An algorithm for the minimization of nonsmooth nonconvex functions using inexact evaluations and its worst-case complexity

Gratton, Serge; Simon, Ehouarn; Toint, Philippe

Published in:
Mathematical Programming

DOI:
10.1007/s10107-020-01466-5

Publication date:
2020

Document Version
Peer reviewed version

An algorithm for the minimization of nonsmooth nonconvex functions using inexact evaluations and its worst-case complexity

S. Gratton [*], E. Simon [†], and Ph. L. Toint [‡]

2 January 2020

Abstract

An adaptive regularization algorithm using inexact function and derivatives evaluations is proposed for the solution of composite nonsmooth nonconvex optimization. It is shown that this algorithm needs at most $O(|\log(\epsilon)|\,\epsilon^{-2})$ evaluations of the problem's functions and their derivatives for finding an ϵ-approximate first-order stationary point. This complexity bound therefore generalizes that provided by [Bellavia, Gurioli, Morini and Toint, 2018] for inexact methods for smooth nonconvex problems, and is within a factor $|\log(\epsilon)|$ of the optimal bound known for smooth and nonsmooth nonconvex minimization with exact evaluations. A practically more restrictive variant of the algorithm with worst-case complexity $O(|\log(\epsilon)| + \epsilon^{-2})$ is also presented.

Keywords: evaluation complexity, nonsmooth problems, nonconvex optimization, composite functions, inexact evaluations.

1 Introduction

We consider the problem of finding a local minimum of the following composite problem:

$$\min_{x\in\mathbb{R}^n} \psi(x) = f(x) + h(c(x)), \tag{1.1}$$

where f is a (possibly nonconvex) function from $\mathbb{R}^n$ into $\mathbb{R}$ whose gradient is Lipschitz continuous, c is a (possibly nonconvex) function from $\mathbb{R}^n$ into $\mathbb{R}^m$, whose Jacobian is also Lipschitz continuous, and where h is a convex (possibly nonsmooth) Lipschitz continuous function from $\mathbb{R}^m$ into $\mathbb{R}$.

Such problems occur in a variety of contexts, like LASSO methods in computational statistics [28], Tikhonov regularization of underdetermined estimation problems [19], compressed

[*]Université de Toulouse, INP, IRIT, Toulouse, France. Email: serge.gratton@enseeiht.fr

[†]Université de Toulouse, INP, IRIT, Toulouse, France. Email: ehouarn.simon@enseeiht.fr

[‡]naXys, University of Namur, Namur, Belgium. Email: philippe.toint@unamur.be. Partially supported by ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

sensing [15], artificial intelligence [22], penalty or projection methods for constrained optimization [9], reduced-precision deep-learning [29], image processing [1], to cite only a few examples. We also refer the reader to the excellent review in [23]. In many of these applications, the function h is cheap to compute, for example if $h(x)$ is the Euclidean, ℓ_1 or ℓ_∞ norm, and its Lipschitz constant is known.

Methods to calculate approximate local solutions of the nonconvex problem (1.1) have been studied for many years. (We do not consider here the abundant literature on the easier convex case, see [16] or [14] for recent instances or [7] for a general text.) If h is differentiable, standard methods include steepest descent, Levenberg-Morrison-Marquardt quadratic regularization algorithms or trust-region techniques (see [13]). In this case, the evaluation complexity (that is the number of times the functions f and c are evaluated for finding an ϵ-approximate first-order point) is proved to be $O(\epsilon^{-2})$ [25, 17]. Moreover, this order is known to be optimal [10]. If h is nonsmooth, applicable methods can be found in the abundant literature for bundle methods (see [20] and the references therein), and also involve the proximal gradient method and its variants [26], as well as the nonsmooth trust-region and quadratic regularization methods of [9] for the nonconvex ones. It was also shown in this paper that the evaluation compexity remains $O(\epsilon^{-2})$ despite nonsmoothness.

Inexact function evaluations are however quite commonly met in practice. For instance, $f(x)$ or $c(x)$ may be the result of some truncated iterative process, making the accuracy of the computed values dependent on the truncation level. Or $f(x)$ or $c(x)$ could be computed as statistical estimates with controlled error (e.g. in subsampling methods for additive problems). Or they may result from the need (for embarked processors) or desire (for high-end supercomputers) to perform their evaluation in restricted arithmetic precision whenever possible. Convergence analysis results for methods with inexact function and/or derivatives values exist [8, 13, 30, 11, 2, 24, 12, 6, 4, 3] and their practical performance considered [8, 18], but all these contributions assume smoothness of the objective function. To the best of the authors' knowledge, the only analysis for nonconvex composite problems allowing inexact evaluations is [20], but this contribution uses assumptions different from those presented below and does not discuss upper complexity bounds.

The contribution of the present paper is threefold.

- We first propose a new regularization method for the nonsmooth problem (1.1) that uses dynamic accuracy.
- We then show that the optimal $O(\epsilon^{-2})$ evaluation complexity bound is preserved when using this algorithm, up to a (typically modest) factor $|\log(\epsilon)|$.
- We finally present a variant of the algorithm for which a better complexity bound of $O(|\log(\epsilon)| + \epsilon^{-2})$ can be proved at the price of losing some practicality.

Our presentation is organized as follows. Section 2 discusses the nature of the inexact evaluations and presents the new regularization algorithm, whose essential properties are then developed in Section 3. The corresponding evaluation complexity bound is derived in Section 4. A practically more restrictive variant of the algorithm with better worst-case complexity is presented in Section 5. What can happen if accuracy is limited is discussed in Section 6. Conclusions are outlined in Section 7.

2 The Adaptive Regularization Algorithm using Dynamic Accuracy

As indicated above, we assume that f, c and their derivatives are computed inexactly but that h is exact and its cost negligible compared to that of obtaining approximate value for f, c or their derivatives. Moreover, we assume that, for some *known* constant $L_h \geq 0$,

$$\|h(v) - h(w)\| \leq L_h \|v - w\| \quad \text{for all } \; v, w \in \mathbb{R}^m. \tag{2.1}$$

where $\|\cdot\|$ denotes the standard Euclidean norm.

2.1 An outline

Our algorithm is iterative and of the adaptive regularization type. It constructs a sequence of iterates $\{x_k\}$, at which the function ψ and its derivatives are computed inexactly. For exact values, we use the notations

$$f_k \stackrel{\rm def}{=} f(x_k), \quad g_k \stackrel{\rm def}{=} g(x_k) = \nabla^1_x f(x_k), \quad c_k \stackrel{\rm def}{=} c(x_k), \quad J_k \stackrel{\rm def}{=} J(x_k) = \nabla^1_x c(x_k)$$

and $\psi_k \stackrel{\rm def}{=} \psi(x_k) = f_k + h(c_k)$. The "linearization"

$$\ell_k(s) \stackrel{\rm def}{=} f_k + g_k^T s + h(c_k + J_k s).$$

will play in important role in what follows. In particular, we use the fact that

$$\min_{\|d\| \leq 1} \ell_k(d) = \ell_k(0) = \psi_k$$

if x_k is a local minimizer of (1.1) [31, Lemma 2.1] and we say that x_k is an ϵ-approximate minimizer if

$$\phi_k \leq \epsilon, \tag{2.2}$$

where

$$\phi_k \stackrel{\rm def}{=} \phi(x_k) \stackrel{\rm def}{=} \ell_k(0) - \min_{\|d\| \leq 1} \ell_k(d) = \max_{\|d\| \leq 1} \Delta\ell_k(d), \tag{2.3}$$

with

$$\Delta\ell_k(s) \stackrel{\rm def}{=} \ell_k(0) - \ell_k(s) = -g_k^T s + h(c_k) - h(c_k + J_k s).$$

If x_k is not such a point, the standard exact regularization algorithm [9] computes a trial step s_k by approximately minimizing the regularized model

$$m_k(s) \stackrel{\rm def}{=} \ell_k(s) + \frac{\sigma_k}{2}\|s\|^2 \tag{2.4}$$

over all $s \in \mathbb{R}^n$, where σ_k is an adaptive "regularization parameter", which is adjusted by the algorithm we described below, starting from an arbirary initial value. This yields the model decrease $\Delta m_k(s_k)$, where

$$\Delta m_k(s) \stackrel{\rm def}{=} m_k(0) - m_k(s) = -g_k^T s + h(c_k) - h(c_k + J_k s) - \frac{\sigma_k}{2}\|s\|^2.$$

The value of the objective function is then computed at the trial point $x_k + s_k$, which is accepted as the new iterate if the achieved reduction $\psi_k - \psi(x_k + s_k)$ compares well with the predicted decrease $\Delta\ell_k(s_k)$. The regularization parameter σ_k is then updated to reflect the quality of this prediction and a new iteration started.

