5-2022

Improved First-Order Techniques for Certain Classes of Convex Optimization

Trevor Squires
Clemson University, tsquire@clemson.edu

Improved first-order techniques for certain classes of convex optimization

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Sciences

---

by
Trevor Squires
May 2022

Accepted by:
Dr. Yuyuan Ouyang, Committee Chair
Dr. Brian Dean
Dr. Yongjia Song
Dr. Boshi Yang

Abstract

The primary concern of this thesis is to explore efficient first-order methods of computing approximate solutions to convex optimization problems. In recent years, these methods have become increasingly desirable as many problems in fields such as machine learning and imaging science have scaled tremendously. Our aim here is to acknowledge the capabilities of such methods and then propose new techniques that extend the reach or accelerate the performance of the existing state-of-the-art literature.

Our novel contributions are as follows. We first show that the popular Conditional Gradient Sliding (CGS) algorithm can be extended in application to objectives with Hölder continuous gradients. CGS has gained much attention in recent years due to its ability to compute an approximate solution without the necessity of a projection oracle. However, it requires both the existence and knowledge of certain smoothness parameters to properly run. We will relax the smoothness requirements and utilize a backtracking linesearch approach to facilitate the algorithm without the knowledge of the relaxed smoothness parameter. In doing so, we will design a new generalized CGS algorithm which also has additional practical benefits over CGS in the smooth case.

Chapter 4 moves our discussion to affinely constrained problems and their limitations. These methods can be solved by alternating direction method of multipliers (ADMM) and are quite popular in fields such as imaging science. We discuss the current lower complexity bounds of solving such problems and suggest a potential method of improvement. By separating the computation of gradient and operator evaluations, we propose two new sliding methods that improve upon the best known convergence rates for such affinely constrained problems. Furthermore, we show that by carefully combining our two new methods, we can obtain a single sliding ADMM method that attains a stronger lower complexity bound for this problem class.

Dedication

I dedicate this dissertation to my friends and family. To my high school classmates, our friendly competitions pushed me to be the best version of myself. In particular, Joshua Hudson always kept me striving for more. I continually revisit our conversations about how "crazy" it would be to get a doctorate in math. Although you are not here to see the finish line, I am forever grateful of our memories. This one's for you, Josh. Thanks for always believing in me.

To Mom and Dad, thanks for unconditionally supporting me through my constant career changes. You always put me in the best positions to succeed. And finally, to my wife, Catherine Kenyon, your encouragement and reliability were crucial these last few years. I never felt worried with you beside me. Without you, this degree and dissertation would not be possible.

Acknowledgments

I would like to thank all of those who have assisted me throughout my graduate career. First of all, I owe a tremendous amount of thanks to my advisor, Dr. Yuyuan Ouyang, for an incredible amount of guidance, preparation, and support during my time as a graduate student. He has served as an excellent mentor not only for conducting high quality research, but also for all around professionalism. I am deeply appreciative for his countless support in all my endeavors.

Dr. Boshi Yang is another member of the OR department that I have been lucky to know. He taught my very first course in optimization and has played a major role in my development as an academic. I'd also like to thank the rest of my committee, Dr. Brian Dean and Dr. Yongjia Song for their support and expertise as they help me complete my dissertation. In general, I would like to thank Clemson and its math department for providing me the opportunity to interact with so many helpful individuals. I have benefited from my interactions with numerous faculty such as Dr. Mishko Mitkovski, Dr. Fei Xue, Dr. Matthew Macauley, and Dr. Christopher McMahan.

Lastly, I'd like to acknowledge my fellow graduate students. I have had the pleasure of knowing excellent senior students to learn from. Sharing an office space with Jason Kurz was one of the best situations one could have asked for. I also appreciate the hours of brainstorming and problem solving with Andrew Pangia. To all of my friends and cohort whose paths have crossed with mine, I have enjoyed your company. I will fondly look back on our game nights, sports teams, and various lunch outings.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This document is intent on finding an $\varepsilon > 0$ solution $\tilde{x}$ to the convex optimization problem

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* \leq \varepsilon$. Here, $f : X \to \mathbb{R}$ is a convex, real-valued function, $\mathbb{R}^n$ is a high dimensional space, and $X \subseteq \mathbb{R}^n$ is a convex, compact set with diameter D_X, or $\mathbb{R}^n$ itself. Under the Euclidean norm $||\cdot||$, we define

$$D_X := \sup_{x,y \in X} ||x - y|| < \infty \tag{1.0.1}$$

when applicable. We restrict ourselves to deterministic first-order algorithms. Although the deterministic restriction makes a sizable cut in our class of algorithms, much of the well received stochastic algorithms are based on a deterministic counterpart. Furthermore, since we assume that n is large, we will find ourselves reluctant to accept algorithms that require higher order information of f as the size of such information does not scale linearly with our problem size. With this issue in mind, we will keep our study to algorithms that only utilize zero- and first-order evaluations of f which we refer to as first-order algorithms. More precisely, the term "deterministic first-order method" is defined by the following oracle description: we say that an iterative algorithm $\mathcal{M}$ for a convex optimization problem (CO) is a deterministic first-order method if it accesses the information of objective function f through a deterministic first-order *oracle* $O_f : \mathbb{R} \times \mathbb{R}^n$, such that $O_f(x) = (f(x), f'(x))$ for any inquiry x, where $f'(x)$ is a subgradient of f at x. Specifically, $\mathcal{M}$ can

be described by a problem independent initial iterate x_0 and a sequence of rules $\{\mathcal{I}_t\}_{t=0}^{\infty}$ such that

$$x_{t+1} = \mathcal{I}_t(O_f(x_0), \ldots, O_f(x_t)),\ \forall t \geq 0.$$

The computational performance of $\mathcal{M}$ is evaluated through its solution accuracy $f(\tilde{x}) - f^*$, in which $\tilde{x}$ is an approximate solution computed by $\mathcal{M}$. Without loss of generality, we can assume that x_t's are both inquiry points to the oracle O and the approximate solution computed by $\mathcal{M}$.

First-order convex optimization in high dimensional spaces is an incredibly active area of research with far reaching applications particularly in the last 30 years. First-order methods have the potential to provide approximate solutions to convex problems while maintaining a low computational complexity. This is in great contrast to higher order methods such as interior point or Newton based methods which provide near exact solutions (i.e. machine precision error) at a slightly higher computational complexity. First-order methods have developed at an accelerated rate recently due to the rising popularity of applications such as machine learning, signal processing, medical imaging, and others. These applications often have large dimension and make higher order methods infeasible. Additionally, applications such as machine learning only require an approximate solution to a problem modeled as (CO) since the model itself is only an approximation to an underlying problem instance. These characteristics make first-order optimization an ideal candidate for solving such applications.

Our research throughout this document is be focused on both the lower and upper complexity bounds of different algorithms and problem settings. For convex optimization problems, the lower complexity bound is concerned with the least number of inquiries to the deterministic first-order oracle in order to compute an ε-approximate solution $\tilde{x}$ such that $f(\tilde{x}) - f^* \leq \varepsilon$ in the asymptotic sense. On the other hand, an upper complexity bound is a depiction of achievable computational performance on solving a specified class of problems by use of a particular algorithm. If the lower complexity bound of a problem class matches the upper complexity bound achieved by a specific algorithm when applied to the aforementioned problem class, we say that said algorithm is optimal for this problem class under the associated oracle. One important aspect of both complexity bounds is the oracle used. By choosing an oracle appropriately, we can simply and accurately portray the complexity behavior of algorithms.

Throughout this document, we consider different settings of (CO) in which we assume

additional properties of f and X. For example, f can be assumed to be sufficiently smooth or strongly convex while X may be assumed to be well structured in order to achieve faster algorithms. While these assumptions restrict our class of problems we can solve, they also potentially accelerate the speed at which we can solve them. This trade-off between algorithm breadth and computational complexity is critical in the discussion of first-order optimization techniques. As we assume more about f or X, our problem class will therefore shrink, and we may potentially achieve a better complexity bound than in the unrestricted setting. However, additional assumptions prevent us from designing truly blackbox algorithms and reduce the number of practical applications. To this end, we prefer to only explore new algorithms under the assumption of additional properties of f and X if our additional assumptions are common in practice *and* an improved upper bound can be established. The rest of this thesis is organized as follows. In Chapter 2 we introduce Nesterov's accelerated gradient descent algorithm and some extensions of the algorithm under different problem settings, in Chapters 3 and 4 we develop new first-order optimization results based upon the well-established methods in Chapter 2 with numerical experiments where appropriate, and in Chapter 5 we summarize the newly developed results with concluding remarks.

Chapter 2

Nesterov's Accelerated Gradient Method and its Extensions

Let us begin exploring methods for solving (CO) under the assumption that f is L-smooth, i.e. f is differentiable and ∇f is Lipschitz continuous with Lipschitz constant $L > 0$. For convex functions in particular, this is equivalent to

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2 \tag{2.0.1}$$

for any $x, y \in X$. We also assume that the projection problem

$$\Pi_X(y) = \Pi(y) := \underset{x \in X}{\operatorname{argmin}} \ ||x - y||^2 \tag{2.0.2}$$

is easy to solve. These two assumptions are quite common in first-order methods and will be the focus in future sections. We refer to problems of (CO) with the L-smoothness assumption as smooth convex optimization. In practice, both L-smoothness and a computationally feasible projection problem are commonplace. Among machine learning in particular, loss functions are frequently modeled using some p-norm, and unit balls/standard simplices are common feasible sets which satisfy this assumption(see [1] for examples). Under this setting, we present Nesterov's accelerated gradient descent algorithm (NAGD) in Algorithm 2.1.

A few remarks can be made regarding NAGD. First, the motivation for each iteration is

Algorithm 2.1 Nesterov's accelerated gradient descent (NAGD)

Start: Select parameters $\gamma_k \in (0,1], \eta_k > 0$. Choose $x_0 \in X$. Set $y_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$z_k = (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1} \tag{2.0.3}$$

$$x_k = \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2 \tag{2.0.4}$$

$$y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k. \tag{2.0.5}$$

end for
Output y_N.

simple: rather than attempting to minimize f directly, we instead iteratively minimize its linear approximation at some point along with a proximal term to keep us close to our previous iterate. Indeed, the subproblem in (2.0.4) is the sum of the linear approximation of f at z_k and a penalty term that decreases as u approaches x_{k-1}. To allow further algorithmic flexibility, NAGD introduces convex combinations z_k and y_k in (2.0.3), (2.0.5) that facilitate what linear approximation we use and what point to output as our solution, respectively. Second, note that the computation of x_k can be rewritten as a projection

$$\begin{aligned} x_k &= \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2 \\ &= \underset{u \in X}{\operatorname{argmin}} \left|\left| u - \left(x_{k-1} - \frac{1}{\eta_k}\nabla f(z_k) \right) \right|\right|^2 \\ &= \Pi \left(x_{k-1} - \frac{1}{\eta_k}\nabla f(z_k) \right) \end{aligned}$$

Since we assumed the projection to be easy to solve, the only significant computational cost per iteration of NAGD is the gradient evaluation in computing x_k. Third, if $\gamma_k \equiv 1$, then $z_k = x_{k-1}, y_k = x_k$ and x_k reduces to a projected gradient descent step. That is, NAGD is equivalent to projected gradient descent with step size $\frac{1}{\eta_k}$ whenever $\gamma_k \equiv 1$ (see [2]).

The convergence analysis of NAGD is well studied. Originally established in [3] and revisited in [4, 5], the following results are proven in [5].

Theorem 2.1. Suppose that we apply NAGD to solve (CO) with parameter $\gamma_k \in [0,1]$. Then the

k-th iterates satisfy

$$f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) \leq \gamma_k\eta_k\left(||x - x_{k-1}||^2 - ||x - x_k||^2\right) + \frac{L\gamma_k^2 - \eta_k\gamma_k}{2}||x_k - x_{k-1}||^2. \tag{2.0.6}$$

Proof. By Lipschitz continuity of gradients and convexity, we have the following:

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k\rangle + \frac{L}{2}||y_k - z_k||^2 \tag{2.0.7}$$

$$f(y_{k-1}) \geq f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle \tag{2.0.8}$$

$$f(x) \geq f(z_k) + \langle \nabla f(z_k), x - z_k\rangle. \tag{2.0.9}$$

Each inequality (2.0.7), (2.0.8), and (2.0.9) follows immediately from convexity of f and (2.0.1). With these inequalities in hand, we write

$$\begin{aligned} f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) \leq\ & f(z_k) + \langle \nabla f(z_k), y_k - z_k\rangle + \frac{L}{2}||y_k - z_k||^2 \\ & - (1-\gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle) \\ & - \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k\rangle). \end{aligned}$$

After simplifying and noting $y_k - z_k = \gamma_k(x_k - x_{k-1})$ from (2.0.5) and (2.0.3), we have

$$\begin{aligned} f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) &\leq \langle \nabla f(z_k), y_k - (1-\gamma_k)y_{k-1} - \gamma_k x\rangle + \frac{L\gamma_k^2}{2}||x_k - x_{k-1}||^2 \\ &= \gamma_k\langle \nabla f(z_k), x_k - x\rangle + \frac{L\gamma_k^2}{2}||x_k - x_{k-1}||^2. \end{aligned}$$

Here, we make use of the definition of y_k in (2.0.5) in the final step. Enforcing the optimality conditions of x_k in (2.0.4) to $\gamma_k\langle \nabla f(z_k), x_k - x\rangle$, we obtain

$$f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) \leq \gamma_k\eta_k\left(||x - x_{k-1}||^2 - ||x - x_k||^2\right) + \frac{L\gamma_k^2 - \eta_k\gamma_k}{2}||x_k - x_{k-1}||^2$$

which completes our proof. □

We now provide two different settings of γ_k and η_k and analyze the convergence rate of NAGD under these parameters.

Corollary 2.1. If we set

$$\gamma_k \equiv 1 \text{ and } \eta_k \equiv L$$

in NAGD, then

$$f(\tilde{y}_N) - f(x^*) \leq \frac{L\left|\left|x^* - x_0\right|\right|^2}{N+1}$$

where $\tilde{y}_N = \sum_{k=0}^{N} y_k/(N+1)$.

Proof. Note that by convexity of f,

$$f(\tilde{y}_N) = f\left(\frac{\sum_{k=0}^{N} y_k}{N+1}\right) \leq \frac{1}{N+1}\sum_{k=0}^{N} f(y_k).$$

Thus, to prove the corollary, it suffices to show that

$$\frac{1}{N+1}\left(\sum_{k=0}^{N} f(y_k)\right) - f(x^*) \leq \frac{L\left|\left|x^* - x_0\right|\right|^2}{N+1}.$$

Following the result of Theorem 2.1 with $\gamma_k \equiv 1$ and $\eta_k \equiv L$, (2.0.6) becomes

$$f(y_k) - f(x) \leq L\left(\left|\left|x - x_{k-1}\right|\right|^2 - \left|\left|x - x_k\right|\right|^2\right) \tag{2.0.10}$$

Summing (2.0.10) over all k, the right hand side becomes a telescoping sum to yield

$$\left(\sum_{k=0}^{N} f(y_k)\right) - (N+1)f(x) \leq L\left(\left|\left|x - x_0\right|\right|^2 - \left|\left|x - x_N\right|\right|^2\right) \leq L\left|\left|x - x_0\right|\right|^2.$$

Setting $x = x^*$ and dividing by $(N+1)$ gives the desired result. □

Corollary 2.2. If we set

$$\gamma_k = \frac{2}{k+1} \text{ and } \eta_k = \frac{2L}{k} \tag{2.0.11}$$

in NAGD, then

$$f(y_N) - f(x^*) \leq \frac{4L}{N(N+1)}\left|\left|x^* - x_0\right|\right|^2.$$

Proof. Letting $x = x^*$ in (2.0.6), noting from (2.0.11) that $1 - \gamma_k = (k-1)/(k+1)$, and multiplying

by $k(k+1)$, we have

$$k(k+1)\left(f(y_k)-f(x^*)\right)-k(k-1)\left(f(y_{k-1})-f(x^*)\right)\leq 4L\left[||x^*-x_{k-1}||^2-||x^*-x_k||^2\right]$$

Summing over k gives us telescoping series that evaluate to

$$N(N+1)\left(f(y_N)-f(x^*)\right)\leq 4L\,||x^*-x_0||^2-4L\,||x^*-x_N||^2\leq 4L\,||x^*-x_0||^2$$

which immediately concludes our result. □

Corollary 2.1 shows that we can prove a $\mathcal{O}(L/\varepsilon)$ convergence result for projected gradient descent using the NAGD framework. However, by choosing γ_k and η_k more aggressively, we can achieve the accelerated complexity of only $\mathcal{O}(\sqrt{L/\varepsilon})$ first-order oracle calls as shown in Corollary 2.2. As a result, NAGD establishes an upper complexity bound for (CO) with an L-smoothness assumption in (2.0.1). While the accelerated convergence rate is indeed a powerful result, [6] by Nemirovski established a lower complexity bound for such a problem class under the first-order oracle assumption. The proof technique for a lower complexity bound alone is one worth reviewing.

In order to construct a lower complexity bound for a problem class, it is sufficient to find a problem instance such that any first-order method struggles to solve it. Nemirovski achieved the latter by constructing a sequence of problem instances such that for any particular first-order method $\mathcal{M}$, each iterate $x_1,\ldots,x_k$ was guaranteed to lie in a particular subspace. By showing that the minimizer among this particular subspace yields an objective value sufficiently far away from the global minimizer, Nemirovski concluded in [6] that no first-order method could achieve better guaranteed performance than the accelerated one given by NAGD. In particular, Nemirovski proved that for any first-order iterative method $\mathcal{M}$ which generates iterates using the first-order oracle assumption, there existed an objective function such that $\mathcal{M}$ requires $\Omega(\sqrt{L/\varepsilon})$ oracle calls to find an ε-approximate solution. With both an upper and lower complexity bound on the order of $\mathcal{O}(\sqrt{L/\varepsilon})$, we conclude that NAGD is optimal for L-smooth instances of (CO) under the first-order oracle. This analysis of the optimality of an algorithm on particular problem classes under certain oracle assumptions will be crucial for improvement in Chapters 3 and 4.

Although NAGD is an optimal first-order method for smooth convex optimization with respect to the first-order oracle, we can continue to extend its usefulness. In this chapter, we explore

some state-of-the-art methods that are immediate adaptations of the NAGD algorithm.

2.1 Universal Gradient Methods

One important assumption for the convergence of NAGD is the smoothness of f. In both the convergence results of Corollary 2.2 and its corresponding parameter setting, the Lipschitz smoothness in (2.0.1) is used quite crucially. However, there are many instances where such Lipschitz constant is not known, may not exist, or the function may not even be differentiable. For example, quadratic objectives of the form

$$f(x) = \frac{1}{2}x^T Ax - b^T x$$

are Lipschitz smooth with constant $L = ||A||$. Explicitly finding L, however, is by no means a trivial task, especially when n is large. In other cases, certain regularization terms in machine learning are modeled using ℓ_1 norms which themselves are nondifferentiable. To allow ourselves further application of accelerated gradient methods, we will need to relax the L-smoothness assumption.

One such possibility is the class of functions with Hölder continuous gradients. Instead of assuming the inequality in (2.0.1), we instead impose the generalized assumption

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M_\nu}{1+\nu} ||x - y||^{1+\nu}, \ \forall x, y \in X. \tag{2.1.1}$$

for some Hölder exponent $\nu \in (0, 1]$ and constant $M_\nu > 0$ along with the previous assumption that X is easy to project to. We refer to functions satisfying (2.0.1) as Hölder smooth which covers both smooth ($\nu = 1$) and weakly smooth ($\nu \in (0, 1)$) cases. In this section and in Chapter 3 to follow, we may also allow for the nonsmooth case ($\nu = 0$) by replacing ∇f with a subgradient f', but these instances are left out for brevity. Nesterov proposed the Universal Gradient Method (UGM) in Algorithm 2.2 for solving (CO) under the Hölder smoothness assumption.

A few immediate remarks can be made regarding UGM. First, if we assume that f is L-smooth with known constant L, then letting $L_k \equiv L$ immediately reduces UGM to NAGD. However, under the Hölder smoothness assumption, we may not have access to such L. Second, to alleviate this issue, Nesterov proposes to find an approximate L_k that satisfies an approximate Lipschitz inequality (2.0.1) locally. Rather than requiring L_k to satisfy (2.0.1) for any $x, y \in X$, we only require that it satisfy the approximate Lipschitz inequality (2.0.1) at y_k and z_k. Third, we must

Algorithm 2.2 Universal Gradient Method (UGM)

Start: Select parameters $\gamma_k \in (0,1], \eta_k > 0$. Choose $x_0 \in X$ and $\varepsilon > 0$. Set $y_0 = x_0$
for $k = 1, \ldots, N$ **do**
Decide $L_k > 0$ satisfying

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} ||y_k - z_k||^2 + \frac{\varepsilon}{2}\gamma_k \tag{2.1.2}$$

where

$$\begin{aligned} z_k =& (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \operatorname*{argmin}_{u \in X} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2, \\ y_k =& (1-\gamma_k)y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for
Output y_N.

specify how L_k is chosen. UGM proposes to search L_k through a backtracking linesearch strategy. In particular, we initialize with any $L_0 \in \mathbb{R}$ and choose $L_1 = 2^i L_0$ where i is the smallest integer such that (2.1.2) is satisfied. At the start of the k-th outer iteration where $k > 1$, we set $L_k = L_{k-1}/2$ and assess the validity of L_k. If it does not satisfy (2.1.2), we keep backtracking and replacing L_k to $2L_k$ until (2.1.2) is satisfied. Fourth, it can be shown that this backtracking procedure only adds a negligible logarithmic term to our complexity and thus such a procedure will not affect our theoretical complexity. Finally, although UGM closely resembles that of NAGD, we should expect performance at least as slow as NAGD since we now cover a broader class of problems.

Indeed, it has been shown in [7] that UGM computes an ε-solution in at most $\mathcal{O}((M_\nu/\varepsilon)^{\frac{2}{1+3\nu}})$ first-order oracle calls. When $\nu = 1$, this matches the lower complexity bound of smooth convex optimization. Furthermore, the classical iteration complexity theory [6] has established that the lower complexity bound on the number of gradient evaluations of ∇f is

$$\Omega\left((M_\nu/\varepsilon)^{\frac{2}{1+3\nu}}\right) \tag{2.1.3}$$

for computing an ε-solution. Thus, UGM is an optimal algorithm for solving Hölder smooth instances of (CO) with respect to the first-order oracle. In addition, UGM does not require the knowledge of Hölder constants ν or M_ν which makes it in some ways more robust than its NAGD counterpart. We will revist universal methods again in Chapter 3.

2.2 Conditional Gradient Methods

In both NAGD and UGM, we assumed that the projection problem (2.0.2) was easy to solve for our feasible set X. Both algorithms are efficient in the sense that they do not require many gradient evaluations. However, there are many cases where the projection is computationally expensive. Take, for example, $X = \text{conv}(v_1, \ldots, v_p)$ for some $p \in \mathbb{N}$. For suitably large p, the projection problem for X is a quadratic program which can be difficult to solve. In particular, if f itself is a quadratic, the projection onto the convex hull can be more difficult to solve than minimizing f. For such problem instances, NAGD and UGM are of little use. In order to have any progress in this direction, we must look for algorithms which do not require a projection, namely projection-free methods.

In 1956, Frank and Wolfe developed one of the earliest first-order methods for solving general convex programming problems without the use of projections (see [8]). They proposed Algorithm 2.3 below for solving (CO) under the L-smoothness assumption and compactness of X.

Algorithm 2.3 Conditional Gradient (CG) method

Start: Select parameters $\gamma_k \in (0, 1], \eta_k > 0$. Choose $y_0 \in X$.
for $k = 1, \ldots, N$ **do**

$$x_k = \underset{x \in X}{\text{argmin}} \langle \nabla f(y_{k-1}), x \rangle \tag{2.2.1}$$
$$y_k = (1 - \alpha_k) y_{k-1} + \alpha_k x_k$$

end for
Output y_N.

The Conditional Gradient (CG) algorithm (also sometimes called Frank-Wolfe or FW), removes the proximal term from the x_k subproblem and chooses to only solve a linear optimization problem over X instead of a projection. The proximal term in (2.0.4) was useful in ensuring that future iterates maintain information from previous iterates, but also made the subproblem potentially computationally infeasible by adding an extra quadratic term. Accordingly, one should expect that CG requires more iterations to compute an ε-solution to (CO), but the cost per iteration no longer includes a projection. Instead, the potentially expensive operations are gradient evaluations, and linear optimization problems. For many problem settings, a linear optimization problem is far more computationally feasible than a projection. Indeed, a linear optimization over the convex hull set previously discussed is simply a linear program. It is shown in [9] that for properly chosen α_k,

CG can compute an ε-solution in $\mathcal{O}(LD_X^2/\varepsilon)$ iterations. Noting that one CG iteration requires one gradient evaluation, we see that although CG does require asymptotically more gradient evaluations for an approximate solution, it does not require any projections. Furthermore, it is also shown in [9] that any algorithm which computes a solution x_k via

$$y_k \in \operatorname*{Argmin}_{x \in X} \langle p_k, x \rangle$$
$$x_k \in \operatorname{conv}\{y_0, \ldots, y_k\}$$

must make at least $\Omega(LD_X^2/\varepsilon)$ linear optimization calls for an ε solution, which makes CG an optimal algorithm in some sense.

We cannot, however, conclude that CG is optimal among first-order methods for gradient evaluations. In fact, we know from the NAGD upper bound that only $\mathcal{O}(\sqrt{LD_X^2/\varepsilon})$ gradient evaluations are required. It should also be noted that we include the D_X^2 here in the convergence result since X must be compact in order for (2.2.1) to have an optimal solution. If one assumes compactness in the previous section, similar diameter terms can be explicitly included. It remains to be shown if one can design a projection-free method to compute an ε-solution with both the optimal number of linear optimizations as well as the optimal number of gradient evaluations. In 2015, Lan proposed the Conditional Gradient Sliding (CGS) method in Algorithm 2.4 to answer this question (see [10]).

Algorithm 2.4 Conditional Gradient Sliding (CGS)

Start: Select parameters $\gamma_k \in (0, 1], \eta_k > 0$. Choose $x_0 \in X$. Set $y_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$z_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_{k-1}, \tag{2.2.2}$$
$$x_k = \text{CndG}(\nabla f(z_k), x_{k-1}, \eta_k, \varepsilon_k)$$
$$y_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_k. \tag{2.2.3}$$

end for
Output y_N.

The idea behind CGS is the following: NAGD provides us with a framework for solving (CO) using an optimal number of gradient evaluations. If the projection subproblem can be addressed in a projection-free way without any additional gradient evaluations, then such an algorithm would be a projection-free method with the optimal number of gradient evaluations. Algorithm 2.4 simply

Algorithm 2.5 Conditional Gradient for Projection Problems (CndG)

Initialize $t = 1$ and $u^0 = u$.
while true **do**
Compute v^t such that

$$\max_{x \in X} \langle g + \beta(u^{t-1} - u), v^t - x \rangle \leq 0 \tag{2.2.4}$$

If

$$\max_{x \in X} \langle g + \beta(u^{t-1} - u), u^{t-1} - x \rangle \leq \eta, \tag{2.2.5}$$

terminate with $u^+ = u^{t-1}$. Otherwise, set

$$u^t = (1 - \alpha^t)u^{t-1} + \alpha^t v^t$$

and $t = t + 1$
end while

replaces x_k with an approximate solution to the projection problem that is computed via a projection-free method in CndG. Here, the CndG method in Algorithm 2.5 is a specialized variant of CG that approximately solves the induced projection problem

$$\min_{x \in X} \phi(x) := \langle g, x \rangle + \frac{\beta}{2} ||x - u||^2 . \tag{2.2.6}$$

There is, however, one issue to be addressed with CGS. If CndG computes a rough approximation x_k to the projection problem, then the iterates z_k, x_k, and y_k will no longer resemble those of the original NAGD iterates. Thus, CGS may require more gradient evaluations to compute a good solution y_N. On the other hand, if CndG computes a precise solution to the projection problem, then we will need to do many CndG iterations and may potentially require more linear optimizations. Lan provided the following parameter setting himself in [10] that addressed both concerns.

Theorem 2.2. If β_k, γ_k, and η_k in Algorithm 2.4 are set to

$$\beta_k = \frac{3L}{k+1}, \gamma_k = \frac{3}{k+2}, \text{ and } \eta_k = \frac{LD_X^2}{k(k+1)}, \forall k \geq 1,$$

then for any $k \geq 1$,

$$f(y_k) - f^* \leq \frac{15LD_X^2}{2(k+1)(k+2)}.$$

As a consequence, with properly chosen α^t, the total number of calls to the first-order and linear optimization oracles performed by the CGS method for finding an ε-solution of (CO) can be bounded by $\mathcal{O}(\sqrt{LD_X^2/\varepsilon})$ and $\mathcal{O}(LD_X^2/\varepsilon)$ respectively.

Thus, we conclude that CGS is optimal in both the number of gradient evaluations and linear optimizations required. This idea of using NAGD as an algorithm template for only computing the optimal number of gradient evaluations is a technique called sliding and was developed in 2010 by Lan [11]. We will return to sliding algorithms Chapter 4.

2.3 Alternating Direction Method of Multipliers

We begin our final algorithm review with a more specialized problem class. For the rest of this section, we will be considering problem instances of the form

$$F^* := \min_{x \in X} f(x) + h(Ax - b) \tag{2.3.1}$$

where we introduce new constants $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ along with a Lipschitz continuous, but possibly nonsmooth function $h : \mathbb{R}^n \to \mathbb{R}$ and assume that $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth. We also assume that the projection mapping involving X and proximal mapping involving h are easy to solve, i.e. (2.0.2) and

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

can be solved exactly with little computational cost. If we let $F(x) = f(x) + h(Ax - b)$, then the above problem falls under the model (CO) with additional structure. However, this specific problem class in (2.3.1) is of particular interest since it is equivalent to an affinely constrained optimization problem

$$F^* := \min_{(x,z) \in \mathcal{H}} F(x, z) := f(x) + h(z) \text{ where } \mathcal{H} := \{ (x, z) \in X \times Z \mid Kx - z = 0 \}. \tag{ACO}$$

Here we introduce an extra variable $z \in Z \subseteq \mathbb{R}^m$ to account for the linear operation Kx in the original problem (2.3.1). Problems of the form in (ACO) have found numerous applications in machine learning and image processing and are of specific interest. Specifically, the two-dimensional total variation based image reconstruction problem arises in many settings. We will see this problem in more detail in the numerical experiments for Chapter 4.

