Locally Linear Convergence for Nonsmooth Convex Optimization via Coupled Smoothing and Momentum

Reza Rahimi Baghbadorani[1], Sergio Grammatico[1], and Peyman Mohajerin Esfahani[1,2]

Abstract. We propose an adaptive accelerated smoothing technique for a nonsmooth convex optimization problem where the smoothing update rule is coupled with the momentum parameter. We also extend the setting to the case where the objective function is the sum of two nonsmooth functions. With regard to convergence rate, we provide the global (optimal) sublinear convergence guarantees of $\mathcal{O}(1/k)$, which is known to be provably optimal for the studied class of functions, along with a local linear rate if the nonsmooth term fulfills a so-call *locally strong convexity condition*. We validate the performance of our algorithm on several problem classes, including regression with the ℓ_1-norm (the Lasso problem), sparse semidefinite programming (the *MaxCut* problem), Nuclear norm minimization with application in model free fault diagnosis, and ℓ_1-regularized model predictive control to showcase the benefits of the coupling. An interesting observation is that although our global convergence result guarantees $\mathcal{O}(1/k)$ convergence, we consistently observe a practical transient convergence rate of $\mathcal{O}(1/k^2)$, followed by asymptotic linear convergence as anticipated by the theoretical result. This two-phase behavior can also be explained in view of the proposed smoothing rule.

Keywords: Adaptive stepsize, Nonsmooth optimization, first-order methods, composite convex optimization

1. Introduction

Objective functions with multiple nonsmooth terms are widely used in optimization-based control, system identification and machine learning. The broad applicability of these formulations across diverse domains calls for efficient algorithms capable of handling nonsmooth optimization problems. For instance, the authors in [2, 37] consider model predictive control (MPC) with nonsmooth ℓ_1 regularizers to obtain sparse control inputs. The authors in [35] use combinations of ℓ_1 and Nuclear norms to model fault detection in model-free dynamical systems. [18] employ ℓ_2–ℓ_1 and ℓ_1–ℓ_1 formulations for image restoration; [12] utilize the TV–ℓ_1 model for image denoising; [47] apply the ℓ_1–ℓ_1 formulation to dictionary learning; and [48] use Nuclear norms for low-rank matrix decomposition in graph neural networks with an ℓ_1 regularizer.

Motivating example. Consider the MPC formulation for a class of constrained linear systems with uncertain state-delays [22]:

$$\begin{aligned} x(k+1) &= \mathbf{A}x(k) + \mathbf{A_d}x(k - N_d(x_k)) + \mathbf{B}u(k), \\ N_d(k) &\in [1, \bar{N}_d]; \quad x(i) = x_i, \quad \forall i \in \{-\bar{N}_d, \dots, 0\}, \\ \|u_i\| &\leq \bar{u}_i, \quad i \in \{1, \dots, m\}, \end{aligned} \tag{1}$$

Date: May 11, 2026.

The authors are with (1) Delft University of Technology and (2) the University of Toronto. This work was supported by the ERC grant TRUST-949796 and the NSERC Discovery grant RGPIN-2025-06544.

where k is the discrete-time index, $x \in \mathbb{R}^n$ represents the system state, $u \in \mathbb{R}^m$ is the input vector, $\bar{u}_i$ are the input constraints, $N_d(k)$ is the uncertain time delay, possibly varying with time, $\bar{N}_d$ is its upper bound, and the matrices $\mathbf{A}$, $\mathbf{A_d}$, and $\mathbf{B}$ define the system dynamics.
Based on an artificial Lyapunov function, a stabilizing condition depending on the upper bound of the uncertain state-delays, together with MPC, is presented in the form of a linear matrix inequality as [22, Eq. (15)]:

$$\min_{u(k|k),\ldots,u(k+N-1|k)} \; J(k) \tag{2}$$
$$\text{s.t. system dynamics and constraints (1)},$$
$$\begin{bmatrix} P - \bar{N}_d Q_d & 0 & (\mathbf{A}+\mathbf{B}K)^\top \\ 0 & Q_d & \mathbf{A}_d^\top \\ \mathbf{A}+\mathbf{B}K & \mathbf{A}_d & P^{-1} \end{bmatrix} \succ 0,$$
$$\begin{bmatrix} Y & K \\ K^\top & P\mu^{-1} \end{bmatrix} \succeq 0, \quad Y_{ij} \leq \bar{u}_j^2, \quad \forall i,j \in \{1,\ldots,m\},$$

where, J is a nonsmooth objective function that can include an ℓ_1 regularizer, e.g., $\sum_i \|\Delta u_i\|_1$, to enforce sparsity in input changes, allowing the controller to respond promptly to disturbances and system variations [37, 2]. We emphasize that, the MPC problem in (2), is complex due to the nonsmooth objective function and SDP constraints, and must be solved at every sampling time.

This motivates fast optimization algorithms for nonsmooth convex problem of the following class:

$$\min_{x\in\mathbb{R}^n} F(x) = \min_{x\in\mathbb{R}^n} f(x) + h(x), \tag{3}$$

where the functions f and h are proper, closed, convex, possibly nonsmooth, but prox-friendly (to be made precise later in the notation section, see (4)). A particular subclass of such problems is composite minimization in which one of these functions is smooth [7, 34, 4]. While the composite minimization problem has been extensively studied, the possibility of having multiple nonsmooth terms remains relatively unexplored. This is primarily due to the fact that the summation of two prox-friendly functions is not necessarily prox-friendly [40]. Let us emphasize that the most common algorithms, such as subgradient [28], mirror descent [6], and bundle methods [42] often suffer from a slow convergence rate of $\mathcal{O}\left(1/\varepsilon^2\right)$ where ε is an apriori desired precision.

Following the seminal works by Nesterov [32, 33], one can exploit the structure of the nonsmooth objective function to propose a smooth approximation and then deploy a first-order accelerated method to find the optimizer of the smooth approximation [30]. Remarkably, Nesterov's algorithm requires only $\mathcal{O}(1/\varepsilon)$ iterations to reach an ε optimal solution. Inspired by this approach, several works have tried to improve and enhance the efficacy of smoothing algorithms in terms of complexity or to customize it for specific applications at hand [29, 31, 12, 16]. In [8], comparisons are made about the advantages and disadvantages of smoothing techniques and proximal-type methods. Based on [33], the authors in [17] utilize a stochastic smoothing technique to improve the scalability of the algorithm for semidefinite programming problems. The smoothing techniques in [50] improve the convergence speed in comparison with conditional gradient algorithms. The authors in [36] study the smoothing technique for minimizing the sum of three functions, where two of which are nonsmooth. The paper [11] studies an adaptive smoothing algorithm with a convergence rate of $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))$.

TABLE 1. Comparison of algorithms for minimizing nonsmooth convex functions.

Algorithm	Smoothing technique	prox-friendly assumption	Local linear convergence rate	Global convergence rate
Subgradient descent [28]	✗	✗	✗	$\mathcal{O}(1/\varepsilon^2)$
Stochastic smoothing [17]	✗	✗	✗	$\mathcal{O}(1/\varepsilon^2)$
Chambolle-Poc [12]	✗	✓	✗	$\mathcal{O}(1/\varepsilon)$
Nesterov's smoothing [32, 33]	✓	✓	✗	$\mathcal{O}(1/\varepsilon)$
Variable smoothing [11]	✓	✓	✗	$\mathcal{O}(\log(1/\varepsilon)/\varepsilon)$
Adaptive smoothing [46]	✓	✓	✗	$\mathcal{O}(1/\varepsilon)$
Proposed adaptive smoothing (Theorems 3.3–3.4)	✓	✓	✓ (under some assumptions)	$\mathcal{O}(1/\varepsilon)$

The closest existing work to our study is the adaptive smoothing technique in [46] that enjoys $\mathcal{O}(1/\varepsilon)$ complexity, which matches the worst-case bound offered by Nesterov's algorithm. Table 1 summarizes the convergence results of this study and compares them with those in the literature. Following the footsteps of Nesterov's smoothing technique, we propose a novel adaptive-smoothing technique where the smoothing parameter decreases with each iteration but as a function of the momentum, which retains the same global convergence rate while improving the asymptotic performance.

Contributions: The contributions of our work are summarized as follows:

(i) **Global optimal sublinear convergence:** We introduce an algorithm with an adaptive smoothing parameter coupled with the momentum term (Algorithm 1). When the smoothing rule is modified to stay away from zero, we provide a global sublinear convergence rate of $\mathcal{O}(1/\varepsilon)$ that matches the optimal worst-case complexity bound for optimizing this class of nonsmooth functions (Theorem 3.3).

(ii) **Locally linear convergence:** When the nonsmooth term $f(x)$ meets a so-called ∞-locally strong convexity condition, we show that Algorithm 1 enjoys a local linear convergence rate (Theorem 3.4). Combined with the global convergence result from the previous section, this implies that an appropriate initial condition for Algorithm 1 ensures a transient optimal sublinear convergence rate followed by an asymptotic linear convergence rate.

(iii) **Multiple nonsmooth terms:** The proposed algorithm allows for multiple nonsmooth terms. Such settings can be computationally challenging as the "prox-friendly" property, a key feature in nonsmooth optimization, is not necessarily additive. Important applications falling into this category include model-free fault diagnosis, nonsmooth model predictive control, sparse regression and sparse semidefinite programming, which are the examples investigated in our numerical section to validate the performance of our proposed algorithm.

To validate the theoretical results of this study, we implement and compare the performance of the proposed algorithm with different existing methods in the literature on various classes of problems including regression with the ℓ_1-norm (the Lasso problem), sparse semidefinite programming (the *MaxCut* problem), and Nuclear norm minimization. Regarding the above contributions, an observation consistently confirmed by these numerical results deserves attention.

An unexpected observation: The smoothness parameter has a direct impact on the stepsize in accelerated algorithms, i.e., the smoother the function, the larger the stepsize. A general rule of thumb is that larger stepsizes lead to faster convergence, and with that in mind, we would expect to

see a slower convergence rate for coupled smoothness parameters. It, however, turns out that this is not the case and the proposed adaptive smoothness parameter has a faster convergence rate of $\mathcal{O}(1/k^2)$ for its transient and asymptotic convergence of $\mathcal{O}(e^{-k})$, as opposed to the existing rate of $\mathcal{O}(1/k)$ [46]. This two-phase behavior with these rates is also evident in the proposed smoothing rule (cf. Lemma 3.2).

Roadmap. The paper is organized as follows: Section 2 reviews Nesterov's smoothing technique for solving nonsmooth minimization. We propose our adaptive-smoothing algorithm in Section 3 and provide the technical proof of theorems in Section 4. Section 5 benchmarks our algorithm in several applications.

Notation. $\mathbb{R}^n$ shows the real vector space, we denote the standard inner product by $\langle\cdot,\cdot\rangle$ and ℓ_p-norm by $\|\cdot\|_p$ (and by $\|\cdot\|$, we mean the Euclidean standard 2-norm). If f is differentiable, $\nabla f(x)$ represents the gradient of f at x. The function f is called L-Lipschitz for some $L>0$ if

$$|f(x)-f(y)|\leq L\|x-y\|,\qquad \forall x,y\in\mathbb{R}^n.$$

The function f is μ-smooth if its gradient is μ-Lipschitz, i.e., $\|\nabla f(x)-\nabla f(y)\|\leq\mu\|x-y\|$. The convex function f is called ρ-locally strongly convex at x^*, if there exists $\varepsilon>0$ so that the function $f(x)-\frac{\rho}{2}\|x\|^2$ is convex over the ball $\mathbb{B}_\varepsilon(x^*)$. We call f ∞-locally strongly convex if f is ρ-locally strongly convex for any $\rho>0$. The Fenchel conjugate of f is defined as $f^*(x):=\sup_y\langle x,y\rangle-f(y)$. The prox operator for function h is defined as

$$\operatorname{prox}_h(x):=\arg\min_u h(u)+\frac{1}{2}\|u-x\|^2. \tag{4}$$

A function is "prox-friendly" if the operator (4) is available (computationally or explicitly). We also denote the gradient mapping of two convex functions f and h by

$$G^f_{\zeta h}(x):=\frac{1}{\zeta}\Big(x-\operatorname{prox}_{\zeta h}\big(x-\zeta\nabla f(x)\big)\Big), \tag{5}$$

where ζ is a positive scalar and has a stepsize interpretation. The gradient mapping is available if f is differentiable and h is prox-friendly.

2. State of the Art on Nonsmooth Optimization

In this section, we review the current state of the art in smoothing nonsmooth functions and discuss a possible challenge that may emerge in dealing with multiple smoothing terms.

2.1. Nesterov's smoothing technique

Consider a possibly nonsmooth convex function f that we assume to be prox-friendly and L_f-Lipschitz continuous. A key object in smoothing techniques is the Moreau envelope defined as

$$f_\mu(x):=\min_y f(y)+\frac{1}{2\mu}\|y-x\|^2. \tag{6}$$

The Moreau envelope is a smooth lower approximation of a function at every point, i.e., $f_\mu(x)\leq f(x)$ for all $x\in\mathbb{R}^n$. By definition, the objective function of the Moreau envelop (6) and the prox operator (4) are closely related, namely, the optimizer of (6) is $\operatorname{prox}_{\mu f}(x)$. The following lemma

indicates several known properties of the smooth approximation (6) that are central for algorithms in this context. For brevity, we skip the proof and refer interested readers to [11] for further details.

Lemma 2.1 (Smoothness regularity)**.** *Let f_μ be defined as in* (6)*. Then, the following holds:*

(i) **Dual reformulation:** $f_\mu(x) = \max_z \left\{ \langle x, z\rangle - f^*(z) - \frac{\mu}{2}\|z\|^2 \right\}$.

(ii) **Uniform bound:** $f_\mu(x) \le f(x) \le f_\mu(x) + \frac{\mu}{2}L_f^2$, *where L_f is the Lipschitz constant of f.*

(iii) **Gradient evaluation:** $\nabla f_\mu(x) = \frac{1}{\mu}\left(x - prox_{\mu f}(x)\right)$.

