Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN MATEMATICA
CICLO XXXIII

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Paolo Salani

Adaptive Regularisation Methods under Inexact Evaluations for Nonconvex Optimisation and Machine Learning Applications

Settore Scientifico Disciplinare MAT/08

Dottorando
Gianmarco Gurioli

Tutore
Prof. Stefania Bellavia

Coordinatore
Prof. Paolo Salani

Anni 2017/2020

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

Mathematical Institute, University of Oxford

Thesis candidated for the label of
DOCTOR EUROPAEUS

Institutions involved:

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM,
University of Oxford

The realisation of this thesis has benefited from fruitful discussions with Prof. Philippe L. Toint, during his annual visits at the University of Florence, and from valuable hints by Prof. Coralia Cartis, during my research period at the Mathematical Institute, University of Oxford.

Dottorando
Gianmarco Gurioli

Tutori
Prof. Stefania Bellavia,
Università degli Studi di Firenze
Prof. Coralia Cartis,
University of Oxford and Balliol College

Coordinatore
Prof. Paolo Salani

Contents

List of Algorithms

List of Tables

List of Figures

Introduction

In this thesis we mainly consider the numerical resolution of the following unconstrained minimisation problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

in which $f : \mathbb{R}^n \to \mathbb{R}$ represents the objective function, that is assumed to be sufficiently smooth ($f \in \mathcal{C}^2(\mathbb{R}^n)$) and bounded below.

In particular, we assume to bump into *large-scale instances* of the problem (i.e. with large values of the number of independent variables n) and to cope with highly nonlinear and/or nonconvex f, since this is the most frequent case when considering real-life applications, such as the one arising in machine learning. In this way, the set of stationary points $\mathcal{P} \overset{\text{def}}{=} \{\, x \in \mathbb{R}^n \mid \nabla f(x) = 0 \,\}$ can be extremely rich compared to the case of convex objective functions, where each stationary point, if any, is a global minimiser or, more restrictively, when looking at the case of strictly convex functions, admitting at most a global minimiser. Within nonconvexity, in fact, $\mathcal{P}$ can be populated by a variety of local/global minimiser, local/global maximiser and saddle points.

In this light, second-order procedures are interesting strategies to handle the situation in which a first-order optimisation solver gets stuck on a saddle point. This is because these methods go beyond function and gradient computations, making use of second-order information, let us say curvature information, cabled in Hessian computations. In this context, this thesis builds on Adaptive Regularisation algorithms using Cubics (ARC).

From a macroscopic point of view, this class of second-order globally convergent methods is based on an iterative scheme that starts from an initial guess $x_0 \in \mathbb{R}^n$ and generates a sequence

$$x_0,\ x_1,\ \ldots,\ x_k,\ \ldots,$$

aiming at reducing the objective from an iteration to the other, in order to have $f(x_k) < f(x_{k-1})$, for $k \geq 1$.

The basic idea is specular to the well-known Trust-Region methods with which ARC schemes share the fact that at each iteration a model based on the Taylor's series of f centered at x_k with increment s is considered and truncated to the second order. From this point on, the two approaches differ. In particular, instead of choosing a *suitably small* trust-region radius $\Delta_k > 0$ and solving the per-iteration problem

$$\min_{s\in\mathbb{R}^n,\ \|s\|\leq\Delta_k} f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \nabla^2 f(x_k) s$$

for obtaining a candidate s_k to define the new iterate $x_k + s_k$, the ARC approach solves the unconstrained minimisation problem

$$\min_{s\in\mathbb{R}^n} f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \nabla^2 f(x_k) s + \frac{\sigma_k}{3}\|s\|^3, \tag{2}$$

choosing a *sufficiently large* regularisation parameter $\sigma_k > 0$. The idea is thus to (approximately) solve (2), selecting the regulariser σ_k adaptively in order to get a suitable

decrease $f(x_k + s_k) < f(x_k)$ and, hence, setting $x_{k+1} = x_k + s_k$.
The choice of ARC methods in turn is not casual, since all the novel variants designed and analysed in this work focus on optimal complexity for reaching a first-order critical point under a certain accuracy and the complexity bounds for ARC methods have been proved to be optimal.

The notion of *complexity* we look at here consists in the worst-case count of outer iterations needed to find an approximate first or second-order critical point (iteration complexity), with the aim of deriving an upper bound on it. The existence of such a bound implies the termination of the algorithm and allows to derive its convergence properties. Moreover, when the per-iteration number of function and its derivatives computations is known, the bound for the worst-case iterations complexity applies also to give an upper bound on the number of function and derivatives evaluation, falling into the range of evaluation complexity. In particular, for ARC schemes employing exact function and derivatives, since each outer iteration of the method requires a function, a gradient and a Hessian computation, the two notions are the same.

In many applications it is necessary (and sometimes also useful) to evaluate $f(x)$ and/or $\nabla^j f(x)$, $j \in \{1, 2\}$, inexactly. We need only think to variable accuracy schemes, evaluations affected by noise, approximations based on subsampling scheme or complicated computations involving truncated iterative processes. As a consequence, a careful investigation of inexact versions of second-order methods has got increasing interest in literature. In particular, in the ARC context, approximate evaluations of f and its derivatives have to be incorporated in (2), in order to reduce the per-iteration cost of the procedure, while preserving optimal complexity and without deteriorating (hopefully) the overall performance of the algorithm.

In so doing, great attention has to be placed on designing levels of resemblance between approximate function/derivatives and their exact counterparts that are reliable and implementable, basing on known and already computed quantities. It should be specified that for maintaining the convergence and complexity properties of the basic exact version of the method, such accuracy agreements usually have to be fulfilled at each iteration k of the scheme, yielding a *deterministic complexity analysis*.

On the other hand, we will see that the practical estimate of f and its derivatives at a certain iterate is not uniquely identified and can be obtained in practice only under a certain probability, generating a non-deterministic algorithm. A relevant example in such a context, common to many machine learning applications, is given by finite-sum minimisation problems, i.e. occurrences of (1) with $f(x) = \frac{1}{N}\sum_{i=1}^{N} \varphi_i(x)$, $\varphi_i : \mathbb{R}^n \to \mathbb{R}$, and N a positive integer, for which accuracy requirements on f and its derivatives can be practically achieved within an arbitrary prescribed probability by means of concentration inequalities (see, e.g., [115]). Therefore, the complexity bounds coming from the deterministic analysis can only be restored in probability, in the sense that they still hold but within a certain probability of success, depending on the probabilities of the function and derivatives approximations to match the prescribed accuracy levels. This is the field of the so-called *probabilistic analysis*, that derives probabilistic conditions under which properties of the deterministic algorithms are preserved without providing an algorithm which is robust against failures to satisfy the adaptive accuracy requirements.

In fact, this approach does not give information about the true decrease of the objective function when the accuracy of the estimates for f or the derivatives are not satisfied; in other words, nothing is said about a possible upper bound on the expected number of steps needed to reach an approximate stationary point for the first time. This upper bound is challenging and its investigation calls for a *stochastic analysis* of the methods.

The major aim of this thesis is essentially to handle the main challenges presented above, exploiting the design, analysis and development of novel variants of ARC methods, employing inexact derivatives and/or function evaluations.

To start with, Part I introduces a suitable reference version of the ARC method, obtained by merging existing basic forms of ARC algorithms, in order to set the general background on adaptive cubic regularisation, rederiving its convergence and, mainly, complexity properties, for higher understanding of the methods and theory presented in the subsequent parts and chapters.

Having set the scene, Part II copes with the need of introducing inexactness in function and derivatives computations while conserving optimal complexity. After setting the finite-sum minimisation framework (Chapter 2), this starts from Chapter 3 with the employment of inexact Hessian information, adaptively chosen, before moving on to Chapter 4, in which an extended framework based on function estimates together with approximate derivatives evaluations is considered.

Part III is then devoted to handle the stochastic complexity analysis of the frameworks presented in Chapter 3 and Chapter 4, addressed in Chapter 5 and Chapter 6, respectively.

Finally, Part IV faces the numerical tests of the proposed methods in the context of supervised learning, ranging from popular machine learning datasets to a real-life machine learning industrial application related to the parametric design of centrifugal pumps.

Notations. $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{R}$ denote the set of integers, natural and real numbers, respectively. Given the scalar or vector or matrix v, and the nonnegative scalar χ, we write $v = \mathcal{O}(\chi)$ if there is a constant g such that $\|v\| \leq g\chi$. Given any set $\mathcal{S}$, $|\mathcal{S}|$ denotes its cardinality. Unless otherwise specified, $\|\cdot\|$ denotes the standard Euclidean norm for vectors and matrices. For a general symmetric tensor S of order $p \geq 1$, we define

$$\|S\|_{[p]} \overset{\text{def}}{=} \max_{\|v\|=1} |S[v]^p| = \max_{\|v_1\|=\cdots=\|v_p\|=1} |S[v_1, \ldots, v_p]| \tag{3}$$

the induced Euclidean norm (see [126, Theorem 2.1] for a proof of the second equality). Given a vector $x \in \mathbb{R}^n$ and a symmetric matrix $A \in \mathbb{R}^{n\times n}$, we denote by $\|x\|_A$ the A-norm of x, defined by $\|x\|_A \overset{\text{def}}{=} \sqrt{x^\top Ax}$. Given n scalars $\{\lambda_i\}_{i=1}^n$, $diag(\lambda_1, \cdots, \lambda_n)$ denotes the diagonal matrix of $\mathbb{R}^{n\times n}$ with diagonal components (i,i) equal to λ_i, for $i \in \{1, \cdots, n\}$. We also denote by $\nabla_x^j f(x)$ the j-th order derivative tensor of f evaluated at x, noting that such a tensor is always symmetric for any $j \geq 2$. $\nabla_x^0 f(x)$ and $\nabla_x^1 f(x)$ are synonyms for $f(x)$ and $\nabla_x f(x)$, respectively. The subscript x in $\nabla_x^j f(x)$, $j \geq 0$, is omitted when obvious from the context. The notation $\nabla_x^j f(x)[s]^\ell$ denotes the j-th derivative tensor applied to j copies of the vector s. We use notations $\text{globmin}_{x\in\mathcal{S}} f(x)$ and $\text{globmax}_{x\in\mathcal{S}} f(x)$ to denote the smallest and biggest value of $f(x)$ over $x \in \mathcal{S}$, respectively. All inexact functional evaluations are indicated by an overbar. $\lceil\alpha\rceil$ and $\lfloor\alpha\rfloor$ denote the smallest integer not smaller than α and the largest integer not exceeding $\alpha \in \mathbb{R}$ ($\alpha \geq 0$), respectively. For symmetric matrices, $\lambda_{\min}(M)$ is the leftmost eigenvalue of M. As usual, $\mathbb{R}^+$ denotes the set of positive real numbers, 0 is used for both the scalar zero and the null vector, while we use the notation $I_n \in \mathbb{R}^{n\times n}$ to denote the identity real matrix of dimension $n \times n$. For nonnegative integers i and j, we denote by $\delta_{ij} \in \{0,1\}$ the Kronecker delta, such that $\delta_{ij} = 1$ if and only if $i = j$. Given k vectors $\{v_i\}_{i=1}^k \subseteq \mathbb{R}^n$, $\text{span}\{v_1, ..., v_k\}$ represents the subspace of $\mathbb{R}^n$ generated by $v_1, ..., v_k$. We use the notation $\mathbb{E}[X]$ to indicate the expected value of a random variable X. In addition, given a random event E, $P(E)$ denotes the probability of E, while $\mathbb{1}_E$ refers to the indicator of the random event E occurring (i.e. $\mathbb{1}_E(z) = 1$ if $z \in E$, otherwise $\mathbb{1}_E(z) = 0$). The notation E^c indicates that event E does not occur. Symbol "$\wedge$" is the logical conjunction operator "and". Finally, we denote by $e_i \in \mathbb{R}^n$ the i-th orthonormal basis vector of $\mathbb{R}^n$ (i.e., $e_i \in \mathbb{R}^n$ has zero entries except from the i-th component which is equal to 1) and by e the Euler's number.

Part I

General Background on Adaptive Cubic Regularisation

Chapter 1

The Adaptive Cubic Regularisation (ARC) Framework

Adaptive Regularisation methods using Cubics (ARC) are Newton-type procedures for solving unconstrained optimisation problems of the form

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

in which $f : \mathbb{R}^n \to \mathbb{R}$ is a sufficiently smooth and possibly nonconvex function. To ensure the well-posedness of the considered problem, the objective function in (1.1) is additionally assumed to be bounded below. Since the seminal work [95] by Nesterov and Polyak, the method is proved to be a globally convergent second-order iterative scheme. In fact, regardless of the initial guess $x_0 \in \mathbb{R}^n$, it generates a sequence of iterates $\{x_k\}_{k\geq 0}$ until a point $\widetilde{x}$ satisfying the approximate first-order optimality condition,

$$\|\nabla f(\widetilde{x})\| \leq \epsilon_1, \tag{1.2}$$

or the second-order approximate optimality condition

$$\|\nabla f(\widetilde{x})\| \leq \epsilon_1 \quad \wedge \quad \lambda_{\min}(\nabla^2 f(\widetilde{x})) \geq -\epsilon_2, \tag{1.3}$$

is met, where $\epsilon_j > 0$, $j \in \{1, 2\}$, denotes the j-th order tolerance. As usual (see, e.g., [40]), conditions (1.2)–(1.3) are the discrete counterparts of the first and second-order optimality necessary conditions, reported below for the sake of completeness.

Theorem 1. (First-Order Necessary Optimality Condition). With reference to problem (1.1), if x^* is a local minimiser and f is continuously differentiable in an open neighbourhood of x^*, then $\nabla f(x^*) = 0$.

Theorem 2. (Second-Order Necessary Optimality Condition). With reference to problem (1.1), if x^* is a local minimiser and $\nabla^2 f$ exists and is continuous in an open neighbourhood of x^*, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

The method can thus be used as a concrete alternative to line search [56] and trust-region [52] algorithms, commonly-used as convergence schemes for unconstrained optimisation to globalise Newton-like iterations.

Like the trust-region approach, the aim of the k-th iteration of ARC methods is, given the current iterate x_k, to find a step $s \in \mathbb{R}^n$ such that $x_k + s$ reduces f, i.e. such that

$$f(x_k + s) < f(x_k).$$

To do that, assuming that f is twice continuously differentiable, both algorithms make use of the Taylor's series as a model to predict the values of the objective function f in a neighborhood of the current iterate x_k.

