FLORE
Repository istituzionale dell'Università degli Studi di Firenze

Adaptive cubic regularization methods with dynamic inexact Hessian information and applications to finite-sum minimization

Questa è la Versione finale referata (Post print/Accepted manuscript) della seguente pubblicazione:

Original Citation:

Adaptive cubic regularization methods with dynamic inexact Hessian information and applications to finite-sum minimization / Stefania Bellavia, G.G.. - In: IMA JOURNAL OF NUMERICAL ANALYSIS. - ISSN 0272-4979. - STAMPA. - 41:(2021), pp. 764-799. [10.1093/imanum/drz076]

Availability:

The webpage https://hdl.handle.net/2158/1182764 of the repository was last updated on 2025-01-27T09:19:38Z

Published version:

DOI: 10.1093/imanum/drz076

Terms of use:

Open Access

La pubblicazione è resa disponibile sotto le norme e i termini della licenza di deposito, secondo quanto stabilito dalla Policy per l'accesso aperto dell'Università degli Studi di Firenze (https://www.sba.unifi.it/upload/policy-oa-2016-1.pdf)

Publisher copyright claim:

La data sopra indicata si riferisce all'ultimo aggiornamento della scheda del Repository FloRe - The above-mentioned date refers to the last update of the record in the Institutional Repository FloRe

(Article begins on next page)

11 November 2026

1

ADAPTIVE CUBIC REGULARIZATION METHODS WITH DYNAMIC INEXACT HESSIAN INFORMATION AND APPLICATIONS TO FINITE-SUM MINIMIZATION

STEFANIA BELLAVIA† GIANMARCO GURIOLI‡ AND BENEDETTA MORINI†

Abstract. We consider the Adaptive Regularization with Cubics approach for solving nonconvex optimization problems and propose a new variant based on inexact Hessian information chosen dynamically. The theoretical analysis of the proposed procedure is given. The key property of ARC framework, constituted by optimal worst-case function/derivative evaluation bounds for first- and second-order critical point, is guaranteed. Application to large-scale finite-sum minimization based on subsampled Hessian is discussed and analyzed in both a deterministic and probabilistic manner and equipped with numerical experiments on synthetic and real datasets.

Key words. Adaptive regularization with cubics; nonconvex optimization; worst-case analysis, finite-sum optimization.

1. Introduction. Numerical methods based on the Adaptive Regularization with Cubics (ARC) constitute an important class of Newton-type procedures for the solution of the unconstrained, possibly nonconvex, optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is smooth and bounded below. Successively to the seminal works [12, 13], ARC methods have become a very active area of research due to their worst-case iteration and computational complexity bounds for achieving a desired level of accuracy in first-order and second-order optimality conditions. Under reasonable assumptions on f and a suitable realization of the adaptive cubic regularization method with derivatives of f up to order 2, Cartis et al. proved that an (ϵ, ϵ_H) approximate first- and second-order critical point is found in at most $O(\max(\epsilon^{-3/2}, \epsilon_H^{-3}))$ iterations, where ϵ and ϵ_H are positive prefixed first-order and second-order optimality tolerances, respectively [7, 13, 14, 15]; this complexity result is known to be sharp and optimal with respect to steepest descent, Newton's method and Newton's method embedded into a linesearch or a trust-region strategy [11, 14].

Of particular practical interest is the ARC algorithm where exact second-derivatives of f are not required [12]. Inexact Hessian information is used and suitable approximations of the Hessian make the algorithm convenient for problems where the evaluation of second-derivatives is expensive. Clearly, the agreement between the Hessian and its approximation characterizes complexity and convergence rate behaviour of the procedure; in [12, 13] the well-known Dennis-Moré condition [21] and slightly stronger agreements are considered.

Recently, Newton-type methods with inexact Hessian information, and possibly inexact function and gradient information, have received large attention see e.g., [2, 3, 4, 5, 8, 9, 16, 19, 24, 31, 32, 34, 35, 36]. The interest in such methods is motivated by problems where the derivative information

†Dipartimento di Ingegneria Industriale, Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia, stefania.bellavia@unifi.it, benedetta.morini@unifi.it. Members of the INdAM Research Group GNCS.

‡Dipartimento di Matematica e Informatica "Ulisse Dini", Università di Firenze, viale G.B. Morgagni 67a, 50134 Firenze, Italia, gianmarco.gurioli@unifi.it. Member of the INdAM Research Group GNCS.

*Work partially supported by INdAM-GNCS under Progetti di Ricerca 2018.

about f is computationally expensive, such as large-scale optimization problems arising in machine learning and data analysis modeled as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{N}\sum_{i=1}^{N}\phi_i(x), \tag{1.2}$$

with N being a positive scalar and $\phi_i : \mathbb{R}^n \to \mathbb{R}$. Experimental studies have shown that second-order methods can be more efficient on badly-scaled or ill-conditioned problems than first-order methods even though inexact Hessian information is built via random sampling methods, see e.g., [5, 8, 19, 24, 35, 36]. In addition, these methods can take advantage of second-order information to escape from saddle points [14, 35]. ARC methods with probabilistic models have been proposed and studied in [16, 19, 24, 34, 35, 36], while a cubic regularized method incorporating variance reduction techniques has been given in [37]; much effort has been devoted to weaken the request on the level of resemblance between the Hessian and its approximation though preserving optimal complexity bounds.

This work focuses on a variant of the ARC methods for problem (1.1) with inexact Hessian information and presents a strategy for choosing the Hessian approximation dynamically. We propose a rule for fixing the desired accuracy in the Hessian approximation and incorporate it into the ARC framework; the agreement between the Hessian of f and its approximation can be loose at the beginning of the iterative process and increases progressively as the norm of stepsize drops below one and a stationary point for (1.1) is approached. The resulting ARC variant employs a potentially milder accuracy requirement on the Hessian approximation than the proposals in [19, 34], without impairing optimal complexity results. The new algorithm is analyzed theoretically and first- and second-order optimal complexity bounds are proved in a deterministic manner; in particular, we show that the complexity bounds and convergence properties of our scheme match those of the ARC methods mentioned above. Our proposal has been motivated by the pervasiveness of finite-sum minimization problems (1.2) and the significant interest in unconstrained optimization methods with inexact Hessian information. Therefore, we discuss the application of our method to this relevant class of problems and show that it is compatible with subsampled Hessian approximations adopted in literature; in this context, we give probabilistic and deterministic results as well as numerical results on a set of nonconvex binary classification problems.

The paper is organized as follows. In Section 2 we briefly review the ARC framework, then in Section 3 we introduce our variant based on a dynamic rule for building the inexact Hessian. The first-order iteration complexity bound of the resulting algorithm is studied in Section 4 along with the asymptotic behaviour of the generated sequence; complexity bounds and convergence to second-order points are analyzed in Section 5. The application of our algorithm to the finite-sum optimization problem is discussed in Section 6, while the relevant differences of our proposal from the closely related works in the literature are discussed in Section 7. Finally, in Section 8 we provide numerical results showing the effectiveness of our adaptive rule.

Notations. The Euclidean vector and matrix norm is denoted as $\|\cdot\|$. Given the scalar or vector or matrix v, and the non-negative scalar χ, we write $v = O(\chi)$ if there is a constant g such that $\|v\| \le g\chi$. Given any set $\mathcal{S}$, $|\mathcal{S}|$ denotes its cardinality.

2. The adaptive regularization framework. The ARC approach for unconstrained optimization, firstly proposed in [23, 30, 33], is based on the use of a cubic model for f and is a globally convergent second-order procedure. If f is smooth and the Hessian matrix $\nabla^2 f$ is globally Lipschitz

continuous on $\mathbb{R}^n$ with ℓ_2-norm Lipschitz constant L, i.e.,

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n, \quad \exists L > 0,$$

then the Taylor's expansion of f at $x_k \in \mathbb{R}^n$ with increment $s \in \mathbb{R}^n$ implies

$$f(x_k + s) \leq f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{L}{6}\|s\|^3 \stackrel{\text{def}}{=} m^C(x_k, s). \tag{2.1}$$

Consequently, any step s satisfying $m^C(x_k, s) < m^C(x_k, 0) = f(x_k)$ provides a reduction of f at $x_k + s$ with respect to the current value $f(x_k)$.

The ARC approach has received growing interest starting from the papers by Cartis et al. [12, 13] where it is not required the knowledge of either exact second-derivatives of f or the Lipschitz constant L. Specifically, the cubic model used at iteration k has the form

$$m(x_k, s, \sigma_k) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T B_k s + \frac{\sigma_k}{3}\|s\|^3 \stackrel{\text{def}}{=} T_2(x_k, s) + \frac{\sigma_k}{3}\|s\|^3, \tag{2.2}$$

where $B_k \in \mathbb{R}^{n \times n}$ is a symmetric approximation of $\nabla^2 f(x_k)$ and $\sigma_k > 0$ is the cubic regularization parameter chosen adaptively to ensure the overestimation property as in (2.1). The relevance of such procedure lies on its worst-case evaluation complexity for finding an ϵ-approximate first-order critical point, i.e., a point $\widehat{x}$ such that

$$\|\nabla f(\widehat{x})\| \leq \epsilon. \tag{2.3}$$

In fact, in [13] worst-case iteration complexity of order $O(\epsilon^{-3/2})$ is proved, provided that: (a) the step s_k is the global minimizer of $m(x_k, s, \sigma_k)$ over a subspace of $\mathbb{R}^n$ including $\nabla f(x_k)$, see e.g. [6, 10, 12]; (b) the actual objective decrease $f(x_k) - f(x_k + s_k)$ is a prefixed fraction of the predicted model reduction $f(x_k) - m(x_k, s_k, \sigma_k)$, i.e.,

$$\pi_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m(x_k, s_k, \sigma_k)} \geq \eta_1, \tag{2.4}$$

for some $\eta_1 \in (0, 1)$; (c) the agreement between $\nabla^2 f(x_k)$ and B_k along s_k is such that

$$\|(\nabla^2 f(x_k) - B_k)s_k\| \leq \chi\|s_k\|^2, \tag{2.5}$$

for all $k \geq 0$ and some constant $\chi > 0$.

The requirement (2.5) is stronger than the Dennis-Moré condition [21] and it is unknown whether it can be ensured theoretically [13]. Kohler and Lucchi [24] suggested to achieve (2.5) by imposing

$$\|\nabla^2 f(x_k) - B_k\| \leq \chi\|s_k\|. \tag{2.6}$$

It is evident that the agreement between $\nabla^2 f(x_k)$ and B_k depends on the steplength which can be determined only after B_k is formed. This issue is circumvented in practice employing the steplength at the previous iteration [24, §5].

Xu et al. [34, 36] analyzed ARC algorithm making a major modification on the level of resemblance between $\nabla^2 f(x_k)$ and B_k over (2.5) requiring

$$\|(\nabla^2 f(x_k) - B_k)s_k\| \leq \mu\|s_k\|, \tag{2.7}$$

with $\mu \in (0,1)$. In practice, (2.7) is achieved imposing $\|\nabla^2 f(x_k) - B_k\| \leq \mu$. In order to retain the optimal complexity of the classical ARC method, $\mu = O(\epsilon)$ is assumed.

We observe that, given a positive υ, the requirement $\|\nabla^2 f(x_k) - B_k\| \leq \upsilon$ can be enforced approximating $\nabla^2 f(x)$ by finite differences or interpolating functions [20]. Moreover, for the class of large-scale finite-sum minimization (1.2), the requirement $\|\nabla^2 f(x_k) - B_k\| \leq \upsilon$ can be satisfied in probability via subsampling in Hessian computation, see e.g., [2, 8, 24, 34].

A main advancement in ARC algorithm was obtained by Birgin et al. in the paper [7] where ARC is generalized to higher order regularized models and significant modifications in the step computation and acceptance criterion are introduced with respect to [12, 13]. The Algorithm 2.1 detailed below is proposed in [7] and here restricted to the version based on second order model and cubic regularization; as in [7] B_k is supposed to be equal to $\nabla^2 f(x_k)$. Remarkably, global optimization of $m(x_k, s, \sigma_k)$ over a subspace of $\mathbb{R}^n$ is no longer required and conditions (2.8)–(2.9) on the step s_k are quite standard in unconstrained optimization when a model is approximately minimized. A further distinguishing feature is that the denominator in (2.10) involves the second-order Taylor expansion of f without the regularizing term, whereas the denominator in (2.4) involves the cubic model $m(x_k, s, \sigma_k)$ itself. Analogously to the algorithm in [13], Algorithm 2.1 finds an ϵ-approximation first-order critical point in at most $O(\epsilon^{-3/2})$ evaluations of f and its derivatives ∇f, $\nabla^2 f$ ([7]).

Algorithm 2.1: ARC algorithm [7]

Step 0: Initialization. Given an initial point x_0, the initial regularizer $\sigma_0 > 0$, the accuracy level ϵ. Given θ, η_1, η_2, γ_1, γ_2, γ_3, $\sigma_{\min}$ s.t.

$$\theta > 0, \quad \sigma_{\min} \in (0, \sigma_0], \quad 0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3.$$

Compute $f(x_0)$ and set $k = 0$.

Step 1: Test for termination. Evaluate $\nabla f(x_k)$. If $\|\nabla f(x_k)\| \leq \epsilon$, terminate with the approximate solution $\widehat{x} = x_k$. Otherwise, compute $B_k = \nabla^2 f(x_k)$.

Step 2: Step computation. Compute the step s_k by approximately minimizing the model $m(x_k, s, \sigma_k)$ w.r.t. s so that

$$m(x_k, s_k, \sigma_k) < m(x_k, 0, \sigma_k), \tag{2.8}$$
$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \leq \theta \|s_k\|^2. \tag{2.9}$$

Step 3: Acceptance of the trial step. Compute $f(x_k + s_k)$ and define

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{T_2(x_k, 0) - T_2(x_k, s_k)}. \tag{2.10}$$

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 parameters update. Set

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k], & \text{if } \rho_k \geq \eta_2, \\ [\sigma_k, \gamma_2 \sigma_k], & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_2 \sigma_k, \gamma_3 \sigma_k], & \text{if } \rho_k < \eta_1. \end{cases} \tag{2.11}$$

Set $k = k + 1$ and go to Step 1 if $\rho_k \geq \eta_1$, or to Step 2 otherwise.

In this work, we propose a variant of Algorithm 2.1 employing a model of the form (2.2) and a matrix B_k such that

$$\|\nabla^2 f(x_k) - B_k\| \leq C_k, \tag{2.12}$$

for all $k \geq 0$ and positive scalars C_k. The accuracy C_k on the inexact Hessian information is dynamically chosen and when the norm of the step is smaller than one it depends on the current gradient's norm. We will show that for properly chosen scalars C_k, condition (2.12) is an implementable rule to achieve (2.5). In addition, in the first phase of the procedure the accuracy imposed on B_k can be less stringent with respect to the proposal made in [19, 34, 36], though preserving the complexity bound $O(\epsilon^{-3/2})$. In the subsequent sections we present and study our variant of the ARC algorithm. We refer to Sections 6 and 7 for a discussion on the application to the finite-sum optimization problem and the comparison with the above mentioned related works in the literature.

3. An adaptive choice of the inexact Hessian. In this section, we propose and study a variant of Algorithm 2.1 which maintains the complexity bound $O(\epsilon^{-3/2})$. Our algorithm is based on the use of an approximation B_k of $\nabla^2 f(x_k)$ in the construction of the cubic model and a rule

for choosing the level of agreement between B_k and $\nabla^2 f(x_k)$. The accuracy requirements in the approximate minimization of $m(x_k, s, \sigma_k)$ consist of (2.8) and a condition on $\|\nabla_s m(x_k, s_k, \sigma_k)\|$ which includes condition (2.9) but it is not limited to it.

