PROXIMAL BUNDLE METHODS AND NONLINEAR ACCELERATION : AN EXPLORATION

A Senior Thesis

Submitted to the Department of Mathematics

by

Qingxuan Jiang

May 2021

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Adrian Lewis. His enthusiasm and knowledge of optimization has constantly inspired me throughout the year of conducting this research. Even though all of our meetings are virtual during this special period, he has made them as engaging as if I were in his office chatting about optimization.

I would also like to thank all the professors from the Cornell Math Department and beyond, for building up my mathematical expertise throughout my years as a math major, and influencing me to pursue research. This includes Professor John Hubbard, Camil Muscalu, James Renegar, Robert Strichartz, Alex Townsend, and many more.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

The main objective of this thesis is to have a better understanding of algorithms for nonsmooth optimization problems

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

where f is in general assumed to be convex and continuous, though not differentiable at all points.

There are several categories of algorithms that are often used to solve such optimization algorithms. One category is variants of the subgradient method, an extension of gradient methods in smooth optimization. Another category is variants of proximal point method, which minimizes the original function plus a quadratic minimizer. Neither of these two categories of methods are ideal, for different reasons: the performance of subgradient methods relies heavily on the choice of an adequate step size, and the proximal point method is only computable for specific examples of objective functions.

In this thesis, we will explore a class of algorithms called the proximal bundle method, which takes ideas from both subgradient method and proximal method above. Specifically, the bundle method can be stated equivalently in primal and dual form. The dual formulation of the bundle method corresponds to a smoothed version of subgradient descent, and the primal formulation corresponds to applying proximal point method on a cutting plane approximation of the objective function. In this sense, the proximal bundle method is theoretically an elegant construct, in the sense that it does not depend on a step size parameter, and it is always computable via solving a quadratic program at each iteration.

However, in practice, while the proximal bundle method works well in terms of requiring relatively fewer number of iterations compared to other algorithms, it is expensive to compute the solution of a quadratic program at each step. Moreover, each iteration of the method becomes increasingly difficult to compute, as the number of constraints in the quadratic program may grow with the number of iterations. Thus, an interesting question to ask is whether we can extrapolate a better iterate from the first k iterates, assuming that computing the next iterate may be very slow or even impossible.

Such ideas lead us to consider the nonlinear acceleration algorithm, which is a general acceleration routine that computes the best linear combination of the first k-iterates to accelerate an optimization algorithm. Whether such acceleration methods can indeed accelerate the proximal bundle method is still under research, and we hope this thesis could give some pointer on how we may approach this question.

The structure of the thesis is as follows. Chapter 2 provides some basic preliminaries in optimization that will be used in later chapters. Chapter 3 gives an introduction to the proximal bundle method, and presents some experimental results on its performance compared to other algorithms [1]. Chapter 4 introduces the nonlinear acceleration algorithm and its regularized variant, and ends with a short note of how we may want to apply such acceleration routine to the proximal bundle method.

[1]The code for the experiment section in the thesis can be found in the Github repository https://github.com/mathreader/nonlinear-acceleration-bundle-method.

CHAPTER 2

PRELIMINARIES

This chapter aims to set up notations and provide a brief summary of concepts in optimization that will be used in later chapters. More details about the definitions and properties in this chapter can be found in [1] and [8]. The reader is encouraged to quickly skim the chapter and go on to the main text starting in Chapter 3, and refer back whenever needed.

2.1 Classes of Functions

Our main goal is to minimize a function $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ over the extended real numbers $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$. This is often written compactly as $\min\limits_{x \in \mathbb{R}^d} f(x)$. We will first define the domain of f as the region where f is finite.

Definition 2.1.1 (Domain of Extended-Value Function)**.** *We define the **domain** of f by dom(f) $= \{x \in \mathbb{R}^d : f(x) < \infty\}$.*

In most cases, we will assume f to have certain nice properties that allow us to find minima easily. Such properties fall into one of the several categories below:

(1) **Finiteness of Objective Value**: While we allow the objective function to take values $+\infty$ or $-\infty$, we often would like to make sure our optimum is finite (and thus the optimization problem is nontrivial). A condition that guarantees is for f to be proper:

Definition 2.1.2 (Properness)**.** *We say f is **proper** if it does not attain $-\infty$, and dom(f) $\neq \emptyset$.*

(2) **Smoothness of Objective Function**: A key property that many optimization problems rely on is smoothness of the objective function f, to varying degrees. We start by looking at a weaker notion of continuity, that has nice interplay with minimization problems:

Definition 2.1.3 (Closedness/Lower-Semicontinuity). *Let us define the epigraph of f as $epi(f) = \{(x, y) : f(x) \leq y, x \in \mathbb{R}^d, y \in \mathbb{R}\}$. Then we say the function f is* ***closed*** *if its epigraph $epi(f)$ is closed as a set. An equivalent definition is that of* ***lower-semicontinuity****, which states that for any convergent sequence $\lim_{n\to\infty} x_n = x$, we have $f(x) \leq \liminf_{n\to\infty} f(x_n)$.*

We also briefly list the standard definitions of (Lipschitz) continuity and differentiability in analysis, just for completeness:

Definition 2.1.4 (Continuity). *We say that f is* ***continuous*** *if for any convergent sequence $\lim_{n\to\infty} x_n = x$, we have $\lim_{n\to\infty} f(x_n) = f(x)$.*

Definition 2.1.5 (Lipschitz Continuity). *We say that f is* ***Lipschitz continuous*** *with constant $L > 0$, if we have $\forall x, y \in \mathbb{R}^d$, $|f(x) - f(y)| \leq L\|x - y\|_2$.*

Definition 2.1.6 (Differentiability). *We say that f is* ***differentiable*** *if for all $x \in \mathbb{R}^d$, the derivative $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$ exists.*

(3) **Convexity**: Convexity properties are often used in optimization to characterize the minima of a function. This also comes in several different variants, that we list below:

Definition 2.1.7 (Convexity). *We say f is* ***convex*** *if for any $t \in (0, 1)$ and $x, y \in \mathbb{R}^d$, we have*

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$$

in case where f is first/second-order differentiable, we can also write equivalent definitions

$$f \text{ is convex} \iff \forall x, y \in \mathbb{R}^d \text{ , } f(x) \geq f(y) + \nabla f(y)(x - y) \iff \forall x \in \mathbb{R}^d \text{ , } \nabla^2 f(x) \geq 0$$

Stronger versions of convexity include the following:

Definition 2.1.8 (Strict Convexity)**.** *We say f is* ***strictly convex*** *if for any $t \in (0, 1)$ and $x, y \in \mathbb{R}^d$, we have the strict inequality*

$$f(tx + (1 - t)y) < tf(x) + (1 - t)f(y)$$

Definition 2.1.9 (Strong Convexity)**.** *We say f is* ***strongly convex*** *with parameter m if we have for all $x, y \in \mathbb{R}^d$,*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq m||x - y||^2 \iff f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{m}{2}||y - x||^2$$

or equivalently, if f is second-order differentiable, we have $\forall x \in \mathbb{R}^d$, $\nabla^2 f(x) \geq mI$.

As a brief note for how this relates to optimization, there is a celebrated theorem ensures that for convex f, the local minima are also global minima. Also, if we have strict/strong convexity, then f has a unique minima on any compact set.

2.2 Gradients and Subgradients

One of the most well-known optimization algorithms in convex optimization, the (sub)gradient descent algorithm, uses information of the gradient and subgradient. We start by introducing the definition of gradient for differentiable functions:

Definition 2.2.1 (Gradient)**.** *For differentiable $f : \mathbb{R}^d \to \mathbb{R}$, we define the* ***gradient*** *$\nabla f(x) = g(x)$ to be a function satisfying*

$$\forall y \in \mathbb{R}^d\ ,\ f(x+ty) = f(x) + t\langle g, y\rangle + o(t) \Longleftrightarrow \lim_{t\to 0} \frac{f(x+ty) - f(x) - t\langle g, y\rangle}{t} = 0$$

it can be shown that the gradient is well-defined and unique.

We now introduce analogous notion of subgradient for convex functions that are not necessarily differentiable.

Definition 2.2.2 (Subgradient/Subdifferential)**.** *For convex $f : \mathbb{R}^d \to \overline{\mathbb{R}}$, we say y is a* ***subgradient*** *of f at x if we have*

$$\forall z \in \mathbb{R}^d\ ,\ f(z) - f(x) \geq \langle y, z - x\rangle$$

the set of all subgradients of f at x is called the ***subdifferential*** *of f at x, and denoted by $\partial f(x)$. Thus, we often write $y \in \partial f(x)$ to indicate y is a subgradient.*

Since the subgradient is not unique, we would like a way to pick a representative from the subdifferential set. A natural idea is to pick the subgradient closest to **0**, as in the following definition of shortest subgradient.

Definition 2.2.3 (Shortest Subgradient)**.** *Let $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ be proper and convex, and $x \in int(dom(f))$ be in the interior of domain of f. Then there exists a unique solution to*

$$\partial^0 f(x) = \min_{g \in \partial f(x)} \|g\|_2$$

and we call $\partial^0 f(x)$ the shortest subgradient of f at x.

We will use several properties of subgradients, listed here:

Theorem 2.2.4 (Properties of Subdifferential)**.** *Let f, g be convex functions, and $h : \mathbb{R} \to \mathbb{R}$ be a nondecreasing convex function. Then we have the following properties for subdifferential*

- *(1)* ***Sum rule****: For $x \in int(dom(f)) \cap int(dom(g))$, we have*

$$\partial(f+g)(x) = \partial f(x) + \partial g(x).$$

- *(2)* ***Chain rule****: If h is differentiable at $x \in int(dom(f))$, then we have*

$$\partial(h \circ f)(x) = h'(f(x))\partial f(x).$$

- *(3)* ***Optimality condition****: We have $0 \in \partial f(x)$ if and only if $f(x) = \inf_{x \in \mathbb{R}^n} f$ is optimal.*

2.3 Directional Derivative and the Max Formula

A related notion to subgradients that will be useful to us is the directional derivative:

Definition 2.3.1 (Directional Derivative)**.** *Let f be proper and convex, let $x \in int(dom(f))$, and let $d \in \mathbb{R}^n$ represent some direction. Then the directional derivative of f at x in direction d always exists, and is defined by*

$$f'(x,d) = \lim_{\alpha \to 0+} \frac{f(x+\alpha d) - f(x)}{\alpha}$$

Directional derivatives are related to subgradients via the max formula:

Theorem 2.3.2 (Max Formula)**.** *Let f be proper and convex, then for any $x \in int(dom(f))$ and $d \in \mathbb{R}^d$, we have*

$$f'(x;d) = \max\{\langle g, d \rangle : g \in \partial f(x)\}$$

2.4 Set-Valued Mappings

In the treatment of some later chapters, we will consider the subdifferential operator ∂f itself as a function, mapping from $x \in \mathbb{R}^n$ to a subset of $\partial f(x) \subseteq \mathbb{R}^n$. Here we present a more formal definition of this from variational analysis, taken from [8, Chapter 5].

Definition 2.4.1 (Set-Valued Mappings)**.** *We say that S is a* ***set-valued mapping*** *from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$ (denoted by $S : \mathbb{R}^{d_1} \rightrightarrows \mathbb{R}^{d_2}$) if for each $x \in \mathbb{R}^{d_1}$, we have $S(x) \subseteq \mathbb{R}^{d_2}$ being a subset of $\mathbb{R}^{d_2}$.*

In some cases, we need to define when such set-valued mappings are continuous, and we provide an analogous setting to the usual continuity definition.

Definition 2.4.2 (Continuity of Set-Valued Mappings)**.** *Let $S : \mathbb{R}^{d_1} \rightrightarrows \mathbb{R}^{d_2}$ be a set-valued mapping.*

- *(1) We say S is* ***outer semicontinuous*** *at x if for any convergence sequences $(x_k, y_k) \to (x, y)$ inside the graph (i.e. $y_k \in S(x_k)$), we have $y \in S(x)$.*
- *(2) We say S is* ***inner semicontinuous*** *at x if for any $y \in S(x)$ and sequence $x_k \to x$, there exists some convergent sequence $y_k \to y$ such that $y_k \in S(x_k)$.*
- *(3) We say S is* ***continuous*** *at $\overline{x}$ if it is both outer and inner semicontinuous.*

CHAPTER 3

PROXIMAL BUNDLE METHOD

In this chapter, we will describe the proximal bundle method, one of the optimization algorithms for black-box optimization of nonsmooth functions.