(iv) **Smoothness:** $f_\mu(\cdot)$ *is* $\frac{1}{\mu}$*-smooth, i.e.,* $\|\nabla f_\mu(x) - \nabla f_\mu(y)\| \le \frac{1}{\mu}\|x - y\|$.

Lemma 2.1 paves the way to a power smoothing technique as follows: The uniform bound (ii) allows us to choose a minimum value for the smoothing parameter so that the smoothed function f_μ remains in a desired ε-vicinity of the original function f. Thanks to the smoothness result in (iv) and under the assumption that f is prox-friendly, one can apply Nesterov's accelerated algorithm using the gradient evaluation (iii) and optimize f_μ. The choice of Nesterov's algorithm (Algorithm 0) is justified by the fact that it is the fastest general convex optimization algorithm for smooth functions [30].

Algorithm 0 Nesterov's accelerated method for $1/\mu$-smooth functions [30]

Input: given initial conditions $y_1 = x_1$, $\beta_0 > 0$, and smoothing constant μ.

$y_{k+1} = x_k - \zeta \nabla f_\mu(x_k), \quad$ with the stepsize $\zeta = \mu$

$x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$

$\beta_{k+1} = \frac{1}{2}\big(1 + \sqrt{1 + 4\beta_k^2}\big), \gamma_k = \dfrac{1 - \beta_k}{\beta_{k+1}}$

This idea is first proposed in [32] in which the smoothness parameter is proposed to be the constant $\mu = 2\varepsilon/L_f^2$ where ε is an apriori desired precision. This yields an overall complexity of $\mathcal{O}(1/\varepsilon)$ in terms of the precision parameter ε. More recently, [46] proposes an adaptive version of the smoothing technique. Our proposed adaptive smoothing technique also follows a similar spirit as in [46], but the key feature is to pair the update rule with the momentum parameter of the accelerated method. Before proceeding with that, we also wish to briefly comment on our motivation for another feature of our proposed algorithm that allows for two nonsmooth terms.

Several studies have been devoted to developing methods for computing the prox-operator (4) of a sum of two (multiple) functions, see the recent work [40, 1, 14, 49] and the references therein. These methods typically provide various assumptions under which we have

$$\operatorname{prox}_{f+h} = \operatorname{prox}_f \circ \operatorname{prox}_h, \tag{7}$$

where "$\circ$" denotes the mapping composition. However, these conditions are still restrictive, and the prox-operator of the sum of two prox-friendly functions may not have a closed-form solution and may be in general computationally demanding (e.g., sparse regression and semi-definite programming). Motivated by this, we develop our algorithm so that it allows for two (multiple) nonsmooth terms, i.e., in (3) both functions f and h are possibly nonsmooth but prox-friendly.

3. Proposed Algorithm and Convergence Analysis

This section presents the main algorithm of this paper and its global and local convergence results.

3.1. Algorithm

The proposed method is given in Algorithm 1. The algorithm follows Nesterov's accelerated method in Algorithm 0 with the difference in the choice of stepsize ζ, which is coupled with the momentum parameter at two consecutive steps β_k and β_{k+1} (Lemma 3.1). This connection is established by looking at the increment of the function as described in one of the preliminary lemmas in Section 4 (Lemma 4.2).

Algorithm 1 Adaptive accelerated smoothing method

Input: given initial conditions $y_1 = x_1$, $\beta_0 > 0$, and the to-be-defined sequence $\big(\mu_k\big)_{k\geq 0}$.

$$\beta_{k+1} = \frac{1+\sqrt{1+4\beta_k^2}}{2}, \quad \gamma_k = \frac{1-\beta_k}{\beta_{k+1}}$$
$$y_{k+1} = x_k - \zeta_k G^{f_{\mu_{k+1}}}_{\zeta_k h}(x_k), \quad \zeta_k = \mu_{k+1}$$
$$x_{k+1} = (1-\gamma_k)y_{k+1} + \gamma_k y_k$$

Our first result quantifies the performance of Algorithm 1 after T iterations for a specific choice of adaptive smoothing sequence $\big(\mu_k\big)_{k\geq 0}$.

Lemma 3.1 (Optimality gap in adaptive smoothing)**.** *Consider the optimization problem in* (3) *and Algorithm 1 with adaptive smoothing variable* $\mu_k = \max\Big\{\mu_{k-1}\Big(\frac{3\beta_k^2}{\beta_{k-1}^2} - 1\Big)^{-1}, c\Big\}$*, for some* $c \geq 0$*. Then, after* T *iterations, we have*

$$F(y_{T+1}) - F^\star \leq \frac{L_f^2 \mu_{T+1}}{2} + \frac{E}{2\beta_T^2 \mu_{T+1}}, \tag{8}$$

where the constant E *is*

$$E = \|u_1\|^2 + \zeta_0 \beta_0^2 \delta_1 + \Big(1 - \frac{\mu_1}{\mu_0}\Big)\frac{3\mu_0}{2} L_f^2 \zeta_0 \beta_0^2,$$

with $\delta_1 = f_{\mu_1}(y_1) + h(y_1) - F^\star$ *and* $u_1 = \beta_1 x_1 - (\beta_1 - 1)y_1 - x^\star$.

Lemma 3.1 serves as a basis for different types of convergence results (global and local) in this study. We note that, the right-hand side of the bound (8) consists of two terms that are dependent on the adaptive smoothing parameter μ_k and can be used to control the optimality gap. The proof of Lemma 3.1 builds on two preparatory lemmas, which we relegate to Section 4 to improve the flow of the paper. Before proceeding with the main results concerning the convergence of Algorithm 1, we first provide a lemma that sheds light on the behavior of the smoothing parameter μ_k that we use in the algorithm.

Lemma 3.2 (Smoothing parameter convergence)**.** *The sequence generated by the first part of* μ_k *in Lemma 3.1, i.e.,* $\mu_{k-1}\big(3\beta_k^2/\beta_{k-1}^2 - 1\big)^{-1}$*, exhibits a transient behavior at the order of* $\mathcal{O}(1/k^2)$ *and an asymptotic linear rate of* $\mathcal{O}(e^{-k})$.

The proof of Lemma 3.2 is rather straightforward and we defer it to Section 4. The convergence behavior of μ_k helps the global and local convergence guarantees of Algorithm 1. In fact, our first result is to show that Algorithm 1 enjoys a global optimal convergence rate of $\mathcal{O}(1/\varepsilon)$ when the smoothing rule of μ_k is uniformly lower bounded away from zero.

Theorem 3.3 (Global sublinear convergence)**.** *Suppose the sequence of the smoothing parameters* $\left(\mu_k\right)_{k\geq 0}$ *is defined as in Lemma 3.1 where* $c = \varepsilon/L_f^2$*. Then, the outcome of Algorithm 1 after* T *iterations satisfies* $F(y_T) - F^* \leq \varepsilon$ *if* $T \geq 2L_f\sqrt{E}/\varepsilon$*.*

Proof of Theorem 3.3. The proof builds on the assertion of Lemma 3.1. From Lemma 3.2, we know that the first phase of μ_k is decreasing with the rate of at least $\mathcal{O}(1/k^2)$. Therefore, the smoothing rule μ_k also converges to ε/L_f^2 with the similar rate of at least $\mathcal{O}(1/k^2)$. It then suffices to determine the minimum number of iterations T so that μ_{T+1} reaches ε/L_f^2. To this end, note that since $\beta_k \geq k/2$ for all k, we have:

$$F(y_{T+1}) - F^\star \leq \frac{\varepsilon}{2} + \frac{2L_f^2 E}{T^2\varepsilon}.$$

Hence, the ε precision is guaranteed if T greater than $2L_f\sqrt{E}/\varepsilon$ which concludes the desired assertion. □

We note that the global sublinear rate provided in Theorem 3.3 is worst-case optimal for the class of nonsmooth functions (3) [32, 33]. Another key message of this study is that under a certain condition over the nonsmoothness, Algorithm 1 can achieve a linear, but local, convergence rate when the smoothing parameter follows the sequences in Lemma 3.2 ($c = 0$ in Lemma 3.1). This two-phase behavior is evident in the smoothing rule as elucidated in Lemma 3.2.

Let us remind that the stepsize in Algorithm 1 is dictated by the smoothing parameter (i.e., $\zeta_k = \mu_{k+1}$). It is surprising to have an exponentially fast diminishing stepsize in optimization algorithms. However, the following result addresses this phenomenon and demonstrates that, under a specific ∞-locally strong convexity condition, such a rate facilitates local linear convergence.

Theorem 3.4 (Local linear convergence)**.** *Consider the optimization problem in* (3) *where the hypotheses of Lemma 3.1 hold with* $c = 0$*. In addition, suppose the nonsmooth term* f *is* ∞*-locally strongly convex at the solution* x^* *of* (3)*, i.e., for any* $\rho > 0$ *the function* $f(x) - \rho\|x\|^2$ *is locally convex in a neighborhood containing* x^**. Then, for any initial condition* $\mu_0 > 0$*, the sequence* $(y_k)_{k\geq 0}$ *generated by Algorithm 1 converges locally linearly to* x^**, i.e., there exist* $\varepsilon > 0$ *and* $\alpha \in (0, 1)$ *such that for all* $x_0 \in \mathbb{B}_\varepsilon(x^*)$ *there exists* k_0 *where for all* $k \geq k_0$*, we have* $F(y_k) - F^* \leq \alpha^{(k-k_0)}(F(y_{k_0}) - F^*)$*.*

Before proceeding with the proof of this theorem, let us note that the key feature leading to local linear convergence is the fact that when f is ∞-locally strongly convex, the condition number of its Hessian $\partial^2 f_\mu$ at its optimizer is uniformly bounded for all sufficiently small μ. This allows the first-order accelerated algorithm to converge locally linearly and independently of the smoothness parameter μ_k. It is also worth noting that the linear convergence rate is uniform for all the initial conditions $x_0 \in \mathbb{B}_\varepsilon(x^*)$. Let us formally explain this as follows.

Proof of Theorem 3.4. Let us define $F_{\mu_k}(x) := f_{\mu_k}(x) + h(x)$. Note that using the basic property of the Moreau envelope in Lemma 2.1(ii), we have the inequality

$$F(y_k) - F(x^*) \leq \frac{\mu_k}{2} L_f^2 + F_{\mu_k}(y_k) - F_{\mu_k}^* \tag{9}$$

where $F_{\mu_k}^* = \min_{x \in \mathbb{R}^n} F_{\mu_k}(x) = F_{\mu_k}(x_{\mu_k}^*)$. Thanks to Lemma 3.2, we know that the term μ_k, and as such the first term of the upper bound (9), converges to zero exponentially fast. Therefore, it suffices to show that $F_{\mu_k}(y_k) - F_{\mu_k}^*$ converges to zero linearly with a rate independently from μ_k. To this end, we recall that the Moreau envelope is locally $\mathcal{C}^2$-differentiable [38]. From the recent work [19], the maximum and minimum Hessian eigenvalues of f_{μ_k} at the optimal point $x_{\mu_k}^*$ can be bounded by

$$\lambda_{\max}\left(\frac{\partial^2}{\partial x^2} f_{\mu_k}(x_{\mu_k}^*)\right) \leq \mu_k^{-1},$$
$$\lambda_{\min}\left(\frac{\partial^2}{\partial x^2} f_{\mu_k}(x_{\mu_k}^*)\right) \geq \left(\rho^{-1} + \mu_k\right)^{-1}.$$

where ρ is the strongly convex parameter of f_{μ_k} (i.e., $f_{\mu_k}(x) - \rho\|x\|^2/2$ is convex). This observation yields the bound on the condition number $\kappa := \lambda_{\max}/\lambda_{\min}$ of the Hessian as $1 \leq \kappa \leq 1 + (\rho\mu_k)^{-1}$. Recall that thanks to the ∞-locally strong convexity feature of f, we know that the parameter ρ can be arbitrarily high as we are closer to the solution x^*. In other words, there exists a sufficiently small $\varepsilon > 0$ where we can choose ρ large enough for all $x \in \mathbb{B}_\varepsilon(x^*)$. Since the (accelerated)-proximal gradient descent methods converge linearly with the rate of $(1 - 1/\kappa)$ [25], we can deduce that for all sufficiently small enough μ_k, Algorithm 1 enjoys linear convergence in a neighborhood of the solution x^*. □

Note that the ∞-local strong convexity of f is essential for having locally linear convergence in Theorem 3.4. Namely, this property enables a linear decrease in the smoothing parameter while the strong convexity parameter, characterized by $\rho(x)$, grows sufficiently fast as we converge to the minimizer. Interestingly, this property holds for many popular nonsmooth terms commonly encountered in applications; see Section 5 for several such examples and also [9] (e.g., Theorem 3.1) for a detailed analysis of the explicit computation of such parameters.

We close this section with two remarks concerning the class of ∞-locally strongly convex functions and the initial value of the smoothing parameter in the proposed algorithm.

Remark 3.5 (∞-locally strong convexity vs. sharpness)**.** *We note that the concept of "∞-locally strong convexity" is closely related to the "sharpness" property* [39]*, but in a slightly weaker manner in the sense that the former is typically used locally, whereas the latter is posed globally. More specifically, a function f is ∞-locally strongly convex with parameter $\rho_\varepsilon^{\infty\text{-}sc}$ (respectively, sharp with the parameter ρ^{sh}) when the following holds:*

$$\infty\text{-locally strong convexity}: \frac{\rho_\varepsilon^{\infty\text{-}sc}}{2}\|x - x^*\|^2 \leq f(x) - f(x^*) \quad \forall\, x \in \mathbb{B}_\varepsilon(x^*), \text{ where } \quad \rho_\varepsilon^{\infty\text{-}sc} \underset{\varepsilon \downarrow 0}{\longrightarrow} \infty,$$

$$\text{Sharpness}: \frac{\rho^{sh}}{2}\|x - x^*\| \leq f(x) - f(x^*) \quad \forall\, x.$$

Remark 3.6 (Avoid switching rule in Algorithm 1)**.** *In view of Theorem 3.4, local linear convergence is achievable if we are sufficiently close to the global optimal point x^*. One can use Algorithm 1 with the lower-bounded smoothing rule (Theorem 3.3) to converge to an arbitrarily close neighborhood*

of the optimal solution wherein linear convergence is guaranteed, and then continue with $c = 0$ in Algorithm 1 to converge to the desired solution with the faster rate anticipated in Theorem 3.4. To avoid this switching mechanism, a promising idea is to choose the initial value μ_0 in a way that ensures a sufficient decrease in the objective function in the first phase behavior of μ_k, and as such, guarantees reaching the desired neighborhood $\mathbb{B}_\varepsilon(x^*)$ where the linear convergence is achievable. Note that the optimal value of μ_0 typically depends on the initial error $\|x^* - x_0\|$, as we elaborate further in the next section.

