The ADMM: Past, Present, and Future

Jonathan Eckstein
Rutgers University, Piscataway, NJ, USA

RUTGERS
Rutgers Business School
Newark and New Brunswick

RUTGERS

Thank You!

- I would like to thank the organizers for the opportunity to give this keynote presentation
    - Jacek Gondzio
    - Stefano Cipolla
    - Fillipo Zanetti
- And Jacek for sponsoring my one-month sabbatical visit to U. Edinburgh

Disclaimer

- Some of the early references I will mention are not online
- I did not bring my copies with me to Edinburgh
- So, there is a chance that some citations I will give are not exactly correct

The ADMM: Past

- The ADMM is now considered a standard optimization algorithm
- But it has an unusual history:

1. It was discovered empirically before it was analyzed mathematically

2. The initial discoverers and analyzers were French applied mathematics researchers specializing in large-scale discretized PDEs

3. Over 20 years elapsed between its initial analysis and its becoming popular

The ADMM – Background from the Standard ALM

In the mid-1970's, this group of researchers (Fortin, Glowinski, Marrocco, Gabay, Mercier) had reformulated their discretized PDEs (roughly) as follows:

- $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a convex function
- $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is a convex function
- M is an $m \times n$ matrix

$$\boxed{\min f(x) + g(Mx)}$$

- Equivalent formulation:

$$\boxed{\begin{array}{ll} \min & f(x) + g(z) \\ \text{ST} & Mx = z \end{array}}$$

Applying the Augmented Lagrangian Method

- To this formulation, they applied the augmented Lagrangian method (ALM)
  - A "hot" new method at the time
- Standard augmented Lagrangian method (ALM) for this formulation:

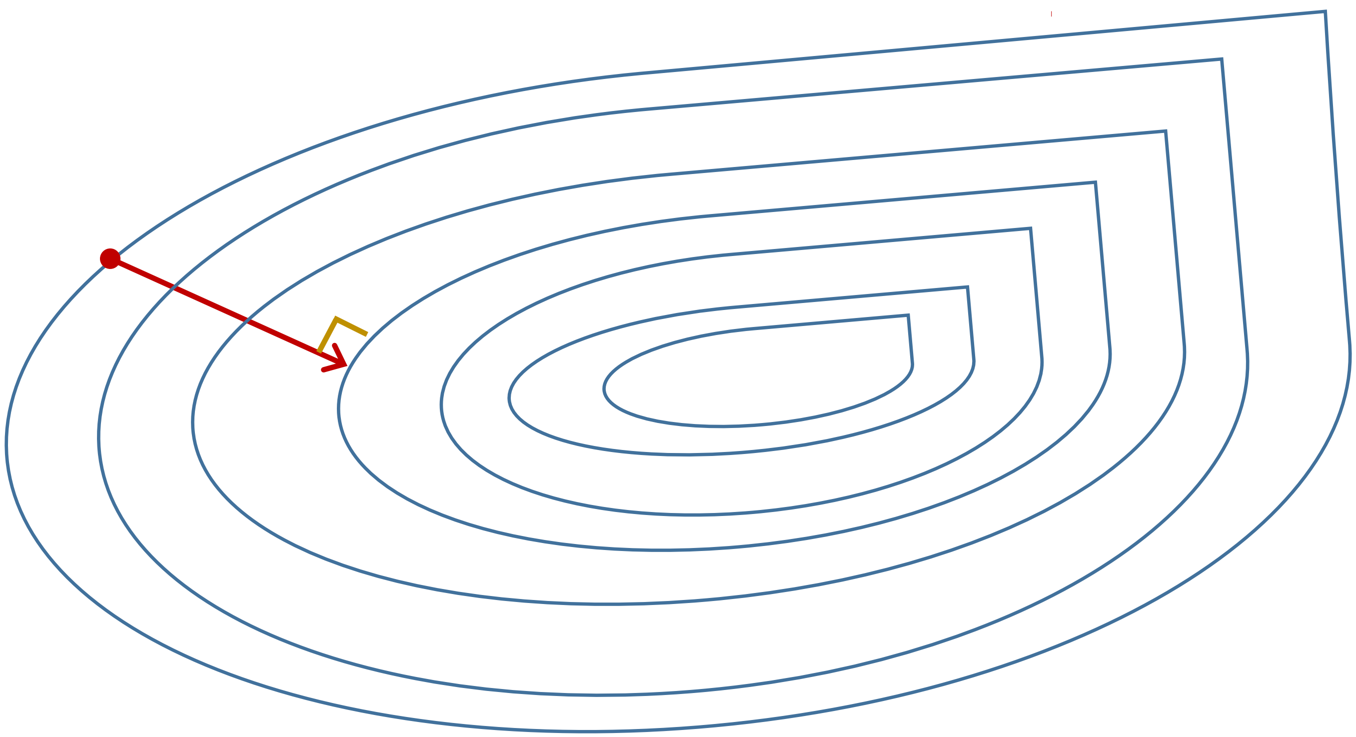
$$\left(x^{k+1}, z^{k+1}\right) \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n, z \in \mathbb{R}^m} \left\{ f(x) + g(z) + \left\langle p^k, Mx - z \right\rangle + \frac{c_k}{2} \|Mx - z\|^2 \right\}$$

$$p^{k+1} = p^k + c_k (Mx^{k+1} - z^{k+1})$$

- Although the equality constraints are gone, the cross terms in $\|Mx - z\|^2$ make the augmented Lagrangian harder to optimize than the ordinary Lagrangian (without $\|Mx - z\|^2$)
  - Cannot handle f and g independently
- But the ALM is much more stable than minimizing the ordinary Lagrangian & multiplier update (subgradient in the dual)

Interpretation of the ALM

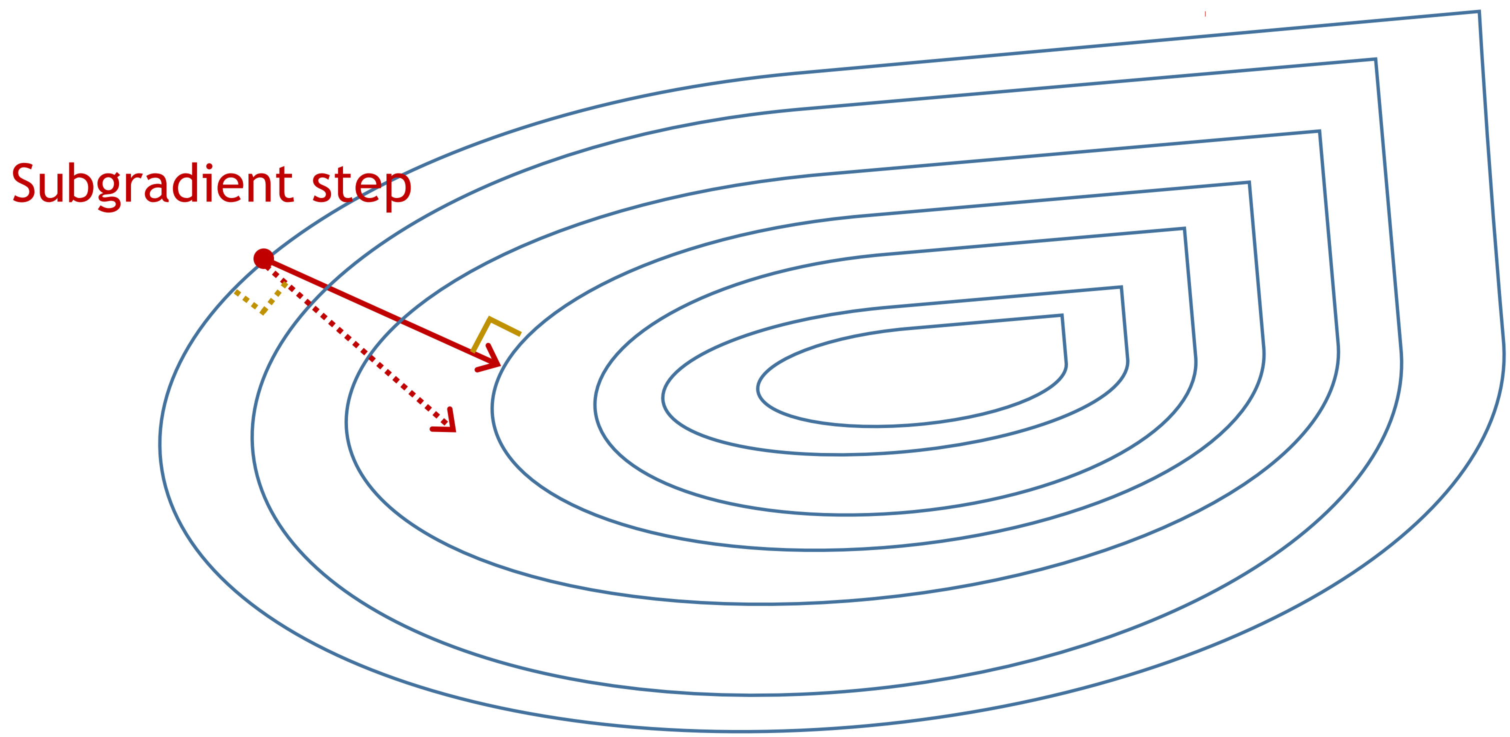
- The ALM method does an implicit subgradient step on the dual problem (as shown by Rockafellar; a form of “dual ascent”)