2.2 An accuracy model

In our context of inexact values for f and c, we will keep the same general algorithm outline, but will also need to take action to handle the fact that evaluations of f, c, g and J are inexact. In order to reach any conclusion regarding the true problem (that is the problem using exact evaluations), we must make some assumptions on the possible errors occuring in the evaluations. To make the discussion more focused, consider the evaluation of the objective function f at a given point $x \in \mathbb{R}^n$. A very general accuracy model would be to assume that we have an accuracy parameter a_f which we can specify and such that the resulting inexact value $\widehat{f}(x, a_f)$ satisfies

$$|\widehat{f}(x, a_f) - f(x)| \leq \kappa_f(x) a_f. \tag{2.5}$$

This implies, in particular, that sufficiently reducing a_f also reduces the error $|\widehat{f}(x, a_f) - f(x)|$. Several cases are then of interest. A first case is when $\kappa_f(x) = \kappa_f$ is independent of x or an upper bound on its value ($\kappa_f(x) \leq \kappa_{f,\max}$ for all x) is known a priori. This occurs, for instance, in some subsampling algorithms for deep learning (see [27]) or binary classification problems modelled by the sigmoid function and least-squares loss (see [3]). Other examples include the case where f is linear combination of basis functions whose coefficients can be computed to prescribed accuracy (in multivariate interpolation, for instance), or more general iterative process whose termination guarantees a final user-specified accuracy. A second case is when $\kappa_f(x)$ depends of x and an upper bound on its value ($\kappa_f(x) \leq \kappa_{f,\max}(x)$) is computable together with $\widehat{f}(x, a_f)$ once x is known. An example is when the *relative* accuracy of $\widehat{f}(x, a_f)$ can be controlled, say when

$$|\widehat{f}(x, a_f) - f(x)| \leq (\zeta + |f(x)|) a_f \quad \text{or} \quad |\widehat{f}(x, a_f) - f(x)| \leq (\zeta + |\widehat{f}(x, a_f)|) a_f \tag{2.6}$$

for some small $\zeta \geq 0$ and some computable function $u(x)$ is known such that $|f(x)| \leq u(x)$ or $|\widehat{f}(x, a_f)| \leq u(x)$. (Of course, $u(x)$ might be constant when global upper bounds on $|f(x)|$ or $|\widehat{f}(x, a_f)|$ are known.) Computation in variable precision using an idealized truncation model (see [18] and references therein) is a framework where the first of the inequalities of (2.6) typically holds. A third case where no bound on $\kappa_f(x)$ is computable could unfortunately also happen. This occurs, for instance, in [21] where a PDE application is described in which $\kappa_f(x)$ depends on an adaptive mesh. However, this third case then prevents any algorithm to include a termination rule relating an inexact solution to a true one. While the algorithm may be proved convergent to a true solution in the limit, it has to remain theoretical and *a priori* evaluation complexity bound can possibly be given for the computation of an approximate solution. Indeed, the authors of [21] derive a trust-region algorithm inspired from [13, Section 10.6] that is convergent to a true solution, but this algorithm does not include any termination rule (and the practical termination criterion used in the numerical illustration is not mentioned).

In order to cover the first two cases (given our purpose to estimate an evaluation complexity bound, we exclude the third), we assume that, given an absolute accuracy request ε_f, we can compute $\overline{f}(x)$ such that

$$\overline{f}(x) \stackrel{\text{def}}{=} \overline{f}(x, \varepsilon_f) \quad \text{with} \quad |\overline{f}(x, \varepsilon_f) - f(x)| \leq \varepsilon_f, \tag{2.7}$$

possibly by evaluating $\widehat{f}(x, a_f)$ in (2.5) for several values of a_f. Let $n_f(x)$ denote the number

of these evaluations. In case 1, defining

$$\overline{f}(x,\varepsilon_f) = \widehat{f}\left(x, \frac{\varepsilon_f}{\kappa_{f,\max}}\right)$$

then ensure that (2.7) holds with $n_f(x) = 1$. The situation in case 2 is more complicated because the evaluation of $\kappa_{f,\max}(x)$ may itself require some additional evaluations. Note that, in the relative accuracy case of (2.6) where the upper bounding function $u(x)$ is known, setting

$$\overline{f}(x,\varepsilon_f) = \widehat{f}\left(x, \frac{\varepsilon_f}{u(x)}\right)$$

also yields $n_f(x) = 1$.

In what follows, we will denote inexactly computed quantities with an overbar and we apply the model discussed above for f to f, g, c, and J, and assume the accuracy parameters ε_f, ε_g, ε_c and ε_J are bounded above and that, given $x_k \in \mathbb{R}^n$, ε_f, ε_g, ε_c and ε_J, approximate values of $f(x)$, $g(x)$, $c(x)$ and $J(x)$ can be computed such that

$$\overline{f}_k = \overline{f}(x_k,\varepsilon_f) \quad \text{and} \quad |\overline{f}_k - f_k| \le \varepsilon_f, \tag{2.8}$$

$$\overline{g}_k = \overline{g}(x_k,\varepsilon_g) \quad \text{and} \quad \|\overline{g}_k - g_k\| \le \varepsilon_g, \tag{2.9}$$

$$\overline{c}_k = \overline{c}(x_k,\varepsilon_c) \quad \text{and} \quad \|\overline{c}_k - c_k\| \le \varepsilon_c, \tag{2.10}$$

$$\overline{J}_k = \overline{J}(x_k,\varepsilon_J) \quad \text{and} \quad \|\overline{J}_k - J_k\| \le \varepsilon_J, \tag{2.11}$$

with associated number of evaluations $n_{f,k}$, $n_{g,k}$, $n_{c,k}$ and $n_{J,k}$. The accuracy level on $\overline{f}$, $\overline{g}$, $\overline{c}$ and $\overline{J}$ is thus dynamic, in the sense that ε_f, ε_g, ε_c and ε_J are specified by the algorithm in order to ensure its meaningful progress.

We will then consider the inexact objective function $\overline{\psi}(x) = \overline{f}(x) + h(\overline{c}(x))$, together with its "linearization" and model given by

$$\overline{\ell}_k(s) \stackrel{\rm def}{=} \overline{f}_k + \overline{g}_k^T s + h(\overline{c}_k + \overline{J}_k s) \ \text{ and } \ \overline{m}_k(s) \stackrel{\rm def}{=} \overline{\ell}_k(s) + \frac{\sigma_k}{2}\|s\|^2,$$

defining their corresponding decreases by

$$\overline{\Delta\ell}_k(s_k) \stackrel{\rm def}{=} -\overline{g}_k^T s + h(\overline{c}_k) - h(\overline{c}_k + \overline{J}_k s) \tag{2.12}$$

and

$$\overline{\Delta m}_k(s) \stackrel{\rm def}{=} -\overline{g}_k^T s + h(\overline{c}_k) - h(\overline{c}_k + \overline{J}_k s) - \frac{\sigma_k}{2}\|s\|^2. \tag{2.13}$$

Finally, the criticality measure ϕ_k will be approximated by

$$\overline{\phi}_k \stackrel{\rm def}{=} \overline{\ell}_k(0) - \min_{\|d\|\le 1} \overline{\ell}_k(d) = \max_{\|d\|\le 1} \overline{\Delta\ell}_k(d).$$

Armed with these definitions, we may establish the following crucial error bounds.

Lemma 2.1 We have that, for any k,

$$|\overline{\psi}_k - \psi_k| \le \varepsilon_f + L_h\varepsilon_c \tag{2.14}$$

and, for any $v \in \mathbb{R}^n$,

$$|\overline{\Delta m}_k(v) - \Delta m_k(v)| = |\overline{\Delta\ell}_k(v) - \Delta\ell_k(v)| \le (\varepsilon_g + L_h\varepsilon_J)\|v\| + 2L_h\varepsilon_c. \tag{2.15}$$

Proof. Using successively (1.1), the triangle inequality, (2.1), (2.8) and (2.10), we obtain that

$$\begin{aligned} |\overline{\psi}_k - \psi_k| &= |\overline{f}_k + h(\overline{c}_k) - f_k - h(c_k)| \\ &\leq |\overline{f}_k - f_k| + |h(\overline{c}_k) - h(c_k)| \\ &\leq \varepsilon_f + L_h \|\overline{c}_k - c_k\| \\ &\leq \varepsilon_f + L_h \varepsilon_c \end{aligned}$$

and hence (2.14) holds. Similarly, using now (2.13), (2.12), the triangle and Cauchy-Schwarz inequalities, (2.1), (2.9), (2.10) and (2.11), we deduce that

$$\begin{aligned} |\overline{\Delta m}_k(v) - \Delta m_k(v)| &= |\overline{\Delta \ell}_k(v) - \Delta \ell_k(v)| \\ &\leq |(\overline{g}_k - g_k)^T v| + |h(\overline{c}_k) - h(c_k)| + |h(\overline{c}_k + \overline{J}_k v) - h(c_k + J_k v)| \\ &\leq \|\overline{g}_k - g_k\| \, \|v\| + L_h \|\overline{c}_k - c_k\| + L_h \|\overline{c}_k + \overline{J}_k v - c_k - J_k v\| \\ &\leq \|\overline{g}_k - g_k\| \, \|v\| + L_h \|\overline{c}_k - c_k\| + L_h (\|\overline{c}_k - c_k\| + \|\overline{J}_k - J_k\| \, \|v\|) \\ &\leq \varepsilon_g \|v\| + L_h \varepsilon_c + L_h (\varepsilon_c + \varepsilon_J \|v\|) \\ &\leq (\varepsilon_g + L_h \varepsilon_J) \|v\| + 2 L_h \varepsilon_c. \end{aligned}$$

□

2.3 The ARLDA algorithm

Broadly inspired by [3], we may now state our inexact adaptive regularization algorithm formally, in two stages. We first describe its global framework on the following page, delegating the more complicated questions of verifying optimality and computing the step to more detailed sub-algorithms to be presented in the second stage.