3.2. **Further discussion, limitation, and future direction**

In this part, we provide additional information and insights concerning the proposed smoothing technique, its limitations, and possible future directions.

Further insights behind the adaptive smoothing rule: The main motivation of the adaptive smoothness parameter is to exploit the possibility of having a larger smoothness parameter μ_k (and as such, optimizing smoother approximate function f_μ), which leads to larger stepsizes and potentially faster convergence rate. However, any stepsize larger than ε/L_f^2 can increase the Lyapunov function of Nesterov's accelerated algorithm, which is the key driving force behind our algorithm. This violation turns out to be dependent on the two subsequent algorithm momentum β_k and β_{k+1} and the two subsequent smoothness parameters μ_k and μ_{k+1}. This observation indeed leads to the smoothing rule in Lemma 3.1. Furthermore, as explained in Lemma 3.2, the smoothing parameter μ_k has an initial decreasing rate of $\mathcal{O}(1/k^2)$, but asymptotically it converges to zero with an exponential rate. This behavior can explain the locally linear convergence of Algorithm 1 discussed in Theorem 3.4.

Extension to the smoothing rule: It can be shown that the smoothing parameter μ_k in Lemma 3.1 can be generalized to $\mu_k = \max\Big\{b\mu_{k-1}\big(\frac{b(a-1)+a}{a-1}\frac{\beta_k^2}{\beta_{k-1}^2} - 1\big)^{-1}, c\Big\}$, where $a > 1$ and $b > 0$ are hyperparameters that control the behavior of the smoothing parameter μ_k. These parameters can be selected to prevent Algorithm 1 from switching, thereby enabling locally linear convergence, as discussed in Remark 3.6. The optimal tuning of these parameters remains unclear to the authors and is a promising direction for future research.

Relation to prior works and the existing performance guarantees: We note that our apriori theoretical results in Theorem 3.3 match the state of the art [32] for the particular choice of algorithm parameters $(a, b) = (1, 0)$, but do not improve the global performance. We wish also to note that this is not a rare precedent in optimization algorithm literature that a new algorithm numerically performs better than its formal apriori guarantees. For instance, the well-known FISTA algorithm proposed by [7] has the same convergence bounds as in Nesterov's method [32] when the latter is restricted to the proximal setting.

Limitation and future direction: The primary limitation of the proposed method lies in the selection of the initial condition μ_0 (see Remark 3.6). When μ_0 is too small and $c = 0$, there is a risk that the algorithm fails to reach the region where linear convergence is attainable. This can result in stagnation of iterations due to the diminishing but still summable rate of decrease in the sequence of stepsizes $\zeta_k = \mu_{k+1}$. Conversely, if μ_0 is excessively large, the algorithm precision is compromised, as it takes longer to reach the linear convergence region. An optimal value for μ_0 may depend on

the initial error $\|x^* - x_0\|$ and the local characteristics of the functions involved in (3) at x^* [26]. Investigating and analyzing these features are promising avenues for future research.

Another future direction is concerned with the relation between the stepsize and smoothing parameters. In this work, the smoothing parameter dictates the stepsize. However, if we untangle this dependency, an adaptive stepsize may support the acceleration of the algorithms while an adaptive smoothing parameter enhances the precision. This adaptive stepsize-adaptive smoothing rule can be a promising research direction.

4. Technical proofs

In this section, we provide the theoretical proof for Section 3 and additional material supporting the technical as well as the numerical investigation of the paper.

4.1. **Details of the Theoretical Analysis**

Our proof for Lemma 3.1 relies on a Lyapunov argument. To this end, we first proceed with two preliminary lemmas.

Lemma 4.1 (Gradient mapping)**.** *Let $G^f_{\zeta h}(x)$ be the gradient mapping defined in* (5) *where ζ is a positive scalar, $f(x)$ is smooth, and $h(x)$ is prox-friendly. Then, we have*

$$h\big(x - \zeta G^f_{\zeta h}(x)\big) \leq h(y) - \big\langle G^f_{\zeta h}(x) - \nabla f(x), y - \big(x - \zeta G^f_{\zeta h}(x)\big)\big\rangle, \quad \forall x, y \in \mathbb{R}^d.$$

Proof of Lemma 4.1. Defining $u = \mathrm{prox}_{\zeta h}(w)$, we can write

$$u = \mathrm{prox}_{\zeta h}(w) \Leftrightarrow u = \arg\min_u h(u) + \frac{1}{2\zeta}||u - w||^2 \Leftrightarrow 0 \in \partial h(u) + \frac{1}{\zeta}(u - w) \Leftrightarrow w - u \in \zeta \partial h(u)$$

By defining $u := x - \zeta G^f_{\zeta h}(x)$ and $w := x - \zeta \nabla f(x)$, we have

$$\underbrace{x - \zeta G^f_{\zeta h}(x)}_{u} = x - \zeta \frac{1}{\zeta}(x - \mathrm{prox}_{\zeta h}(x - \zeta \nabla f(x))) =$$

$$\mathrm{prox}_{\zeta h}(\underbrace{x - \zeta \nabla f(x)}_{w}) \Rightarrow G^f_{\zeta h}(x) - \nabla f(x) \in \partial h(x - \zeta G^f_{\zeta h}(x)).$$

Using the convexity of h, we can write

$$h(x - \zeta G^f_{\zeta h}(x)) \leq h(y) - \langle G^f_{\zeta h}(x) - \nabla f(x), y - (x - \zeta G^f_{\zeta h}(x))\rangle.$$

□

Lemma 4.1 is an inherent property of the gradient mapping and essentially represents a convex inequality that is particularly helpful to control the increment of the original function F in (3).

Lemma 4.2 (Increment bound)**.** *Suppose function f is prox-friendly and f_μ is the smooth approximation* (6)*. Considering the update $y_{k+1} = x_k - \zeta_k G^{f_{\mu_{k+1}}}_{\zeta_k h}(x_k)$, for any $z \in \mathbb{R}^d$ we have*

$$f_{\mu_{k+1}}(y_{k+1}) - f(z) + h(y_{k+1}) - h(z) \leq -\frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}\big\langle y_{k+1} - x_k, x_k - z\big\rangle \qquad (10)$$

Proof of Lemma 4.2. By using the uniform boundedness of f_μ and the definition of convexity, we have

$$\begin{aligned}
f_{\mu_{k+1}}(y_{k+1}) - f(z) + h(y_{k+1}) - h(z) &\overset{(ii)}{\leq} f_{\mu_{k+1}}(x_k - \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k)) - f_{\mu_{k+1}}(z) \\
&+ h(x_k - \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k)) - h(z) \leq f_{\mu_{k+1}}(x_k - \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k)) - f_{\mu_{k+1}}(x_k) \\
&+ \langle \nabla f_{\mu_{k+1}}(x_k), x_k - z\rangle - \underbrace{\langle G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k) - \nabla f_{\mu_{k+1}}(x_k), z - (x_k - \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k))\rangle}_{\textbf{Lemma } 4.1} \\
&\leq \langle \nabla f_{\mu_{k+1}}(x_k), x_k - \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k) - x_k\rangle + \frac{\zeta_k^2}{2\mu_{k+1}}\|G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k)\|^2 + \langle G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k), y_{k+1} - z\rangle \\
&+ \langle \nabla f_{\mu_{k+1}}(x_k), \zeta_k G_{\zeta_k h}^{f_{\mu_{k+1}}}(x_k)\rangle \qquad (11)
\end{aligned}$$

The last inequality is valid as a result of the smoothness property of the function $f_{\mu_{k+1}}$ (iv). By adding and subtracting $\frac{1}{\zeta_k}\langle y_{k+1} - x_k, x_k\rangle$ in (11), we obtain

$$\begin{aligned}
f_{\mu_{k+1}}(y_{k+1}) - f(z) + h(y_{k+1}) - h(z) \leq &\underbrace{\langle y_{k+1} - x_k, \frac{1}{2\mu_{k+1}}\|y_{k+1} - x_k\|^2 - \frac{1}{2\mu_{k+1}}\|y_{k+1} - x_k\|^2\rangle}_{=0} \\
&- \frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}\langle y_{k+1} - x_k, x_k - z\rangle.
\end{aligned}$$

□

Lemma 4.2 plays a key role in the proof of Lemma 3.1. The increment bound (10) allows for the inclusion of a momentum term that emerges in acceleration. We are now in a position to prove Lemma 3.1.

Proof of Lemma 3.1. Here, we present the proof for the general selection of the smoothing parameter, as discussed in Section 3.2. In the first step, by applying (10) and Lemma 2.6 in [11] for two cases of $z = y_k$ and $z = x^*$ to arrive at

$$\begin{aligned}
f_{\mu_{k+1}}(y_{k+1}) - f_{\mu_k}(y_k) - \frac{(\mu_k - \mu_{k+1})}{2}L_f^2 + h(y_{k+1}) - h(y_k) &\overset{\text{(ii)}}{\leq} f_{\mu_{k+1}}(y_{k+1}) - f(y_k) \\
+ h(y_{k+1}) - h(y_k) \leq -\frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}\langle y_{k+1} - x_k, x_k - y_k\rangle. \qquad &(12a) \\
f_{\mu_{k+1}}(y_{k+1}) - f^* + h(y_{k+1}) - h^* \leq -\frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}\langle y_{k+1} - x_k, x_k - x^*\rangle. \qquad &(12b)
\end{aligned}$$

Let us define $\delta_k := f_{\mu_k}(y_k) + h(y_k) - f^* - h^*$. Then, multiplying (12a) by $(\beta_k - 1)$ and adding the two sides of the inequality to (12b) yields

$$\begin{aligned}
&\beta_k\delta_{k+1} - (\beta_k - 1)\delta_k - \frac{(\mu_k - \mu_{k+1})}{2}L_f^2(\beta_k - 1) \leq \\
&-\frac{\beta_k}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}\langle y_{k+1} - x_k, \beta_k x_k - (\beta_k - 1)y_k - x^*\rangle.
\end{aligned}$$

Multiplying the above inequality by $\zeta_k\beta_k$ and considering $\beta_{k-1}^2 := \beta_k^2 - \beta_k$ and $\zeta_k \leq \zeta_{k-1}$, we have

$$\begin{aligned}
\zeta_k\beta_k^2\delta_{k+1} - \zeta_{k-1}\beta_{k-1}^2\delta_k - \frac{(\mu_k - \mu_{k+1})}{2}L_f^2\zeta_k\beta_{k-1}^2 \leq -\frac{1}{2}\Big(&\|\beta_k(y_{k+1} - x_k)\|^2 \\
&+ 2\beta_k\langle y_{k+1} - x_k, \beta_k x_k - (\beta_k - 1)y_k - x^*\rangle\Big) \qquad (13)
\end{aligned}$$

The right-hand side of (13) can be equivalently written as

$$\|\beta_k(y_{k+1}-x_k)\|^2+2\beta_k\langle y_{k+1}-x_k,\beta_k x_k-(\beta_k-1)y_k-x^*\rangle=$$
$$\|\beta_k y_{k+1}-(\beta_k-1)y_k-x^*\|^2-\|\beta_k x_k-(\beta_k-1)y_k-x^*\|^2. \tag{14}$$

By substituting (14) into (13) and by rearranging the inequality, we have

$$\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-\frac{(\mu_k-\mu_{k+1})}{2}L_f^2\zeta_k\beta_{k-1}^2\le-\frac{1}{2}\Big(\|\beta_k y_{k+1}-(\beta_k-1)y_k-x^*\|^2 - \|\beta_k x_k-(\beta_k-1)y_k-x^*\|^2\Big) \tag{15}$$

Using the update rule of x_{k+1} on the right-hand side of (15) reduces to

$$\beta_k y_{k+1}-(\beta_k-1)y_k-x^*=\beta_{k+1}x_{k+1}-(\beta_{k+1}-1)y_{k+1}-x^* \tag{16}$$

which is equivalent to

$$x_{k+1}=\frac{(-1+\beta_k+\beta_{k+1})}{\beta_{k+1}}y_{k+1}+\frac{1-\beta_k}{\beta_{k+1}}y_k,$$

By combining (15) and (16) with $u_k=\beta_k x_k-(\beta_k-1)y_k-x^*$, we obtain

$$\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-\frac{(\mu_k-\mu_{k+1})}{2}L_f^2\zeta_k\beta_{k-1}^2\le\frac{1}{2}\Big(\|u_k\|^2-\|u_{k+1}\|^2\Big), \tag{17}$$

By defining $\omega_k:=\mu_{k+1}/\mu_k$, we can rewrite (17) as

$$\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-\frac{\mu_k(1-\omega_k)}{2}L_f^2\zeta_k\beta_{k-1}^2\le\frac{1}{2}\Big(\|u_k\|^2-\|u_{k+1}\|^2\Big), \tag{18}$$

where $0<\omega_k\le 1$. Now, we consider two cases. First, we assume that μ_k strictly decreases in each iteration. To enforce μ_k being strictly decreasing we impose the monotonicity condition $\mu_k<\mu_{k-1}$. Then,

$$-a\frac{\mu_{k-1}}{2}L_f^2\zeta_{k-1}\beta_{k-1}^2+(a-1)\frac{\mu_k}{2}L_f^2\zeta_k\beta_{k-1}^2<-a\frac{\mu_k}{2}L_f^2\zeta_k\beta_{k-1}^2+(a-1)\frac{\mu_k}{2}L_f^2\zeta_k\beta_{k-1}^2=-\frac{\mu_k}{2}L_f^2\zeta_k\beta_{k-1}^2 \tag{19}$$

