# On Averaging and Extrapolation for Gradient Descent

Alan Luner[*] Benjamin Grimmer[†]

**Abstract**

This work considers the effect of averaging, and more generally extrapolation, of the iterates of gradient descent in smooth convex optimization. After running the method, rather than reporting the final iterate, one can report either a convex combination of the iterates (averaging) or a generic combination of the iterates (extrapolation). For several common stepsize sequences, including recently developed accelerated periodically long stepsize schemes, we show averaging cannot improve gradient descent's worst-case performance and is, in fact, strictly worse than simply returning the last iterate. In contrast, we prove a conceptually simple and computationally cheap extrapolation scheme strictly improves the worst-case convergence rate: when initialized at the origin, reporting $(1+1/\sqrt{16N\log(N)})x_N$ rather than $x_N$ improves the best possible worst-case performance by the same amount as conducting $O(\sqrt{N/\log(N)})$ more gradient steps. Our analysis and characterizations of the best-possible convergence guarantees are computer-aided, using performance estimation problems. Numerically, we find similar (small) benefits from such simple extrapolation for a range of gradient methods.

## 1 Introduction

There is a long history of using strategic weightings of iterates to produce better solutions in convex optimization. In many nonsmooth optimization settings, returning the final iterate is known to provide provably suboptimal worst-case guarantees [1]. A common and well-studied resolution to this problem is to output instead an average (convex combination) of the iterates [2–5]. In smooth optimization, on the other hand, averaging is not as commonplace; instead, one often simply returns the final iterate. One intuitive justification for not averaging is that most theory ensures some monotone decrease, ensuring the last iterate outperforms the rest, e.g., decreasing objective values for gradient descent. Beyond the convex combinations underlying averaging, one could consider extrapolations from the iterates, utilizing a not necessarily convex combination of them. Each step of a momentum method, like Polyak's heavy ball method, can be viewed as extrapolating from recent steps to a (hopefully) better next iterate. The use of more sophisticated extrapolation techniques, like Anderson acceleration [6, 7], have also proven useful in smooth optimization.

Basic questions remain about the importance (or lack thereof) of averaging and extrapolation in smooth optimization. Our primary focus in this paper is on smooth optimization via gradient descent. Extensions to other accelerated gradient methods will be considered afterward. For a convex function $f:\mathbb{R}^m\to\mathbb{R}$ with $L$-Lipschitz gradient, gradient descent iterates

$$x_{k+1} = x_k - \frac{h_k}{L}\nabla f(x_k)\ , \tag{1.1}$$

given initial point $x_0$ and a pre-determined stepsize sequence $h=(h_0,\dots,h_{N-1})$. Existing convergence rate theory can handle gradient descent with stepsizes $h_k$ *constant* in $(0,2)$, $h_k$ *dynamically*

[*]Johns Hopkins University, Department of Applied Mathematics and Statistics, `aluner1@jhu.edu`
[†]Johns Hopkins University, Department of Applied Mathematics and Statistics, `grimmer@jhu.edu`

*increasing* up towards two [8], and $h_k$ as a special *silver* sequence of long stepsizes, frequently greatly exceeding length two [9]. Existing worst-case convergence analysis for all of these stepsize patterns applies to the final iterate, not utilizing any averaging. This motivates our first major question:

> *Can averaging improve the worst-case performance guarantees of gradient descent (in objective gap or gradient size) for any common stepsizes choices (Constant, Dynamically Increasing, or Silver)?*

We prove it cannot. See the informal statement of this result in Theorem 1.1 and Section 3 where this claim is formalized and proven.

The fact that averaging cannot help gradient descent motivates the consideration of reporting an extrapolation. Rather than returning a convex combination of the iterates, one could return a linear combination of gradient descent iterates. Our second major question is then:

> *Can extrapolating improve the worst-case performance guarantees of gradient descent?*

We prove it can and show, in fact, a very simple, computationally cheap extrapolation approach suffices. We consider the following intuitive extrapolation strategy. After $N$ steps of gradient descent, instead of returning $x_N$, report

$$x_0 + c(x_N - x_0)$$

for some fixed extrapolation factor $c > 1$. This amounts to extrapolating from $x_N$ away from $x_0$. This is natural as one may suspect the direction of progress from the initial iterate to the terminal iterate to "roughly" point towards optimal. With $x_0$ chosen as the origin, this simplifies to reporting $cx_N$, adding effectively no additional computational or memory storage costs. We exactly characterize the resulting improvement in worst-case performance guarantee under proper selection of the extrapolation factor $c$. See the informal statement of this result in Theorem 1.2 and Section 4 where this claim is formalized and proven.

To provide such exact descriptions of worst-case algorithm performance, we turn to performance estimation. The Performance Estimation Problem (PEP), first proposed by Drori and Teboulle in [10], established a new perspective framing a method's worst-case performance as an optimization problem. Building on this initial proposal, Taylor et al. [11] showed the PEP search for a worst-case problem instance is equivalent to a Semidefinite Program (SDP). This approach to optimization problems is a doubly important tool. First, PEP serves as a black box to numerically compute worst-case convergence rates under varied algorithmic configurations. Observing these numerical results can help identify patterns and shortcomings/slackness in current performance guarantees. Second, observing the exact worst-case problem instances and corresponding dual optimality certificates produced by PEP can often translate into proving new and improved convergence guarantees. For readers unfamiliar with PEP, we refer to the library and resources of [12] and the foundational early paper [13].

Particularly relevant to our investigation, the PEP framework has seen substantial use in the design of stepsizes for gradient descent. In their paper introducing PEP, Drori and Teboulle proved a tight $O(1/N)$ convergence bound [10, Theorem 3.1] for gradient descent with constant stepsize $h \in (0, 1]$. They also conjectured a similar bound [10, Conjecture 3.1] for $h \in (1, 2)$ based on strong numerical evidence from performance estimation. More recently, by solving a related nonconvex problem via branch-and-bound, Das Gupta et al. [14] numerically identified "globally optimal" stepsize patterns for fixed small $N \leq 25$, which were often much larger than length two. Through their general branch-and-bound method, one can optimize an algorithm's hyperparameters, such as the stepsize sequence or averaging weights, for any reasonably small fixed number of iterations;

this ability to answer questions of global optimality offers a useful comparison for our later results. Motivated by these numerical results, Grimmer et al. [15, 16] used the PEP framework to analyze specific "straightforward" sequences with frequent long steps ($h_k > 2$), eventually showing a slightly accelerated convergence rate of $O\left(1/N^{1.0564}\right)$. Concurrently, Altschuler and Parrilo [9] following up on the earlier Master's thesis work [17] introduced the *silver* stepsize sequence for gradient descent, which attains a stronger rate of $O\left(1/N^{1.2716}\right)$.

## 1.1 Our Contributions

Our two primary contributions show that averaging does not improve (and, in fact, strictly worsens) the worst-case convergence guarantees for gradient descent under common stepsize sequences, while a very simple extrapolation can strictly improve gradient descent performance. Our analysis in both settings is tight, having exactly matching worst-case problem instances derived from associated PEP solutions. We measure solution quality both by function value gap and gradient norm. These two main results are informally stated below with $\Delta_N$ denoting the standard simplex in $\mathbb{R}^N$ and $\mathfrak{F}_{L,D}$ denoting the set of all considered smooth convex problem instances: namely, all pairs $(x_0, f)$ with $f$ convex and $L$-smooth and $x_0$ at most distance $D$ from a minimizer of $f$.

**Theorem 1.1** (Informal, The Optimal Averaging is to Not Average)**.** *Gradient descent has* $\sigma = (0, \ldots, 0, 1)$ *uniquely minimize both worst-case performance measures below of*

$$\min_{\sigma\in\Delta_{N+1}} \max_{(x_0,f)\in\mathfrak{F}_{L,D}} f\left(\sum_{j=0}^{N}\sigma_j x_j\right) - \inf f = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} \tag{3.6}$$

*and*

$$\min_{\sigma\in\Delta_{N+1}} \max_{(x_0,f)\in\mathfrak{F}_{L,D}} \left\|\nabla f\left(\sum_{j=0}^{N}\sigma_j x_j\right)\right\| = \frac{LD}{\sum_{i=0}^{N-1} h_i + 1}\ . \tag{3.7}$$

*when the stepsizes* $h_k$ *are chosen as any of a constant* $h \in (0, 1]$*, a constant* $h \in (1, 2)$ *(assuming* $N$ *sufficiently large and [10, Conjecture 3.1] holds), dynamically increasing as proposed by Teboulle and Vaisbourd [8], or as silver long steps as proposed by Altschuler and Parrilo [9] (assuming* $N$ *is one less than a power of two and our Conjecture 3.2 holds).*

**Theorem 1.2** (Informal, Strict Gain of $\sqrt{N/\log(N)}$ from Simple Extrapolation)**.** *For any constant stepsize* $h \in (0, 1]$ *and simple extrapolation factor* $c$ *up to size* $1 + O\left(\frac{1}{\sqrt{N\log N}}\right)$*, gradient descent has*

$$\max_{(x_0,f)\in\mathfrak{F}_{L,D}} f(x_0 + c(x_N - x_0)) - \inf f = \frac{LD^2}{4Nhc + 2}\ . \tag{1.2}$$

*For example, setting* $x_0 = 0$ *without loss of generality,*

$$\max_{(x_0,f)\in\mathfrak{F}_{L,D}} f\left(\left(1 + \frac{1}{4\sqrt{N\log(N)}}\right)x_N\right) - \inf f = \frac{LD^2}{4Nh + \sqrt{\frac{N}{\log(N)}}h + 2}\ . \tag{1.3}$$

Note that gradient descent is not optimal among all first-order methods for smooth convex optimization as accelerated methods provide faster rates of $O(1/N^2)$ [18, 19]. As a result, our theory should not be viewed as advancing the state-of-the-art in performance for this general class. Instead, gradient descent provides a structured and fundamental smooth optimization setting where we can provide exact proof that averaging offers no gains and quantify the provable gains simple extrapolation can provide. This simple extrapolation may have practical relevance in settings where memory is the dominant computational constraint (See Remark 2). It can be easily implemented at the end of any first-order method, with no a priori knowledge of the number of steps to be taken. To extend our main results beyond gradient descent, we provide numerical characterizations of the worst-case performance of several common accelerated methods with extrapolation. We find nearly universal improvements (small but nonzero) in the worst-case guarantees from simple extrapolations. Extensions to projected and proximal gradient methods are also discussed when applicable.

**Outline.** First, Section 2 introduces our notation and the performance estimation problem framework upon which our results are primarily built. Section 3 then proves our negative results regarding the use of averaging in gradient descent (for any of a range of common stepsizes). Section 4 proves our positive result on the benefits of even very simple extrapolation on worst-case performance. Finally, Section 5 presents numerical results strongly indicating the degree to which our insights from gradient descent generalize. A few algebraic simplifications are deferred to the associated `Mathematica` [20] notebook available at `github.com/alanluner/GDAvgExtrap`.

## 2 Preliminaries and Performance Estimation Problems

In this section, we introduce Performance Estimation Problems (PEP) and the specific problems relevant to our analysis. Consider a smooth convex minimization problem of the form

$$\min_{x\in\mathbb{R}^m} f(x) \tag{2.1}$$

where $f:\mathbb{R}^m\to\mathbb{R}$ is $L$-smooth (i.e. $\nabla f$ is $L$-Lipschitz) and convex. We assume a minimizer $x_\star$ of $f$ exists and suppose a point $x_0$ is known with $\|x_0-x_\star\|\leq D$ for some $D\in\mathbb{R}_+$. Throughout, we denote the Euclidean inner product by $\langle\cdot,\cdot\rangle$, and all norms are the associated two-norm. Note that a differentiable function $f$ is $L$-smooth and convex if and only if

$$f(y)\geq f(x)+\langle\nabla f(x),y-x\rangle+\frac{1}{2L}\|\nabla f(x)-\nabla f(y)\|^2 \qquad \forall x,y\in\mathbb{R}^m\ . \tag{2.2}$$

One particularly useful smooth convex function is the "Huber" function, which often occurs as a worst-case problem for smooth optimization, as highlighted in [11] and many related works [8,10,13,19]. We denote this simple one-dimensional function by

$$\phi_{L,\eta}(x):=\begin{cases}\frac{L}{2}x^2 & \text{if } |x|\leq\eta\\ L\eta|x|-\frac{L\eta^2}{2} & \text{if } |x|>\eta\ .\end{cases} \tag{2.3}$$

We primarily consider applying gradient descent by iterating as in (1.1) using a pre-determined stepsize sequence $h=(h_0,\dots,h_{N-1})$. Rather than simply reporting the terminal iterate $x_N$, we consider reporting an averaged/extrapolated point $x_\sigma$, defined by

$$x_\sigma=\sum_{j=0}^{N}\sigma_j x_j \tag{2.4}$$

given weights $\sigma=(\sigma_0,\dots,\sigma_N)$. Note no modification is made to the iterates of gradient descent; the only change is to which point is ultimately reported and, hence, where performance (in the objective gap or gradient norm) is measured.

We broadly refer to this scheme as *general extrapolation.* As a specific case, we define *iterate averaging* as any choice of $\sigma$ such that $\sigma \geq 0$ and $\sum_{j=0}^{N} \sigma_j = 1$. We will say that an averaging scheme $\sigma$ is *nondegenerate* provided that $\sum_{j=0}^{N} \sigma_j x_j \neq x_N$ (i.e. $\sigma \neq (0, \ldots, 0, 1)$). Observe that our framing of $x_\sigma$ is equivalent to

$$\begin{aligned} x_\sigma = \sum_{j=0}^{N} \sigma_j x_j &= \sum_{j=0}^{N} \sigma_j \left( x_0 - \frac{1}{L} \sum_{k=0}^{j-1} h_k \nabla f(x_k) \right) \qquad (2.5) \\ &= x_0 - \frac{1}{L} \sum_{k=0}^{N-1} \left( h_k \sum_{j=k+1}^{N} \sigma_j \right) \nabla f(x_k) \ . \end{aligned}$$

So a general extrapolation can be denoted by $x_0 + \operatorname{span}\{\nabla f(x_0), \ldots, \nabla f(x_{N-1})\}$.

## 2.1 Performance Estimation Problems with Averaging/Extrapolation

When evaluating the convergence of an optimization method, there are several different performance measures commonly considered. We will introduce the PEP framework using the objective gap $f(x) - f(x_\star)$ as the measure of algorithm performance. Although this is done to ease our initial development, we will also present results on guaranteeing a small gradient norm, $\|\nabla f(x)\|$. In particular, our primary focus is to compare the worst-case performance of an averaged or extrapolated point $x_\sigma$ with that of the last iterate $x_N$.