In the ARLDA [(1)] algorithm on the next page, $\varepsilon_f^{\max}$, $\varepsilon_g^{\max}$, $\varepsilon_c^{\max}$ and $\varepsilon_J^{\max}$ stand for upper bounds on ε_f, ε_g, ε_c and ε_J, and ω_k can be viewed as an iteration dependent relative accuracy level on f, ψ and the model decreases.

A few comments on this first view of the algorithm are now useful.

1. The words "If unavailable" at the beginning of Step 1 will turn out to be fairly important. In a context where the values of $\overline{f}_k$, $\overline{g}_k$, $\overline{c}_k$ and $\overline{J}_k$ may need to be computed several times but with different accuracy requirements in the course of the same iteration (as we will see below), they indicate that, if one of these functions has already been computed at the current iterate with the desired accuracy, it need not (of course) be recomputed. This imposes the minor task of keeping track of the smallest value of the relevant ε for which each of these functions has been evaluated at the current iterate.

2. Observe that the relative accuracy threshold ω_k is recurred from iteration to iteration, and the absolute accuracy requirements ε_f, ε_g, ε_c and ε_J are then determined in the hope to enforce the relative error (see (2.17) and (2.18) for $\overline{f}$ at $x_k + s_k$).

[(1)] For Adaptive Regularization with Lipschitz model and Dynamic Accuracy.

Algorithm 2.1: The ARLDA Algorithm

Step 0: Initialization. An initial point x_0 and an initial regularization parameter $\sigma_0 > 0$ are given, as well as an accuracy level $\epsilon \in (0,1)$. The constants α, κ_ω, η_1, η_2, γ_1, γ_2, γ_3, $\varepsilon_f^{\max}$, $\varepsilon_g^{\max}$, $\varepsilon_c^{\max}$, $\varepsilon_J^{\max}$, γ_ε and $\sigma_{\min}$ are also given and satisfy $\sigma_{\min} \in (0, \sigma_0]$,

$$0 < \eta_1 \leq \eta_2 < 1, \;\; 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3, \;\; \alpha \in (0,1), \;\; \gamma_\varepsilon \in (0,1).$$

Choose $\varepsilon_f \leq \varepsilon_f^{\max}$, $\varepsilon_g \leq \varepsilon_g^{\max}$, $\varepsilon_c \leq \varepsilon_c^{\max}$, $\varepsilon_J \leq \varepsilon_J^{\max}$ and $\kappa_\omega \in (0, \frac{1}{3}\alpha\eta_1]$ such that $\omega_0 = \varepsilon_f + L_h\varepsilon_c \leq \min[\kappa_\omega, \sigma_0^{-1}]$. Set $k = 0$.

Step 1: Compute the optimality measure and check for termination.

If unavailable, compute $\overline{f}_k$, $\overline{g}_k$, $\overline{c}_k$ and $\overline{J}_k$ satisfying (2.8)–(2.11). Apply Algorithm 2.2 to check for termination with the iterate x_k and $\overline{\psi}(x_k) = \overline{f}_k + h(\overline{c}_k)$, or to obtain $\overline{\phi}_k > \epsilon/(1+\omega_k)$ if termination does not occur.

Step 2: Step calculation. Apply Algorithm 2.3 to approximately minimize $\overline{m}_k(s)$ and obtain a step s_k and the corresponding linearized decrease $\overline{\Delta\ell}_k(s_k)$ such that

$$\overline{\Delta\ell}_k(s_k) \geq \frac{1}{4}\min\left\{1, \frac{\overline{\phi}_k}{\sigma_k}\right\}\overline{\phi}_k. \tag{2.16}$$

Step 3: Acceptance of the trial point. Possibly reduce ε_f to ensure that

$$\varepsilon_f \leq \omega_k\overline{\Delta\ell}_k(s_k). \tag{2.17}$$

If ε_f has been reduced, recompute $\overline{f}_k(x_k, \varepsilon_f)$ to ensure (2.8). Then compute $\overline{f}_k(x_k+s_k, \varepsilon_f)$ such that

$$|\overline{f}_k(x_k+s_k, \varepsilon_f) - f(x_k+s_k)| \leq \varepsilon_f, \tag{2.18}$$

set $\overline{\psi}(x_k+s_k) = \overline{f}_k(x_k+s_k, \varepsilon_f) + h(\overline{c}_k(x_k+s_k, \varepsilon_c))$, $\overline{\psi}_k = \overline{f}_k(x_k, \varepsilon_f) + h(\overline{c}_k(x_k, \varepsilon_c))$ and define

$$\rho_k = \frac{\overline{\psi}_k - \overline{\psi}(x_k+s_k)}{\overline{\Delta\ell}_k(s_k)}. \tag{2.19}$$

If $\rho_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$; otherwise define $x_{k+1} = x_k$.

Step 4: Regularization parameter update. Set

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k] & \text{if } \rho_k \geq \eta_2, \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \rho_k < \eta_1. \end{cases} \tag{2.20}$$

Step 5: Relative accuracy update. Set

$$\omega_{k+1} = \min\left[\kappa_\omega, \frac{1}{\sigma_{k+1}}\right] \tag{2.21}$$

and redefine $\varepsilon_f \leq \varepsilon_f^{\max}$, $\varepsilon_g \leq \varepsilon_g^{\max}$, $\varepsilon_c \leq \varepsilon_c^{\max}$ and $\varepsilon_J \leq \varepsilon_J^{\max}$ such that $\varepsilon_f + L_h\varepsilon_c \leq \omega_{k+1}$. Increment k by one and go to Step 1.

3. However, the redefinition of the absolute accuracy requirements in Step 5 leaves much freedom. One possible implementation of this redefinition would be to set

$$\begin{aligned} \varepsilon_f &= \min\left[\varepsilon_f^{\max}, \frac{\omega_{k+1}}{\omega_k}\varepsilon_f\right], \quad \varepsilon_g = \min\left[\varepsilon_g^{\max}, \frac{\omega_{k+1}}{\omega_k}\varepsilon_g\right], \\ \varepsilon_c &= \min\left[\varepsilon_c^{\max}, \frac{\omega_{k+1}}{\omega_k}\varepsilon_c\right], \quad \varepsilon_J = \min\left[\varepsilon_J^{\max}, \frac{\omega_{k+1}}{\omega_k}\varepsilon_J\right], \end{aligned} \tag{2.22}$$

   but this is by no means the only possible choice. In particular, any choice of $\varepsilon_g \leq \varepsilon_g^{\max}$ and $\varepsilon_J \leq \varepsilon_J^{\max}$ is permitted. Observe that since the sequence $\{\sigma_k\}$ produced by (2.20) (or (2.22)) need not be monotonically increasing, the sequence $\{\omega_k\}$ constructed in (2.21) need not be decreasing. We present an alternative to this choice in Section 5.

4. We will verify in Lemma 3.1 below that the sufficient-decrease requirement (2.16) is fairly loose. In fact the constant $\frac{1}{4}$ in this condition can be replaced by any constant in $(0, \frac{1}{2})$ and/or $\overline{\phi}_k$ replaced by ϵ without affecting our theoretical results.

5. When exact functions values can be computed (i.e. $\varepsilon_f = \varepsilon_g = \varepsilon_c = \varepsilon_J = 0$), the ARLDA algorithm essentially reduces to the regularization algorithm of [9]. It is also close in spirit to the ARpDA algorithm for $p = 1$ (AR1DA) of [3] when $h = 0$ and the problem becomes smooth, but the step computation is simpler in this reference because $\overline{\Delta\ell}_k(s)$ only involves derivatives' values in that case.

6. The algorithm presented in [20] in the context of general nonsmooth functions assumes the Lipschitz nature of the objective function but does not use any Lipschitz constant explictly. This is of course more general than the context discussed here. However, this point of view ignores the problem's structure and we are not aware of any complexity result for methods at that level of generality.

The purpose of Step 1 of the ARLDA algorithm is to check for termination by computing a value of $\overline{\phi}_k$ which is *relatively* sufficiently accurate. As can be expected, computing a relatively accurate value when $\overline{\phi}_k$ itself tends to zero may be too demanding, but we nevertheless design a mechanism that will allow us to prove (in Lemma 3.3 below) that true ϵ-optimality can be reached in this case. The details of the resulting Step 1 are given in Algorithm 2.2 on the following page. Observe that this algorithm introduces a possible loop on the accuracy requirement, between Step 1.3 and Step 1.

Once the algorithm has determined in Step 1 that termination cannot occur at the current iterate, it next computes s_k in Step 2. In this computation, the relative accuracy of the "linearized decrease" $\overline{\Delta\ell}_k(s_k)$ must again be assessed. This is achieved in Algorithm 2.3 on the next page.

As for Algorithm 2.2, this algorithm introduces a possible loop on the accuracy requirement, between Step 2.3 and Step 1. We will show (in Lemma 3.5 below) that these loops are finite, and thus that the ARLDA algorithm is well-defined.