From this point on the two approaches differ and let us briefly recall the main frame of the Trust-Region framework in [52]. To start with, the k-th iteration of trust-region schemes considers a ball centered in x_k with radius Δ_k in which we trust that the following quadratic model

$$m_k^{TR}(s) \overset{\text{def}}{=} f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \overline{\nabla^2 f}(x_k) s, \tag{1.4}$$

used to predict $f(x_k + s)$, is reliable. This is, mainly, the Taylor's expansion centered at x_k with increment s, truncated to the second order in which $\overline{\nabla^2 f}(x_k)$ can represent the true Hessian $\nabla^2 f \in \mathbb{R}^{n\times n}$ evaluated at x_k (Newton Trust-Region method) or it can be replaced by an approximation given by a symmetric matrix $\overline{\nabla^2 f}(x_k)$ (Quasi-Newton Trust-Region method), in order to decrease the per-iteration computational cost of the algorithm. Hence, it is expected to be a good estimate if $\|s\|$ is sufficiently small and $\overline{\nabla^2 f}(x_k)$ is a suitably good approximation of $\nabla^2 f(x_k)$. In fact, assuming that $\|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s\| = \mathcal{O}(\|s\|^2)$, the second-order Taylor's expansion of f at x_k coupled with the definition (1.4) gives:

$$\begin{aligned} f(x_k + s) &= f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \nabla^2 f(x_k) s + \mathcal{O}(\|s\|^3) \qquad (1.5)\\ &= m_k^{TR}(s) + \frac{1}{2} s^\top \left(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\right) s + \mathcal{O}(\|s\|^3) \\ &= m_k^{TR}(s) + \mathcal{O}(\|s\|^3) \end{aligned}$$

and, hence, $f(x_k + s) - m_k^{TR}(s) = \mathcal{O}(\|s\|^3)$. In this view, a radius $\Delta_k > 0$ coming from the previous iteration is considered and the new trial iterate $x_k + s$ generated at iteration k is required to lay in the ball

$$\mathcal{B}_k = \{\, x \in \mathbb{R}^n \mid \|x - x_k\|_* \le \Delta_k \,\},$$

in which $\|\cdot\|_*$ denotes a generic vectorial norm. A common choice is to define $\mathcal{B}_k$ referring to the Euclidean norm (see, e.g., [97]). The step s_k is then defined by solving the following quadratically constrained quadratic minimisation problem, known as the trust-region minimisation subproblem:

$$\min_{s\in\mathbb{R}^n,\ \|s\|\le\Delta_k} m_k^{TR}(s) = \min_{s\in\mathbb{R}^n,\ \|s\|\le\Delta_k} f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \overline{\nabla^2 f}(x_k) s. \tag{1.6}$$

It is worth noting that both the objective and the constraint (that can be rewritten as $s^\top s \le \Delta_k^2$ in (1.6)) are quadratic. Moreover, exact minimisation in (1.6) is not needed to gain global convergence and prove complexity, but it suffices to find a step s_k that induce a "sufficient" decrease in the model. See [97, 52] for techniques employing exact minimisation of the trust-region subproblem (1.6) and [52] for inexact resolutions. In addition, we do not forget that the subproblem minimisation is aimed at obtaining a reduction of the objective. To this purpose, the following quantities are computed at iteration k.

- *Actual reduction* of the objective function: $f(x_k) - f(x_k + s_k)$;
- *Predicted reduction* of the objective function: $m_k^{TR}(0) - m_k^{TR}(s_k)$;

- *Decrease ratio*:

$$\rho_k^{TR} \overset{\text{def}}{=} \frac{f(x_k) - f(x_k + s_k)}{m_k^{TR}(0) - m_k^{TR}(s_k)}. \tag{1.7}$$

Note that, when minimising (1.6), $s = 0 \in \mathcal{B}_k$ implies that $m_k^{TR}(s_k) \leq m_k^{TR}(0)$, so the predicted reduction is nonnegative by definition and the sign of ρ_k^{TR} is given by its numerator. If $\rho_k^{TR} \geq \eta_1 \in (0, 1)$, then a sufficient decrease of the objective function has actually been gained, meaning that $f(x_k + s_k) \leq f(x_k) - \eta_1(m_k^{TR}(0) - m_k^{TR}(s_k))$, and thus the step s_k is accepted to update the iterate, setting $x_{k+1} \overset{\text{def}}{=} x_k + s_k$. Otherwise, the step s_k is rejected. Finally, the guideline to update the radius Δ_k relies on the fact that (1.7) provides a measure of accordance between the model used and the function over the step, in the sense that the larger is the ratio ρ_k^{TR} (close to 1), the better is the agreement between the two.

Note that:

$$\rho_k^{TR} \simeq 1 \quad \Rightarrow \quad m_k^{TR}(s_k) \simeq f(x_k + s_k),$$

being $m_k^{TR}(0) = f(x_k)$ by definition of $m_k^{TR}(s)$ in (1.4). Therefore, if ρ_k^{TR} is large (in practice above a prescribed decrease fraction $\eta_2 \in [\eta_1, 1)$, with $0 \leq \eta_1 < 1$), then enlarging Δ_k is considered licit, solving the trust-region subproblem (1.6) at the next iteration in a larger ball; on the other hand, if ρ_k^{TR} is small (even close to be null), the model is not considered as a good approximation of the objective function f in the ball $\mathcal{B}_k$ and the radius Δ_k is reduced. The detailed iteration k of the Trust-Region algorithm in [52] is reported below.

Algorithm 1 The Trust-Region algorithm [52](iteration k).

Step 0: Initialisation. Given the current iterate x_k, the radius $\Delta_k > 0$, the trust-region $\mathcal{B}_k$, the constants η_1, η_2, γ_1, γ_2, s.t.

$$0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 \leq \gamma_2 < 1. \tag{1.8}$$

Compute $f(x_k)$.

Step 1: Model definition. Build the model $m_k^{TR}(s)$ according to (1.4).

Step 2: Step computation. Compute a step s_k by approximately solving the trust-region subproblem (1.6) that "sufficiently reduces" the model.

Step 3: Acceptance of the trial step. Compute $f(x_k + s_k)$ and ρ_k^{TR} defined in (1.7):

$$\rho_k^{TR} = \frac{f(x_k) - f(x_k + s_k)}{m_k^{TR}(0) - m_k^{TR}(s_k)}.$$

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

Step 4: Trust-region radius update. Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty], & \text{if } \rho_k^{TR} \geq \eta_2, \\ [\gamma_2\Delta_k, \Delta_k], & \text{if } \rho_k^{TR} \in [\eta_1, \eta_2), \\ [\gamma_1\Delta_k, \gamma_2\Delta_k], & \text{if } \rho_k^{TR} < \eta_1. \end{cases}$$

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

The ARC approach stems from the idea of replacing the subproblem (1.6) by an *unconstrained* optimisation subproblem to be solved at each iteration, in which the model is still based on the Taylor's expansion truncated to the second order, but incremented by an additional cubic term.

To get more insight, let us assume that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous on a neighborhood of $\mathbb{R}^n$ containing the path of iterates $\{x_k\}_{k\geq 0}$, being $L > 0$ its 2-norm Lipschitz constant (see Assumption 1.2.2), then from the Taylor's expansion of f centered at x_k with increment s, it first follows that

$$\begin{aligned} f(x_k+s) &= f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \nabla^2 f(x_k) s + \\ &\quad + \int_0^1 (1-\tau) s^\top (\nabla^2 f(x_k+\tau s) - \nabla^2 f(x_k)) s \, d\tau \qquad (1.9) \\ &\overset{\text{def}}{=} T_2^E(x_k, s) + \int_0^1 (1-\tau) s^\top (\nabla^2 f(x_k+\tau s) - \nabla^2 f(x_k)) s \, d\tau \\ &\leq T_2^E(x_k, s) + \frac{1}{6} L \|s\|^3 \overset{\text{def}}{=} m_k^C(s), \qquad (1.10) \end{aligned}$$

for all $s \in \mathbb{R}^n$, where $T_2^E(x_k, s)$ is such that $T_2^E(x_k, 0) = f(x_k)$ and denotes, for later use, the Taylor's expansion of f centered at x_k with increment s truncated to the second-order.

The idea is indeed to overestimate $f(x_k+s)$ by the cubic model $m_k^C(s)$. Thus, so long as

$$m_k^C(s) < m_k^C(0) = f(x_k),$$

the new iterate $x_{k+1} = x_k + s_k$ reduces $f(x)$, providing $f(x_{k+1}) < f(x_k)$. The minimisation of the model $m_k^C(s)$ can indeed be considered to generate the step s_k, forming the basis for a new unconstrained optimisation algorithm.

The whole thesis elaborates on this algorithm, developing different independent lines of research. We briefly reviewed here the main contributions on the topic.

The overestimation in (1.10) was firstly considered in the unpublished technical report [70] written in 1981 by Griewank within the context of affine-invariant variants of Newton's method which are globally convergent to second-order critical points. In that work, a variant of (1.10) is introduced. The core of this variant mainly consists in replacing $L/2 > 0$ by a k-dependent term $\sigma_k > 0$, considering the new model definition

$$m_k^G(s) \overset{\text{def}}{=} T_2^E(x_k, s) + \frac{1}{3} \sigma_k \|s\|_{G_k}^3, \qquad (1.11)$$

where $\sigma_k \|\cdot\|_{G_k}$ is iteratively chosen to ensure the overestimation in (1.10), while preserving the affine-invariant structure. In the work, all the minimisers (local or global) of the model which provide descent are characterised and global convergence to second-order critical points is proved if the step s is computed by finding any second-order minimiser of the model inducing a descent of the objective f. Concerning the convergence analysis, quadratic local convergence is proved and the convergence proofs assume a global Hölder condition on the Hessian of the objective function together with the assumption that the sequence of symmetric positive definite matrices G_k stabilises over iterations. Using a variant of the nonlinear conjugate-gradient method, he also exploits the case in which the minimisation of (1.11) is approximately performed, showing the rate of convergence and giving some preliminary numerical results.

In more recent years, Nesterov and Polyak (see [95]) reconsidered the model (1.10) showing that the resulting algorithm has a better global iteration worst-case complexity bound with respect to the steepest descent method, if the step is computed via global minimisation and the Hessian $\nabla^2 f(x)$ is globally Lipschitz continuous. The main contribution of the paper is related to global worst-case complexity bounds for different problem classes, including some nonconvex cases. It is also shown that the search direction can be computed by standard linear algebra techniques.

In this context, the notion of *iteration complexity* consists in the worst-case number of outer iterations required by the considered algorithm to reach a first or second-order critical point, while when the bound refers to the number of function and its derivatives evaluation, we fall into the range of *evaluation complexity*. Usually (and this is the case of this thesis), the study of evaluation complexity also includes iteration complexity, since it stems from it counting the number of function and derivatives evaluations needed at each outer iteration of the method. Although no numerical results were provided, global convergence to second-order critical points and asymptotically quadratic rate of convergence were analysed.

Subsequently, Nesterov has proposed in [94] more sophisticated methods to further improve the complexity bounds in the convex case.

In 2007, the work [122] by Wieser, Deuflhard and Erdmann independently turned to the same line of thought, motivated (as Griewank) by the design of an affine-invariant version of Newton's method. Their approach directly evolved from techniques for convex problems and, elaborating on [58], it makes use of the cubic model (1.11) with the choice $G_k = \sigma_k G$ where G is positive definite and σ_k is an estimate of the global Lipschitz constant, following similar updating rules for σ_k as the ones by Griewank. The proposed method does not consider global model minimisation, but rather uses approximate techniques for finding a local minimiser, such as Krylov's subspace techniques and nonlinear conjugate-gradients. Again, global Lipschitz continuity is assumed, but no formal convergence or complexity analysis is presented, while limited but encouraging numerical experiments are discussed.

We start here from the ARC formulation given in [40, 41] by Cartis, Gould and Toint, aimed at unifying and extending the previous contributions into a coherent and numerically efficient algorithmic framework, for which global and asymptotic convergence results can be proved under weaker assumptions and with simpler proofs, while preserving the complexity bounds shown in [95].

At each iteration of their approach, an approximate global minimiser of a local cubic regularisation model for the objective function, which remains computationally-viable even for large-scale problems, is determined and this ensures a significant improvement in the objective so long as the Hessian of the objective is locally Lipschitz continuous, without insisting in that $\nabla^2 f(x)$ be globally Lipschitz or Hölder continuous for what concerns the convergence properties to first-order critical points, while the globally Lipschitz continuity of the Hessian is needed for convergence to second-order critical points and the proof of complexity. Let us turn to the overestimation in (1.10). It is worth noting that the cubic term in the definition of m_k^C depends on the Lipschitz constant L of the Hessian and, hence, it implicitly assumes the knowledge of that constant to be computed at step k. Such an assumption can be quite stringent, especially when minimising nonconvex objective functions.

It is for this reason that the adaptive cubic regularisation framework deals with the idea of getting rid of the Lipschitz constant, trying to overestimate it by the adaptive cubic regulariser σ_k. If

$$\sigma_k \geq \frac{L}{2}, \tag{1.12}$$

the following inequality directly follows from (1.10):

$$f(x_k + s) \leq T_2^E(x_k, s) + \frac{1}{3}\sigma_k \|s\|^3 \overset{\text{def}}{=} m_k^E(s). \tag{1.13}$$

Consequently, any vector $s \in \mathbb{R}^n$ satisfying

$$m_k^E(s) < m_k^E(0) = f(x_k)$$

provides a reduction of the objective function f at $x_k + s$ with respect to the current value

$f(x_k)$. From now on, the Euclidean norm is considered for simplicity of exposition. Hence, the idea of the ARC framework is to (approximately) minimise the cubic model (1.13) at each iteration of the method, choosing the cubic regularisation parameter σ_k with the aim of ensuring the overestimation property in (1.13).

The authors in [40, 41] also allows for inexactness in the Hessian computation, redefining the model as the following regularised approximate Taylor's series

$$\begin{aligned} m_k(s) &\overset{\text{def}}{=} f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \overline{\nabla^2 f}(x_k) s + \frac{1}{3}\sigma_k \|s\|^3 \\ &\overset{\text{def}}{=} \widehat{T}_2(x_k, s) + \frac{1}{3}\sigma_k \|s\|^3, \end{aligned} \tag{1.14}$$

such that

$$\nabla_s m_k(s) = \nabla_s \widehat{T}_2(x_k, s) + \sigma_k s\|s\| = \nabla f(x_k) + \overline{\nabla^2 f}(x_k) s + \sigma_k s \|s\|, \tag{1.15}$$

where $\overline{\nabla^2 f}(x_k)$ is a symmetric approximation to the local Hessian $\nabla^2 f(x_k)$ in the cubic model on each iteration. This may be highly useful in practice, enabling to reduce the computational cost related to the exact Hessian computation.

Of course, the replacement of the true Hessian $\nabla^2 f(x_k)$ with its approximation $\overline{\nabla^2 f}(x_k)$ in the model definition will affect the overestimation property of the model, since (when $\overline{\nabla^2 f}(x_k) \neq \nabla^2 f(x_k)$) it is no longer guaranteed that

$$f(x_k + s) \le m_k(s), \tag{1.16}$$

for any $s \in \mathbb{R}^n$ and under the validity of (1.12). The regulariser σ_k therefore performs a double task: it may account not only for the discrepancy between the objective function and its second order Taylor's expansion, but also for the difference between the exact and the approximate Hessian. Nevertheless, it can be proved that if the Hessian approximation $\overline{\nabla^2 f}(x_k)$ is accurate enough, then a different lower bound on σ_k (compared to (1.12)) still ensures that each vector $s \in \mathbb{R}^n$ such that $m_k(s) < m_k(0) = f(x_k)$ provides $f(x_k+s) < f(x_k)$. We will further discuss this point in the next section, in which the reference ARC algorithm is introduced.

Alternatively, a secant-like strategy is employed in [29] to define the cubic term in the used cubic model. Other ways of estimating such a term are also given in [21], via a mixed factorisation based on the Bunch-Parlett-Kaufman decomposition, and in [70], where the local cubic terms are bounded using ellipsoidal norms.

