More First-Order Algorithms: Dual, Primal-Dual and "1.5-Order" Acceleration

Yinyu Ye

http://www.stanford.edu/~yyye

(Chapters 8, 9, 14, 15)

The Lagrangian Method with Multipliers (LMM)

We consider

$$f^* := \min_{\mathbf{x}} \quad f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{x} \in X. \tag{1}$$

Recall the Lagrangian function:

$$L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{h}(\mathbf{x}).$$

and the dual function:

$$\phi(\mathbf{y}) := \min_{\mathbf{x} \in X} L(\mathbf{x}, \mathbf{y}); \tag{2}$$

and the dual problem

$$(f^* \geq) \phi^* := \max_{\mathbf{y}} \quad \phi(\mathbf{y}). \tag{3}$$

In many cases, one can find $\mathbf{y}^*$ of dual problem (3), a unconstrained optimization problem, then go ahead to find $\mathbf{x}^*$ using (2).

Although this would be guaranteed for only convex optimization, one can always construct a local convex region constraint in (2) around a local minimizer so that the approach still works.

The Gradient and Hessian of ϕ

Let $\mathbf{x}(\mathbf{y})$ be a minimizer of (2). Then

$$\phi(\mathbf{y}) = f(\mathbf{x}(\mathbf{y})) - \mathbf{y}^T \mathbf{h}(\mathbf{x}(\mathbf{y}))$$

Thus,

$$\begin{aligned} \nabla\phi(\mathbf{y}) &= \nabla f(\mathbf{x}(\mathbf{y}))^T \nabla \mathbf{x}(\mathbf{y}) - \mathbf{y}^T \nabla \mathbf{h}(\mathbf{x}(\mathbf{y})) \nabla \mathbf{x}(\mathbf{y}) - \mathbf{h}(\mathbf{x}(\mathbf{y})) \\ &= (\nabla f(\mathbf{x}(\mathbf{y}))^T - \mathbf{y}^T \nabla \mathbf{h}(\mathbf{x}(\mathbf{y}))) \nabla \mathbf{x}(\mathbf{y}) - \mathbf{h}(\mathbf{x}(\mathbf{y})) \\ &= -\mathbf{h}(\mathbf{x}(\mathbf{y})). \end{aligned}$$

Similarly, we can derive

$$\nabla^2\phi(\mathbf{y}) = -\nabla \mathbf{h}(\mathbf{x}(\mathbf{y})) \left(\nabla^2_{\mathbf{x}} L(\mathbf{x}(\mathbf{y}), \mathbf{y})\right)^{-1} \nabla \mathbf{h}(\mathbf{x}(\mathbf{y}))^T,$$

where $\nabla^2_{\mathbf{x}} L(\mathbf{x}(\mathbf{y}), \mathbf{y})$ is the Hessian of the Lagrangian function that is assumed to be positive definite at any (local) minimizer.

A Toy Example

minimize $(x_1 - 1)^2 + (x_2 - 1)^2$

subject to $x_1 + 2x_2 - 1 = 0, \quad 2x_1 + x_2 - 1 = 0.$

$$L(\mathbf{x}, \mathbf{y}) = (x_1 - 1)^2 + (x_2 - 1)^2 - y_1(x_1 + 2x_2 - 1) - y_2(2x_1 + x_2 - 1).$$

$$x_1 = 0.5y_1 + y_2 + 1, \quad x_2 = y_1 + 0.5y_2 + 1.$$

$$\phi(\mathbf{y}) = -1.25y_1^2 - 1.25y_2^2 - 2y_1y_2 - 2y_1 - 2y_2.$$

$$\nabla\phi(\mathbf{y}) = \begin{pmatrix} 2.5y_1 + 2y_2 + 2 \\ 2y_1 + 2.5y_2 1 + 2 \end{pmatrix},$$

$$\nabla^2\phi(\mathbf{y}) = -\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}^{-1}\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}^T = -\begin{pmatrix} 2.5 & 2 \\ 2 & 2.5 \end{pmatrix}$$

A Fisher Market Example

$$\text{minimize} \quad -5\log(2x_1 + x_2) - 8\log(3x_3 + x_4)$$

$$\text{subject to} \quad x_1 + x_3 = 1, \quad x_2 + x_4 = 1, \quad \mathbf{x} \geq \mathbf{0}.$$

$$L(\mathbf{x}(\geq \mathbf{0}), \mathbf{y}) = -5\log(2x_1 + x_2) - 8\log(3x_3 + x_4) - y_1(x_1 + x_3 - 1) - y_2(x_2 + x_4 - 1).$$

Start from $\mathbf{y}^0 > \mathbf{0}$, at the kth step, compute $\mathbf{x}^{k+1}$ from

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} L(\mathbf{x}(\geq \mathbf{0}), \mathbf{y}^k),$$

then let

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \frac{1}{\beta}(A\mathbf{x}^{k+1} - \mathbf{b}).$$

(FisherexampleLMM.m of Chapter 14)

The Augmented Lagrangian Function

In both theory and practice, we often consider an Augmented Lagrangian function (ALF)

$$L_a(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{h}(\mathbf{x}) + \frac{\beta}{2} \|\mathbf{h}(\mathbf{x})\|^2,$$

which corresponds to an equivalent problem of (1):

$$f^* := \min \quad f(\mathbf{x}) + \tfrac{\beta}{2} \|\mathbf{h}(\mathbf{x})\|^2 \quad \text{s.t.} \quad \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{x} \in X.$$

where a quadratic penalty term on constraint violation is added, besides the original linear penalty term.

Note that, although at feasibility the additional square term in objective is redundant, it helps to improve convexity of the Lagrangian function

For the Fisher example, the function is

$$\begin{aligned} & L_a(\mathbf{x}(\geq \mathbf{0}), \mathbf{y}) \\ = \quad & -5 \log(2x_1 + x_2) - 8 \log(3x_3 + x_4) - y_1(x_1 + x_3 - 1) - y_2(x_2 + x_4 - 1) \\ & + \tfrac{\beta}{2}((x_1 + x_3 - 1)^2 + (x_2 + x_4 - 1)^2). \end{aligned}$$

The Augmented Lagrangian Dual

Now the dual function:

$$\phi_a(\mathbf{y}) = \min_{\mathbf{x}\in X} L_a(\mathbf{x}, \mathbf{y}); \tag{4}$$

and the dual problem

$$(f^* \geq)\phi_a^* := \max \quad \phi_a(\mathbf{y}). \tag{5}$$

Note that the dual function approximately satisfies $\frac{1}{\beta}$-Lipschitz condition (see Chapter 14 of L&Y).

For the convex optimization case, say $\mathbf{h}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$, we have

$$\nabla^2 L_a(\mathbf{x}, \mathbf{y}) = \nabla^2 f(\mathbf{x}) + \beta(A^T A).$$

Therefore, the Hessian of the dual would be more likely be Positive Definite so that the convergence speed may be accelerated.

The Augmented Lagrangian Method with Multipliers (ALMM)

Start from any $(\mathbf{x}^0 \in X, \mathbf{y}^0)$, we compute a new iterate pair

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in X} L_a(\mathbf{x}, \mathbf{y}^k), \text{ and } \mathbf{y}^{k+1} = \mathbf{y}^k - \beta \mathbf{h}(\mathbf{x}^{k+1}).$$

The computation of $\mathbf{x}$ is used to calculate the gradient vector of $\phi_a(\mathbf{y})$, which is a steepest ascent direction.

The method converges just like the SDM, because the dual function satisfies $\frac{1}{\beta}$-Lipschitz condition.

Other SDM strategies may be adapted to update $\mathbf{y}$ (the BB, ASDM, Conjugate, Quasi-Newton ...).

One drawback of ALMM is computing of $\mathbf{x}$ can be much more complicated!

Alternating Direction Method with Multipliers (ADMM) with Two-Blocks

Here we consider structured problem

$$\min \quad f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) \quad \text{s.t.} \quad A_1\mathbf{x}_1 + A_2\mathbf{x}_2 = \mathbf{b},\ \mathbf{x}_1 \in X_1,\ \mathbf{x}_2 \in X_2.$$

Consider

$$L(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) - \mathbf{y}^T(A_1\mathbf{x}_1 + A_2\mathbf{x}_2 - \mathbf{b}) + \frac{\beta}{2}\|A_1\mathbf{x}_1 + A_2\mathbf{x}_2 - \mathbf{b}\|^2.$$

Then, for any given $(\mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{y}^k)$, we compute a new iterate

$$\begin{aligned}
\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1 \in X_1} L(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{y}^k), \\
\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2 \in X_2} L(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{y}^k), \\
\mathbf{y}^{k+1} &= \mathbf{y}^k - \beta(A_1\mathbf{x}_1^{k+1} + A_2\mathbf{x}_2^{k+1} - \mathbf{b}).
\end{aligned}$$

Again, we can prove that the iterates converge with the same speed for any convex objectives.

The ADMM method resembles the Block Coordinate Descent Method (BCD) in computing $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2)$, which produces an inexact dual gradient. However, the separate computation of $(\mathbf{x}_1; \mathbf{x}_2)$ would be much easier!

Direct Application of ADMM to Dual Linear Programming I

Consider the dual LP

$$\begin{array}{ll} \text{maximize}_{(\mathbf{y},\mathbf{s})} & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} & A^T\mathbf{y}+\mathbf{s}=\mathbf{c},\ \mathbf{s}\geq\mathbf{0}. \end{array}$$

The augmented Lagrangian function would be

$$L(\mathbf{y},\mathbf{s},\mathbf{x}) = -\mathbf{b}^T\mathbf{y} - \mathbf{x}^T(A^T\mathbf{y}+\mathbf{s}-\mathbf{c}) + \frac{\beta}{2}\|A^T\mathbf{y}+\mathbf{s}-\mathbf{c}\|^2,$$

where β is a positive parameter, and $\mathbf{x}$ is the multiplier vector.