In Sections 3.1 and 3.2, we introduce two separate formulations of the proximal bundle method: ϵ-steepest descent and the cutting plane method. We then describe how these two formulations are related to each other using a duality argument in Section 3.3. Finally in Section 3.4, we present some experiments comparing the performance of proximal bundle method to other optimization algorithms that work on nonsmooth functions.

For a more detailed introduction to proximal bundle method, the interested reader is referred to [11, Chapter XIII, XIV, XV, XVI].

3.1 ϵ-Steepest Descent

We start by introducing a dual formulation of the the proximal bundle method, the ϵ-steepest descent algorithm. This algorithm can be thought of as an extension of the line search method for subgradient descent.

3.1.1 Extending Line Search to Subgradient Methods

Motivation (Line Search for Gradient Descent)**.** Recall that in smooth optimization, one of the most widely used algorithm is the gradient descent algorithm: If we want to solve the problem $\min\limits_{x \in \mathbb{R}^n} f(x)$ with differentiable $f : \mathbb{R}^n \to \mathbb{R}$, we can perform the iteration

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

where α_k is a step size to be determined in each step.

In cases where f is Lipschitz continuous with constant L, we can simply pick the fixed step size $\alpha_k = \frac{2}{L}$, and this guarantees convergence of gradient descent. However in practice, even if the objective function is indeed Lipschitz continuous, we often do not know the Lipschitz constant to set up the step size in advance. In this case, we will apply a line search to the step size, where we approximately solve the problem

$$\min_{\alpha} f(x_k - \alpha \nabla f(x_k))$$

and set α_k to be the approximate solution. Under certain conditions on the approximate solution (for example, the Armijo-Wolfe condition [7]), we can guarantee that the algorithm converges.

A natural question that we may ask here is the following:

Question. Is it possible for us to apply line search to subgradient descent?

To answer this question, we will first set up subgradient descent in an analogous framework as gradient descent. Our goal is now to solve $\min\limits_{x \in \mathbb{R}^n} f(x)$ with convex $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ that is not necessarily differentiable. The subgradient method performs the iteration

$$x_{k+1} = x_k - \alpha_k g_k \text{ where } g_k \in \partial f(x_k).$$

Here we observe that unlike the gradient, the subgradient g_k is not necessarily unique. For simplicity, we will just consider picking the shortest subgradient $g_k = \partial^0 f(x_k)$.

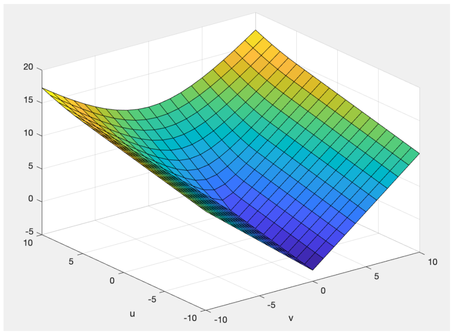
While it is tempting to continue mimicking the step of applying line search to find α_k for each shortest gradient, it turns out that even with exact line search, the algorithm will not necessarily output the optimal solution, as shown in the following example.

Example (Subgradient Method with Line Search May Fail on Nonsmooth Problems)**.** *To illustrate a scenario where line search on shortest subgradients may fail, we consider the function*

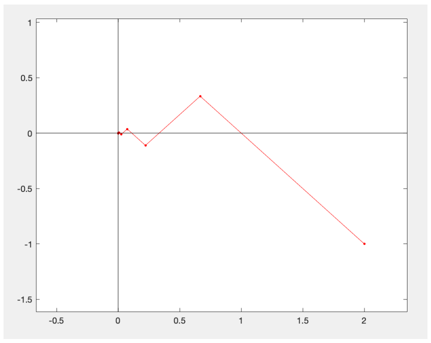
$$f(u,v) = \begin{cases} \sqrt{u^2 + 2v^2}\ , \textit{ if } u \geq \frac{|v|}{2} \\ \frac{1}{3}(u + 4|v|)\ , \textit{ otherwise} \end{cases}$$

which is illustrated in Figure 3.3 (a). We can see that the function has minimum $-\infty$ when we pick $v = 0$ and let $u \to -\infty$.

Figure 3.1: Illustration of Pathological Function and Iterates for Subgradient Method with Line Search



(a) 3D plot of pathological function that we want to apply line search on. Note the linear "valley" of the function when $v = 0, u < 0$, which has different properties from the region with $u \geq 0$.



(b) 2D plot of iterates of subgradient method with line search, starting at $(2, -1)$ (bottom-right point) and moving toward the left. We see that the iterates converge to $(0, 0)$.

Now consider the behavior of this function when we run a line search on shortest *subgradients, beginning at the point* $(u_0, v_0) = (2, -1)$.

We first compute the subgradient set at each point:

$$\partial f(u,v) = \begin{cases} \left(\frac{u}{\sqrt{u^2+2v^2}}, \frac{2v}{\sqrt{u^2+2v^2}}\right) \text{ , } \textit{if } u > \frac{|v|}{2} \\ \left(\frac{1}{3}, \frac{4}{3} sign(v)\right) \text{ , } \textit{if } u \le \frac{|v|}{2} \textit{ and } v \neq 0 \\ \frac{1}{3} \times [-1, 1] \text{ , } \textit{if } u \le \frac{|v|}{2} \textit{ and } v = 0 \end{cases}$$

and thus the shortest subgradient is given by

$$g(u,v) = \begin{cases} \left(\frac{u}{\sqrt{u^2+2v^2}}, \frac{2v}{\sqrt{u^2+2v^2}}\right) \text{ , } \textit{if } u > \frac{|v|}{2} \\ \left(\frac{1}{3}, \frac{4}{3} sign(v)\right) \text{ , } \textit{if } u \le \frac{|v|}{2} \end{cases}$$

Now starting from $(u_0, v_0) = (2, -1)$, *we can compute the first few iterates of the subgradient method with line search using a computer program. The first* 10 *iterates are shown in Table 3.1, and plotted in 3.3 (b). We can see that while the objective value does decrease throughout the process, the iterates converge to the point* $(0, 0)$, *which is not the minimizer of the function.*

Iteration	u	v	Objective Value
0	2	-1	2.4494
1	$6.6667 * 10^{-1}$	$3.333 * 10^{-1}$	$8.164 * 10^{-1}$
2	$2.2222 * 10^{-1}$	$-1.111 * 10^{-1}$	$2.722 * 10^{-1}$
3	$7.4074 * 10^{-2}$	$3.7037 * 10^{-2}$	$9.072 * 10^{-2}$
4	$2.4690 * 10^{-2}$	$-1.2347 * 10^{-2}$	$3.024 * 10^{-2}$
5	$8.2265 * 10^{-3}$	$4.1192 * 10^{-3}$	$1.008 * 10^{-2}$
6	$2.7455 * 10^{-3}$	$-1.3697 * 10^{-3}$	$3.360 * 10^{-3}$
7	$9.0409 * 10^{-4}$	$4.6763 * 10^{-4}$	$1.120 * 10^{-3}$
8	$3.1312 * 10^{-4}$	$-1.4371 * 10^{-4}$	$3.733 * 10^{-4}$
9	$8.9834 * 10^{-5}$	$6.12515 * 10^{-5}$	$1.248 * 10^{-4}$

Table 3.1: Table of First 10 Iterates for Subgradient Method with Line Search in Pathological Example

The main issue with the above counterexample lies in the fact that the sign function is not continuous at 0, and thus the shortest subdifferential $g(u, v)$ is also not continuous around $(0,0)$. This means the subdifferential at $(0,0)$ does not give very useful information about the subdifferential at some nearby point of $(0,0)$, and this causes the iterates to become stuck at that point. One remedy to this problem is to introduce a continuous version of subdifferential, called ϵ-subdifferential, which we will discuss in the next section.

3.1.2 The ϵ-Differential

As motivated by the above section, we will introduce the ϵ-subdifferential as a "smoothed" version of the original subdifferential definition.

Definition 3.1.1 (ϵ-Subdifferential)**.** *For any $\epsilon > 0$, we define the **ϵ-subdifferential** of f at x to be*

$$\partial_\epsilon f(x) = \{y \in \mathbb{R}^n : \forall z \in \mathbb{R}^n, \langle y, z - x\rangle \leq f(z) - f(x) + \epsilon\}$$

We then state several useful properties of the ϵ-subdifferential. Proofs of these properties can be found in [11, Chapter XIII].

Theorem 3.1.2 (Properties of ϵ-Subdifferential)**.** *The ϵ-subdifferential satisfies the following properties:*

- *(1) **Match with subdifferential definition**: $\partial_\epsilon f(x) = \partial f(x)$ when $\epsilon = 0$.*
- *(2) **Monotonicity**: The subdifferential set $\partial_\epsilon f(x)$ grows with ϵ.*
- *(3) **Closedness and convexity**: $\partial_\epsilon f(x)$ is closed and convex.*
- *(4) **Optimality condition**: $0 \in \partial_\epsilon f(x)$ if and only if x is ϵ-optimal, i.e. $f(x) \leq \inf f + \epsilon$.*

We now give a simple example of the ϵ-subdifferential on the function $f(x) = |x|$, which demonstrates how we convert its discontinuous subdifferential into a continuous counterpart.

Example (ϵ-Subdifferential of Absolute Value Function). *For the absolute value function $f(x) = |x|$, its subgradient is given by*

$$\partial f(x) = \begin{cases} 1 \text{ , } \textit{if } x > 0 \\ [-1,1] \text{ , } \textit{if } x = 0 \\ -1 \text{ , } \textit{if } x < 0 \end{cases}$$

We can show that ∂f is not a continuous set-valued mapping as defined in Definition 2.4.2: If we pick the sequence $x_k = \frac{1}{k}$ and $y_k \in \partial f(x_k)$, we see that $x_k \to 0$ and also $y_k = 1$ for all k. This means $y_k \to 1$, and thus it cannot converge to any other value in $[-1, 1) \subseteq \partial f(0)$. This implies ∂f is not inner semicontinuous, and thus ∂f is not continuous.

The ϵ-subdifferential of f is given by

$$\partial_\epsilon f(x) = \{y \in \mathbb{R} : \forall z \in \mathbb{R}, y(z - x) \leq |z| - |x| + \epsilon\}$$

WLOG $x > 0$. Then we have three cases to consider:

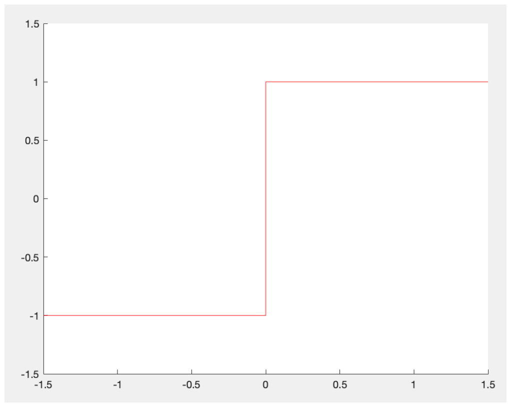
- *Case 1 ($z > x > 0$): We have $y(z - x) \leq z - x + \epsilon$, so $y \leq 1 + \frac{\epsilon}{z-x}$. As $z \to_+ x$, this implies $y \leq 1$.*
- *Case 2 ($x > z > 0$): We again have $y(z - x) \leq z - x + \epsilon$, so $y \geq 1 + \frac{\epsilon}{z-x}$. As $z \to_+ 0$, this gives $y \geq 1 - \frac{\epsilon}{x}$.*
- *Case 3 ($x > 0 > z$): We have $y(z - x) \leq -z - x + \epsilon$, so $y \geq -1 + \frac{\epsilon - 2x}{z-x}$. Now if $x \geq \frac{\epsilon}{2}$, then picking $z \to_- 0$ gives $y \geq 1 - \frac{\epsilon}{x}$. Otherwise, if $x < \frac{\epsilon}{2}$, then picking $z \to -\infty$ gives $y \geq -1$.*

