A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization

Hongzhou Lin, Julien Mairal, Zaid Harchaoui

▶ **To cite this version:**

Hongzhou Lin, Julien Mairal, Zaid Harchaoui. A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization. 2017. hal-01376079v2

HAL Id: hal-01376079
https://hal.science/hal-01376079v2

Preprint submitted on 5 Apr 2017 (v2), last revised 28 Jan 2019 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization*

Hongzhou Lin
Inria
hongzhou.lin@inria.fr

Julien Mairal
Inria
julien.mairal@inria.fr

Zaid Harchaoui
University of Washington
zaid@uw.edu

April 5, 2017

Abstract

We propose a generic approach to accelerate gradient-based optimization algorithms with quasi-Newton principles. The proposed scheme, called QuickeNing, can be applied to incremental first-order methods such as stochastic variance-reduced gradient (SVRG) or incremental surrogate optimization (MISO). It is also compatible with composite objectives, meaning that it has the ability to provide exactly sparse solutions when the objective involves a sparsity-inducing regularization. QuickeNing relies on limited-memory BFGS rules, making it appropriate for solving high-dimensional optimization problems. Besides, it enjoys a worst-case linear convergence rate for strongly convex problems. We present experimental results where QuickeNing gives significant improvements over competing methods for solving large-scale high-dimensional machine learning problems.

1 Introduction

Convex composite optimization arises in many scientific fields, such as image and signal processing or machine learning. It consists of minimizing a real-valued function composed of two convex terms:

$$\min_{x\in\mathbb{R}^d} \Big\{ f(x) \triangleq f_0(x) + \psi(x) \Big\}, \tag{1}$$

where f_0 is smooth with Lipschitz continuous derivatives, and ψ is a regularization function which is not necessarily differentiable. For instance, the ℓ_1-norm $\psi(x) = \|x\|_1$ is popular in image processing [14, 29] for encouraging sparse solutions; composite minimization also encompasses constrained minimization when considering extended-valued indicator functions ψ that may take the value $+\infty$ outside of a convex set $\mathcal{C}$ and 0 inside (see [22]). In general, algorithms that are dedicated to composite optimization only require to be able to compute efficiently the proximal operator of ψ:

$$p_\psi(y) \triangleq \operatorname*{arg\,min}_{x\in\mathbb{R}^d} \left\{ \psi(x) + \frac{1}{2}\|x-y\|^2 \right\},$$

where $\|\cdot\|$ denotes the Euclidean norm. Note that when ψ is an indicator function, the proximal operator corresponds to the simple Euclidean projection.

To solve (1), significant efforts have been devoted to (i) extending techniques for smooth optimization to deal with composite terms; (ii) exploiting the underlying structure of the problem—is f a finite sum of independent terms? Is ψ separable in different blocks of coordinates? (iii) exploiting the local curvature of the smooth term f to achieve faster convergence than gradient-based approaches when the dimension d is very

*This work was supported by the ERC grant SOLARIS (number 714381), a grant from ANR (MACARON project ANR-14-CE23-0003-01), the program "Learning in Machines and Brains" (CIFAR), and the project Titan (CNRS-Mastodons).

large. Typically, the first point is well understood in the context of optimal first-order methods, see [1, 36], and the last point is tackled with effective heuristics such as L-BFGS when the problem is smooth [27, 37]. Yet, solving all these problems at the same time is challenging: this is precisely the focus of this paper.

In particular, a problem of interest that has first motivated our work is that of empirical risk minimization (ERM); the problem arises in machine learning and can be formulated as the minimization of a composite function $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\}, \tag{2}$$

where the functions f_i are convex and smooth with Lipschitz continuous derivatives, and ψ is a composite term, possibly non-smooth. The function f_i measures the fit of some model parameters x to a specific data point indexed by i, and ψ is a regularization penalty to prevent over-fitting. To exploit the sum structure of f, a large number of randomized incremental gradient-based techniques have been proposed, such as SAG [45], SAGA [10], SDCA [46], SVRG [47], Finito [11], or MISO [28]. These approaches access a single gradient $\nabla f_i(x)$ at every iteration instead of the full gradient $(1/n) \sum_{i=1}^{n} \nabla f_i(x)$ and achieve lower computational complexity in expectation than optimal first-order methods [1, 36] under a few assumptions. Yet, these methods are unable to exploit the curvature of the objective function; this is indeed also the case for variants that are accelerated in the sense of Nesterov [16, 26, 46].

To tackle (2), dedicated first-order methods are often the default choice in machine learning, but it is also known that standard Quasi-Newton approaches can sometimes be surprisingly effective in the smooth case—that is when $\psi = 0$, see, e.g., [45] for extensive benchmarks. Since the dimension of the problem d is typically very large ($d \geq 10\,000$), "limited memory" variants of these algorithms, such as L-BFGS, are necessary to achieve the desired scalability [27, 37]. The theoretical guarantees offered by L-BFGS are somewhat weak, meaning that it does not outperform accelerated first-order methods in terms of worst-case convergence rate, and also it is not guaranteed to correctly approximate the Hessian of the objective. Yet, it remains one of the greatest practical success of smooth optimization. Adapting L-BFGS to composite and structured problems, such as the finite sum of functions (2), is of utmost importance nowadays.

For instance, there have been several attempts to develop a proximal Quasi-Newton method [7, 24, 43, 48]. These algorithms typically require computing many times the proximal operator of ψ with respect to a variable metric, which may be as computationally demanding as solving the original problem. Quasi-Newton steps were also incorporated as local search steps into accelerated first-order methods to further enhance their numerical performance [19]. More related to our work, L-BFGS is combined with SVRG for minimizing smooth finite sums in [20]. The scope of our approach is broader beyond the case of SVRG. We present a generic Quasi-Newton scheme, applicable to a large-class of first-order methods for composite optimization, including other incremental algorithms [10, 11, 28, 45, 46] and block coordinate descent methods [39, 40]

More precisely, our main contribution is a generic meta-algorithm, called QuickeNing (the letters "Q" and "N" stand for Quasi-Newton), which uses a given optimization method to solve a sequence of auxiliary problems up to some appropriate accuracy, resulting in faster global convergence in practice. The resulting scheme admits a simple interpretation: *it may be seen as applying the L-BFGS algorithm with inexact (but accurate enough) gradients to the Moreau-Yosida regularization of the objective.* As a result, our approach is (i) generic, as stated previously; (ii) despite the smoothing of the objective, the sub-problems that we solve are composite ones, which may lead to exactly sparse iterates when a sparsity-inducing regularization is involved, e.g., the ℓ_1-norm; (iii) it admits a worst-case linear convergence rate for strongly convex problems similar to that of gradient descent, which is typically the best guarantees obtained for L-BFGS schemes in the literature; (iv) it is easy to use and does not require using a line search algorithm, which is sometimes computationally expensive and difficult to calibrate in classical Quasi-Newton methods.

The idea of combining second-order or quasi-Newton methods with Moreau-Yosida regularization is in fact relatively old. It may be traced back to variable metric proximal bundle methods [9, 18, 30], which use BFGS updates on the Moreau-Yosida smoothing of the objective and bundle methods to approximately solve the corresponding sub-problems. Our approach revisits this principle with *a limited-memory variant* (to deal with large dimension d), with *an alternative strategy to line search schemes*—which is useful when f is a

large sum of n functions as in (2)—and with *restart strategies for the sub-problems with a global complexity analysis* that is more relevant than convergence rates that do not take into account the cost per iteration.

To demonstrate the effectiveness of our scheme in practice, we evaluate QuickeNing on regularized logistic regression and regularized least-squares, which smooth and non smooth regularization penalities such as the Elastic-Net [49]. We use large-scale machine learning datasets and show that QuickeNing performs at least as well as the recently proposed Catalyst [26] and as the classical L-BFGS scheme in all experiments, and significantly outperforms them in many cases.

The paper is organized as follows: Section 2 presents related work on Quasi-Newton methods such as L-BFGS; we introduce QuickeNing as well as basic properties of the Moreau-Yosida regularization in Section 3, and we provide a convergence analysis in Section 4; Section 5 is devoted to experiments and Section 6 concludes the paper.

2 Related work and preliminaries

The history of quasi-Newton methods can be traced back to the 1950's [4, 23, 38]. Quasi-Newton methods often lead to significantly faster convergence in practice compared to simpler gradient-based methods for solving smooth optimization problems [44]. Yet, a theoretical analysis of quasi-Newton methods that explains their impressive empirical behavior on a wide range of problems is still an open topic. Here, we review briefly the well-known BFGS algorithm in Section 2.1, its limited memory variant [37], and a few recent extensions. Then, we present earlier works that combine proximal point and Quasi-Newton algorithms in Section 2.2, which is related to the strategy we use in our paper.

2.1 Quasi-Newton methods for smooth optimization

The most popular Quasi-Newton method is BFGS, named after its inventors (Broyden-Fletcher-Goldfarb-Shanno), and its limited variant L-BFGS [38]. These approaches will be the workhorses of the QuickeNing meta-algorithm. Consider a smooth convex objective f to be minimized, the BFGS method constructs at iteration k a couple (x_k, B_k) with the following update:

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k) \quad \text{and} \quad B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}, \tag{3}$$

where α_k is a suitable stepsize and

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

The condition $y_k^\top s_k > 0$ and the positive definiteness of B_k are guaranteed as soon as f is strongly convex. To determine the stepsize α_k, Wolfe's line-search is a simple choice which provides linear convergence rate in the worst case. In addition, if the objective is twice differentiable and the Hessian is Lipschitz continuous, the convergence is asymptotically superlinear [38].

The limited memory variant L-BFGS [37] overcomes the issue of storing B_k for large d, by replacing it by another positive definite matrix—say $\bar{B}_k$—which can be built from a "generating list" of at most l pairs of vectors $\{(s_i^k, y_i^k)\}_{i=0\dots j}$ along with an initial diagonal matrix $\bar{B}_0$. Formally, $\bar{B}_k$ can be computed by applying at most l times a recursion similar to (3) involving all pairs of the generating list. Between iteration k and $k+1$, the generating list is incrementally updated, by removing the oldest pair in the list (when $j = l$) and adding a new one. What makes the approach appealing is the ability of computing $H_k z = B_k^{-1} z$ for any vector z with only $O(ld)$ floating point operations instead of $O(d^3)$ for a naive implementation with matrix inversion. The price to pay is that superlinear convergence becomes out of reach in contrast to BFGS.

L-BFGS is thus appropriate for high-dimensional problems (when d is large), but it still requires computing the full gradient at each iteration, which may be cumbersome in the large sum setting (2). This motivated stochastic counterparts of the Quasi-Newton method (SQN) [6]. The direct application substituting the full gradient $\nabla f(x_k)$ by a stochastic estimate is unfortunately not convergent. Instead, the SQN method [6] uses

a product of a sub-sampled Hessian and s_k to approximate y_k. SQN can be complemented by a variance reduction scheme such as SVRG [20, 31].

2.2 Combining the proximal point algorithm and Quasi-Newton

There are several ways to extend Quasi-Newton methods for nonsmooth optimization. A first approach consists in minimizing successive quadratic approximations. A major drawback is then the computation of the proximal operator. Since B_k is dense and changes over the iterations, an exact solution to the proximal operator is usually not available. An inexact variant has been analyzed in [7], but it is unclear whether or not the quadratic approximation could be solved inexactly in an efficient manner.