While problem (2.3.1) is equivalent to (ACO) with $Z = \mathbb{R}^m$, our analysis of problem (ACO) can be extended to a general closed convex set Z. By studying the augmented Lagrangian formula-

tion of (ACO), namely,

$$\mathcal{L}(\rho; u, v, w) = f(u) + h(w) + \langle v, Ku - w\rangle + \frac{\rho}{2}\|Ku - w\|^2,$$

we can apply the classical alternating direction method of multipliers (ADMM) to solve (ACO). The iterations of ADMM are described in Algorithm 2.6. In ADMM, the x_k and z_k steps minimize

Algorithm 2.6 Alternating direction method of multipliers (ADMM) for (ACO)

Start: Choose $x_0 \in X$, $y_0 \in \mathbb{R}^m$, and $z_0 \in Z$.
for $k = 1, \ldots, N$ **do**
Compute

$$x_k = \operatorname*{argmin}_{u \in X} \mathcal{L}_\rho(u, y_{k-1}, z_{k-1}), \tag{2.3.2}$$

$$z_k = \operatorname*{argmin}_{w \in \mathbb{R}^m} \mathcal{L}_\rho(x_k, y_{k-1}, w), \tag{2.3.3}$$

$$y_k = y_{k-1} + \rho(Kx_k - z_k). \tag{2.3.4}$$

end for
Output x_N.

the augmented Lagrangian $\mathcal{L}_\rho(u, v, w)$ alternatively with respect to its primal variables. The y_k step is a dual ascent step on the dual of problem (ACO). First developed in [12, 13], the ADMM described above can be understood as a two-block variant of the augmented Lagrangian method (ALM) proposed in [14, 15]. There has been extensive study on the analysis of ALM and ADMM. See, e.g., the monograph [16] and the references within for the review on ALM and ADMM. See also [17, 18] and the references within for the convergence analysis of first-order method variants of ADMM that uses gradient computations ∇f and linear operations (involving K and $K^\top$). It is known that the rate of convergence of ADMM is on the order $\mathcal{O}((L + ||K||)/k)$ (see [19, 20]).

However, it is possible that the projection problem involving x_k is not easy to solve even under our assumption that X is easy to project to. Specifically, the optimization problem in the x_k iteration (2.3.2) is

$$x_k = \operatorname*{argmin}_{u \in X} f(u) + \langle K^\top y_{k-1}, u\rangle + \frac{\rho}{2}\|Ku - z_{k-1}\|^2.$$

The above problem is not necessarily easy to solve due to the function $f(u)$ and the quadratic term $\|Ku - z_{k-1}\|^2$. To avoid solving such sophisticated problems, one may consider replacing the

troublesome terms with their linear approximations:

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

Replacing the x_k step in (2.3.2) by the above approximation, we obtain a variant of ADMM commonly known as the linearized ADMM (L-ADMM). The linearized ADMM can be analyzed through the oracle complexity theory with oracle

$$O(x, y) = (f(x), \nabla f(x), Kx, K^T y). \tag{2.3.5}$$

Here, we choose to include Kx and $K^T y$ in the oracle since those matrix-vector multiplications may be computationally expensive. Each iteration requires one inquiry on the gradient $\nabla f(x_{k-1})$ and one inquiry on the matrix-vector multiplications involving K and $K^\top$ (note that it requires Kx_{k-1} and Kx_k in each iteration but Kx_{k-1} can be treated as an inquiry that has already been made in the previous iteration).

Some variants of linearized ADMM and their convergence analysis are studied in [17]. It is proved that the aforementioned linearized ADMM has rate of convergence $\mathcal{O}((L + \|K\|)/k)$. Moreover, an accelerated linearized ADMM is proposed in [17] which has improved rate of convergence. The key idea is based on the observation of the accelerated gradient method in (2.0.3)–(2.0.5) that the introduction of convex combinations z_k and y_k can improve the rate of convergence when minimizing smooth convex functions. The iterations of the accelerated linearized preconditioned ADMM proposed in [17] for solving problem (2.3.1) is described in Algorithm 2.7.

We will refer to Algorithm 2.7 as the accelerated ADMM (A-ADMM). It is proved in [17] that A-ADMM has rate of convergence $\mathcal{O}(L/k^2 + \|K\|/k)$. Such a rate is better than that of the linearized ADMM, especially when L is much larger than $\|K\|$. The key differences between the A-ADMM and the L-ADMM that contributed to the improved rate are the introduction of z_k and y_k, and the variable constants θ_k, τ_k and ρ_k (while in L-ADMM) $\theta_k \equiv \tau_k \equiv \rho_k \equiv \rho$). The A-ADMM can also be analyzed through the oracle complexity theory; in each iteration it requires one inquiry on the gradient $\nabla f(z_k)$ and one inquiry on the matrix-vector multiplications involving K and $K^\top$.

With our previously defined oracle, we are now ready to review the efficiency of any first-

Algorithm 2.7 Accelerated ADMM (A-ADMM) for (ACO)

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

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

end for
Output y_N.

order methods for solving (ACO) that access information of the function f and the operator K through the first-order oracle O. Under this oracle setting, L-ADMM requires at most $\mathcal{O}((L + ||K||)/\varepsilon)$ inquiries to O in (2.3.5) as was shown in [17]. It's improved version A-ADMM, achieves a better complexity with respect to the Lipschitz constant, namely $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$. In addition, a worst-case instance of (2.3.1) was designed in [18] such that any first-order method that calls O in (2.3.5) will need at least $\Omega(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ inquiries for an ε-solution. We can then conclude that A-ADMM is theoretically unimprovable under our particular oracle assumption.

Chapter 3

Universal Conditional Gradient Sliding

In Chapter 2, we discussed a class of algorithms known as conditional gradient methods for solving (CO) without the use of projections. These projection free methods were shown to be optimal with respect to the number of linear optimizations required, and in the case of CGS in Algorithm 2.4, also optimal with respect to the number of gradient evaluations. However, a key assumption was that our objective function f satisfied the L-smooth inequality in (2.0.1). Also in Chapter 2, we discussed methods of relaxing this condition. In this section, we attempt to combine the two to design an optimal projection free algorithm which does not require the objective to be L-smooth.

Let us restate our problem at hand. We study first-order projection free methods for computing ε-approximate solutions to convex optimization problems of the form (CO) where $X \subseteq \mathbb{R}^n$ is a high-dimensional compact, convex set, and f is a convex function that satisfies the Hölder smoothness condition (2.1.1). For the general case when $\nu \in (0, 1]$, universal methods have been developed in [21, 22] that compute ε-solutions with at most $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{1}{\nu}})$ gradient evaluations of ∇f and linear objective optimization subproblems solves. Focusing on the number of gradient evaluations of ∇f required by the aforementioned projection-free methods, we can observe a significant gap with the lower complexity bound in (2.1.3). For example, the number of gradient evaluations required by the universal methods in [21, 22] is upper bounded by $\mathcal{O}(1/\varepsilon^3)$ when $\nu = 1/3$. This complexity

is significantly worse than the lower complexity bound in (2.1.3) which is of order $\Omega(1/\varepsilon)$ when $\nu = 1/3$. For the special smooth case (when $\nu = 1$), the number of gradient evaluations required by the methods in [9, 23, 24] are upper bounded by $\mathcal{O}(1/\varepsilon)$ and is significantly worse than the $\mathcal{O}(1/\sqrt{\varepsilon})$ lower complexity bound in (2.1.3).

The conditional gradient sliding technique discussed previously allows us to close the gap in the gradient evaluations whenever $\nu = 1$. However, it remains to be seen if we can obtain the $\mathcal{O}((M_\nu D_X^2/\varepsilon)^{\frac{2}{1+3\nu}})$ complexity for gradient evaluations in the weakly smooth case. In this chapter, we propose to close the remaining gap in the gradient evaluations of ∇f between its upper complexity bounds in projection-free methods and the lower complexity bounds in (2.1.3). Specifically, we propose a novel first-order projection-free method, namely the universal conditional gradient sliding (UCGS) method, that is able to compute an ε-solution of the problem (CO) without requiring any projections or knowledge of the smoothness information (ν, M_ν). The framework of UCGS is built around that of the universal gradient and conditional gradient sliding methods in [7] and [10], respectively. The outline of this chapter is summarized below.

First, in terms of gradient evaluations of ∇f or calls to the first-order oracle, the total number of evaluations required by the proposed UCGS method for computing an ε-solution is upper bounded uniformly by $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{2}{1+3\nu}})$ for any $\nu \in (0, 1]$. Such a bound matches the lower complexity bound in (2.1.3). To the best of our knowledge, this is the first first-order projection-free method that is able to achieve such gradient evaluation complexity bound uniformly for smooth and weakly smooth convex optimization problems.

Second, the total number of calls to the linear optimization oracle required by the proposed UCGS method for computing an ε-solution is upper bounded uniformly by $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{4}{1+3\nu}})$ for any $\nu \in (0, 1]$. Comparing with the $\mathcal{O}((M_\nu D_X^{1+\nu}/\epsilon)^{\frac{1}{\nu}})$ result [21, 22] in the literature, the proposed UCGS method has the same complexity when $\nu = 1$ and is significantly better for all $\nu \in (0, 1)$. For example, when $\nu = 1/3$, the UCGS method has a significantly better complexity of $\mathcal{O}(1/\varepsilon^2)$ compared the $\mathcal{O}(1/\varepsilon^3)$ result in [21, 22]. Within the class of sliding-type algorithms following the work of [10, 11], to the best of our knowledge, this is the first time a sliding-type algorithm is able to improve not only the gradient complexity but also the overall complexity for computing an approximate solution.

Third, the proposed UCGS method is able to achieve the aforementioned complexity bounds without any knowledge of the smooth information (ν, M_ν) of the objective function. Therefore, it is

a universal method that is able to solve weakly smooth and smooth convex optimization problems with the best possible $\nu \in (0, 1]$ and $M_\nu > 0$. Note that in the special smooth case when $\nu = 1$, the proposed UCGS method can be understood as an extension of the CGS method with additional features for practical implementation. In such case, it has the same complexity results as the CGS method and its backtracking linesearch edition in terms of both gradient evaluations of ∇f and linear objective subproblems (see [25]). However, unlike the linesearch edition, by incorporating a different backtracking linesearch strategy with a novel parameter choice, UCGS no longer require any information on the continuity constant M_1. UCGS also allows that all linear objective optimization subproblems be solved approximately within certain accuracy while maintain the same complexity results.

3.1 Conditional Gradient Sliding in the Hölder Case

In this section, we analyze the conditional gradient and conditional gradient sliding methods under the relaxed Hölder smooth condition. While CGS can already achieve better gradient evaluation than that of CG for problems with Lipschitz continuous gradients, our result covers a more general case of problems with Hölder continuous gradients. Moreover, we also show a theoretical result that CGS can also achieve better complexity on linear objective optimizations than that of CG when the Hölder continuity exponent $\nu \in (0, 1)$. Such theoretical result is particularly interesting within the class of sliding-type algorithms following the works of [10, 11]. To the best of our knowledge, this is the first time a sliding-type algorithm is able to improve not only the gradient complexity but also the overall complexity for computing an approximate solution.

Let us consider again the CGS algorithm in Algorithm 2.4 and make a few additional remarks. First, in both Algorithms 2.4 and in the sequel, we refer to the operations between increments in t as an inner iteration and that of k as an outer iteration. To distinguish inner and outer iteration descriptions, we use subscripts and superscripts to denote outer and inner iteration indices, respectively. Second, the relation (2.2.4) in the CndG procedure allows for both projection-based and projection-free implementations. For example, if we require $\eta_k \equiv 0$, then the CndG procedure solves a projection problem and the iterate x_k computed by the procedure is an optimal

solution to the projection problem

$$\min_{x\in X} \phi_k(x) := \langle \nabla f(z_k), x\rangle + \frac{\beta_k}{2}\left|\left|x - x_{k-1}\right|\right|^2.$$

Consequently, CGS reduces to a version of Nesterov's accelerated gradient method in Algorithm 2.1 (see, e.g., [2]).Third, if $\beta_k \equiv 0$, then satisfying the condition (2.2.4) becomes a linear objection optimization and it takes exactly one inner iteration for CndG to compute an optimal solution to this subproblem. Note that by the description of v^t in (2.2.4), v^t is the optimal solution to the linear subproblem. If instead we allow the right hand side of (2.2.4) to be nonzero, then we can study practical implementation variants of CG that solve the linear objective optimization subproblem approximately (see, e.g., [24] and the references within). However, we will focus on theoretical analysis in this section; the approximate linear subproblem implementation will be discussed in next section.

In this section, we show that the CGS method can be applied to not only smooth, but also weakly smooth problems. An interesting discovery we make use of this section, is that the use of sliding also reduces the total number of inner iterations, and consequently linear objective optimizations when the Hölder exponent $\nu \in (0, 1)$.

We now analyze the performance of Algorithm 2.4 under various parameter settings. We begin by building a recurrence relation on the outer iterates. Such recurrence provides us a tool for performing complexity analysis on CGS.

Proposition 3.1. Suppose that $\gamma_k \in [0, 1]$ for all k in Algorithm 2.4. Then

$$\begin{aligned} &f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) \\ \leq &\gamma_k\eta_k + \frac{\beta_k\gamma_k}{2}(||x_{k-1} - x||^2 - ||x_k - x||^2) \\ &- \frac{\beta_k\gamma_k}{2}\left|\left|x_k - x_{k-1}\right|\right|^2 + \frac{M_\nu\gamma_k^{1+\nu}}{1+\nu}\left|\left|x_k - x_{k-1}\right|\right|^{1+\nu},\ \forall k \geq 1, x \in X. \end{aligned} \tag{3.1.1}$$

Specifically, if $\nu \in (0, 1)$ and $\beta_k > 0$ for all k, then

$$\begin{aligned} &f(y_k) - (1-\gamma_k)f(y_{k-1}) - \gamma_k f(x) \\ \leq &\gamma_k\eta_k + \frac{\beta_k\gamma_k}{2}(||x_{k-1} - x||^2 - ||x_k - x||^2) + \xi_k,\ \forall k \geq 1, x \in X, \end{aligned} \tag{3.1.2}$$

where

$$\xi_k := \frac{1-\nu}{2(1+\nu)} M_\nu^{\frac{2}{1-\nu}} \left(\frac{\gamma_k}{\beta_k}\right)^{\frac{1+\nu}{1-\nu}}. \tag{3.1.3}$$

Proof. From the Hölder condition (2.1.1) and the convexity of f we have

$$\begin{aligned}
&f(y_k) - (1-\gamma_k) f(y_{k-1}) - \gamma_k f(x) \\
\le& f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{M_\nu}{1+\nu} \|y_k - z_k\|^{1+\nu} \\
&- (1-\gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle) - \gamma_k (f(z_k) + \langle \nabla f(z_k), x - z_k \rangle) \\
=& \gamma_k \langle \nabla f(z_k), x_k - x \rangle + \frac{M_\nu \gamma_k^{1+\nu}}{1+\nu} \|x_k - x_{k-1}\|^{1+\nu}.
\end{aligned}$$

Here the last equality is from the definitions of z_k and y_k in (2.2.2) and (2.2.3) respectively. Noting that x_k is computed from the CndG procedure which satisfies (2.2.5), we have

$$\begin{aligned}
\eta_k &\ge \langle \nabla f(z_k) + \beta_k (x_k - x_{k-1}), x_k - x \rangle \\
&= \langle \nabla f(z_k), x_k - x \rangle + \frac{\beta_k}{2} (\|x_k - x_{k-1}\|^2 + \|x_k - x\|^2 - \|x_{k-1} - x\|^2)
\end{aligned}$$

for any $x \in X$. Summarizing the above two relations we obtain (3.1.1). By Young's inequality (applied to the product of $(\beta_k \gamma_k/(1+\nu))^{(1+\nu)/2} \|x_k - x_{k-1}\|^{1+\nu}$ and $M_\nu (\gamma_k/\beta_k)^{(1+\nu)/2} (1+\nu)^{-(1-\nu)/2}$ with exponents $2/(1+\nu)$ and $2/(1-\nu)$ respectively) we conclude the next result (3.1.2) from (3.1.1). □

In Proposition 3.1 above, there is a recurrence relation concerning weights $(1-\gamma_k)$. The following notation is be used in the sequel for analyzing the complexity of CGS:

$$\Gamma_k = \begin{cases} 1 & k = 1 \\ \Gamma_{k-1}(1-\gamma_k) & k > 1. \end{cases} \tag{3.1.4}$$

We use the following simple lemma for analyzing the sum of recurrent terms.

Lemma 3.1. Suppose that $\{a_k\}, \{b_k\} \subset \mathbb{R}$ and $\{\gamma_k\} \subset [0,1]$ are sequences that satisfy $\gamma_1 = 1$ and

$$a_k \le (1-\gamma_k) a_{k-1} + \gamma_k b_k, \ \forall k \ge 1. \tag{3.1.5}$$

Then we have

$$a_k \leq \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} b_i, \tag{3.1.6}$$

for all $k \geq 1$.

Proof. Dividing both sides of (3.1.5) by Γ_k we obtain a series of inequalities with telescoping terms. Summing up we obtain (3.1.6). □

We are now ready to derive results on the complexity of CG as a special case of CGS. Theorem 3.1 below is a known complexity result of CG for problems with Hölder continuous gradients [21, 22].

Theorem 3.1 (see also [21, 22])**.** Suppose that we apply CGS in Algorithm 2.4 with parameters $\beta_k \equiv 0$, $\eta_k \equiv 0$, $\gamma_k = 2/(k+1)$ to compute an ε-solution to problem (CO) with Hölder exponent ν and constant M_ν. Then CGS requires at most N_{grad} gradient evaluations and N_{lin} linear objective optimizations, in which

$$N_{\text{lin}} = N_{\text{grad}} = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{1}{\nu}}\right). \tag{3.1.7}$$

Proof. Since $\gamma_k = 2/(k+1)$, by (3.1.4) we have $\Gamma_k = 2/(k(k+1))$ and hence $\gamma_k/\Gamma_k = k$. Applying Proposition 3.1 and noting Lemma 3.1 with our parameter settings, we have for any $x \in X$ that

$$f(y_N) - f(x) \leq \frac{2M_\nu}{N(N+1)(1+\nu)} \sum_{k=1}^{N} k \left(\frac{2}{k+1}\right)^{\nu} \|x_k - x_{k-1}\|^{1+\nu} \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^\nu}\right).$$

Thus, in order to obtain an ε-solution, we require at most N_{grad} outer iterations. Moreover, noting that $\alpha^1 = 1$ and $\beta = 0$ in the CndG procedure, comparing (2.2.4) and (4.1.2) we observe that CndG always terminates after one inner iteration. Therefore, the total number of gradient evaluations and linear optimizations must both be upper bounded by (3.1.7). □

As pointed in the remarks after the description of Algorithm 2.4, CGS with $\beta_k \equiv 0$ reduces to a CG variant with the same convergence results. Therefore, Theorem 3.1 above provides a complexity result for the CG algorithm applied to functions with Hölder continuity exponent $\nu \in (0, 1]$. One achieves similar results to Theorem 3.1 when choosing different γ_k (e.g., $\gamma_k = 1/k$; see [21] for other choices of γ_k). It should also be noted that the choice of $\eta_k \equiv 0$ does not affect the above analysis;

indeed, with $\beta = 0$ and any $\eta \geq 0$, the CndG procedure always terminates after one inner iteration. However, as we describe below, if $\beta > 0$, the choice of η will affect the number of inner iterations performed by the CndG procedure before termination. The proposition below is a known complexity result (see Theorem 2.2(c) in [10]) of CG for solving projection problems. For completeness, we prove it later as an immediate consequence of Proposition 3.3.

Proposition 3.2. In the CndG procedure for computing an approximate solution to the projection problem (2.2.6), if choose $\alpha^t = 2/(t+1)$, then

$$\min_{j=0,\ldots,t} \max_{x \in X} \langle \nabla \phi(u^j), u^j - x \rangle \leq \frac{6\beta D_X^2}{t}, \ \forall t \geq 1.$$

Proposition 3.2 provides insight on the number of inner iterations required by the CndG procedure in Algorithm 2.5 to solve the projection problem (2.2.6) approximately. For example, if we set $\eta_k \geq 6\beta_k D_X^2$, then the CndG procedure always terminates after exactly one iteration. Noting that $u^0 = u$ in CndG, we can observe that CGS reduces to CG not only when $\beta_k \equiv 0$ (as stated previously in the remarks of CGS and after Theorem 3.1), but also when $\beta_k > 0$ and $\eta_k \geq 6\beta_k D_X^2$. The latter observation is important for our analysis: as described in the following theorem, for problems with Hölder continuous exponent $\nu \in (0, 1)$, the latter observation allows us to perform a simple analysis of CG that is different from the current literature [21, 22]. Such simple analysis leads to our interesting discovery that sliding could improve the complexity of linear objective optimizations.

Theorem 3.2 (see also [21, 22])**.** Assume in (CO) that the Hölder exponent $\nu \in (0, 1)$. Suppose that we apply CGS in Algorithm 2.4 with parameters $\beta_k > 0$, $\eta_k = 6\beta_k D_X^2$, and $\alpha^1 = 1$ in Algorithm 2.5. Then we have for any $x \in X$ that

$$\begin{aligned} f(y_N) - f(x) \leq & \Gamma_N \sum_{k=1}^{N} \frac{\beta_k \gamma_k}{2\Gamma_k} (12 D_X^2 + ||x_{k-1} - x||^2 - ||x_k - x||^2) \\ & + \frac{1}{\Gamma_k} \frac{1-\nu}{2(1+\nu)} M_\nu^{\frac{2}{1-\nu}} \left(\frac{\gamma_k}{\beta_k} \right)^{\frac{1+\nu}{1-\nu}}. \end{aligned} \tag{3.1.8}$$

Specifically, if we set $\beta_k = M_\nu \gamma_k^\nu / D_X^{1-\nu}$ and $\gamma_k = 2/(k+1)$, to compute an ε-solution to problem (CO) with Hölder exponent ν and constant M_ν, CGS requires at most N_{grad} gradient evaluations and N_{lin} linear objective optimizations, in which $N_{\text{lin}} = N_{\text{grad}} = \mathcal{O}\left(\left(M_\nu D_X^{1+\nu} / \varepsilon \right)^{\frac{1}{\nu}} \right)$.

Proof. Applying Proposition 3.1 and noting Lemma 3.1, with our choice of η_k we have (3.1.8). Consequently,

$$f(y_N) - f(x) \leq \Gamma_N \sum_{k=1}^{N} \frac{7\beta_k \gamma_k D_X^2}{\Gamma_k} + \frac{1}{\Gamma_k} \frac{1-\nu}{2(1+\nu)} M_\nu^{\frac{2}{1-\nu}} \left(\frac{\gamma_k}{\beta_k}\right)^{\frac{1+\nu}{1-\nu}}, \ \forall x \in X. \tag{3.1.9}$$

Substituting to (3.1.9) the values of β_k, γ_k, and noting Γ_k from (3.1.4), we have

$$f(y_N) - f(x) \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^2}\right) \sum_{k=1}^{N} \frac{k}{(k+1)^\nu} \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^\nu}\right).$$

Thus, in order to obtain an ε-solution, we require at most N_{grad} outer iterations. Moreover, noting that $\alpha^1 = 1$ and $\eta = 6\beta D_X^2$ in the CndG procedure, by Proposition 3.2 we observe that the CndG procedure always terminates after one inner iteration. Therefore the total number of linear optimizations is upper bounded by $N_{lin} = N_{grad}$. □

In the above theorem, we observe an imperfection by in the derivation from (3.1.8) and (3.1.9), although we obtain the same complexity result of CG as in Theorem 3.1. Specifically, due to the existence of the dominant term D_X^2, we can only simply bound the telescoping difference $(||x_{k-1} - x||^2 - ||x_k - x||^2)$ by D_X^2 to obtain (3.1.9). As a consequence, even if we attempt to choose the best $\beta_k = \mathcal{O}(M_\nu \gamma_k^\nu / D_X^{1-\nu})$ to minimize the right hand side of (3.1.9), the complexity result remains to be $\mathcal{O}\left(\left(M_\nu D_X^{1+\nu}/\varepsilon\right)^{\frac{1}{\nu}}\right)$. Noting that the imperfection we observe is due to the choice that $\eta_k = 6\beta_k D_X^2$, we may choose a smaller η_k setting to improve the complexity results, as stated in the theorem below.

Theorem 3.3. Assume in problem (CO) that the Hölder exponent $\nu \in (0, 1)$. Suppose that we apply CGS in Algorithm 2.4 with parameters

$$\beta_k = \frac{M_\nu k^{\frac{1-3\nu}{2}}}{D_X^{1-\nu}}, \eta_k = \frac{6\beta_k D_X^2}{k}, \text{ and } \gamma_k = \frac{2}{k+1}.$$

Then, to compute an ε-solution to problem (CO) with Hölder exponent ν and constant M_ν, CGS requires at most N_{grad} gradient evaluations and N_{lin} linear objective optimizations, in which

$$N_{\text{grad}} = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right) \text{ and } N_{\text{lin}} = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{4}{1+3\nu}}\right).$$

for any $\nu \in (0,1)$.

Proof. Since $\gamma_k = 2/(k+1)$, we have $\Gamma_k = 2/(k(k+1))$ and hence $\gamma_k/\Gamma_k = k$. Applying Proposition 3.1 and noting Lemma 3.1, with our choice of parameters

$$\begin{aligned} f(y_N) - f(x) \leq & \frac{2}{N(N+1)} \sum_{k=1}^{N} 6\beta_k D_X^2 + \frac{k\beta_k}{2} \left(||x_{k-1} - x||^2 - ||x_k - x||^2 \right) \\ & + \frac{\xi_k}{\Gamma_k}, \ \forall x \in X. \end{aligned}$$

Noting that $k\beta_k$ is increasing, we have

$$\begin{aligned} & \sum_{k=1}^{N} k\beta_k \left(||x_{k-1} - x||^2 - ||x_k - x||^2 \right) \\ =& \beta_1 \|x_0 - x\|^2 + \sum_{k=1}^{N} ((k+1)\beta_{k+1} - k\beta_k) \|x_k - x\|^2 - N\beta_N \|x_N - x\|^2 \\ \leq & \beta_1 D_X^2 + \sum_{k=1}^{N} ((k+1)\beta_{k+1} - k\beta_k) D_X^2 = N\beta_N D_X^2. \end{aligned}$$

Combining the above two relations and noting our choice of β_k and the description of ξ_k in (3.1.3) we have

$$f(y_N) - f(x) \leq \mathcal{O}\left(\frac{M_\nu D_X^{1+\nu}}{N^2} \right) \left[\sum_{k=1}^{N} k^{\frac{1-3\nu}{2}} + N^{\frac{3-3\nu}{2}} + \sum_{k=1}^{N} \frac{k^{\frac{1+3\nu^2}{2(1-\nu)}}}{(k+1)^{\frac{2\nu}{1-\nu}}} \right], \ \forall x \in X. \quad (3.1.10)$$

Thus, to obtain an ε-solution, we need at most N_{grad} outer iterations, or equivalently, at most N_{grad} gradient evaluations. Also, from Proposition 3.2, if $\eta_k = 6\beta_k D_X^2/k$, then we perform at most k inner iterations per outer iteration. Thus, the total number of inner iterations and consequently linear objective optimizations is upper bounded by

$$\sum_{k=1}^{N_{\text{grad}}} k \leq \mathcal{O}(N_{\text{grad}}^2) = \mathcal{O}\left(\left(\frac{M_\nu D_X^{1+\nu}}{\varepsilon} \right)^{\frac{4}{1+3\nu}} \right).$$

The proof is now complete. □

Note that by the choice of η_k and Proposition 3.2, Theorem 3.3 provides a complexity result for a version of CGS with the sliding feature for solving problem (CO). Comparing Theorems 3.2 and 3.3, the key difference in the proofs is the additional $1/k$ factor in η_k in Theorem 3.3. With

the additional factor, the three terms at the right hand side of (3.1.10) are of the same order with respect to N, resolving the imperfection we noticed previously in (3.1.9) in the proof of Theorem 3.2. In doing so, we achieve the optimal lower complexity bound of gradient evaluations (2.1.3) for first-order methods. Interestingly, we can discover that the number of linear optimizations required in Theorem 3.3 is also significantly reduced comparing with that in Theorem 3.2, since $4/(1+3\nu) < 1/\nu$ for all $\nu \in (0, 1)$.

It should be noted that we exclude the case $\nu = 1$ case in Theorem 3.3 only for convenience of our analysis, since our focus in this section is mainly the theoretical analysis on improving the state-of-the-art complexity bounds [21, 22] when $\nu \in (0, 1)$. By slightly modifying the proof of Theorem 3.3 we can also achieve the same complexity results as the state-of-the-art in [10]. We include the $\nu = 1$ case in the convergence analysis of practical implementation in the next section.

We conclude this section with several comments regarding the implementation of CGS in Algorithm 2.4. Note that the sliding result shown in Theorem 3.3 requires a parameter choice β_k that assumes the knowledge of Hölder exponent $\nu \in (0, 1)$ and constant M_ν. Unfortunately, the knowledge of the best ν and M_ν for the performance of CGS may not be easily accessible in practice. Furthermore, the proposed Algorithm 2.4 has no termination criterion for verifying whether the current approximate solution y_k is an ε-solution. Lastly, there may exist problem instances in which a solution v^t to the linear subproblem (2.2.4) cannot be computed exactly and instead we can only compute an approximate solution. In the next section, we propose an algorithm called universal conditional gradient sliding (UCGS) that utilizes a backtracking linesearch scheme with an implementable stopping criterion to achieve better practical performance than Algorithm 2.4. We also analyze its convergence under an approximate solution to linear subproblem (2.2.4).

3.2 The Universal Conditional Gradient Sliding Method

In this section, we propose a practical universal conditional gradient sliding (UCGS) method that addresses the above issues of CGS. The proposed UCGS algorithm is described in Algorithm 3.1.

