A Simple Adaptive Proximal Gradient Method for Nonconvex Optimization [*]

Zilong Ye[†], Shiqian Ma[‡], Junfeng Yang[†] and Danqing Zhou[§]

October 8, 2025

Abstract

Consider composite nonconvex optimization problems where the objective function consists of a smooth nonconvex term (with Lipschitz-continuous gradient) and a convex (possibly nonsmooth) term. Existing parameter-free methods for such problems often rely on complex multi-loop structures, require line searches, or depend on restrictive assumptions (e.g., bounded iterates). To address these limitations, we introduce a novel adaptive proximal gradient method (referred to as AdaPGNC) that features a simple single-loop structure, eliminates the need for line searches, and only requires the gradient's Lipschitz continuity to ensure convergence. Furthermore, AdaPGNC achieves the theoretically optimal iteration/gradient evaluation complexity of $\mathcal{O}(\varepsilon^{-2})$ for finding an ε-stationary point. Our core innovation lies in designing an adaptive step size strategy that leverages upper and lower curvature estimates. A key technical contribution is the development of a novel Lyapunov function that effectively balances the function value gap and the norm-squared of consecutive iterate differences, serving as a central component in our convergence analysis. Preliminary experimental results indicate that AdaPGNC demonstrates competitive performance on several benchmark nonconvex (and convex) problems against state-of-the-art parameter-free methods.

Keywords: adaptive proximal gradient, parameter-free, line-search-free, Lipschitz gradient, composite optimization

1 Introduction

We consider the composite optimization problem:

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + h(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and $h : \mathbb{R}^n \to \mathbb{R}$ is a proper, closed and convex function (possibly nonsmooth). Throughout the paper, we assume that the set of solutions of problem (1) is nonempty, i.e., there exists $x_* \in \mathbb{R}^n$ such that $F(x_*) = F_* \triangleq \inf_{x\in\mathbb{R}^n} F(x) > -\infty$. A classical method for solving this problem is the proximal gradient method, which follows the following iteration formula:

$$x_{k+1} = \mathrm{prox}_{\lambda_k h}(x_k - \lambda_k \nabla f(x_k)), \quad k = 0, 1, 2, \ldots. \tag{2}$$

[*]**Funding:** This work was supported by the National Science Foundation Division of Mathematical Sciences [Grant 2243650]; the Division of Computing and Communication Foundations [Grants 2308597 and 2311275]; the Division of Electrical, Communications and Cyber Systems [Grant 2326591]; the National Natural Science Foundation of China [Grants 12431011 and 12371301]; the Key Laboratory of NSLSCS, Ministry of Education of China.

[†]School of Mathematics, Nanjing University, Nanjing, Jiangsu 210093, People's Republic of China (Email: `zilongye@smail.nju.edu.cn, jfyang@nju.edu.cn`).

[‡]Department of Computational Applied Math and Operations Research, Rice University, Houston, Texas 77005 (Email: `sqma@rice.edu`)

[§]School of Mathematics, Nanjing Audit University, Nanjing, Jiangsu 211815, People's Republic of China (Email: `dqkyo@foxmail.com`)

Here, $x_0 \in \mathbb{R}^n$ is an arbitrary starting point, and $\lambda_k > 0$ represents the step size at the k-th iteration. The proximal operator of h is defined as

$$\operatorname{prox}_{th}(x) := \arg\min_{y\in\mathbb{R}^n} \left\{h(y) + \frac{1}{2t}\|y-x\|^2\right\}, \quad \forall x \in \mathbb{R}^n,\ t > 0.$$

Note that (2) reduces to the gradient descent (GD) if $h \equiv 0$. The key question to ensure convergence lies in how to determine the step size λ_k at each iteration. When f is convex and globally L-smooth, its gradient operator ∇f satisfies

$$f(x) + \langle \nabla f(x), y - x\rangle \le f(y), \quad \forall x, y \in \mathbb{R}^n, \tag{3}$$

and there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n. \tag{4}$$

Under such conditions, any constant step size $\lambda_k = \lambda \in (0, 2/L)$ suffices to ensure the pointwise convergence of the sequence $\{x_k\}$ generated by (2) to a minimizer of f; see, e.g., [2, Thm. 10.24]. Moreover, in this setting, the function value gap $F(x_k) - F_*$ converges at a sublinear rate of $\mathcal{O}(1/k)$; see, e.g., [2, Thm. 10.21]. Nevertheless, determining the Lipschitz constant is challenging in practical settings. Consequently, λ_k needs to be tuned repeatedly, or we can determine λ_k via line-search, such as backtracking or Wolfe-Powell line-search. However, line-search entails additional computational overhead: for each trial point, the evaluation of the function value and, in some cases, the gradient is required. This can be impractical in certain applications. For instance, in the context of large-scale problems in machine learning, computing a single function value typically necessitates a full pass over the dataset, an operation that is often prohibitive. This motivates the adoption of simpler step size selection strategies, such as diminishing step sizes or adaptive step size determination that adjusts dynamically throughout the iteration process.

Recently, extensive research has focused on adaptive step size selection throughout the iteration process. The key motivations for this line of work are as follows: First, computing the global Lipschitz constant L can be computationally expensive. For instance, in logistic regression, L is proportional to the maximum eigenvalue of $X^\top X$ (where X denotes the data matrix). Calculating this maximum eigenvalue is challenging in high-dimensional settings, and becomes even more cumbersome in federated or distributed frameworks, where data are stored across distinct locations. Second, gradient descent with fixed step sizes derived from the worst-case L tends to be overly conservative in practice. This leads to inefficient updates in "flat" regions of the objective function, where significantly larger step sizes would be permissible. Finally, the objective function f may only be locally L-smooth, that is, the inequality in (4) holds solely for x, y within a bounded region. A typical example of this scenario is convex optimization problems with cubic regularization. A seminal contribution to this line of research is due to Malitsky and Mishchenko [18], who proposed an adaptive gradient descent (AdGD) method for (1) with $h \equiv 0$, utilizing the following step size rule:

$$\lambda_k = \min\left\{\sqrt{1+\theta_{k-1}}\lambda_{k-1}, \frac{1}{2L_k}\right\} \quad \text{with} \quad L_k = \frac{\|\nabla f(x_k) - \nabla f(x_{k-1})\|}{\|x_k - x_{k-1}\|}, \quad \theta_{k-1} = \frac{\lambda_{k-1}}{\lambda_{k-2}}. \tag{5}$$

Here, L_k estimates the local curvature of f, while the first term in the minimum defining λ_k regulates the growth rate of the step size. This step size strategy guarantees the monotonic decrease of a specific Lyapunov function, which in turn ensures the boundedness of the iterates and the convergence of AdGD under convexity and the weaker local L-smoothness condition. Subsequently, AdGD was refined in [19] and extended to composite convex optimization problems in [13, 14]. More recently, Zhou et al. [26] proposed an adaptive gradient method based on the short Barzilai-Borwein step size [1] for (composite) convex optimization problems, which achieves the same convergence results as those in the globally L-smooth setting. We also mention the recent work [16] and [22], which use an adaptive estimation for the Lipschitz constant and achieve the accelerated convergence rate in function value gap: $F(x_k) - F_* = \mathcal{O}(1/k^2)$.

In this paper, we further investigate the nonconvex yet globally L-smooth setting, where the Lipschitz smoothness condition (4) holds for some constant $L > 0$ and all $x, y \in \mathbb{R}^n$. Under the L-smoothness condition (4), there exists a constant $l \in [0, L]$ such that

$$-\frac{l}{2}\|x-y\|^2 \le f(x) - f(y) - \langle \nabla f(y), x - y\rangle \le \frac{L}{2}\|x-y\|^2, \quad \forall x, y \in \mathbb{R}^n. \tag{6}$$

When $l = 0$, the function f is convex; otherwise, f is weakly convex, following the terminology in, e.g., [5, 6]. In the following, we refer to L and l as the upper curvature and lower curvature, respectively. We propose an algorithm termed AdaPGNC, which requires no prior knowledge of L and l, a property that renders the algorithm parameter-free. We shall estimate L and l using the current and previous iterates, and the resulting estimates are then used to construct the step sizes. Furthermore, a key technique underpinning our algorithm is the introduction of a summable nonnegative sequence to control the growth rate of step sizes, a strategy that has also been adopted in recent work [9, 8, 17].

In the rest of this section, we first review several existing parameter-free methods for nonconvex optimization. All these methods assume that f is globally L-smooth and satisfies the curvature condition (6) for some $l \in [0, L]$. Recall that an algorithm is considered parameter-free if it does not require prior knowledge of the upper and lower curvature parameters L and l. We then summarize the contributions of this paper and its organization.

1.1 Existing parameter-free methods for nonconvex optimization

Recently, Lan et al. proposed a parameter-free method named NASCAR (*Nonconvex Acceleration via Strongly Convex Accumulative Regularization*, [12, Alg. 7.2]) for solving the nonconvex optimization problem (1) with $h \equiv 0$. It requires solving a sequence of strongly convex regularization subproblems, involves four nested loops, and relies on line-search. Furthermore, it needs $\mathcal{O}(1/\varepsilon^2)$ gradient evaluations to reach an ε-stationary point x satisfying $\|\nabla f(x)\| \leq \varepsilon$. While this result is optimal for $l = L$ [3] and represents the state of the art for $l \in (0, L)$, the four-nested-loop structure of the method undermines its practical effectiveness. Another limitation of NASCAR is that it requires the precision parameter ε as an input to the algorithm, since ε is necessary for solving its sequence of strongly convex regularization subproblems. Marumo and Takeda proposed a parameter-free method named AGDR (*Accelerated Gradient Descent with Restart*, [20, Alg. 4.1]) under the same setting as Lan et al. [12, Alg. 7.2], but with an extra second-order smoothness condition. AGDR features a simple single-loop structure but relies on a restart technique, and it requires $\mathcal{O}(1/\varepsilon^{7/4})$ gradient evaluations to reach an ε-stationary point. Notably, this improved convergence result is attributed to the aforementioned additional second-order smoothness assumption. However, these two algorithms cannot solve the general composite problem (1). Kong proposed a parameter-free method named PF.APD (*Parameter-Free with Accelerated Proximal Descent*, [10, Alg. 3.4]) to solve the nonconvex composite optimization problem (1). The algorithmic structure and theoretical results of PF.APD are completely analogous to those of NASCAR [12, Alg. 7.2]: it iteratively solves a sequence of strongly convex regularization subproblems, involves four nested loops, relies on line-search, and requires $\mathcal{O}(1/\varepsilon^2)$ gradient evaluations to reach an ε-stationary point x satisfying $\|\nabla f(x) + h'(x)\| \leq \varepsilon$ for some subgradient $h'(x) \in \partial h(x)$. For problem (1) where h being the indicator function of a compact convex set Ω (i.e., a nonconvex optimization problem with a compact convex constraint), Lan et al. proposed a parameter-free method called AC-PG (*Auto-Conditioned Projected Gradient*, [11, Alg. 2]). AC-PG method adopts a single loop structure yet employs a monotonically nonincreasing step size strategy based on the local curvature estimation:

$$\mathcal{L}_k = 2\big(f(x_k) - f(x_{k-1}) - \langle \nabla f(x_{k-1}), x_k - x_{k-1}\rangle\big)/\|x_k - x_{k-1}\|^2.$$

