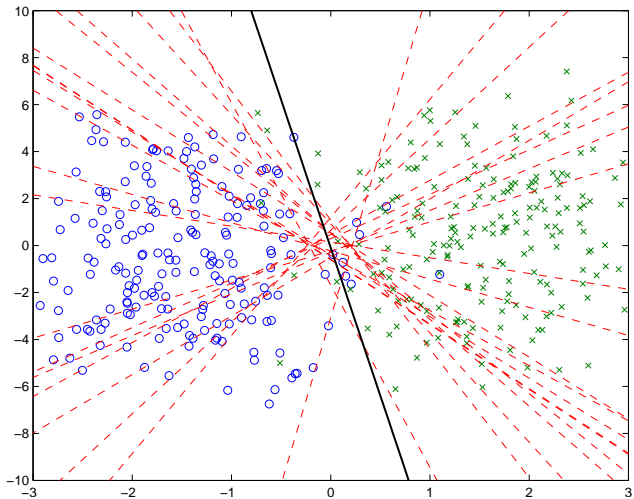
- ▶ data (examples)  $(a_i, b_i)$ ,  $i = 1, \dots, N$ ,  $a_i \in \mathbf{R}^n$ ,  $b_i \in \{-1, +1\}$
- ▶ linear classifier  $\text{sign}(a^T w + v)$ , with weight  $w$ , offset  $v$
- ▶ margin for  $i$ th example is  $b_i(a_i^T w + v)$ ; want margin to be positive
- ▶ loss for  $i$ th example is  $l(b_i(a_i^T w + v))$ 
  - $l$  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



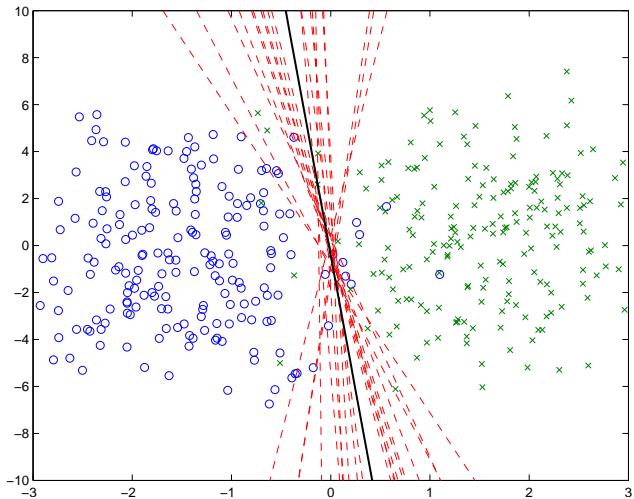
## Consensus SVM example

- ▶ 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

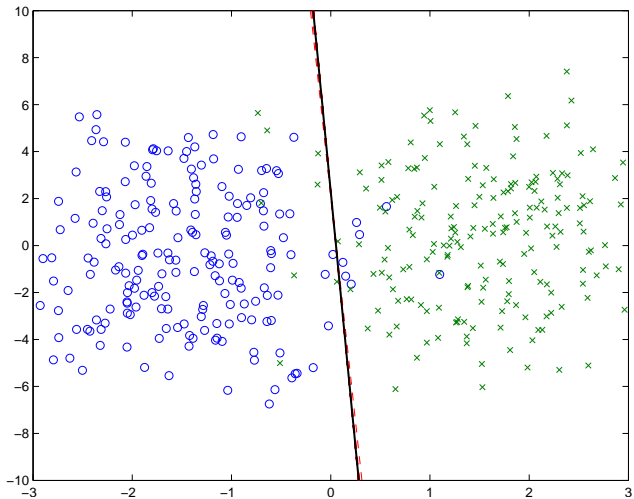
# Iteration 1



## Iteration 5



## Iteration 40



## Distributed lasso example

- ▶ example with **dense**  $A \in \mathbf{R}^{400000 \times 8000}$  (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

## 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 convergence 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