Algorithm 3.1 Universal conditional gradient sliding (UCGS) method

Start: Choose tolerance $\varepsilon > 0$ and initial iteration $x_0 \in X$. Set $y_0 = x_0$.
for $k = 1, 2, \ldots,$ **do**
Decide $L_k > 0$ such that

$$f(y_k) \le f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} ||y_k - z_k||^2 + \frac{\varepsilon}{2}\gamma_k \tag{3.2.1}$$

where

$$\gamma_k := \begin{cases} 1, & k = 1 \\ \text{positive solution to } \Gamma_{k-1}(1-\gamma_k) = \frac{L_k \gamma_k^2}{k}, & k > 1 \end{cases} \tag{3.2.2}$$

$$z_k := (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1} \tag{3.2.3}$$

$$x_k := \text{ACndG}(\nabla f(z_k), x_{k-1}, \beta_k, \eta_k) \tag{3.2.4}$$

$$y_k := (1-\gamma_k) y_{k-1} + \gamma_k x_k \tag{3.2.5}$$

$$\Gamma_k := \frac{L_k \gamma_k^2}{k}. \tag{3.2.6}$$

Compute an approximate solution s_k to the problem

$$\min_{x \in X} \ell_k(x) := \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left(f(z_i) + \langle \nabla f(z_i), x - z_i \rangle \right) \tag{3.2.7}$$

such that $\ell_k(s_k) - \min_{x \in X} \ell_k(x) \le \varepsilon_k$. Terminate and output y_k as an approximate solution if

$$f(y_k) - \ell_k(s_k) + \varepsilon_k \le \varepsilon. \tag{3.2.8}$$

end for

procedure $u^+ = \text{ACndG}(g, u, \beta, \eta)$
Goal: Compute u^+ such that $\max_{x \in X} \langle \nabla \phi(u^+), u^+ - x \rangle \le \eta$, where

$$\phi(x) := \langle g, x \rangle + \frac{\beta}{2} \|x - u\|^2. \tag{3.2.9}$$

Start: Set $u^0 = u$.
for $t = 1, 2, \ldots,$ **do**
Compute a δ^t-approximate solution v^t to the problem $\min_{x \in X} \langle \nabla \phi(u^{t-1}), x \rangle$ such that

$$\langle g + \beta(u^{t-1} - u), v^t \rangle - \min_{x \in X} \langle g + \beta(u^{t-1} - u), x \rangle \le \delta^t. \tag{3.2.10}$$

Terminate with $u^+ := u^{t-1}$ if

$$\langle g + \beta(u^{t-1} - u), u^{t-1} - v^t \rangle + \delta^t \le \eta. \tag{3.2.11}$$

Otherwise, compute $u^t = (1 - \alpha^t) u^{t-1} + \alpha^t v^t$.
end for
end procedure

Let us make a few remarks regarding Algorithm 3.1. First, the approximate conditional gradient method (ACndG) procedure in Algorithm 3.1 is a generalization of the CndG procedure discussed in the previous section. Specifically, whenever $\delta^t \equiv 0$, ACndG and CndG are equivalent. Note also that an appropriate choice for parameter α^t can be computed through an exact linesearch, namely,

$$\alpha^t := \min\left\{1, \frac{\langle g - \beta(u - u^{t-1}), u^{t-1} - v^t\rangle}{\beta\,||v^t - u^{t-1}||^2}\right\}. \tag{3.2.12}$$

It is easy to observe that the above α^t is the optimal solution to the exact linesearch problem $\min_{\alpha\in[0,1]} \phi((1-\alpha)u^{t-1} + \alpha v^t)$. Second, if the objective function f in problem (CO) has Lipschitz continuous gradient (so $\nu = 1$) with Lipschitz constant M_1, then UCGS can be understood as extension of CGS with added features for practical implementation. The new features include a backtracking linesearch strategy that computes adaptive estimates L_k for the Lipschitz constant M_1, the possibility of computing only approximate solutions to linear subproblems, and a termination criterion for verifying whether an approximate solution to problem (CO) has been computed. Third, the choice of γ_k and Γ_k in (3.2.6) and (3.2.2) implies that (3.1.4) holds.

Furthermore, it can be shown that for $k > 1$, the solution to (3.2.2) is given by

$$\gamma_k = \frac{2\sqrt{k\Gamma_{k-1}}}{\sqrt{4L_k + k\Gamma_{k-1}} + \sqrt{k\Gamma_{k-1}}}.$$

Observe that $\gamma_k \in (0, 1)$. Consequently, the recursively described approximate solution y_k is the convex combination of $x_1, \ldots, x_k$. Also the point z_k for gradient evaluation is a convex combination of $x_1, \ldots, x_{k-1}$. Such recursive description first appeared in Nesterov's seminal accelerated gradient algorithm (see, e.g., [2]) and is also used in the CGS algorithm [10] and the universal gradient algorithms studied in [7]. However, our choice of γ_k is novel and is different from the ones in [7, 10, 2]. In fact, to our knowledge, none of the settings of γ_k in [7, 2, 10] are suitable for CGS-type algorithms with adaptive L_k. In the only previous work [25] that successfully developed a linesearch scheme for CGS, γ_k needs to satisfy a more sophisticated cubic equation and L_k needs to be monotone increasing. As we describe below, such monotonicity restriction on L_k is removed in our proposed UCGS method.

A few remarks on the practical implementation of Algorithm 3.1 are also in place. First,

Algorithm 3.1 proposes that we find $L_k > 0$ such that (3.2.1) is satisfied. The condition (3.2.1) originated from the framework of inexact oracle in [26] and is also used in [7]. We proposed to search for such L_k through the same backtracking linesearch strategy in the discussion surrounding Algorithm 2.2. Through this backtracking linesearch strategy, we ensure that our choice of L_k is adaptive and that performance is independent of the choice of L_0. Previous literature [25] on backtracking linesearch strategy of CGS require monotonicity of L_k and may suffer from a poorly chosen L_0. Second, our termination criterion is based on (3.2.8). We can observe immediately that if the parameter $\varepsilon_k \equiv 0$, i.e., s_k is the exact solution to problem (3.2.7), then when (3.2.8) is satisfied, y_k will be ε-approximation solution to problem (CO). To see this, note from (3.1.4) that

$$\Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} = 1 \tag{3.2.13}$$

and consequently

$$f(y_k) - f^* \leq f(y_k) - \min_{x \in X} \ell_k(x) = f(y_k) - \ell_k(s_k).$$

Such termination criterion also appeared in the previous literature (see, e.g., [7, 25]). For the case when $\varepsilon_k > 0$, we show later in Theorem 3.4 that allowing approximate solution s_k with properly chosen accuracy ε_k will not affect the complexity results of UCGS.

We present convergence analysis for the UCGS algorithm proposed above, beginning with some results on the inner iteration complexity. The following lemma resembles a combination of the proofs of Theorem 2.2(c) in [10] and Theorem 5.2 in [24] on the analysis of conditional gradient method with approximate linear objective optimization subproblems for solving projection problems.

Lemma 3.2. Suppose that $\lambda^t \in [0, 1]$ is any predetermined sequence satisfying $\lambda^1 = 1$. In the ACndG procedure, if α^t is chosen such that

$$\phi(u^t) \leq \phi((1 - \lambda^t)u^{t-1} + \lambda^t v^t), \ \forall t \geq 1, \tag{3.2.14}$$

then we have

$$\sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle \leq \left[\delta^1 + \sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} \left(\delta^j + \Lambda^{j-1} \sum_{i=1}^{j-1} \frac{\lambda^i}{\Lambda^i} \delta^i \right) \right] + \frac{\beta D_X^2}{2} \left[1 + \sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} \left(\lambda^j + \Lambda^{j-1} \sum_{i=1}^{j-1} \frac{(\lambda^i)^2}{\Lambda^i} \right) \right]$$

for all $t \geq 2$, where

$$\Lambda^t := \begin{cases} 1 & \text{when } t = 1 \\ \Lambda^{t-1}(1 - \lambda^t) & \text{when } t > 1. \end{cases} \tag{3.2.15}$$

Proof. Observing that the function $\phi(x)$ in (3.2.9) is a strongly convex function with Lipschitz continuous (with constant β) gradient, using the assumption (3.2.14), and noting the definition of approximate solution v^t in (3.2.10), we have

$$\begin{aligned} & \phi(u^t) - (1 - \lambda^t)\phi(u^{t-1}) - \lambda^t(\phi(u^{t-1}) + \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle) \\ \leq & \phi((1 - \lambda^t)u^{t-1} + \lambda^t v^t) - \phi(u^{t-1}) - \lambda^t \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle \\ \leq & \lambda^t \langle \nabla \phi(u^{t-1}), v^t - u^{t-1} \rangle + \frac{\beta(\lambda^t)^2}{2} \left|\left| v^t - u^{t-1} \right|\right|^2 - \lambda^t \langle \nabla \phi(u^{t-1}), x - u^{t-1} \rangle \\ = & \lambda^t \langle \nabla \phi(u^{t-1}), v^t - x \rangle + \frac{\beta(\lambda^t)^2}{2} \left|\left| v^t - u^{t-1} \right|\right|^2 \\ \leq & \lambda^t \delta^t + \frac{\beta D_X^2 (\lambda^t)^2}{2}, \ \forall x \in X, t \geq 1. \end{aligned} \tag{3.2.16}$$

Defining $x^* := \operatorname{argmin}_{x \in X} \phi(x)$, from the above relation we have for any $t \geq 2$ that

$$\begin{aligned} & \sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} \max_{x \in X} \langle \nabla \phi(u^{j-1}), u^{j-1} - x \rangle \\ \leq & \sum_{j=2}^{t} \frac{1}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] - \frac{1}{\Lambda^j} [\phi(u^j) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j} \\ = & [\phi(u^1) - \phi(x^*)] - \frac{1}{\Lambda^t} [\phi(u^t) - \phi(x^*)] \\ & + \sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j} \\ \leq & [\phi(u^1) - \phi(x^*)] + \sum_{j=2}^{t} \frac{\lambda^j}{\Lambda^j} [\phi(u^{j-1}) - \phi(x^*)] + \frac{\lambda^j}{\Lambda^j} \delta^j + \frac{\beta D_X^2 (\lambda^j)^2}{2\Lambda^j}. \end{aligned} \tag{3.2.17}$$

Here in the equality we use the following observations from the definition of Λ^t in (3.2.15): $1/\Lambda^1 = 1$ and $1/\Lambda^j = 1/\Lambda^{j-1} + \lambda^j/\Lambda^j$ for all $j \geq 2$,

To finish the proof it suffices to bound $\phi(u^{j-1}) - \phi(x^*)$ for any $j \geq 2$. Observing that $\phi(x)$ in (3.2.9) is strongly convex and quadratic with

$$\frac{\beta}{2}\left|\left|x - u^{t-1}\right|\right|^2 = \phi(x) - (\phi(u^{t-1}) + \langle \nabla\phi(u^{t-1}), x - u^{t-1}\rangle),\ \forall x \in X, t \geq 1,$$

we have from (3.2.16) (with $x = x^*$) that

$$[\phi(u^t) - \phi(x^*)] - (1-\lambda^t)[\phi(u^{t-1}) - \phi(x^*)] \leq \lambda^t\delta^t + \frac{\beta D_X^2(\lambda^t)^2}{2} - \frac{\beta\lambda^t}{2}\left|\left|x^* - u^{t-1}\right|\right|^2.$$

Applying Lemma 3.1 to the above recurrence relation and ignoring negative terms at the right hand side, we have

$$\phi(u^t) - \phi(x^*) \leq \Lambda^t \sum_{i=1}^{t} \frac{\lambda^i}{\Lambda^i}\delta^i + \frac{\beta D_X^2(\lambda^i)^2}{2\Lambda^i},\ \forall t \geq 1.$$

We conclude the lemma immediately by applying the above bound to (3.2.17) and rearranging terms. □

The complexity result of the above lemma depends on a predetermined sequence $\{\lambda^t\}$. In the proposition below, we provide a complexity result from an example choice of $\{\lambda^t\}$.

Proposition 3.3. In the ACndG procedure, at termination we have

$$\max_{x\in X} \langle \nabla\phi(u^+), u^+ - x\rangle \leq \eta. \tag{3.2.18}$$

Moreover, if $\delta^t = \sigma\beta D_X^2/t$ for certain $\sigma \geq 0$ and α^t is chosen such that

$$\phi(u^t) \leq \phi\left(\frac{t-1}{t+1}u^{t-1} + \frac{2}{t+1}v^t\right), \tag{3.2.19}$$

then we have for any $t \geq 1$ that

$$\begin{aligned}\min_{j=1,\ldots,t+1} \langle \nabla\phi(u^{j-1}), u^{j-1} - v^j \rangle \leq & \min_{j=1,\ldots,t+1} \max_{x\in X} \langle \nabla\phi(u^{j-1}), u^{j-1} - x \rangle \\ \leq & \frac{6(\sigma+1)\beta D_X^2}{t}.\end{aligned} \tag{3.2.20}$$

Specifically, it takes at most

$$T := 1 + \left\lceil \frac{(7\sigma+6)\beta D_X^2}{\eta} \right\rceil \tag{3.2.21}$$

iterations for the ACndG procedure to terminate.

Proof. From the definition of the approximate solution v^t in (3.2.10), if the termination criterion in (3.2.11) of the ACndG procedure is satisfied, then the output $u^+ = u^{t-1}$ satisfies

$$\begin{aligned}\max_{x\in X} \langle \nabla\phi(u^{t-1}), u^{t-1} - x \rangle &= \max_{x\in X} \langle \nabla\phi(u^{t-1}), u^{t-1} - v^t \rangle + \langle \nabla\phi(u^{t-1}), v^t - x \rangle \\ &\leq (\eta - \delta^t) + \delta^t = \eta.\end{aligned}$$

Therefore (3.2.18) holds. To conclude the proposition it suffices to estimate the rate of convergence of

$$\max_{x\in X} \langle \nabla\phi(u^{t-1}), u^{t-1} - x \rangle.$$

To analyze the rate, let us choose $\lambda^t = 2/(t+1)$ and apply Lemma 3.2. Then $\Lambda^t = 2/(t(t+1))$ and

$$\begin{aligned}&\sum_{j=2}^{t} j \cdot \max_{x\in X} \langle \phi(u^{j-1}), u^{j-1} - x \rangle \\ \leq & \sigma\beta D_X^2 \left[1 + \sum_{j=2}^{t} \left(1 + \frac{2}{j-1} \sum_{i=1}^{j-1} 1 \right) \right] + \frac{\beta D_X^2}{2} \left[1 + \sum_{j=2}^{t} \left(\frac{2j}{j+1} + \frac{2}{j-1} \sum_{i=1}^{j-1} \frac{2i}{i+1} \right) \right] \\ < & \sigma\beta D_X^2 (3t-2) + \frac{\beta D_X^2}{2}(6t-5), \ \forall t \geq 2.\end{aligned}$$

Noting that $\sum_{j=2}^{t} j = (t+2)(t-1)/2$, we have

$$\begin{aligned}\min_{j=2,\ldots,t} \max_{x\in X} \langle \nabla\phi(u^{j-1}), u^{j-1} - x\rangle \leq & \frac{2}{(t+2)(t-1)} \sum_{j=2}^{t} j \cdot \max_{x\in X} \langle \phi(u^{j-1}), u^{j-1} - x\rangle \\ < & \frac{6(\sigma+1)\beta D_X^2}{t-1},\ \forall t \geq 2.\end{aligned}$$

Using the above result and observing that

$$\begin{aligned}\min_{j=1,\ldots,t+1} \langle \nabla\phi(u^{j-1}), u^{j-1} - v^j\rangle \leq & \min_{j=1,\ldots,t+1} \max_{x\in X} \langle \nabla\phi(u^{j-1}), u^{j-1} - x\rangle \\ \leq & \min_{j=2,\ldots,t+1} \max_{x\in X} \langle \nabla\phi(u^{j-1}), u^{j-1} - x\rangle,\ \forall t \geq 1\end{aligned}$$

we conclude (3.2.20). Moreover, from (3.2.20) and noting the choice of δ^t, the termination criterion (3.2.11) holds whenever

$$\frac{6(\sigma+1)\beta D_X^2}{t-1} + \frac{\sigma\beta D_X^2}{t} \leq \eta.$$

Noting the definition of T in (3.2.21), the above condition clearly holds for all $t \geq T$. □

In the above proposition, $\sigma \geq 0$ in the definition of δ^t is a parameter related to the accuracy of approximately solving linear objective optimization subproblems. Note that there may also exist other possible choice of δ^t. For example, similar complexity result can be derived by choosing $\delta^t = \sigma\eta$. The benefit of our proposed choice $\delta^t = \sigma\beta D_X^2/t$ from the perspective of practical implementation is that it allows adaptive error of the approximate solution v^t to the linear subproblems and larger error can be admissible when t is small.

As a side note, recalling that ACndG procedure reduces to CndG procedure in Algorithm 2.5, we can observe that Proposition 3.2 in the previous section is a direct consequence of the above result:

Proof of Proposition 3.2. Noting that the CndG procedure described in Algorithm 2.5 is equivalent to the ACndG procedure with $\delta^t \equiv 0$, applying Proposition 3.3 above with $\alpha^t = 2/(t+1)$, we conclude the proposition immediately from (3.2.20). □

From the above two proofs, it is clear that Proposition 3.3 is different from Proposition 3.2 in the previous section, since it shows us that we can compute an approximate solution, rather

than an exact one, to (3.2.10) and proceed with the convergence analysis. We will eventually utilize Proposition 3.3 to establish an upper bound on the number of inner iterations that Algorithm 3.1 requires to compute an ε-solution. We now continue on to the outer iteration analysis, starting with a few results that establish the relation between our computed L_k in the linesearch scheme and the underlying Hölder exponent ν and constant M_ν in (2.1.1). We use the following lemma that appeared in [7].

Lemma 3.3. For any $\delta > 0$ and any L such that

$$L \geq \left(\frac{1-\nu}{1+\nu} \cdot \frac{1}{\tau}\right)^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}},$$

where ν and M_ν are the Hölder continuity exponent and constant in (2.1.1), we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} ||y - x||^2 + \frac{\tau}{2}, \ \forall x, y \in X. \tag{3.2.22}$$

Proof. See Lemma 1 of [7]. □

Note that for $\nu = 1$, the term $\left(\frac{1-\nu}{1+\nu}\right)^{\frac{1-\nu}{1+\nu}}$ can be handled using a continuity argument $\lim_{\nu \to 1} \left(\frac{1-\nu}{1+\nu}\right)^{\frac{1-\nu}{1+\nu}} = 1$. We state an immediate corollary of the above lemma below.

Corollary 3.1. Any $L_k > 0$ chosen by Algorithm 3.1 according to (3.2.1) must also satisfy

$$L_k \leq 2\left(\frac{1-\nu}{1+\nu} \cdot \frac{1}{\varepsilon\gamma_k}\right)^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}}$$

Proof. Suppose that L_k does not satisfy (3.2.22). Then applying Proposition 3.4 with $\tau = \varepsilon\gamma_k$ implies that $L_k/2$ satisfies (3.2.1), contradicting the fact that L_k was chosen at step k following the proposed backtracking linesearch implementation (see the remark on practical implementation after the description of Algorithm 3.1). □

The above result is an immediate consequence of the backtracking linesearch strategy we use to find a suitable L_k that satisfies (3.2.1). Based on the above result, we can estimate a bound of $L_k\gamma_k^2$ in the proposition below. Recalling that $\Gamma_k = L_k\gamma_k^2/k$ from (3.2.6) in Algorithm 3.1, the following lemma provides also a bound of Γ_k that is important for the outer iteration complexity analysis.

Lemma 3.4. Let $L_k > 0$ be chosen by (3.2.1) of Algorithm 3.1 at step k, then

$$L_k \gamma_k^2 \leq \frac{C_\nu M_\nu^{\frac{2}{1+\nu}}}{k^{\frac{1+3\nu}{1+\nu}} \varepsilon^{\frac{1-\nu}{1+\nu}}}$$

where

$$C_\nu := \left(\frac{1+2\nu}{1+3\nu}\right)^{\frac{1+3\nu}{1+\nu}} \left(\frac{1-\nu}{1+\nu}\right)^{\frac{1-\nu}{1+\nu}} 2^{\frac{4+10\nu}{1+\nu}} \tag{3.2.23}$$

is a constant depending only on ν.

Proof. The case when $k = 1$ is immediate from Corollary 3.1. Therefore, throughout the proof we assume that $k \geq 2$. Since we set $\Gamma_k = L_k \gamma_k^2 / k$ in Algorithm 3.1, we can prove this proposition by bounding Γ_k. Set $s := (1+\nu)/(1+3\nu)$. Since $\nu \in [0,1]$, we have $s \in [1/2, 1]$. We study the quantity $1/\Gamma_k^s - 1/\Gamma_{k-1}^s$, which can be rewritten as

$$\begin{aligned}
\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} &= \frac{\left(\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s}\right)\left(\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}}\right)}{\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}}} \\
&= \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}} - \frac{1}{\Gamma_k}\left(\frac{\Gamma_k}{\Gamma_{k-1}}\right)^s + \frac{1}{\Gamma_{k-1}}\left(\frac{\Gamma_{k-1}}{\Gamma_k}\right)^s}{\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}}}.
\end{aligned}$$

Here, noting from the relation of Γ_k and Γ_{k-1} in (3.1.4) that $\Gamma_k \leq \Gamma_{k-1}$ and recalling that $s \in [1/2, 1]$, we can make two observations. First, we have $\Gamma_k^{2s-1} \leq \Gamma_{k-1}^{2s-1}$, and hence

$$-\frac{1}{\Gamma_k}\left(\frac{\Gamma_k}{\Gamma_{k-1}}\right)^s + \frac{1}{\Gamma_{k-1}}\left(\frac{\Gamma_{k-1}}{\Gamma_k}\right)^s \geq 0.$$

Second, we have $\Gamma_k^{1-s} \leq \Gamma_{k-1}^{1-s}$, and hence

$$\frac{1}{\Gamma_k^{1-s}} + \frac{1}{\Gamma_{k-1}^{1-s}} \leq \frac{2}{\Gamma_k^{1-s}}.$$

Combining the above two observations and recalling that $s = (1+\nu)/(1+3\nu)$ and the relations concerning Γ_k and Γ_{k-1} in (3.1.4), we have that

$$\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} \geq \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}}{\frac{2}{\Gamma_k^{1-s}}} = \frac{\frac{\gamma_k}{\Gamma_k}}{\frac{2}{\Gamma_k^{1-s}}} = \frac{\gamma_k}{2} \Gamma_k^{-\frac{1+\nu}{1+3\nu}}. \tag{3.2.24}$$

We can further bound the last expression in the above relation. Indeed, recalling that $\Gamma_k = L_k\gamma_k^2/k$ and applying Corollary 3.1, we have the inequality

$$\frac{\gamma_k^2}{k\Gamma_k} = \frac{1}{L_k} \geq \frac{1}{2M_\nu^{\frac{2}{1+\nu}}}\left(\frac{1+\nu}{1-\nu}\cdot\varepsilon\gamma_k\right)^{\frac{1-\nu}{1+\nu}}.$$

In the above recall that we can use a continuity argument for the $\nu = 1$ case since $(1-\nu)^{-(1-\nu)} \to 1$ as $\nu \to 1$. Rearranging terms in the above relation, we have

$$\gamma_k\Gamma_k^{-\frac{1+\nu}{1+3\nu}} \geq \left(\frac{1+\nu}{1-\nu}\right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}}k^{\frac{1+\nu}{1+3\nu}}}{2^{\frac{1+\nu}{1+3\nu}}M_\nu^{\frac{2}{1+3\nu}}}.$$

Applying the above bound to (3.2.24), it follows that

$$\frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_{k-1}^s} \geq \left(\frac{1+\nu}{1-\nu}\right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}}k^{\frac{1+\nu}{1+3\nu}}}{2^{\frac{2+4\nu}{1+3\nu}}M_\nu^{\frac{2}{1+3\nu}}}.$$

Summing the above from $i = 2$ to k and using the fact that

$$\sum_{i=2}^{k} i^{\frac{1+\nu}{1+3\nu}} \geq \int_1^k u^{\frac{1+\nu}{1+3\nu}}du = \frac{1+3\nu}{2+4\nu}\cdot\left(k^{\frac{2+4\nu}{1+3\nu}} - 1\right) \geq \frac{1+3\nu}{4+8\nu}k^{\frac{2+4\nu}{1+3\nu}},\ \forall k \geq 2$$

and the definition of C_ν in (3.2.23), we obtain

$$\frac{1}{\Gamma_k^s} \geq \frac{1}{\Gamma_k^s} - \frac{1}{\Gamma_1^s} \geq \left(\frac{1+\nu}{1-\nu}\right)^{\frac{1-\nu}{1+3\nu}} \frac{\varepsilon^{\frac{1-\nu}{1+3\nu}}}{2^{\frac{4+10\nu}{1+3\nu}}M_\nu^{\frac{2}{1+3\nu}}}\frac{1+3\nu}{1+2\nu}k^{\frac{2+4\nu}{1+3\nu}}$$

Recalling that $s = (1+\nu)/(1+3\nu)$ and $\Gamma_k = L_k\gamma_k^2/k$, we conclude the proposition immediately from the above result. □

It should be noted that the technique utilized in Lemma 3.4 is similar to that of the proof surrounding equation (4.4) in [7]. However, note that the choice of parameter γ_k in UCGS is different from the one in [7]. Therefore, the proof in [7] needs to be adapted to the above proof. With the help of Lemma 3.4, we are now ready to prove our primary convergence properties on the proposed UCGS algorithm. We start with the following proposition that resembles the outer iteration analysis in Proposition 3.1 of the previous section.

Proposition 3.4. Suppose that the parameters in Algorithm 3.1 satisfy $\beta_k \geq L_k\gamma_k$ for all k. Then

for any $x \in X$,

$$f(y_k) - \ell_k(x) \leq \frac{\varepsilon}{2} + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \beta_i}{2\Gamma_i} \left(||x - x_{i-1}||^2 - ||x - x_i||^2 \right)^2 + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{\Gamma_i}.$$

Proof. Fix any $x \in X$. From the definitions of $\ell_k(x)$ and y_k in (3.2.1) and (3.2.5) respectively, we have

$$\begin{aligned}
\frac{1}{\Gamma_k}\ell_k(x) =& \sum_{i=1}^{k} \frac{1}{\Gamma_i} \left(\gamma_i f(z_i) + \gamma_i \langle \nabla f(z_i), x - x_i \rangle + \langle \nabla f(z_i), \gamma_i (x_i - z_i) \rangle \right) \\
=& \sum_{i=1}^{k} \frac{1}{\Gamma_i} \left(f(z_i) + \langle \nabla f(z_i), y_i - z_i \rangle \right) + \frac{\gamma_i}{\Gamma_i} \langle \nabla f(z_i), x - x_i \rangle \\
& - \frac{1 - \gamma_i}{\Gamma_i} (f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i \rangle)
\end{aligned}$$

We now bound three terms in the above relation. First, by convexity of f,

$$-(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i \rangle) \geq -f(y_{i-1}).$$

Second, by our choice of L_k in (3.2.1) and the definitions of y_k and z_k in (3.2.5) and (3.2.3) respectively, we have

$$\begin{aligned}
f(z_i) + \langle \nabla f(z_i), y_i - z_i \rangle \geq& f(y_i) - \frac{L_i}{2} ||y_i - z_i||^2 - \frac{\varepsilon}{2}\gamma_i \\
=& f(y_i) - \frac{L_i \gamma_i^2}{2} ||x_i - x_{i-1}||^2 - \frac{\varepsilon}{2}\gamma_i.
\end{aligned}$$

Lastly, using the result (3.2.18) in Lemma 3.2 and noting the definition of $\phi(x)$ in (3.2.9), we obtain the following result during the termination of the ACndG procedure in computing x_i:

$$\begin{aligned}
\langle \nabla f(z_i), x - x_i \rangle \geq& \beta_i \langle x_i - x_{i-1}, x_i - x \rangle - \eta_i \\
=& -\frac{\beta_i}{2} \left(||x - x_{i-1}||^2 - ||x_i - x_{i-1}||^2 - ||x - x_i||^2 \right) - \eta_i \\
\geq& -\frac{\beta_i}{2} \left(||x - x_{i-1}||^2 - ||x - x_i||^2 \right) - \eta_i + \frac{L_i^2}{2} ||x_i - x_{i-1}||^2 .
\end{aligned}$$

In the last inequality above we use our assumption that $\beta_k \geq L_k \gamma_k$ for all k. Based on the above

three observations and rearranging terms we obtain that

$$\begin{aligned}\frac{1}{\Gamma_k}\ell_k(x) \geq & \sum_{i=1}^{k} \frac{1}{\Gamma_i}\left(f(y_i) - (1-\gamma_i)f(y_{i-1}) - \frac{\gamma_i\beta_i}{2}(||x-x_{i-1}||^2 - ||x-x_i||^2)\right) \\ & - \frac{1}{\Gamma_i}\left(\frac{\varepsilon}{2}\gamma_i + \gamma_i\eta_i\right) \\ = & \frac{f(y_k)}{\Gamma_k} - \sum_{i=1}^{k}\frac{\gamma_i\beta_i}{2\Gamma_i}(||x-x_{i-1}||^2 - ||x-x_i||^2) - \sum_{i=1}^{k}\frac{\gamma_i\eta_i}{\Gamma_i} - \frac{\varepsilon}{2\Gamma_k}.\end{aligned}$$

Here in the last equality we use the relations (3.1.4) and (3.2.13) and the fact that $\gamma_1 = 1$ in its definition (3.2.2). We conclude the result by multiplying by Γ_k and rearranging terms. □

With the help of Propositions 3.3, Lemma 3.4, and Proposition 3.4, we are ready to present the complexity results of UCGS in the following theorem.

Theorem 3.4. Suppose that we apply UCGS described in Algorithm 3.1 with parameters

$$\beta_k = L_k\gamma_k,\ \eta_k = \frac{L_k\gamma_k D_X^2}{k},\ \text{and}\ \varepsilon_k = \frac{\sigma L_k\gamma_k^2 D_X^2}{2}, \tag{3.2.25}$$

and α^t in (3.2.12) and $\delta^t = \sigma\beta D_X^2/t$ in the ACndG procedure, where $\sigma \geq 0$ is a parameter related to the accuracy of approximately solving linear objective optimization subproblems. Then Algorithm 3.1 terminates with an ε-solution after at most N_{iter} outer iterations where

$$N_{\text{iter}} := \left\lceil 16\left(\frac{(3+\sigma)^{\frac{1+\nu}{2}}M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right\rceil$$

As a consequence, the total number of gradient evaluations and linear objective optimizations performed by UCGS to find an ε solution of (CO) can be bounded by $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{2}{1+3\nu}})$ and $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{4}{1+3\nu}})$ respectively.