This first part of the thesis is organised as follows. Section 1.1 introduces a suitable version of the ARC algorithm considered as the basic scheme for its extensions in the upcoming parts of the thesis. Such a version is configured as a combination of existing ARC algorithms and, hence, it requires a first research investigation to analyse its worst-case complexity analysis and convergence to first and second-order critical points, that is the focus of Section 1.2. Finally, Section 1.3 gives useful guidelines for approximate minimising the cubic subproblem involved within the ARC framework.

1.1 The ARC algorithm

In this section we recall the main facts about the ARC algorithm. In order to make it useful for the lines of research developed in the next chapters, the ARC scheme is introduced embedding the model definition (1.14), like [40], in the scheme of [22, Algorithm 1 (case $p = 2$)]. We preliminary assume that

$$f \in \mathcal{C}^1(\mathbb{R}^n).$$

Algorithm 2 The ARC algorithm.

```
Step 0: Initialisation. Given an initial point $x_0$, the initial regulariser $\sigma_0 > 0$, the positive
accuracy level $\epsilon_1$. Given $\theta, \eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3, \sigma_{\min}$ s.t.

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

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

Step 1: Test for termination. Evaluate $\nabla f(x_k)$. If $\|\nabla f(x_k)\| \le \epsilon_1$, terminate with the approximate
solution $\widehat{x} = x_k$; otherwise, compute the model $m_k(s)$ as defined in (1.14).

Step 2: Step computation. Compute the step $s_k$ by approximately minimising the model $m_k(s)$ w.r.t. $s$ so that

    $m_k(s) < m_k(0),$                                   (1.17)
    $\|\nabla_s m_k(s)\| \le \theta \|s_k\|^2.$          (1.18)

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

    $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)}.$     (1.19)

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

Step 4: Regularisation parameters update. Set

    $\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k], & \text{if } \rho_k \ge \eta_2, \quad (\textit{very successful iteration}) \\ [\sigma_k, \gamma_2 \sigma_k], & \text{if } \rho_k \in [\eta_1, \eta_2), \quad (\textit{successful iteration}) \\ [\gamma_2 \sigma_k, \gamma_3 \sigma_k], & \text{if } \rho_k < \eta_1, \quad (\textit{unsuccessful iteration}). \end{cases}$     (1.20)

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

All the initial parameters and constants are given at Step 0, where f is also evaluated at the starting point x_0. The test for termination is then considered at Step 1. If it is not fulfilled, the model (1.14) is computed, where $\overline{\nabla^2 f}(x_k)$ is an approximation to the Hessian of f (provided the latter exists). We will see the usual condition on the agreement between the Hessian approximation $\overline{\nabla^2 f}(x_k)$ and the true one in the next section (see Assumption 1.2.3). At Step 3, given an estimate x_k of a critical point of f, a step s_k is computed satisfying (1.17)–(1.18). Once $f(x_k + s_k)$ is evaluated, the magnitude of the decrease ratio ρ_k decides about the acceptance of the trial point $x_k + s_k$. The step s_k is accepted and the new iterate x_{k+1} is set to $x_k + s_k$ whenever a reasonable fraction of the predicted model decrease $f(x_k) - \widehat{T}_2(x_k, s_k)$ is realised by the actual decrease in the objective, namely $f(x_k) - f(x_k + s_k)$. This is measured by computing the ratio ρ_k in (1.19) and requiring ρ_k to be greater than a prescribed positive constant $\eta_1 \in (0, 1)$. We will shortly see (Remark 2 on page 15) that ρ_k is well defined, provided that (1.17) holds. Since the current regulariser σ_k has resulted in a successful step, there is no reason to increase it, and indeed there may be benefits in decreasing it if a good agreement between model and function is observed. By contrast, if ρ_k is smaller than η_1, we judge that the improvement in objective is not sufficient; indeed, as already noticed in the introduction, there is no improvement if $\rho_k \le 0$. If this happens, the step will be rejected and x_{k+1} left as x_k. Under these circumstances, the only recourse available is to increase σ_k prior to the next iteration with the implicit intention of reducing the size of the step, aiming at satisfying the (local) overestimation property.

This property is represented by the one in (1.13) in the particular case in which $\overline{\nabla^2 f}(x_k) =$

$\nabla^2 f(x_k)$, otherwise a new overestimation property has to be proved. Moreover, evidence of the fact that the updating rule at Step 4 ensures the termination of the scheme has to be given. These two latter aspects will be fully addressed in the analysis of the next section.

Remark 1. We stress that the use of the condition (1.17), in place of an inequality like $m_k(s_k) \leq m_k(0)$, entains that a nonzero step s_k satisfying (1.17)–(1.18) can always be computed by Algorithm 2 before termination. Therefore, two possible situations have to be excluded. The first one is when the model $m_k(s)$ built at iteration k admits only the minimiser $\widehat{s}_k = 0$ (for which $m_k(\widehat{s}_k) = 0 = m_k(0)$). In such a case, from (1.15) we have $0 = \nabla_s m_k(\widehat{s}_k) = \nabla f(x_k)$ and Algorithm 2 would have already stopped, due to the fulfillment of the termination criterion $\|\nabla f(x_k)\| \leq \epsilon_1$. It remains thus to exclude the case in which the model $m_k(s)$ has a minimiser $\overline{s}_k \neq \widehat{s}_k = 0$, such that $m_k(\overline{s}_k) = m_k(0)$. To cope with that, recalling that by definition $m_k(0) = f(x_k)$, we notice that $m_k(\overline{s}_k) = m_k(0)$ implies

$$\nabla f(x_k)^\top \overline{s}_k + \frac{1}{2}\overline{s}_k^\top \overline{\nabla^2 f}(x_k)\overline{s}_k + \frac{1}{3}\sigma_k\|\overline{s}_k\|^3 = 0. \tag{1.21}$$

Moreover, if $\overline{s}_k$ is a minimiser of $m_k(s)$, it has to satisfy the necessary optimality condition (1.3), that together with (1.14)–(1.15) implies that

$$\begin{aligned} 0 &= \nabla_s m_k(\overline{s}_k) = \nabla f(x_k) + \overline{\nabla^2 f}(x_k)\overline{s}_k + \sigma_k \overline{s}_k\|\overline{s}_k\|, & (1.22)\\ 0 &\leq \overline{s}_k^\top \nabla^2 m_k(\overline{s}_k)\overline{s}_k = \overline{s}_k^\top \overline{\nabla^2 f}(x_k)\overline{s}_k + \sigma_k\|\overline{s}_k\|^3. & (1.23)\end{aligned}$$

Left-multiplying (1.22) by $\overline{s}_k^\top$ and substracting (1.21) from the resulting equation, we get that

$$\frac{1}{2}\overline{s}_k^\top \overline{\nabla^2 f}(x_k)\overline{s}_k + \frac{2}{3}\sigma_k\|\overline{s}_k\|^3 \tag{1.24}$$

is null, which is excluded by the validity of (1.23). In fact, (1.23) is equivalent to

$$\frac{1}{2}\overline{s}_k^\top \overline{\nabla^2 f}(x_k)\overline{s}_k \geq -\frac{1}{2}\sigma_k\|\overline{s}_k\|^3,$$

implying from (1.24) that

$$\frac{1}{2}\overline{s}_k^\top \overline{\nabla^2 f}(x_k)\overline{s}_k + \frac{2}{3}\sigma_k\|\overline{s}_k\|^3 \geq \left(\frac{2}{3} - \frac{1}{2}\right)\sigma_k\|\overline{s}_k\|^3 = \frac{1}{6}\sigma_k\|\overline{s}_k\|^3 > 0,$$

where the last inequality is due to $\overline{s}_k \neq 0$ and $\sigma_k \geq \sigma_{\min} > 0$.
The same argument applies to the ARC-DH algorithm in Chapter 3 and the SARC-IGDH algorithm in Chapter 5.

We proceed by deriving a preliminar result on the model decrease obtained under the condition (1.17), also ensuring the well-definitiveness of ρ_k in (1.19).

Lemma 3. With reference to the model definition (1.14), assume that Assumption 1.2.1(ii) and (1.17) hold. Then,

$$\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) \geq \frac{\sigma_k}{3}\|s_k\|^3. \tag{1.25}$$

Proof. Because of (1.17) and (1.14):

$$0 < m_k(0) - m_k(s_k) = \widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) - \frac{\sigma_k}{3}\|s_k\|^3, \tag{1.26}$$

which implies the desired bound. □

Remark 2. Referring to Algorithm 2, we underline that from (1.25) and the regularisation parameter updating rule (1.20) it follows:

$$\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) \geq \frac{\sigma_k}{3}\|s_k\|^3 \geq \frac{\sigma_{\min}}{3}\|s_k\|^3 > 0, \tag{1.27}$$

in which $\|s_k\| \neq 0$ because of (1.17). Therefore, the ratio (1.19) is well-defined for all $k \geq 0$ and the same argument applies to [22, Algorithm 7].

The analogy between the construction of the ARC algorithm and of the basic Trust-Region method defined by Algorithm 1 is superficially evident in the choice of the measure ρ_k, i.e. the threshold for step acceptance and for the updating rule of σ_k, which is specular of the one used to update the trust-region radius Δ_k in Algorithm 1 (compare the Step 4 of Algorithm 2 with that of Algorithm 1). As noticed in [40], at a deeper level the parameter σ_k might be viewed as the *reciprocal* of the trust-region radius and it has here to be incremented suitably in order to let the overestimation property (1.16) with $s = s_k$ hold.

Following [40], the model definition (1.14) is given referring to the Euclidean norm. Anyway, due to the equivalence of norms on $\mathbb{R}^n$, the 2-norm could be replaced by the A-norm, given a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$. It can be shown that the convergence properties of Algorithm 2 established in what follows remain true in such a more general setting, although some of the constants involved change accordingly. The use of different norms may be viewed as an attempt to achieve affine invariance, as in the spirit of [70, 122]. Also, regularisation terms of the form $\|s_k\|^i$, $i > 2$, may be employed in (1.14) in place of the cubic term. Complexity aspects of such a modification are discussed in [41], while in [70] this extension is considered to cope with the possibility of Hölder rather than Lipschitz continuous Hessians.

The above Algorithm 2 is essentially [22, Algorithm 7] written for a third-order model and employing a Hessian approximation $\overline{\nabla^2 f}(x_k)$ to replace the exact value $\nabla^2 f(x_k)$. On the other hand, it reduces to [40, Algorithm 2.1] if the modified steps 2–3 reported below are considered within the scheme of Algorithm 2.

Algorithm 3 Modified Steps 2–3 of the ARC algorithm (Algorithm 2).

Step 2: Step computation. Compute a step s_k for which

$$m_k(s_k) \leq m_k(s_k^C), \tag{1.28}$$

where the Cauchy point s_k^C is defined by

$$s_k^C = -\alpha_k^C \nabla f(x_k), \qquad \alpha_k^C = \arg\min_{\alpha\in\mathbb{R},\ \alpha>0} m_k(-\alpha\nabla f(x_k)). \tag{1.29}$$

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

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\widehat{T}_2(x_k, 0) - m_k(s_k)}. \tag{1.30}$$

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

As for Trust-Region methods, we note that the Cauchy point is computationally inexpensive as it is a one-dimensional minimisation of a (two-piece) cubic polynomial; this involves finding roots of a quadratic polynomial and requires a matrix-vector and three vector products. Resorting to the Cauchy decrease is quite informative, since it enables to keep a tight control on the effort spent in the step computation. Its use is considered in [38, 40, 41] to prove convergence up to second-order critical points and to derive the complexity results. In particular (consider for simplicity of exposure $\epsilon_1 = \epsilon_2 = \epsilon$ in (1.2)–(1.3)), requiring the mild Cauchy condition (1.28) on the step, as in Algorithm 3, the authors in [41] obtain an upper bound on the total number of iterations the ARC algorithm takes to drive the norm of the gradient of f below ϵ that is of order ϵ^2, so the same as for the steepest descent method (see [96]). This is to be expected since the Cauchy-point condition requires no more than a move in the negative gradient direction.

The steepest descent-like complexity bound can be improved when s_k is the global minimiser of the model (1.14) in a subspace containing the gradient $\nabla f(x_k)$ and an appropriate termination criterion is employed (see, e.g., (3.26), (3.27)). In particular, assuming the Lipschitz continuity of $\nabla^2 f(x)$, and requiring $\overline{\nabla^2 f}(x_k)$ to be "sufficiently close" to $\nabla^2 f(x_k)$ along s_k (i.e. (1.31)), the authors in [40] show that their ARC algorithm has an overall worst-case iteration count of order $\epsilon^{-3/2}$ for generating $\|\nabla f(x_k)\| \leq \epsilon$, and of order ϵ^{-3} for achieving approximate nonnegative curvature in a subspace containing s_k. These bounds match those proved by Nesterov and Polyak in [95, Section 3] for their Algorithm 3.3. By contrast, the Cauchy-point requirement is not needed in [22] to derive the result of complexity (and, therefore, of convergence) for first-order stationary points, nor global optimisation is required at the level of the minimisation subproblem considered at Step 3 of Algorithm 2.

Therefore, instead of referring to results from [40, Algorithm 2.1] (proved in [40, 41]), from which Algorithm 2 inherits the use of Hessian approximations, or [22, Algorithm 1 (case $p = 2$)], with which Algorithm 2 shares the definition of ρ_k, we will recall then main properties of the ARC framework by building directly on Algorithm 2. This, in turn, will require the preliminary research effort of combining techniques from [40, 41, 22], but it will be helpful for later use, when dealing with the novel extensions of the basic ARC algorithm (Algorithm 2) considered in the next chapters of this thesis.

1.2 Worst-case evaluation complexity analysis of the basic ARC algorithm

