# Sliding Alternating Direction Method of Multipliers

Yuyuan Ouyang and Trevor Squires[1]

INFORMS Annual Meeting 2020

[1]This research is partially supported by US Dept. of the Air Force grant FA9453-19-1-0078 and NSF grant DMS-1913006.

# Problem Setting

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ using a first order method.

# Problem Setting

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ using a first order method.

- $X \subset \mathbb{R}^n$ is closed, convex
- $K \in \mathbb{R}^{m \times n}$
- $f$ and $h$ are real-valued, convex functions

# Problem Setting

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \qquad \text{(CO)}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ using a first order method.

- $X \subset \mathbb{R}^n$ is closed, convex
- $K \in \mathbb{R}^{m \times n}$
- $f$ and $h$ are real-valued, convex functions

We further assume that

- $\nabla f$ is Lipschitz continuous with Lipschitz constant $L$
- $X$ is easy to project to
- The proximal mapping problem involving $h(\cdot)$ is easy, i.e.

  $$\min_{w \in \mathbb{R}^m} h(w) + \frac{\rho}{2}\|w - z\|^2$$

  can be solved quickly.

In algorithms to follow, it is sometimes useful to view (CO) as an affinely constrained optimization problem.

**Affinely Convex Optimization**

Equivalently, we rewrite (CO) as

$$F^* := \min_{x \in X, z \in Z} f(x) + h(z) \text{ s.t. } Kx - b = z. \tag{ACO}$$

# Problem Setting

**Convex Optimization**

Our problem of interest is the minimization problem

$$F^* := \min_{x \in X} f(x) + h(Kx - b). \tag{CO}$$

Examples of (CO) include

- $f(x) = \frac{1}{2} ||Ax - b||_2^2, f(x) = \sum_{i=1}^{N} 2 \log \left(1 + \exp\left(-b_{(i)}(a_i^T x + y)\right)\right)$
- $h(x) = \lambda ||x||_p$, $p = 1, 2$, or $\infty$
- $X = \{x \mid ||x|| \leq 1\}, X = \{x \mid \sum_i x_i = 1, x \geq 0\}$

# Problem Setting

**Convex Optimization**

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

Two questions come to mind:

# Problem Setting

**Convex Optimization**

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

Two questions come to mind:

- How do we measure the efficiency of an algorithm?

# Problem Setting

**Convex Optimization**

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

Two questions come to mind:

- How do we measure the efficiency of an algorithm?
- What does it mean to be "as quickly as possible"?

# Oracle Complexity Theory

**Convex Optimization**

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

How do we measure the efficiency of an algorithm?

## Oracle Complexity Theory

### Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

How do we measure the efficiency of an algorithm?

- define some oracle $\mathcal{O} : \mathbb{R}^n \to S$
- oracle complexity theory assumes a method $\mathcal{M}$ accesses information only through querying $\mathcal{O}$ during each iteration
- efficiency is measured by the number of times $\mathcal{O}$ is queried by $\mathcal{M}$

### Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

How do we measure the efficiency of an algorithm?

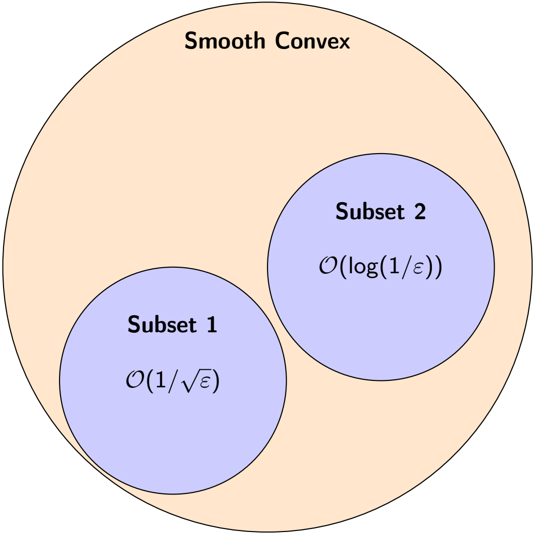
- define some oracle $\mathcal{O} : \mathbb{R}^n \to S$
- oracle complexity theory assumes a method $\mathcal{M}$ accesses information only through querying $\mathcal{O}$ during each iteration
- efficiency is measured by the number of times $\mathcal{O}$ is queried by $\mathcal{M}$