In all, we see that we always have $\max\{-1, 1-\frac{\epsilon}{x}\} \leq y \leq 1$. *Similarly for* $x \leq 0$, *we must have* $-1 \leq y \leq \min\{1, -1-\frac{\epsilon}{x}\}$. *Thus, the final* ϵ*-subdifferential formula is given by*

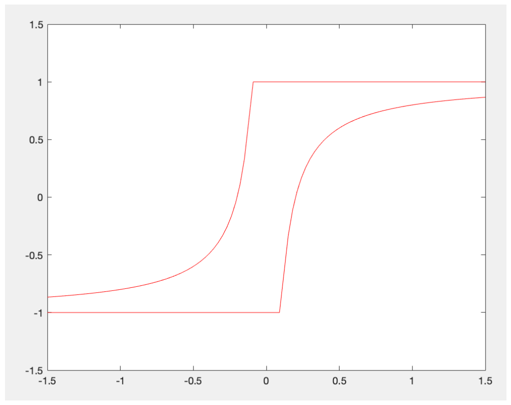
$$\partial_\epsilon f(x) = \begin{cases} [1-\frac{\epsilon}{x}, 1]\text{ , } \textit{if } x > \frac{\epsilon}{2} \\ [-1, 1]\text{ , } \textit{if } -\frac{\epsilon}{2} \leq x \leq \frac{\epsilon}{2} \\ [-1, -1-\frac{\epsilon}{x}]\text{ , } \textit{if } x < -\frac{\epsilon}{2} \end{cases}$$

Through some lengthy computations, we can verify that the ϵ*-subdifferential is continuous as a set-valued mapping, though this is best illustrated by plotting its region, as we do in Figure 3.2. We can see that in the* ϵ*-subdifferential region on the right, if we pick any convergent sequence in the x axis, we can always pick points in the y-axis such that they converge to any target y-value within the region, thus verifying the continuity of* $\partial_\epsilon f$.

Figure 3.2: Illustration of original subdifferential and ϵ-subdifferential for $f(x) = |x|$.



(a) The original subdifferential includes the red lines.



(b) The ϵ-subdifferential is the region between the red lines, which is indeed a "smoothed" version of the original subdifferential on the left.

We now introduce smoothed versions of directional derivative in Definition

2.3.1, and also state a corresponding version of the max formula in Theorem 2.3.2.

Definition 3.1.3 (ϵ-Directional Derivative)**.**

$$f'_\epsilon(x;d) := \inf_{t>0} \frac{f(x+td) - f(x) + \epsilon}{t}$$

Theorem 3.1.4 (Max Formula for ϵ-Directional Derivative)**.** *The ϵ-subdifferential and ϵ-directional derivative are related in the following way:*

$$f'_\epsilon(x;d) = \max_{y \in \partial_\epsilon f(x)} \langle y, d \rangle$$

Given the above tools, we are now ready to state the statement of the ϵ-steepest descent algorithm in the next section.

3.1.3 ϵ-Steepest Descent Algorithm

In this section, we illustrate how we can modify the subgradient descent algorithm such that it works with the ϵ-subdifferential. We start by defining the descent direction to be the opposite direction of the shortest ϵ-subgradient.

Definition 3.1.5 (ϵ-Steepest Descent Direction)**.** *We define the ϵ-steepest descent direction of f at x to be*

$$d_\epsilon = -proj_{\partial_\epsilon f(x)}(0)$$

We now present the ϵ-steepest descent algorithm, which performs line search on the above descent direction at each step.

Algorithm 1: ϵ-Steepest Descent Algorithm

```
**Input**: Error tolerance $\epsilon$, maximum iteration $max\_iter$, initial iterate $x_0$
$k = 0$
while $k <= max\_iter$ do
    $d_k = -proj_{\partial_\epsilon f(x_k)}(0)$
    if $d_k = 0$ then
        break
    else
        Apply line search to find $t_k = \min_{t>0} f(x_k + td_k)$
        $x_{k+1} = x_k + t_k d_k$
    end
    $k = k + 1$
end
return $x_k$
```

Theorem 3.1.6 (Convergence of ϵ-Steepest Descent). *Each step of ϵ-steepest descent decreases the objective value by ϵ, i.e. we have*

$$f(x_{k+1}) < f(x_k) - \epsilon$$

Thus, if the above algorithm is allowed to run indefinitely, it will terminate at some x' with $f(x') - \min f \leq \epsilon$, thus within ϵ of optimal objective value.

Proof. From the max formula in Theorem 3.1.4, we know that

$$\inf_{t>0} \frac{f(x_k + td_k) - f(x_k) + \epsilon}{t} = f'_\epsilon(x_k; d_k) = \max_{y \in \partial_\epsilon f(x)} \langle y, d_k \rangle = -\|d_k\|^2 < 0$$

Now if $t^* > 0$ achieves the infimum above, we then have

$$f(x_k + t^* d_k) < f(x_k) - \epsilon - t^* \|d_k\|^2 < f(x_k) - \epsilon$$

and thus

$$f(x_{k+1}) = \min_{t>0} f(x_k + td_k) \leq f(x_k + t^* d_k) < f(x_k) - \epsilon$$

and this proves the theorem. □

Remark. *We note that each iteration of the above algorithm requires us to compute the projection onto $\partial_\epsilon f(x_k)$. Since ϵ-subdifferentials are often hard to compute for general optimization problems, it is difficult to generate a closed form formula for the projection, thus making the algorithm hard to implement. In the next section, we will discuss ways to approximate the ϵ-subdifferential that makes the projection step easier.*

3.1.4 Bundle Iteration

We now present the bundle iteration, an important tool that allows us to "approximate" an arbitrary set Q via another set P (typically a polytope that is easy to analyze and compute with). Let us assume that we are given a set $Q \subseteq \mathbb{R}^n$ that is nonempty, compact and convex. Our goal is to decide if $0 \in Q$. (Note that once we know how to do this, we can also decide whether $q \in Q$ for arbitrary q, based on translation.)

The algorithm requires the following oracle:

Assumption (Oracle for Bundle Iteration)**.** *We assume that we have an oracle that does the following: For set Q and any input $p \in \mathbb{R}^n$, the oracle either finds $q \in Q$ such that $\langle p, q \rangle \leq 0$, or determines that such q does not exist.*

With the above oracle, we can now present the bundle iteration algorithm:

```
Algorithm 2: Bundle Iteration
Input: A point $q_0 \in Q$, maximum iteration $max\_iter$
Algorithm: $P = \{q_0\}$
$k = 0$
result = False
while $k <= max\_iter$ do
    $p = proj_P(0)$
    Call oracle with set $P$ and input $p$
    if oracle finds $q$ then
        $P = conv(P, q)$
    else
        result = True
        break
    $k = k + 1$
return result
```

Theorem 3.1.7 (Guarantees for Bundle Iteration)**.** *The bundle iteration will terminate (i.e. oracle returns non-existence of q in finite iterations) if and only if $0 \notin Q$.*

Proof. We will prove the contrapositive of the statement.

(1) One direction of this is easy: If $0 \in Q$, then the oracle can repeatedly pick $q = 0$ that always satisfies $\langle p, q \rangle \leq 0$, so the algorithm would never terminate.

(2) For the other direction, assume that the bundle iteration does not terminate. Then, we get a sequence of polytopes $\{P_k\}$ with corresponding shortest vectors $\{p_k\}$. We also note from construction that P_k is a convex combination of points in Q, and this combined with Q being convex gives $P_k \subseteq Q$ for all k.

From bundle iteration construction, each P_{k+1} contains p_k and some $q_k \in Q$

satisfying $\langle p_k, q_k \rangle \leq 0$. Moreover, since p_{k+1} is the shortest vector in P_{k+1}, we have $\|p_{k+1}\| \leq \|p_k\|$. Thus, the sequence $\{\|p_k\|\}_k$ is decreasing and bounded below by 0, so it converges.

Also, from characterization of p_{k+1} as shortest vector, we have

$$\langle p_{k+1}, p_k - p_{k+1} \rangle \geq 0 \Longleftrightarrow \langle p_{k+1}, p_k \rangle \geq \|p_{k+1}\|^2$$

$$\langle p_{k+1}, q_k - p_{k+1} \rangle \geq 0 \Longleftrightarrow \langle p_{k+1}, q_k \rangle \geq \|p_{k+1}\|^2$$

Thus, we have as $k \to \infty$,

$$\|p_k - p_{k+1}\|^2 = \|p_k\|^2 + \|p_{k+1}\|^2 - 2\langle p_{k+1}, p_k \rangle \leq \|p_k\|^2 - \|p_{k+1}\|^2 \to 0$$

$$\|p_{k+1}\|^2 \leq \langle p_{k+1}, q_k \rangle \leq \langle p_{k+1} - p_k, q_k \rangle \leq \|p_{k+1} - p_k\| \cdot \|q_k\| \to 0$$

as $\|q_k\|$ is bounded from Q being compact. This proves that $p_k \to 0$. Since Q is closed with $p_k \in P_k \subseteq Q$, we know that the limit point 0 satisfies $0 \in Q$.

□

3.1.5 ϵ-steepest descent algorithm with bundle approximation

We now design a approximated version of the ϵ-steepest descent algorithm based on the bundle iteration. Our goal is again to minimize a function f, which we assume to be proper, convex and continuous. The algorithm keeps track of an iterate x and a polytope approximation P of Q. The main iteration performs two types of steps: If ϵ-steepest descent on the approximation P gives sufficient decrease, we perform a serious step where x is updated; otherwise, we perform a null step where x is kept the same but P is updated to better approximate the set Q. The algorithm is shown below:

```
Algorithm 3: $\epsilon$-steepest descent with bundle iteration
Input: A point $q_0 \in Q$, initial iterate $x_0$, maximum iteration $max\_iter$
Algorithm: $P_0 = \{q_0\}$
$k = 0$
while $k <= max\_iter$ do
    $d_k = -proj_{P_k}(0)$
    Apply line search to find $t_k = \min_{t>0} f(x_k + td_k)$
    if $f(x_k + t_k d_k) < f(x_k) - \epsilon$ then
        $x_{k+1} = x_k + t_k d_k$
        Pick $P_{k+1} = \{g_{k+1}\}$ for some $g_{k+1} \in \partial f(x_{k+1})$
    else
        $x_{k+1} = x_k$
        Call bundle iteration with set $P_k$
        if bundle iteration terminates with output $q$ then
            $P_{k+1} = conv(P_k, \{q\})$
        else
            break
    $k = k + 1$
return $x_k$
```

The serious step with steepest descent is similar to what we have done in the original ϵ-steepest descent algorithm, so our analysis mainly focus on the null step. We want to ensure that (1) whenever $f(x_k) > \min f + \epsilon$, the bundle iteration can always find a vector q and (2) we only need to perform a finite number of null steps before the next serious step. These two properties are formulated and proven in the two theorems below:

Theorem 3.1.8 (Existence of Solution to Bundle Iteration)**.** *Assume that we have*

$f'_\epsilon(x;d) \geq 0$. Then there exists some $q \in \partial_\epsilon f(x)$ with $\langle p, q\rangle \leq 0$, so the oracle can output some q such that we can perform the bundle iteration.

Proof. Let $t > 0$ be the parameter that minimizes $\frac{1}{t}\left(f(x+td) - f(x) + \epsilon\right)$.

Step 1 (Applying convexity of f): Since f is convex, we know that the function

$$g(s) = \left(f(x+sd) - f(x) + \epsilon\right) - \frac{s}{t}\left(f(x+td) - f(x) + \epsilon\right) \geq 0$$

is also convex, and is minimized at $s = t$.