A second approach consists in applying Quasi-Newton methods after Moreau-Yosida smoothing or equivalently a proximal point algorithm on the objective function [5, 9, 17, 18]. The underlying idea is to first smooth and improve the conditioning of the objective function, then apply a Quasi-Newton method on the resulting surrogate objective function. This implies minimizing a sequence of sub-problems. Formally, the Moreau-Yosida regularization of the objective f is defined as the infimal convolution

$$F(x) \triangleq \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2} \|z - x\|^2 \right\}, \tag{4}$$

where κ is a positive scalar. The image $p(x)$ of x under the proximal mapping is defined as the solution of the previous problem, which is unique by strong convexity. The Moreau-Yosida regularization admits classical properties characterizing the smoothness of the Moreau envelope; see [25] for elementary proofs.

Proposition 1 (Basic properties of the Moreau-Yosida regularization). *Let us consider the Moreau-Yosida regularization F of the convex function f defined in (4); Then, the following statements hold*

1. *Minimizing f and F are equivalent in the sense that*

$$\min_{x \in \mathbb{R}^p} F(x) = \min_{x \in \mathbb{R}^p} f(x),$$

   and the solution set of the two above problems coincide with each other.

2. *F is continuously differentiable even when f is not and*

$$\nabla F(x) = \kappa(x - p(x)), \tag{5}$$

   where $p(x)$ is the proximal mapping, which we have defined previously as the solution of (4). Moreover the gradient ∇F is Lipschitz continuous with constant $L_F = \kappa$.

3. *F is convex; moreover, when f is μ-strongly convex with respect to the Euclidean norm, F is μ_F-strongly convex with $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$.*

Interestingly, F inherits all the convex properties of f and more importantly it is always continuously differentiable. Moreover, F can be arbitrarily well conditioned by choosing a small value of κ. Naturally, a naive approach for minimizing a possibly non-smooth function f is to apply a first-order method on F since both functions admit the same solutions, which yields well-known algorithms.

The proximal point algorithm. Consider gradient descent with step size $1/L_F = 1/\kappa$ to minimize F:

$$x_{k+1} = x_k - \frac{1}{\kappa} \nabla F(x_k).$$

By rewriting the gradient $\nabla F(x_k)$ as $\kappa(x_k - p(x_k))$, we obtain

$$x_{k+1} = p(x_k) = \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2} \|z - x_k\|^2 \right\}. \tag{6}$$

This is exactly the proximal point algorithm [41].

Accelerated inexact proximal point algorithm. Since gradient descent on F yields the proximal point algorithm, it is also natural to apply an accelerated first-order method to get faster convergence. To that effect, Nesterov's algorithm [32] uses a two-stage update:

$$x_{k+1} = y_k - \frac{1}{\kappa}\nabla F(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k),$$

where β_{k+1} is a specific extrapolation parameter (see [33]). We may now rewrite the update using the value of ∇F given in (5), which gives:

$$x_{k+1} = p(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k)$$

This is known as the accelerated proximal point algorithm introduced by Güler [21]. These techniques are conceptually interesting, but they are not directly applicable in general, since computing the gradient of F requires the exact solution $p(x)$ of (4), for which no closed-form is typically available. It is thus necessary to use an approximate solution, which implies defining an inexactness criterion that is easy to check and to control the accuracy of the gradient approximation to ensure convergence. This is notably the approach studied in [12, 21, 42, 26].

Quasi-Newton after the Moreau-Yosida regularization. As mentioned previously, Quasi-Newton approaches have also been applied after Moreau-Yosida smoothing in various ways. For instance, in [9], the sub-problems are solved inexactly by proximal bundle methods, and in [17] by using a sophisticated sufficient descent criterion. The proposed QuickeNing algorithm, which will be presented in the next section, follows this line of research. In contrast to previous works, we propose several restart strategies for solving the sub-problems and give a convergence analysis taking into account the inner-loop complexity. Furthermore, we propose a simple strategy to remove the need of a line-search scheme and simple rules to set the parameters.

3 QuickeNing: a Quasi-Newton meta-algorithm

We now present the QuickeNing method in Algorithm 1, which consists of applying L-BFGS rules on the smoothed objective F with inexact gradients. Each gradient approximation is the result of a minimization problem tackled with the algorithm $\mathcal{M}$, used as a sub-routine. The outer loop of the algorithm performs L-BFGS quasi-Newton updates. The method $\mathcal{M}$ can be any algorithm of the user's choice, as long as it enjoys linear convergence rate for strongly convex problems. More details about the choice of the parameter κ and about the inexactness criterion to use will be given next.

3.1 The main algorithm

We now discuss the main algorithm components and its main features.

Outer-loop: inexact L-BFGS. The L-BFGS rule is the standard one and consists in updating incrementally a generating list of vectors $\{(s_i, y_i)\}_{i=1\dots j}$, which implicitly defines the L-BFGS matrix. We use here the two-loop recursion detailed in [38, Algorithm 7.4] and use skipping steps when the condition $s_i^\top y_i > 0$ is not satisfied, in order to ensure the positive-definiteness of the L-BFGS matrix H_k (see [15]). A more stringent condition for skipping steps may also be used for specific methods $\mathcal{M}$, as discussed in Appendix A.

Inner-loop: approximate Moreau-Yosida envelope. The minimization algorithm $\mathcal{M}$ is used to compute an approximate Moreau-Yosida envelope of the objective. The procedure ApproxGradient() calls the minimization algorithm $\mathcal{M}$ to minimize the sub-problem (8). When the problem is solved exactly, the function returns the exact values $g = \nabla F(x)$, $F_z = F(x)$ and $z = p(x)$. Otherwise, a stopping criterion should be used. In this paper, we consider the three following cases:

Algorithm 1 QuickeNing

input Initial point x_0 in $\mathbb{R}^p$; number of iterations K; parameter $\kappa > 0$; minimization algorithm $\mathcal{M}$.

1: Initialization: $(g_0, F_0, z_0) = \mathsf{ApproxGradient}\,(x_0, \mathcal{M})$; L-BFGS matrix $H_0 = (1/\kappa)I$.
2: **for** $k = 0, \ldots, K-1$ **do**
3: Perform the Quasi-Newton step

$$x_{\text{test}} = x_k - H_k g_k.$$

4: Estimate the new gradient and the Moreau-Yosida function value

$$(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \mathsf{ApproxGradient}\,(x_{\text{test}}, \mathcal{M})\ .$$

5: **if** sufficient approximate decrease is obtained

$$F_{\text{test}} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2, \tag{7}$$

then
6: Accept the new iterate: $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.
7: **else**
8: Update the current iterate with the proximal mapping: $x_{k+1} = z_k$.

$$(g_{k+1}, F_{k+1}, z_{k+1}) = \mathsf{ApproxGradient}\,(x_{k+1}, \mathcal{M})\ .$$

9: **end if**
10: update $H_{k+1} = \mathsf{L\text{-}BFGS}(H_k, x_{k+1} - x_k, g_{k+1} - g_k)$.
11: **end for**

output last proximal mapping z_K (solution).

Algorithm 2 Generic procedure ApproxGradient

input Current point x in $\mathbb{R}^p$; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using an optimization method $\mathcal{M}$:

$$z \approx \operatorname*{arg\,min}_{v \in \mathbb{R}^d} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v - x\|^2 \right\}, \tag{8}$$

using the restart strategies and stopping criterions described later in this section.
2: Estimate the gradient $\nabla F(x)$ of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output approximate gradient estimate g, objective value $F_a \triangleq h(z)$, proximal mapping z.

(a) defining a pre-defined decreasing sequence $(\varepsilon_k)_{k\geq 0}$ and stopping criterion based on function values—that is, stop the optimization method $\mathcal{M}$ when the approximate solution of (8) satisfies $h(z) - h^* \leq \varepsilon_k$, where where $h^* = \min_{z\in\mathbb{R}^d} h(z)$, which may be checked by computing a duality gap. This is the less practical of the three strategies presented here.

(b) defining an adaptive sequence $(\varepsilon_k)_{k\geq 0}$, which depends on quantities that are available at iteration k.

(c) using a pre-defined budget $T_{\mathcal{M}}$ in terms of number of iterations of the method $\mathcal{M}$ at iteration k.

Section 4 will provide theoretical and practical insight about these parameter choices. Before that, we discuss the requirements on f and $\mathcal{M}$ and restart strategies to solve the sub-problems.

Requirements on f. When f is μ-strongly-convex, we show that the method achieves a linear convergence rate (which takes into account the complexity of solving the sub-problems with $\mathcal{M}$); to obtain this guarantee, the accuracy for solving the sub-problems needs to decrease at the same linear rate. In principle, all strategies (a)-(b)-(c) above may be used to stop $\mathcal{M}$, as discussed in Section 4.

When f is convex but not strongly convex (meaning $\mu = 0$), it is possible to achieve the classical $O(1/k)$ worst-case convergence rate on function values, by using a particular strategy (b) and by replacing the approximate descent condition (7) by a slightly stronger one $f(x_{\text{test}}) \leq f(z_k)$, see again details in Section 4.

Requirements on $\mathcal{M}$ and restarts. We have mentioned that QuickeNing applies to methods $\mathcal{M}$ with linear convergence rates for strongly-convex problems. More precisely, we consider methods $\mathcal{M}$ that are always able produce a sequence of iterates $(w_t)_{t\geq 0}$ for solving each sub-problem (8) such that

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^t (h(w_0) - h^*) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0, \tag{9}$$

where w_0 is the initial point given to $\mathcal{M}$. When (9) is satisfied, we call $\mathcal{M}$ a *primal* method. QuickeNing can also afford non-deterministic methods $\mathcal{M}$ that satisfy (9) in expectation; in such a case, our complexity results also hold in expectation. The condition typically holds for many primal gradient-based optimization techniques, such as block-coordinate descent algorithms [35, 40], or some incremental algorithms [10, 45]. The restart strategy derived from our convergence analysis in Section 4 is simply $w_0 = x_{\text{test}}$.

Then, we also consider the class of *dual* methods MISO/Finito [11, 28], which are analyzed in Appendix A, where the convergence rate of $\mathcal{M}$ is stated in terms of dual certificate instead of primal quantities $h(w_t) - h^*$. For such approaches, we consider at iteration k the restart strategy $w_0 = z_k + \frac{\kappa}{\kappa+\mu}(x_{\text{test}} - x_k)$, which was originally used in the accelerated stochastic dual coordinate ascent method [46] and in Catalyst [26]. Finally, before giving more precise theoretical statements, we discuss briefly two important features of the algorithm.

Handling composite objective functions. In machine learning or signal processing, convex composite objectives (1) with a non-smooth penalty ψ are typically formulated to encourage solutions with specific characteristics; in particular, the ℓ_1-norm is known to provide sparsity. Smoothing techniques [34] may allow us to solve the optimization problem up to some chosen accuracy, but they provide solutions that do not inherit the properties induced by the non-smoothness of the objective. To illustrate what we mean by this sentence, we may consider smoothing the ℓ_1-norm, leading to a solution vector with small coefficients, but not with exact zeroes. When the goal is to perform model selection—that is, understanding which variables are important to explain a phenomenon, exact sparsity is seen as an asset, and optimization techniques dedicated to composite problems such as FISTA [1] are often preferred.