Our analysis is carried out under the following Assumptions on the function f and the matrix B_k used in the model (2.2).

ASSUMPTION 3.1. *The objective function f is twice continuously differentiable on $\mathbb{R}^n$ and its Hessian is Lipschitz continuous on the path of iterates with Lipschitz constant L,*

$$\|\nabla^2 f(x_k + \beta s_k) - \nabla^2 f(x_k)\| \leq L\beta\|s_k\|, \quad \forall k \geq 0, \quad \beta \in [0, 1].$$

ASSUMPTION 3.2. *For all $k \geq 0$ and some $\kappa_B \geq 0$, it holds*

$$\|B_k\| \leq \kappa_B.$$

Further, we suppose that the step s_k computed has the following properties.

ASSUMPTION 3.3. *For all $k \geq 0$ and some $0 \leq \theta_k \leq \theta$, $\theta \in [0, 1)$, s_k satisfies*

$$m(x_k, s_k, \sigma_k) < m(x_k, 0, \sigma_k), \tag{3.1}$$
$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \leq \theta_k \|\nabla f(x_k)\|. \tag{3.2}$$

By (2.1) and (2.2) it easily follows

$$m^C(x_k, s) = T_2(x_k, s) + \frac{1}{2} s^T (\nabla^2 f(x_k) - B_k)s + \frac{L}{6}\|s\|^3. \tag{3.3}$$

Then, (2.1) yields

$$f(x_k + s) \leq T_2(x_k, s) + E_k(s), \tag{3.4}$$

where

$$E_k(s) = \frac{1}{2}\|\nabla^2 f(x_k) - B_k\|\|s\|^2 + \frac{L}{6}\|s\|^3. \tag{3.5}$$

Now, we make our key requirement on the agreement between B_k and $\nabla^2 f(x_k)$ and analyze its effects on ARC algorithm.

ASSUMPTION 3.4. *Let $B_k \in \mathbb{R}^{n\times n}$ satisfy*

$$\Delta_k = \nabla^2 f(x_k) - B_k, \quad \|\Delta_k\| \leq C_k, \tag{3.6}$$
$$C_k = C, \qquad \text{if} \quad \|s_k\| \geq 1, \tag{3.7}$$
$$C_k \leq \alpha(1 - \theta)\|\nabla f(x_k)\|, \quad \text{if} \quad \|s_k\| < 1, \tag{3.8}$$

for all $k \geq 0$, with α, C_k and C positive scalars, $s_k \in \mathbb{R}^n$ and $\theta \in [0, 1)$ as in Assumption 3.3.

Bounds on $\|\Delta_k\|$ and on $E_k(s_k)$ involving $\|s_k\|$ are derived below and give $E_k(s_k) = O(\|s_k\|^3)$.

LEMMA 3.1. *Let Assumptions 3.1–3.4 hold, and let $E_k(s)$ and Δ_k as in (3.5), (3.6). Then*

$$\|\Delta_k\| \leq \begin{cases} C\|s_k\|, & \text{if } \|s_k\| \geq 1, \\ \alpha(\kappa_B + \sigma_k)\|s_k\|, & \text{if } \|s_k\| < 1, \end{cases} \tag{3.9}$$

and

$$E_k(s_k) \leq \begin{cases} \frac{1}{2}\left(C + \frac{L}{3}\right)\|s_k\|^3, & \text{if } \|s_k\| \geq 1, \\ \frac{1}{2}\left(\alpha(\kappa_B + \sigma_k) + \frac{L}{3}\right)\|s_k\|^3, & \text{if } \|s_k\| < 1. \end{cases} \tag{3.10}$$

Proof. First consider the case $\|s_k\| \geq 1$. Trivially, the inequality in (3.6) gives (3.9) and

$$E_k(s_k) \leq \frac{1}{2}C\|s_k\|^3 + \frac{L}{6}\|s_k\|^3,$$

i.e., the first bound in (3.10).

Suppose now that $\|s_k\| < 1$. Using (3.2), Assumptions 3.2 and 3.3, we obtain

$$\begin{aligned} \theta\|\nabla f(x_k)\| &\geq \|\nabla_s m(x_k, s_k, \sigma_k)\| \\ &= \|\, \nabla f(x_k) + B_k s_k + \sigma_k s_k \|s_k\| \,\| \\ &\geq \|\nabla f(x_k)\| - \|B_k\|\, \|s_k\| - \sigma_k \|s_k\|^2 \\ &\geq \|\nabla f(x_k)\| - \kappa_B \|s_k\| - \sigma_k \|s_k\|, \end{aligned} \tag{3.11}$$

which gives

$$\|s_k\| \geq \frac{(1-\theta)\|\nabla f(x_k)\|}{\kappa_B + \sigma_k}. \tag{3.12}$$

Thus, (3.6) and (3.8) yield

$$\|\Delta_k\| \leq C_k = \frac{C_k}{\|s_k\|}\|s_k\| \leq \frac{C_k(\kappa_B + \sigma_k)}{(1-\theta)\|\nabla f(x_k)\|}\|s_k\|.$$

Finally, (3.8) implies (3.9) and this along with (3.5) gives (3.10). □

Taking into account the previous result and assuming $\alpha \in [0, 2/3)$, we can establish when the overestimation property $f(x_k + s_k) \leq m(x_k, s_k, \sigma_k)$ is verified. Using (2.2), (3.3), (3.4) we see that if $E_k(s_k) \leq \sigma_k\|s_k\|^3/3$, then $m^C(x_k, s_k) \leq m(x_k, s_k, \sigma_k)$ which implies that $m(x_k, s_k, \sigma_k)$ overestimates $f(x_k + s)$.

If $\|s_k\| \geq 1$ and

$$\frac{1}{2}\left(C + \frac{L}{3}\right) \leq \frac{\sigma_k}{3}, \quad \text{i.e.,} \quad \sigma_k \geq \frac{3C + L}{2},$$

then (3.10) implies $m^C(x_k, s_k) \leq m(x_k, s_k, \sigma_k)$. Analogously, if $\|s_k\| < 1$

$$\frac{1}{2}\left(\alpha(\kappa_B + \sigma_k) + \frac{L}{3}\right) \leq \frac{\sigma_k}{3} \quad \text{i.e.,} \quad \sigma_k \geq \frac{3\alpha\kappa_B + L}{2 - 3\alpha},$$

then (3.10) implies $m^C(x_k, s_k) \leq m(x_k, s_k, \sigma_k)$.

We can now deduce an important upper bound on the regularization parameter σ_k.

LEMMA 3.2. *Let Assumptions 3.1–3.4 hold. Suppose that the scalar α in Assumption 3.4 is such that $\alpha \in \left[0, \frac{2}{3}\right)$ and that the constant η_2 in Algorithm 2.1 is such that $\eta_2 \in \left(0, \frac{2-3\alpha}{2}\right)$. Then it holds*

$$\sigma_k \leq \sigma_{\max} \stackrel{\text{def}}{=} \max\left\{\sigma_0, \gamma_3 \frac{3C+L}{2(1-\eta_2)}, \gamma_3 \frac{3\alpha\kappa_B + L}{2-3\alpha-2\eta_2}\right\} \quad \forall k \geq 0, \tag{3.13}$$

where γ_3 is the constant used in (2.11).

Proof. Let us derive conditions on σ_k ensuring $\rho_k \geq \eta_2$. By (2.2) and (3.1) it follows $\|s_k\| \neq 0$ and

$$0 < m(x_k, 0, \sigma_k) - m(x_k, s_k, \sigma_k) = T_2(x_k, 0) - T_2(x_k, s_k) - \frac{\sigma_k}{3}\|s_k\|^3. \tag{3.14}$$

Thus

$$T_2(x_k, 0) - T_2(x_k, s_k) > \frac{\sigma_k}{3}\|s_k\|^3 > 0, \tag{3.15}$$

and by (3.4) and the fact that $E_k(s_k) > 0$

$$1 - \rho_k = \frac{f(x_k + s_k) - T_2(x_k, s_k)}{T_2(x_k, 0) - T_2(x_k, s_k)} \leq \frac{E_k(s_k)}{T_2(x_k, 0) - T_2(x_k, s_k)} < \frac{3E_k(s_k)}{\sigma_k\|s_k\|^3}. \tag{3.16}$$

If $\|s_k\| \geq 1$, using (3.10) we obtain

$$1 - \rho_k < \frac{3}{2\sigma_k}\left(C + \frac{L}{3}\right),$$

and $\rho_k \geq \eta_2$ is guaranteed when

$$\sigma_k \geq \frac{3C+L}{2(1-\eta_2)}.$$

On the other hand, if $\|s_k\| < 1$ then (3.10) and (3.16) give

$$1 - \rho_k < \frac{3}{2\sigma_k}\left(\alpha(\kappa_B + \sigma_k) + \frac{L}{3}\right),$$

and $\rho_k \geq \eta_2$ is guaranteed when

$$\sigma_k \geq \frac{3\alpha\kappa_B + L}{2-3\alpha-2\eta_2},$$

noting that the denominator is strictly positive by assumption. Then, the updating rule (2.11) implies $\sigma_{k+1} \leq \sigma_k$ in case $\rho_k \geq \eta_2$ and, more generally, inequality (3.13). ∎

An important consequence of Lemma 3.1 and Lemma 3.2 is that (2.12) implies

$$\|\nabla^2 f(x_k) - B_k\| \leq \max(C, \alpha(\kappa_B + \sigma_{\max}))\|s_k\|, \tag{3.17}$$

for all $k \geq 0$, and consequently condition (2.5) is satisfied.

In Lemma 3.2, the value of α in (3.8) determines the accuracy of B_k as an approximation to $\nabla^2 f(x_k)$ and the admitted maximum value of η_2. For decreasing values of α, the accuracy of the Hessian approximation increases and η_2 reaches one. On the other hand, if α tends to $\frac{2}{3}$ then the accuracy of the Hessian approximation reduces, η_2 tends to zero and $\sigma_{\max}$ tends to infinity.*

On the base of the previous analysis we sketch our version of Algorithm 2.1 denoted as Algorithm 3.1. The main feature is the adaptive rule for adjusting the agreement between B_k and $\nabla^2 f(x_k)$ as specified in Assumption 3.4. At the beginning of kth iteration, the variable flag is equal to either 1 or 0 and determines the value of C_k; specifically $C_k = C$ if flag $= 1$, $C_k = \alpha(1-\theta)\|\nabla f(x_k)\|$ otherwise with $\nabla f(x_k)$ being available (at iteration $k = 0$, flag is set equal to 1). Scalars C and α are initialized at Step 0; the choice of α and η_2 is made in accordance to the results presented above. Then, B_k is computed at Step 2 and the trial step s_k is computed at Step 3.

Step 4 is devoted to a check on the accordance between C_k and $\|s_k\|$ since (3.8) is required to hold if $\|s_k\| < 1$ whereas $\|s_k\|$ can be determined only after B_k is formed. Therefore, at the end of a successful iteration the value of flag is fixed accordingly to the steplength of the last step. Successively, once B_k and s_k have been computed, if $\|s_k\| < 1$, flag $= 1$ and $C > \alpha(1-\theta)\|\nabla f(x_k)\|$ hold, then the step is rejected and the iteration is *unsuccessful*; variable flag is set equal to 0 and B_k is recomputed at the successive iteration. This unsuccessful iteration is ascribed to the choice of matrix B_k, hence the regularization parameter is left unchanged. On the other hand, if the level of accuracy in matrix B_k with respect to $\nabla^2 f(x_k)$ fulfills the requests (3.7)–(3.8), in Step 5 we proceed for acceptance of the trial steps and update of the regularizing parameter as in Algorithm 2.1. Summarizing, by construction, *Assumption 3.4 is satisfied at every successful iteration and at any unsuccessful iteration detected in Step 5.*

*Values $\eta_2 = \frac{3}{4}$ and $\eta_2 = \frac{9}{10}$ used in the literature for the trust-region and ARC frameworks are achieved setting $\alpha = \frac{1}{6}$ and $\alpha = \frac{1}{15}$, respectively.

Algorithm 3.1: ARC algorithm with dynamic Hessian accuracy

Step 0: Initialization. Given an initial point x_0, the initial regularizer $\sigma_0 > 0$, the accuracy level ϵ. Given θ, α, η_1, η_2, γ_1, γ_2, γ_3, $\sigma_{\min}$, C s.t.

$$0 < \theta < 1,\ \alpha \in \left[0, \frac{2}{3}\right),\ \sigma_{\min} \in (0, \sigma_0],\ 0 < \eta_1 \leq \eta_2 < \frac{2-3\alpha}{2},\ 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3,\ C > 0$$

Compute $f(x_0)$ and set $k = 0$, $C_0 = C$, flag $= 1$.

Step 1: Test for termination. If $\|\nabla f(x_k)\| \leq \epsilon$, terminate with the current solution $\widehat{x} = x_k$.
Step 2: Hessian approximation. Compute B_k satisfying (3.6).
Step 3: Step computation. Choose $\theta_k \leq \theta$. Compute the step s_k satisfying (3.1) and (3.2).
Step 4: Check on $\|s_k\|$.
If $\|s_k\| < 1$ and flag $= 1$ and $C > \alpha(1-\theta)\|\nabla f(x_k)\|$
set $x_{k+1} = x_k$, $\sigma_{k+1} = \sigma_k$, (*unsuccessful iteration*)
set $C_{k+1} = \alpha(1-\theta)\|\nabla f(x_k)\|$, flag $= 0$,
set $k = k+1$ and go to Step 2.
Step 5: Acceptance of the trial step and parameters update.
Compute $f(x_k + s_k)$ and ρ_k in (2.10). If $\rho_k \geq \eta_1$
define $x_{k+1} = x_k + s_k$, set

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k], & \text{if } \rho_k \geq \eta_2, \quad \textit{(very successful iteration)} \\ [\sigma_k, \gamma_2\sigma_k], & \text{if } \rho_k \in [\eta_1, \eta_2), \quad \textit{(successful iteration)} \end{cases}$$

If $\|s_k\| \geq 1$ set $C_{k+1} = C$, flag $= 1$.
Otherwise set $C_{k+1} = \alpha(1-\theta)\|\nabla f(x_{k+1})\|$, flag $= 0$.
Set $k = k+1$ and go to Step 1.
else
define $x_{k+1} = x_k$, $\sigma_{k+1} \in [\gamma_2\sigma_k, \gamma_3\sigma_k]$, (*unsuccessful iteration*)
$C_{k+1} = C_k$, $B_{k+1} = B_k$,
set $k = k+1$ and go to Step 3.

Finally, both flag and C_k are updated in Step 5 as follows. If the iteration is *successful*, we update flag and C_k following (3.7)–(3.8) and using the norm of the accepted trial step; clearly, this is a prediction as the step s_{k+1} is not available at this stage and such a setting may be rejected at Step 4 of the successive iteration. If the iteration is *unsuccessful*, then we do not change either C_k or B_k.