The inequality (19) inspire us to define μ_k as

$$\mu_k=\left(\frac{b(a-1)+a}{a-1}\frac{\beta_k^2}{\beta_{k-1}^2}\mu_k-b\mu_{k-1}\right)\Rightarrow\ \mu_k=\frac{b\mu_{k-1}}{\dfrac{b(a-1)+a}{a-1}\dfrac{\beta_k^2}{\beta_{k-1}^2}-1}. \tag{20}$$

Using the definition of μ_k and its decreasing rate (Lemma 3.2), it can be easily shown that $1-\omega_k\le 1-\omega_{k-1}$. Then, by substituting μ_k in the left-hand side of (19) and using (18), we arrive at a Lyapunov-like inequality

$$\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-(1-\omega_{k-1})(ab-b+a)\frac{\mu_{k-1}}{2}L_f^2\zeta_{k-1}\beta_{k-1}^2 + (1-\omega_k)(ab-b+a)\frac{\mu_k}{2}L_f^2\zeta_k\beta_k^2\le\frac{1}{2}\Big(\|u_k\|^2-\|u_{k+1}\|^2\Big) \tag{21}$$

By summing up the inequalities in (21) from $k=1$ to $k=T$, one obtains

$$\zeta_T\beta_T^2\delta_{T+1}-\zeta_0\beta_0^2\delta_1-(1-\omega_0)(ab-b+a)\frac{\mu_0}{2}L_f^2\zeta_0\beta_0^2\le\frac{1}{2}\|u_1\|^2-\frac{1}{2}\|u_{T+1}\|^2\le\frac{1}{2}||u_1||^2, \tag{22}$$

which implies

$$\delta_{T+1}\le\frac{E}{2\zeta_T\beta_T^2},\quad E=\|u_1\|^2+\zeta_0\beta_0^2\delta_1+(1-\omega_0)(ab-b+a)\frac{\mu_0}{2}L_f^2\zeta_0\beta_0^2. \tag{23}$$

Second, we assume that the smoothing parameter μ_k strictly decreases by (20) until it reaches to some a-priori value, and then we use the fixed μ_k afterward. By summing up the inequalities in (21) from $k = 1$ to $k = K_\varepsilon$ (the iteration index that we fix μ_k), one obtains

$$\zeta_{K_\varepsilon}\beta_{K_\varepsilon}^2\delta_{K_\varepsilon+1} - \zeta_0\beta_0^2\delta_1 - (1-\omega_0)(ab-b+a)\frac{\mu_0}{2}L_f^2\zeta_0\beta_0^2 \le \frac{1}{2}\|u_1\|^2 - \frac{1}{2}\|u_{K_\varepsilon+1}\|^2, \tag{24}$$

Now, by fixing μ_k, we know that $\omega_k = 1$ for all $k \ge K_\varepsilon + 1$ and then by summing up the inequalities in (18) from $k = K_\varepsilon + 1$ to $k = T$, one can obtain

$$\zeta_T\beta_T^2\delta_{T+1} - \zeta_{K_\varepsilon}\beta_{K_\varepsilon}^2\delta_{K_\varepsilon+1} \le \frac{1}{2}\|u_{K_\varepsilon+1}\|^2 - \frac{1}{2}\|u_{T+1}\|^2, \tag{25}$$

Summing (24) and (25) yields the same inequality as (23). Finally, by the definition of δ_{T+1}, (ii), and (23), we then have

$$F(y_{T+1}) - F^* \le \frac{L_f^2}{2}\mu_{T+1} + \frac{E}{2\zeta_T\beta_T^2}. \tag{26}$$

The result in Lemma 3.1 can be easily derived by considering $a = 2$ and $b = 1$. □

Proof of Lemma 3.2. We present the proof for the general selection of the smoothing parameter, as discussed in Section 3.2. Defining $\alpha_k := \big[\big(\frac{\beta_{k+1}}{\beta_k}\big)^2 + \big(\frac{a}{b(a-1)}\big)\big(\frac{\beta_{k+1}}{\beta_k}\big)^2 - \frac{1}{b}\big]^{-1}$, the first part of the smoothing parameter μ_k can be explicitly described as $\mu_{k+1} = \prod_{i\le k}\alpha_i$. To show $\mu_k \le \frac{C_0}{k^2}$ for some constant C_0, it suffices to show $\alpha_i \le \mathrm{e}^{-2/i}$ for all $i \ge k_0$ where k_0 is a constant. This claim relies on the fact that this inequality implies the following:

$$\mu_{k+1} = \prod_{i\le k}\alpha_i \le C_0\mathrm{e}^{\sum_{i\le k} -2/i} \le C_0\mathrm{e}^{-2\log(k)} \le \frac{C_0}{k^2},$$

where $C_0 = \prod_{i<k_0}\alpha_i$. To complete the proof, we show $\alpha_i \le \mathrm{e}^{-2/i}$ for all sufficiently large i by the following argument:
If $k_0 = 2\big(\log(1+\frac{1}{b(a-1)})\big)^{-1} \le i$, then $\mathrm{e}^{2/i} \le 1 + \frac{1}{b(a-1)}$, which consequently implies:

$$\mathrm{e}^{2/i} \le \big(\frac{\beta_i}{\beta_{i-1}}\big)^2 + \frac{1}{b(a-1)}\Big(a\big(\frac{\beta_i}{\beta_{i-1}}\big)^2 - a + 1\Big) = \alpha_i^{-1}.$$

In the last argument, we use the increasing property of $\beta_i \ge \beta_{i-1} > 0$ (i.e., $\big(\frac{\beta_i}{\beta_{i-1}}\big)^2 > 1$), which is a property of the momentum update (line 2 in Algorithm 1). Finally, to show the linear convergence, we show that α_k stays uniformly away from 1 as k increases, ensuring a linear rate for large k. To this end, note that by the definition, we have

$$\begin{aligned}\alpha_k &:= \Big[\big(\frac{\beta_{k+1}}{\beta_k}\big)^2 + \Big(\frac{a}{b(a-1)}\Big)\big(\frac{\beta_{k+1}}{\beta_k}\big)^2 - \frac{1}{b}\Big]^{-1} = \Big[\big(\frac{\beta_{k+1}}{\beta_k}\big)^2\Big(1 + \frac{a}{b(a-1)}\Big) - \frac{1}{b}\Big]^{-1} \\ &\le \Big[\Big(1 + \frac{a}{b(a-1)}\Big) - \frac{1}{b}\Big]^{-1} = \Big[1 + \frac{1}{b(a-1)}\Big]^{-1} = \frac{b(a-1)}{b(a-1)+1} < 1,\end{aligned}$$

where the second line inequality follows from the monotonicity of the momentum sequence $\beta_{k+1} \ge \beta_k$ and the fact that the coefficient $1 + a/b(a-1) > 0$ (the latter is also ensured by the feasible range of $a > 1$ and $b > 0$). This concludes that α_k is uniformly smaller than 1 by the positive margin of $(b(a-1)+1)^{-1}$.
For validation, Figure 1 depicts the sequence in the first part of smoothing parameter μ_k, as defined in Lemma 3.2. As illustrated in the figure, the smoothing parameter initially enjoys a convergence rate of $\mathcal{O}(1/k^2)$ but in the second phase, adopts an asymptotically linearly convergence rate. □

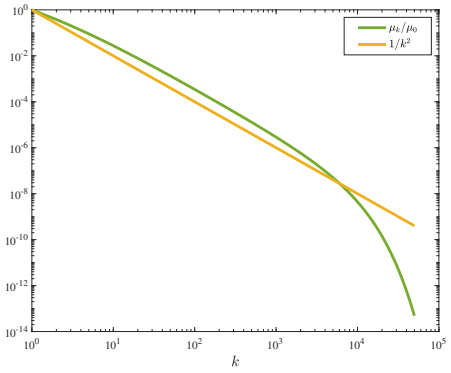


FIGURE 1. Convergence rate of μ_k.

5. Numerical experiments

We demonstrate the performance of Algorithm 1 (with $c = 0$) on four popular classes of problems studied in the machine learning and control literature: (1) regression problems for both combination of the ℓ_1 and ℓ_2 norms borrowed from [48, 45], (2) the *MaxCut* problem belonging to the class of semidefinite programming from [41], (3) the Nuclear norm minimization problem and its application in model-free fault diagnosis from [3, 35], and (4) ℓ_1-regularized model predictive control from [2]. To evaluate our performance, we compare our proposed algorithm (Algorithm (1)) with the following methods from the literature: (i) Sub-gradient descent (SGD) [28], the standard optimization algorithm, (ii) Chambolle-Pock (CP) [12], the state-of-the-art method for sparse regression, or the stochastic smoothing technique (stoch-smooth) [17], a relatively recent method for semidefintie programming, (iii) Nesterov's smoothing (Nes-smooth) [32], and (iv) Tran-Dinh (TD) [46], that is the closest in spirit to our proposed method.

Note that in light of Remark 3.5, we use sharp convex functions in this study, and the simulations include examples that satisfy the sharpness property. An example of an ∞-locally strongly convex function is $\|\cdot\|_1$. Using the definition of ∞-locally strongly convex functions and the sharpness properties of $\|\cdot\|_1$, there exists a neighborhood of x^* where, for any $\rho > 0$, the function is ρ-strongly convex [39, 26, 9]. Several important prox-friendly functions fall into the category as well, including: $\|x\|_\infty$ and the Nuclear norm of matrices (see the tables "Prox Calculus Rules" and "Prox Computations" on pages 448 and 449 in [5]). Additional examples, formed as combinations of such functions, are also discussed in the literature, for instance, [27] considers the total variation, ℓ_∞-norm, $\ell_1 - \ell_2$-norm, and the Nuclear norm. Indeed, the widespread applicability of these functions across various domains motivated us to include several numerical examples in this section. We would also like to highlight that the class of problems studied in this work encompasses several key applications in image processing, image reconstruction, system identification, and control. Specifically, in [18], the authors employ $\ell_2 - \ell_1$ and $\ell_1 - \ell_1$ optimization for image restoration. In [12], the TV-ℓ_1 model is used for image denoising, while in [47], the $\ell_1 - \ell_1$ problem is applied to dictionary learning. Lastly, the ℓ_1-Nuclear norm problem is employed in data-driven fault diagnosis control in [35].

5.1. **Regression problems**

We consider the ℓ_i-ℓ_1-regularized LASSO problem

$$F^* := \min_{x\in\mathbb{R}^n} \mathcal{R}(x) + \eta\|x\|_1, \tag{27}$$

where $\mathcal{R}(x) = \|Bx - b\|_i,\ i = \{1, 2\}$, is a regularized function. the parameters B and b are given and $\eta \geq 0$ is a weighting coefficient. Functions f and h in (3) can be written as $f(x) := \mathcal{R}(x)$ and $h(x) := \eta\|x\|_1$, respectively. For the function f, we use the smoothing approximation (6).

The test data is generated as follows: Matrix $B \in \mathbb{R}^{n\times n}$ is generated randomly using the standard Gaussian distribution $\mathcal{N}(0, 1)$. For simulation, we consider two cases as in [46]. In the first case, we use uncorrelated data, while in the second case (reported in the Appendix A), we generate B with 50% correlated columns as $B(:, j + 1) = 0.5B(:, j) + \text{randn}(:)$. The observed measurement vector b is generated as $b := Bx^\natural + \mathcal{N}(0, 0.05)$, where $x^\natural$ is generated randomly using $\mathcal{N}(0, 1)$. For the simulation, we set $n = 100$ and the coefficient parameter η is chosen as suggested in [46, 10]. All algorithms are initialized at the same point chosen randomly. In Nes-smooth method the stepsize is $2 \cdot 10^{-3}/L_f^2\|B\|^2$ for achieving desired error $\varepsilon = 10^{-3}$ and the stepsizes in primal and dual updates in CP are $1/\|B\|$. In TD and the proposed method, the stepsize is $\zeta = \mu_k/\|B\|^2$. The convergence speed is significantly affected by the initial condition of μ in TD, and following the suggestion by [46], we set $\mu_0 = \mu^* = \|B\| \cdot \|x_0 - x^*\|/\sqrt{3L_f^2}$ for the TD algorithm.

• ℓ_1-ℓ_1**-regularized LASSO.** Here, $i = 1$ for $\mathcal{R}(x)$ in (27). Then, f can be written as $f(x) := \|Bx - b\|_1 = \max_y\big\{\langle B^\top y, x\rangle - \langle b, y\rangle : \|y\|_\infty \leq 1\big\}$. Hence, we can smooth f using the quadratic prox-function to obtain $f_\mu(x) := \max_y \Big\{\langle B^\top y, x\rangle - \langle b, y\rangle - \frac{\mu}{2}\|y\|^2 : y \in \mathbb{B}_\infty\Big\}$. Using the analytical solution, one can observe that $\nabla f_\mu(x) = \text{proj}_{\mathbb{B}_\infty}\left(\mu^{-1}(Bx - b)\right)$ and $L_f^2 = n$.

• ℓ_2-ℓ_1**-regularized LASSO.** We consider $i = 2$ for $\mathcal{R}(x)$ in (27). Functions f can be written as $f(x) := \|Bx - b\|_2 = \max_y\big\{\langle B^\top y, x\rangle - \langle b, y\rangle : \|y\|_2 \leq 1\big\}$. Hence, we similarly have $f_\mu(x) := \max_y \Big\{\langle B^\top y, x\rangle - \langle b, y\rangle - \frac{\mu}{2}\|y\|^2 : y \in \mathbb{B}_2\Big\}$. In this case, we can show that $\nabla f_\mu(x) = \text{proj}_{\mathbb{B}_2}\left(\mu^{-1}(Bx - b)\right)$, and $L_f^2 = 1$.

Figures 2a and 2d report the results for (27) with $i = 1, 2$ where all algorithms are initialized at the same point chosen randomly. As shown in Figures 2a and 2d, the proposed algorithm outperforms other methods.