Then, one might be concerned that our scheme operates on the smoothed objective F, leading to iterates $(x_k)_{k\geq 0}$ that may suffer from the above "non-sparse" issue, assuming that ψ is the ℓ_1-norm. Yet, our approach also provides iterates $(z_k)_{k\geq 0}$ that are computed using the original optimization method $\mathcal{M}$ we wish to accelerate. When $\mathcal{M}$ handles composite problems without smoothing, typically when $\mathcal{M}$ is a proximal block-coordinate, or incremental method, the iterates $(z_k)_{k\geq 0}$ may be sparse. For this reason, our theoretical analysis presented in Section 4 studies the convergence of the sequence $(f(z_k))_{k\geq 0}$ to the solution f^*.

On the absence of line-search scheme. Another key property of the QuickeNing algorithm is that it does not require using a line-search scheme to select a step-size, which is typically necessary to ensure the monotonic descent (and convergence) of classical BFGS and L-BFGS algorithms [37]. In the context of the Moreau-Yosida regularization, any line-search would be prohibitive, since it would require to evaluate the function F multiple times, hence solving the sub-problems (8) as many times at every iteration.[1] Here, we choose a simple strategy that selects $x_{k+1} = z_k$ when the sufficient descent condition (7) is not satisfied. z_k is indeed a good candidate since it is obtained by performing one step of the inexact proximal point algorithm, which we have described above, and whose convergence properties are well understood. In practice, we have observed that the sufficient decrease condition is almost all the time satisfied, given the choice of parameters provided in the experimental section.

4 Convergence and complexity analysis

In this section, we study the convergence of the QuickeNing algorithm—that is, the rate of convergence to zero of the quantities $(f(z_k)-f^*)_{k\geq 0}$ and $(F(x_k)-F^*)_{k\geq 0}$, and also its computational complexity, which takes into account the cost of solving the sub-problems (8). We consider several cases, leading to different variants of the parameter choices, restart and approximation strategies. We apply QuickeNing to the MISO/Finito algorithms and establish the resulting global complexity bounds. We start by stating the main properties of the gradient approximation in Section 4.1. Then, we analyze the convergence of the outer loop algorithm in Section 4.2, and then Section 4.3 is devoted to the global complexity analysis.

4.1 Properties of the gradient approximation

The next lemma is classical and provides approximation guarantees about the quantities returned by the ApproxGradient procedure (Algorithm 2); see [3, 18]. We recall here the proof for completeness.

Lemma 1 (**Approximation quality of the gradient approximation**). *Consider a vector x in $\mathbb{R}^d$, a positive scalar ε and compute*

$$z \approx \operatorname*{arg\,min}_{v\in\mathbb{R}^d} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v-x\|^2 \right\},$$

such that $h(z) - h^* \leq \varepsilon$, where $h^* = \min_{v\in\mathbb{R}^d} h(v)$. As in Algorithm 2, define $g = \kappa(x-z)$ and $F_a = h(z)$. Then, the following inequalities hold

$$F(x) \leq F_{\mathrm{a}} \leq F(x) + \varepsilon, \tag{10}$$

$$\|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}}, \tag{11}$$

$$\|g - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}. \tag{12}$$

Proof. (10) is straightforward by definition of $h(z)$. Since f is convex, the function h is κ-strongly convex, and (11) follows from

$$\frac{\kappa}{2}\|z-p(x)\|^2 \leq h(z) - h(p(x)) = h(z) - h^* \leq \varepsilon,$$

where we recall that $p(x)$ minimizes h. Finally, we obtain (12) from

$$g - \nabla F(x) = \kappa(x-z) - \kappa(x-p(x)) = \kappa(p(x)-z),$$

by using the definitions of g and the property (5). □

[1] Note that in the context of BFGS applied to F, a heuristic line-search that requires evaluating f instead of F is also proposed in [9], but this heuristic does not guarantee the algorithm convergence and is still computationally demanding when f is a large sum of functions as in (2).

This lemma allows us to quantify the quality of the gradient and function value approximations in terms of ε, which is a key to control the error accumulation of our algorithm. Before moving to a first convergence result, a few additional technical inequalities are needed.

Lemma 2 (**Simple inequalities regarding the gradient approximation).** *Consider the same quantities introduced in Lemma 1. Then,*

$$f(z) = F_{\mathrm{a}} - \frac{1}{2\kappa}\|g\|^2, \tag{13}$$

$$\frac{1}{2}\|\nabla F(x)\|^2 - 2\kappa\varepsilon \leq \|g\|^2 \leq 2(\|\nabla F(x)\|^2 + 2\kappa\varepsilon). \tag{14}$$

Proof. Eq. (13) is obtained by using the definitions of g and noticing that $F_{\mathrm{a}} = h(z)$. Eq. (14) follows from

$$\begin{aligned} \|\nabla F(x)\|^2 &\leq 2(\|\nabla F(x) - g\|^2 + \|g\|^2) \\ &\leq 2(2\kappa\varepsilon + \|g\|^2) \qquad \text{(from (12)).} \end{aligned}$$

Interchanging $\nabla F(x)$ and g gives the right-hand side inequality. □

4.2 Convergence analysis of the outer loop

We are now in shape to establish the convergence of the QuickeNing meta-algorithm, without considering yet the cost of solving the sub-problems (8). We first remark that the following relation always holds

$$(g_k, F_k, z_k) = \mathsf{ApproxGradient}(x_k, \mathcal{M}) \;, \tag{15}$$

which allows us to apply Lemma 2 for all $k \geq 0$. In the following analysis, we will always call ε_k the precision (in the sense of Lemma 1) of solving the sub-problem (8) related to the call (15).

Lemma 3 (**Approximate descent property).** *Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 1 where z_k is obtained by solving a sub-problem (8) with accuracy ε_k. Then,*

$$F(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{1}{4\kappa}\|\nabla F(x_k)\|^2 + 2\varepsilon_k. \tag{16}$$

Proof. By (13), we have

$$f(z_k) = F_k - \frac{1}{2\kappa}\|g_k\|^2.$$

To show the first inequality, we consider two cases:

- When the condition (7) is violated, then we set $x_{k+1} = z_k$. As a result,

$$F(x_{k+1}) = \min_{z\in\mathbb{R}^k} \left\{ f(z) + \frac{\kappa}{2}\|z - x_{k+1}\|^2 \right\} \leq f(x_{k+1}) = f(z_k).$$

- Otherwise, we have $F(x_{k+1}) \leq F_{k+1} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2 = f(z_k)$.

Thus, we have $F(x_{k+1}) \leq f(z_k)$ in both cases and we may now prove the second inequality:

$$\begin{aligned} f(z_k) &= F_k - \frac{1}{2\kappa}\|g_k\|^2 \\ &\leq F(x_k) + \varepsilon_k - \left(\frac{1}{4\kappa}\|\nabla F(x_k)\|^2 - \varepsilon_k\right) \quad \text{(from (10) and (14))} \\ &= F(x_k) - \frac{1}{4\kappa}\|\nabla F(x_k)\|^2 + 2\varepsilon_k. \end{aligned}$$

This concludes the proof. □

This lemma gives us a first intuition about the natural choice of the accuracy ε_k to use, which should be of the same order as $\|\nabla F(x_k)\|^2$. Unfortunately, the quantity $\|\nabla F(x_k)\|^2$ is not known since computing its value requires solving exactly the subproblem (8). We propose in the next section several practical strategies to set the sequences $(\varepsilon_k)_{k\geq 0}$. Before that, we now use the approximate descent property of the previous lemma to control the accumulation of errors across iterations.

4.2.1 Strongly-convex objectives.

We start by analyzing the convergence of our algorithm for strongly-convex objectives.

Proposition 2 (Convergence of Algorithm 1 with theoretical sequence $(\varepsilon_k)_{k\geq 0}$.). *Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{4(\mu+\kappa)}$. Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 1 where z_k is obtained by solving a sub-problem (8) with accuracy ε_k where*

$$\varepsilon_k \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2. \tag{17}$$

Then,

$$F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq (1-\rho)^{k+1}\,(f(x_0) - f^*).$$

Proof. We may apply Lemma 3, and from (16), we obtain

$$\begin{aligned} F(x_{k+1}) \leq f(z_k) &\leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 \\ &\leq F(x_k) - F^* - \frac{\mu_F}{4\kappa}(F(x_k) - F^*) \\ &\leq (1-\rho)\,(F(x_k) - F^*) \\ &\leq (1-\rho)^{k+1}\,(f(x_0) - f^*), \end{aligned}$$

where the second inequality uses a classical relation [33, Theorem 2.1.10] for strongly-convex functions, and the third one simply uses the definition of ρ given in Proposition 4. □

The previous proposition is interesting from a theoretical point of view, but, as discussed previously, the quantity $\|\nabla F(x_k)\|^2$ is unknown in advance. Therefore, the above criterion does not seem practical at first sight. The next proposition presents one way to use them concretely.

Proposition 3 (On using Proposition 2 in practice). *The following condition implies the inequality (17) in Proposition 2:*

$$\varepsilon_k \leq \frac{1}{36\kappa}\|g_k\|^2.$$

Proof. We may use (14) from Lemma 2, which gives

$$\|g_k\|^2 \leq 2\left(\|\nabla F(x_k)\|^2 + \frac{1}{18}\|g_k\|^2\right),$$

which implies

$$\frac{4}{9}\|g_k\|^2 \leq \|\nabla F(x_k)\|^2 \quad \text{and thus} \quad \varepsilon_k \leq \frac{1}{36\kappa}\|g_k\|^2 \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2.$$

□

While we do have access to $\|\nabla F(x_k)\|^2$, we do have access to the estimate gradient $g_k = \kappa(x_k - z_k)$. This immediately suggests the following stopping criterion when applying the method $\mathcal{M}$ to (8)

$$h(w_t) - h^* \leq \frac{\kappa}{36}\|w_t - x\|^2 \tag{18}$$

where $(w_t)_{t\geq 0}$ is the sequence produced by $\mathcal{M}$ to solve the sub-problem (8). Again, such a condition can often be checked by computing duality gap. This choice ensures the linear convergence of the sequences $(F(x_k))_{k\geq 0}$ and $(f(z_k))_{k\geq 0}$ to f^*. We shall also see in Section 4.3 that the condition may also be enforced by simply using a constant number of iterations of the method $\mathcal{M}$ when the right restart strategy is used. Another possibility is to use a pre-defined sequence $(\varepsilon_k)_{k\geq 0}$, similar to Catalyst [26].