Specifically, AC-PG determines the step size using historical information of $\mathcal{L}_k$:

$$\lambda_{k+1} = 1/\max\{\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_k\} > 0 \ \forall k \geq 0 \text{ and } \lambda_0 = 1/\mathcal{L}_0 \geq 1/L.$$

To ensure the projected gradient has a norm no greater than ε, the AC-PG method requires $\mathcal{O}(1/\varepsilon^2)$ iterations. This complexity result heavily depends on the boundedness of Ω; in particular, the squared diameter of Ω is concealed within the big $\mathcal{O}$ notation. Very recently, Hoai and Thai proposed a NPG2 method (*nonconvex proximal gradient*, [9, Alg. 4.2]) for solving the composite nonconvex problem (1), where f is assumed to satisfy that for any u, v, the function $\langle \nabla f(u+t(v-u)), v-u\rangle$ is quasiconvex with respect to $t \in [0, 1]$. This method estimates the upper curvature L as in (5) and adopts a summable nonnegative sequence technique to control the growth rate of step sizes. Furthermore, NPG2 features a simple single-loop structure and achieves an $\mathcal{O}(1/\varepsilon^2)$ gradient evaluation complexity, ensuring the proximal gradient has a norm not greater than ε (see [9, Thm. 4.1] for details).

While the aforementioned algorithms achieve parameter-free adaptation, each is plagued by notable limitations. Specifically, both NASCAR and PF.APD feature four nested loops and necessitate line-search for estimating the local upper curvature, which leads to prohibitively complex implementations and renders their efficiency far from practical. Although AC-PG simplifies the algorithmic structure, it is restricted to problems with compact convex constraints. On the other hand, NPG2 relaxes this boundedness requirement but imposes a quasiconvexity assumption on $\langle \nabla f(u + t(v-u)), v-u\rangle$ with respect to $t \in [0,1]$, a condition that may prove challenging to verify in practice. Regarding AGDR, while it can solve (1), it requires second-order smoothness for conducting its convergence analysis, greatly limiting its applicability.

1.2 Summary of contributions

To the best of our knowledge, no existing method simultaneously satisfies all the following features: (a) having a simple single-loop structure; (b) admitting convergence analysis under first-order Lipschitz continuity alone (with no additional assumptions); (c) requiring no line-search or restart; (d) being applicable to the general composite nonconvex problem (1). To bridge this gap, we propose AdaPGNC (*Adaptive Proximal Gradient for NonConvex Optimization*) in this paper, which achieves all the aforementioned features simultaneously. Specifically, AdaPGNC is parameter-free and line-search-free, adopts a simple single-loop structure, and requires no restart. Additionally, it solves the generic composite nonconvex problem (1). Furthermore, under the blanket global L-smooth condition (4), we show that AdaPGNC finds an ε-stationary point within $\mathcal{O}(1/\varepsilon^2)$ iterations, which equals the number of gradient evaluations, as the algorithm only requires one gradient computation per iteration. This complexity result matches the state-of-the-art for $l \in (0, L)$ and is optimal when $l = L$. At each iteration, AdaPGNC adaptively determines the step size by incorporating local curvature information and restricts step size growth via a nonnegative summable sequence. It naturally extends Malitsky and Mishchenko's AdGD algorithm (see [18]) to general nonconvex problems with a convex regularizer. For a clear comparison, the features of the reviewed parameter-free methods and our AdaPGNC are summarized in Table 1.

Table 1: Comparison of features, restrictions, and/or additional assumptions (excluding the blanket global L-smooth condition) for the reviewed parameter-free methods and our AdaPGNC. Here, "LS" denotes line-search.

Algorithm	Restrictions &/or Additional Assump.	Without LS	Without Restart	#Loops
NASCAR [12, Alg. 7.2]	$h \equiv 0$	✗	✓	4
AGDR [20, Alg. 4.1]	$h \equiv 0$ 2nd-order smoothness	✓	✗	1
PF.APD [10, Alg. 3.4]	none	✗	✓	4
AC-PG [11, Alg. 2]	h is an indicator func. of a compact set Ω	✓	✓	1
NPG2 [9, Alg. 4.2]	$\langle \nabla f(u + t(v-u)), v-u\rangle$ quasi-conv in t	✓	✓	1
AdaPGNC (Ours)	**none**	✓	✓	**1**

1.3 Notation and organization

Our notation is quite standard. Let $\mathbb{R}^n$ be the n-dimensional Euclidean space. For $u, v \in \mathbb{R}^n$, $\langle u, v\rangle$ denotes their inner product, and $\|u\|$ denotes the Euclidean norm. For a matrix U, we let $\|U\|_F$ be the Frobenius norm of U. For a proper closed convex function $h : \mathbb{R}^n \to (-\infty, \infty]$, its subdifferential at a point $x \in \mathbb{R}^n$ is defined as $\partial h(x) = \{v \in \mathbb{R}^n \mid h(y) \geq h(x) + \langle v, y-x\rangle \text{ for all } y \in \mathbb{R}^n\}$. We adopt the convention that $c/0 = +\infty$ for any $c > 0$, and use the notation $[u]_+ = \max\{u, 0\}$.

The rest of this paper is organized as follows. In Section 2, we first present the AdaPGNC algorithm for the composite problem (1). Next, Section 3 is dedicated to establishing the convergence results of AdaPGNC, where we rigorously derive its theoretical guarantees. Section 4 then reports preliminary numerical results: we compare AdaPGNC with several state-of-the-art parameter-free methods on a set of benchmark nonconvex problems and the logistic regression problem to validate its practical performance. Finally, concluding remarks and potential future directions are drawn in Section 5.

2 The proposed algorithm: AdaPGNC

In this section, we present our AdaPGNC algorithm for the composite nonconvex optimization problem (1). We first formalize the blanket assumptions underlying our analysis.

Assumption 1. *We assume that (i) $h : \mathbb{R}^n \to (-\infty, \infty]$ is proper, closed and convex, (ii) $f : \mathbb{R}^n \to \mathbb{R}$ is globally L-smooth, i.e., it satisfies* (4) *for some $L > 0$ and all $x, y \in \mathbb{R}^n$, and (iii) F is lower bounded, i.e., $F_* \triangleq \inf_{x\in\mathbb{R}^n} F(x) > -\infty$.*

Recall that, under Assumption 1 (*ii*), there exists some $l \in [0, L]$ such that the two inequalities in (6) hold for all $x, y \in \mathbb{R}^n$. For the smooth function f in (1), we adopt the vanilla proximal gradient to handle its optimization. We first estimate the upper and lower curvature of f appropriately using local information at each iteration; these estimates are then used to construct the step size. Furthermore, inspired by [9], we introduce a nonnegative summable sequence to regulate the step size's growth rate, which plays a critical role in the convergence analysis. Specifically, the upper and lower curvatures of f at the k-th iteration are estimated respectively by

$$L_k = \frac{\|\nabla f(x_k) - \nabla f(x_{k-1})\|}{\|x_k - x_{k-1}\|} \quad \text{and} \quad l_k = \frac{2\big(f(x_k) - f(x_{k-1}) + \langle \nabla f(x_k), x_{k-1} - x_k\rangle\big)}{\|x_k - x_{k-1}\|^2}. \tag{7}$$

It follows from (4) and (6) that $L_k \leq L$ and $l_k \leq l \leq L$. On the other hand, for the potentially nonsmooth function h in (1), which is proper, closed, and convex according to Assumption 1 (*i*), we resort to the proximal operator for its optimization. The proposed AdaPGNC for solving problem (1) is summarized in Algorithm 1.

Algorithm 1 Adaptive **P**roximal **G**radient Method for **N**on**C**onvex (AdaPGNC) Problem (1)

```
1: Input: $x_0 \in \mathbb{R}^n$, $\lambda_0 > 0$, and a nonnegative sequence $\{\rho_k\}_{k=0}^{\infty}$ such that $\sum_{k=0}^{\infty} \rho_k < +\infty$.
2: $x_1 = \mathrm{prox}_{\lambda_0 h}(x_0 - \lambda_0 \nabla f(x_0))$
3: for $k = 1, 2, \ldots$ do
4:     Compute $L_k$ and $l_k$ according to (7)
5:     if $l_k \leq 0$ then
6:         $\lambda_k = \min\left\{\sqrt{1+\rho_{k-1}}\lambda_{k-1}, \frac{1}{L_k}\right\}$
7:     else
8:         $\lambda_k = \min\left\{\sqrt{1+\rho_{k-1}}\lambda_{k-1}, \frac{1}{\sqrt{2}L_k}, \sqrt{\frac{\lambda_{k-1}}{2l_k}}\right\}$
9:     end if
10:    $x_{k+1} = \mathrm{prox}_{\lambda_k h}(x_k - \lambda_k \nabla f(x_k))$
11: end for
```

Note that from (7), the validity of $l_k \leq 0$ is equivalent to the satisfaction of the subgradient inequality (3), e.g., $f(x_{k-1}) \geq f(x_k) + \langle \nabla f(x_k), x_{k-1} - x_k\rangle$, which indicates the detection of local convexity. In this case, we set $\lambda_k = \min\{\sqrt{1+\rho_{k-1}}\lambda_{k-1}, 1/L_k\}$: the first term in the minimum regulates the step size's growth rate, while the second term approximates the inverse of the upper curvature. When $l_k \leq 0$ is violated (indicating nonconvexity is detected), we adopt a more conservative step size, as specified in Line 8 of Algorithm 1. Note that the second and third terms in the minimum in Line 8 of Algorithm 1 are derived from our analysis, and their setting allows some flexibility, which will be elaborated on in Remark 2.