- The step direction is a subgradient of the function at the end of the step, not the beginning

Interpretation of the ALM

- The ALM method does an implicit subgradient step on the dual problem (as shown by Rockafellar; a form of "dual ascent")



- The step direction is a subgradient of the function at the end of the step, not the beginning
- Much more stable, but at the cost of those cross terms

Alternating Directions

- To make the cross terms less painful, Glowinski and Marrocco (1976) suggested an alternating direction method for the inner problem:
  - Minimize over x with z held fixed
  - Then minimize over z with x held fixed
- They suggested executing this loop a fixed number of times, then update the multipliers
- This inner iteration was later shown to converge by Tseng (2001) under fairly loose assumptions
  - But not in a fixed number of steps
  - Unless you can show that you have (at least approximately) minimized the inner problem, the multiplier update is no longer dual ascent

The ADMM is Born

- Interestingly, Glowinski and Marrocco observed the best performance when making only one pass through x and z at every iteration – the ADMM:

$$\boxed{\begin{aligned}
x^{k+1} &\in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ f(x) + g(z^k) + \left\langle p^k, Mx - z^k \right\rangle + \tfrac{c}{2} \left\| Mx - z^k \right\|^2 \right\} \\
z^{k+1} &\in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{ f(x^{k+1}) + g(z) + \left\langle p^k, Mx^{k+1} - z \right\rangle + \tfrac{c}{2} \left\| Mx^{k+1} - z \right\|^2 \right\} \\
p^{k+1} &= p^k + c(Mx^{k+1} - z^{k+1})
\end{aligned}}$$

Omitting constants from the minimands,

$$\begin{aligned}
x^{k+1} &\in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ f(x) + \left\langle p^k, Mx \right\rangle + \tfrac{c}{2} \left\| Mx - z^k \right\|^2 \right\} \\
z^{k+1} &\in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{ g(z) - \left\langle p^k, z \right\rangle + \tfrac{c}{2} \left\| Mx^{k+1} - z \right\|^2 \right\} \\
p^{k+1} &= p^k + c(Mx^{k+1} - z^{k+1})
\end{aligned}$$

No Theory Yet

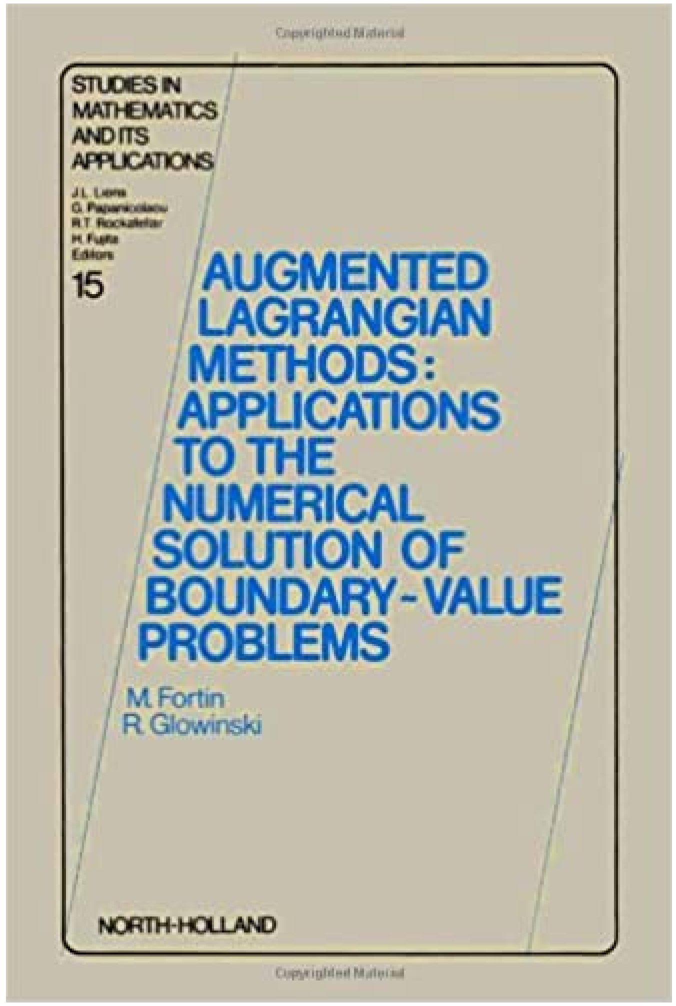
- But they did not have any theory to support this result
- The multiplier update is definitely not dual ascent now, because we are nowhere close to minimizing the augmented Lagrangian

The ADMM Does Not Approximate the ALM

- I have done experiments in which I use alternating minimization for the inner problem until some (rigorous) approximation criterion for the augmented Lagrangian is met, then update the multipliers
- This generally produces far fewer multiplier updates (usually an order of magnitude or so)
- But many orders of magnitude more total inner iterations
- Alternating minimization is in general a poor algorithm for the inner problems
- So how to understand the convergence of the ADMM?

Four Years After Glowinski & Marrocco, Some Theory

- The following edited volume of papers appeared in 1983



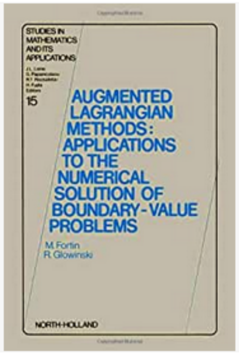
Aside: Amazon's Search Engine

- When I searched for this book on Amazon, I got

Aside: Amazon’s Search Engine

- When I searched for this book on Amazon, I got

augmented lagrangian applications See all 57 results



Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems (Studies in Mathematics and Its Applications, V. 15) (English and French Edition)

French Edition | by Michel Fortin , R. Glowinski, et al. | Jul 1, 1983

Hardcover

Best Seller

Paulas Choice--SKIN PERFECTING 2% BHA Liquid Salicylic Acid Exfoliant--Facial Exfoliant for Blackheads, Enlarged Pores, Wrinkles & Fine Lines, 4 oz Bottle

4 Fl Oz (Pack of 1)

60,366

$32.00 ($8.00/Fl Oz)