The classification of successful and unsuccessful iterations of the Algorithm 3.1 between 0 and k can be made introducing the sets

$$\mathcal{S}_k = \{\, 0 \leq j \leq k \mid j \text{ successful in the sense of Step 5} \,\}, \tag{3.18}$$
$$\mathcal{U}_{k,1} = \{\, 0 \leq j \leq k \mid j \text{ unsuccessful in the sense of Step 5} \,\}, \tag{3.19}$$
$$\mathcal{U}_{k,2} = \{\, 0 \leq j \leq k \mid j \text{ unsuccessful in the sense of Step 4} \,\}. \tag{3.20}$$

More insight into the settings of C_k and σ_k in our algorithm, first we note that C_k satisfies

$$C_k = \alpha\omega(s_k)(1-\theta)\|\nabla f(x_k)\| + (1-\omega(s_k))C,$$

where $\omega : W \to \{\,0, 1\,\}$ denotes the characteristic function of $W = \{s_k : \|s_k\| < 1\}$. It follows that if

$$\|\nabla f(x)\| \leq \kappa_g,$$

for all x in an open convex set X containing $\{x_k\}$ and some positive κ_g, then $C_k \leq \max\{C, \alpha(1-\theta)\kappa_g\}$.

Second, we observe that the update of σ_k is not affected by unsuccessful iterations in the sense of Step 4. In fact, we have $\sigma_{k+1} = \sigma_k$ whenever an unsuccessful iteration occurs at Step 4 and the rule for adapting σ_j, $j \leq k$, has the form

$$\sigma_{j+1} \geq \gamma_1\sigma_j, \quad j \in \mathcal{S}_k, \tag{3.21}$$
$$\sigma_{j+1} \geq \gamma_2\sigma_j, \quad j \in \mathcal{U}_{k,1}, \tag{3.22}$$
$$\sigma_{j+1} = \sigma_j, \quad j \in \mathcal{U}_{k,2}. \tag{3.23}$$

As a consequence, the upper bound on the scalars σ_k established in Lemma 3.2 is still valid.

4. Complexity analysis. In this section we study the iteration complexity of Algorithm 3.1 assuming that f is bounded below, i.e., there exists f_{low} such that

$$f(x) \geq f_{low}, \ \forall x \in \mathbb{R}^n.$$

We consider two possible stopping criteria for the approximate minimization of model m_k at Step 3. Given $\theta \in (0,1)$, the first criterion has the form

$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \leq \theta \min\left(\|s_k\|^2, \|\nabla f(x_k)\|\right), \tag{4.1}$$

which amounts to (3.2) with $\theta_k = \theta \min\left(1, \frac{\|s_k\|^2}{\|\nabla f(x_k)\|}\right)$. The second criterion is considered in [12, Eqn. (3.28)] and takes the form

$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \leq \theta \min(1, \|s_k\|)\|\nabla f(x_k)\|. \tag{4.2}$$

It corresponds to the choice $\theta_k = \theta \min(1, \|s_k\|)$ in (3.2).

LEMMA 4.1. *Let Assumptions 3.1 and 3.2 hold. Suppose that* $\alpha \in \left[0, \frac{2}{3}\right)$ *and* $\eta_2 \in \left(0, \frac{2-3\alpha}{2}\right)$ *in Algorithm 3.1. Then, at iteration* $k \in \mathcal{S}_k \cup \mathcal{U}_{k,1}$

$$\|s_k\| \geq \sqrt{\zeta\|\nabla f(x_k + s_k)\|},$$

for some positive ζ*, both when* s_k *satisfies (4.1) and when* $= s_k$ *satisfies (4.2) and the norm of the Hessian is bounded above by a constant* κ_H *on the path of iterates,*

$$\|\nabla^2 f(x_k + \beta s_k)\| \leq \kappa_H, \quad \forall k \geq 0, \quad \beta \in [0,1]. \tag{4.3}$$

Proof. Taylor expansions of f and ∇f give

$$f(x_k+s) = f(x_k) + s^T\nabla f(x_k) + \frac{1}{2}s^T\nabla^2 f(x_k)s + \int_0^1 (1-\tau)s^T(\nabla^2 f(x_k+\tau s) - \nabla^2 f(x_k))s\, d\tau,$$
$$\nabla f(x_k+s_k) = \nabla f(x_k) + \int_0^1 \nabla^2 f(x_k+ts_k)s_k dt. \tag{4.4}$$

Then, noting that the assumptions of Lemma 3.2 hold at iterations $k \in \mathcal{S}_k \cup \mathcal{U}_{k,1}$, using the Lipschitz continuity of $\nabla^2 f$, (3.7), (3.9) (valid at $k \in \mathcal{S}_k \cup \mathcal{U}_{k,1}$) and (3.13), we derive

$$\begin{aligned}\|\nabla f(x_k+s_k) - \nabla_s T_2(x_k,s_k)\| &= \|\nabla f(x_k+s_k) - \nabla f(x_k) - B_k s_k\| \\ &\le \|\Delta_k s_k\| + \int_0^1 \|(\nabla^2 f(x_k+\tau s_k) - \nabla^2 f(x_k))s_k\|\, d\tau \\ &\le \|\Delta_k\|\|s_k\| + \frac{L}{2}\|s_k\|^2 \\ &\le \left(\max(C, \alpha(\kappa_B+\sigma_{\max})) + \frac{L}{2}\right)\|s_k\|^2. \end{aligned} \tag{4.5}$$

Moreover, by (3.11)

$$\begin{aligned}\nabla f(x_k+s_k) &= \nabla f(x_k+s_k) - \nabla_s T_2(x_k,s_k) + \nabla_s T_2(x_k,s_k) + \sigma_k\|s_k\|s_k - \sigma_k\|s_k\|s_k \\ &= \nabla f(x_k+s_k) - \nabla_s T_2(x_k,s_k) + \nabla_s m(x_k,s_k,\sigma_k) - \sigma_k\|s_k\|s_k. \end{aligned} \tag{4.6}$$

Now consider the case s_k satisfying (4.1). Condition (4.1) along with (4.6) and (4.5) yield

$$\|\nabla f(x_k+s_k)\| \le \left(\max(C, \alpha(\kappa_B+\sigma_{\max})) + \frac{L}{2} + \theta + \sigma_{\max}\right)\|s_k\|^2,$$

which gives the claim with $\zeta = 1/\left(\max\ (C, \alpha(\kappa_B+\sigma_{\max})) + L/2 + \theta + \sigma_{\max}\right)$.

We turn now the attention to the case s_k satisfying (4.2). Combining (4.4) and the boundness of the Hessian we have

$$\|\nabla f(x_k)\| \le \|\nabla f(x_k+s_k)\| + \kappa_H\|s_k\|,$$

and by (4.2)

$$\begin{aligned}\|\nabla_s m(x_k,s_k,\sigma_k)\| &\le \theta\min(1,\|s_k\|)\|\nabla f(x_k+s_k)\| + \theta\min(1,\|s_k\|)\kappa_H\|s_k\| \\ &\le \theta\|\nabla f(x_k+s_k)\| + \theta\kappa_H\|s_k\|^2. \end{aligned}$$

Thus, (4.5) and (4.6) give

$$(1-\theta)\|\nabla f(x_k+s_k)\| \le \left(\max(C, \alpha(\kappa_B+\sigma_{\max})) + L/2 + \theta\kappa_H + \sigma_{\max}\right)\|s_k\|^2,$$

and the claim follows with $\zeta = (1-\theta)/(\max(C, \alpha(\kappa_B+\sigma_{\max})) + L/2 + \theta\kappa_H + \sigma_{\max})$. ∎

THEOREM 4.2. *Suppose that f in (1.1) is lower bounded by f_{low} and the assumptions of Lemma 4.1 hold. Then Algorithm 3.1 requires at most*

$$\mathcal{I}_S = \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon^{3/2}} \right\rfloor, \tag{4.7}$$

successful iterations and at most

$$\mathcal{I}_T = \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon^{3/2}} \right\rfloor \left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right) + \lfloor \kappa_u (f(x_0) - f_{low}) \rfloor ,$$

iterations to produce an iterate $x_{\widehat{k}}$ satisfying (2.3), with $\kappa_s = \frac{3}{\eta_1 \sigma_{\min} \zeta^{3/2}}$ and ζ as in Lemma 4.1, and $\kappa_u = \frac{3}{\eta_1 \sigma_{\min}}$.

Proof. The mechanism of Algorithm 3.1 for updating σ_k has the form (3.21)–(3.23). An unsuccessful iteration in $\mathcal{U}_{k,2}$ does not affect the value of the regularization parameter as $\sigma_{k+1} = \sigma_k$. Moreover, the assumptions of Lemma 3.2 hold at iterations $k \in \mathcal{S}_k$. Hence, $\sigma_k \leq \sigma_{\max}$, for all $k \geq 0$, due to Lemma 3.2.

The upper bound on the cardinality $|\mathcal{S}_k|$ of $\mathcal{S}_k$ follows from [7, Theorem 2.5]. Then, by using (3.15) and Lemma 4.1, at each successful iteration before termination it holds

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\geq \eta_1 (T_2(x_k, 0) - T_2(x_k, s_k)) \\ &\geq \eta_1 \frac{\sigma_k}{3} \|s_k\|^3 \qquad (4.8) \\ &\geq \eta_1 \frac{\sigma_{\min}}{3} \zeta^{3/2} \|\nabla f(x_k + s_k)\|^{3/2} \\ &\stackrel{\text{def}}{=} \kappa_s^{-1} \|\nabla f(x_k + s_k)\|^{3/2}. \qquad (4.9) \end{aligned}$$

Consequently, before termination (2.3) it holds $f(x_k) - f(x_k + s_k) \geq \kappa_s^{-1} \epsilon^{3/2}$ which implies

$$f(x_0) - f(x_{k+1}) = \sum_{j \in \mathcal{S}_k} (f(x_j) - f(x_j + s_j)) \geq |\mathcal{S}_k| \kappa_s^{-1} \epsilon^{3/2},$$

and (4.7).

The upper bound on $|\mathcal{U}_{k,1}|$ follows from [7, Lemma 2.4]. In particular, by (3.21)–(3.23) it holds $\sigma_0 \gamma_1^{|\mathcal{S}_k|} \gamma_2^{|\mathcal{U}_{k,1}|} \leq \sigma_k$ and (3.13) implies

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

As for $|\mathcal{U}_{k,2}|$, it is less or equal than the number of successful iterations with $\|s_k\| \geq 1$. By construction, an unsuccessful iteration in $\mathcal{U}_{k,2}$ occurs at most once between two successful iterations with the first one such that flag = 1, and it can not occur between two successful iterations if flag is null at the first of such iterations. In fact, flag is reassigned only at the end of a successful iteration and can be set to one only in case of successful iteration with $\|s_k\| \geq 1$, see Step 5 of Algorithm 3.1, except for the first iteration. If the case flag = 1 and $\|s_k\| < 1$ occurs then flag is set to zero and is not further changed until the subsequent successful iteration. Moreover, as flag is initialized to one, at most one additional unsuccessful iteration in $\mathcal{U}_{k,2}$ may occur before the first successful iteration.

Noting that, by (4.8),

$$\begin{aligned} f(x_0) - f(x_{k+1}) &= \sum_{j \in \mathcal{S}_k} (f(x_j) - f(x_j + s_j)) \\ &\geq \sum_{\substack{j \in \mathcal{S}_k \\ \|s_k\| \geq 1}} (f(x_j) - f(x_j + s_j)) \\ &\geq \eta_1 \frac{\sigma_{\min}}{3} \sum_{\substack{j \in \mathcal{S}_k \\ \|s_k\| \geq 1}} \|s_k\|^3, \end{aligned}$$

we have

$$f(x_0) - f_{low} \geq \eta_1 \frac{\sigma_{\min}}{3} |\mathcal{U}_{k,2}| \stackrel{\text{def}}{=} \kappa_u^{-1} |\mathcal{U}_{k,2}|$$

Then, we obtain $|\mathcal{U}_{k,2}| \leq \lfloor \kappa_u (f(x_0) - f_{\text{low}}) \rfloor$ and the proof is concluded. □

The complexity analysis presented above implies

$$\liminf_{k \to \infty} ||\nabla f(x_k)|| = 0.$$

Further characterizations of the asymptotic behaviour of $\|\nabla f(x_k)\|$ and $\|s_k\|$ are given below where the sets $\mathcal{S}, \mathcal{U}_1, \mathcal{U}_2$ are defined as

$$\begin{aligned} \mathcal{S} &= \{k \geq 0 : \ k \ \text{successful or very successful in the sense of Step 5}\}, \\ \mathcal{U}_1 &= \{k \geq 0 : \ k \ \text{unsuccessful in the sense of Step 5}\}, \\ \mathcal{U}_2 &= \{k \geq 0 : \ k \ \text{unsuccessful in the sense of Step 4}\}. \end{aligned}$$

THEOREM 4.3. *Suppose that f in (1.1) is lower bounded by f_{low}, and that the assumptions of Theorem 4.2 hold. Then, the steps s_k and the iterates x_k generated by Algorithm 3.1 satisfy*

$$\|s_k\| \to 0, \quad \text{as } k \to \infty, \quad k \in \mathcal{S}, \tag{4.10}$$

and

$$\|\nabla f(x_k)\| \to 0, \quad \text{as } k \to \infty. \tag{4.11}$$

Moreover, unsuccessful iterations in $\mathcal{U}_2$ do not occur eventually.

Proof. The first claim is proved paralleling [12, Lemma 5.1]. In particular, by (3.14)

$$f(x_k) - f(x_{k+1}) \geq \eta_1 (T_2(x_k, 0) - T_2(x_k, s_k)) \geq \eta_1 \frac{\sigma_{\min}}{3} \|s_k\|^3, \ k \in \mathcal{S}.$$

Since f is lower bounded by f_{low}, one has

$$f(x_0) - f_{low} \geq f(x_0) - f(x_{k+1}) = \sum_{j=0,\, j \in S}^{k} (f(x_j) - f(x_{j+1})) \geq \eta_1 \frac{\sigma_{\min}}{3} \sum_{j=0,\, j \in \mathcal{S}}^{k} \|s_j\|^3, \quad k \geq 0,$$

which implies convergence of the series $\sum_{k=0,\ k\in\mathcal{S}}^{\infty} \|s_k\|^3$ and the first claim as a consequence.

As for $\|\nabla f(x_k)\|$, Lemma 4.1 provides

$$\zeta\|\nabla f(x_{k+1})\| \leq \|s_k\|^2 \to 0, \quad \text{as } k \to \infty,\ k \in \mathcal{S}.$$

This fact along with $\nabla f(x_{k+1}) = \nabla f(x_k)$ at unsuccessful iterations provides the convergence of $\{\|\nabla f(x_k)\|\}$ to zero.

Finally, the behaviour of $\{\|s_k\|\}_{k\in\mathcal{S}}$ implies that eventually all successful iterations are such that $\|s_k\| < 1$. Thus, the mechanism of Algorithm 3.1 gives flag $= 0$ for all k sufficiently large and unsuccessful iterations in the sense of Step 4 can not occur. ∎

5. Convergence to second order critical point. In this section we focus on the convergence of the sequence generated by our procedure to second-order critical points x^*:

$$\nabla f(x^*) = 0 \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(x^*)) \geq 0.$$

First, we analyze the asymptotic behaviour of $\{x_k\}$ in the case where the approximate Hessian B_k becomes positive definite along a converging subsequence of $\{x_k\}$. In such a context, we show q-quadratic convergence of $\{x_k\}$ under an additional mild requirement on the step, namely the Cauchy condition. Second, we consider the case where the model B_k is not convex and obtain a second order complexity bound in accordance with the study of Cartis et al. [14].