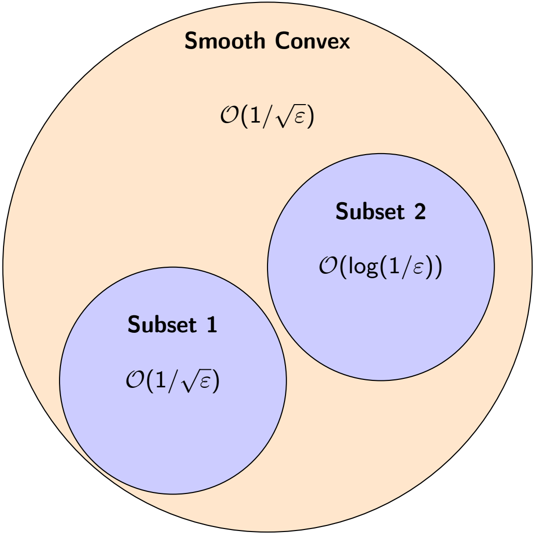
Example: Gradient Descent

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

can be evaluated under oracle $\mathcal{O}(x) = (f(x), \nabla f(x))$. It requires on the order of $1/\varepsilon$ oracle queries to obtain an $\varepsilon$-solution.

# Lower Complexity Bound - a worst case problem instance




Smooth Convex
$\mathcal{O}(1/\sqrt{\varepsilon})$
Subset 2
$\mathcal{O}(\log(1/\varepsilon))$
Subset 1
$\mathcal{O}(1/\sqrt{\varepsilon})$

**Convex Optimization**

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

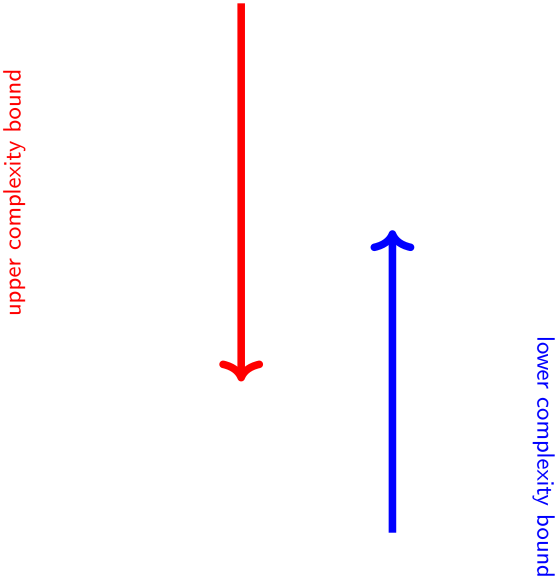
such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

What does it mean to be "as quickly as possible"?

- define some oracle $\mathcal{O} : \mathbb{R}^n \to S$
- show the method matches the lower complexity bound of the corresponding oracle

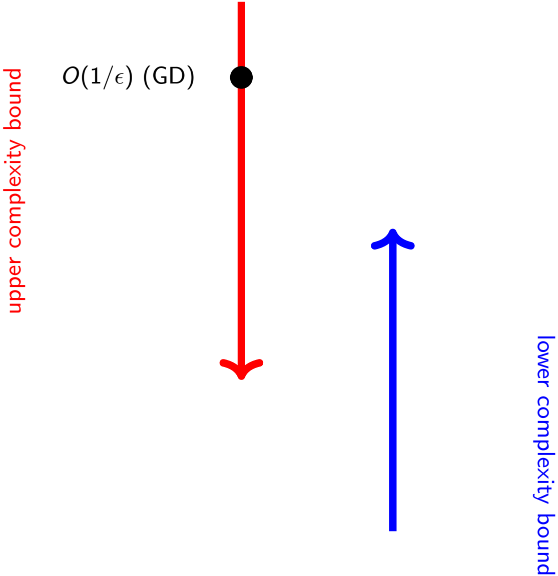
## Algorithm Optimality

What does it mean to be "as quickly as possible"?