$30.40 with Subscribe & Save discount

prime FREE One-Day

Get it **Tomorrow, May 17**

Climate Pledge Friendly

Two Proofs of Convergence in this Book

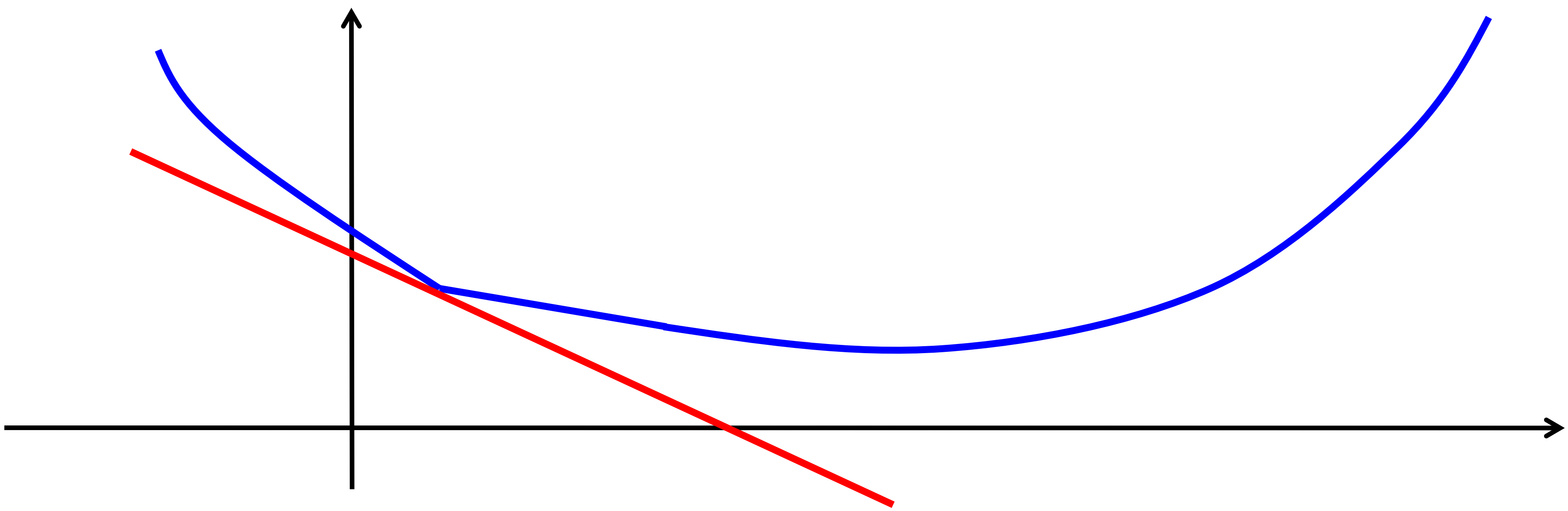
- In Fortin & Glowinski (1983), a convergence proof using a variational inequality analysis
- In Gabay (1983), a proof showing that the ADMM is an operator splitting method
  - The "Douglas-Rachford" splitting method for monotone (set-valued) operators analyzed by Lion and Mercier in 1979
  - Applied to the dual of $\min f(x) + g(Mx)$
  - Operator splitting methods also have their roots in the PDE world – so relatively natural for these researchers to have this insight

- I will follow the Gabay path since it is more intuitive
- The relationship between the two proofs could still use clarification
  - They lead to different forms of over-relaxation

Subgradients of a Convex Function

- Suppose that $d:\mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ is a convex function
- d may not be smooth, but it has subgradients
- $\partial d(x)$ denotes the set of subgradients of d at x :

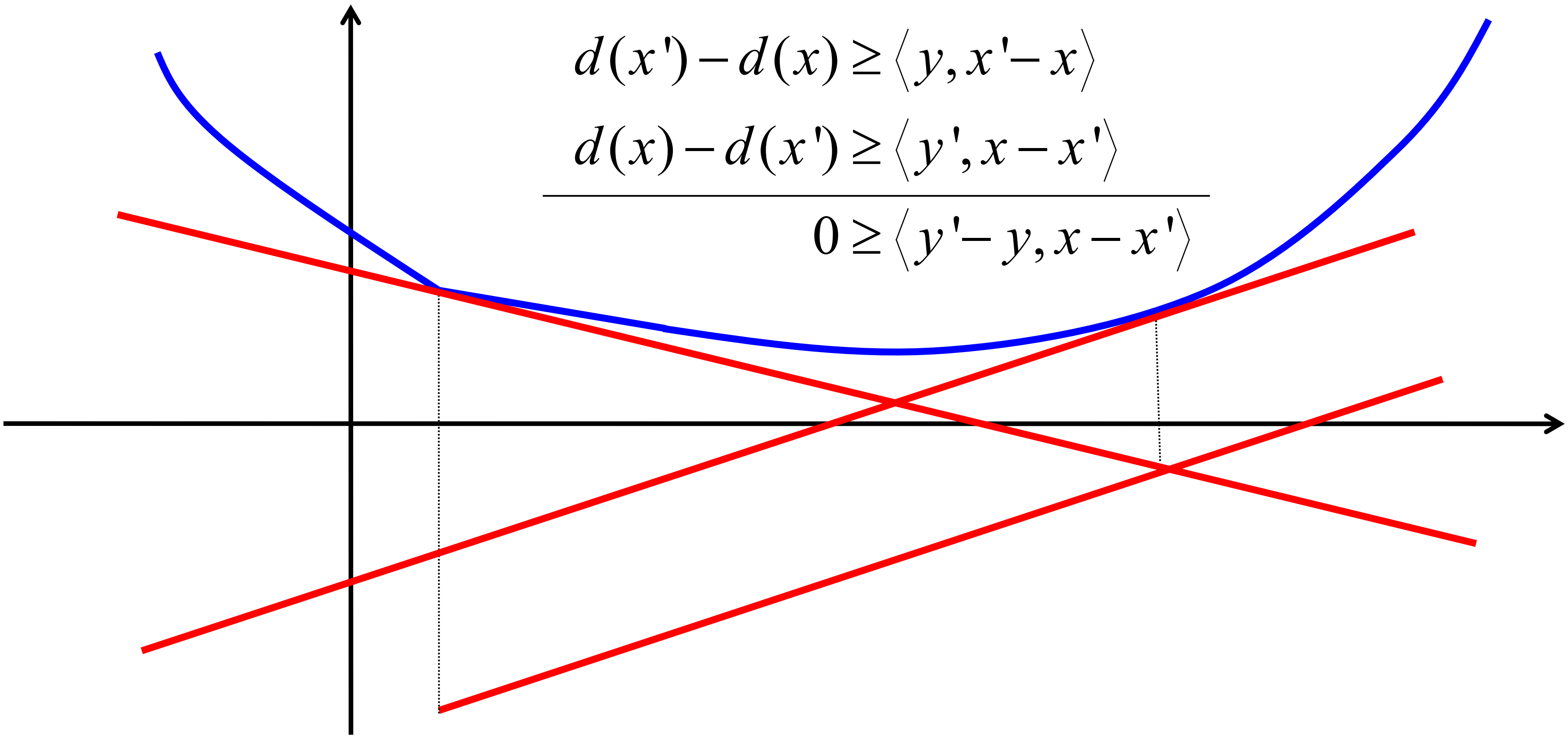
$$\partial d(x) = \left\{ y \;\middle|\; d(x') \geq d(x) + \langle y, x' - x \rangle \ \forall x' \in \mathbb{R}^m \right\}$$



Monotonicity

- Subgradient maps of convex functions are monotone

$$\boxed{y \in \partial d(x),\ y' \in \partial d(x') \quad \Rightarrow \quad \langle x - x', y - y' \rangle \geq 0}$$



$$\begin{aligned} d(x') - d(x) &\geq \langle y, x' - x \rangle \\ d(x) - d(x') &\geq \langle y', x - x' \rangle \\ \hline 0 &\geq \langle y' - y, x - x' \rangle \end{aligned}$$

- This condition is a natural generalization to higher dimension of a function being monotone nondecreasing

The Dual of $\min f(x)+g(Mx)$

- The dual of $\min f(x)+g(Mx)$ can be written in the form $\min_p d_1(p)+d_2(p)$, for two convex functions d_1 and d_2
- Namely, $d_1(p)=f^*(-M^\top p)$ and $d_2(p)=g^*(p)$

$$
\begin{aligned}
d(p) &= \min_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \left\{ L(x,z,p) \right\} \\
&= \min_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \left\{ f(x)+g(z)+\left\langle p, Mx-z \right\rangle \right\} \\
&= \min_{x\in\mathbb{R}^n} \left\{ f(x)+\left\langle M^\top p, x \right\rangle \right\} + \min_{z\in\mathbb{R}^m} \left\{ g(z)-\left\langle p, z \right\rangle \right\} \\
&= \underbrace{f^*(-M^\top p)}_{d_1(p)} + \underbrace{g^*(p)}_{d_2(p)} \\
&= d_1(p)+d_2(p)
\end{aligned}
$$