Algorithm 2.2: Check for termination in Algorithm 2.1

Step 1.1. Solve

$$\max_{\|d\|\leq 1} \overline{\Delta\ell}_k(d) \tag{2.23}$$

to obtain a global maximizer d_k and the corresponding $\overline{\Delta\ell}_k(d_k)$.

Step 1.2.

- If
$$\varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \leq \omega_k\overline{\Delta\ell}_k(d_k), \tag{2.24}$$
then
  - define $\overline{\phi}_k = \overline{\Delta\ell}_k(d_k)$;
  - if $\overline{\phi}_k \leq \epsilon/(1+\omega_k)$, terminate the ARLDA algorithm with `exit` $= 1$;
  - else go to Step 2 of the ARLDA algorithm.
- If
$$\overline{\Delta\ell}_k(d_k) \leq \tfrac{1}{2}\epsilon \ \text{ and } \ \varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \leq \tfrac{1}{2}\epsilon, \tag{2.25}$$
terminate the ARLDA algorithm with `exit` $= 2$.

Step 1.3: Multiply ε_g, ε_c and ε_J by γ_ε and restart Step 1 of the ARLDA algorithm.

Algorithm 2.3: Compute the step s_k in Algorithm 2.1

Step 2.1: Approximately solve

$$\min_{s\in\mathbb{R}^n} \overline{m}_k(s) \tag{2.26}$$

to obtain a step s_k together with $\overline{\Delta m}_k(s_k)$ and $\overline{\Delta\ell}_k(s_k)$.

Step 2.2: If

$$(\varepsilon_g + L_h\varepsilon_J)\|s_k\| + 2L_h\varepsilon_c \leq \omega_k\overline{\Delta\ell}_k(s_k), \tag{2.27}$$

go to Step 3 of the ARLDA algorithm.

Step 2.3: Otherwise multiply ε_g, ε_c and ε_J by γ_ε and return to Step 1 of the ARLDA algorithm.

3 Properties of the ARLDA algorithm

Having defined the algorithm, we turn to establishing some of its properties, which will be central to the forthcoming complexity analysis. We first verify that the requirement (2.16) can always be achieved.

Lemma 3.1 A step s_k satisfying

$$\overline{\Delta \ell}_k(s_k) \geq \overline{\Delta m}_k(s_k) \geq \frac{1}{2} \min \left\{ 1, \frac{\overline{\phi}_k}{\sigma_k} \right\} \overline{\phi}_k \tag{3.1}$$

(and hence also satisfying (2.16)) can always be computed.

Proof. The first inequality results from (2.13) and (2.12). The second is given in [9, Lemma 2.5], and hinges on the convexity of h. □

We next show an alternative lower bound on the linearized decrease, directly resulting from the model's definition.

Lemma 3.2 For all $k \geq 0$, we have that

$$\overline{\Delta \ell}_k(s_k) \geq \frac{\sigma_k}{2} \|s_k\|^2. \tag{3.2}$$

Moreover, as long as the algorithm has not terminated,

$$\overline{\Delta \ell}_k(s_k) \geq \delta_k(\epsilon) \stackrel{\rm def}{=} \frac{1}{16} \min \left\{ 1, \frac{\epsilon}{\sigma_k} \right\} \epsilon. \tag{3.3}$$

Proof. The definitions (2.13) and (2.12) imply that, for all k,

$$0 < \overline{\Delta m}_k(s_k) = \overline{\Delta \ell}_k(s_k) - \frac{\sigma_k}{2} \|s_k\|^2$$

and (3.2) follows. We also have that, using (2.16) and the fact that $\overline{\phi}_k > \epsilon/(1+\omega_k)$ if termination does not occur,

$$\overline{\Delta \ell}_k(s_k) \geq \frac{1}{4} \min \left\{ 1, \frac{\epsilon}{\sigma_k(1+\omega_k)} \right\} \frac{\epsilon}{1+\omega_k}.$$

Hence (3.3) results from (2.21) and the inequality $\omega_k \leq \kappa_\omega \leq \frac{1}{3}\alpha\eta_1 < 1$. □

Our next step is to prove that, if termination occurs, the current iterate is a first-order ϵ-approximate minimizer, as requested.

Lemma 3.3 (Inspired by [3, Lemma 3.2]) If the ARLDA algorithm terminates, then

$$\phi_k \leq \epsilon \tag{3.4}$$

and x_k is a first-order approximate necessary minimizer.

Proof. Suppose first that the ARLDA algorithm terminates at iteration k with `exit` $= 1$ in Step 1.2. From the mechanism of this step, we have that (2.24) holds and thus, for each d with $\|d\| \leq 1$

$$(\varepsilon_g + L_h\varepsilon_J)\|d\| + 2L_h\varepsilon_c \leq \varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \leq \omega_k\overline{\Delta\ell}_k(d_k).$$

As a consequence, (2.15) ensures that, for all d with $\|d\| \leq 1$,

$$|\overline{\Delta\ell}_k(d) - \Delta\ell_k(d)| \leq \omega_k\overline{\Delta\ell}_k(d_k).$$

Hence,

$$\begin{aligned} \Delta\ell_k(d) &\leq \overline{\Delta\ell}_k(d) + |\overline{\Delta\ell}_k(d) - \Delta\ell_k(d)| \\ &\leq \overline{\Delta\ell}_k(d_k) + |\overline{\Delta\ell}_k(d) - \Delta\ell_k(d)| \\ &\leq (1+\omega_k)\overline{\Delta\ell}_k(d_k). \end{aligned} \tag{3.5}$$

where we have used that $\overline{\Delta\ell}_k(d) \leq \overline{\Delta\ell}_k(d_k)$ by definition of d_k to derive the second inequality. As a consequence, for all d with $\|d\| \leq 1$,

$$\max\left\{0, \Delta\ell_k(d)\right\} \leq (1+\omega_k)\overline{\Delta\ell}_k(d_k) = (1+\omega_k)\overline{\phi}_k \leq \epsilon,$$

where we have used the definition of $\overline{\phi}_k$ to obtain the last inequality. The conclusion (3.4) then follows from (2.3).

Suppose now that the ARLDA algorithm terminates with `exit` $= 2$ (in Step 1.2). We then obtain, using the first two inequalities of (3.5), (2.25) and (2.15), that, for every d such that $\|d\| \leq 1$,

$$\begin{aligned} \Delta\ell_k(d) &\leq \overline{\Delta\ell}_k(d_k) + |\overline{\Delta\ell}_k(d) - \Delta\ell_k(d)| \\ &\leq \tfrac{1}{2}\epsilon + (\varepsilon_g + L_h\varepsilon_J)\|d\| + 2L_h\varepsilon_c \\ &\leq \tfrac{1}{2}\epsilon + \varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \\ &\leq \tfrac{1}{2}\epsilon + \tfrac{1}{2}\epsilon = \epsilon, \end{aligned}$$

which, combined with (2.3), again implies (3.4) for this case. □

We now establish a useful property of Step 2 (Algorithm 2.3).

Lemma 3.4 Suppose that, at Step 2.2,

$$\|s_k\| \geq \theta_k \stackrel{\rm def}{=} \frac{1}{\omega_k\sigma_{\min}}\left[\varepsilon_g^{\max} + L_h\varepsilon_J^{\max} + \sqrt{(\varepsilon_g^{\max} + L_h\varepsilon_J^{\max})^2 + 4L_h\varepsilon_c^{\max}}\right]. \tag{3.6}$$

Then (2.27) is satisfied and the branch to Step 3 of the ARLDA algorithm is executed.

Proof. Step 2 of the ARLDA algorithm terminates as soon as (2.27) holds, which, in view of (3.2) is guaranteed whenever $\|s_k\|$ exceeds the largest root of

$$(\varepsilon_g + L_h\varepsilon_J)\|s_k\| + 2L_h\varepsilon_c = \tfrac{1}{2}\omega_k\sigma_k\|s_k\|^2.$$

given by

$$\frac{1}{\omega_k\sigma_k}\left[\varepsilon_g + L_h\varepsilon_J + \sqrt{(\varepsilon_g + L_h\varepsilon_J)^2 + 4L_h\varepsilon_c\omega_k\sigma_k}\right],$$

which is itself bounded above by θ_k as defined in (3.6) because of the inequality $\sigma_k \geq \sigma_{\min}$, (2.21) and the fact that $\varepsilon_f \leq \varepsilon_f^{\max}$, $\varepsilon_g \leq \varepsilon_g^{\max}$, $\varepsilon_c \leq \varepsilon_c^{\max}$ and $\varepsilon_J \leq \varepsilon_J^{\max}$. □

It is also necessary (as announced above) to prove that the accuracy loops within iteration k are finite, and thus that the ARLDA algorithm is well-defined. We therefore give explicit bounds on the maximum number of these accuracy loops and the resulting number of evaluations of the problem's inexact functions.