THEOREM 5.1. *Suppose that f in (1.1) is lower bounded by f_{low}, and that the assumptions of Theorem 4.2 hold. Suppose that $\{x_{k_i}\}$ is a subsequence of successful iterates converging to some x^* and that B_{k_i} are positive definite whenever x_{k_i} is sufficiently close to x^*. Then*

i) $x_k \to x^*$ as $k \to \infty$ and x^* is second-order critical.

ii) If s_k satisfies

$$m(x_k, s_k, \sigma_k) \leq m(x_k, s_k^C, \sigma_k),\ \forall k \geq 0, \tag{5.1}$$

where s_k^C is the Cauchy step, i.e.

$$s_k^C = -\alpha_k^C \nabla f(x_k) \quad \text{and} \quad \alpha_k^C = \operatorname*{argmin}_{\alpha\geq 0} m_k(x_k, -\alpha\nabla f(x_k), \sigma_k),$$

then all the iterations are eventually successful and $x_k \to x^*$ q-quadratically.

Proof. i) From (3.17) and (4.10), it follows

$$\|\nabla^2 f(x_k) - B_k\| \leq \max(C, \alpha(\kappa_B + \sigma_{\max}))\|s_k\| \to 0, \quad k \to \infty,\ k \in \mathcal{S}. \tag{5.2}$$

As a consequence, standard perturbation results on the eigenvalues of symmetric matrices and the convergence of $\{x_{k_i}\}$ to x^* give that $\nabla^2 f(x^*)$ is positive definite. Thus, x^* is an isolated limit point and the claim *i)* is completed by using (4.10) and [28, Lemma 4.10].

ii) From the convergence of $\{x_k\}$ to x^*, (5.2) and the positive definiteness of $\nabla^2 f(x^*)$ it follows that

$$\lambda_{\min}(B_k) \geq \underline{\lambda} > 0, \quad \forall k \in \mathcal{S} \text{ sufficiently large.}$$

Moreover, we know that unsuccessful iterations in $\mathcal{U}_2$ do not occur eventually. Then, taking into account that B_k is not modified along the unsuccessful iterations in $\mathcal{U}_1$, we conclude that

$$\lambda_{\min}(B_k) \geq \underline{\lambda}, \quad \forall k \text{ sufficiently large.}$$

In order to show that all the iterations are eventually successful, we start using (3.2), (3.11) and obtain

$$\frac{\|s_k\|}{\|(B_k + \sigma_k\|s_k\|I)^{-1}\|} - \|\nabla f(x_k)\| \leq \theta\|\nabla f(x_k)\|.$$

Since

$$\|(B_k + \sigma_k\|s_k\|I)^{-1}\| = \frac{1}{\lambda_{\min}(B_k) + \sigma_k\|s_k\|} \leq \frac{1}{\underline{\lambda}},$$

we get

$$\|s_k\| \leq \frac{1+\theta}{\underline{\lambda}}\,\|\nabla f(x_k)\|, \quad \forall k \text{ sufficiently large,} \tag{5.3}$$

and $\|s_k\| \to 0$ due to (4.11). Moreover, by (3.14), (5.1) and [12, Lemma 2.1]

$$\begin{aligned} T_2(x_k,0) - T_2(x_k,s_k) &\geq m(x_k,0,\sigma_k) - m(x_k,s_k,\sigma_k) \\ &\geq \frac{\|\nabla f(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|\nabla f(x_k)\|}{1+\|B_k\|}, \frac{1}{2}\sqrt{\frac{\|\nabla f(x_k)\|}{\sigma_k}}\right), \end{aligned} \tag{5.4}$$

and Assumption 3.2 and Lemma 3.2 yield

$$T_2(x_k,0) - T_2(x_k,s_k) \geq \frac{\|\nabla f(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|\nabla f(x_k)\|}{1+\kappa_B}, \frac{1}{2}\sqrt{\frac{\|\nabla f(x_k)\|}{\sigma_{\max}}}\right).$$

Thus, eventually (4.11) and (5.3) give

$$T_2(x_k,0) - T_2(x_k,s_k) \geq \frac{\|\nabla f(x_k)\|^2}{6\sqrt{2}(1+\kappa_B)} \geq \frac{\underline{\lambda}^2}{6\sqrt{2}(1+\kappa_B)(1+\theta)^2}\|s_k\|^2 \stackrel{\text{def}}{=} \kappa_c\|s_k\|^2,$$

and by (3.10) and (3.4)

$$1 - \rho_k = \frac{f(x_k+s_k) - T_2(x_k,s_k)}{T_2(x_k,0) - T_2(x_k,s_k)} \leq \frac{E_k(s_k)}{\kappa_c\|s_k\|^2} < \frac{(\alpha(\kappa_B+\sigma_{\max}) + L/3)\|s_k\|^3}{2\kappa_c\|s_k\|^2},$$

i.e., $\rho_k \to 1$ and the iterations are very successful eventually.

Finally, (5.3) and Lemma 4.1 provide

$$\|\nabla f(x_{k+1})\| \leq \frac{\|s_k\|^2}{\zeta} \leq \frac{(1+\theta)^2}{\zeta\underline{\lambda}^2}\|\nabla f(x_k)\|^2, \quad \forall k \text{ sufficiently large,}$$

and the q-quadratic convergence of the sequence $\{x_k\}$ follows in a standard way by means of the Taylor's expansion. □

Dropping the assumption that B_k is positive definite, convergence to second order critical points can be studied. Following [14] where a modification of the ARC algorithm in [12] is proposed, we

equip Algorithm 3.1 with a further stopping criterion and impose an additional condition on the step. First, Algorithm 3.1 is stopped when

$$\|\nabla f(x_k)\| \leq \epsilon \quad \text{and} \quad \lambda_{\min}(B_k) \geq -\epsilon_H, \quad \epsilon,\ \epsilon_H > 0, \tag{5.5}$$

which represents the approximate counterpart of the second-order optimality conditions with the Hessian matrix approximated by B_k. The above criterion does not imply, in general, vicinity to local minima, as well as it does not guarantee the iterates to be distant from saddle points. Then, the possibility of referring to the strict-saddle property [26] may play a significant role; indeed (5.5) implies closeness to a local minimum for sufficiently small values of the tolerances ϵ and ϵ_H.

Second, the trial step s_k computed in Step 2.2 of Algorithm 3.1 is required to satisfy the following additional condition: if B_k is not positive semidefinite, then

$$m(x_k, s_k, \sigma_k) \leq m(x_k, s_k^E, \sigma_k), \tag{5.6}$$

where s_k^E is defined as

$$s_k^E = \alpha_k^E u_k \quad \text{and} \quad \alpha_k^E = \operatorname*{argmin}_{\alpha \geq 0} m_k(\alpha u_k), \tag{5.7}$$

and u_k is an approximation of the eigenvector of B_k associated with its smallest eigenvalue $\lambda_{\min}(B_k)$, in the sense that

$$\nabla f(x_k)^T u_k \leq 0 \quad \text{and} \quad u_k^T B_k u_k \leq \kappa_{\text{snc}} \lambda_{\min}(B_k) \|u_k\|^2, \tag{5.8}$$

for some constant $k_{\text{snc}} \in (0, 1]$. Note that the minimization in (5.7) is global which implies

$$\nabla f(x_k)^T s_k^E + (s_k^E)^T B_k s_k^E + \sigma_k \|s_k^E\|^3 = 0, \tag{5.9}$$

$$(s_k^E)^T B_k s_k^E + \sigma_k \|s_k^E\|^3 \geq 0. \tag{5.10}$$

We refer to the resulting algorithm as ARC Second Order critical point (ARC_SO). The termination criterion adopted here does not affect the mechanism for updating σ_k, then the upper bound $\sigma_{\max}$ on σ_k given in Lemma 3.2 is still valid.

Let $\widetilde{\mathcal{S}}_k$ denote the set of indices of successful iterations of ARC_SO whenever $\|\nabla f(x_k)\| > \epsilon$ and/or $\lambda_{\min}(B_k) < -\epsilon_H$, i.e., the indices of successful iterations before (5.5) is met. Following [14] we also let $\widetilde{\mathcal{S}}_k^{(1)}$ be the set of indices of successful iterations where $\|\nabla f(x_k)\| > \epsilon$ and $\widetilde{\mathcal{S}}_k^{(2)}$ be the set of indices of successful iterations where $\lambda_{\min}(B_k) < -\epsilon_H$. Let $\widetilde{\mathcal{U}}_{k,1}$ and $\widetilde{\mathcal{U}}_{k,2}$ denote the set of unsuccessful iterations of ARC_SO analogously to (3.19) and (3.20). Remarkably, the cardinality of both $\widetilde{\mathcal{S}}_k^{(1)}$ and $\widetilde{\mathcal{U}}_{k,2}$ is the same as in Algorithm 3.1, see Theorem 4.2, while proceeding as in Theorem 4.2 the cardinality of $\widetilde{\mathcal{U}}_{k,1}$ is bounded in terms of the number of successful iterations $\widetilde{\mathcal{S}}_k$, see also [14, Lemma 2.6]. Hence, it remains to derive the cardinality of $\widetilde{\mathcal{S}}_k^{(2)}$.

LEMMA 5.2. *Suppose that f in (1.1) is lower bounded by f_{low} and the assumptions of Theorem 4.2 hold. Suppose that s_k satisfies (5.6). Then, the number of successful iterations of Algorithm ARC_SO with $\lambda_{\min}(B_k) < -\epsilon_H$ is bounded above by*

$$\left\lfloor \kappa_e \frac{f(x_0) - f_{low}}{\epsilon_H^3} \right\rfloor,$$

where $\kappa_e = \frac{6\sigma_{\max}^2}{\eta_1 \kappa_{\rm snc}^3}$.

Proof. The proof parallels that of [14, Lemma 2.8]. We have

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\geq \eta_1 (T_2(x_k, 0) - T_2(x_k, s_k)) \\ &= \eta_1 (m(x_k, 0, \sigma_k) - m(x_k, s_k, \sigma_k) + \frac{\sigma_k}{3}\|s_k\|^3) \\ &\geq \eta_1 (m(x_k, 0, \sigma_k) - m(x_k, s_k^E, \sigma_k)) \\ &\geq \eta_1 \frac{\sigma_k}{6}\|s_k^E\|^3 \\ &\geq \eta_1 \frac{-\kappa_{\rm snc}^3 \lambda_{\min}(B_k)^3}{6\sigma_{\max}^2} \\ &\geq \eta_1 \frac{\kappa_{\rm snc}^3 \epsilon_H^3}{6\sigma_{\max}^2} \end{aligned} \tag{5.11}$$

in which we have used (3.14), (5.6), (5.9), (5.10) and (5.8). As a consequence, letting κ_e as in the statement of the theorem, before termination it holds

$$f(x_0) - f_{low} \geq f(x_0) - f(x_{k+1}) \geq \sum_{j \in \widetilde{\mathcal{S}}_k^{(2)}} (f(x_j) - f(x_j + s_j)) \geq |\widetilde{\mathcal{S}}_k^{(2)}| \kappa_e^{-1} \epsilon_H^3,$$

and the claim follows. ∎

We thus conclude that Algorithm ARC_SO produces an iterate $x_{\widehat{k}}$ satisfying (5.5) within at most

$$O\left(\max(\epsilon^{-3/2}, \epsilon_H^{-3})\right),$$

iterations, in accordance with the complexity result in [14].

6. Finite sum minimization. Large-scale instances of the finite-sum problem (1.2) can be conveniently solved by subsampled procedures where $\nabla f^2(x_k)$ is approximated by randomly sampling component functions ϕ_i [8]. The resulting approximation of $\nabla f^2(x_k)$ takes the form

$$\nabla^2 f_{\mathcal{D}_k}(x_k) = \frac{1}{|\mathcal{D}_k|} \sum_{i \in \mathcal{D}_k} \nabla^2 \phi_i(x_k), \tag{6.1}$$

with $\mathcal{D}_k \subset \{1, 2, \ldots, N\}$ and $|\mathcal{D}_k|$ being the so-called sample size.

We discuss the application of Algorithm 3.1 to problem (1.2) with

$$B_k = \nabla^2 f_{\mathcal{D}_k}(x_k), \tag{6.2}$$

giving both deterministic and probabilistic results. The application of Algorithm 3.1 to problem (1.2) with such Hessian approximation is supported by results in the literature which give the sample size required to obtain B_k satisfying condition (3.6) in probability and will be addressed below.

Let us make the following assumption on the objective function.

ASSUMPTION 6.1. *Suppose that, for any $x \in \mathbb{R}^n$, there exist non-negative upper bounds $\kappa_\phi(x)$ such that*

$$\max_{i \in \{1,\ldots,N\}} \|\nabla^2 \phi_i(x)\| \leq \kappa_\phi(x).$$

Uniform and non-uniform sampling strategies have been proposed [8, 19, 24, 34, 35]; for instance, the following Lemma provides the size of uniform sampling which probabilistically satisfies (3.6).

LEMMA 6.1. *Assume that Assumption 6.1 holds, $C_k > 0$ is given, the subsample $\mathcal{D}_k$ is chosen randomly and uniformly from $\{1, 2, \ldots, N\}$ and B_k is as in (6.2). Then, given $\bar{\delta} \in (0, 1)$,*

$$Pr(||\nabla^2 f(x_k) - B_k|| \leq C_k) \geq 1 - \bar{\delta}, \tag{6.3}$$

whenever the cardinality $|\mathcal{D}_k|$ of $\mathcal{D}_k$ satisfies

$$|\mathcal{D}_k| \geq \min\left\{N, \left\lceil \frac{4\kappa_\phi(x_k)}{C_k}\left(\frac{2\kappa_\phi(x_k)}{C_k} + \frac{1}{3}\right)\log\left(\frac{2n}{\bar{\delta}}\right)\right\rceil\right\} \tag{6.4}$$

Proof. See [2, Theorem 7.2]. □

We hereafter assume the existence of $\overline{\kappa}_\phi \geq 0$ such that

$$\sup_{x\in\mathbb{R}^n} \kappa_\phi(x) \leq \overline{\kappa}_\phi, \tag{6.5}$$

yielding $\sup_{x\in\mathbb{R}^n} \|\nabla^2 f(x)\| \leq \overline{\kappa}_\phi$ and Assumption 3.2 with $\kappa_B = \overline{\kappa}_\phi$.

We first give deterministic results, namely properties which are valid independently from Assumption 3.4 on B_k, now guaranteed with probability $1-\bar{\delta}$ by Lemma 6.1. In the following theorem the only requirement on B_k is the boundness of its norm, i.e. Assumption 3.2; concerning the trial step s_k, the Cauchy condition (5.1) is assumed.[†]

THEOREM 6.2. *Let $f \in C^2(\mathbb{R}^n)$. Suppose that f in (1.1) is lower bounded by f_{low}, Assumption 6.1, conditions (5.1) and (6.5) hold. Then,*

i) Given $\epsilon > 0$, Algorithm 3.1 takes at most $O(\epsilon^{-2})$ successful iterations to satisfy $\|\nabla f(x_k)\| < \epsilon$.

ii) $\|\nabla f(x_k)\| \to 0$, as $k \to \infty$ and therefore all the accumulation points of the sequence $\{x_k\}$, if any, are first-order stationary points.

iii) If $\{x_{k_i}\}$ is a subsequence of iterates converging to some x^* such that $\nabla f^2(x^*)$ is definite positive, then $x_k \to x^*$ as $k \to \infty$.