Direct Application of ADMM to Dual Linear Programming II

The ADMM for the dual is straightforward: starting from any $\mathbf{y}^0$, $\mathbf{s}^0 \geq \mathbf{0}$, and multiplier $\mathbf{x}^0$,

- Update variable $\mathbf{y}$:

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} L(\mathbf{y}, \mathbf{s}^k, \mathbf{x}^k);$$

- Update slack variable $\mathbf{s}$:

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\geq\mathbf{0}} L(\mathbf{y}^{k+1}, \mathbf{s}, \mathbf{x}^k);$$

- Update multipliers $\mathbf{x}$:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \beta(A^T\mathbf{y}^{k+1} + \mathbf{s}^{k+1} - \mathbf{c}).$$

Note that the updates of $\mathbf{y}$ is a least-squares problem with constant matrix, and the update of $\mathbf{s}$ has a simple close form. (Also note that $\mathbf{x}$ would be non-positive at the end, since we changed maximization to minimization of the dual.) (ADMMDualLP.m of Chapter 14)

To split $\mathbf{y}$ into multi blocks and update cyclically in random order?

ADMM for Solving the Fisher Example

$$\begin{array}{ll} \text{minimize} & -5\log(2x_1+x_2)-8\log(3x_3+x_3) \\ \text{subject to} & x_1+x_3=1, \quad x_2+x_4=1, \quad \mathbf{x}\geq \mathbf{0}. \end{array}$$

$$\begin{array}{ll} \text{minimize} & -5\log(u_1)-8\log(u_2) \\ \text{subject to} & x_1+x_3-1=0, \quad x_2+x_4-1=0, \\ & 2x_1+x_2-u_1=0, \quad 3x_3+x_4-u_2=0, \\ & \mathbf{x}-\mathbf{s}=\mathbf{0}, \quad \mathbf{s}\geq\mathbf{0}. \end{array}$$

$$L(\mathbf{x},\mathbf{u},\mathbf{s}(\geq\mathbf{0}),\mathbf{y}) = -5\log(u_1)-8\log(u_2)-y_1(x_1+x_3-1)-y_2(x_2+x_4-1)$$
$$-y_3(2x_1+x_2-u_1)-y_4(3x_3+x_4-u_2)-\mathbf{y}_{5:8}^T(\mathbf{x}-\mathbf{s})+$$
$$\frac{\beta}{2}[(x_1+x_3-1)^2+(x_2+x_4-1)^2+(2x_1+x_2-u_1)^2+(3x_3+x_4-u_2)^2+\|\mathbf{x}-\mathbf{s}\|^2].$$

Let the first block primal variables be $\mathbf{x}$ and the second be $(\mathbf{u},\mathbf{s})$. Then start from $\mathbf{y}^0$ repeat the ADMM steps. Note that all primal variables have close-form solutions (FisherexanpleADMM.m of Chapter 14).

ADMM for SNL

Recall that SNL can be represented as a quartic polynomial minimization and it is a nonconvex problem.

Applying the variable-splitting, it becomes constrained bi-convex minimization problem

$$\begin{array}{ll} \min_{\mathbf{x}_i,\mathbf{z}_i} & \sum_{(i,j)\in N_x}((\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{z}_i-\mathbf{z}_j)-d_{ij}^2)^2+\sum_{(k,j)\in N_a}((\mathbf{a}_k-\mathbf{x}_j)^T(\mathbf{a}_k-\mathbf{z}_j)-\hat{d}_{kj}^2)^2 \\ \text{s.t.} & \mathbf{x}_i=\mathbf{z}_i,\ \forall i. \end{array}$$

The augmented Lagrangian function would be

$$\begin{array}{rl} & L_a(\mathbf{x}_i,\mathbf{z}_i,\mathbf{y}_i) \\ = & \sum_{(i,j)\in N_x}((\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{z}_i-\mathbf{z}_j)-d_{ij}^2)^2+\sum_{(k,j)\in N_a}((\mathbf{a}_k-\mathbf{x}_j)^T(\mathbf{a}_k-\mathbf{z}_j)-\hat{d}_{kj}^2)^2 \\ & -\sum_i \mathbf{y}_i^T(\mathbf{x}_i-\mathbf{z}_i)+\frac{\beta}{2}\sum_i\|\mathbf{x}_i-\mathbf{z}_i\|^2. \end{array}$$

Then one can treat $\mathbf{x}_i$'s as the first block of variables and $\mathbf{z}_i$'s the second block, and apply ADMM.

Minimizer $\mathbf{x}$'s of the Lagrangian function, when $\mathbf{z}_i$, $\mathbf{y}_i$'s are fixed, is the solution of a strongly convex quadratic minimization.

The ADMM with Three Blocks?

What about ADMM for

$$\min \quad f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + f_3(\mathbf{x}_3) \quad \text{s.t.} \quad A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + A_3\mathbf{x}_3 = \mathbf{b},$$

where the Lagrangian function

$$\begin{aligned} L(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{y}) = \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + f_3(\mathbf{x}_3) - \mathbf{y}^T(A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + A_3\mathbf{x}_3 - \mathbf{b}) \\ & + \tfrac{\beta}{2}\|A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + A_3\mathbf{x}_3 - \mathbf{b}\|^2. \end{aligned}$$

Then, for any given $(\mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{x}_3^k, \mathbf{y}^k)$, we compute a new iterate

$$\begin{aligned} \mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} L(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{x}_3^k, \mathbf{y}^k), \\ \mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} L(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{x}_3^k, \mathbf{y}^k), \\ \mathbf{x}_3^{k+1} &= \arg\min_{\mathbf{x}_3} L(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \mathbf{x}_3, \mathbf{y}^k), \\ \mathbf{y}^{k+1} &= \mathbf{y}^k - \beta(A_1\mathbf{x}^{k+1} + A_2\mathbf{x}_2^{k+1} + A_3\mathbf{x}_3^{k+1} - \mathbf{b}). \end{aligned}$$

The Direct Extension does Not Converge

Theorem 1 *There existing an example where the direct extension of ADMM of three blocks is not necessarily convergent for any choice of* β*. Moreover, for any randomly generated initial point, ADMM diverges with probability one.*

The problem with unique solution $\mathbf{x}^* = \mathbf{0}$:

$$\min \quad 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 \qquad \text{s.t.} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{0},$$

Does the smaller step-size ($1 > \gamma > 0$) dual update work? Answer: it remains divergent when solving

$$\min \quad 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 \qquad \text{s.t.} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1+\gamma \\ 1 & 1+\gamma & 1+\gamma \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{0},$$

(ADMMdiverexm.m of Chapter 14)

The Algorithmic Mapping is Not Contracting

The ADMM with $\beta = 1$ is a linear matrix mapping

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 5 & 7 & 9 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 & 1 & 0 \\ 1 & 2 & 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{k+1} \\ \mathbf{y}^{k+1} \end{pmatrix} = \begin{pmatrix} 0 & -4 & -5 & 1 & 1 & 1 \\ 0 & 0 & -7 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{k} \\ \mathbf{y}^{k} \end{pmatrix}.$$

which can be reduced to

$$\begin{pmatrix} x_2^{k+1} \\ x_3^{k+1} \\ \mathbf{y}^{k+1} \end{pmatrix} = M \begin{pmatrix} x_2^{k} \\ x_3^{k} \\ \mathbf{y}^{k} \end{pmatrix},$$

where

$$M = \frac{1}{162}\begin{pmatrix} 144 & -9 & -9 & -9 & 18 \\ 8 & 157 & -5 & 13 & -8 \\ 64 & 122 & 122 & -58 & -64 \\ 56 & -35 & -35 & 91 & -56 \\ -88 & -26 & -26 & -62 & 88 \end{pmatrix}.$$

But the spectral radius of the matrix, $\rho(M) = 1.0087 > 1$, which implies that the mapping is not a contraction.

Multi-Block Problems and ADMM

In general, consider a convex optimization problem

$$\begin{aligned} \min_{\mathbf{x}\in R^N} \quad & f_1(\mathbf{x}_1) + \cdots + f_n(\mathbf{x}_n), \\ \text{s.t.} \quad & A\mathbf{x} := A_1\mathbf{x}_1 + \cdots + A_n\mathbf{x}_n = \mathbf{b}, \\ & \mathbf{x}_i \in \mathcal{X}_i \subset R^{d_i},\ i = 1, \ldots, n. \end{aligned} \tag{6}$$

$$L(\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{y}) = \sum_i f_i(x_i) - \mathbf{y}^T(\sum_i A_i\mathbf{x}_i - \mathbf{b}) + \frac{\beta}{2}\|\sum_i A_i\mathbf{x}_i - \mathbf{b}\|^2$$

The direct Cyclic Extension Multi-block ADMM:

$$\begin{aligned} \mathbf{x}_1 &\longleftarrow \arg\min_{\mathbf{x}_1\in\mathcal{X}_1} L(\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{y}), \\ &\vdots \\ \mathbf{x}_n &\longleftarrow \arg\min_{\mathbf{x}_n\in\mathcal{X}_n} L(\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{y}), \\ \mathbf{y} &\longleftarrow \mathbf{y} - \beta(A\mathbf{x} - \mathbf{b}), \end{aligned}$$

Randomly Permuted ADMM

Random-Permuted ADMM (RP-ADMM): in each round, draw a random permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ of $\{1, \ldots, n\}$, and use the

$$\text{Update Order}: \mathbf{x}_{\sigma(1)} \to \mathbf{x}_{\sigma(2)} \to \cdots \to \mathbf{x}_{\sigma(n)} \to \mathbf{y}.$$

- This is equivalent to a random sample without replacement so it costs nothing.
- Interpretation: Force "absolute fairness" among blocks.
- Simulation Test Result on solving linear equations: always converges!

Any theory behind the success?

We produced a positive result for ADMM on solving the system of linear equations.

(ADMMrandperm.m of Chapter 14)

Random Permuted ADMM for Linear Systems

Consider solving a nonsingular square system of linear equations ($f_i = 0, \ \forall i$).

$$\begin{aligned} \min_{\mathbf{x} \in R^N} \quad & 0, \\ \text{s.t.} \quad & A_1\mathbf{x}_1 + \cdots + A_n\mathbf{x}_n = \mathbf{b}, \end{aligned}$$

RP-ADMM generates $\mathbf{z}^k$, an r.v., depending on

$$\boldsymbol{\xi}_k = (\sigma_1, \ldots, \sigma_k), \quad \mathbf{z}^i = M_{\sigma_i}\mathbf{z}^{i-1}, \ i = 1, ..., k,$$

where σ_i is the random permutation at i-th round.