Proof. From the definition of s_k in Algorithm 3.1, we have that if the termination criterion of UCGS in (3.2.8) holds, then y_k is an ε-solution to problem (CO). Let us evaluate the number of gradient evaluations, or equivalently, the number of outer iterations of UCGS in order to compute an ε-solution y_k. Applying Proposition 3.4 with our choice of parameters we have

$$f(y_k) - \ell_k(x) = \frac{\varepsilon}{2} + \frac{\Gamma_k}{2}\sum_{i=1}^{k} i(||x-x_{i-1}||^2 - ||x-x_i||^2) + \Gamma_k k D_X^2,\ \forall x \in X. \tag{3.2.26}$$

The second term above can be further simplified by noting from the compactness of X and the definition of the diameter D_X in (1.0.1). Indeed, we have

$$\begin{aligned}
&\sum_{i=1}^{k} i\left(||x-x_{i-1}||^2 - ||x-x_i||^2\right) \\
&= ||x-x_0||^2 + \sum_{i=2}^{k}(i-(i-1))\,||x-x_{i-1}||^2 - k\,||x-x_k||^2 \qquad (3.2.27)\\
&\leq D_X^2 + \sum_{i=2}^{k} D_X^2 = \; kD_X^2, \;\forall x \in X.
\end{aligned}$$

Thus, we may continue by applying (3.2.27) to (3.2.26) with $x = s_k$ to conclude that

$$f(y_k) - \ell_k(s_k) \leq \frac{\varepsilon}{2} + \frac{\Gamma_k}{2} k D_X^2 + \Gamma_k k D_X^2 = \frac{\varepsilon}{2} + \frac{3L_k\gamma_k^2}{2} D_X^2.$$

Here we make use of the description of Γ_k in (3.2.6) for the last equality. In view of the above result and the value of parameter ε_k in (3.2.25), y_k satisfies the termination criterion (3.2.8) of UCGS and hence becomes an ε-solution whenever k satisfies the relation $(3+\sigma)L_k\gamma_k^2 D_X^2 \leq \varepsilon$. Applying Lemma 3.4, it follows that such relation holds whenever

$$\frac{(3+\sigma)C_\nu D_X^2 M_\nu^{\frac{2}{1+\nu}}}{k^{\frac{1+3\nu}{1+\nu}}\varepsilon^{\frac{1-\nu}{1+\nu}}} \leq \varepsilon, \text{ i.e., } k \geq C_\nu^{\frac{1+\nu}{1+3\nu}} \left(\frac{(3+\sigma)^{\frac{1+\nu}{2}} M_\nu D_X^{1+\nu}}{\varepsilon}\right)^{\frac{2}{1+3\nu}}.$$

Noting that C_ν defined in (3.2.23) is a constant that depends only on $\nu \in [0,1]$ and observing that $C_\nu^{\frac{1+\nu}{1+3\nu}} \leq 16$ for all $\nu \in [0,1]$, we conclude that whenever $k \geq N_{\text{iter}}$, y_k is an ε-solution.

To compute the total number of gradient evaluations required, note that each outer iteration of UCGS requires us to search for a valid L_k before proceeding. From the discussion following Algorithm 3.1 regarding how to find a valid L_k, let i_k denote the number of times that we set $L_k = 2L_k$ on iteration k before a valid L_k is found. Since the validation of each L_k requires a gradient evaluation, the total number of gradient evaluations of UCGS is

$$N_{\text{grad}} := \sum_{k=1}^{N_{\text{iter}}} 1 + i_k. \qquad (3.2.28)$$

Noting from our linesearch strategy that $L_k = \frac{1}{2}2^{i_k}L_{k-1}$, we have that $i_k - 1 = \log \frac{L_k}{L_{k-1}}$. Thus,

$$N_{\text{grad}} = \sum_{k=1}^{N_{\text{iter}}} 1 + i_k = \sum_{k=1}^{N_{\text{iter}}} 2 + \log L_k - \log L_{k-1} \leq 2N_{\text{iter}} + \log L_{N_{\text{iter}}}.$$

It suffices to show that the logarithmic term above is indeed negligible. Using Corollary 3.1, we have that

$$\begin{aligned} \log L_k &\leq \log \left[\left(\frac{1-\nu}{1+\nu} \cdot \frac{1}{\varepsilon\gamma_k} \right)^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}} \right] \\ &\leq \frac{1-\nu}{1+\nu} \left(\log \left(\frac{1}{\varepsilon} \right) + \log \left(\frac{1}{\gamma_k} \right) \right) + \frac{2}{1+\nu} \log M_\nu \end{aligned} \tag{3.2.29}$$

since $(1-\nu)/(1+\nu) \leq 1$ for $\nu \in [0,1]$. To handle the $1/\gamma_k$ term, noting that since the termination criterion in (3.2.8) is met whenever $(3+\sigma)L_k\gamma_k^2 D_X^2 \leq \varepsilon$, we have that

$$L_k\gamma_k^2 \leq \frac{\varepsilon}{(3+\sigma)D_X^2}$$

upon termination. Thus, by running Algorithm 3.1 with the stopping criterion in (3.2.8), we can be sure that

$$L_k\gamma_k^2 \geq \frac{\varepsilon}{(3+\sigma)D_X^2}.$$

We can use the above inequality to then derive

$$\log \left(\frac{1}{\gamma_k} \right) \leq \frac{1}{2} \log \left(\frac{(3+\sigma)D_X^2}{\varepsilon} \right) + \frac{1}{2} \log L_k \tag{3.2.30}$$

Applying (3.2.30) to (3.2.29), we obtain

$$\log L_k \leq \frac{2(1-\nu)}{1+3\nu} \log \left(\frac{1}{\varepsilon} \right) + \frac{1-\nu}{1+\nu} \log \left(\frac{(3+\sigma)D_X^2}{\varepsilon} \right) + \frac{4}{1+3\nu} \log M_\nu$$

which is independent of k. Thus, we have that, up to a negligible logarithmic term, UCGS requires $\mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations of f to compute an ε-solution.

It suffices to compute the number of linear objective optimizations that UCGS requires for computing an ε-solution. Let us estimate the maximal number of inner iterations required before the termination criterion (3.2.11) is satisfied. Recall from the remark after (3.2.12) that α^t is the

best linesearch parameter and hence satisfies assumption (3.2.19) in Proposition 3.3. Applying Proposition 3.3 and noting the definition of approximate solution v^t in (3.2.10), we have that the maximal number of linear objective optimizations performed at the k-th call to the ACndG procedure is at most

$$T_k := 1 + \left\lceil \frac{(7\sigma + 6)\beta_k D_X^2}{\eta_k} \right\rceil = 1 + \lceil (7\sigma + 6)k \rceil.$$

Noting that there is a total of $1 + (1 + i_k)T_k$ linear optimizations in the k-th outer iteration, we can compute the total number of linear optimizations to be

$$N_{\text{lin}} := \sum_{k=1}^{N_{\text{iter}}} 1 + (1 + i_k)T_k \leq N_{\text{iter}} + T_{N_{\text{iter}}} \sum_{k-1}^{N_{\text{iter}}} (1 + i_k)$$

since T_k is increasing in k. Applying (3.2.28) to the above, we have

$$N_{\text{lin}} \leq N_{\text{iter}} + T_{N_{\text{iter}}} N_{\text{grad}}$$

Since $T_N = \mathcal{O}(N)$, we conclude that $N_{\text{lin}} = \mathcal{O}((M_\nu D_X^{1+\nu}/\varepsilon)^{\frac{4}{1+3\nu}})$. □

We conclude this section with a few remarks on the above complexity results of UCGS. First, we note that UCGS is similar to the Fast Gradient Method presented in [7] in the sense that the number of gradient evaluations generalizes the accelearated gradient descent method in [2]. From Theorem 3.4, number of gradient evaluations required by UCGS to compute an approximate solution is $\mathcal{O}(\left(M_\nu D_X^{1+\nu}/\varepsilon\right)^{\frac{2}{1+3\nu}})$. In the smooth case when $\nu = 1$, this becomes $\mathcal{O}\left(\sqrt{M_1 D_X^2/\varepsilon}\right)$ which matches the complexities of gradient evaluations in [7, 2]. Second, unlike FGM that requires exact solutions to projection subproblems, we have a bound on the number of linear objective optimizations required to solve the projection subproblem (3.2.4). From this perspective, UCGS is a generalization of CGS in [10] as a universal method that covers not only the smooth case (when $\nu = 1$) but also the weakly smooth case (when $\nu \in (0, 1)$), without requiring any knowledge of Hölder exponent ν and constant M_ν . Indeed, when $\nu = 1$ our complexity on the number of linear objective optimizations is on the order of $\mathcal{O}(M_1 D_X^2/\varepsilon)$, which matches the that of CGS in [10]. Third, the number of linear objective optimizations and gradient evaluations when CG is applied to (CO) was shown to be $\mathcal{O}\left(\left(M_\nu D^{1+\nu}/\varepsilon\right)^{\frac{1}{\nu}}\right)$ in [21, 22]. In view of Theorem 3.4, UCGS benefits from sliding and only requires $\mathcal{O}\left(\left(M_\nu D^{1+\nu}/\varepsilon\right)^{\frac{2}{1+3\nu}}\right)$ gradient evaluations and $\mathcal{O}\left(\left(M_\nu D^{1+\nu}/\varepsilon\right)^{\frac{4}{1+3\nu}}\right)$ linear objective

optimizations which are both improvements over the results in [21, 22] whenever $\nu \in (0, 1]$. Fourth, our proposed UCGS method is not only a universal method generalization of the CGS method. Indeed, there are more features added for practical implementation: it has an implementable exit criterion and allows for an approximate solution to (3.2.10). Note that such added features of the UCGS does not affect its theoretical complexity. Finally, we use the same accuracy constant σ for approximately solving the linear subproblems in setting the parameters ε_k and δ^t. It is easy to change the proof if we use different accuracy constants for ε_k and δ^t.

3.3 Numerical Experiments

Our goal in this section is to present preliminary results from our numerical experiments. We compare the performance of our proposed UCGS algorithm with that of the CG method for Hölder smooth objectives in [21] in two numerical experiments described below. The experiments are performed using MATLAB R2018b.

In the first experiment, we consider the problem

$$\min_{x \in \text{conv}(V)} f(x) := ||Ax - b||_2$$

with $V = \{v_1, \ldots, v_p\} \subseteq \mathbb{R}^n$, $\text{conv}(V) := \{x \in \mathbb{R}^n : \exists \lambda \in \Delta_p \text{ s.t. } x = \sum_{j=1}^p \lambda_i v_i\}$, and $\Delta_p := \{\lambda \in \mathbb{R}^p : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0\}$ is the standard simplex. In this experiment, we generate vectors v_i uniformly in $[0, 1]^n$. The matrix $A \in \mathbb{R}^{m \times n}$ is a Gaussian randomly generated sparse matrix with density d. For this experiment, we fix the number of vectors in the set V to be $p = 500$ and set $m = 2n$. The linear objective optimization subproblem is a linear program over the standard simplex and can be computed easily.

For our second experiment, we solve the problem

$$\min_{X \in \text{Spe}_n} f(X) := \sum_{i=1}^m ||X - A_i||_2$$

where $\text{Spe}_n := \{X \in \mathbb{R}^{n \times n} : \text{tr}(X) = 1, X \succeq 0\}$ is the standard spectrahedron and $A_i \in \text{Spe}_n$ for each $i = 1, \ldots, m$. The matrices A_i are obtained by randomly generating an $n \times n$ matrix whose entries follow uniform $[0, 1]$ distributions and then projecting it into Spe_n. The linear objective optimization problem over Spe_n is equivalent to a smallest eigenvalue problem, which is solved by

MATLAB's eigs() function. Note that a solution to the smallest eigenvalue problem will not be exact, and therefore we benefit from being able to solve the linear subproblems approximately.

In our experiments, UCGS terminates whenever an ε-solution with tolerance $\varepsilon = 10^{-3}$ is computed. We terminate CG when its computational time exceeds twice the amount that UCGS spent before termination. Note that both models in the experiments have nonsmooth objective functions, but are still differentiable at many feasible points. Therefore, they may benefit from a universal method for $\nu \in (0, 1]$.

		UCGS				**CG**		
n	d	**GE**	**LO**	**Time**	**Error**	**Iter**	**Time**	**Error**
2500	0.2	66	2690	6.71	$9.945e-4$	572	13.42	$9.7086e1$
2500	0.4	60	3679	9.08	$9.976e-4$	524	18.17	$1.404e2$
2500	0.6	62	245	2.64	$9.678e-4$	146	5.29	$5.598e2$
2500	0.8	57	3176	8.45	$9.768e-4$	399	16.93	$2.400e2$
5000	0.2	71	286	7.13	$9.882e-4$	178	14.32	$6.037e2$
5000	0.4	42	52	4.89	$9.585e-4$	84	9.81	$1.689e3$
5000	0.6	68	4564	36.14	$9.727e-4$	483	72.40	$3.527e2$
5000	0.8	67	419	12.91	$9.815e-4$	161	25.94	$1.165e3$
10000	0.2	85	12269	150.51	$9.96e-4$	915	301.21	$2.449e2$
10000	0.4	69	12614	157.39	$9.916e-4$	636	315.27	$4.734e2$
10000	0.6	70	16063	205.87	$9.821e-4$	653	412.14	$5.423e2$
10000	0.8	69	12707	180.65	$9.862e-4$	473	361.73	$8.162e2$

Table 3.1: Minimizing over a convex hull. Here, we report the gradient evaluations (outer iterations) and linear objective optimization (inner iterations) for UCGS as well as the error that it terminated with. For CG, we allow it to run for twice the amount of time that UCGS took. We then report the number of iterations and whether terminating objective value was better than that of UCGS.

The results from the numerical experiments are documented in Tables 3.1 and 3.2. Column 1 indicates the sizes n[1] whereas the second column represents either the density of A or the value of m for experiments 1 and 2 respectively. Columns 3 and 4 denote the number of outer iterations, i.e. gradient evaluations (GE), and inner iterations, i.e. linear objective optimization (LO), respectively that UCGS performed before terminating with the desired tolerance. Columns 5 and 6 present the time (in seconds) used and error upon termination of UCGS. For CG, we report the total number of iterations (Iter) performed, the computational time (in seconds) required and the final error in Columns 7, 8, and 9. Note that if the time of CG is twice that of UCGS, then the error is not expected to be below our specified tolerance.

Let us make a few comments regarding the results in Tables 3.1 and 3.2. For the convex hull

[1]Note that the length of the vectors are n and n^2 in the first and second experiments respectively.

		UCGS				CG		
n	m	**GE**	**LO**	**Time**	**Error**	**Iter**	**Time**	**Error**
50	50	1354	8493	9.87	$9.992e-4$	6908	19.74	$6.073e-3$
50	100	1767	11138	13.09	$9.994e-4$	7038	26.19	$1.172e-2$
50	200	2425	15173	25.39	$9.995e-4$	8273	50.79	$2.271e-2$
100	50	1836	13056	159.61	$9.980e-4$	11648	319.25	$3.225e-3$
100	100	2347	16816	216.59	$9.990e-4$	13372	433.20	$5.634e-3$
100	200	3296	23836	310.16	$9.984e-4$	16053	620.36	$9.892e-3$
200	50	1722	33673	470.71	$9.989e-4$	15966	941.43	$3.308e-3$
200	100	2314	46323	730.69	$9.994e-4$	17033	1461.42	$6.870e-3$
200	200	3154	64511	1086.42	$9.992e-4$	19762	2172.85	$1.015e-2$

Table 3.2: Minimizing over the standard spectrahedron. Here, we report the gradient evaluations (outer iterations) and linear objective optimization (inner iterations) for UCGS as well as the error that it terminated with. For CG, we allow it to run for twice the amount of time that UCGS took. We then report the number of iterations and whether terminating objective value was better than that of UCGS.

experiment in Table 3.1, we see that the excessive number of gradient evaluations of CG prevents it from being competitive. The gradient of our objective function requires a matrix multiplication of increasingly dense matrices. As these densities tend to 1, the gradient evaluations become more computationally expensive, and CG cannot report as good of a solution as UCGS with even in twice the allotted time, because it requires much more gradient evaluations to compute an approximate solution. We also note the necessity of a projection-free algorithm for this feasible set since the projection onto the convex hull requires the solving of a quadratic program. For any moderately sized n, this quadratic program is computationally infeasible to solve. For example, one iteration of FGM in [7] applied to the problem instance with $n = 2500$ and $d = 0.2$ takes at least 20 seconds, which is three times as long as UCGS took to converge.

The second experiment over the standard spectrahedron removes the previous difficulty of computing the gradient. In the second experiment, the cost of the gradient evaluation is almost negligible. However, we still see in Table 3.2 that UCGS outperforms CG. In this case, the superior linear objective optimization complexity of UCGS can be seen by noting that CG performs one linear objective optimization per iteration. Thus, even with a comparable amount of linear objective optimizations, CG can still not match the complexity of UCGS. This directly highlights the differences in the linear objective optimization complexity mentions previously. We also observe the effectiveness of the implementable stopping criterion which enabled us to terminate when an ε-solution was achieved.

Chapter 4

Sliding Alternating Direction Method of Multipliers

Recall in Chapter 1 that the accelerated ADMM algorithm (A-ADMM) achieves optimal performance with respect to the oracle in (2.3.5) when applied to the problem in (ACO). However, from our discussion surrounding NAGD and Corollary 2.2, it is known that only $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations are required for a first-order method. In this sense, A-ADMM computes more than the optimal number of gradient evaluations. From the oracle complexity analysis point of view, the complexity results of NAGD reveal a drawback in the assumption of the first-order oracle (2.3.5). Since $f(x)$ and $h(Kx)$ are separate functions in the definition of problem (ACO), it is only reasonable to consider their oracles separately. Specifically, let $O_f : \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}$ such that $O_f(x) = (f(x), \nabla f(x))$ for any inquiry point x, and an oracle $O_K : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m \times \mathbb{R}^n$ such that $O_K(u, v) = (Ku, K^\top v)$ for any inquiry point (u, v). The performance of a first-order algorithm should be evaluated by its number of inquires to O_K and O_f respectively.

Let us further explore this idea by considering the separable problem

$$\min_{x \in X, z \in Z} f(x) + h(Az - b) \tag{4.0.1}$$

as a generalization of (2.3.1) and (ACO). Per our discussion of NAGD, if $h \equiv 0$, then we should expect at least $\Omega(\sqrt{L/\varepsilon})$ gradient evaluations to compute an ε-solution to (4.0.1). On the other

hand if $f \equiv 0$, then, it was shown in [27] that proximal based methods require at least $\Omega(||K||/\varepsilon)$ operator evaluations. Thus, under the combined oracle in (2.3.5), it is natural that problems of the form (ACO) require $\Omega(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ calls to compute an ε solution. An interesting question is whether or not we can design algorithms that can benefit from separating the calls to $\mathcal{O}_f$ and $\mathcal{O}_K$.

This idea of counting different oracle inquiries separately is the driving issue behind sliding algorithms as we saw in the CGS method in Chapters 1 and 3. From our previous discussion, applying A-ADMM to solve problem (ACO) computes an ε-approximate solution with only $\mathcal{O}(\sqrt{L/\varepsilon}+||K||/\varepsilon)$ iterations. However, since a gradient evaluation is required for each iteration, we still need to compute $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ gradient evaluations. Conversely, if we apply NAGD in Algorithm 2.1 to solve problem (ACO), we can already compute an ε-approximate solution with $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations of ∇f, finishing part of our objective in algorithm design. The only drawback is that the optimization subproblem in the x_k step (2.0.4) may be difficult to solve. Naturally, we may choose to solve the subproblem approximately through certain first-order methods.

4.1 Gradient Sliding

Our proposed method GS-ADMM will attempt to combine variants of the above NAGD and A-ADMM algorithms. In particular, we will make use of the linearized ADMM (L-ADMM) in Algorithm 4.1 in order to solve the subproblem (2.0.4) approximately. By controlling the number of iterations used by L-ADMM in each iteration of NAGD, we will keep the operator evaluations to a minimum while still maintaining the $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluation complexity of NAGD. Solving the related projection problem (2.0.4) is a crucial component of sliding algorithms. To continue, we now discuss solving the appropriate (2.0.4) and analyze the performance of L-ADMM applied to the subproblem in (2.0.4) under various parameter settings.

Let us describe the subproblem (2.0.4) in NAGD abstractly as follows:

$$\min_{u \in X} \phi(u) := \langle g, u \rangle + h(Ku) + \frac{\beta}{2}\|u - x\|^2. \tag{4.1.1}$$

Here, ϕ contains an $h(Ku)$ term rather than a linear approximation since h was assumed to be nondifferentiable and simple. To solve the above problem, we can reformulate it to an affinely

constrained problem:

$$\min_{(u,w)\in\mathcal{H}} \Phi(u,w) := \langle g, u\rangle + h(w) + \frac{\beta}{2}\|u-x\|^2, \tag{4.1.2}$$

where recall that $\mathcal{H} := \{\, (u,w) \in X \times Z \mid Ku - w = 0 \,\}$. We may now apply the ADMM in Algorithm 2.6 to solve the above problem. Using (u_t, v_t, w_t) as the iterates, we have the following iterations:

$$\begin{aligned} u_t =& \operatorname*{argmin}_{u\in X} \langle g, u\rangle + \frac{\beta}{2}\|u-x\|^2 + \langle v_{t-1}, Ku\rangle + \frac{\sigma}{2}\|Ku - w_{t-1}\|^2 \\ w_t =& \operatorname*{argmin}_{w\in\mathbb{R}^m} h(w) - \langle v_{t-1}, w\rangle + \frac{\sigma}{2}\|w - Ku_t\|^2 \\ v_t =& v_{t-1} - \sigma(Ku_t - w_t). \end{aligned} \tag{4.1.3}$$

Note again that for large dense matrix K, the optimization problem in (4.1.3) may be difficult to solve. However, as discussed in Chapter 1, we may use L-ADMM variant by replacing the quadratic function $(\sigma/2)\|Ku - w_{t-1}\|^2$ with its linear approximation. The L-ADMM algorithm is described in Algorithm 4.1. Note that the in the description of L-ADMM in Algorithm 4.1 we use variable parameters θ_t, τ_t, and ρ_t instead of the constant parameter σ above for generality. Furthermore, we see that (4.1.4) and (4.1.5) are computed via a projection and proximal solve, respectively.

Algorithm 4.1 Linearized ADMM (L-ADMM) for solving problem (4.1.2)

Start: Choose $u_0 \in X$, $v_0 \in \mathbb{R}^m$, and $w_0 \in Z$.
for $t = 1, \ldots, T$ **do**
Compute

$$\begin{aligned} u_t = \operatorname*{argmin}_{u\in X} \langle g, u\rangle + \frac{\beta}{2}\|u-x\|^2 + \langle K^\top(v_{t-1} + \theta_t(Ku_{t-1} - w_{t-1})), u\rangle \\ + \frac{\eta_t}{2}\|u - u_{t-1}\|^2 \end{aligned} \tag{4.1.4}$$

$$w_t = \operatorname*{argmin}_{w\in Z} h(w) - \langle v_{t-1}, w\rangle + \frac{\tau_t}{2}\|w - Ku_t\|^2 \tag{4.1.5}$$

$$v_t = v_{t-1} + \rho_t(Ku_t - w_t) \tag{4.1.6}$$

end for

The convergence analysis of L-ADMM is well studied; for example, the analysis in [17] can be applied directly. Specifically, we have the following result.

Proposition 4.1. Suppose that the parameters η_t and θ_t in Algorithm 4.1 satisfy

$$\eta_t \geq \theta_t \|K\|^2. \tag{4.1.7}$$

For any $(u, w) \in \mathcal{H}$ and any $v \in \mathbb{R}^m$, we have

$$\Phi(\tilde{u}_T, \tilde{w}_T) - \Phi(u, w) \leq \Delta_T(u, w),$$

where

$$\tilde{u}_T := \frac{1}{T}\sum_{t=1}^{T} u_t \text{ and } \tilde{w}_T := \frac{1}{T}\sum_{t=1}^{T} w_t \tag{4.1.8}$$

are the averages of sequences $\{u_t\}_{t=1}^T$ and $\{w_t\}_{t=1}^T$, and

$$\begin{aligned}\Delta_T(u, w) :=& \frac{1}{T}\sum_{t=1}^{T}\Bigg[\frac{\eta_t}{2}\|u - u_{t-1}\|^2 - \frac{\beta + \eta_t}{2}\|u_t - u\|^2 + \frac{1}{2\rho_t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) \\ &+ \frac{\theta_t}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) - \frac{\theta_t}{2}(\|Ku_{t-1} - Ku\|^2 - \|Ku_t - Ku\|^2) \\ &- \frac{\tau_t - \rho_t}{2\rho_t^2}\|v_{t-1} - v_t\|^2 + \frac{\tau_t - \theta_t}{2}\|Ku_t - Ku\|^2 - \frac{\tau_t - \theta_t}{2}\|w_t - w\|^2 \Bigg].\end{aligned} \tag{4.1.9}$$

Proof. Let us fix any $(x, w) \in \mathcal{H}$. From the description of $\tilde{u}_t$ and $\tilde{w}_t$ in (4.1.8) and the convexity of Φ in (4.1.2) we have

$$\Phi(\tilde{u}_T, \tilde{w}_T) - \Phi(u, w) \leq \frac{1}{T}[\Phi(u_t, w_t) - \Phi(u, w)]. \tag{4.1.10}$$

Let us estimate a bound of $\Phi(u_t, w_t) - \Phi(u, w)$. From the optimality condition of u_t in (4.1.4) we have

$$\begin{aligned}&\langle g + K^\top v_{t-1} + \theta_t K^\top(Ku_{t-1} - w_{t-1}), u_t - u\rangle \\ \leq& \frac{\beta}{2}\left(\|u - x\|^2 - \|u_t - x\|^2 - \|u_t - u\|^2\right) \\ &+ \frac{\eta_t}{2}\left(\|u - u_{t-1}\|^2 - \|u_t - u_{t-1}\|^2 - \|u_t - u\|^2\right).\end{aligned}$$

From the optimality condition of w_t in (4.1.5) we also have

$$\begin{aligned}&\langle -v_{t-1}, w_t - w\rangle + h(w_t) - h(w)\\ \leq &\frac{\tau_t}{2}(\|w - Ku_t\|^2 - \|w_t - Ku_t\|^2 - \|w_t - w\|^2).\end{aligned}$$

Applying the above two relations and the definition of Φ in (4.1.2) we have

$$\begin{aligned}&\Phi(u_t, w_t) - \Phi(u, w)\\ =&\langle g, u_t - u\rangle + h(w_t) - h(w) + \frac{\beta}{2}\|u_t - x\|^2 - \frac{\beta}{2}\|u - x\|^2\\ \leq &-\langle K^\top v_{t-1} + \theta_t K^\top(Ku_{t-1} - w_{t-1}), u_t - u\rangle - \frac{\beta}{2}\|u_t - u\|^2\\ &+ \frac{\eta_t}{2}\left(\|u - u_{t-1}\|^2 - \|u_t - u_{t-1}\|^2 - \|u_t - u\|^2\right)\\ &+ \langle v_{t-1}, w_t - w\rangle + \frac{\tau_t}{2}(\|w - Ku_t\|^2 - \|w_t - Ku_t\|^2 - \|w_t - w\|^2).\end{aligned}$$

In the above, noting that $(u, w) \in \mathcal{H}$ (thus $Ku = w$) and using the description of v_t in (4.1.6) we have

$$\begin{aligned}&-\langle K^\top v_{t-1} + \theta_t K^\top(Ku_{t-1} - w_{t-1}), u_t - u\rangle + \langle v_{t-1}, w_t - w\rangle - \frac{\tau_t}{2}\|w_t - Ku_t\|^2\\ =&-\frac{\tau_t}{2}\|w_t - Ku_t\|^2 + \langle v_{t-1}, w_t - Ku_t\rangle - \theta_t\langle Ku_{t-1} - w_{t-1}, Ku_t - Ku\rangle\\ =&-\frac{\tau_t}{2\rho_t^2}\|v_{t-1} - v_t\|^2 + \frac{1}{2\rho_t}\left(\|v_{t-1} - v_t\|^2 + \|v_{t-1}\|^2 - \|v_t\|^2\right)\\ &-\frac{\theta_t}{2}(\|Ku_{t-1} - Ku\|^2 - \|Ku_{t-1} - Ku_t\|^2 - \|w_{t-1} - w\|^2 + \|w_{t-1} - Ku_t\|^2).\end{aligned} \tag{4.1.11}$$

Summarizing the above two relations, rearranging terms, and noting that $Ku = w$, we obtain

$$\begin{aligned}&\Phi(u_t, w_t) - \Phi(u, w)\\ \leq &\frac{\eta_t}{2}\|u - u_{t-1}\|^2 - \frac{\beta + \eta_t}{2}\|u_t - u\|^2 + \left(-\frac{\eta_t}{2}\|u_t - u_{t-1}\|^2 + \frac{\theta_t}{2}\|Ku_{t-1} - Ku_t\|^2\right)\\ &+ \frac{1}{2\rho_t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) + \frac{\theta_t}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2)\\ &- \frac{\theta_t}{2}(\|Ku_{t-1} - Ku\|^2 - \|Ku_t - Ku\|^2)\\ &- \frac{\tau_t - \rho_t}{2\rho_t^2}\|v_{t-1} - v_t\|^2 + \frac{\tau_t - \theta_t}{2}\|Ku_t - Ku\|^2 - \frac{\tau_t - \theta_t}{2}\|w_t - w\|^2 - \frac{\theta_t}{2}\|w_{t-1} - Ku_t\|^2\end{aligned}$$

Applying the above result to (4.1.10) and observing from (4.1.7) that

$$-\frac{\eta_t}{2}\|u_t - u_{t-1}\|^2 + \frac{\theta_t}{2}\|Ku_{t-1} - Ku_t\|^2 \leq -\frac{\eta_t - \theta_t\|K\|^2}{2}\|u_t - u_{t-1}\|^2 \leq 0,$$

we conclude (4.1.9).

□

There are many possible parameter choices that apply to the above proposition. In the following corollary we provide one example of parameter choices.