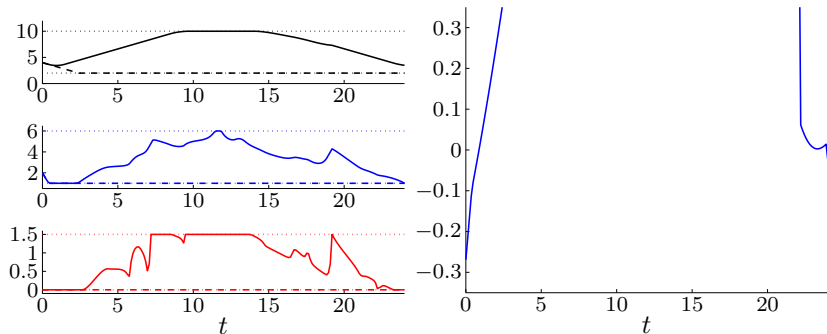
## Smart grid example

10 devices

- ▶ 3 generators
- ▶ 2 fixed loads
- ▶ 1 shiftable load
- ▶ 1 EV charging systems
- ▶ 1 battery
- ▶ 1 HVAC system
- ▶ 1 external tie

# Convergence

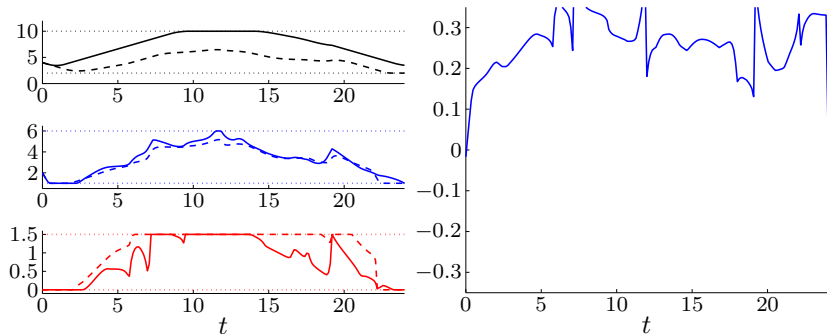
iteration:  $k = 1$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

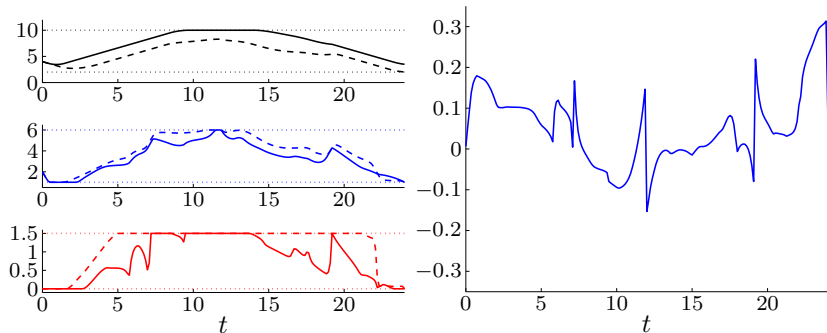
iteration:  $k = 3$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

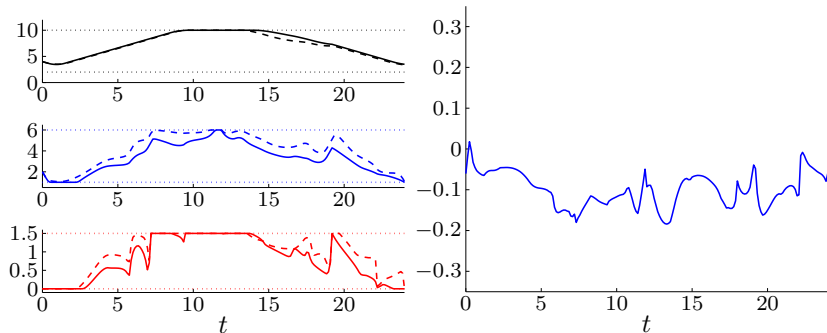
iteration:  $k = 5$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