Lemma 3.5 Each iteration k of the ARLDA algorithm involves at most three evaluations of $\overline{f}$, $2 + \nu_k(\epsilon)$ evaluations of $\overline{c}$ and $1 + \nu_k(\epsilon)$ evaluations of $\overline{g}$ and $\overline{J}$, where $\nu_k(\epsilon)$, the number of times that the accuracy thresholds ε_g, ε_c and ε_J have been reduced by Steps 1.3 or 2.3 at iteration k, satisfies the bound

$$\nu_k(\epsilon) \stackrel{\rm def}{=} \frac{|\log\left((\varepsilon_g^{\max} + L_h\varepsilon_J^{\max})\max\{1,\theta_k\} + 2L_h\varepsilon_c^{\max}\right) - \log\left(\omega_k \min\{\tfrac{1}{2}\epsilon, \delta_k(\epsilon)\}\right)|}{|\log(\gamma_\varepsilon)|}, \quad (3.7)$$

and where $\delta_k(\epsilon)$ and θ_k are defined in (3.3) and (3.6), respectively.
As a consequence, each iteration k of the ARLDA algorithm involves at most $3n_{f,k}$ evaluations of $\widehat{f}$, $(2 + \nu_k(\epsilon))n_{c,k}$ evaluations of $\widehat{c}$, $(1 + \nu_k(\epsilon))n_{g,k}$ evaluations of $\widehat{g}$ and $(1 + \nu_k(\epsilon))n_{J,k}$ evaluations of $\widehat{J}$.

Proof. In order to prove this result, we have to bound the number of times the accuracy-improving loops (Step 1.3–Step 1) and (Step 2.3–Step 1) are being executed.

Observe first that, at the beginning of every iteration, ε_g, ε_J and ε_c are bounded above by $\varepsilon_g^{\max}$, $\varepsilon_J^{\max}$ and $\varepsilon_c^{\max}$, respectively. Morever, the mechanism of Algorithms 2.2 and 2.3 ensures that they can only be reduced within these algorithms, and that this reduction is obtained by multiplication with the constant $\gamma_\varepsilon < 1$. Thus, if i is the number of times ε_g, ε_J and ε_c have been reduced in Steps 1.3 or 2.3, then

$$\varepsilon_g \leq \gamma_\varepsilon^i\varepsilon_g^{\max}, \quad \varepsilon_J \leq \gamma_\varepsilon^i\varepsilon_J^{\max} \text{ and } \varepsilon_c \leq \gamma_\varepsilon^i\varepsilon_c^{\max}. \quad (3.8)$$

Consider the loop (Step 1.3–Step 1) and suppose that

$$\varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \leq \tfrac{1}{2}\omega_k\epsilon. \quad (3.9)$$

First consider the case where $\overline{\Delta\ell}_k(d_k) \geq \frac{1}{2}\epsilon$. Combining this last inequality with (3.9) gives that (2.24) holds and thus the loop (Step 1.3–Step 1) is terminated by either exiting the ARLDA algorithm with `exit` $= 1$ or going to its Step 2. Suppose now that (3.9) holds and that $\overline{\Delta\ell}_k(d_k) < \frac{1}{2}\epsilon$. Then (2.25) holds and the loop is terminated by exiting the ARLDA algorithm with `exit` $= 2$. Thus, using (3.8) and (3.9), the loop is not activated if i is large enough to ensure that

$$\gamma_\varepsilon^i \, (\varepsilon_g^{\max} + L_h \varepsilon_J^{\max} + 2L_h \varepsilon_c^{\max}) \leq \tfrac{1}{2}\omega_k \epsilon. \tag{3.10}$$

The situation is similar for the loop (Step 2.3–Step 1): the mechanism of Algorithm 2.3 ensures that the loop is not activated when (2.27) holds. Suppose first that $\|s_k\|$ remains below θ_k (as defined in (3.6)) for all iterations of the loop (Step 2.3–Step 1). Then, in view of (3.8) and (3.3), (2.27) must hold at the latest when

$$\gamma_\varepsilon^i \left((\varepsilon_g^{\max} + L_h \varepsilon_J^{\max})\theta_k + 2L_h \varepsilon_c^{\max} \right) \leq \omega_k \delta_k(\epsilon) \tag{3.11}$$

where $\delta_k(\epsilon)$ is defined in (3.3). If $\|s_k\|$ happens to exceed θ_k before (3.11) is satisfied, then (2.27) is also satisfied earlier because of Lemma 3.4 and the loop terminated. We therefore deduce from (3.10) and (3.11) that the loops (Step 1.3–Step 1) and (Step 2.3–Step 1) can be activated at most $\nu_k(\epsilon)$ times during the complete k-th iteration of the ARLDA algorithm, where $\nu_k(\epsilon)$ is given by (3.7).

Since they are be evaluated once in each of these loops and also (possibly) once in the beginning of Step 1, $\overline{g}(x_k, \varepsilon_g)$, and $\overline{J}(x_k, \varepsilon_j)$ are thus computed at most $1 + \nu_k(\epsilon)$ times. Evaluations of $\overline{c}$ occurs with those of $\overline{g}$, and $\overline{J}$, but also in Step 3 where $\overline{c}(x_k + s_k, \varepsilon_c)$ is evaluated. Observe that there is no need to recompute $\overline{c}(x_k, \varepsilon_c)$ in Step 3 because the necessary accuracy is ensured by Step 2, since (2.27) must hold, which in turn implies that $L_h \varepsilon_c \leq \frac{1}{2}\omega_k \overline{\Delta\ell}_k(s_k)$. Thus $\overline{c}$ is evaluated at most $2 + \nu_k(\epsilon)$ times at each iteration. Finally, $\overline{f}$ is evaluated at most three times per iteration (possibly once in Step 1 and possibly twice in Step 3). This concludes the proof of the first part of the theorem. The conclusions in terms of $\widehat{f}$, $\widehat{g}$, $\widehat{c}$ and $\widehat{J}$ follow from the definitions of $n_{f,k}$, $n_{g,k}$, $n_{c,k}$ and $n_{J,k}$, respectively. □

We next bound the error on the successive values of the objective function.

Lemma 3.6 We have that, for all $k \geq 0$,

$$|\overline{\psi}_k - \psi_k| \leq \tfrac{3}{2}\omega_k \overline{\Delta\ell}_k(s_k) \quad \text{and} \quad |\overline{\psi}_k^+ - \psi_k^+| \leq \tfrac{3}{2}\omega_k \overline{\Delta\ell}_k(s_k). \tag{3.12}$$

where $\psi_k^+ = \psi(x_k + s_k)$.

Proof. When ρ_k is computed in Step 3, it must be because Step 2 has been completed, and hence, as we noted above, (2.27) must hold, which in turn implies that $L_h \varepsilon_c \leq \frac{1}{2}\omega_k \overline{\Delta\ell}_k(s_k)$. Thus the desired inequalities follow from (2.8), (2.18), (2.17) and (2.14). □

We finally recall a standard result on successful versus unsuccessful iterations.

Lemma 3.7 [5, Theorem 2.4] Let

$$\mathcal{S}_k = \{j \in \{1,\ldots,k\} \mid \rho_j \geq \eta_1\} \quad \text{and} \quad \mathcal{U}_k = \{1,\ldots,k\} \setminus \mathcal{S}_k \tag{3.13}$$

be the sets of *successful* and *unsuccessful* iterations, respectively. The mechanism of Algorithm 2.1 guarantees that, if

$$\sigma_k \leq \sigma_{\max}, \tag{3.14}$$

for some $\sigma_{\max} > 0$, then

$$k+1 \leq |\mathcal{S}_k| \left(1 + \frac{|\log \gamma_1|}{\log \gamma_2}\right) + \frac{1}{\log \gamma_2} \log\left(\frac{\sigma_{\max}}{\sigma_0}\right). \tag{3.15}$$

This shows that it is sufficient, for establishing the overall evaluation complexity of the ARLDA algorithm, to bound the maximum number of evaluations at successful iterations.

4 Worst-case evaluation complexity

We are now in position to start our evaluation complexity proper. In order to make it formally coherent, we start by explicitly stating our assumptions on the problem.

AS.1. f and c are continuously differentiable in $\mathbb{R}^n$.

AS.2. There exist non-negative constants L_g and L_J such that, for all $k \geq 0$, and for all x, y in $\mathcal{L}_0 = \{v \in \mathbb{R}^n \mid \psi(v) \leq \psi(x_0)\}$,

$$\|g(x) - g(y)\| \leq 2L_g\|x-y\| \ \text{ and } \ \|J(x) - J(y)\| \leq 2L_J\|x-y\|. \tag{4.1}$$

AS.3 There exists a constant $L_h \geq 0$ such that (2.1) holds.

AS.4 There exists a constant $\psi_{\rm low}$ such that $\psi(x) \geq \psi_{\rm low}$ for all $x \in \mathbb{R}^n$.

We stress that we only assume the existence of L_g and L_J, but we do not assume that their value is known. Since our algorithm uses L_h, AS.3 is necessary. A first (and standard) consequence of AS.1-AS.2 is the following result on error bounds for f and c at a trial point $x+s$.

Lemma 4.1 Suppose that AS.1 and AS.2 hold. Then, for all $x, s \in \mathbb{R}^n$,

$$|f(x+s) - (f(x) + g(x)^T s)| \leq L_g\|s\|^2 \ \text{ and } \ \|c(x+s) - (c(x) + J(x)s)\| \leq L_J\|s\|^2$$

Proof. See [10, Lemma 2.1]. □

We may then use the bounds to establish the following important bound on the regularization parameter.