Step 2 (Simplifying the optimality conditions): From optimality condition (Theorem 2.2.4 (3)), we have $0 \in \partial g(t)$, so

$$0 \in \partial\left(f(x + \cdot\, d) - f(x) + \epsilon - \frac{\cdot}{t}(f(x+td) - f(x) + \epsilon)\right)(t)$$

where the dot above represent the variable that we re taking subdifferential on. Now we can apply the sum rule of subdifferential (Theorem 2.2.4 (1)) to get

$$\frac{1}{t}\left(f(x+td) - f(x) + \epsilon\right) \in \partial\left(f(x + \cdot\, d)\right)(t)$$

Step 3 (Proving inner product inequality): From the chain rule for subdifferential (Theorem 2.2.4 (2)), we have

$$\partial\left(f(x + \cdot\, d)\right)(t) = \langle \nabla_t(x+td), \partial f(x+td)\rangle = \langle d, \partial f(x+td)\rangle$$

Thus, there exists $q \in \partial f(x+td)$ with

$$\langle d, q\rangle = \frac{1}{t}\left(f(x+td) - f(x) + \epsilon\right) = f'_\epsilon(x;d) \geq 0 \Rightarrow \langle p, q\rangle = \langle -d, q\rangle \leq 0$$

from t being the optimal choice and $f'_\epsilon(x;d) \geq 0$ being the assumption for entering the null step.

Step 4 (Proving containment inside ϵ-subdifferential): Moreover, since $q \in \partial f(x + td)$, we know that

$$\forall z \in \mathbb{R}^n\ , \langle z - (x + td), q\rangle \leq f(z) - f(x + td)$$

Also, we know that

$$-t\langle d, q\rangle \leq 0 \leq f(x + td) - f(x) + \epsilon$$

and summing the above two inequalities gives

$$\forall z \in \mathbb{R}^n\ , \langle z - x, q\rangle \leq f(z) - f(x) + \epsilon$$

and thus $q \in \partial_\epsilon f(x)$, so there is indeed at least one choice of q that the bundle iteration can output. □

Theorem 3.1.9 (Finite Null Steps between Serious Steps)**.** *Assuming that $f(x) > \inf f + \epsilon$, so the current iterate is not within ϵ of the optimal value (and thus a serious step can be performed). Then we can only perform a finite number of null steps before we perform the serious step.*

Proof. Since $f(x) > \inf f + \epsilon$, we know that $0 \notin \partial_\epsilon f(x)$. Thus, we know that the bundle iteration applied to $Q = \partial_\epsilon f(x)$ and initial polytope $P = \{g\}$ for some $g \in \partial f(x) \subseteq Q$ must terminate in finite number of steps.

At the point where the bundle iteration terminates, we have some direction p such that for any $q \in Q$, we have $\langle p, q\rangle > 0$. This means if we pick $d = -p$, then $\langle d, q\rangle < 0$ for any $q \in \partial_\epsilon f(x)$.

We know from Step 3 in the proof of above Theorem 3.1.8 that we can write $f'_\epsilon(x; d) = \langle d, q_0\rangle$ for some $q_0 \in \partial f(x+td) \subseteq \partial_\epsilon f(x)$. This means we have $f'_\epsilon(x; d) < 0$.

Thus, at the beginning of the next iteration of the algorithm, we will perform the serious step, and thus the number of null steps is finite before we get to a serious step. □

Remark. *One thing to note about the above theorem is that the number of null steps between serious steps depends on the number of bundle iterations needed, and there is no explicit upper bound on this number for the general setting. This makes it relatively hard to estimate the worst case bound for this algorithm - which is only possible for specific simple objective functions f.*

In the next section, we will consider an alternate, primal way of interpreting the proximal bundle methods via cutting planes.

3.2 Cutting Plane Method

This section discusses a primal formulation of the bundle method via cutting planes, which dates back to 1960 in the paper [3].

3.2.1 Cutting Plane Method

The setting of cutting plane method is similar to what we have in the previous case: We would like to minimize convex $f : \mathbb{R}^n \to \mathbb{R}$ over closed convex $X \subseteq \mathbb{R}^n$. We have access to an oracle that takes input $x \in X$ and outputs a subgradient $\partial f(x)$.

Our first goal here is to define what a "cutting plane" is in this setting. Given any (x_j, g_j) with $g_j \in \partial f(x_j)$, we can construct a plane $h_j : X \to \mathbb{R}$ given by the equation

$$h_j(x) = f(x_j) + \langle g_j . x - x_j \rangle$$

From definition of subgradient, we know that

$$\forall x \in X \ , \ f(x) \geq f(x_j) + \langle g_j . x - x_j \rangle = h_j(x)$$

and thus we know that $h_j(x)$ gives a lower bound of the function $f(x)$.

This means if we take the maximum of several such cutting planes, we would get a lower bound of f that becomes a better approximation as we increase the number of cutting planes. Thus, if we are given points $x_0, \ldots, x_k \in X$ with subgradients $g_0, \ldots, g_k$, then we can consider the cutting plane model of the function:

$$f_k(x) = \max_{0 \leq j \leq k} h_j(x) = \max_{0 \leq j \leq k} \left\{ f(x_j) + \langle g_j, x - x_j \rangle \right\}$$

Given the above model, a natural choice is for each iterate to be the minimizer of the model generated by the previous iterates. This is given by the following linear programming problem:

$$\min_{x \in X} \max_{0 \leq j \leq k} \left\{ f(x_j) + \langle g_j, x - x_j \rangle \right\} \iff \begin{array}{ll} \min\limits_{x \in X, y \in \mathbb{R}} & y \\ \text{s.t.} & y - \langle g_j, x \rangle \geq f(x_j) - \langle g_j, x_j \rangle \end{array} \tag{3.1}$$

With the above component, we can derive the following algorithm.

Algorithm 4: Cutting Plane Method

```
Input: Initial iterate $x_0 \in X$, maximum iteration $max\_iter$
Algorithm:
$k = 0$
result = False
while $k <= max\_iter$ do
    Solve the Linear Program (3.1), and let $x_{k+1}$ be its solution
    Find $g_{k+1} \in \partial f(x_{k+1})$
    $k = k + 1$
return result
```

We show convergence of the above algorithm in the following theorem.

Theorem 3.2.1 (Convergence of Cutting Plane Method in Limit Point)**.** *If f is continuous, then any limit point of $\{x_k\}$ generated by the above cutting plane method minimizes f over X.*

Proof. We know that the model satisfies

$$\forall x \in X \text{ , } f_k(x) = \max_{0 \le j < k} \{f(x_j) + \langle g_j, x - x_j \rangle\} \le f(x) \tag{3.2}$$

Hence, for all $0 \le j < k$, we have

$$f(x_j) + \langle g_j, x_k - x_j \rangle \le f_k(x_k) \le f_k(x) \le f(x)$$

where first inequality comes from definition of maximum function, second comes from optimality of x_k, and third inequality comes from (3.2) above.

Since $f_k \le f_{k+1}$ (as we are taking maximum of more functions), we know that the minimizer also has increasing values, i.e. $f_k(x_k) \le f_{k+1}(x_{k+1})$, so $\{f_k(x_k)\}$ is a non-decreasing sequence.

Suppose $x_k \to \overline{x}$ as $k \to \infty$ in a subsequence $K \subseteq \mathbb{N}$. Set $x = \overline{x}$. Then $\{f_k(x_k)\}$ is bounded above by $f(\overline{x})$, thus converges. Moreover, for $j \to \infty$ with $k > j \in K$, we have

$$\lim_{k\to\infty}(f(x_j) + \langle g_j, x_k - x_j\rangle) \leq \lim_{k\to\infty} f_k(x_k) \leq f(\overline{x})$$

By continuity of f, we have $f(x_j) \to f(\overline{x})$, with the subgradients g_j uniformly bounded, and thus $x_k - x_j \to 0$. This implies $\lim\limits_{k\to\infty} f_k(x_k) = f(\overline{x})$, since both sides of the above inequality are equal to $f(\overline{x})$.

Since $f_k \leq f$, $f_k(x_k) \leq f(x^*)$ for optimal solution x^*, and the above implies $f(\overline{x}) \leq f(x^*)$, so we must have equality and thus $\overline{x}$ being an optimal solution. □

Remark. *The algorithm above is unstable since we are only guaranteed that the limit point of the sequence converges, yet the original sequence may not converge. To avoid this problem, we will consider a proximal version of this method in the next section.*

3.2.2 Proximal Cutting Plane Method

In this section, we consider the a proximal version of the cutting plane method above, with the intention of making the obtained sequence $\{x_k\}$ itself converge to a minimizer of f.

We start by defining some basic notions of proximal method:

Definition 3.2.2 (Moreau Envelope and Proximal Operator). *For convex function $f : \mathbb{R}^n \to \mathbb{R}$ and closed, convex set $X \subset \mathbb{R}^n$, we define its **Moreau envelope** of f at y as the minimum of the following optimization problem*

$$f_\rho(y) = \min_{x\in X}\left\{f(x) + \frac{\rho}{2}\|x - y\|^2\right\}$$

and we define the ***proximal operator*** *of f at y as one such x that achieves the minimum:*

$$prox_f(y) = \arg\min_{x\in X}\left\{f(x) + \frac{\rho}{2}\|x-y\|^2\right\}$$

where in both definitions, $\rho > 0$ is a regularization parameter.

The well-known proximal point method performs the following iteration

$$x_{k+1} = prox_f(x_k)$$

and for closed convex proper f, any fixed point of the proximal operator is also the minimizer of f, so any algorithm that finds the fixed point of this map would give the minimizer. Thus, if we have a closed form for the proximal map, then we can compute its minimizer easily.

However, for general functions the proximal operator does not have a closed form formula. In this case, we need to approximate f in the objective by a simple function that allows us to perform the proximal method.

Given the above idea, we design our algorithm via several steps:

Step 1 (Approximation of Proximal Point Method): Motivated by the cutting plane method, we can approximate f using the cutting plane approximation f_k, and this gives us an approximate iterate of the form:

$$\begin{aligned} z_{k+1} &= prox_{f_k}(x_k) = \arg\min_{x\in X}\left\{f_k(x) + \frac{\rho}{2}\|x-x_k\|^2\right\} \\ &= \arg\min_{x\in X}\left\{\max_{0\le j\le k}\{f(x_j) + \langle g_j, x-x_j\rangle\} + \frac{\rho}{2}\|x-x_k\|^2\right\} \\ &= \arg\min_{x\in X}\max_{0\le j\le k}\left\{f(x_j) + \langle g_j, x-x_j\rangle + \frac{\rho}{2}\|x-x_k\|^2\right\} \end{aligned}$$

To save computation, we note that the constraints in $f_k(x) = \max_{0\le j\le k}\{f(x_j)+\langle g_j, x-x_j\rangle\}$ are not necessarily all needed in the model. If we denote $J_k \subseteq \{0, 1, \cdots, k\}$ as the actual active constraints, then we can write

$$z_{k+1} = \arg\min_{x \in X} \max_{j \in J_k} \left\{ f(x_j) + \langle g_j, x - x_j \rangle + \frac{\rho}{2} \|x - x_k\|^2 \right\}$$

and we can further rewrite this in terms of a quadratic program:

$$\begin{aligned} \min_{x \in X, \tau \in \mathbb{R}} \quad & \tau + \frac{\rho}{2}\|x - x_k\|^2 \\ \text{s.t.} \quad & \tau \geq f(x_j) + \langle g_j, x - x_j \rangle \quad \forall j \in J_k \end{aligned} \tag{3.3}$$

Step 2 (Designing ϵ-Optimality Condition): Our next goal is to find a condition to determine that we are within ϵ of the optimal solution. To motivate this, recall that our final goal is to find some x_k with

$$f(x_k) - f(x^*) < \epsilon$$

where x^* is one solution that achieves the minimum. Since we have $f \geq f_k$, and x^* and z_{k+1} minimizes the two functions respectively, we have $f(x^*) \geq f_k(z_{k+1})$.

This implies once we have the condition

$$f(x_k) - f_k(z_{k+1}) < \epsilon$$

then we are guaranteed that we are within ϵ of the true solution.

Step 3 (Sufficient Descent Condition, Serious and Null Step): Finally, to ensure that our solution gives sufficient descent, we will check the following condition:

$$f(z_{k+1}) \leq f(x_k) - \beta\left(f(x_k) - f_k(z_{k+1})\right)$$

Note that this implies

$$f(x_k) - f(z_{k+1}) \geq \beta\left(f(x_k) - f_k(z_{k+1})\right) \geq \beta\epsilon$$

Thus, if the above condition hold, we will perform the serious step of updating $x_{k+1} = z_{k+1}$, and this will give a improvement of at least $\beta\epsilon$. Otherwise, we will keep $x_{k+1} = x_k$ and improve the approximation.