Comparison with AdGD. Our step size design for AdaPGNC builds upon the AdGD framework [18] but introduces several key modifications. First, we replace the parameter θ_k in (5) with ρ_k to better regulate the step size growth. This change is crucial because we constrain ρ_k to be a summable series, which prevents the potential divergence of λ_k that could occur if θ_k were non-summable. Second, we employ a more aggressive step size of $1/L_k$ instead of $1/(2L_k)$ used in (5) for convex problems. Third, to handle nonconvexity, we refine the step size by reducing the coefficient from 2 to $\sqrt{2}$ and incorporating an extra term $\sqrt{\lambda_{k-1}/(2l_k)}$. The coefficient $1/(\sqrt{2}L_k)$ aligns with the refined AdGD method in [19], while the additional term is our novel contribution for nonconvex settings. The second and third terms in the minimum operation in Line 8 of Algorithm 1 are derived from our convergence analysis, and we note that their specific form admits flexibility, as will be elaborated in Remark 2.

3 Convergence analysis

In this section, we carry out the convergence analysis of Algorithm 1. First, we present some preliminaries for this analysis, including establishing the boundedness of the step sizes $\{\lambda_k\}$, recalling useful lemmas from the literature, and introducing a Lyapunov function (a key component for the complexity analysis). Then, we derive the convergence results for the general nonconvex case, followed by additional results when the function f is convex. For brevity, in what follows, we will not repeatedly mention the standing Assumption 1 and will take its validity for granted.

3.1 Preliminaries

Proposition 1 (boundedness of $\{\lambda_k\}$)**.** *Let $\{\lambda_k\}$ be the sequence of step sizes generated by Algorithm 1, and define $P := \sum_{k=0}^{\infty} \rho_k$. Then, $\{\lambda_k\}$ is bounded below and above by positive constants $\lambda > 0$ and $\Lambda > 0$, respectively. Specifically, we have*

$$\lambda := \min\big(\lambda_0, 1/(2L)\big) \le \lambda_k \le \Lambda := \lambda_0 \exp(P/2), \quad \forall k \ge 0. \tag{8}$$

Proof. We first establish the upper bound. From Algorithm 1, we have $\lambda_{i+1}/\lambda_i \le \sqrt{1+\rho_i}$ for all $i \ge 0$, and thus $\ln \lambda_{i+1} - \ln \lambda_i = \ln(\lambda_{i+1}/\lambda_i) \le \ln \sqrt{1+\rho_i} \le \rho_i/2$. Summing this inequality over $i = 0, \ldots, k$ and recalling that $P = \sum_{i=0}^{\infty} \rho_i$, we obtain $\ln \lambda_{k+1} - \ln \lambda_0 \le \sum_{i=0}^{k} \rho_i/2 \le P/2$. This further implies $\lambda_k \le \lambda_0 \exp(P/2)$ for all $k \ge 0$. Next, we establish the lower bound by induction. The base case: $\lambda_0 \ge \lambda$ holds trivially by the definition of λ. For the inductive step, suppose that $\lambda_k \ge \lambda$ holds for some integer $k \ge 0$; we then show this inequality must also hold for $k+1$. We divide the proof into three cases: (i) If $\lambda_{k+1} = \sqrt{1+\rho_k}\lambda_k$, then $\lambda_{k+1} \ge \lambda_k \ge \lambda$ (the second inequality follows from the inductive hypothesis). (ii) If $\lambda_{k+1} = 1/(\sqrt{2}L_{k+1})$ or $\lambda_{k+1} = 1/L_{k+1}$, then recalling that $L_k \le L$ for all $k \ge 0$, we have $\lambda_{k+1} \ge 1/(\sqrt{2}L) \ge \lambda$. (iii) If $\lambda_{k+1} = \sqrt{\lambda_k/(2l_{k+1})}$ (which possibly applies only when $l_{k+1} > 0$), then by the inductive hypothesis, together with $l_{k+1} \le l \le L$ and the definition of λ, we obtain $\lambda_{k+1} \ge \sqrt{\lambda/(2L)} \ge \lambda$. Therefore, $\lambda_k \ge \lambda$ holds for all $k \ge 0$. $\square$

An equivalent form of the proximal gradient step $x_{k+1} = \mathrm{prox}_{\lambda_k h}(x_k - \lambda_k \nabla f(x_k))$ is given by

$$x_{k+1} = x_k - \lambda_k(\nabla f(x_k) + h'(x_{k+1})) \quad \text{for some} \quad h'(x_{k+1}) \in \partial h(x_{k+1}). \tag{9}$$

In this paper, we employ the following gradient mapping to quantify the accuracy of an approximate solution:

$$\mathcal{G}_\eta(x) := \big(x - \mathrm{prox}_{\eta h}(x - \eta \nabla f(x))\big)/\eta, \quad \text{for some} \quad \eta > 0. \tag{10}$$

This measure is commonly used in the context of composite optimization problems, see, e.g., [21, 9]. When $h \equiv 0$, we have $\mathcal{G}_\eta(x) = \nabla f(x)$ for any $\eta > 0$. Moreover, it follows directly from (9) and the definition of $\mathcal{G}_\eta(x)$ in (10) that $\|\mathcal{G}_{\lambda_k}(x_k)\| = \|(x_{k+1} - x_k)/\lambda_k\| = \|\nabla f(x_k) + h'(x_{k+1})\|$. In this paper, we call a point $\hat{x} \in \mathbb{R}^n$ as an ε-stationary point if $\|\mathcal{G}_\Lambda(\hat{x})\| \le \varepsilon$ for some $\varepsilon > 0$, where Λ is defined in (8).

Next, we recall two useful lemmas from the literature, both of which hold under conditions (i) and (ii) of our Assumption 1.

Lemma 1 (see [19, Lem. 12])**.** *For iterates $\{x_k\}$ of the proximal gradient method with arbitrary step sizes $\{\lambda_k\}$, it holds*

$$\|\nabla f(x_k) + h'(x_{k+1})\| \le \|\nabla f(x_k) + h'(x_k)\|. \tag{11}$$

Lemma 2 (see [2, Thm. 10.9])**.** *For any $\eta_1 \ge \eta_2 > 0$, we have $\|\mathcal{G}_{\eta_1}(x)\| \le \|\mathcal{G}_{\eta_2}(x)\|$ for all $x \in \mathbb{R}^n$.*

For simplicity, in the rest of this section, we denote $\mathcal{G}_k := \mathcal{G}_{\lambda_k}(x_k)$. In adaptive algorithms, the sequence of objective function values $\{F(x_k)\}$ is not necessarily monotone. To analyze the convergence rate of AdaPGNC, we construct a Lyapunov function that incorporates both the objective function value gap and the iterate movement, defined as follows:

$$V_k := \omega_k\Big(F(x_k) - F_* + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2\Big) \ge 0, \quad \forall k \ge 0, \tag{12}$$

where (for convenience, we define $\lambda_{-1} = \lambda_0$ and $\rho_{-1} = 0$)

$$\omega_k := \frac{\lambda_k^2}{\lambda_0^2 \prod_{i=1}^k (1+\rho_{i-1}) \prod_{i=1}^k \sqrt{1+\rho_{i-2}}}, \quad \forall k \geq 1 \text{ and } \omega_0 = 1. \tag{13}$$

Next, we prove some basic properties of $\{\omega_k\}$.

Lemma 3 (properties of $\{\omega_k\}$)**.** *Let $\{\omega_k\}$ be defined as in (13). Then, we have*

$$\omega_{k+1}\lambda_k^2(1+\rho_k)\sqrt{1+\rho_{k-1}} = \omega_k \lambda_{k+1}^2, \quad \forall k \geq 0. \tag{14}$$

Moreover, $\{\omega_k\}$ is bounded below: $\omega_k \geq \omega := (\lambda^2/\lambda_0^2)\exp(-3P/2) > 0$ for all $k \geq 1$.

Proof. Fix $k \geq 0$ arbitrarily. Equality (14) follows directly from the definition of ω_k in (13). Furthermore, by leveraging the inequality $\ln(1+x) \leq x$ (which holds for all $x \geq 0$) and the definition of P, we obtain

$$\prod_{i=1}^k (1+\rho_{i-1}) = \exp\Big(\sum_{i=1}^k \ln(1+\rho_{i-1})\Big) \leq \exp\Big(\sum_{i=1}^k \rho_{i-1}\Big) \leq \exp(P).$$

Similarly, we have $\prod_{i=1}^k \sqrt{1+\rho_{i-2}} \leq \exp(P/2)$. Further, by noting that $\lambda_k \geq \lambda$ for all k, we deduce from (13) that $\omega_k \geq (\lambda^2/\lambda_0^2)\exp(-3P/2)$. This completes the proof. □

3.2 Convergence analysis for the nonconvex setting

In this subsection, we establish convergence results for the general nonconvex setting. To begin with, we present a technical lemma, whose proof follows from straightforward calculations.

Lemma 4. *Let $\{x_k\}$ and $\{\lambda_k\}$ denote the sequences generated by Algorithm 1. Then, for all $k \geq 1$, it holds that*

$$\|x_{k+1} - x_k\|^2 \leq \Big(\lambda_k^2 L_k^2 + \frac{\lambda_k^2}{\lambda_{k-1}} l_k\Big)\|x_k - x_{k-1}\|^2 + \frac{2\lambda_k^2}{\lambda_{k-1}}\big(F(x_{k-1}) - F(x_k)\big) - \lambda_k^2 \|\mathcal{G}_{k-1}\|^2. \tag{15}$$

Proof. It follows from (7) and (9) that

$$\begin{aligned} f(x_{k-1}) &\overset{(7)}{=} f(x_k) + \langle \nabla f(x_k), x_{k-1} - x_k\rangle - \frac{l_k}{2}\|x_k - x_{k-1}\|^2 \\ &\overset{(9)}{=} f(x_k) + \lambda_{k-1}\langle \nabla f(x_k), \nabla f(x_{k-1}) + h'(x_k)\rangle - \frac{l_k}{2}\|x_k - x_{k-1}\|^2. \end{aligned} \tag{16}$$

Since $h'(x_k) \in \partial h(x_k)$, the subgradient inequality for the convex function h implies that

$$h(x_{k-1}) \geq h(x_k) + \langle h'(x_k), x_{k-1} - x_k\rangle \overset{(9)}{=} h(x_k) + \lambda_{k-1}\langle h'(x_k), \nabla f(x_{k-1}) + h'(x_k)\rangle. \tag{17}$$