One can frame deriving a convergence guarantee for an algorithm in terms of understanding its performance on a worst-case problem instance. From this perspective, worst-case analysis is an optimization problem: find the problem instance with the maximum final/reported objective gap, gradient norm, etc. We define our worst-case performance $p_{N,L,D}(\sigma, h)$ by

$$p_{N,L,D}(\sigma, h) := \begin{cases} \max_{x_0, x_\star, f} & f(x_\sigma) - f(x_\star) \\ \text{s.t.} & f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2 \quad \forall x, y \\ & \|x_0 - x_\star\| \leq D \\ & \nabla f(x_\star) = 0 \\ & x_{k+1} = x_k - \frac{h_k}{L} \nabla f(x_k) \qquad k = 0, \ldots, N-2 \\ & x_\sigma = x_0 - \frac{1}{L} \sum_{k=0}^{N-1} \left( h_k \sum_{j=k+1}^{N} \sigma_j \right) \nabla f(x_k) \end{cases} \qquad (2.6)$$

where our first constraint comes from the standard identity for $L$-smooth, convex functions (2.2). We use the alternate form (2.5) of $x_\sigma$ to remove its dependence on $x_N$. Consequently, we can exclude $x_N$ from our problem formulation. Equivalently, one could view $x_\sigma$ as a modified replacement of $x_N$.

Finite dimensional relaxations of this formulation were first considered by Drori and Teboulle [10]. Subsequently and quite surprisingly, the Interpolation Theorem of Taylor et al. [11] established an equivalent finite-dimensional problem one could consider. Rather than enforce our constraints for all points in our domain, it is only necessary to enforce them along the points of interest in our algorithm. Denoting $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, and $I_N^\star = \{\star, 0, 1, \ldots, N-1, \sigma\}$, our problem becomes

$$
p_{N,L,D}(\sigma,h) = \begin{cases} \max_{x_0,x_\star,f} & f(x_\sigma) - f(x_\star) \\ \text{s.t.} & f_i \geq f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \quad \forall i \neq j \in I_N^\star \\ & \|x_0 - x_\star\| \leq D \\ & g_\star = 0 \\ & x_{k+1} = x_k - \frac{h_k}{L} g_k \qquad k = 0,\dots,N-2 \\ & x_\sigma = x_0 - \frac{1}{L}\sum_{k=0}^{N-1}\left(h_k \sum_{j=k+1}^{N}\sigma_j\right) g_k \ . \end{cases} \tag{2.7}
$$

By translation, we can also fix $x_\star = 0$ and $f_\star = 0$ without loss of generality.

Finally, following the methods in many previous works [11,13,21], we slightly relax our discrete problem (2.7) to form a solvable SDP. We adopt the notation used in [14] and introduce it here. We define

$$
\begin{aligned}
F &= [f_0|f_1|\dots|f_{N-1}|f_\sigma] \in \mathbb{R}^{1\times(N+1)} \\
H &= [x_0|g_0|g_1|\dots|g_{N-1}|g_\sigma] \in \mathbb{R}^{d\times(N+2)} \\
G &= H^T H \in \mathbb{S}_+^{N+2}
\end{aligned}
$$

where $f_\sigma = f(x_\sigma)$ and $g_\sigma = \nabla f(x_\sigma)$. We define special vectors for selecting particular elements of our matrices using standard basis vectors $e_i$ (the corresponding space for $e_i$ should be clear from context if not specified):

$$
\begin{aligned}
&\mathbf{g}_\star = 0 \in \mathbb{R}^{N+2} \\
&\mathbf{g_i} = e_{i+2} \in \mathbb{R}^{N+2} \qquad i = 0,\dots,N-1 \\
&\mathbf{x_0} = e_1 \in \mathbb{R}^{N+2} \\
&\mathbf{x}_\star = 0 \in \mathbb{R}^{N+2} \\
&\mathbf{x_{i+1}} = \mathbf{x_i} - \frac{h_i}{L}\mathbf{g_i} \qquad i = 0,\dots,N-2 \\
&\mathbf{f}_\star = 0 \in \mathbb{R}^{N+1} \\
&\mathbf{f_i} = e_{i+1} \in \mathbb{R}^{N+1} \qquad i = 0,\dots,N-1
\end{aligned}
$$

and to account for $x_\sigma$, we define

$$
\begin{aligned}
&\mathbf{g}_\sigma = e_{N+2} \in \mathbb{R}^{N+2} \\
&\mathbf{x}_\sigma = \mathbf{x}_0 - \frac{1}{L}\sum_{k=0}^{N-1}\left(h_k \sum_{j=k+1}^{N}\sigma_j\right)\mathbf{g_k} \\
&\mathbf{f}_\sigma = e_{N+1} \in \mathbb{R}^{N+1} \ .
\end{aligned}
$$

Through this construction, we have encoded the gradient steps of the algorithm into our matrices $F$, $G$, and $H$; we have $x_i = H\mathbf{x_i}$, $g_i = H\mathbf{g_i}$, and $f_i = F\mathbf{f_i}$ (and similar for $x_\sigma$ and $x_\star$). Next, using the symmetric outer product $x \odot y = \frac{1}{2}(xy^T + yx^T)$, define

$$
\begin{aligned}
A_{i,j}(h) &= \mathbf{g_j} \odot (\mathbf{x_i} - \mathbf{x_j}) \in \mathbb{S}^{N+2} \\
B_{i,j}(h) &= (\mathbf{x_i} - \mathbf{x_j}) \odot (\mathbf{x_i} - \mathbf{x_j}) \in \mathbb{S}^{N+2} \\
C_{i,j} &= (\mathbf{g_i} - \mathbf{g_j}) \odot (\mathbf{g_i} - \mathbf{g_j}) \in \mathbb{S}^{N+2} \\
a_{i,j} &= \mathbf{f_j} - \mathbf{f_i} \in \mathbb{R}^{N+1}
\end{aligned}
$$

for all $i, j \in I_N^\star$, with $i \neq j$. These matrices satisfy the useful identities

$$\begin{aligned}
\langle g_j, x_i - x_j \rangle &= \mathrm{Tr} G A_{i,j}(h) \\
\|x_i - x_j\|^2 &= \mathrm{Tr} G B_{i,j}(h) \\
\|g_i - g_j\|^2 &= \mathrm{Tr} G C_{i,j} \ .
\end{aligned}$$

These definitions enable a succinct SDP relaxation of the performance estimation problem:

$$p_{N,L,D}(\sigma, h) \leq \begin{cases} \max_{F,G} & F a_{\star,\sigma} \\ \text{s.t.} & F a_{i,j} + \mathrm{Tr} G A_{i,j}(h) + \frac{1}{2L} \mathrm{Tr} G C_{i,j} \leq 0 \quad \forall i \neq j \in I_N^\star \\ & G \succeq 0 \\ & \mathrm{Tr} G B_{0,\star} \leq D^2 \ . \end{cases} \tag{2.8}$$

Since this SDP is a relaxation of our original problem, its solution is an upper bound for $p_{N,L,D}(\sigma, h)$. The SDP can be made equivalent by applying an additional assumption that the problem dimension $m$ is at least $N + 2$ [11]. However, the inequality above will be sufficient for our analysis, so for now, we omit this additional rank assumption. This completes our derivation of the PEP SDP for the averaged/extrapolated performance measure $f(x_\sigma) - f(x_\star)$. Other common performance measures would follow a very similar derivation. Going forward, we simplify our notation to $p(\sigma) := p_{N,L,D}(\sigma, h)$, but note that our solution is a function of each of those now hidden parameters. We especially emphasize the role of $N$ as a known and fixed parameter in the optimization problem.

Finally, we define the dual SDP in preparation for our use of the dual certificate in our later proofs. Introducing dual variables $\lambda_{i,j} \in \mathbb{R}$, $v \in \mathbb{R}$, and $Z \in \mathbb{S}^{(N+2)\times(N+2)}$, we let

$$d(\sigma) := \begin{cases} \min_{v,\lambda,Z} & v D^2 \\ \text{s.t.} & \sum_{i \neq j} \lambda_{i,j} a_{i,j} - a_{\star,\sigma} = 0 \quad \forall i \neq j \in I_N^\star \\ & v B_{0,\star} + \sum_{i \neq j} \lambda_{i,j} (A_{i,j}(h) + \frac{1}{2L} C_{i,j}) = Z \quad \forall i \neq j \in I_N^\star \\ & Z \succeq 0 \\ & v \geq 0, \lambda_{i,j} \geq 0 \quad \forall i \neq j \in I_N^\star \end{cases} \tag{2.9}$$

with the role of $\sigma$ implicit in the various vector and matrix definitions. This SDP (2.8) and its dual (2.9) provide an incredibly useful framework for analyzing our worst-case problem instances [10, 11, 14–17]. This dual problem will play a central role in proving our new convergence guarantees.

## 3 Averaging is Strictly Worse than the Last Iterate

In this section, we address our first major question regarding the effect of iterate averaging on gradient descent's convergence guarantees. Our approach to establishing that averaging cannot benefit (and in fact harms) worst-case convergence guarantees works by first observing a common structure in the tight convergence bounds for many common stepsize selections (Section 3.1) and second, showing any stepsize with this worst-case structure cannot benefit from averaging (Section 3.2).

### 3.1 Common Structure Among Tight Last Iterate Convergence Guarantees

The performance estimation framework has enabled the development of exactly tight convergence rates for gradient descent. That is, convergence guarantees that are attained with equality by some problem instance, establishing that no better guarantee is possible. Below, we summarize the known (or conjectured) tight convergence rates for four different families of stepsizes $h_k$. For each considered stepsize policy on any $L$-smooth convex $f$ with $\|x_0 - x_\star\| \leq D$, these tight rates for gradient descent take the form

$$f(x_N) - f(x_\star) \leq \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} \tag{3.1}$$

and

$$\|\nabla f(x_N)\| \leq \frac{LD}{\sum_{i=0}^{N-1} h_i + 1}\ . \tag{3.2}$$

Before discussing particular stepsize selections, we first note that no stepsize sequence $h_k$ can produce a rate faster than those above. This follows from considering simple Huber functions as prior works [8, 10, 11, 13, 19] have identified as common worst-case problem instances. While we do not focus on momentum methods in this section, we note that the prevalence of the Huber function appears to be shared for momentum methods such as Nesterov acceleration and OGM, as conjectured in [11].

**Lemma 3.1.** *For any $N, L, D > 0$ and $h_k > 0$, there exists a problem instance with $L$-smooth convex $f$ and $\|x_0 - x_\star\| \leq D$ such that*

$$f(x_N) - f(x_\star) = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2}\ .$$

*There also exists an $L$-smooth convex $f$ with*

$$\|\nabla f(x_N)\| = \frac{LD}{\sum_{i=0}^{N-1} h_i + 1}\ .$$

*Proof.* In both cases, this problem instance takes the form of $x_0 = D$ and $f = \phi_{L,\eta}$ as a Huber function. Note that provided $\eta \leq D/(\sum_{i=0}^{N-1} h_i + 1)$, the first $N$ iterates of gradient descent all remain larger than $\eta$, being given by $x_k = D - \eta\sum_{i=0}^{k-1} h_i$ and having constant gradient $\nabla f(x_k) = \eta L$. To bound objective gap convergence, selecting $\eta = D/(2\sum_{i=0}^{N-1} h_i + 1)$ has $f(x_N) - f(x_\star) = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2}$. To bound gradient norm convergence, selecting $\eta = D/(\sum_{i=0}^{N-1} h_i + 1)$ has $\|\nabla f(x_N)\| = \frac{LD}{\sum_{i=0}^{N-1} h_i + 1}$. □

As a result, guarantees of the form (3.1) and (3.2) holding imply more strongly that

$$\max_{(x_0,f)\in\mathfrak{F}_{L,D}} f(x_N) - \inf f = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} \tag{C1}$$

and

$$\max_{(x_0,f)\in\mathfrak{F}_{L,D}} \|\nabla f(x_N)\| = \frac{LD}{\sum_{i=0}^{N-1} h_i + 1} \tag{C2}$$

hold respectively. We show below in Theorems 3.1 and 3.2 that these two conditions imply averaging is strictly worse than returning the last iterate in terms of worst-case guarantees.

**Constant Stepsizes** $h_k = h \in (0, 1]$ Perhaps the simplest setting of gradient descent is the use of a constant stepsize $h_k = h \in (0, 1]$. The seminal work [10] and subsequently [8] showed such stepsizes have guarantees of the form (3.1) and (3.2), giving bounds of $LD^2/(4Nh + 2)$ and $LD/(Nh + 1)$ on objective gap and gradient norm convergence, respectively. Note tightness of these bounds is easily verified by the examples in Lemma 3.1, and so the conditions (C1) and (C2) hold. For extensions of these ideas to monotone operators, see [22, Remark 4.10].

**Constant Stepsizes $h_k = h \in (1,2)$** When using constant stepsizes $h_k = h \in (1,2)$, the conjectured tight convergence rates from [10, Conjecture 3.1] and [11, Conjecture 3], respectively, are that

$$f(x_N) - f(x_\star) \leq \frac{LD^2}{2} \max\left\{\frac{1}{2Nh+1}, |1-h|^{2N}\right\} \tag{3.3}$$

and

$$\|f(x_N)\| \leq LD \max\left\{\frac{1}{Nh+1}, |1-h|^{N}\right\} \ . \tag{3.4}$$

Provided $N$ is large enough relative to the fixed value of $h$, each first case above dominates, and the (conjectured) convergence guarantees take the form (3.1) and (3.2). Again, tightness and the conditions (C1) and (C2) follow from Lemma 3.1.

**Dynamic Stepsizes $h_k \to 2$** Note convergence cannot be guaranteed for gradient descent with $h_k$ constant and greater than or equal to two. A sequence of stepsizes $h_k$ approaching this boundary was proposed and analyzed by [8]. They considered

$$h_k = \frac{-\sum_{i=0}^{k-1} h_i + \sqrt{\left(\sum_{i=0}^{k-1} h_i\right)^2 + 8\left(\sum_{i=0}^{k-1} h_i + 1\right)}}{2} \tag{3.5}$$

with $h_0 = \sqrt{2}$. Theorem 4 of [8] established tight convergence guarantees of the common form (3.1) and (3.2), establishing conditions (C1) and (C2) hold.

**Silver Stepsizes $h_k$ (often much greater than two)** Going beyond the limitation of stepsizes being at most length two, Altschuler and Parrilo [9] considered the "silver" stepsize pattern: Using the silver ratio, $\rho = 1 + \sqrt{2}$, the sequence $h^{(N)}$ is defined recursively for any $N = 2^m - 1$ by concatenation $h^{(2N+1)} = (h^{(N)}, 1 + \rho^{m-1}, h^{(N)})$ and with $h^{(1)} = \sqrt{2}$. Note this includes arbitrarily large stepsizes, having $h_{2k} \approx k^{1.2716}$ whenever $k$ is a power of two. Theorem 1.1 of [9] showed silver stepsizes have an accelerated convergence rate of $f(x_N) - f(x_\star) = O(LD^2/N^{1.2716})$. Numerically computing the worst-case performance of silver stepsizes via PEP, we find for every $N$ one less than a power of two, the convergence matches (3.1) and (3.2), see Table 1. This leads us to make the following conjecture.

