Alternating Direction Method of Multipliers

Stephen Boyd

Lund University, 23/8/12

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 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: $\quad$ maximize $g(y)$
- recover $x^\star = \operatorname{argmin}_x L(x, y^\star)$

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

$$
\begin{array}{rcll}
x^{k+1} & := & \operatorname{argmin}_x L(x, y^k) & \textit{// } x\textit{-minimization} \\
y^{k+1} & := & y^k + \alpha^k (A x^{k+1} - b) & \textit{// dual update}
\end{array}
$$

- works, with lots of strong assumptions

Dual decomposition

- suppose f is separable:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \ldots, x_N)$$

- 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

$$x_i^{k+1} \quad := \quad \operatorname*{argmin}_{x_i} L_i(x_i, y^k)$$

  which can be carried out in parallel

Dual decomposition

- dual decomposition (Everett, Dantzig, Wolfe, Benders 1960–65)

$$
\begin{aligned}
x_i^{k+1} &:= \operatorname{argmin}_{x_i} L_i(x_i, y^k), \quad i = 1, \ldots, N \\
y^{k+1} &:= y^k + \alpha^k (\textstyle\sum_{i=1}^N A_i x_i^{k+1} - b)
\end{aligned}
$$

- 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

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)

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

(note specific dual update step length ρ)

Method of multipliers dual update step

- optimality conditions (for differentiable f):

$$Ax^\star - b = 0, \qquad \nabla f(x^\star) + A^T y^\star = 0$$

  (primal and dual feasibility)

- since x^{k+1} minimizes $L_\rho(x, y^k)$

$$\begin{array}{rcl} 0 & = & \nabla_x L_\rho(x^{k+1}, y^k) \\ & = & \nabla_x f(x^{k+1}) + A^T \left(y^k + \rho(Ax^{k+1} - b)\right) \\ & = & \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{array}$$

- 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: $Ax^{k+1} - b \to 0$

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

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)

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

  – 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:

$$\begin{array}{lcll} x^{k+1} & := & \operatorname{argmin}_x L_\rho(x, z^k, y^k) & \textit{// } x\textit{-minimization} \\ z^{k+1} & := & \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) & \textit{// } z\textit{-minimization} \\ y^{k+1} & := & y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) & \textit{// dual update} \end{array}$$

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 $k \to \infty$

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):

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

Convergence

- assume (very little!)
  - f, g convex, closed, proper
  - L_0 has a saddle point
- then ADMM converges:
  - iterates approach feasibility: $Ax^k + Bz^k - c \to 0$
  - objective approaches optimal value: $f(x^k) + g(z^k) \to p^\star$

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,

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \qquad x = (x_1, \ldots, x_N)$$

- 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 $A = I$

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

- some special cases:

$f = I_C$ (indicator fct. of set C)	$x^+ := \Pi_C(v)$ (projection onto C)
$f = \lambda\|\cdot\|_1$ (ℓ_1 norm)	$x_i^+ := S_{\lambda/\rho}(v_i)$ (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

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:

$$\begin{array}{lll} x^{k+1} & := & \operatorname*{argmin}_x \left(f(x) + (\rho/2)\|x - z^k + u^k\|_2^2 \right) \\ z^{k+1} & := & \Pi_{\mathcal{C}}(x^{k+1} + u^k) \\ u^{k+1} & := & u^k + x^{k+1} - z^{k+1} \end{array}$$

Lasso

- lasso problem:

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

- ADMM form:

$$\begin{array}{ll} \text{minimize} & (1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} & x - z = 0 \end{array}$$

- ADMM:

$$\begin{array}{rcl} x^{k+1} & := & (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) \\ z^{k+1} & := & S_{\lambda/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} & := & y^k + \rho(x^{k+1} - z^{k+1}) \end{array}$$

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 λ'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 Σ^{-1} sparse (*i.e.*, Gaussian Markov random field)
- estimate Σ^{-1} via ℓ_1 regularized maximum likelihood

$$\text{minimize} \quad \mathbf{Tr}(SX) - \log\det X + \lambda \|X\|_1$$

- methods: COVSEL (Banerjee et al 2008), graphical lasso (FHT 2008)