Combining (16) and (17), we obtain

$$F(x_{k-1}) \geq F(x_k) + \lambda_{k-1}\langle \nabla f(x_k) + h'(x_k), \nabla f(x_{k-1}) + h'(x_k)\rangle - \frac{l_k}{2}\|x_k - x_{k-1}\|^2. \tag{18}$$

Then, we can upper bound $\|x_{k+1} - x_k\|^2$ as:

$$\begin{aligned} &\|x^{k+1} - x^k\|^2 \overset{(9)}{=} \lambda_k^2\|\nabla f(x_k) + h'(x_{k+1})\|^2 \overset{(11)}{\leq} \lambda_k^2 \|\nabla f(x_k) + h'(x_k)\|^2 \\ &= \lambda_k^2\|\nabla f(x_k) - \nabla f(x_{k-1})\|^2 - \lambda_k^2\|\mathcal{G}_{k-1}\|^2 + 2\lambda_k^2\langle \nabla f(x_k) + h'(x_k), \nabla f(x_{k-1}) + h'(x_k)\rangle \\ &\overset{(7)}{=} \lambda_k^2 L_k^2 \|x_k - x_{k-1}\|^2 - \lambda_k^2\|\mathcal{G}_{k-1}\|^2 + 2\lambda_k^2\langle \nabla f(x_k) + h'(x_k), \nabla f(x_{k-1}) + h'(x_k)\rangle \\ &\overset{(18)}{\leq} \lambda_k^2 L_k^2 \|x_k - x_{k-1}\|^2 - \lambda_k^2\|\mathcal{G}_{k-1}\|^2 + \frac{2\lambda_k^2}{\lambda_{k-1}}\Big(F(x_{k-1}) - F(x_k) + \frac{l_k}{2}\|x_k - x_{k-1}\|^2\Big) \\ &= \text{right-hand side of (15)}, \end{aligned} \tag{19}$$

where the second equality follows from $\mathcal{G}_{k-1} = \mathcal{G}_{\lambda_{k-1}}(x_{k-1}) = \nabla f(x_{k-1}) + h'(x_k)$. This completes the proof. □

Now, we are ready to present the main convergence results of AdaPGNC in the general nonconvex setting.

Theorem 1. *Let $\{x_k\}$ and $\{\lambda_k\}$ denote the sequences generated by Algorithm 1, V_k be defined as in (12), and ω_k be defined as in (13). Recall also that λ and Λ are defined in (8). Then, for all $k \geq 1$, it holds that*

$$V_k \leq V_{k-1} - \frac{\omega_k \lambda_{k-1}}{2} \|\mathcal{G}_{k-1}\|^2. \tag{20}$$

This further implies $\lim_{k\to\infty} \|\mathcal{G}_\Lambda(x_k)\| = 0$ and $\min_{0\leq i\leq k-1} \|\mathcal{G}_\Lambda(x_i)\|^2 \leq \min_{0\leq i\leq k-1} \|\mathcal{G}_i\|^2 \leq \frac{2V_0}{\omega\lambda k}$. Hence, for any $\varepsilon > 0$, to generate a point x satisfying $\|\mathcal{G}_\Lambda(x)\| \leq \varepsilon$, the number of iterations (or, equivalently, the number of gradient evaluations) is at most $\frac{2V_0}{\omega\lambda\varepsilon^2}$, which is of order $\mathcal{O}(1/\varepsilon^2)$.

Proof. From the definition of λ_k in Algorithm 1, we can deduce straightforwardly that

$$\lambda_k^2 L_k^2 + \frac{\lambda_k^2}{\lambda_{k-1}} l_k \leq 1. \tag{21}$$

By using (21) to amplify the first term on the right-hand side of (15) and rearranging the resulting terms, we deduce

$$F(x_k) + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2 \leq F(x_{k-1}) + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_k - x_{k-1}\|^2 - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2. \tag{22}$$

Recall that $\lambda_k \leq \sqrt{1+\rho_{k-1}}\lambda_{k-1}$ holds for all $k \geq 1$. Let $k \geq 1$ be arbitrarily fixed. We next split the discussion into two mutually exclusive cases: (a) $\lambda_{k-1}/\lambda_k^2 \leq \lambda_{k-2}/\lambda_{k-1}^2$, and (b) $\lambda_{k-1}/\lambda_k^2 > \lambda_{k-2}/\lambda_{k-1}^2$. For case (a), we have $\lambda_{k-1}/\lambda_k^2 \leq \lambda_{k-2}/\lambda_{k-1}^2$, which together with (22), implies that

$$\begin{aligned}
&F(x_k) - F_* + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2 \\
\overset{(22),(a)}{\leq}\; & F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2 - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2 \\
\leq\; & \Big(F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2\Big)\frac{\lambda_{k-1}^2}{\lambda_k^2}(1+\rho_{k-1})\sqrt{1+\rho_{k-2}} - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2.
\end{aligned}$$

For case (b), we have $\lambda_{k-1}^3/(\lambda_k^2\lambda_{k-2}) > 1$, which together with (22), implies that

$$\begin{aligned}
&F(x_k) - F_* + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2 \\
\overset{(22),(b)}{\leq}\; & \Big(F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2\Big)\frac{\lambda_{k-1}^3}{\lambda_k^2\lambda_{k-2}} - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2 \\
\leq\; & \Big(F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2\Big)\frac{\lambda_{k-1}^2}{\lambda_k^2}(1+\rho_{k-1})\sqrt{1+\rho_{k-2}} - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2.
\end{aligned}$$

In both cases, we arrived at the same inequality

$$\begin{aligned}
&F(x_k) - F_* + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2 \\
\leq&\Big(F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2\Big)\frac{\lambda_{k-1}^2}{\lambda_k^2}(1+\rho_{k-1})\sqrt{1+\rho_{k-2}} - \frac{\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2.
\end{aligned} \tag{23}$$

Multiplying both sides of (23) by ω_k (as defined in (13)), and utilizing the definition of V_k in (12) together with the equality in (14), we derive (20) immediately as follows:

$$\begin{aligned}
V_k \overset{(12)}{=}\; & \omega_k\Big(F(x_k) - F_* + \frac{\lambda_{k-1}}{2\lambda_k^2}\|x_{k+1} - x_k\|^2\Big) \\
\overset{(23),(14)}{\leq}\; & \omega_{k-1}\Big(F(x_{k-1}) - F_* + \frac{\lambda_{k-2}}{2\lambda_{k-1}^2}\|x_k - x_{k-1}\|^2\Big) - \frac{\omega_k\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2 \\
\overset{(12)}{=}\; & V_{k-1} - \frac{\omega_k\lambda_{k-1}}{2}\|\mathcal{G}_{k-1}\|^2.
\end{aligned}$$

By replacing the index k with i in the above inequality and then summing the inequality over $i = 1, \dots, k$, we obtain

$$\sum_{i=1}^{k} \frac{\omega_i \lambda_{i-1}}{2} \|\mathcal{G}_{i-1}\|^2 \le V_0 - V_k \le V_0. \tag{24}$$

Since $\omega_i \ge \omega > 0$ (see Lemma 3) and $\lambda_i \ge \lambda > 0$ (see Eq. (8)), it follows easily from (24) that $\lim_{k\to\infty} \|\mathcal{G}_k\| = 0$ and $\min_{0\le i\le k-1} \|\mathcal{G}_i\|^2 \le \frac{2V_0}{\omega\lambda k}$. Furthermore, since $\|\mathcal{G}_\Lambda(x_i)\| \le \|\mathcal{G}_{\lambda_i}(x_i)\| = \|\mathcal{G}_i\|$ for all i, which is implied by Lemma 2 together with the fact that $\lambda_i \le \Lambda$, we immediately obtain $\lim_{k\to\infty} \|\mathcal{G}_\Lambda(x_k)\| = 0$. The remaining claim of the theorem follows straightforwardly. □

Remark 1. *In Algorithm 1 (Lines 6 and 8), we adopt $\sqrt{1+\rho_k}$ to regulate the growth rate of the step size λ_k, but the square root form here can adopt some other choices. In fact, $\sqrt{1+\rho_k}$ can be replaced by $(1+\rho_k)^q$ for any constant $q > 0$, and this modification does not undermine the algorithm's theoretical guarantees. In fact, similar convergence results can still be established with only minor adjustments to the proofs. Specifically, the key adjustments are twofold: first, redefine the coefficient ω_k (denoted as $\omega_{k,q}$ here to distinguish it from the original ω_k) as follows:*

$$\omega_{k,q} = \frac{\lambda_k^2}{\lambda_0^2 \prod_{i=1}^{k}(1+\rho_{i-1})^{2q} \prod_{i=1}^{k}(1+\rho_{i-2})^{q}}, \quad \forall k \ge 1.$$

Second, update the lower bound λ and upper bound Λ of the step size λ_k; additionally, update the definition of ω in Lemma 4 to ensure it serves as a valid positive lower bound for $\omega_{k,q}$. With these adjustments in place, the remainder of the convergence proof remains entirely unchanged.

Remark 2. *From the above analysis, the key factors ensuring the convergence results in Theorem 1 are twofold: first, the boundedness of $\{\lambda_k\}$ established in Proposition 1; second, the satisfaction of condition* (21). *Guided by this insight, we can replace the step size rule in Algorithm 1 (Lines 5-9) with the following more relaxed one: $\lambda_k = \min\left\{\sqrt{1+\rho_k}\lambda_{k-1}, \left([L_k^2 + l_k/\lambda_{k-1}]_+\right)^{-\frac{1}{2}}\right\}$. In particular, this relaxed rule retains two critical theoretical properties: it still guarantees the satisfaction of* (21)*, and it can potentially yield larger step sizes than the original rule in Algorithm 1. Moreover, the boundedness of $\{\lambda_k\}$ can be established following a similar line of reasoning as in Proposition 1, and all subsequent theoretical results can be derived analogously under this relaxed step size rule. We did not adopt it in the main algorithm, however, because our current step size choice demonstrated better performance in our experiments.*

3.3 Convergence analysis for the convex setting

In this subsection, we establish additional convergence results, measured by the function value gap, for Algorithm 1 under the convexity assumption of f. First, we derive an upper bound for $\sum_{i=1}^{k} \lambda_i^2 \|\nabla f(x_i) + h'(x_i)\|^2$; notably, this bound does not rely on the convexity of f. Recall that $\lambda \le \lambda_k \le \Lambda$ for all k, with $\lambda, \Lambda > 0$ as defined in (8), and $\omega_k \ge \omega > 0$ for all k, with ω as defined in Lemma 3.