Proof. i). The claim follows from Lemma 3.1–3.3 and Corollary 3.4 in [13]. In fact, despite the acceptance criterion in [13] is (2.4) instead of (2.10), we can rely on the proof of [13, Lemma 3.2] thanks to (5.4) and considering that

$$f(x_k + s_k) - T_2(x_k, s_k) \leq 2\overline{\kappa}_\phi \|s_k\|^2, \quad k \geq 0.$$

ii) The sub-optimal complexity result in Item *i)* guarantees that $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$ and that the number of successful iterations is not finite. Moreover, $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$ follows by Assumption 6.1 , (6.5) and [12, Corollary 2.6].

iii) Proceeding as in Theorem 4.3 we obtain (4.10). Since $\nabla^2 f(x^*)$ in positive definite, x^* is an isolated limit point; consequently, (4.10) and Lemma [28, Lemma 4.10] yield the claim. □

Focusing on the optimal complexity result, we observe that Algorithm 3.1 requires at most $O(\epsilon^{-3/2})$ iterations to satisfy $\|\nabla f(x_k)\| \leq \epsilon$ with probability $1 - \delta$, $\delta \in (0, 1)$, provided that the

[†]This result is valid independently from the specific form of f considered in this section, provided that the norm of the Hessian of f is bounded in an open convex set containing all the sequence $\{x_k\}$ and Assumptions 3.2 holds.

sample size is chosen accordingly to (6.4) and $\bar{\delta}$ is suitable chosen. In fact, let $\mathcal{E}_i$ be the event: "the relation $\|\nabla^2 f(x_i) - B_i\| \leq C_i$ holds at iteration i, $1 \leq i \leq k$", and $\mathcal{E}(k)$ be the event: "the relation $\|\nabla^2 f(x_i) - B_i\| \leq C_i$ holds for the entire k iterations". If the events $\mathcal{E}_i$ are independent, then due to (6.3)

$$Pr(\mathcal{E}(k)) \equiv Pr\left(\bigcap_{i=1}^{k} \mathcal{E}_i\right) = (1-\bar{\delta})^k.$$

Thus, requiring that the event $\mathcal{E}(k)$ occurs with probability $1-\delta$, we obtain

$$Pr(\mathcal{E}(k)) = (1-\bar{\delta})^k = 1-\delta, \quad \text{i.e.,} \quad \bar{\delta} = 1 - \sqrt[k]{1-\delta} = O\left(\frac{\delta}{k}\right).$$

Taking into account the iteration complexity, we set $k = O\big(\epsilon^{-3/2}\big)$ and deduce the following choice of $\bar{\delta}$:

$$\bar{\delta} = O(\delta\epsilon^{3/2}). \tag{6.6}$$

Summarizing, choosing, at each iteration, $\bar{\delta}$ according to (6.6) and the sample size according to (6.4), the complexity result in Theorem 4.2 holds with probability of success $1-\delta$. We underline that the resulting per-iteration failure probability $\bar{\delta}$ is not too demanding in what concerns the sample size, because it influences only the logarithmic factor in (6.4), see [34].

Observe that (6.4) and $C_k = \alpha(1-\theta)\|\nabla f(x_k)\|$ yield $|\mathcal{D}_k| = O(\|\nabla f(x_k)\|^{-2})$ as long as N is large enough so that full sample size is not reached. Hence, in the general case, $|\mathcal{D}_k|$ is expected to grow along the iteration and reach values of order ϵ^{-2} at termination.

In the specific case where $k \in \mathcal{S} \cup \mathcal{U}_1$, $C_k = \alpha(1-\theta)\|\nabla f(x_k)\|$ and $\lambda_{\min}(B_k) \geq \underline{\lambda}$ for some positive $\underline{\lambda}$, using (5.3) and Lemma 4.1 we obtain

$$\|\nabla f(x_k)\| \geq \frac{\underline{\lambda}}{1+\theta}\|s_k\| \geq \frac{\sqrt{\zeta}\underline{\lambda}}{1+\theta}\sqrt{\|\nabla f(x_{k+1})\|}.$$

Then $\|\nabla f(x_k)\| \geq \frac{\sqrt{\zeta}\epsilon\underline{\lambda}}{1+\theta}$, provided that the algorithm does not terminate at iteration $k+1$; consequently, $|\mathcal{D}_k|$ is expected to grow along such iterations and reach values of order ϵ^{-1} eventually. On the other hand, the sample size for Hessian approximation is expected to be small with respect to $O(\epsilon^{-1})$ when C_k is set equal to the arbitrar constant accuracy C, hence the iterations at which $C_k = C$ can be neglected within this analysis. Taking into account that $\mathcal{U}_2$ does not depend on ϵ we can claim that, with probability $1-\delta$, at most $O(\epsilon^{-5/2})$ $\nabla^2\phi_i$-evaluations are required to compute an ϵ-approximate first order point, provided that $\lambda_{\min}(B_k) \geq \underline{\lambda}$ at all iterations where $C_k = \alpha(1-\theta)\|\nabla f(x_k)\|$. This is ensured for the subclass of problems where functions ϕ_i are strongly convex. Problems of this type arise, for instance, in classification procedures. For this subclass of problems, Theorem 5.1, Item *ii*) also ensures that, for k sufficiently large, say $k \geq \bar{k}$, with probability $(1-\bar{\delta})^{k_o}$ there exists $M > 0$ such that

$$\|x_{k+1} - x^*\| \leq M\|x_k - x^*\|^2, \quad k = \bar{k}, \ldots, \bar{k} + k_o - 1,$$

where x^* the unique minimizer. Specifically, proceeding as in [32, Theorem 2] and denoting with $\mathcal{E}_i$ the event: "the relation $\|\nabla^2 f(x_i) - B_i\| \leq C_i$ holds at iteration i, $i \geq \bar{k}$", we have that the overall

success probability in consecutive k_o iterations is

$$Pr\left(\bigcap_{i=\bar{k}}^{\bar{k}+k_o-1} \mathcal{E}_i\right) = (1-\bar{\delta})^{k_o},$$

which concludes our argument.

7. Related work. Variants of ARC based on suitable approximations of the gradient and/or the Hessian of f have been discussed in a few recent lines of work reviewed in this section. Besides the algorithm in [12, 13, 14], which employs approximations for the Hessian and is suited for a generic nonconvex function f, works [19, 16, 24, 34, 36] propose variants of the algorithm given in [12] where the gradient and/or the Hessian approximations can be performed via subsampling techniques [5, 8] and are applicable to the relevant class of large-scale finite-sum minimization (1.2) arising in machine learning; probabilistic/stochastic complexity and convergence analysis is carried out.

Cartis et al. [12, 13, 14] analyze ARC framework under varying assumptions on the Hessian approximation B_k and establish optimal and sub-optimal worst-case iteration bounds for first- and second-order optimality. First-order complexity was shown to be of $O(\epsilon^{-2})$ iterations under Assumption 3.2 and, as mentioned in Section 2, of $O(\epsilon^{-3/2})$ iterations when, in addition, B_k resembles the true Hessian and condition (2.5) is satisfied.

Kohler et al. [24] propose and study a variant of ARC algorithm suited for finite-sum minimization not necessarily convex. A subsampling scheme for the gradient and the Hessian of f is applied while maintaining first-order complexity of $O(\epsilon^{-3/2})$ iterations. The sampling scheme provided guarantees that the subsampled gradient $g(x_k)$ satisfies

$$\|\nabla f(x_k) - g(x_k)\| \leq M\|s_k\|^2, \quad \forall k \geq 0,\, M > 0, \tag{7.1}$$

with prefixed probability, and thesubsampled Hessian B_k satisfies condition (2.5) with prefixed probability. As specified in Section 2, condition (2.5) is enforced via (2.6) and since the steplength can be determined only after $g(x_k)$ and B_k are formed, the steplength at the previous iteration is taken

Cartis and Scheinberg [16] analyze a probabilistic cubic regularization variant where conditions (7.1) and (2.5) are satisfied with sufficiently high probability. Enforcing such conditions in a practical setting calls for an (inner) iterative process which requires a step computation at each repetition; in the worst-case derivatives accuracy may reach order $O(\epsilon)$ at each iteration (see also [2]).

As mentioned in Section 2, Xu et al. [34] develop and study a version of ARC algorithm where a major modification on the level of resemblance between $\nabla^2 f(x_k)$ and B_k is made over (2.5). Matrix B_k is supposed to satisfy Assumption 3.2 and

$$\|(\nabla^2 f(x_k) - B_k)s_k\| \leq \mu\|s_k\|, \quad \mu \in (0,1), \tag{7.2}$$

and the latter condition can be enforced building B_k such that $\|\nabla^2 f(x_k) - B_k\| \leq \mu$. Non convex finite-sum minimization is the motivating application for the proposal, and uniform and non-uniform sampling strategies are provided to construct matrices B_k satisfying $\|\nabla^2 f(x_k) - B_k\| \leq \mu$ with prefixed probability. In particular, unlike the rule in [24], the rule for choosing the sample size at iteration k does not depend on the step s_k which is not available when B_k has to be built. Worst-case iteration count of order $\epsilon^{-3/2}$ is shown when $\mu = O(\epsilon)$, while sub-optimal worst-case iteration

count of order ϵ^{-2} is achieved if $\mu = O(\sqrt{\epsilon})$. Note that the accuracy requirement on B_k is fixed along the iterations and depends on the accuracy requirement on the gradient's norm, that is on the gradient's norm at the final iteration. Then, when the Hessian of problem (1.2) is approximated via subsampling with accuracy ϵ, $O(\epsilon^{-2})$ evaluations of matrices $\nabla^2\phi_i$ are needed at each iteration, assuming N sufficiently large. Additionally, the use of approximate gradient via subsampling is addressed in [36].

Chen et al. [19] propose an ARC procedure for convex optimization via random sampling. Function f is convex and defined as finite-sum (1.2) of possibly nonconvex functions. Semidefinite positive subsampled approximations B_k satisfying $\|\nabla^2 f(x_k) - B_k\| \leq \mu_k$, $\mu_k \in (0,1)$, are built with a prefixed probability. Iteration complexity of order $O(\epsilon^{-1/3})$ is proved with respect to the fulfillment of condition $f(x_k) - f(x^*) \leq \epsilon$, x^* being the global minimum of (1.2); the scalar μ_k is updated as $\mu_{k+1} = O(\min(\mu_k, \|\nabla f(x_k)\|))$, and the model $m(x_k, s, \sigma_k)$ is minimized on a subspace of $\mathbb{R}^n$ imposing the strict condition $\|\nabla_s m(x_k, s_k, \sigma_k)\| \leq \theta \min(\|\nabla f(x_k)\|, \|\nabla f(x_k)\|^3, \|s_k\|^2)$, $\theta \in (0,1)$.

Summarizing, our proposal differs from the above works in the following respects. In [24] the upper bound in (2.6) is replaced by a bound computed using information from the previous iteration and no check on the fulfillment of (2.6) is made, while in [16] the error in Hessian approximation is dynamically reduced to fulfill (2.6); on the contrary our accuracy requirement C_k is computable and condition (2.5) is satisfied at every successful iteration and at any unsuccessful iteration detected in Step 5 without deteriorating computational complexity. Our proposal improves upon [34, 36] in the construction of B_k as the level of resemblance between $\nabla^2 f(x_k)$ and B_k is not maintained fixed along iterations but adaptively chosen, remaining less stringent than the first-order ϵ tolerance when the constant accuracy C is selected by the adaptive procedure or, otherwise, whether the current gradient's norm is sufficiently high (see, e.g., (3.6)–(3.8)); it improves upon [19] as the prescribed accuracy on B_k (and the sample size) may reduce at some iteration, the ultimate accuracy on $\|\nabla_s m(x_k, s_k, \sigma_k)\|$ is milder, and our complexity results are optimal for nonconvex problems while the analysis in [19] is limited to convex problems.

8. Numerical results. In this section we present the performance of our ARC Algorithm 3.1 and show that it can be computationally more convenient than ARC variants in the literature. Our numerical validation is based on inexact Hessians built via uniform subsampling and rule (6.4) for choosing the sample size. The results obtained indicate that suitable levels of accuracy in Hessian approximation and careful adaptations of rule (6.4) improve efficiency of existing procedures exploiting subsampled Hessians. Experiments are performed on nonconvex finite-sum problems arising within the framework of binary classification.

Given the training data $\{a_i, y_i\}_{i=1}^N$ where $a_i \in \mathbb{R}^d$ and $y_i \in \{0,1\}$ represent the i-th feature vector and label respectively, we minimize the empirical risk using a least-squares loss f with sigmoid function. The minimization problems then takes the form:

$$\min_{x\in\mathbb{R}^d} f(x) = \min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} \phi_i(x) = \min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} \left(y_i - \sigma\left(a_i^T x\right)\right)^2, \tag{8.1}$$

with the sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \qquad z \in \mathbb{R},$$

used as a model for predicting the values of the labels. The gradient and the Hessian of the

component functions $\phi_i(x)$, $i \in \{1, ..., N\}$, in (8.1) take the form:

$$\nabla\phi_i(x) = -2e^{-a_i^T x}\left(1 + e^{-a_i^T x}\right)^{-2}\left(y_i - \left(1 + e^{-a_i^T x}\right)^{-1}\right)a_i, \tag{8.2}$$

$$\nabla^2\phi_i(x) = -2e^{-a_i^T x}\left(1 + e^{-a_i^T x}\right)^{-4}\left(y_i\left(\left(e^{-a_i^T x}\right)^2 - 1\right) + 1 - 2e^{-a_i^T x}\right)a_i a_i^T. \tag{8.3}$$

Problem (8.1) can be seen as a neural network without hidden layers and zero bias and we refer to f as the training loss. Trivially it has form (1.2) with $n = d$

For each dataset, a number N_T of testing data $\{\bar{a}_i, \bar{y}_i\}_{i=1}^{N_T}$ is used to validate the computed model and the testing loss measured as $\dfrac{1}{N_T}\displaystyle\sum_{i=1}^{N_T}\left(\bar{y}_i - \sigma\left(\bar{a}_i^T x\right)\right)^2$.

Implementation issues concerning the considered procedures are introduced in Section 8.1. In Sections 8.2 -8.3 we give statistics of our runs. We test different ARC variants and rules for choosing the sample size of inexact Hessians and we perform two sets of experiments. First, in Section 8.2 we compare ARC variants with optimal complexity on a set of synthetic datasets from [5]. Algorithm 3.1 is compared with versions of ARC employing: (*i*) exact Hessians; (*ii*) inexact Hessians B_k with accuracy requirement (2.12) and $C_k = \epsilon$, $\forall k \geq 0$ [34, 35]; (*iii*) inexact Hessians B_k with accuracy requirement (2.6) implemented as suggested in [24], i.e., the unavailable information $\|s_k\|$ on the right-hand side is replaced with $\|s_{k-1}\|$, for $k > 0$. Second, in Section 8.3 we compare a suboptimal variant of our adaptive strategy with an ARC procedure where inexact Hessians are built using a fixed small sample size. These experiments are motivated by pervasiveness of prefixed small sample sizes in practical implementations. In fact, inequality (6.4) yields to full sample when high accuracy is imposed, i.e. when C_k is sufficiently small, and sample sizes $|\mathcal{D}_k|$ equal to a prefixed fraction of N are often employed in literature even though first-order complexity becomes $O(\epsilon^{-2})$ [3, 4, 5, 35].