Splitting the Dual

- This is the same as solving

$$0 \in \partial(d_1 + d_2)(p)$$

- Unless things are really ugly, the same as solving

$$0 \in \partial d_1(p) + \partial d_2(p)$$

where $+$ denotes the Minkowski sum of sets

$$A + B = \{a + b \mid a \in A, b \in B\}$$

Resolvents

- Suppose that T is any point-to-set map on $\mathbb{R}^n$ that is monotone: $y \in T(x),\, y' \in T(x') \quad \Rightarrow \quad \langle x - x', y - y' \rangle \geq 0$
- Consider any fixed scalar $c > 0$
- Then the resolvent of T with stepsize c is $J_{cT} = (I + cT)^{-1}$
- The same operation as an implicit step in ODE/PDE integration
- Conceptually, to evaluate $J_{cT}(r)$:
    - Find x, y such that $x + cy = r$ and $y \in T(x)$ (can only be done one way if T is monotone)
    - Return x

Resolvents and "Reflectants"

- If T is monotone, then the resolvent $J_{cT} = (I + cT)^{-1}$ is defined everywhere, single valued, and firmly nonexpansive

$$(\forall x, x') \quad \|J(x) - J(x')\|^2 \le \|x - x'\|^2 - \|(x - J(x)) - (x' - J(x'))\|^2$$

- And the "reflectant" $R_{cT} = 2J_{cT} - I$ is defined everywhere, single valued, and nonexpansive

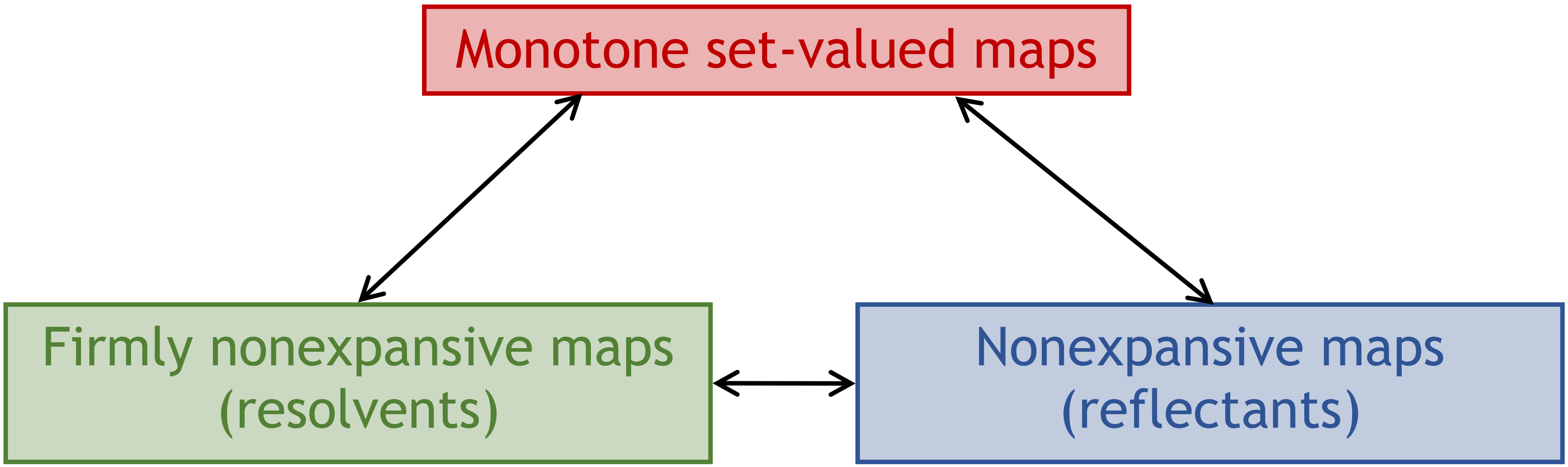
$$(\forall x, x') \quad \|R(x) - R(x')\|^2 \le \|x - x'\|^2$$

- Conceptually, evaluating the reflectant amounts to:
  - Find x, y such that $x + cy = r$ and $y \in T(x)$ (can only be done one way if T is monotone)
  - Return $2x - r = 2x - (x + cy) = x - cy$

$$R_{cT} = 2J_{cT} - I \qquad \Leftrightarrow \qquad J_{cT} = \tfrac{1}{2}R_{cT} + \tfrac{1}{2}I$$

Symmetric Relationships

- The relationships between monotone operators, resolvents, and reflectants are symmetric in all directions



  - A map is firmly nonexpansive if and only if it is the resolvent of some monotone operator
  - A map is nonexpansive if and only if it is the reflectant of some monotone operator
  - A map J is firmly nonexpansive if and only if it is of the form $J = \frac{1}{2}R + \frac{1}{2}I$, where R is nonexpansive

Convergence of the ADMM I

Fundamentally, the convergence theory of the ADMM relies on a very simple observation:

The composition of two nonexpansive mappings is nonexpansive

- Nonexpansive map $R_{c\partial d_1} : \mathbb{R}^m \to \mathbb{R}^m$ corresponding to d_1
- Nonexpansive map $R_{c\partial d_2} : \mathbb{R}^m \to \mathbb{R}^m$ corresponding to d_2
- Their composition $R_{c\partial d_1} \circ R_{c\partial d_2}$ is nonexpansive

Furthermore, the fixed points of $R_{c\partial d_1} \circ R_{c\partial d_2}$ are of the form

$$\left\{ p + cz \,\middle|\, z \in \partial d_2(p), -z \in \partial d_1(p) \right\}$$

Sketch of proof. $p + cz \overset{N_{c\partial d_2}}{\mapsto} p - cz \overset{N_{c\partial d_1}}{\mapsto} p + cz$. Easy to show this is the only possibility (two equations in two unknowns). $\square$