Sparse inverse covariance selection via ADMM

- ADMM form:

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

- ADMM:

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

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

- Σ^{-1} is 1000×1000 with 10^4 nonzeros
  - graphical lasso (Fortran): 20 seconds – 3 minutes
  - ADMM (Matlab): 3 – 10 minutes
  - (depends on choice of λ)

- very rough experiment, but with no special tuning, ADMM is in ballpark of recent specialized methods

- (for comparison, COVSEL takes 25$+$ min when Σ^{-1} is a 400×400 tridiagonal matrix)

Outline

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 ith block of training data

- ADMM form:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & x_i - z = 0 \end{array}$$

  - 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

- $L_\rho(x,z,y) = \sum_{i=1}^N \left(f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2 \right)$

- ADMM:

$$
\begin{aligned}
x_i^{k+1} &:= \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{kT}(x_i - z^k) + (\rho/2)\|x_i - z^k\|_2^2 \right) \\
z^{k+1} &:= \frac{1}{N} \sum_{i=1}^N \left(x_i^{k+1} + (1/\rho) y_i^k \right) \\
y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - z^{k+1})
\end{aligned}
$$

- with regularization, averaging in z update is followed by $\mathbf{prox}_{g,\rho}$

Consensus optimization via ADMM

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

$$
\begin{array}{rcl}
x_i^{k+1} & := & \operatorname*{argmin}_{x_i} \left(f_i(x_i) + y_i^{kT}(x_i - \overline{x}^k) + (\rho/2)\|x_i - \overline{x}^k\|_2^2 \right) \\
y_i^{k+1} & := & y_i^k + \rho(x_i^{k+1} - \overline{x}^{k+1})
\end{array}
$$

  where $\overline{x}^k = (1/N)\sum_{i=1}^N x_i^k$

- in each iteration
  - gather x_i^k and average to get $\overline{x}^k$
  - scatter the average $\overline{x}^k$ 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 ith data block
- x_i^{k+1} is MAP estimate under prior $\mathcal{N}(\overline{x}^k + (1/\rho)y_i^k, \rho I)$
- 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

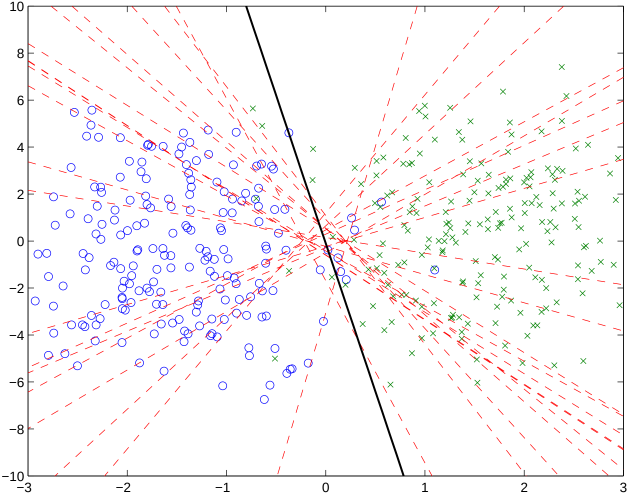
Consensus classification

- data (examples) (a_i, b_i), $i = 1, \ldots, N$, $a_i \in \mathbf{R}^n$, $b_i \in \{-1, +1\}$
- linear classifier $\mathbf{sign}(a^T w + v)$, with weight w, offset v
- margin for ith example is $b_i(a_i^T w + v)$; want margin to be positive
- loss for ith 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 (ℓ_2, ℓ_1, . . .)
- split data and use ADMM consensus to solve

