Restarts subject to approximate sharpness: A parameter-free and optimal scheme for first-order methods

Ben Adcock, Matthew J. Colbrook, Maksym Neyra-Nesterenko

Abstract

Sharpness is an almost generic assumption in continuous optimization that bounds the distance from minima by objective function suboptimality. It leads to the acceleration of first-order methods via *restarts*. However, sharpness involves problem-specific constants that are typically unknown, and previous restart schemes reduce convergence rates. Moreover, such schemes are challenging to apply in the presence of noise or approximate model classes (e.g., in compressive imaging or learning problems), and typically assume that the first-order method used produces feasible iterates. We consider the assumption of *approximate sharpness*, a generalization of sharpness that incorporates an unknown constant perturbation to the objective function error. This constant offers greater robustness (e.g., with respect to noise or relaxation of model classes) for finding approximate minimizers. By employing a new type of search over the unknown constants, we design a restart scheme that applies to general first-order methods and does not require the first-order method to produce feasible iterates. Our scheme maintains the same convergence rate as when assuming knowledge of the constants. The rates of convergence we obtain for various first-order methods either match the optimal rates or improve on previously established rates for a wide range of problems. We showcase our restart scheme on several examples and point to future applications and developments of our framework and theory.

Keywords: First-order methods, Restarting and acceleration, Approximate sharpness, Convex optimization, Convergence rates, Inverse problems

Mathematics Subject Classification: 65K0, 65B99, 68Q25, 90C25, 90C60

1 Introduction

First-order methods are the workhorse of much of modern continuous optimization [6, 10, 24, 59]. They are widely used to solve large-scale problems because of their excellent scalability and easiness of implementation. However, standard first-order methods often converge slowly, for instance, when applied to nonsmooth objective functions or functions lacking strong convexity. This has motivated a large amount of work on speeding up such methods [11, 30, 48, 55, 60, 64, 65].

Recently there has been significant interest in using *restarts* to accelerate the convergence of first-order methods [1, 13, 27, 33, 34, 37, 39, 44, 46, 47, 49, 52, 57, 61, 62, 66, 68, 69, 71]. A restart scheme repeatedly takes the output of an optimization algorithm instance as the initial point of a new instance or "restart", and additionally may reselect the algorithm parameters before executing the new instance. Under the right conditions, the objective error and feasibility gap decay faster for the restarted scheme than for the underlying (unrestarted) first-order method.

[2] *Corresponding author:* m.colbrook@damtp.cam.ac.uk
DAMTP, Centre for Mathematical Sciences, University of Cambridge, UK

However, as discussed below, existing restart schemes either require somewhat restrictive assumptions in which various constants are known, or attain suboptimal convergence rates. This paper overcomes these limitations. We introduce a general restart scheme that applies to a broad class of convex optimization problems, generalizes and improves upon various existing schemes, and leads to optimal complexity bounds for a wide range of problems.

1.1 The problem

We consider the general convex optimization problem

$$\min_{x \in Q} f(x), \tag{1.1}$$

where $f : D \to \mathbb{R}$ is a proper, closed convex function with non-empty effective domain $D \subseteq \mathbb{C}^n$, and $Q \subseteq \mathbb{C}^n$ is a closed, convex set with $Q \subset D$. Let $\hat{f}$ denote the optimal value of (1.1) and $\widehat{X} \subset Q$ denote its set of minimizers, where we assume that $\widehat{X}$ is non-empty.

Our key assumption is that f satisfies the following *approximate sharpness condition*

$$d(x, \widehat{X}) \leq \left(\frac{f(x) - \hat{f} + g_Q(x) + \eta}{\alpha} \right)^{1/\beta}, \qquad \forall x \in D, \tag{1.2}$$

for a metric d on $\mathbb{C}^n$ and some constants $\alpha > 0$, $\beta \geq 1$, $\eta \geq 0$. We slightly abuse notation by defining $d(x, S) := \inf_{z \in S} d(x, z)$ for a set $S \subseteq \mathbb{C}^n$. Here, $g_Q : D \to \mathbb{R}_+$ is a function satisfying

$$g_Q(x) = 0 \iff x \in Q$$

and for any sequence $\{x_m\} \subset D$, $d(x_m, Q) \to 0$ implies $g(x_m) \to 0$. In this paper, we assume that the function g_Q is known, but that the constants η, α and β (or a subset thereof) are unknown. We refer to g_Q as the *feasibility gap* function and $f - \hat{f}$ as the *objective (function) error*.

To formulate a restart scheme that accelerates an optimization algorithm solving (1.1), we assume that f satisfies (1.2), and that we have access to an optimization algorithm $\Gamma : \mathbb{R}_{++} \times \mathbb{R}_{++} \times D \to D$ that defines a map $(\delta, \epsilon, x_0) \mapsto x$, with the property that

$$d(x_0, \widehat{X}) \leq \delta \quad \implies \quad f(x) - \hat{f} + g_Q(x) \leq \epsilon, \text{ where } x = \Gamma(\delta, \epsilon, x_0). \tag{1.3}$$

In essence, for an initial value x_0 within distance δ of an optimal solution, the algorithm produces an output x that is ϵ-suboptimal, i.e., $f(x) - \hat{f} \leq \epsilon$, and ϵ-feasible, i.e., $g_Q(x) \leq \epsilon$, for (1.1). Assumption (1.3) is a generic condition that appear in typical convergence analysis of first-order methods. In Section 4, we describe various examples of first-order optimization methods that yield algorithms satisfying this assumption. See also [66].

The algorithms Γ we consider in this paper are iterative. We define the *cost function* $\mathcal{C}_\Gamma : \mathbb{R}_{++} \times \mathbb{R}_{++} \to \mathbb{N}$, where $\mathcal{C}_\Gamma(\delta, \epsilon)$ represents an upper bound on the number of iterations Γ needs to compute $x = \Gamma(\delta, \epsilon, x_0)$ for any starting value x_0 satisfying $d(x_0, \widehat{X}) \leq \delta$. One can generalize this framework to also consider cost in terms of floating point operations or other measures of time complexity. It is assumed that $\mathcal{C}_\Gamma$ is nondecreasing in its first argument and nonincreasing in its second argument. Examples are given in Section 4 for various first-order methods.

1.2 Motivations

The assumption (1.2) is much weaker than typical assumptions for acceleration, such as strong convexity. It can be considered an *approximate* version of the sharpness condition considered in [69]

(see (1.6)). We discuss its links to other error bounds in Section 1.4. There are two key differences between (1.2) and sharpness. First, we do not assume that the sharpness condition is exact, i.e., we have an additional $\eta \geq 0$ term that controls the approximation. This is very important in many applications and for noisy data, and provides greater *robustness* of our results. For example, when considering sparse recovery, (1.2) covers both *noisy* measurements and *approximately* sparse vectors [27], which is more realistic than *exact* sparse recovery from *noiseless* measurements. Second, we do not require iterates of our algorithm to be feasible, and this is captured by the additional feasibility gap function g_Q. This adds further flexibility and efficiency when selecting the first-order method for the restart scheme (e.g., the primal-dual algorithm considered in Section 4.5).

The other key motivation for this work is that we do not assume knowledge of the constants α, β, and η. When these parameters are known, it is relatively straightforward to derive a restart scheme. However, the constants are rarely known in practice. For example, sharpness holds for general subanalytic convex functions [17], but the proof of this result uses topological arguments that are far from constructive. As another example, in a sparse recovery problem, η depends on the noise level and the sparsity level of the unknown vector, neither of which are typically known. In some applications, one may have bounds for one or more of these constants. Nevertheless, if such bounds are loose – for instance, global bounds may be highly pessimistic near minimizers – this can lead to inefficient schemes. Our method obviates the need for such bounds. However, it also allows the user to input such prior information (e.g., exact values of or ranges for the constants) if these are available.

1.3 Contributions

The following theorem, which follows directly from the results presented in Section 3, summarizes our main convergence rates result.

Theorem 1.1. *Let α, β and η be (unknown) approximate sharpness constants of f in (1.2). Consider Algorithm 2 for fixed $a, b > 1$, $r < 1$, $\alpha_0 > 0$, $\beta_0 \geq 1$ and the choices of schedule criterion and assignment functions described in Section 3.2. Then running Algorithm 2 with*

$$t \gtrsim K(\varepsilon), \quad \varepsilon \to 0^+,$$

(total inner) iterations, where $K(\varepsilon)$ is given in (3.3), implies that

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}.$$

Let $\beta_* = b^{\lceil \log_b(\beta/\beta_0) \rceil}\beta_0$. If, in addition, $\mathcal{C}_\Gamma$ satisfies

$$\mathcal{C}_\Gamma(\delta, \epsilon) \leq C\delta^{d_1}/\epsilon^{d_2} + 1, \qquad C, d_1, d_2 > 0, \tag{1.4}$$

for all $\delta, \epsilon > 0$, then we have

$$K(\varepsilon) \leq \hat{C} \begin{cases} \epsilon_0^{d_1/\beta_* - d_2} \lceil \log(\epsilon_0/\varepsilon) \rceil, & \text{if } d_2 \leq d_1/\beta_*, \\ \varepsilon^{d_1/\beta_* - d_2} \lceil \log(\epsilon_0/\varepsilon) \rceil, & \text{if } d_2 > d_1/\beta_*, \end{cases} \tag{1.5}$$

where $\hat{C}$ is independent of ε (but depends on $r, a, b, \alpha, \beta_*, \alpha_0, \beta_0, d_1$ and d_2). Explicit forms for $\hat{C}$ in (1.5) are given in Section 3.

A few comments are in order. First, note that ε is not a parameter of the algorithm: it is only used to describe the algorithm's behavior as the number of iterations increases. Second, it is possible

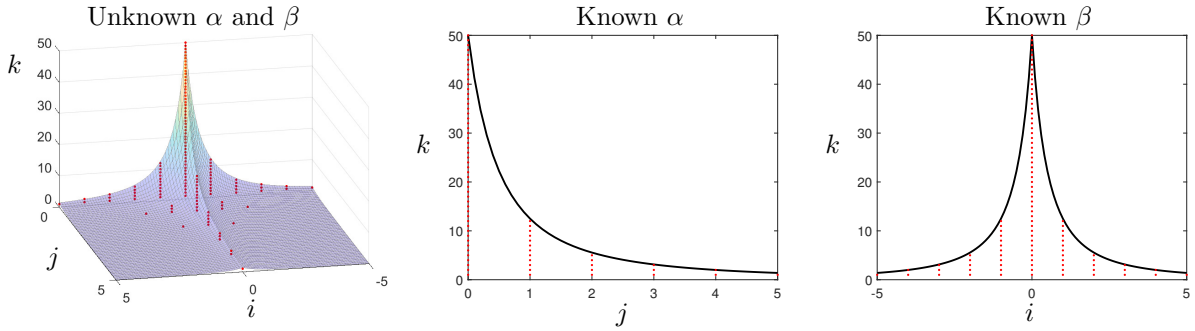

Figure 1: Level curves of $h = 50$ for the schedule criterion functions h in Corollary 3.3 (left panel), Corollary 3.4 (middle panel) and Corollary 3.5 (right panel) with $c_1 = c_2 = 2$. The level curves describe the search order. The red dots show the corresponding indices (i, j, k) in the set defined in (3.4). The index i indicates the parameter search value $a^i\alpha_0$ for α. The index j indicates the parameter search value $b^j\beta_0$ for β. The height (i.e., k) indicates the total number of inner iterations for a fixed (i, j).

for a problem (1.1) to satisfy the approximate sharpness condition (1.2) for different parameters α, β and η, which may give different convergence rates and constant $\hat{C}$ in (1.5). If so, Theorem 1.1 says that for a given accuracy threshold $\varepsilon \geq \eta$, we can take the best rate of convergence/iteration bound over different approximate sharpness constants. Third, Theorem 1.1 does not guarantee a decrease of the objective function error below η as $\varepsilon \to 0^+$. This is quite reasonable in practice. For example, in the case of sparse recovery from noisy measurements, η is the magnitude of the noise level. Therefore there is little benefit in decreasing the objective function error below η, since the error in the recovered vector will generally be $\mathcal{O}(\eta)$. Fourth, the assumption in (1.4) is generic for convergence rates of first-order methods. We present some examples in Section 4. The $+1$ term is included in (1.4) since we often have a bound of the form

$$\mathcal{C}_\Gamma(\delta, \epsilon) \leq \left\lceil C\delta^{d_1}/\epsilon^{d_2} \right\rceil.$$

Finally, the parameters $\alpha_0 > 0$ and $\beta_0 \geq 1$ in Algorithm 2 are estimates for the true α, β. If no estimates are known, we can set $\alpha_0 = \beta_0 = 1$. We also include the case that either or both of α and β are known in our analysis (see the Corollaries in Section 3.2). The parameter $r \in (0, 1)$ is a scale factor that adjusts the parameters of the first-order method at each restart. As we discuss in Section 2, a good choice is $r = \mathrm{e}^{-1/d_2}$.

Our scheme performs a grid search over parameters α, β using the bases $a, b > 1$. The order of the search is based on a so-called *schedule criterion* (see Definition 3.1 and Fig. 1). This new idea allows flexibility depending on which parameters are known and which are unknown, and leads to a unified framework for proving convergence results (e.g., using Theorem 3.2). We postpone the details until Section 3, but, in particular, this new framework allows us to search over a nonuniform grid (Corollary 3.3) that searches more in iteration space as opposed to parameter index space (see left panel of Fig. 1). This is key to developing a search method for unknown parameters that does not suffer from reduced convergence rates.

Suppose now that $\eta \lesssim \varepsilon$. When Algorithm 2 is applied with a suitable first-order method, it leads to optimal[1] complexity bounds for a wide range of different convex optimization problems, *without knowledge of α and β*. Table 1 summarizes some of these bounds and the following correspond to an example for each row:

[1]By optimal, we mean optimal in the number of oracle calls to f, its gradient (where appropriate) or suitable proximal maps. For the first-order methods we discuss, this number will always be bounded by a small multiple of the number of iterations.

Objective function class/structure	Asymptotic bound for $K(\varepsilon)$	Example method
$L-$smooth See Definition 4.2 (NB: must have $\beta \geq 2$)	$\beta = 2$: $\sqrt{L/\alpha} \cdot \log(1/\varepsilon)$ $\beta > 2$: $\frac{\sqrt{L}}{\alpha^{1/\beta_*}} \cdot \frac{1}{\varepsilon^{1/2-1/\beta_*}}$	Nesterov's method $d_1 = 1$, $d_2 = 1/2$ See Section 4.1
$(u, v)-$smoothable See Definition 4.5	$\beta = 1$: $\frac{\sqrt{ab}}{\alpha} \cdot \log(1/\varepsilon)$ $\beta > 1$: $\frac{\sqrt{ab}}{\alpha^{1/\beta_*}} \cdot \frac{1}{\varepsilon^{1-1/\beta_*}}$	Nesterov's method with smoothing $d_1 = 1$, $d_2 = 1$ See Section 4.2
Hölder smooth, parameter $\nu \in [0, 1]$ See Definition 4.8 (NB: must have $\beta \geq 1 + \nu$)	$\beta = 1+\nu$: $\frac{M_\nu^{\frac{2}{1+3\nu}}}{\alpha^{\frac{2}{(1+3\nu)}}} \cdot \log(1/\varepsilon)$ $\beta > 1+\nu$: $\frac{M_\nu^{\frac{2}{1+3\nu}}}{\alpha^{\frac{2+2\nu}{\beta_*(1+3\nu)}}} \cdot \frac{1}{\varepsilon^{\frac{2(\beta_*-1-\nu)}{\beta_*(1+3\nu)}}}$	Universal fast gradient method $d_1 = (2 + 2\nu)/(1 + 3\nu)$ $d_2 = 2/(1 + 3\nu)$ See Section 4.3
$f(x)=q(x)+g(x)+h(Bx)$, q is L_q-smooth, $\sup_{z\in\mathrm{dom}(h)} \inf_{y\in\partial h(z)} \|y\| \leq L_h$, $\|B\| \leq L_B$	$\beta = 1$: $\frac{L_B L_h + L_q}{\alpha} \cdot \log(1/\varepsilon)$ $\beta > 1$: $\frac{L_B L_h + L_q}{\alpha^{1/\beta_*}} \cdot \frac{1}{\varepsilon^{1-1/\beta_*}}$	Primal-dual algorithm $d_1 = 1$, $d_2 = 1$ See Section 4.4
$f(x)=q(x)+g(x)+h(Bx)$, q is L_q-smooth, $\sup_{z\in\mathrm{dom}(h)} \inf_{y\in\partial h(z)} \|y\| \leq L_h$, $\|A\| \leq L_A$, $\|B\| \leq L_B$, $Q=\{x : Ax \in C\}$, $g_Q(x)=\kappa \inf_{z\in C}\|Ax - z\|$	$\beta = 1$: $\frac{\kappa L_A + L_B L_h + L_q}{\alpha} \cdot \log(1/\varepsilon)$ $\beta > 1$: $\frac{\kappa L_A + L_B L_h + L_q}{\alpha^{1/\beta_*}} \cdot \frac{1}{\varepsilon^{1-1/\beta_*}}$	Primal-dual algorithm with constraints $d_1 = 1$, $d_2 = 1$ See Section 4.5

Table 1: Asymptotic cost bounds (as $\varepsilon \downarrow 0$ for $\eta \lesssim \varepsilon$) and suitable first-order methods for Algorithm 2 when applied to different classes of objective functions. Note that whenever the bound is a polynomial in $\log(1/\varepsilon)$, we have $\beta_* = \beta$.

- For L-smooth functions (Definition 4.2) with $\beta = 2$, a well-known lower bound for the subclass of strongly convex smooth functions is $\mathcal{O}(\sqrt{L/\alpha}\log(1/\varepsilon))$ [54]. If $\beta > 2$ then the optimal lower bound is $\mathcal{O}(\sqrt{L}\alpha^{-1/\beta}/\varepsilon^{1/2-1/\beta})$ [53, page 26]. In both cases, we achieve these optimal bounds with our algorithm using, for example, Nesterov's method. This is an improvement (by at least a factor of $\log(1/\varepsilon)$) over the restart scheme presented in [66].

- Suppose that the objective function f is L_f-Lipschitz and has linear growth. Such functions are $(1, L_f^2/2)$-smoothable (Definition 4.5). When $\beta = 1$, the combination of our algorithm and Nesterov's method with smoothing has complexity $\mathcal{O}\left(\log(1/\varepsilon)\right)$. This is an improvement over the restart scheme presented in [66], which has complexity $\mathcal{O}\left(\log^2(1/\varepsilon)\right)$ for such functions. Similarly, for general (u, v)-smoothable objective functions, we improve (by at least a factor of $\log(1/\varepsilon)$) on the results over the restart scheme presented in [66].

- For Hölder smooth functions (see Definition 4.8), the bound in Table 1 matches (with β replaced by β_*) the optimal bound from [53, page 26]. This is an improvement (by at least a factor of $\log(1/\varepsilon)$) over the restart scheme presented in [66].

- There is little work on optimal rates for saddle point problems, a challenge being that there are different measures of error (see [70]). Hence we cannot claim that the final two rows of

Table 1 yield optimal rates. Nevertheless, they yield significantly faster convergence rates than unrestarted first-order methods for saddle point problems.