Proposition 4 (Convergence of Algorithm 1 with pre-defined sequence). *Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{4(\mu+\kappa)}$. Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 1 where z_k is obtained by solving a sub-problem (8) with accuracy ε_k for all k. Then,*

$$F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq (1-2\rho)^{k+1}\left(f(x_0) - f^*\right) + 2\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i.$$

Proof. First, we recall from Proposition 1 that F is μ_F-strongly convex with parameter $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$ and that the optimal value of the objectives f and F coincide—in other words, and $F^* = f^*$. We may then apply Lemma 3 and

$$\begin{aligned}
F(x_{k+1}) - F^* \leq f(z_k) - f^* &\leq F(x_k) - F^* - \frac{1}{4\kappa}\|\nabla F(x_k)\|^2 + 2\varepsilon_k,\\
&\leq F(x_k) - F^* - \frac{\mu_F}{2\kappa}(F(x_k) - F^*) + 2\varepsilon_k,\\
&\leq (1-2\rho)\left(F(x_k) - F^*\right) + 2\varepsilon_k,\\
&\leq (1-2\rho)^{k+1}\left(F(x_0) - F^*\right) + 2\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i,\\
&\leq (1-2\rho)^{k+1}\left(f(x_0) - f^*\right) + 2\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i,
\end{aligned}$$

where the last inequality simply comes from the upper-bound $F \leq f$. □

Corrolary 1 (Recommended pre-defined sequence $(\varepsilon_k)_{k\geq 0}$). *If the sequence $(\varepsilon_k)_{k\geq 0}$ in Proposition 4 satisfies*

$$\varepsilon_k = \frac{1}{2}C(1-\rho)^{k+1} \quad \text{with} \quad C \geq f(x_0) - f^*.$$

then

$$F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq \frac{C}{\rho}\left(1-\rho\right)^{k+2}\left(f(x_0) - f^*\right). \tag{19}$$

Proof. From Proposition 4, we have

$$\begin{aligned}
F(x_{k+1}) - F^* \leq f(z_k) - f^* &\leq (1-2\rho)^{k+1}\left(f(x_0) - f^*\right) + 2\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i,\\
&\leq (1-2\rho)^{k+1}\,C + \sum_{i=0}^{k}(1-2\rho)^{k-i}\,(1-\rho)^{i+1}C,\\
&= C\,\frac{(1-\rho)^{k+2} - (1-2\rho)^{k+2}}{(1-\rho) - (1-2\rho)},\\
&\leq \frac{C}{\rho}(1-\rho)^{k+2}.
\end{aligned}$$

□

We remark that, of course, the quantity $f(x_0) - f^*$ is unknown in practice, but the proposition only requires an upper-bound, e.g., a duality gap, or simply $f(x_0)$ if the function is non-negative. Next, we present a similar complexity analysis for convex, but not strongly-convex objectives.

4.2.2 Convex but not strongly-convex objectives.

The goal of this section is to show that a minor modification of the algorithm yields the classical $O(1/k)$ convergence rate of the sequences $(F(x_k))_{k\geq 0}$ and $(f(z_k))_{k\geq 0}$, which can be shown to be slightly stronger than the previous one, when the accuracy ε_k is small enough. Specifically, we replace the descent condition (7) by $f(x_{\text{test}}) \leq f(z_k)$. We may now use the new descent condition to derive a convergence rate for non-strongly convex function.

Proposition 5 (Convergence of Algorithm 1 for convex, but not strongly-convex objectives). *Let f be a convex function, which attains its minimum at x^*. Assume that the level sets of f are bounded. More precisely, assume that there exists $R > 0$ such that*

$$\forall y \in \mathbb{R}^p \quad s.t. \quad f(y) \leq f(x_0) \quad then \quad \|y - x^*\| \leq R.$$

Assume that the sub-problems (8) are solved up to accuracy $\varepsilon_k \leq \frac{1}{2\kappa}\|g_k\|^2$ for any k, and replace the descent condition (7) in Algorithm 1 by $f(x_{test}) \leq f(z_k)$; then, the sequence $(x_k)_{k\geq 0}$ satisfies

$$f(x_k) - f^* \leq \frac{2\kappa R^2}{k+3}.$$

Proof. First, we remark that with the new condition, we always have

$$f(x_{k+1}) \leq f(z_k),$$

since $x_{k+1} = z_k$ when the descent condition $f(x_{\text{test}}) \leq f(z_k)$ is not satisfied. Moreover, according to Lemma 1,

$$f(z_k) + \frac{\kappa}{2}\|z_k - x_k\|^2 = F_k \leq F(x_k) + \varepsilon_k,$$

and then, by combining the two previous inequalities

$$f(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{\kappa}{2}\|z_k - x_k\|^2 + \varepsilon_k = F(x_k) - \frac{1}{2\kappa}\|g_k\|^2 + \varepsilon_k \leq F(x_k).$$

This relation allows us to use a similar proof technique introduced in [36] in the context of the proximal gradient descent method

$$\begin{aligned} f(x_{k+1}) \leq F(x_k) &= \min_{z\in\mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x_k\|^2 \right\} \\ &\leq \min_{\alpha\in[0,1]} \left\{ f(\alpha x^* + (1-\alpha)x_k) + \frac{\kappa\alpha^2}{2}\|x^* - x_k\|^2 \right\} \\ &\leq \min_{\alpha\in[0,1]} \left\{ \alpha f(x^*) + (1-\alpha)f(x_k) + \frac{\kappa\alpha^2}{2}\|x^* - x_k\|^2 \right\}. \end{aligned}$$

Since $f(x_k)$ is necessarily decreasing (take $\alpha = 0$ in the previous relation), we may use the bounded level set assumption, which leads to

$$f(x_{k+1}) - f^* \leq \min_{\alpha\in[0,1]} \left\{ (1-\alpha)(f(x_k) - f^*) + \frac{\kappa\alpha^2 R^2}{2} \right\}.$$

1. If $f(x_k) - f^* \geq \kappa R^2$, then $\alpha^* = 1$ and $f(x_{k+1}) - f^* \leq \frac{\kappa R^2}{2}$.
2. Otherwise, $\alpha^* = \frac{f(x_k) - f^*}{\kappa R^2}$, which gives

$$r_{k+1} \leq r_k(1 - \frac{r_k}{2\kappa R^2}) \leq \frac{\kappa R^2}{2},$$

where $r_k \triangleq f(x_k) - f^*$. Thus

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_k(1 - \frac{r_k}{2\kappa R^2})} \geq \frac{1}{r_k}\left(1 + \frac{r_k}{2\kappa R^2}\right) = \frac{1}{r_k} + \frac{1}{2\kappa R^2}.$$

In both case, $r_1 \leq \frac{\kappa R^2}{2}$ and there

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_1} + \frac{k}{2\kappa R^2} \geq \frac{k+4}{2\kappa R^2},$$

which is sufficient to conclude. □

4.3 Complexity analysis of the inner loop

In this section, we compute the complexity of solving the sub-problems (8) using the method $\mathcal{M}$, which allows us to obtain the global complexity of QuickeNing, when using the various approximation strategies described in the previous section. We analyze here primal approaches $\mathcal{M}$, which achieve the convergence guarantees (9), and present in Appendix A a similar analysis for the approaches MISO/Finito whose convergence rate does not satisfy (9). In both cases, our main result is that all sub-problems can be solved with an appropriate accuracy in a constant number $T_{\mathcal{M}}$ of iterations. One of the key is to choose the right restart point, as detailed next. The first lemma characterizes the quality of the initialization $w_0 = x$ for solving the sub-problem (8) when the objective function is smooth.

Lemma 4 (Restart for primal methods - smooth case). *If f differentiable with L-Lipschitz continuous gradients, we can initialize the method $\mathcal{M}$ with $w_0 = x$ for solving (8). Then, we have the guarantee that*

$$h(w_0) - h^* \leq \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2. \tag{20}$$

Proof. Denote by w^* the minimizer of h. Then, we have the optimality condition $\nabla f(w^*) + \kappa(w^* - x) = 0$. As a result,

$$\begin{aligned}
h(w_0) - h^* &= f(x) - \left(f(w^*) + \frac{\kappa}{2}\left\|w^* - x\right\|^2\right) \\
&\leq f(w^*) + \langle \nabla f(w^*), x - w^* \rangle + \frac{L}{2}\|x - w^*\|^2 - \left(f(w^*) + \frac{\kappa}{2}\|w^* - x\|^2\right) \\
&= \frac{L+\kappa}{2}\|w^* - x\|^2 \\
&= \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2.
\end{aligned}$$

□

The inequality in the proof of Lemma 4 relies on the smoothness of f, which does not hold for composite problems. The next lemma addresses this issue.

Lemma 5 (Restart for primal methods - composite case). *Consider the composite optimization problem (1). We may initialize the method $\mathcal{M}$ for solving (8) with*

$$w_0 = \operatorname*{arg\,min}_{w \in \mathbb{R}^d} \left\{ f_0(x) + \langle \nabla f_0(x), w - x \rangle + \frac{L+\kappa}{2}\|w - x\|^2 + \psi(w) \right\}. \tag{21}$$

Then,

$$h(w_0) - h^* \leq \frac{L}{2\kappa^2}\|\nabla F(x)\|^2.$$

Proof. We use the inequality corresponding to Lemma 2.3 in [1]: for any w,

$$h(w) - h(w_0) \geq \frac{L'}{2}\|w_0 - x\|^2 + L'\langle w_0 - x, x - w \rangle, \tag{22}$$

with $L' = L + \kappa$. Then, we apply this inequality to $w = w^*$, and

$$h(w_0) - h^* \geq -\frac{L'}{2}\|w_0 - x\|^2 - L'\langle w_0 - x, x - w^* \rangle \leq \frac{L'}{2}\|x - w^*\|^2 = \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2.$$

□

The lemmas presented above are important, since they allow us to derive the global complexity of QuickeNing in the scenarios described in Propositions 2, 3, and 5. Specifically, it shows us that the conditions in both cases will be satisfied in a constant number of iterations of the method $\mathcal{M}$.

Proposition 6 (**Inner-loop complexity for Algorithm 1).** *Assume that f is differentiable with L-Lipschitz continuous gradients. Consider Algorithm 1 with the restart strategy described in Lemma 4 or in Lemma 5. Assume that the primal method $\mathcal{M}$ applied in the inner loop produces a sequence $(w_t)_{t\geq 0}$ for each sub-problem (8) such that*

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h(w_0)-h^*) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0. \tag{23}$$

Then, the sub-problems are solved with enough accuracy to apply Propositions 2, 3, and 5 in at most $T_{\mathcal{M}}$ iterations with

$$T_{\mathcal{M}} = \frac{1}{\tau_{\mathcal{M}}} \log\left(19 C_{\mathcal{M}} \frac{L+\kappa}{\kappa}\right).$$

Proof. The proposition is a direct consequence of the restart strategy in previous lemmas. Consider at iteration k, we want to approximate the proximal mapping according to x_k. With the given $T_{\mathcal{M}}$ (which we abbreviate by T), we have

$$\begin{aligned}
h(w_T) - h^* &\leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^T(h(w_0)-h^*) \\
&\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T}(h(w_0)-h^*) \\
&\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T} \frac{L+\kappa}{2\kappa^2}\|\nabla F(x_k)\|^2 \quad \text{(By Lemma 4 and Lemma 5)} \\
&= \frac{1}{38\kappa}\|\nabla F(x_k)\|^2.
\end{aligned}$$

It reamins to show that this accuracy is in fact small enough to apply Propositions 2, 3, and 5. From Lemma 2, we know that $\|g_k\|$ and $\|\nabla F(x_k)\|$ are related through:

$$\|\nabla F(x_k)\|^2 \leq \|g_k\|^2 + 2\kappa\varepsilon_k \leq \|g_k\|^2 + \frac{1}{19}\|\nabla F(x_k)\|^2.$$

Thus

$$h(w_T) - h^* \leq \frac{1}{38\kappa}\|\nabla F(x_k)\|^2 \leq \frac{1}{36\kappa}\|g_k\|^2.$$

This is exactly the accuracy required in Proposition 3. From which we easily derived the accuracy in Propositions 2 and 5.