The main reason to consider the ARC framework in place of other globalisation strategies, such as Newton-type methods embedded into a line search or a trust-region scheme, lies on its optimal complexity. That is to say that, given the first-order ϵ_1 tolerance, an ϵ_1-approximate first-order stationary point is reached, in the worst-case, in at most $\mathcal{O}(\epsilon_1^{-3/2})$ iterations (see, e.g. [95, 41, 30, 22]). On the other hand, basic trust-region and line search schemes take at most $\mathcal{O}(\epsilon_1^{-2})$ iterations to drive the norm of the gradient of f below ϵ_1 (see, e.g., [38, 42, 96]). We observe that due to the direct correspondence between the count of ARC iterations and the number of function and derivatives evaluations these bounds also apply to the latter (evaluation complexity). Moreover, in [31] it has been shown that the bound $\mathcal{O}(\epsilon_1^{-3/2})$ for computing an ϵ_1-approximate first-order stationary point is optimal among methods operating on functions with Lipschitz continuous Hessian. Further results showing optimal complexity bounds for approximate first-order stationary points can be found in [87], where a cubic regularisation counterpart of a variable-norm trust-region method for unconstrained minimisation is considered. Concerning the analogous bounds for convergence to second-order critical points, an (ϵ_1, ϵ_2)-approximate first and second -order critical point is found by ARC in at most $\mathcal{O}(\max(\epsilon_1^{-3/2}, \epsilon_2^{-3}))$ iterations (see, [40, 22, 38, 37]), instead of the $\mathcal{O}(\max(\epsilon_1^{-2}, \epsilon_2^{-3}))$ bound gained by trust-region procedures (see, e.g., [42]).

For this purpose, this section is devoted to derive the worst-case evaluation complexity analysis of the ARC framework considered in Algorithm 2, basing on the analysis in [40, 41, 22].

First of all, let us consider the following preliminary assumptions on f.

Assumption 1.2.1. *With reference to problem* (1.1)*, the objective function f is assumed to be:*

(i) bounded below by f_{low}, for all $x \in \mathbb{R}^n$;

(ii) twice continuously differentiable, i.e. $f \in \mathcal{C}^2(\mathbb{R}^n)$.

Assumption 1.2.2. *The Hessian of the objective function f in* (1.1) *is Lipschitz continuous on the path of iterates with Lipschitz constant L, i.e.*

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

We further assume that the Hessian approximation $\overline{\nabla^2 f}(x_k)$ considered for the model construction at Step 1 of Algorithm 2 satisfies the following agreement along the step s_k.

Assumption 1.2.3. *With reference to problem* (1.1)*, for all $k \ge 0$ and some constant $\chi > 0$, the agreement between $\nabla^2 f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ along s_k is such that*

$$\|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\| \le \chi\|s_k\|^2. \tag{1.31}$$

As it will be helpful for the upcoming chapters, we observe that a way of imposing (1.31) consists in prescribing that

$$\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\| \le \chi\|s_k\|, \tag{1.32}$$

which, by virtue of the Cauchy-Swarz inequality, trivially implies that

$$\|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\| \le \|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\|\|s_k\| \le \|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\| \le \chi\|s_k\|^2.$$

We underline that the theoretical requirement (1.31) is stronger than the well-known Dennis-Moré condition

$$\lim_{k\to+\infty} \frac{\|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\|}{\|s_k\|} = 0 \tag{1.33}$$

in [57], that is achieved by proper Quasi-Newton updates. Ensuring theoretically condition (1.32) is a crucial point for a practical implementation of the algorithm and it will be the focus of the next chapter.

Having set the main assumptions on f, we first ensure the validity of the overestimation property (1.16) for any $s \in \mathbb{R}^n$, given the Hessian approximation requirement (1.31) and provided that the cubic regularisation parameter σ_k is sufficiently large. As already noticed in the previous subsection (at the end of page 10), this motivates the decreasing of the regulariser σ_k on very successful iterations in the design of the basic Algorithm 2.

Lemma 4. With reference to problem (1.1) and the model definition in (1.14), let Assumption 1.2.1(ii), Assumption 1.2.2 and Assumption 1.2.3 hold.
For all $k \ge 0$:

$$f(x_k + s_k) \le m_k(s_k), \tag{1.34}$$

provided that $\sigma_k \ge (3\chi + L)/2$.

Proof. Due to Assumption 1.2.1(ii) and Assumption 1.2.2, (1.10) with $s = s_k$ holds. Consequently, by virtue of (1.31) and recalling the model definition in (1.14),

$$\begin{aligned}
f(x_k+s_k) &\leq f(x_k)+\nabla f(x_k)s_k+\frac{1}{2}s_k^\top\nabla^2 f(x_k)s_k+\frac{1}{6}L\|s_k\|^3\\
&= f(x_k)+\nabla f(x_k)s_k+\frac{1}{2}s_k^\top(\nabla^2 f(x_k)-\overline{\nabla^2 f}(x_k))s_k+\frac{1}{2}s_k^\top\overline{\nabla^2 f}(x_k)s_k+\frac{1}{6}L\|s_k\|^3\\
&\leq f(x_k)+\nabla f(x_k)s_k+\frac{1}{2}s_k^\top\overline{\nabla^2 f}(x_k)s_k+\frac{1}{2}\|s_k\|\|(\nabla^2 f(x_k)-\overline{\nabla^2 f}(x_k))s_k\|+\frac{1}{6}L\|s_k\|^3\\
&\leq f(x_k)+\nabla f(x_k)s_k+\frac{1}{2}s_k^\top\overline{\nabla^2 f}(x_k)s_k+\frac{1}{2}\left(\chi+\frac{L}{3}\right)\|s_k\|^3\\
&= \widehat{T}_2(x_k,s)+\frac{1}{2}\left(\chi+\frac{L}{3}\right)\|s_k\|^3 \qquad (1.35)\\
&= m_k(s_k)-\frac{1}{3}\sigma_k\|s_k\|^3+\frac{1}{2}\left(\chi+\frac{L}{3}\right)\|s_k\|^3.
\end{aligned}$$

The thesis then follows provided that

$$\frac{1}{2}\left(\chi+\frac{L}{3}\right)-\frac{\sigma_k}{3}\leq 0,$$

i.e., $\sigma_k \geq (3\chi+L)/2$.

□

Remark 3. Recalling the model definition (1.14), such that $m_k(s)=f(x_k)$ when s is null, we remark that (1.34) and (1.17) imply that

$$f(x_k+s_k)\leq m_k(s_k)<m_k(0)=f(x_k),$$

resulting in a reduction of the objective function f when moving from x_k to the new iterate x_k+s_k.

We now assure that the updating rule in (1.20) is such that the regularisation parameter σ_k remains bounded above through all iterations, providing the termination of Algorithm 2.

Lemma 5. With reference to Algorithm 2, let Assumption 1.2.1(ii), Assumption 1.2.2 and Assumption 1.2.3 hold. For all $k\geq 0$,

$$\sigma_k \leq \sigma_{\max} \overset{\text{def}}{=} \max\left[\sigma_0, \gamma_3\frac{3\chi+L}{2(1-\eta_2)}\right], \qquad (1.36)$$

where σ_0 and γ_3 are the constants defined in (1.8).

Proof. We first derive conditions on σ_k ensuring a very successful iteration. From (1.35) we have that

$$f(x_k+s_k)-\widehat{T}_2(x_k,s_k)\leq \frac{1}{2}\left(\chi+\frac{L}{3}\right)\|s_k\|^3. \qquad (1.37)$$

Consequently, using (1.27) and recalling (1.19), we then deduce that

$$1-\rho_k \leq \frac{f(x_k+s_k)-\widehat{T}_2(x_k,s_k)}{\widehat{T}_2(x_k,0)-\widehat{T}_2(x_k,s_k)}\leq\frac{3}{2\sigma_k}\left(\chi+\frac{L}{3}\right)$$

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

$$\frac{3}{2\sigma_k}\left(\chi+\frac{L}{3}\right)\leq 1-\eta_2,$$

which is equivalent to

$$\sigma_k \geq \frac{3\chi + L}{2(1-\eta_2)} \overset{\text{def}}{=} \sigma^*. \tag{1.38}$$

To sum up, according to the updating rule (1.20), we have that if (1.38) holds, then $\rho_k \geq \eta_2$ and $\sigma_{k+1} \leq \sigma_k$ (σ_k is not increased); otherwise it can happen that $\eta_1 \leq \rho_k < \eta_2$ or $\rho_k < \eta_1$, obtaining that $\sigma_{k+1} \geq \sigma_k$ (σ_k not decreased) with $\sigma_{k+1} < \gamma_3\sigma^*$. In fact, by virtue of (1.20), $\rho_k < \eta_2$ and $\sigma_{k+1} \geq \gamma_3\sigma^*$ would imply that $\sigma_k \geq \frac{\gamma_3}{t}\sigma^*$, $1 \leq t \leq \gamma_3$, and, hence, $\sigma_k \geq \sigma^*$, contradicting the inequality $\rho_k < \eta_2$. The bound in (1.36) then follows by taking into account the generality of σ_0. $\square$

Our next step, very much in the line of the theory proposed in [41, 22], is to show that the step length cannot be arbitrarily small compared with the gradient of the objective function f at the trial point $x_k + s_k$.

Lemma 6. Let Assumption 1.2.1(ii), Assumption 1.2.2 and Assumption 1.2.3 hold. Then,

$$\|s_k\| \geq \sqrt{\zeta^* \|\nabla f(x_k + s_k)\|}, \tag{1.39}$$

for some positive ζ^*, when s_k satisfies (1.18).

Proof. The Taylor's expansion (1.9) implies

$$\nabla f(x_k + s) = \nabla f(x_k) + \nabla^2 f(x_k)s + 2\int_0^1 (1-\tau)(\nabla^2 f(x_k + \tau s) - \nabla^2 f(x_k))s\, d\tau. \tag{1.40}$$

Recalling the $\widehat{T}_2(x_k, s)$ definition in (1.14), we also note that

$$\nabla_s \widehat{T}_2(x_k, s) = \nabla f(x_k) + \overline{\nabla^2 f}(x_k)s.$$

Consequently, using Assumption 1.2.2 and Assumption 1.2.3, we derive

$$\begin{aligned}
\|\nabla f(x_k + s_k) - \nabla_s \widehat{T}_2(x_k, s_k)\| &= \|\nabla f(x_k + s_k) - \nabla f(x_k) - \overline{\nabla^2 f}(x_k)s_k\| \\
&\leq \|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s_k\| \\
&\quad + 2\int_0^1 (1-\tau)\|(\nabla^2 f(x_k + \tau s_k) - \nabla^2 f(x_k))s_k\|\, d\tau \\
&\leq \chi\|s_k\|^2 + 2\int_0^1 L\tau(1-\tau)\|s_k\|^2\, d\tau \\
&\leq \left(\chi + \frac{L}{3}\right)\|s_k\|^2.
\end{aligned} \tag{1.41}$$

In addition, recalling (1.15),

$$\begin{aligned}
\nabla f(x_k + s_k) &= \nabla f(x_k + s_k) - \nabla_s \widehat{T}_2(x_k, s_k) + \nabla_s \widehat{T}_2(x_k, s_k) + \sigma_k\|s_k\|s_k - \sigma_k\|s_k\|s_k \\
&= \nabla f(x_k + s_k) - \nabla_s \widehat{T}_2(x_k, s_k) + \nabla_s m(s_k) - \sigma_k\|s_k\|s_k,
\end{aligned} \tag{1.42}$$

yielding

$$\begin{aligned}
\|\nabla f(x_k + s_k)\| &\leq \|\nabla f(x_k + s_k) - \nabla_s \widehat{T}_2(x_k, s_k)\| + \|\nabla_s m(s_k)\| + \sigma_k\|s_k\|^2 \\
&\leq \left(\chi + \frac{L}{3} + \theta + \sigma_{\max}\right)\|s_k\|^2,
\end{aligned}$$

in which we have used (1.41)–(1.42), (1.18) and (1.36). The claim is then given defining

$$\zeta^* \overset{\text{def}}{=} \left(\chi + \frac{L}{3} + \theta + \sigma_{\max}\right)^{-1} > 0.$$

□

We now bound the number of unsuccessful iterations as a function of the number of successful ones. The proof is the same as [22, Lemma 2.4] (see also [41, Theorem 2.1]) and it is here reported for clarity of presentation. To do so, let us introduce the disjointed sets $\mathcal{S}_k$ and $\mathcal{U}_k$ for classifying successful (or very successful) and unsuccessful iterations of Algorithm 2, between 0 and k, defined as

$$\mathcal{S}_k \overset{\text{def}}{=} \{\, 0 \le j \le k \;\mid \rho_j \ge \eta_1 \,\}, \qquad \mathcal{U}_k \overset{\text{def}}{=} \{\, 0 \le j \le k \;\mid \rho_j < \eta_1 \,\}. \tag{1.43}$$

Lemma 7. The mechanism of Algorithm 2 ensures that, if $\sigma_k \le \sigma_{\max}$ for some $\sigma_{\max} > 0$, then the total number of iterations $|\mathcal{S}_k| + |\mathcal{U}_k|$ is bounded above by

$$|\mathcal{S}_k| \left(1 + \frac{|\log \gamma_1|}{\log \gamma_2}\right) + \frac{1}{\log \gamma_2} \log\left(\frac{\sigma_{\max}}{\sigma_0}\right). \tag{1.44}$$

Proof. The regularisation parameter update (1.20) gives that, for each k,

$$\begin{aligned} \gamma_1\sigma_j &\le \max(\gamma_1\sigma_j, \sigma_{\min}) \le \sigma_{j+1}, \quad j \in \mathcal{S}_k, \\ \gamma_2\sigma_j &\le \sigma_{j+1}, \quad j \in \mathcal{U}_k. \end{aligned}$$

Thus we deduce inductively that

$$\sigma_0 \gamma_1^{|\mathcal{S}_k|} \gamma_2^{|\mathcal{U}_k|} \le \sigma_k.$$

We therefore obtain, using the hypothesis $\sigma_k \le \sigma_{\max}$, that

$$|\mathcal{S}_k| \log \gamma_1 + |\mathcal{U}_k| \log \gamma_2 \le \log\left(\frac{\sigma_{\max}}{\sigma_0}\right),$$

which implies that

$$|\mathcal{U}_k| \le -|\mathcal{S}_k| \frac{\log \gamma_1}{\log \gamma_2} + \frac{1}{\log \gamma_2} \log\left(\frac{\sigma_{\max}}{\sigma_0}\right),$$

since $\gamma_2 > 1$. The desired result (1.44) then follows from the inequality $\gamma_1 < 1$ given by (1.8). □

Using all the above results, we are now in position to state our main evaluation complexity result.

Theorem 8. (Worst-case evaluation complexity of Algorithm 2). With reference to Algorithm 2, let Assumption 1.2.1, Assumption 1.2.2 and Assumption 1.2.3 hold. Then, given $\epsilon_1 > 0$, Algorithm 2 takes at most

$$\left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor \tag{1.45}$$

successful iterations, each involving one evaluation of f, of its gradient and approximate Hessian, and at most

$$\left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor \left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right) \tag{1.46}$$

total iterations to produce an iterate x^* such that $\|\nabla f(x^*)\| \le \epsilon_1$, where $\sigma_{\max}$ is defined in (1.36), with

$$\kappa_s \stackrel{\text{def}}{=} \frac{3}{\eta_1 \sigma_{\min} {\zeta^*}^{3/2}} > 0$$

and ζ^* as in Lemma 6.

Proof. The proof parallels the one of [22, Theorem 2.5]. First note that the we are under the validity of the assumptions of Lemma 5, so the bound (1.36) holds. At each successful iteration at each successful iteration before termination, from (1.27) and (1.39), we have that

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\ge \eta_1(\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)) \\ &\ge \eta_1 \frac{\sigma_{\min}}{3}\|s_k\|^3 \\ &\ge \eta_1 \frac{\sigma_{\min}}{3} {\zeta^*}^{3/2}\|\nabla f(x_k + s_k)\|^{3/2} \\ &\stackrel{\text{def}}{=} \kappa_s^{-1}\|\nabla f(x_k + s_k)\|^{3/2}. \end{aligned} \tag{1.47}$$

Consequently, before termination (1.2), it holds

$$f(x_k) - f(x_k + s_k) \ge \kappa_s^{-1}\|\nabla f(x_k + s_k)\|^{3/2} \ge \kappa_s^{-1}\epsilon_1^{3/2},$$

implying that

$$f(x_0) - f_{low} \ge f(x_0) - f(x_{k+1}) = \sum_{j \in \mathcal{S}_k} (f(x_j) - f(x_j + s_j)) \ge |\mathcal{S}_k| \kappa_s^{-1}\epsilon_1^{3/2},$$

in which we have used Assumption 1.2.1(i) and, hence,

$$|\mathcal{S}_k| \le \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor. \tag{1.48}$$

Having proved (1.45), we can directly invoked Lemma 7 to obtain the upper bound (1.46) on the total number of iterations $|\mathcal{S}_k| + |\mathcal{U}_k|$, yielding

$$\begin{aligned} |\mathcal{S}_k| + |\mathcal{U}_k| &\le |S_k|\left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right) \\ &\le \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor \left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right), \end{aligned} \tag{1.49}$$

in which we have used (1.48) to derive the last inequality. □

We have shown that, under suitable smoothness assumptions, an ϵ_1-approximate stationary point is found by Algorithm 2 in at most $\mathcal{O}(\epsilon_1^{-3/2})$ iterations and evaluations of the objective function, its gradient and approximate Hessian. The result in Theorem 8 is made possible by the introduction in [22] of two main innovations:

- a weaker termination conditions on the model minimisation subproblem (no global optimisation is required at all);
- a reformulation of the ratio of achieved versus predicted decreases (i.e. (1.19)), where the model is limited to the Taylor's approximation.

Of course, each iteration of the proposed algorithm requires the approximate minimisation of a typically nonconvex regularised cubic model, but this minimisation does not involve additional computation of the objective function of the original problem or of its derivatives and, therefore, its cost does not affect the evaluation complexity of Algorithm 2, as already noticed at the beginning of the subsection. Which numerical procedure is better for this task is not the aim of this thesis (for instance, one might think of applying an efficient first-order method on the model), but a guidance to do that is given in Section 1.3.

1.2.1 Convergence to first-order critical points

As a direct consequence of the complexity analysis presented in the previous section, we have that

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0. \tag{1.50}$$

Furthermore, let us introduce the following sets to directly address the set of successful/very successful and unsuccessful iterations:

$$\begin{aligned} \mathcal{S} &= \{k \geq 0 |\, k \text{ successful or very successful}\}, \\ \mathcal{U} &= \{k \geq 0 |\, k \text{ unsuccessful}\}. \end{aligned}$$

We can now parallel [40, Lemma 5.1] in order to turn the liminf convergence in (1.50) into a lim-type convergence result, which is the aim of this subsection.

Theorem 9. Let Assumption 1.2.1, Assumption 1.2.2 and Assumption 1.2.3 hold. Then, the step s_k and the iterates $\{x_k\}_{k\geq 0}$ generated by Algorithm 2 satisfy

$$\|s_k\| \to 0, \quad \text{as } k \to \infty,\ k \in \mathcal{S} \tag{1.51}$$

and

$$\|\nabla f(x_k)\| \to 0, \quad \text{as } k \to \infty. \tag{1.52}$$

Proof. By (1.27),

$$f(x_k) - f(x_{k+1}) \geq \eta_1(\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)) \geq \eta_1 \frac{\sigma_{\min}}{3}\|s_k\|^3,\ k \in \mathcal{S}.$$

Since, by Assumption 1.2.1(i), f is bounded below by f_{low}, one has that

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

which implies the convergence of the series $\sum_{k=0,\ k\in\mathcal{S}}^{\infty} \|s_k\|^3$ and, hence,

$$\|s_k\|^3 \to 0, \quad \text{as } k \to \infty,\ k \in \mathcal{S},$$

giving the first claim of the thesis.
As for $\|\nabla f(x_k)\|$, Lemma 6 ensures that

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

This fact, along with $\nabla f(x_{k+1}) = \nabla f(x_k)$ at unsuccessful iterations (i.e. for $k \in \mathcal{U}$), provides the vanishing of the sequence $\{\|\nabla f(x_k)\|\}_{k\geq 0}$, as $k \to \infty$. □

The optimal complexity result given by Theorem 8, implying (1.52) and here proved for the basic ARC scheme described by Algorithm 2, will be the common thread for the extensions of the ARC framework considered in the following chapters of this thesis. Under suitable assumptions, we will analyse in the next subsection its generalisation to the case of second-order critical points, in which quadratic local rate of convergence will be proved.

An intermediate result, showing local quadratic rate of convergence to first-order critical points, was shown in [40] for Algorithm 2 with the modified steps in Algorithm 3, under a set of stricter assumptions compared to the ones required by Theorem 9. The result, given by [40, Corollary 4.10], is reported below.

Theorem 10. [40, Corollary 4.10] With reference to Algorithm 2 with the modified steps in Algorithm 3, let Assumption 1.2.1(ii), Assumption 1.2.2 and Assumption 1.2.3 hold. Assume also that:

i) $\nabla f(x)$ is Lipschitz continuous in an open set containing all the iterates $\{x_k\}_{k\geq 0}$;

ii) the step s_k is required to satisfy $\nabla f(x_k)^\top s_k + s_k^\top \overline{\nabla^2 f}(x_k) s_k + \sigma_k \|s_k\|^3 = 0$, choosing θ in (1.18) as $\theta_k = \kappa_\theta \min[1, \|s_k\|]$, for $k \geq 0$ and some $\kappa_\theta \in (0,1)$;

iii) the gradient is uniformly continuous on the sequence of iterates $\{x_k\}_{k\geq 0}$, namely

$$\|\nabla f(x_{t_i}) - \nabla f(x_{\ell_i})\| \to 0, \text{ whenever } \|x_{t_i} - x_{\ell_i}\| \to 0,\ i \to \infty;$$

iv) the level of resemblance between the approximate Hessian $\overline{\nabla^2 f}(x_k)$ and the true one $\nabla^2 f(x_k)$ at iteration k is such that

$$\frac{\|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\|}{\|s_k\|} \to 0, \text{ whenever } \|\nabla f(x_k)\| \to 0;$$

v) the norms of the Hessian approximations are uniformly bounded above: $\|\overline{\nabla^2 f}(x_k)\| \leq \kappa_B$, for all $k \geq 0$ and some $\kappa_B \geq 0$;

vi) $x_k \to x^*$, as $k \to \infty$, with $\nabla^2 f(x^*)$ positive definite.

Then, $\{\nabla f(x_k)\}_{k\geq 0}$ converges to zero and $\{x_k\}_{k\geq 0}$ converges to x^* quadratically, as $k \to \infty$.

1.2.2 Convergence to second-order critical points

After having exploited convergence to first-order critical points, we now carry on the analysis focusing on the convergence of the sequence generated by Algorithm 2 to second-order critical points, i.e. to a points x^* satisfying (1.3). In so doing, the two following scenarios are considered.

- We study the asymptotic behaviour of $\{x_k\}_{k\geq 0}$ under the assumption that the approximate Hessian $\overline{\nabla^2 f}(x_k)$ at x_k becomes positive definite along a converging subsequence of $\{x_k\}_{k\geq 0}$. In this situation, local quadratic rate of convergence is restored provided that, as in [40], the step satisfies the Cauchy condition (1.28).

- We analyse the case in which the Hessian approximation $\overline{\nabla^2 f}(x_k)$ is not convex, obtaining a second-order complexity bound in accordance with the study in [38].

To address the fist claim, let us report the following preliminary assumption and results.

Assumption 1.2.4. *With reference to problem* (1.1)*, for all* $k \geq 0$ *and some* $\kappa_B \geq 0$ *it holds*

$$\|\overline{\nabla^2 f}(x_k)\| \leq \kappa_B.$$

Lemma 11. [91, Lemma 4.10]. Let x^* be an isolated limit point of a sequence $\{x_k\}_{k\geq 0}$ in $\mathbb{R}^n$. If $\{x_k\}_{k\geq 0}$ does not converge, then there is a subsequence $\{x_{\ell_j}\}_{j\geq 0}$ which converges to x^* and $\epsilon > 0$ such that $\|x_{\ell_j+1} - x_{\ell_j}\| \geq \epsilon$.

Proof. [91, Lemma 4.10]. Choose $\epsilon > 0$ so that if $\|x - x^*\| \leq \epsilon$ and x is a limit point of $\{x_k\}_{k\geq 0}$, then $x = x^*$. If $\|x_{k_j} - x^*\| \leq \epsilon$, then define ℓ_j by

$$\ell_j = \max\{\ell \mid \|x_i - x^*\| \leq \epsilon,\ i = k_j, \cdots, \ell\}.$$

In this manner, a subsequence $\{x_{\ell_j}\}_{j\geq 0}$ is defined such that

$$\|x_{\ell_j} - x^*\| \leq \epsilon, \quad \|x_{\ell_j+1} - x^*\| > \epsilon.$$

It follows that $\{x_{\ell_j}\}_{j\geq 0}$ converges to x^* and thus $\|x_{\ell_j} - x^*\| \leq \frac{\epsilon}{2}$ for all ℓ_j sufficiently large. Hence,

$$\|x_{\ell_j+1} - x_{\ell_j}\| \geq \|x_{\ell_j+1} - x^*\| - \|x_{\ell_j} - x^*\| \geq \frac{\epsilon}{2},$$

as desired. □

Lemma 12. [40, Lemma 2.1]. Suppose that the step s_k satisfies the Cauchy condition (1.28)–(1.29). Then, for $k \geq 0$, we have that

$$\begin{aligned} f(x_k) - m_k(s_k) &\geq f(x_k) - m_k(s_k^C) \geq \\ &\geq \frac{\|\nabla f(x_k)\|^2}{6\sqrt{2}\max\left(1 + \|\overline{\nabla^2 f}(x_k)\|, 2\sqrt{\sigma_k\|\nabla f(x_k)\|}\right)} \\ &= \frac{\|\nabla f(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|\nabla f(x_k)\|}{1 + \|\overline{\nabla^2 f}(x_k)\|}, \frac{1}{2}\sqrt{\frac{\|\nabla f(x_k)\|}{\sigma_k}}\right). \end{aligned} \tag{1.53}$$

Proof. [40, Lemma 2.1]. Due to (1.28) and since the first inequality in (1.53) is straightforward, it remains to show the second inequality in (1.53). For any $\alpha \geq 0$, using the

Cauchy-Schwarz inequality, we have that

$$\begin{aligned} m_k(s_k^C) - f(x_k) &\leq m_k(-\alpha\nabla f(x_k)) - f(x_k) \\ &= -\alpha\|\nabla f(x_k)\|^2 + \frac{1}{2}\alpha^2\nabla f(x_k)^\top\overline{\nabla^2 f}(x_k)\nabla f(x_k) + \frac{1}{3}\alpha^3\sigma_k\|\nabla f(x_k)\|^3 \\ &\leq \alpha\|\nabla f(x_k)\|^2\left(-1 + \frac{1}{2}\alpha\|\overline{\nabla^2 f}(x_k)\| + \frac{1}{3}\alpha^2\sigma_k\|\nabla f(x_k)\|\right). \end{aligned} \tag{1.54}$$

Now $m_k(s_k^C) \leq f(x_k)$, provided that

$$-1 + \frac{1}{2}\alpha\|\overline{\nabla^2 f}(x_k)\| + \frac{1}{3}\alpha^2\sigma_k\|\nabla f(x_k)\| \leq 0$$

and $\alpha \geq 0$ in the latter two inequalities, being equivalent to

$$\alpha \in [0, \overline{\alpha}_k], \qquad \overline{\alpha}_k \overset{\text{def}}{=} \frac{3}{2\sigma_k\|\nabla f(x_k)\|}\left[-\frac{1}{2}\|\overline{\nabla^2 f}(x_k)\| + \sqrt{\frac{1}{4}\|\overline{\nabla^2 f}(x_k)\|^2 + \frac{4}{3}\sigma_k\|\nabla f(x_k)\|}\right].$$

Furthermore, we can express $\overline{\alpha}_k$ as

$$\overline{\alpha}_k = 2\left(\frac{1}{2}\|\overline{\nabla^2 f}(x_k)\| + \sqrt{\frac{1}{4}\|\overline{\nabla^2 f}(x_k)\|^2 + \frac{4}{3}\sigma_k\|\nabla f(x_k)\|}\right)^{-1}.$$

Letting

$$\theta_k \overset{\text{def}}{=} \left[\sqrt{2}\max\left(1 + \|\overline{\nabla^2 f}(x_k)\|, 2\sqrt{\sigma_k\|\nabla f(x_k)\|}\right)\right]^{-1} \tag{1.55}$$

and employing the inequalities

$$\begin{aligned} \sqrt{\frac{1}{4}\|\overline{\nabla^2 f}(x_k)\|^2 + \frac{4}{3}\sigma_k\|\nabla f(x_k)\|} &\leq \frac{1}{2}\|\overline{\nabla^2 f}(x_k)\| + \frac{2}{\sqrt{3}}\sqrt{\sigma_k\|\nabla f(x_k)\|} \\ &\leq 2\max\left(\frac{1}{2}\|\overline{\nabla^2 f}(x_k)\|, \frac{2}{\sqrt{3}}\sqrt{\sigma_k\|\nabla f(x_k)\|}\right) \\ &\leq \sqrt{2}\max\left(1 + \|\overline{\nabla^2 f}(x_k)\|, 2\sqrt{\sigma_k\|\nabla f(x_k)\|}\right) \end{aligned}$$

and

$$\frac{1}{2}\|\overline{\nabla^2 f}(x_k)\| \leq \max\left(1 + \|\overline{\nabla^2 f}(x_k)\|, 2\sqrt{\sigma_k\|\nabla f(x_k)\|}\right),$$

it follows that $0 < \theta_k \leq \overline{\alpha}_k$. Thus, substituting the value of θ_k in the last inequality in (1.54), we obtain that

$$\begin{aligned} m_k(s_k^C) - f(x_k) \leq\ & \frac{\|\nabla f(x_k)\|^2}{\sqrt{2}\max\left(1 + \|\overline{\nabla^2 f}(x_k)\|, 2\sqrt{\sigma_k\|\nabla f(x_k)\|}\right)} \\ & \cdot\left(-1 + \frac{1}{2}\theta_k\|\overline{\nabla^2 f}(x_k)\| + \frac{1}{3}\theta_k^2\sigma_k\|\nabla f(x_k)\|\right) \leq 0. \end{aligned} \tag{1.56}$$

It now follows from the definition of θ_k in (1.55) that $\theta_k\|\overline{\nabla^2 f}(x_k)\| \leq 1$ and $\theta_k^2\sigma_k\|\nabla f(x_k)\| \leq 1$, so that the expression in the curly brackets in (1.56) is bounded above by $-1/6$. This and (1.56) imply the second inequality in (1.53). □

Remark 4. We have already noticed that an important point in the design of ARC-type algorithms is the well definition of the per-iteration decrease ratio ρ_k (see, e.g., (1.19) in Algorithm 2). This property has been highlighted for Algorithm 2 and [22, Algorithm 7] in Remark 2, since its denominator cannot be null. For the sake of completeness, let us now prove that the denominator of the corresponding decrease ratio (1.30) in [40, Algorithm

2.1] (i.e. Algorithm 2 with the modified steps in Algorithm 3) is strictly positive as well. We first note that the Cauchy point s_k^C required at Step 2 of Algorithm 3 is the global minimiser of $m_k(s)$ in (1.14) over the subspace generated by $\{-\nabla f(x_k)\}$, implying that

$$\nabla_s m_k(s_k)^\top s_k = \nabla f(x_k)^\top s_k + s_k^\top \overline{\nabla^2 f}(x_k) s_k + \sigma_k \|s_k\|^3 = 0, \tag{1.57}$$

$$s_k^\top \overline{\nabla^2 f}(x_k) s_k + \sigma_k \|s_k\|^3 \geq 0. \tag{1.58}$$

are satisfied with $s_k = s_k^C$ (see, e.g., Lemma 17). Recalling the model definition (1.14) and condition (1.28), then the denominator of ρ_k in (1.30) satisfies the equality:

$$\begin{aligned} f(x_k) - m_k(s_k) &\geq f(x_k) - m_k(s_k^C) = -\nabla f(x_k)^\top s_k^C - \frac{1}{2}(s_k^C)^\top \overline{\nabla^2 f}(x_k) s_k^C - \frac{1}{3}\sigma_k \|s_k^C\|^3 \\ &= -\nabla f(x_k)^\top s_k^C - \left(1 - \frac{1}{2}\right)(s_k^C)^\top \overline{\nabla^2 f}(x_k) s_k^C - \left(1 - \frac{2}{3}\right)\sigma_k \|s_k^C\|^3 \\ &= \frac{1}{2}(s_k^C)^\top \overline{\nabla^2 f}(x_k) s_k^C + \frac{2}{3}\sigma_k \|s_k^C\|^3 \qquad (1.59) \\ &\geq \left(-\frac{1}{2} + \frac{2}{3}\right)\sigma_k \|s_k^C\|^3 = \frac{1}{6}\sigma_k \|s_k^C\|^3. \qquad (1.60) \end{aligned}$$

If

$$\nabla f(x_k) \neq 0, \quad \text{for all } k \geq 0, \tag{1.61}$$

then (1.59) implies $s_k^C \neq 0$. To see this, assume $s_k^C = 0$. Then, (1.59) gives $f(x_k) = m_k(s_k^C)$, contradicting

$$f(x_k) - m_k(s_k^C) > 0, \quad \text{for all } k \geq 0,$$

which follows from (1.53) and (1.61). The proof is thus concluded because whenever $s_k^C \neq 0$, then (1.60) ensures that $f(x_k) - m_k(s_k) > 0$ and, hence, ρ_k in (1.30) is well defined, since $\sigma_k \geq \sigma_{\min} > 0$ for all $k \geq 0$.

We can now proceed in the study of the convergence to second-order stationary points by stating the following theorem.

Theorem 13. Let Assumption 1.2.1(i), Assumption 1.2.2 and (1.32) hold. Suppose that $\{x_{k_i}\}_{k_i \geq 0}$ is a subsequence of successful iterates converging to some x^* and that the approximate Hessians $\overline{\nabla^2 f}(x_{k_i})$ is 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.

$ii)$ Further assume the validity of Assumption 1.2.4. If s_k satisfies the Cauchy condition (1.28)–(1.29) for all $k \geq 0$, then all the iterations are eventually successful and $\{x_k\}_{k \geq 0}$ converges to x^* quadratically.

Proof. We first notice that, due to (1.32), Assumption 1.2.3 holds.

$i)$ From (1.32) and (1.51), we note that

$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \chi \|s_k\| \to 0, \quad k \to \infty,\ k \in \mathcal{S}. \tag{1.62}$$

From standard perturbation results on the eigenvalues of symmetric matrices and taking into account the convergence of $\{x_{k_i}\}$ to x^*, it then follows that $\nabla^2 f(x^*)$ is positive definite (hence invertible). Therefore (see [77]), x^* is an isolated limit point and the claim $i)$ is proved by virtue of (1.51) and Lemma 11.

$ii)$ From the convergence of $\{x_k\}_{k \geq 0}$ to x^*, (1.62) and the positive definitiveness of