Denote the expected iterate $\phi^k := E_{\boldsymbol{\xi}_k}(\mathbf{z}^k)$

Theorem 2 *The expected output converges to the unique solution of the linear system equations any integer $N \geq 1$.*

Remark: Expected convergence $\neq$ convergence, but is a strong evidence for convergence for solving most problems, e.g., when iterates are bounded.

The Average Mapping is a Contraction

- The update equation of RP-ADMM is

$$\mathbf{z}^{k+1} = M_\sigma \mathbf{z}^k,$$

where $M_\sigma \in R^{2N\times 2N}$ depend on σ.

- Define the expected update matrix as

$$M = E_\sigma(M_\sigma) = \frac{1}{n!}\sum_\sigma M_\sigma.$$

Theorem 3 *The spectral radius of M, $\rho(M)$, is strictly less than 1 for any integer $N \geq 1$.*

Remark: For A in the divergence example, $\rho(M_\sigma) > 1$ for any σ

– Averaging Helps, a lot.

RP-ADMM for Linear Constrained Convex QP

In general, consider a convex quadratic optimization problem

$$\begin{aligned} \min_{\mathbf{x}\in R^N} \quad & \mathbf{c}_1^T\mathbf{x}_1 + \cdots + \mathbf{c}_n^T\mathbf{x}_n + \tfrac{1}{2}\mathbf{x}^T Q\mathbf{x}, \\ \text{s.t.} \quad & A\mathbf{x} := A_1\mathbf{x}_1 + \cdots + A_n\mathbf{x}_n = \mathbf{b}. \end{aligned} \tag{7}$$

Theorem 4 *Under some technical assumptions, the expected output of randomly permuted ADMM converges to the solution of the original problem for any integer* $N \geq 1$.

Extensions and Research Directions (Suggested Optional Project)

- Non-square system of linear equations – “yes”
- Non-separable convex quadratic minimization with linear equality constraints – “yes”
- Convergence w.h.p.??
- Generalize to inequality systems or convex optimization at large??
- Generalize to non-convex optimization??
- ADMM where, in every iteration, each block are randomly assembled without replacement??

Software Implementation Based on ADMM

SCS: `http://www.stanford.edu/~boyd/cvx` for CLP

ABIP+: `https://github.com/sepvar/ABIP` for solving LP

RACQP: `https://github.com/kmihic/RACQP` for quadratic minimization with mixed continuous and integer decision variables.

Primal-Dual Hybrid Gradient (PDHG) Method

For the standard-form LP:

$$\text{(Primal)} \quad \min \ \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \ A\mathbf{x} = \mathbf{b}\,, \ \mathbf{x} \geq \mathbf{0}$$

Instead of directly dealing with the original problem, we deal with the primal-dual form to avoid doing expensive projections onto the feasible set $\{\mathbf{x} : \ A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geq \mathbf{0}\}$. With the

$$\text{(Dual)} \quad \max \ \mathbf{b}^\top y \quad \text{s.t.} \ A^\top \mathbf{y} \leq \mathbf{c}$$

we construct Lagrangian:

$$\text{(Primal-dual)} \quad \min_{\mathbf{x} \geq \mathbf{0}} \max_{\mathbf{y}} \ L(\mathbf{x}, \mathbf{y}) := \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{y} - \mathbf{x}^\top A^\top \mathbf{y}$$

Saddle point: The $(\mathbf{x}^\star, \mathbf{y}^\star)$ with $\mathbf{x}^\star \geq \mathbf{0}$ is a saddle point of $L(\mathbf{x}, \mathbf{y})$ if for any $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{y}$:

$L(\mathbf{x}^\star, \mathbf{y}) \leq L(\mathbf{x}^\star, \mathbf{y}^\star) \leq L(\mathbf{x}, \mathbf{y}^\star)$

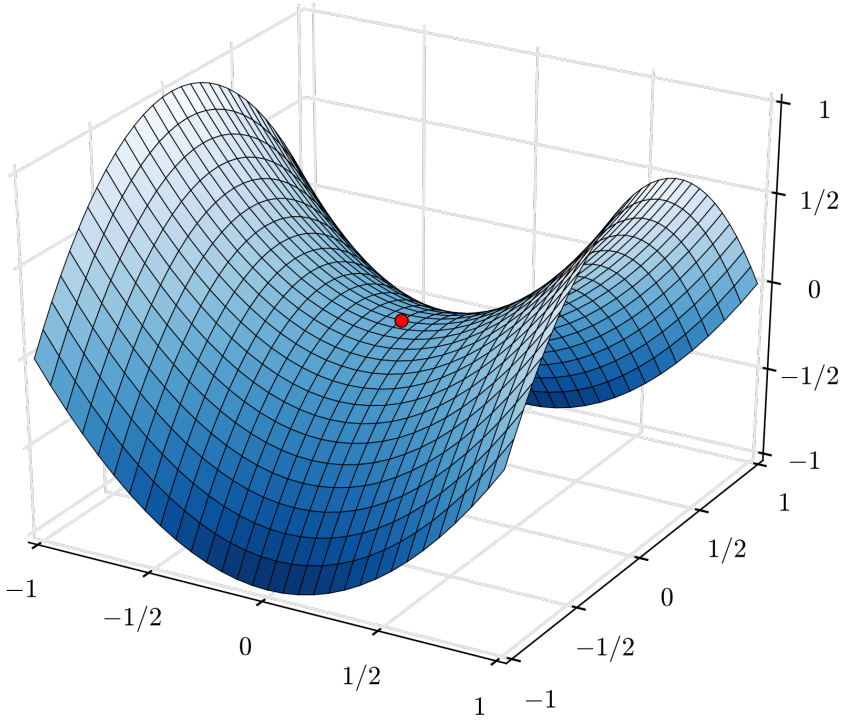


Figure 1: Saddle point

Classical Gradient Descent-Ascent:

$$\mathbf{x}^{k+1} = \text{Proj}_{\mathbf{x}\geq \mathbf{0}}\left(\mathbf{x}^k - \eta\left(\mathbf{c} - A^\top \mathbf{y}^k\right)\right) = \left(\mathbf{x}^k - \eta\left(\mathbf{c} - A^\top \mathbf{y}^k\right)\right)^+$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \eta\left(\mathbf{b} - A\mathbf{x}^k\right)$$

Gradient Descent-Ascent May Not Converge

For example, in the saddle point problem $\max_{x\geq 0}\min_y \; xy$, we have closed form of each iteration:

$$\begin{pmatrix} x^k \\ y^k \end{pmatrix} = \begin{pmatrix} 1 & \eta \\ -\eta & 1 \end{pmatrix}^k \begin{pmatrix} x^0 \\ y^0 \end{pmatrix}$$

The determinant of $\begin{pmatrix} 1 & \eta \\ -\eta & 1 \end{pmatrix}$ is always strictly larger than 1 no matter how small η is, which means this approach never converges.

Primal-Dual Hybrid Gradient (PDHG) Method

$$x^{k+1} = \text{Proj}_{x\geq 0}\left(x^k - \eta\left(c - A^\top y^k\right)\right)$$
$$y^{k+1} = y^k + \eta\left(b - A(2x^{k+1} - x^k)\right)$$

The update of y^k uses the idea of momentum or two-point mapping.

For the small saddle point problem $\max_{x\geq 0}\min_y \; xy$, we also have closed form of each iteration:

$$\begin{pmatrix} x^k \\ y^k \end{pmatrix} = \begin{pmatrix} 1 & \eta \\ -\eta & 1-2\eta^2 \end{pmatrix}^k \begin{pmatrix} x^0 \\ y^0 \end{pmatrix}$$

The modulus of eigenvalues of $\begin{pmatrix} 1 & \eta \\ -\eta & 1-2\eta^2 \end{pmatrix}$ are smaller than 1 if $\eta < 1$, so PDHG converges.

"A first-order primal-dual algorithm for convex problems with applications to imaging," Chambolle/Pock, Journal of mathematical imaging and vision, 2011.

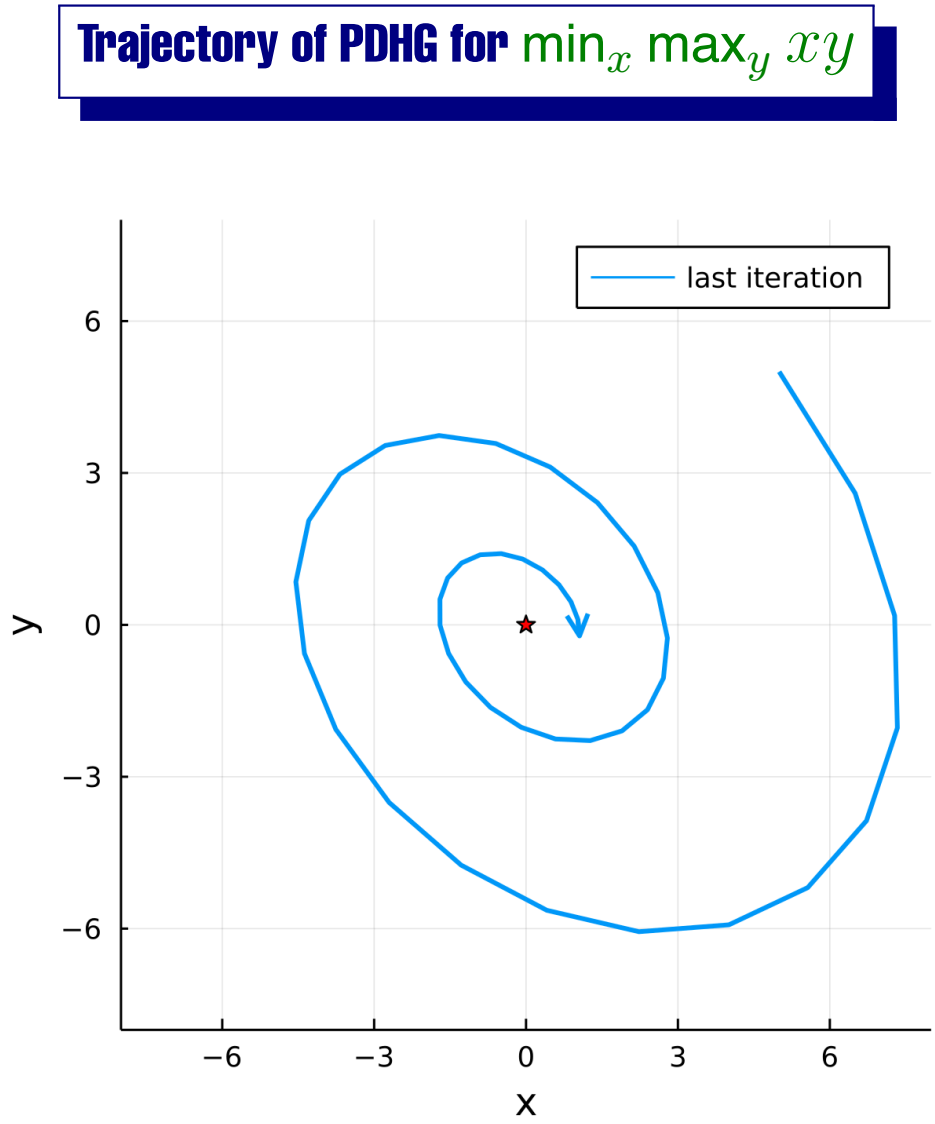


Figure 2: Final iterates of PDHG . Figure courtesy Lu.