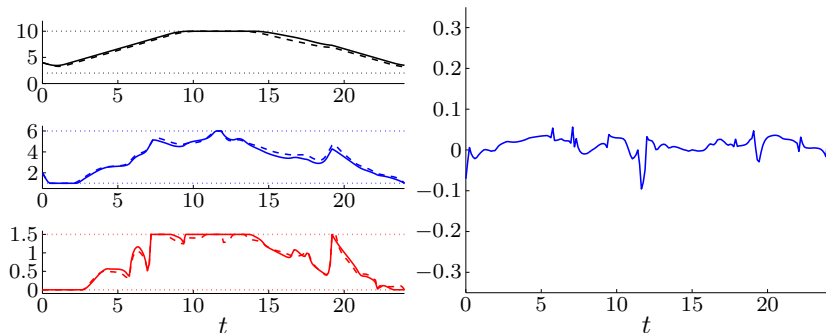
iteration:  $k = 10$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

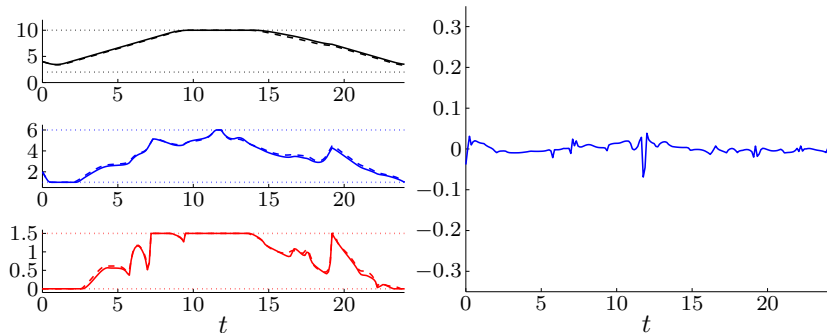
iteration:  $k = 15$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

iteration:  $k = 20$

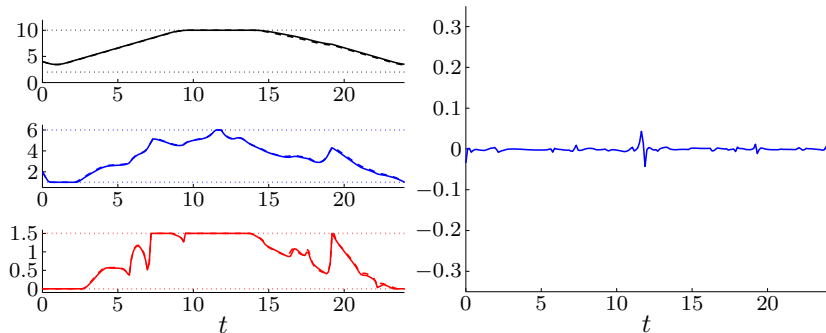


- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$



# Convergence

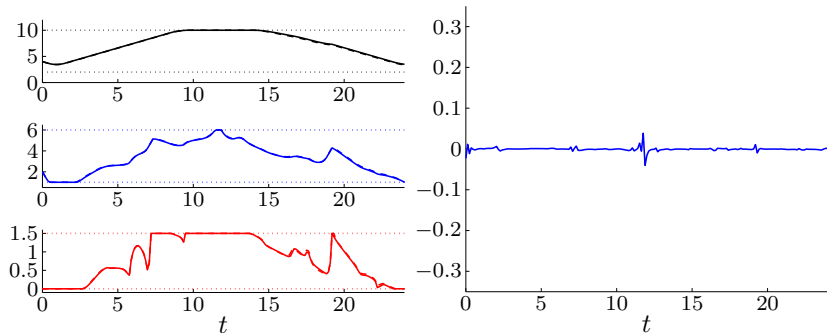
iteration:  $k = 25$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