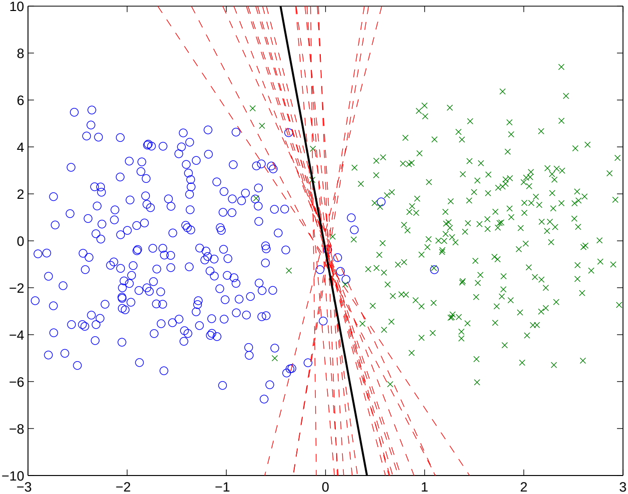
Consensus SVM example

- hinge loss $l(u) = (1-u)_+$ with ℓ_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

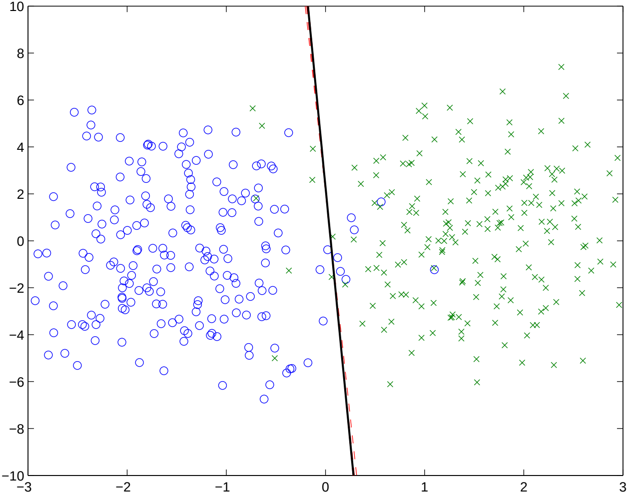
Iteration 1



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 $y^\star$ is a set of valid prices for the goods
- suggests real or virtual cash payment $(y^\star)^T x_i$ 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:

$$\begin{aligned} x_i^{k+1} &:= \operatorname*{argmin}_{x_i} \left(f_i(x_i) + (\rho/2)\|x_i - x_i^k + \overline{x}^k + u^k\|_2^2 \right) \\ u^{k+1} &:= u^k + \overline{x}^{k+1} \end{aligned}$$

- unscaled form:

$$\begin{aligned} x_i^{k+1} &:= \operatorname*{argmin}_{x_i} \left(f_i(x_i) + y^{kT} x_i + (\rho/2)\|x_i - (x_i^k - \overline{x}^k)\|_2^2 \right) \\ y^{k+1} &:= y^k + \rho \overline{x}^{k+1} \end{aligned}$$

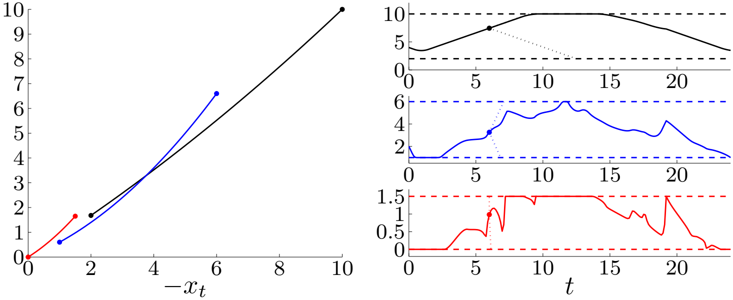
Interpretation as tâtonnement process

- *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

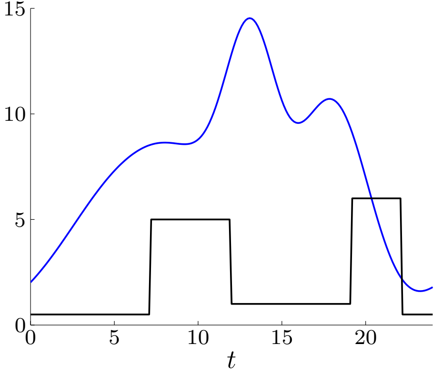
- N devices exchange power in time periods $t = 1, \ldots, T$
- $x_i \in \mathbf{R}^T$ is power flow *profile* for device i
- $f_i(x_i)$ is cost of profile x_i (and encodes constraints)
- $x_1 + \cdots + x_N = 0$ is energy balance (in each time period)
- dynamic energy management problem is exchange 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