Finally, it is worth pointing out two straightforward generalization of the assumptions in Section 1.1 where our algorithms and results also hold.

First, the approximate sharpness condition (1.2) can be further generalized to consider any fixed set $Y \subseteq D$ as opposed to $\widehat{X}$. This is expressed as

$$d(x, y) \leq \left(\frac{f(x) - f(y) + g_Q(x) + \eta}{\alpha} \right)^{1/\beta}, \qquad \forall x \in D,\ y \in Y,$$

where α, β, η, and g_Q are defined the same way as in the (1.2). With a suitable generalization of (1.3), much of the work presented here can be extended to this general setting. Note that this is of particular interest whenever the exact minimizer of the associated optimization problem is not desired. In sparse recovery, the ground truth vector being recovered from noisy measurements is often not the minimizer of the associated optimization problem (e.g., see Section 5 or [27]). It is sufficient when the recovered vector's measurements match the original measurements up to a noise level. Similarly, when training overparameterized models in machine learning, e.g., deep neural networks, a balance between training error and generalization error is preferred as opposed to solely minimizing the training error.

Second, our restart procedure for unknown constants always decreases the sum of the objective and feasibility gap functions after each restart. Moreover, we only make use of (1.2) in our analysis each time we restart, so it suffices that we only need (1.2) to hold in the *sublevel* set

$$\{x \in D : f(x) + g_Q(x) \leq f(x_0) + g_Q(x_0)\}$$

for a starting vector $x_0 \in D$.

1.4 Connections with previous work

Recently, there has been a large amount of work on adaptive first-order methods [33, 34, 37, 39, 52, 62, 71]. Adaptive methods seek to learn when to restart a first-order method by trying various values for the method's parameters and observing consequences over a number of iterations. A catalyst for this body of work was provided by Nesterov [57], where he designed an accelerated (line search) method for L-smooth objective functions f (see Section 4.1) with an optimal convergence rate $\mathcal{O}(\sqrt{L/\varepsilon})$ without needing L as an input. In the same paper, Nesterov considered strongly convex objective functions with a grid search for approximating the strong convexity parameter. By narrowing the class of objective functions, this led to an adaptive method with a dramatically improved convergence rate ($\mathcal{O}(\log(1/\varepsilon))$ vs. $\mathcal{O}(1/\sqrt{\varepsilon})$), even without having to know the Lipschitz constant or strong convexity parameter.

The complexity of first-order methods is usually controlled by smoothness assumptions on the objective function, such as Lipschitz continuity of its gradient. Additional assumptions on the objective function such as strong and uniform convexity provide, respectively, linear and faster polynomial rates of convergence [55]. Restart schemes for strongly convex or uniformly convex functions have been studied in [44, 49, 53, 57]. However, strong or uniform convexity is often too restrictive an assumption in many applications.

An assumption more general than strong or uniform convexity is sharpness:

$$d(x, \widehat{X}) \leq \left(\frac{f(x) - \hat{f}}{\alpha} \right)^{1/\beta}, \qquad \forall x \in Q, \tag{1.6}$$

also known as a Hölderian growth/error bound or a Łojasiewicz-type inequality. For example, Nemirovskii and Nesterov [53] linked a "strict minimum" condition similar to (1.6) (with known constants) with faster convergence rates using restart schemes for smooth objective functions. For further use of Łojasiewicz-type inequalities for first-order methods, see [7, 18, 19, 36, 45]. Hölderian error bounds were first introduced by Hoffman [43] to study systems of linear inequalities, and extended to convex optimization in [8, 20, 21, 51, 67]. Łojasiewicz showed that (1.6) holds generically for real analytic and subanalytic functions [50], and Bolte, Daniilidis, and Lewis extended this result to nonsmooth subanalytic convex functions [17]. However, the proofs of these results use topological arguments that are far from constructive. Hence, without further case-by-case analysis of problems and outside of some particular cases (e.g., strong convexity), we cannot assume that suitable constants in (1.6) are known.

An example of (1.6) for $\beta = 1$ was considered in [68] (see also [16]), where the authors use a restarted NESTA algorithm [12] for the exact recovery of sparse vectors from noiseless measurements. The approximate sharpness condition (1.2) was first considered in [27] for the case of $\beta = 1$, and known α and η, to allow the recovery of approximately sparse vectors from noisy measurements and further related examples. Here the parameter $\eta > 0$ is crucial, both in practice and to allow analysis. See also [1, 61]. Though similar to the sharpness condition in (1.6), our more general assumption in (1.2) differs in two important ways, discussed above. First, we do not assume that the sharpness condition is exact ($\eta > 0$), and, second, we do not require iterates of our algorithm to be feasible (the function g_Q). It is also important to re-emphasize that, in this paper, we do not assume that the approximate sharpness constants are known.

The η term in (1.2) is expected and natural. For example, in [28] it was shown that there are well-conditioned recovery problems for which stable and accurate neural networks exist, but no training algorithm can obtain them. The existence of a training algorithm depends on the amount/type of training data and the accuracy required. However, under certain conditions, one can train an appropriate neural network: [28] links trainability to a special case of (1.2), and links the accuracy possible via training to the corresponding η term. In the setting of inexact input, the noise parameter appears as a limitation on the ability of an algorithm [9]. These phenomena occur even if the algorithm is only expected to work on a restricted class of inputs that are 'nice' or 'natural' for the problem under consideration. The results of [9, 28] lead to the phenomenon of generalized hardness of approximation (see also [38]), where it is possible to obtain solutions up to some threshold, but beyond that threshold it becomes impossible. This threshold is strongly related to η in the standard cases.

Most restart schemes are designed for a narrow family of first-order methods, and typically rely on learning approximations of the parameter values characterizing functions in a particular class (e.g., learning the Lipschitz constant L when f is assumed to be L-smooth, or the constants α and β in (1.6)). There are two notable exceptions related to the present paper. First, Roulet and d'Aspremont [69] consider all f possessing sharpness, and having Hölder continuous gradient with exponent $0 < \nu \leq 1$. The restart schemes of [69] result in optimal complexity bounds when particular algorithms are employed in the schemes, assuming scheme parameters are set to appropriate values that, however, are generally unknown in practice. However, for smooth f (i.e., $\nu = 1$), [69] develops an adaptive grid search procedure within the scheme to accurately approximate the required values, leading to an overall complexity that is optimal up to logarithmic factors. Second, Renegar and Grimmer [66] provide a simple scheme for restarting (generic) first-order methods. Multiple instances are run that communicate their improvements in objective value to one another, possibly triggering restarts. Their restart scheme only depends on how much the objective value has been decreased and does not attempt to learn parameter values. The scheme in [66] leads to nearly optimal complexity bounds for quite general classes of functions. This method

Notation	Meaning
f	Proper convex function
D	Effective domain of f
Q	Closed, convex subset of $\mathbb{R}^n$ or $\mathbb{C}^n$
g_Q	Sharpness feasibility gap function, identically zero on Q
$\hat{f}$	Minimum value of objective function over Q
$\widehat{X}$	Set of minimizers of f
d	Metric on $\mathbb{R}^n$ or $\mathbb{C}^n$
η	Sharpness gap constant
α	Sharpness scaling constant
β	Sharpness exponentiation constant
δ	Distance bound between initial point to optimum points
ε	Bound on sum of objective function error and feasibility gap
ϵ_j	Sum of objective function error and feasibility gap at jth restart initial point
Γ	Optimization algorithm
$\mathcal{C}_\Gamma$	Cost function that outputs the number of iterates
ϕ	Mapping of current algorithm step to parameter subscripts (i, j, k)
h	Function defining classes of maps ϕ as abstract execution order of restart scheme
χ_C	Indicator function of a set C ($\chi_C(x) = 0$ if $x \in C$, $\chi_C(x) = \infty$ otherwise)
$\|\cdot\|$	Unless otherwise stated, the Euclidean norm on $\mathbb{C}^n$ or the induced 2-norm on $\mathbb{C}^{m\times n}$
$\langle\cdot,\cdot\rangle$	Unless otherwise stated, the Euclidean inner product on $\mathbb{C}^n$
$\langle\cdot,\cdot\rangle_{\mathbb{R}}$	Unless otherwise stated, $\langle x, y\rangle_{\mathbb{R}} = \mathrm{Re}\,(\langle x, y\rangle)$ for $x, y \in \mathbb{C}^n$
$\mathbb{R}_+$	Non-negative real numbers
$\mathbb{R}_{++}$	Positive real numbers
$\mathbb{N}_0$	Non-negative integers $\{0\} \cup \mathbb{N}$

Table 2: Notation used throughout the paper.

differs quite significantly from ours in that it does not assume an underlying sharpness condition (1.6) (although such a condition is used in the analysis to obtain explicit complexity bounds). However, as observed previously, by assuming (1.2) we are able to obtain better and essentially optimal rates that avoid additional factors of $\log(1/\varepsilon)$. Moreover, in contrast to [66], our method is independent of the total number of iterations, and we do not need to specify the total number of iterations in advance. Further, we also address the practical case of approximate sharpness and allow the case of infeasible iterates (the convergence analysis of [66] relies on $\eta = 0$ and that iterates are feasible).

1.5 Notation and outline

For ease of reference, Section 1.5 outlines the notation used throughout the paper. The remainder of this paper is organized as follows. In Section 2, we introduce a restart scheme in the case where η is unknown, but α and β are known. This transpires to be significantly simpler than the general case. Next, in Section 3 we introduce and analyze the full restart scheme when all three constants are potentially unknown. In Section 4, we apply this restart scheme to different problems with various first-order methods, leading, in particular, to the results described in Table 1. Next, in Section 5 we present a series of numerical experiments illustrating the restart schemes in different applications. Finally, we end in Section 6 with conclusions and open problems.

Algorithm 1: Restart scheme for unknown η.

Input : Optimization algorithm Γ for (1.1), initial vector $x_0 \in D$, upper bound ϵ_0 such that $f(x_0) - \hat{f} + g_Q(x_0) \leq \epsilon_0$, constants $\alpha > 0$ and $\beta \geq 1$ such that (1.2) holds (for possibly unknown $\eta \geq 0$), $r \in (0,1)$, and number of restart iterations $t \in \mathbb{N}$.

Output: Final iterate x_t approximating a solution to (1.1)

for $k = 0, 1, \ldots, t-1$ **do**
- $\epsilon_{k+1} \leftarrow r\epsilon_k$;
- $\delta_{k+1} \leftarrow \left(\frac{2\epsilon_k}{\alpha}\right)^{1/\beta}$;
- $z \leftarrow \Gamma\left(\delta_{k+1}, \epsilon_{k+1}, x_k\right)$;
- $x_{k+1} \leftarrow \operatorname{argmin}\left\{f(x) + g_Q(x) : x = x_k \text{ or } x = z\right\}$;

end

2 Restart scheme for unknown η but known α and β

To formulate a restart scheme within the setup of Section 1.1, observe that the approximate sharpness condition (1.2) relates $d(x, \widehat{X})$ to the objective function error $f(x) - \hat{f}$ and feasibility gap $g_Q(x)$. The upper bound in the approximate sharpness condition can be used as δ for the algorithm Γ, and ϵ set as a rescaling of the previous sum of objective error and feasibility gap $f(x) - \hat{f} + g_Q(x)$. However, in practice, we may not know the exact values of the objective error $f(x) - \hat{f}$ and feasibility gap $g_Q(x)$. It is, instead, enough to know upper bounds for these quantities.

We first consider the case where α, β are known, but η is unknown. This simpler case provides insight into the solution of the full problem considered in Section 3. We define a restart scheme under this assumption in Algorithm 1. Using (1.2) and Γ, it is easy to see inductively that for any t with $\epsilon_t \geq \eta$, Algorithm 1 produces iterates $x_0, x_1, \ldots, x_t \in D$ that satisfy

$$\begin{aligned} &f(x_k) - \hat{f} + g_Q(x_k) \leq \epsilon_k, \\ &d(x_k, \widehat{X}) \leq \left(\frac{f(x_k) - \hat{f} + g_Q(x_k) + \eta}{\alpha}\right)^{1/\beta} \leq \left(\frac{\epsilon_k + \eta}{\alpha}\right)^{1/\beta} \leq \left(\frac{2\epsilon_k}{\alpha}\right)^{1/\beta}, \qquad 0 \leq k \leq t. \end{aligned} \tag{2.1}$$

In addition, the total number of inner iterations used in Algorithm 1 is at most

$$\sum_{k=0}^{t-1} \mathcal{C}_\Gamma\left(\left(\frac{2\epsilon_k}{\alpha}\right)^{1/\beta}, \epsilon_{k+1}\right).$$

Under further assumptions about the function $\mathcal{C}_\Gamma$, we can show that the iterates produced by the restart scheme yield linear (if $d_2 = d_1\beta$) or fast algebraic (if $d_2 > d_1\beta$) decay of $f(x_k) - \hat{f} + g_Q(x_k)$ in k down to a finite tolerance proportional to η. Hence, this property holds for both the objective error $f(x_k) - \hat{f}$ and feasibility gap $g_Q(x_k)$. We state and prove this in the following theorem. Note that these additional assumptions are not arbitrary and will appear in our examples later.

Theorem 2.1. *Consider Algorithm 1 and its corresponding inputs. For any $\varepsilon \in (0, \epsilon_0)$, if we run Algorithm 1 with $t \geq \lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil$, then*

$$f(x_t) - \hat{f} + g_Q(x_t) \leq \max\{\eta, \varepsilon\}. \tag{2.2}$$

Suppose, in addition, that for all $\delta, \epsilon > 0$, $\mathcal{C}_\Gamma$ satisfies

$$\mathcal{C}_\Gamma(\delta, \epsilon) \leq C\delta^{d_1}/\epsilon^{d_2} + 1, \qquad C, d_1, d_2 > 0.$$

Then the total number of iterations of Γ needed to compute an x_t with (2.2) is at most

$$\left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}} \cdot \begin{cases} \frac{1-r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r)\rceil |d_2-d_1/\beta|}}{1-r^{|d_2-d_1/\beta|}} \cdot \frac{1}{\epsilon_0^{d_2-d_1/\beta}}, & \text{if } d_2 < d_1/\beta, \\ \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil, & \text{if } d_2 = d_1/\beta, \\ \frac{1-r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r)\rceil |d_2-d_1/\beta|}}{1-r^{|d_2-d_1/\beta|}} \cdot \frac{1}{\varepsilon^{d_2-d_1/\beta}}, & \text{if } d_2 > d_1/\beta. \end{cases} \tag{2.3}$$

Note that the cases in (2.3) match in the limit $d_2 - d_1/\beta \to 0$.

Proof of Theorem 2.1. The statement of the theorem is unchanged if we assume that $\varepsilon \geq \eta$. Hence, we may assume without loss of generality that $\varepsilon \geq \eta$. Let $s = \lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil$, then $\epsilon_{s-1} = r^{s-1}\epsilon_0 \geq \varepsilon \geq \eta$. It follows that we are in the regime where (2.1) holds and hence

$$f(x_{s-1}) - \hat{f} + g_Q(x_{s-1}) \leq \epsilon_{s-1}, \qquad d(x_{s-1}, \widehat{X}) \leq \left(\frac{2\epsilon_{s-1}}{\alpha}\right)^{1/\beta}.$$

Then by line 4 of Algorithm 1 and the choice of s, we have

$$f(z) - \hat{f} + g_Q(z) \leq \epsilon_s \leq \varepsilon, \quad z = \Gamma(\delta_s, \epsilon_s, x_{s-1}).$$

Due to the argmin taken in Algorithm 1, (2.2) follows. The total number of iterations, T, needed to reach such an x_s is bounded by

$$T \leq \sum_{k=0}^{s-1} \mathcal{C}_\Gamma\left(\left(\frac{2\epsilon_k}{\alpha}\right)^{1/\beta}, \epsilon_{k+1}\right) \leq s + C\sum_{k=0}^{s-1} \frac{(2\epsilon_k)^{d_1/\beta}}{\alpha^{d_1/\beta}\epsilon_{k+1}^{d_2}} = s + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}} \sum_{k=0}^{s-1} \frac{1}{\epsilon_k^{d_2-d_1/\beta}}.$$

In the case that $d_2 = d_1/\beta$, then $\epsilon_k^{d_2-d_1/\beta} = 1$ and we obtain

$$T \leq s + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}} s = \left(1 + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}}\right) \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil.$$

If $d_2 \neq d_1/\beta$, we use that $\epsilon_k = r^k\epsilon_0$ and sum the geometric series to obtain

$$T \leq \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}} \frac{1 - r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r)\rceil (d_1/\beta - d_2)}}{1 - r^{d_1/\beta - d_2}} \frac{1}{\epsilon_0^{d_2-d_1/\beta}}. \tag{2.4}$$

If $d_2 > d_1/\beta$, then since $\epsilon_0 \geq \varepsilon/r^{s-1}$, we have $\epsilon_0^{d_2-d_1/\beta} \geq \varepsilon^{d_2-d_1/\beta} r^{d_2-d_1/\beta}/r^{s(d_2-d_1/\beta)}$. Substituting this into (2.4) and rearranging yields

$$T \leq \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \frac{C2^{d_1/\beta}}{\alpha^{d_1/\beta} r^{d_2}} \frac{1 - r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r)\rceil (d_2 - d_1/\beta)}}{1 - r^{d_2 - d_1/\beta}} \frac{1}{\varepsilon^{d_2-d_1/\beta}}.$$

The result follows by considering the three separate cases in (2.3). □

Remark 2.2 (How to choose r)**.** *Suppose that $d_2 = d_1/\beta$ and that*

$$\left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil \leq 2\frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)}.$$

Using this new bound instead, the total number of iterations T performed by Γ is bounded by

$$T \leq \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \frac{C2^{d_1/\beta+1}}{\alpha^{d_1/\beta}} \log(\epsilon_0/\varepsilon) \frac{r^{-d_2}}{\log(1/r)}.$$

Hence T is bounded by an ε-dependent constant times $r^{-d_2}/\log(1/r)$, which can be minimized analytically by choosing $r = \mathrm{e}^{-1/d_2}$. Note that the optimal r here does not depend on the approximate sharpness constants. Therefore, one has

$$T \leq \lceil d_2 \log(\epsilon_0/\varepsilon) \rceil + \frac{C\mathrm{e}d_2 2^{d_1/\beta+1}}{\alpha^{d_1/\beta}} \log(\epsilon_0/\varepsilon)$$

This is meaningful in terms of choosing one less parameter, namely r for Algorithm 1.

An optimal value of r can also be found for the case $d_2 > d_1/\beta$. However, this optimal value depends on ε in a complicated manner. In the limit $\varepsilon \downarrow 0$, the optimal choice is

$$r = \left(\frac{d_2}{2d_2 - d_1/\beta} \right)^{\frac{1}{d_2 - d_1/\beta}},$$

which does depend on the sharpness constant β. As $d_2 - d_1/\beta \downarrow 0$, this choice converges to the choice $r = \mathrm{e}^{-1/d_2}$, that is obtained when $d_2 = d_1/\beta$. Similarly, if $d_2 < d_1/\beta$, then the optimal choice depends on ε in a complicated manner but converges to the choice $r = \mathrm{e}^{-1/d_2}$ as $d_2 - d_1/\beta \uparrow 0$.