Corollary 4.1. Suppose that the parameters in Algorithm 4.1 are set to

$$\eta_t = \theta\|K\|^2 + \beta(t-1), \tau_t = \theta_t \equiv \theta, \text{ and } \rho_t \equiv \sigma \tag{4.1.12}$$

for some positive constants θ and σ with $\theta \geq \sigma$. We have for all $(u, w) \in \mathcal{H}$ that

$$\begin{aligned}\Delta_T(u,w) \leq & \frac{\theta\|K\|^2}{2T}(\|u_0 - u\|^2 - \|u_T - u\|^2) - \frac{\beta}{2}\|u_T - u\|^2 + \frac{1}{2\sigma T}\left(\|v_0\|^2 - \|v_T\|^2\right) \\ & + \frac{\theta}{2T}(\|w_0 - w\|^2 - \|w_T - w\|^2) - \frac{\theta}{2T}(\|Ku_0 - Ku\|^2 - \|Ku_T - Ku\|^2).\end{aligned}$$

Proof. Observing that $\eta_t \geq \theta_t\|K\|^2$, hence condition (4.1.7) holds and we can apply Proposition 4.1. Noting that $\tau_t \equiv \theta \geq \sigma \equiv \rho_t$ and applying the parameter setting (4.1.12) to the description of $\Delta_T(u, w)$ in (4.1.9) and simplifying terms, we have

$$\begin{aligned}\Delta_T(u,w) \leq & \frac{1}{T}\sum_{t=1}^{T}\Bigg[\left(\frac{\theta\|K\|^2}{2} + \frac{\beta(t-1)}{2}\right)\|u_{t-1} - u\|^2 - \left(\frac{\theta\|K\|^2}{2} + \frac{\beta t}{2}\right)\|u_t - u\|^2 \\ & + \frac{1}{2\sigma}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) \\ & + \frac{\theta}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) - \frac{\theta}{2}(\|Ku_{t-1} - Ku\|^2 - \|Ku_t - Ku\|^2)\Bigg] \\ = & \frac{\theta\|K\|^2}{2T}(\|u_0 - u\|^2 - \|u_T - u\|^2) - \frac{\beta}{2}\|u_T - u\|^2 + \frac{1}{2\sigma T}\left(\|v_0\|^2 - \|v_T\|^2\right) \\ & + \frac{\theta}{2T}(\|w_0 - w\|^2 - \|w_T - w\|^2) - \frac{\theta}{2T}(\|Ku_0 - Ku\|^2 - \|Ku_T - Ku\|^2).\end{aligned}$$

□

Now that we have a method of solving (2.0.4), we are ready to propose our sliding algorithm for solving problem (CO). Applying L-ADMM to solve the associated NAGD subproblem in (4.1.1),

we obtain the gradient sliding ADMM (GS-ADMM) algorithm, described in Algorithm 4.2.

Algorithm 4.2 Gradient sliding alternating direction method of multipliers (GS-ADMM)

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

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

Compute

$$\underline{x}_k = (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1} \tag{4.1.13}$$

$$(\tilde{x}_k, \tilde{z}_k, x_k, y_k, z_k,) = \text{ApproxGS}(\nabla f(\underline{x}_k), x_{k-1}, y_{k-1}, z_{k-1}, \beta_k, T_k) \tag{4.1.14}$$

$$\overline{x}_k = (1-\gamma_k)\overline{x}_{k-1} + \gamma_k \tilde{x}_k \tag{4.1.15}$$

$$\overline{z}_k = (1-\gamma_k)\overline{z}_{k-1} + \gamma_k \tilde{z}_k \tag{4.1.16}$$

end for

Output $\overline{x}_N$.

procedure $(\tilde{z}^+, \tilde{x}^+, x^+, y^+, z^+) = \textsc{ApproxGS}(g, x, y, z, \beta, T)$

Apply L-ADMM in Algorithm 4.1 to solve problem (4.1.2) with initial values $u_0 = x$, $v_0 = y$, and $w_0 = z$. Obtain the iterates u_T, v_T, and w_T of the algorithm and the average $\tilde{w}_T$ and $\tilde{u}_T$ (defined in (4.1.8)) after T iterations.

Output $\tilde{z}^+ := \tilde{w}_T$, $\tilde{x}^+ := \tilde{u}_T$, $x^+ := u_T$, $y^+ := v_T$, and $z^+ := w_T$.

end procedure

In GS-ADMM, iterations (4.1.13)–(4.1.15) resembles that of NAGD in Algorithm 2.1, while the subproblem in the x_k step (2.0.4) in NAGD is replaced by a call to the ApproxGS procedure. Specifically, in the k-th call to the ApproxGS procedure (4.1.14), the procedure performs T_k iterations of the L-ADMM algorithm described in Algorithm 4.1 to compute an approximate solution to the problem in the x_k step (2.0.4) in NAGD, or equivalently, problem (4.1.2) with $g = \nabla f(\underline{x}_k)$ and $x = x_{k-1}$. By Proposition 4.1, for properly chosen parameters we have

$$\left[\langle \nabla f(\underline{x}_k), \tilde{x}_k\rangle + h(\tilde{z}_k) + \frac{\beta_k}{2}\|\tilde{x}_k - x_{k-1}\|^2\right] - \left[\langle \nabla f(\underline{x}_k), u\rangle + h(w) + \frac{\beta_k}{2}\|u - x_{k-1}\|^2\right] \tag{4.1.17}$$
$$\leq \Delta_{T_k}(u, w),\ \forall (u, w) \in \mathcal{H}.$$

Note that the results x_{k-1}, y_{k-1}, and z_{k-1} computed during the $(k-1)$-th call of the ApproxGS procedure are supplied as initial values in the k-th call, allowing a "warm-start" of the L-ADMM algorithm. The approximated average solution $\tilde{u}_{T_k}$ computed after T_k iterations of the L-ADMM algorithm is delivered to $\tilde{x}_k$, which is then used to compute the approximate solution output $\overline{x}_k$. Using the above result (4.1.17), we can derive the convergence of Algorithm 4.2.

Theorem 4.1. Suppose that the parameters η_t and θ_t in Algorithm 4.2 satisfy condition (4.1.7)

and we choose β_k and γ_k such that

$$\beta_k \geq L\gamma_k, \tag{4.1.18}$$

then we have

$$F(\overline{x}_k, \overline{z}_k) - (1-\gamma_k)F(\overline{x}_{k-1}, \overline{z}_{k-1}) - \gamma_k F(u,w) \leq \gamma_k \left[\frac{\beta_k}{2}\|x_{k-1} - u\|^2 + \Delta_{T_k}(u,w) \right], \ \forall u, w \in \mathcal{H}. \tag{4.1.19}$$

Here F is defined in problem (ACO) and $\Delta_{T_k}(u,w)$ follows the definition in (4.1.9).

Proof. Let us fix any $(u,w) \in \mathcal{H}$. By the Lipschitz condition (2.0.1) and the convexity of f, we have

$$\begin{aligned}
& f(\overline{x}_k) - (1-\gamma_k)f(\overline{x}_{k-1}) - \gamma_k f(u) \\
\leq & f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), \overline{x}_k - \underline{x}_k \rangle + \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2 \\
& - (1-\gamma_k)[f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), \overline{x}_{k-1} - \underline{x}_k \rangle] \\
& - \gamma_k [f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), u - \underline{x}_k \rangle] \\
= & \langle \nabla f(\underline{x}_k), \overline{x}_k - (1-\gamma_k)\overline{x}_{k-1} - \gamma_k u \rangle + \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2 \\
= & \gamma_k \left[\langle \nabla f(\underline{x}_k), \tilde{x}_k - u \rangle + \frac{L\gamma_k}{2}\|\tilde{x}_k - x_{k-1}\|^2 \right].
\end{aligned}$$

Here in the last inequality we use the description of $\overline{x}_k$ and $\underline{x}_k$ in (4.1.15) and (4.1.13). Also, by the convexity of h and the description of $\overline{z}_k$ in (4.1.16) we have

$$h(\overline{z}_k) - (1-\gamma_k)h(\overline{z}_{k-1}) - \gamma_k h(w) \leq \gamma_k (h(\tilde{z}_k) - h(w)).$$

Applying the above two relations and (4.1.17),

$$\begin{aligned}
& [F(\overline{x}_k, \overline{z}_k) - F(u,w)] - (1-\gamma_k)\left[F(\overline{x}_{k-1}, \overline{z}_{k-1}) - F(u,w)\right] \\
\leq & \gamma_k \left[\langle \nabla f(\underline{x}_k), \tilde{x}_k - u \rangle + \frac{L\gamma_k}{2}\|\tilde{x}_k - x_{k-1}\|^2 + h(\tilde{z}_k) - h(w) \right]. \\
\leq & \gamma_k \left[-\frac{\beta_k - L\gamma_k}{2}\|\tilde{x}_k - x_{k-1}\|^2 + \frac{\beta_k}{2}\|u - x_{k-1}\|^2 + \Delta_{T_k}(u,w) \right].
\end{aligned}$$

Applying (4.1.18) to the above we conclude (4.1.19). □

In the following corollary we describe a possible parameter setting of Algorithm 4.2.

Corollary 4.2. Suppose that the parameters β_k, γ_k, and T_k in Algorithm 4.2 are set to

$$T_k \equiv T, \gamma_k = \frac{2}{k+1} \text{ and } \beta_k = \frac{2L}{k}, \tag{4.1.20}$$

and that the parameters in the k-th call to the ApproxGS procedure in Algorithm 4.2 are set to

$$\eta_t = \frac{\sigma N}{k}\|K\|^2 + \beta(t-1), \tau_t = \theta_t \equiv \frac{\sigma N}{k}, \text{ and } \rho_t \equiv \frac{\sigma k}{N} \tag{4.1.21}$$

for some positive integer T and real number σ. Then we have for all $(u, w) \in \mathcal{H}$ that

$$\begin{aligned} &F(\overline{x}_k, \overline{z}_k) - F(u, w) \\ &\leq \left[\frac{2L}{N(N+1)} + \frac{\sigma\|K\|^2}{T(N+1)}\right]\|x_0 - u\|^2 + \frac{1}{\sigma T(N+1)}\left(\|y_0\|^2 - \|y_N\|^2\right) + \frac{\sigma}{T(N+1)}\|z_0 - w\|^2. \end{aligned} \tag{4.1.22}$$

Proof. Note that the parameters described in (4.1.21) satisfies (4.1.12) (with $\theta := \sigma N/k$ and $\sigma{:=}\sigma k/N$). Using the parameter values in (4.1.20) and (4.1.21), applying Corollary 4.1 and noting the initial value and output of the k-th call to the ApproxGS procedure, we have for any $(u, w) \in \mathcal{H}$ that

$$\begin{aligned} \Delta_{T_k}(u, w) \leq &\frac{\sigma N\|K\|^2}{2Tk}(\|x_{k-1} - u\|^2 - \|x_k - u\|^2) - \frac{L}{k}\|x_k - u\|^2 + \frac{N}{2\sigma Tk}\left(\|y_{k-1}\|^2 - \|y_k\|^2\right) \\ &+ \frac{\sigma N}{2Tk}(\|z_{k-1} - w\|^2 - \|z_k - w\|^2) - \frac{\sigma N}{2Tk}(\|Kx_{k-1} - Ku\|^2 - \|Kx_k - Ku\|^2). \end{aligned}$$

Applying (4.1.20) and the above result to Theorem 4.1, the result (4.1.20) in the theorem becomes

$$\begin{aligned} &[F(\overline{x}_k, \overline{z}_k) - F(u, w))] - \frac{k-1}{k+1}\left[F(\overline{x}_{k-1}, \overline{z}_{k-1}) - F(u, w))\right] \\ \leq &\frac{2}{k+1}\left[\frac{L}{k}(\|x_{k-1} - u\|^2 - \|x_k - u\|^2)\right. \\ &+ \frac{\sigma N\|K\|^2}{2Tk}(\|x_{k-1} - u\|^2 - \|x_k - u\|^2) + \frac{N}{2\sigma Tk}\left(\|y_{k-1}\|^2 - \|y_k\|^2\right) \\ &\left.+ \frac{\sigma N}{2Tk}(\|z_{k-1} - w\|^2 - \|z_k - w\|^2) - \frac{\sigma N}{2Tk}(\|Kx_{k-1} - Ku\|^2 - \|Kx_k - Ku\|^2)\right]. \end{aligned}$$

Multiplying both sides by $[k(k+1)]/[N(N+1)]$ and summing from $k = 1, \ldots, N$ we have

$$\begin{aligned}
&F(\overline{x}_k, \overline{z}_k) - F(u, w) \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{\sigma\|K\|^2}{T(N+1)}\right](\|x_0 - u\|^2 - \|x_N - u\|^2) + \frac{1}{\sigma T(N+1)}\left(\|y_0\|^2 - \|y_N\|^2\right) \\
&+ \frac{\sigma}{T(N+1)}(\|z_0 - w\|^2 - \|z_N - w\|^2) - \frac{\sigma}{T(N+1)}(\|Kx_0 - Ku\|^2 - \|Kx_N - Ku\|^2).
\end{aligned}$$

In the above, note that

$$\begin{aligned}
&- \left[\frac{2L}{N(N+1)} + \frac{\sigma\|K\|^2}{T(N+1)}\right]\|x_N - u\|^2 - \frac{\sigma}{T(N+1)}\|z_N - w\|^2 \\
&- \frac{\sigma}{T(N+1)}(\|Kx_0 - Ku\|^2 - \|Kx_N - Ku\|^2) \\
\leq &- \frac{\sigma\|K\|^2}{T(N+1)}\|x_N - u\|^2 + \frac{\sigma}{T(N+1)}\|Kx_N - Ku\|^2 \leq 0,
\end{aligned}$$

We conclude (4.1.22). □

Let us briefly consider a simple form of GS-ADMM with $T_k \equiv T = 1$. Since we only run one iteration of Algorithm 4.1 in each call to the ApproxGS procedure, we can write the computation of x_k, y_k, and z_k explicitly. Specifically, using the parameters in the above corollary with $T = 1$, we have the following steps in the k-th iteration of GS-ADMM:

$$\begin{aligned}
\underline{x}_k &= \frac{k-1}{k+1}\overline{x}_{k-1} + \frac{2}{k+1}x_{k-1} \\
x_k &= \mathrm{argmin}_{u \in X}\langle \nabla f(\underline{x}_k), u\rangle + \left\langle K^\top\left(y_{k-1} + \frac{\sigma N}{k}(Kx_{k-1} - z_{k-1})\right), u\right\rangle \\
&\qquad + \frac{2L + \sigma N\|K\|^2}{2k}\|u - x_{k-1}\|^2 \\
z_k &= \mathrm{argmin}_{w \in Z}\, h(w) - \langle y_{k-1}, w\rangle + \frac{\sigma N}{2k}\|z - Kx_k\|^2 \\
y_k &= y_{k-1} + \frac{\sigma k}{N}(Kx_t - z_t) \\
\overline{x}_k &= \frac{k-1}{k+1}\overline{x}_{k-1} + \frac{2}{k+1}x_k
\end{aligned}$$

The above iteration is indeed the same as that in A-ADMM described in Algorithm 2.7 (with $\theta_k = \tau_k = \sigma N/k$, $\sigma_k = \sigma k/N$, and $\eta_k = (2L + \sigma N\|K\|^2)/k$). The parameters we choose are in fact exactly the same as the ones described in Theorem 2.9 in [17]. Therefore, we conclude that

GS-ADMM includes A-ADMM as a special case (when $T_k \equiv T = 1$). Equivalently, A-ADMM can be interpreted as an algorithm that uses one iteration of L-ADMM to compute an approximate solution to the possibly sophisticated subproblem in the x_k step (2.0.4) of NAGD.

However, although A-ADMM can be interpreted clearly under the GS-ADMM framework as the case when $T_k \equiv T = 1$, from the result (4.1.22) in Theorem 4.1, we can observe that $T = 1$ is not necessarily the best choice. This is because that the right hand side of result (4.1.22) is either in order $\mathcal{O}(1/N^2)$ or $\mathcal{O}(1/(TN))$. If we are allowed to select $T_k \equiv T = \mathcal{O}(N)$ instead of $T_k \equiv 1$ in GS-ADMM, we can possibly obtain an estimate in (4.1.22) that has significantly better dependence on N.

Corollary 4.3. Suppose that $Y := \operatorname{dom} h^*$ is compact, and that the parameters in Algorithm 4.2 are set as in Corollary 4.2. Then we have

$$f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \left[\frac{2L}{N(N+1)} + \frac{2\sigma\|K\|^2}{T(N+1)}\right] D_X^2 + \frac{1}{\sigma T(N+1)} D_Y^2, \tag{4.1.23}$$

where D_X and D_Y are upper estimates of the distance from x_0 to the set of optimal solutions to problem (CO) and the largest norm $\sup_{y\in Y} \|y\|$ among elements in Y, respectively. Specifically, if we set

$$\sigma := \frac{D_Y}{\|K\| D_X} \text{ and } T := \left\lceil \frac{N\|K\| D_Y}{L D_X} \right\rceil \tag{4.1.24}$$

in the parameter settings in Corollary 4.2, then the number of gradient evaluations of ∇f and operator evaluators (involving K and $K^\top$) are bounded by

$$N_{\nabla f} := \sqrt{\frac{5LD_X^2}{\varepsilon}} \text{ and } N_K := \frac{5\|K\| D_X D_Y}{\varepsilon} + \sqrt{\frac{5LD_X^2}{\varepsilon}}$$

respectively.

Proof. Consider the k-th call to the ApproxGS procedure. Noting that the ApproxGS procedure performance $T_k \equiv T$ iterations of L-ADMM in Algorithm 4.1 with $\rho_t \equiv \sigma k/N$, by the description of

v_t in (4.1.6) we have

$$v_t = v_{t-1} + \frac{\sigma k}{N}(Ku_t - w_t).$$

Summing from $t = 1, \ldots, T_k$ and noting the definition of average sequence $\tilde{u}_t$ and $\tilde{w}_t$ in (4.1.8) we have

$$v_{T_k} = v_0 + \frac{\sigma k}{N}\sum_{t=1}^{T_k}(Ku_t - w_t) = v_0 + \frac{\sigma k T_k}{N}(K\tilde{u}_{T_k} - \tilde{w}_{T_k}).$$

Observing the setting of initial and output values in the description of the ApproxGS procedure in Algorithm 4.2 and recalling that $T_k \equiv T$, the above becomes

$$y_k = y_{k-1} + \frac{\sigma k T}{N}(K\tilde{x}_k - \tilde{z}_k).$$

Applying the above result, the description of $\overline{x}_k$ and $\overline{z}_k$ in (4.1.15) and (4.1.16) respectively, and noting that the parameter $\gamma_k = 2/(k(k+1))$, we have the following recursion:

$$\begin{aligned} K\overline{x}_k - \overline{z}_k =& \frac{k-1}{k+1}(K\overline{x}_{k-1} - \overline{z}_{k-1}) + \frac{2}{k+1}(\tilde{x}_k - \tilde{z}_k) \\ =& \frac{k-1}{k+1}(K\overline{x}_{k-1} - \overline{z}_{k-1}) + \frac{2N}{\sigma k(k+1)T}(y_k - y_{k-1}). \end{aligned}$$

Multiplying by $[k(k+1)]/[N(N+1)]$ and summing from $k = 1, \ldots, N$ we have

$$K\overline{x}_k - \overline{z}_k = \frac{2}{\sigma T(N+1)}(y_N - y_0).$$

Therefore, we have the following relation for any $v \in Y$:

$$\begin{aligned} \langle v, K\overline{x}_k - \overline{z}_k \rangle =& \frac{2}{\sigma T(N+1)}\langle v, y_N - y_0 \rangle \\ =& \frac{1}{\sigma T(N+1)}\left[\|y_N\|^2 - \|y_N - v\|^2 - \|y_0\|^2 + \|y_0 - v\|^2\right]. \end{aligned}$$

Applying the above relation to the result (4.1.22) of Corollary 4.2, we obtain

$$\begin{aligned}
&F(\overline{x}_k, \overline{z}_k) - F(u, w) + \langle v, K\overline{x}_k - \overline{z}_k \rangle \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{\sigma \|K\|^2}{T(N+1)} \right] \|x_0 - u\|^2 + \frac{1}{\sigma T(N+1)} \left(\|y_0 - v\|^2 - \|y_N - v\|^2 \right) \\
&+ \frac{\sigma}{T(N+1)} \|z_0 - w\|^2, \ \forall v \in Y.
\end{aligned}$$

In the above result, applying the initial value $y_0 = 0$ and $z_0 = Kx_0$ in Algorithm 4.2, the definition of D_X and D_Y, choosing $u = x^*$ and $w = z^* = Kx^*$, and noting that $\|Kx_0 - Kx^*\| \leq \|K\| \|x_0 - x^*\| \leq D_X$, we have

$$\begin{aligned}
&F(\overline{x}_k, \overline{z}_k) - F^* + \langle v, K\overline{x}_k - \overline{z}_k \rangle \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{\sigma \|K\|^2}{T(N+1)} \right] \|x_0 - x^*\|^2 + \frac{1}{\sigma T(N+1)} \|v\|^2 \\
&+ \frac{\sigma}{T(N+1)} \|Kx_0 - Kx^*\|^2 \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{2\sigma \|K\|^2}{T(N+1)} \right] D_X^2 + \frac{1}{\sigma T(N+1)} D_Y^2, \ \forall v \in Y.
\end{aligned}$$

Letting $v = h'(K\overline{x}_k) \in Y$ be a subgradient of h at $K\overline{x}_k$, we have

$$h(\overline{z}_k) + \langle v, K\overline{x}_k - \overline{z}_k \rangle \geq h(K\overline{x}_k),$$

thus

$$\begin{aligned}
&f(\overline{x}_k) + h(K\overline{x}_k) - F^* \\
\leq &f(\overline{x}_k) + h(\overline{z}_k) + \langle v, K\overline{x}_k - \overline{z}_k \rangle - F^* \\
= &F(\overline{x}_k, \overline{z}_k) - F^* + \langle v, K\overline{x}_k - \overline{z}_k \rangle \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{2\sigma \|K\|^2}{T(N+1)} \right] D_X^2 + \frac{1}{\sigma T(N+1)} D_Y^2.
\end{aligned}$$

Specifically, setting σ and T to (4.1.24) we have

$$\begin{aligned}
&f(\overline{x}_k) + h(K\overline{x}_k) - F^* \\
\leq &\left[\frac{2L}{N(N+1)} + \frac{2L}{N(N+1)}\right] D_X^2 + \frac{L\|x_0 - x^*\|^2}{N(N+1)} = \frac{5LD_X^2}{N(N+1)}.
\end{aligned}$$

Let us evaluate the number of gradient evaluations and K operations needed to compute an ε-approximate solution to (CO). In order to have $f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \varepsilon$, we need

$$N \geq N_{\nabla f} := \sqrt{\frac{5LD_X^2}{\varepsilon}}$$

iterations of GS-ADMM. With $N := N_{\nabla f}$ iterations of GS-ADMM, we perform $N_{\nabla f}$ gradient evaluations of ∇f and the same number of calls to the ApproxGS procedure. In the k-th call to the ApproxGS procedure we run $T_k \equiv T$ iterations of L-ADMM with T operator evaluations involving K and $K^\top$. Therefore, the total number of K and $K^\top$ operations is bounded by

$$\begin{aligned}
N_K :=& \sum_{k=1}^{N_{\nabla f}} T_k = \sum_{k=1}^{N_{\nabla f}} \left\lceil \frac{N\|K\|D_Y}{LD_X} \right\rceil = N_{\nabla f} \left\lceil N_{\nabla f} \frac{\|K\|D_Y}{LD_X} \right\rceil \\
\leq& N_{\nabla f} \left(N_{\nabla f} \frac{\|K\|D_Y}{LD_X} + 1 \right) \\
=& \frac{5\|K\|D_X D_Y}{\varepsilon} + \sqrt{\frac{5LD_X^2}{\varepsilon}}.
\end{aligned}$$

□

A few remarks are in place for the above result. First, when X and Y are both compact and we have knowledge of both diameters, then we can simply set D_X and D_Y to be the diameters of X and Y respectively. Second, from the setting in (4.1.24), only the ratio between D_Y and D_X is required for achieving the above complexity result, instead of the knowledge of both values. Finally, from the above result, in order to compute an ε-approximate solution to (ACO), the gradient evaluations of ∇f and operator evaluations (involving K and $K^\top$) required by GS-ADMM are bounded by

$$\mathcal{O}\left(\sqrt{\tfrac{LD_X^2}{\varepsilon}}\right) \text{ and } \mathcal{O}\left(\sqrt{\tfrac{LD_X^2}{\varepsilon}} + \tfrac{\|K\|D_X D_Y}{\varepsilon}\right)$$

respectively. Let us compare the above result with that of A-ADMM. Let set σ as in (4.1.24) but $T := 1$, then by (4.1.23) we have

$$f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \left[\frac{2L}{N(N+1)} + \frac{2\|K\|D_Y}{(N+1)D_X}\right] D_X^2 + \frac{\|K\|D_X D_Y}{N+1} = \frac{2LD_X^2}{N(N+1)} + \frac{3\|K\|D_X D_Y}{N+1}.$$

Since A-ADMM performs one gradient evaluation of f and operator evaluation (involving K and $K^\top$) in each iteration, in order to compute an ε-solution to (ACO) both gradient and operator evaluations are bounded by

$$\mathcal{O}\left(\sqrt{\frac{LD_X^2}{\varepsilon}} + \frac{\|K\|D_X D_Y}{\varepsilon}\right). \tag{4.1.25}$$

Therefore, the GS-ADMM has significantly better performance than that of A-ADMM in terms of the number of evaluations of ∇f required. Looking further, we can see that the quantity $\mu := \frac{||K||D_Y}{LD_X}$ plays an important role in the differences between GS-ADMM and A-ADMM. Whenever $\mu \leq 1/N$, we see that from (4.1.24) that $T = 1$ in GS-ADMM. That is, for small enough values of μ, GS-ADMM reduces to A-ADMM. Put differently, A-ADMM achieves the optimal gradient evaluation complexity whenever $\mu \leq 1/N$. Indeed, applying this relation, the evaluations of ∇f in (4.1.25) reduces to $\mathcal{O}(\sqrt{LD_X^2/\varepsilon})$. In view of this fact, we note that A-ADMM achieves optimal gradient evaluation complexity for problems with a particular set of problem dependent parameters, namely $\mu \leq 1/N$. The novelty of GS-ADMM is that this method achieves the optimal number of gradient evaluations for any set of L, $||K||$, D_X, and D_Y.

Note, however, that due to the choice of parameters in (4.1.21) and T in (4.1.24), we must specify in advance the maximum number of iterations N. Such a property is not ideal as it does not allow us to perform more iterations if our original desired tolerance is not sufficient. In the numerical experiments later, we will see a problem instance that suffers from this phenomenon. Therefore, it is often desirable to design modifications whose parameters are not dependent on the max iteration count N. Before discussing further sliding algorithms, we now turn to addressing this problem for GS-ADMM and consequently A-ADMM as well.

4.2 Operator Sliding

Previously, we applied sliding to ADMM type algorithms to reduce the number of overall gradient evaluations required for an ε-solution. Similarly, we can also use sliding to improve the number of operator evaluations required. Here, we present a modified ADMM type algorithm which computes an ε-approximate solution to (ACO) with $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ gradient evaluations and $\mathcal{O}(||K||/\varepsilon)$ operator evaluations.

Similar to our previous discussion of GS-ADMM, our proposed operator sliding ADMM (OS-ADMM) algorithm in this section is based on the works of NAGD and ADMM. Here, ADMM will serve as our foundation algorithm and we will utilize a more aggressive strategy of computing the x_k subproblem. Rather than linearizing the problem, we will instead use NAGD iterations to compute an approximate solution to the x_k subproblem in (2.3.2). In doing so, we will show that an ε-approximate solution is achieved in only $\mathcal{O}(||K||/\varepsilon)$ iterations of ADMM. By properly choosing the tolerance of the approximate solution of (2.3.2), we will also show that we can keep the gradient evaluations to the aforementioned $\mathcal{O}(\sqrt{L/\varepsilon}+||K||/\varepsilon)$. To do so, let us consider the NAGD algorithm for solving problem

$$\min_{u\in X} \psi(u) := f(u) + \langle l, u\rangle + \tfrac{\eta}{2}\|u - x\|^2. \tag{4.2.1}$$

Applying a modified NAGD, which we will refer to as NEST, we have the algorithm described in Algorithm 4.3. For convergence of Algorithm 4.3, we have the following proposition and its corollary.