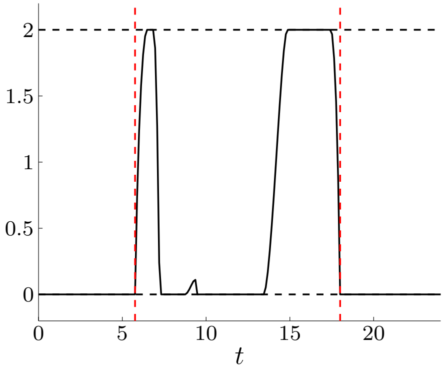
- 3 example generators
- left: generator costs/limits; right: ramp constraints
- can add cost for power changes

Fixed loads



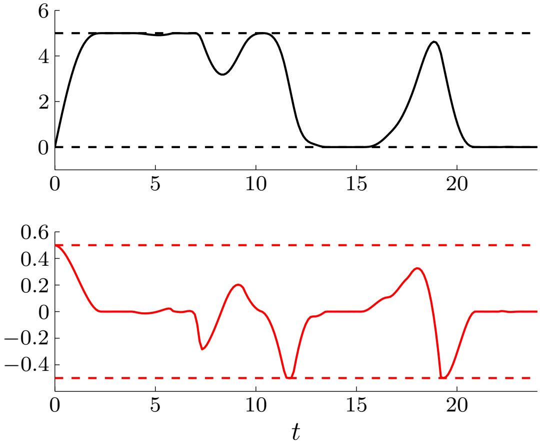
- 2 example fixed loads
- cost is $+\infty$ for not supplying load; zero otherwise

Shiftable load



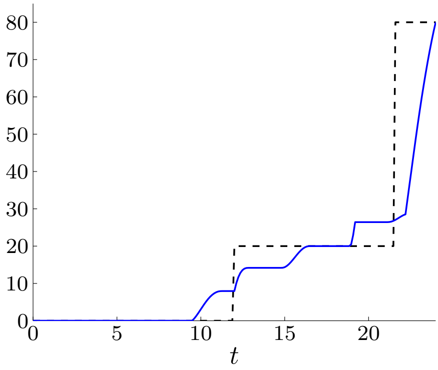
- total energy consumed over an interval must exceed given minimum level
- limits on energy consumed in each period
- cost is $+\infty$ for violating constraints; zero otherwise

Battery energy storage system



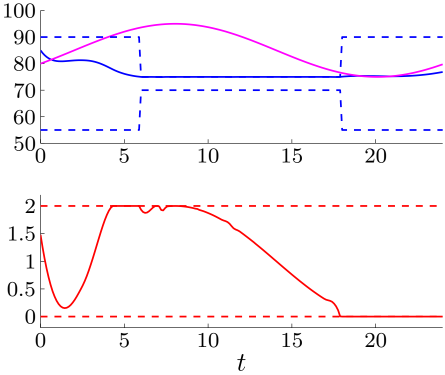
- energy store with maximum capacity, charge/discharge limits
- black: battery charge, red: charge/discharge profile
- cost is $+\infty$ for violating constraints; zero otherwise

Electric vehicle charging system



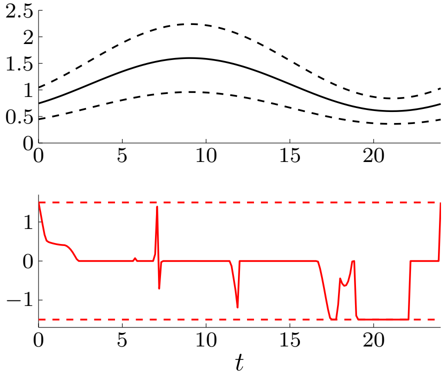
- black: desired charge profile; blue: charge profile
- shortfall cost for not meeting desired charge