Step 4 (Constraint Update): At the end of each step, we compute $g_{k+1} \in \partial f(x_{k+1})$, and generate a new cutting plane

$$h_{k+1}(x) = f(x_{k+1}) + \langle g_{k+1}, x - x_{k+1} \rangle$$

We then update the constraint set to contain this new cutting plane, and all old cutting planes that pass through the minimum $(x_{k+1}, f_k(x_{k+1}))$, as these are the only cutting planes that may pass through the minimum of the newly generated $f_{k+1}(x)$.

Combining all the above steps give the following algorithm [9]:

Algorithm 5: Proximal Cutting Plane Method

```
Input      : Initial point $x_0 \in X$, subgradient $g_0 \in \partial F(x_0)$, proximal
             parameter $\rho$, descent parameter $\beta \in (0,1)$, precision $\epsilon > 0$
Initialize: Current index $k \leftarrow 1$, Iterate $z_0 \leftarrow x_0$, Active index set $J_0 = \{0\}$

repeat
    // Proximal Update
    Compute $z_{k+1} = \text{prox}_{f_k}(x_k)$ from quadratic programming

    // Check $\epsilon$-optimality
    if $f(x_k) - f_k(z_{k+1}) \leq \epsilon$ then
        break
    else
        // Serious and Null Step
        if $f(z_{k+1}) \leq f(x_k) - \beta(f(x_k) - f_k(z_{k+1}))$ then
            $x_{k+1} = z_{k+1}$
        else
            $x_{k+1} = x_k$

        // Update constraint set and cutting plane
        Compute subgradient $g_{k+1} \in \partial f(x_{k+1})$ using the oracle
        $J_{k+1} = \{k+1\} \cup \{j \in J_k : f(z_j) + \langle g_j, z_{k+1} - z_j \rangle = f_k(z_{k+1})\}$
        Define $f_{k+1}(x) = \max_{j \in J_{k+1}} \{f(z_j) + \langle g_j, x - z_j \rangle\}$
    $k = k + 1$
end;
return $x_k$
```

3.3 Relation between Two Formulations

In this section, we will reveal the relationship between ϵ-steepest descent with bundle iteration (Algorithm 3) and the proximal cutting plane method (Algorithm 5). The two algorithms are similar in the sense that they share the structure of two types of steps - serious steps where we have guaranteed sufficient descent, and null steps where we improve the approximation. This section gives a sketch of ideas for showing that the proximal cutting plane method is in some sense also an ϵ-steepest descent algorithm, where we keep track of iterates x_k and an approximation P_k of $Q = \partial_\epsilon f(x_k)$. This relationship suggests why these ideas combined give the name "proximal bundle method".

Step 1 (Setting up notation): We will start from the proximal cutting plane method and set up some additional notation. Let us denote the error of the cutting plane generated by x_j at target point x_k to be

$$p_j = f(x_k) - f(x_j) - \langle g_j, x_k - x_j \rangle \geq 0$$

Practically, we can rewrite this in the following form

$$f(x_k) + \langle g_j, x - x_k \rangle + p_j = f(x_j) + \langle g_j, x - x_j \rangle$$

We will also denote $d_k = x - x_k$ to be the descent direction in which we move toward the next iterate.

Recall that in the proximal cutting plane formulation, we defined $z_{k+1} = prox_{f_k}(x_k)$ to be the solution of the problem

$$\begin{aligned} \underset{x \in X, \tau \in \mathbb{R}}{\operatorname{argmin}} \quad & \tau + \frac{\rho}{2} \|x - x_k\|^2 \\ \text{s.t.} \quad & \tau \geq f(x_j) + \langle g_j, x - x_j \rangle \quad \forall j \in J_k \end{aligned}$$

Step 2 (Simplification of notation): We first rewrite the above problem in terms of error p_j and descent direction d_k.

$$\begin{array}{llcll} \underset{x\in X,\tau\in\mathbb{R}}{\operatorname{argmin}} & \tau + \frac{\rho}{2}\|x - x_k\|^2 & & \underset{d_k\in X+x_k,\tau\in\mathbb{R}}{\operatorname{argmin}} & \tau + \frac{\rho}{2}\|d_k\|^2 \\ & & \Longleftrightarrow & & \\ \text{s.t.} & \tau \geq f(x_j) + \langle g_j, x - x_j\rangle \quad \forall j \in J_k & & \text{s.t.} & \tau \geq \langle g_j, d_k\rangle + p_j \quad \forall j \in J_k \end{array}$$

where we incorporated the constant $f(x_k)$ into τ.

Step 3 (Conversion to dual problem): Now consider the Lagrangian dual problem of the right optimization problem above:

$$\max_{\lambda\geq 0} \min_{d_k\in X+x_k,\tau\in\mathbb{R}} \left\{\tau + \frac{\rho}{2}\|d_k\|^2 + \sum_{j\in J_k} \lambda_j \left(\langle g_j, d_k\rangle - p_j - \tau\right)\right\}$$

Step 4 (Completing the square): Now we can complete the square for the quadratic dual objective as

$$\begin{aligned} &\frac{\rho}{2}\left(\|d_k\|^2 + \sum_{j\in J_k} \frac{2\lambda_j}{\rho}\langle g_j, d_k\rangle\right) - \sum_{j\in J_k} \lambda_j p_j + \left(1 - \sum_{j\in J_k} \lambda_j\right)\tau \\ &= \frac{\rho}{2}\left\|d_k + \frac{1}{\rho}\sum_{j\in J_k} \lambda_j g_j\right\|^2 + \left(1 - \sum_{j\in J_k} \lambda_j\right)\tau - \frac{1}{2\rho}\left\|\sum_{j\in J_k} \lambda_j g_j\right\|^2 - \sum_{j\in J_k} \lambda_j p_j \end{aligned}$$

Step 5 (Optimal values for d_k and λ):

- From the Lagrange conditions, we know that we must have $\sum\limits_{j\in J_k} \lambda_j = 1$.
- Moreover, the optimal descent direction is $d_k = -\frac{1}{\rho}\sum\limits_{j\in J_k} \lambda_j g_j$, a weighted average of subgradients g_j.
- With the above choices, the optimization problem reduces to

$$\min_{\lambda\in\Delta_k} \left\{\sum_{j\in J_k} \lambda_j p_j + \frac{1}{2\rho}\left\|\sum_{j\in J_k} \lambda_j g_j\right\|^2\right\}$$

where $\Delta_k = \{\lambda \geq 0 : \sum\limits_{j\in J_k} \lambda_j = 1, \lambda_l = 0 \text{ for } l \notin J_k\}$.

Step 6 (Conversion back to primal problem): If we apply duality again on the above problem, we get

$$\min_{\lambda \in \Delta_k} \left\{ \left\| \sum_{j \in J_k} \lambda_j g_j \right\| : \sum_{j \in J_k} \lambda_j p_j \leq \epsilon \right\}$$

for some $\epsilon > 0$ that depends on ρ. This means if we define

$$P_\epsilon^k = \left\{ \sum_{j \in J_k} \lambda_j g_j : \lambda \in \Delta_k, \sum_{j \in J_k} \lambda_j p_j \leq \epsilon \right\}$$

Then we have $-\rho d_k \in P_\epsilon^k$ being the shortest vector in P_ϵ^k, so $z_{k+1} = x_k + d_k$ is indeed a point that can be picked from line search in the opposite direction of $-\rho d_k$. Thus, each z_k can be thought of as generated via a line search in the direction d_k, which is the shortest vector in the approximation P_ϵ^k of $\partial_\epsilon f(x_k)$.

Step 7 (Containment of P_ϵ^k in ϵ-subdifferential): we finally verify the assumption $P_\epsilon^k \subseteq \partial_\epsilon(x_k)$ used in the analysis of ϵ-steepest descent of bundle iteration.

- Since $g_j \in \partial f(x_j)$, we know that $\forall x \in X$, we have $f(x) - f(x_j) \geq \langle g_j, x - x_j \rangle$, so

$$f(x_k) - f(x) - \langle g_j, x_k - x \rangle \leq f(x_k) - f(x_j) - \langle g_j, x_k - x_j \rangle$$

- Now suppose $y \in P_\epsilon^k$, then we can write $y = \sum_{j \in J_k} \lambda_j g_j$ for some $\lambda \in \Delta_k$ with $\sum_{j \in J_k} \lambda_j p_j \leq \epsilon$. This implies

$$\begin{aligned} f(x_k) - f(x) - \langle y, x_k - x \rangle &= f(x_k) - f(x) - \left\langle \sum_{j \in J_k} \lambda_j g_j, x_k - x \right\rangle \\ &= \sum_{j \in J_k} \lambda_j \left(f(x_k) - f(x) - \langle g_j, x_k - x \rangle \right) \\ &\leq \sum_{j \in J_k} \lambda_j \left(f(x_k) - f(x_j) - \langle g_j, x_k - x_j \rangle \right) = \sum_{j \in J_k} \lambda_j p_j \leq \epsilon \end{aligned}$$

  and thus $\langle y, x - x_k \rangle \leq f(x) - f(x_k) + \epsilon$, so $y \in \partial_\epsilon f(x_k)$. This shows $P_\epsilon^k \subseteq \partial_\epsilon f(x_k)$.

3.4 Experiment: Min-Max Problem with Quadratic Objectives

In this section, we verify the empirical performance of the proximal bundle method by comparing it to several other baseline methods for nonsmooth optimization, specifically the subgradient method, prox-linear method and the BFGS method.

We will look at a specific example of optimization problem, given by the minimization of the maximum of a set of quadratic functions:

$$\min_{x \in \mathbb{R}^n} f(x) \text{ for } f(x) = \max_{1 \le i \le m} f_i(x) = \max_{1 \le i \le m} \left\{ \frac{1}{2} x^\top A_i x + b_i^\top x \right\}$$

where each $A_i \in \mathbb{R}^{n \times n}$ is positive semidefinite, and each $b_i \in \mathbb{R}^n$ is a positive vector.

Our first goal is to design an oracle for the subgradient of this objective, as this will be used by the optimization method. Specifically, we have

$$f_i(x) = \frac{1}{2} x^\top A_i x + b_i^\top x \Rightarrow \nabla f_i(x) = A_i x + b_i$$

and thus we can design the oracle for $\partial f(x)$ as follows: The oracle first finds one index $1 \le i \le m$ that maximizes $x^\top A_i x + b_i^\top x$ (if there are many, it will output an arbitrary one). It then outputs the subgradient as $\nabla f_i(x) = A_i x + b_i$.

We now briefly describe several baseline methods that will be compared with the proximal bundle method.

3.4.1 Baseline Method: Subgradient Method

We start by looking at one of the simplest nonsmooth optimization optimization algorithms, the subgradient descent algorithm, with the iteration

$$x_{k+1} = x_k - \alpha_k g_k$$

for $g_k \in \partial f(x_k)$ being a subgradient. Here we will choose the step size based on the nonsummable diminishing step length rule: $\alpha_k = \frac{\gamma_k}{||g_k||_2}$ with $\gamma_k = \frac{1}{k}$.

Algorithm 6: Subgradient Method for Min-Max Problem

```
Input: Initial iterate $x_0 \in X$, maximum iteration $max\_iter$
Algorithm: $k = 0$
while $k <= max\_iter$ do
    $x_{k+1} = x_k - \frac{1}{k||g_k||_2} g_k$
    $k = k + 1$
return $x_k$
```

3.4.2 Baseline Method: Prox-linear Method

The prox-linear method [5] is an algorithm that is used to solve composite optimization problems, of the form

$$\inf_{x \in \mathbb{R}^n} g(H(n))$$

where $H : \mathbb{R}^n \to \mathbb{R}^m$ being C^2-smooth and $g : \mathbb{R}^m \to \mathbb{R}$ convex and finite.