Average Trajectory of PDHG for $\min_x \max_y xy$

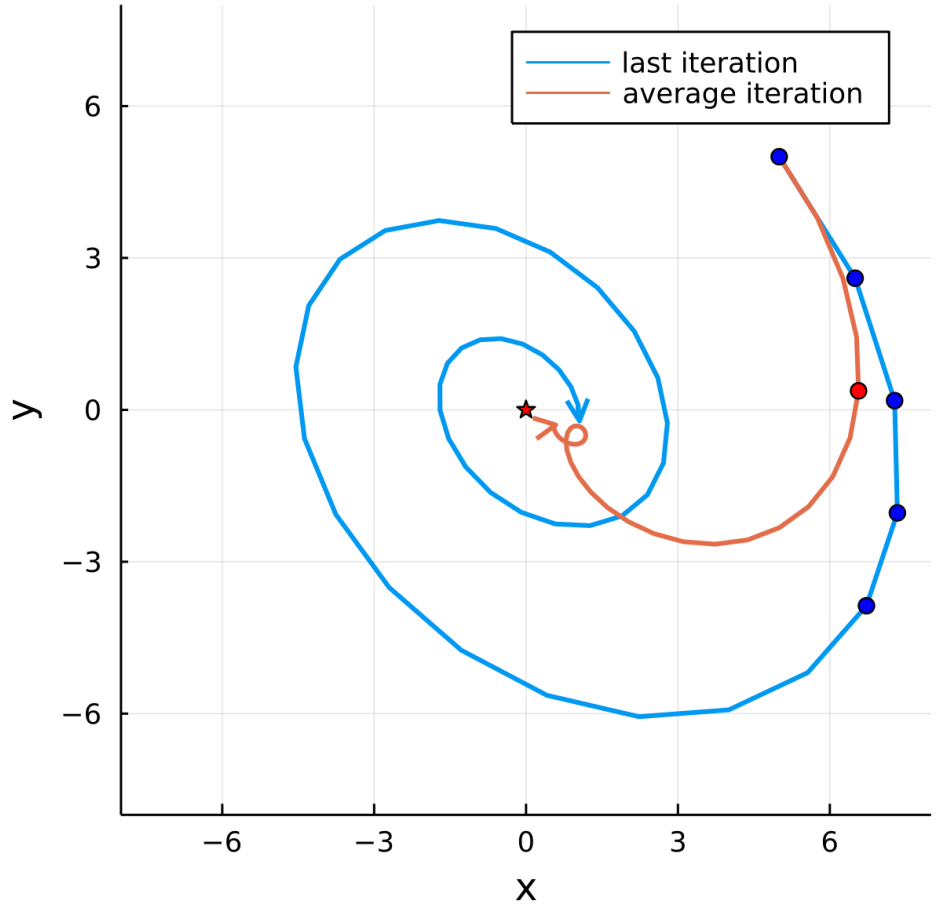


Figure 3: Average iterates of PDHG . Figure courtesy Lu.

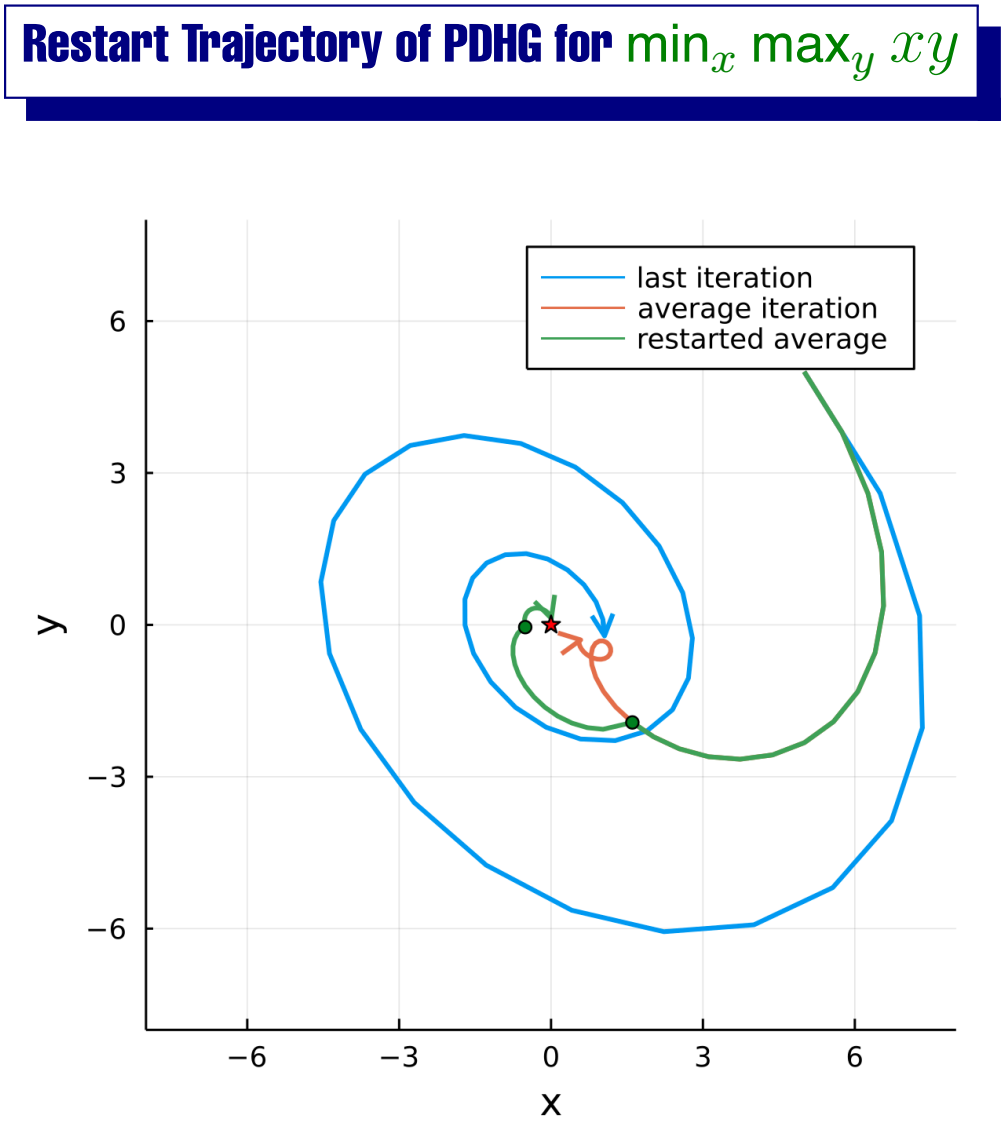


Figure 4: Restarted iterates of PDHG . Figure courtesy Lu.

PHDG with Restarts

Algorithm 1: Restarted-PDHG

Input: Initial iterate $z^{0,0} := (x^{0,0}, y^{0,0})$, $n \leftarrow 0$;
repeat
  initialize the inner loop: inner loop counter $k \leftarrow 0$;
  repeat
    conduct one step of PDHG: $z^{n,k+1} \leftarrow \mathrm{PDHG}(z^{n,k})$;
    compute the average iterate in the inner loop.
    $\bar{z}^{n,k+1} \leftarrow \frac{1}{k+1} \sum_{i=1}^{k+1} z^{n,i}$;
    $k \leftarrow k + 1$;
  until $\bar{z}^{n,k}$ **satisfies some (verifiable) restart condition** ;
  restart the outer loop: $z^{n+1,0} \leftarrow \bar{z}^{n,k}$, $n \leftarrow n + 1$;
until $z^{n,0}$ satisfies some convergence condition ;
Output: $z^{n,0}$ $(= (x^{n,0}, y^{n,0}))$

Restart Scheme: Fixed frequency restart, adaptive restart ...