**Conjecture 3.2.** *Consider a gradient descent algorithm of fixed length $N = 2^k - 1$ and the corresponding Silver step sequence $h^{(N)}$. Then any $L$-smooth convex $f$ has*

$$f(x_N) - f(x_\star) \leq \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} = \frac{LD^2}{4\rho^{1+\log_2(N+1)} - 2} \ ,$$
$$\|f(x_N)\| \leq \frac{LD}{\sum_{i=0}^{N-1} h_i + 1} = \frac{LD}{\rho^{1+\log_2(N+1)}} \ .$$

If true, tightness of these bounds is immediate from Lemma 3.1, giving conditions (C1) and (C2).

## 3.2 Suboptimal Convergence from Averaging

The above tight characterizations (down to the constants) of the worst-case performance of gradient descent's final iterate provide a direct quantity to compare the performance of a proposed averaging scheme against. We find that the conditions (C1) and (C2) imply every nondegenerate iterate averaging $\sigma$ provides a strictly worse final objective value. This is formalized for objective gap

| | $N=1$ | $N=3$ | $N=7$ | $N=15$ | $N=31$ | $N=63$ | $N=127$ |
|---|---|---|---|---|---|---|---|
| Silver Obj. Gap PEP | 0.13060 | 0.04692 | 0.01842 | 0.00747 | 0.00307 | 0.00127 | 0.00058 |
| Difference from (C1) | -8.567e-10 | 1.581e-8 | 5.148e-11 | -2.942e-9 | -5.057e-12 | -3.232e-8 | -2.422e-8 |
| Silver Grad. Norm PEP | 0.41421 | 0.17157 | 0.07107 | 0.02944 | 0.01219 | 0.00505 | 0.00233 |
| Difference from (C2) | -3.669e-14 | -1.976e-13 | -6.178e-12 | -5.630e-9 | -1.204e-8 | -3.685e-10 | 7.602e-13 |

Table 1: Numerical results from `Mosek` with feasibility tolerances set as $10^{-12}$ solving the PEP SDP for the worst-case objective gap and gradient norm of $x_N$ after $N$ steps of the silver stepsize sequence. Differences from the value in Conjecture 3.2 are presented, which remain relatively small.

convergence in Theorem 3.1 and gradient norm convergence in Theorem 3.2. Similar to Lemma 3.1, the proof of these results is based on considering related Huber functions.

As an immediate consequence of these theorems, averaging cannot benefit (and strictly worsens) the convergence guarantees for gradient descent with $h_k$ constant less than one or as the dynamic sequence in (3.5). Further, if $N$ is sufficiently large and the numerically supported conjecture of [10, Conjecture 3.1] holds, constant stepsizes between one and two cannot benefit from averaging. Similarly, if $N$ is one less than a power of two and the numerically supported Conjecture 3.2 holds, the accelerated rate following from silver stepsizes cannot benefit from averaging. As an aside, numerical PEP evaluations indicate the conditions (C1) and (C2) do not appear to hold for the straightforward sequences studied by [15, 16], indicating they may benefit from averaging.

**Theorem 3.1.** *Consider gradient descent with stepsizes $h = (h_0, \dots, h_{N-1}) > 0$. For any $L, D > 0$, if the worst final iterate objective gap is characterized by* (C1), *then*

$$\min_{\sigma\in\Delta_{N+1}} \max_{(x_0,f)\in\mathfrak{F}_{L,D}} f\left(\sum_{j=0}^{N}\sigma_j x_j\right) - \inf f = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} \tag{3.6}$$

*with $\sigma = (0, \dots, 0, 1)$ as the unique minimizer.*

*Proof.* To prove this result, it suffices to show that for any nondegenerate $\sigma$, there exists an $L$-smooth convex function $f$ and initialization $\|x_0 - x_\star\| \le D$ such that

$$f(x_\sigma) - f(x_\star) > \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2} \ .$$

We consider the same Huber function establishing tightness of the last iterate convergence in Lemma 3.1. Let $x_0 = D$ and $f = \phi_{L,\eta}$ with $\eta = D/(2\sum_{i=0}^{N-1} h_i + 1)$. Noting $\eta \le D/(\sum_{i=0}^{N-1} h_i + 1)$, each $k \le N$ has $x_k = D - \eta\sum_{i=0}^{k-1} h_i \ge \eta$ and $\nabla f(x_k) = \eta L$. Observe that $f$ is linear on the convex hull of the iterates (since it is linear on $x \ge \eta$) with

$$f(x_0) = \frac{LD^2}{2\sum_{i=0}^{N-1} h_i + 1} - \frac{LD^2}{2(2\sum_{i=0}^{N-1} h_i + 1)^2} \ ,$$
$$f(x_N) = \frac{LD^2}{2(2\sum_{i=0}^{N-1} h_i + 1)} < f(x_0) \ .$$

Hence $f(x_N) < f(x_k)$ for all $k < N$. Then for any nondegenerate $\sigma$, this linearity ensures

$$f(x_\sigma) - f(x_\star) = f(x_\sigma) = \sum_{j=0}^{N}\sigma_j f(x_j) > f(x_N) = \frac{LD^2}{4\sum_{i=0}^{N-1} h_i + 2}$$

Therefore, the degenerate averaging $\sigma = (0, \dots, 0, 1)$ minimizes $\max_{(x_0,f)} f(x_\sigma) - \inf f$ uniquely. $\square$

**Theorem 3.2.** *Consider gradient descent with stepsizes $h = (h_0, \dots, h_{N-1}) > 0$. For any $L, D > 0$, if the worst final iterate gradient norm is characterized by* (C2)*, then*

$$\min_{\sigma \in \Delta_{N+1}} \max_{(x_0,f)\in\mathcal{F}_{L,D}} \left\| \nabla f\left(\sum_{j=0}^{N} \sigma_j x_j\right)\right\| = \frac{LD}{\sum_{i=0}^{N-1} h_i + 1} \tag{3.7}$$

*with $\sigma = (0, \dots, 0, 1)$ as the unique minimizer.*

*Proof.* To prove this, we require a perturbation of the tight example for the last iterate gradient norm convergence used Lemma 3.1. Consider $x_0 = D$ and the Huber function $f = \phi_{L,\eta}$ with

$$\eta = \frac{D}{\sum_{i=0}^{N-1} h_i + 1} + \delta$$

for some small $\delta > 0$. Note any selection of $\delta < \frac{Dh_{N-1}}{(\sum_{i=0}^{N-1} h_i+1)(\sum_{i=0}^{N-2} h_i+1)}$ has $D - \eta\sum_{i=0}^{N-2} h_i > \eta$. As a result, the first $N$ iterates of gradient descent are still given by $x_k = D - \eta\sum_{i=0}^{k-1} h_i$. Thus, any averaging $\sigma$ produces an output point

$$x_\sigma = \sum_{j=0}^{N} \sigma_j x_j = \sum_{j=0}^{N} \sigma_j (D - \eta \sum_{i=0}^{j-1} h_i) = D - \eta \sum_{j=0}^{N} \sigma_j \sum_{i=0}^{j-1} h_i \ .$$

Additionally, suppose that $\delta$ is chosen small enough that

$$\left(\sum_{j=0}^{N} \sigma_j \sum_{i=0}^{j-1} h_i\right)\delta < \frac{D}{\sum_{i=0}^{N-1} h_i + 1}\left(\sum_{i=0}^{N-1} h_i - \sum_{j=0}^{N} \sigma_j \sum_{i=0}^{j-1} h_i\right) \ .$$

Note that for any nondegenerate averaging, $\sum_{j=0}^{N} \sigma_j \sum_{i=0}^{j-1} h_i < \sum_{i=0}^{N-1} h_i$, so the right-hand side above is strictly positive. Plugging this into our formula for $x_\sigma$ with $\eta = \frac{D}{\sum_{i=0}^{N-1} h_i+1} + \delta$ yields

$$x_\sigma = D - \left(\frac{D}{\sum_{i=0}^{N-1} h_i + 1} + \delta\right)\sum_{j=0}^{N} \sigma_j \sum_{i=0}^{j-1} h_i > \frac{D}{\sum_{i=0}^{N-1} h_i + 1} \ .$$

By construction, the Huber objective function at $x_\sigma > \eta$ then has $\nabla f(x_\sigma) > \frac{D}{\sum_{i=0}^{N-1} h_i+1}$. Any choice of $\delta > 0$ less than our two needed upper bounds suffices. Since these bounds are positive for any nondegenerate iterate averaging, we conclude the degenerate averaging $\sigma = (0, \dots, 0, 1)$ uniquely minimizes the worst-case gradient norm $\max_{(x_0,f)} \|\nabla f(x_\sigma)\|$. □

**Remark 1** (Extension to Projected/Proximal Gradient Descent)**.** Reasoning virtually identical to the above can be applied to the proximal gradient method. Consider minimizing an additive composite function $\psi = f + r$ where $f, r$ are both convex. Denote the proximal mapping by

$$\mathrm{prox}_{tr}(x) := \mathrm{argmin}\left\{ r(u) + \frac{1}{2t}\|u - x\|^2 : u \in \mathbb{R}^m \right\} \ .$$

and then iterate according to $x_{k+1} = \mathrm{prox}_{h_k r}(x_k - \frac{h_k}{L}\nabla f(x_k))$. We note that projected gradient descent is a special case where we take $r$ as the indicator function of a closed, convex set. In [8, Theorem 7], Teboulle and Vaisbourd proved for any $L$-smooth $f$, the proximal gradient method with constant stepsize $h \in (0,1]$ satisfies

$$\psi(x_N) - \psi(x_\star) \le \frac{LD^2}{4Nh}$$

and

$$\|\psi_{h/L}(x_{N-1})\| \le \frac{LD}{Nh}$$

where

$$\psi_{h/L}(x_{N-1}) = \Big((x_{N-1} - \frac{h}{L}\nabla f(x_{N-1})) - (x^+_{N-1} - \frac{h}{L}\nabla f(x^+_{N-1}))\Big)\, L/h$$

and $x^+ = \mathrm{prox}_{(h/L)r}\left(x - \frac{h}{L}\nabla f(x)\right)$. Note that $\psi_{h/L}(x_{N-1}) \in \partial\psi(x_N)$, so these results are clear analogues of those in Theorems 3.1 and 3.2. The bounds above are shown to be tight by the simple example of $f(x) = \eta x$ and $r$ as the indicator function over $\mathbb{R}_+$, choosing $\eta = \frac{D}{2Nh}$ and $\eta = \frac{D}{Nh}$, respectively. The earliest occurrences of these tight lower bound examples to our knowledge are [23, Appendix 2.8] and [11, Section 4]. Consequently, applying our above argument, averaging cannot improve (and strictly worsens) convergence guarantees for proximal or projected gradient descent with constant stepsize $h \in (0,1]$. As a note, although PEP is not used for proximal settings here, such natural extensions exist [13] and have seen much use [12, 24–26].

## 4 Extrapolations are Strictly Better than the Last Iterate

Given averaging gradient descent's iterates cannot improve its convergence bounds, we now remove the restriction that $\sigma$ provides a convex combination, allowing it to be freely chosen from $\mathbb{R}^{N+1}$. As discussed in Section 2, this allows $x_\sigma$ to be anything in $x_0 + \mathrm{span}\{\nabla f(x_0), \dots, \nabla f(x_{N-1})\}$.

An initial problem to investigate in this larger parameter space is to determine the optimal choice of $\sigma$ for a fixed number of iterations $N$ and fixed step sequence $h$. Given $N, L, D > 0$ and stepsizes $h$, this amounts to solving the following nonconvex minimization problem

$$\min_{\sigma \in \mathbb{R}^{N+1}} p(\sigma)\ .$$

Note that expanding the definition of $p(\sigma)$ gives a nonconvex minimax problem. Das Gupta et al. [14] introduced a spatial branch-and-bound approach tailored to numerically globally solving such problems. We apply their method to the new setting of optimizing extrapolation weights $\sigma$. Fixing $N = 10$ and constant stepsize $h = 1$, Figure 1 shows the optimal $\sigma$ for minimizing the objective gap $f(x_\sigma) - f(x_\star)$ and for minimizing the gradient norm $\|\nabla f(x_\sigma)\|$. We observe that the components of $\sigma$ are dominated by the weights of the last two components. It is worth noting, however, that approximating these final two weights and setting the remaining weights to zero significantly worsens the performance. Thus, the subtle structure (highlighted in Figure 1) for components $k < N-1$ appears to be integral to the optimal averaging's success. The first four columns of Table 2 show these best possible improvements to the worst-case performance as $N$ varies.

While the optimal $\sigma$ offers a noticeable improvement to the previous convergence bound, we did not identify a clear pattern in the optimal values. Even given a construction for the optimal extrapolation for each $N$, applying it would require prior knowledge of $N$ when beginning the method to construct $x_\sigma$ incrementally as a second vector in memory while running gradient descent. To avoid these shortcomings, we restrict $\sigma$ to a particularly convenient structure, namely

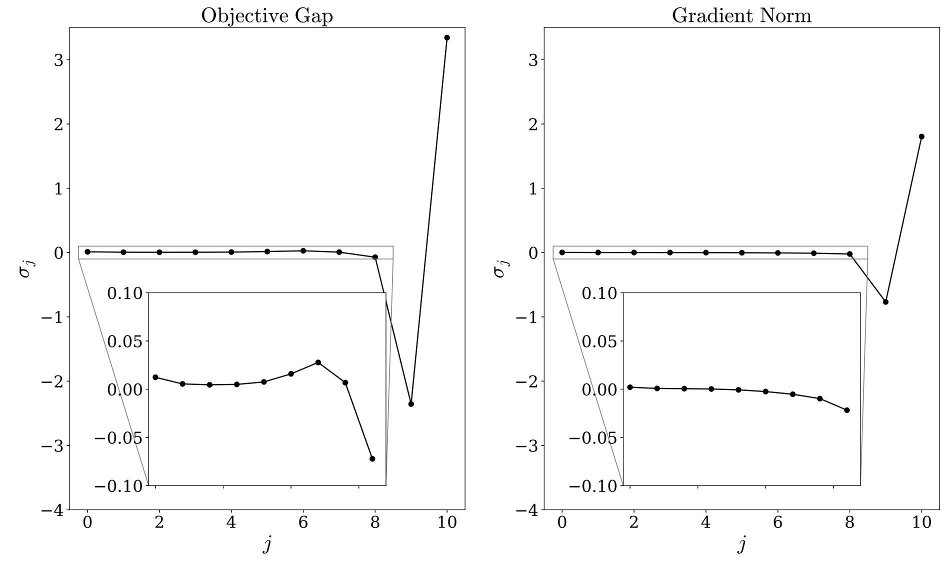

Figure 1: Optimal extrapolation choice of $\sigma$ for $N = 10$ under different performance measures.

| | Last Iterate/Averaging | | Optimal Extrapolation | | Optimal Simple Extrapolation | |
|---|---|---|---|---|---|---|
| $N$ | Obj Gap | Grad Norm | Obj Gap | Grad Norm | Obj Gap | Grad Norm |
| 5 | 0.0455 | 0.1667 | 0.0365 | 0.1483 | 0.0390 | 0.1546 |
| 10 | 0.0238 | 0.0909 | 0.0200 | 0.0844 | 0.0213 | 0.0871 |
| 15 | 0.0161 | 0.0625 | 0.0139 | 0.0592 | 0.0147 | 0.0607 |
| 20 | 0.0122 | 0.0476 | 0.0107 | 0.0456 | 0.0113 | 0.0466 |
| 25 | 0.0098 | 0.0385 | 0.0087 | 0.0371 | 0.0091 | 0.0378 |

Table 2: Numerical comparison between gradient descent's worst case objective gap and gradient norm at its last iterate $x_N$, at the best possible extrapolated point $\sum \sigma_j x_j$, and at the best possible simple extrapolation point $x_0 + c(x_N - x_0)$, fixing $L = D = h = 1$.