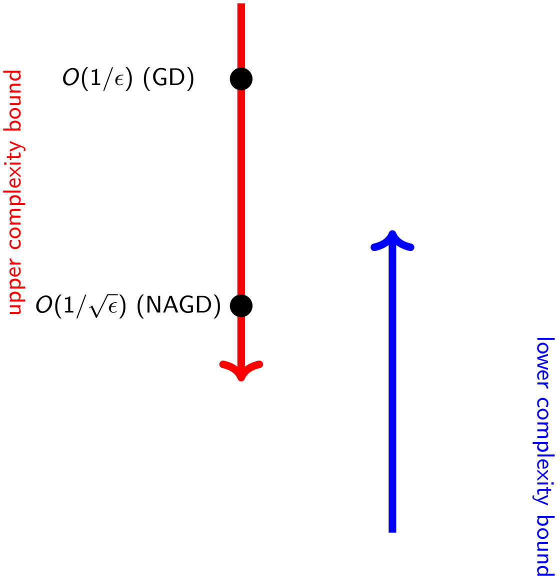
# Algorithm Optimality

What does it mean to be "as quickly as possible"?

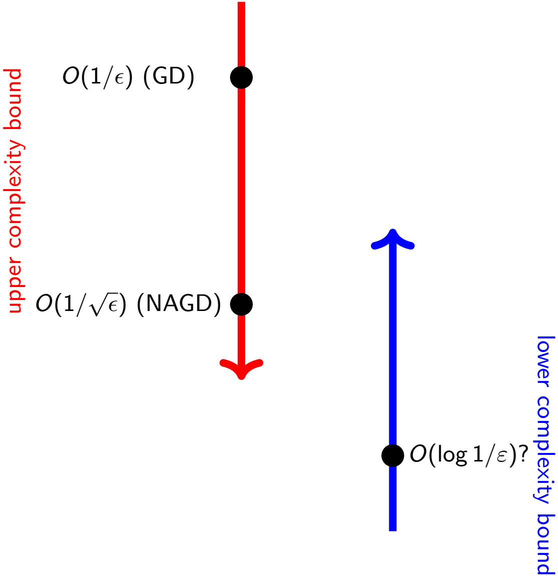


# Algorithm Optimality

What does it mean to be "as quickly as possible"?

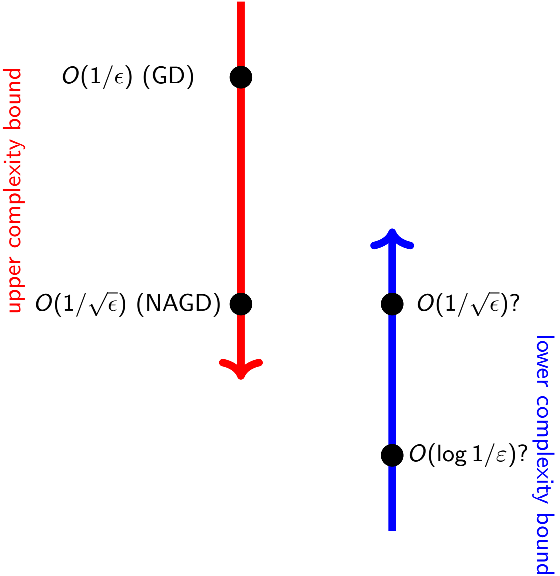


What does it mean to be "as quickly as possible"?