Lemma 5 (an upper bound for $\sum_{i=1}^{k} \lambda_i^2 \|\nabla f(x_i) + h'(x_i)\|^2$)**.** *Let $\{x_k\}$ and $\{\lambda_k\}$ denote the sequences generated by Algorithm 1. Then, it holds that*

$$\sum_{i=1}^{k} \lambda_i^2 \|\nabla f(x_i) + h'(x_i)\|^2 \le \mathcal{S} := \frac{\Lambda^2}{\lambda}\Big(\frac{2\Lambda^2}{\omega\lambda^2} V_0 + 2\big(F(x_0) - F_*\big)\Big) > 0. \tag{25}$$

Proof. It follows from (19) and (21) that

$$\begin{aligned} \lambda_k^2 \|\nabla f(x_k) + h'(x_k)\|^2 &\overset{(19)}{\le} \Big(\lambda_k^2 L_k^2 + \frac{\lambda_k^2}{\lambda_{k-1}} l_k\Big)\|x_k - x_{k-1}\|^2 + \frac{2\lambda_k^2}{\lambda_{k-1}}\big(F(x_{k-1}) - F(x_k)\big) \\ &\overset{(21)}{\le} \|x_k - x_{k-1}\|^2 + \frac{2\lambda_k^2}{\lambda_{k-1}}\big(F(x_{k-1}) - F(x_k)\big). \end{aligned} \tag{26}$$

Multiplying both sides of (26) by $\lambda_{k-1}/\lambda_k^2$, and utilizing (9) along with the relation $\mathcal{G}_{k-1} = \nabla f(x_{k-1}) + h'(x_k)$, we can reformulate (26) equivalently as

$$\lambda_{k-1}\|\nabla f(x_k) + h'(x_k)\|^2 \leq \frac{\lambda_{k-1}^3}{\lambda_k^2}\|\mathcal{G}_{k-1}\|^2 + 2\big(F(x_{k-1}) - F(x_k)\big).$$

By replacing the index k with i in the above inequality, then telescoping the resulting inequality over $i = 1, \ldots, k$, and further utilizing $F(x_k) \geq F_*$, we derive

$$\begin{aligned}\sum_{i=1}^{k}\lambda_{i-1}\|\nabla f(x_i) + h'(x_i)\|^2 &\leq \sum_{i=1}^{k}\frac{\lambda_{i-1}^3}{\lambda_i^2}\|\mathcal{G}_{i-1}\|^2 + 2\big(F(x_0) - F_*\big) \\ &\overset{(24)}{\leq} C := \frac{2\Lambda^2}{\omega\lambda^2}V_0 + 2\big(F(x_0) - F_*\big), \end{aligned} \tag{27}$$

where the second inequality also leverages the bounds $\lambda \leq \lambda_i \leq \Lambda$ and $\omega_i \geq \omega$ (holding for all i). Reapplying the bounds $\lambda \leq \lambda_i \leq \Lambda$ (valid for all i), we can further derive

$$\sum_{i=1}^{k}\lambda_i^2\|\nabla f(x_i) + h'(x_i)\|^2 = \sum_{i=1}^{k}\frac{\lambda_i^2}{\lambda_{i-1}}\cdot\lambda_{i-1}\|\nabla f(x_i) + h'(x_i)\|^2 \leq \frac{\Lambda^2 C}{\lambda} = \mathcal{S},$$

where $C > 0$ is defined as in (27) and $\mathcal{S}$ is defined in (25). This completes the proof. □

Next, we establish additional convergence rate results by leveraging the bound given in (25), under the convexity assumption on f.

Theorem 2. *Assume that f is convex and that the set of minimizers of F is nonempty. Let $\{x_k\}$ denote the sequence generated by Algorithm 1. Then, we have*

$$\min_{1\leq i\leq k} F(x_i) - F_* \leq \frac{\|x_1 - x_*\|^2 + \mathcal{S}}{2\lambda k} = \mathcal{O}\left(\frac{1}{k}\right),$$

where $x_* \in \mathbb{R}^n$ is any minimizer of F, i.e., $F(x_*) = F_*$, and $\mathcal{S} > 0$ is as defined in (25)*. Moreover, we have the following ergodic convergence rate result:*

$$F(\bar{x}_k) - F_* \leq \frac{\|x_1 - x_*\|^2 + \mathcal{S}}{2\lambda k} = \mathcal{O}\left(\frac{1}{k}\right), \text{ where } \bar{x}_k = \frac{\sum_{i=1}^k \lambda_i x_i}{\sum_{i=1}^k \lambda_i}. \tag{28}$$

This implies that to guarantee $F(\bar{x}_k) - F_* \leq \varepsilon$ for any given $\varepsilon > 0$, no more than $\mathcal{O}(1/\varepsilon)$ iterations are required.

Proof. By utilizing the convexity of h, we have

$$\lambda_k(h(x_{k+1}) - h(x_*)) \leq \lambda_k\langle h'(x_{k+1}), x_{k+1} - x_*\rangle \overset{(9)}{=} \langle x_k - x_{k+1} - \lambda_k\nabla f(x_k), x_{k+1} - x_*\rangle. \tag{29}$$

Multiply both sides of (29) by 2 and add it to the following identity

$$\|x_{k+1} - x_*\|^2 = \|x_k - x_*\|^2 - \|x_{k+1} - x_k\|^2 + 2\langle x_{k+1} - x_*, x_{k+1} - x_k\rangle,$$

we deduce

$$\|x_{k+1} - x_*\|^2 + 2\lambda_k(h(x_{k+1}) - h(x_*)) \leq \|x_k - x_*\|^2 - \|x_{k+1} - x_k\|^2 + 2\lambda_k\langle\nabla f(x_k), x_* - x_{k+1}\rangle. \tag{30}$$

Next, we estimate the term $\langle\nabla f(x_k), x_* - x_{k+1}\rangle$ in (30) by

$$\begin{aligned}\langle\nabla f(x_k), x_* - x_{k+1}\rangle &= \langle\nabla f(x_k), x_* - x_k\rangle + \langle\nabla f(x_k) + h'(x_k), x_k - x_{k+1}\rangle - \langle h'(x_k), x_k - x_{k+1}\rangle \\ &\leq f(x_*) - f(x_k) + \langle\nabla f(x_k) + h'(x_k), x_k - x_{k+1}\rangle + h(x_{k+1}) - h(x_k), \end{aligned} \tag{31}$$

where the inequality follows from the convexity of f and h. Plugging (31) into (30), rearranging the terms, and using $F = f + h$, we obtain

$$\begin{aligned}\|x_{k+1} - x_*\|^2 + 2\lambda_k(F(x_k) - F_*) &\leq \|x_k - x_*\|^2 - \|x_{k+1} - x_k\|^2 + 2\lambda_k\langle\nabla f(x_k) + h'(x_k), x_k - x_{k+1}\rangle \\ &\leq \|x_k - x_*\|^2 + \lambda_k^2\|\nabla f(x_k) + h'(x_k)\|^2,\end{aligned}$$

where the second inequality here follows from $2\langle u, v\rangle \leq \|u\|^2 + \|v\|^2$. By replacing the index k with i in the above inequality, then telescoping the resulting inequality over $i = 1, \ldots, k$, we deduce

$$2\sum_{i=1}^{k} \lambda_i (F(x_i) - F_*) \leq \|x_1 - x_*\|^2 + \sum_{i=1}^{k} \lambda_i^2 \|\nabla f(x_i) + h'(x_i)\|^2 \overset{(25)}{\leq} \|x_1 - x_*\|^2 + \mathcal{S}. \tag{32}$$

It then follows straightforwardly from (32) that

$$\min_{1\leq i\leq k} F(x_i) - F_* \leq \frac{\|x_1 - x_*\|^2 + \mathcal{S}}{2\sum_{i=1}^{k} \lambda_i} \leq \frac{\|x_1 - x_*\|^2 + \mathcal{S}}{2\lambda k} = \mathcal{O}\left(\frac{1}{k}\right). \tag{33}$$

Furthermore, following a similar derivation to that of (33), but instead leveraging the convexity of F and Jensen's inequality, we can derive the ergodic convergence rate result given in (28) from (32). We omit the details for brevity, and this concludes the proof. $\square$

Remark 3. *Let $\varepsilon > 0$. From* (33)*, it can be inferred that choosing $k = \lceil (\|x_1 - x_*\|^2 + \mathcal{S})/(2\lambda\varepsilon)\rceil$ is sufficient to guarantee $\min_{1\leq i\leq k} F(x_i) - F_* \leq \varepsilon$. In other words, to ensure the existence of some index $i_0 \in \{1, \ldots, k\}$ satisfying $F(x_{i_0}) - F_* \leq \varepsilon$, no more than $\mathcal{O}(1/\varepsilon)$ iterations are needed, a complexity bound directly implied by the expression for k.*

Remark 4. *When the objective function f is convex, we always have $l_k \leq 0$. Leveraging this property, we can incorporate the short Barzilai-Borwein (BB) step size (cf. [1, 26]) into the design of a novel step size formula, given by*

$$\lambda_k = \min\left\{\sqrt{1+\rho_k}\lambda_{k-1}, \lambda_k^{BB}\right\} \quad \textit{with} \quad \lambda_k^{BB} := \frac{\langle \nabla f(x_k) - \nabla f(x_{k-1}), x_k - x_{k-1}\rangle}{\|\nabla f(x_k) - \nabla f(x_{k-1})\|^2}. \tag{34}$$

The theoretical validity of this step size choice is justified by the following two key observations: (i) Owing to the Lipschitz continuity of ∇f and the convexity of f, λ_k^{BB} is inherently bounded below by $1/L$ (see, e.g., [21, Thm. 2.1.5]). Then, by incorporating this observation into the proof of Proposition 1, we can show that $\min\{\lambda_0, 1/L\} \leq \lambda_k \leq \Lambda$. (ii) By the Cauchy-Schwarz inequality, λ_k always satisfies $\lambda_k \leq \lambda_k^{BB} \leq \|x_k - x_{k-1}\|/\|\nabla f(x_k) - \nabla f(x_{k-1})\| = 1/L_k$. Given that $l_k \leq 0$, the step size λ_k automatically satisfies the general step size condition specified in (21)*. This extension underscores the flexibility of condition* (21) *with respect to step size selection.*

4 Numerical experiments

In this section, we evaluate the performance of our proposed AdaPGNC algorithm and variants on five benchmark optimization problems, comparing them with several state-of-the-art parameter-free methods. Of the five test problems, four are nonconvex, and one is the convex logistic regression problem. All experiments were implemented in Python on a workstation equipped with an AMD Ryzen 9 5900HX CPU and 16GB RAM.