$\sigma = (-(c-1), 0, \dots, 0, c)$ for any extrapolation factor $c \geq 1$. This corresponds to

$$x_\sigma = x_0 + c(x_N - x_0) = x_0 - c \sum_{k=0}^{N-1} \frac{h_k}{L} g_k \ . \tag{4.1}$$

Hence, one can interpret $x_\sigma$ as extrapolating along the vector from $x_0$ to $x_N$. We refer to this particular method of extrapolation as *simple extrapolation*. Note when $c = 1$, this reverts back to simply returning gradient descent's last iterate $x_\sigma = x_N$.

This scheme's simplicity offers a few advantages. To perform this extrapolation, one needs only the initial point $x_0$ and terminal point $x_N$; no auxiliary vectors or enlarged memory storage are required. Then, whenever the gradient descent iteration is stopped, this extrapolation can be computed without requiring a prior knowledge of $N$. The constant $c$ can be selected on-the-fly once the method stops. See Corollary 4.2 for several convenient, provably good formulas one could use, setting $c = 1 + \Theta(\sqrt{1/N \log(N)})$.

In the remainder of this section, we focus on gradient descent with constant stepsize $h \in (0, 1]$ and demonstrate a provable benefit to simple extrapolation. As a result, we show that including this simple rescaling at the final step provides a provably good improvement "for free". In Table 2, we include the optimal performance of simple extrapolation, though this will be discussed in more detail in Section 5.1.

**Remark 2** (Computational Considerations)**.** In large-scale optimization settings where memory is a *severely* limiting factor, the simplicity of this extrapolation becomes relevant. While accelerated momentum methods achieve faster convergence, they require the simultaneous storage of two vectors rather than just the current iterate $x_k$. For such problem instances where memory usage is at capacity from storing a single vector $x_k$, doubling the memory storage is not an option, and hence momentum methods cannot be used. When $x_0$ is a structured point such as the origin, simple extrapolations can improve convergence while incurring no added memory costs.

The increased memory costs of momentum methods are also avoided by the recently developed gradient descent accelerations from longer stepsizes in the sequence, e.g., the Silver pattern's $O(1/N^{1.2716})$ rate [9]. Such methods may be state-of-the-art in certain severely memory-constrained settings. In Section 5, we show numerically that simple extrapolation also (slightly) improves the worst-case performance using silver stepsizes, offering a computational benefit at nearly no cost.

## 4.1 Improved Convergence Guarantees from Simple Extrapolations

We are now prepared to present our main theorem quantifying the improvement in worst-case performance derived from simple extrapolations. To first discuss the consequences of our theory, we defer the proof of this result to Section 4.2. Our analytic results below focus on the objective gap as the performance measure. In Section 5.1, we return to numerically consider the minimization of the reported gradient norm, establishing similar improvements. However, our Conjecture 5.1 suggests that simple extrapolation is less effective at improving the final gradient norm size, hence our focus on the objective gap.

**Theorem 4.1.** *For any convex $L$-smooth function $f$ with minimizer $x_\star$, consider gradient descent with constant stepsizes $h_k = h \in (0, 1]$. For any $N > 0$, define the function*

$$\psi_N(c) := 1 - \sum_{i=0}^{N-1} \frac{(c-1)(2Nc - 2N + 1)(2Nc + 1)}{2Nc + 4Nci - 2i^2 + 1} > 0 \tag{4.2}$$

| $N$ | $c_{\text{crit}}$ | $c_u$ | $c_\ell$ | $1+1/(4\sqrt{N}\log N)$ |
|---|---|---|---|---|
| 1 | 1.5 | 1.672029 | 1.359869 | - |
| 10 | 1.121974 | 1.158494 | 1.104917 | 1.052099 |
| 100 | 1.034804 | 1.041404 | 1.028570 | 1.011650 |
| $10^4$ | 1.002878 | 1.003171 | 1.002235 | 1.000824 |
| $10^6$ | 1.000246 | 1.000263 | 1.000186 | 1.000067 |
| $10^8$ | 1.000022 | 1.000023 | 1.000016 | 1.000006 |

Table 3: Approximate values of $c_{\text{crit}}$ and the upper and lower bounds from Proposition 4.1.

*and let* $c_{\text{crit}}$ *be the largest root of* $\psi_N(c)$. *For any* $c \in [1, c_{\text{crit}}]$, $x_\sigma = x_0 + c(x_N - x_0)$ *satisfies*

$$f(x_\sigma) - f(x_\star) \leq \frac{LD^2}{4Nhc+2} \tag{4.3}$$

*where* $D = \|x_0 - x_\star\|$. *Moreover, for any* $L, D, N > 0$, *there exists a function* $f$ *where equality holds.*

In terms of PEP introduced in Section 2, this theorem provides an exact description of the worst-case performance of small extrapolations: the extrapolation $\sigma = (-(c-1), 0, \ldots, 0, c)$ has

$$p(\sigma) = \frac{LD^2}{4Nhc+2} \qquad \forall c \in [1, c_{\text{crit}}] \ .$$

Setting $c$ as large as possible (within the theorem's bounds) provides the strongest performance guarantee. To quantify the level of resulting benefit, we next bound how large the root $c_{\text{crit}}$ is for a given value of $N$.

The definition of $c_{\text{crit}}$ as the largest root of $\psi_N(c)$ unfortunately yields no simple closed-form expression. Considering the case of $N = 2$, our function is $\psi_2(c) = \frac{-64c^3+112c^2-36c-1}{12c-1}$, so $c_{\text{crit}}$ is the largest root of $-64c^3 + 112c^2 - 36c - 1 = 0$, approximately 1.312285. In general, given any $N$, $c_{\text{crit}}$ will be the largest root of a degree $N+1$ polynomial. In Table 3, we include numerical approximations of $c_{\text{crit}}$ for various values of $N$. The following proposition shows $c_{\text{crit}}$ shrinks at rate $1 + \Theta\left(\frac{1}{\sqrt{N}\log N}\right)$.

**Proposition 4.1.** *For N>1, the largest root* $c_{\text{crit}}$ *of* $\psi_N(c)$ *is bounded below by*

$$c_{\text{crit}} > \underbrace{1 - \frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{1}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}}}_{c_\ell} > 1 + \frac{1}{4\sqrt{N}\log N} \tag{4.4}$$

*where* $\gamma \approx 0.5772$ *is the Euler-Mascheroni constant. Moreover, this bound is nearly tight as*

$$c_{\text{crit}} < \underbrace{1 - \frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{3}{(3N+1)(\log N + \frac{4N-1}{8N^2} + \gamma)}}}_{c_u} \ .$$

This proposition is easily proven by establishing upper and lower bounds on $\psi_N(c)$ but involves some in-depth calculations. We therefore defer the proof to Appendix A.3. Combining these simpler lower bounds on $c_{\text{crit}}$ with Theorem 4.1 yields the following explicit convergence bounds for gradient descent with simple extrapolations.

**Corollary 4.2.** *For any convex $L$-smooth function $f$ with minimizer $x_\star$ and $N > 1$, gradient descent with constant stepsizes $h_k = h \in (0, 1]$ and simple extrapolation by factor*

$$c = 1 - \frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{1}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}} \tag{4.5}$$

*has $x_\sigma = x_0 + c(x_N - x_0)$ satisfy*

$$f(x_\sigma) - f(x_\star) \le \frac{LD^2}{4Nh + \left(\sqrt{1 + \frac{16N^2}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}} - 1\right) h + 2} \tag{4.6}$$

*where $D = \|x_0 - x_\star\|$. As a simpler, weaker bound, setting $c = 1 + \frac{1}{4\sqrt{N \log N}}$ ensures that*

$$f(x_\sigma) - f(x_\star) \le \frac{LD^2}{4Nh + \sqrt{\frac{N}{\log N}} h + 2} \ . \tag{4.7}$$

This result shows that adding a simple extrapolation achieves the same worst-case convergence improvement as taking an additional $O(\sqrt{N/\log N})$ gradient steps. Note this gain increases with $N$ while the cost of the extrapolation step remains fixed (a vector-scalar multiplication and a vector sum). While this is still a lower-order improvement, this level of benefit from such a simple modification is rather surprising.

Finally, given the benefits of simple extrapolation proven above, we next consider its limitations. In the theorem below, we formalize the notion that too large of an extrapolation has a negative effect. An extrapolation that is too large may "overshoot" the minimizer, leading to worse performance. In particular, we show simple extrapolation factors any larger than $1 + O(1/\sqrt{N})$ can perform strictly worse than the known worst-case for gradient descent's last iterate.

**Proposition 4.3.** *There exists a convex $L$-smooth function $f$ with minimizer $x_\star$ such that gradient descent with constant stepsizes $h_k = h \in (0, 1]$ and any*

$$c > \hat{c} := \frac{1 + \sqrt{\frac{1}{2Nh+1}}}{1 - (1-h)^N}$$

*has $x_\sigma = x_0 + c(x_N - x_0)$ satisfy*

$$f(x_\sigma) - f(x_\star) > \frac{LD^2}{4Nh + 2} \ge f(x_N) - f(x_\star) \ .$$

*Proof.* Consider $f(x) = \frac{L}{2}x^2$ and $x_0 = D$. Then gradient descent (1.1) has iterates contract towards zero with $x_N = (1-h)^N D$. As a result, $x_\sigma = x_0 + c(x_N - x_0) = D + c(D(1-h)^N - D)$ and so $f(x_\sigma) = \frac{LD^2}{2}(1 + c((1-h)^N - 1))^2$. Plugging in the assumed lower bound on $c$, it immediately follows that

$$f(x_\sigma) > \frac{LD^2}{2}\left(1 + \frac{1 + \sqrt{\frac{1}{2Nh+1}}}{1 - (1-h)^N}((1-h)^N - 1)\right)^2 = \frac{LD^2}{4Nh + 2} \ .$$

The second claim inequality follows from (C1). □

From the results of Theorem 4.1 and Proposition 4.3, it is clear that there exists some optimal extrapolation factor $c_{\text{opt}} \in [c_{\text{crit}}, \hat{c}]$. Hence the optimal extrapolation factor shrinks at a rate between $1 + \Theta(1/\sqrt{N \log(N)})$ and $1 + \Theta(1/\sqrt{N})$. In Section 5, our results suggest that $c_{\text{opt}}$ closely follows the $1 + \Theta(1/\sqrt{N \log(N)})$ behavior of $c_{\text{crit}}$, see Figure 2. Consequently, we expect the development of more sophisticated worst-case functions than the quadratic above would close this gap.

## 4.2 Proof of Theorem 4.1

Performance estimation provides the foundation for our exact analysis of the impact of simple extrapolation on worst-case guarantees. Recalling our previous notation for the associated PEP problem (2.7), now parameterized by $c$ instead of $\sigma$, we denote

$$p(c) := \begin{cases} \max_{x_0,x_\star,f} & f(x_\sigma) - f(x_\star) \\ \text{s.t.} & f_i \geq f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \qquad \forall i \neq j \in I_N^\star \\ & \|x_0 - x_\star\| \leq D \\ & g_\star = 0 \\ & x_{k+1} = x_k - \frac{h}{L}g_k \qquad k = 0, \dots, N-2 \\ & x_\sigma = x_0 - \sum_{k=0}^{N-1} \frac{ch}{L} g_k \ . \end{cases}$$

Then following the same SDP relaxation (2.8) and applying weak duality, let $d(c)$ denote the upper bounding SDP (2.9), now parameterized by $c$ instead of $\sigma$. To prove our claimed upper bound on $f(x_\sigma) - f(x_\star)$, it then suffices to show the same upper bound on $d(c)$. This can be done by providing any dual feasible solution with an objective value matching our claimed convergence guarantee.

In the special case of $c = 1$ (that is, just reporting the last iterate without extrapolation), [10] proved a bound matching our claim precisely through this process of constructing a dual certificate. Our derivation of a dual certificate can be viewed as a generalization of their original method to simple extrapolations. Many of the steps are nearly identical; in such instances, we will simply refer the reader to [10].

Our construction of candidate dual solutions utilizes the following parameters

$$\begin{aligned} r_i &= \frac{ic}{2Nc - i + 1} \qquad i = 1, \dots, N \\ t &= \frac{L}{2Nhc + 1} \ . \end{aligned} \tag{4.8}$$

Given these, we consider dual solutions to $d(c)$ given by

$$v = \frac{1}{2}t \tag{4.9}$$

$$\lambda_{i,j} = \begin{cases} r_1 & \text{if } i = -1, j = 0 \\ r_{j+1} - r_j & \text{if } i = -1, 1 \leq j \leq N-1 \\ 1 - r_N & \text{if } i = -1, j = N \\ r_j & \text{if } i = j-1, 1 \leq j \leq N \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where for simplicity we let indices $-1$ and $N$ correspond to $\star$ and $\sigma$, respectively. When $c = 1$ and the considered extrapolation disappears, this construction exactly reduces to that of [10], in which the authors use $r_i = \frac{i}{2N-i+1}$ and $t = \frac{L}{2Nh+1}$ to prove the bound $LD^2/(4Nh+2)$.