In any of these cases, the same argument for optimal r applies to the algorithms in Section 3. In the case that β is unknown, we recommend the choice $r = \mathrm{e}^{-1/d_2}$. ♦

3 Restart scheme for unknown α, β and η

In the event that the constants α, β of (1.2) are unknown, we introduce a logarithmic grid search on each of α and β, running multiple instances of Algorithm 1, and aggregating results that minimize the objective error and feasibility gap. Even if suitable *global* α and β are known, the following algorithm is useful since it also takes advantage of sharper versions of (1.2) that only hold *locally* around optimal points.

3.1 The algorithm

To introduce the algorithm, we need some additional notation and definitions. This will allow us to define a new general scheme for logarithmic grid searches, with examples given in Section 3.2.

Definition 3.1. Consider an infinite subset $S \subseteq \mathbb{Z} \times \mathbb{N}_0 \times \mathbb{N}$. Let $h : \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_{++} \to \mathbb{R}_{++}$ be a function that is nondecreasing in its first and second arguments, and strictly increasing in its third argument. We call such an h a *schedule criterion function*, or simply a *schedule criterion*. Given a schedule criterion h, an *h-assignment over S* is a bijection $\phi : \mathbb{N} \to S$ satisfying

$$h(|i'|, j', k') \leq h(|i|, j, k) \quad \iff \quad \phi^{-1}(i', j', k') \leq \phi^{-1}(i, j, k), \tag{3.1}$$

for all $(i, j, k), (i', j', k') \in S$. ▲

Let $a, b > 1$ be constants. Our algorithm employs logarithmic search grids for the unknown parameters α and β. Specifically, we consider the values $\alpha_i = a^i \alpha_0$ for $i \in \mathbb{Z}$ and $\beta_j = b^j \beta_0$ for $j \in \mathbb{N}_0$, where we assume that α_0, β_0 are additional inputs with $\alpha_0 > 0$ and $\beta \geq \beta_0 \geq 1$. In essence, our algorithm applies the restart scheme described in Algorithm 1 with the values α_i and β_j for each i and j. However, it does so according to a particular schedule, specified by the functions h and ϕ. The schedule criterion and assignment together control the execution order of Algorithm 1 instances for each i and j. Note that the lower bound β_0 in the definition of the β_j is to capture additional knowledge that may be available (see, e.g., the examples in Section 4), and may be set

Algorithm 2: Restart scheme for unknown α, β and η in (1.2) via grid search.

Input : Optimization algorithm Γ for (1.1), bijection ϕ as in Definition 3.1, initial vector $x^{(0)} \in D$, upper bound ϵ_0 such that $f(x^{(0)}) - \hat{f} + g_Q(x^{(0)}) \leq \epsilon_0$, constants $a, b > 1$, $r \in (0,1)$, $\alpha_0 > 0$, $\beta_0 \geq 1$ and total number of inner iterations $t \in \mathbb{N}$.

Output: Final iterate $x^{(t)}$ approximating a solution to (1.1).

1 Initialize $x^{(0)} = x_0$, $U_{i,j} = 0$, $V_{i,j} = 0$, $\epsilon_{i,j,0} = \epsilon_0$ for all $i \in \mathbb{Z}, j \in \mathbb{N}_0$;
2 **for** $m = 0, 1, \ldots, t-1$ **do**
3 $\quad (i,j,k) \leftarrow \phi(m+1)$;
4 $\quad \alpha_i \leftarrow a^i \alpha_0$, $\beta_j \leftarrow b^j \beta_0$, $U \leftarrow U_{i,j}$, $V \leftarrow V_{i,j}$;
5 $\quad \epsilon_{i,j,U+1} \leftarrow r\epsilon_{i,j,U}$;
6 $\quad$ **if** $2\epsilon_{i,j,U} > \alpha_i$ **then**
7 $\quad\quad \delta_{i,j,U+1} \leftarrow \left(\frac{2\epsilon_{i,j,U}}{\alpha_i}\right)^{\min\{b/\beta_j, 1/\beta_0\}}$;
8 $\quad$ **else**
9 $\quad\quad \delta_{i,j,U+1} \leftarrow \left(\frac{2\epsilon_{i,j,U}}{\alpha_i}\right)^{1/\beta_j}$;
10 $\quad$ **end**
11 $\quad$ **if** $V + \mathcal{C}_\Gamma(\delta_{i,j,U+1}, \epsilon_{i,j,U+1}) \leq k$ **then**
12 $\quad\quad z^{(m)} \leftarrow \Gamma\left(\delta_{i,j,U+1}, \epsilon_{i,j,U+1}, x^{(m)}\right)$;
13 $\quad\quad x^{(m+1)} \leftarrow \operatorname{argmin}\left\{f(x) + g_Q(x) : x = z^{(m)} \text{ or } x = x^{(m)}\right\}$;
14 $\quad\quad V_{i,j} \leftarrow V + \mathcal{C}_\Gamma(\delta_{i,j,U+1}, \epsilon_{i,j,U+1})$;
15 $\quad\quad U_{i,j} \leftarrow U + 1$;
16 $\quad$ **else**
17 $\quad\quad x^{(m+1)} = x^{(m)}$;
18 $\quad$ **end**
19 **end**

to 1 if no such knowledge is available. Similarly, the constant α_0 centers the search grid for α and can be set to 1.

Our algorithm is presented in Algorithm 2. It proceeds as follows. At step $m \in \{0, \ldots, t-1\}$ it first applies the bijection ϕ to obtain the tuple $(i,j,k) = \phi(m+1)$. The first two entries give the approximate sharpness parameter values $\alpha_i = a^i\alpha_0$ and $\beta_j = b^j\beta_0$. The final entry k is a counter, which is an upper bound for the total number of iterations used by the algorithm for these parameter values. We also have two further counters associated with each double (i,j). The counter $V_{i,j}$ counts the total number of inner iterations of Γ used by the restart scheme with these parameters. The second counter $U_{i,j}$ counts the number of completed restarts (outer iterations) corresponding to these parameters.

Having obtained a tuple $(i,j,k) = \phi(m+1)$, the algorithm proceeds as follows. First, much as in line 2 of Algorithm 1, it updates the first scaling parameter in line 5. Then, reminiscent of line 3 of Algorithm 1, it updates the other scaling parameter in lines 6-10. This step is more involved, a complication that arises because the true parameter β is unknown.

The next lines, lines 11-16, are similar to lines 4-5 of Algorithm 1. The main difference is the inclusion of the if statement, which is done to control the computational cost. It stipulates that a restart be performed (line 12) if the total cost (including the proposed restart) does not exceed the counter k (line 11). If this is not the case, then no restart is performed, and the algorithm moves on to the next step.

We now present a general result on this algorithm. It relates the total number of inner iterations of Γ used by Algorithm 2 to produce a solution within a desired error to intrinsic properties of the schedule criterion function h. With this in hand, we derive explicit bounds for specific choices of h in Section 3.2.

Theorem 3.2. *Let $S \subseteq \mathbb{Z} \times \mathbb{N}_0 \times \mathbb{N}$ be an infinite subset, h be a schedule criterion, and ϕ an h-assignment over S. Let α, β and η be approximate sharpness constants of f in (1.2). Consider Algorithm 2 for fixed $a, b > 1$. Define the (unknown) indices*

$$I = \lfloor \log_a(\alpha/\alpha_0) \rfloor, \qquad J = \lceil \log_b(\beta/\beta_0) \rceil$$

and the corresponding constants

$$\alpha_* = a^I \alpha_0 \leq \alpha, \qquad \beta_* = b^J \beta_0 \geq \beta.$$

Then for $q \in \mathbb{N}$ we have

$$\delta_{I,J,q} = \left[\max\left\{1, \frac{2r^{q-1}\epsilon_0}{\alpha_*}\right\} \right]^{\min\{b/\beta_*, 1/\beta_0\}} \left[\min\left\{1, \frac{2r^{q-1}\epsilon_0}{\alpha_*}\right\} \right]^{1/\beta_*} \tag{3.2}$$

Now, for any $\varepsilon \in (0, \epsilon_0)$, let

$$K(\varepsilon) := K(\varepsilon, \alpha, \beta, \eta) = \sum_{q=1}^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil} \mathcal{C}_\Gamma\left(\delta_{I,J,q}, r^q \epsilon_0\right) \tag{3.3}$$

and suppose that $(I, J, K(\varepsilon)) \in S$. Then the total number of inner iterations of Γ needed by Algorithm 2 to compute $x^{(t)}$ with

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\},$$

is bounded by the cardinality of the set

$$\left\{(i', j', k') \in S : h(|i'|, j', l') \leq h\left(|I|, J, K(\varepsilon)\right)\right\}. \tag{3.4}$$

In addition, if $\mathcal{C}_\Gamma$ satisfies

$$\mathcal{C}_\Gamma(\delta, \epsilon) \leq C\delta^{d_1}/\epsilon^{d_2} + 1, \qquad C, d_1, d_2 > 0, \tag{3.5}$$

for all $\delta, \epsilon > 0$, then we have

$$K(\varepsilon) \leq \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \max\left\{ \left(\frac{2\epsilon_0}{\alpha_*}\right)^{d_1 \min\left\{\frac{b-1}{\beta_*}, \frac{1}{\beta_0} - \frac{1}{\beta_*}\right\}}, 1 \right\} \times$$
$$\frac{C 2^{d_1/\beta_*}}{\alpha_*^{d_1/\beta_*} r^{d_2}} \cdot \begin{cases} \frac{1 - r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil |d_2 - d_1/\beta_*|}}{1 - r^{|d_2 - d_1/\beta_*|}} \cdot \frac{1}{\epsilon_0^{d_2 - d_1/\beta_*}}, & \text{if } d_2 < d_1/\beta_*, \\ \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil, & \text{if } d_2 = d_1/\beta_*, \\ \frac{1 - r^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil |d_2 - d_1/\beta_*|}}{1 - r^{|d_2 - d_1/\beta_*|}} \cdot \frac{1}{\varepsilon^{d_2 - d_1/\beta_*}}, & \text{if } d_2 > d_1/\beta_*. \end{cases} \tag{3.6}$$

Proof. Since $\epsilon_{i,j,q-1} = r^{q-1}\epsilon_0$ for all $q \in \mathbb{N}$, (3.2) must hold by considering the two separate cases defining $\delta_{I,J,q}$. Similar to the proof of Theorem 2.1, we may assume without loss of generality that $\varepsilon \geq \eta$. Note that, due to (1.2),

$$d(x, \widehat{X}) \leq \left(\frac{f(x) - \hat{f} + g_Q(x) + \eta}{\alpha_*} \right)^{1/\beta}, \qquad \forall x \in D. \tag{3.7}$$

Now consider the following adapted version of the iterates in Algorithm 1:

for $p = 0, 1, \ldots$ **do**
  $\epsilon_{p+1} \leftarrow r\epsilon_p$;
  if $2\epsilon_p > \alpha_*$ **then**
    $\delta_{p+1} \leftarrow \left(\frac{2\epsilon_p}{\alpha_*}\right)^{\min\{b/\beta_*, 1/\beta_0\}}$;
  else
    $\delta_{p+1} \leftarrow \left(\frac{2\epsilon_p}{\alpha_*}\right)^{1/\beta_*}$;
  end
  $z \leftarrow \Gamma\left(\delta_{p+1}, \epsilon_{p+1}, x_p\right)$;
  $x_{p+1} \leftarrow \operatorname{argmin}\left\{f(x) + g_Q(x) : x = x_p \text{ or } x = z\right\}$;
end

It is easy to see inductively that for any l with $\epsilon_l \geq \eta$ the above produces iterates $\{x_0, x_1, \ldots, x_l\} \subset D$ satisfying

$$f(x_p) - \hat{f} + g_Q(x_p) \leq \epsilon_p, \qquad d(x_p, \widehat{X}) \leq \delta_{p+1}, \qquad 0 \leq p \leq l.$$

The only difference to the previous argument for Algorithm 1 is the use of (3.7), and the fact that

$$\left(\frac{f(x_p) - \hat{f} + g_Q(x_p) + \eta}{\alpha_*} \right)^{1/\beta} \leq \left(\frac{2\epsilon_p}{\alpha_*}\right)^{1/\beta} \leq \begin{cases} \left(\frac{2\epsilon_p}{\alpha_*}\right)^{\min\{b/\beta_*, 1/\beta_0\}}, & \text{if } 2\epsilon_p > \alpha_* \\ \left(\frac{2\epsilon_p}{\alpha_*}\right)^{1/\beta_*}, & \text{otherwise.} \end{cases}$$

Here, we use the fact that $\beta \geq \beta_0$ in the first case.

In Algorithm 2, each $U_{i,j}$ plays the role of the index p in the above iterates (i.e., counting the number of restarts for a fixed (i, j)) and $V_{i,j}$ counts the total number of inner iterations that have been executed by the algorithm Γ for the approximate sharpness constants given by the double index (i, j). The fact that we take minimizers of $f + g_Q$ across different indices does not alter the above inductive argument, since the argument only depends on bounding the value of $f - \hat{f} + g_Q$. Moreover, since h is strictly increasing in its final argument and satisfies (3.1), the counter index k counts successively through $\mathbb{N}$ for any fixed (i, j) as the for loop in Algorithm 2 proceeds. It follows that if $\phi(m+1) = (I, J, k)$, $V_{I,J} + \mathcal{C}_\Gamma\left(\delta_{I,J,U_{I,J}+1}, \epsilon_{I,J,U_{I,J}+1}, x^{(m)}\right) \leq k$ and $\epsilon_{I,J,U_{I,J}} \geq \eta$, then

$$f(x^{(m+1)}) - \hat{f} + g_Q(x^{(m+1)}) \leq \epsilon_{I,J,U_{I,J}+1} = r^{U_{I,J}+1}\epsilon_0. \tag{3.8}$$

Hence, for Algorithm 2 to produce an iterate with

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}, \tag{3.9}$$

it is sufficient to reach an m with $\phi(m+1) = (I, J, k)$ such that

$$k \geq \sum_{q=1}^{\lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil} \mathcal{C}_\Gamma\left(\delta_{I,J,q}, \epsilon_{I,J,q}\right) = K(\varepsilon) \tag{3.10}$$

and execute the resulting restart. To see why this is the case, notice that if k satisfies this inequality, then the number of restart iterations performed by the algorithm for the parameter values (I, J) is at least $\lceil \log(\epsilon_0/\varepsilon)/\log(1/r) \rceil$. Plugging this into (3.8) gives the desired bound (3.9).

Now consider the set in (3.4). Due to (3.1), we notice that this set is equivalent to

$$\{(i', j', k') \in S : \phi^{-1}(i', j', k') \leq m+1\},$$

where $\phi(m+1) = (I, J, K(\varepsilon))$. Notice that if a tuple (i', j', k') belongs to this set, then (i', j', k'') belongs to the set for every $1 \leq k'' \leq k'$. Thus, the number of terms in this set corresponding to the pair (i', j') is precisely the total number of inner iterations performed by the algorithm at the corresponding parameter values up to step m. We immediately deduce that the cardinality of the set (3.4) is a bound for the total number of inner iterations performed by the algorithm across all parameter values up to step m, as required.

To finish the proof, we must show that (3.6) holds under the additional assumption (3.5) on $\mathcal{C}_\Gamma$. Suppose first that $\delta_{I,J,q} > 1$, then

$$\begin{aligned}
\mathcal{C}_\Gamma\left(\delta_{I,J,q}, r^q\epsilon_0\right) &\leq C\left(\frac{2r^{q-1}\epsilon_0}{\alpha_*}\right)^{d_1 \min\{b/\beta_*, 1/\beta_0\}} (r^q\epsilon_0)^{-d_2} + 1 \\
&\leq C\left(\frac{2\epsilon_0}{\alpha_*}\right)^{d_1[\min\{b/\beta_*, 1/\beta_0\} - 1/\beta_*]} \left(\frac{2r^{q-1}\epsilon_0}{\alpha_*}\right)^{d_1/\beta_*} (r^q\epsilon_0)^{-d_2} + 1 \\
&= \frac{C}{r^{d_2}}\left(\frac{2\epsilon_0}{\alpha_*}\right)^{d_1[\min\{b/\beta_*, 1/\beta_0\} - 1/\beta_*]} \left(\frac{2}{\alpha_*}\right)^{d_1/\beta_*} \left(r^{q-1}\epsilon_0\right)^{-d_2 + d_1/\beta_*} + 1.
\end{aligned}$$

Similarly, if $\delta_{I,J,q} \leq 1$, then

$$\mathcal{C}_\Gamma\left(\delta_{I,J,q}, r^q\epsilon_0\right) \leq \frac{C}{r^{d_2}}\left(\frac{2}{\alpha_*}\right)^{d_1/\beta_*} \left(r^{q-1}\epsilon_0\right)^{-d_2 + d_1/\beta_*} + 1.$$

From (3.10), it follows that

$$K(\varepsilon) \leq \left\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \right\rceil + \max\left\{\left(\frac{2\epsilon_0}{\alpha_*}\right)^{d_1[\min\{b/\beta_*, 1/\beta_0\} - 1/\beta_*]}, 1\right\} \cdot \frac{C 2^{d_1/\beta_*}}{\alpha_*^{d_1/\beta_*} r^{d_2}} \cdot \sum_{k=0}^{\lceil \frac{\log(\epsilon_0/\varepsilon)}{\log(1/r)} \rceil - 1} \frac{1}{(r^k\epsilon_0)^{d_2 - d_1/\beta_*}}.$$

We now note that the only difference between this bound for $K(\varepsilon)$ and the bound for T in the proof of Theorem 2.1 is the factor that maximizes over the terms in curly brackets and the replacement of α and β by α_* and β_*, respectively. The result now follows by using the same arguments as in the proof of Theorem 2.1. □

3.2 Choices of schedule criterion functions and assignments

The total number of inner iterations of Γ needed for Algorithm 2 depends on the choice of h and ϕ. We examine some choices and state them as corollaries. Examples are shown in Fig. 1.