5.2. ***MaxCut*** **(semidefinite programming)**

An important application of the first-order smoothing technique is to solve semidefinite programming. This class can essentially be reduced to minimizing the maximum eigenvalue of a matrix as follows:

$$\min_{X\in\mathbb{S}^n} \lambda_{\mathbf{max}}(X) \tag{28}$$

It is worth noting that the primal form of all semidefinite programs with a fixed trace can be rewritten as (28). As shown in [33], the smooth approximation of $\lambda_{\mathbf{max}}(X)$ can be written as $f_\mu(X) = \mu \log\left(\sum_{i=1}^n \exp(\lambda_i(X)/\mu)\right)$, which is convex and twice differentiable with a gradient

$$\nabla f_\mu(X) =$$

$$\left(\sum_{i=1}^{n}\exp(\lambda_i(X)/\mu)\right)^{-1}\sum_{i=1}^{n}\exp(\lambda_i(X)/\mu)q_iq_i^{\top},$$

where q_i is the i^{th} column of the unitary matrix Q in the eigen-decomposition $Q\Sigma Q^T$ of X and $\lambda_1 \geq ... \geq \lambda_n$ are the eigenvalues of X. In addition, $f_\mu(X)$ fulfills the uniform boundedness inequality (ii) with $L_f^2 = 2\log n$ and the smoothness parameter $1/\mu$ (cf. (iv) in Lemma 2.1). The numerical performance of the smoothing technique is evaluated for the *MaxCut* relaxation which is written in the primal form of the semidefinite program as follows:

$$\max_{X\in\mathbb{S}^n}\mathbf{Tr}(CX) \qquad \text{s.t.} \quad \text{diag}(X)=1, \quad X\succcurlyeq 0 \tag{29}$$

Matrix C is generated using the Wishart distribution with $C = G^\top G/\|G\|_2^2$, where G is a standard Gaussian matrix. Here, we consider a regularized dual form of (29) as suggested in [21]

$$\min_{y\in\mathbb{R}^n}\lambda_{\mathbf{max}}(C+\text{diag}(y)) - \langle \mathbf{1}, y\rangle + \eta\mathcal{R}(y), \tag{30}$$

where η is a regularization parameter. Figure 2b and 2e show the simulation results for minimizing problem (30) with $n = 100$, 2000 iterations, and two different choices of $\mathcal{R}(y)$. In the Nes-smooth method, the stepsize is 10^{-3} for achieving a desired error $\varepsilon = 10^{-3}$. The parameters required in the stoch-smooth method are chosen according to [17]. The initial condition for μ in the TD method is $\mu_0 = \mu^* = \|x_0 - x^*\|/\sqrt{6\log n}$. Additional simulations for different η are reported in the Appendix A.

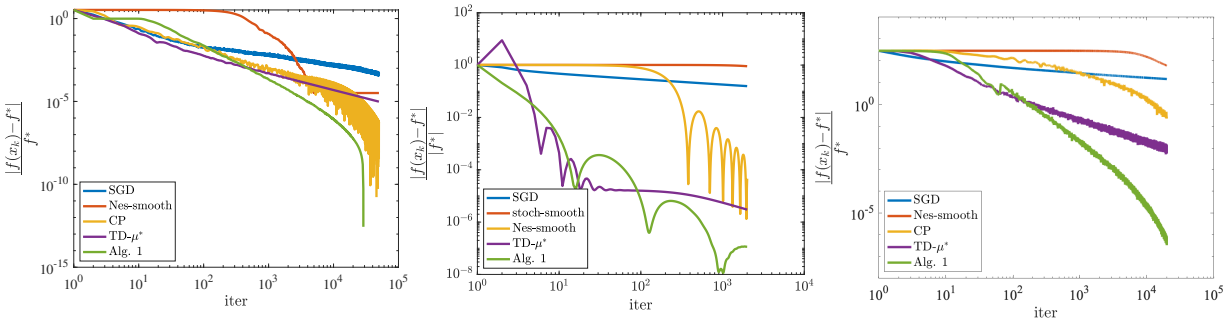
5.3. **Nuclear norm minimization (Model-free Fault Diagnosis)**

The Nuclear norm regularization is an important problem emerging in several applications including matrix completion [3, 23], compressed sensing [43], principal component analysis [20], system identification and fault diagnosis [44]. In this work, we consider model-free fault diagnosis as in [35], where the authors proposed a model-free, data-driven fault diagnosis approach that aims to identify the system and diagnose faults simultaneously, thereby eliminating the need for an extensive identification phase prior to fault detection. The proposed method reformulates the problem as a convex optimization problem involving the summation of nonsmooth terms. To enhance computational efficiency, proximal and splitting-type algorithms (similar to Nes-smooth, CP, and TD methods) are employed for online implementation. The problem formulation is expressed as follows (see equation (5) in [35]):

$$F^* := \min_{x,X_L,X_S}\frac{1}{2}\|y - Hx\|^2 + \tau\|X_L\|_* + \lambda\|X_S\|_1. \tag{31}$$

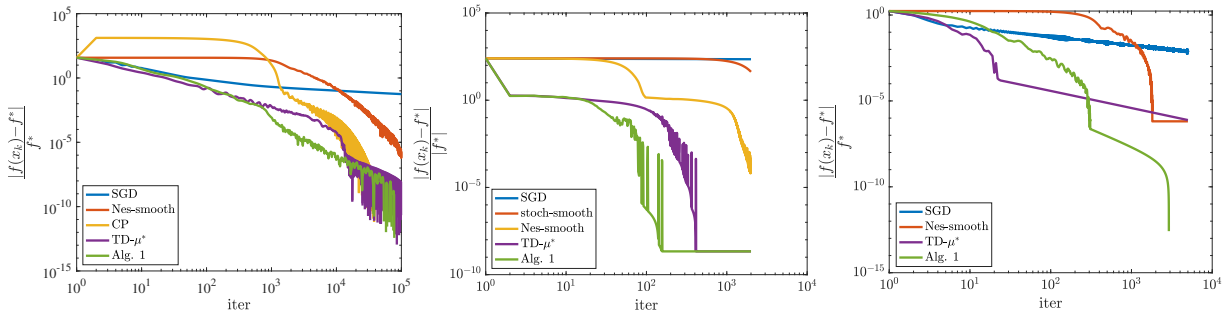
Here, we consider the same buck converter example in [35] with same parameters and solve the problem using different methods. Here, $X_L \in \mathbb{R}^{m_1\times m_2}$ and $X_S \in \mathbb{R}^{n_1\times n_2}$ are linear maps of x that select and rearrange its elements into a low-rank matrix and a sparse vector, respectively. H is a Toeplitz matrix corresponding to the input, output, and fault signals, y denotes the output data, and $\tau \geq 0$ and $\lambda \geq 0$ are regularization parameters. The Nuclear norm of a matrix X, denoted $\|X\|_*$, is the sum of its singular values, equivalently the ℓ_1-norm of the singular values. Both nonsmooth terms in (31) are prox-friendly. Therefore, functions f and h in (3) can be expressed as $f(x) = \frac{1}{2}\|y - Hx\|^2 + \lambda\|X_S\|_1$ and $h(x) = \tau\|X_L\|_*$ and can be smoothed using (6).
Figures 2c depict the results of optimizing (31). All algorithms are initialized at the same randomly

chosen point and run for 20,000 iterations. For the Nes-smooth method, the stepsize is set to $2 \cdot 10^{-3}/(\|H\|^2 + \lambda)$ to achieve a precision level of $\varepsilon = 10^{-3}$. In the primal and dual updates of CP, the stepsizes are set to $1/(\|H\|^2 + \lambda)$, while in TD and the proposed method, the stepsize is determined based on $\mu_k/(\|H\|^2 + \lambda)$.



(A) ℓ_1-ℓ_1-regularized LASSO (27). (B) $\mathcal{R}(y) = \|y\|_2^2$ and $\eta = 0.05$ (30).

(C) Fault diagnosis (31).



(D) ℓ_2-ℓ_1-regularized LASSO (27). (E) $\mathcal{R}(y) = \|y\|_1$ and $\eta = 1$ (30). (F) ℓ_1-Regularized MPC (32).

FIGURE 2. Numerical results for four problem classes. The first column shows regression problems with uncorrelated data (Subsection 5.1); the second, the *MaxCut* problem with different choices of $\mathcal{R}(y)$ (Subsection 5.2); and the third, model-free fault diagnosis with Nuclear norm minimization (Subsection 5.3) and ℓ_1-Regularized MPC (Subsection 5.4).

5.4. ℓ_1-regularized model predictive control

The authors in [2] consider the ℓ_1-regularized MPC problem with horizon H to reduce actuator activity in the quadruple tank system. They use a linearized state-space model and solve the following MPC problem [24, Eq. (1)]:

$$\begin{aligned} \min_{x,u} \quad & J(x,u) := \sum_{i=0}^{H-1} \Big(\|x_i - x_{\mathrm{ref},i}\|_Q^2 + \lambda \|\Delta u_i\|_1 \Big) + \|x_H - x_{\mathrm{ref},H}\|_Q^2 \\ \text{s.t.} \quad & x_{i+1} = \mathbf{A}x_i + \mathbf{B}u_i, \quad \forall i \in \{0,\dots,H-1\}, \\ & u_{\min} \le u_i \le u_{\max}, \qquad \forall i \in \{0,\dots,H-1\}, \\ & \Delta u_i = u_i - u_{i-1}, \quad \forall i \in \{0,\dots,H-1\}. \end{aligned} \tag{32}$$

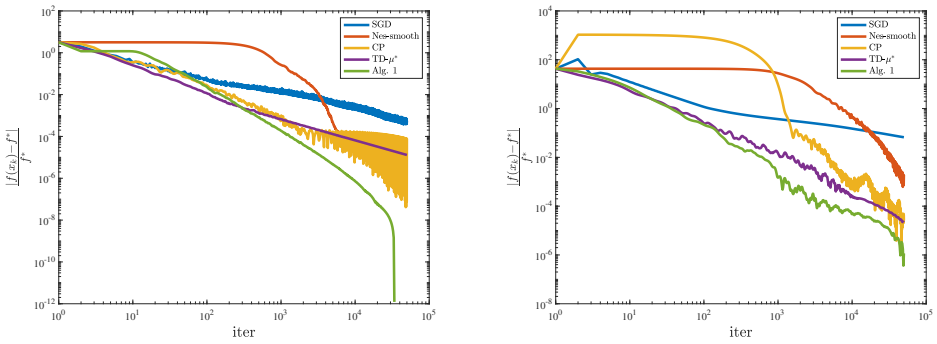
We use the same parameters as in [2] for the system dynamics and solve problem (32) using different methods. The objective function can be represented by $f = J(x, u)$ and $h = i(\chi)$ in (3), where χ is the feasible set of problem (32), and $i(\chi)$ is its indicator function. Therefore, the nonsmooth ℓ_1 term in f, which is prox-friendly, can be smoothed. Figure 2f shows the results; as can be seen and mentioned in Remark 3.6, the proposed method exhibits two convergence phases, including an initial $\mathcal{O}(1/k^2)$ phase followed by linear convergence. Note that CP method cannot be applied to this problem because we have more than one nonsmooth terms in the objective function.

We provide additional numerical experiments using the real-world dataset LIBSVM [13], as well as extensive simulations with increasing problem dimensions. These results are presented in the Appendix. Furthermore, we note that the ADMM method is suitable for some of the aforementioned problems, and additional simulations for comparison are provided in the Appendix as well. However, ADMM may fail to converge, converge very slowly, or be difficult to implement when multiple nonsmooth terms are present, particularly when they appear in a summation that cannot be easily split [15].

Appendix A. Additional Numerical Results

This section presents additional numerical experiments showcasing the performance of the proposed method in the case of regression problem (27) and *MaxCut* (30) and also the comparison with the ADMM and Algorithm 1 with constant term $c = \varepsilon/L_f^2$ (Theorem 3.3, Alg. 1-2 in Figures).

- Figure 3 contains plots for (27) with $i = 1, 2$ and correlated data where all algorithms are initialized at the same point chosen randomly.

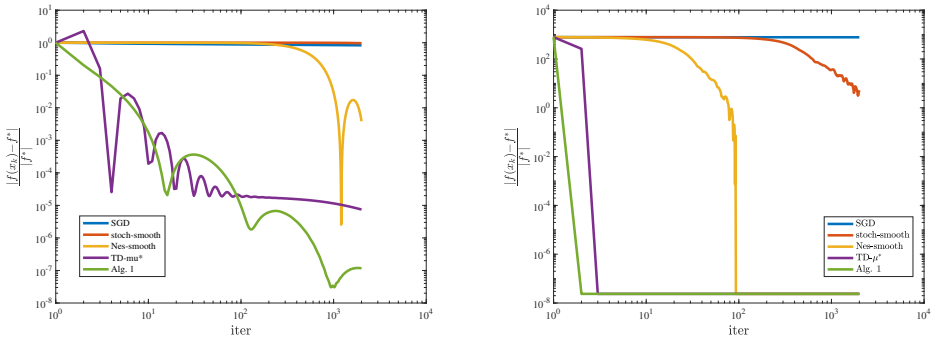


(A) ℓ_1-ℓ_1-regularized LASSO. (B) ℓ_2-ℓ_1-regularized LASSO.

Figure 3. Numerical results for regression problems in Subsection 5.1 (correlated data).

- Figure 4 illustrates the simulation results for the minimizing problem (30) different choice of η.

- Compared with ADMM: it seems that (3) is suitable for ADMM algorithm. Then, in Figures 5 and 6 we have also implemented our proposed algorithm (both versions with $c = 0$ and $c = \varepsilon/L_f^2$ in the smoothing rule) together with the current four existing approaches in the paper and ADMM from Algorithm 2 in [5] on two real datasets from the regression data of "abalone" and "mg" from



(A) $\mathcal{R}(y) = \|y\|_2^2$ and $\eta = 0.005$. (B) $\mathcal{R}(y) = \|y\|_1$ and $\eta = 10$.

FIGURE 4. Numerical results for *MaxCut* problem with different choice of $\mathcal{R}(y)$ in Subsection 5.2.