In Section 1, we reviewed various parameter-free optimization algorithms for nonconvex settings. Among these, NASCAR [12] and PF.APD [10] feature intricate multi-loop structures, rendering them unsuitable for general applications. On the other hand, AC-PG [11] requires the problem being solved to have a bounded domain—this is inconsistent with our study and experimental focus. Thus, we do not compare our algorithms with NASCAR, PF.APD and AC-PG. Our experiments proceed as follows.

- First, we compare AdaPGNC with AGDR in [20, Alg. 4.1] and GD-LS (gradient descent with line-search) on three nonconvex problems satisfying Assumption 1.
- Second, we test a nonnegative matrix factorization (NMF) problem that does not satisfy Assumption 1. For this NMF problem, we compare AdaPGNC only with NPG1 and NPG2: [9] reports that NPG1 and NPG2 outperform line-search techniques, and AGDR is unsuitable for solving this problem as it requires estimating the Lipschitz constant of $\nabla^2 f$ via M_k, which, by its definition in [20], can diverge to infinity.

- Finally, we evaluate the BB-type variant of AdaPGNC given in (34) against AdaBB and AdaBB3 proposed in [26] on the convex logistic regression problem.

For AdaPGNC, we employ two distinct sequences ρ_k, inspired by [18, 8, 9]:

$$\rho_k^{(1)} = \min\left\{\frac{\lambda_k}{\lambda_{k-1}}, \frac{100(\ln(k+1))^4}{(k+1)^{1.1}}\right\} \quad \text{and} \quad \rho_k^{(2)} = \frac{100(\ln(k+1))^4}{(k+1)^{1.1}}, \quad \forall k \geq 1,$$

with $\rho_0 = 10^{10}$ for both cases. A large ρ_0 permits virtually unrestricted initial growth of λ_1, enabling rapid adjustment to local curvature. The initial step size λ_0 is problem-dependent and will be specified later. For direct comparability, our algorithm was integrated into established experimental frameworks: the codebase from [20, 9] for nonconvex test problems and from [18] for the convex logistic regression problem. For unbiased comparison, all algorithmic parameters under consideration were adjusted to the most stable and efficient values reported in their original publications [20, 26, 9]. The details of all the algorithms compared are summarized in Table 2.

Table 2: Summary of compared algorithms and their parameter configurations.

Algorithm	**Description and Parameters**
AdaPGNC-1	Algorithm 1 with $\rho_k = \rho_k^{(1)}$ for $k \geq 1$ and $\rho_0 = 10^{10}$.
AdaPGNC-2	Algorithm 1 with $\rho_k = \rho_k^{(2)}$ for $k \geq 1$ and $\rho_0 = 10^{10}$.
GD-LS	Gradient descent with Armijo line-search. Backtracking finds the smallest nonnegative integer $m_k \geq 0$ such that: $f(x_k - 10^{-3} \cdot 2^{-m_k}\nabla f(x_k)) \leq f(x_k) - 10^{-3} \cdot 2^{-(m_k+1)}\|\nabla f(x_k)\|^2$.
AGDR	[20, Alg. 4.1] with $(L_{\text{init}}, M_0, \alpha, \beta) = (10^{-3}, 10^{-16}, 2, 0.9)$.
NPG1	[9, Alg. 3.1] with $\gamma_{k-1} = 0.1(\ln k)^{5.7}/k^{1.1}$ for $k \geq 1$, $c_0 = 0.7$, and $c_1 = 0.69$.
NPG2	[9, Alg. 4.1] with $\gamma_{k-1} = 0.1(\ln k)^{5.7}/k^{1.1}$ for $k \geq 1$, $c_0 = 0.99$, and $c_1 = 0.98$.
AdaPGNC-BB-1	Algorithm 1 with step size (34), $\rho_k = \rho_k^{(1)}$ for $k \geq 1$, and $\rho_0 = 10^{10}$.
AdaPGNC-BB-2	Algorithm 1 with step size (34), $\rho_k = \rho_k^{(2)}$ for $k \geq 1$, and $\rho_0 = 10^{10}$.
AdaBB **AdaBB3**	The two top-performing variants of AdaBB in [26, Table 2], implemented with their original parameters.

4.1 Experiments on four nonconvex problems

This subsection evaluates the performance of the algorithms on four nonconvex problems: a classification problem, an autoencoder training problem, a low-rank matrix completion problem, and the NMF problem. For the first three problems, we set the initial step size $\lambda_0 = 1$ for AdaPGNC-1 and AdaPGNC-2. We report the minimal function values achieved within time T, defined as

$$\begin{aligned} f_T &= \text{the minimal function value } f(x_k) \text{ until time } T, \\ g_T &= \text{the minimal gradient norm } \|\nabla f(x_k)\| \text{ until time } T, \end{aligned}$$

and plot them against runtime. For the NMF problem, we set a uniform initial step size of $\lambda_0 = 0.001$ for AdaPGNC-1, AdaPGNC-2, and NPG. Details of each problem are elaborated below.

Classification problem. We consider the nonconvex classification problem:

$$\min_{\omega \in \mathbb{R}^n} f(\omega) = \frac{1}{N}\sum_{i=1}^{N} \ell_{CE}(y_i, \phi(z_i; \omega)), \tag{35}$$

which minimizes the cross-entropy loss over a neural network. The dataset comprises $N = 10{,}000$ samples randomly selected from the MNIST dataset, where inputs $\{z_i\}_{i=1}^N \subset \mathbb{R}^{784}$ are images and outputs $\{y_i\}_{i=1}^N \subset \{0,1\}^{10}$ are one-hot encoded labels. The network model $\phi(\cdot;\omega) : \mathbb{R}^{784} \to \mathbb{R}^{10}$ is constructed as a three-layer fully-connected neural network, incorporating bias and utilizing logistic sigmoid activation, where $\sigma(z) = 1/(1+\exp(-z))$. The layer dimensions are 784, 32, 16, and 10, resulting in n = 25,818 trainable parameters $\omega \in \mathbb{R}^n$. The cross-entropy loss, denoted as ℓ_{CE}, for a label $y_i = (y_{i,1}, \dots, y_{i,10})^\top$ and a prediction $p_i = \phi(z_i;\omega) = (p_{i,1}, \dots, p_{i,10})^\top$ is defined as $\ell_{CE}(y_i, p_i) = -\sum_{k=1}^{10} y_{i,k} \log p_{i,k}$. The default `flax.linen.Module.init` method is used for parameter initialization. We terminate the compared algorithms either after 500 seconds have elapsed or when the gradient norm is less than 10^{-10}. Figure 1 shows the comparison results for minimal function values, minimal gradient norms, and step sizes.

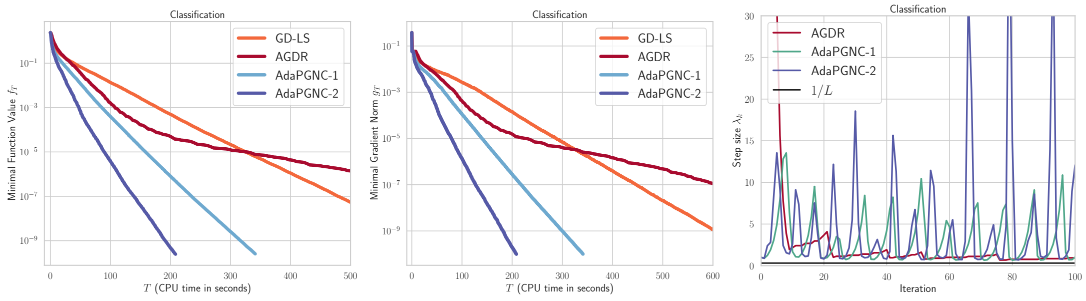

Figure 1: Comparison results for the classification problem.

Figure 1 shows that AdaPGNC-2 converges to the lowest objective values and gradient norms, followed by AdaPGNC-1; both outperform AGDR and GD-LS. The superior performance of AdaPGNC-2 stems from its step size behavior: as shown in the third plot, AdaPGNC-2 achieves larger step sizes than both AdaPGNC-1 and AGDR. This indicates that AdaPGNC-2's adaptive mechanism more effectively leverages favorable local curvature, enabling larger steps that contribute to its accelerated convergence.

Autoencoder training problem. We next consider the nonconvex autoencoder problem, formulated as follows:

$$\min_{\vartheta \in \mathbb{R}^n} f(\vartheta) = \frac{1}{2MN} \sum_{i=1}^N \|u_i - \varphi(u_i;\vartheta)\|^2, \tag{36}$$

whose purpose is to learn a compressed data representation by minimizing the reconstruction error, measured by the squared L^2-norm. The model $\varphi(\cdot;\vartheta) : \mathbb{R}^{784} \to \mathbb{R}^{784}$ is a four-layer fully-connected neural network with bias parameters and sigmoid activation. The layer dimensions are 784, 32, 16, 32, and 784, resulting in n = 52,064 trainable parameters $\vartheta \in \mathbb{R}^n$. We use the same MNIST dataset ($\{u_i\}_{i=1}^N \subset \mathbb{R}^{784}$ with $M = 784$) as in the classification problem (35). The default `flax.linen.Module.init` method is used for parameter initialization. The same as before, we terminate the compared algorithms either after 500 seconds have elapsed or when the gradient norm is found to be less than 10^{-10}. Figure 2 shows the detailed comparison results.

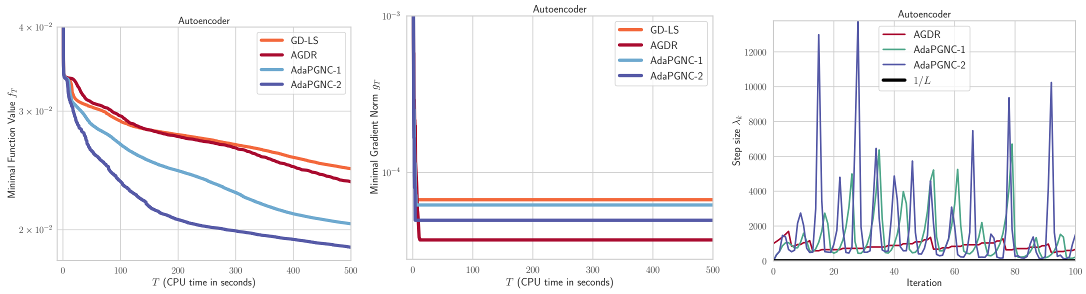

Figure 2: Comparison results for the autoencoder training problem.