$\nabla^2 f(x^*)$, it follows that

$$\lambda_{\min}(\overline{\nabla^2 f}(x_k)) \geq \underline{\lambda} > 0, \quad \forall k \in \mathcal{S} \text{ sufficiently large.} \tag{1.63}$$

Then, recalling that $\overline{\nabla^2 f}(x_k)$ is not modified along unsuccessful iterations, we have that

$$\lambda_{\min}(\overline{\nabla^2 f}(x_k)) \geq \underline{\lambda}, \quad \forall k \text{ sufficiently large.} \tag{1.64}$$

Using (1.18) and recalling (1.15), we have that

$$\begin{aligned}
\theta\|\nabla f(x_k)\| &\geq \|\nabla_s m_k(s_k)\| \\
&= \left\| \nabla f(x_k) + \overline{\nabla^2 f}(x_k)s_k + \sigma_k s_k \|s_k\| \right\| \\
&\geq \|(\overline{\nabla^2 f}(x_k) + \sigma_k\|s_k\| I_n)s_k\| - \|\nabla f(x_k)\| \\
&\geq \frac{\|s_k\|}{\left\|(\overline{\nabla^2 f}(x_k) + \sigma_k\|s_k\|I_n)^{-1}\right\|} - \|\nabla f(x_k)\|,
\end{aligned} \tag{1.65}$$

where the last inequality follows thanks to the fact that for each nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and vector $x \in \mathbb{R}^n$, $\|s\| = \|I_n s\| = \|A^{-1}As\| \leq \|A^{-1}\|\|As\|$, giving $\|As\| \geq \|s\|/\|A^{-1}\|$. Moreover, recalling (1.64) and taking into account that

$$\left\|(\overline{\nabla^2 f}(x_k) + \sigma_k\|s_k\|I_n)^{-1}\right\| = \frac{1}{\lambda_{\min}(\overline{\nabla^2 f}(x_k)) + \sigma_k\|s_k\|} \leq \frac{1}{\underline{\lambda}},$$

inequality (1.65) yields

$$\|s_k\| \leq \frac{1+\theta}{\underline{\lambda}}\|\nabla f(x_k)\|, \quad \forall k \text{ sufficiently large} \tag{1.66}$$

and $\|s_k\| \to 0$, as $k \to \infty$, due to (1.52). Moreover, by (1.29), (1.26) and [40, Lemma 2.1],

$$\begin{aligned}
\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) &\geq m_k(0) - m_k(s_k) \\
&\geq \frac{\|\nabla f(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|\nabla f(x_k)\|}{1 + \|\overline{\nabla^2 f}(x_k)\|}, \frac{1}{2}\sqrt{\frac{\|\nabla f(x_k)\|}{\sigma_k}} \right).
\end{aligned} \tag{1.67}$$

Consequently, Assumption 1.2.4 and (1.36) yield

$$\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) \geq \frac{\|\nabla f(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|\nabla f(x_k)\|}{1 + \kappa_B}, \frac{1}{2}\sqrt{\frac{\|\nabla f(x_k)\|}{\sigma_{\max}}} \right).$$

Thus, (1.52) and (1.66) imply that for k sufficiently large,

$$\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) \quad \geq \quad \frac{\|\nabla f(x_k)\|^2}{6\sqrt{2}(1+\kappa_B)} \geq \frac{\underline{\lambda}^2}{6\sqrt{2}(1+\kappa_B)(1+\theta)^2}\|s_k\|^2 \overset{\text{def}}{=} \kappa_c\|s_k\|^2,$$

and, hence, by (1.19) and (1.37),

$$1 - \rho_k = \frac{f(x_k + s_k) - \widehat{T}_2(x_k, s_k)}{\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)} \leq \frac{\frac{1}{2}\left(\chi + \frac{L}{3}\right)\|s_k\|^3}{\kappa_c\|s_k\|^2} \leq \frac{(\chi + L/3)}{2\kappa_c}\|s_k\|.$$

As a result, $\rho_k \to 1$ and the iterations are eventually very successful. Last, (1.66) and (1.39) provide

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

Then, $\{\|\nabla f(x_k)\|\}_{k\geq 0}$ converges quadratically to zero and the quadratic rate of convergence of the sequence $\{x_k\}_{k\geq 0}$ follows in a standard way by means of the Taylor's expansion. □

The assumption on the positive definitiveness of the Hessian approximations close to x^* can be removed restoring the convergence to second-order critical points. To this aim, we need to equip, as in [38], Algorithm 2 with the following stopping criterion

$$\|\nabla f(x_k)\| \leq \epsilon_1 \quad \text{and} \quad \lambda_{\min}(\overline{\nabla^2 f}(x_k)) \geq -\epsilon_2, \quad \epsilon_1,\ \epsilon_2 > 0, \tag{1.68}$$

which represents, as stated the beginning of the chapter, the approximate counterpart of the second-order optimality condition (1.3) with the Hessian matrix at x_k approximated by $\overline{\nabla^2 f}(x_k)$. Moreover, the trial step s_k computed at Step 2 of Algorithm 2 is required to satisfy the following additional condition: if $\overline{\nabla^2 f}(x_k)$ is not positive semidefinite, then

$$m_k(s_k) \leq m_k(s_k^E), \tag{1.69}$$

where s_k^E is the so-called eigenpoint, defined as

$$s_k^E = \alpha_k^E u_k \quad \text{and} \quad \alpha_k^E = \arg\min_{\alpha\geq 0} m_k(\alpha u_k) \tag{1.70}$$

and u_k is an approximation of the eigenvector of $\overline{\nabla^2 f}(x_k)$ associated with its smallest eigenvalue $\lambda_{\min}(\overline{\nabla^2 f}(x_k))$, in the sense that

$$\nabla f(x_k)^\top u_k \leq 0 \quad \text{and} \quad u_k^\top \overline{\nabla^2 f}(x_k) u_k \leq \kappa_{snc}\lambda_{\min}(\overline{\nabla^2 f}(x_k))\|u_k\|^2, \tag{1.71}$$

for some constant $\kappa_{snc} \in (0,1]$. We underline that the minimisation in (1.70) is global over the subspace generated by αu_k and thus (see, e.g., Lemma 17) (1.57)–(1.58) are satisfied with $s_k = s_k^E$.

The above modifications can be helpfully summarised in the following steps.

Algorithm 4 Modified Steps 0–2 of the ARC algorithm (Algorithm 2).

Step 0: Initialisation. Given an initial point x_0, the initial regulariser $\sigma_0 > 0$, the positive accuracy levels ϵ_1, ϵ_2. 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 (1.68) is satisfied, terminate with the approximate solution $\widehat{x} = x_k$. Otherwise, compute the model $m_k(s)$ as defined in (1.14).

Step 2: Step computation. Compute the step s_k by approximately minimising the model $m_k(s)$ with respect to s so that

$$m_k(s) < m_k(0), \qquad \|\nabla_s m_k(s)\| \leq \theta\|s_k\|^2,$$

and (1.69) is fulfilled.

We hereafter refer to Algorithm 2 with the modified steps 0–2 defined as in Algorithm 4 as ARC-II. Of course, the termination criterion adopted at Step 1 does not affect the mechanism for updating σ_k, then the upper bound $\sigma_{\max}$ on σ_k given by (1.36) is still valid.

Let us denote by:

- $\widetilde{\mathcal{S}}_k$, the set of indices of successful iterations of ARC-II whenever $\|\nabla f(x_k)\| > \epsilon_1$ and/or $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) < -\epsilon_2$, i.e., the indices of successful iterations before (1.68) is met;

- $\widetilde{\mathcal{S}}_k^{(1)}$, the set of indices of successful iterations where $\|\nabla f(x_k)\| > \epsilon_1$;
- $\widetilde{\mathcal{S}}_k^{(2)}$, the set of indices of successful iterations where $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) < -\epsilon_2$;
- $\widetilde{\mathcal{U}}_k$, the set of unsuccessful iterations of ARC-II.

It is worth noting that the cardinality of $\widetilde{\mathcal{S}}_k^{(1)}$ is the same as in Algorithm 2 (see Theorem 8), while proceeding as in Theorem 8 the cardinality of $\widetilde{\mathcal{U}}_k$ is still bounded in terms of the number of successful iterations $\widetilde{\mathcal{S}}_k$ (see also [41, Theorem 2.1]). It indeed remains to derive the cardinality of $\widetilde{\mathcal{S}}_k^{(2)}$.

Lemma 14. [38, Lemma 2.8] Let Assumption 1.2.1, Assumption 1.2.2 and Assumption 1.2.3 hold. Suppose that s_k satisfies (1.69)–(1.70). Then, the number of successful iterations of Algorithm ARC-II with $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) < -\epsilon_2$ is bounded above by

$$\left\lfloor \kappa_e \frac{f(x_0) - f_{low}}{\epsilon_2^3} \right\rfloor,$$

with $\kappa_e \overset{\text{def}}{=} \frac{6\sigma_{\max}^2}{\eta_1 \kappa_{\text{snc}}^3}$.

Proof. The proof parallels that of [38, Lemma 2.8]. We first note that (1.57)–(1.58) with $s_k = s_k^E$ imply

$$\begin{aligned} m_k(0) - m_k(s_k^E) &= -\nabla f(x_k)^\top s_k^E - \frac{1}{2}(s_k^E)^\top \overline{\nabla^2 f}(x_k) s_k^E - \frac{\sigma_k}{3}\|s_k^E\|^3 \\ &= \sigma_k \|s_k^E\|^3 + \frac{1}{2}(s_k^E)^\top \overline{\nabla^2 f}(x_k) s_k^E - \frac{\sigma_k}{3}\|s_k^E\|^3 \\ &\geq \left(1 - \frac{1}{2} - \frac{1}{3}\right)\sigma_k \|s_k^E\|^3 = \frac{1}{6}\sigma_k \|s_k^E\|^3. \end{aligned} \tag{1.72}$$

Moreover, using (1.58) with $s_k = s_k^E$ and (1.71),

$$\sigma_k \|s_k^E\| \geq -\frac{(s_k^E)^\top \overline{\nabla^2 f}(x_k) s_k^E}{\|s_k^E\|^2} \geq -\kappa_{snc}\lambda_{\min},$$

which by (1.36) gives

$$\|s_k^E\| \geq \frac{-\kappa_{snc}\lambda_{\min}}{\sigma_k} \geq \frac{-\kappa_{snc}\lambda_{\min}}{\sigma_{\max}}. \tag{1.73}$$

Assume now that $k \in \widetilde{\mathcal{S}}_k^{(2)}$, we have that

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\geq \eta_1(\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)) = \eta_1(m_k(0) - m_k(s_k) + \underbrace{\frac{\sigma_k}{3}\|s_k\|^3}_{>0}) \\ &> \eta_1(m_k(0) - m_k(s_k^E)) \geq \eta_1 \frac{\sigma_k}{6}\|s_k^E\|^3 \\ &\geq \eta_1 \frac{-\kappa_{snc}^3 \lambda_{\min}(\overline{\nabla^2 f}(x_k))^3}{6\sigma_{\max}^2} \\ &> \eta_1 \frac{\kappa_{snc}^3 \epsilon_2^3}{6\sigma_{\max}^2}, \end{aligned}$$

in which we have used (1.26), (1.69), (1.72), (1.73) and recalling that $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) < -\epsilon_2$

for $k \in \widetilde{\mathcal{S}}_k^{(2)}$. Consequently, defining $\kappa_e \overset{\text{def}}{=} \frac{6\sigma_{\max}^2}{\eta_1 \kappa_{\text{snc}}^3}$, before termination it holds

$$f(x_0) - f_{low} \geq f(x_0) - f(x_{k+1}) \geq \sum_{j \in \widetilde{\mathcal{S}}_k^{(2)}} (f(x_j) - f(x_j + s_j)) \geq |\widetilde{\mathcal{S}}_k^{(2)}| \kappa_e^{-1} \epsilon_2^3$$

and, hence,

$$|\widetilde{\mathcal{S}}_k^{(2)}| \leq \kappa_e \frac{f(x_0) - f_{low}}{\epsilon_2^3},$$

which completes the proof. □

We thus conclude that Algorithm ARC-II produces an iterate $x_{\widehat{k}}$ satisfying (1.68) within at most

$$\mathcal{O}\left(\max(\epsilon_1^{-3/2}, \epsilon_2^{-3})\right)$$

iterations, in accordance with the complexity result in [38].

1.3 Guidelines for approximately minimising the cubic model

Despite the complexity and convergence properties of the ARC algorithm derived in the previous subsections, its practical efficiency ultimately relies on the ability of exactly or approximately minimise m_k in (1.14). Which numerical procedure is preferable for this task is beyond the scope of this thesis, but let us end this first chapter giving useful guidelines to approximately minimise the cubic model m_k, after a characterisation of the global minimiser of the cubic model over $\mathbb{R}^n$ is given.

The presented exposition quotes the innovative presentation in [40] and is organised as follows. Though m_k is nonconvex, the first theorem reported from [40] in Subsection 1.3.1 provides a characterisation of its global solutions over $\mathbb{R}^n$, from which the model can be globally minimised using a factorisation of the matrix $\overline{\nabla^2 f}(x_k)$. Nevertheless, this could be less viable when dealing with large-scale optimisation, hence Subsection 1.3.2 studies the cubic model minimisation in a sequence of nested subspaces, while Subsection 1.3.3 considers the computation of cheaper and approximate minimisers of the model, without involving explicit factorisations of $\overline{\nabla^2 f}(x_k)$, but only matrix-vector products.

1.3.1 Characterisation of the minimiser of the cubic model

This subsection is aimed at giving necessary and sufficient optimality conditions for the global minimiser of the cubic model m_k. We will follow the scheme in [40], which is closer to the trust-region approach compared to the analogous one in [95] and [70].

For fixed k, $k \geq 0$, let us rewrite the gradient of the model in (1.15) and its second-order derivative as

$$\nabla_s m_k(s) = \nabla f(x_k) + \overline{\nabla^2 f}(x_k)s + \lambda s, \qquad \nabla_s^2 m_k(s) = \overline{\nabla^2 f}(x_k) + \lambda I_n + \lambda \left(\frac{s}{\|s\|}\right)\left(\frac{s}{\|s\|}\right)^\top, \quad (1.74)$$

letting $\lambda = \sigma_k \|s\|$. The following global optimality result holds true from [40].

Theorem 15. [40, Theorem 3.1]. Any s_k^* is a global minimiser of $m_k(s)$ over $\mathbb{R}^n$ if and only if it satisfies the system of equations

$$\left(\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n\right) s_k^* = -\nabla f(x_k), \tag{1.75}$$

where $\lambda_k^* = \sigma_k \|s_k^*\|$ and $\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n$ is positive semidefinite. If $\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n$ is positive definite, then s_k^* is unique.

Proof. [40, Theorem 3.1]. In this proof, the iteration subscript k is omitted for simplicity. Firstly, let s^* be a global minimiser of $m_k(s)$ over $\mathbb{R}^n$. It follows from (1.74) and the first and second-order necessary optimality conditions at s^* that

$$\nabla f(x) + \left(\overline{\nabla^2 f}(x) + \lambda^* I_n\right) s^* = 0,$$

and hence that (1.75) holds and

$$\omega^\top \left[\overline{\nabla^2 f}(x) + \lambda^* I_n + \lambda^* \left(\frac{s^*}{\|s^*\|}\right)\left(\frac{s^*}{\|s^*\|}\right)^\top\right] \omega \geq 0, \tag{1.76}$$

for all $\omega \in \mathbb{R}^n$. If $s^* = 0$, by $\lambda^* = \sigma\|s\|$ we have $\lambda^* = 0$ and (1.76) implies that $\overline{\nabla^2 f}(x)$ is positive semidefinite, which gives the required result. We indeed just need to consider $s^* \neq 0$. There are two cases. Firstly, suppose that $\omega^\top s^* = 0$. In this case, it immediately follows from (1.76) that

$$\omega^\top \left(\overline{\nabla^2 f}(x) + \lambda^* I_n\right) \omega \geq 0, \tag{1.77}$$

for all ω for which $\omega^\top s^* = 0$. It thus remains to consider vectors ω for which $\omega^\top s^* \neq 0$. Since ω and s^* are not orthogonal, the line $s^* + \alpha\omega$ intersects the ball centered in the origin of radius $\|s^*\|$ at two points, s^* and $u^* \neq s^*$, say, and thus

$$\|u^*\| = \|s^*\|. \tag{1.78}$$

We let $\omega^* = u^* - s^*$ and note that ω^* is parallel to ω. Since s^* is a global minimiser, we immediately have that

$$\begin{aligned}
0 &\leq m(u^*) - m(s^*) \\
&= \nabla f(x)^\top (u^* - s^*) + \frac{1}{2}(u^*)^\top \overline{\nabla^2 f}(x) u^* - \frac{1}{2}(s^*)^\top \overline{\nabla^2 f}(x) s^* + \frac{\sigma}{3}\left(\|u^*\|^3 - \|s^*\|^3\right) \\
&= \nabla f(x)^\top (u^* - s^*) + \frac{1}{2}(u^*)^\top \overline{\nabla^2 f}(x) u^* - \frac{1}{2}(s^*)^\top \overline{\nabla^2 f}(x) s^*,
\end{aligned} \tag{1.79}$$