Corollary 3.3 (Unknown α and β)**.** *Suppose that $S = \mathbb{Z} \times \mathbb{N}_0 \times \mathbb{N}$ and let*

$$h(x_1, x_2, x_3) = (x_1 + 1)^{c_1}(x_2 + 1)^{c_2} x_3, \quad c_1, c_2 > 1$$

be a schedule criterion with h-assignment ϕ. Then for any $\varepsilon \in (0, \epsilon_0)$, running Algorithm 2 with

$$t \geq 2c_1c_2\tau/[(c_1 - 1)(c_2 - 1)], \quad \tau = (|\lfloor \log_a(\alpha/\alpha_0) \rfloor| + 1)^{c_1}(|\lceil \log_b(\beta/\beta_0) \rceil| + 1)^{c_2} K(\varepsilon),$$

where $K(\varepsilon)$ is as in (3.3), implies that

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}.$$

Proof. It suffices to prove that the stated lower bound on t is an upper bound for the cardinality of the set (3.4) from Theorem 3.2. We do this by finding an upper bound on the number of solutions to $n_1^{c_1} n_2^{c_2} n_3 \leq \tau$ where $n_1, n_2, n_3 \in \mathbb{N}$. By directly counting, the number of solutions is bounded by

$$\sum_{n_1=1}^{\tau^{1/c_1}} \sum_{n_2=1}^{\left(\frac{\tau}{n_1^{c_1}}\right)^{\frac{1}{c_2}}} \frac{\tau}{n_1^{c_1} n_2^{c_2}} \leq \tau \sum_{n_1=1}^{\infty} \frac{1}{n_1^{c_1}} \sum_{n_2=1}^{\infty} \frac{1}{n_2^{c_2}}.$$

We have that

$$\sum_{n_1=1}^{\infty} \frac{1}{n_1^{c_1}} \leq 1 + \int_1^{\infty} \frac{dx}{x^{c_1}} = \frac{c_1}{c_1 - 1}.$$

It follows that the number of solutions is bounded by $\tau c_1 c_2/((c_1-1)(c_2-1))$. Each counted solution (n_1, n_2, n_3) defines *at most* two tuples (i', j', k') in the set (3.4), namely $i' = \pm(n_1 - 1)$, $j' = n_2 - 1$, $k' = n_3$. In reverse, each tuple (i', j', k') of the set (3.4) is always associated with a single solution (n_1, n_2, n_3), namely $n_1 = |i'| + 1$, $n_2 = j' + 1$, $n_3 = k'$. It then follows that that the set (3.4) is bounded by $2\tau c_1 c_2/((c_1 - 1)(c_2 - 1))$. □

We compare the cost in Corollary 3.3 to that of Theorem 2.1 under the assumption (3.5). Let $\hat{K}(\varepsilon)$ be the cost in (2.3).

$$K(\varepsilon) \lesssim \hat{K}(\varepsilon) \begin{cases} 1, & \text{if } \beta = \beta_* \text{ or } d_2 \leq d_1/\beta_*, \\ \frac{1}{\varepsilon^{d_1(1/\beta - 1/\beta_*)}}, & \text{otherwise.} \end{cases} \tag{3.11}$$

It follows that if $\beta = \beta_*$ or $d_2 \leq d_1/\beta_*$, the cost of Algorithm 2 is of the same order as $\hat{K}(\varepsilon)$. If neither of these hold, then the cost of Algorithm 2 is of the order of $\varepsilon^{-d_1(1/\beta - 1/\beta_*)}$ times the cost of Algorithm 1. Note that the order of this extra algebraic dependence can be made arbitrarily small by taking b close to 1, at the expense of a factor in the term τ that grows as $\log_b(\beta/\beta_0)^{c_2}$.

We now consider the cases where either α or β is known.

Corollary 3.4 (Known α)**.** *Suppose that $\alpha = a^i \alpha_0$. Let $S = \{i\} \times \mathbb{N}_0 \times \mathbb{N}$ and $h(x_1, x_2, x_3) = (x_2 + 1)^{c_2} x_3$, $c_2 > 1$, be a schedule criterion. Then given any h-assignment ϕ and any $\varepsilon \in (0, \epsilon_0)$, running Algorithm 2 with*

$$t \geq c_2\tau/(c_2 - 1), \quad \tau = (|\lceil \log_b(\beta/\beta_0) \rceil| + 1)^{c_2} K(\varepsilon),$$

where $K(\varepsilon)$ is as in (3.3), implies that

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}.$$

Proof. The result follows after modifying the proof of Corollary 3.3 as follows. First, find an upper bound to the number of solutions to $n_2^{c_2} n_3 \leq \tau$ for $n_2, n_3 \in \mathbb{N}$. Now find the correspondence between the solutions and the tuples (i', j', k') of (3.4), where i' is now fixed. □

For the case of known β, we alter Algorithm 2 by removing the if statement in line 6 and always using the update rule in line 9.

Corollary 3.5 (Known β)**.** *Suppose that $\beta = \beta_0$ is known, $S = \mathbb{Z} \times \{0\} \times \mathbb{N}$ and $h(x_1, x_2, x_3) = (x_1 + 1)^{c_1} x_3$, $c_1 > 1$, is a schedule criterion. Then given any h-assignment ϕ and any $\varepsilon \in (0, \epsilon_0)$, running Algorithm 2 and*

$$t \geq 2c_1\tau/(c_1 - 1), \quad \tau = (|\lfloor \log_a(\alpha/\alpha_0) \rfloor| + 1)^{c_1} K(\varepsilon),$$

where $K(\varepsilon)$ is as in (3.3), implies that

$$f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}.$$

Proof. Similar to the previous proof, the result follows after modifying the proof of Corollary 3.3. First, find an upper bound to the number of solutions to $n_1^{c_1} n_3 \leq \tau$ for $n_1, n_3 \in \mathbb{N}$. Now find the correspondence between the solutions and the tuples (i', j', k') of (3.4), where j' is now fixed. □

Remark 3.6 (How to choose a, b)**.** *In the case of Corollary 3.5 and assuming* (3.5)*, we can select an optimal value of a. From Corollary 3.5 and $\alpha_* \geq \alpha/a$, the part of τ that depends on a is bounded by $\mathcal{O}((|\lfloor \log_a(\alpha/\alpha_0) \rfloor| + 1)^{c_1} a^{d_1/\beta})$. We can upper bound this further by both dropping the floor function and, then dropping the $+1$ in brackets. We are then led to minimizing*

$$|\log_a(\alpha/\alpha_0)|^{c_1} a^{d_1/\beta} = |\log(\alpha/\alpha_0)|^{c_1} a^{d_1/\beta} / \log(a)^{c_1}.$$

Under these assumptions, the optimal value of a is $\mathrm{e}^{c_1\beta/d_1}$. Note that in the case of Corollary 3.4, there is no clear optimal choice for b since the optimal choice is ε-dependent. ♦

Remark 3.7 (How to choose c_1, c_2)**.** *For Corollaries 3.3 and 3.5, an optimal choice of $c_1 > 1$ exists but it depends on the unknown parameter α. To see this, minimize the lower bound of t in the aforementioned corollaries with respect to c_1, noting that the only term in τ that depends on c_1 is $(|\lfloor \log_a(\alpha/\alpha_0) \rfloor| + 1)^{c_1}$. Assuming $\alpha_0 \neq \alpha$, this gives*

$$c_1 = \frac{1 + \sqrt{1 + \frac{4}{\log(|\lfloor \log_a(\alpha/\alpha_0) \rfloor|+1)}}}{2}.$$

By the same reasoning, for Corollaries 3.3 and 3.4 and $\beta_0 \neq \beta$, the optimal choice of $c_2 > 1$ depends on the unknown parameter β and is given by

$$c_2 = \frac{1 + \sqrt{1 + \frac{4}{\log(|\lceil \log_b(\beta/\beta_0) \rceil|+1)}}}{2}.$$

Intuitively, if α_0 is far from α then c_1 should be closer to 1, and similarly for β_0 and β regarding c_2. In the absence of prior knowledge, we recommend a sensible default such as $c_1 = c_2 = 2$. ♦

Finally, to emphasize the generality of our algorithm, we consider the case where α and β are known to lie within explicit ranges. In this case, we modify set S based on these ranges and choose a schedule criterion function $h(x_1, x_2, x_3)$ depending on x_3 only. The following result is immediate.

Corollary 3.8 (Known ranges for α, β)**.** *Suppose we have integers*

$$i_{\min} \leq i_{\max}, \quad 0 \leq j_{\min} \leq j_{\max},$$

for which

$$\alpha \in [a^{i_{\min}}\alpha_0, a^{i_{\max}}\alpha_0], \quad \beta \in [b^{j_{\min}}\beta_0, b^{j_{\max}}\beta_0].$$

Let

$$S = \{i_{\min}, i_{\min}+1, \ldots, i_{\max}\} \times \{j_{\min}, j_{\min}+1, \ldots, j_{\max}\} \times \mathbb{N},$$

and $h(x_1, x_2, x_3) = x_3$ be a schedule criterion. Then given any h-assignment ϕ and any $\varepsilon \in (0, \epsilon_0)$, running Algorithm 2 with

$$t \geq (i_{\max} - i_{\min} + 1)(j_{\max} - j_{\min} + 1)K(\varepsilon),$$

where $K(\varepsilon)$ is as in (3.3), implies $f(x^{(t)}) - \hat{f} + g_Q(x^{(t)}) \leq \max\{\eta, \varepsilon\}$.

Note that Algorithm 2 is sequential. However, one can readily devise a parallel implementation that runs Algorithm 1 in parallel over each pair (i, j) and then minimizes $f + g_Q$ over all instances at the end of the process.

4 Examples

In this section, we present various examples of first-order methods that can be used in our restart scheme for different problem settings. In particular, we describe the methods that lead to the various results in Table 1. We do this by explicitly deriving a method $\Gamma : \mathbb{R}_{++} \times \mathbb{R}_{++} \times D \to D$ that satisfies (1.3) and give an explicit bound for the cost function $C_\Gamma(\delta, \epsilon, x_0)$ of the form $C\delta^{d_1}/\epsilon^{d_2} + 1$ for suitable d_1 and d_2.

Remark 4.1 (Optimization over $\mathbb{C}$)**.** *In convex analysis and continuous optimization, it is standard to consider function inputs lying in a finite-dimensional vector space over $\mathbb{R}$. The results described below are extended to $\mathbb{C}$, but this treatment does not always arise in the original papers for the first-order methods. We are interested in the domain of f being a subset of $\mathbb{C}^n$. Hence, we consider the natural isomorphism between $\mathbb{C}^n$ and $\mathbb{R}^{2n}$ given by: if $z = x + \mathrm{i}y \in \mathbb{C}^n$ with $x, y \in \mathbb{R}^n$, then $z \mapsto (x, y)$. We refer to z as the complex representation and (x, y) as the real representation. Now, one proceeds to do convex analysis and continuous optimization in the real representation, then express the results in the equivalent complex representation. Fortunately, not much needs to change (at least symbolically) when switching between real and complex representations.*

For example, the Euclidean inner products $\langle \cdot, \cdot \rangle$ have to be substituted with their real part, i.e., $\langle \cdot, \cdot \rangle_{\mathbb{R}} := \mathrm{Re}\,\langle \cdot, \cdot \rangle$. Another example pertains to the differentiability of f. Specifically, for $x, y \in \mathbb{R}^n$, we say that f is differentiable at $z = x + \mathrm{i}y \in D \subseteq \mathbb{C}^n$ if and only if $\mathrm{Re}\,(f)$ is (real) differentiable at (x, y). To define the gradient, denote ∇_x and ∇_y as the vector of partial derivatives corresponding to variables x and y, respectively. Then $\nabla f := \nabla_x \mathrm{Re}\,(f) + \mathrm{i}\nabla_y \mathrm{Re}\,(f)$, noting that because f is real-valued, we have $\mathrm{Im}\,(f) \equiv 0$. Other parts of convex analysis, such as convexity, functions, proximal mappings, subgradients, and so on, also extend to a complex vector domain by applying the definitions to the real representation of complex vectors. ♦

Algorithm 3: Nesterov's method

Input : An L-smooth function f and closed, convex set $Q \subseteq \mathbb{C}^n$ as in (1.1), prox-function $p(\cdot; x_0)$ with strong convexity constant σ_p and unique minimizer $x_0 \in Q$, sequences $\{\gamma_j\}_{j=0}^{\infty}$ and $\{\tau_j\}_{j=0}^{\infty}$, and number of iterations N.

Output: The vector x_N, which estimates a minimizer of (1.1).

$z_0 \leftarrow x_0$

for $j = 0, 1, \ldots, N-1$ **do**

  $x_{j+1} \leftarrow \underset{x \in Q}{\operatorname{argmin}} \frac{L}{2}\|x - z_j\|_{\ell^2}^2 + \langle \nabla f(z_j), x - z_j \rangle_{\mathbb{R}}$

  $v_j \leftarrow \underset{x \in Q}{\operatorname{argmin}} \frac{L}{\sigma_p} p(x; x_0) + \sum_{i=0}^{j} \gamma_i \langle \nabla f(z_i), x - z_i \rangle_{\mathbb{R}}$

  $z_{j+1} \leftarrow \tau_j v_j + (1 - \tau_j) x_{j+1}$

end

4.1 Nesterov's method for L-smooth functions

For our first example, we consider Nesterov's method [56], an accelerated projected gradient descent algorithm for general constrained convex optimization problems. Specifically, the algorithm aims to solve (1.1) in the special case when f is convex and L-smooth:

Definition 4.2. A function $f : \mathbb{C}^n \to \mathbb{R}$ is *L-smooth* over $Q \subseteq \mathbb{C}^n$ if it is Fréchet differentiable in an open set containing Q, and for all x, y in this set, its gradient ∇f has the Lipschitz property

$$\|\nabla f(x) - \nabla f(y)\|_{\ell^2} \leq L\|x - y\|_{\ell^2}. \qquad \blacktriangle$$

Nesterov's method is given in Algorithm 3. The algorithm uses the notion of a *prox-function* p. Here $p : Q \to \mathbb{R}$ is a proper, closed and strongly convex function with strong convexity constant $\sigma_p > 0$, that, in addition, satisfies $\min_{x \in Q} p(x) = 0$. Let $x_0 = \operatorname{argmin}_{x \in Q} p(x)$ be the unique minimizer of p. To make this dependence explicit, we write $p(\cdot) = p(\cdot; x_0)$. A common and simple choice of prox-function is $p(x; x_0) = \frac{1}{2}\|x - x_0\|_{\ell^2}^2$ with $\sigma_p = 1$. This will be useful when we express Nesterov's method *with smoothing*, in terms of Γ. We now state Nesterov's main result that gives a bound for $f(x_k) - f(x)$, for any $x \in Q$.

Lemma 4.3 (Nesterov's theorem)**.** *Let $Q \subseteq \mathbb{C}^n$ be nonempty, closed and convex, f a convex L-smooth function over Q. In addition, let $p : Q \to \mathbb{R}$ be a proper, closed and strongly convex function over Q with strong convexity constant $\sigma_p > 0$ with $\min_{x \in Q} p(x) = 0$. Then Algorithm 3 with*

$$\gamma_j = \frac{j+1}{2}, \qquad \tau_j = \frac{2}{j+3}, \qquad x_0 = \underset{x \in Q}{\operatorname{argmin}}\ p(x),$$

generates a sequence $\{x_k\}_{k=1}^{\infty} \subset Q$ satisfying

$$f(x_k) - f(x) \leq \frac{4Lp(x; x_0)}{k(k+1)\sigma_p}, \qquad \forall x \in Q. \tag{4.1}$$

Lemma 4.3 consists of two modifications of [56, Theorem 2]. First, we do not assume Q is bounded, as the results in the original work do not use this. Second, we allow $x \in Q$ instead of $x \in \widehat{X}$. The proof in the original work does not use the optimality of x, and only requires x to be feasible. We utilize this property when considering Nesterov's method with smoothing. The following is now immediate.

Proposition 4.4. *Let $Q \subseteq \mathbb{C}^n$ be nonempty, closed and convex, f a convex L-smooth function over Q (Definition 4.2). Given input $(\delta, \epsilon, x_0) \in \mathbb{R}_+ \times \mathbb{R}_+ \times Q$, let $\Gamma(\delta, \epsilon, x_0)$ be the output of Algorithm 3 with*

$$p(x; x_0) = \frac{1}{2}\|x - x_0\|_{\ell^2}^2, \quad \gamma_j = \frac{j+1}{2}, \quad \tau_j = \frac{2}{j+3}, \quad N = \left\lceil \frac{\delta\sqrt{2L}}{\sqrt{\epsilon}} \right\rceil.$$

Then (1.3) holds with $g_Q \equiv 0$. Specifically,

$$f(\Gamma(\delta, \epsilon, x_0)) - \hat{f} \leq \epsilon, \quad \forall x_0 \in Q \text{ with } d(x_0, \widehat{X}) \leq \delta, \tag{4.2}$$

where d is the metric induced by the ℓ^2-norm. It follows that we can take

$$\mathcal{C}_\Gamma(\delta, \epsilon) = \left\lceil \frac{\delta\sqrt{2L}}{\sqrt{\epsilon}} \right\rceil. \tag{4.3}$$

Proposition 4.4 shows that we can take $d_1 = 1$ and $d_2 = 1/2$ in the cost bound (3.5) for Nesterov's method (without smoothing). If f is $L-$smooth and satisfies (1.2) with $\eta = 0$, then $\beta \geq 2$. It follows that we can take $\beta_0 = 2$. Theorem 3.2 now implies the rates in the first row of Table 1.

Several other remarks are in order. First, in Nesterov's method, the iterates x_j are always feasible since the corresponding update step returns a point in Q. Thus in Proposition 4.4 we do not have to define g_Q since Γ trivially satisfies (1.3) with $g_Q \equiv 0$. Finally, in Nesterov's method, the requirement $x_0 \in Q$ can be relaxed. For instance, we only require f is L-smooth over the union of Q and an open neighborhood of x_0 for some $L > 0$ to start with $x_0 \notin Q$.

4.2 Nesterov's method for (u, v)-smoothable functions

We can extend Nesterov's method to solve (1.1) without assuming that f is differentiable. This is done via *smoothing*. For this, we need the following definition from [10, Definition 10.43] (extended to functions with complex-vector domains).

Definition 4.5. Let $u, v > 0$. A convex function $f : \mathbb{C}^n \to \mathbb{R}$ is called (u, v)-*smoothable* if for any $\mu > 0$ there exists a convex differentiable function $f_\mu : \mathbb{C}^n \to \mathbb{R}$ such that

1. $f_\mu(x) \leq f(x) \leq f_\mu(x) + v\mu$ for all $x \in \mathbb{C}^n$
2. f_μ is $\frac{u}{\mu}$-smooth over $\mathbb{C}^n$

The function f_μ is referred to as a $\frac{1}{\mu}$-*smooth approximation* of f with parameters (u, v), and μ is referred to as the *smoothing parameter*. ▲

Smoothing is a framework that approximates f arbitrarily closely by a family of smooth functions, i.e., functions with Lipschitz gradients. This means that we can apply Nesterov's method to a smooth approximation of f, and also analyze the objective error in terms of f. The following provides a modified version of Lemma 4.3 for (a, b)-smoothable f, and is proven in Appendix A.1.

Lemma 4.6. *Let $f : \mathbb{C}^n \to \mathbb{R}$ be a convex (u, v)-smoothable function. Given any $\mu > 0$, let f_μ be a $\frac{1}{\mu}$-smooth approximation of f with parameters (u, v). Then taking Q, p, γ_j, τ_j, x_0 as in Lemma 4.3 and applying Algorithm 3 to the function f_μ produces a sequence $\{x_k\}_{k=1}^\infty$ satisfying*

$$f(x_k) - f(x) \leq \frac{4up(x; x_0)}{\mu k(k+1)\sigma_p} + v\mu, \qquad x \in Q. \tag{4.4}$$

The following proposition shows that Nesterov's method with smoothing can be formulated as an algorithm Γ in our framework, and is proven in Appendix A.1.