Restart Strategies I

Fixed Frequency Restart:

Suppose α and $\|A\|$ are known to the user, restart the algorithm every

$$\left\lceil \frac{8\|A\|}{\alpha} \right\rceil$$

iterations

Adaptive Restart:

Restart the algorithm whenever the normalized duality gap has sufficient decay

$$\rho\left(\left\|\bar{\mathbf{z}}^{n,t} - \mathbf{z}^{n,0}\right\|; \bar{\mathbf{z}}^{n,t}\right) \leq 0.5 \cdot \rho\left(\left\|\mathbf{z}^{n,0} - \mathbf{z}^{n-1,0}\right\|; \mathbf{z}^{n,0}\right).$$

Restart Strategies II

With either of the above restart schemes, we have the following linear convergence guarantee:

Theorem 5 (Applegate et al. 2023) *The restarted PDHG finds a solution* $\mathbf{z}$ *such that* $dist(\mathbf{z}, Z^\star) \leq \varepsilon$ *within*

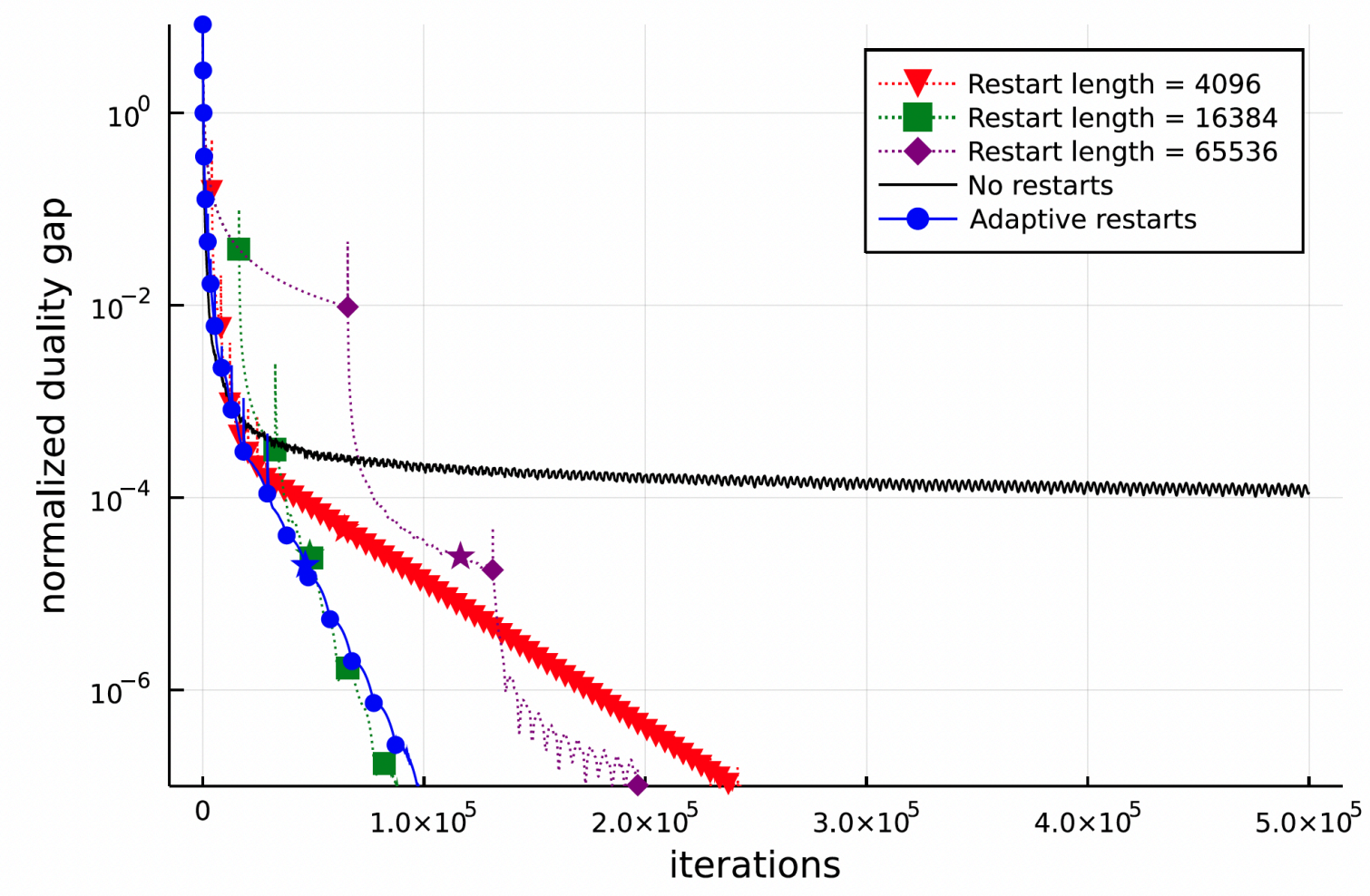
$$O\left(\frac{\|A\|}{\alpha} \cdot \log\left(\frac{1}{\varepsilon}\right)\right)$$

iterations.

An important research question: How to explain and understand α? And how to improve α? [see *Xiong/Freund arXiv:2312.14774, Lu/Yang arXiv:2307.03664, Hinders arXiv:2309.03988*]

Numerical Experiments

LP relaxation of a quadratic assignment problem:



FOMs on GPU I

FOM requires only matrix-vector multiplication:

1. Matrix-vector multiplication is very suitable for utilizing modern hardware and distributed computation
2. Matrix factorization free. The memory usage is relatively low.

FOM is particularly good at solving large instances

With help of GPU, recent research shows PDHG is fast when the problem becomes large, see

- *"cuPDLP. jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia", Lu/Yang, arXiv:2311.12180, 2023*.
- *"cuPDLP-C: A Strengthened Implementation of cuPDLP for Linear Programming by C language," Lu et al. arXiv:2312.14832, 2023*.

"Accelerating Low-Rank Factorization-Based Semidefinite Programming Algorithms on GPU" Han et al., arXiv:2407.15049.

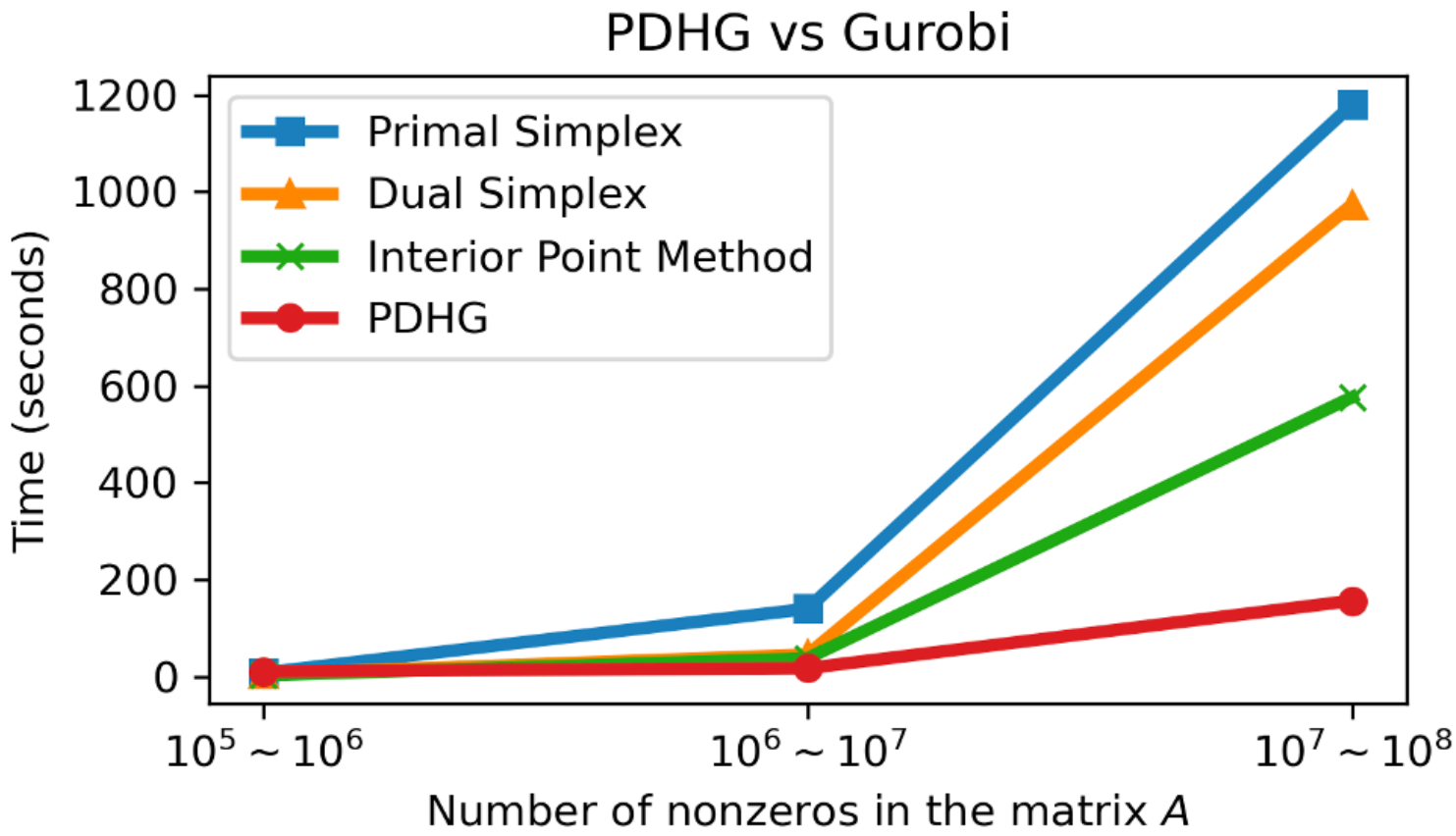


Figure 5: Geometric average runtime on problems from LP relaxations of MIPLIB 2017 (termination tolerance 10^{-4})

FOMs on GPU II

The SDP FOMs, based on ALMM and ADMM, also show dramatic speedup using GPU:

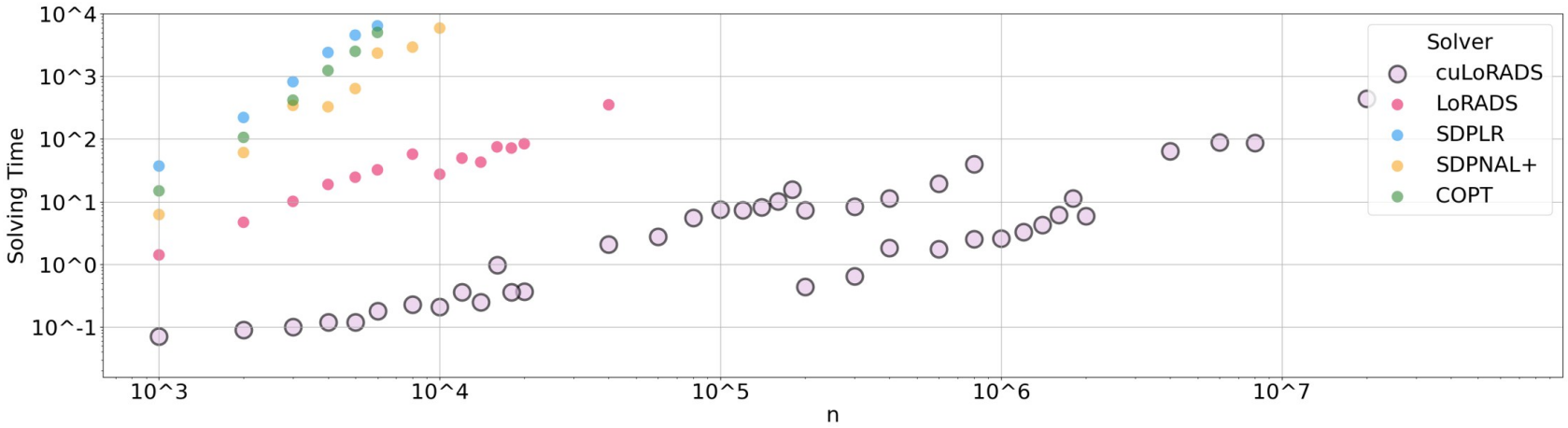


Figure 6: Solving a classic Max-Cut SDP relaxation Problems

Many ongoing projects and more improvements are expected!