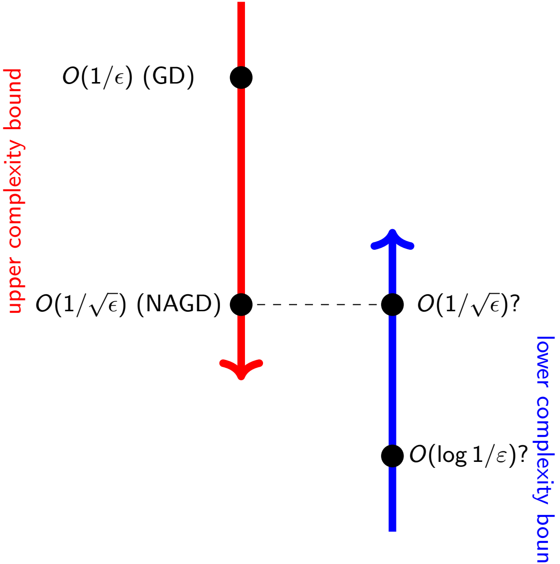


# Algorithm Optimality

What does it mean to be "as quickly as possible"?



What does it mean to be "as quickly as possible"?



# Solving (CO)

**Algorithm 1** Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $\bar{x}_0 := x_0$

**for** $k = 1, \ldots, N$ **do**

$$\begin{aligned} \underline{x}_k &= (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_{k-1}, \\ x_k &= \operatorname*{argmin}_{u \in X} \langle \nabla f(\underline{x}_k), u\rangle + h(Ku - b) + \frac{\eta_k}{2}\|u - x_{k-1}\|^2, \\ \bar{x}_k &= (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_k. \end{aligned}$$

**end for**

Output $\bar{x}_N$.

- reduces to gradient descent when $X = \mathbb{R}^n, h \equiv 0, \gamma_k = 1$
- computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ iterations
- is optimal for solving problems such as (CO) using oracle $\mathcal{O}(x) = (f(x), \nabla f(x))$ ([1])

**Algorithm 2** Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $\bar{x}_0 := x_0$

**for** $k = 1, \dots, N$ **do**

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(\underline{x}_k), u \rangle + h(Ku - b) + \frac{\eta_k}{2} \|u - x_{k-1}\|^2, \\ \bar{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_k. \end{aligned}$$

**end for**

Output $\bar{x}_N$.

- reduces to gradient descent when $X = \mathbb{R}^n, h \equiv 0, \gamma_k = 1$
- computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ iterations
- is optimal for solving problems such as (CO) using oracle $\mathcal{O}(x) = (f(x), \nabla f(x))$ ([1])
- potentially problematic subproblem
- need another oracle to measure the ambiguity in subproblem difficulty

**Algorithm 3** Nesterov's Smoothing Algorithm (NEST-S)

Start: Choose $x_0 \in X$. Set $\bar{x}_0 := x_0$
**for** $k = 1, \dots, N$ **do**

$$\begin{aligned}
\underline{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_{k-1}, \\
y_k =& \underset{v \in \mathbb{R}^m}{\operatorname{argmin}} - \langle K\underline{x}_k, v\rangle + h^*(v) + \frac{\rho}{2}\|v\|^2, \\
x_k =& \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(\underline{x}_k) + K^\top y_k \rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2, \\
\bar{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_k.
\end{aligned}$$

**end for**
Output $\bar{x}_N$.

- replaces nonsmooth $h$ with smooth approximation
- subproblem becomes easy, but need to compute smooth approximation of $h$
- computes $\varepsilon$-solution in $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$ oracle calls of $\mathcal{O}(x, y) = (\nabla f(x), Kx, K^T y)$

**Algorithm 4** Alternating Direction Method of Multipliers (ADMM)

Start: Choose $x_0 \in X$. Set $y_0 := 0$ and $z_0 := Kx_0$.