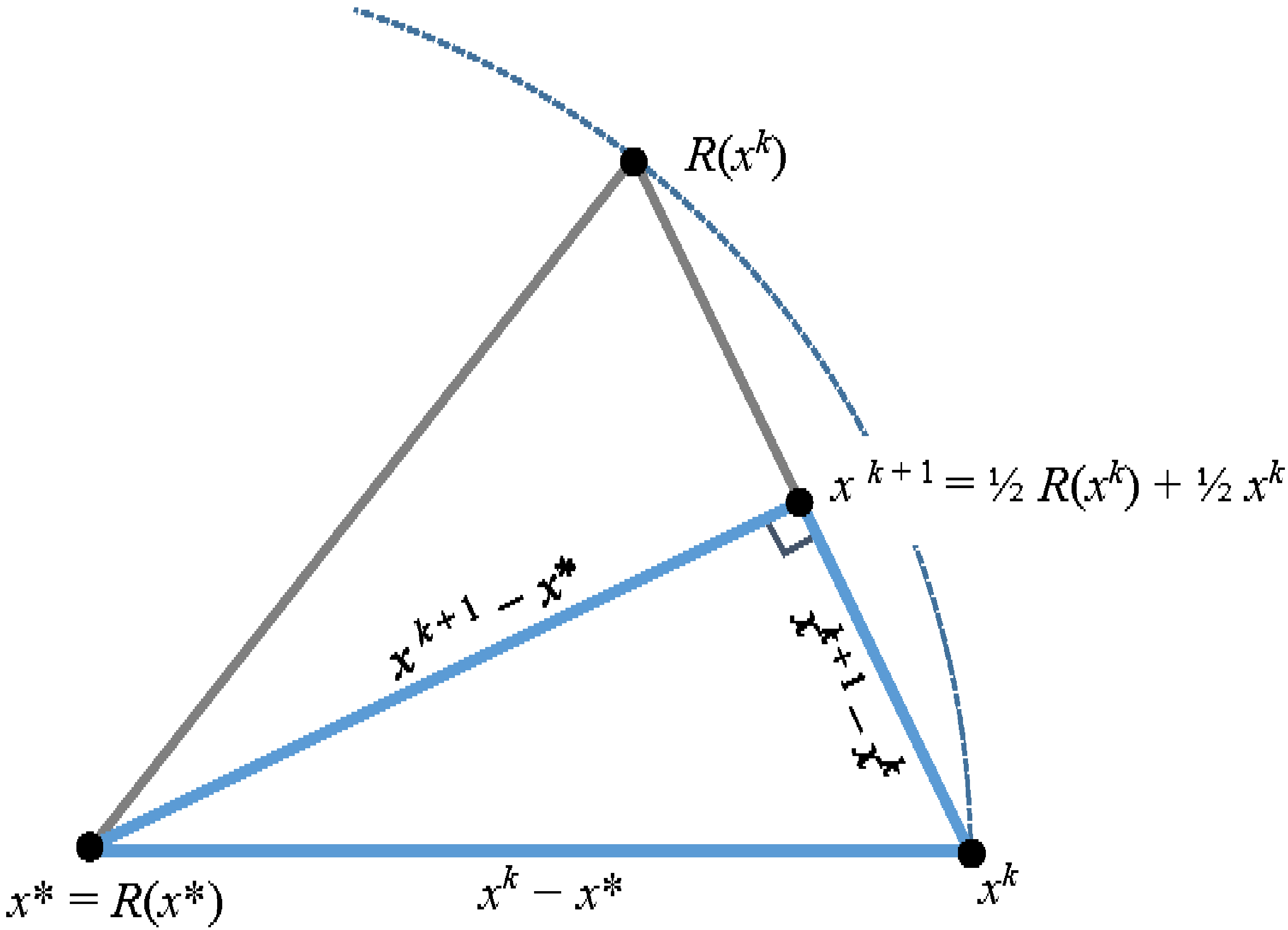
Convergence of the ADMM II

- From a fixed point t of $R_{c\partial d_1} \circ R_{c\partial d_2}$, we can find the optimal dual solution by just applying $J_{c\partial d_2}(t)$, and we can also easily find the primal solution
- It would be nice to just iterate the map $R_{c\partial d_1} \circ R_{c\partial d_2}$ to converge to a fixed point, but since its Lipschitz constant is 1, this process might just "orbit" around the set of fixed points
- But (Krasnosel'skii 1955), if we blend it with the identity, it will converge

$$s^{k+1} = \tfrac{1}{2} s^k + \tfrac{1}{2}\left(R_{c\partial d_1} \circ R_{c\partial d_2}\right)(s^k) = \tfrac{1}{2} s^k + \tfrac{1}{2} R_{c\partial d_1}\left(R_{c\partial d_2}(s^k)\right)$$

- This is the essence of "Douglas-Rachford splitting"
- Converts the nonexpansive map $R_{c\partial d_1} \circ R_{c\partial d_2}$ to a firmly nonexpansive one (with the same fixed points)

Picture of Krasnosel'skii



What is Meant by "Operator Splitting"?

- Douglas-Rachford splitting is a kind of operator splitting
- We are solving the problem $\min_p d_1(p) + d_2(p)$
- Or (usually) equivalently $0 \in \partial d_1(p) + \partial d_2(p)$
- But we only deal with the individual reflectant maps $R_{c\partial d_1}$ and $R_{c\partial d_2}$ respectively associated with with d_1 and d_2
- That's the essence of operator splitting

Getting the History Right

- Douglas and Rachford had a different representation of the operations in their method, but equivalent
- However, the original Douglas-Rachford publication was only for linear operators
  - Applied to very specific linear operators related to the discretized 2-D heat equation
- Lions and Mercier (1979) generalized the idea from linear maps to general monotone set-valued maps (but kept the name)
- Gabay (1983) showed that the ADMM is just this idea applied to the dual of $\min f(x) + g(Mx)$
- The composition-of-nonexpansive maps interpretation may first be found (as an aside) in Lawrence & Spingarn (1987)
- E & Bertsekas (1992) contains some equivalent analysis and exploits the relationship with the proximal point algorithm to derive approximate and over-relaxed versions

Some Insights from the Convergence Analysis

- ADMM convergence based on evaluation of $R_{c\partial d_1} \circ R_{c\partial d_2}$, which is not an approximation of of $R_{c(\partial d_1 + \partial d_2)}$ (the mapping for ALM)

- Unlike the ALM, changing c in the ADMM is problematic because it shifts the set of fixed points $\left\{ p + cv \,\middle|\, v \in \partial d_2(p), -v \in \partial d_1(p) \right\}$ of $R_{c\partial d_1} \circ R_{c\partial d_2}$
  - There are results for variable c, but they need extra assumptions and get "messy"

- Also, the $R_{c\partial d_1} \circ R_{c\partial d_2}$ convergence theory of the ADMM does not have a "clean" extension to more than two blocks:

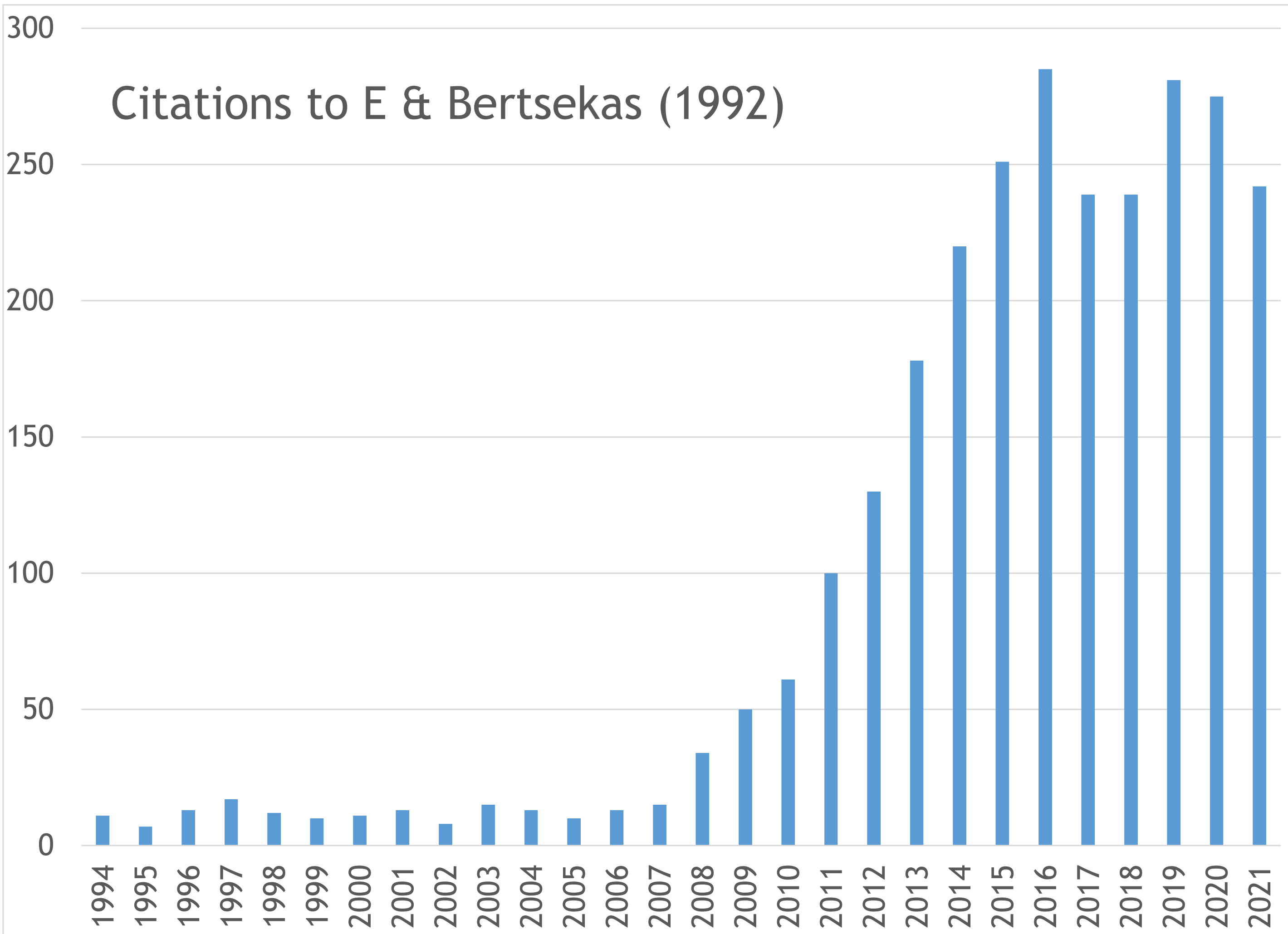
$$\min_{x \in \mathbb{R}^n} \left\{ f_1(M_1 x_1) + f_2(M_2 x_2) + \cdots + f_p(M_p x_p) \right\}$$

$R_{c\partial d_1} \circ R_{c\partial d_2} \circ \cdots \circ R_{c\partial d_p}$ does not have "nice" fixed points