Proposition 4.7. *Let $Q \subseteq \mathbb{C}^n$ be nonempty, closed and convex, and $f : \mathbb{C}^n \to \mathbb{R}$ a convex (u, v)-smoothable function (Definition 4.5). Given input $(\delta, \epsilon, x_0) \in \mathbb{R}_+ \times \mathbb{R}_+ \times Q$, let $\Gamma(\delta, \epsilon, x_0)$ be the output of Algorithm 3 applied to function f_μ with*

$$\mu = \frac{\epsilon}{2v}, \quad p(x; x_0) = \frac{1}{2}\|x - x_0\|_{\ell^2}^2, \quad \gamma_j = \frac{j+1}{2}, \quad \tau_j = \frac{2}{j+3}, \quad N = \left\lceil \frac{2\sqrt{2uv} \cdot \delta}{\epsilon} \right\rceil.$$

Then

$$f(\Gamma(\delta, \epsilon, x_0)) - \hat{f} \leq \epsilon, \qquad \forall x_0 \in Q \text{ satisfying } d(x_0, \widehat{X}) \leq \delta,$$

where d is the metric induced by the ℓ^2-norm. It follows that we can set

$$\mathcal{C}_\Gamma(\delta, \epsilon, x_0) = \left\lceil \frac{2\sqrt{2uv} \cdot \delta}{\epsilon} \right\rceil.$$

This result shows that we can take $d_1 = 1$ and $d_2 = 1$ in (3.5) in the case of Nesterov's method with smoothing. Theorem 3.2 now implies the rates in the second row of Table 1.

The following discussion considers a standard example of smoothing that is closely related to proximal maps, from [10, Theorem 10.51]. If $f : \mathbb{C}^n \to \mathbb{R}$ is convex and Lipschitz continuous with Lipschitz constant L_f, then it is $(1, L_f^2/2)$-smoothable. In particular, the *Moreau envelope* with parameter $\mu > 0$ is a $\frac{1}{\mu}$-smooth approximation of f with parameters $(1, L_f^2)$. Given a convex function $f : \mathbb{C}^n \to \mathbb{R}$ and $\mu > 0$, the *Moreau envelope* of f is the function

$$M_f^\mu(x) = \min_{y \in \mathbb{C}^n} \left\{ f(y) + \frac{1}{2\mu}\|x - y\|_{\ell^2}^2 \right\}. \tag{4.5}$$

The number μ is referred to as the *smoothing parameter*. The Moreau envelope M_f^μ is well-defined, and the minimization problem defined in (4.5) has a unique solution corresponding to $\operatorname{prox}_{\mu f}(x)$, i.e., the proximal map of μf at x [10, Theorem 6.3]. The Moreau envelope of f is also $\frac{1}{\mu}$-smooth over its domain, where for any x we have

$$\nabla M_f^\mu(x) = \frac{1}{\mu}(x - \operatorname{prox}_{\mu f}(x)).$$

Examples of Moreau envelopes of functions can be found in [10, Section 6.7].

4.3 The universal fast gradient method

We next consider Hölder smooth functions, which are a natural way of interpolating between nonsmooth and smooth objective functions.

Definition 4.8. A function $q : \mathbb{C}^n \to \mathbb{R}$ is Hölder smooth over $Q \subseteq \mathbb{C}^n$ with parameter $\nu \in [0, 1]$ if

$$\|\nabla q(x) - \nabla q(y)\|_{\ell^2} \leq M_\nu \|x - y\|_{\ell^2}^\nu, \qquad \forall\, x, y \in Q, \nabla q(x) \in \partial q(x), \nabla q(y) \in \partial q(y). \qquad \blacktriangle$$

We consider the universal fast gradient method [58] for the problem

$$\min_{x \in Q} f(x), \qquad f(x) := q(x) + g(x), \tag{4.6}$$

Algorithm 4: Universal fast gradient method

Input : $\epsilon > 0$, $L_0 > 0$, $\phi_0(x) = 0$, $y_0 = x_0$, $A_0 = 0$.

Output: The vector x_N, which estimates a minimizer of (4.6).

for $k = 0, 1, \ldots, N$ **do**
- $v_k \leftarrow \mathrm{prox}_{\phi_k, Q}(x_0)$
- $i_k \leftarrow -1$
- **do**
  - $i_k \leftarrow i_k + 1$
  - Compute a_{k+1,i_k} from the equation $a_{k+1,i_k}^2 = \frac{1}{2^{i_k} L_k}(A_k + a_{k+1,i_k})$.
  - $A_{k+1,i_k} \leftarrow A_k + a_{k+1,i_k}$
  - $\tau_{k,i_k} \leftarrow a_{k+1,i_k} / A_{k+1,i_k}$
  - $x_{k+1,i_k} \leftarrow \tau_{k,i_k} v_k + (1 - \tau_{k,i_k}) y_k$
  - Choose a subgradient $\nabla q(x_{k+1,i_k}) \in \partial q(x_{k+1,i_k})$.
  - $\hat{\phi}_{k+1,i_k}(x) \leftarrow a_{k+1,i_k}[\langle \nabla q(x_{k+1,i_k}), x \rangle_{\mathbb{R}} + g(x)]$
  - $\hat{x}_{k+1,i_k} \leftarrow \mathrm{prox}_{\hat{\phi}_{k+1,i_k}, Q}(v_k)$
  - $y_{k+1,i_k} \leftarrow \tau_{k,i_k} \hat{x}_{k+1,i_k} + (1 - \tau_{k,i_k}) y_k$
- **while** $q(y_{k+1,i_k}) > q(x_{k+1,i_k}) + \langle \nabla q(x_{k+1,i_k}), y_{k+1,i_k} - x_{k+1,i_k} \rangle_{\mathbb{R}} + 2^{i_k - 1} L_k \|y_{k+1,i_k} - x_{k+1,i_k}\|_{\ell^2}^2 + \frac{\epsilon}{2} \tau_{k,i_k}$
- $x_{k+1} \leftarrow x_{k+1,i_k}$, $y_{k+1} \leftarrow y_{k+1,i_k}$, $a_{k+1} \leftarrow a_{k+1,i_k}$, $\tau_k \leftarrow \tau_{k,i_k}$
- $A_{k+1} \leftarrow A_k + a_{k+1}$, $L_{k+1} \leftarrow 2^{i_k - 1} L_k$
- $\phi_{k+1}(x) \leftarrow \phi_k(x) + a_{k+1}[q(x_{k+1}) + \langle \nabla q(x_{k+1}), x - x_{k+1} \rangle_{\mathbb{R}} + g(x)]$.

end

where q is a proper convex function that is Hölder smooth for some $\nu \in [0, 1]$, and g is a closed convex function whose proximal map,

$$\mathrm{prox}_{cg, Q}(x) = \operatorname*{argmin}_{y \in Q} \left\{ c \cdot g(y) + \frac{1}{2} \|x - y\|_{\ell^2}^2 \right\},$$

is straightforward to compute. The iterates of the universal fast gradient method are summarized in Algorithm 4.

Lemma 4.9 (Theorem 3 of [58])**.** *Let $Q \subseteq \mathbb{C}^n$ be nonempty, closed and convex, q a proper convex function that is Hölder smooth for some $\nu \in [0, 1]$ and $M_\nu < \infty$ (Definition 4.8), and g a closed convex function. Then Algorithm 4 generates a sequence $\{x_k\}_{k=1}^\infty \subset Q$ satisfying*

$$f(x_k) - \hat{f} \leq \left(\frac{2^{2+4\nu} M_\nu^2}{\epsilon^{1-\nu} k^{1+3\nu}} \right)^{\frac{1}{1+\nu}} \frac{d(x_0, \widehat{X})^2}{2} + \frac{\epsilon}{2}, \qquad \forall x \in Q, \tag{4.7}$$

where d is the metric induced by the ℓ^2-norm.

By choosing k to match the two terms on the right-hand side of (4.7), the following proposition is immediate.

Proposition 4.10. *Let $Q \subseteq \mathbb{C}^n$ be nonempty, closed and convex, q a proper convex function is Hölder smooth for some $\nu \in [0, 1]$ and $M_\nu \geq 0$ (Definition 4.8), and g a closed convex function. Given input $(\delta, \epsilon, x_0) \in \mathbb{R}_+ \times \mathbb{R}_+ \times Q$, let $\Gamma(\delta, \epsilon, x_0)$ be the output of Algorithm 4 with*

$$N = \left\lceil \frac{2^{\frac{2+4\nu}{1+3\nu}} M_\nu^{\frac{2}{1+3\nu}} \delta^{\frac{2+2\nu}{1+3\nu}}}{\epsilon^{\frac{2}{1+3\nu}}} \right\rceil.$$

Algorithm 5: Primal-dual algorithm for the problem (4.8).

Input : Initial vectors $x_0 \in \mathbb{C}^n$ and $y_0 \in \mathbb{C}^m$, proximal step sizes $\tau, \sigma > 0$, number of iterations N, matrix $B \in \mathbb{C}^{m \times n}$, and routines for appropriate proximal maps.

Output: Final ergodic average X_N approximating a solution to (4.8).

Initiate with $x^{(0)} = x_0$, $y_1^{(0)} = y_0$, $X_0 = 0$, and $Y_0 = 0$.

for $j = 0, \dots, N-1$ **do**

$x^{(j+1)} \leftarrow \mathrm{prox}_{\tau g}\left(x^{(j)} - \tau B^* y^{(j)} - \tau \nabla q(x^{(j)})\right)$;

$y^{(j+1)} \leftarrow \mathrm{prox}_{\sigma h^*}\left(y^{(j)} + \sigma B(2x^{(j+1)} - x^{(j)})\right)$;

$X_{j+1} \leftarrow \frac{1}{j+1}\left(jX_j + x^{(j+1)}\right)$;

$Y_{j+1} \leftarrow \frac{1}{j+1}\left(jY_j + y^{(j+1)}\right)$;

end

Then

$$f(\Gamma(\delta, \epsilon, x_0)) - \hat{f} \leq \epsilon, \qquad \forall x_0 \in Q \text{ satisfying } d(x_0, \widehat{X}) \leq \delta,$$

where d is the metric induced by the ℓ^2-norm. It follows that we can set

$$\mathcal{C}_\Gamma(\delta, \epsilon, x_0) = \left\lceil \frac{2^{\frac{2+4\nu}{1+3\nu}} M_\nu^{\frac{2}{1+3\nu}} \delta^{\frac{2+2\nu}{1+3\nu}}}{\epsilon^{\frac{2}{1+3\nu}}} \right\rceil.$$

Proposition 4.10 shows that we can take $d_1 = (2+2\nu)/(1+3\nu)$ and $d_2 = 2/(1+3\nu)$ for the universal fast gradient method. Note that if q satisfies both (1.2) for $\eta = 0$ and Definition 4.8, then $\beta \geq 1+\nu$ [69]. Therefore, we take $\beta_0 = 1+\nu$. Theorem 3.2 now implies the rates in the third row of Table 1.

4.4 The primal-dual iteration for unconstrained problems

We now consider Chambolle and Pock's primal-dual algorithm [23, 25]. The primal-dual hybrid gradient (PDHG) algorithm is a popular method to solve saddle point problems [22,31,63]. Consider the problem

$$\min_{x \in \mathbb{C}^n} f(x), \qquad f(x) := q(x) + g(x) + h(Bx), \tag{4.8}$$

where: $B \in \mathbb{C}^{m \times n}$ with $\|B\| \leq L_B$; q is a proper, lower semicontinuous, convex function, and is L_q-smooth; and g, h are proper, lower semicontinuous, convex functions whose proximal maps are straightforward to compute. We also use the standard Euclidean metric for d in (1.2) and write the primal-dual iterates in their simplified form accordingly.

The saddle-point problem associated with (4.8) is

$$\min_{x \in \mathbb{C}^n} \max_{y \in \mathbb{C}^m} \mathcal{L}(x, y) := \langle Bx, y \rangle_{\mathbb{R}} + q(x) + g(x) - h^*(y). \tag{4.9}$$

The primal-dual iterates are summarized in Algorithm 5, where the output is the ergodic average of the primal-dual iterates. Note that the primal-dual algorithm allows us to easily deal with the matrix B, which can be difficult with other first-order methods. If $\tau(\sigma L_B^2 + L_q) \leq 1$, then [25, Theorem 1] shows that for any $x \in \mathbb{C}^n$ and $y \in \mathbb{C}^m$,

$$\mathcal{L}(X_k, y) - \mathcal{L}(x, Y_k) \leq \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\|y - y^{(0)}\|^2}{\sigma}\right). \tag{4.10}$$

The following lemma is a simple consequence of this bound and is proven in Appendix A.2.

Lemma 4.11. *Consider the primal-dual iterates in Algorithm 5. If $\tau(\sigma L_B^2 + L_q) \leq 1$, then*

$$f(X_k) - f(x) \leq \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\|y - y^{(0)}\|^2}{\sigma}\right), \quad \forall x \in \mathbb{C}^n,\ y \in \partial h(BX_k). \tag{4.11}$$

We can take the infimum over $y \in \partial h(BX_k)$ on the right-hand side of (4.11) to obtain

$$f(X_k) - f(x) \leq \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\sup_{z\in\mathrm{dom}(h)} \inf_{y\in\partial h(z)} \|y - y^{(0)}\|^2}{\sigma}\right), \quad \forall x \in \mathbb{C}^n. \tag{4.12}$$

To bound the right-hand side, we take $y^{(0)} = 0$ and consider the case where h is such that there always exist points y in the subdifferential of h for which $\|y\|$ is not too large. Note that this always holds if, for example, h is Lipschitz continuous and its domain is open [10, Theorem 3.61]. The following proposition now shows how this falls into the framework of our restart scheme and is proven in Appendix A.2.

Proposition 4.12. *Suppose that*

$$\sup_{z\in\mathrm{dom}(h)} \inf_{y\in\partial h(z)} \|y\| \leq L_h < \infty. \tag{4.13}$$

Given input $(\delta, \epsilon, x_0) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{C}^n$, let $\Gamma(\delta, \epsilon, x_0)$ be the output of Algorithm 5 with

$$y_0 = 0, \quad \tau = \frac{\delta}{L_B L_h + \delta L_q}, \quad \sigma = \frac{L_h}{\delta L_B}, \quad N = \left\lceil \frac{\delta}{\epsilon}\left(2L_B L_h + \delta L_q\right)\right\rceil.$$

Then

$$f(\Gamma(\delta, \epsilon, x_0)) - \hat{f} \leq \epsilon, \quad \forall x_0 \text{ with } d(x_0, \widehat{X}) \leq \delta. \tag{4.14}$$

It follows that we can take

$$\mathcal{C}_\Gamma(\delta, \epsilon, x_0) = \left\lceil \frac{\delta}{\epsilon}\left(2L_B L_h + \delta L_q\right)\right\rceil. \tag{4.15}$$

Assuming that δ is bounded, Proposition 4.12 shows that we can take $d_1 = 1$ and $d_2 = 1$ for the primal-dual algorithm. Theorem 3.2 now implies the rates in the fourth row of Table 1.

4.5 The primal-dual iterations for constrained problems

We now consider primal-dual iterations, but for the constrained problem

$$\min_{x\in\mathbb{C}^n} f(x) + \chi_C(Ax), \qquad f(x) := q(x) + g(x) + h(Bx), \tag{4.16}$$

with the same assumptions on q, g, h and B as in Section 4.4, but now with the additional term $\chi_C(Ax)$. Here, C is a closed and non-empty convex set, χ_C is its indicator function of C and $A \in \mathbb{C}^{m'\times n}$ with $\|A\| \leq L_A$. This fits into our framework with the choice

$$Q = \{x \in \mathbb{C}^n : Ax \in C\}, \qquad g_Q(x) = g_Q(\kappa; x) = \kappa \cdot \inf_{z\in C} \|Ax - z\|,$$

for $\kappa > 0$. Note that κ is an additional parameter that can be chosen to balance the rate of reduction in the feasibility gap versus the objective function error. It is possible to formulate a projected version of the primal-dual iteration. However, like with Nesterov's method, this is only

Algorithm 6: Primal-dual algorithm for the constrained problem (4.16).

Input : Initial vectors $x_0 \in \mathbb{C}^n$, $[y_0]_1 \in \mathbb{C}^m$ and $[y_0]_2 \in \mathbb{C}^{m'}$, proximal step sizes $\tau, \sigma_1, \sigma_2 > 0$, number of iterations N, matrices $B \in \mathbb{C}^{m\times n}$ and $A \in \mathbb{C}^{m'\times n}$, and routines for appropriate proximal maps.

Output: Final ergodic average X_N approximating a solution to (4.16).

Initiate with $x^{(0)} = x_0$, $y_1^{(0)} = [y_0]_1$, $y_2^{(0)} = [y_0]_2$, $X_0 = 0$, $[Y_0]_1 = 0$, and $[Y_0]_2 = 0$.

for $j = 0, \dots, N-1$ **do**

$x^{(j+1)} \leftarrow \text{prox}_{\tau g}\left(x^{(j)} - \tau B^* y_1^{(j)} - \tau A^* y_2^{(j)} - \tau \nabla q(x^{(j)})\right)$;

$y_1^{(j+1)} \leftarrow \text{prox}_{\sigma_1 h^*}\left(y_1^{(j)} + \sigma_1 B(2x^{(j+1)} - x^{(j)})\right)$;

$y_2^{(j+1)} \leftarrow y_2^{(j)} + \sigma_2 A(2x^{(j+1)} - x^{(j)}) - \sigma_2 \mathcal{P}_C\left(y_2^{(j)}/\sigma_2 + A(2x^{(j+1)} - x^{(j)})\right)$;

$X_{j+1} \leftarrow \frac{1}{j+1}\left(jX_j + x^{(j+1)}\right)$;

$[Y_{j+1}]_1 \leftarrow \frac{1}{j+1}\left(j[Y_j]_1 + y_1^{(j+1)}\right)$;

$[Y_{j+1}]_2 \leftarrow \frac{1}{j+1}\left(j[Y_j]_2 + y_2^{(j+1)}\right)$;

end

possible when the projection onto Q can be easily computed. In this section, we consider a primal-dual iteration for (4.16) that only involves computing the projection onto the set C, at the price of producing non-feasible iterates.

The saddle-point problem associated with (4.16) is

$$\min_{x\in\mathbb{C}^n} \max_{y_1\in\mathbb{C}^m} \max_{y_2\in\mathbb{C}^{m'}} \mathcal{L}_C(x, y_1, y_2) := \langle Bx, y_1\rangle_{\mathbb{R}} + q(x) + g(x) - h^*(y_1) + \langle Ax, y_2\rangle_{\mathbb{R}} - \sup_{z\in C}\langle z, y_2\rangle_{\mathbb{R}}. \quad (4.17)$$

The primal-dual iterates are summarized in Algorithm 6, where, again, the output is the ergodic average of the primal-dual iterates. We have included three proximal step sizes τ, σ_1 and σ_2, which correspond to the primal variable and the two dual variables, respectively. To compute the proximal map associated with the second dual variable, we use Moreau's identity to write

$$\text{prox}_{\sigma_2\chi_C^*}(y) = y - \sigma_2 \mathcal{P}_C(y/\sigma_2),$$

where $\mathcal{P}_C$ denotes the projection onto C (with respect to the standard Euclidean norm).

If $\tau(\sigma_1 L_B^2 + \sigma_2 L_A^2 + L_q) \le 1$, then a straightforward adaption of [25, Theorem 1] shows that for any $x \in \mathbb{C}^n$, $y_1 \in \mathbb{C}^m$ and $y_2 \in \mathbb{C}^{m'}$,

$$\mathcal{L}_C\left(X_k, y_1, y_2\right) - \mathcal{L}_C\left(x, [Y_k]_1, [Y_k]_2\right) \le \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\|y_1 - y_1^{(0)}\|^2}{\sigma_1} + \frac{\|y_2 - y_2^{(0)}\|^2}{\sigma_2}\right). \quad (4.18)$$

We now have the following lemma and resulting proposition, both of which are proven in Appendix A.3.