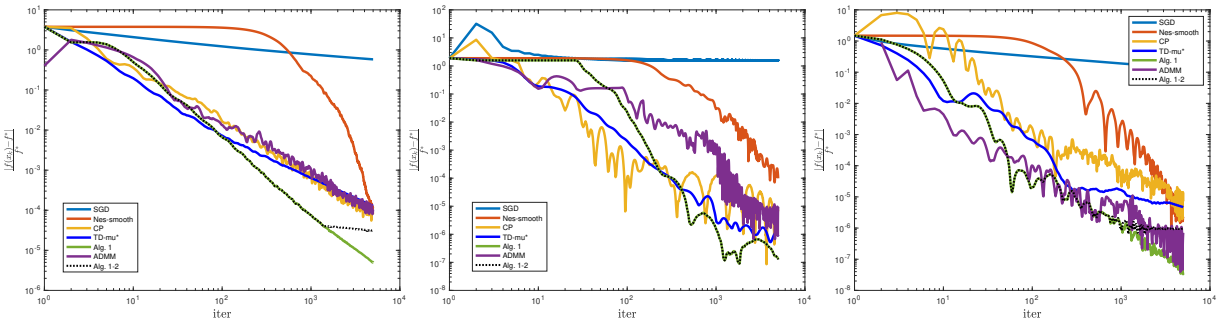
TABLE 2. Applications of Selected Regression Datasets from LIBSVM.

Dataset	**Application / Context**	**Reference**
"mg" (Mackey-Glass) (Fig. 5–6)	Time-series regression problem modeling a gas furnace system. The goal is to predict CO_2 concentration based on methane input and past outputs. Used in system identification and control engineering.	LIBSVM dataset; Box–Jenkins studies [13]
"abalone" (Fig. 5–6)	Predict the age of abalone (a type of mollusk) from physical measurements such as shell length, diameter, and weight. Used in marine biology for studying growth and population dynamics.	UCI Abalone dataset; LIBSVM [13]
"cpusmall"	Predict the relative performance of CPUs from their hardware specifications (e.g., memory, clock speed). Used in benchmarking and performance modeling in computer architecture.	StatLib CPU performance dataset; LIBSVM [13]
"triazines"	Predict the biological activity of triazine chemical compounds using molecular descriptors. Commonly used in chemoinformatics and quantitative structure activity relationship (QSAR) modeling for drug discovery.	QSAR modeling studies; Statlog; LIBSVM [13]

LIBSVM [13]. We run the ADMM with three different penalty parameters ($\varrho = 0.1, 1, 10$) and report the best one.

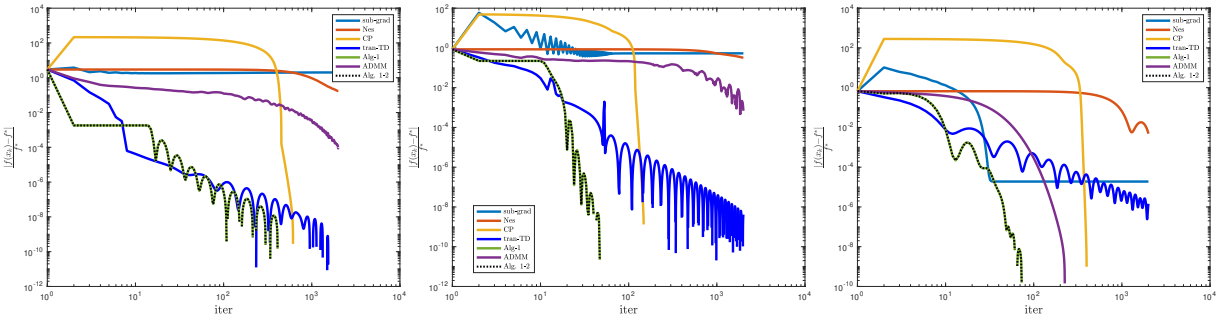
• Figure 7 contains plots for the regression problem (27) with $i = 1, 2$ and correlated data (setting similar to Figures 2a and 2d). All algorithms are initialized at the same randomly chosen point. This includes comparisons of Algorithm 1 with constant term $c = \varepsilon/L_f^2$ (Theorem 3.3) and various values of ε, as well as the case where the constant term $c = 0$.

• Real-world applications and dimentionality: Indeed, the applicability of the proposed method is an important consideration. The widespread use of nonsmooth convex optimization across various domains motivated us to include 16 diverse numerical examples in the paper. Relevant applications in image processing, image reconstruction, system identification, and control, particularly those



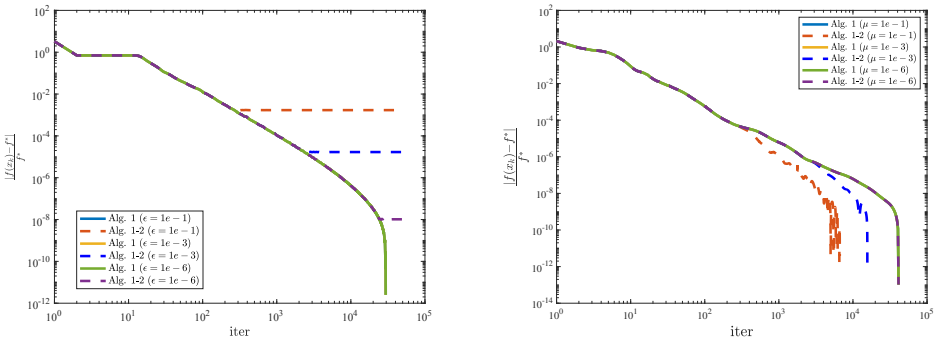
(A) setting similar to Fig. 2a. (B) real dataset ("abalone"). (C) real dataset ("mg").

FIGURE 5. Numerical results for ℓ_1-ℓ_1 regression problem as in (27).



(A) setting similar to Fig. 2d. (B) real dataset ("abalone"). (C) real dataset ("mg").

FIGURE 6. Numerical results for ℓ_1-ℓ_2 regression problem as in (27).



(A) ℓ_1-ℓ_1-regularized LASSO. (B) ℓ_2-ℓ_1-regularized LASSO.

FIGURE 7. Numerical results for regression problems with different values of ε.

involving the summation of nonsmooth terms, are discussed with references in Numerical results section. Additionally, in the Additional Numerical Results, we include simulations using real-world datasets "mg" and "abalone" from the LIBSVM regression collection. To provide further comparison, we perform 14 additional simulations by increasing the data dimensionality (5-20 times larger) under similar settings across various problem types where we report the optimality gap $\frac{|f(x_k)-f(x^*)|}{|f(x^*)|}$ at 10 equally spaced points throughout iterations. We also included new real-world datasets, "cpusmall" and "trianzime," from LIBSVM. Table 2 summarizes the application domain of each dataset.

- **ℓ_1-ℓ_1-regularized LASSO**

$n = 1000$, $m = 100$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**7.74**	$2.27{\cdot}10^{-2}$	$1.69{\cdot}10^{-2}$	$1.45{\cdot}10^{-2}$	$1.28{\cdot}10^{-2}$	$1.19{\cdot}10^{-2}$	$1.13{\cdot}10^{-2}$	$1.01{\cdot}10^{-2}$	$9.76{\cdot}10^{-3}$	$9.36{\cdot}10^{-3}$
Nes-smooth	**7.74**	3.43	$6.47{\cdot}10^{-1}$	$2.79{\cdot}10^{-1}$	$1.44{\cdot}10^{-1}$	$7.50{\cdot}10^{-2}$	$3.89{\cdot}10^{-2}$	$1.96{\cdot}10^{-2}$	$1.01{\cdot}10^{-2}$	$4.24{\cdot}10^{-3}$
CP	**7.74**	$3.84{\cdot}10^{-4}$	$1.39{\cdot}10^{-4}$	$1.27{\cdot}10^{-4}$	$4.85{\cdot}10^{-5}$	$5.46{\cdot}10^{-5}$	$7.41{\cdot}10^{-5}$	$2.55{\cdot}10^{-5}$	$4.44{\cdot}10^{-5}$	$1.58{\cdot}10^{-5}$
TD-μ^*	**7.74**	$2.26{\cdot}10^{-4}$	$4.24{\cdot}10^{-5}$	$2.26{\cdot}10^{-5}$	$1.56{\cdot}10^{-5}$	$1.27{\cdot}10^{-5}$	$1.01{\cdot}10^{-5}$	$8.94{\cdot}10^{-6}$	$7.40{\cdot}10^{-6}$	$6.72{\cdot}10^{-6}$
Alg. 1	**7.74**	$\mathbf{7.37{\cdot}10^{-5}}$	$\mathbf{1.55{\cdot}10^{-5}}$	$\mathbf{5.04{\cdot}10^{-6}}$	$\mathbf{2.35{\cdot}10^{-6}}$	$\mathbf{1.49{\cdot}10^{-6}}$	$\mathbf{1.01{\cdot}10^{-6}}$	$\mathbf{6.94{\cdot}10^{-7}}$	$\mathbf{5.10{\cdot}10^{-7}}$	$\mathbf{3.53{\cdot}10^{-7}}$

$n = 100$, $m = 1000$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**17.79**	$1.37{\cdot}10^{-3}$	$8.91{\cdot}10^{-4}$	$4.82{\cdot}10^{-4}$	$3.57{\cdot}10^{-4}$	$3.43{\cdot}10^{-4}$	$2.60{\cdot}10^{-4}$	$2.35{\cdot}10^{-4}$	$2.01{\cdot}10^{-4}$	$2.15{\cdot}10^{-4}$
Nes-smooth	**17.79**	1.75	$3.05{\cdot}10^{-1}$	$2.40{\cdot}10^{-2}$	$\mathbf{3.88{\cdot}10^{-5}}$	$3.02{\cdot}10^{-5}$	$3.00{\cdot}10^{-5}$	$3.01{\cdot}10^{-5}$	$3.00{\cdot}10^{-5}$	$3.01{\cdot}10^{-5}$
CP	**17.79**	$1.56{\cdot}10^{-3}$	$1.69{\cdot}10^{-3}$	$1.07{\cdot}10^{-3}$	$8.38{\cdot}10^{-4}$	$6.70{\cdot}10^{-4}$	$6.77{\cdot}10^{-4}$	$5.31{\cdot}10^{-4}$	$4.49{\cdot}10^{-4}$	$3.78{\cdot}10^{-4}$
TD-μ^*	**17.79**	$1.17{\cdot}10^{-3}$	$6.65{\cdot}10^{-4}$	$5.02{\cdot}10^{-4}$	$3.90{\cdot}10^{-4}$	$3.13{\cdot}10^{-4}$	$2.54{\cdot}10^{-4}$	$2.14{\cdot}10^{-4}$	$1.91{\cdot}10^{-4}$	$1.74{\cdot}10^{-4}$
Alg. 1	**17.79**	$\mathbf{5.90{\cdot}10^{-4}}$	$\mathbf{1.61{\cdot}10^{-4}}$	$\mathbf{7.64{\cdot}10^{-5}}$	$4.26{\cdot}10^{-5}$	$\mathbf{2.73{\cdot}10^{-5}}$	$\mathbf{1.89{\cdot}10^{-5}}$	$\mathbf{1.39{\cdot}10^{-5}}$	$\mathbf{1.07{\cdot}10^{-5}}$	$\mathbf{8.52{\cdot}10^{-6}}$

$n = 2000$, $m = 500$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**5.28**	$1.63{\cdot}10^{-2}$	$1.19{\cdot}10^{-2}$	$9.75{\cdot}10^{-3}$	$8.45{\cdot}10^{-3}$	$7.67{\cdot}10^{-3}$	$7.07{\cdot}10^{-3}$	$6.45{\cdot}10^{-3}$	$5.85{\cdot}10^{-3}$	$5.48{\cdot}10^{-3}$
Nes-smooth	**5.28**	2.46	$4.80{\cdot}10^{-1}$	$2.11{\cdot}10^{-1}$	$1.05{\cdot}10^{-1}$	$5.46{\cdot}10^{-2}$	$2.83{\cdot}10^{-2}$	$1.53{\cdot}10^{-2}$	$7.30{\cdot}10^{-3}$	$3.41{\cdot}10^{-3}$
CP	**5.28**	$1.22{\cdot}10^{-4}$	$4.21{\cdot}10^{-5}$	$2.43{\cdot}10^{-5}$	$1.31{\cdot}10^{-5}$	$8.99{\cdot}10^{-6}$	$7.86{\cdot}10^{-6}$	$4.77{\cdot}10^{-6}$	$3.40{\cdot}10^{-6}$	$2.92{\cdot}10^{-6}$
TD-μ^*	**5.28**	$6.05{\cdot}10^{-5}$	$2.62{\cdot}10^{-5}$	$1.66{\cdot}10^{-5}$	$1.19{\cdot}10^{-5}$	$9.45{\cdot}10^{-6}$	$7.92{\cdot}10^{-6}$	$6.95{\cdot}10^{-6}$	$5.92{\cdot}10^{-6}$	$5.29{\cdot}10^{-6}$
Alg. 1	**5.28**	$\mathbf{2.30{\cdot}10^{-5}}$	$\mathbf{4.83{\cdot}10^{-6}}$	$\mathbf{2.35{\cdot}10^{-6}}$	$\mathbf{1.39{\cdot}10^{-6}}$	$\mathbf{1.02{\cdot}10^{-6}}$	$\mathbf{7.62{\cdot}10^{-7}}$	$\mathbf{5.83{\cdot}10^{-7}}$	$\mathbf{4.72{\cdot}10^{-7}}$	$\mathbf{3.93{\cdot}10^{-7}}$