We claim by construction these satisfy the first linear constraint $\sum_{i\neq j} \lambda_{i,j} a_{i,j} - a_{\star,\sigma} = 0$. Further, we claim that the second linear constraint is satisfied by setting $Z = \frac{1}{2}S_c(r,t)$ where $S_c(r,t)$ is defined as the following block matrix

$$S_c(r,t) = \begin{pmatrix} t & q_c^T \\ q_c & \frac{1-h}{L}Q_c + \frac{h}{L}W_c \end{pmatrix} \tag{4.11}$$

with components given by $q_c = (-r_1, r_1 - r_2, \dots, r_{N-1} - r_N, r_N - 1)^T$,

$$Q_c = \begin{pmatrix} 2r_1 & -r_1 & & & & \\ -r_1 & 2r_2 & -r_2 & & & \\ & -r_2 & 2r_3 & -r_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -r_{N-1} & 2r_N & -r_N \\ & & & & -r_N & 1 \end{pmatrix}, \tag{4.12}$$

and

$$W_c = \begin{pmatrix} 2r_1 & r_2 - r_1 & \dots & r_N - r_{N-1} & c - r_N \\ r_2 - r_1 & 2r_2 & \dots & r_N - r_{N-1} & c - r_N \\ \vdots & & \ddots & & \vdots \\ r_N - r_{N-1} & r_N - r_{N-1} & \dots & 2r_N & c - r_N \\ c - r_N & c - r_N & \dots & c - r_N & 1 \end{pmatrix}. \tag{4.13}$$

For completeness, these two claimed identities are also verified in Appendix A.1. To complete our proof, we verify nonnegativity of $\lambda$ (Section 4.2.1) and positive semidefiniteness of $Z$ (Section 4.2.2) establishing dual feasibility, from which we can apply weak duality to prove our claimed rate (Section 4.2.3). Lastly, we show a Huber function can again be used to show our resulting worst-case bound is the best possible (Section 4.2.4).

**4.2.1 Verification of Nonnegativity of $\lambda$** We can easily confirm that $\lambda_{i,j} \geq 0$ for all $i, j \in I_N^\star$. Clearly $r_i > 0$ for all $i = 1, \dots, N$, provided that $c > \frac{N-1}{2N}$. And similarly, for $c > \frac{N-1}{2N}$, we see that $r_{i+1} - r_i = \frac{c(2Nc+1)}{(2Nc-i)(2Nc-i+1)} > 0$. Lastly, we have $1 - r_N = \frac{Nc-N+1}{2Nc-N+1} > 0$ for $c > \frac{N-1}{N}$. We therefore have that $\lambda \geq 0$ for $c > \frac{N-1}{N}$, and the remaining verification steps will hold for such $c$ as well. However, for the purposes of simple extrapolation, we only consider $c \geq 1$.

**4.2.2 Verification of Positive Semidefiniteness of $Z$** Note it is equivalent to show $S_c$ is positive semidefinite since it is just a rescaling of $Z$. Our first lemma is a generalization of Drori and Teboulle's proof of [10, Lemma 3.3] that shows the matrix $W_c$ is positive semidefinite for $c = 1$.

**Lemma 4.4.** *Let $W_c$ be defined as in* (4.13) *with entries set as in* (4.8) *and denote the $k$-th principal submatrix of $W_c$ by $M_k(c)$. Then for $k = 0, \dots, N-1$,*

$$\det M_k(c) = c^{k+1} \left(1 + \sum_{i=0}^{k} \frac{2Nc - 2k - 1}{2Nc + 4Nci - 2i^2 + 1}\right) \prod_{i=0}^{k} \frac{2Nc + 4Nci - 2i^2 + 1}{(2Nc - i)^2} \tag{4.14}$$

*and*

$$\det M_N(c) = c^N \left(1 - \sum_{i=0}^{N-1} \frac{(c-1)(2Nc - 2N + 1)(2Nc + 1)}{2Nc + 4Nci - 2i^2 + 1}\right) \prod_{i=0}^{N-1} \frac{2Nc + 4Nci - 2i^2 + 1}{(2Nc - i)^2}. \tag{4.15}$$

The proof of this generalized result is deferred to Appendix A.2. This explicit formula for the determinant yields the following result on the positive definiteness of $W_c$.

**Lemma 4.5.** *Let $W_c$ be defined as in* (4.13) *with entries set as in* (4.8). *Then $W_c$ is positive definite if and only if $\psi_N(c) > 0$, with $\psi_N(c)$ defined as in* (4.2)

*Proof.* Recall from Sylvester's criterion that a symmetric matrix is positive definite if and only if all leading principal submatrices of a matrix have positive determinant. We can easily see from (4.14) that for $k < N$, $\det M_k(c)$ is the product of all positive factors, so $\det M_k(c) > 0$. Therefore, by Sylvester's criterion, $W_c$ is positive definite if and only if $\det M_N(c)$. Finally, observe that in (4.15), $\det M_N(c)$ is the product of $\psi_N(c)$ with a sequence of positive factors. We therefore conclude that $\det M_N(c) > 0$ if and only if $\psi_N(c) > 0$ and the result follows. $\square$

We now examine the matrix $Q_c$.

**Lemma 4.6.** *Let $Q_c$ be defined as in* (4.12) *with entries set as in* (4.8)*. Then $Q_c$ is positive definite.*

*Proof.* Consider any $y = (y_0, \dots, y_N)^T \neq 0$. We calculate

$$\begin{aligned} y^T Q_c y &= \sum_{i=0}^{N-1} 2r_{i+1} y_i^2 - 2\sum_{i=0}^{N-1} r_{i+1} y_i y_{i+1} + y_N^2 \\ &= \sum_{i=0}^{N-1} r_{i+1}(y_{i+1} - y_i)^2 + r_1 y_0^2 + \sum_{i=1}^{N-1} (r_{i+1} - r_i) y_i^2 + (1 - r_N) y_N^2 \ . \end{aligned}$$

The positivity of the first and second terms is immediate, and it was shown in Section 4.2.1 that $r_{i+1} - r_i > 0$ and $1 - r_N > 0$. Therefore, we have $y^T Q_c y > 0$. $\square$

Finally, we combine each of these results to prove the following claim

**Lemma 4.7.** *If $\psi_N(c) > 0$ then $S_c(r,t)$ is positive semidefinite.*

*Proof.* For simplicity, we will denote $S_c = S_c(r,t)$. Since $\psi_N(c) > 0$, by Lemmas 4.5 and 4.6, we know that both $Q_c$ and $W_c$ are positive definite. Observe that for $h \in (0,1]$, $\frac{1-h}{L}Q_c + \frac{h}{L}W_c$ must be positive definite as it is a convex combination of positive definite matrices. We can therefore consider the Schur complement of $\frac{1-h}{L}Q_c + \frac{h}{L}W_c$ in $S_c$. From a well-known result on the Schur complement, we know that $S_c$ is positive semidefinite if and only if $t - q_c^T \left(\frac{1-h}{L}Q_c + \frac{h}{L}W_c\right)^{-1} q_c \geq 0$. However, we also have the standard identity

$$\det S_c = \left(t - q_c^T \left(\frac{1-h}{L}Q_c + \frac{h}{L}W_c\right)^{-1} q_c\right) \det\left(\frac{1-h}{L}Q_c + \frac{h}{L}W_c\right) \tag{4.16}$$

Next define the vector $u_c = (\frac{2Nhc+1}{L}, 1, \dots, 1)^T$. It is easily verified that $S_c u_c = 0$ (see `Mathematica proof 4.2`). Consequently, $\det S_c = 0$. But $\det\left(\frac{1-h}{L}Q_c + \frac{h}{L}W_c\right) > 0$, so from (4.16) we must have

$$\left(t - q_c^T \left(\frac{1-h}{L}Q_c + \frac{h}{L}W_c\right)^{-1} q_c\right) = 0.$$

Therefore, $S_c$ is positive semidefinite. $\square$

All that remains is to show selecting $c < c_{\text{crit}}$ ensures $\psi_N(c) > 0$. For $c > 1$, the denominator $2Nc + 4Nci - 2i^2 + 1$ is always positive, so $\psi_N(c)$ is a continuous rational polynomial. And note that $\psi_N(1) = 1 > 0$. A quick calculation verifies that $\psi_N'(c) < 0$ for all $c > 1$ with $\lim_{c\to\infty} \psi_N(c) = -\infty$. From this, $\psi_N(c)$ has exactly one root larger than 1, which must be $c_{\text{crit}}$. Moreover, by continuity, $\psi_N(c) \geq 0$ for $c \in [1, c_{\text{crit}}]$ and $\psi_N(c) < 0$ for $c > c_{\text{crit}}$.

#### 4.2.3 Deriving Claimed Extrapolation Guarantee from Weak Duality

Since our proposed dual certificate is feasible, using that $v = \frac{1}{2}t = \frac{L}{4Nhc+2}$ as in (4.9), we conclude that $d(c) \le vD^2 = \frac{LD^2}{4Nhc+2}$. Then, by applying weak duality, for any $L$-smooth convex $f$,

$$f(x_\sigma) - f(x_\star) \le p(c) \le d(c) \le \frac{LD^2}{4Nhc+2} \ .$$

#### 4.2.4 Claimed Extrapolation Guarantee is Tight

Finally, we demonstrate that our bound (4.3) is tight. Our proof once again is an adjustment of the tight example in [10, Theorem 3.2] to account for the extrapolation $c$. Consider $x_0 = D$ and $f = \phi_{L,\eta}$ with $\eta = D/(2Nhc+1)$. Then gradient descent's iterates are $x_k = D(1 - kh\eta)$ for $k = 1, \dots, N$. Taking a simple extrapolation yields

$$x_\sigma = x_0 + c(x_N - x_0) = D - Nhc\eta = (Nhc+1)\eta \ .$$

Since $x_\sigma > \eta$, we can therefore evaluate

$$f(x_\sigma) - f(x_\star) = f(x_\sigma) - 0 = L\eta^2(Nhc+1) - \frac{L\eta^2}{2} = \frac{LD^2}{4Nhc+2} \ .$$

#### 4.2.5 Remarks on Assumptions of the Proof of Theorem 4.1

We briefly draw attention to our claim in Section 4.2.2 above that $\psi_N(c) < 0$ for $c > c_{\text{crit}}$. This fact shows that our particular certificate construction breaks down for $c > c_{\text{crit}}$. Indeed, our numerical experiments in Section 5 indicate that (4.3) is, in fact, false for $c > c_{\text{crit}}$. Similarly, our certificate construction breaks down for constant stepsizes beyond the assumed $h \le 1$. If $h > 1$, then we can no longer guarantee that $\frac{1-h}{L}Q_c + \frac{h}{L}W_c$ is positive definite. A different family of dual certificates would be needed to provide guarantees under extrapolation with stepsizes longer than one.

## 5 Numerical Extensions

Beyond enabling the preceding proofs, the associated performance estimation problems enable numerically surveying the effects of averaging and extrapolation. Here, we address three extensions of our theory, utilizing `Mosek` [27] via `JuMP` [28] for any numerical solves. First, Section 5.1 shows results paralleling those of Section 4 appear to hold when considering the worst-case gradient norm rather than the worst-case objective gap. For both of these settings, we compute the simple extrapolation factor minimizing the worst-case performance in both the objective gap and gradient norm, finding our theory nearly matches the optimal factor. Then, in contrast to the many prior work finding one-dimensional functions characterize the worst-case behavior of gradient descent's last iterate, Section 5.2 computes its worst-case performance with and without a restriction to one-dimensional problems, showing these are not sufficient to describe the performance of extrapolations. Lastly, Section 5.3 surveys the effect of simple extrapolations across a range of (accelerated) gradient methods, finding consistent, small benefits.

Recall the PEP problem (2.7) can be written equivalently as the SDP (2.8) provided the dimension $m$ of functions $f$ considered is at least $N+2$. Consequently, if no restriction is made on the dimension of the considered problems (i.e., high-dimension examples are allowed), we can numerically compute optimal solutions to the PEP problem via any commercial interior point method solver. Here, we consider both the worst case measured in terms of the objective gap at $x_\sigma$ as well as the gradient norm $\|g_\sigma\| = \|\nabla f(x_\sigma)\|$. To distinguish these, we denote the performance estimation problem maximizing the reported point's objective gap (as was considered throughout Section 4) by

$$p_{\text{obj}}(c) = \begin{cases} \max_{F,G} & Fa_{\star,\sigma} \\ \text{s.t.} & Fa_{i,j} + \mathrm{Tr} GA_{i,j}(h) + \frac{1}{2L}\mathrm{Tr} GC_{i,j} \le 0 \qquad \forall i \neq j \in I_N^\star \\ & G \succeq 0 \\ & \mathrm{Tr} GB_{0,\star} \le D^2 \ . \end{cases}$$

We denote the alternative performance estimation problem maximizing the reported point's gradient norm, formulated as an SDP, by

$$p_{\text{grad}}(c) = \begin{cases} \max_{F,G} & \mathrm{Tr} GC_{\star,\sigma} \\ \text{s.t.} & Fa_{i,j} + \mathrm{Tr} GA_{i,j}(h) + \frac{1}{2L}\mathrm{Tr} GC_{i,j} \le 0 \qquad \forall i \neq j \in I_N^\star \\ & G \succeq 0 \\ & \mathrm{Tr} GB_{0,\star} \le D^2 \ . \end{cases}$$

In both cases, the equalities above hold by virtue of assuming $m \ge N+2$.

## 5.1 Optimal Extrapolations for Objective Gap and Gradient Norm

Numerically, we find much of our results on the benefits of extrapolation on the worst-case final objective gap carry over to the worst-case final gradient norm. However, the exact dual certificates $v, \lambda, Z$ underlying our proof do not constitute proofs for the setting of gradient norms. We expect dual certificates proving a benefit from small extrapolations exist but have not identified their structure and hence leave proving such parallel results open. Theory similar to the H-duality theory of [29] may facilitate doing so without repeating and re-engineering the proof.

In Figure 2, we plot both $p_{\text{obj}}$ and $p_{\text{grad}}$ as the extrapolation factors varies for fixed $N = 7$, $h = 1$, and $L = D = 1$. In overall structure, these two functions look very similar. For $c$ in an interval near one, Theorem 4.1 ensures $p_{\text{obj}}(c) = LD^2/(4Nhc+2)$. We see similarly in the gradient norm case that an interval near one has $p_{\text{grad}}(c) = LD/(Nhc+1)$, improving on its tight last iterate bound [8] of $\|\nabla f(x_N)\| \le \frac{LD}{Nh+1}$. Reasoning mirroring the tightness of Huber functions in Theorem 4.1 provides an example attaining this conjectured gradient norm rate under extrapolation. Let $x_0 = D$ and again consider $f = \phi_{L,\eta}$ as in (2.3) with $\eta = \frac{D}{Nhc+1}$. A simple calculation shows that $x_N = D - Nh\eta$ and

$$x_\sigma = x_0 + c(x_N - x_0) = D - \frac{DNhc}{Nhc+1} = \frac{D}{Nhc+1} = \eta \ .$$

We then have $\|g_\sigma\| = \|\nabla f(\eta)\| = \frac{LD}{Nhc+1}$.