**for** $k = 1, \dots, N$ **do**

$$x_k = \underset{u \in X}{\text{argmin}}\, f(u) + \langle y_{k-1}, Ku - b - z_{k-1} \rangle + \frac{\eta_k}{2} \|Ku - b - z_{k-1}\|^2$$

$$z_k = \underset{w \in \mathbb{R}^m}{\text{argmin}} - \langle y_{k-1}, w \rangle + h(w) + \frac{\tau_k}{2} \|Kx_k - w\|^2,$$

$$y_k = y_{k-1} - \rho_k (Kx_k - z_k).$$

**end for**

Output $x_N$.

- alternates updating primal and dual variables
- computes $\varepsilon$-solution in $\mathcal{O}((L + \|K\|)/\varepsilon)$ oracle calls of $\mathcal{O}(x, y) = (\nabla f(x), Kx, K^T y)$

**Algorithm 5** Linearized Alternating Direction Method of Multipliers (L-ADMM)

Start: Choose $x_0 \in X$. Set $y_0 := 0$ and $z_0 := Kx_0$.
**for** $k = 1, \ldots, N$ **do**

$$x_k = \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(x_k), u \rangle + K^\top (y_{k-1} + \theta_k (Kx_{k-1} - z_{k-1})), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2$$

$$z_k = \underset{w \in \mathbb{R}^m}{\operatorname{argmin}} - \langle y_{k-1}, w \rangle + h(w) + \frac{\tau_k}{2} \|Kx_k - w\|^2,$$

$$y_k = y_{k-1} - \rho_k (Kx_k - z_k).$$

**end for**
Output $x_N$.

- alternates updating primal and dual variables
- computes $\varepsilon$-solution in $\mathcal{O}((L + ||K||)/\varepsilon)$ oracle calls of $\mathcal{O}(x, y) = (\nabla f(x), Kx, K^T y)$
- introduces linearization for simpler $x_k$ subproblems

## Solving (CO)

**Algorithm 6** Accelerated Alternating Direction Method of Multipliers (A-ADMM)

Start: Choose $x_0 \in X$. Set $\overline{x}_0 := x_0$, $y_0 := 0$, and $z_0 := Kx_0$.

**for** $k = 1, \ldots, N$ **do**

$$\begin{aligned}
\underline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1}, \\
x_k =& \operatorname*{argmin}_{u \in X} \langle \nabla f(\underline{x}_k), u\rangle + K^\top(y_{k-1} + \theta_k(Kx_{k-1} - z_{k-1})), u\rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2 \\
z_k =& \operatorname*{argmin}_{w \in \mathbb{R}^m} -\langle y_{k-1}, w\rangle + h(w) + \frac{\tau_k}{2}\|Kx_k - w\|^2, \\
y_k =& y_{k-1} - \rho_k(Kx_k - z_k). \\
\overline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_k.
\end{aligned}$$

**end for**

Output $\overline{x}_N$.

- acceleration motivated by NAGD
- was shown in [2] that it computes $\varepsilon$-solution in $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$ oracle calls of $\mathcal{O}(x, y) = (\nabla f(x), Kx, K^T y)$

# Improving the Oracle Idea

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

A few remarks:

1. For oracle $\mathcal{O}(x) = (\nabla f(x), Kx, K^T y)$, both A-ADMM and NEST-S only require $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| /\varepsilon)$ calls.
2. In [3], it was shown that $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| /\varepsilon)$ is the lower complexity bound for problems of the form (CO) using oracle $(\nabla f(x), Kx, K^T y)$.

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

A few remarks:

1. For oracle $\mathcal{O}(x) = (\nabla f(x), Kx, K^T y)$, both A-ADMM and NEST-S only require $\mathcal{O}(\sqrt{L/\varepsilon} + \|K\|/\varepsilon)$ calls.
2. In [3], it was shown that $\mathcal{O}(\sqrt{L/\varepsilon} + \|K\|/\varepsilon)$ is the lower complexity bound for problems of the form (CO) using oracle $(\nabla f(x), Kx, K^T y)$.