8.1. Implementation issues. The implementation of the main phases of ARC variants is given in this section.

The cubic regularization parameter is initialized as $\sigma_0 = 10^{-1}$ and its minimum value is $\sigma_{\min} = 10^{-5}$. The parameters η_1, η_2, γ_1, γ_2, γ_3 and α are fixed as

$$\eta_1 = 0.1, \quad \eta_2 = 0.8, \quad \gamma_1 = 0.5, \quad \gamma_2 = 1.5, \quad \gamma_3 = 2, \quad \alpha = 0.1,$$

while the failure probability $\overline{\delta}$ in (6.3) is set equal to 0.2. The initial guess is the zero vector $x_0 = (0, ..., 0)^T \in \mathbb{R}^d$ in all runs.

The minimization of the cubic model in Step 3 of Algorithm 3.1 is performed by the Barzilai-Borwein gradient method [1] combined with a nonmonotone linesearch following the proposal in [6]. The major per iteration cost of such Barzilai-Borwein process is one Hessian-vector product, needed to compute the gradient of the cubic model. The threshold used in the termination criterion (3.2) is $\theta_k = 0.5$, $k \geq 0$.

As for the termination criteria for ARC methods, we imposed a maximum of 500 iterations and we declared a successful termination when one of the two following conditions is met:

$$\|\nabla f(x_k)\| \leq \epsilon, \quad |f(x_k) - f(x_{k-1})| \leq 10^{-6}|f(x_k)|, \quad \epsilon = 10^{-3}.$$

In order to measure the computational cost, as in [5] we use the number of Effective Gradient Evaluations (EGE), that is the sum of function and Hessian-vector product evaluations. This is a

pertinent measure since the major cost in the evaluation of each component function ϕ_i, $1 \leq i \leq N$, at $x \in \mathbb{R}^d$ consists in the computation of the scalar product $a_i^T x$. Once evaluated, this scalar product can be reused for obtaining $\nabla\phi_i(x)$, while the computation of $\nabla^2\phi_i(x)$ times a vector $v \in \mathbb{R}^d$ requires the scalar product $a_i^T v$ and it is as expensive as one $\phi_i(x)$ evaluation (see (8.2)–(8.3)). Consequently, each full Hessian-vector product costs as one function or gradient evaluation. When $|\mathcal{D}_k|$ samples are used for the Hessian approximation B_k, the cost of one matrix-vector product of the form $B_k v$ is counted as $|\mathcal{D}_k|/N$ EGE.

The algorithms were implemented in Fortran language and run on an Intel Core i5, 1.8 GHz $\times$ 1 CPU, 8 GB RAM.

8.2. Synthetic datasets. The first class of databases we consider is a set of synthetic datasets from [5], firstly proposed in [29]. These datasets have been constructed so that Hessians have condition numbers of order up to 10^7 and a wide spectrum of the eigeinvalues and allow testing on moderately ill-conditioned problems. We scaled them, in order to have entries in the interval $[0, 1]$, as follows. Let $D \in \mathbb{R}^{(N+N_T)\times d}$ be the matrix containing the training and testing features of the original dataset, that is

$$e_i^T D = a_i^T \quad \text{for } i \in \{1, ..., N\}, \quad e_{N+i}^T D = \bar{a}_i^T \quad \text{for } i \in \{1, ..., N_T\},$$

and let $m_j = \min_{i\in\{1,...,N+N_T\}} D_{ij}$ and $M_j = \max_{i\in\{1,...,N+N_T\}} D_{ij}$, for $j = 1, \ldots, d$. Then, matrix D is scaled as

$$D_{ij} \stackrel{\text{def}}{=} \frac{D_{ij} - m_j}{M_j - m_j}, \quad \text{for } i \in \{1, ..., N + N_T\},\ j \in \{1, \ldots, d\}.$$

The computation of the matrix B_k accordingly to (6.4) involves the constant

$$\kappa_\phi(x_k) = \max_{i\in\{1,...,N\}} \left\{ 2e^{-a_i^T x_k} \left(1 + e^{-a_i^T x_k}\right)^{-4} \left| y_i \left(\left(e^{-a_i^T x_k}\right)^2 - 1 \right) + 1 - 2e^{-a_i^T x_k} \right| \|a_i\|^2 \right\}.$$

Since the values $a_i^T x_k$, $1 \leq i \leq N$, are available from the exact computation of $f(x_k)$, we evaluated $\kappa_\phi(x_k)$ at the (offline) extra cost of computing $\|a_i\|^2$, $1 \leq i \leq N$.

In our implementation of Algorithm 3.1 the value of C used in (3.7) whenever $\|s_k\| \geq 1$ is such that $|\mathcal{D}_0|$ computed via (6.4) with $C_0 = C$ satisfies $|\mathcal{D}_0|/N = 0.1$. We shall hereafter refer to the implementation of Algorithm 3.1 as *ARC-Dynamic*. The numerical tests in this section compare *ARC-Dynamic*, with the following variants.

- *ARC-Full*: Algorithm 3.1 employing exact Hessians;
- *ARC-Sub*: Algorithm 3.1 employing inexact Hessian B_k and accuracy $C_k = \epsilon$, for all $k \geq 0$, i.e.

$$\|\nabla^2 f(x_k) - B_k\| \leq \epsilon, \quad \forall k \geq 0, \tag{8.4}$$

  as suggested in [34, 35];
- *ARC-KL*: Algorithm 3.1 employing inexact Hessian B_k and accuracy $C_k = \chi\|s_{k-1}\|$, for all $k \geq 1$. In other words, we use the accuracy requirement (2.6) replacing, as suggested in [24], the unavailable information $\|s_k\|$ on the right-hand side of (2.6) with the norm of the step s_{k-1}, i.e.,

$$\|\nabla^2 f(x_k) - B_k\| \leq \chi\|s_{k-1}\|, \quad \forall k \geq 1. \tag{8.5}$$

To make a fair comparison with *ARC-Dynamic*, the sample size $|\mathcal{D}_0|$ is set equal to 10% of the number of samples, since the first step has not been computed yet. Moreover, χ is chosen so that the sample size $|\mathcal{D}_1|$ resulting from (6.4) with $C_1 = \chi\|s_0\|$ is 10% of the total number of samples.

The synthetic datasets are listed in Table 8.1. For each dataset, the number N of training samples, the feature dimension d and the testing size N_T are reported. We also display the 2-norm condition number *cond* of the Hessian matrix at the approximate first-order optimal point (computed with ARC method, exact Hessian and stopping tolerance $\epsilon = 10^{-3}$) and the value of the scalar C selected.

Dataset	Training N	d	Testing N_T	*cond*	C
Synthetic1	9000	100	1000	$2.5 \cdot 10^4$	1.0101
Synthetic2	9000	100	1000	$1.4 \cdot 10^5$	1.0343
Synthetic3	9000	100	1000	$4.2 \cdot 10^7$	1.0406
Synthetic4	90000	100	10000	$4.1 \cdot 10^4$	0.2982
Synthetic6	90000	100	10000	$5.0 \cdot 10^6$	0.3184

Table 8.1: Synthetic datasets. Number of training samples (N), feature dimension (d), number of testing samples (N_T), 2-norm condition number of the Hessian matrix at computed solution (*cond*), scalar C used in forming Hessian estimates (C).

In Table 8.2 we report the results on all the synthetic datasets obtained with *ARC-Dynamic* and values C as in Table 8.1. Since the selection of the subsets $\mathcal{D}_k$ is made randomly (and uniformly) at each iteration, statistics in the forthcoming tables are averaged over 20 runs. We display: the total number of iterations (n-iter), the value of EGE at termination (EGE), the worst (Save-W), best (Save-B) and mean (Save-M) percentages of savings obtained by *ARC-Dynamic* with respect to *ARC-Sub* and *ARC-KL* in terms of EGE. To give more insights, in what follows we focus on Synthetic1 and Synthetic6 as they are representative of what we have observed in our experimentation.

In Tables 8.3 and 8.4 we report statistics for these problems solved with our algorithm and constant C different from the value in Table 8.1; we refer to such runs as *ARC-Dynamic(C)*. We duplicate the results given in Table 8.2 for sake of readibility.

In Figure 8.1 we additionally show the decrease of the training loss and the testing loss versus the number of EGE and in Figure 8.2 we plot the gradient norm versus EGE. In all the Figures we consider *ARC-Dynamic*, *ARC-Dynamic(C)*, *ARC-Sub* and *ARC-KL*. A representative run is considered for each method; in Figure 8.1 we do not plot *ARC-Dynamic*(1.00) as it overlaps with *ARC-Dynamic*.

Some comments are in order:

- Condition (8.4) in *ARC-Sub* yields a too high sample size at each iteration. The adaptive strategies *ARC-Dynamic* and *ARC-KL* outperform *ARC-Sub* as in the latter algorithm the cost for computing the Hessians is not compensated by the gain in convergence rate.
- Focusing on the two adaptive strategies *ARC-Dynamic* and *ARC-KL*, Table 8.2 shows that on average the former is less expensive than the latter. Figure 8.1 shows that *ARC-KL* is fast in the first stage of the convergence history, becoming progressively slower as the norm of

Dataset	*ARC-Dynamic*		*ARC-Sub*			*ARC-KL*		
	n-iter	EGE	Save-W	Save-B	Save-M	Save-W	Save-B	Save-M
Synthetic1	17.2	103.7	38%	53%	44%	-5%	43%	20%
Synthetic2	16.7	89.5	47%	63%	55%	-33%	50%	18%
Synthetic3	17.1	94.6	46%	61%	51%	-11%	47%	20%
Synthetic4	15.6	85.3	58%	62%	60%	-21%	22%	5%
Synthetic6	15.2	67.4	60%	66%	63%	-5%	36%	16%

Table 8.2: The columns are divided in three different groups. *ARC-Dynamic*: average number of iterations (n-iter) and EGE at termination. *ARC-Sub*: worst (Save-W), best (Save-B) and mean (Save-M) percentages of saving obtained by *ARC-Dynamic* over *ARC-Sub* on the synthetic datasets. *ARC-KL*: worst (Save-W), best (Save-B) and mean (Save-M) percentages of saving obtained by *ARC-Dynamic* over *ARC-KL* on the synthetic datasets.

Method	n-iter	EGE	Save-W	Save-B	Save-M
ARC-Dynamic	17.2	103.7	38%	53%	44%
ARC-Dynamic(0.50)	15.4	145.4	9%	29%	21%
ARC-Dynamic(0.75)	16.5	112.6	27%	46%	39%
ARC-Dynamic(1.00)	16.8	104.0	36%	53%	43%
ARC-Dynamic(1.25)	18.7	115.1	26%	54%	37%

Table 8.3: Synthetic1 dataset. Average number of iterations (n-iter), EGE, and worst (Save-W), best (Save-B) and mean (Save-M) percentages of saving obtained by *ARC-Dynamic* over *ARC-Sub*.

the step starts changing significantly from an iteration to the other (see Figure 8.3). In fact, the implementation of *ARC-KL* relies on the assumption that $\|s_k\|$ is well approximated by $\|s_{k-1}\|$ and this is not always true. In particular, Figure 8.3 shows that the norm of the step changes slowly initially while in the remaining iterations it oscillates and successive values differ by some orders of magnitude. This behaviour affects the euclidean norm of the gradient as shown in Figure 8.2. We observe that such norm, depicted against EGE, oscillates in *ARC-KL*, while this is not the case in *ARC-Dynamic* and *ARC-Dynamic(C)*.

- Focusing on our proposed adaptive strategy, Figure 8.4 shows that *ARC-Dynamic* uses sets $\mathcal{D}_k$ whose cardinality varies adaptively through iterations and it is considerably smaller than N in most iterations. Moreover, the performance of *ARC-Dynamic* appears to be quite insensitive to the choice of scalar C. In fact, computational savings of *ARC-Dynamic* over *ARC-Sub* are achieved with various values of C, including those reported in Table 8.1.

The synthetic datasets used provide moderately ill-conditioned problems and motivate the use of second order methods. Indeed, second order methods show their strength since all the tested procedures manage to reduce the norm of the gradient and provide a small classification error. This is shown in Table 8.5 where the average accuracy achieved by the methods under comparison is reported. We outline that, the difference between the percentages reported in each column and

Method	n-iter	EGE	Save-W	Save-B	Save-M
ARC-Dynamic	15.2	67.4	60%	66%	63%
ARC-Dynamic(0.25)	15.1	78.9	53%	59%	57%
ARC-Dynamic(0.50)	15.9	58.5	57%	70%	68%
ARC-Dynamic(0.75)	16.6	61.5	54%	73%	66%
ARC-Dynamic(1.00)	16.8	64.1	46%	74%	65%

Table 8.4: Synthetic6 dataset. Average number of iterations (n-iter), EGE, and worst (Save-W), best (Save-B) and mean (Save-M) percentages of saving obtained by *ARC-Dynamic* over *ARC-Sub*.

Method	Synthetic1	Synthetic2	Synthetic3	Synthetic4	Synthetic6
ARC-Dynamic	98.00%	96.80%	97.10%	97.85%	97.98%
ARC-Dynamic(0.25)	—	—	—	98.09%	98.08%
ARC-Dynamic(0.50)	97.60%	96.40%	96.90%	98.19%	98.23%
ARC-Dynamic(0.75)	98.10%	96.60%	97.20%	98.02%	98.11%
ARC-Dynamic(1.00)	97.20%	96.60%	96.10%	98.15%	97.96%
ARC-Dynamic(1.25)	98.00%	96.60%	96.90%	—	—
ARC-Sub	97.50%	96.60%	97.00%	98.13%	97.87%
ARC-KL	97.80%	96.60%	96.70%	98.13%	97.98%

Table 8.5: Synthetic datasets. Binary classification rate on the testing set employed by *ARC-Dynamic*, *ARC-Dynamic(C)*, $C \in \{0.25, 0.5, 0.75, 1, 1.25\}$, *ARC-KL* and *ARC-Sub*, mean values over 20 runs.

their mean value ranges from 0.20% (best case) to $0,74\%$ (worst case), with an average of $0,38\%$. Thus, all the ARC variants reach a high accuracy in the testing phase and the preferable one is the variant requiring the lowest number of EGE at termination.

As a final comment, our experiments show that, despite ill-conditioning, an accurate approximation of the Hessian is not required and accuracy dynamically chosen along iterations works well in practice. Adaptive thresholds for the Hessian approximations yield to procedures computationally more convenient than those using constant and tiny thresholds and do not lack ability in solving the problems.

8.3. Real datasets. In this section we present our second set of numerical results, performed on the machine learning datasets using subsampled ARC variants with deterministic suboptimal complexity of $O(\epsilon^{-2})$. More in depth, we compare our adaptive strategy with the version of ARC considered in [35] where, at each iteration, the Hessian is approximated via subsampling on a set with prefixed and small cardinality.