The algorithm is an iterative method that performs the following update:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \left\{ g\left(H(x_k) + \nabla H(x_k)^\top (x - x_k)\right) + \frac{\lambda}{2} ||x - x_k||^2 \right\}$$

To apply this to the min-max problem, we will define

$$g\begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \max_{i=1,\cdots,m} x_i, H(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

Then our prox-linear algorithm would simplify to

$$\begin{aligned} x_{k+1} &= \arg\min_x \left\{ g\begin{pmatrix} f_1(x_k) + \nabla f_1(x_k)^\top (x - x_k) \\ \vdots \\ f_m(x_k) + \nabla f_m(x_k)^\top (x - x_k) \end{pmatrix} + \frac{\lambda}{2}\|x - x_k\|^2 \right\} \\ &= \arg\min_x \left\{ \max_{i=1,\cdots,m} \{f_i(x_k) + \nabla f_i(x_k)^\top (x - x_k)\} + \frac{\lambda}{2}\|x - x_k\|^2 \right\} \\ &= \arg\min_x \max_{i=1,\cdots,m} \left\{ f_i(x_k) + \nabla f_i(x_k)^\top (x - x_k) + \frac{\lambda}{2}\|x - x_k\|^2 \right\} \end{aligned}$$

Thus, each iteration reduces to solving a quadratic program

$$\begin{aligned} &\min_x \ t + \frac{\lambda}{2}\|x - x_k\|^2 \\ &\text{s.t. } t \geq f_i(x_k) + \nabla f_i(x_k)^\top (x - x_k) \text{ for } i = 1, \cdots, m \end{aligned} \tag{3.4}$$

And this gives the following algorithm:

Algorithm 7: Prox-linear Method for Min-Max Problem

```
Input: Initial iterate $x_0 \in X$, maximum iteration $max\_iter$
Algorithm: $k = 0$
while $k <= max\_iter$ do
    Solve the Quadratic Program (3.4), and let $x_{k+1}$ be its solution
    $k = k + 1$
return $x_k$
```

3.4.3 Baseline Method: BFGS Method

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [7] is an iterative method for solving nonlinear optimization algorithms. Here we use a nonsmooth version of the algorithm proposed in [4].

Algorithm 8: BFGS Method for Min-Max Problem

```
Input: Initial point $x_0 \in \mathbb{R}^n$, initial hessian $B_0 \in \mathbb{R}^{n\times n}$, constant
 $c_1, c_2 \in [0, 1]$, maximum iteration $max\_iter$
Algorithm: $k = 0$
while $k <= max\_iter$ do
    $p_k = -B_k \nabla f(x_k)$
    Find approximate solution $t_k$ to $\min_{t>0}\{f(x_k + tp_k)\}$ via line search.
    $s_k = t_k p_k$
    if $f$ is not differentiable at $x_{k+1}$ or $\nabla f_{k+1} = 0$ then
        break
    else
        $s_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
        $B_{k+1} = B_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{B_k s_k s_k^\top B_k^\top}{s_k^\top B_k s_k}$
    $k = k + 1$
return $x_k$
```

Here, we used an modified line search procedure, based on the Armijo-Wolfe conditions: For constants $c_1, c_2 \in [0, 1]$, let

$$h(t) = f(x_k + tp_k) - f(x_k)\ , \ s = \limsup_{t\to 0} \frac{h(t)}{t} < 0$$

Then we want our approximate condition to satisfy

- (1) **Armijo Condition**: $h(t) < c_1 st$
- (2) **Wolfe Condition**: h is differentiable at t with $h'(t) > c_2 s$

and these are reflected in the following line search algorithm:

Algorithm 9: BFGS Line Search

```
$\alpha = 0$
$\beta = +\infty$
$t = 1$
repeat
    if $h(t) \geq c_1 st$ then
        $\beta = t$
    else if $h'(t) \leq c_2 s$ then
        $\alpha = t$
    else
        break
    if $\beta < +\infty$ then
        $t = \frac{\alpha+\beta}{2}$
    else
        $t = 2\alpha$
until;
```

3.4.4 Proximal Bundle Method

Our implementation of proximal bundle method follows directly from the proximal cutting plane algorithm in Algorithm 5.

3.4.5 Comparison of Results

Again, recall that our goal is to solve the problem

$$\min_{x \in \mathbb{R}^n} f(x) \text{ for } f(x) = \max_{1 \leq i \leq m} f_i(x) = \max_{1 \leq i \leq m} \left\{ \frac{1}{2} x^\top A_i x + b_i^\top x \right\}$$

where each $A_i \in \mathbb{R}^{n \times n}$ is positive semidefinite, and each $b_i \in \mathbb{R}^n$ is a positive vector.

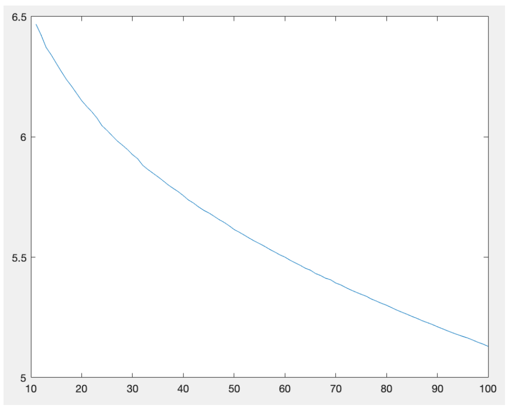
In the experiments below, we pick $n = 100$, $m = 20$, and thus consider maximum of 20 quadratic objectives with matrices and vectors of size $A_i \in \mathbb{R}^{100\times 100}$ and $b_i \in \mathbb{R}^{100}$. Moreover, we will assume that $\sum_{i=1}^{20} b_i = 0$, to ensure that the minimum is achieved at $x = 0$ with optimal value 0. We then run the prox-linear method, the BFGS method, and the proximal bundle method, each for a maximum of 100 iterations.

The results are shown in the following table. Graphs of convergence (semi-log plots of the objective value between the 11-th and 100-th iterate) are shown on the next page.

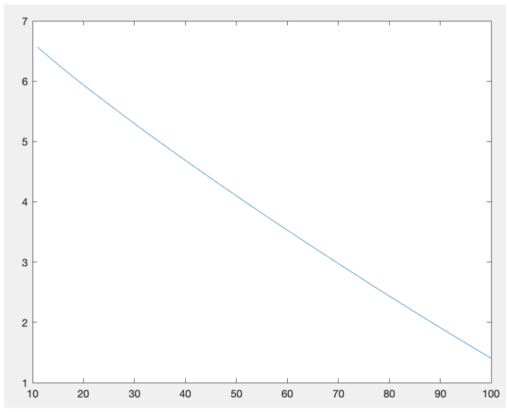
Method	Optimal Value	Time Spent (s)
Subgradient method	$1.690 * 10^2$	0.188
Prox-linear method	$4.070 * 10^0$	88.116
BFGS	$9.174 * 10^{-1}$	7.933
Proximal bundle method	$1.123 * 10^{-4}$	213.625

Table 3.2: Table of Results for Running the Methods on Min-Max Problem

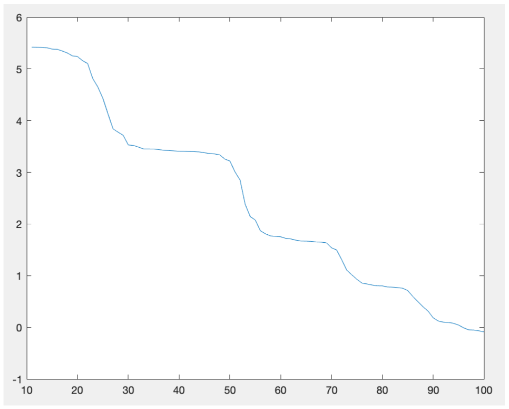
Figure 3.3: Experimental Results on Min-max Problem



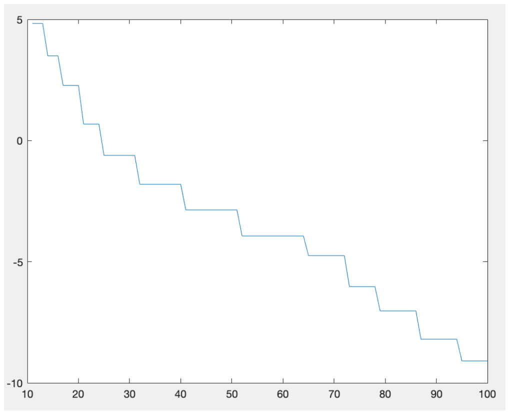
(a) Semi-log plot of convergence of subgradient method. While not shown in the first 100 iterations, the convergence becomes significantly slower later on.



(b) Semi-log plot of convergence of proxlinear method. The graph shows linear convergence that is faster than subgradient method.



(c) Semi-log plot of convergence of BFGS method. The flat and steep regions represent the local property of the function on the optimization trajectory.



(d) Semi-log plot of convergence of proximal bundle method. Here the horizontal regions represent the null steps, and the steep regions represent the serious steps.

From the above results, we can see that the proximal bundle method gives the best result under a fixed number of iterations, though it is typically much slower than the other three methods. If we look deeper into the implementation of the proximal bundle method, we can see that the number of constraints in the quadratic program solved by the proximal bundle method may grow linearly with the number of iterations. Thus, the proximal bundle method may become significantly slower when the number of iterations become larger.

This phenomenon motivates us to explore the following question: Assuming that the later iterates of proximal bundle method may be hard to compute, is it possible for us to extrapolate more information from a finite set of early iterates of the proximal bundle method (instead of just keeping track of the last iterate)?

In the next section, we would explore a recent line of work on nonlinear acceleration methods, an acceleration scheme that finds the best linear combination of past iterates to improve the convergence rate, and think about how this may be used to improve the proximal bundle method.

CHAPTER 4

NONLINEAR ACCELERATION

In this chapter, we describe the regularized nonlinear acceleration algorithm [10], a general acceleration scheme that accelerate a general optimization algorithm based on the best linear combination of past iterates.

Section 4.1 starts by introducing our problem setting. In Section 4.2, we first introduce the nonlinear acceleration algorithm without regularization, and we proceed to describe the regularized version in Section 4.3. The final short Section 4.4 is a short discussion on the feasibility and difficulty of applying this acceleration scheme to the proximal bundle method.

4.1 Problem Setting

We start by defining the setting of nonlinear acceleration. Consider a general unconstrained optimization problem $\min_{x \in \mathbb{R}^n} f(x)$, for $f : \mathbb{R}^n \to \mathbb{R}$ strongly convex and Lipschitz continuous.

Our goal is to analyze a black-box algorithm $g : \mathbb{R}^n \to \mathbb{R}^n$ whose fixed-point is the minimum of f. Specifically, we assume the following properties:

- The fixed point x^* of g is unique.
- $\nabla g(x^*)$ exists and is symmetric.
- $\nabla g(x^*) \preceq \sigma I$ for some $\sigma < 1$.

To analyze the behavior of g as an iteration map, we would consider its first-order Taylor approximation around x^*:

$$\begin{aligned}\tilde{x}_{k+1} &= g(\tilde{x}_k) = g(x^*) + \nabla g(x^*)(\tilde{x}_k - x^*) + e_k \\ &= x^* + \nabla g(x^*)(\tilde{x}_k - x^*) + e_k\end{aligned}$$

where $e_k = O(\|x_k - x^*\|^2)$ is the second-order error.

In the next section, we will start simple and analyze properties of the linear approximation

$$x_{k+1} = x^* + \nabla g(x^*)(x_k - x^*).$$

4.2 Nonlinear Acceleration for Linear Iterates

As motivated above, we will look at the linear iteration

$$x_{k+1} = x^* + \nabla g(x^*)(x_k - x^*)$$

We start by analyzing the convergence rate of the sequence $\{x_k\}$ generated by this iteration. Let us denote $G = \nabla g(x^*)$. Then for any $i \in \mathbb{N}$, we have

$$x_{i+1} - x^* = G(x_i - x^*) \Rightarrow x_i - x^* = G^i(x_0 - x^*)$$

Thus, the original sequence $\{x_i\}$ satisfies

$$\|x_i - x^*\| \leq \|G\|^i \|x_0 - x^*\| \leq \sigma^i \|x_0 - x^*\|$$

and thus converges linearly with rate σ.