  - Must use a product-space reformulation, or make things "messier"

The ADMM: More Past

- The standard theory of the ADMM was settled by the early 90’s
- But it remained an obscure algorithm for 15+ years
- During the period 2008-2014, things changed



The ADMM: Present

- Now the ADMM is considered part of the standard optimization "toolbox"

Some typical current applications:

- Image denoising
- Data fitting / machine learning
  - Along with other operator-splitting methods, like forward-backward
- Stochastic programming (progressive hedging)
- …
- Even some general conic QP solvers

- Also, a dizzying profusion of new variants (not covered much here)

Features of Successful Applications

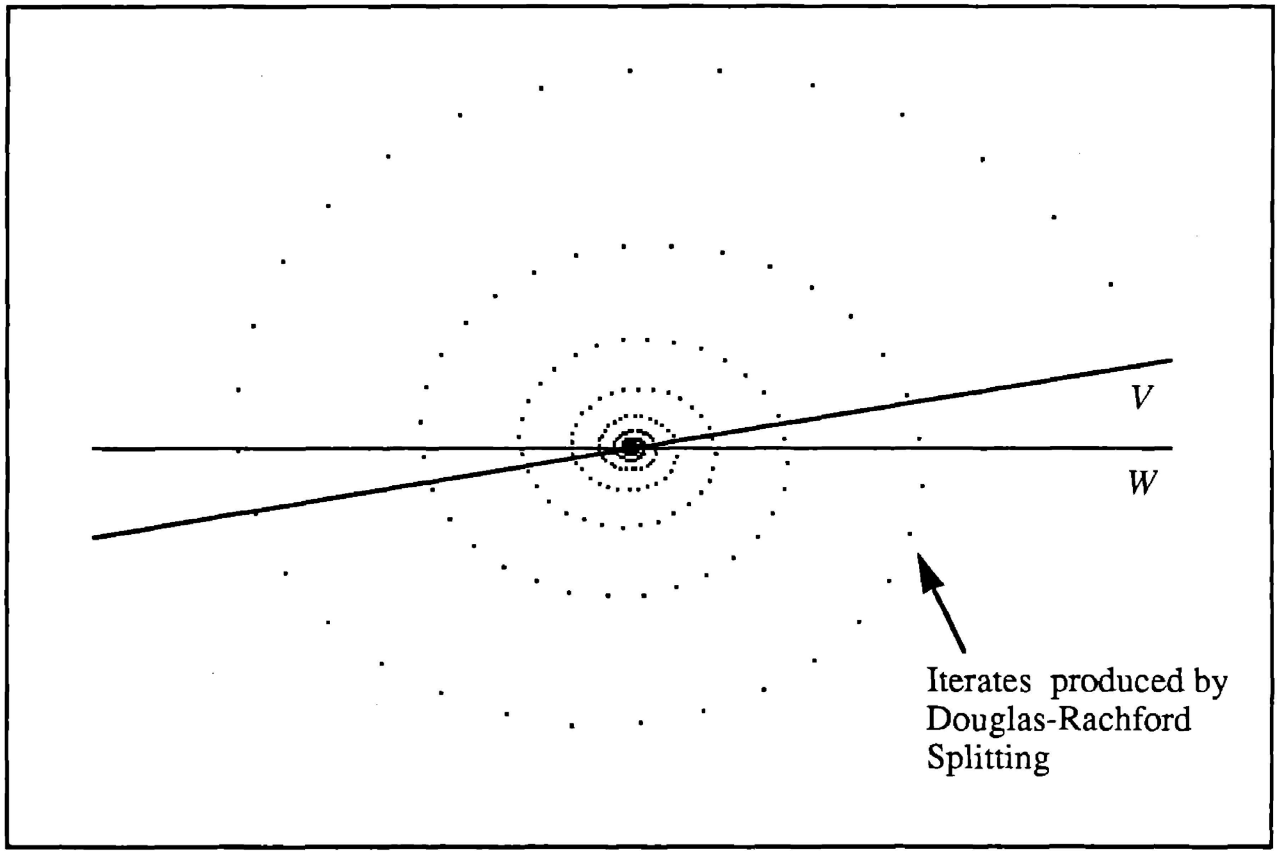
1. Low accuracy solutions are sufficient
   (fairly common knowledge)

2. Nonlinear convex objectives can often work better than linear ones

3. Do not try to "atomize" problems
   (although that can be tempting for parallelism)

Low Accuracy Requirement

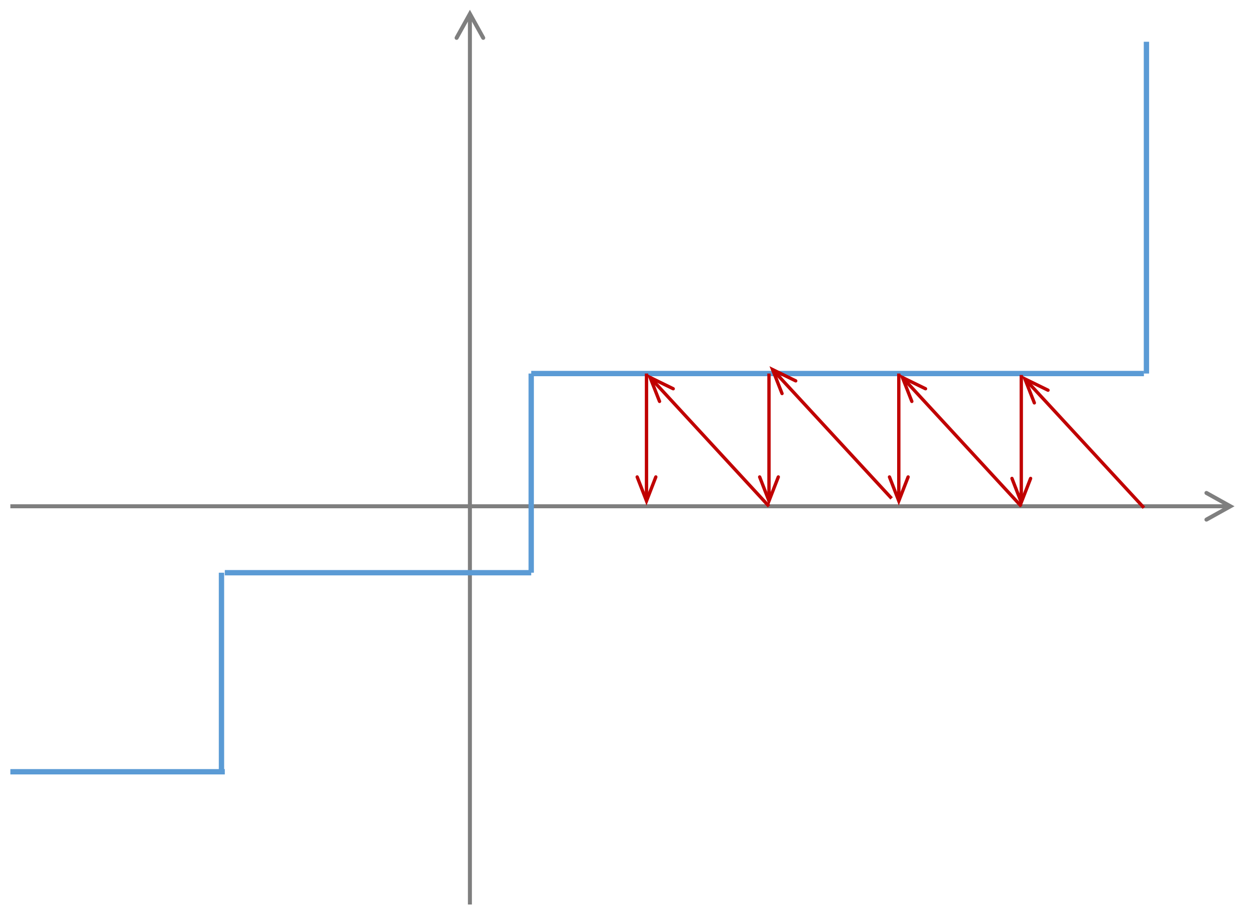
- The ADMM does not have very fast asymptotic / tail convergence
- It is typically linear/geometric, but the constant can be poor



- However, applications like machine learning and image denoising typically don't require high-accuracy solutions