where the last equality follows from (1.78). But left-multiplication in (1.75) by $(s^* - u^*)^\top$ gives that

$$(u^* - s^*)^\top \nabla f(x) = (s^* - u^*)^\top \overline{\nabla^2 f}(x) s^* + \lambda^* (s^* - u^*)^\top s^*. \tag{1.80}$$

In addition, (1.78) shows that

$$(s^* - u^*)^\top s^* = \frac{1}{2}(s^*)^\top s^* + \frac{1}{2}(u^*)^\top u^* - (u^*)^\top s^* = \frac{1}{2}(\omega^*)^\top \omega^*. \tag{1.81}$$

Thus, combining (1.79)–(1.80), we find that

$$\begin{aligned} 0 &\leq \frac{1}{2}\lambda^*(\omega^*)^\top\omega^* + \frac{1}{2}(u^*)^\top\overline{\nabla^2 f}(x)u^* - \frac{1}{2}(s^*)^\top\overline{\nabla^2 f}(x)s^* \\ &\quad +(s^*)^\top\overline{\nabla^2 f}(x)s^* - (u^*)^\top\overline{\nabla^2 f}(x)s^* \\ &= \frac{1}{2}(\omega^*)^\top\left(\overline{\nabla^2 f}(x) + \lambda^* I_n\right)\omega^*, \end{aligned} \tag{1.82}$$

from which we deduce

$$\omega^\top\left(\overline{\nabla^2 f}(x) + \lambda^* I_n\right)\omega \geq 0, \tag{1.83}$$

for all ω such that $\omega^\top s^* \neq 0$, since ω^* is parallel to ω. Hence, (1.77) and (1.83) together show that $\overline{\nabla^2 f}(x) + \lambda^* I_n$ is positive semidefinite. The uniqueness of s^* when $\overline{\nabla^2 f}(x) + \lambda^* I_n$ is positive definite, hence invertible, follows from (1.75) by *reductio ad absurdum*. We directly refer to [40, Proof of Theorem 3.1] for the sufficiency implication. □

We can now state the following relevant remark, linking the trust-region radius Δ_k of the Trust-Region framework (see, e.g., Algorithm 1) to the regulariser σ_k of ARC (see, e.g., Algorithm 2).

Remark 5. The result in Theorem 15, together with its proof, is definitely similar to those for the trust-region subproblem (1.6), as it can be seen from [52, Corollary 7.2.2], reported below. Therefore, recalling that s_k^* would satisfy (1.75), we have from Theorem 15, when $\|s_k^*\| = \Delta_k$, that

$$\sigma_k = \frac{\lambda_k^*}{\Delta_k}$$

and, hence, *one might interpret the regulariser σ_k of the ARC framework as inversely proportional to the trust-region radius Δ_k.*

Theorem 16. [52, Corollary 7.2.2] With reference to Algorithm 1 (Trust-Region), any global minimiser $s_k^{*,TR}$ of (1.4) subject to $\|s_k^{*,TR}\| \leq \Delta_k$ satisfies the system of equations

$$\left(\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n\right) s_k^{*,TR} = -\nabla f(x_k),$$

where $\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n$ is positive semidefinite and the Lagrange multiplier λ_k^* for the constraint satisfies $\lambda_k^* \geq 0$, $\lambda_k^*(\|s_k^{*,TR}\| - \Delta_k) = 0$. If $\overline{\nabla^2 f}(x_k) + \lambda_k^* I_n$ is positive definite, then $s_k^{*,TR}$ is unique.

Always on the basis of Theorem 15, methodologies to compute the global minimiser s_k^* are reported in [40, Section 6.1]. Following that lines, we here give the main idea, leaving the details in the Appendix (Section A.1). Throughout the remaining part of this section, we drop the major iteration subscript k for convenience. Due to Theorem 15, to compute the global model minimiser of $m_k(s)$ over $\mathbb{R}^n$ we search for a pair (s, λ) such that

$$(\overline{\nabla^2 f}(x) + \lambda I_n)s = -\nabla f(x), \quad \text{and} \quad \lambda^2 = \sigma^2 s^\top s \tag{1.84}$$

and for which $\overline{\nabla^2 f}(x) + \lambda I_n$ is positive semidefinite. As noticed by the authors in [40, 70], we can suppose that, in analogy with the trust-region approach (see, e.g., [52, Section 7.3.1]), $\nabla^2 f(x)$ admits an eigen-decomposition

$$\overline{\nabla^2 f}(x) = U^\top \Lambda U,$$

giving

$$B(\lambda) \overset{\text{def}}{=} \overline{\nabla^2 f}(x) + \lambda I_n = U^\top(\Lambda + \lambda I_n)U, \tag{1.85}$$

where $\Lambda \overset{\text{def}}{=} diag(\lambda_1, \cdots, \lambda_n)$, with $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and U is an orthonormal matrix of associated eigenvectors. Therefore, from Theorem 15, the value of λ we seek must satisfy $\lambda \geq -\lambda_1$ as only then is $B(\lambda)$ positive semidefinite.
Suppose that $\lambda > -\lambda_1$, then $B(\lambda)$ is positive definite and let

$$s(\lambda) = -B(\lambda)^{-1}\nabla f(x) = -U^\top(\Lambda + \lambda I_n)^{-1}U\nabla f(x), \tag{1.86}$$

where the last equality is due to (1.85). But (see (1.75)) the value of λ^* is the solution of the nonlinear equation $\|s(\lambda)\|^2 = \lambda^2/\sigma^2$. The analysis reported in the Appendix (Section A.1) shows that a safeguarded univariated Newton's iteration can be considered for the solution of the nonlinear equation $\|s(\lambda)\|^2 = \lambda^2/\sigma^2$, with $s(\lambda)$ given by (1.86). The application of the Newton's method to such an equation requires the resolution of a sequence of linear systems

$$B(\lambda^{(j)})s = (\overline{\nabla^2 f}(x) + \lambda^{(j)} I_n)s = -\nabla f(x), \tag{1.87}$$

where $\lambda^{(j)}$ is the j-th Newton's iterate.

1.3.2 Cubic model minimisation in a subspace

The computation of the model global minimiser via the Newton's iteration may be computational demanding for large-scale problems, since a sequence of $n \times n$ linear systems (1.87) has to be solved.

As in the case of trust-region methods, a much more useful approach in practice is to compute an approximate global minimiser of $m_k(s)$ which corresponds to globally minimising the model over a sequence of nested Krylov's subspaces, where each subproblem is computationally quite inexpensive (see Subsection 1.3.3). It is important to note that the step computed with this strategy still satisfies (1.57)–(1.58), as stated in the following lemma.

Lemma 17. [40, Lemma 3.2] Let s_k be the global minimiser of m_k over some subspace $\mathcal{L}_k \subseteq \mathbb{R}^n$. Then, s_k satisfies (1.57)–(1.58). Furthermore, letting Q_k denote any orthogonal matrix whose columns form a basis of $\mathcal{L}_k$, we have that

$$Q_k^\top \overline{\nabla^2 f}(x_k) Q_k + \sigma_k \|s_k\| I_n \text{ is positive semidefinite.} \tag{1.88}$$

Proof. [40, Lemma 3.2] Let s_k be the global minimiser of m_k over some $\mathcal{L}_k$, then s_k solves

$$\min_{s \in \mathcal{L}_k} m_k(s). \tag{1.89}$$

Let ℓ denote the dimension of the subspace $\mathcal{L}_k$ and let $Q_k \in \mathbb{R}^{n \times \ell}$ be an orthogonal matrix whose columns form a basis of $\mathcal{L}_k$. Thus, $Q_k^\top Q_k = I_\ell$ and for all $s \in \mathcal{L}_k$, we have $s = Q_k u$, for some $u \in \mathbb{R}^\ell$. Recalling that s_k solves (1.89) and letting

$$s_k = Q_k u_k, \tag{1.90}$$

we have that u_k is the global minimiser of

$$\min_{u \in \mathbb{R}^\ell} m_{k,r}(u) \overset{\text{def}}{=} f(x_k) + (Q_k^\top \nabla f(x_k))^\top u + \frac{1}{2} u^\top Q_k^\top \overline{\nabla^2 f}(x_k) Q_k u + \frac{1}{3}\sigma_k \|u\|^3,$$

where we have used that the Euclidean norm is invariant with respect to multiplication by orthogonal matrices[*]. Applying Theorem 15 to the reduced model $m_{k,r}$ and u_k, it follows that

$$Q_k^\top \overline{\nabla^2 f}(x_k) Q_k u_k + \sigma_k \|u_k\| u_k = -Q_k^\top \nabla f(x_k),$$

and left-multiplication by $u_k^\top \in \mathbb{R}^{1\times\ell}$ gives

$$u_k^\top Q_k^\top \overline{\nabla^2 f}(x_k) Q_k u_k + \sigma_k \|u_k\|^3 = -u_k^\top Q_k^\top \nabla f(x_k),$$

which is the same as (1.57), recalling that (1.90) and the invariancy of the Euclidean norm with respect to multiplication by orthogonal matrices give

$$u_k = Q_k^\top s_k \quad \text{with} \quad \|u_k\| = \|Q_k^\top s_k\| = \|s_k\|. \tag{1.91}$$

Moreover, Theorem 15 implies that $Q_k^\top \overline{\nabla^2 f}(x_k) Q_k + \sigma_k \|u_k\| I_n$ is positive semidefinite. Consequently, due to (1.91), this is (1.88) and also implies

$$u_k^\top Q_k^\top \overline{\nabla^2 f}(x_k) Q_k u_k + \sigma_k \|u_k\|^3 \geq 0,$$

which is, by virtue of (1.91), (1.58). □

Remark 6. As anticipated on page 28, we remark that Theorem 15 implies that the step s_k^E satisfying conditions (1.70) also fulfill (1.57)–(1.58) considered in the hypotheses of Lemma 14, since it is defined (see (1.70)) as the global minimiser of $m_k(s)$ along the subspace generated by αu_k. The same is true for the Cauchy point s_k^C in (1.29), used in Theorem 13, where the global minimisation is made over the subspace generated by $\{-\nabla f(x_k)\}$. On the other hand, requiring that s_k satisfies (1.57) may not necessarily imply (1.28). Nevertheless, when globally minimising m_k over successive subspaces, the Cauchy condition (1.28) can be easily ensured by including $\nabla f(x_k)$ in each of the subspaces. This is the approach considered in [40] where the subspaces generated by Lanczos method naturally include the gradient (see Subection 1.3.3).

1.3.3 Methods for approximately minimising the cubic model

We now describe the procedure outlined in [40] to compute an approximate minimiser of $m_k(s)$ that is the global minimiser over a proper subspace $\mathcal{L}_k \subseteq \mathbb{R}^n$. In particular, the method proposed in [40] requires only Hessian-vector products rather than access to the Hessian itself (a so-called "matrix-free" approach) and, hence, it may be used in principle for large unstructured problems. This approach has been also used in [65] for the Trust-Region method. In both cases, the Lanczos method [63, 64] is used to build an orthogonal basis $\{q_0, \cdots, q_j\}$, $j \geq 0$, for the Krylov's subspace:

$$\mathcal{K}_j \overset{\text{def}}{=} \mathcal{K}(\overline{\nabla^2 f}(x_k), \nabla f(x_k), j) = \operatorname{span}\{\nabla f(x_k), \overline{\nabla^2 f}(x_k)\nabla f(x_k), \cdots, (\overline{\nabla^2 f}(x_k))^j \nabla f(x_k)\}, \tag{1.92}$$

formed by successively applying $\nabla^2 f(x_k)$ to $\nabla f(x_k)$. Let us briefly recall the main facts about the Lanczos iteration.

We start by $\mathcal{K}(\overline{\nabla^2 f}(x_k), \nabla f(x_k), 0) = \operatorname{span}\{q_0\}$, with $q_0 = \nabla f(x_k)/\|\nabla f(x_k)\|$. Thus,

$$\mathcal{K}_j = \mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, j).$$

Suppose now that we have found a suitable orthonormal basis $\{q_s\}_{s=0}^i$ for $\mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, i)$,

[*]For all vectors $u \in \mathbb{R}^\ell$ and orthogonal matrices $Q \in \mathbb{R}^{n\times l}$ it holds: $\|Qu\|^2 = (Qu)^\top(Qu) = u^\top(Q^\top Q)u = u^\top I_\ell u = \|u\|^2$, implying that $\|Qu\| = \|u\|$.

for all $i \in \{0, ..., t\}$ with $t \in \{0, ..., j-1\}$, i.e.

$$\mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, i) = \mathrm{span}\{q_0, ..., q_i\}, \quad q_r^\top q_s = \delta_{rs}, \quad s, r \in \{0, ..., i\}, \quad i \in \{0, ..., t\}.$$

The construction of the next basis vector q_{t+1} such that

$$\mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, t+1) = \mathrm{span}\{q_0, ..., q_t, q_{t+1}\}$$

is based on the fact that q_{t+1} we seek has to lie in the span of the previous $\{q_i\}_{i=0}^t$ and $\overline{\nabla^2 f}(x_k)q_t$. This is due to [52, Lemma 5.2.1], reported below.

Lemma 18. [52, Lemma 5.2.1] Suppose that

$$\mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, i) = \mathrm{span}\{q_0, q_1, ..., q_i\}, \tag{1.93}$$

for $i \in \{0, ..., t+1\}$ and that the q_i are mutually orthonormal. Then,

$$q_{i+1} \in \mathrm{span}\{q_0, q_1, ..., q_i, \overline{\nabla^2 f}(x_k)q_i\}, \tag{1.94}$$

for all $i \in \{0, ..., t\}$.

Proof. The proof, as it appears in [52], is reported in the Appendix (Section A.2). □

As q_{t+1} is required to be orthonormal to $\{q_i\}_{i=0}^t$, it has to contain a nonzero component of $\overline{\nabla^2 f}(x_k)q_t$. It thus suffices to find a vector

$$y_{t+1} = \sum_{i=0}^{t} \theta_{ti} q_i + \overline{\nabla^2 f}(x_k)q_t \tag{1.95}$$

that is orthogonal to q_i, for $i \in \{0, ..., t\}$, and then to normalise y_{t+1}, setting

$$q_{t+1} \overset{\text{def}}{=} \frac{y_{t+1}}{\gamma_{t+1}}, \qquad \gamma_{t+1} \overset{\text{def}}{=} \|y_{t+1}\|. \tag{1.96}$$

At a first sight (1.95) appears to require that all the $\{q_i\}_{i=0}^t$ are available, but it can in practice be computed solely as a linear combination of $\overline{\nabla^2 f}(x_k)q_t$, q_t and q_{t-1}. In fact, given $r \in \{0, ..., t-2\}$, we can use the orthogonality of y_{t+1} and q_r, together with the mutual orthonormality of the $\{q_i\}_{i=0}^t$, to derive, from (1.95), that

$$0 = q_r^\top y_{t+1} = \sum_{i=0}^{t} \theta_{ti} q_r^\top q_i + q_r^\top \overline{\nabla^2 f}(x_k)q_t = \theta_{tr} + q_r^\top \overline{\nabla^2 f}(x_k)q_t. \tag{1.97}$$

As the $\{q_i\}_{i=0}^t$ are mutually orthonormal, (1.94) implies

$$\overline{\nabla^2 f}(x_k)q_r \in \mathrm{span}\{q_0, ..., q_r, q_{r+1}\}$$

and, hence, that

$$q_r^\top \overline{\nabla^2 f}(x_k)q_t = 0,$$

for all $r \in \{0, ..., t-2\}$. Consequently (recall (1.97)), $\theta_{tr} = 0$ for all $r \in \{0, ..., t-2\}$. Combining this latter equalities with (1.95), the definition of q_{t+1} in (1.96) and (1.97), we obtain that

$$y_{t+1} = \gamma_{t+1}q_{t+1} = \overline{\nabla^2 f}(x_k)q_t - q_t^\top \overline{\nabla^2 f}(x_k)q_t q_t - q_{t-1}^\top \overline{\nabla^2 f}(x_k)q_t q_{t-1}. \tag{1.98}$$

This relationship leads to an alternatine expression for γ_{t+1} in (1.96). To this purpose, taking the inner product of (1.98) with q_{t+1} and using once again the mutual orthonormality of $\{q_i\}_{i=0}^{t+1}$ gives

$$\begin{aligned}\gamma_{t+1} &= \gamma_{t+1} q_{t+1}^\top q_{t+1}\\ &= q_{t+1}^\top \overline{\nabla^2 f}(x_k) q_t - (q_t^\top \overline{\nabla^2 f}(x_k) q_t)(q_{t+1}^\top q_t) - (q_{t-1}^\top \overline{\nabla^2 f}(x_k) q_t)(q_{t+1}^\top q_{t-1})\\ &= q_{t+1}^\top \overline{\nabla^2 f}(x_k) q_t.\end{aligned}$$

Therefore, $\gamma_t = q_t^\top \overline{\nabla^2 f}(x_k) q_{t-1} = q_{t-1}^\top \overline{\nabla^2 f}(x_k) q_t$ and we can rewrite (1.95) as

$$y_{t+1} = \gamma_{t+1} q_{t+1} = \overline{\nabla^2 f}(x_k) q_t - \delta_t q_t - \gamma_t q_{t-1}, \tag{1.99}$$

setting $\delta_t \overset{\text{def}}{=} q_t^\top \overline{\nabla^2 f}(x_k) q_t$.

Summarising, we may compute orthonormal bases of the Krylov's subspace $\mathcal{K}_j$, for $j \geq 0$, with the following algorithm.

Algorithm 5 The Lanczos method [52] for an orthonormal basis of $\mathcal{K}_t$, $t \geq 0$.