Our adaptive choice of $|\mathcal{D}_k|$ is implemented by introducing safeguards in (6.4). Whenever $\|s_k\| < 1$ we choose the cardinality of $\mathcal{D}_k$ according the following rule:

$$|\mathcal{D}_k| = \max\left\{0.05N, \min\left\{0.1N, \left\lceil \frac{4\rho}{C_k}\left(\frac{2\rho}{C_k}+\frac{1}{3}\right)\log\left(\frac{2n}{\bar{\delta}}\right)\right\rceil\right\}\right\}, \quad k \geq 0, \tag{8.6}$$

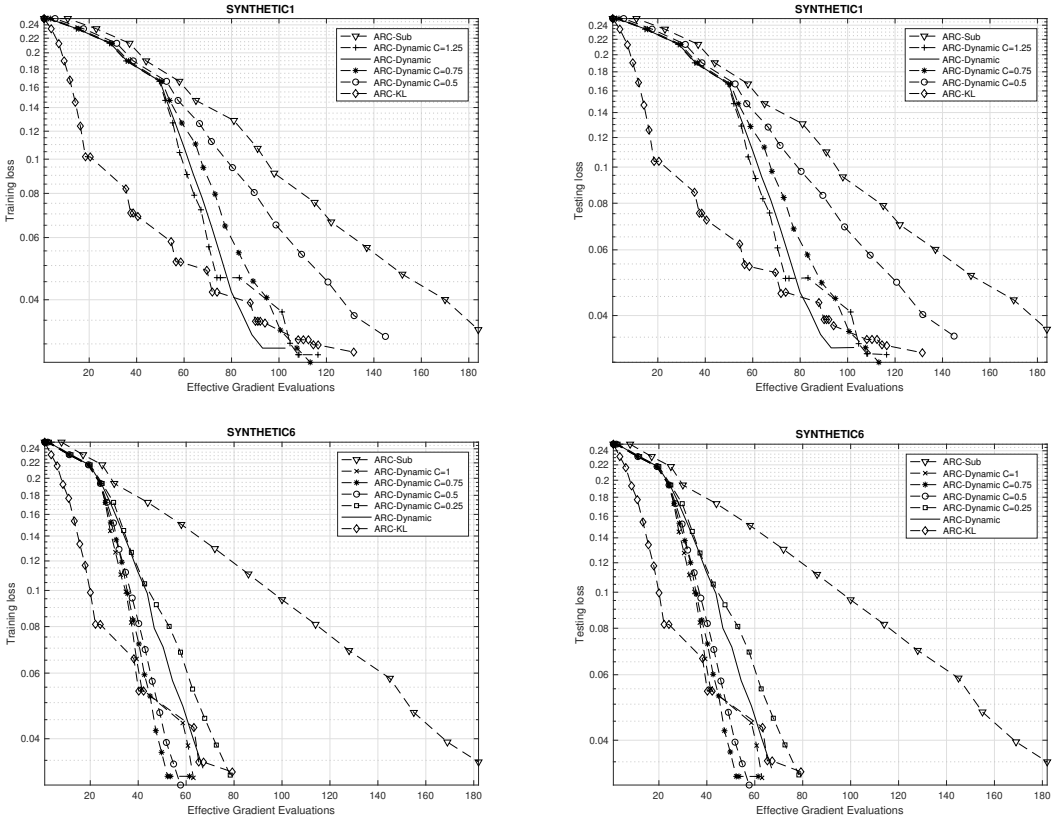

Figure 8.1: Comparison of *ARC-Dynamic*, *ARC-Dynamic* C with $C = 0.25$, $C = 0.5$, $C = 0.75$, $C = 1$, $C = 1.25$, ARC-KL and *ARC-Sub* against EGE. Each row corresponds to a different synthetic dataset. Training loss (left) and testing loss (right) against EGE, logarithmic scale is on the y axis.

with $\rho > 0$. Clearly, $|\mathcal{D}_k|/N$ varies in the range $[0.05, 0.1]$ for all $k \geq 0$, allowing us to compare our adaptive strategy with strategies employing fixed small sample sizes. The scalar ρ is chosen so that $|\mathcal{D}_k| = 0.1N$ when $C_k = \alpha(1-\theta)\epsilon^{2/3}$, i.e., the value of C_k corresponding to $\|\nabla f(x_k)\| = \epsilon^{2/3}$ and $\|s_k\| < 1$. Whenever $\|s_k\| \geq 1$, the scalar C used in (3.7) is fixed so that $|\mathcal{D}_k| = 0.05N$.

Guidelines for our rule are: sample size $0.05N$ is used when $\|s_k\| \geq 1$, larger sample size, up to $0.1N$, is used eventually. Clearly, under this rule the Hessian sample size depends on the ratio ρ/C_k.

We compare *ARC-Dynamic* with the above choice of $|\mathcal{D}_k|$ against its variant using Hessian approximations obtained by subsampling a small constant fraction of examples. We will refer to the latter algorithm as *ARC-Fix(p)* where $p \in (0, 1)$ is the prefixed constant fraction of the N examples used for building the Hessian approximations.

In Table 8.6 we list the datasets used and for sake of completeness, the value of the ratio ρ/C

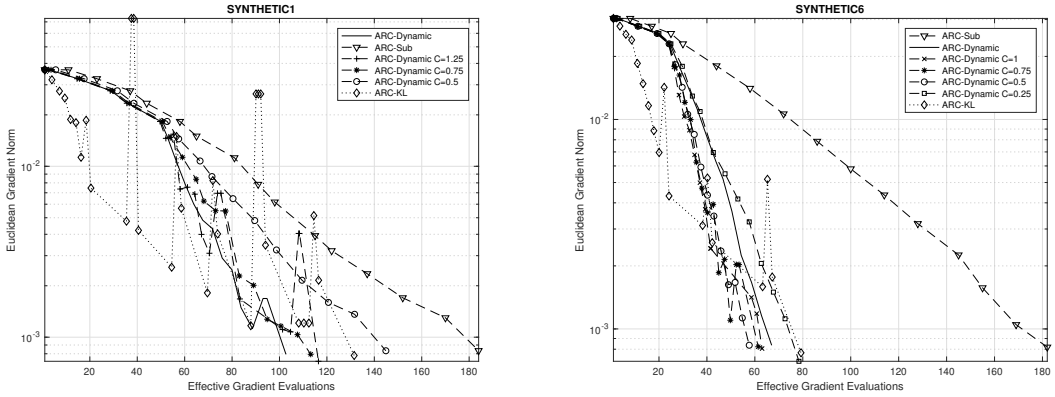

Figure 8.2: Synthetic datasets, euclidean norm of the gradient against EGE (training set), logarithmic scale is on the y axis. *ARC-Dynamic*, *ARC-Dynamic* C with $C = 0.25$, $C = 0.5$, $C = 0.75$, $C = 1$, $C = 1.25$, *ARC-KL* and *ARC-Sub*.

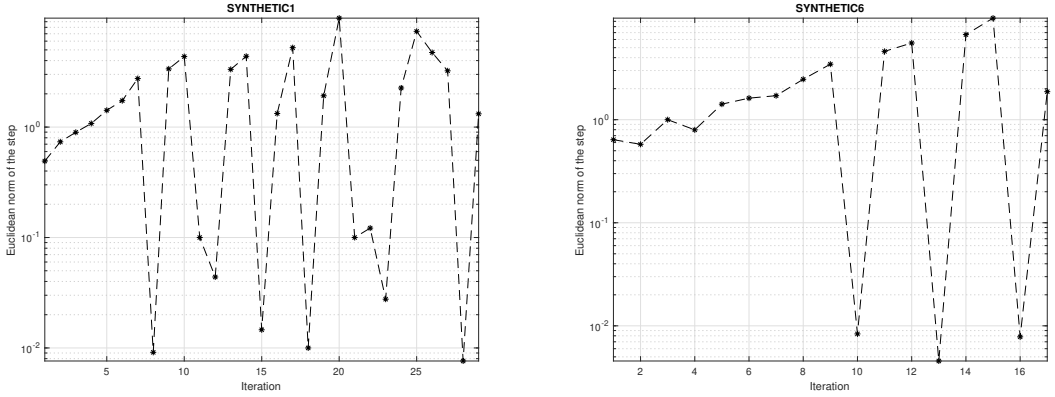

Figure 8.3: Synthetic datasets, 2-norm of the step against iterations via *ARC-KL*. Logarithmic scale on the y-axis.

determining the Hessian sample size whenever $\|s_k\| \geq 1$ is used. The MNIST dataset is here used for binary classification, labelling even digits with 1 and odd digits with 0. In the same table, in the column with header ϵ we report the used stopping tolerance. All test problems have been solved with $\epsilon = 10^{-3}$ except for Cina0 and HTRU2, where the tolerance has been increased to 10^{-2}, since for lower values of ϵ we had no longer improvements on the decrease of the training and the testing loss, regardless of the method used. By contrast, Mushroom was solved also using the tighter tolerance $\epsilon = 10^{-5}$ as below threshold $\epsilon = 10^{-3}$ further reduction in training and testing loss was observed and the percentage of failures in classification on the testing set dropped from 1% to zero. This can be observed in Table 8.7 where we report the average percentage of testing

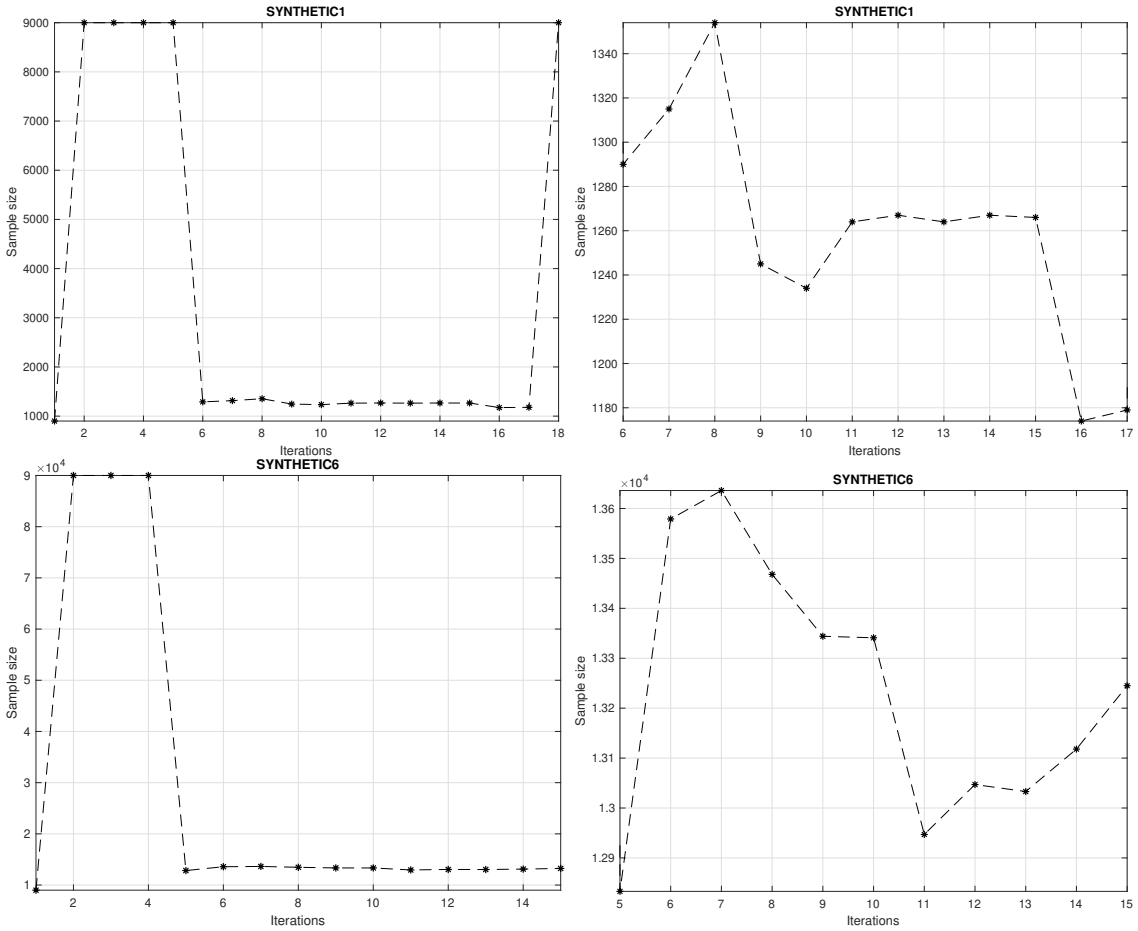

Figure 8.4: Synthetic datasets. Sample size for Hessian approximations against iterations (left). Portions of figures showing low sample sizes (right).

set data correctly classified. We also underline that, the gap between the percentages reported in each column and their mean value varies from 0% (best case) to $0,89\%$ (worst case), for an average of $0,20\%$. Therefore, the different ARC methods considered achieve a high level of accuracy in the testing phase.

In Table 8.8 we report, for each considered test problem and for each method under comparison, the average number of EGE performed on 20 runs. We compare the performance of *ARC-Dynamic* with that of *ARC-Full* and *ARC-Fix(p)*, $p \in \{0.01, 0.05, 0.1, 0.2\}$.

Focusing on the strategies employing a prefixed sample size, Table 8.8 shows that there is not a clear winner, as their performance depend on the specific dataset. However, all of them are clearly preferable to ARC with full Hessian, confirming that uniformly sampling the Hessian on a low number of example is enough and there is no point to compute the full Hessian in these applications. On the other hand, *ARC-Dynamic* always terminates with the lowest number of EGE

Dataset	Training N	d	Testing N_T	ϵ	ρ/C
Mushroom [27]	6503	112	1621	10^{-3}	2.3241
				10^{-5}	2.3241
HTRU2 [27]	10000	8	7898	10^{-2}	3.6942
Cina0 [17]	10000	132	6033	10^{-2}	2.8671
Gisette [27]	5000	5000	1000	10^{-3}	1.6182
MNIST [25]	60000	784	10000	10^{-3}	6.3841
A9A [27]	22793	123	9768	10^{-3}	4.3922
Ijcnn1 [18]	49990	22	91701	10^{-3}	7.5283
Reged0 [18]	400	999	100	10^{-3}	0.4443

Table 8.6: Real datasets. Size of the training set (Training N), problem dimension (d), size of the testing set (Testing N_T), tolerance ϵ for approximate optimality (ϵ) and the ratio ρ/C used for computing sample sizes.