Algorithm 4.3 NEST for solving (4.2.1)

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

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

Compute

$$\underline{x}_k = (1-\lambda)\tilde{x}_0 + \lambda(1-\gamma_k)\tilde{x}_{k-1} + \lambda\gamma_k x_{k-1}, \tag{4.2.2}$$

$$x_k = \operatorname*{argmin}_{u\in X} \langle \nabla f(\underline{x}_k) + l, u\rangle + \tfrac{\eta}{2}\|u - x\|^2 + \tfrac{\beta_k}{2}\|u - x_{k-1}\|^2, \tag{4.2.3}$$

$$\tilde{x}_k = (1-\gamma_k)\tilde{x}_{k-1} + \gamma_k x_k. \tag{4.2.4}$$

end for

We will again make use the following quantity:

$$\Gamma_k := \begin{cases} 1 & k = 1 \\ (1-\gamma_k)\Gamma_{k-1} & k > 1. \end{cases} \tag{4.2.5}$$

Proposition 4.2. Suppose that $\gamma_1 = 1$, $\gamma_k \in (0,1)$, $k = 2, \ldots, N$, and

$$\beta_k \geq L\lambda\gamma_k. \tag{4.2.6}$$

Then we have

$$\begin{aligned} &[\psi(\overline{x}_N) - \psi(u)] - (1-\lambda)[\psi(\overline{x}_0) - \psi(u)] \\ \leq &\tfrac{\eta}{2}\|\overline{x}_N - x\|^2 - \tfrac{\eta(1-\lambda)}{2}\|\overline{x}_0 - x\|^2 - \tfrac{\eta\lambda}{2}\|\tilde{x}_N - x\|^2 + \lambda\Upsilon_N(u),\ \forall u \in X, \end{aligned}$$

where

$$\overline{x}_k := (1-\lambda)\tilde{x}_0 + \lambda\tilde{x}_k,\ \forall k = 0, 1, \ldots, N, \text{ and} \tag{4.2.7}$$

$$\Upsilon_N(u) := \Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k}\left[\frac{\beta_k}{2}\|x_{k-1} - u\|^2 - \frac{\beta_k+\eta}{2}\|x_k - u\|^2\right]. \tag{4.2.8}$$

Proof. Let us fix an arbitrary $u \in X$ and define

$$v := (1-\lambda)\tilde{x}_0 + \lambda u \tag{4.2.9}$$

We have the following observations regarding our defined v, $\underline{x}_k$, $\tilde{x}_k$ and $\overline{x}_k$ in (4.2.9), (4.2.2), (4.2.4), and (4.2.7) respectively:

$$\begin{aligned} \overline{x}_k - \underline{x}_k =& \lambda[\tilde{x}_k - (1-\gamma_k)\tilde{x}_{k-1}] - \lambda\gamma_k x_{k-1} = \lambda\gamma_k(x_k - x_{k-1}), \\ \overline{x}_k - (1-\gamma_k)\overline{x}_{k-1} - \gamma_k v =& (\overline{x}_k - \overline{x}_{k-1}) + \gamma_k(\overline{x}_{k-1} - v) = \lambda(\tilde{x}_k - \tilde{x}_{k-1}) + \lambda\gamma_k(\tilde{x}_{k-1} - u) \\ =& \lambda[\tilde{x}_k - (1-\gamma_k)\tilde{x}_{k-1}] - \lambda\gamma_k u = \lambda\gamma_k(x_k - u). \end{aligned}$$

Using the above observations and the convexity and Lipschitz continuity of f in (2.0.1), we have

$$\langle l, \overline{x}_k - (1-\gamma_k)\overline{x}_{k-1} - \gamma_k v\rangle = \lambda\gamma_k\langle l, x_k - u\rangle$$

and

$$\begin{aligned} f(\overline{x}_k) - (1-\gamma_k) f(\overline{x}_{k-1}) - \gamma_k f(v) \leq & f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), \overline{x}_k - \underline{x}_k \rangle + \tfrac{L}{2} \|\overline{x}_k - \underline{x}_k\|^2 \\ & - (1-\gamma_k)[f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), \overline{x}_{k-1} - \underline{x}_k \rangle] \\ & - \gamma_k [f(\underline{x}_k) + \langle \nabla f(\underline{x}_k), v - \underline{x}_k \rangle] \\ = & \lambda \gamma_k \left[\langle \nabla f(\underline{x}_k), x_k - u \rangle + \tfrac{L \lambda \gamma_k}{2} \|x_k - x_{k-1}\|^2 \right]. \end{aligned}$$

Also, from the optimality condition of x_k in (4.2.3) we obtain

$$\begin{aligned} \langle \nabla f(\underline{x}_k) + l, x_k - u \rangle \leq & \tfrac{\beta_k}{2} (\|u - x_{k-1}\|^2 - \|x_k - x_{k-1}\|^2 - \|x_k - u\|^2) \\ & + \tfrac{\eta}{2} (\|u - x\|^2 - \|x_k - x\|^2 - \|x_k - u\|^2) \\ = & \tfrac{\beta_k}{2} \|x_{k-1} - u\|^2 - \tfrac{\beta_k + \eta}{2} \|x_k - u\|^2 - \tfrac{\beta_k}{2} \|x_{k-1} - x_k\|^2 \\ & + \tfrac{\eta}{2} (\|u - x\|^2 - \|x_k - x\|^2). \end{aligned}$$

Using the above three relations and noting (4.2.6) we have the recurrence relation

$$\begin{aligned} & \{[f(\overline{x}_k) + \langle l, \overline{x}_k \rangle] - [f(v) - \langle l, v \rangle]\} - (1-\gamma_k)\{[f(\overline{x}_{k-1}) - \langle l, \overline{x}_{k-1} \rangle] - [f(v) - \langle l, v \rangle]\} \\ = & f(\overline{x}_k) - (1-\gamma_k) f(\overline{x}_{k-1}) - \gamma_k f(v) + \langle l, \overline{x}_k - (1-\gamma_k)\overline{x}_{k-1} - \gamma_k v \rangle \\ \leq & \lambda \gamma_k \left[\langle \nabla f(\underline{x}_k), x_k - u \rangle + \tfrac{L \lambda \gamma_k}{2} \|x_k - x_{k-1}\|^2 + \langle l, x_k - u \rangle \right] \qquad (4.2.10) \\ \leq & \lambda \gamma_k \left[\tfrac{\beta_k}{2} \|x_{k-1} - u\|^2 - \tfrac{\beta_k + \eta}{2} \|x_k - u\|^2 + \tfrac{\eta}{2} (\|u - x\|^2 - \|x_k - x\|^2) - \tfrac{\beta_k - L\lambda\gamma_k}{2} \|x_{k-1} - x_k\|^2 \right] \\ \leq & \lambda \gamma_k \left[\tfrac{\beta_k}{2} \|x_{k-1} - u\|^2 - \tfrac{\beta_k + \eta}{2} \|x_k - u\|^2 + \tfrac{\eta}{2} (\|u - x\|^2 - \|x_k - x\|^2) \right]. \end{aligned}$$

Note from the description of $\tilde{x}_k$ in (4.2.4) and convexity of $||\cdot||^2$ that

$$\begin{aligned} ||\tilde{x}_k - u||^2 &= ||(1-\gamma_k)(\tilde{x}_{k-1} - u) + \gamma_k (x_k - u)||^2 \\ &\leq (1-\gamma_k) ||\tilde{x}_{k-1} - u||^2 + \gamma_k ||x_k - u||^2. \end{aligned}$$

Summing from $k = 1, \ldots, N$ in the above relation we conclude

$$||\tilde{x}_N - u||^2 \leq \Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k} ||x_k - u||^2 = \frac{2}{N(N+1)} \sum_{k=1}^N k ||x_k - u||^2. \qquad (4.2.11)$$

Thus, from the definition of Γ_k in (4.2.5), our assumption on parameters in (4.2.6), the relation in (4.2.11), and that $\gamma_1 = 1$, multiplying by Γ_N/Γ_k and summing from $k = 1, \ldots, N$ in (4.2.10) we have that

$$\begin{aligned}
&[f(\overline{x}_N) + \langle l, \overline{x}_N\rangle] - [f(v) - \langle l, v\rangle] \\
\leq &\lambda\Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k}\left[\frac{\beta_k}{2}\|x_{k-1} - u\|^2 - \frac{\beta_k+\eta}{2}\|x_k - u\|^2 + \frac{\eta}{2}(\|u - x\|^2 - \|x_k - x\|^2)\right] \\
\leq &\frac{\eta\lambda}{2}(\|u - x\|^2 - \|\tilde{x}_N - x\|^2) + \lambda\Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k}\left[\frac{\beta_k}{2}\|x_{k-1} - u\|^2 - \frac{\beta_k+\eta}{2}\|x_k - u\|^2\right]
\end{aligned}$$

where in the last inequality we used the relation

$$\Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k} = 1.$$

Observing that $\overline{x}_0 = \tilde{x}_0$ and that $(1-\lambda)\overline{x}_0 - \lambda u = v$, using the above result we conclude

$$\begin{aligned}
&[\psi(\overline{x}_N) - \psi(u)] - (1-\lambda)[\psi(\overline{x}_0) - \psi(u)] \\
=&[f(\overline{x}_N) - (1-\lambda)f(\overline{x}_0) - \lambda f(u)] + [\langle l, \overline{x}_N - (1-\lambda)\overline{x}_0 - \lambda u\rangle] \\
&+ \tfrac{\eta}{2}\|\overline{x}_N - x\|^2 - \tfrac{\eta(1-\lambda)}{2}\|\overline{x}_0 - x\|^2 - \tfrac{\eta\lambda}{2}\|u - x\|^2 \\
\leq&[f(\overline{x}_N) - f(v)] + [\langle l, \overline{x}_N - v\rangle] + \tfrac{\eta}{2}\|\overline{x}_N - x\|^2 \\
&- \tfrac{\eta(1-\lambda)}{2}\|\overline{x}_0 - x\|^2 - \tfrac{\eta\lambda}{2}\|u - x\|^2 \\
\leq&\tfrac{\eta}{2}\|\overline{x}_N - x\|^2 - \tfrac{\eta(1-\lambda)}{2}\|\overline{x}_0 - x\|^2 - \tfrac{\eta\lambda}{2}\|\tilde{x}_N - x\|^2 \\
&+ \lambda\Gamma_N \textstyle\sum_{k=1}^N \frac{\gamma_k}{\Gamma_k}\left[\frac{\beta_k}{2}\|x_{k-1} - u\|^2 - \frac{\beta_k+\eta}{2}\|x_k - u\|^2\right]
\end{aligned}$$

which finishes the proof. □

An immediate corollary of Proposition 4.2 is a potential parameter setting for simplified analysis.

Corollary 4.4. If the parameters of Algorithm 4.3 are set to

$$\gamma_k = \tfrac{2}{k+1},\ \beta_k = \tfrac{2L\xi}{k} \text{ where } \xi \geq \lambda, \tag{4.2.12}$$

then for all $u \in X$ we have

$$\Upsilon_N(u) = \tfrac{2L\xi}{N(N+1)}(\|x_0 - u\|^2 - \|x_N - u\|^2) - \tfrac{\eta}{2}\|\tilde{x}_N - u\|^2.$$

Proof. By (4.2.12) and (4.2.5) we have $\Gamma_k = 2/(k(k+1))$. Applying the value of Γ_k and the parameter settings in (4.2.12) to (4.2.8) we have

$$\begin{aligned}\Upsilon_N(u) =& \tfrac{2}{N(N+1)} \textstyle\sum_{k=1}^N \left[L\xi\|x_{k-1} - u\|^2 - \left(L\xi + \tfrac{\eta}{2}\right)\|x_k - u\|^2\right] \\ =& \tfrac{2L\xi}{N(N+1)}(\|x_0 - u\|^2 - \|x_N - u\|^2) - \tfrac{2}{N(N+1)}\tfrac{\eta}{2}\textstyle\sum_{k=1}^N k\|x_k - u\|^2\end{aligned}$$

in which we immediately conclude the corollary by noting (4.2.11). □

Algorithm 4.4 Operator sliding alternating direction method of multipliers (OS-ADMM)

Start: Choose $u_0 \in X$. Set $\overline{u}_0 := u_0$, $\tilde{u}_0 := u_0$, $v_0 := 0$ and $w_0 := Ku_0$.
for $t = 1, \ldots, T$ **do**
Compute

$$l_t = K^\top(v_{t-1} + \theta_k(K\tilde{u}_{t-1} - w_{t-1})) \tag{4.2.13}$$
$$(\tilde{u}_t, u_t) = \text{ApproxOS}(l_t, u_{t-1}, \overline{u}_{t-1}, \eta_t, N_t) \tag{4.2.14}$$
$$\overline{u}_t = (1 - \lambda_t)\overline{u}_{t-1} + \lambda\tilde{u}_t \tag{4.2.15}$$
$$w_t = \operatorname*{argmin}_{w \in \mathbb{R}^m} -\langle v_{k-1}, w\rangle + h(w) + \tfrac{\tau_k}{2}\|K\tilde{u}_t - w\|^2 \tag{4.2.16}$$
$$v_t = v_{t-1} + \rho_k(K\tilde{u}_t - w_t) \tag{4.2.17}$$

end for
Output $\overline{x}_N$.
procedure $(\tilde{u}^+, u^+) =$ ApproxOS$(g, u, \overline{u}, v, w, \beta, N)$
Apply NEST in Algorithm 4.3 to solve problem (4.2.1) with $x = \tilde{u}_{t-1}$ and initial values $x_0 = u_{t-1}$, $\tilde{x}_0 = \overline{u}_{t-1}$. Obtain the iterates x_N, $\tilde{x}_N$, and $\overline{x}_N$ of the algorithm.
Output $\tilde{u}^+ := \tilde{x}_N$ and $u^+ := x_N$.
end procedure

We can now describe our novel contribution of this section, the Operator Sliding ADMM (OS-ADMM) algorithm shown in Algorithm 4.4. We can make the following observations regarding OS-ADMM. The iterations (4.2.16) and (4.2.17) resemble that of the ADMM iterations in (2.3.3) and (2.3.4). Indeed, the updating of the artificial and dual variables of OS-ADMM are the same as in ADMM. However, the primal variable, denoted x_k in Algorithm 2.6 and $\tilde{u}_t$ in Algorithm 4.4, is computed approximately in OS-ADMM since the gradient step for the primal variable, i.e. a minimization of the form (2.3.2), is not necessarily easy to compute. The call to ApproxOS

in (4.2.14) is effectively applying NAGD iterations for computing an approximate solution for the primal variable. That is, OS-ADMM can be viewed as an ADMM procedure which computes its primal update using NAGD iterations.

Noting the initial and output iterates of the ApproxOS procedure, we can observe that $\bar{u}_t = \bar{x}_{N_t}$. Moreover, by Proposition 4.2, for properly chosen parameters we have

$$\begin{aligned}
&[f(\overline{u}_t) - f(u) + \langle l_t, \overline{u}_t - u\rangle + \tfrac{\eta_t}{2}\|\overline{u}_t - \tilde{u}_{t-1}\|^2 - \tfrac{\eta_t}{2}\|u - \tilde{u}_{t-1}\|^2] \\
&- (1-\lambda_t)[f(\overline{u}_{t-1}) - f(u) + \langle l_t, \overline{u}_{t-1} - u\rangle + \tfrac{\eta_t}{2}\|\overline{u}_{t-1} - \tilde{u}_{t-1}\|^2 - \tfrac{\eta_t}{2}\|u - \tilde{u}_{t-1}\|^2] \\
\le &\tfrac{\eta_t}{2}\|\overline{u}_t - \tilde{u}_{t-1}\|^2 - \tfrac{\eta_t(1-\lambda_t)}{2}\|\overline{u}_{t-1} - \tilde{u}_{t-1}\|^2 - \tfrac{\eta_t\lambda_t}{2}\|\tilde{u}_t - \tilde{u}_{t-1}\|^2 + \Upsilon_{N_t}(u),\ \forall u \in X.
\end{aligned}$$

Noting the description of $\overline{u}_t$ in (4.2.15) and simplifying the above relation, we have

$$\begin{aligned}
f(\overline{u}_t) - (1-\lambda_t)f(\overline{u}_{t-1}) - \lambda_t f(u) \le &-\lambda_t \big[\ \langle l_t, \tilde{u}_t - u\rangle - \tfrac{\eta_t}{2}\|\tilde{u}_t - \tilde{u}_{t-1}\|^2 \\
&+ \tfrac{\eta_t}{2}\|\tilde{u}_{t-1} - u\|^2 + \Upsilon_{N_t}(u)\ \big],\ \forall u \in X.
\end{aligned} \tag{4.2.18}$$

Using (4.2.18), we can establish the convergence property of Algorithm 4.4.

Theorem 4.2. Suppose that the parameters in Algorithm 4.4 satisfy (4.2.12) and

$$\lambda_1 = 1,\ \theta_t = \tau_t \ge \rho_t,\ \eta_t \ge \theta_t\|K\|^2. \tag{4.2.19}$$

Then we have

$$\begin{aligned}
&[F(\overline{u}_t, \overline{w}_t) - F(u,w)] - (1-\lambda_t)[F(\overline{u}_{t-1}, \overline{w}_{t-1}) - F(u,w)] \\
\le &\lambda_t \big[\ \tfrac{\eta_t}{2}\|\tilde{u}_{t-1} - u\|^2 + \Upsilon_{N_t}(u) + \tfrac{1}{2\rho_t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) + \tfrac{\theta_t}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) \\
&- \tfrac{\theta_t}{2}(\|K\tilde{u}_{t-1} - Ku\|^2 - \|K\tilde{u}_t - Ku\|^2)\ \big],
\end{aligned}$$

where F is defined in problem (ACO) and $\Upsilon_{N_t}(u)$ follows the definition in (4.2.8).

Proof. Let us fix any $(u,w) \in \mathcal{H}$. By the definition of $\mathcal{H}$ we have $Ku = w$. From the optimality condition of (4.2.16) we have

$$\begin{aligned}
\langle -v_{t-1}, w_t - w\rangle + h(w_t) - h(w) \le &\tfrac{\tau_t}{2}(\|w - K\tilde{u}_t\|^2 - \|w_t - K\tilde{u}_t\|^2 - \|w_t - w\|^2) \\
= &\tfrac{\tau_t}{2}(\|Ku - K\tilde{u}_t\|^2 - \|w_t - K\tilde{u}_t\|^2 - \|w_t - w\|^2).
\end{aligned}$$

Also, by the description of l_t in (4.2.13), noting that $Ku = w$, and using (4.1.11), we have

$$\begin{aligned}
&- \langle l_t, \tilde{u}_t - u\rangle + \langle v_{t-1}, w_t - w\rangle - \tfrac{\tau_t}{2}\|w_t - K\tilde{u}_t\|^2 \\
=&- \langle K^\top v_{t-1} + \theta_t K^\top(K\tilde{u}_{t-1} - w_{t-1}), \tilde{u}_t - u\rangle + \langle v_{t-1}, w_t - w\rangle - \tfrac{\tau_t}{2}\|w_t - K\tilde{u}_t\|^2 \\
=&- \theta_t\langle K\tilde{u}_{t-1} - w_{t-1}, K\tilde{u}_t - Ku\rangle + \langle v_{t-1}, w_t - K\tilde{u}_t\rangle - \tfrac{\tau_t}{2}\|w_t - K\tilde{u}_t\|^2 \\
=&- \tfrac{\theta_t}{2}(\|K\tilde{u}_{t-1} - Ku\|^2 - \|K\tilde{u}_{t-1} - K\tilde{u}_t\|^2 - \|w_{t-1} - w\|^2 + \|w_{t-1} - K\tilde{u}_t\|^2) \\
&+ \tfrac{1}{2\rho_t}\left(\|v_{t-1} - v_t\|^2 + \|v_{t-1}\|^2 - \|v_t\|^2\right) - \tfrac{\tau_t}{2\rho_t^2}\|v_{t-1} - v_t\|^2.
\end{aligned}$$

Therefore, combining the two above relations we have

$$\begin{aligned}
h(w_t) - h(w) - \langle l_t, \tilde{u}_t - u\rangle \leq& \tfrac{1}{2\rho_t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) + \tfrac{\theta_t}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) \\
&- \tfrac{\theta_t}{2}(\|K\tilde{u}_{t-1} - Ku\|^2 - \|K\tilde{u}_t - Ku\|^2) + \tfrac{\theta_t}{2}\|K\|^2\|\tilde{u}_t - \tilde{u}_{t-1}\|^2,
\end{aligned}$$

where the last inequality is from our condition of parameters in (4.2.19). Using the above relation, (4.2.18), the description of F in problem (ACO), and the convexity of function h, we have

$$\begin{aligned}
&[F(\overline{u}_t, \overline{w}_t) - F(u, w)] - (1 - \lambda_t)[F(\overline{u}_{t-1}, \overline{w}_{t-1}) - F(u, w)] \\
=&[f(\overline{u}_t) - (1 - \lambda_t)f(\overline{u}_{t-1}) - \lambda_t f(u)] + [h(\overline{w}_t) - (1 - \lambda_t)h(\overline{w}_{t-1}) - \lambda_t h(u)] \\
\leq&[f(\overline{u}_t) - (1 - \lambda_t)f(\overline{u}_{t-1}) - \lambda_t f(u)] + \lambda_t[h(w_t) - h(u)] \\
\leq&\lambda_t\Big[- \tfrac{\eta_t}{2}\|\tilde{u}_t - \tilde{u}_{t-1}\|^2 + \tfrac{\eta_t}{2}\|\tilde{u}_{t-1} - u\|^2 + \Upsilon_{N_t}(u) + \tfrac{\theta_t}{2}\|K\|^2\|\tilde{u}_t - \tilde{u}_{t-1}\|^2 \\
&+ \tfrac{1}{2\rho_t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) + \tfrac{\theta_t}{2}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) \\
&- \tfrac{\theta_t}{2}(\|K\tilde{u}_{t-1} - Ku\|^2 - \|K\tilde{u}_t - Ku\|^2)\Big].
\end{aligned}$$

We conclude the proposition immediately by noting that $\eta_t \geq \theta_t\|K\|^2$ in the condition (4.2.19). □

In the following corollary we describe a possible parameter setting of Algorithm 4.4.

Corollary 4.5. Suppose that the parameters in Algorithm 4.4 are set to

$$N_t \equiv N, \lambda_t = \tfrac{2}{t+1},\ \eta_t = \tfrac{\sigma T}{t}\|K\|^2,\ \tau_t = \theta_t = \tfrac{\sigma T}{t},\ \rho_t = \tfrac{\sigma t}{T}, \tag{4.2.20}$$

and that the parameters in the k-th call to the ApproxOS procedure in Algorithm 4.4 are set to

$$\gamma_k = \tfrac{2}{k+1},\ \beta_k = \tfrac{2L}{kt} \tag{4.2.21}$$

for some positive integer N and real number σ. Then we have for all $(u, w) \in \mathcal{H}$ that

$$F(\overline{u}_T, \overline{w}_T) - F(u, w) \leq \tfrac{8L}{N(N+1)T(T+1)}(\|u_0 - u\|^2 - \|u_T - u\|^2) + \tfrac{1}{\sigma(T+1)}(\|v_0\|^2 - \|v_T\|^2).$$

Proof. Note that the parameters described in (4.2.20) and (4.2.21) satisfy condition (4.2.6) in Proposition 4.2 (with $\lambda = \lambda_t = 2/(t+1)$). Using the parameter values, applying Corollary 4.4 (with $\xi = 2/t$ and $\lambda = \lambda_t = 2/(t+1)$), and noting the initial value and output of the t-th call to the ApproxOS procedure, we have

$$\Upsilon_{N_t}(u) = \tfrac{4L}{N_t(N_t+1)t}(\|u_{t-1} - u\|^2 - \|u_t - u\|^2) - \tfrac{\sigma T\|K\|^2}{2t}\|\tilde{u}_t - u\|^2.$$

Applying the parameter settings in (4.2.20) and the above result to Theorem 4.1, we have

$$\begin{aligned}
&[F(\overline{u}_t, \overline{w}_t) - F(u, w)] - \tfrac{t-1}{t+1}[F(\overline{u}_{t-1}, \overline{w}_{t-1}) - F(u, w)] \\
\leq &\tfrac{2}{t+1}\Big[\ \tfrac{\sigma T\|K\|^2}{2t}\|\tilde{u}_{t-1} - u\|^2 + \tfrac{4L}{N(N+1)t}(\|u_{t-1} - u\|^2 - \|u_t - u\|^2) - \tfrac{\sigma T\|K\|^2}{2t}\|\tilde{u}_t - u\|^2 \\
&+ \tfrac{T}{2\sigma t}\left(\|v_{t-1}\|^2 - \|v_t\|^2\right) + \tfrac{\sigma T}{2t}(\|w_{t-1} - w\|^2 - \|w_t - w\|^2) \\
&- \tfrac{\sigma T}{2t}(\|K\tilde{u}_{t-1} - Ku\|^2 - \|K\tilde{u}_t - Ku\|^2)\ \Big].
\end{aligned}$$

We may then multiply by $[t(t+1)]/[T(T+1)]$ and summing from $t = 1, \ldots, T$ to obtain

$$\begin{aligned}
F(\overline{u}_T, \overline{w}_T) - F(u, w) \leq &\tfrac{\sigma\|K\|^2}{T+1}(\|\tilde{u}_0 - u\|^2 - \|\tilde{u}_T - u\|^2) + \tfrac{8L}{N(N+1)T(T+1)}(\|u_0 - u\|^2 - \|u_T - u\|^2) \\
&+ \tfrac{1}{\sigma(T+1)}(\|v_0\|^2 - \|v_T\|^2) + \tfrac{\sigma}{T+1}(\|w_0 - w\|^2 - \|w_T - w\|^2) \\
&- \tfrac{\sigma}{T+1}(\|K\tilde{u}_0 - Ku\|^2 - \|K\tilde{u}_T - Ku\|^2).
\end{aligned}$$

We conclude the corollary by noting from the above that

$$\begin{aligned}
&-\tfrac{\sigma\|K\|^2}{T+1}\|\tilde{u}_T-u\|^2-\tfrac{\sigma}{T+1}\|w_T-w\|^2-\tfrac{\sigma}{T+1}(\|K\tilde{u}_0-Ku\|^2-\|K\tilde{u}_T-Ku\|^2)\\
&\leq-\tfrac{\sigma\|K\|^2}{T+1}\|\tilde{u}_T-u\|^2+\tfrac{\sigma}{T+1}\|K\tilde{u}_T-Ku\|^2\\
&\leq 0.
\end{aligned}$$

□

Just as in the case for GS-ADMM, the convergence in Theorem 4.2 allows us to choose a set of parameters that generates an upper complexity bound on minimizing problems of the form in (ACO).

Corollary 4.6. Suppose that $Y := \operatorname{dom} h^*$ is compact, and that the parameters in Algorithm 4.4 are set as in Corollary 4.5. Then we have

$$f(\overline{x}_k)+h(K\overline{x}_k)-F^*\leq\tfrac{8L}{N(N+1)T(T+1)}D_X^2+\tfrac{1}{\sigma(T+1)}D_Y^2,$$

where, recall, D_X and D_Y are upper estimates of the distance from x_0 to the set of optimal solutions to problem (ACO) and the largest norm $\sup_{y\in Y}\|y\|$ among elements in Y, respectively. Specifically, if we set

$$\sigma:=\tfrac{D_Y}{\|K\|D_X}\text{ and }N:=\left\lceil\sqrt{\tfrac{LD_X}{T\|K\|D_Y}}\right\rceil \tag{4.2.22}$$

in the parameter settings in Corollary 4.5, then the number of gradient evaluations of ∇f and operator evaluators (involving K and $K^\top$) are bounded by

$$T_K:=\tfrac{9\|K\|D_XD_Y}{\varepsilon}\text{ and }T_{\nabla f}:=\sqrt{\tfrac{9LD_X^2}{\varepsilon}}+\tfrac{9\|K\|D_XD_Y}{\varepsilon}$$

respectively.

Proof. Similar to the proof of Corollary 4.3 we have

$$f(\overline{x}_k)+h(K\overline{x}_k)-F^*\leq\tfrac{8L}{N(N+1)T(T+1)}D_X^2+\tfrac{1}{\sigma T(N+1)}D_Y^2.$$

Noting the choices of σ and N in (4.2.22) we have

$$f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \frac{8LT\|K\|D_Y}{T(T+1)LD_X} D_X^2 + \frac{\|K\|D_X D_Y}{T+1} = \frac{9||K\|D_X D_Y}{T+1}.$$

Let us evaluate the number of gradient evaluations and K operations need to compute an ε-approximate solution to (ACO). In order to have $f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \varepsilon$, we need $T \geq T_K := \frac{9\|K\|D_X D_Y}{\varepsilon}$ iterations of the OS-ADMM algorithm in Algorithm 4.4. With $T := T_K$ iterations of OS-ADMM, we perform T_K operator evaluations involving K and $K^\top$ and the same number of calls to the ApproxOS procedure. In the t-th call to the ApproxOS procedure we run $N_t \equiv N$ iterations of NEST with N gradient evaluations of ∇f. Therefore, the total number of gradient operations is bounded by

$$T_{\nabla f} := \sum_{t=1}^{T_K} N_t = T_K \left\lceil \sqrt{\frac{LD_X}{T_K\|K\|D_Y}} \right\rceil \leq T_K \left(\sqrt{\frac{LD_X}{T_K\|K\|D_Y}} + 1 \right) = \sqrt{\frac{9LD_X^2}{\varepsilon}} + \frac{9\|K\|D_X D_Y}{\varepsilon}.$$

□

4.3 Lower Complexity Bounds for ADMM algorithms

Throughout this chapter, we have introduced two sliding techniques that are both designed to solve problems of the form (ACO). In the discussion surrounding (4.0.1), we argue that to compute an ε-solution, at least $\Omega(\sqrt{L/\varepsilon})$ gradient evaluations and $\Omega(||K||/\varepsilon)$ operator evaluations must be computed. Viewed in this way, it is clear from the previous sections that GS-ADMM and OS-ADMM are optimal algorithms for (ACO) with respect to gradient and operator evaluations required, respectively. Here, we argue that if properly chosen, either GS-ADMM or OS-ADMM will have optimal calls to both the gradient and operator oracle for any given problem of the form in (ACO).

Algorithm 4.5 Sliding Alternating Direction Method of Multipliers (S-ADMM)

if $\pi := \frac{||K||D_Y}{\sqrt{\varepsilon}\sqrt{L}} \geq 1$ **then**
  Apply GS-ADMM.
else
  Apply OS-ADMM.
end if

Theorem 4.3. Let S-ADMM in Algorithm 4.5 be applied to compute an ε-solution to (ACO). Then S-ADMM requires only $\mathcal{O}(\sqrt{LD_X^2/\varepsilon})$ gradient evaluations of f and $\mathcal{O}(||K||/\varepsilon)$ operator evaluations.

Proof. Fix some tolerance $\varepsilon > 0$ and suppose that the quantity $\pi := \frac{||K||D_Y}{\sqrt{\varepsilon}\sqrt{L}}$ is known. Whenever $\pi \geq 1$, S-ADMM applies GS-ADMM to minimize (ACO). Noting the results regarding the gradient and operator evaluation complexities for GS-ADMM from Corollary 4.3 and that

$$N_K := \sqrt{\frac{5LD_X^2}{\varepsilon}} + \frac{5||K||D_X D_Y}{\varepsilon} \leq \sqrt{\frac{5LD_X^2}{\varepsilon}} \cdot \frac{||K||D_Y}{\sqrt{\varepsilon}\sqrt{L}} + \frac{5||K||D_X D_Y}{\varepsilon} = \mathcal{O}\left(\frac{||K||D_X D_Y}{\varepsilon}\right)$$

since $\pi \geq 1$, we see that S-ADMM satisfies the conclusion of the theorem whenever $\pi \geq 1$. On the other hand, if $\pi \leq 1$, then S-ADMM applies OS-ADMM to solve (ACO). From our results in Corollary 4.6 and the assumption on π, we have

$$N_{\nabla f} := \sqrt{\frac{9LD_X^2}{\varepsilon}} + \frac{9||K||D_X D_Y}{\varepsilon} \leq \sqrt{\frac{9LD_X^2}{\varepsilon}} + \frac{9||K||D_X D_Y}{\varepsilon} \cdot \frac{\sqrt{\varepsilon}\sqrt{L}}{||K||D_Y} = \mathcal{O}\left(\sqrt{\frac{LD_X^2}{\varepsilon}}\right)$$

which satisfies the conclusion of the theorem. □

That is, by using π dictate sliding, S-ADMM is an optimal algorithm for solving (ACO) with respect to both gradient and operator evaluations required.

4.4 Dual Regularized Gradient Sliding

There has been an effort in recent literature to resolve the dependence on the maximum iteration count N. Specifically for ADMM type algorithms, [28] proposed a technique called dual regularization which is sufficient for eliminating the need for N in the choosing of inner iteration parameters. In this section, we will briefly show that dual regularization can be also be successful when applied to sliding ADMM procedures. We provide a proof of concept by presenting a practical version of GS-ADMM using dual regularization that does not require N in the parameter setting.