We then consider a linear combination $\sum_{i=0}^{k} c_i x_i$ of the iterates up to step k. The error can be written in the form

$$\sum_{i=0}^{k} c_i x_i = \left(\sum_{i=0}^{k} c_i\right) x^* + \left(\sum_{i=0}^{k} c_i G^i\right)(x_0 - x^*)$$

If we write $p(z) = \sum_{i=0}^{k} c_i z^i$ and assume $\sum_{i=0}^{k} c_i = 1$, then we have

$$\sum_{i=0}^{k} c_i x_i - x^* = \left(\sum_{i=0}^{k} c_i G^i\right)(x_0 - x^*) = p(G)(x_0 - x^*)$$

Thus, to minimize the norm of the LHS, we just need to find

$$\left\|\sum_{i=0}^{k} c_i^* x_i - x^*\right\| = \min_{c \in \mathbb{R}^{k+1}: c^\top \mathbf{1} = 1} \left\|\left(\sum_{i=0}^{k} c_i G^i\right)(x_0 - x^*)\right\| = \min_{p \in \mathbb{R}_k[x]: p(1)=1} \|p(G)(x_0 - x^*)\| \quad (4.1)$$

We now state a theorem on the convergence rate of the best linear combination iterate c^*.

Theorem 4.2.1 (Convergence Guarantee for Nonlinear Acceleration on Linear Iteration). *Let $c^* \in \mathbb{R}^{k+1}$ be the optimal solution for the minimization problem in equation (4.1). Assume that m is the number of distinct eigenvalues of G, and $k < m$. We then have the following convergence bound:*

$$\left\|\sum_{i=0}^{k} c_i^* x_i - x^*\right\| \leq \frac{2\beta^k}{1 + \beta^{2k}} \|x_0 - x^*\|$$

where

$$\beta = \frac{1 - \sqrt{1 - \sigma}}{1 + \sqrt{1 - \sigma}}$$

Proof. **Step 1 (Conversion to min-max problem)**: Consider the eigenvalue decomposition $G = Q^* \Lambda Q$, then we have

$$\begin{aligned} \|p(G)(x_0 - x^*)\|_2 &= \|Q^* p(\Lambda) Q(x_0 - x^*)\|_2 \\ &\leq \|p(\Lambda)\|_2 \, \|x_0 - x^*\|_2 = \max_{1 \leq i \leq m} |p(\lambda_i)| \cdot \|x_0 - x^*\|_2 \end{aligned}$$

Since $0 \leq G \leq \sigma$, we know that $0 \leq \lambda_i \leq \sigma$. Thus,

$$\min_{p \in \mathbb{R}_k[x]: p(1)=1} \|p(G)(x_0 - x^*)\|_2 \leq \min_{p \in \mathbb{R}_k[x]: p(1)=1} \max_{\lambda \in [0, \sigma]} |p(\lambda)| \cdot \|x_0 - x^*\|_2$$

Step 2 (Deriving optimal convergence rate from Chebyshev polynomials): The above problem reminds us of the definition of Chebyshev polynomials:

$$C_k(x) = \underset{p \in \mathbb{R}_k[x]:p(1)=1}{\arg\min} \max_{x \in [-1,1]} |p(x)|$$

Indeed, we can rescale x from $[-1, 1]$ to $[0, \sigma]$, and define

$$T_k(x, \sigma) = \frac{C_k(t(x, \sigma))}{C_k(t(1, \sigma))} \text{ where } t(x, \sigma) = \frac{2x - \sigma}{\sigma}$$

then T_k would solve our shifted problem, with

$$\min_{p \in \mathbb{R}_k[x]:p(1)=1} \max_{\lambda \in [0,\sigma]} |p(\lambda)| = \max_{\lambda \in [0,\sigma]} |T_k(\lambda, \sigma)| = |T_k(\sigma, \sigma)| = \frac{2\beta^k}{1 + \beta^{2k}}$$

with β defined as in the theorem. □

To give some intuition on the new convergence rate $\beta = \frac{1-\sqrt{1-\sigma}}{1+\sqrt{1-\sigma}}$ versus the original convergence rate σ, we plot the values of them as σ increases. We see that the new convergence rate is always better than the old one, thus verifying that we indeed get acceleration via the method.

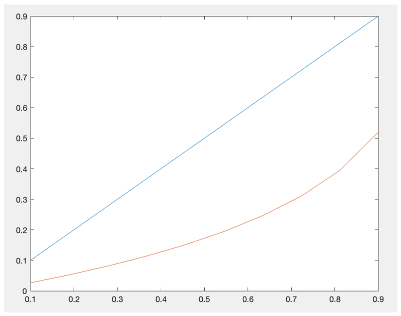


Figure 4.1: Comparing the two convergence rates. Blue line plots σ and orange line plots β. We can see that we always have $\beta < \sigma$.

We finally describe how we may compute the optimal weights c^* - in fact, this can be done using simple matrix operations instead of the Chebyshev polynomials. To do this, we recall that we would like to solve

$$\left\|\sum_{i=0}^{k} c_i^* x_i - x^*\right\| = \min_{c\in\mathbb{R}^{k+1}:c^\top\mathbf{1}=1} \left\|\left(\sum_{i=0}^{k} c_i G^i\right)(x_0 - x^*)\right\|$$

If we denote $r_i = x_{i+1} - x_i$, then we have

$$r_i = x_{i+1} - x_i = (G - I)G^i(x_0 - x^*)$$

and thus we have

$$\begin{aligned}\left\|\sum_{i=0}^{k} c_i^* x_i - x^*\right\| &= \min_{c\in\mathbb{R}^{k+1}:c^\top\mathbf{1}=1} \left\|\left(\sum_{i=0}^{k} c_i G^i\right)(x_0 - x^*)\right\| \\ &= \min_{c\in\mathbb{R}^{k+1}:c^\top\mathbf{1}=1} \left\|(G-I)^{-1}\left(\sum_{i=0}^{k} c_i r_i\right)\right\| \\ &\leq \|(G-I)^{-1}\| \min_{c\in\mathbb{R}^{k+1}:c^\top\mathbf{1}=1} \|Rc\|\end{aligned}$$

Thus, we know that the optimal c satisfies

$$c^* = \operatorname*{arg\,min}_{c\in\mathbb{R}^{k+1}:c^\top\mathbf{1}=1} \|Rc\|_2$$

To find an explicit solution, we apply Lagrange multipliers. Let

$$g(c,\mu) = c^\top R^\top Rc + \mu(1 - \mathbf{1}^\top c)$$

Then we have

$$\nabla_c g(c,\mu) = 2R^\top Rc - \mu\mathbf{1} = 0 \Rightarrow c = \frac{\mu}{2}(R^\top R)^{-1}\mathbf{1}$$

Now to ensure $1^\top c = 1$, we just need to normalize our value of c, so we have

$$c = \frac{z}{\sum_{i=0}^{k} z_i} \text{ where } z = (R^\top R)^{-1}\mathbf{1}$$

Alternatively, we can write

$$c^* = \frac{(R^\top R)^{-1}\mathbf{1}}{\mathbf{1}^\top(R^\top R)^{-1}\mathbf{1}}$$

Thus, we can compute the best k-th iterate $x^* = \sum_{i=0}^{k} c_i^* x_i$ via the following algorithm:

Algorithm 10: Nonlinear Acceleration

```
**Requires:** Iterates $x_0, \dots, x_{k+1}$;
**Algorithm:**
    $r_i = x_{i+1} - x_i$, $R = [r_0, \dots, r_k]$
    $z = (R^\top R)^{-1}\mathbf{1}$        $c = z / \left(\sum_{i=0}^{k} z_i\right)$ **Output:** $x^* = \sum_{i=0}^{k} c_i x_i$
```

4.3 Regularized Nonlinear Acceleration for Nonlinear Iterates

In this section, we would like to analysis how the above nonlinear acceleration algorithm works on nonlinear problems. Numerical issues that are revealed in the analysis would then lead us to consider a regularized version of nonlinear acceleration for general problems.

4.3.1 Sensitivity Analysis of Nonlinear Acceleration

Let us set $\tilde{x}_0 = x_0$, and consider two sequences $\{\tilde{x}_k\}$ and $\{x_k\}$, generated by the original iteration map and linear approximation:

$$\tilde{x}_{k+1} = g(\tilde{x}_k)$$

$$x_{k+1} = x^* + \nabla g(x^*)(x_k - x^*)$$

We will also use the following notations:

$$r_i = x_{i+1} - x_i \ , \tilde{r}_i = \tilde{x}_{i+1} - \tilde{x}_i$$

$$R = [r_0, \dots, r_k] \in \mathbb{R}^{n\times(k+1)}, \tilde{R} = [\tilde{r}_0, \dots, \tilde{r}_k] \in \mathbb{R}^{n\times(k+1)}$$

$$P = \tilde{R}^\top \tilde{R} - R^\top R$$

If we apply nonlinear acceleration to the two sequences, we will get respectively

$$\tilde{c}^* = \frac{(\tilde{R}^\top \tilde{R})^{-1}\mathbf{1}}{\mathbf{1}^\top (\tilde{R}^\top \tilde{R})^{-1}\mathbf{1}}, c^* = \frac{(R^\top R)^{-1}\mathbf{1}}{\mathbf{1}^\top (R^\top R)^{-1}\mathbf{1}}$$

We now state a theorem in [10, Proposition 3.1] that allows us to bound the difference $\|\tilde{c}^* - c^*\|$.

Theorem 4.3.1 (Sensitivity of Nonlinear Acceleration). *With the above notation, we have the following bound:*

$$\|\tilde{c}^* - c^*\| \leq \|P\| \cdot \|(R^\top R + P)^{-1}\| \cdot \|c^*\|$$

Now we note that the residual matrix R has the Krylov form

$$R = [r_0, Gr_0, \dots, G^k r_0]$$

and thus the condition number of R grows exponentially as k increases, and so does its perturbation $\|(R^\top R + P)^{-1}\|$. Thus, directly applying nonlinear acceleration on g would give very different results than on the linear approximation, and thus may lead to poor convergence bounds. This motivates us to use a regularization scheme to ensure the norm $\|\tilde{c}^* - c^*\|$ does not become arbitrarily large.

4.3.2 Regularized Nonlinear Acceleration

We now consider adding an additional regularization term to the original problem in equation (4.2):

$$c^* = \underset{c\in\mathbb{R}^{k+1}: c^\top \mathbf{1}=1}{\arg\min} \|\tilde{R}c\|_2^2 + \lambda\|c\|_2^2$$

To compute its solution, we again find its Lagrange multiplier:

$$g(c,\mu) = c^\top \tilde{R}^\top \tilde{R} c + \lambda c^\top c + \mu(1 - \mathbf{1}^\top c)$$

Then we have

$$\nabla_c g(c,\mu) = 2\tilde{R}^\top \tilde{R} c + 2\lambda c - \mu \mathbf{1} = 0 \Rightarrow c = \frac{\mu}{2}(\tilde{R}^\top \tilde{R} + \lambda I)^{-1}\mathbf{1}$$

Now to ensure $\mathbf{1}^\top c = 1$, we just need to normalize our value of c, so we have

$$c = \frac{z}{\sum\limits_{i=0}^{k} z_i} \text{ where } z = (\tilde{R}^\top \tilde{R} + \lambda I)^{-1}\mathbf{1}$$

Alternatively, we can write

$$\tilde{c}^*_\lambda = \frac{(\tilde{R}^\top \tilde{R} + \lambda I)^{-1}\mathbf{1}}{\mathbf{1}^\top (\tilde{R}^\top \tilde{R} + \lambda I)^{-1}\mathbf{1}}$$

and this leads to the following algorithm:

```
Algorithm 11: Regularized Nonlinear Acceleration
  Requires: Iterates $\tilde{x}_0, \ldots, \tilde{x}_{k+1}$;
  Algorithm:
      $\tilde{r}_i = \tilde{x}_{i+1} - \tilde{x}_i$, $R = [\tilde{r}_0, \ldots, \tilde{r}_k]$
      $z = (\tilde{R}^\top \tilde{R} + \lambda I)^{-1}\mathbf{1}$;
      $c = z / \left( \sum_{i=0}^{k} z_i \right)$;
  Output: $x^* = \sum_{i=0}^{k} c_i x_i$.
```

We now present some results from [10] on analysis of this algorithm without proof. First, [10, Proposition 3.2] gives the effect of the regularization term on the norm of $\|\tilde{c}^* - c^*\|$.