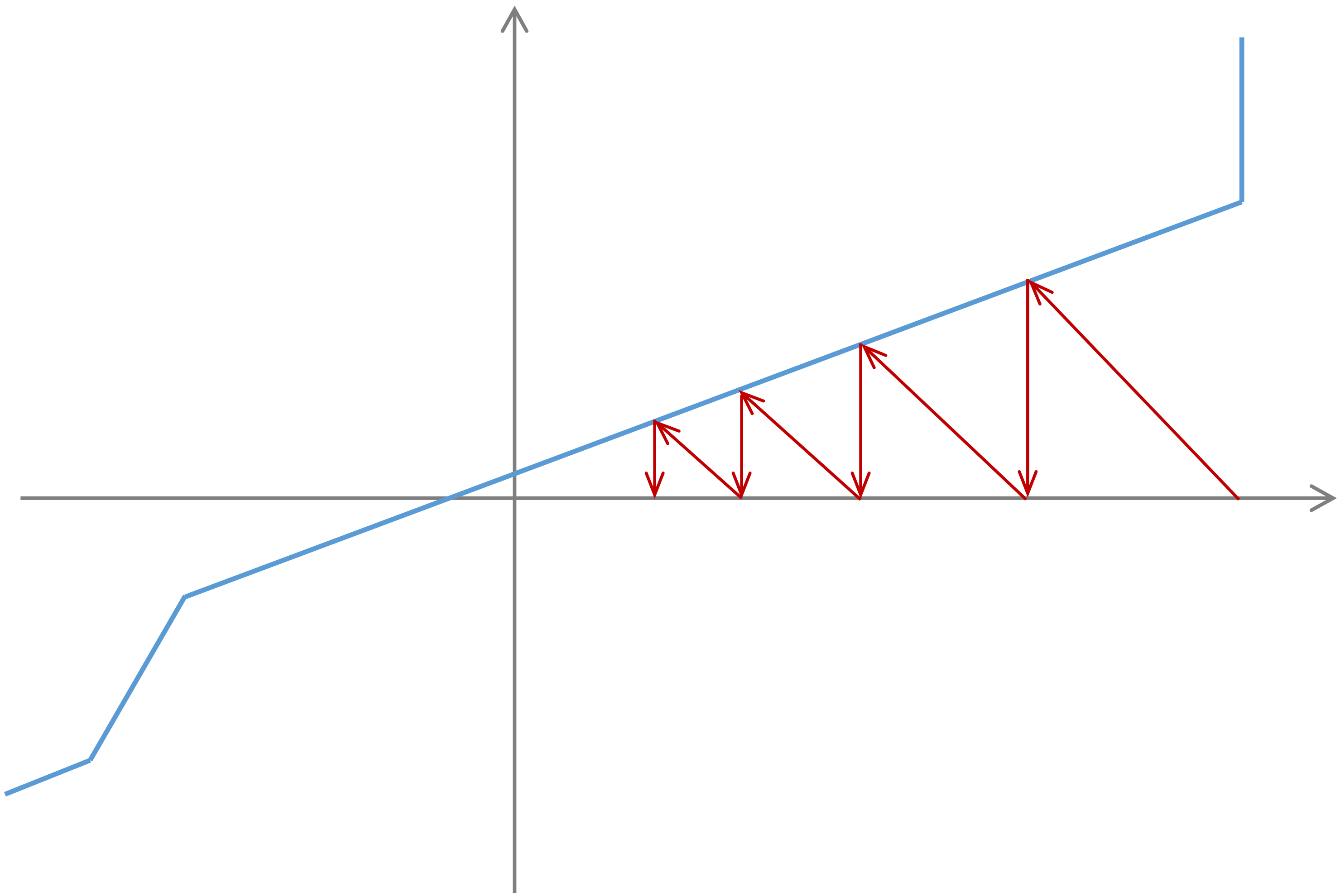
Nonlinear Objectives (Intuition)

- The ADMM is a form of the proximal point algorithm (PPA) (shown for example in E & Bertsekas 1992)
- Solving an LP-like problem with the PPA



Nonlinear Objectives (Intuition)

- Solving a QP-like problem with the PPA



Don't Atomize Problems

In successful ADMM applications…

- At least one side of the splitting (f or g) should model a substantial portion of the global interconnections between problem elements
- My former postdoc Patrick Johnstone calls this property "being meaty"
- What does that mean?

Don't Atomize Problems II

- Example: in data fitting / ML problems, one often has a structure like

$$\min_x \ell(x) + r(x)$$

- ℓ is a smooth loss function
- r is a regularizer (for example, an L_1 penalty)

- All the connections between the model parameters x and fitting the observations are contained in ℓ
- So, ℓ is "meaty" (it contains essentially all the connections between model elements) and one can set $f = \ell, g = r, M = I$

Don't Atomize Problems III

In OSQP (Stellato *et al*. 2020)

- f models all the linear relationships within the model
- g contains only conic constraints on individual vectors
- So, f is "meaty"

In E & Ferris 1998 for optimal control problems

- f enforces a block-tridiagonal linear system capturing all the time dynamics in the model
- g enforces all the inequalities and nonsmooth elements (confined within each time step)
- So, f is "meaty"

Don't Atomize Models IV

In progressive hedging for stochastic programming problems (originally Rockafellar and Wets 1991)

- f contains the entire time dynamics within each scenario
- g enforces "nonanticipativity" (not seeing the future) relationships between scenarios
- So, they are both fairly "meaty"

A Tempting Example of "Non-Meatiness": The Classic "Transportation" Problem

Given a bipartite graph (S, D, E),

$$\boxed{\begin{array}{lll} \min & r^\top x & \\ \text{ST} & \displaystyle\sum_{j:(i,j)\in E} x_{ij} = s_i & \forall i \in S \\ & \displaystyle\sum_{i:(i,j)\in E} x_{ij} = d_j & \forall i \in D \\ & x_{ij} \geq 0 & \forall (i,j) \in E \end{array}}$$

- Illustration of how the ADMM can lead to highly parallel algorithms
- But ones that are typically not competitively efficient

Modeling Transportation in the ADMM Form

With $x, z \in \mathbb{R}^{|E|}$,

$$f(x) = \begin{cases} \frac{1}{2} r^\top x, & \text{if } x \ge 0 \text{ and } \sum_{j:(i,j)\in E} x_{ij} = s_i \quad \forall i \in S \\ +\infty & \text{otherwise} \end{cases}$$

$$g(z) = \begin{cases} \frac{1}{2} r^\top z, & \text{if } z \ge 0 \text{ and } \sum_{i:(i,j)\in E} z_{ij} = d_j \quad \forall i \in D \\ +\infty & \text{otherwise} \end{cases}$$

Then the problem reduces to

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

- The x-minimization step separates by source node $i \in S$
- The z-minimization step separates by destination node $j \in D$

ADMM for Transportation

For example, the x minimization reduces to, for each $i \in S$

$$
\begin{aligned}
\min \quad & \sum_{j:(i,j)\in E} r_{ij}x_{ij} + \sum_{j:(i,j)\in E} p_{ij}x_{ij} + \tfrac{c}{2}\sum_{j:(i,j)\in E} (x_{ij} - z_{ij})^2 \\
\text{ST} \quad & \sum_{j:(i,j)\in E} x_{ij} = s_i \\
& x_{ij} \geq 0 \qquad\qquad \forall j:(i,j)\in E
\end{aligned}
$$

This is just projection on a simplex, so it's an easy problem

- A simple implementation is $\mathrm{O}(\delta \log \delta)$, where δ is the node degree
- Can be done in $\mathrm{O}(\delta)$ time if one is careful (related to linear-time median finding; only matters for large δ)

- The z minimization step is similar

ADMM for Transportation — (Parallel) Implementation

High potential for parallelism:

- x minimization consists of $|S|$ independent, easy tasks
- z minimization consists of $|D|$ independent, easy tasks
- Multiplier update consists of $|E|$ independent, easy tasks
- Simple communication pattern between these tasks
- An example of how ADMM can lead to highly parallel algorithms
- Studied these kinds of applications in my dissertation

- Unfortunately, it's very slow compared to network simplex etc.
- And parallelism is not enough to save it