1.5-Order Algorithm: Quasi-Newton Method I

$$\mathbf{x}^{k+1} = \mathbf{x}^k - S^k \nabla f(\mathbf{x}^k),$$

for a symmetric matrix S^k with a step-size α^k. When S^k can be a nonnegative diagonal matrix, then it is the scaled steepest descent method we described earlier. In general, when S^k is positive definite, direction $-S^k \nabla f(\mathbf{x}^k)$ is a descent direction (why?).

For convex quadratic minimization, the linear convergence rate then depends on λ_{max} and λ_{min} represent the largest and smallest eigenvalues of a matrix. Thus, S^k can be viewed as a Preconditioner–typically an approximation of the Hessian matrix inverse, and can be learned from a regression model: let $\mathbf{p}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ and $\mathbf{q}^k := \mathbf{g}(\mathbf{x}^{k+1}) - \mathbf{g}(\mathbf{x}^k) = Q(\mathbf{x}^{k+1} - \mathbf{x}^k) = Q\mathbf{p}^k,\ k = 0, 1, ...$. We actually learn Q^{-1} from $Q^{-1}\mathbf{q}^k = \mathbf{p}_k,\ k = 0, 1, ...$ The process start with H^k, $k = 0, 1, ...$, where the rank of H^k is k, that is, we each step lean a rank-one update: given H^{k-1}, $\mathbf{q}^k$, $\mathbf{p}^k$ we solve $(h_0 \cdot H^{k-1} + \mathbf{h}^k(\mathbf{h}^k)^T)\mathbf{q}^k = \mathbf{p}^k$ for vector $\mathbf{h}^k$. Then after n iterations, we build up $H^n = Q^{-1}$.

1.5-Order Algorithm: Quasi-Newton Method II

One can simply let

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k(\frac{n-k}{n}I + \frac{k}{n}H^k)\mathbf{g}(\mathbf{x}^k),$$

which is similar to the Conjugate Gradient method.

A popular method, BFGS, is given as follows (thre are multiple typos in the text): start from $\mathbf{x}^0$ and set $S^0 = I$, let

$$\mathbf{d}^k = -S^k\mathbf{g}(\mathbf{x}^k) = -S^k\nabla f(\mathbf{x}^k),$$

and iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{d}^k.$$

Then update

$$S^{k+1} = S^k + \left(1 + \frac{(\mathbf{q}^k)^T S^k \mathbf{q}^k}{(\mathbf{p}^k)^T\mathbf{q}^k}\right)\frac{\mathbf{p}^k(\mathbf{p}^k)^T}{(\mathbf{p}^k)^T\mathbf{q}^k} - \frac{\mathbf{p}^k(\mathbf{q}^k)^T S^k + S^k\mathbf{q}^k(\mathbf{p}^k)^T}{(\mathbf{p}^k)^T\mathbf{q}^k}.$$

1.5-Order Algorithm: The Ellipsoid Method

Ellipsoids are just sets of the form

$$E = \{\mathbf{y} \in \mathbf{R}^m : (\mathbf{y} - \bar{\mathbf{y}})^T B^{-1} (\mathbf{y} - \bar{\mathbf{y}}) \leq 1\}$$

where $\bar{\mathbf{y}} \in R^m$ is a given point (called the center) and B is a symmetric positive definite matrix of dimension m. We can use the notation ell$(\bar{\mathbf{y}}, B)$ to specify the ellipsoid E defined above. Note that

$$\text{vol}(E) = (\det B)^{1/2} \text{vol}(S(\mathbf{0}, 1)).$$

where $S(\mathbf{0}, 1)$ is the unit sphere in $\mathbf{R}^m$.

A Half-Ellipsoid

By a Half-Ellipsoid of E, we mean the set

$$\frac{1}{2}E_a := \{\mathbf{y} \in E : \mathbf{a}^T\mathbf{y} \leq \mathbf{a}^T\bar{\mathbf{y}}\}$$

for a given non-zero vector $\mathbf{a} \in \mathbf{R}^m$ where $\bar{\mathbf{y}}$ is the center of E – the intersection of the ellipsoid and a plane cutting through the center.

We are interested in finding a new ellipsoid containing $\frac{1}{2}E_a$ with the least volume.

- How small could it be?
- How easy could it be constructed?

The New Containing Ellipsoid

The new ellipsoid $E^+ = \text{ell}(\bar{\mathbf{y}}^+, B^+)$ can be constructed as follows. Define

$$\tau := \frac{1}{m+1}, \qquad \delta := \frac{m^2}{m^2-1}, \qquad \sigma := 2\tau.$$

And let

$$\bar{\mathbf{y}}^+ := \bar{\mathbf{y}} - \frac{\tau}{(\mathbf{a}^T B \mathbf{a})^{1/2}} B\mathbf{a},$$

$$B^+ := \delta\left(B - \sigma \frac{B\mathbf{a}\mathbf{a}^T B}{\mathbf{a}^T B \mathbf{a}}\right).$$

Theorem 6 *Ellipsoid $E^+ = \text{ell}(\bar{\mathbf{y}}^+, B^+)$ defined as above is the ellipsoid of least volume containing $\frac{1}{2}E_a$. Moreover,*

$$\frac{\text{vol}(E^+)}{\text{vol}(E)} \leq \exp\left(-\frac{1}{2(m+1)}\right)$$

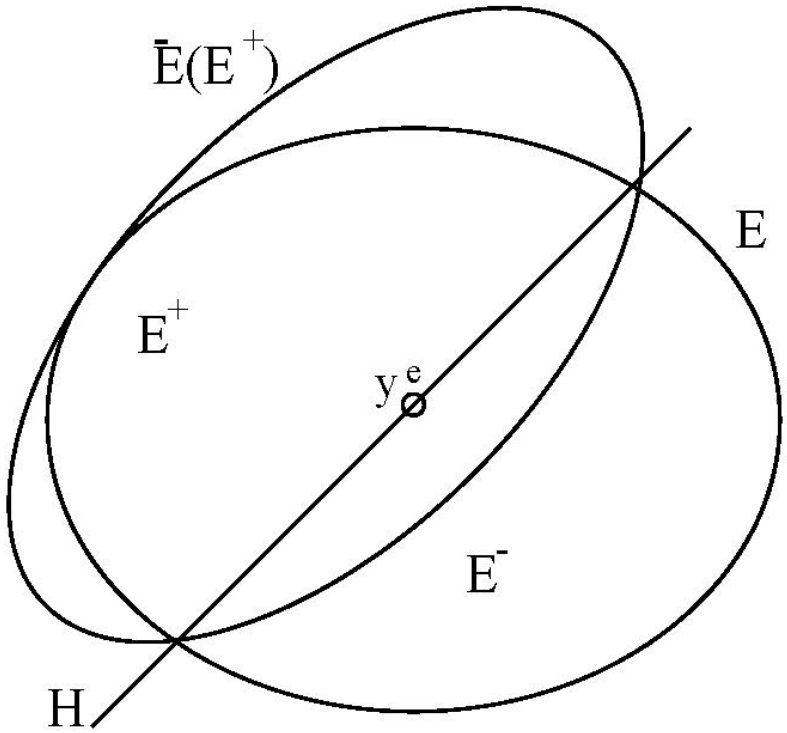


Figure 7: The least volume ellipsoid containing a half ellipsoid

(Ellip...LP...m of Chapter 10)

The Ellipsoid Method for Minimizing a Convex Function

Consider $\min_{\mathbf{x}} \ f(\mathbf{x})$:

- Initialization: Set the initial ellipsoid (ball) as $B^0 = \frac{1}{R^2} I$ centered at an initial solution $\mathbf{x}^0$ where R is sufficiently large such it contains an optimal solution.
- For $k = 0, 1, \ldots$ do

  If not terminated:

  - Compute the (sub)gradient vector $\nabla f(\mathbf{x}^k)$,
  - Let the cutting-plane be $\{\mathbf{x} : \ \nabla f(\mathbf{x}^k)^T \mathbf{x} \leq f(\mathbf{x}^k)^T \mathbf{x}^k\}$ and form the half ellipsoid; and update $\mathbf{x}^k$ and B^k as described earlier.

(Ellip...QP...m of Chapter 10)

1.5-Order Algorithm: ADAM - Adaptive Learning-Rate and Moment Estimation

The most popular method using Stochastic Gradient $\hat{\mathbf{g}}^k$ at the kth iteration such as

$$\mathbf{d}^k = (1-\beta_1)\hat{\mathbf{g}}^k + \beta_1 \mathbf{d}^{k-1}$$
$$\mathbf{s}^k = (1-\beta_2)\hat{\mathbf{g}}^k .* \hat{\mathbf{g}}^k + \beta_2 \mathbf{s}^{k-1}$$
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \,\mathrm{Diag}(\mathbf{s}^k)^{-1/2}\mathbf{d}^k.$$

for some parameters β_1 and β_2.

One can see that the Scaling Matrix is Diagonal and Positive Definite.

ADAMW: update by adding the regularization $0,5\lambda\|\mathbf{x}\|^2$ to the loss function:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k(\mathrm{Diag}(\mathbf{s}^k)^{-1/2}\mathbf{d}^k + \lambda\mathbf{x}^k).$$

ADAM_mini: according to the Block-Diagonal Structure of Hessian, each block use the same learning-rate/scaling-weight to save memories.