Lemma 4.13. *Consider the primal-dual algorithm in Algorithm 6 with $y_2^{(0)} = 0$. If $\tau(\sigma_1 L_B^2 + \sigma_2 L_A^2 + L_q) \le 1$, then for any $\kappa > 0$*

$$f(X_k) - f(x) + g_Q(\kappa; X_k) \le \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\|y_1 - y_1^{(0)}\|^2}{\sigma_1} + \frac{\kappa^2}{\sigma_2}\right), \quad \forall x \in Q,\ y_1 \in \partial h(BX_k). \quad (4.19)$$

Proposition 4.14. *Suppose that*

$$\sup_{z\in\mathrm{dom}(h)} \inf_{y\in\partial h(z)} \|y\| \leq L_h < \infty. \tag{4.20}$$

Given input $(\delta, \epsilon, x_0) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{C}^n$*, let* $\Gamma(\delta, \epsilon, x_0)$ *be the output of Algorithm 6 with*

$$[y_0]_1 = 0, [y_0]_2 = 0, \tau = \frac{\delta}{\kappa L_A + L_h L_B + \delta L_q}, \sigma_1 = \frac{L_h}{\delta L_B}, \sigma_2 = \frac{\kappa}{\delta L_A}, N = \left\lceil \frac{\delta\left(2\kappa L_A + 2L_h L_B + \delta L_q\right)}{\epsilon} \right\rceil.$$

Then

$$f(\Gamma(\delta, \epsilon, x_0)) - \hat{f} + g_Q(\kappa; \hat{x}) \leq \epsilon, \quad \forall x_0 \text{ with } d(x_0, \widehat{X}) \leq \delta. \tag{4.21}$$

It follows that we can take

$$\mathcal{C}_\Gamma(\delta, \epsilon, x_0) = \left\lceil \frac{\delta\left(2\kappa L_A + 2L_h L_B + \delta L_q\right)}{\epsilon} \right\rceil. \tag{4.22}$$

Assuming that δ is bounded, Proposition 4.14 shows that we can take $d_1 = 1$ and $d_2 = 1$ for the primal-dual algorithm. Theorem 3.2 now implies the rates in the final row of Table 1.

5 Numerical experiments

We implement several numerical experiments for the general restart scheme (Algorithm 2) applied to three different problems. The first is a simple sparse recovery problem modeled as QCBP, which is solved using the primal-dual iteration for constrained problems (Algorithm 6). Second, we consider image reconstruction from Fourier measurements via TV minimization. The reconstruction is computed using NESTA [12], where NESTA is an accelerated projected gradient descent algorithm derived from Nesterov's method (Algorithm 3) with smoothing. Third, we perform feature selection on three real-world datasets. This selection is done by solving a SR-LASSO problem on the data with unconstrained primal-dual iterations (Algorithm 5).

Before discussing the examples in turn, we make some general remarks about the implementation. First, we use the schedule criteria from Section 3.2, and for parameters we always set $c_1 = c_2 = 2$, $b = \mathrm{e}$, $r = \mathrm{e}^{-1}$, and $a = e^{c_1\beta/d_1}$ for unknown α but known β (Corollary 3.5), otherwise $a = e^{c_1/d_1}$ if both are unknown (Corollary 3.3). Assignments from the schedule criteria are obtained by enumerating and sorting solutions of the respective Diophantine equations found in the proofs of Corollaries 3.3 to 3.5. The choice of r is motivated by Remark 2.2 and the choice of a by Remark 3.6. The choice of c_1 and c_2 were arbitrary, with the intent of being sane defaults, and otherwise can be tuned to improve performance.

Second, when using the restart scheme for primal-dual iterations, we store and perform restarts on the dual variables for each instance indexed by (i, j).

Third, we use a simple workaround to handle finite precision arithmetic. In the grid search for the restart scheme, the sharpness parameter α_i can be arbitrarily large or small, and β_j can be arbitrarily large. Also, the adaptive restart parameters $\delta = \delta_{i,j,U}$ and $\epsilon = \epsilon_{i,j,U}$ can become arbitrarily small. Regarding the grid indices, we limit i and j so that

$$|i| \leq \lfloor \log_a(1/\epsilon_{\mathrm{mach}}) \rfloor, \qquad j \leq \lfloor \log_b(1/\epsilon_{\mathrm{mach}}) \rfloor,$$

where ϵ_{mach} is machine epsilon. Regarding the adaptive parameters, after the assignments of $\delta_{i,j,U+1}$ and $\epsilon_{i,j,U+1}$ in Algorithm 2, we insert the updates $\delta_{i,j,U+1} := \max(\delta_{i,j,U+1}, 10\epsilon_{\mathrm{mach}})$ and $\epsilon_{i,j,U+1} := \max(\epsilon_{i,j,U+1}, 10\epsilon_{\mathrm{mach}})$ to avoid setting them to zero.

Fourth, we slightly modify the primal-dual algorithm to improve overall performance. For each $j \geq 1$, we track a separate iterate $\widetilde{X}_j$ defined by

$$\widetilde{X}_j = \text{argmin}_{i=1,\dots,j} f(X_i) + \kappa g_Q(X_i), \qquad j \geq 1.$$

The iterates $\{\widetilde{X}_j\}_{j\geq 1}$ are not used in the primal-dual algorithm, but are instead used to evaluate the reconstruction or objective error in our experiments. In addition, the algorithm returns $\widetilde{X}_N$ as its final iterate. We similarly track a separate iterate for the dual variables, selecting them based on an evaluation of the Lagrangian (4.17) with $\widetilde{X}_j$. Note that choosing to output $\widetilde{X}_N$ instead of X_N is theoretically justified, since if (1.3) holds, then our modification would still satisfy (1.3) for the same parameters (δ, ϵ, x_0).

5.1 Sparse recovery via QCBP

We consider reconstructing a vector $x \in \mathbb{R}^n$ from noisy measurements $y = Ax + e \in \mathbb{R}^m$, where $A \in \mathbb{R}^{m\times n}$ is a matrix whose entries are i.i.d. Gaussian random variables with mean zero and variance $1/m$, and $e \in \mathbb{R}^m$ is a noise vector satisfying $\|e\|_{\ell^2} \leq \varsigma$ for some noise level $\varsigma > 0$. For a positive integer n, we write $[n] = \{1, 2, \dots, M\}$. Given a vector $z = (z_i)_{i=1}^n \in \mathbb{C}^n$ and $S \subseteq [n]$, the vector z_S has ith entry z_i if $i \in S$, and is zero otherwise. The best s-term approximation error of z is defined as

$$\sigma_s(z)_{\ell^1} = \min\{\|u_S - z\|_{\ell^1} : u \in \mathbb{C}^n,\ S \subseteq [n],\ |S| \leq s\}.$$

We assume that x is *approximately* s-sparse, in the sense that its best s-term approximation error $\sigma_s(x)_{\ell^1}$ is small. The recovery of x is formulated as solving the QCBP problem

$$\min_{z\in\mathbb{R}^n} \|z\|_{\ell^1} \text{ subject to } \|Az - y\|_{\ell^2} \leq \varsigma. \tag{5.1}$$

We use the following condition on the matrix A to ensure that approximate sharpness holds.

Definition 5.1 (Robust null space property, e.g., Definition 5.14 of [4])**.** The matrix $A \in \mathbb{C}^{m\times n}$ satisfies the *robust Null Space Property (rNSP)* with constants $0 < \rho < 1$ and $\gamma > 0$ if

$$\|v_S\|_{\ell^2} \leq \frac{\rho}{\sqrt{s}} \|v_{S^{\complement}}\|_{\ell^1} + \gamma \|Av\|_{\ell^2},$$

for all $v \in \mathbb{C}^n$ and $S \subseteq [M]$ with $|S| \leq s$. ▲

In [27, Theorem 3.3], it was shown that the robust null space property (rNSP) implies approximate sharpness. We restate the result in the notation of this paper for completeness.

Proposition 5.2 (Approximate sharpness of ℓ^1-norm for QCBP sparse recovery)**.** *Let $\varsigma > 0$. Suppose $A \in \mathbb{C}^{m\times n}$ has the rNSP of order s with constants $0 < \rho < 1$, $\gamma > 0$. Let $y \in \mathbb{C}^m$, $D = \mathbb{C}^n$, $Q = \{x \in \mathbb{C}^n : \|Ax - y\|_{\ell^2} \leq \varsigma\}$ and $f(x) = \|x\|_{\ell^1}$. Then the approximate sharpness condition* (1.2) *holds with*

$$g_Q(z; \sqrt{s}) = \sqrt{s}\max\{\|Az - y\|_{\ell^2} - \varsigma, 0\}, \quad \alpha = \hat{c}_1\sqrt{s}, \quad \beta = 1, \quad \eta = \hat{c}_2\sigma_s(x)_{\ell^1} + \hat{c}_3\varsigma\sqrt{s},$$

for constants $\hat{c}_1, \hat{c}_2, \hat{c}_3 > 0$ are constants depending only on ρ and γ.

The theory of compressed sensing [4, 35] aims to construct (random) matrices satisfying the rNSP, which is itself implied by the better-known *Restricted Isometry Property* (RIP). For example, if A is a Gaussian random matrix, then it satisfies the rNSP with probability at least $1-\varepsilon$, provided $m \geq C \cdot (s \cdot \log(\mathrm{e}N/s) + \log(2/\varepsilon))$ (see, e.g., [4, Theorem 5.22]). However, a sharp value of the constant C, and therefore also the rNSP constants ρ and γ, is unknown. This implies that the approximate sharpness constants α and η are also unknown. This motivates using the restart scheme (Algorithm 2), which does not require knowledge of α or η, to solve (5.1).

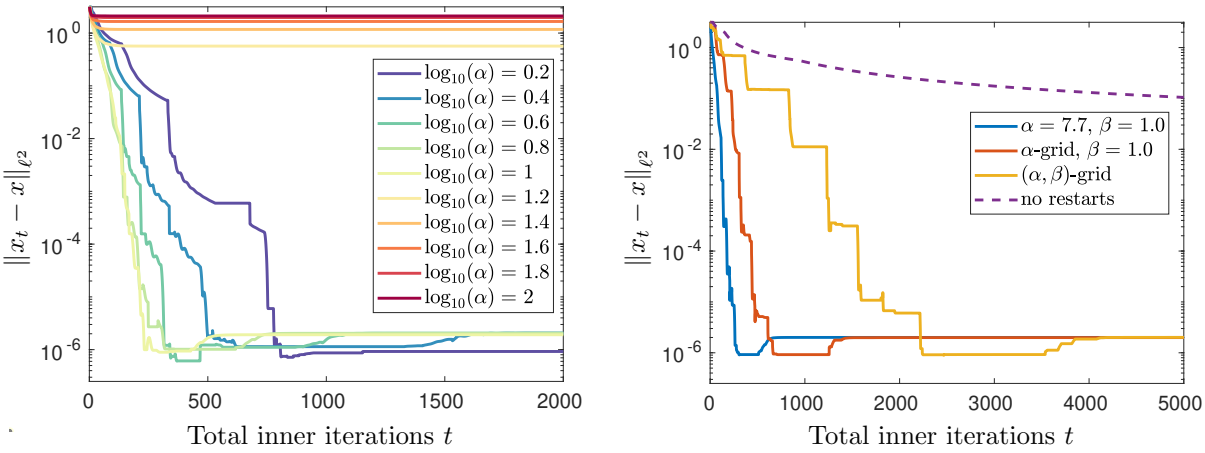

Figure 2: Reconstruction error of restarted primal-dual iteration for QCBP with $\varsigma = 10^{-6}$. Left: The restart scheme with fixed sharpness constants $\beta = 1$ and various α. Right: Various different schemes (including restarted and unrestarted schemes).

5.1.1 Experimental setup

We use the primal-dual iteration for constrained problems (Algorithm 6) to solve the sparse recovery problem. This can be done by expressing QCBP in (5.1) as (4.16) with

$$q \equiv 0, \quad h \equiv 0, \quad B = 0, \quad g(x) = \|x\|_{\ell^1}, \quad C = \{z \in \mathbb{C}^N : \|z - y\|_{\ell^2} \leq \varsigma\}.$$

Given these choices, the proximal map of τg is the shrinkage-thresholding operator, and the projection map is straightforward to compute since C is a shifted ℓ^2-ball. Moreover, we have $h^*(z) = +\infty$ whenever $z \neq 0$, and is zero otherwise. Therefore the proximal map $\mathrm{prox}_{\sigma_1 h^*}(x) = \|x\|_{\ell^2}^2/2$, and thus $y_1^{(j)} = 0$ for all $j > 0$ if the initial data $y_1^{(0)} = 0$. In essence, we can ignore the parameter σ_1 and updating the iterates $y_1^{(j)}$ in the primal-dual iterations (Algorithm 6). The error bound derived in Lemma 4.13 holds with the σ_1 term omitted.

Unless stated otherwise, the parameters used are ambient dimension $n = 128$, sparsity level $s = 10$, measurements $m = 60$, noise level $\varsigma = 10^{-6}$. The ground truth vector x is exactly sparse with s of its entries (randomly selected) corresponding to i.i.d. standard normal entries. The noise vector e is selected uniformly random on the ℓ^2-ball of radius ς and thus $\|e\|_{\ell^2} = \varsigma$. The objective function is $f(x) = \|x\|_{\ell^1}$ and the feasibility gap is given by $g_Q(x; \kappa) = \kappa \cdot \max\{\|Ax - y\|_{\ell^2} - \varsigma, 0\}$, which is derived from Section 4.5. The feasibility gap weight is set to $\kappa = \sqrt{m}$ from Proposition 5.2, noting that $s \leq m$ in general. In addition, $\alpha_0 = \sqrt{m}$, $\beta_0 = 1$. The choice of α_0 is also motivated by Proposition 5.2.

5.1.2 Results

Fig. 2 shows the performance of the restart scheme in Algorithm 1 for various fixed values of α and $\beta = 1$. For smaller α, the error decreases linearly down to the noise level $\varsigma = 10^{-6}$. This agrees with Theorem 2.1. Increasing α leads to fast linear convergence, up to a threshold (between 10^1 and $10^{1.2}$). After this point, the performance of the restart scheme abruptly breaks down since large α violates the approximate sharpness condition (1.2).

To overcome such parameter sensitivity, we use Algorithm 2. Fig. 2 also compares the performance of the restart scheme with fixed $(\alpha, \beta) = (\sqrt{m}, 1)$ with restart schemes that (i) perform a

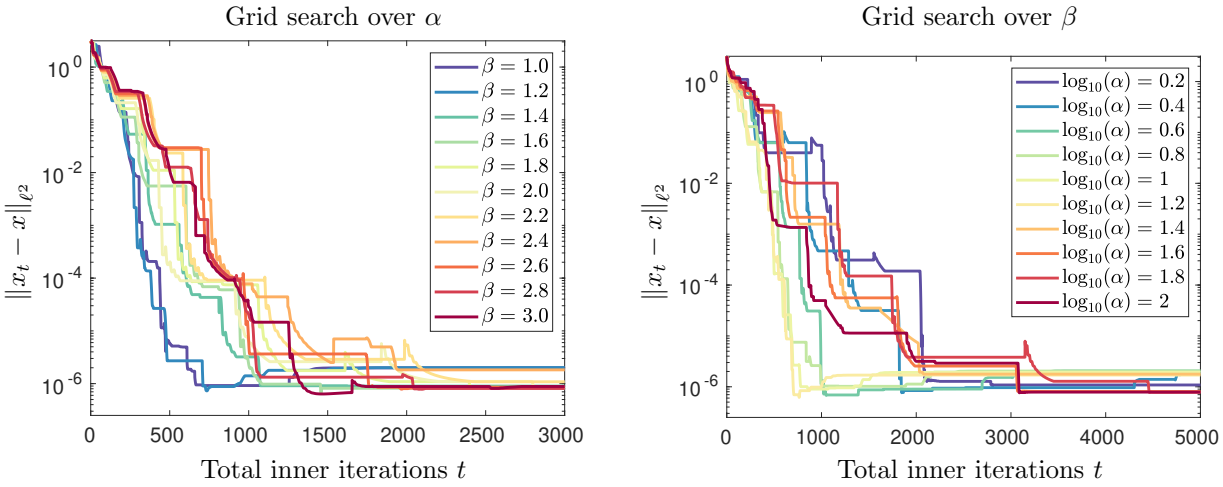

Figure 3: Reconstruction error of restarted primal-dual iteration for QCBP with $\varsigma = 10^{-6}$. Left: The restart scheme with grid search over α and various fixed β. Right: The restart scheme with grid search over β and various fixed α.

grid search over α, for fixed $\beta = 1$, and (ii) perform a grid search over both α and β. Both grid search schemes exhibit linear convergence, in agreement with Theorem 1.1. They converge less rapidly than the scheme with fixed (α, β), but require no empirical parameter tuning. Note that all restart schemes significantly outperform the unrestarted primal-dual iteration ("no restarts").

Next, we consider two cases of grid searching over exactly one sharpness constant and leaving the other fixed. Fig. 3 shows the results for fixed α with β grid search and fixed β with α grid search. Both yield linear decay, although at a slightly worse rate. A key point to note is the potential benefit of grid searching. Compare the reconstruction error with those for the fixed restart schemes in Fig. 2 with $\log_{10}(\alpha) \geq 1.2$ and $\beta = 1$. In the fixed constant scheme, these parameter choices stall the error. However, β grid search overcomes this and manages to reconstruct x within a tolerance proportional to ς after sufficiently many restarts.

Finally, Fig. 4 considers the effect on the restart schemes when changing the noise level ς. In all cases, the restart schemes linearly decay to a tolerance proportional to ς, and outperform the unrestarted primal-dual iterations.

5.2 Image reconstruction via TV minimization

In this experiment, we consider image reconstruction with Fourier measurements – a sensing modality with applications notably in Magnetic Resonance Imaging (MRI) [4]. Specifically, we consider the recovery of a vector $x \in \mathbb{R}^n$ from noisy Fourier measurements $y = Ax + e \in \mathbb{C}^m$, where $A \in \mathbb{C}^{m \times n}$ corresponds to a subsampled Fourier matrix and $e \in \mathbb{C}^m$ models noise or perturbations. The vector x is a vectorized complex 2-D image $X \in \mathbb{C}^{R \times R}$, where $n = R^2$ for some positive power-of-two integer R. The matrix A has the form $A = m^{-1/2} P_\Omega F$, where $F \in \mathbb{C}^{n \times n}$ is the 2-D discrete Fourier transform and $\Omega \subseteq n$ is a sampling mask with $|\Omega| = m$. Here, Ω defines the matrix $P_\Omega \in \mathbb{C}^{m \times n}$, which selects the rows of F by index according to the indices in Ω. Lastly, $\|e\|_{\ell^2} \leq \varsigma$ for some noise level $\varsigma > 0$. A widely used tool for reconstructing x from y is the *total variation (TV) minimization* problem

$$\min_{z \in \mathbb{C}^n} \|Vz\|_{\ell^1} \text{ subject to } \|Az - y\|_{\ell^2} \leq \varsigma,$$

where V is the 2-D (anisotropic) discrete gradient transform with periodic boundary conditions [3].