Slowness Intuition and the Moral of the Story

- Both sides of the splitting decompose into optimizations that only “see” individual nodes
- The whole “big picture” is left to the ADMM to coordinate
- But ADMM / DR is not an outstanding linear equation solver
- So it takes a long time for all the pieces of the problem to come into alignment

- ADMM is a useful algorithm
- But don’t ask too much of it

- Don’t leave the entire coordination of small problem elements to the ADMM
- Keep at least some of the global connections within f or g

Approximate Iterations

- With “meatiness” of subproblems comes the need to solve them inexactly
- E & Bertsekas 1992 was the first publication to rigorously cover inexact solution of subproblems
- But requires bounding the distance between the approximate iterate and the exact one
- In general, finding such a bound can be difficult
- I have repeatedly seen people using only the distance-based 1992 approximation result, not realizing that there is more recent work on the subject

Better Approximation Criteria

Two papers:

- E & Yao 2017 (COAP)
- E & Yao 2018 (*Math Programming* A)

- Neither require estimating the distance to the exact subproblem solution
- So, easier to implement in general than the 1992 criterion
- Absolute error criteria involve a summable sequence of allowable errors that is (formally) an external parameter
- Relative error criteria use a single parameter to compare two quantities generated by the algorithm
  - One of which would be zero in the exact case
- The above two papers contain both kinds
- If you are using inexact ADMM, please look at these papers!

The ADMM: Future

- What can are the directions for the future?

1. Other operator splitting methods might make and impact

2. Upper and lower bounds

3. A wider range of applications, if we can solve the tail convergence issues

Other Operator Splitting Methods

- For a long while, there were basically three classes of operator splitting method
  - Forward-backward (generalizes gradient projection)
  - Douglas-Rachford (as in the ADMM), which forms a continuous family with Peaceman-Rachford
  - Double-backward: solves $\min_p d_1(p) + \frac{c}{2}\|p-q\|^2 + d_2(q)$ instead of $\min_p d_1(p) + d_2(p)$
- Once the math programming and machine learning communities got interested in operator splitting, new varieties started appearing
  - Forward-backward-forward (Tseng 200)
  - Projective splitting (starting with E & Svaiter 2008)
  - Forward-reflected-backward (Malitsky & Tam 2020) …

Other Splitting Methods

- The picture is not clear yet
- There are so many methods and variations now
- And so many problems to apply them too
- But there are specific cases in which new operator splitting methods can outperform the ADMM

An Example

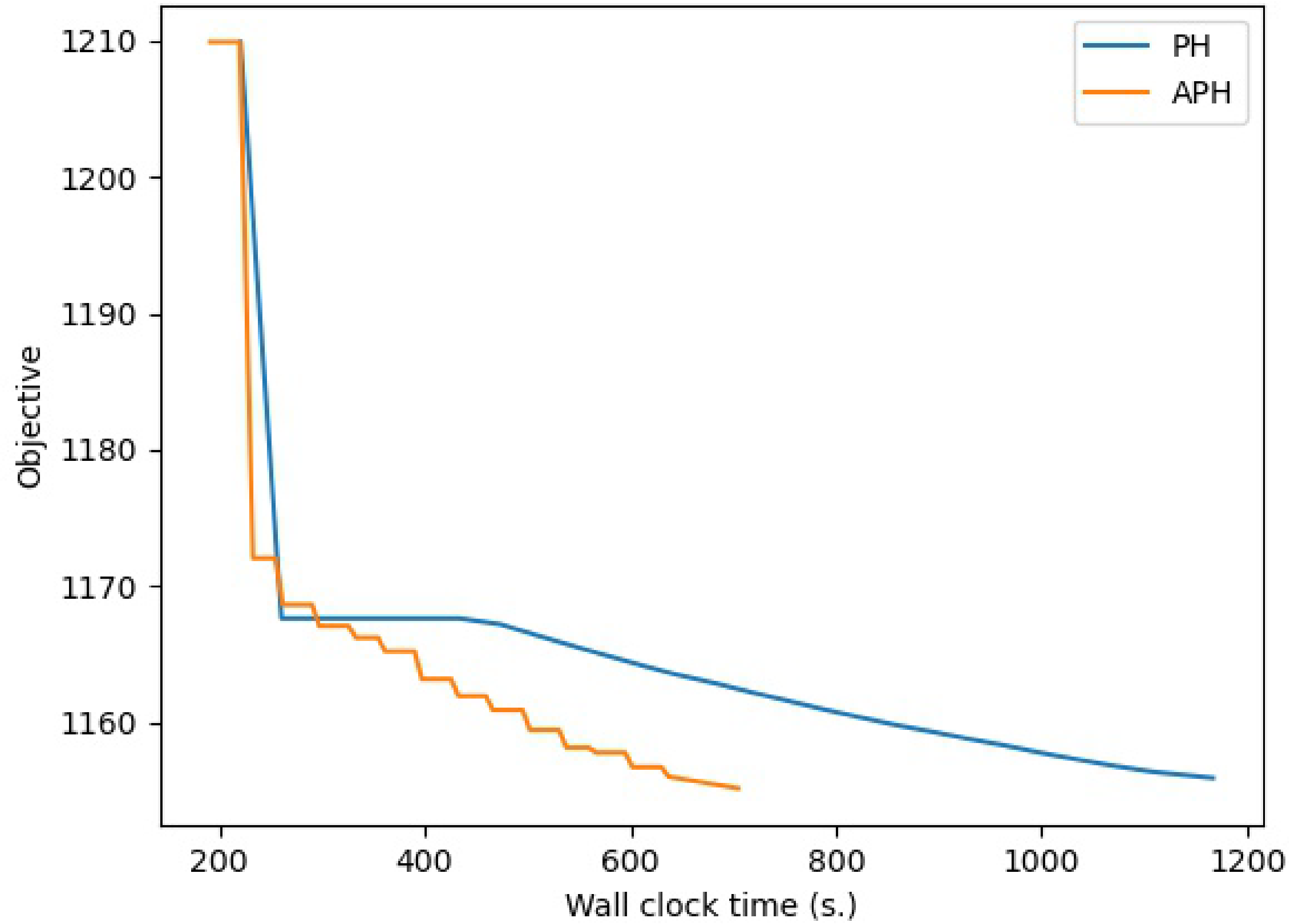
Progressive hedging (PH) for stochastic programming:

- The x minimization separately optimizes a quadratic perturbation of each scenario
- The z minimization and multiplier update try to make your overall strategy non-clairvoyant

“Projective Hedging”

- Apply an “asynchronous” projective splitting variant (Combettes and E 2018, E 2017) to the same problem
- Obtain a similar algorithm to PH, except
  - You don’t have to optimize every scenario at every iteration
  - You can just optimize a subset
  - Called asynchronous projective hedging (APH)
- Generally, if you don’t re-optimize all the scenarios
  - The convergence slows down somewhat
  - But each iteration takes much less time
  - So overall time may be reduced, as in…

An Example Stochastic Programming Problem



- 1,000,000 scenarios! (And 5 stages)
- 600 processor cores
- Here, APH only solves 10% of the scenarios at each iteration

Upper and Lower Bounds

- ADMM, like many other operator-splitting methods…
- Converges asymptotically to solution (both primal and dual)
- But is typically never feasible: $x = z$ only in the limit
  - So in general there are no upper bounds
  - Although in ML problems everything is usually feasible, so upper bounds are easy
- And doesn’t provide lower bounds
  - Since it never truly minimizes the (augmented) Lagrangian
- But if you’re solving problems to low accuracy, you would often like to have upper and lower bounds
- Workarounds are generally application-specific
  - E 2020 gives a possible lower bound when all else fails…
- It would be nice to have a systematic approach

Tail Convergence

- Slow tail convergence is probably the biggest issue with ADMM-class methods
- There are "accelerated" versions
- But these often address the global rates: $O(1/k)$ etc.
- Whereas the real issue is tends to be slow linear/geometric asymptotic tail
  - Asymptotically much faster than $O(1/k)$, $O(1/k^2)$, etc.
  - But still too slow
- Sometimes one can periodically test and try to "jump" to a basic solution (in simplex terms, or some generalization); see for example E & Ferris 1998
- But that's not very satisfactory in general
- If we could speed up the tail, we could see a lot more applications (Patrinos etc. are working on this topic)

Thank you once again!