□

4.4 Global complexity of QuickeNing

Finally, we can use the previous result to upper-bound the complexity of the QuickeNing algorithm in terms of iterations of the method $\mathcal{M}$ for minimizing f up to ε.

Proposition 7 (**Worst-case global complexity for Algorithm 1).** *Under the setting of Proposition 6, the number of iterations of the method $\mathcal{M}$ (or expected number if $\mathcal{M}$ is non-deterministic) to guarantee the optimality condition $f(z_k) - f^* \leq \varepsilon$ is*

- *for μ-strongly-convex problems:*

$$2T_{\mathcal{M}} \times \frac{\mu+\kappa}{\mu}\log\left(\frac{f(x_0)-f^*}{\varepsilon}\right) = 2\frac{\mu+\kappa}{\tau_{\mathcal{M}}\mu}\log\left(\frac{f(x_0)-f^*}{\varepsilon}\right)\log\left(19C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right).$$

- *for convex but not strongly convex problems:*

$$2T_{\mathcal{M}} \times \frac{2\kappa R^2}{\varepsilon} = \frac{4\kappa R^2}{\tau_{\mathcal{M}}\varepsilon}\log\left(19C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right).$$

Proof. Since in the worst case, there are two proximal problems to be solved at each outer-loop iteration, the total number of call of method $\mathcal{M}$ is at most $2T_{\mathcal{M}}$ times the number of outer-loop iterations. The result follows immediately from Propositions 2 and 5. □

We give below the worst-case global complexity of QuickeNing when applied to two optimization methods $\mathcal{M}$ of interest. As we shall see, in terms of worst-case complexity bounds, QuickeNing does not lead to an improved convergence rate. It is worthwhile to underline, though, that this result is not surprising since it is often the case for L-BFGS-type methods, for which an important gap remains between theory and practice. When comparing the vanilla L-BFGS algorithm with the gradient descent method, one outperforms the other in many practical cases, but never in theory.

Proposition 7 and its application to the two examples below show that, in terms of worse-case complexity, the QuickeNing scheme leaves the convergence rate unchanged. Linearly-converging methods remain linear-converging when used within the QuickeNing scheme.

Example 1. *Consider gradient descent with fixed constant step-size $1/L$ as the optimization method $\mathcal{M}$. Gradient descent (GD) minimizes f to ε accuracy in $L/\mu \log(1/\varepsilon)$ iterations. Quickening-GD minimizes f to ε accuracy in $2(L+\kappa)/\mu \log(1/\varepsilon)$ iterations (when ignoring logarithmic terms).*

Example 2. *Consider the stochastic variance-reduced gradient (SVRG) as the optimization method $\mathcal{M}$. SVRG minimizes f to ε accuracy in*

$$O\left(\max\left\{n, \frac{L}{\mu}\right\} \log\left(\frac{1}{\varepsilon}\right)\right)$$

iterations in expectation. QuickeNing-SVRG with minimizes f to ε accuracy in

$$O\left(\max\left\{\frac{\mu+\kappa}{\mu}n, \frac{L+\kappa}{\mu}\right\} \log\left(\frac{1}{\varepsilon}\right)\right)$$

iterations in expectation.

Choice of κ. The worst-case complexity theory also does not seem to offer any guidance regarding the choice of κ for the QuickeNing acceleration to be most effective. We shall experiment with a heuristic, consisting in choosing the κ that would lead to the most acceleration for the related Catalyst acceleration scheme [26]. We present empirical evidence in support of this heuristic in Section 5.

5 Experiments and practical details

In this section, we present experimental results obtained by applying QuickeNing on SVRG (Section 5.3) and on the proximal gradient descent algorithm ISTA (Section 5.4), and we compare the resulting methods to other acceleration techniques and to the L-BFGS algorithm. We also study the behavior of QuickeNing when changing the smoothing parameter κ and the limited-memory parameter l (Section 5.5). Before that, we first present in Section 5.1 the formulations and datasets that we use, and also discuss in Section 5.2 various experimental and practical choices.

5.1 Formulations and datasets

We consider three common optimization problems in machine learning and signal processing. The problems may admit a particular structure (large finite sum, composite, strong convexity). For each formulation, we also consider a training set $(b_i, a_i)_{i=1}^n$ of n data points, where the b_i's are scalars in $\{-1, +1\}$ and the a_i are feature vectors in $\mathbb{R}^d$. Then, the goal is to fit a linear model x in $\mathbb{R}^p$ such that the scalar b_i can be well predicted by the inner-product $\approx a_i^\top x$, or by its sign. Specifically, the three formulations we consider are listed below.

- ℓ_2**-regularized Logistic Regression**:

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} \log\left(1+\exp(-b_i\, a_i^T x)\right) + \frac{\mu}{2}\|x\|^2,$$

  which leads to a μ-strongly convex smooth optimization problem.

- ℓ_1**-regularized Linear Regression (LASSO)**:

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2n}\sum_{i=1}^{n} (b_i - a_i^T x)^2 + \lambda\|x\|_1,$$

  which is non smooth and convex but not strongly convex.

- $\ell_1 - \ell_2^2$**-regularized Linear Regression (Elastic-Net)**:

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2n}\sum_{i=1}^{n} (b_i - a_i^T x)^2 + \lambda\|x\|_1 + \frac{\mu}{2}\|x\|^2,$$

  which is based on the Elastic-Net regularization [49] leading to strongly-convex optimization problem.

Since we normalize the feature vectors a_i, a natural upper-bound on the Lipschitz constant L of the unregularized objective can be easily obtained with $L_{\text{logistic}} = 1/4$, $L_{\text{elastic-net}} = 1$ and $L_{\text{lasso}} = 1$. In the experiments, we consider relatively ill-conditioned problems with the regularization parameter $\mu = 1/(100n)$. Experiments with other values of μ are provided in the supplemental material. The ℓ_1-regularization parameter is set to $\lambda = 1/n$ for the Elastic-Net formulation; for the Lasso problem, we consider a logarithmic grid $10^i/n$, with $i = -3, -2, \ldots, 3$, and we select the parameter λ that provides a sparse optimal solution closest to 10% non-zero coefficients.

Datasets. We consider four standard machine learning datasets with different characteristics in terms of size and dimension, which are described below:

name	covtype	alpha	real-sim	rcv1
n	581 012	250 000	72 309	781 265
d	54	500	20 958	47 152

5.2 Choice of hyper-parameters and variants

Before presenting the numerical results, we discuss the choice of default parameters used in the experiments as well as different variants.

Choice of method $\mathcal{M}$. We consider the acceleration of the proximal SVRG algorithm [47] since it is able to adapt to the problem structure we consider: large sum of functions and composite regularization. We also consider accelerating the ISTA algorithm [1], which allow us to perform a comparison with the natural baselines FISTA and L-BFGS for smooth problems.

Stopping criteria for the inner loop. The default stopping criteria is to solve the subproblem until accuracy ε_k such that $\varepsilon_k \leq \frac{1}{36}\|g_k\|^2$, which reduces to the simple criterion (18). Although we have shown that such accuracy is attainable in constant $T_{\mathcal{M}}$ iteration, the theoretical value of $T_{\mathcal{M}}$ could be very large depending on the conditioning of the problem. A natural heuristic is to always perform exactly one pass over the data in the inner loop without checking any stopping criteria as proposed in [26].

When applying QuickeNing to SVRG and ISTA, we call the one-pass variant **QuickeNing-SVRG1** and **QuickeNing-ISTA1**, respectively, and the one using stopping criterion (18) **QuickeNing-SVRG2** and **QuickeNing-ISTA2**. Finally, another variant that could be considered is to give a pre-defined sequence of $(\varepsilon_k)_{k\geq 0}$ and stop when the current accuracy is smaller than ε_k, which is checked by using Fenchel duality; typically, this variant is less practical and is thus omitted from the experiments.

Choice of regularization parameter κ. As the global complexity is a function of κ, the most intuitive choice is to choose κ to minimize the complexity. However, such analysis leads to $\kappa = 0$, which is not quite helpful since this suggests directly performing $\mathcal{M}$ on the original problem. The reason is that the convergence of L-BFGS is hard to characterize and theoretical rates of convergence can be pessimistic. Noting that for smooth functions, L-BFGS often outperforms Nesterov's accelerated gradient method, it is reasonable to expect QuickeNing achieves a similar complexity bound as Catalyst. Therefore, we choose κ identical as in Catalyst algorithm, which is L for gradient descent and $L/2n$ for SVRG. Later in this section, a comparison of different scaling of κ is performed in order to demonstrate the relevance of this strategy.

Choice of limited memory parameter l. The default setting is $l = 100$. We show a comparison with smaller values to study the influence of this parameter.

Sufficient decrease condition. We use the sufficient decrease condition suggested by the theory:

- for strongly convex objectives, we use $F_{\text{test}} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2$;
- for convex, but non-strongly convex objectives, we use $f(x_{k+1}) \leq f(z_k)$. Nevertheless, we will also evaluate experimentally the strategy for strongly-convex objectives in this context.

Comparison metric. For all experiments, we use the number of gradient evaluations as a natural metric, assuming this is the computational bottleneck of all methods considered. This is indeed the case here since the L-BFGS step cost $O(dl)$ floating-point operations [38], whereas evaluating the gradient of the full objective costs $O(nd)$, with $d \leq n$ and $l \ll d$. A similar choice was made in other works [45].

5.3 QuickeNing-SVRG for minimizing large sums of functions

We now apply QuickeNing to SVRG and compare different variants.

- **SVRG**: the Prox-SVRG algorithm of [47] with default parameters $m = 1$ and $\eta = 1/L$, where L is the upper-bound on Lipschitz constant of the gradient, as described in the Section 5.2.
- **Catalyst-SVRG**: The catalyst meta-algorithm of [26] applied to Prox-SVRG;
- **L-BFGS** (for smooth objectives): Since implementing effectively L-BFGS with a line-search algorithm is a bit involved, we use the implementation of Mark Schmidt[2], which has been widely used in other comparisons [45]. The limited memory parameter l is also set to 100. We use it for the logistic regression experiment since L-BFGS is defined only for smooth objectives.
- **QuickeNing-SVRG1** as detailed above. For the Lasso problem, we also evaluate this heuristic with the sufficient decrease condition used for strongly convex problems. We call this heuristic variant **QuickeNing-SVRG1***.
- **QuickeNing-SVRG2**, as detailed above.