$n = 500$, $m = 2000$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**2.29**	$\mathbf{4.29{\cdot}10^{-5}}$	$\mathbf{1.58{\cdot}10^{-5}}$	$9.28{\cdot}10^{-6}$	$6.58{\cdot}10^{-6}$	$5.70{\cdot}10^{-6}$	$4.64{\cdot}10^{-6}$	$4.01{\cdot}10^{-6}$	$3.60{\cdot}10^{-6}$	$3.73{\cdot}10^{-6}$
Nes-smooth	**2.29**	$5.63{\cdot}10^{-1}$	$9.02{\cdot}10^{-2}$	$3.23{\cdot}10^{-2}$	$1.51{\cdot}10^{-2}$	$6.63{\cdot}10^{-3}$	$1.89{\cdot}10^{-3}$	$1.44{\cdot}10^{-4}$	$6.14{\cdot}10^{-6}$	$4.24{\cdot}10^{-6}$
CP	**2.29**	$4.77{\cdot}10^{-4}$	$2.50{\cdot}10^{-4}$	$1.87{\cdot}10^{-4}$	$1.49{\cdot}10^{-4}$	$1.17{\cdot}10^{-4}$	$9.05{\cdot}10^{-5}$	$7.90{\cdot}10^{-5}$	$6.35{\cdot}10^{-5}$	$6.03{\cdot}10^{-5}$
TD-μ^*	**2.29**	$4.37{\cdot}10^{-4}$	$1.84{\cdot}10^{-4}$	$1.21{\cdot}10^{-4}$	$9.20{\cdot}10^{-5}$	$7.50{\cdot}10^{-5}$	$6.34{\cdot}10^{-5}$	$5.46{\cdot}10^{-5}$	$4.77{\cdot}10^{-5}$	$4.23{\cdot}10^{-5}$
Alg. 1	**2.29**	$8.54{\cdot}10^{-5}$	$2.06{\cdot}10^{-5}$	$\mathbf{8.55{\cdot}10^{-6}}$	$\mathbf{4.58{\cdot}10^{-6}}$	$\mathbf{2.77{\cdot}10^{-6}}$	$\mathbf{1.82{\cdot}10^{-6}}$	$\mathbf{1.27{\cdot}10^{-6}}$	$\mathbf{9.21{\cdot}10^{-7}}$	$\mathbf{6.86{\cdot}10^{-7}}$

cpusmall dataset $n = 12$, $m = 8192$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{1.62{\cdot}10^{5}}$	$3.56{\cdot}10^{2}$	$7.61{\cdot}10^{1}$	$7.54{\cdot}10^{1}$	$3.46{\cdot}10^{2}$	$3.44{\cdot}10^{2}$	$7.32{\cdot}10^{1}$	$7.24{\cdot}10^{1}$	$3.36{\cdot}10^{2}$	$3.33{\cdot}10^{2}$
Nes-smooth	$\mathbf{1.62{\cdot}10^{5}}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$	$1.62{\cdot}10^{5}$
CP	$\mathbf{1.62{\cdot}10^{5}}$	$1.64{\cdot}10^{3}$	$7.07{\cdot}10^{2}$	$3.43{\cdot}10^{2}$	$4.45{\cdot}10^{2}$	$7.03{\cdot}10^{2}$	$4.57{\cdot}10^{2}$	$2.86{\cdot}10^{2}$	$6.77{\cdot}10^{2}$	$6.48{\cdot}10^{2}$
TD-μ^*	$\mathbf{1.62{\cdot}10^{5}}$	$6.99{\cdot}10^{1}$	$4.43{\cdot}10^{1}$	$1.62{\cdot}10^{1}$	$1.36{\cdot}10^{1}$	$1.86{\cdot}10^{1}$	$1.30{\cdot}10^{1}$	$9.40{\cdot}10^{0}$	$1.41{\cdot}10^{1}$	$1.36{\cdot}10^{1}$
Alg. 1	$\mathbf{1.62{\cdot}10^{5}}$	$\mathbf{1.43{\cdot}10^{0}}$	$\mathbf{1.14{\cdot}10^{0}}$	$\mathbf{9.00{\cdot}10^{-1}}$	$\mathbf{7.27{\cdot}10^{-1}}$	$\mathbf{5.97{\cdot}10^{-1}}$	$\mathbf{4.98{\cdot}10^{-1}}$	$\mathbf{4.20{\cdot}10^{-1}}$	$\mathbf{3.60{\cdot}10^{-1}}$	$\mathbf{3.13{\cdot}10^{-1}}$

trianzines dataset $n = 60$, $m = 186$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{2.16{\cdot}10^{2}}$	$5.02{\cdot}10^{-1}$	$7.84{\cdot}10^{-2}$	$9.15{\cdot}10^{-2}$	$6.76{\cdot}10^{-2}$	$8.46{\cdot}10^{-2}$	$4.20{\cdot}10^{-2}$	$4.92{\cdot}10^{-2}$	$3.91{\cdot}10^{-2}$	$2.60{\cdot}10^{-2}$
Nes-smooth	$\mathbf{2.16{\cdot}10^{2}}$	$1.25{\cdot}10^{1}$	$5.22{\cdot}10^{0}$	$9.11{\cdot}10^{-1}$	$5.03{\cdot}10^{-1}$	$2.80{\cdot}10^{-1}$	$1.74{\cdot}10^{-1}$	$1.19{\cdot}10^{-1}$	$7.86{\cdot}10^{-2}$	$4.46{\cdot}10^{-2}$
CP	$\mathbf{2.16{\cdot}10^{2}}$	$1.03{\cdot}10^{-1}$	$4.34{\cdot}10^{-2}$	$2.97{\cdot}10^{-2}$	$2.61{\cdot}10^{-2}$	$1.55{\cdot}10^{-2}$	$1.54{\cdot}10^{-2}$	$1.44{\cdot}10^{-2}$	$8.14{\cdot}10^{-3}$	$9.77{\cdot}10^{-3}$
TD-μ^*	$\mathbf{2.16{\cdot}10^{2}}$	$2.21{\cdot}10^{-2}$	$9.72{\cdot}10^{-3}$	$4.57{\cdot}10^{-3}$	$4.05{\cdot}10^{-3}$	$3.36{\cdot}10^{-3}$	$2.55{\cdot}10^{-3}$	$1.99{\cdot}10^{-3}$	$1.87{\cdot}10^{-3}$	$1.59{\cdot}10^{-3}$
Alg. 1	$\mathbf{2.16{\cdot}10^{2}}$	$\mathbf{1.71{\cdot}10^{-2}}$	$\mathbf{4.48{\cdot}10^{-3}}$	$\mathbf{1.72{\cdot}10^{-3}}$	$\mathbf{8.98{\cdot}10^{-4}}$	$\mathbf{6.66{\cdot}10^{-4}}$	$\mathbf{4.08{\cdot}10^{-4}}$	$\mathbf{3.58{\cdot}10^{-4}}$	$\mathbf{2.63{\cdot}10^{-4}}$	$\mathbf{1.86{\cdot}10^{-4}}$

- **ℓ_2-ℓ_1-regularized LASSO**

$n = 1000$, $m = 100$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**7.57**	$5.75{\cdot}10^{-2}$	$4.42{\cdot}10^{-2}$	$3.89{\cdot}10^{-2}$	$3.58{\cdot}10^{-2}$	$3.38{\cdot}10^{-2}$	$3.24{\cdot}10^{-2}$	$3.12{\cdot}10^{-2}$	$3.03{\cdot}10^{-2}$	$2.94{\cdot}10^{-2}$
Nes-smooth	**7.57**	$7.20{\cdot}10^{0}$	$6.19{\cdot}10^{0}$	$4.76{\cdot}10^{0}$	$3.32{\cdot}10^{0}$	$2.39{\cdot}10^{0}$	$1.66{\cdot}10^{0}$	$1.15{\cdot}10^{0}$	$8.18{\cdot}10^{-1}$	$6.66{\cdot}10^{-1}$
CP	**7.57**	$8.60{\cdot}10^{0}$	$5.86{\cdot}10^{-2}$	$1.98{\cdot}10^{-2}$	$9.62{\cdot}10^{-3}$	$5.73{\cdot}10^{-3}$	$3.58{\cdot}10^{-3}$	$2.51{\cdot}10^{-3}$	$1.76{\cdot}10^{-3}$	$1.27{\cdot}10^{-3}$
TD-μ^*	**7.57**	$\mathbf{2.76{\cdot}10^{-3}}$	$1.37{\cdot}10^{-3}$	$9.06{\cdot}10^{-4}$	$6.74{\cdot}10^{-4}$	$5.36{\cdot}10^{-4}$	$4.45{\cdot}10^{-4}$	$3.82{\cdot}10^{-4}$	$3.34{\cdot}10^{-4}$	$2.97{\cdot}10^{-4}$
Alg. 1	**7.57**	$3.63{\cdot}10^{-3}$	$\mathbf{9.11{\cdot}10^{-4}}$	$\mathbf{4.06{\cdot}10^{-4}}$	$\mathbf{2.28{\cdot}10^{-4}}$	$\mathbf{1.46{\cdot}10^{-4}}$	$\mathbf{1.01{\cdot}10^{-4}}$	$\mathbf{7.43{\cdot}10^{-5}}$	$\mathbf{5.70{\cdot}10^{-5}}$	$\mathbf{4.50{\cdot}10^{-5}}$

$n = 100$, $m = 1000$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{1.91{\cdot}10^{2}}$	$1.61{\cdot}10^{-2}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$
Nes-smooth	$\mathbf{1.91{\cdot}10^{2}}$	$1.79{\cdot}10^{2}$	$1.45{\cdot}10^{2}$	$9.21{\cdot}10^{1}$	$3.15{\cdot}10^{1}$	$2.10{\cdot}10^{1}$	$1.82{\cdot}10^{1}$	$9.64{\cdot}10^{0}$	$9.05{\cdot}10^{0}$	$7.50{\cdot}10^{0}$
CP	$\mathbf{1.91{\cdot}10^{2}}$	$2.39{\cdot}10^{3}$	$1.06{\cdot}10^{0}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$
TD-μ^*	$\mathbf{1.91{\cdot}10^{2}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$
Alg. 1	$\mathbf{1.91{\cdot}10^{2}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$	$\mathbf{8.33{\cdot}10^{-9}}$

$n = 2000$, $m = 500$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**4.04**	$2.92{\cdot}10^{-2}$	$2.30{\cdot}10^{-2}$	$2.01{\cdot}10^{-2}$	$1.86{\cdot}10^{-2}$	$1.76{\cdot}10^{-2}$	$1.67{\cdot}10^{-2}$	$1.61{\cdot}10^{-2}$	$1.56{\cdot}10^{-2}$	$1.52{\cdot}10^{-2}$
Nes-smooth	**4.04**	$3.96{\cdot}10^{0}$	$3.72{\cdot}10^{0}$	$3.33{\cdot}10^{0}$	$2.84{\cdot}10^{0}$	$2.30{\cdot}10^{0}$	$1.79{\cdot}10^{0}$	$1.35{\cdot}10^{0}$	$1.04{\cdot}10^{0}$	$8.24{\cdot}10^{-1}$
CP	**4.04**	$5.29{\cdot}10^{1}$	$1.20{\cdot}10^{1}$	$2.07{\cdot}10^{-1}$	$3.14{\cdot}10^{-2}$	$1.36{\cdot}10^{-2}$	$6.94{\cdot}10^{-3}$	$4.32{\cdot}10^{-3}$	$2.94{\cdot}10^{-3}$	$2.15{\cdot}10^{-3}$
TD-μ^*	**4.04**	$\mathbf{1.07{\cdot}10^{-3}}$	$5.23{\cdot}10^{-4}$	$3.46{\cdot}10^{-4}$	$2.57{\cdot}10^{-4}$	$2.05{\cdot}10^{-4}$	$1.70{\cdot}10^{-4}$	$1.46{\cdot}10^{-4}$	$1.27{\cdot}10^{-4}$	$1.13{\cdot}10^{-4}$
Alg. 1	**4.04**	$1.49{\cdot}10^{-3}$	$\mathbf{3.82{\cdot}10^{-4}}$	$\mathbf{1.71{\cdot}10^{-4}}$	$\mathbf{9.69{\cdot}10^{-5}}$	$\mathbf{6.22{\cdot}10^{-5}}$	$\mathbf{4.33{\cdot}10^{-5}}$	$\mathbf{3.22{\cdot}10^{-5}}$	$\mathbf{2.48{\cdot}10^{-5}}$	$\mathbf{1.98{\cdot}10^{-5}}$

$n = 500$, $m = 2000$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{2.43{\cdot}10^{2}}$	$3.47{\cdot}10^{-1}$	$9.03{\cdot}10^{-3}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$
Nes-smooth	$\mathbf{2.43{\cdot}10^{2}}$	$2.39{\cdot}10^{2}$	$2.28{\cdot}10^{2}$	$2.10{\cdot}10^{2}$	$1.85{\cdot}10^{2}$	$1.55{\cdot}10^{2}$	$1.22{\cdot}10^{2}$	$8.85{\cdot}10^{1}$	$5.97{\cdot}10^{1}$	$4.04{\cdot}10^{1}$
CP	$\mathbf{2.43{\cdot}10^{2}}$	$4.64{\cdot}10^{3}$	$2.80{\cdot}10^{3}$	$1.33{\cdot}10^{3}$	$3.53{\cdot}10^{2}$	$8.71{\cdot}10^{-4}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$
TD-μ^*	$\mathbf{2.43{\cdot}10^{2}}$	$8.29{\cdot}10^{-8}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$
Alg. 1	$\mathbf{2.43{\cdot}10^{2}}$	$\mathbf{9.25{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$	$\mathbf{9.27{\cdot}10^{-9}}$

cpusmall dataset $n = 12$, $m = 8192$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{9.97{\cdot}10^{4}}$	$5.38{\cdot}10^{4}$	$3.23{\cdot}10^{5}$	$2.63{\cdot}10^{5}$	$8.66{\cdot}10^{3}$	$3.71{\cdot}10^{3}$	$1.86{\cdot}10^{5}$	$1.74{\cdot}10^{5}$	$4.28{\cdot}10^{3}$	$3.14{\cdot}10^{3}$
Nes-smooth	$\mathbf{9.97{\cdot}10^{4}}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$	$9.97{\cdot}10^{4}$
CP	$\mathbf{9.97{\cdot}10^{4}}$	$4.97{\cdot}10^{3}$	$2.01{\cdot}10^{3}$	$2.66{\cdot}10^{2}$	$9.82{\cdot}10^{2}$	$7.48{\cdot}10^{2}$	$2.57{\cdot}10^{2}$	$7.99{\cdot}10^{1}$	$1.65{\cdot}10^{2}$	$1.13{\cdot}10^{2}$
TD-μ^*	$\mathbf{9.97{\cdot}10^{4}}$	$4.67{\cdot}10^{1}$	$2.20{\cdot}10^{1}$	$1.57{\cdot}10^{1}$	$9.63{\cdot}10^{0}$	$5.76{\cdot}10^{0}$	$5.82{\cdot}10^{0}$	$7.20{\cdot}10^{0}$	$7.53{\cdot}10^{0}$	$6.60{\cdot}10^{0}$
Alg. 1	$\mathbf{9.97{\cdot}10^{4}}$	$\mathbf{1.61{\cdot}10^{-1}}$	$\mathbf{1.31{\cdot}10^{-1}}$	$\mathbf{7.50{\cdot}10^{-2}}$	$\mathbf{5.57{\cdot}10^{-2}}$	$\mathbf{5.03{\cdot}10^{-2}}$	$\mathbf{4.84{\cdot}10^{-2}}$	$\mathbf{4.76{\cdot}10^{-2}}$	$\mathbf{4.72{\cdot}10^{-2}}$	$\mathbf{4.70{\cdot}10^{-2}}$