Lemma 4.2 Suppose that AS.1-AS.3 hold. Then there exists a constant $\sigma_{\max} \geq \max\{1, \sigma_0\}$ such that, for all $k \geq 0$,

$$\sigma_k \leq \sigma_{\max} \stackrel{\rm def}{=} \max\left\{\sigma_0, \gamma_3 \frac{4 + 2(L_g + L_h L_J)}{1 - \eta_2}, \frac{1}{\kappa_\omega}\right\} \ \text{ and } \ \omega_k \geq \frac{1}{\sigma_{\max}}. \tag{4.2}$$

Proof. Observe first that, if the algorithm has not terminated at iteration k, then (2.27) must hold, and hence (2.15) (with $v = s_k$) implies that

$$|\overline{\Delta\ell}_k(s_k) - \Delta\ell_k(s_k| \leq \omega_k \overline{\Delta\ell}_k(s_k). \tag{4.3}$$

We then have that

$$\begin{aligned}
|\rho_k - 1| &= \frac{|\overline{\psi}_k - \overline{\psi}_k^+ - \overline{\Delta\ell}_k + \Delta\ell_k(s_k) - \Delta\ell_k(s_k)|}{\overline{\Delta\ell}_k(s_k)} \\
&\leq \frac{1}{\overline{\Delta\ell}_k s_k)}\Big[|\overline{\psi}_k - \psi_k| + |\overline{\psi}_k^+ - \psi_k^+| + |\overline{\Delta\ell}_k(s_k) - \Delta\ell_k(s_k)| \\
&\qquad\qquad + |\psi_k^+ - (\psi_k - \Delta\ell_k(s_k))|\Big] \\
&\leq \frac{1}{\overline{\Delta\ell}_k(s_k)}\Big[4\omega_k \overline{\Delta\ell}_k(s_k) + |\psi_k^+ - (\psi_k - \Delta\ell_k(s_k))|\Big],
\end{aligned}$$

where we also used (2.19), the triangle inequality to derive the first inequality, while the second results from (3.12) and (4.3). Now, because of the triangle inequality, (4.1), Lemma 4.1 and (2.1), we see that

$$\begin{aligned}
|\psi_k^+ - (\psi_k - \Delta\ell_k(s_k))| &= |f_k^+ + h(c_k^+) - f_k - h(c_k) - g_k^T s_k + h(c_k) - h(c_k + J_k s_k)| \\
&\leq |f_k^+ - (f_k + g_k^T s_k)| + |h(c_k^+) - h(c_k + J_k s_k)| \\
&\leq |f_k^+ - (f_k + g_k^T s_k)| + L_h \|c_k^+ - c_k + J_k s_k\| \\
&\leq L_g \|s_k\|^2 + L_h L_J \|s_k\|^2.
\end{aligned}$$

Thus, combining the two last displays,

$$|\rho_k - 1| \leq 4\omega_k + (L_g + L_h L_J)\frac{\|s_k\|^2}{\overline{\Delta\ell}_k(s_k)}. \tag{4.4}$$

Taking now (2.21) and the inequality of (3.2) into account, we deduce that

$$|\rho_k - 1| \leq \frac{1}{\sigma_k}\Big[4 + 2(L_g + L_h L_J)\Big] \leq 1 - \eta_2 \quad \text{whenever} \quad \sigma_k \geq \frac{4 + 2(L_g + L_h L_J)}{1 - \eta_2}, \tag{4.5}$$

in which case $\rho_k \geq \eta_2 \geq \eta_1$, iteration k is successful (i.e. $k \in \mathcal{S}_k$) and $\sigma_{k+1} \leq \sigma_k$. Assume now that $k + 1$ is the first index such that

$$\sigma_{k+1} > \gamma_3 \frac{4 + 2(L_g + L_h L_J)}{1 - \eta_2}, \tag{4.6}$$

Then $\sigma_k < \sigma_{k+1}$ and the mechanism of Step 4 in the ARLDA algorithm implies that

$$\sigma_k > \frac{4 + 2(L_g + L_h L_J)}{1 - \eta_2}.$$

Thus, because of (4.5), $\rho_k \geq \eta_2$, which in turn ensures that

$$\sigma_{k+1} \leq \sigma_k < \sigma_{k+1}$$

which is impossible. Hence no k can exist such that (4.6) holds and (4.2) therefore holds for all k. The lower bound on ω_k follows from (2.21) and the fact that (4.2) ensures that $(1/\sigma_{\max}) \leq \kappa_\omega$. □

The bound (4.2) is important, in particular because it allows, in conjunction with AS.2, to simplify the bound on the complexity of a single iteration of the ARLDA algorithm, making this bound only dependent on ϵ (i.e. dropping the dependence on k).

Lemma 4.3 Suppose that AS.1-AS.3 hold. Then we have that, before termination, each iteration of the ARLDA algorithms evaluates $\overline{f}$ at most three times, $\overline{c}$ at most $2 + \nu(\epsilon)$ times, and $\overline{g}$ and $\overline{J}$ at most $1 + \nu(\epsilon)$ times, where

$$\nu(\epsilon) \stackrel{\rm def}{=} \left\lfloor \frac{|2\log\left(\epsilon\right)| + |\log\left((\varepsilon_g^{\max} + L_h\varepsilon_J^{\max})\theta + 2L_h\varepsilon_c^{\max}\right) + 2\log(4\sigma_{\max})|}{|\log(\gamma_\varepsilon)|} \right\rfloor \tag{4.7}$$

with

$$\theta \stackrel{\rm def}{=} \max\left\{1, \frac{\sigma_{\max}}{\sigma_{\min}}\left[\varepsilon_g^{\max} + L_h\varepsilon_J^{\max} + \sqrt{(\varepsilon_g^{\max} + L_h\varepsilon_J^{\max})^2 + 4L_h\varepsilon_c^{\max}}\right]\right\}. \tag{4.8}$$

As a consequence, each iteration k of the ARLDA algorithm involves at most $3n_{f,k}$ evaluations of $\widehat{f}$, $(2 + \nu(\epsilon))n_{c,k}$ evaluations of $\widehat{c}$, $(1 + \nu(\epsilon))n_{g,k}$ evaluations of $\widehat{g}$ and $(1 + \nu(\epsilon))n_{J,k}$ evaluations of $\widehat{J}$.

Proof. We observe that, because of (3.3), (4.2) and the inequalities $\epsilon \leq 1 \leq \sigma_{\max}$ and the second part of (4.2),

$$\omega_k\delta_k(\epsilon) \geq \frac{\omega_k}{16}\min\left\{1, \frac{\epsilon}{\sigma_{\max}}\right\}\epsilon \geq \frac{\epsilon^2}{16\sigma_{\max}^2} \quad \text{and} \quad \tfrac{1}{2}\omega_k\epsilon \geq \frac{\epsilon^2}{16\sigma_{\max}^2}.$$

Moreover, the second part of (4.2) and (3.6) imply that $\theta_k \leq \theta$, a value independent of k and ϵ. Using these bounds in (3.7), we see that

$$\nu_k(\epsilon) \leq \frac{\log\left(\frac{\epsilon^2}{16\sigma_{\max}^2}\right) - \log\left((\varepsilon_g^{\max} + L_h\varepsilon_J^{\max})\theta + 2L_h\varepsilon_c^{\max}\right)}{\log(\gamma_\varepsilon)}$$

which, with Lemma 3.5, the second part of (4.2) and the observation that the above value only depends on ϵ, concludes the proof of the first part of the theorem. The last conclusion again follows from the definitions of $n_{f,k}$, $n_{g,k}$, $n_{c,k}$ and $n_{J,k}$, respectively. □

Following a well-worn path in complexity analysis, we may now use a telescopic sum argument involving successive objective function's decreases at successful iterations and Lemmas 3.1, 3.7 and 4.3 to deduce our final result.

Theorem 4.4 Suppose that AS.1-AS.4 hold. Then the ARLDA algorithm terminates with $\phi_k \leq \epsilon$ in at most

$$\tau(\epsilon) \text{ iterations}, \quad 3\tau(\epsilon) \text{ evaluations of } \overline{f}, \quad (2+\nu(\epsilon))\tau(\epsilon) \text{ evaluations of } \overline{c}$$

$$\text{and } (2+\nu(\epsilon))\tau(\epsilon) \text{ evaluations of } \overline{g} \text{ and } \overline{J},$$

where

$$\tau(\epsilon) \stackrel{\rm def}{=} \left\lfloor \frac{8\sigma_{\max}\big(\psi(x_0)-\psi_{\rm low}\big)}{\eta_1(1-\alpha)}\,\epsilon^{-2}+1 \right\rfloor \left(1+\frac{|\log\gamma_1|}{\log\gamma_2}\right)+\frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right), \tag{4.9}$$

$\nu(\epsilon)$ is defined in (4.7) and $\sigma_{\max}$ is defined in (4.2).
As a consequence, the ARLDA algorithm terminates with $\phi_k \leq \epsilon$ in at most

- $3\sum_{k=0}^{\tau(\epsilon)} n_{f,k}$ evaluations of $\widehat{f}$,
- $\sum_{k=0}^{\tau(\epsilon)} (2+\nu(\epsilon))n_{c,k}$ evaluations of $\widehat{c}$,
- $\sum_{k=0}^{\tau(\epsilon)} (1+\nu(\epsilon))n_{g,k}$ evaluations of $\widehat{g}$ and
- $\sum_{k=0}^{\tau(\epsilon)} (1+\nu(\epsilon))n_{J,k}$ evaluations of $\widehat{J}$.