We then make the following observations

1. Whenever we simply count gradient evaluations, the problem can be solved in $\mathcal{O}(\sqrt{L/\varepsilon})$ calls.
2. Whenever operator evaluations are involved, the number of calls increases to $\mathcal{O}(\sqrt{L/\varepsilon} + \|K\|/\varepsilon)$.

# Improving the Oracle Idea

## Convex Optimization

Our problem of interest is computing an $\varepsilon$-solution $\tilde{x}$ to

$$F^* := \min_{x \in X} F(x) := f(x) + h(Kx - b) \tag{CO}$$

such that $F(\tilde{x}) - F^* < \varepsilon$ as quickly as possible.

A few remarks:

1. For oracle $\mathcal{O}(x) = (\nabla f(x), Kx, K^T y)$, both A-ADMM and NEST-S only require $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$ calls.
2. In [3], it was shown that $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$ is the lower complexity bound for problems of the form (CO) using oracle $(\nabla f(x), Kx, K^T y)$.

We then make the following observations

1. Whenever we simply count gradient evaluations, the problem can be solved in $\mathcal{O}(\sqrt{L/\varepsilon})$ calls.
2. Whenever operator evaluations are involved, the number of calls increases to $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| / \varepsilon)$.

Perhaps there is an algorithm that keeps $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations while still keeps the operator evaluations low.

**Algorithm 7** Gradient Sliding (GS)

Start: Choose $x_0 \in X$. Set $\bar{x}_0 := x_0$

**for** $k = 1, \dots, N$ **do**

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \text{Subgradient}(\langle \nabla f(\underline{x}_k), u \rangle + h(Ku - b) + \frac{\eta_k}{2}\|u - x_{k-1}\|^2) \\ \bar{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_k. \end{aligned}$$

**end for**

Output $\overline{x}_N$.

- same as NAGD, but solves subproblem using subgradient method
- was shown in [4] that it computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient calls, but $\mathcal{O}(\sqrt{L/\varepsilon} + (||K|| / \varepsilon)^2)$ operator calls

# Solving (CO) with Sliding

**Algorithm 8** Gradient Sliding (GS)

Start: Choose $x_0 \in X$. Set $\bar{x}_0 := x_0$
**for** $k = 1, \dots, N$ **do**

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \text{Subgradient}(\langle \nabla f(\underline{x}_k), u\rangle + h(Ku - b) + \frac{\eta_k}{2}\|u - x_{k-1}\|^2) \\ \bar{x}_k =& (1-\gamma_k)\bar{x}_{k-1} + \gamma_k x_k. \end{aligned}$$

**end for**
Output $\overline{x}_N$.

- same as NAGD, but solves subproblem using subgradient method
- was shown in [4] that it computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient calls, but $\mathcal{O}(\sqrt{L/\varepsilon} + (||K|| /\varepsilon)^2)$ operator calls
- improves gradient calls from A-ADMM and NEST-S, but increases operator calls

**Algorithm 9** Gradient sliding alternating direction method of multipliers (GS-ADMM)

Start: Choose $x_0 \in X$ and set $\overline{x}_0 := x_0$
**for** $k = 1, \ldots, N$ **do**

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1} \\ (\tilde{x}_k, x_k, y_k, z_k) =& \text{ApproxGS}(\nabla f(\underline{x}_k), x_{k-1}, y_{k-1}, z_{k-1}) \\ \overline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k \tilde{x}_k \end{aligned}$$

**end for**
Output $\overline{x}_N$.

- here, the subproblem is approximately solved using a variant of L-ADMM (ApproxGS)
- computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient calls and $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| /\varepsilon)$ operator calls

# Solving (CO) with Sliding

**Algorithm 10** Gradient sliding alternating direction method of multipliers (GS-ADMM)

Start: Choose $x_0 \in X$ and set $\overline{x}_0 := x_0$

**for** $k = 1, \ldots, N$ **do**

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1} \\ (\tilde{x}_k, x_k, y_k, z_k) =& \text{ApproxGS}(\nabla f(\underline{x}_k), x_{k-1}, y_{k-1}, z_{k-1}) \\ \overline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k \tilde{x}_k \end{aligned}$$

**end for**

Output $\overline{x}_N$.