As shown in Figure 2, AdaPGNC-2 achieves the steepest reduction in function values, followed by AdaPGNC-1, AGDR, and GD-LS. In terms of gradient norms, AGDR reaches its minimum value the fastest, with AdaPGNC-2, AdaPGNC-1, and GD-LS following in sequence. The plateaus observed in the gradient norm plot after approximately 20 seconds occur because each algorithm has found its minimum achievable gradient norm by that time; since we report the best value found up to T seconds, the curves flatten once no further improvement is made. Consistent with previous observations, both AdaPGNC variants attain significantly larger step sizes than AGDR. This likely explains the superior performance of AdaPGNC variants in minimizing the objective function.

Low-rank matrix completion problem. We next consider the low-rank matrix completion problem [24, 25], formulated as follows:

$$\min_{Z=(U,V)\in\mathbb{R}^{p\times r}\times\mathbb{R}^{q\times r}} f(Z) = \frac{1}{2N}\sum_{(i,j,s)\in\Omega}\left((UV^\top)_{ij} - s\right)^2 + \frac{1}{2N}\left\|U^\top U - V^\top V\right\|_F^2. \tag{37}$$

The goal is to find a rank-r approximation $UV^\top$ that completes missing entries in a matrix, using a set Ω of N observed entries (i,j,s) where s denotes the value of the (i,j)-th element. The regularization term $||U^\top U - V^\top V||_F^2$ ensures consistency between the factors $U \in \mathbb{R}^{p\times r}$ and $V \in \mathbb{R}^{q\times r}$ (see [23] for details). We evaluated the algorithms on the MovieLens-100K and MovieLens-1M collaborative filtering datasets [7]. Table 3 presents the details about the data and matrix dimensions for both datasets. We terminate the compared algorithms either after 600 seconds have elapsed or when the gradient norm is found to be less than 10^{-10}. The experimental results are shown in Figures 3 and 4 for the two datasets.

Table 3: Details about the data and matrix dimensions for the matrix completion problem (37).

Dataset	**MovieLens-100K**	**MovieLens-1M**
Rank: r	100	500
Row dimension of U**:** p	943	6,040
Row dimension of V**:** q	1,682	3,900
Number of observed entries: N	100,000	1,000,209
Number of parameters: $(p+q)r$	262,500	4,970,000

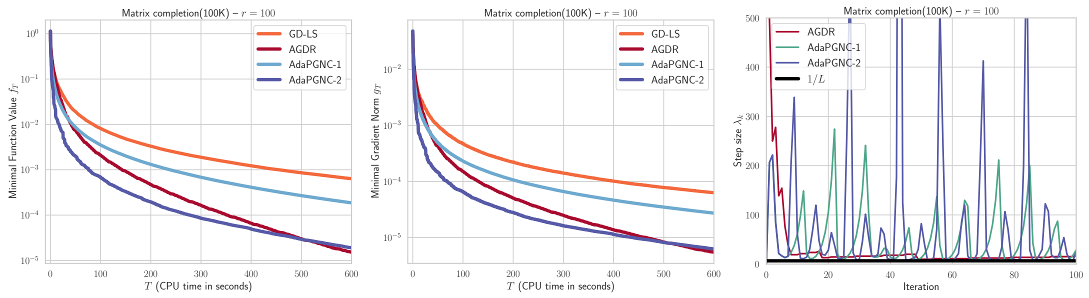

Figure 3: Comparison results for the matrix completion problem on MovieLens-100K.

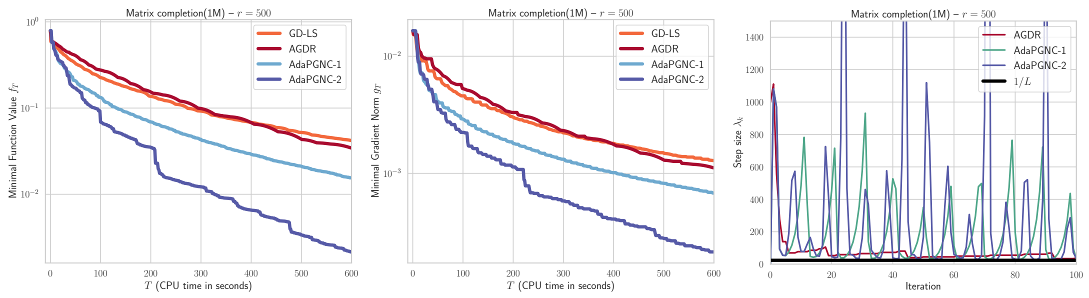

Figure 4: Comparison results for the matrix completion problem on MovieLens-1M.

Figures 3 and 4 show that for the smaller MovieLens-100K dataset, AdaPGNC-2 and AGDR perform best, followed by AdaPGNC-1 and GD-LS. For the larger MovieLens-1M dataset, however, both AdaPGNC-2 and AdaPGNC-1 outperform AGDR and GD-LS significantly in reducing the objective value and gradient norm. This observation suggests that the performance advantage of AdaPGNC algorithms becomes more pronounced as problem dimension increases. Furthermore, the step size plots reveal that AdaPGNC-2 attains significantly larger step sizes than both AdaPGNC-1 and AGDR across both datasets. This consistent behavior demonstrates that AdaPGNC-2's adaptive mechanism adapts more effectively to local curvature, enabling more efficient progress toward convergence.

Nonnegative matrix factorization (NMF) problem. We next consider the following NMF problem [15]:

$$\min_{Z:=(U,V)\in\mathbb{R}_+^{n\times r}\times\mathbb{R}_+^{m\times r}} f(Z)=\frac{1}{2}\|UV^\top-A\|_F^2, \tag{38}$$

where $A\in\mathbb{R}^{n\times m}$ is a given data matrix, r is a positive integer representing the factorization rank. This problem does not satisfy Assumption 1 because the objective function is not globally L-smooth; therefore, our algorithms are treated as heuristic methods for solving it. Consistent with [19], we generate A by multiplying matrices B and $C^\top$, where $B\in\mathbb{R}_+^{m\times r}$ and $C\in\mathbb{R}_+^{n\times r}$ have entries sampled from a normal distribution $\mathcal{N}(0,1)$. Negative entries of B and C are replaced with zeros. All methods are initialized with two random matrices (U_0,V_0). For each tuple (n,r,m), we run the algorithms with 10 different random seeds. We terminate the compared algorithms when the gradient residual $\|\mathcal{G}_k\|$ is found to be less than 10^{-6}. Table 4 presents the average comparison results for AdaPGNC and NPG.

Table 4 shows that across all tested scenarios, AdaPGNC-2 almost always achieves the best performance in terms of number of iterations, CPU time, and optimality residuals (GradRes and OptGap in the table). AdaPGNC-1 performs slightly worse than AdaPGNC-2, and both AdaPGNC variants outperform the two NPG variants significantly.

4.2 Experiments on a convex logistic regression problem

In this subsection, we consider the following convex logistic regression problem with ℓ_2-square regularization:

$$\min_{x\in\mathbb{R}^n} f(x)=-\frac{1}{m}\sum_{i=1}^m\left[y_i\log(\sigma(a_i^\top x))+(1-y_i)\log(1-\sigma(a_i^\top x))\right]+\frac{\gamma}{2}\|x\|^2, \tag{39}$$

where $a_i\in\mathbb{R}^n$, $y_i\in\{0,1\}$, $\sigma(z)=1/(1+\exp(-z))$ is the sigmoid function, m is the number of observations, and $\gamma>0$ is a regularization parameter. The gradient of f is given by $\nabla f(x)=\frac{1}{m}\sum_{i=1}^m a_i(\sigma(a_i^\top x)-y_i)+\gamma x$, and f is L-smooth with $L=\frac{1}{4}\lambda_{\max}(A^\top A)+\gamma$, where $A=(a_1,\ldots,a_m)^\top$ and $\lambda_{\max}(A^\top A)$ stands for the largest eigenvalue of $AA^\top$. We conducted experiments on the mushrooms, w8a, and covtype datasets from LIBSVM [4]. Details and parameters of these datasets are provided in Table 5.

For this test, we compared AdaPGNC-BB-1 and AdaPGNC-BB-2 with AdaBB and AdaBB3 (see Table 2). All four algorithms use an initial step size of $\lambda_0=10^{-10}$. Figure 5 plots the optimality gap $\min_{0\le i\le k}(f(x_i)-f_*)$ and gradient norm $\min_{0\le i\le k}\|\nabla f(x_i)\|$ against iteration count, where f_* is approximated by running AdaPGNC-BB-1 and AdaPGNC-BB-2 for an additional 1000 iterations beyond the specified maximum. The results show that both AdaPGNC variants consistently outperform AdaBB and AdaBB3 across all datasets, demonstrating superior efficiency and effectiveness in solving these logistic regression problems. Additionally, Figure 6 presents the step sizes generated by the four algorithms, revealing that our methods generally produce larger step sizes than AdaBB and AdaBB3—this helps explain their improved performance.

5 Conclusions

This paper proposes AdaPGNC, a parameter-free and line-search-free adaptive proximal gradient method for general nonconvex optimization problems. The proposed algorithm extends the capabilities of classical adaptive gradient methods to general composite nonconvex problems, while eliminating

Table 4: Average results for the NMF problem (38). In the table, "GradRes" denotes $\min_k \|\mathcal{G}_k\|$, where $\|\mathcal{G}_k\| = \|\mathcal{G}_{\lambda_k}(Z_k)\| = \|(Z_{k+1} - Z_k)/\lambda_k\|$; "OptGap" denotes $\min_k f(Z_k) - f_*$, with f_* approximated by the minimum of $f(Z_k)$ over all iterations and tested algorithms; and Time is measured in seconds. Best results are highlighted in bold.