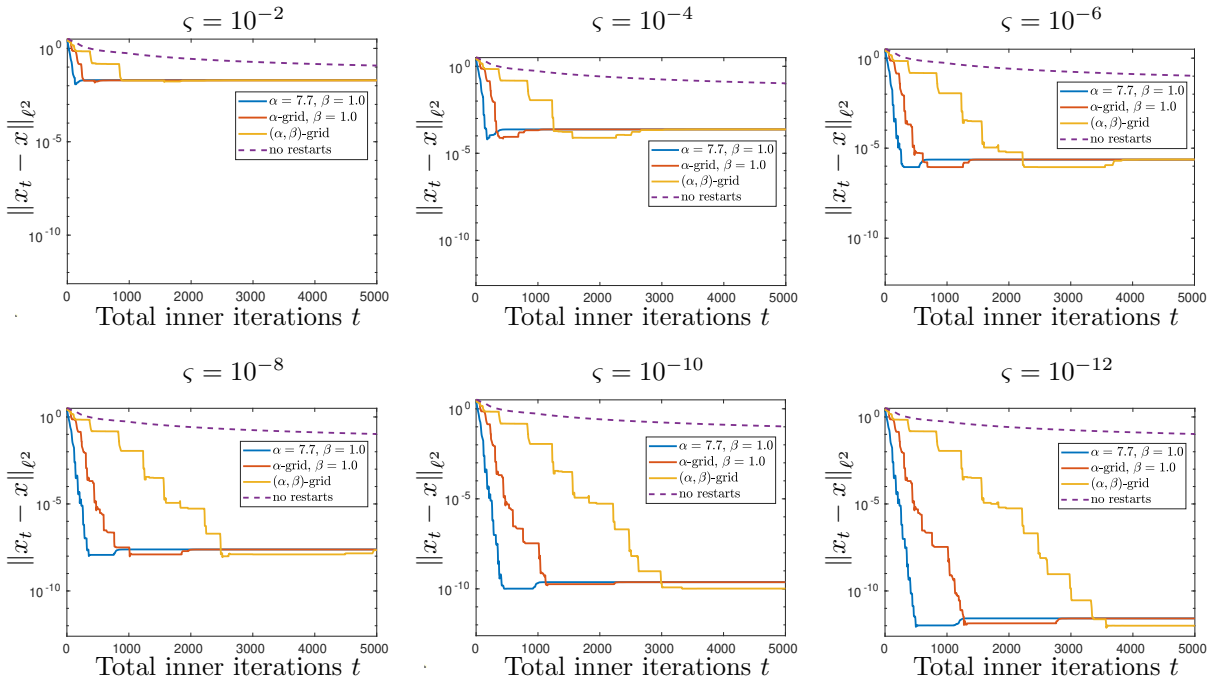

Figure 4: Reconstruction error of restarted primal-dual iteration for QCBP with $\varsigma = 10^{-2k}$ for $k = 1, 2, \ldots, 6$. Each plot includes the various (restarted and unrestarted) schemes.

Similar to the sparse recovery problem described in the previous section, the TV-Fourier image reconstruction problem can be shown to have the approximate sharpness condition (1.2) with high probability under a suitable random sampling pattern Ω. Stating and proving this is somewhat more involved, but can be done with a careful adaptation of the analysis within [3, Sec. 7.4].

5.2.1 Experimental setup

The first-order solver we use is NESTA (NESTerov's Algorithm), an accelerated projected gradient descent algorithm used to solve problems of the form

$$\min_{z \in \mathbb{C}^n} \|W^* z\|_{\ell^1} \text{ subject to } \|Az - y\|_{\ell^2} \leq \varsigma, \qquad W \in \mathbb{C}^{n \times m'},$$

where TV minimization is a special case with $W = V^\top$. NESTA is derived from Nesterov's method with smoothing, where the objective function $f(z) = \|W^* z\|_{\ell^1}$ is smoothed by replacing the ℓ^1-norm with its Moreau envelope. This yields a $1/\mu$-smooth approximation $f_\mu(z) = \|W^* z\|_{\ell^1, \mu}$ of f with parameters $(\|W^*\|_{\ell^2}^2, m'/2)$. Here $\|w\|_{\ell^1, \mu} = \sum_{i=1}^{m'} |w_i|_\mu$ for $w = (w_i)_{i=1}^{m'}$ and $|\cdot|_\mu$ is the complex Huber function (see, e.g., [61]). In particular, we have $\|V\|_{\ell^2} = 2\sqrt{2}$ for TV minimization in 2-D.

The second part of the derivation of NESTA is finding closed-form expressions for the update steps. In general, this is not possible to do except in special cases. However, NESTA considers A with orthonormal rows up to a constant factor, i.e., $AA^* = \nu I$ for some $\nu > 0$. Such an assumption yields a closed form for the update formulas and is not unreasonable since many forward operators in compressive imaging have orthonormal rows. For example, with the subsampled Fourier matrix we have $AA^* = (N/m)I$, and hence the desired property holds with $\nu = N/m$.

We reconstruct an $R \times R$ GPLU phantom image [42] with $R = 512$ so that the ambient dimension is $n = 512^2$. The noise e is uniformly sampled from an ℓ^2-ball of radius $\varsigma = 10^{-5}$, and so $\|e\|_{\ell^2} = \varsigma$. Two sampling masks are considered for the subsampled Fourier matrix A and are shown in Fig. 5.

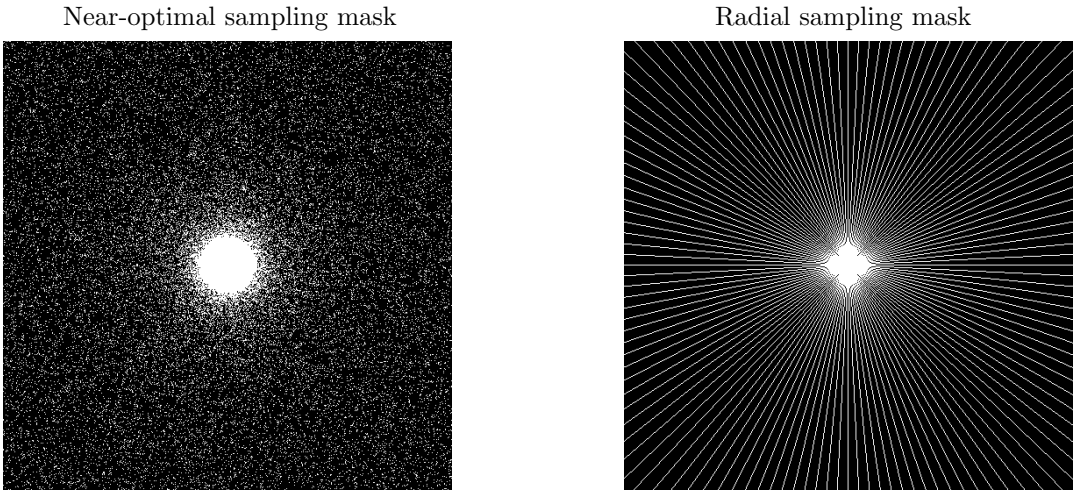

Figure 5: Sampling patterns for the Fourier measurements used in the image reconstruction experiments.

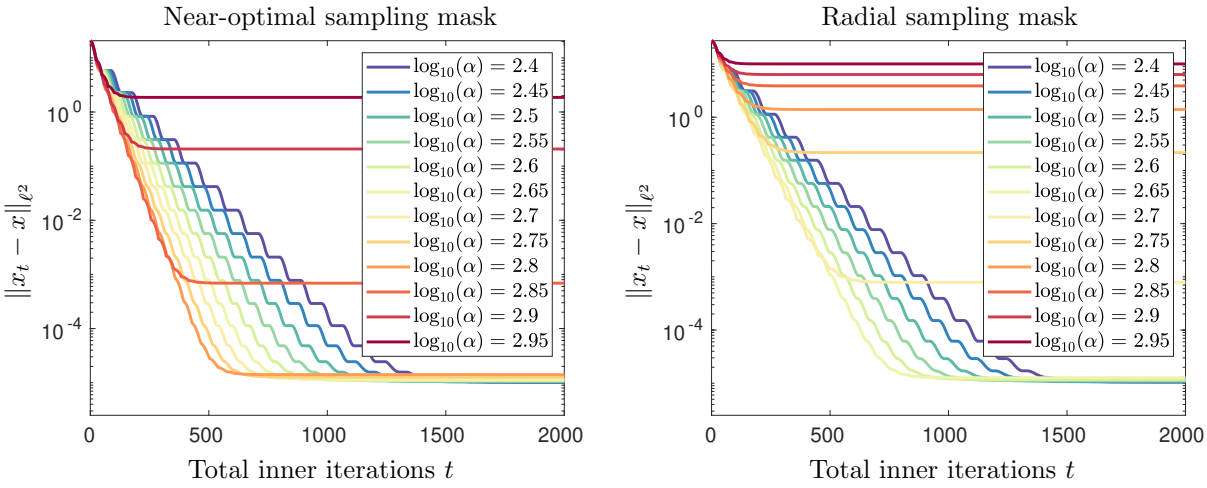

Figure 6: Reconstruction error of restarted NESTA for TV minimization with $\varsigma = 10^{-5}$, and with the near-optimal and radial sampling masks, respectively. The restart scheme uses fixed sharpness constants $\beta = 1$ and various α.

The first is a near-optimal sampling scheme [3, Sec. 4.2] and the second is a radial sampling scheme, where the latter is common in practice. Each mask yields approximately a 12.5% sampling rate. For the restart scheme, the objective function is $f(z) = \|Vx\|_{\ell^1}$ and the feasibility gap $g_Q \equiv 0$ since NESTA always produces feasible iterates. The smoothing parameters μ are handled directly by the restarting procedure and explicitly depend on $\epsilon_{i,j,U}$ (see Proposition 4.7). The main two experiments are done for each of the two sampling masks. Lastly, we choose $\alpha_0 = \sqrt{m}$, $\beta_0 = 1$. The choice of α_0 is motivated by [27, Theorem 6.3] which generalizes Proposition 5.2.

5.2.2 Results

First, we run the restart scheme with fixed sharpness constants (no grid search) corresponding to pairs (α, β) with $\beta = 1$ and various α values. The reconstruction error versus total inner iterations is plotted in Fig. 6 with near-optimal sampling (left) and radial sampling (right). The results are very similar to the first sparse recovery via QCBP experiment. Again, the rate of decay corresponds

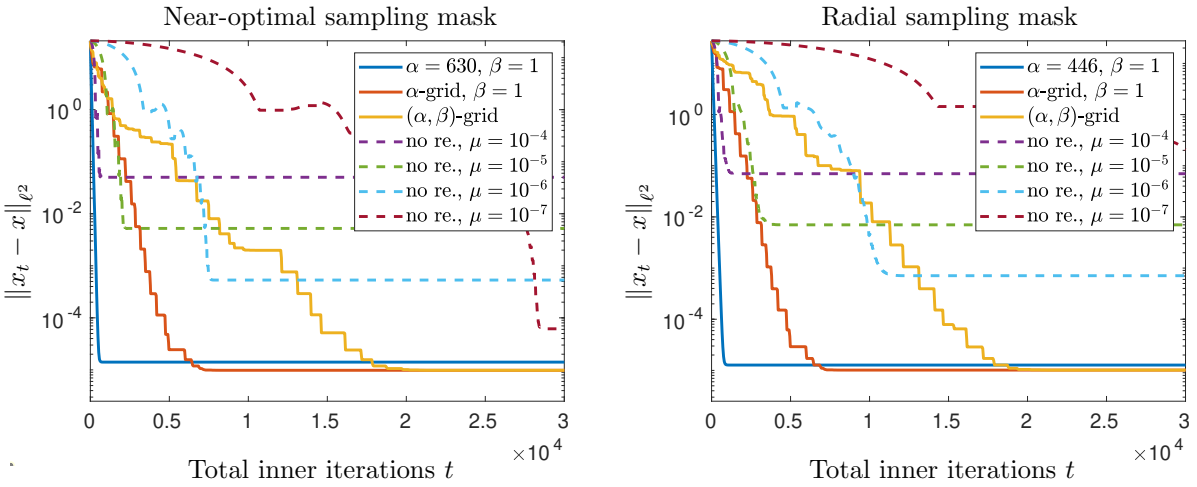

Figure 7: Reconstruction error of restarted NESTA for TV minimization with $\varsigma = 10^{-5}$, and with the near-optimal and radial sampling masks, respectively. Various different (restarted and unrestarted) schemes are used.

to linear decay as anticipated from Theorem 3.2. The convergence rate improves as α increases, up to a threshold (about $\alpha = 630$ for near-optimal sampling, and about $\alpha = 446$ for radial sampling), where afterwards the limiting tolerance increases steadily, yielding poor reconstruction results. This phenomenon is discussed in the first experiment of sparse recovery via QCBP. A key observation is how changing the sampling mask changes the threshold α value. This motivates using a grid search to avoid having to tune α as a parameter for different sampling masks.

In the second experiment, we compare the reconstruction errors of several restart schemes, together with standalone NESTA (i.e., no restarts) with various smoothing parameters. This is shown in Fig. 7 with near-optimal sampling (left) and radial sampling (right). The smoothing parameters used are $\mu = 10^i \varsigma$, $i \in \{-2, 1, 0, 1\}$. The results are analogous to the fourth experiment with sparse recovery via QCBP. We note that the radial sampling mask produces slightly lower convergence rates than the near-optimal scheme. Moreover, we observe that converging to the limiting tolerance of NESTA is sensitive to the choice of smoothing parameter μ. By making μ smaller, we better approximate the original problem and thus the reconstruction, but require more iterations to achieve a better approximation. In contrast, restarting NESTA via Algorithm 2 does not require any tuning of the smoothing parameter and outperforms the non-restarted algorithm.

5.3 Feature selection via SR-LASSO

Our final experiment considers feature selection via the *Square Root LASSO (SR-LASSO)* problem [2, 14, 15, 72]. Let $X \in \mathbb{R}^{m \times n}$ be a data matrix, where each row corresponds to a data point and each column corresponds to a feature, and $y \in \mathbb{R}^m$ the label vector for the data points. Since we wish to learn an *affine* mapping from data points to labels, we augment X by appending a new column consisting of ones, with the augmentation denoted by $A \in \mathbb{R}^{m \times (n+1)}$. Now fix $\lambda > 0$. Then we seek a vector $x \in \mathbb{R}^{n+1}$ that solves the SR-LASSO problem

$$\min_{z \in \mathbb{R}^{n+1}} \|Az - y\|_{\ell^2} + \lambda \|z\|_{\ell^1}.$$

An advantage of this problem over the classical LASSO is that it requires less tuning of the parameter λ as the problem instance or noise level changes. See [72] for discussion and recovery conditions for this problem. Feature selection is done by identifying the indices of close-to-zero entries of x,

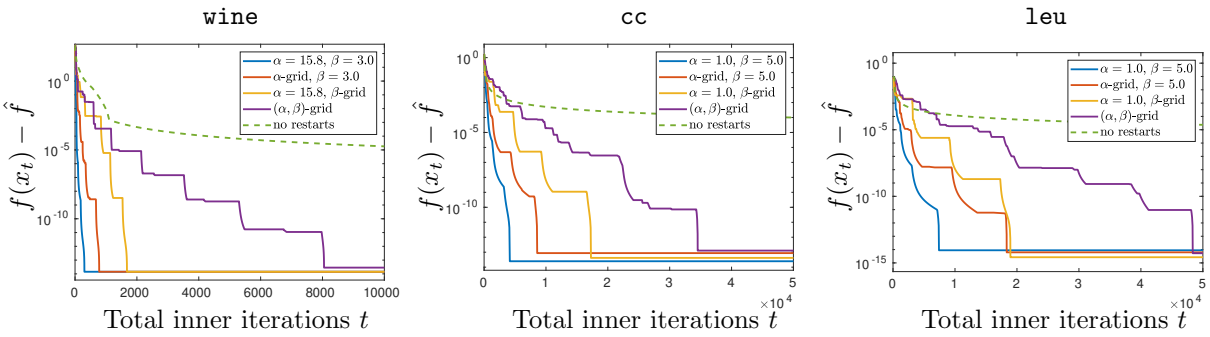

Figure 8: Objective error versus the total inner iteration of various (restarted and unrestarted) schemes of primal-dual iteration for SR-LASSO. The plots correspond to three different datasets.

which are the features to discard. This reduces the number of columns of X for future processing or analysis.

The SR-LASSO is a well-known tool in high-dimensional statistics. It can also be used for sparse recovery problems, in which case approximate sharpness follows (like it did with QCBP) from the rNSP (Definition 5.1) [27]. However, in the feature selection problem, properties such as the rNSP are unlikely to hold. In this case, more general recovery conditions for SR-LASSO (and LASSO), such as the *compatibility condition* [72], are more useful. Under these conditions, one also has approximate sharpness with unknown constants.

5.3.1 Setup

We use the unconstrained primal-dual iterations (Algorithm 5) to solve SR-LASSO. We can express SR-LASSO as (4.16) by

$$q \equiv 0, \quad g(x) = \lambda \|x\|_{\ell^1}, \quad h(Bx) = \|Bx - y\|_{\ell^2}, \quad B = A.$$

From this, the primal-dual updates can be computed explicitly. The proximal map τg is the shrinkage-thresholding operator and the proximal map of σh^* is a projection map onto the ℓ^2-ball. In either case, the proximal maps are straightforward to compute. For three different datasets, we compare the SR-LASSO objective error of various unrestarted and restarted schemes. The minimum of SR-LASSO for each dataset is computed using CVX [40, 41] with high precision and the SDPT3 solver and is used to compute the objective errors in Figs. 8 and 9.

We use three datasets: wine quality (`wine`) [29] with $m = 6497$ points and $n = 11$ features, colon cancer (`cc`) [26] with $m = 62$ points and $n = 2000$ features, and leukemia (`leu`) [26] with $m = 38$ points and $n = 7129$ features. The `wine` data corresponds to a regression task of predicting wine quality, `cc` and `leu` are two-class classification tasks of diagnosing illness based on data features. We use $\lambda = 3$, 2, and 4 for the `wine`, `cc`, and `leu` datasets, respectively. We measure sparsity s of $\hat{x}$ by interpreting an entry to be non-zero if its absolute value is greater than 10^{-5}. The values α_0 and β_0 are chosen empirically as estimates of the true sharpness constants α and β, respectively.