Consider an adaptation of L-ADMM presented in Algorithm 4.6 and let us compare the two procedures. We can immediately note the differences in the underlying minimization subproblems (4.1.4) and (4.1.5) in L-ADMM and its counterparts (4.4.1) and (4.4.2). In both pairs, we can see that an additional norm squared regularizer has been added to each minimization in the dual regularized algorithm. We can view these as penalties for moving too far away from our initial

Algorithm 4.6 Dual Regularized Linearized ADMM

Start: Choose $u_0 \in X$, $v_0 \in \mathbb{R}^m$, and $w_0 \in Z$.

for $t = 1, \ldots, T$ **do**

Compute

$$u_t = \underset{u \in X}{\operatorname{argmin}} \langle g, u \rangle + \tfrac{\beta}{2}\|u - u_0\|^2 + \langle K^\top(v_{t-1} + \theta_t(Ku_{t-1} - w_{t-1})) - \alpha_t(Ku_{t-1} - Ku_0), u \rangle + \tfrac{\eta_t}{2}\|u - u_{t-1}\|^2 + \tfrac{\xi_t}{2}\|u - u_0\|^2 \tag{4.4.1}$$

$$w_t = \underset{w \in Z}{\operatorname{argmin}} h(w) - \langle v_{t-1}, w \rangle + \tfrac{\tau_t}{2}\|w - Ku_t\|^2 + \tfrac{\zeta_t}{2}\|w - w_0\|^2 \tag{4.4.2}$$

$$v_t = \tfrac{1}{1+\rho_t\delta_t} v_{t-1} + \tfrac{\rho_t\delta_t}{1+\rho_t\delta_t} v_0 + \tfrac{\rho_t}{1+\rho_t\delta_t}(Ku_t - w_t) \tag{4.4.3}$$

end for

iterate. We also notice a difference in (4.1.6) and (4.4.3). This too can be explained through a regularizer. Indeed, note that

$$\begin{aligned}
&\underset{v \in \mathbb{R}^m}{\operatorname{argmin}} - \langle Ku_t - w_t, v \rangle + \tfrac{1}{2\rho_t}\|v - v_{t-1}\|^2 + \tfrac{\delta_t}{2}\|v - v_0\|^2 \\
&\quad = \tfrac{1/\rho_t}{(1/\rho_t)+\delta_t} v_{t-1} + \tfrac{\delta_t}{(1/\rho_t)+\delta_t} v_0 + \tfrac{1}{(1/\rho_t)+\delta_t}(Ku_t - w_t) \\
&\quad = \tfrac{1}{1+\rho_t\delta_t} v_{t-1} + \tfrac{\rho_t\delta_t}{1+\rho_t\delta_t} v_0 + \tfrac{\rho_t}{1+\rho_t\delta_t}(Ku_t - w_t) \\
&\quad = v_t.
\end{aligned}$$

That is, the computation of (4.4.3) can be viewed as solving the above optimization problem which has a regularization term with penalty δ_t. Whenever $\delta_t \equiv 0$, we recover (4.1.6). Thus, the only difference between Algorithms 4.1 and 4.6 is the presence of regularization terms on the minimization subproblems. However, despite only being small changes to the subproblem minimizations, these regularization terms allow us to choose alternative parameters for Algorithm 4.6 that do not require N. Furthermore, since these subproblem modifications are only the addition of regularization terms, solving (4.4.1) and (4.4.2) is no more difficult than solving (4.1.4) and (4.1.5) respectively. The benefit of the regularization terms can be seen in the convergence analysis. For the rest of the section, we will concern ourselves with the analysis of algorithms that are slight modifications of the ones already discussed in this section. Accordingly, much of the remaining analysis is similar to our previous discussion. Below we state a proposition regarding the convergence rate of Algorithm 4.6 that is very similar to that of Proposition 4.1.

Proposition 4.3. Suppose that Algorithm 4.6 is used to minimize (4.1.2). Then for all $t \geq 1$ and

$(u,w) \in \mathcal{H}$, we have $\Phi(\tilde{u}_T, \tilde{w}_T) - \Phi(u,w) \leq \Lambda_T(u,w)$ where

$$\tilde{u}_T := \tfrac{2}{T(T+1)} \sum_{t=1}^{T} t u_t \text{ and } \tilde{w}_T := \tfrac{2}{T(T+1)} \sum_{t=1}^{T} t w_t \tag{4.4.4}$$

are weighted averages of sequences $\{u_t\}_{t=1}^T$ and $\{w_t\}_{t=1}^T$, and

$$\begin{aligned}
\Lambda_T(u,w) := \tfrac{2}{T(T+1)} \sum_{t=1}^{T} t \Big[& \left(\tfrac{\eta_t}{2} ||u_{t-1} - u||^2 - \tfrac{\eta_t + \beta + \xi_t}{2} ||u_t - u||^2 \right) + \tfrac{\xi_t}{2} ||u_0 - u||^2 \\
& + \left(\tfrac{1}{2\rho_t} ||v_{t-1}||^2 - \left(\tfrac{1}{2\rho_t} + \tfrac{\delta_t}{2} \right) ||v_t||^2 \right) + \tfrac{\delta_t}{2} ||v_0||^2 \\
& + \left(\tfrac{\theta_t}{2} ||w_{t-1} - w||^2 - \tfrac{\tau_t + \zeta_t}{2} ||w_t - w||^2 \right) + \tfrac{\zeta_t}{2} ||w_0 - w||^2 \\
& + \left(\tfrac{\tau_t}{2} ||Ku_t - Ku||^2 - \tfrac{\theta_t - \alpha_t}{2} ||Ku_{t-1} - Ku||^2 \right) \\
& - \tfrac{1}{2} \left(\eta_t ||u_{t-1} - u_t||^2 - (\theta_t - \alpha_t) ||Ku_{t-1} - Ku_t||^2 \right) \\
& - \tfrac{\tau_t - \rho_t}{2} ||w_t - Ku_t||^2 - \tfrac{1}{2} (\xi_t - \alpha_t ||K||^2) ||u_0 - u_t||^2 \\
& - \tfrac{\alpha_t}{2} ||Ku_0 - Ku||^2 \Big].
\end{aligned} \tag{4.4.5}$$

Specifically, if the parameters in Algorithm 4.1 are set to

$$\tau_t = \theta_t \equiv \sigma, \rho_t \equiv \rho, \zeta_t = \alpha_t = \tfrac{\sigma}{t}, \xi_t = \tfrac{\sigma}{t} ||K||^2, \eta_t = \beta \left(\tfrac{t-1}{2} \right) + \sigma ||K||^2, \text{ and } \delta_t = \tfrac{1}{\rho t} \tag{4.4.6}$$

for some positive constants σ and ρ, then have for all $(u,w) \in \mathcal{H}$ that

$$\begin{aligned}
\Lambda_T(u,w) \leq & \tfrac{2}{T(T+1)} \Big[\left(\tfrac{\sigma}{2} ||K||^2 ||u_0 - u||^2 - \left(\tfrac{\beta T(T+1)}{4} + \tfrac{\sigma}{2} ||K||^2 (T+1) \right) ||u_T - u||^2 \right) \\
& + \left(\tfrac{1}{2\rho} ||v_0||^2 - \tfrac{T+1}{2\rho} ||v_T||^2 \right) + \left(\tfrac{\sigma}{2} ||w_0 - w||^2 - \tfrac{\sigma}{2} (T+1) ||w_T - w||^2 \right) \\
& + \left(\tfrac{T\sigma}{2} ||Ku_T - Ku||^2 - \tfrac{T\sigma}{2} ||Ku_0 - Ku||^2 \right) + \tfrac{T\sigma}{2} ||K||^2 ||u_0 - u||^2 + \tfrac{T}{2\rho} ||v_0||^2 \\
& + \tfrac{T\sigma}{2} ||w_0 - w||^2 \Big].
\end{aligned} \tag{4.4.7}$$

Proof. Fix any $(u,w) \in \mathcal{H}$ and note that from the definitions of $\tilde{u}_t$ and $\tilde{w}_t$ in (4.4.4) we have

$$\Phi(\tilde{u}_T, \tilde{w}_T) - \Phi(u,w) \leq \tfrac{2}{T(T+1)} \sum_{t=1}^{T} t \left[\Phi(u_t, w_t) - \Phi(u,w) \right]$$

by the convexity of Φ. In accordance with the above, we will need to estimate a bound on $\Phi(u_t, w_t) -$

$\Phi(u, w)$. From the optimality condition of (4.4.1), we have

$$\begin{aligned} 0 \geq &\langle K^T(v_{t-1} + \theta_t(Ku_{t-1} - w_{t-1}) - \alpha_t(Ku_{t-1} - Kx)) + \eta_t(u_t - u_{t-1}) + \xi_t(u_t - x), u_t - u\rangle \\ &+ \langle g, u_t - u\rangle + \beta\langle u_t - x, u_t - u\rangle \end{aligned}$$

for any $u \in X$. Rearranging, this becomes

$$\begin{aligned} \langle g, u_t - u\rangle + \beta\langle u_t - x, u_t - u\rangle \leq &- \langle K^T(v_{t-1} + \theta_t(Ku_{t-1} - w_{t-1}) - \alpha_t(Ku_{t-1} - Kx)), u_t - u\rangle \\ &- \eta_t\langle u_t - u_{t-1}, u_t - u\rangle - \xi_t\langle u_t - x, u_t - u\rangle. \end{aligned}$$

Expanding the inner products and rearranging again, we obtain

$$\begin{aligned} &\langle g, u_t - u\rangle + \tfrac{\beta}{2}\left(||u_t - x||^2 - ||u - x||^2\right) \\ &\quad\leq - \langle K^T(v_{t-1} + \theta_t(Ku_{t-1} - w_{t-1}) - \alpha_t(Ku_{t-1} - Kx)), u_t - u\rangle \\ &\quad\quad + \tfrac{\eta_t}{2}\left(||u - u_{t-1}||^2 - ||u_t - u_{t-1}||^2 - ||u_t - u||^2\right) \\ &\quad\quad + \tfrac{\xi_t}{2}\left(||u - x||^2 - ||u_t - x||^2 - ||u_t - u||^2\right) - \tfrac{\beta}{2}||u_t - u||^2. \end{aligned} \tag{4.4.8}$$

Similarly, by the optimality condition of (4.4.2), we can show that

$$\begin{aligned} h(w_t) - h(w) \leq &\langle v_{t-1}, w_t - w\rangle + \tfrac{\tau_t}{2}\left(||w - Ku_t||^2 - ||w_t - Ku_t||^2 - ||w - w_t||^2\right) \\ &+ \tfrac{\zeta_t}{2}\left(||w - z||^2 - ||w_t - z||^2 - ||w - w_t||^2\right) \end{aligned} \tag{4.4.9}$$

for all $w \in Z$. It follows that

$$\begin{aligned} \Phi(u_t, w_t) - \Phi(u, w) = &\langle g, u_t - u\rangle + h(w_t) - h(w) + \tfrac{\beta}{2}\left(||u - x||^2 - ||u_t - x||^2\right) \\ \leq &\alpha_t\langle Ku_{t-1} - Kx, Ku_t - Ku\rangle - \theta_t\langle Ku_{t-1} - w_{t-1}, Ku_t - Ku\rangle \\ &+ \tfrac{\eta_t}{2}\left(||u - u_{t-1}||^2 - ||u_t - u_{t-1}||^2 - ||u_t - u||^2\right) \\ &+ \tfrac{\xi_t}{2}\left(||u - x||^2 - ||u_t - x||^2 - ||u_t - u||^2\right) \\ &+ \tfrac{\tau_t}{2}\left(||w - Ku_t||^2 - ||w_t - Ku_t||^2 - ||w - w_t||^2\right) \\ &+ \tfrac{\zeta_t}{2}\left(||w - z||^2 - ||w_t - z||^2 - ||w - w_t||^2\right) \\ &+ \langle v_{t-1}, w_t - w\rangle - \langle v_{t-1}, Ku_t - Ku\rangle - \tfrac{\beta}{2}||u_t - u||^2 \end{aligned} \tag{4.4.10}$$

by combining the relations (4.4.8) and (4.4.9). We can further rewrite the first two lines of (4.4.10). By the fact that $(u, w) \in \mathcal{H}$ and (4.4.3), we can write

$$\begin{aligned}\langle v_{t-1}, w_t - w\rangle - \langle v_{t-1}, Ku_t - Ku\rangle &= \tfrac{\rho_t}{2}\left|\left|w_t - Ku_t\right|\right|^2 + \tfrac{1}{2\rho_t}\left|\left|v_{t-1}\right|\right|^2 - \tfrac{1}{2\rho_t}\left|\left|v_{t-1} - \rho_t(w_t - Ku_t)\right|\right|^2 \\ &= \tfrac{\rho_t}{2}\left|\left|w_t - Ku_t\right|\right|^2 + \tfrac{1}{2\rho_t}\left|\left|v_{t-1}\right|\right|^2 - \tfrac{1}{2\rho_t}\left|\left|v_t + \rho_t\delta_t(v_t - y)\right|\right|^2 \\ &\leq \tfrac{\rho_t}{2}\left|\left|w_t - Ku_t\right|\right|^2 + \tfrac{1}{2\rho_t}\left|\left|v_{t-1}\right|\right|^2 - \left(\tfrac{1}{2\rho_t} + \tfrac{\delta_t}{2}\right)\left|\left|v_t\right|\right|^2 + \tfrac{\delta_t}{2}y.\end{aligned}$$

Furthermore, we can expand inner products to generate the following bound

$$\begin{aligned}\alpha_t\langle Ku_{t-1} - Kx, Ku_t - Ku\rangle - \theta_t\langle Ku_{t-1} - w_{t-1}, Ku_t - Ku\rangle& \\ \leq \tfrac{\alpha_t - \theta_t}{2}\left(\left|\left|Ku_{t-1} - Ku\right|\right|^2 - \left|\left|Ku_{t-1} - Ku_t\right|\right|^2\right)& \\ + \tfrac{\alpha_t}{2}\left|\left|K\right|\right|^2\left|\left|x - u_t\right|\right|^2 + \tfrac{\theta_t}{2}\left|\left|w_{t-1} - w\right|\right|^2 - \tfrac{\alpha_t}{2}\left|\left|Kx - Ku\right|\right|^2.&\end{aligned}$$

We may then apply the above two relations to (4.4.10) and the use the definition of $\Phi(u, w)$ in (4.1.2) to obtain

$$\begin{aligned}\Phi(u_t, w_t) - \Phi(u, w) =& \langle g, u_t - u\rangle + h(w_t) - h(w) + \tfrac{\beta}{2}\left(\left|\left|u - x\right|\right|^2 - \left|\left|u_t - x\right|\right|^2\right) \\ \leq& \tfrac{\alpha_t - \theta_t}{2}\left(\left|\left|Ku_{t-1} - Ku\right|\right|^2 - \left|\left|Ku_{t-1} - Ku_t\right|\right|^2\right) \\ &+ \tfrac{\alpha_t}{2}\left|\left|K\right|\right|^2\left|\left|u_0 - u_t\right|\right|^2 + \tfrac{\theta_t}{2}\left|\left|w_{t-1} - w\right|\right|^2 - \tfrac{\alpha_t}{2}\left|\left|Ku_0 - Ku\right|\right|^2 \\ &+ \tfrac{\eta_t}{2}\left(\left|\left|u - u_{t-1}\right|\right|^2 - \left|\left|u_t - u_{t-1}\right|\right|^2 - \left|\left|u_t - u\right|\right|^2\right) \\ &+ \tfrac{\xi_t}{2}\left(\left|\left|u - u_0\right|\right|^2 - \left|\left|u_t - u_0\right|\right|^2 - \left|\left|u_t - u\right|\right|^2\right) \\ &+ \tfrac{\tau_t}{2}\left(\left|\left|w - Ku_t\right|\right|^2 - \left|\left|w_t - Ku_t\right|\right|^2 - \left|\left|w - w_t\right|\right|^2\right) \\ &+ \tfrac{\zeta_t}{2}\left(\left|\left|w - w_0\right|\right|^2 - \left|\left|w_t - w_0\right|\right|^2 - \left|\left|w - w_t\right|\right|^2\right) \\ &+ \tfrac{\rho_t}{2}\left|\left|w_t - Ku_t\right|\right|^2 + \tfrac{1}{2\rho_t}\left|\left|v_{t-1}\right|\right|^2 - \left(\tfrac{1}{2\rho_t} + \tfrac{\delta_t}{2}\right)\left|\left|v_t\right|\right|^2 + \tfrac{\delta_t}{2}\left|\left|v_0\right|\right|^2 \\ &- \tfrac{\beta}{2}\left|\left|u_t - u\right|\right|^2,\end{aligned}$$

which immediately implies the conclusion in (4.4.5) by combining the similar terms. To show (4.4.7) and finish the proof, let the parameters of Algorithm 4.6 be chosen according to (4.4.6) and evaluate the telescoping sums. In particular, note that since $\xi_t \geq \alpha_t \left|\left|K\right|\right|^2$, $\theta_t \geq \alpha_t, \tau_t \geq \rho_t$, and $\eta_t \geq (\theta_t - \alpha_t)\left|\left|K\right|\right|^2$ for all $t \geq 1$, the relation (4.4.7) follows immediately. □

Let us now compare Proposition 4.3 to Proposition 4.1 and its corollary, Corollary 4.1, as there are only a few key differences. First, we note that the outputs $\tilde{u}_T$ and $\tilde{w}_T$ are both convex combinations of previously computed iterates, but differ slightly in their weights. Second, by enforcing $\xi_t = \delta_t = \zeta_t = \alpha_t \equiv 0$, we see that the sum of (4.4.5) is a slightly rearranged and scaled equivalent of (4.1.9). Lastly, we observe that the parameters in (4.4.6) are not constant with respect to t. In particular, note how (4.1.9) and (4.4.5) depend on ρ_t and θ_t. In order for the sums in (4.1.9) to telescope nicely, we are required to choose ρ_t and θ_t that are constant with respect to the sum. This restriction is not prevalent in (4.4.5) because the additional regularization parameters $\eta_t, \delta_t, \zeta_t$, and α_t can be chosen to better facilitate telescoping.

Similar to the vanilla version of GS-ADMM in Algorithm 4.2, now that we have a method of solving (4.1.2), we now present Algorithm 4.7: a dual regularized version of GS-ADMM which has the same theoretical properties of Algorithm 4.2, while also having the additional benefit of not requiring N a priori. To avoid an overly complicated naming system, we refer to both Algorithms 4.2 and 4.7 as GS-ADMM and will specifically list equation numbers when referencing dual regularization.

Algorithm 4.7 Dual Regularized GS-ADMM

Start: Choose $x_0 \in X$ and set $\overline{x}_0 := x_0$. Set $y_0 := 0$ and $z_0 := Kx_0$.
for $k = 1, \ldots, N$ **do**
Compute

$$\begin{aligned} \underline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1} \\ (\tilde{x}_k, \tilde{z}_k, x_k, y_k, z_k,) =& \text{ApproxDRGS}(\nabla f(\underline{x}_k), x_{k-1}, y_{k-1}, z_{k-1}, \beta_k, T_k) \qquad (4.4.11) \\ \overline{x}_k =& (1-\gamma_k)\overline{x}_{k-1} + \gamma_k \tilde{x}_k \qquad (4.4.12) \\ \overline{z}_k =& (1-\gamma_k)\overline{z}_{k-1} + \gamma_k \tilde{z}_k \qquad (4.4.13) \end{aligned}$$

end for
Output $\overline{x}_N$.
procedure $(\tilde{x}^+, \tilde{z}^+, x^+, y^+, z^+) =$ ApproxDRGS(g, x, y, z, β, T)
Apply Algorithm 4.6 to solve problem (4.1.2) with initial values $u_0 = x$, $v_0 = y$, and $w_0 = z$. Obtain the iterates u_T, v_T, and w_T of the algorithm and weighted sums $\tilde{w}_T$ and $\tilde{u}_T$ (defined in (4.4.4)) after T iterations.
Output $\tilde{z}^+ := \tilde{w}_T$, $\tilde{x}^+ := \tilde{u}_T$, $x^+ := u_T$, $y^+ := v_T$, and $z^+ := w_T$.
end procedure

We can see now that Proposition 4.3 and the call to ApproxDRGS in (4.4.11) implies that

for properly chosen parameters we have

$$\left[\langle \nabla f(\underline{x}_k), \tilde{x}_k \rangle + h(\tilde{z}_k) + \tfrac{\beta_k}{2}\|\tilde{x}_k - x_{k-1}\|^2\right] - \left[\langle \nabla f(\underline{x}_k), u \rangle + h(w) + \tfrac{\beta_k}{2}\|u - x_{k-1}\|^2\right] \leq \Lambda_{T_k}(u, w) \tag{4.4.14}$$

for all $(u, w) \in \mathcal{H}$ which is a similar result to (4.1.17) in the vanilla GS-ADMM case. This allows us to prove convergence results on Algorithm 4.7. In particular, Proposition 4.3 and its surrounding discussion shows that we can utilize the regularization parameters to choose more flexible parameters in Algorithm 4.6. The following theorem shows that we can also make a better choice of T_k, and as a result, nearly be free of our dependency on N.

Theorem 4.4. Suppose that the parameters in the k-th call to the ApproxDRGS procedure in Algorithm 4.6 are chosen according to (4.4.6) as

$$\tau_t = \theta_t \equiv \sigma, \rho_t \equiv \rho, \zeta_t = \alpha_t = \tfrac{\sigma}{t}, \xi_t = \tfrac{\sigma}{t}\left|\left|K\right|\right|^2, \eta_t = \beta\left(\tfrac{t-1}{2}\right) + \sigma\left|\left|K\right|\right|^2, \text{ and } \delta_t = \tfrac{1}{\rho t}$$

with

$$\sigma := \chi\tfrac{\lceil \mu k \rceil}{\mu k}, \rho := \chi\tfrac{\mu k}{\lceil \mu k \rceil} \tag{4.4.15}$$

depending on some constant $\chi > 0$, and that γ_k, β_k, and T_k in Algorithm 4.7 are set as

$$\beta_k = \tfrac{2L}{k}, \gamma_k = \tfrac{2}{k+1}, T_k = \lceil \mu k \rceil \tag{4.4.16}$$

for some $\mu > 0$. Here, $\lceil a \rceil$ denotes the standard ceiling function which returns the smallest integer larger than a. Then by the definition of D_X in (1.0.1)

$$F(\overline{x}_N, \overline{z}_N) - F(u, w) \leq \left(\tfrac{2L}{N(N+1)} + \tfrac{3\chi||K||^2}{N(N+1)\mu}\right) D_X^2 - \tfrac{2}{\chi N(N+1)\mu}\left|\left|y_N\right|\right|^2 \tag{4.4.17}$$

for any $(u, w) \in \mathcal{H}$ and $N \geq 1$.

Proof. By the strong convexity and convexity of f and h respectively along with the definition of F

in (ACO), we have

$$\begin{aligned}
F(\overline{x}_k,\overline{z}_k)&-(1-\gamma_k)F(\overline{x}_{k-1},\overline{z}_{k-1})-\gamma_k F(u,w)\\
&\leq\gamma_k\left(\langle\nabla f(x_k),\tilde{x}_k-u\rangle+h(\tilde{z}_k)-h(w)+\tfrac{L\gamma_k}{2}\left\|\tilde{x}_k-x_{k-1}\right\|^2\right)\\
&\leq\gamma_k\left(\langle\nabla f(x_k),\tilde{x}_k-u\rangle+h(\tilde{z}_k)-h(w)+\tfrac{\beta_k}{2}\left\|\tilde{x}_k-x_{k-1}\right\|^2\right)\\
&\quad+\tfrac{\beta_k\gamma_k}{2}\left\|x_{k-1}-u\right\|^2-\tfrac{\beta_k\gamma_k}{2}\left\|x_{k-1}-u\right\|^2\\
&=\gamma_k\left[\langle\nabla f(\underline{x}_k),\tilde{x}_k\rangle+h(\tilde{z}_k)+\tfrac{\beta_k}{2}\left\|\tilde{x}_k-x_{k-1}\right\|^2\right.\\
&\quad\left.-\left(\langle\nabla f(\underline{x}_k),u\rangle+h(w)+\tfrac{\beta_k}{2}\left\|x_{k-1}-u\right\|^2\right)\right]+\tfrac{\beta_k\gamma_k}{2}\left\|x_{k-1}-u\right\|^2
\end{aligned}$$

where we used the fact that $\beta_k \geq L\gamma_k$ for any $k \geq 1$. Noting both the inputs and outputs of the procedure call of ApproxDRGS in (4.4.11) and recalling the discussion surrounding (4.4.14), we may apply Proposition 4.3 to the above and continue with

$$\begin{aligned}
F(\overline{x}_k,\overline{z}_k)&-(1-\gamma_k)F(\overline{x}_{k-1},\overline{z}_{k-1})-\gamma_k F(u,w)\\
&\leq\tfrac{2}{T_k(T_k+1)}\left[\tfrac{\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|K\right\|^2\left(\left\|x_{k-1}-u\right\|^2-(T_k+1)\left\|x_k-u\right\|^2\right)\right.\\
&\quad-\tfrac{\beta_k T_k(T_k+1)}{4}\left\|x_k-u\right\|^2+\tfrac{T_k\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|K\right\|^2\left\|x_{k-1}-u\right\|^2\\
&\quad+\tfrac{T_k+1}{2\chi}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|y_{k-1}\right\|^2-\tfrac{T_k+1}{2\chi}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|y_k\right\|^2\\
&\quad+\tfrac{(T_k+1)\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|z_{k-1}-w\right\|^2-\tfrac{(T_k+1)\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|z_k-w\right\|^2\\
&\quad\left.+\tfrac{T_k\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|Kx_k-Ku\right\|^2-\tfrac{T_k\chi}{2}\tfrac{\lceil\mu k\rceil}{\mu k}\left\|Kx_{k-1}-Ku\right\|^2\right]\\
&\quad+\tfrac{\beta_k\gamma_k}{2}\left\|x_{k-1}-u\right\|^2\\
&\leq\tfrac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1}-u\right\|^2-\left\|x_k-u\right\|^2\right)\\
&\quad+\tfrac{\lceil\mu k\rceil}{\mu k}\left[\left(\tfrac{\chi\gamma_k}{T_k}\left\|K\right\|^2\left\|x_{k-1}-u\right\|^2-\tfrac{\chi\gamma_k}{T_k}\left\|K\right\|^2\left\|x_k-u\right\|^2\right)\right.\\
&\quad+\tfrac{\gamma_k}{\chi T_k}\left(\left\|y_{k-1}\right\|^2-\left\|y_k\right\|^2\right)+\tfrac{\chi\gamma_k}{T_k}\left(\left\|z_{k-1}-w\right\|^2-\left\|z_k-w\right\|^2\right)\\
&\quad\left.+\tfrac{\gamma_k\chi}{T_k+1}\left(\left\|Kx_k-Ku\right\|^2-\left\|Kx_{k-1}-Ku\right\|^2\right)\right].
\end{aligned}$$

Rearranging and applying the parameters according to (4.4.16), this becomes

$$\begin{aligned}
&F(\overline{x}_k,\overline{z}_k)-(1-\gamma_k)F(\overline{x}_{k-1},\overline{z}_{k-1})-\gamma_k F(u,w)\\
&\quad\le\left(\tfrac{\beta_k\gamma_k}{2}+\tfrac{\gamma_k\chi\lceil\mu k\rceil||K||^2}{(\mu k)T_k}\right)\left(||x_{k-1}-u||^2-||x_k-u||^2\right)\\
&\qquad+\tfrac{\lceil\mu k\rceil}{\mu k}\left[\tfrac{\gamma_k}{\chi T_k}\left(||y_{k-1}||^2-||y_k||^2\right)\right.\\
&\qquad+\tfrac{\gamma_k\chi}{T_k}\left(||z_{k-1}-w||^2-||z_k-w||^2\right)\\
&\qquad\left.+\tfrac{\gamma_k\chi}{T_k+1}\left(||Kx_k-Ku||^2-||Kx_{k-1}-Ku||^2\right)\right] \qquad (4.4.18)\\
&\quad\le\tfrac{2L\mu+2\chi||K||^2}{k(k+1)\mu}\left(||x_{k-1}-u||^2-||x_k-u||^2\right)\\
&\qquad+\tfrac{2}{\chi k(k+1)\mu}\left(||y_{k-1}||^2-||y_k||^2\right)\\
&\qquad+\tfrac{2\chi}{k(k+1)\mu}\left(||z_{k-1}-w||^2-||z_k-w||^2\right)\\
&\qquad+\tfrac{2\chi\lceil\mu k\rceil}{k(k+1)(\lceil\mu k\rceil+1)\mu}\left(||Kx_k-Ku||^2-||Kx_{k-1}-Ku||^2\right).
\end{aligned}$$

Let us now multiply (4.4.18) by $k(k+1)\mu$ and sum from $k=1,\dots,N$. Noting the telescoping sum, the initializations of $y_0=0$ and $z_0=Kx_0$, and the fact that $(u,w)\in H$, this becomes

$$\begin{aligned}
&N(N+1)\mu\left(F\overline{x}_N,\overline{z}_N\right)-F(u,w))\\
&\quad\le\left(2L\mu+2\chi\,||K||^2\right)\left(||x_0-u||^2-||x_N-u||^2\right)+2\chi\,||Kx_0-Ku||^2\\
&\qquad+2\chi\textstyle\sum_{k=1}^N\left(\frac{\lceil\mu k\rceil}{\lceil\mu k\rceil+1}\right)\left(||Kx_k-Ku||^2-||Kx_{k-1}-Ku||^2\right)-\frac{2}{\chi}\,||y_N||^2\\
&\quad\le\left(2L\mu+2\chi\,||K||^2\right)\left(||x_0-u||^2-||x_N-u||^2\right)+2\chi\,||Kx_0-Ku||^2\\
&\qquad+2\chi\textstyle\sum_{k=1}^N\left(1-\frac{1}{\lceil\mu k\rceil+1}\right)\left(||Kx_k-Ku||^2-||Kx_{k-1}-Ku||^2\right)-\frac{2}{\chi}\,||y_N||^2\\
&\quad\le\left(2L\mu+2\chi\,||K||^2\right)\left(||x_0-u||^2-||x_N-u||^2\right)+2\chi\,||Kx_0-Ku||^2\\
&\qquad+2\chi\left(||Kx_N-Ku||^2-||Kx_0-Ku||\right)-\frac{2}{\chi}\,||y_N||^2\\
&\qquad+2\chi\textstyle\sum_{k=1}^N\frac{1}{\lceil\mu k\rceil+1}\left(||Kx_{k-1}-Ku||^2-||Kx_k-Ku||^2\right)
\end{aligned}$$

where in the last inequality, we split the sum into two. Using Cauchy-Schwarz, combining like terms,

and resolving the final summation, we conclude that

$$\begin{aligned}
&N(N+1)\mu\left(F\overline{x}_N, \overline{z}_N\right) - F(u,w)) \\
&\quad \leq \left(2L\mu + 2\chi\left|\left|K\right|\right|^2\right)\left(\left|\left|x_0 - u\right|\right|^2 - \left|\left|x_N - u\right|\right|^2\right) - \tfrac{2}{\chi}\left|\left|y_N\right|\right|^2 + 2\chi\left|\left|Kx_0 - Ku\right|\right|^2 \\
&\qquad + 2\chi\left(\left|\left|Kx_N - Ku\right|\right|^2 - \left|\left|Kx_0 - Ku\right|\right|\right) + \tfrac{2\chi}{\lceil\mu\rceil+1}\left|\left|Kx_0 - Ku\right|\right|^2 \\
&\quad \leq \left(2L\mu + 2\chi\left|\left|K\right|\right|^2\right)\left(\left|\left|x_0 - u\right|\right|^2 - \left|\left|x_N - u\right|\right|^2\right) - \tfrac{2}{\chi}\left|\left|y_N\right|\right|^2 \\
&\qquad + 2\chi\left|\left|K\right|\right|^2\left|\left|x_N - u\right|\right|^2 + \chi\left|\left|K\right|\right|^2\left|\left|x_0 - u\right|\right|^2 \\
&\quad \leq (2L\mu + 3\chi\left|\left|K\right|\right|^2)\left|\left|x_0 - u\right|\right|^2 - 2L\mu\left|\left|x_N - u\right|\right|^2 - \tfrac{2}{\chi}\left|\left|y_N\right|\right|^2 \\
&\quad \leq (2L\mu + 3\chi\left|\left|K\right|\right|^2)\left|\left|x_0 - u\right|\right|^2 - \tfrac{2}{\chi}\left|\left|y_N\right|\right|^2
\end{aligned}$$

which reduces to (4.4.17) after dividing by $N(N+1)\mu$ and upper bounding it using the definition of D_X in (1.0.1).