trianzines dataset $n = 60$, $m = 186$

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{6.58{\cdot}10^{1}}$	$6.57{\cdot}10^{1}$	$6.56{\cdot}10^{1}$	$6.56{\cdot}10^{1}$	$6.55{\cdot}10^{1}$	$6.55{\cdot}10^{1}$	$6.55{\cdot}10^{1}$	$6.55{\cdot}10^{1}$	$6.54{\cdot}10^{1}$	$6.54{\cdot}10^{1}$
Nes-smooth	$\mathbf{6.58{\cdot}10^{1}}$	$4.92{\cdot}10^{1}$	$2.66{\cdot}10^{1}$	$2.11{\cdot}10^{1}$	$1.76{\cdot}10^{1}$	$1.61{\cdot}10^{1}$	$1.45{\cdot}10^{1}$	$1.33{\cdot}10^{1}$	$1.18{\cdot}10^{1}$	$1.02{\cdot}10^{1}$
CP	$\mathbf{6.58{\cdot}10^{1}}$	$1.71{\cdot}10^{1}$	$5.04{\cdot}10^{-1}$	$2.77{\cdot}10^{-1}$	$1.79{\cdot}10^{-1}$	$1.16{\cdot}10^{-1}$	$6.81{\cdot}10^{-2}$	$4.12{\cdot}10^{-2}$	$2.07{\cdot}10^{-2}$	$2.85{\cdot}10^{-3}$
TD-μ^*	$\mathbf{6.58{\cdot}10^{1}}$	$1.59{\cdot}10^{-1}$	$5.98{\cdot}10^{-2}$	$3.59{\cdot}10^{-2}$	$1.88{\cdot}10^{-2}$	$1.77{\cdot}10^{-2}$	$1.28{\cdot}10^{-2}$	$9.91{\cdot}10^{-3}$	$8.18{\cdot}10^{-3}$	$6.61{\cdot}10^{-3}$
Alg. 1	$\mathbf{6.58{\cdot}10^{1}}$	$\mathbf{5.21{\cdot}10^{-2}}$	$\mathbf{3.92{\cdot}10^{-4}}$	$\mathbf{2.98{\cdot}10^{-5}}$	$\mathbf{2.61{\cdot}10^{-5}}$	$\mathbf{1.86{\cdot}10^{-5}}$	$\mathbf{9.84{\cdot}10^{-6}}$	$\mathbf{6.39{\cdot}10^{-6}}$	$\mathbf{5.92{\cdot}10^{-6}}$	$\mathbf{5.76{\cdot}10^{-6}}$

- **MaxCut problem**

$\mathcal{R}(y) = \|y\|_1$, $\eta = 1$, dimension of matrix = 300

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	$\mathbf{2.50{\cdot}10^{2}}$	$2.30{\cdot}10^{2}$	$2.28{\cdot}10^{2}$	$2.27{\cdot}10^{2}$	$2.26{\cdot}10^{2}$	$2.25{\cdot}10^{2}$	$2.25{\cdot}10^{2}$	$2.24{\cdot}10^{2}$	$2.24{\cdot}10^{2}$	$2.24{\cdot}10^{2}$
stoch-smooth	$\mathbf{2.50{\cdot}10^{2}}$	$2.46{\cdot}10^{2}$	$2.34{\cdot}10^{2}$	$2.15{\cdot}10^{2}$	$1.90{\cdot}10^{2}$	$1.61{\cdot}10^{2}$	$1.30{\cdot}10^{2}$	$9.73{\cdot}10^{1}$	$6.77{\cdot}10^{1}$	$4.23{\cdot}10^{1}$
Nes-smooth	$\mathbf{2.50{\cdot}10^{2}}$	$1.16{\cdot}10^{0}$	$9.45{\cdot}10^{-1}$	$7.01{\cdot}10^{-1}$	$4.65{\cdot}10^{-1}$	$2.23{\cdot}10^{-1}$	$3.66{\cdot}10^{-2}$	$2.55{\cdot}10^{-3}$	$3.80{\cdot}10^{-4}$	$5.92{\cdot}10^{-5}$
TD-μ^*	$\mathbf{2.50{\cdot}10^{2}}$	$1.26{\cdot}10^{-2}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$
Alg. 1	$\mathbf{2.50{\cdot}10^{2}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$	$\mathbf{2.16{\cdot}10^{-9}}$

$\mathcal{R}(y) = \|y\|^2$, $\eta = 0.05$, dimension of matrix = 300

Algorithm	point #1	point #2	point #3	point #4	point #5	point #6	point #7	point #8	point #9	point #10
SGD	**1.01**	$2.44{\cdot}10^{-1}$	$2.12{\cdot}10^{-1}$	$1.96{\cdot}10^{-1}$	$1.85{\cdot}10^{-1}$	$1.77{\cdot}10^{-1}$	$1.70{\cdot}10^{-1}$	$1.65{\cdot}10^{-1}$	$1.61{\cdot}10^{-1}$	$1.57{\cdot}10^{-1}$
stoch-smooth	**1.01**	$1.01{\cdot}10^{0}$	$1.01{\cdot}10^{0}$	$1.00{\cdot}10^{0}$	$9.92{\cdot}10^{-1}$	$9.79{\cdot}10^{-1}$	$9.65{\cdot}10^{-1}$	$9.47{\cdot}10^{-1}$	$9.28{\cdot}10^{-1}$	$9.06{\cdot}10^{-1}$
Nes-smooth	**1.01**	$2.35{\cdot}10^{-1}$	$9.90{\cdot}10^{-3}$	$7.45{\cdot}10^{-4}$	$3.01{\cdot}10^{-3}$	$1.15{\cdot}10^{-3}$	$5.71{\cdot}10^{-6}$	$3.45{\cdot}10^{-4}$	$3.67{\cdot}10^{-4}$	$4.55{\cdot}10^{-5}$
TD-μ^*	**1.01**	$1.38{\cdot}10^{-5}$	$1.01{\cdot}10^{-5}$	$7.66{\cdot}10^{-6}$	$6.10{\cdot}10^{-6}$	$5.05{\cdot}10^{-6}$	$4.31{\cdot}10^{-6}$	$3.76{\cdot}10^{-6}$	$3.33{\cdot}10^{-6}$	$3.00{\cdot}10^{-6}$
Alg. 1	**1.01**	$\mathbf{1.30{\cdot}10^{-7}}$	$\mathbf{3.13{\cdot}10^{-8}}$	$\mathbf{1.35{\cdot}10^{-8}}$	$\mathbf{7.36{\cdot}10^{-9}}$	$\mathbf{4.55{\cdot}10^{-9}}$	$\mathbf{3.04{\cdot}10^{-9}}$	$\mathbf{2.14{\cdot}10^{-9}}$	$\mathbf{1.57{\cdot}10^{-9}}$	$\mathbf{1.18{\cdot}10^{-9}}$

References

[1] Feriel Abboud, Emilie Chouzenoux, Jean-Christophe Pesquet, Jean-Hugues Chenot, and Louis Laborelli. Dual block-coordinate forward-backward algorithm with application to deconvolution and deinterlacing of video sequences. *Journal of Mathematical Imaging and Vision*, 59:415–431, 2017.

[2] Mariette Annergren, Anders Hansson, and Bo Wahlberg. An admm algorithm for solving ℓ_1 regularized mpc. In *IEEE Conference on Decision and Control*, 2012.

[3] Haim Avron, Satyen Kale, Shiva Kasiviswanathan, and Vikas Sindhwani. Efficient and practical stochastic subgradient descent for nuclear norm regularization. *preprint available at arXiv:1206.6384*, 2012.

[4] Reza Rahimi Baghbadorani, Sergio Grammatico, and Peyman Mohajerin Esfahani. Adaptive accelerated composite minimization. *preprint available at arXiv:2405.03414*, 2024.

[5] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[6] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

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

[8] Amir Beck and Marc Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[9] Yunier Bello-Cruz, Guoyin Li, and Tran Thai An Nghia. Quadratic growth conditions and uniqueness of optimal solution to lasso. *Journal of Optimization Theory and Applications*, 194(1):167–190, 2022.

[10] Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistic*, 37(4):1705–1732, 2009.

[11] Radu Ioan Boţ and Christopher Hendrich. A variable smoothing algorithm for solving convex optimization problems. *An Official Journal of the Spanish Society of Statistics and Operations Research*, 23:124–150, 2015.

[12] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145, 2011.

[13] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[14] Caroline Chaux, Jean-Christophe Pesquet, and Nelly Pustelnik. Nested iterative algorithms for convex constrained image recovery problems. *SIAM Journal on Imaging Sciences*, 2(2):730–762, 2009.

[15] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1):57–79, 2016.

[16] Alexandre dAspremont, Laurent Ghaoui, Michael Jordan, and Gert Lanckriet. A direct formulation for sparse pca using semidefinite programming. *Advances in Neural Information Processing Systems*, 2004.

[17] Alexandre dAspremont and Noureddine Karoui. A stochastic smoothing algorithm for semidefinite programming. *SIAM Journal on Optimization*, 24(3):1138–1177, 2014.

[18] Haoying Fu, Michael K Ng, Mila Nikolova, and Jesse L Barlow. Efficient minimization methods of mixed l2-l1 and l1-l1 norms for image restoration. *SIAM Journal on Scientific computing*, 27(6):1881–1902, 2006.

[19] Benjamin Grimmer, Haihao Lu, Pratik Worah, and Vahab Mirrokni. The landscape of the proximal point method for nonconvex–nonconcave minimax optimization. *Mathematical Programming*, 201(1):373–407, 2023.

[20] Yuhong Guo. Robust transfer principal component analysis with rank constraints. *Advances in Neural Information Processing Systems*, 2013.

[21] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[22] Xiao-Bing Hu and Wen-Hua Chen. Model predictive control for constrained systems with uncertain state-delays. *International Journal of Robust and Nonlinear Control*, 14(17):1421–1432, 2004.

[23] Martin Jaggi, Marek Sulovsk, et al. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[24] Karl Henrik Johansson, Alexander Horch, Olle Wijk, and Anders Hansson. Teaching multivariable control using the quadruple-tank process. In *38th IEEE Conference on Decision and Control*, volume 1. IEEE, 1999.

[25] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.

[26] Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Activity identification and local linear convergence of forward–backward-type methods. *SIAM Journal on Optimization*, 27(1):408–437, 2017.

[27] Jingwei Liang, Jalal M Fadili, and Gabriel Peyré. Local linear convergence of forward–backward under partial smoothness. *Advances in neural information processing systems*, 27, 2014.

[28] Arkadij Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[29] Becker S Bobin J Nesta. A fast and accurate first-order method for sparse recovery, 2009.

[30] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[31] Yurii Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1), 2005.

[32] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

[33] Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007.

[34] Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[35] Jacques Noom, Oleg Soloviev, and Michel Verhaegen. Proximal-based recursive implementation for model-free data-driven fault diagnosis. *Automatica*, 165:111656, 2024.

[36] Francesco Orabona, Andreas Argyriou, and Nathan Srebro. Prisma: Proximal iterative smoothing algorithm. *preprint available at arXiv:1206.2372*, 2012.

[37] Sina Khoshfetrat Pakazad, Henrik Ohlsson, and Lennart Ljung. Sparse control using sum-of-norms regularized model predictive control. In *52nd IEEE Conference on Decision and Control*, pages 5758–5763. IEEE, 2013.

[38] René A Poliquin and R Tyrrell Rockafellar. Generalized hessian properties of regularized nonsmooth functions. *SIAM Journal on Optimization*, 6(4):1121–1137, 1996.

[39] Boris Polyak. *Introduction to Optimization*. New York, NY, USA: Optimization Software, 1987.

[40] Nelly Pustelnik and Laurent Condat. Proximity operator of a sum of functions; application to depth map estimation. *IEEE Signal Processing Letters*, 24(12):1827–1831, 2017.

[41] Mehmet Fatih Sahin, Armin Eftekhari, Ahmet Alacaoglu, Fabian Latorre, and Volkan Cevher. An inexact augmented lagrangian framework for nonconvex optimization with nonlinear constraints. *Advances in Neural Information Processing Systems*, 2019.

[42] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

[43] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. *Advances in Neural Information Processing Systems*, 2004.

[44] Yue Sun, Samet Oymak, and Maryam Fazel. System identification via nuclear norm regularization. *preprint available at arXiv:2203.16673*, 2022.

[45] Christos Thrampoulidis, Ehsan Abbasi, and Babak Hassibi. Lasso with non-linear measurements is equivalent to one with linear measurements. *Advances in Neural Information Processing Systems*, 2015.

[46] Quoc Tran-Dinh. Adaptive smoothing algorithms for nonsmooth composite convex minimization. *Computational Optimization and Applications*, 66:425–451, 2017.

[47] Shanshan Wang, Qiegen Liu, Yong Xia, Pei Dong, Jianhua Luo, Qiu Huang, and David Dagan Feng. Dictionary learning based impulse noise removal via l1–l1 minimization. *Signal Processing*, 93(9):2696–2708, 2013.

[48] Junfeng Yang and Yin Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.

[49] Yao-Liang Yu. On decomposing the proximal map. *Advances in Neural Information Processing Systems*, 2013.

[50] Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. A conditional-gradient-based augmented lagrangian framework. *ICML*, 2019.