Proof. If iteration k is successful (i.e. $k \in \mathcal{S}_k$) and the ARLDA algorithm has not terminated yet, one has that

$$\begin{aligned}
\psi(x_k)-\psi(x_{k+1}) &\geq [\overline{\psi}_k(x_k)-\overline{\psi}_k(x_{k+1})] - 3\omega_k\overline{\Delta\ell}_k(s_k) \\
&\geq \eta_1\overline{\Delta\ell}_k(s_k) - \alpha\eta_1\overline{\Delta\ell}_k(s_k) \\
&\geq \frac{\eta_1(1-\alpha)}{2}\min\left\{1,\frac{\overline{\phi}_k}{\sigma_k}\right\}\overline{\phi}_k \\
&\geq \frac{\eta_1(1-\alpha)}{2}\min\left\{1,\frac{\epsilon}{\sigma_{\max}(1+\omega_k)}\right\}\frac{\epsilon}{1+\omega_k} \\
&= \frac{\eta_1(1-\alpha)\epsilon^2}{2\sigma_{\max}(1+\omega_k)^2},
\end{aligned}$$

where we used (3.12), (2.19), (3.1) and (4.2), the fact that $\overline{\phi}_k > \epsilon/(1+\omega_k)$ before termination, that $\sigma_{\max} \geq 1$ and the inequality $\epsilon \leq 1$. Thus $\psi(x_k)$ is monotonically decreasing, and one then deduces that

$$\psi(x_0) - \psi(x_{k+1}) \geq \frac{\eta_1(1-\alpha)\epsilon^2}{2\sigma_{\max}(1+\omega_k)^2} |\mathcal{S}_k|.$$

Using that ψ is bounded below by $\psi_{\rm low}$ and the inequalities $\omega_k \leq \kappa_\omega < 1$, we conclude that

$$|\mathcal{S}_k| \leq \frac{2\sigma_{\max}(1+\omega_k)^2}{\eta_1(1-\alpha)}(\psi(x_0) - \psi_{\rm low})\epsilon^{-2} < \frac{8\sigma_{\max}}{\eta_1(1-\alpha)}(\psi(x_0) - \psi_{\rm low})\epsilon^{-2}$$

until termination. Lemmas 3.7 and 4.2 are then invoked to compute the upper bound on the total number of iterations $\tau(\epsilon)$, and Lemma 4.3 is invoked to bound the number of evaluations. The final bounds on the number of evaluations of $\widehat{f}$, $\widehat{g}$, $\widehat{c}$ and $\widehat{J}$ follow once more from the definitions of $n_{f,k}$, $n_{g,k}$, $n_{c,k}$ and $n_{J,k}$, respectively. □

If, as is usual in evaluation complexity analysis, one focuses on the maximum number of evaluations expressed as the order in ϵ, the bound of Theorem 4.4 may be simplified, whenever $n_{f,k}$, $n_{g,k}$, $n_{c,k}$ and $n_{J,k}$ can be bounded above uniformly, to

$$O\Big(|\log(\epsilon)|\,\epsilon^{-2}\Big) \text{ evaluations,} \tag{4.10}$$

which is identical in order to the bound obtained for the inexact first-order regularization method AR1DA in [3].

5 An algorithmic variant with monotonic accuracy thresholds

As in [3], we now consider a variant of the ARLDA algorithm for which a better worst-case complexity bound can be proved, at the price of a significantly more rigid dynamic accuracy strategy. Whether or not this could result in a less efficient optimization process strongly depends on the problem-dependent cost of more accurate evaluations.

Suppose that the relatively loose conditions for updating ε_f, ε_g, ε_c and ε_J and the end of Step 5 of the ARLDA algorithm are replaced by

$$\text{If necessary, } \boxed{\text{decrease}}\ \varepsilon_f,\ \varepsilon_g,\ \varepsilon_c \text{ and } \varepsilon_J \text{ to ensure that } \varepsilon_f + L_h\varepsilon_c \leq \omega_{k+1}. \tag{5.1}$$

In this case, ε_f, ε_g, ε_c and ε_J all decrease monotonically. As a consequence, the number of times they are reduced by multiplication with γ_ϵ is still bounded by $\nu(\epsilon)$ as given in (4.7), but this bound now holds for reductions at Steps 1.3 or 2.3 *across all iterations* (instead of at iteration k only). We may therefore revise Theorem (4.4) as follows.

Theorem 5.1 Suppose that AS.1-AS.4 hold. Then the variant of the ARLDA algorithm using the update (5.1) terminates with $\phi_k \leq \epsilon$ in at most

$$\tau(\epsilon) \text{ iterations, } 3\tau(\epsilon) \text{ evaluations of } \overline{f}, \quad (\nu(\epsilon)+2\tau(\epsilon)) \text{ evaluations of } \overline{c}$$

$$\text{and } (\nu(\epsilon)+\tau(\epsilon)) \text{ evaluations of } \overline{g} \text{ and } \overline{J},$$

where $\tau(\epsilon)$ is defined in (4.9) and $\nu(\epsilon)$ is defined in (4.7).
As a consequence, the variant of ARLDA algorithm terminates with $\phi_k \leq \epsilon$ in at most

- $3\sum_{k=0}^{\tau(\epsilon)} n_{f,k}$ evaluations of $\widehat{f}$,
- $\nu(\epsilon) \max_{k\in\{0,\ldots,\tau(\epsilon)\}} n_{c,k} + 2\sum_{k=0}^{\tau(\epsilon)} n_{c,k}$ evaluations of $\widehat{c}$,
- $\nu(\epsilon) \max_{k\in\{0,\ldots,\tau(\epsilon)\}} n_{g,k} + \sum_{k=0}^{\tau(\epsilon)} n_{g,k}$ evaluations of $\widehat{g}$ and
- $\nu(\epsilon) \max_{k\in\{0,\ldots,\tau(\epsilon)\}} n_{J,k} + \sum_{k=0}^{\tau(\epsilon)} n_{J,k}$ evaluations of $\widehat{J}$.

Proof. The proof is identical to that of Theorem 4.4 except for the very last argument, where one now needs to take the revised interpretation of $\nu(\epsilon)$ into account to derive the maximum number of approximate evaluations of g, c and J. □

Observe that expressing this new bound in order of ϵ now gives

$$O\Big(|\log(\epsilon)| + \epsilon^{-2}\Big) \text{ evaluations,}$$

which typically improves upon (4.10) and extends the bound known in the smooth case for the $p = 1$ variant of the ARpDA algorithm with monotonic accuracy [3]. But, as indicated above this improved bound comes at the price of the more restrictive updating rule (5.1). In particular this rule means that a (potentially large) number of iterations will require an accuracy on g, c and J which is tighter than what is actually needed for the algorithm's progress.

6 Discussion

The theory presented above supposes a somewhat ideal world, where arbitrarily high accuracy may be requested for the evaluation of the problem's function values and their derivatives. In practice however, such requests are likely to be too demanding, for instance due to limitations of computer arithmetic. It may thus happen that evaluating $\overline{f}$, $\overline{c}$, $\overline{g}$ or $\overline{J}$ becomes impossible, especially if ψ is locally very nonlinear causing σ_k to increase and ω_k to decrease.

A first comment is that algorithmic precautions may be taken, in the framework of the present theory, to make this event less likely. The most obvious one is to use the ARLDA algorithm itself (instead of its variant of Section 5). Secondly, it is important to choose the final accuracy ϵ large enough to ensure that satisfying

$$\varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c \approx \tfrac{1}{2}\epsilon \tag{6.1}$$

(the second inequality in (2.25)) is at all possible. Moreover, as one expects $\overline{\Delta\ell}(s_k)$ to be of the order of ϵ and $\|s_k\|$ to be of the order of $\sqrt{\epsilon}$ when converging, (2.27) and (2.21) suggest that the condition

$$(\varepsilon_g + L_h\varepsilon_J)\sqrt{\epsilon} + 2L_h\varepsilon_c \approx \frac{\epsilon}{\sigma_{\max}} \tag{6.2}$$

should be achievable, where $\sigma_{\max}$ is given by (4.2). Assuming the term in σ_0 does not dominate is this latter expression, the condition (6.2) becomes

$$(\varepsilon_g + L_h\varepsilon_J)\sqrt{\epsilon} + 2L_h\varepsilon_c \approx \frac{\epsilon(1-\eta_2)}{\gamma_3(3+2(L_g+L_hL_J))}. \tag{6.3}$$

Similarly, (2.17) and (2.21) indicate that

$$\varepsilon_f \approx \frac{\epsilon(1-\eta_2)}{\gamma_3(3+2(L_g+L_hL_J))} \tag{6.4}$$

should also be achievable. This discussion furthermore indicates that limiting the growth of σ_k as much as possible by choosing moderate values of γ_2 and γ_3 in (2.20) might be a good idea. A third possibility is to "balance" the accuracy requests between ε_g, ε_J and ε_c in order to satisfy (2.24) and (2.27), depending on the value of L_h. For instance, if L_h is large, one might consider choosing ε_g smaller to allow for a larger ε_c. In view of (2.27), this is even more important if $\|s_k\|$ is small (as can be expected when converging). Finally, since (2.17) and (2.27) involve $\overline{\Delta\ell}_k(s_k)$ in their right-hand side, computing the step s_k more accurately than requested by (2.16) may also be helpful.