5.3.2 Results

Fig. 8 shows the performance of various restart schemes for this problem on the three different datasets. In all cases, the restarted schemes outperform the unrestarted scheme. The suitable values of α and β differ significantly across the datasets, indicating that the optimal sharpness parameters are problem-dependent. This further demonstrated in Fig. 9, where we show the restart scheme for various fixed β and grid search over α - the restart schemes with choices of $\beta > 1$ outperform

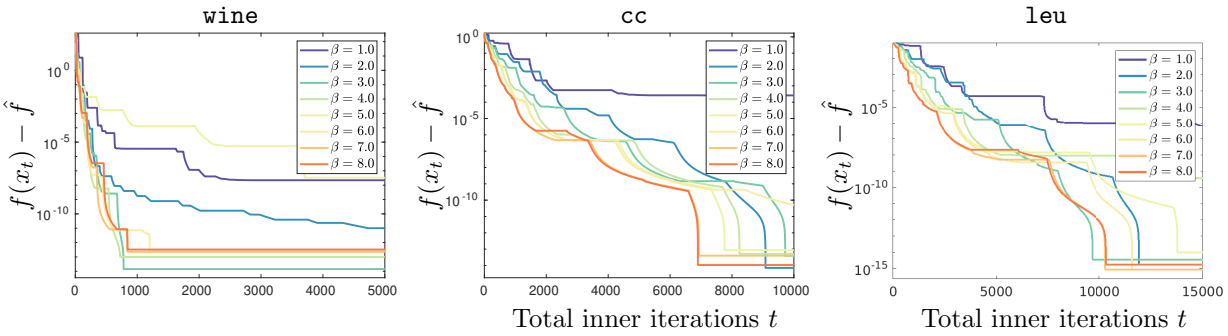

Figure 9: Objective error versus the total inner iteration of restarted primal-dual iteration for SR-LASSO. The plots correspond to grid search over α with various fixed β for three different datasets.

the schemes that use $\beta = 1$. This is in contrast to the sparse recovery example, where theory and experiment suggest $\beta = 1$ as a good choice. This phenomenon is unsurprising since the approximate sharpness condition (see (1.2)) for this problem is expected to be highly dependent on the data. Nonetheless, using our grid search scheme, we obviate the need for estimating or tuning these parameters.

6 Conclusion

We provided a framework for the optimal acceleration of first-order methods under approximate sharpness conditions. These conditions generalize sharpness by incorporating an unknown constant perturbation to the objective error, offering greater robustness to noise or model classes. Our scheme can achieve optimal convergence rates for a wide variety of problems, despite not assuming knowledge of the constants appearing in (1.2). Moreover, we do not require the first-order method to produce feasible iterates, a flexibility that is useful when employing methods such as primal-dual iterations. As illustrated by our numerical experiments, our schemes are also practical, and often lead to significant improvements over unrestarted schemes or restart schemes with poor parameter choices.

There are numerous possible avenues for future research and extensions of our framework. One avenue involves replacing the metric in (1.2) by a Bregman distance, and acceleration for convex optimization problems in Banach spaces. Another involves applications to (non-convex) bilevel optimization schemes. For saddle-point problems such as (4.9) and (4.17), it may be possible to develop similar restart schemes based on primal-dual gaps replacing $f(x) - \hat{f}$ in (1.2), see [5] and [32] for primal-dual gap sharpness and restart schemes in the cases of $\beta = 1$ and $\beta = 2$, respectively. See also [46, 47] for recent work on restarts based on gap functions for Frank-Wolfe algorithms. Finally, there is the extension of our restart schemes to handle stochastic first-order methods, including larger-scale machine learning problems.

A Miscellaneous proofs

In this section, we prove several results that were stated in Section 4.

A.1 Nesterov's method with smoothing

Proof of Lemma 4.6. Applying Lemma 4.3 with the function f_μ and using the second part of Definition 4.5 gives

$$f_\mu(x_k) - f_\mu(x) \leq \frac{4up(x; x_0)}{\mu k(k+1)\sigma_p}.$$

Now using both inequalities in the first part of Definition 4.5 gives the result. □

Proof of Proposition 4.7. Suppose that $x_0 \in Q$ with $d(x_0, \widehat{X}) \leq \delta$. Then by Lemma 4.6 with $\hat{x} \in \widehat{X} \subseteq Q$, we have

$$f(x_N) - \hat{f} \leq \frac{4up(\hat{x}; x_0)}{\mu N(N+1)\sigma_p} + v\mu.$$

Using $\frac{1}{N(N+1)} \leq \frac{1}{N^2}$, $\sigma_p = 1$ and $p(\hat{x}) \leq \frac{1}{2}\delta^2$ by choice of p, we get

$$f(x_N) - \hat{f} \leq \frac{2u\delta^2}{\mu N^2} + v\mu.$$

Substituting $\mu = \frac{\epsilon}{2v}$ and using that $N \geq 2\sqrt{2uv} \cdot \frac{\delta}{\epsilon}$ gives the result. □

A.2 Primal-dual iterations for unconstrained problems

Proof of Lemma 4.11. We use (4.10) and prove bounds on each of the terms on the left-hand side. First, we have

$$\mathcal{L}(X_k, y) = \langle BX_k, y\rangle_{\mathbb{R}} + q(X_k) + g(X_k) - h^*(y).$$

Since h is convex and lower semicontinuous, $h^{**} = h$. It follows that

$$h(BX_k) = \max_{y \in \mathbb{C}^m} \langle BX_k, y\rangle_{\mathbb{R}} - h^*(y) = -\min_{y \in \mathbb{C}^m} (h^*(y) - \langle BX_k, y\rangle_{\mathbb{R}}).$$

The objective function is convex and lower semicontinuous, and the set of minimizers is y such that

$$0 \in \partial\left(h^*(\cdot) - \langle \cdot, BX_k\rangle\right)(y) = \partial h^*(y) - BX_k.$$

Rearranging and using the Legendre–Fenchel identity, we deduce that this set of minimizers is precisely $\partial h(BX_k)$. It follows that

$$\mathcal{L}(X_k, y) = f(X_k), \qquad \forall y \in \partial h(BX_k). \tag{A.1}$$

Second, we have

$$\mathcal{L}(x, Y_k) = \langle Bx, Y_k\rangle_{\mathbb{R}} + q(x) + g(x) - h^*(Y_k).$$

The above argument shows that

$$h(Bx) = \max_{y \in \mathbb{C}^m} \langle Bx, y\rangle_{\mathbb{R}} - h^*(y) \geq \langle Bx, Y_k\rangle_{\mathbb{R}} - h^*(Y_k).$$

It follows that

$$\mathcal{L}(x, Y_k) \leq f(x). \tag{A.2}$$

The bound (4.11) now follows by combining (A.1) and (A.2). □

Proof of Proposition 4.12. First, consider general $\tau, \sigma > 0$ with $\tau(\sigma L_B^2 + L_q) = 1$. For input x_0 with $d(x_0, \widehat{X}) \leq \delta$, (4.13) and (4.12) imply that for $x \in \widehat{X}$,

$$f(X_N) - \hat{f} \leq \frac{1}{N}\left(\frac{\delta^2}{\tau} + \frac{L_h^2}{\sigma}\right) = \frac{1}{N}\left(\sigma\delta^2 L_B^2 + \frac{L_h^2}{\sigma} + \delta^2 L_q\right).$$

Choosing the step size $\sigma > 0$ to minimize the right-hand side leads to

$$\sigma = \frac{L_h}{\delta L_B}, \quad \tau = \frac{\delta}{L_B L_h + \delta L_q}, \quad f(X_N) - \hat{f} \leq \frac{\delta}{N}\left(2L_B L_h + \delta L_q\right).$$

Equations (4.14) and (4.15) now follow by taking $N = \left\lceil \frac{\delta}{\epsilon}\left(2L_B L_h + \delta L_q\right)\right\rceil$. □

A.3 Primal-dual iterations for constrained problems

Proof of Lemma 4.13. Using the same arguments as the proof of Lemma 4.11, (4.18) implies that for $y_2^{(0)} = 0$,

$$\begin{aligned} &f(X_k) - f(x) + \langle AX_k, y_2\rangle_{\mathbb{R}} - \sup_{z\in C}\langle z, y_2\rangle_{\mathbb{R}} - \langle Ax, [Y_k]_2\rangle_{\mathbb{R}} + \sup_{z\in C}\langle z, [Y_k]_2\rangle_{\mathbb{R}} \\ &\quad \leq \frac{1}{k}\left(\frac{\|x - x^{(0)}\|^2}{\tau} + \frac{\|y_1 - y_1^{(0)}\|^2}{\sigma_1} + \frac{\|y_2\|^2}{\sigma_2}\right), \quad \forall x \in \mathbb{C}^n,\ y_1 \in \partial h(BX_k),\ y_2 \in \mathbb{C}^{m'}. \end{aligned}$$

If $x \in Q$, then

$$-\langle Ax, [Y_k]_2\rangle_{\mathbb{R}} + \sup_{z\in C}\langle z, [Y_k]_2\rangle_{\mathbb{R}} \geq 0.$$

Let $\hat{z} \in C$ be of minimal distance to AX_k and let y_2 be a multiple of $AX_k - \hat{z}$ such that y_2 has norm κ. Since C is convex, the following holds [10, Theorem 6.41]

$$\langle z, y_2\rangle_{\mathbb{R}} \leq \langle \hat{z}, y_2\rangle_{\mathbb{R}}, \quad \forall z \in C.$$

It follows that

$$\langle AX_k, y_2\rangle_{\mathbb{R}} - \sup_{z\in C}\langle z, y_2\rangle_{\mathbb{R}} \geq \langle AX_k - \hat{z}, y_2\rangle_{\mathbb{R}} = \kappa \cdot \inf_{z\in C}\|AX_k - z\| = g_Q(\kappa; X_k).$$

Combining the inequalities yields (4.19). □

Proof of Proposition 4.14. First, consider general $\tau, \sigma_1, \sigma_2 > 0$ with $\tau(\sigma_1 L_B^2 + \sigma_2 L_A^2 + L_q) = 1$. For input x_0 with $d(x_0, \widehat{X}) \leq \delta$, we argue as in the proof of Proposition 4.12 (but now using Lemma 4.13) to obtain

$$f(X_N) - \hat{f} + g_Q(\kappa; X_N) \leq \frac{1}{N}\left(\frac{\delta^2}{\tau} + \frac{L_h^2}{\sigma_1} + \frac{\kappa^2}{\sigma_2}\right) = \frac{1}{N}\left(\sigma_1\delta^2 L_B^2 + \frac{L_h^2}{\sigma_1} + \sigma_2\delta^2 L_A^2 + \frac{\kappa^2}{\sigma_2} + \delta^2 L_q\right). \tag{A.3}$$

Optimizing the proximal step sizes leads to

$$\tau = \frac{\delta}{\kappa L_A + L_h L_B + \delta L_q}, \quad \sigma_1 = \frac{L_h}{\delta L_B}, \quad \sigma_2 = \frac{\kappa}{\delta L_A}.$$

Substituting these values into (A.3) leads to

$$f(X_N) - \hat{f} + g_Q(X_N) \leq \frac{\delta}{N}\left(2\kappa L_A + 2L_h L_B + \delta L_q\right).$$

The rest of the proof follows the same argument as the proof of Proposition 4.12. □

References

[1] B. Adcock, S. Brugiapaglia, N. Dexter, and S. Moraga. On efficient algorithms for computing near-best polynomial approximations to high-dimensional, Hilbert-valued functions from limited samples. *arXiv preprint arXiv:2203.13908*, 2022.

[2] B. Adcock, S. Brugiapaglia, and C. G. Webster. *Sparse Polynomial Approximation of High-Dimensional Functions*. Comput. Sci. Eng. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2022.

[3] B. Adcock, N. Dexter, and Q. Xu. Improved recovery guarantees and sampling strategies for TV minimization in compressive imaging. *SIAM J. Imaging Sci.*, 14(3):1149–1183, 2021.

[4] B. Adcock and A. Hansen. *Compressive Imaging: Structure, Sampling, Learning*. CUP, 2021.

[5] D. Applegate, O. Hinder, H. Lu, and M. Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *Mathematical Programming*, pages 1–52, 2022.

[6] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*. Elsevier, 2018.

[7] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-łojasiewicz inequality. *Math. Oper. Res.*, 35(2):438–457, 2010.

[8] A. Auslender and J.-P. Crouzeix. Global regularity theorems. *Math. Oper. Res.*, 13(2):243–253, 1988.

[9] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale's 9th problem. *arXiv preprint arXiv:2110.15734*, 2021.

[10] A. Beck. *First-order methods in optimization*. SIAM, 2017.

[11] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

[12] S. Becker, J. Bobin, and E. J. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM J. Imaging Sci.*, 4(1):1–39, 2011.

[13] S. Becker, E. J. Candès, and M. C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Math. Program. Comput.*, 3(3):165, 2011.

[14] A. Belloni, V. Chernozhukov, and L. Wang. Square-root LASSO: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[15] A. Belloni, V. Chernozhukov, and L. Wang. Pivotal estimation via square-root LASSO in nonparametric regression. *Ann. Statist.*, 42(2):757–788, 2014.

[16] A. Ben-Tal and A. Nemirovski. Lectures on modern convex optimization. 2020/2021.

[17] J. Bolte, A. Daniilidis, and A. Lewis. The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM J. Optim.*, 17(4):1205–1223, 2007.

[18] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Math. Program.*, 165(2):471–507, 2017.

[19] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 146(1):459–494, 2014.

[20] J. Burke and S. Deng. Weak sharp minima revisited Part I: basic theory. *Control Cybernet.*, 31:439–469, 2002.

[21] J. V. Burke and M. C. Ferris. Weak sharp minima in mathematical programming. *SIAM J. Control Optim.*, 31(5):1340–1359, 1993.

[22] A. Chambolle, M. J. Ehrhardt, P. Richtárik, and C. Schonlieb. Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications. *SIAM J. Optim.*, 28(4), 2018.

[23] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40(1):120–145, 2011.

[24] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[25] A. Chambolle and T. Pock. On the ergodic convergence rates of a first-order primal–dual algorithm. *Math. Program.*, 159(1-2):253–287, 2016.

[26] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.

[27] M. J. Colbrook. WARPd: A linearly convergent first-order primal-dual algorithm for inverse problems with approximate sharpness conditions. *SIAM Journal on Imaging Sciences*, 15(3):1539–1575, 2022.

[28] M. J. Colbrook, V. Antun, and A. C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and smale's 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, 2022.

[29] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Wine Quality. UCI Machine Learning Repository, 2009.

[30] A. d'Aspremont, D. Scieur, A. Taylor, et al. Acceleration methods. *Foundations and Trends in Optimization*, 5(1-2):1–245, 2021.

[31] E. Esser, X. Zhang, and T. F. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM J. Imaging Sci.*, 3(4), 2010.

[32] O. Fercoq. Quadratic error bound of the smoothed gap and the restarted averaged primal-dual hybrid gradient. *arXiv preprint arXiv:2206.03041*, 2022.

[33] O. Fercoq and Z. Qu. Restarting accelerated gradient methods with a rough strong convexity estimate. *arXiv:1609.07358*, 2016.

[34] O. Fercoq and Z. Qu. Adaptive restart of accelerated gradient methods under local quadratic growth condition. *IMA Journal of Numerical Analysis*, 39(4):2069–2095, 2019.

[35] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*. Springer, 2013.

[36] P. Frankel, G. Garrigos, and J. Peypouquet. Splitting methods with variable metric for Kurdyka–łojasiewicz functions and general convergence rates. *J. Optim. Theory Appl.*, 165(3):874–900, 2015.

[37] R. M. Freund and H. Lu. New computational guarantees for solving convex optimization problems with first order methods, via a function growth condition measure. *Math. Program.*, 170(2):445–477, 2018.

[38] L. E. Gazdag and A. C. Hansen. Generalised hardness of approximation and the SCI hierarchy - On determining the boundaries of training algorithms in AI. *arXiv preprint arXiv:2209.06715*, 2022.

[39] P. Giselsson and S. Boyd. Monotonicity and restart in fast gradient methods. In *IEEE Conf Decis Control*, pages 5058–5063. IEEE, 2014.

[40] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[41] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[42] M. Guerquin-Kern, L. Lejeune, K. P. Pruessmann, and M. Unser. Realistic analytical phantoms for parallel Magnetic Resonance Imaging. *IEEE Trans. Med. Imag.*, 31(3):626–636, 2012.

[43] A. J. Hoffman. On approximate solutions of systems of linear inequalities. *J. Research Nat. Bur. Standards*, 49(4), 1952.

[44] A. Iouditski and Y. Nesterov. Primal-dual subgradient methods for minimizing uniformly convex functions. *arXiv preprint arXiv:1401.1792*, 2014.

[45] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-łojasiewicz condition. In *Mach Learn Knowl Discov Databases*, pages 795–811. Springer, 2016.

[46] T. Kerdreux, A. d'Aspremont, and S. Pokutta. Restarting Frank-Wolfe. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1275–1283. PMLR, 2019.

[47] T. Kerdreux, A. d'Aspremont, and S. Pokutta. Restarting frank–wolfe: Faster rates under hölderian error bounds. *Journal of Optimization Theory and Applications*, 192(3):799–829, 2022.

[48] D. Kim and J. A. Fessler. Optimized first-order methods for smooth convex minimization. *Mathematical programming*, 159(1):81–107, 2016.

[49] Q. Lin and L. Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. In *International Conference on Machine Learning*, pages 73–81. PMLR, 2014.

[50] S. Lojasiewicz. Une propriété topologique des sous-ensembles analytiques réels. *Les équations aux dérivées partielles*, 117:87–89, 1963.

[51] O. L. Mangasarian. A condition number for differentiable convex inequalities. *Math. Oper. Res.*, 10(2):175–179, 1985.

[52] I. Necoara, Y. Nesterov, and F. Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Math. Program.*, 175(1):69–107, 2019.

[53] A. S. Nemirovskii and Y. E. Nesterov. Optimal methods of smooth convex minimization. *USSR Comput. Math. Math. Phys.*, 25(2):21–30, 1985.

[54] A. S. Nemirovskij and D. B. Yudin. Problem complexity and method efficiency in optimization. 1983.

[55] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[56] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, May 2005.

[57] Y. Nesterov. Gradient methods for minimizing composite functions. *Math. Program.*, 140(1):125–161, 2013.

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

[59] Y. Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[60] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR,*, volume 269, pages 543–547, 1983.

[61] M. Neyra-Nesterenko and B. Adcock. NESTANets: Stable, accurate and efficient neural networks for analysis-sparse inverse problems. *arXiv:2203.00804*, 2022.

[62] B. O'donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Found. Comput. Math.*, 15(3):715–732, 2015.

[63] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the Mumford–Shah functional. In *IEEE Int Conf Comput Vis*, pages 1133–1140. IEEE, 2009.

[64] J. Renegar. "efficient" subgradient methods for general convex optimization. *SIAM Journal on Optimization*, 26(4):2649–2676, 2016.

[65] J. Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1):1–35, 2019.

[66] J. Renegar and B. Grimmer. A simple nearly optimal restart scheme for speeding up first-order methods. *Foundations of Computational Mathematics*, pages 1–46, 2021.

[67] S. M. Robinson. An application of error bounds for convex programming in a linear space. *SIAM J. Control*, 13(2):271–273, 1975.

[68] V. Roulet, N. Boumal, and A. d'Aspremont. Computational complexity versus statistical performance on sparse recovery problems. *Inf. Inference*, 9(1):1–32, 2020.

[69] V. Roulet and A. d'Aspremont. Sharpness, restart, and acceleration. *SIAM J. Optim.*, 30(1):262–289, 2020.

[70] O. Rynkiewicz. Lower bounds and primal-dual methods for affinely constrained convex optimization under metric subregularity, 2020.

[71] W. Su, S. Boyd, and E. Candes. A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. *Advances in neural information processing systems*, 27, 2014.

[72] S. van de Geer. *Estimation and Testing Under Sparsity: École d'Été de Probabilités de Saint-Flour XLV – 2015*, volume 2159 of *Lecture Notes in Math.* Springer, Cham, Switzerland, 2016.