The result of the comparison is presented in Figure 1 and leads to the conclusions below, showing that **QuickeNing-SVRG1** is a safe heuristic, which never decreases the speed of the method **SVRG**:

[2]available here `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`

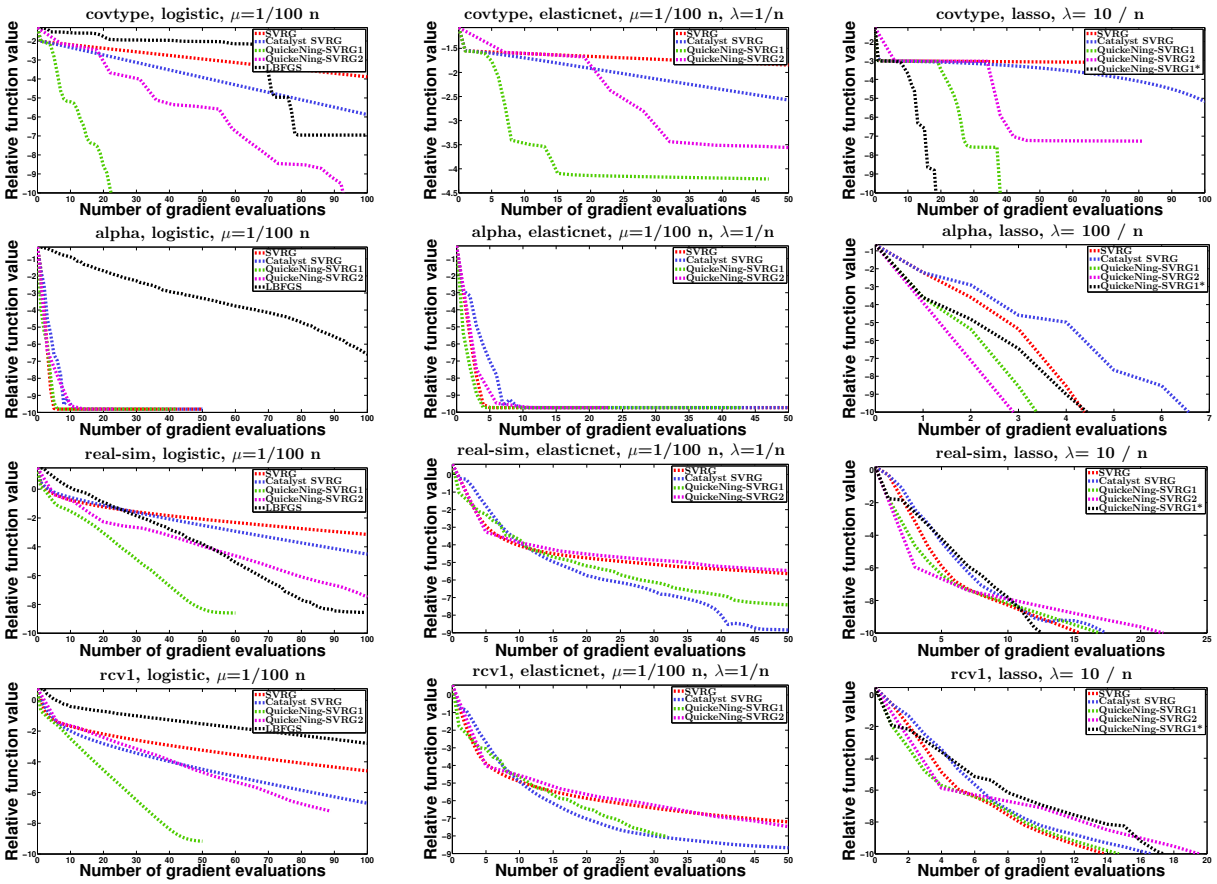

Figure 1: Experimental study of the performance of QuickeNing-SVRG for minimizing large sums of functions. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

- **L-BFGS** is less competitive than other approaches that exploit the sum structure of the objective, except on the dataset real-sim; the difference in performance with the SVRG-based approaches can be very important (see dataset alpha).
- **QuickeNing-SVRG1** is significantly faster than or on par with **SVRG** and **QuickeNing-SVRG2**.
- **QuickeNing-SVRG2** is significantly faster than, or on par with, or only slightly slower than **SVRG**.
- **QuickeNing-SVRG1** is significantly faster, or on par with **Catalyst-SVRG** except in two cases where **Catalyst-SVRG** performs slightly better.
- there is no clear conclusion regarding the performance of **QuickeNing-SVRG1*** vs **QuickeNing-SVRG1** for the Lasso problem.

5.4 QuickeNing-ISTA and comparison with L-BFGS

The previous experiments have included a comparison between L-BFGS and approaches that are able to exploit the sum structure of the objective. It is then interesting to study the behavior of QuickeNing when applied to a basic proximal gradient descent algorithm such as ISTA. Specifically, we now consider

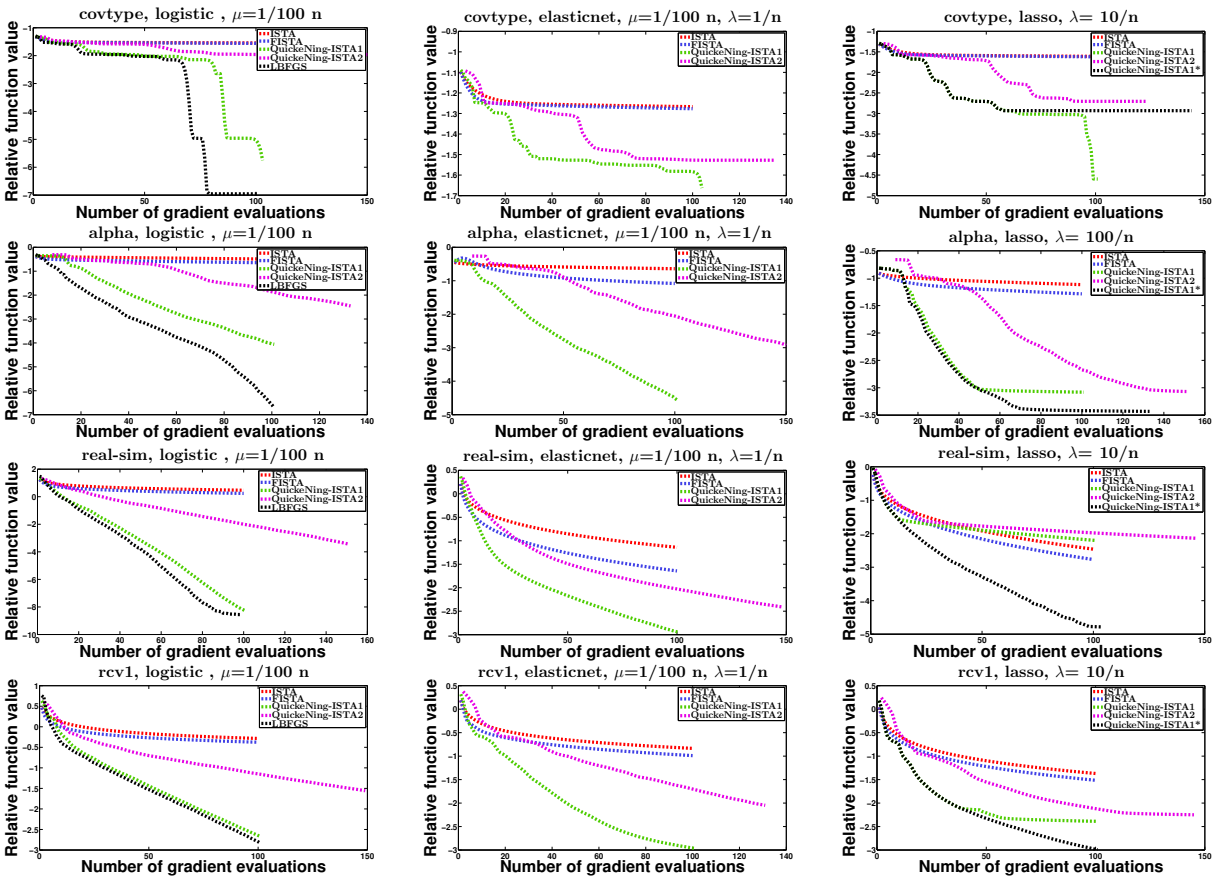

Figure 2: Experimental study of the performance of QuickeNing-ISTA. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

- **ISTA**: the classical proximal gradient descent algorithm ISTA [1] with back-tracking line-search to automatically adjust the Lipschitz constant of the gradient objective;
- **FISTA**: the accelerated variant of ISTA from [1].
- **L-BFGS**, **QuickeNing-ISTA1**, **QuickeNing-ISTA2**, and **QuickeNing-ISTA1***, as in the previous section when replacing SVRG by ISTA.

The results are reported in Figure 2 and lead to the following conclusions

- **L-BFGS** is slightly better on average than **QuickeNing-ISTA1** for smooth problems, which is not surprising since we use a state-of-the-art implementation with a well-calibrated line search.
- **QuickeNing-ISTA1** is always significantly faster than, or on par with **QuickeNing-ISTA2**.
- The **QuickeNing-ISTA** approaches are significantly faster than **FISTA** in 15 cases out of 16.
- for Lasso problems, **QuickeNing-ISTA1*** is sometimes faster than **QuickeNing-ISTA1**.

5.5 Experimental study of hyper-parameters l and κ

In this section, we study the influence of the limited memory parameter l and of the regularization parameter κ in QuickeNing. More precisely, we start with the parameter l and try the method **QuickeNing-SVRG1** with the values $l = 1, 2, 5, 10, 20, 100$. Note that all previous experiments were conducted with $l = 100$, which is the most expensive in terms of memory and computational cost for the L-BFGS step. The results are presented in Figure 3. Interestingly, the experiment suggests that having a large value for l is not necessarily the best choice, especially for composite problems where the solution is sparse, where $l = 5$ seems to be a reasonable choice in practice.

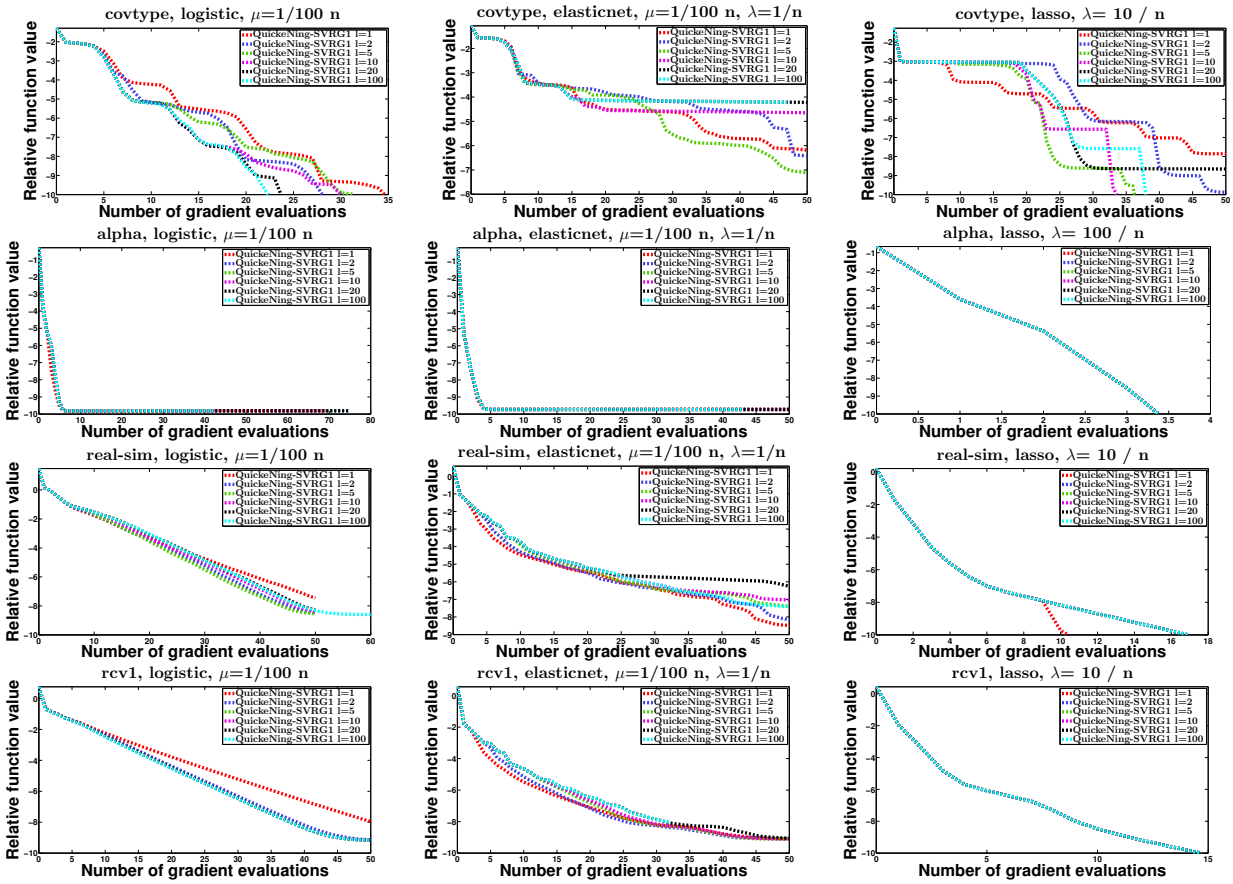

Figure 3: Experimental study of influence of the limited-memory parameter l for QuickeNing-SVRG1. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