The Gradient Method (GM) with Online-Scaling

Consider unconstrained convex optimization problem

$$f^* = \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

- $f(\mathbf{x})$ is L-smooth and μ-strongly convex (for convex QP, they are the max and min e-values of Hessian)

$$f(\mathbf{y}) + \nabla f(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) + \frac{\mu}{2}\|\mathbf{x}-\mathbf{y}\|^2 \le f(\mathbf{x}) \le f(\mathbf{y}) + \nabla f(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|^2$$

- Recall gradient descent with $1/L$ stepsize: $\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)$ gives linear convergence

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) \le (1 - \frac{\mu}{L})(f(\mathbf{x}^k) - f(\mathbf{x}^*))$$

- where $\kappa = L/\mu$ is known as the condition number and greatly affects practical algorithm performance

Effect of Condition Number on Gradient Descent

Condition number characterizes the heterogeneity of the optimization landscape:

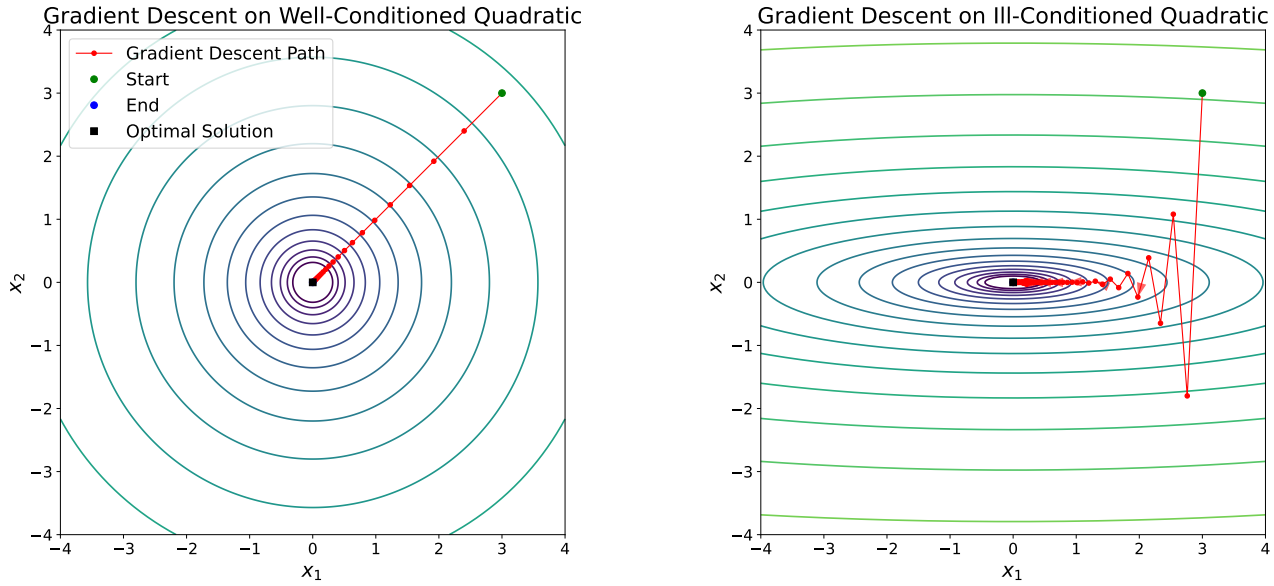


Figure 8: Gradient descent on left: $x^2 + y^2$ and right: $x^2 + 100y^2$. A highly heterogeneous landscape slows down convergence

A large condition number implies slow convergence for GM. A standard way of alleviation is preconditioning.

Preconditioning and Optimal Preconditioner

Preconditioning reduces the heterogeneity of the landscape by applying a scaling matrix S, known as preconditioner, to the GM update

$$\mathbf{x}^{k+1} = \mathbf{x}^k - S\nabla f(\mathbf{x}^k).$$

– Applying an affine transformation to the problem to get an improved condition number $\kappa_S < \kappa$

– Given a set of candidate scaling matrices $\mathcal{S}$, the optimal preconditioner S^* solves

$$\kappa^* = \min_{S\in\mathcal{S}} \kappa_S = \min_{\kappa} \kappa \quad \text{subject to} \quad \tfrac{1}{\kappa} I \preceq S\nabla^2 f(\mathbf{x})S \preceq I \quad \text{for all } \mathbf{x}$$

and optimally scaled (preconditioned) gradient descent $x^{k+1} = x^k - S^*\nabla f(x^k)$ gives

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) \leq (1 - \tfrac{1}{\kappa^*})(f(\mathbf{x}^k) - f(\mathbf{x}^*))$$

and $\mathcal{O}(\kappa^* \log(1/\varepsilon))$ complexity.

Optimal scaling matrix S^*, even restricted to diagonal matrices, can greatly improve convergence.

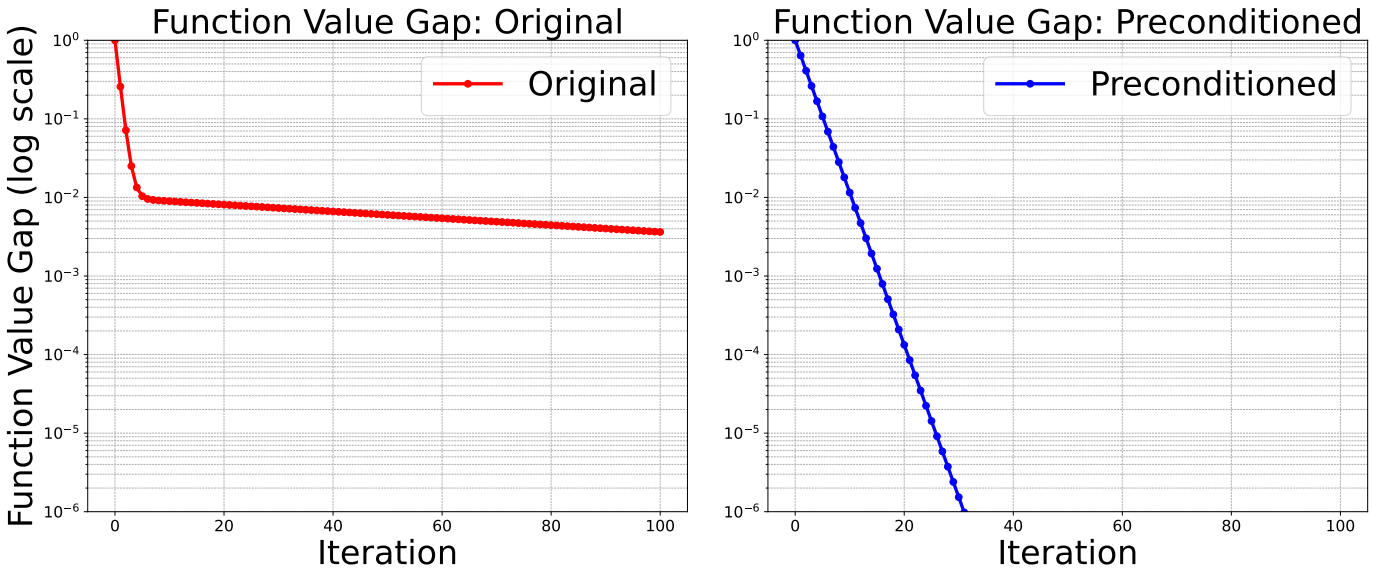


Figure 9: Optimal scaling matrix greatly improves the convergence of gradient descent

However, even for a quadratic function, finding S^* requires solving an SDP of dimension n.

Is it possible to achieve $\mathcal{O}(\kappa^* \log(1/\varepsilon))$ convergence without even computing S^*?

Answer: although we don't know S^*, we can learn it on the fly with online learning.

Online Learning Preliminaries

Consider a sequential game with a Principal and an Agent. At round k,

- Principal makes decision S^k
- Agent gives Principal feedback $\ell_k(S^k)$
- Principal adjusts based on the $\ell_k(S^k)$ using online learning algorithm $\mathcal{A}$

If the feedbacks $\{\ell_k(S^k)\}$ satisfy certain regularity/truthful conditions, online learning algorithms generate a sequence of $\{S^k\}$ such that the cumulative regret

$$\sum_{k=1}^{K} \ell_k(S^k) - \min_{S\in\mathcal{S}} \sum_{k=1}^{K} \ell_k(S) \leq \rho_K$$

and ρ_K is sublinear in K : $\lim_{K\to\infty} \rho_K/K = 0$.

– We can do as well as the optimal **hindsight** on average.

– GM fits in the framework and can be viewed as such a collaborative game!

Gradient Descent as a Collaborative Game

Consider a sequential game with a Scaling Planner (SP) and an Gradient Executioner (GE)

- Scaling Planner decides a scaling matrix S^k: $\mathbf{x}^{k+1} = \mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)$
- Gradient Executioner gives SP feedback $\ell_k(S^k)$ (?)
- SP adjusts based on the $\ell_k(S^k)$ using online learning algorithm $\mathcal{A}$ (?)

Q1. What feedback should we choose? Since we want to get fast linear convergence:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) \leq (1 - \tfrac{1}{\kappa^*})(f(\mathbf{x}^k) - f(\mathbf{x}^*))$$

A very natural choice is the linear convergence rate after one step:

$$\ell_{\mathbf{x}}(S) = \frac{f(\mathbf{x} - S\nabla f(\mathbf{x})) - f(\mathbf{x}^*)}{f(\mathbf{x}) - f(\mathbf{x}^*)}$$

Q2. How to update S? (Online) Gradient Descent suffices:

$$S^{k+1} = S^k - \eta \nabla \ell_k(S^k)$$

Online Scaled Gradient Method (OSGM)

At iteration k

- SP passes matrix S^k to GE: $\mathbf{x}^{k+1} = \mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)$
- GE gives player feedback $\ell_k(S^k) = \frac{f(\mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)) - f(\mathbf{x}^*)}{f(\mathbf{x}^k) - f(\mathbf{x}^*)}$
- SP performs the OGD step

$$S^{k+1} = S^k - \eta \nabla \ell_k(S^k) = S^k - \eta \frac{\nabla f(\mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)) \nabla f(\mathbf{x}^k)^\top}{f(\mathbf{x}^k) - f(\mathbf{x}^*)}$$

Lemma 1 *$\ell_{\mathbf{x}}(S)$ is $2L^2$-smooth, convex and lower bounded by 0.*

Lemma 2 *For smooth, convex, and lower-bounded losses, online gradient descent generates a sequence of scaling matrices $\{S^k\}$ such that*

$$\frac{1}{K} \sum_{k=1}^K \ell_k(S^k) - \min_{S \in \mathcal{S}} \frac{1}{K} \sum_{k=1}^K \ell_k(S) \leq \mathcal{O}(\frac{1}{\sqrt{K}}).$$

The SP is competitive with any fixed scaling matrix, including the hindsight or optimal S^*.