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 $p^{\mathrm{ext}}(t) \pm \gamma(t)$
- solid: $p^{\mathrm{ext}}(t)$; dashed: $p^{\mathrm{ext}}(t) \pm \gamma(t)$

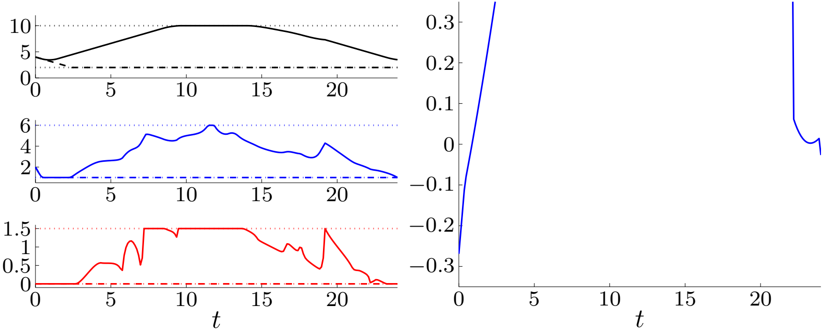
Smart grid example

10 devices

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

Convergence

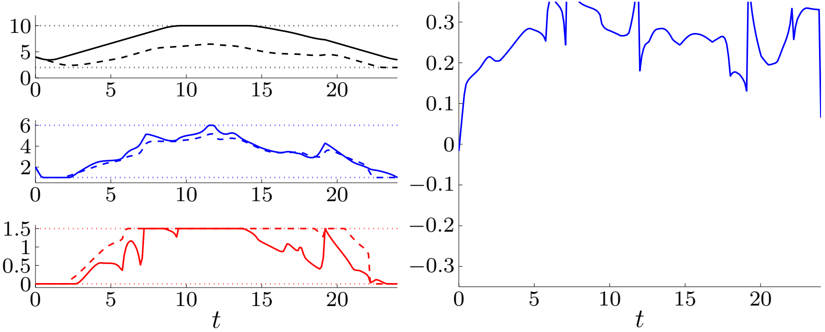
iteration: $k = 1$



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

Convergence

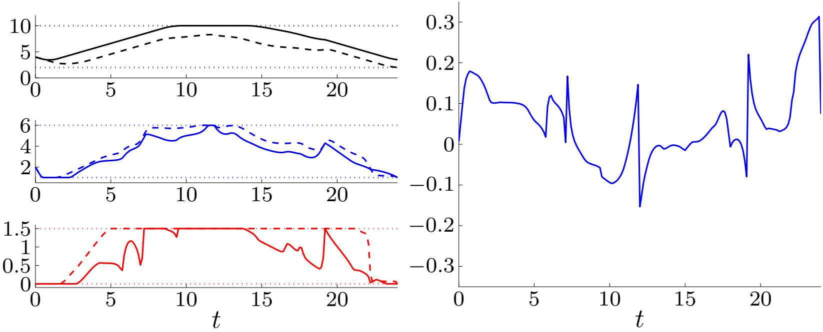
iteration: $k = 3$



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

Convergence

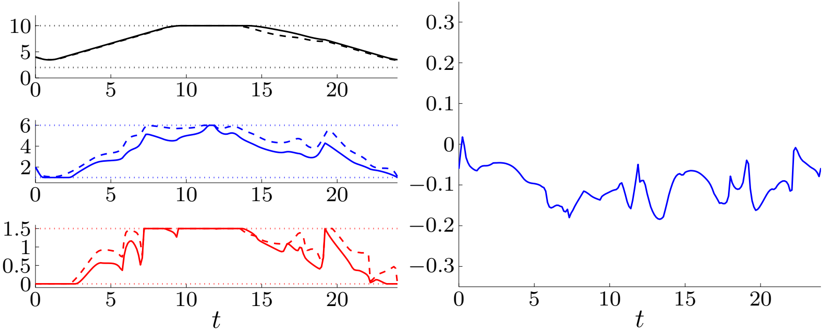
iteration: $k = 5$