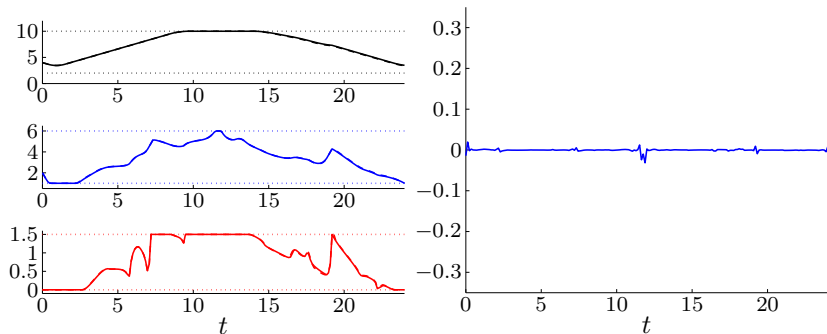
iteration:  $k = 30$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

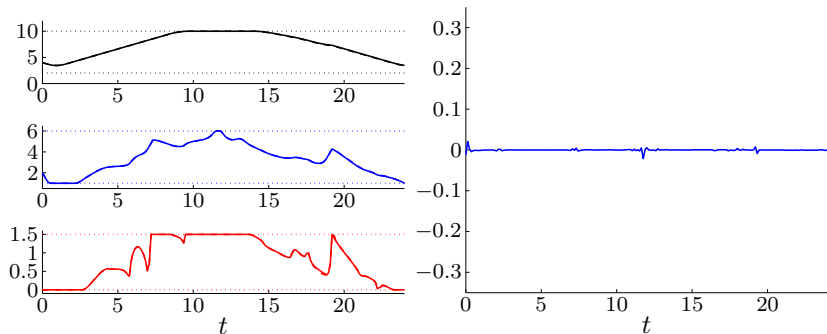
iteration:  $k = 35$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

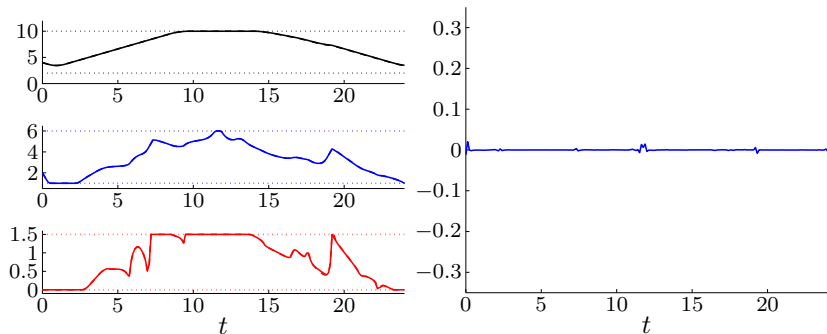
iteration:  $k = 40$



- ▶ left: solid: optimal generator profile, dashed: profile at  $k$ th iteration
- ▶ right: residual vector  $\bar{x}^k$

# Convergence

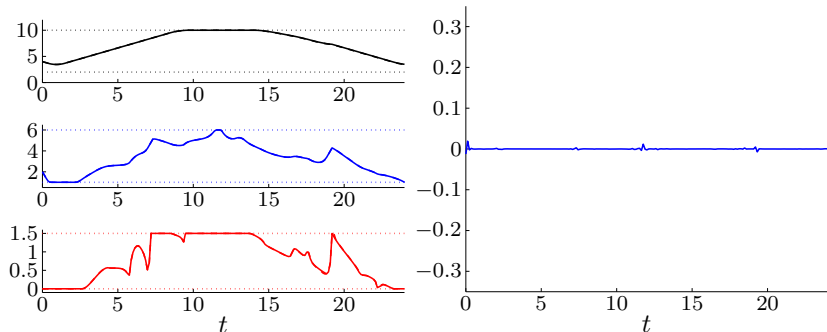
iteration:  $k = 45$



- ▶ 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**

# 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