Method	Mushroom		HTRU2	Cina0	Gisette	MNIST	A9A	Ijcnn1	Reged0
	$\epsilon = 10^{-3}$	$\epsilon = 10^{-5}$							
ARC-Dynamic	99.38%	100%	98.20%	91.88%	97.40%	89.92%	84.81%	91.76%	96.00%
ARC-Fix(0.01)	99.07%	100%	98.21%	91.80%	97.60%	89.84%	84.83%	91.95%	96.00%
ARC-Fix(0.05)	98.83%	100%	98.19%	91.84%	97.50%	89.83%	84.76%	91.75%	96.00%
ARC-Fix(0.1)	99.32%	100%	98.20%	91.88%	97.50%	89.77%	84.78%	91.69%	96.00%
ARC-Fix(0.2)	99.20%	100%	98.24%	92.76%	97.30%	89.82%	84.83%	91.70%	96.00%
ARC-Full	98.77%	100%	98.27%	93.10%	97.50%	89.82%	84.87%	91.67%	96.00%

Table 8.7: Real datasets. Binary classification rate on the testing set employed by *ARC-Dynamic*, *ARC-Fix(p)*, $p \in \{0.01, 0.05, 0.1, 2\}$ and *ARC-Full*, mean values over 20 runs.

and gains over the most effective runs with *ARC-Fix(p)* range from 11% to 27% in 7 out of 9 test problems and are larger than 20% in the solution of HTRU2, Cina0, MNIST, Reged0. This is confirmed by the performance profile displayed in Figure 8.5. Denoting by T the set of test problems in Table 8.6, by $S = \{$*ARC-Dynamic*, $\{$*ARC-Fix(p)*$\}_{p\in\{0.01,0.05,0.1,0.2\}}\}$ the set of the considered methods and by $E_{t,s}$ the number of EGE (at termination) to solve the problem $t \in T$ by the solver $s \in S$, the performance profile [22] for each $s \in S$ is defined as the fraction

$$\rho_s(\tau) = \frac{1}{|T|} \left| \left\{ t \in T : r_{t,s} = \frac{E_{t,s}}{\min\{E_{t,s} : s \in S\}} \leq \tau \right\} \right|, \qquad \tau \geq 1,$$

of problems in T solved by the method s with a performance ratio $r_{t,s}$ within a fraction τ of the best solver. Comparing the values of $\rho_s(1)$, $s \in S$, it can be seen that *ARC-Dynamic* outperforms the other solvers in the solution of all test problems. As already commented, the performances of the *ARC-Fix(p)* methods are instead more controversial. More specifically, *ARC-Fix(0.01)* and *ARC-Fix(0.2)* seem to be overall less efficient, even if *ARC-Fix(0.01)* is within a fraction $\tau = 1.08$ from the best solver on about 11% of the problems. *ARC-Fix(0.05)* solved all the problems within $\tau = 1.9$

Method	Mushroom $\epsilon = 10^{-3}$	Mushroom $\epsilon = 10^{-5}$	HTRU2	Cina0	Gisette	MNIST	A9A	Ijcnn1	Reged0
ARC-Dynamic	29.8	75.3	52.2	260.5	195.9	53.4	24.1	26.6	395.6
ARC-Fix(0.01)	41.5	140.1	87.0	405.2	397.3	136.1	37.0	28.4	600.3
ARC-Fix(0.05)	35.5	88.7	86.2	335.6	221.0	101.5	26.2	28.7	503.2
ARC-Fix(0.1)	39.6	92.1	76.1	340.7	231.0	72.8	28.2	31.3	796.3
ARC-Fix(0.2)	38.1	110.7	69.1	453.4	268.8	73.5	34.5	36.1	1353.5
ARC-Full	92.0	264.0	158.0	2300.0	836.0	173.0	87.0	78.0	6932.0

Table 8.8: Real datasets. Number of EGE employed by *ARC-Dynamic*, *ARC-Fix(p)*, $p \in \{0.01, 0.05, 0.1, 2\}$ and *ARC-Full*, mean values over 20 runs.

while, within such a value of τ, *ARC-Fix(0.1)* and *ARC-Fix(0.2)* solved 89% of the problems and *ARC-Fix(0.01)* solved 78% of the problems. Moreover, *ARC-Fix(0.2)* method requires a number of EGE which is within $\tau = 3.4$ from the best one to solve all the problems. Finally, in all runs we observed that the decreases of the training and testing loss with *ARC-Dynamic* is either comparable or faster than with *ARC-Fix(p)*. This features is displayed in Figures 8.6 where the training and testing loss is plotted versus the number of EGE; representative runs reported concern datasets MNIST and Gisette.

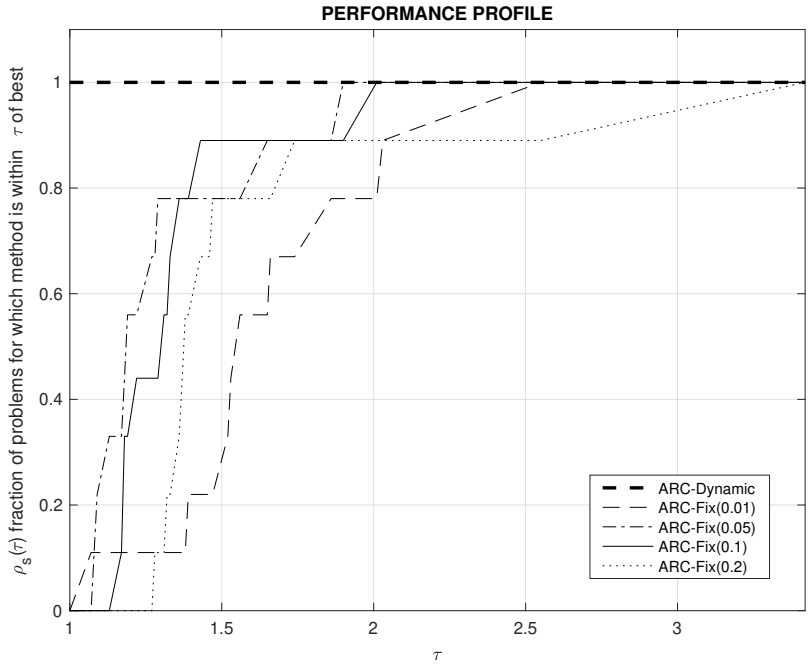

Figure 8.5: Performance profile (EGE count) on [1, 3.4] for real datasets.

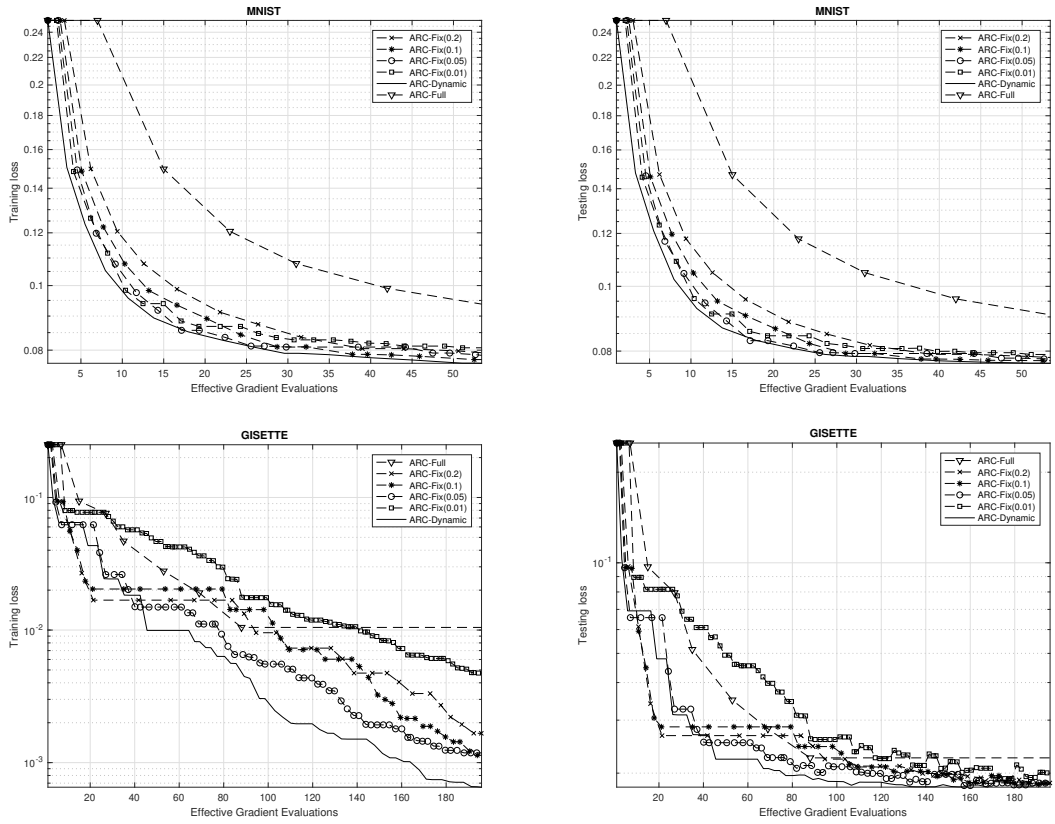

Figure 8.6: MNIST dataset (top), Gisette dataset (bottom), training loss (left) and testing loss (right) against EGE, logarithmic scale is on the y axis. *ARC-Dynamic*, *ARC-Fix(p)* with $p = 0.2$, $p = 0.1$, $p = 0.05$, $p = 0.01$ and *ARC-Full*.

9. Conclusions and perspectives. We proposed an ARC algorithm for solving nonconvex optimization problems based on a dynamic rule for building inexact Hessian information. The new algorithm maintains the distinguishing features of ARC framework, i.e., the optimal worst-case iteration bound for first- and second-order critical points. Application to large-scale finite-sum minimization is sketched and analyzed.

In case of sums of strictly convex functions the adaptivity allows to improve complexity results in terms of component Hessian evaluations over approaches that do not employ adaptive rules.

We tested the new algorithm on a large number of problems and compared its performance with the performance of ARC variants with optimal complexity and the performance of ARC variants employing a prefixed small Hessian sample size and showing suboptimal complexity. The former comparison was carried out on synthetic moderately ill-conditioned datasets while the latter comparison was carried out on machine learning datasets from the literature. Numerical results highlight that adaptiveness allows to reduce the overall computational effort and that the perfor-

mance of the proposed method is quite problem independent while strategies taking a prefixed fraction of samples require a trial and error procedure to set the most efficient sample size. In this work we have assumed that the objective function and the gradient are exact. Extensions of the approach proposed to the case where both function and gradient are evaluated with adaptive accuracy is desirable as well as the employment of variance reduction techniques. Moreover, convergence properties have been analyzed both under deterministic and probabilistic conditions, in the latter case properties of the deterministic algorithm are preserved in high probability. However, this analysis does not give indication on the properties of the method when the adaptive accuracy requirement is not satisfied. A stochastic analysis, in the spirit of [16], would be of interest and it is the topic of future research.

10. Acknowledgements. The authors wish to thank Raghu Bollapragada for gently providing the synthetic datasets and the referees for their insightful comments.

REFERENCES

[1] J. Barzilai, J. M. Borwein (1988) Two-Point Step Size Gradient Methods, *IMA Journal of Numerical Analysis*, **8**, pp. 14–148.

[2] S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint (2019) Adaptive Regularization Algorithms with Inexact Evaluations for Nonconvex Optimization. *SIAM Journal on Optimization*, **29**, pp. 2281–2915.

[3] S. Bellavia, N. Krejic, N. Krklec Jerinkic (2018) Subsampled Inexact Newton methods for minimizing large sums of convex functions. *IMA Journal of Numerical Analysis*, DOI: 10.1093/imanum/drz027.

[4] S. Bellavia, N. Krejic, B. Morini, Inexact restoration with subsampled trust-region methods for finite-sum minimization. *arXiv:1902.01710*.

[5] A.S. Berahas, R. Bollapragada, J. Nocedal (2017) An investigation of Newton-sketch and subsampled Newton methods. *arXiv:1705.06211*.

[6] T. Bianconcini, G. Liuzzi, B. Morini, M. Sciandrone (2015) On the use of iterative methods in cubic regularization for unconstrained optimization. *Comp. Optim. Appl.*, **60**, 35–57.

[7] E.G. Birgin, J.L. Gardenghi, J.M. Martínez, S.A. Santos and Ph.L. Toint (2017) Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Math. Progr., Ser. A* **163**, 359-368.

[8] L. Bottou, F.E. Curtis, J. Nocedal (2018) Optimization Methods for Large-Scale Machine Learning, *SIAM Review*, **60**, 223-311.

[9] R.H. Byrd, G.M. Chin, W. Neveitt, J. Nocedal (2018) On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning. *SIAM J. Optim.*, **21**, 977–995.

[10] Y. Carmon, J.C. Duchi (2019) Gradient descent efficiently finds the cubic-regularized non-convex Newton step, *SIAM J. Optim.*, **29**, 2146–2178.

[11] C. Cartis, N.I.M. Gould and Ph.L. Toint (2010) On the complexity of steepest descent, Newton's and regularized Newton's method for nonconvex unconstrained optimization. *SIAM J. Optim.*, **20**, 2833–2852.

[12] C. Cartis, N.I.M. Gould, Ph.L. Toint (2011) Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Math. Progr., Ser. A*, **127**, 245-295.

[13] C. Cartis, N.I.M. Gould, Ph.L. Toint (2011) Adaptive cubic overestimation methods for unconstrained optimization. Part II: worst-case function and derivative-evaluation complexity. *Math. Progr., Ser. A*, **130**, 295–319

[14] C. Cartis, N.I.M. Gould, Ph.L. Toint (2012) Complexity bounds for second-order optimality in unconstrained optimization. *J. Complex.*, **28**, 93–108.

[15] C. Cartis, N.I.M. Gould and Ph.L. Toint (2012) An adaptive cubic regularisation algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. *IMA J. Numer. Anal.*, **32**, 1662–1695.

[16] C. Cartis, K. Scheinberg (2018) Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Math. Progr., Ser. A*, **169**, 337–375.

[17] (2008) Causality workbench team, A marketing dataset, `http://www.causality.inf.ethz.ch/data/CINA.html`.

[18] C.C. Chang, C.J. Lin (2011) LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, **2**(3):27 `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

[19] X. Chen, B. Jiang, T. Lin, S. Zhang (2018) On Adaptive Cubic Regularized Newton's Methods for Convex Optimization via Random Sampling. *arXiv:1802.05426*.

[20] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. (2000) *Trust-Region Methods*. No. 1 in the 'MPS–SIAM series on optimization', Philadelphia, USA: SIAM.

[21] J.E. Dennis, J.J. Moré (1974) A characterization of superlinear convergence and its application to quasi-Newton methods. *Math. Comput.*, **28**, 549–560.

[22] E. D. Dolan, J. J. Moré (2002). Benchmarking optimization software with performance profiles. *Math. Program.*, **91**(2), 201–213.

[23] A. Griewank (1981) The modification of Newton's method for unconstrained optimization by bounding cubic terms. *Technical Report NA/12*, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, United Kingdom

[24] J.M. Kohler, A. Lucchi (2017) Subsampled cubic regularization for non-convex optimization *34th International Conference on Machine Learning, ICML 2017*. **4**, 2988-3011.

[25] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE, **8**6 2278-2324. MNIST database available at `http://yann.lecun.com/exdb/mnist/.`

[26] J.D. Lee, M. Simchowitz, M.I. Jordan, B. Recht (2016) Gradient Descent Only Converges to Minimizers. *JMRL: Workshop and Conference Proceedings*, **49**, 1–12.

[27] M. Lichman(2013) UCI machine learning repository, `https://archive.ics.uci.edu/ml/index.php.`

[28] J.J. Moré, D.C. Sorensen. (1983) Computing a trust region step. *SIAM J. Sci. Statist. Comput.*, **4**, 553–572.

[29] I. Mukherjee, K. Canini, R. Frongillo, Y. Singer (2013). *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, 17–32.

[30] Y. Nesterov and B.T. Polyak (2006) Cubic regularization of Newton's method and its global performance. *Math. Progr., Ser. A*, **108**, 177–205.

[31] M. Pilanci and M. J. Wainwright. (2017) Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM J. Optim.*, **27**, 205–245.

[32] F. Roosta-Khorasani, M.W. Mahoney. (2019) Sub-Sampled Newton Methods, *Math. Prog*, **174**, 293–326.

[33] M. Weiser, P. Deuflhard, B. Erdmann (2007) Affine conjugate adaptive Newton methods for nonlinear elastomechanics. *Optim. Methods Softw.*, **22**, 413–431.

[34] P. Xu, F. Roosta-Khorasani, M.W. Mahoney (2019) Newton-Type Methods for Non-Convex Optimization Under Inexact Hessian Information, *Math. Prog*, https://doi.org/10.1007/s10107-019-01405-z.

[35] P. Xu, F. Roosta-Khorasani, M.W. Mahoney (2017) Second-order optimization for non-convex machine learning: an empirical study. *arXiv:1708.07827*.

[36] Z. Yao, P. Xu, F.Roosta-Khorasani, M.W. Mahoney (2018) Inexact non-convex Newton-type methods. *arXiv:1802.06925*.

[37] D. Zhou, P. Xu, Q. Gu, (2019) Stochastic Variance-Reduced Cubic Regularization Methods. Journal of Machine Learning Research, **20**, pp.1–47.