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

Convergence

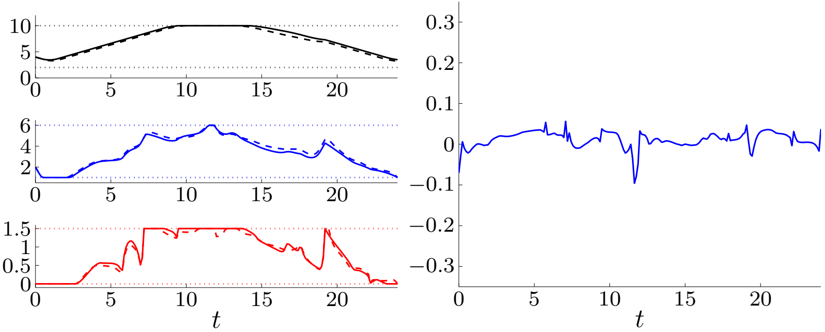
iteration: $k = 10$



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

Convergence

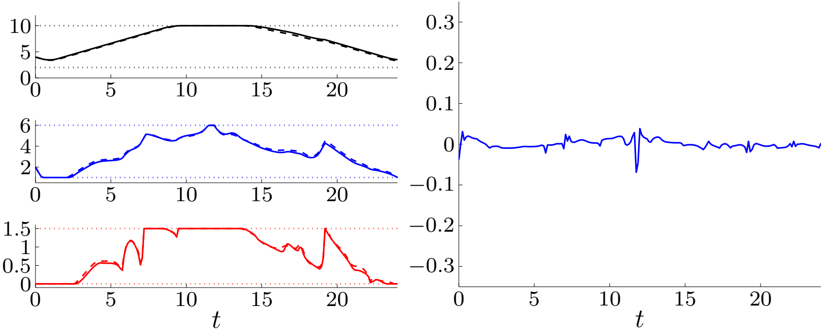
iteration: $k = 15$



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

Convergence

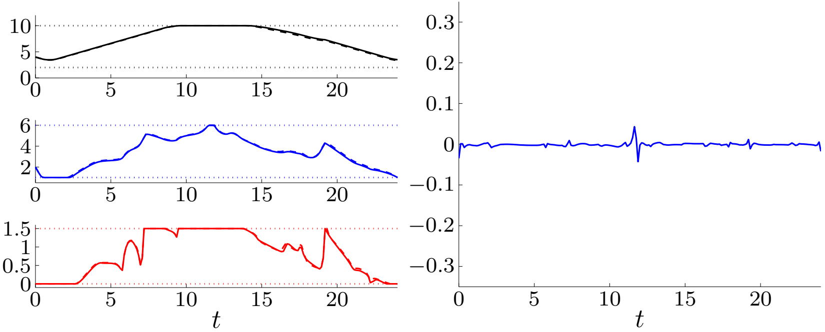
iteration: $k = 20$



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

Convergence

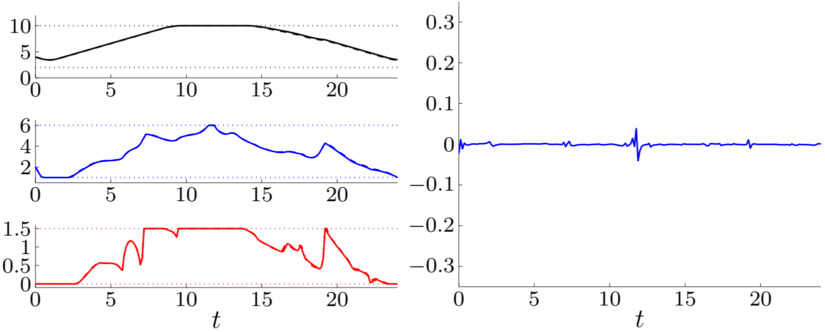
iteration: $k = 25$



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

Convergence

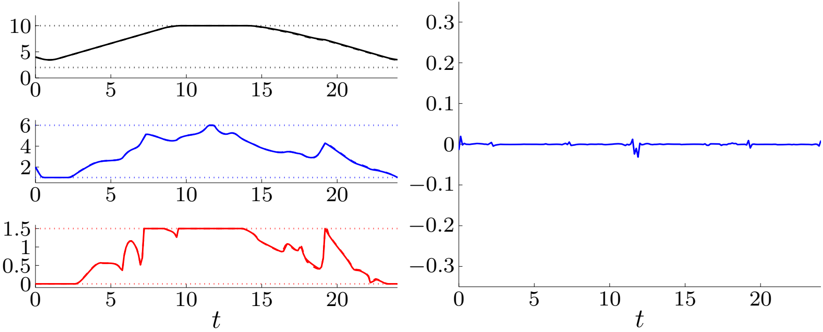