Theorem 4.3.2 (Upper Bound on Length of Weight Vector from Regularization)**.** *We have the following upper bounds on norm of weight and difference:*

$$\|\tilde{c}^*_\lambda\| \leq \sqrt{\frac{\lambda + \|\tilde{R}\|^2}{(k+1)\lambda}}, \|\tilde{c}^*_\lambda - c^*_\lambda\| \leq \frac{\|P\|}{\lambda}\|c^*_\lambda\|$$

Finally, [10, Proposition 3.5] gives a convergence guarantee for the regularized nonlinear acceleration, based on decomposition of error into several components and bounding them one by one:

Theorem 4.3.3 (Convergence Guarantee for Regularized Nonlinear Acceleration)**.** *We can decompose the error into the following:*

$$\sum_{i=0}^{k}(\tilde{c}_\lambda^*)_i\tilde{x}_i - x^* = \underbrace{\left(\sum_{i=0}^{k}(c_\lambda^*)_i x_i - x^*\right)}_{\textit{Linear case bound}} + \underbrace{\left(\sum_{i=0}^{k}(\tilde{c}_\lambda^* - c_\lambda^*)_i x_i\right)}_{\textit{Stability}} + \underbrace{\left(\sum_{i=0}^{k}(\tilde{c}_\lambda^*)(\tilde{x}_i - x_i)\right)}_{\textit{Nonlinearity}}$$

Let $\overline{x}$ be an arbitrary point in $\mathbb{R}^n$, and we denote $\overline{X}, \mathcal{E} \in \mathbb{R}^{n\times(k+1)}$ with $\overline{X}_i = x_i - \overline{x}$, $\mathcal{E}_i = \tilde{x}_i - x_i$. Let $\kappa = \frac{1}{1-\sigma}$, and S_σ be the maximum of regularized Chebyshev polynomial, defined by

$$C_\sigma^*(x,k,\alpha) = \underset{C\in\mathbb{R}_k[x]:C(1)=1}{\arg\min}\ \max_{x\in[0,\sigma]} C^2(x) + \alpha\|C\|^2$$

$$S_\sigma(k,\alpha) = \sqrt{\max_{x\in[0,\sigma]}(C_\sigma^*(x,k,\alpha))^2 + \alpha\|C_\sigma^*(x,k,\alpha)\|^2}$$

then we have the following bounds:

$$\left\|\sum_{i=0}^{k}(c_\lambda^*)_i x_i - x^*\right\| \le \kappa\sqrt{S_\sigma^2(k,\overline{\lambda})\|x_0 - x^*\|^2 - \lambda\|c_\lambda^*\|^2}$$

$$\left\|\sum_{i=0}^{k}(\tilde{c}_\lambda^* - c_\lambda^*)_i x_i\right\| \le \|\overline{X}\|\frac{\|P\|}{\lambda}\|c_\lambda^*\|$$

$$\left\|\sum_{i=0}^{k}(\tilde{c}_\lambda^*)(\tilde{x}_i - x_i)\right\| \le \frac{\|\mathcal{E}\|}{\sqrt{k+1}}\sqrt{1 + \frac{\|\tilde{R}\|^2}{\lambda}}$$

and combining them gives the upper bound on error for regularized nonlinear acceleration.

4.4 Discussions and Future Work

This section provides some discussion on how we may think about applying nonlinear acceleration on the proximal bundle method.

4.4.1 Nonlinear Acceleration of Proximal Methods

First, we observe that for the proximal bundle method, we need to apply the nonlinear acceleration method on the serious steps only, since these are the steps that actually update x_k.

However, a direct application of the nonlinear acceleration algorithm will likely not work, since each serious step in the proximal bundle method comes from a different iteration map:

$$x_{k+1} = g_k(x_k) = prox_{f_k}(x_k)$$

where f_k is the current approximation model of the function f, that differs in each serious step. Since the nonlinear acceleration method depends on the iteration map being fixed (and that is how we can use an argument based on powers of gradient and polynomials), we cannot directly apply the method.

One way to ensure that the iteration map is the same throughout the process is for us to take a step back and consider accelerating the original proximal point method without approximation:

$$x_{k+1} = g(x_k) = prox_f(x_k)$$

Beyond the brief discussion in [2], one interesting idea in this direction comes from the paper [6], which applies nonlinear acceleration on the specific instance of proximal gradient method. We will describe the idea of this below:

Recall that the proximal gradient method is an optimization algorithm that solves the optimization problem

$$\min_{x \in \mathbb{R}^n} \{f(x) + g(x)\}$$

with f smooth and L-Lipschitz, and g proper closed and convex. The proximal gradient method computes the following recursively:

$$x_{k+1} = prox_{\lambda g}(x_k - \lambda \nabla f(x_k))$$

We would like to find the fixed point to either of the following:

$$h_1(x) = prox_{\lambda g}(x - \lambda \nabla f(x))$$

The authors in [6] proposed that instead of applying nonlinear acceleration directly to the original iteration, it is easier to rewrite the iteration as

$$\begin{cases} y_{k+1} = x_k - \lambda \nabla f(x_k) \\ x_{k+1} = prox_{\lambda g}(y_{k+1}) \end{cases}$$

and then consider applying nonlinear acceleration on $\{y_k\}$ instead, to find the fixed point of the iteration map

$$h_2(y) = prox_{\lambda g}(y) - \lambda \nabla f(prox_{\lambda g}(y))$$

While this method does not seem to extend naturally to general proximal point methods, it may be possible to apply a similar idea to other optimization algorithms that has an iteration map that alternates between multiple components.

4.4.2 Manifold Approximation of Proximal Methods

Another idea that may be interesting for future work is analyze how proximal methods (or other nonsmooth optimization algorithms) can be approximated by simple optimization algorithms on a local manifold. The idea is motivated by the following example of proximal point method:

Example (Example of proximal point method on nonsmooth problem)**.** *In this example, we will consider the optimization problem*

$$\min_{x\in\mathbb{R}^2} f(x) \textit{ with } f\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = |u| + v^2$$

This problem has the global minimum 0 *achieved at* $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$*. Our goal is to apply the proximal point method and see its behavior on this problem.*

Step 1: We first simplify the proximal point method computation. *Let* $\lambda > 0$ *be the proximal parameter, and* $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ *be the initial iterate. We will denote* $x_k^\lambda = \begin{bmatrix} u_k^\lambda \\ v_k^\lambda \end{bmatrix}$ *to be the k-th iterate. We then have*

$$\begin{aligned}\begin{bmatrix} u_{k+1}^\lambda \\ v_{k+1}^\lambda \end{bmatrix} = x_{k+1}^\lambda &= \min_{x\in\mathbb{R}^2} \left\{ f(x) + \frac{\lambda}{2}\|x - x_k^\lambda\|^2 \right\} \\ &= \min_{u,v\in\mathbb{R}} \left\{ |u| + v^2 + \frac{\lambda}{2}(u - u_k^\lambda)^2 + \frac{\lambda}{2}(v - v_k^\lambda)^2 \right\}\end{aligned}$$

Now we can split the above problem into minimizing u and v separately.

Step 2: We then solve the problem for u***.***

$$u_{k+1}^\lambda = \min_u \left\{ |u| + \frac{\lambda}{2}(u - u_k^\lambda)^2 \right\} \Rightarrow 0 \in \partial(|u|)(u_{k+1}^\lambda) + \lambda(u_{k+1}^\lambda - u_k^\lambda)$$

From subdifferential computation of $g(u) = |u|$*, we have*

$$\begin{cases} \lambda(u_{k+1}^\lambda - u_k^\lambda) = 1 \text{ , } \textit{if } u_{k+1}^\lambda > 0 \\ \lambda(u_{k+1}^\lambda - u_k^\lambda) \in [-1, 1] \text{ , } \textit{if } u_{k+1}^\lambda = 0 \\ \lambda(u_{k+1}^\lambda - u_k^\lambda) = -1 \text{ , } \textit{if } u_{k+1}^\lambda < 0 \end{cases}$$

and thus

$$u_{k+1}^{\lambda} = \begin{cases} u_k^{\lambda} - \frac{1}{\lambda} \text{ , } \textit{if } u_k^{\lambda} > \frac{1}{\lambda} \\ 0 \text{ , } \textit{if } u_k^{\lambda} \in [-\frac{1}{\lambda}, \frac{1}{\lambda}] \\ u_k^{\lambda} + \frac{1}{\lambda} \text{ , } \textit{if } u_k^{\lambda} < \frac{1}{\lambda} \end{cases}$$

Step 3: We now solve the problem for** v**.

$$v_{k+1}^{\lambda} = \min_{v} \left\{ v^2 + \frac{\lambda}{2}(v - v_k^{\lambda})^2 \right\} \Rightarrow 2v_{k+1}^{\lambda} + \lambda(v_{k+1}^{\lambda} - v_k^{\lambda}) = 0$$

$$\Rightarrow v_{k+1}^{\lambda} = \frac{\lambda}{\lambda + 2} v_k^{\lambda}$$

Step 4: We finally combine all of our results. *We have the proximal point sequence* $x_k^{\lambda} = \begin{bmatrix} u_k^{\lambda} \\ v_k^{\lambda} \end{bmatrix}$ *is given by the recursion formulas*

$$u_{k+1}^{\lambda} = \begin{cases} u_k^{\lambda} - \frac{1}{\lambda} & \text{, } \textit{if } u_k^{\lambda} > \frac{1}{\lambda} \\ 0 & \text{, } \textit{if } u_k^{\lambda} \in [-\frac{1}{\lambda}, \frac{1}{\lambda}] \\ u_k^{\lambda} + \frac{1}{\lambda} & \text{, } \textit{if } u_k^{\lambda} < \frac{1}{\lambda} \end{cases} \text{ , } \textit{and } v_{k+1}^{\lambda} = \frac{\lambda}{\lambda + 2} v_k^{\lambda}$$

Now if we plug in $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$*, we have*

$$x_k^{\lambda} = \begin{bmatrix} \max\{1 - \frac{k}{\lambda}, 0\} \\ (\frac{\lambda}{\lambda+2})^k \end{bmatrix}$$

Analysis of Iterates*: Now if we look at the above iterates, we see that in the first* $\lceil \frac{\lambda}{k} \rceil$ *iterations, it will take steps that move in both coordinates. After that, the u-coordinate becomes* 0 *and stays there, while we continue to move the v-coordinate to become closer to the optimum.*

Thus, after $\lceil \frac{\lambda}{k} \rceil$ iterations, we are essentially minimizing the function on the 1-*dimensional subspace*

$$V = \left\{ \begin{bmatrix} 0 \\ v \end{bmatrix} : v \in \mathbb{R} \right\}$$

where the function becomes $f\left(\begin{bmatrix} 0 \\ v \end{bmatrix}\right) = v^2$, *which is a smooth function.*

The above example illustrates that many nonsmooth optimization problems, while being nonsmooth on the entire space, can become a smooth problem once restricted to a submanifold of the space. In fact, as we have seen in Section 3.4.5, the convergence of nonsmooth optimization algorithms like BFGS depends on the local properties of the function around a small neighborhood the iterate, which we can also interpret as a approximation of a manifold.

Based on our above example and discussion, we would like to explore in the future how we can apply the manifold idea to our acceleration of proximal methods. The goal is thus two-fold: (1) To develop nonlinear acceleration methods on simple manifold optimization algorithms, and (2) to understand to what extent proximal point method and proximal bundle method can be seen as approximation of manifold optimization algorithms, at least for specific instances of optimization problems.

CHAPTER 5

BIBLIOGRAPHY

[1] Amir Beck. *First-order methods in optimization*. Society for Industrial and Applied Mathematics, 2017.

[2] Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *arXiv preprint arXiv:2101.09545*, 2021.

[3] James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[4] Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

[5] Adrian S Lewis and Stephen J Wright. A proximal method for composite minimization. *Mathematical Programming*, 158(1):501–546, 2016.

[6] Vien Mai and Mikael Johansson. Anderson acceleration of proximal gradient methods. In *International Conference on Machine Learning*, pages 6620–6629. PMLR, 2020.

[7] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[8] R Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

[9] Andrzej Ruszczynski. *Nonlinear Optimization*. Princeton university press, 2011.

[10] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. *Mathematical Programming*, 179(1):47–83, 2020.

[11] Jean-Baptiste Hiriart Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. Springer-Verlag, 1996.