We consider two notable values of $c$ for each performance measure, the critical point $c_{\text{crit}}$ where $p(c)$ deviates from our simple formulas (which was discussed substanially in Section 4) and the point $c_{\text{opt}}$ minimizing $p(c)$. We denote these by

$$c_{\text{crit,obj}} = \sup\left\{ c \ge 1 \mid p_{\text{obj}}(c) = \frac{LD^2}{4Nhc+2} \right\} \ , \qquad c_{\text{opt,obj}} = \operatorname{argmin}\, p_{\text{obj}}(c) \ ,$$

$$c_{\text{crit,grad}} = \sup\left\{ c \ge 1 \mid p_{\text{grad}}(c) = \frac{LD}{Nhc+1} \right\} \ , \qquad c_{\text{opt,grad}} = \operatorname{argmin}\, p_{\text{grad}}(c) \ .$$

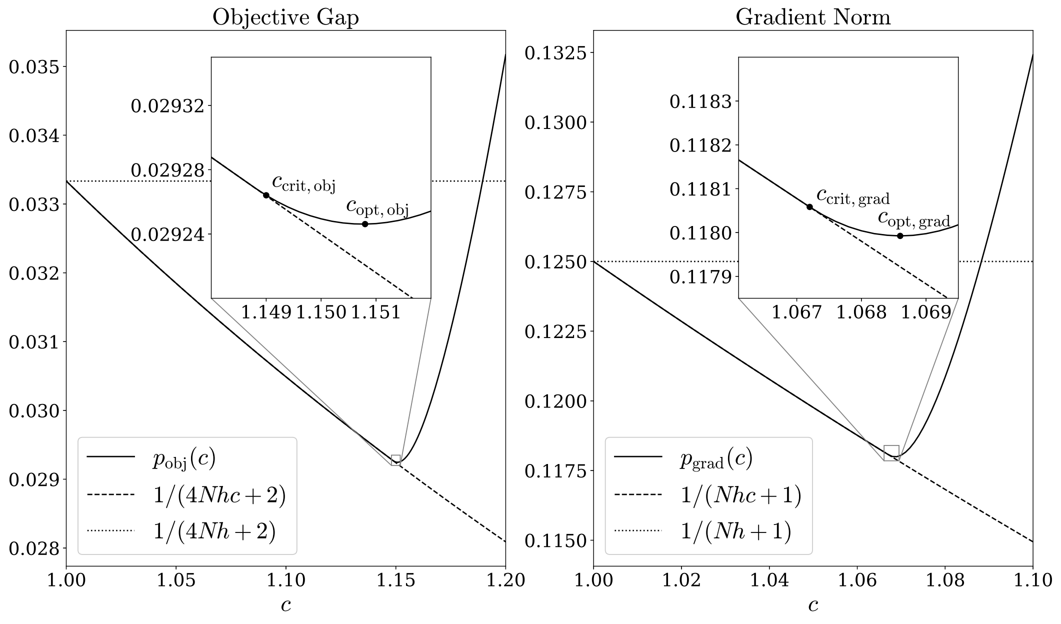

Figure 2: Near-optimality of $c_{\text{crit}}$ for objective gap and gradient norm with $N = 7, L = D = h = 1$.

For both performance measures in Figure 2, we see the critical point characterized by our theory is nearly but not quite optimal. Contrasting these measures, note that $c_{\text{crit,grad}} \neq c_{\text{crit,obj}}$, with $c_{\text{crit,grad}}$ consistently less than $c_{\text{crit,obj}}$, as shown in Table 4. This indicates that extrapolations should be more conservative when seeking a small gradient norm. Moreover, numerically, we observe a faster rate of decay for $c_{\text{crit,grad}}$ as compared to $c_{\text{crit,obj}}$; this is clearly illustrated in Figure 3. Specifically, we see that $c_{\text{crit,grad}}$ appears to be shrinking at rate $1 + O(1/N)$, motivating the following conjecture.

**Conjecture 5.1.** *Consider a gradient descent algorithm of fixed length $N$, constant stepsize $h \in (0, 1]$, and with $\|x_0 - x_\star\| = D$. There exists $c_{\text{crit,grad}} = 1 + \Theta(1/N)$ such that for any $c \in [1, c_{\text{crit,grad}}]$, simple extrapolation by factor $c$ has a tight worst-case bound of $\|\nabla f(x_\sigma)\| \leq \frac{LD}{Nhc+1}$.*

If this conjecture is correct, simple extrapolations would be a provably less effective strategy for reducing the reported gradient norm than for the objective gap. Whereas an extrapolation of size $1 + \Theta(1/\sqrt{N \log(N)})$ gave the same improvement in objective value as $\Theta(\sqrt{N/\log(N)})$ additional gradient steps, a simple extrapolation of size $1 + \Theta(1/N)$ would only improve the gradient norm guarantee by the same amount as a constant number of additional gradient steps.

## 5.2 Limitations of One-Dimensional Worst-Case Functions

For $c \leq c_{\text{crit}}$, we have shown in Theorem 4.1 that there exists a one-dimensional function that attains our worst-case performance. Similarly, regarding averaging, in Section 3.2, we showed that one-dimensional functions were sufficient to describe the worst-case behavior of any averaging scheme $\sigma$. A natural question to ask in light of this is whether the same is true for simple extrapolation, that is, whether one-dimensional functions are sufficient for attaining the worst-case behavior for all simple extrapolations. To answer this, we can modify our original SDP (2.8) by adding an additional

| $N$ | $c_{\text{crit,obj}}$ | $c_{\text{opt,obj}}$ | $c_{\text{crit,grad}}$ | $c_{\text{opt,grad}}$ |
|---|---|---|---|---|
| 1 | 1.5 | 1.5 | 1.4142 | 1.4142 |
| 5 | 1.1800 | 1.1821 | 1.0931 | 1.0950 |
| 10 | 1.1220 | 1.1238 | 1.0471 | 1.0481 |
| 25 | 1.0739 | 1.0752 | 1.0182 | 1.0187 |
| 50 | 1.0507 | 1.0516 | 1.0086 | 1.0090 |

Table 4: The critical and optimal extrapolation factors for objective gap and gradient norm.

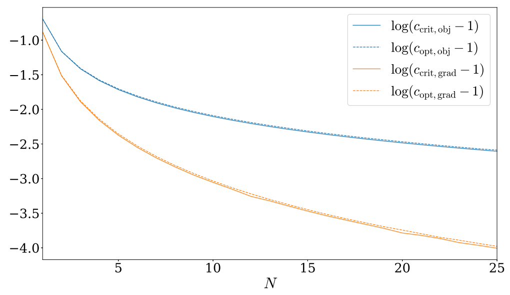

Figure 3: Similarity of critical and optimal factors and divergence between objective and gradient.

constraint that $G = ww^T$ for some vector $w \in \mathbb{R}^{N+2}$. This yields the problem

$$p_{\text{obj,1D}}(c) := \begin{cases} \max_{F,w} & Fa_{\star,\sigma} \\ \text{s.t.} & Fa_{i,j} + \text{Tr}GA_{i,j}(h) + \frac{1}{2L}\text{Tr}GC_{i,j} \le 0, \quad \forall i \ne j \in I_N^\star \\ & G \succeq 0 \\ & \text{Tr}GB_{0,\star} \le D^2 \\ & G = ww^T \; . \end{cases}$$

Due to the rank-one constraint, this problem becomes nonconvex, so we must now perform global rather than local optimization. However, for small $N$, this is still computationally feasible. In Figure 4, we plot our results for various $N$ with $h = 1$. For $c < c_{\text{crit}}$, we already know the worst-case function is $\phi_{L,\eta}$ from Theorem 4.1. Figure 4 shows that for $c > c_{\text{crit}}$, the worst-case performance, in general, deviates from the worst-case performance on one-dimensional objectives. As a result, our theory in Theorem 4.1 and Proposition 4.3 appears to capture the full range of extrapolations characterized by one-dimensional worst-case problem instances. Hence, characterizing the optimal factor $c_{\text{opt}}$ and measuring the (small) gap between it and $c_{\text{crit}}$ will fundamentally require the design of worst-case problem instances beyond Huber functions and quadratics.

The dimension of problems required can be predicted by examining the rank of optimal primal solutions $G$ of (2.8). To see this, recall that the columns of the Cholesky decomposition, $G = H^T H$, gives the gradient vectors seen in that worst-case problem instance. Numerically, even for large $c$, we find the optimal $G$ is rank two, indicating the worst-case problem instances for $c > c_{\text{crit}}$ only require one additional dimension.

Our numerical results indicate that among one-dimensional functions, the worst-case is always a Huber function or a quadratic function. This can be seen in Figure 4 as $p_{\text{obj,1D}}$ appears to be piecewise smooth with two distinct pieces. The first piece, as discussed above, numerically equals $LD^2/(4Nhc+2)$, which is attained by considering $\phi_{L,\eta}$. The second piece numerically matches the curve $\frac{LD^2}{2}(1-c+c(1-h)^N)^2$; this is the exact value achieved by simple extrapolation on the quadratic function $f(x) = \frac{L}{2}x^2$. The analogous result for gradient norm seems to hold numerically as well. Therefore, in one dimension, the worst-case instances of simple extrapolation seem to be easily described. Setting $h = 1$ and computing the minimum of these two functions, the best extrapolation factor in one-dimension is $c = 1 - \frac{1}{4N} + \sqrt{\frac{1}{2N} + \frac{1}{16N^2}}$ which attains a convergence rate of $LD^2/(4N + \sqrt{8N+1} + 1)$. This provides a lower bound on the magnitude of rate improvement that simple extrapolation that can achieve among problems of any dimension, which is within a log

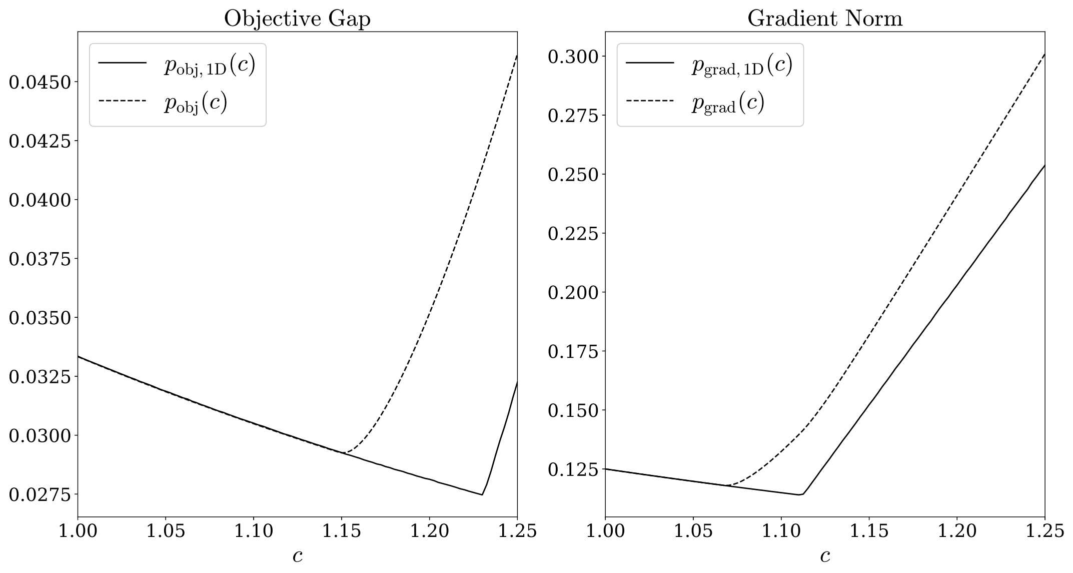

Figure 4: Comparison of PEP restricted to one-dimensional functions with $N = 7$, $L = D = h = 1$.

term of our Corollary 4.2.

## 5.3 Application of Simple Extrapolations to Other Algorithms

Our discussion of extrapolation up to this point has focused on gradient descent with constant stepsizes $h_k = h \in (0, 1]$. This simplifying assumption enabled Section 4's exact characterization of the impact of extrapolation on worst-case performance. As our final numerical extension, we relax this restriction and consider the impact of simple extrapolation on a range of common first-order methods. In particular, we consider the collection of gradient descent stepsize selections introduced in Section 3.1, Nesterov's classic accelerated method [18], Polyak's heavy ball method [30], and the Optimal Gradient Method [19]. The performance estimation SDP (2.8) was previously defined explicitly for gradient descent via the construction of our $\mathbf{x_i}$ vectors, and consequently the matrices $A_{i,j}(h)$ and $B_{i,j}(h)$. This restriction is not fundamental as PEP can apply to any gradient method by adjusting the definition of $\mathbf{x_i}$ to match the method's update scheme.

Although PEP techniques can also naturally generalize to describe proximal or projected gradient methods, extrapolations do not make sense in such settings. Reporting a point outside the convex hull of the algorithm's iterates may set $x_\sigma$ as an infeasible point. Consequently, we focus on unconstrained problems.

The range of methods considered possesses a range of different orders of convergence guarantees for their last iterate. The gradient descent schemes with stepsize bounded above by two and Polyak's heavy ball method [31] all have the last iterates objective gap converges at a rate of $O(1/T)$ in the worst case. Gradient descent with silver stepsizes [9] has the last iterate's objective gap converge at rate $O(1/T^{1.2716})$. The last iterate of Nesterov's accelerated method [18] attains the optimal order of convergence among all gradient methods of $O(1/T^2)$. Even better, the Optimal Gradient Method (OGM) [19] (discussed using PEP techniques) attains the best possible $O(1/T^2)$ rate exactly among all gradient methods, see [32] for the matching lower bound.