As indicated, these stategies may still be insufficient because the high nonlinearity inherent to the problem causes σ_k to grow or because the conditions (6.1)–(6.4) are too restrictive to hold in practice. If failure to compute one of the problem's function occurs with values of σ_k barely ensuring successful iterations, we contend that this is signal that the algorithm should be stopped as it has exhausted its "descent potential" on the exact objective function. Three cases must be considered. The first is when the value of $\overline{\Delta\ell}_k(d_k)$ cannot be proved to be significant enough for its value to be interpreted as the optimality measure $\overline{\phi}_k$ (this likely to happen for quite small values of $\overline{\Delta\ell}_k(d_k)$). This implies that the link between ϕ_k and $\overline{\phi}_k$ is lost, but the proof of Lemma 3.3 nevertheless indicates that "noisy optimality" is achieved in the sense that, for all d with $\|d\| \leq 1$,

$$\Delta\ell_k(d) \leq \max\{\tfrac{1}{2}\epsilon, \overline{\Delta\ell}_k(d_k)\} + \varepsilon_g + L_h\varepsilon_J + 2L_h\varepsilon_c.$$

The second case is when (2.27) cannot be satisfied, meaning that $\overline{\Delta\ell}_k(s_k)$ cannot be made accurate enough (due to failing evaluations of $\overline{c}$, $\overline{g}$ or $\overline{J}$) to make the latter significant compared with the inaccuracy noise. Because of the form of (2.27), it is possible that backtracking along the step s_k could improve the situation, as convexity of ℓ_k leaves the possibility that $\|\beta s_k\|$ decreases faster that $\overline{\Delta\ell}_k(\beta s_k)$ for β tending to zero in $(0,1]$, thereby allowing (2.27) to hold

for some β. If this is the case, minimization can be pursued, possibly at the price of loosing the complexity guarantee of Theorem (4.4) if $\overline{\Delta\ell}_k(\beta s_k)$ is too small compared to ϵ^2. If (2.27) cannot be enforced, this means that progress based on the model cannot be guaranteed, and the algorithm should then be stopped. A similar situation occurs in the third case, where the computation of $\overline{f}(x_k, \varepsilon_f)$ or $\overline{f}(x_k + s_k, \varepsilon_f)$ fails. This then means that the decrease in the objective function value is obscured by inaccuracies and cannot be meaningfully compared to the predicted decrease. A purely deterministic algorithm, like ARLDA, must therefore abandon. But, as we have noted, it is not because the correct working of the method is no longer *guaranteed* that significant objective function decrease may not happen by chance. Attempting some re-evaluations and/or recomputations of $\overline{\Delta\ell}_k(s_k)$ may, with some luck, allow progress. It is therefore not unreasonable to consider such an effort-limited "trial-and-error" heuristic, close to random-direction search, if the algorithm stalls due to impractical accuracy requests. Obviously, this is beyond the theory we have presented.

We conclude this section by an important observation. Since the mechanism of requiring adaptive absolute errors on the inexactly computed quantities is identical to that used in [3], the probabilistic complexity analysis derived in this reference remains valid for our case. Moreover, if either f or c is computed by subsampling sums of many independent terms (as is frequent in machine learning applications), the sample size estimators presented in [3, Theorem 6.2] may also be used in our framework.

7 Conclusion and perspectives

For solving the possibly nonsmooth and nonconvex composite problem (1.1), we have proposed an adaptive regularization algorithm using inexact evaluations of the problem's functions and their first derivative, whose evaluation complexity is $O\big(|\log(\epsilon)|\,\epsilon^{-2}\big)$. This complexity bound is within a factor $|\log(\epsilon)|$ of the known optimal bound for first-order methods using exact derivatives for smooth [10] or nonsmooth composite [9] problems. It also generalizes the bound derived in [3] to the composite nonsmooth case. We have also shown that a practically more restrictive variant of the algorithm has $O\big(|\log(\epsilon)| + \epsilon^{-2}\big)$ complexity.

Our method and analysis can easily be extended to cover two other cases of potential interest. The first is when g and J are merely β-Hölder continuous rather that Lipschitz-continuous, and the second is to set-constrained problems $\min \psi(x)$ for $x \in \mathcal{F}$, where the constraints are *inexpensive* in the sense that their/evaluation/enforcement has a negligible cost compared to that of evaluating f, g, c or J. We have refrained from including the generality needed to cover these two extensions here for clarity of exposition, and we refer the reader to [10, 3] for details. We also note that, as in [10] (for instance), the Lipschitz conditions of AS.2 need only to apply on each segment of the "path of iterates" $\cup_{k\geq 0}[x_k, x_k+1]$ for our results to hold.

The authors are aware that there is considerable room for an updating strategy for ε_f, ε_g, ε_c and ε_J which is more practical than uniform multiplication by γ_ε or simple updates of the form (2.22). One expects their worst-case complexity to lie between $O\big(|\log(\epsilon)|\,\epsilon^{-2}\big)$ and $O\big(|\log(\epsilon)| + \epsilon^{-2}\big)$ depending on how much non-monotonicity is allowed. They should be considered in a (desirable) numerical study of the new methods.

Acknowledgment

The third author is grateful to the ENSEEIHT (INP, Toulouse) for providing a friendly research environment for the duration of this research project.

References

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

[2] S. Bellavia, G. Gurioli, and B. Morini. Theoretical study of an adaptive cubic regularization method with dynamic inexact Hessian information. arXiv:1808.06239, 2018.

[3] S. Bellavia, G. Gurioli, B. Morini, and Ph. L. Toint. Adaptive regularization algorithms with inexact evaluations for nonconvex optimization. *SIAM Journal on Optimization*, 29:4:2881-2915, 2020.

[4] E. Bergou, Y. Diouane, V. Kungurtsev, and C. W. Royer. A subsampling line-search method with second-order results. arXiv:1810.07211, 2018.

[5] E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint. Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Mathematical Programming, Series A*, 163(1):359–368, 2017.

[6] J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg. Convergence rate analysis of a stochastic trust region method via supermartingales. arXiv:1609.07428v3, 2018.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

[8] R. G. Carter. Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information. *SIAM Journal on Scientific and Statistical Computing*, 14(2):368–388, 1993.

[9] C. Cartis, N. I. M. Gould, and Ph. L. Toint. On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal on Optimization*, 21(4):1721–1739, 2011.

[10] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints. *SIAM Journal on Optimization*, to appear, 2020.

[11] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming, Series A*, 159(2):337–375, 2018.

[12] X. Chen, B. Jiang, T. Lin, and S. Zhang. On adaptive cubic regularization Newton's methods for convex optimization via random sampling. arXiv:1802.05426, 2018.

[13] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[14] D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

[15] D. L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

[16] J. Duchi and F. Ruan. Stochastic methods for composite and weakly convex optimization problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018.

[17] S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008.

[18] S. Gratton and Ph. L. Toint. A note on solving nonlinear optimization problems in variable precision. arXiv:1812.03467, 2018.

[19] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, USA, 1998.

[20] W. Hare, C. Sagastizábal, and M. Solodov. A proximal bundle method for nonsmooth nonconvex functions with inexact information. *Computational Optimization and Applications*, 63:1–28, 2016.

[21] D. P. Kouri, M. Heinkenscloss, D. Rizdal, and B.G. van Bloemen-Waanders. Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty. *SISC*, 36(6):A3011A3029, 2014.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] A. S. Lewis and S. J. Wright. A proximal method for composite minimization. *Mathematical Programming, Series A*, 158:501–546, 2016.

[24] L. Liu, X. Liu, C.-J. Hsieh, and D. Tao. Stochastic second-order methods for non-convex optimization with inexact Hessian and gradient. arXiv:1809.09853, 2018.

[25] Yu. Nesterov. *Introductory Lectures on Convex Optimization.* Applied Optimization. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[26] S. Reddi, S. Sra, B. Póczos, and A. Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1145–1153. Curran Associates, Inc., 2016.

[27] F. Roosta-Khorani and M. W. Mahoney. Sub-sampled Newton methods. *Mathematical Programming, Series A*, 174(1-2):293–326, 2019.

[28] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

[29] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *32nd Conference on Neural Information Processing Systems*, 2018.

[30] P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Newton-type methods for non-convex optimization under inexact Hessian information. arXiv:1708.07164v3, 2017.

[31] Y. Yuan. Conditions for convergence of trust region algorithms for nonsmooth optimization. *Mathematical Programming*, 31(2):220–228, 1985.