Dimensions	**Metrics**	**Average results for ten random data**			
(n, r, m)		**NPG1**	**NPG2**	**AdaPGNC-1**	**AdaPGNC-2**
(2000, 20, 3000)	Iterations	1,368.1	1,149.4	743.8	**651.8**
	GradRes	9.8e-07	9.7e-07	**9.4E-07**	**9.4E-07**
	OptGap	1.1e-15	6.8e-16	**4.5E-16**	5.8E-16
	Time	76.3	63.8	42.3	**36.6**
(3000, 20, 2000)	Iterations	1,411.8	1,167	752.6	**645.3**
	GradRes	9.8e-07	9.8e-07	9.6E-07	**9.4E-07**
	OptGap	5.8e-16	8.1e-16	3.1E-16	**2.1E-16**
	Time	83.7	69.0	44.8	**38.4**
(3000, 20, 3000)	Iterations	1,496.7	1,233.9	758.1	**644.4**
	GradRes	9.9e-07	9.8e-07	9.6E-07	**9.1E-07**
	OptGap	1.3e-15	9.5e-16	6.7E-16	**4.4E-17**
	Time	123.1	101.7	62.8	**53.5**
(2000, 30, 3000)	Iterations	2,864.4	2,424.2	1,360.2	**1,178.6**
	GradRes	9.9E-07	9.9E-07	9.7E-07	**9.5E-07**
	OptGap	6.9E-16	1.1E-15	7.1E-16	**2.4E-16**
	Time	167.4	141.9	81.0	**70.4**
(3000, 30, 2000)	Iterations	2,875.3	2,461.5	1,391.2	**1,178.6**
	GradRes	9.9E-07	9.9E-07	9.8E-07	**9.7E-07**
	OptGap	1.1E-15	9E-16	5.3E-16	**7.8E-17**
	Time	178.0	152.4	82.3	**75.0**
(3000, 30, 3000)	Iterations	3,076.7	2,582	1,362.6	**1,130.4**
	GradRes	9.9E-07	9.9E-07	9.8E-07	**9.6E-07**
	OptGap	7.1E-16	9.2E-16	4.9E-16	**2.2E-16**
	Time	261.6	219.2	117.3	**97.3**

Table 5: Datasets used in the logistic regression problem.

Dataset	m	n	L	γ	Max Iteration
mushrooms	8124	112	2.59	L/m	1,000
w8a	49,749	300	0.66	L/m	3,000
covtype	581,012	54	5.04×10^6	$L/(10m)$	10,000

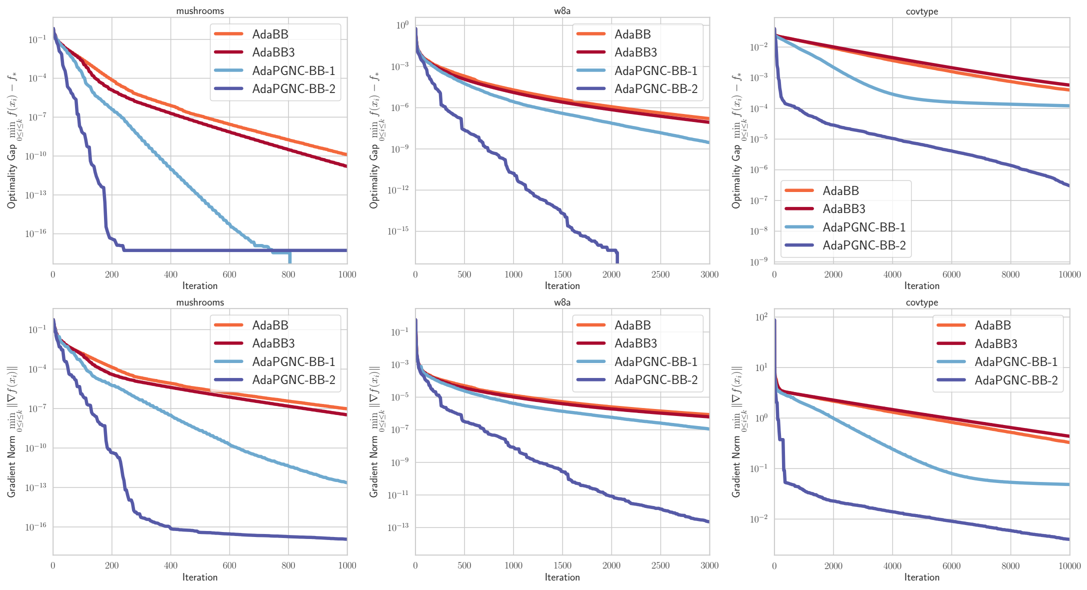

Figure 5: Comparison results for the logistic regression problem across the three datasets.

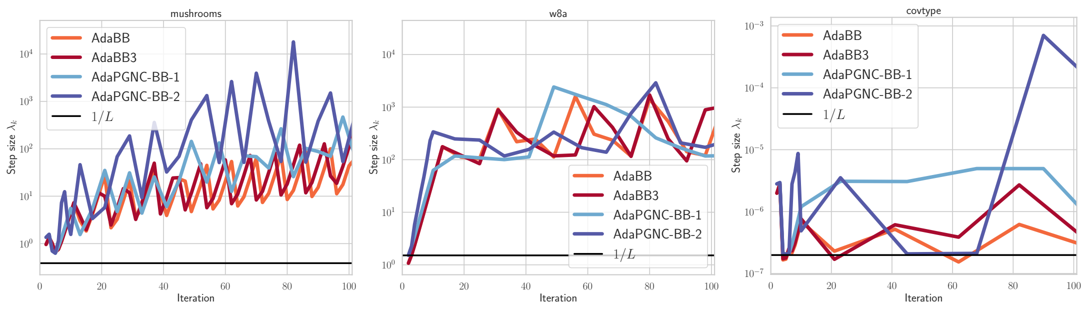

Figure 6: Step sizes generated by AdaBB, AdaBB3, AdaPGNC-BB-1 and AdaPGNC-BB-2.

the need for the bounded-iterate assumption or second-order smoothness requirements. Furthermore, it features a single-loop structure and requires no restarts. Theoretically, we construct a novel Lyapunov function to establish convergence rates under mild conditions, while simultaneously resolving the challenging question raised in [12] regarding the feasibility of simplified parameter-free frameworks for nonconvex optimization. Thus, this work bridges the gap between algorithmic simplicity and theoretical guarantees for general composite nonconvex optimization problems. Even in the convex setting, our analysis offers new insights into the design of adaptive gradient methods—for instance, the short Barzilai-Borwein (BB) step size can be seamlessly incorporated into the step size design. Numerical experiments further demonstrate the competitive performance of AdaPGNC against various benchmark parameter-free methods for both nonconvex and convex optimization problems, validating its practical efficiency alongside its theoretical guarantees. An interesting direction for future work is to develop meaningful heuristics for determining the nonnegative summable sequence $\{\rho_k\}$ dynamically. Currently, this sequence must be determined offline.

References

[1] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[2] A. Beck, *First-Order Methods in Optimization*, SIAM, Philadelphia, PA, 2017.

[3] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *Lower bounds for finding stationary points I*, Math. Program., 184 (2020), pp. 71–120.

[4] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intell. Syst. Technol., 2 (2011), pp. 1–27.

[5] D. Davis and D. Drusvyatskiy, *Stochastic model-based minimization of weakly convex functions*, SIAM J. Optim., 29 (2019), pp. 207–239.

[6] D. Drusvyatskiy, *The Proximal Point Method Revisited*, preprint, arXiv:1712.06038, (2017).

[7] F. M. Harper and J. A. Konstan, *The Movielens datasets: History and context*, ACM Trans. Interact. Intell. Syst., 5 (2015), pp. 1–19.

[8] P. T. Hoai, N. T. Vinh, and N. P. H. Chung, *A novel stepsize for gradient descent method*, Oper. Res. Lett., 53 (2024), pp. No. 107072, 8.

[9] P. T. Hoaia and N. P. D. Thaia, *Composite Optimization Models via Proximal Gradient Method with a Novel Enhanced Adaptive Stepsize*, Optimization Online, preprint, `https://optimization-online.org/2010/08/2714/`, (2010).

[10] W. Kong, *Complexity-optimal and parameter-free first-order methods for finding stationary points of composite optimization problems*, SIAM J. Optim., 34 (2024), pp. 3005–3032.

[11] G. Lan, T. Li, and Y. Xu, *Projected gradient methods for nonconvex and stochastic optimization: New complexities and auto-conditioned stepsizes*, preprint, arXiv:2412.14291, (2024).

[12] G. Lan, Y. Ouyang, and Z. Zhang, *Optimal and parameter-free gradient minimization methods for convex and nonconvex optimization*, preprint, arXiv:2310.12139, (2024).

[13] P. Latafat, A. Themelis, and P. Patrinos, *On the convergence of adaptive first order methods: Proximal gradient and alternating minimization algorithms*, in Proceedings of the 6th Annual Learning for Dynamics and Control Conference, vol. 242 of Proc. Mach. Learn. Res., PMLR, 2024, pp. 197–208.

[14] P. Latafat, A. Themelis, L. Stella, and P. Patrinos, *Adaptive proximal algorithms for convex optimization under local lipschitz continuity of the gradient*, Math. Program., (2024), pp. 1–39.

[15] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.

[16] T. Li and G. Lan, *A simple uniformly optimal method without line search for convex optimization*, Math. Program., (2025), pp. 1–38.

[17] H. Liu, T. Wang, and Z. Liu, *Some modified fast iterative shrinkage thresholding algorithms with a new adaptive non-monotone stepsize strategy for nonsmooth and convex minimization problems*, Comput. Optim. Appl., 83 (2022), pp. 651–691.

[18] Y. Malitsky and K. Mishchenko, *Adaptive gradient descent without descent*, in Proceedings of the 37th International Conference on Machine Learning, vol. 119 of Proc. Mach. Learn. Res., PMLR, 2020, pp. 6702–6712.

[19] Y. Malitsky and K. Mishchenko, *Adaptive proximal gradient method for convex optimization*, Adv. Neural Inf. Process. Syst., 37 (2024), pp. 100670–100697.

[20] N. Marumo and A. Takeda, *Parameter-free accelerated gradient descent for nonconvex minimization*, SIAM J. Optim., 34 (2024), pp. 2093–2120.

[21] Y. Nesterov, *Lectures on Convex Optimization*, Springer Optim. Appl. 137, Springer, New York, 2018.

[22] J. J. Suh and S. Ma, *An adaptive and parameter-free Nesterov's accelerated gradient method for convex optimization*, preprint, arXiv:2505.11670, (2025).

[23] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht, *Low-rank solutions of linear matrix equations via procrustes flow*, in Proceedings of the 33rd International Conference on Machine Learning, vol. 48 of Proc. Mach. Learn. Res., PMLR, 2016, pp. 964–973.

[24] F. Wen, L. Chu, P. Liu, and R. C. Qiu, *A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning*, IEEE Access, 6 (2018), pp. 69883–69906.

[25] Q. Yao and J. T. Kwok, *Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity*, J. Mach. Learn. Res., 18 (2018), pp. 1–52.

[26] D. Zhou, S. Ma, and J. Yang, *AdaBB: Adaptive Barzilai-Borwein method for convex optimization*, Math. Oper. Res., (2025).