Convergence Analysis

Convergence analysis in three lines:

By regret bound:

$$\frac{1}{K}\sum_{k=1}^{K}\ell_k(S^k) \le \min_{S\in\mathcal{S}} \frac{1}{K}\sum_{k=1}^{K}\ell_k(S) + \mathcal{O}(\frac{1}{\sqrt{K}})$$

By definition and arithmetic-geometric mean inequality:

$$\begin{array}{rcl} \frac{f(\mathbf{x}^{K+1})-f(\mathbf{x}^*)}{f(\mathbf{x}^1)-f(\mathbf{x}^*)} & = & \prod_{k=1}^{K}\frac{f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)}{f(\mathbf{x}^k)-f(\mathbf{x}^*)} \\ & \le & \left(\frac{1}{K}\sum_{k=1}^{K}\ell_k(S^k)\right)^K \\ & \le & \left(\min_{S\in\mathcal{S}}\frac{1}{K}\sum_{k=1}^{K}\ell_k(S) + \frac{O(1)}{\sqrt{K}}\right)^K \end{array}$$

$\min_{S\in\mathcal{S}}\frac{1}{K}\sum_{k=1}^{K}\ell_k(S) \le \frac{1}{K}\sum_{k=1}^{K}\ell_k(S^*) \le 1-\frac{O(1)}{\kappa^*}$ and the convergence rate is

$$(1-\frac{1}{\kappa^*}+\frac{O(1)}{\sqrt{K}})^K \approx (1-\frac{1}{\kappa^*})^K.$$

We don't know S^*, but we perform as well as it as K is large enough.

More Online Scaled Gradient Method (OSGM)

At iteration k

- GE gives player feedback $\ell_k(S^k) = \frac{f(\mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)) - f(\mathbf{x}^*)}{f(\mathbf{x}^k) - f(\mathbf{x}^*)}$
- SP performs the OGD step

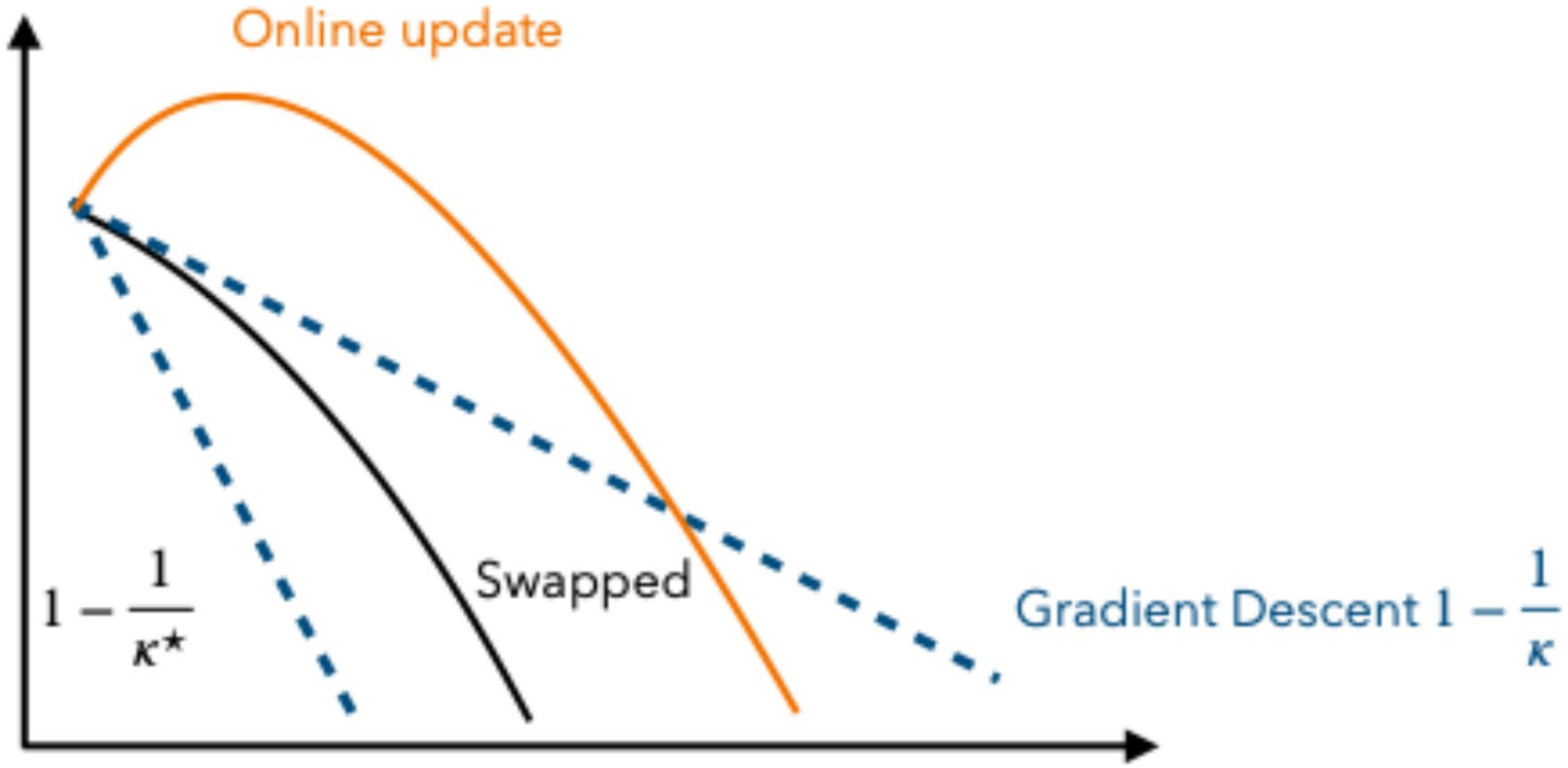
$$S^{k+1} = S^k - \eta \nabla \ell_k(S^k) = S^k - \eta \frac{\nabla f(\mathbf{x}^k - S^k \nabla f(\mathbf{x}^k)) \nabla f(\mathbf{x}^k)^\top}{f(\mathbf{x}^k) - f(\mathbf{x}^*)}$$

- SP passes matrix S^{k+1} to GE : $\mathbf{x}^{k+1} = \mathbf{x}^k - S^{k+1} \nabla f(\mathbf{x}^k)$

Lemma 3 *For smooth, convex, and lower-bounded losses, online gradient descent generates a sequence of scaling matrices* $\{S^k\}$ *such that*

$$\frac{1}{K} \sum_{k=1}^K \ell_k(S^k) - \min_{S \in \mathcal{S}} \frac{1}{K} \sum_{k=1}^K \ell_k(S) \leq \mathcal{O}(\frac{1}{K}).$$

SP becomes more competitive:


Online update
Swapped
$1-\frac{1}{\kappa^\star}$
Gradient Descent $1-\frac{1}{\kappa}$

Other Convergence Properties of OSGM

Theorem 7 *If $f(x^*)$ is unknown, then OSGM has $\mathcal{O}(\kappa^* \log^2(1/\varepsilon))$ complexity*

Theorem 8 *If $\mathcal{S} = \mathbb{R}^{n\times n}$ and $f(\mathbf{x})$ has Lipschitz continuous Hessian, then OSGM has local superlinear convergence: $\mathcal{O}((\frac{1}{\sqrt{K}})^K)$.*

Theorem 9 *If $\mathcal{S} = \mathbb{R}^{n\times n}$ and under other regularity conditions, the scaling matrices $\{S^k\}$ generated by OSGM converge to $(\nabla^2 f(\mathbf{x}^*))^{-1}$.*

Similar results appeared in Quasi-Newton methods, but the analysis in OSGM is much simpler.

Numerical Experiments

Linear least squares $f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2$

Three typical examples of $\mathcal{S}$: scalar times identity, diagonal matrix, and full matrix

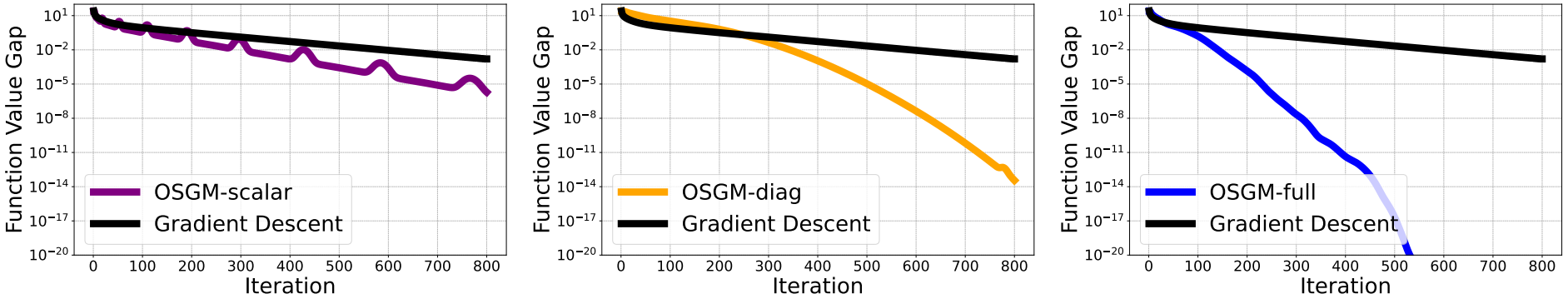


Figure 10: Convergence of OSGM. Left: diagonal scaling matrix; right: full scaling matrix

A larger candidate set $\mathcal{S}$ gives a more competitive S^*.

(osgm...m of Chapter 8 or 10)

Extensions and Future Directions

- "Gao, W., Chu, Y. C., Ye, Y., & Udell, M. (2024). Gradient Methods with Online Scaling. arXiv preprint arXiv:2411.01803."

Future work:

- Can the convergence be further accelerated to $\mathcal{O}(\sqrt{\kappa^*}\log(1/\varepsilon))$ with momentum directions?
- How to reduce variances with reporting stochastic gradient descent?
- How to do constrained optimization?
- What happens in a nonconvex landscape?
- Connection with Quasi-Newton methods?
- Generalization to other, such as high-order, iterative methods?