The next experiment consists of studying the robustness of QuickeNing to the smoothing parameter κ. We present in Figure 4 an experiments by trying the values $\kappa = 10^i \kappa_0$, for $i = -3, -2, \ldots, 2, 3$, where κ_0 is the default parameter that we used in the previous experiments. The conclusion is clear: QuickeNing clearly slows down when using a larger smoothing parameter than κ_0, but it is very robust to small values of κ (and in fact it even performs better for smaller values than κ_0 in one experiment).

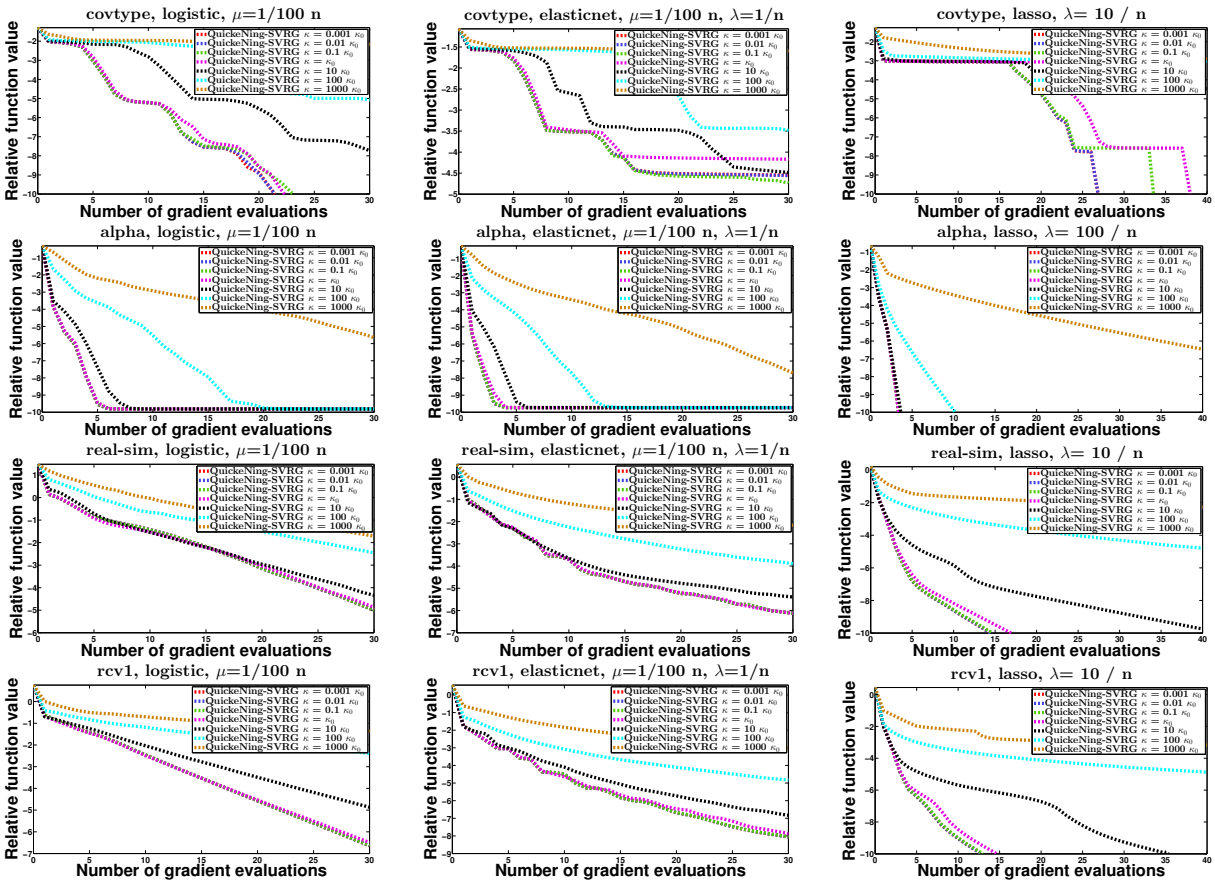

Figure 4: Experimental study of influence of the smoothing parameter κ for QuickeNing-SVRG1. κ_0 denotes the default choice used in the previous experiments. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

6 Discussions and concluding remarks

A few questions naturally arise regarding the QuickeNing scheme: one may wonder whether or not our convergence rates may be improved, or if the Moreau-Yosida regularization could be replaced by another smoothing technique. In this section, we discuss these two points and present concluding remarks.

6.1 Discussion of convergence rates

In this paper, we have established the linear convergence of QuickeNing for strongly convex objectives when sub-problems are solved with enough accuracy. Since QuickeNing uses Quasi-Newton steps, one might have expected a superlinear convergence rate as several Quasi-Newton algorithms often enjoy [8]. The situation is as follows. Consider the BFGS Quasi-Newton algorithm (without limited memory). As shown in [9], if the sequence $(\varepsilon_k)_{k\geq 0}$ decreases super-linearly, then, it is possible to design a scheme similar to QuickeNing that indeed enjoys a super-linear convergence rate. There are two major downsides though. The scheme of [9] with such a fast rate requires performing a line-search on F and a super-linearly decreasing sequence $(\varepsilon_k)_{k\geq 0}$ implies an exponentially growing number of iterations in the inner-loops. These two issues make this approach impractical.

Another potential strategy for obtaining a faster convergence rate consists in interleaving a Nesterov-type extrapolation step in the QuickeNing algorithm. Indeed, the convergence rate of QuickeNing scales linearly

in the condition number μ_F/L_F, which suggests that a faster convergence rate could be obtained using a Nesterov-type acceleration scheme. Empirically, we did not observe any benefit of such a strategy, probably because of the pessimistic nature of the convergence rates that are typically obtained for Quasi-Newton approaches based on L-BFGS. Obtaining a linear convergence rate for an L-BFGS algorithm is still an important sanity check, but to the best of our knowledge, the gap in performance between these worst-case rates and practice has always been huge for this class of algorithms.

6.2 Other types of smoothing

Algorithm 2 (ApproxGradient) corresponds to applying the Moreau-Yosida regularization first before computing an estimate g of the gradient, which is a particular instance of infimal convolution smoothing [2], whose family also encompasses the so-called Nesterov smoothing [2]. Other ways to smooth a function include randomization techniques [13] or specific strategies tailored for the objective at hand.

One of the main purposes of the Moreau-Yosida regularization is to provide a better conditioning. As recalled in Proposition 1, the gradient of the Moreau-Yosida-smoothed function F is Lipschitz continuous regardless of whether the original function is continuously differentiable or not. Furthermore, the conditioning of F is improved with respect to the original function, with a condition number depending on the amount of smoothing. As highlighted in [2], this property is also shared by other types of infimal convolutions. Therefore, QuickeNing could potentially be extended to such types of smoothing in place of the Moreau-Yosida regularization. A major advantage of our approach, though, is its outstanding simplicity.

6.3 Concluding remarks

To conclude, we have proposed a generic mechanism, QuickeNing, to accelerate existing first-order optimization algorithms with quasi-Newton-type rules to update a variable metric along the iterations. QuickeNing's main features are the compatibility with composite optimization and its practical performance when combined with incremental approaches. The absence of line-search scheme makes it also easy to implement and use, making it a promising tool for solving large-scale machine learning problems. A few questions remain however open regarding the use of the method in a pure stochastic optimization setting, and the gap in performance between worst-case convergence analysis and practice is significant. We are planning to address the first question about stochastic optimization in future work; the second question is unfortunately difficult and is probably one of the main open question in the literature about L-BFGS methods.

A Variant for the dual approaches MISO/Finito

We focus here on complexity results for the methods MISO/Finito [11, 26], whose convergence rate is given in terms of dual certificate. Specifically, for solving a sub-problem (8), these methods $\mathcal{M}$ are able to produce a sequence $(w_t)_{t\geq 0}$ such that

$$\mathbb{E}[h(w_t) - d(w_t)] \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h^* - d(w_0)) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0. \tag{24}$$

where the function d satisfies $d(w_t) \leq h^*$ for all t. For such approaches, we consider the same restart strategy as in Catalyst [26]—that is, we define

$$w_0 = z_k + \frac{\kappa}{\kappa+\mu}(x_{\text{test}} - x_k) \tag{25}$$

at iteration k, Line 4, of Algorithm 1; If the method is also called at Line 8, we simply replace x_{test} by x_{k+1}. The next modification is a variant of the L-BFGS rule, which we present in Section A.1. Then, we present the inner-loop complexity in Section A.2, which allows us to derive the global complexity analysis of the corresponding QuickeNing scheme.

A.1 Modified L-BFGS rule for MISO/Finito

We use classical L-BFGS rules, which maintain a "generating list" of at most l pairs of vectors $\{(s_i^k, y_i^k)\}_{i=0\ldots j}$ along with an initial diagonal matrix $H_0 = \bar{B}_0^{-1}$. Given a new pair (s, y), the list is updated by removing the oldest pair and replacing it by a new one. With inexact gradients, it is well known that it is sometimes necessary to skip an L-BFGS update in order to ensure the positive-definiteness of the L-BFGS matrix [15]. Typically, these steps occur whenever the condition $s^\top y > 0$ is not satisfied. The convergence analysis for MISO/Finito presented in Section A.2 suggests instead the stronger condition

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s. \tag{26}$$

for $0 < c_1, c_2 \leq 1$, which represents strong-convexity and Lipschitz gradient inequalities when using exact gradients with $c_1 = c_2 = 1$. Intuitively, they are still valid when the gradient accuracy is good enough or when c_1, c_2 are small enough. The role of c_1 and c_2 is to allow us controlling the eigenvalues of the L-BFGS matrix, which is a classical step for analyzing the convergence of L-BFGS algorithms [27] [27]. This is achieved by the next lemma.