Figure 5 shows the worst-case objective gap for each considered algorithm as the simple ex-

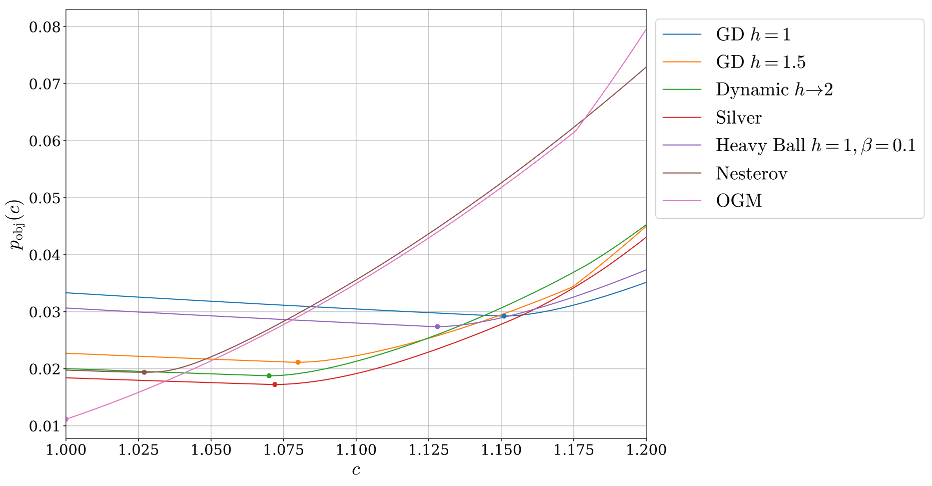

Figure 5: Effect of Simple extrapolation on worst-case reported objective gap of various gradient methods with $N = 7$, $L = D = 1$. The optimal choice of $c$ for each algorithm is marked by a dot.

| N | GD | GD | Dynamic | Silver | Heavy Ball | Nesterov | OGM |
|---|---|---|---|---|---|---|---|
| | $h = 1$ | $h = 1.5$ | $h \to 2$ | | $h = 1, \beta = 0.1$ | | |
| 3 | 1.2447 | 1.1033 | 1.0884 | 1.1029 | 1.1855 | 1.1254 | 1.0 |
| 7 | 1.1508 | 1.0795 | 1.0703 | 1.0718 | 1.1279 | 1.0267 | 1.0 |
| 15 | 1.0991 | 1.0585 | 1.0519 | 1.0454 | 1.0868 | 1.0065 | 1.0 |
| 31 | 1.0667 | 1.0419 | 1.0372 | 1.0284 | 1.0596 | 1.0017 | 1.0 |
| 63 | 1.0454 | 1.0297 | 1.0275 | 1.0181 | 1.0407 | 1.0002 | 1.0 |

Table 5: Numerically computed optimal simple extrapolation factor $c_{\text{opt}}$ for various methods and $N$.

trapolation factor varies with $N = 7$ and $L = D = 1$. We can see that every method except OGM has a non-negligible convergence improvement from applying a simple extrapolation. This holds even for Nesterov acceleration, which already has an optimal convergence order. Note that, as one should expect, extrapolation cannot improve the convergence rate of OGM as the method is already exactly optimal. While exact comparisons between the optimal amount of extrapolation for each algorithm remain dependent on $N$, the general ordering of when each method stops benefiting from extrapolation remains fairly consistent. To this end, Table 5 shows the numerically computed optimal extrapolation factor for each method up to $N = 63$. This shows a widespread (small) benefit from simple extrapolation and hints at the potential broad applicability of this as a simple, effectively-free (in computation and memory costs) postprocessing step for smooth convex optimization.

**Acknowledgements.** This work was supported in part by the Air Force Office of Scientific Research under award number FA9550-23-1-0531. We thank Adrien Taylor for providing kind and direct feedback on this work's positioning among the growing PEP literature.

## References

[1] Moslem Zamani and François Glineur. Exact convergence rate of the last iterate in subgradient methods. *arXiv:2307.11134*, 2023.

[2] Guanghui Lan. *First-order and Stochastic Optimization Methods for Machine Learning.* Springer, 2020.

[3] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page I–71–I–79, 2013.

[4] E. Gustavsson, M. Patriksson, and AB Strömberg. Primal convergence from dual subgradient methods for convex optimization. *Mathematical Programming*, 150:365–390, 2015.

[5] Benjamin Grimmer and Danlin Li. Some primal-dual theory for subgradient methods for strongly convex optimization. *arXiv:2305.17323*, 2024.

[6] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, 1965.

[7] Vien V. Mai and Mikael Johansson. Anderson acceleration of proximal gradient methods. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, 2020.

[8] Marc Teboulle and Yakov Vaisbourd. An elementary approach to tight worst case complexity analysis of gradient based methods. *Mathematical Programming*, 2022.

[9] Jason M. Altschuler and Pablo A. Parrilo. Acceleration by stepsize hedging ii: Silver stepsize schedule for smooth convex optimization. *arXiv:2309.16530*, 2023.

[10] Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: A novel approach. *Mathematical Programming*, 145(1–2):451–482, 2014.

[11] Adrien Taylor, Julien Hendrickx, and François Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161:307–345, 2017.

[12] Baptiste Goujaud, Céline Moucer, François Glineur, Julien Hendrickx, Adrien Taylor, and Aymeric Dieuleveut. PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python. *arXiv preprint arXiv:2201.04040*, 2022.

[13] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, 27(3):1283–1313, 2017.

[14] Shuvomoy Das Gupta, Bart P.G. Van Parys, and Ernest K. Ryu. Branch-and-bound performance estimation programming: A unified methodology for constructing optimal optimization method. *Mathematical Programming*, 2023.

[15] Benjamin Grimmer. Provably faster gradient descent via long steps. *arXiv:2307.06324*, 2023.

[16] Benjamin Grimmer, Kevin Shu, and Alex L. Wang. Accelerated gradient descent via long steps. *arXiv:2309.09961*, 2023.

[17] Jason Altschuler. *Greed, hedging, and acceleration in convex optimization.* PhD thesis, Massachusetts Institute of Technology, 2018.

[18] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

[19] Donghwan Kim and Jeffrey A. Fessler. Optimized first-order methods for smooth convex minimization. *Mathematical Programming*, 159(1–2):81–107, 2016.

[20] Wolfram Research, Inc. Mathematica, Version 14.0. Champaign, IL, 2024.

[21] Yoel Drori and Adrien B. Taylor. Efficient first-order methods for convex minimization: a constructive approach. *Mathematical Programming*, 184(1–2):183–220, 2020.

[22] Felix Lieder. *Projection Based Methods for Conic Linear Programming.* PhD thesis, Heinrich-Heine-Universität Düsseldorf, 2018.

[23] Yoel Drori. *Contributions to the complexity analysis of optimization algorithms.* PhD thesis, Tel-Aviv University, 2014.

[24] Guoyong Gu and Junfeng Yang. Tight sublinear convergence rate of the proximal point algorithm for maximal monotone inclusion problems. *SIAM Journal on Optimization*, 30(3):1905–1921, 2020.

[25] Donghwan Kim. Accelerated proximal point method for maximally monotone operators. *Math. Program.*, 190(1–2):57–87, nov 2021.

[26] Mathieu Barré, Adrien Taylor, and Francis Bach. Principled analyses and design of first-order methods with inexact proximal operators. *Mathematical Programming*, 201:185–230, 2023.

[27] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[28] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[29] Jaeyeon Kim, Asuman Ozdaglar, Chanwoo Park, and Ernest K. Ryu. Time-reversed dissipation induces duality between minimizing gradient norm and function value. *arXiv:2305.06628*, 2023.

[30] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[31] Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson. Global convergence of the heavy-ball method for convex optimization. In *2015 European Control Conference (ECC)*, pages 310–315, 2015.

[32] Yoel Drori. The exact information-based complexity of smooth convex minimization. *J. Complex.*, 39:1–16, 2016.

## A Deferred Proofs

### A.1 Verifying Construction of Section 4.2

First we claim $\sum_{i\neq j}\lambda_{i,j}a_{i,j}-a_{\star,\sigma}=0$. This is equivalent to

$$\sum_{j=-1}^{N}\lambda_{j,k}-\sum_{j=-1}^{N}\lambda_{k,j}=\begin{cases}-1 & \text{if } k=-1\\ 1 & \text{if } k=N\\ 0 & \text{otherwise}\end{cases}.$$

One can easily verify this as

$$\begin{aligned}
&\sum_{j=-1}^{N}\lambda_{j,-1}-\sum_{j=-1}^{N}\lambda_{-1,j}=0-(r_1+(r_2-r_1)+\cdots+(r_N-r_{N-1})+(1-r_N))=-1\\
&\sum_{j=-1}^{N}\lambda_{j,0}-\sum_{j=-1}^{N}\lambda_{0,j}=r_1-r_1=0\\
&\sum_{j=-1}^{N}\lambda_{j,k}-\sum_{j=-1}^{N}\lambda_{k,j}=(r_{k+1}-r_k+r_k)-r_{k+1}=0 \qquad \forall k=1,\dots,N-1\\
&\sum_{j=-1}^{N}\lambda_{j,N}-\sum_{j=-1}^{N}\lambda_{N,j}=(1-r_N+r_N)-0=1\ .
\end{aligned}$$

Our second claim is that $Z = \frac{1}{2}S_c(r,t)$. By construction, we can rewrite $S_c(r,t)$ as

$$S_c(r,t)_{i,j} = \begin{cases} t & \text{if } i=j=-1 \\ -r_1 & \text{if } i=-1, j=0 \text{ or } i=0, j=-1 \\ r_j - r_{j+1} & \text{if } i=-1, 1 \le j \le N-1 \text{ or } 1 \le i \le N-1, j=-1 \\ r_N - 1 & \text{if } i=-1, j=N \text{ or } i=N, j=-1 \\ \frac{h}{L}(c - r_N) & \text{if } i=N, 0 \le j \le N-2 \text{ or } 0 \le i \le N-2, j=N \\ \frac{2r_{j+1}}{L} & \text{if } 0 \le i=j \le N-1 \\ \frac{1}{L} & \text{if } i=j=N \\ \frac{1}{L}(hr_{j+1} - r_j) & \text{if } i=j+1 \le N-1 \text{ or } j=i+1 \le N-1 \\ \frac{1}{L}(hc - r_N) & \text{if } i=N, j=N-1 \text{ or } i=N-1, j=N \\ \frac{h}{L}(r_{j+1} - r_j) & \text{if } j+2 \le i \le N-1 \text{ or } i+2 \le j \le N-1\ . \end{cases} \tag{A.1}$$

Now using our selected dual variables (4.9), and expanding our special matrices, we can write

$$\begin{aligned} Z &= vB_{0,\star} + r_1\left(A_{\star,0} + \frac{1}{2L}C_{\star,0}\right) + (1-r_N)\left(A_{\star,\sigma} + \frac{1}{2L}C_{\star,\sigma}\right) + \sum_{k=1}^{N-1}(r_{k+1}-r_k)\left(A_{\star,k} + \frac{1}{2L}C_{\star,k}\right) \\ &\quad + \sum_{k=1}^{N-1} r_k\left(A_{k-1,k} + \frac{1}{2L}C_{k-1,k}\right) + r_N\left(A_{N-1,\sigma} + \frac{1}{2L}C_{N-1,\sigma}\right) \\ &= \frac{t}{2}\mathbf{x_0}\mathbf{x_0}^T + \frac{r_1}{2}\left(\mathbf{g_0}\mathbf{g_0}^T - (\mathbf{x_0}\mathbf{g_0}^T + \mathbf{g_0}\mathbf{x_0}^T)\right) \\ &\quad + \frac{1-r_N}{2}\left(\frac{1}{L}\mathbf{g_\sigma}\mathbf{g_\sigma}^T - (\mathbf{x_0}\mathbf{g_\sigma}^T + \mathbf{g_\sigma}\mathbf{x_0}^T) + \frac{ch}{L}\sum_{l=0}^{N-1}(\mathbf{g_\sigma}\mathbf{g_l}^T + \mathbf{g_l}\mathbf{g_\sigma}^T)\right) \\ &\quad + \sum_{k=0}^{N-1}\frac{r_{k+1}-r_k}{2}\left(\frac{1}{L}\mathbf{g_k}\mathbf{g_k}^T - (\mathbf{x_0}\mathbf{g_k}^T + \mathbf{g_k}\mathbf{x_0}^T) + \frac{h}{L}\sum_{l=0}^{k-1}(\mathbf{g_k}\mathbf{g_l}^T + \mathbf{g_l}\mathbf{g_k}^T)\right) \\ &\quad + \sum_{k=1}^{N-1}\frac{r_k}{2L}\left((h-1)(\mathbf{g_{k-1}}\mathbf{g_k}^T + \mathbf{g_k}\mathbf{g_{k-1}}^T) + \mathbf{g_{k-1}}\mathbf{g_{k-1}}^T + \mathbf{g_k}\mathbf{g_k}^T\right) \\ &\quad + \frac{r_N}{2L}\left(\mathbf{g_{N-1}}\mathbf{g_{N-1}}^T + \mathbf{g_\sigma}\mathbf{g_\sigma}^T + (hc-1)(\mathbf{g_{N-1}}\mathbf{g_\sigma}^T + \mathbf{g_\sigma}\mathbf{g_{N-1}}^T) + h(c-1)\sum_{l=0}^{N-2}(\mathbf{g_l}\mathbf{g_\sigma}^T + \mathbf{g_\sigma}\mathbf{g_l}^T)\right)\ . \end{aligned}$$

It is now straightforward to check that $Z_{i,j} = \frac{1}{2}S_c(r,t)_{i,j}$ in each of the cases in (A.1).

## A.2 Proof of Determinant Formula (Lemma 4.4)

For simplicity, we will follow the same notation used by [10, Lemma 3.3], but we will assign more generalized values. However, note many of these variables overlap with other unrelated concepts in this paper, so these definitions should be recognized only in the context of this proof.