□

Comparing Theorem 4.4 with Theorem 4.1 and Corollary 4.2, we see that the dual regularized version does not require the usage of N outside of the constants χ and μ which we have not yet specified. That is, if χ and μ are chosen to be free of N, then so will the entirety of Algorithm 4.7. We answer this question of whether or not these parameters can be chosen in this way as well as conclude our final GS-ADMM result through Corollary 4.7 below.

Corollary 4.7. Suppose that $Y := \operatorname{dom} h^*$ is compact, and that the parameters in Algorithm 4.2 are set as in Theorem 4.4. Then we have

$$f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq \left(\tfrac{2L}{N(N+1)} + \tfrac{3\chi||K||^2}{N(N+1)\mu}\right)D_X^2 + \tfrac{2D_Y^2}{\chi N(N+1)\mu}, \tag{4.4.19}$$

where D_X and D_Y are upper estimates of the distance from x_0 to the set of optimal solutions to problem (ACO) and the largest norm $\sup_{y\in Y}\|y\|$ among elements in Y, respectively. Specifically in the parameter settings in Theorem 4.4, if we set

$$\chi := \tfrac{D_Y}{||K||D_X}, \text{ and } \mu := \tfrac{||K||D_Y}{LD_X} \tag{4.4.20}$$

then the number of gradient evaluations of ∇f and operator evaluations (involving K and $K^\top$) are

bounded by $N_{\nabla f} := \sqrt{\frac{7LD_X^2}{\varepsilon}}$ and $N_K := \frac{7\|K\|D_X D_Y}{\varepsilon} + \sqrt{\frac{7LD_X^2}{\varepsilon}}$ respectively.

Proof. Observe that for any $v \in Y$ such that v is a subgradient of h at $K\overline{x}_k$, we have

$$h(K\overline{x}_k) \leq h(\overline{z}_k) + \langle v, K\overline{x}_k - \overline{z}_k \rangle$$

and consequently

$$\begin{aligned} f(\overline{x}_k) + h(K\overline{x}_k) - F^* \leq & f(\overline{x}_k) + h(\overline{z}_k) + \langle v, K\overline{x}_k - \overline{z}_k \rangle - F^* \\ = & F(\overline{x}_k, \overline{z}_k) - F^* + \langle v, K\overline{x}_k - \overline{z}_k \rangle. \end{aligned} \tag{4.4.21}$$

Thus, in view of Theorem 4.4, to obtain a result of the form in (4.4.19), it suffices to bound $\langle v, K\overline{x}_k - \overline{z}_k \rangle$. Note first that by the definition of (4.4.3), the setting of ρ_t and δ_t in (4.4.6), and the definitions of $\tilde{u}_t$ and $\tilde{w}_t$ in (4.4.4) that

$$\begin{aligned} K\tilde{u}_T - \tilde{w}_T &= \tfrac{2}{T(T+1)} \textstyle\sum_{t=1}^{T} t(Ku_t - w_t) \\ &= \tfrac{2}{\rho T(T+1)} \textstyle\sum_{t=1}^{T} \left[((t+1)v_t - tv_{t-1}) - v_0\right] \\ &= \tfrac{2}{\rho T(T+1)} \left[(T+1)v_T - v_0 - Tv_0\right]. \end{aligned}$$

Now consider the k-th call to ApproxDRGS in (4.4.11). By the outputs of ApproxDRGS, the above equality, and the setting of $T_k = k$ in (4.4.16), it follows that

$$\begin{aligned} K\tilde{x}_k - \tilde{z}_k &= \tfrac{2}{\rho\lceil\mu k\rceil(\lceil\mu k\rceil+1)} \left[(\lceil\mu k\rceil + 1)y_k - y_{k-1} - \lceil\mu k\rceil y_{k-1}\right] \\ &= \tfrac{2}{\rho\lceil\mu k\rceil} \left(y_k - y_{k-1}\right) \end{aligned}$$

where recall that v_0 is initialized to y_{k-1}. Using this equality, we can show that

$$\begin{aligned} K\overline{x}_k - \overline{z}_k &= K((1-\gamma_k)\overline{x}_{k-1} + \gamma_k\tilde{x}_k) - ((1-\gamma_k)\overline{z}_{k-1} + \gamma_k\tilde{z}_k) \\ &= \tfrac{k-1}{k+1} \left(K\overline{x}_{k-1} - \overline{z}_{k-1}\right) + \tfrac{2}{k+1} \left(K\tilde{x}_k - \tilde{z}_k\right) \\ &= \tfrac{k-1}{k+1} \left(K\overline{x}_{k-1} - \overline{z}_{k-1}\right) + \tfrac{4}{\rho(k+1)\lceil\mu k\rceil} \left(y_k - y_{k-1}\right) \\ &= \tfrac{k-1}{k+1} \left(K\overline{x}_{k-1} - \overline{z}_{k-1}\right) + \tfrac{4}{\chi N(N+1)\mu} (y_k - y_{k-1}) \end{aligned}$$

where we used the definitions of $\overline{x}_k$ and $\overline{z}_k$ in (4.4.12) and (4.4.13) respectively, the setting of $\gamma_k =$

$2/(k+1)$ in (4.4.16) to obtain the first equality, and the definition of ρ in (4.4.15). Multiplying the above, summing over $k = 1, \ldots, N$, and rearranging, we have that $K\overline{x}_N - \overline{z}_N = \frac{4}{\chi N(N+1)\mu}(y_N - y_0)$, which directly implies that

$$\begin{aligned}\langle v, K\overline{x}_N - \overline{z}_N\rangle &= \tfrac{4}{\chi N(N+1)\mu}\langle v, y_N - y_0\rangle \\ &= \tfrac{2}{\chi N(N+1)\mu}\left(||y_N||^2 - ||y_N - v||^2 - ||y_0||^2 + ||y_0 - v||^2\right) \\ &\leq \tfrac{2}{\chi N(N+1)\mu}\left(||y_N||^2 + ||v||^2\right)\end{aligned} \tag{4.4.22}$$

since we initialize $y_0 = 0$. We may then combine (4.4.22), (4.4.21), and Theorem 4.4 to show that

$$f(\overline{x}_N) + h(K\overline{x}_N) - F^* \leq F(\overline{x}_N, \overline{z}_N) - F^* + \langle v, K\overline{x}_k - \overline{z}_k\rangle \leq \left(\tfrac{2L\mu + 3\chi||K||^2}{N(N+1)\mu}\right) D_X^2 + \tfrac{2D_Y^2}{\chi N(N+1)\mu}$$

which proves (4.4.19). Applying our choice of χ and μ from (4.4.20), we conclude that

$$f(\overline{x}_N) + h(K\overline{x}_N) - F^* \leq \tfrac{7LD_X^2}{N^2}.$$

Consequently, in order obtain an ε solution, we need $N \geq N_{\nabla f} := \sqrt{\frac{7LD_X^2}{\varepsilon}}$ iterations of Algorithm 4.7. Since each iteration performs exactly one gradient evaluation, we therefore need at least $N_{\nabla f}$ gradient evaluations for an ε solution.

To count the number of operator evaluations, we note that each iteration of Algorithm 4.7 performs one call to Algorithm 4.6 with T_k inner iterations. Since each inner iteration of Algorithm 4.6 requires one operator evaluation, we conclude that the total number of operator evaluations required for an ε solution using Algorithm 4.7 is bounded above by

$$N_K := \textstyle\sum_{k=1}^{N_{\nabla f}} T_k \leq \mu \sum_{k=1}^{N_{\nabla f}} k + N_{\nabla f} \leq \mu N_{\nabla f}^2 + N_{\nabla f} = \frac{7||K||D_X D_Y}{\varepsilon} + \sqrt{\frac{7LD_X^2}{\varepsilon}}.$$

□

We conclude this section by noting that Corollary 4.7 provides the same theoretical convergence rate as its vanilla GS-ADMM counterpart in Corollary 4.3, and is also able to eliminate the knowledge of N in the parameter setting using additional regularization terms and novel parameter choices.

4.5 Numerical Experiments

Our goal now is to present preliminary results from our numerical experiments. We will be comparing the performance of our sliding algorithms with two algorithms, Nesterov Smoothing (NEST-S) and A-ADMM. A-ADMM we discussed previously and NEST-S is a NAGD variant that is able to solve (ACO) with the same iteration complexities as A-ADMM (see [29] for more details). Let us briefly consider an example that highlights the limitations of the single oracle approach. Consider the following instance of convex problem (2.3.1) with $f(x) := \langle Qx, x\rangle/2 - \langle q, x\rangle$ and $h(z) := \|z - b\|$:

$$F^* := \min_{x\in\mathbb{R}^n} \tfrac{1}{2}\langle Qx, x\rangle - \langle q, x\rangle + \|Kx - b\|. \tag{4.5.1}$$

Here $n := 2k$,

$$Q := \tfrac{L}{4}U^\top SU,\ q := \tfrac{L}{2}U^\top e_{2k},\ K := \tfrac{RM}{2}V^\top BU,\ b := \tfrac{RM}{2}V^\top e,\ R := \tfrac{2\sqrt{2k+3}L}{M}$$

$$S := \begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix},\ B := \begin{bmatrix} & & & & -1 & 1 \\ & & & \iddots & \iddots & \\ & & -1 & 1 & & \\ & -1 & 1 & & & \\ 1 & & & & & \end{bmatrix}, \tag{4.5.2}$$

e is a $2k$-dimensional vector of 1's, and U and V are orthogonal matrices such that $VKU^\top$ is the singular value decomposition of B. Observe that $\|B\| \le 2$ since for any $x \in \mathbb{R}^{2k}$ we have

$$\begin{aligned}\|Bx\|^2 =& \left(x^{(2k)} - x^{(2k-1)}\right)^2 + \cdots + \left(x^{(2)} - x^{(1)}\right)^2 + \left(x^{(1)}\right)^2 \\ \le& 2\left(\left(x^{(2k)}\right)^2 + \left(x^{(2k-1)}\right)^2\right) + \cdots + 2\left(\left(x^{(2)}\right)^2 + \left(x^{(1)}\right)^2\right) + \left(x^{(1)}\right)^2 \le 4\|x\|^2.\end{aligned}$$

Also, note that B is a nonsingular symmetric matrix with positive and negative eigenvalues, so the columns of V and U are both eigenvectors of B, and may only differ by their signs. K is a diagonal matrix by the singular value decomposition construction, with

$$\|K\| = \tfrac{RM}{2}\|B\| \le RM = 2\sqrt{2k+3}L.$$

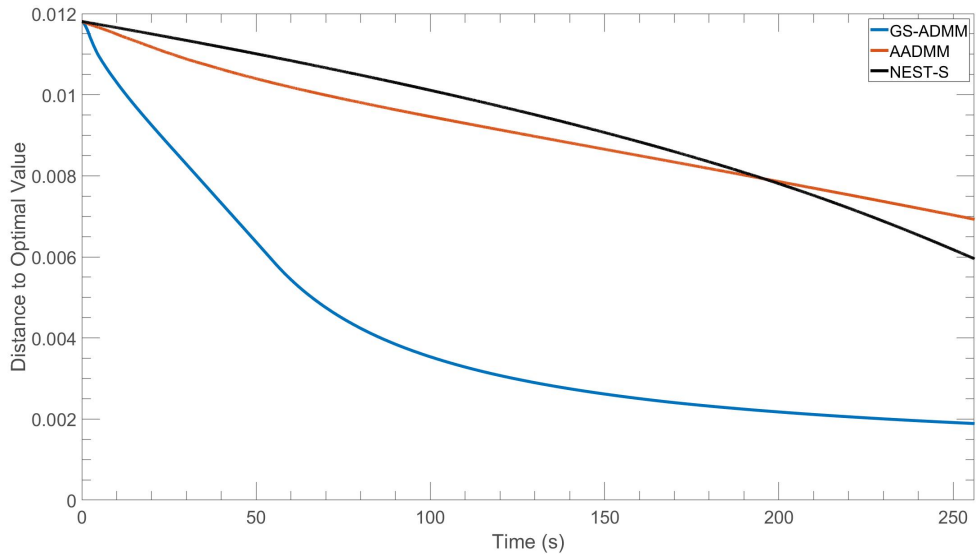

Figure 4.1: A comparison of GS-ADMM and existing algorithms.

The problem (4.5.1) is a modified version of the worst-case instance described in Theorem 4.1 of [18]. The difference is that in the problem instance there S is set to BB and U and V are identity matrix. In our setting of S and B in (4.5.2), S and BB differ only by the top-left entry (the value is 1 for S and 2 for BB). It is also worth pointing out that the matrix S is also closely related to a block in the matrix A_k in Chapter 2.1.2 of [2] for worst-case smooth convex optimization instance. However, unlike the matrix in [2] the top-left and bottom-right entries of S are 1 instead of 2. Accordingly, since the problem instance (4.5.1) is so similar to the work in [18] which provided an example of a worst-case problem instance, we might expect A-ADMM and NEST-S to have near worst-case behavior.

Furthermore, since Q is dense and K is sparse, the gradient evaluations of f will dominate the running time of these implementations. Thus, if the required gradient evaluations for solving (4.5.1) are high, a gradient sliding modification would be an effective improvement. Indeed, as evidenced in Figure 4.1, we see that the GS-ADMM algorithm outperforms A-ADMM and NEST-S for the problem instance in (4.5.1). The same experiment can be designed to showcase a need for operator sliding as well. Similarly, through a change of variable, one can modify the problem formulation described in (4.5.1) to obtain a worst-case instance of the same form, but with a sparse Q and a dense K. Applying OS-ADMM to the problem will then require fewer operator evaluations,

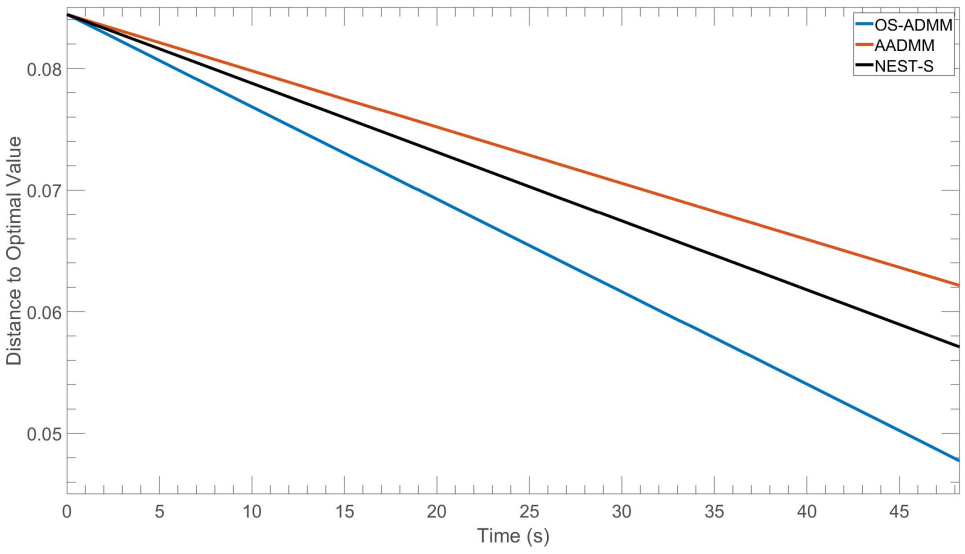

Figure 4.2: A comparison of OS-ADMM and existing algorithms.

and thus requires less computational time. Figure 4.2 demonstrates this behavior.

However, although such datasets exist in theory, we may be unlikely to encounter such a problem in practice. With this in mind, our primary focus will be on another problem. For our main experiment, we consider a more realistic comparison of S-ADMM, A-ADMM, and NEST-S on the following image reconstruction problem:

$$\min_{x \in X} \tfrac{1}{2} ||Ax - f||^2 + \lambda ||Dx||_{2,1}, \tag{4.5.3}$$

where x is a two dimensional image matrix reshaped into an n vector that is to be reconstructed, $\lambda ||Dx||_{2,1}$ is the discrete total variation seminorm regularization term with smoothing parameter λ, A is the acquisition matrix, f is the observed data, and X is a ball centered at the origin (see [30]). For this particular experiment, we let f be some fixed image with some standard normally distributed random noise added component wisely. Our acquisition matrix A is an m by $n = 2m$ matrix with normally distributed entries. The image reconstruction problem in (4.5.3) is particularlly well suited for the advantages of S-ADMM. First, this problem fits nicely into our proposed model in (2.3.1) since the projection onto X is trivial and the proximal problem of the discrete total variation seminorm is equivalent to projecting a vector onto the total variation ball. Second, the task of image

reconstruction benefits greatly from our dual-regularized variants that do not require N. Since it is often hard to associate a tolerance ε with the desired characteristics of a solution, it is often necessary to solve (4.5.3) to varying degrees of tolerance - a task which is not possible without the dual regularization variants or restarts.

Before we discuss the results, let us first note the impact of parameters n and λ and how they affect the model and theory. The parameter λ is the smoothness parameter in the total variation model. It balances the sharpness of the optimal solution with how closely each individual pixel of solution matches the original image. This parameter is a hyperparameter in the sense that the optimal value of λ for modeling purposes depends on the image. Theoretically, our λ is going to control ratio $||K||/L$. That is, for the model in (4.5.3), $L = ||A||^2$, and $||K|| = \lambda ||D||$. Whenever $||K||$ is small enough, the difference between gradient complexities of $\mathcal{O}(\sqrt{L/\varepsilon})$ from S-ADMM and $\mathcal{O}(\sqrt{L/\varepsilon} + ||K||/\varepsilon)$ from NEST-S and A-ADMM will be negligible. If $||K||$ is sufficiently large, however, S-ADMM will require significantly fewer of the expensive gradient evaluations. Thus, we should expect A-ADMM and NEST-S to perform better comparatively to S-ADMM whenever λ is small. On the other hand, the parameter n represents resolution of our initial image, i.e. the number of total entries in its gray-scale matrix representation. As n grows large, the amount of time needed to perform a single gradient evaluation grows quadratically in n. Thus, for large n, we expect to see even a small difference in gradient evaluation complexities result in a large difference of outputs.

With these trends in mind, we applied S-ADMM, A-ADMM, and NEST-S to two different images with varying resolutions and smoothness parameters. The experiments were conducted in the following way: S-ADMM was allowed to run for a fixed number of (outer) iterations and then both A-ADMM and NEST-S were run for the same amount of computational time spent on the S-ADMM iterations. The results are displayed in Tables 4.1 and 4.2. The columns can be described as follows: columns 1 and 2 are the aforementioned variable parameters, columns 3-7 depict the number of gradient evaluations, operator evaluations, final objective value, and relative error of the solution of S-ADMM, columns 8-10 showcase the total number of iterations, i.e. both the number of gradient and operator evaluations, the final objective value, and the relative error of the solution of A-ADMM, and columns 11-13 represent the same for NEST-S. Both tables have similar trends in results so we will discuss them simultaneously.

First, we notice that in all experiments, S-ADMM performs significantly less gradient evaluations and has more time to spend on operator evaluations. We will, however, focus our comparison

between S-ADMM and A-ADMM since NEST-S appears to perform unnaturally poorly for this problem. Second, as predicted, the objective values of S-ADMM are better than that of A-ADMM and NEST-S when $\log \lambda \in \{0, -1\}$, but get worse when λ grows smaller. However, as n grows larger in each experiment, this difference becomes more in favor of S-ADMM. This agrees with our previous discussion that suggests the differences in gradient evaluation complexity is displayed more clearly whenever n is large. Third, we note that the best relative error in Table 4.1 occurres when $\lambda = 10^{-1}$. That is, S-ADMM outperforms the other methods for optimally chosen λ. A similar argument can be made for the Lenna image whose numerical experiments are presented in Table 4.2. We conclude then that S-ADMM is a preferred method for image reconstructions problems, particularly when n is large.

		S-ADMM				A-ADMM			NEST-S		
$\sqrt{n}$	$\log\lambda$	**GE**	**OE**	**Obj Val**	**Rel Err**	**Iter**	**Obj Val**	**Rel Err**	**Iter**	**Obj Val**	**Rel Err**
100	0	500	125250	$1.00e+05$	11.51	18742	$1.03e+05$	11.47	18282	$1.85e+06$	102.39
150	0	500	125250	$1.86e+05$	9.52	8073	$2.17e+05$	9.23	8345	$3.74e+06$	92.55
200	0	500	125250	$2.86e+05$	8.19	4663	$3.76e+05$	7.28	4708	$5.57e+06$	77.69
100	-1	500	125250	$1.33e+04$	7.69	15135	$1.36e+04$	7.68	14834	$1.72e+05$	95.42
150	-1	500	125250	$2.47e+04$	6.03	3976	$2.88e+04$	6.96	4104	$2.47e+05$	63.49
200	-1	500	125250	$3.72e+04$	4.95	2608	$4.66e+04$	6.35	2684	$3.75e+05$	59.12
100	-2	500	125250	$1.65e+03$	9.52	14613	$1.42e+03$	7.48	15197	$1.38e+04$	76.63
150	-2	500	125250	$3.18e+03$	8.08	4228	$3.14e+03$	7.61	4093	$2.07e+04$	62.27
200	-2	500	125250	$4.97e+03$	7.10	2558	$7.07e+03$	12.53	2699	$3.77e+04$	66.90
100	-3	500	125250	$5.78e+02$	49.17	14351	$1.44e+02$	7.32	14535	$9.44e+02$	60.03
150	-3	500	125250	$1.30e+03$	49.43	4900	$4.36e+02$	13.16	4665	$2.16e+03$	68.85
200	-3	500	125250	$2.31e+03$	49.54	2772	$1.82e+03$	39.62	2522	$3.96e+03$	70.61

Table 4.1: Minimization of the total variation problem on our dog image

Minimization of the total variation problem on our dog image. We report the gradient evaluations, operator evaluations, terminating objective value, and the relative error of the reported solution for S-ADMM, A-ADMM, and NEST-S. For A-ADMM and NEST-S, the number of both gradient and operator evaluations are equal to the number of iterations.

		S-ADMM				A-ADMM			NEST-S		
$\sqrt{n}$	$\log\lambda$	**GE**	**OE**	**Obj Val**	**Rel Err**	**Iter**	**Obj Val**	**Rel Err**	**Iter**	**Obj Val**	**Rel Err**
100	0	500	125250	$8.18e+04$	6.17	15078	$8.75e+04$	6.19	14906	$2.23e+06$	84.79
150	0	500	125250	$1.50e+05$	5.02	3786	$2.12e+05$	5.26	3805	$4.08e+06$	70.78
200	0	500	125250	$2.31e+05$	4.44	2438	$3.49e+05$	4.17	2714	$6.12e+06$	61.19
100	-1	500	125250	$1.06e+04$	3.80	14884	$1.10e+04$	3.83	15266	$2.19e+05$	83.06
150	-1	500	125250	$1.96e+04$	3.06	3718	$2.51e+04$	4.60	3907	$3.24e+05$	59.24
200	-1	500	125250	$3.04e+04$	2.64	2456	$4.66e+04$	5.29	2577	$5.23e+05$	60.07
100	-2	500	125250	$1.90e+03$	9.15	15132	$1.17e+03$	3.53	14937	$1.74e+04$	67.99
150	-2	500	125250	$3.86e+03$	8.87	3787	$3.41e+03$	6.99	3831	$2.92e+04$	64.19
200	-2	500	125250	$6.48e+03$	8.81	2527	$9.73e+03$	12.91	2527	$5.40e+04$	68.15
100	-3	500	125250	$9.42e+02$	55.76	14598	$1.19e+02$	3.66	14298	$1.31e+03$	61.44
150	-3	500	125250	$2.12e+03$	54.61	3958	$6.08e+02$	12.67	3831	$3.12e+03$	69.50
200	-3	500	125250	$3.75e+03$	54.44	2274	$3.13e+03$	37.99	2369	$5.63e+03$	70.09

Table 4.2: Minimization of the total variation problem on the Lenna image

Minimization of the total variation problem on the Lenna image. We report the gradient evaluations, operator evaluations, terminating objective value, and the relative error of the reported solution for S-ADMM, A-ADMM, and NEST-S. For A-ADMM and NEST-S, the number of both gradient and operator evaluations are equal to the number of iterations.

Chapter 5

Conclusions and Discussion

The landscape of first-order convex optimization has quickly evolved as applications become increasingly large scale. From the optimization perspective, there are two natural avenues of improvement. For certain applications, there exists the possibility of a new, accelerated algorithm that is able to compute an approximate solution with less expensive operations than existing literature. On the other hand, we can expand the domain of some popular methods to achieve improved area of applicability. In this thesis, we discussed improvements in both directions.

We began in Chapter 3 by studying the popular CGS algorithm. Preferred due to its simplicity of implementation, the CGS algorithm is able to compute approximate solutions to smooth convex problems without the need for a projection oracle. Instead, it performs linear optimizations in its iterations. As discussed there, the CGS algorithm is an optimal algorithm for this problem class with respect to the number of linear optimizations and gradient evaluations computed. However, continual advancements can be made by relaxing some of the assumptions required by CGS. In Chapter 3, we proposed a new universal method, namely UCGS, that is able to achieve generalized CGS behavior on a wider class of functions. UCGS extends the applications of CGS to not only Lipschitz smooth, but also Hölder smooth problems while also removing the necessary knowledge of smoothness parameters. UCGS also maintains the optimal number of gradient evaluations for its expanded class of problems and improves upon the current number of linear optimizations required.

We then turned our attention to solving a more structured class, affinely constrained optimization problems. Current literature would suggest that there exists optimal algorithms NEST-S and A-ADMM for computing approximate solutions with respect to certain oracles. However, by

relaxing our class of algorithms, in Chapter 4 we proposed two new sliding algorithms that were preferable to the aforementioned methods both in practice and in theory. In theory, each algorithm reduces the number of calls to a particular operator required for an approximate solution. Such improvement was made possible by analyzing these methods under a new oracle approach. In practice, we showed in the numerical experiments that there exists problem sets such that the reduction in gradient/operator evaluations was enough for these sliding methods to be computationally faster than existing methods. Furthermore, we provided a more practical implementation of the gradient sliding algorithm using dual regularization. This variant relieves us of the N restriction in the parameter setting and was shown in the numerical experiments to increase functionality. Finally, we combined the two methods into a single sliding algorithm, S-ADMM, which was able to simultaneously achieve optimal gradient and operator evaluation complexity. Such an algorithm would not be possible under the current approach, but by using a two oracle analysis, we were able improve upon the existing methods.

There are still many open questions. In the UCGS algorithm, we were able to achieve the optimal gradient evaluation complexity for the class of Hölder smooth objectives while also improving the linear optimization complexity. In the smooth case, we know that this linear optimization complexity is optimal. However, for the Hölder smooth setting, the lower complexity bound is currently unknown. It is even shown in [31] that there exists methods that require even fewer linear optimizations in the nonsmooth case.

For the S-ADMM setting, although we have removed the N using dual regularization, the parameter settings still require the diameter of our feasible space for implementation. This is frequently the case for sliding algorithms including our own UCGS. How to remove such parameters is an open question for sliding algorithms. Furthermore, we have shown that dual regularization is sufficient to remove N from our GS-ADMM algorithm, but to design a similarly practical variant of S-ADMM, we must also propose a dual regularized OS-ADMM algorithm. We believe a similar method can be derived by the same technique, but such task has not been studied to our knowledge.

Bibliography

[1] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[2] Y. E. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, Massachusetts, 2004.

[3] Y. E. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN SSSR*, 269:543–547, 1983. translated as Soviet Math. Docl.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[5] Trevor Squires. Worst case datasets for solving binary logistic regression via deterministic first-order methods. unpublished thesis, 2020.

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

[7] Y. E. Nesterov. Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152(1-2):381–404, 2015.

[8] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[9] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[10] Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.

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

[12] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[13] Roland Glowinski and A Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[14] Magnus R Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[15] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In *Optimization (Sympos., Univ. Keele, Keele, 1968)*, pages 283–298. Academic Press, London, 1969.

[16] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[17] 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.

[18] 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.

[19] Bingsheng He and Xiaoming Yuan. On the O(1/n) convergence rate of the Douglas-Rdachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[20] Renato DC Monteiro and Benar F Svaiter. Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. *SIAM Journal on Optimization*, 23(1):475–507, 2013.

[21] Yu Nesterov. Complexity bounds for primal-dual methods minimizing the model of objective function. *Mathematical Programming*, 171(1):311–330, 2018.

[22] Saeed Ghadimi. Conditional gradient type methods for composite nonlinear and stochastic optimization. *Mathematical Programming*, 173(1):431–464, 2019.

[23] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1-2):75–112, 2015.

[24] Robert M Freund and Paul Grigas. New analysis and results for the Frank–Wolfe method. *Mathematical Programming*, 155(1-2):199–230, 2016.

[25] Hamid Nazari and Yuyuan Ouyang. Backtracking linesearch for conditional gradient sliding. *arXiv preprint arXiv:2006.05272*, 2020.

[26] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1):37–75, 2014.

[27] Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[28] Guanghui Lan, Edwin Romeijn, and Zhiqiang Zhou. Conditional gradient methods for convex optimization with function constraints. *arXiv preprint arXiv:2007.00153*, 2020.

[29] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[30] Yunmei Chen, William W Hager, Maryam Yashtini, Xiaojing Ye, and Hongchao Zhang. Bregman operator splitting with variable stepsize for total variation image reconstruction. *Computational Optimization and Applications*, 54(2):317–342, 2013.

[31] Kiran Koshy Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Projection efficient subgradient method and optimal nonsmooth frank-wolfe method. *arXiv preprint arXiv:2010.01848*, 2020.