Lemma 6 (Largest eigenvalue of the matrix H_k). *Let H_k be constructed by the L-BFGS update with skipping step (26); then,*

$$\lambda_{\max}(H_k) \leq \frac{1}{\kappa c_1^l}\left(1 + \frac{\kappa}{\mu}\right)^l \left(d + \frac{l}{c_2}\right)^{d+l-1},$$

where l is the limited memory parameter and d is the dimension of H_k.

Proof. By construction of the L-BFGS rule, H_k is in fact the inverse of the approximate hessian $\bar{B}_k$, which is constructed from $\bar{B}_0 = \kappa I$ and the generating list $\{(s_i^k, y_i^k)\}_{i=0\ldots j}$ by performing recursively at most l times the BFGS update rule:

$$B' = B + \frac{yy^\top}{y^\top s} - \frac{Bss^\top B}{s^\top Bs},$$

where (s, y) is a pair from the generating list, and B takes the new value B' after each recursion. After scanning all elements of the list and performing $j \leq l$ times the recursion, we obtain $\bar{B}_k$.

We may now proceed and characterize some constants $\lambda_1, \lambda_2 > 0$ such that

$$\lambda_1 \geq \lambda_{\max}(\bar{B}_k) \geq \lambda_{\min}(\bar{B}_k) \geq \lambda_2.$$

Classical analysis shows that with the BFGS update rule, B' is positive definite as soon as B is also positive definite and $s^\top y > 0$. By construction, $s_i^{k\top} y_i^k > 0$ is ensured by (26) for any (s_i^k, y_i^k) in the generating list. Consequently, a simple recurrence shows that all matrices $\bar{B}_k$ are positive definite and therefore $H_k = \bar{B}_k^{-1}$ are well defined. To show that the eigenvalues of $\bar{B}_k$ are bounded, we follow in part the classical analysis of [27] by studying the trace and determinants of $\bar{B}_k$. First,

$$\mathrm{Tr}(B') = \mathrm{Tr}(B) - \frac{\|Bs\|^2}{s^\top Bs} + \frac{\|y\|^2}{y^\top s}.$$

Since the pairs (s, y) from the generating list satisfy $c_2/L_F\|y\|^2 < y^\top s$ from (26),

$$\mathrm{Tr}(B') \leq \mathrm{Tr}(B) + \frac{L_F}{c_2}.$$

Each $\bar{B}_k$ is obtained by performing at most l times BFGS update from $\bar{B}_0$ and $L_F = \kappa$, which leads to

$$\mathrm{Tr}(\bar{B}_k) \leq \mathrm{Tr}(\bar{B}_0) + \frac{lL_F}{c_2} = \kappa\left(d + \frac{l}{c_2}\right).$$

Since $\bar{B}_k$ is definite positive, then $\lambda_{\max}(\bar{B}_k) \leq \mathrm{Tr}(\bar{B}_k) \leq \lambda_1$ with $\lambda_1 = \kappa\,(d + l/c_2)$. Furthermore, the determinant of B' satisfies the following relation:

$$\det(B') = \det(B)\frac{y^\top s}{s^\top B s} = \det(B)\,\frac{y^\top s}{s^\top s}\,\frac{s^\top s}{s^\top B s}.$$

Again, since any pair (s, y) in the generating list satisfies $c_1\mu_F\|s\|^2 < y^\top s$ from (26) and $\lambda_{\max}(B)$ is bounded by λ_1, we obtain

$$\det(B') \geq \det(B)\cdot c_1\mu_F \cdot \frac{1}{\lambda_{\max}(B)} \geq \det(B)\frac{c_1\mu_F}{\lambda_1}.$$

As a result, since $\bar{B}_k$ is obtained by performing at most l times update, we have

$$\det(\bar{B}_k) \geq \det(\bar{B}_0)\left(\frac{c_1\mu_F}{\lambda_1}\right)^l = \kappa^d\left(\frac{c_1\mu_F}{\lambda_1}\right)^l.$$

Therefore,

$$\lambda_{\min}(\bar{B}_k) \geq \frac{\det(\bar{B}_k)}{\lambda_{\max}(\bar{B}_k)^{d-1}} \geq \frac{\det(\bar{B}_k)}{\lambda_1^{d-1}} \geq \frac{(c_1\mu_F)^l\kappa^d}{\lambda_1^{d+l-1}}.$$

We conclude by remarking that

$$\lambda_{\max}(H_k) = \frac{1}{\lambda_{\min}(\bar{B}_k)} \quad \leq \quad \frac{\lambda_1^{d+l-1}}{(c_1\mu_F)^l\kappa^d} = \frac{(\kappa(d+\frac{l}{c_2}))^{d+l-1}}{(c_1\frac{\mu\kappa}{\mu+\kappa})^l\kappa^d} = \frac{1}{\kappa c_1^l}\left(1+\frac{\kappa}{\mu}\right)^l\left(d+\frac{l}{c_2}\right)^{d+l-1}.$$

□

A.2 Inner-loop analysis for MISO/Finito

We now discuss the complexity analysis for the dual approaches MISO/Finito, which uses a few results from [26] such as the following lemma:

Lemma 7 (ε**-accuracy by moving the prox-center ([26], Lemma D.5)).** *Consider Algorithm 1 used with the method MISO/Finito. Assume that at iteration $k-1$, we compute z_k by solving a sub-problem (8) up to accuracy ε_k in the sense of (24). Then, the restart (25) for sub-problem (8) at iteration k satisfies*

$$h^* - d(w_0) \leq \varepsilon_k + \frac{\kappa^2}{2(\kappa+\mu)}\|x - x_k\|^2,$$

where $x = x_{test}$ when considering the sub-problem of Line 4 of Algorithm 1 and $x = x_{k+1}$ otherwise.

Then, we control the deviation $\|x - x_k\|^2$ in the previous lemma and obtain the next relation, which assumes that the sequence $(\varepsilon_k)_{k\geq 0}$ of accuracies is pre-defined.

Lemma 8 (ε**-accuracy in QuickeNing with pre-defined sequence** $(\varepsilon_k)_{k\geq 0}$**).** *Consider the setting of Corollary 1, where the objective is strongly convex and the predefined sequence $\varepsilon_k = \frac{1}{2}C(1-\rho)^{k+1}$ is used within Algorithm 1. Under the same assumptions made in Lemma 7, we have*

$$h^* - d(w_0) \leq \varepsilon_k\left(1 + 2\left(\frac{C}{\rho}+1\right)\kappa^2 D\right),$$

where $D = \lambda_{\max}(H_k)^2$ when considering the sub-problem of Line 4 of Algorithm 1 and $D = 1/\kappa^2$ for the sub-problem of Line 8.

Proof. We start with the case $x = x_{\text{test}}$.

$$\begin{aligned}
\|x_{\text{test}} - x_k\|^2 = \|H_k g_k\|^2 &\leq \lambda_{\max}(H_k)^2 \|g_k\|^2 \\
&\leq 2\lambda_{\max}(H_k)^2(\|\nabla F(x_k)\|^2 + 2\kappa\varepsilon_k) \quad (\text{ from (14) }) \\
&\leq 2\lambda_{\max}(H_k)^2(2\kappa(F(x_k) - F^*) + 2\kappa\varepsilon_k) \quad (\text{Lipschitz gradient}) \\
&\leq 4\kappa\lambda_{\max}(H_k)^2 \left(\frac{C}{\rho}(1-\rho)^{k+2} + \varepsilon_k\right) \quad (\text{from (19)}) \\
&= 4\kappa \left(\frac{C}{\rho} + 1\right) \lambda_{\max}(H_k)^2 \varepsilon_k.
\end{aligned}$$

Applying Lemma 7 with $x = x_{\text{test}}$ gives the result. Next, we can replace in the previous proof, x_{test} by $x_{k+1} = z_k$, and by noticing that $z_k - x_k = g_k/\kappa$, it is sufficient to replace H_k by $1/\kappa I_d$ and to follow exactly the same analysis to conclude. □

By combining the previous lemma with the convergence guarantee (24), we see that we can solve the sub-problems (8) at iteration k up to accuracy $\varepsilon_{k+1} = (1-\rho)\varepsilon_k$ in a constant number of iterations $T_{\mathcal{M}}$ of the method $\mathcal{M}$. The arguments are in fact exactly the same as in Proposition 3.2 of Catalyst [26]. The only caveat is that $T_{\mathcal{M}}$ increases linearly with the dimension d, which is also a limitation of classical worst-case convergence analysis of L-BFGS methods. Note that our previous analysis for primal methods, which was presented in Section 4, does not suffer from this issue.

References

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

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

[3] D. P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

[4] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[5] J. Burke and M. Qian. On the superlinear convergence of the variable metric proximal point algorithm using Broyden and BFGS matrix secant updating. *Mathematical Programming*, 88(1):157–181, 2000.

[6] R. Byrd, S. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[7] R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for L-1 regularized optimization. *Mathematical Programming*, 157(2):375–396, 2015.

[8] R. H. Byrd, J. Nocedal, and Y.-X. Yuan. Global convergence of a case of quasi-Newton methods on convex problems. *SIAM Journal on Numerical Analysis*, 24(5):1171–1190, 1987.

[9] X. Chen and M. Fukushima. Proximal quasi-Newton methods for nondifferentiable convex optimization. *Mathematical Programming*, 85(2):313–334, 1999.

[10] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[11] A. Defazio, J. Domke, and T. S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2014.

[12] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[13] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[14] M. Elad. *Sparse and Redundant Representations*. Springer, 2010.

[15] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

[16] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2015.

[17] M. Fuentes, J. Malick, and C. Lemaréchal. Descentwise inexact proximal algorithms for smooth optimization. *Computational Optimization and Applications*, 53(3):755–769, 2012.

[18] M. Fukushima and L. Qi. A globally and superlinearly convergent algorithm for nonsmooth convex minimization. *SIAM Journal on Optimization*, 6(4):1106–1120, 1996.

[19] S. Ghadimi, G. Lan, and H. Zhang. Generalized Uniformly Optimal Methods for Nonlinear Programming. *arxiv:1508.07384*, 2015.

[20] R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2016.

[21] O. Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[22] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I.* Springer, 1996.

[23] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms. II.* Springer, 1996.

[24] J. Lee, Y. Sun, and M. Saunders. Proximal Newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[25] C. Lemaréchal and C. Sagastizábal. Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

[26] H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[27] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[28] J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[29] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[30] R. Mifflin. A quasi-second-order proximal bundle algorithm. *Mathematical Programming*, 73(1):51–72, 1996.

[31] P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[32] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[33] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[34] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[35] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

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

[37] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[38] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[39] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[40] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[41] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[42] S. Salzo and S. Villa. Inexact and accelerated proximal point algorithms. *Journal of Convex Analysis*, 19(4):1167–1192, 2012.

[43] K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, 160(1):495–529, 2016.

[44] M. Schmidt, D. Kim, and S. Sra. *Projected Newton-type methods in machine learning*, pages 305–330. MIT Press, 2011.

[45] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 160(1):83–112, 2017.

[46] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2016.

[47] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[48] J. Yu, S. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to non-smooth convex optimization. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2008.

[49] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.