First, observe that $W_c$ has a very specific structure. Using this structure, we can write the $k$-th principal submatrix for $M_k$ as

$$M_k = \begin{pmatrix} d_0 & a_1 & a_2 & \dots & a_{k-1} & a_k \\ a_1 & d_1 & a_2 & & a_{k-1} & a_k \\ a_2 & a_2 & d_2 & & a_{k-1} & a_k \\ \vdots & & & \ddots & & \vdots \\ a_{k-1} & a_{k-1} & a_{k-1} & & d_{k-1} & a_k \\ a_k & a_k & a_k & \dots & a_k & d_k \end{pmatrix}$$

and we consider its determinant. Drori and Teboulle derived the following recursion equation

$$\det M_k = \left( d_k - \frac{2a_k^2}{a_{k-1}} + \frac{a_k^2 d_{k-1}}{a_{k-1}^2} \right) \det M_{k-1} - a_k^2 \left(1 - \frac{d_{k-1}}{a_{k-1}}\right)^2 \det M_{k-2} \tag{A.2}$$

with $\det M_0 = d_0$ and $\det M_1 = d_0 d_1 - a_1^2$. We set

$$\begin{aligned} d_i &= 2r_{i+1} = \frac{2c(i+1)}{2Nc-i} && i = 0, \dots, N-1 \\ d_N &= 1 \\ a_i &= r_{i+1} - r_i = \frac{(i+1)c}{2Nc-i} - \frac{ic}{2Nc-i+1} && i = 1, \dots, N-1 \\ a_N &= c - r_N = c - \frac{Nc}{2Nc-N+1} \ . \end{aligned}$$

Then denoting $\alpha_k = d_k - \frac{2a_k^2}{a_{k-1}} + \frac{a_k^2 d_{k-1}}{a_{k-1}^2}$ and $\beta_k = a_k^2 \left(1 - \frac{d_{k-1}}{a_{k-1}}\right)^2$, for $k = 0, \dots, N-1$, we have

$$\alpha_k = 4c \frac{((2Nc+1)k - k^2 - 1)}{(2Nc-k)^2} \tag{A.3}$$

and

$$\beta_k = \frac{c^2 \left(4kNc - 2Nc - 2k^2 + 4k - 1\right)^2}{(2Nc-k)^2 (2Nc-k+1)^2} \ . \tag{A.4}$$

The recursive equation (A.2) can then be expressed nicely as

$$\det M_k = \alpha_k \det M_{k-1} - \beta_k \det M_{k-2} \ . \tag{A.5}$$

We further define for $i = 0, \dots, N-1$,

$$\begin{aligned} f_i &= c^{i+1} \\ g_i &= 2Nc - 2i - 1 \\ x_i &= \frac{1}{2Nc + 4Nci - 2i^2 + 1} \\ y_i &= \frac{2Nc + 4Nci - 2i^2 + 1}{(2Nc-i)^2} \ . \end{aligned}$$

Next, we verify that (4.14) holds for the base cases, $M_0$ and $M_1$:

$$\begin{aligned}
\det M_0 &= c\left(1+\frac{2Nc-1}{2Nc+1}\right)\frac{2Nc+1}{(2Nc)^2}\\
&= \frac{1}{N} = d_0\\
\det M_1 &= c^2\left(1+\frac{2Nc-3}{2Nc+1}+\frac{2Nc-3}{6Nc-1}\right)\left(\frac{2Nc+1}{(2Nc)^2}\frac{6Nc-1}{(2Nc-1)^2}\right)\\
&= \frac{28N^2c^2-20Nc-1}{4N^2(2Nc-1)^2} = d_0d_1 - a_1^2 \ .
\end{aligned}$$

Now, we claim the recurrence (A.5) ensures that for $k = 0, \ldots, N-1$,

$$\det M_k = f_k\left(1+g_k\sum_{i=0}^{k}x_i\right)\prod_{i=0}^{k}y_i \ .$$

This result follows from the exact argument in [10, Lemma 3.3], using our updated values defined above. This completes our proof for $k < N$.

Finally, we consider $M_N$. Note that the formulas for $d_N$ and $a_N$ differ from their counterparts for $k < N$. So to derive an equation for $M_N$, we simply apply the recursion (A.5) to $M_{N-1}$ and $M_{N-2}$ and simplify. We compute $\alpha_N$ and $\beta_N$ as

$$\begin{aligned}
\alpha_N &= d_N - \frac{2a_N^2}{a_{N-1}} + \frac{a_N^2 d_{N-1}}{a_{N-1}^2} = \frac{\zeta(c)}{(2Nc+1)^2}\\
\beta_N &= a_N^2\left(1-\frac{d_{N-1}}{a_{N-1}}\right)^2 = (2Nc-2N+1)^2\,\frac{c^2\left(4N^2c-2Nc-2N^2+4N-1\right)^2}{(2Nc+1)^2(2Nc-N+1)^2}
\end{aligned}$$

where

$$\begin{aligned}
\zeta(c) = 16N^3(N-1)c^4 - 8N^2(5N^2-9N+4)c^3 + 4N(8N^3-22N^2+20N-5)c^2\\
- 2(4N^4-16N^3+21N^2-13N+2)c + 1 \ .
\end{aligned}$$

Applying our recursion (A.5), and after significant simplification (see `Mathematica proof A.2` for verification), we arrive at our desired formula

$$\det M_N(c) = \det W_c = c^N\left(1-(c-1)(2Nc-2N+1)(2Nc+1)\sum_{i=0}^{N-1}x_i\right)\prod_{i=0}^{N-1}y_i \ .$$

## A.3 Proof of Proposition 4.1

Recall our definition of $\psi_N(c)$:

$$\psi_N(c) = 1 - \sum_{i=0}^{N-1}\frac{(c-1)(2Nc-2N+1)(2Nc+1)}{2Nc+4Nci-2i^2+1} \ . \tag{4.2}$$

Before going forward, we perform a simple change of variables with $s$, where $c = 1 + \frac{s}{2N}$. This will significantly simplify our analysis to follow. So removing the $N$ subscript, we define $\psi$ as a function of $s$:

$$\psi_N(c) = \psi(s) := 1 - \frac{1}{2N}\sum_{i=0}^{N-1}\frac{s(s+1)(2N+s+1)}{2N+4Ni+(2i+1)s-2i^2+1} \ . \tag{A.6}$$

We now focus on approximating $s_{\text{crit}}$, where $s_{\text{crit}}$ is the largest root of $\psi(s)$, or equivalently, $s_{\text{crit}} = 2N(c_{\text{crit}} - 1)$.

Our approach for approximating $s_{\text{crit}}$ will be to find upper and lower bounding functions for $\psi(s)$. We can then find a closed form expression for the roots of these bounding functions, which will in turn act as bounds for $s_{\text{crit}}$. Define

$$\psi_\ell(s) = 1 - \left(\log N + \frac{1}{2N} + \gamma\right)\frac{s(s+1)(2N+1)}{4N^2} \tag{A.7}$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant. We claim below that $\psi_\ell(s)$ is a valid lower bound for $\psi(s)$ and defer the calculations to the following section.

**Lemma A.1.** *For all $s > 0$, $\psi_\ell(s) < \psi(s)$.*

So $\psi_\ell(s)$ is a lower bound for $\psi(s)$, with a much simpler form. Observe that $\psi_\ell(s)$ is quadratic with respect to $s$, it is concave, and $\psi_\ell(0) = 1 > 0$, so it must have a single positive root, which we will call $s_\ell$. And critically, this root $s_\ell$ is a lower bound for $s_{\text{crit}}$. We can easily find $s_\ell$ by the quadratic equation:

$$s_\ell = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{16N^2}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}}\ .$$

And therefore, we can define $c_\ell < c_{\text{crit}}$ by

$$c_\ell = 1 - \frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{1}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}}\ .$$

This bound $c_\ell$ gives us the precise form we present in Proposition 4.1.

We now derive an upper bound $c_u$ through similar approach. Define

$$\psi_u(s) = 1 - \left(\log N + \frac{4N-1}{8N^2} + \gamma\right)\frac{s(s+1)(3N+1)}{12N^2} \tag{A.8}$$

with $\gamma$ again the Euler-Mascheroni constant. We again defer the calculations to the following section.

**Lemma A.2.** *For all $0 < s \leq N$, $\psi_u(s) < \psi(s)$.*

By the same argument as for the lower bound, we know that the largest root $s_u$ of $\psi_u(s)$ is an upper bound for $s_{\text{crit}}$, provided that $s_u \leq N$. Solving our quadratic equation we have

$$s_u = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{48N^2}{(3N+1)(\log N + \frac{4N-1}{8N^2} + \gamma)}}\ .$$

One can easily check $s_u \leq N$, so the needed condition is satisfied. Consequently, we define $c_u > c_{\text{crit}}$ by

$$c_u = 1 - \frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{3}{(3N+1)(\log N + \frac{4N-1}{8N^2} + \gamma)}}\ .$$

To summarize, from Lemmas A.1 and A.2, we know for all $N$ that $c_\ell \leq c_{\text{crit}} \leq c_u$. The proposition's first claim follows by applying Theorem 4.1 to $c_\ell < c_{\text{crit}}$. The second claim similarly can be obtained by verifying that $1 + 1/(4\sqrt{N\log(N)}) < c_\ell < c_{\text{crit}}$

Lastly, we observe these upper and lower bounds suffice to determine the asymptotic behavior of $c_{\text{crit}}$. We claim that $c_\ell = 1 + \Theta\left(\frac{1}{\sqrt{N\log(N)}}\right)$ and similarly $c_u = 1 + \Theta\left(\frac{1}{\sqrt{N\log(N)}}\right)$, from which we can conclude $c_{\text{crit}} = 1 + \Theta\left(\frac{1}{\sqrt{N\log(N)}}\right)$. These claimed asymptotic behaviors of the upper and lower bounds follow from the following simple limit calculations

$$\begin{aligned}
\lim_{N\to\infty} \frac{c_\ell - 1}{\frac{1}{\sqrt{N\log(N)}}} &= \lim_{N\to\infty}\left(-\frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{1}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}}\right)\sqrt{N\log(N)} \\
&= \lim_{N\to\infty} -\frac{\sqrt{N\log(N)}}{4N} + \sqrt{\frac{N\log(N)}{16N^2} + \frac{N\log(N)}{(2N+1)(\log N + \frac{1}{2N} + \gamma)}} = \frac{\sqrt{2}}{2}\ , \\
\lim_{N\to\infty} \frac{c_u - 1}{\frac{1}{\sqrt{N\log(N)}}} &= \lim_{N\to\infty}\left(-\frac{1}{4N} + \sqrt{\frac{1}{16N^2} + \frac{3}{(3N+1)(\log N + \frac{4N-1}{8N^2} + \gamma)}}\right)\sqrt{N\log(N)} \\
&= \lim_{N\to\infty} -\frac{\sqrt{N\log(N)}}{4N} + \sqrt{\frac{N\log(N)}{16N^2} + \frac{3N\log(N)}{(3N+1)(\log N + \frac{4N-1}{8N^2} + \gamma)}} = 1\ .
\end{aligned}$$

## A.4 Proof of Lemma A.1

We prove this lower bound in two steps, using an intermediate function $\hat{\psi}_\ell(s)$. Define

$$\hat{\psi}_\ell(s) = 1 - \frac{1}{2N}\sum_{i=0}^{N-1} \frac{s(s+1)(2N+1)}{2N + 4Ni - 2i^2 + 1}\ .$$

We first show that $\hat{\psi}_\ell(s) \le \psi(s)$. We calculate

$$\begin{aligned}
\psi(s) - \hat{\psi}_\ell(s) &= \left(1 - \frac{1}{2N}\sum_{i=0}^{N-1}\frac{s(s+1)(2N+s+1)}{2N+4Ni+(2i+1)s-2i^2+1}\right) - \left(1 - \frac{1}{2N}\sum_{i=0}^{N-1}\frac{s(s+1)(2N+1)}{2N+4Ni-2i^2+1}\right) \\
&= \frac{s(s+1)}{2N}\left(\sum_{i=0}^{N-1}\frac{2N+1}{2N+4Ni-2i^2+1} - \sum_{i=0}^{N-1}\frac{2N+s+1}{2N+4Ni+(2i+1)s-2i^2+1}\right) \\
&= \frac{s(s+1)}{2N}\left(\sum_{i=0}^{N-1}\frac{2i(i+1)s}{(2N+4Ni+(2i+1)s-2i^2+1)(2N+4Ni-2i^2+1)}\right) \\
&\ge 0\ .
\end{aligned}$$

For the final inequality, we use the fact that both terms of the denominator are always positive for $i = 0, \ldots, N$. Next we show that $\psi_\ell(s) < \hat{\psi}_\ell(s)$:

$$\begin{aligned}
\hat{\psi}_\ell(s) &= 1 - \frac{1}{2N}\sum_{i=0}^{N-1}\frac{s(s+1)(2N+1)}{2N(1+2i-i\frac{i}{N}+\frac{1}{2N})} \\
&> 1 - \frac{1}{2N}\sum_{i=0}^{N-1}\frac{s(s+1)(2N+1)}{2N(1+i)} \\
&= 1 - \frac{s(s+1)(2N+1)}{4N^2}\sum_{i=0}^{N-1}\frac{1}{i+1} \\
&\ge 1 - \frac{s(s+1)(2N+1)}{4N^2}\left(\log N + \frac{1}{2N} + \gamma\right) \\
&= \psi_\ell(s)
\end{aligned}$$

where the second inequality uses the harmonic series upper bound $\sum_{i=1}^{N}\frac{1}{i} \le \log N + \frac{1}{2N} + \gamma$.

### A.5 Proof of Lemma A.2

We again define an intermediate function $\hat{\psi}_u(s)$:

$$\hat{\psi}_u(s) = 1 - \frac{1}{2N}\sum_{i=0}^{N-1} \frac{s(s+1)(3N+1)}{3N+6Ni-2i^2+1} \ .$$

We first show that $\hat{\psi}_u(s) \geq \psi(s)$:

$$\begin{aligned}
\psi(s) - \hat{\psi}_u(s) &= \left(1 - \sum_{i=0}^{N-1} \frac{s(s+1)(2N+s+1)}{2N+4Ni+(2i+1)s-2i^2+1}\right) - \left(1 - \frac{1}{2N}\sum_{i=0}^{N-1} \frac{s(s+1)(3N+1)}{3N+6Ni-2i^2+1}\right) \\
&= \frac{s(s+1)}{2N}\left(\sum_{i=0}^{N-1} \frac{3N+1}{3N+6Ni-2i^2+1} - \frac{2N+s+1}{2N+4Ni+(2i+1)s-2i^2+1}\right) \\
&= \frac{s(s+1)}{2N}\left(\sum_{i=0}^{N-1} \frac{-2i(i+1)(N-s)}{(3N+6Ni-2i^2+1)(2N+4Ni+(2i+1)s-2i^2+1}\right) \\
&\leq 0 \ .
\end{aligned}$$

The additional assumption here that $s \leq N$ is just a formality for our intermediate bound. As we note in the discussion following Lemma A.2, we do not need to consider $s > N$. Finally, we show that $\psi_u > \hat{\psi}_u(s)$.

$$\begin{aligned}
\hat{\psi}_u(s) &= 1 - \frac{1}{2N}\sum_{i=0}^{N-1} \frac{s(s+1)(3N+1)}{3N(1+2i-\frac{2i^2}{3N}+\frac{1}{3N})} \\
&< 1 - \frac{1}{2N}\sum_{i=0}^{N-1} \frac{s(s+1)(3N+1)}{3N(2+2i)} \\
&= 1 - \frac{s(s+1)(3N+1)}{12N^2}\sum_{i=0}^{N-1} \frac{1}{i+1} \\
&\leq 1 - \frac{s(s+1)(3N+1)}{12N^2}\left(\log N + \frac{4N-1}{8N^2} + \gamma\right) \\
&= \psi_u(s)
\end{aligned}$$

where the second inequality uses the harmonic series lower bound $\sum_{i=1}^{N} \frac{1}{i} \geq \log N + \frac{4N-1}{8N^2} + \gamma$.