iteration: $k = 30$



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

Convergence

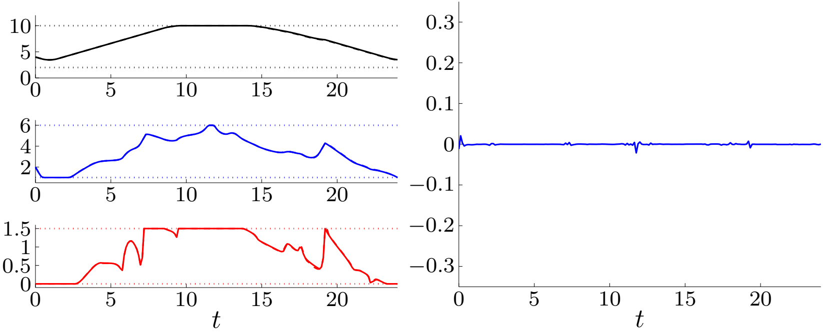
iteration: $k = 35$



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

Convergence

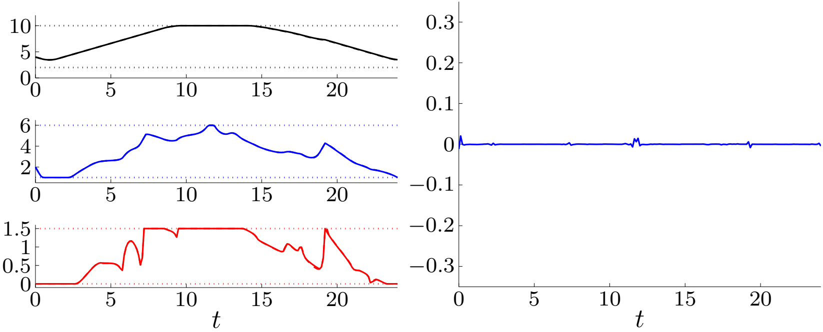
iteration: $k = 40$



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

Convergence

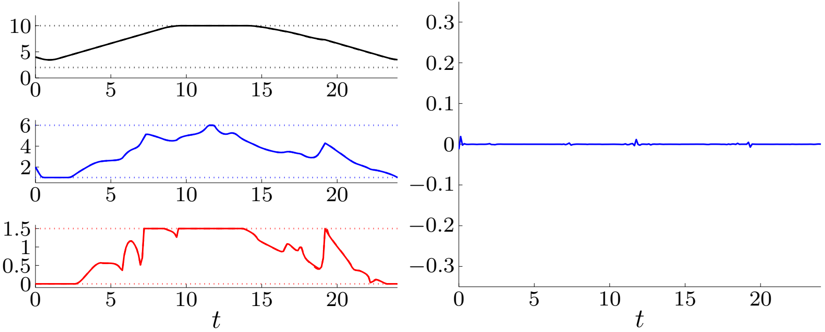
iteration: $k = 45$



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

Convergence

iteration: $k = 50$



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

Outline

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