- here, the subproblem is approximately solved using a variant of L-ADMM (ApproxGS)
- computes $\varepsilon$-solution in only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient calls and $\mathcal{O}(\sqrt{L/\varepsilon} + ||K|| /\varepsilon)$ operator calls
- clear improvement from A-ADMM and NEST-S

# Comparison of Algorithms

| | | NAGD | NEST-S | L-ADMM | A-ADMM | GS | AGS | GS-ASMM | OS-ADMM |
|---|---|---|---|---|---|---|---|---|---|
| **Gradient Evals** | $\mathcal{O}\left(\frac{\sqrt{L}}{\epsilon}\right)$ | X | | | | X | X | X | |
| | $\mathcal{O}\left(\frac{L+\|\|K\|\|}{\epsilon}\right)$ | | | X | | | | | |
| | $\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}}+\frac{\|\|K\|\|}{\epsilon}\right)$ | | X | | X | | | | X |
| **Operator Evals** | $\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}}+\frac{\|\|K\|\|}{\epsilon}\right)$ | | X | | X | | X | X | |
| | $\mathcal{O}\left(\frac{L+\|\|K\|\|}{\epsilon}\right)$ | | | X | | | | | |
| | $\mathcal{O}\left(\frac{L+\|\|K\|\|}{\epsilon^2}\right)$ | | | | | X | | | |
| | $\mathcal{O}\left(\frac{\|\|K\|\|}{\epsilon}\right)$ | | | | | | | | X |

# Numerical Example

Problem setting:

- motivated from worst-case instance in [3]

## Numerical Example

Problem setting:

- motivated from worst-case instance in [3]
- $f(x) = \frac{1}{2}\left|\left|Ax - b\right|\right|_2^2, h(x) = \lambda\left|\left|Kx - b\right|\right|_2$

# Numerical Example

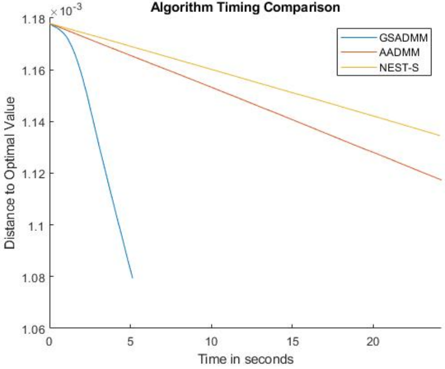
Problem setting:

- motivated from worst-case instance in [3]
- $f(x) = \frac{1}{2} \|Ax - b\|_2^2 , h(x) = \lambda \|Kx - b\|_2$
- sparse $K$, dense $A$

# Numerical Example

Problem setting:

- motivated from worst-case instance in [3]
- $f(x) = \frac{1}{2} ||Ax - b||_2^2 , h(x) = \lambda ||Kx - b||_2$
- sparse $K$, dense $A$



# References


A. Nemirovski and D. Yudin.
*Problem complexity and method efficiency in optimization*.
Wiley-Interscience Series in Discrete Mathematics. John Wiley, XV, 1983.

Yuyuan Ouyang, Yunmei Chen, Guanghui Lan, and Jr. Eduardo Pasiliao.
An accelerated linearized alternating direction method of multipliers.
*SIAM Journal on Imaging Sciences*, 8(1):644–681, 2015.

Yuyuan Ouyang and Yangyang Xu.
Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems.
*Mathematical Programming*, pages 1–35, 2019.

Guanghui Lan.
Gradient sliding for composite optimization.
*Mathematical Programming*, 159(1-2):201–235, 2016.