```
Given $\nabla f(x_k)$, set $y_0 = \nabla f(x_k)$, $q_{-1} = 0$.
For $t = 0, 1, ...$:
    1. Set $\gamma_t = \|y_t\|$.
    2. Set $q_t = y_t/\gamma_t$.
    3. Set $\delta_t = q_t^\top \overline{\nabla^2 f}(x_k) q_t$.
    4. Compute $y_{t+1} = \overline{\nabla^2 f}(x_k) q_t - \delta_t q_t - \gamma_t q_{t-1}$.
```

It is thus evident that the major cost of the Lanczos iteration consists in the Hessian-vector product $\overline{\nabla^2 f}(x_k) q_t$, to be computed at each iteration $t \geq 0$.

In matrix terms, setting $Q_t \overset{\text{def}}{=} (q_0, \cdots, q_t) \in \mathbb{R}^{n\times(t+1)}$ and $q_{-1} = 0$, then (1.99) can be written in a more compact way as

$$\overline{\nabla^2 f}(x_k) Q_t - Q_t T_t = \gamma_{t+1} q_{t+1} e_{t+1}^\top, \tag{1.100}$$

where $T_t \in \mathbb{R}^{(t+1)\times(t+1)}$ is a symmetric tridiagonal matrix with diagonal elements $(\delta_0, \delta_1, ..., \delta_t)$, upper and lower diagonal elements $(\gamma_1, \gamma_2, ..., \gamma_t)$. Moreover, the columns of Q_t are orthonormal, i.e.

$$Q_t^\top q_{t+1} = 0 \tag{1.101}$$

$$Q_t^\top Q_t = I_{t+1} \tag{1.102}$$

Left-multiplication in (1.100) by $Q_t^\top$ and (1.101)–(1.102) give:

$$Q_t^\top \overline{\nabla^2 f}(x_k) Q_t = T_t. \tag{1.103}$$

After $(j+1)$ steps, the Lanczos method indeed generates a matrix $Q_j \overset{\text{def}}{=} (q_0, \cdots, q_j) \in \mathbb{R}^{n\times(j+1)}$ with orthogonal columns $\{q_t\}_{t=0}^j \subseteq \mathbb{R}^n$ that span the Krylov's subspace (1.92) generated by this method. Such a matrix satisfies the properties (1.101)–(1.103) with $t = j$.

Let us now see how exploiting the Lanczos basis to compute a global minimiser of $m_k(s)$ over $\mathcal{K}_j$. We can take advantage of the tridiagonal structure of $T_j \in \mathbb{R}^{(j+1)\times(j+1)}$ and search for a minimiser $s_j^{(k)}$ of $m_k(s)$ in the range of the basis Q_j. To do so we first note that for all $s \in \mathcal{K}_j$ there exists a vector $u \in \mathbb{R}^{j+1}$ such that $s = Q_j u$. Using this and

(1.102)–(1.103) with $t = j$, the problem $\min_{s\in\mathcal{K}_j} m_k(s)$ reduces to

$$\min_{u\in\mathbb{R}^{j+1}} \widetilde{m}_j^{(k)}(u) \overset{\text{def}}{=} f(x_k) + \nabla f(x_k)^\top Q_j e_1 u^\top e_1 + \frac{1}{2}u^\top T_j u + \frac{1}{3}\sigma\|u\|^3, \tag{1.104}$$

with $e_1 \in \mathbb{R}^{j+1}$. We indeed search for a solution $u_j^{(k)} \in \mathbb{R}^{j+1}$ of problem (1.104), defining $s_j^{(k)} = Q_j u_j^{(k)}$. Such iteration on j is pursued until an iterate $s_{\bar{j}}^{(k)}$ fulfilling a suitable termination criterion (see, e.g., (3.26), (3.27)) is found.

Remark 7. As noticed in [40, Section 6.2], there are a number of relevant observations that can be made.

- As T_j is tridiagonal, even when n is large, it is feasible to use the method based on factorisation described in Appendix (Section A.1) to compute the solution of $\min_{s\in\mathcal{K}_j} m_k(s)$.
- Having found $u_j^{(k)}$, the matrix Q_j is needed to recover $s_j^{(k)}$, and thus the Lanczos vectors q_j will either need to be saved on backing store or regenerated when required.
- As a sequence of such problems is solved and as T_j only changes by the addition of an extra diagonal and superdiagonal entry, solution data from one subproblem may be useful for starting the next.
- The authors in [40] underline that employing this approach within Algorithm 2 with the modified steps in Algorithm 3 benefits from the theoretical guarantees of convergence and satisfies the complexity bounds developed in [41]. To see this one can set $\mathcal{L}_k = \mathcal{S}_j$ in Lemma 17 and note that the current gradient is included in all the subspaces $\mathcal{S}_j$.

We end this chapter mentioning that, upon Lanczos-type iterations where the minimisation is done via nested (lower dimensional) Krylov's subspaces, a number of procedures have been proposed to approximately minimise the cubic model (1.14) at each iteration of the ARC framework. These methods range from the described approach in [40], up to minimisation via gradient descent (see, e.g., [1, 33, 32]) or the Barzilai-Borwein gradient method [103, 71, 20]. All these techniques require the evaluation of the gradient (1.15) of the model $m_k(s)$ at iteration k, this in turn needs the computation of the Hessian-vector product $\overline{\nabla^2 f}(x_k)s$.

If the exact Hessian is considered, the product $\nabla^2 f(x_k)s$ can be approximated by finite-difference, with at most two gradient evaluations [20, 32].
The basis for doing that is once again the Taylor's expansion. In fact (see, e.g., [97]), when second derivatives of f exist and are Lipschitz continuous in a neighborhood x_k, we have that

$$\nabla f(x_k + hs) = \nabla f(x_k) + h\nabla^2 f(x_k)s + \underbrace{\frac{1}{2}h^2\nabla^3 f(x_k)[s]^2 + \mathcal{O}(h^3)}_{=\mathcal{O}(h^2)}, \tag{1.105}$$

$$\nabla f(x_k - hs) = \nabla f(x_k) - h\nabla^2 f(x_k)s + \frac{1}{2}h^2\nabla^3 f(x_k)[s]^2 + \mathcal{O}(h^3), \tag{1.106}$$

so that, from (1.105),

$$\nabla^2 f(x_k)s = \frac{\nabla f(x_k + hs) - \nabla f(x_k)}{h} + \mathcal{O}(h)$$

and, hence, we can set

$$\frac{\nabla f(x_k + hs) - \nabla f(x_k)}{h} \simeq \nabla^2 f(x_k)s, \tag{1.107}$$

choosing a small positive scalar h. In this respect, the authors in [20] consider $h = 2 \cdot 10^{-6} \frac{1+\|x_k\|}{\max[10^{-5}, \|s\|]}$. The approximation error is $\mathcal{O}(h)$ and the cost for obtaining the estimation is an additional single gradient evaluation at $x_k + hs$, since $\nabla f(x_k)$ is already available at iteration k when computing the Hessian-vector product $\nabla^2 f(x_k)s$. The formula (1.107) corresponds to the so-called *forward-difference approximation*.

The accuracy of the Hessian-vector product approximation can even be increased considering a central-difference formula, obtained by substracting (1.106) from (1.105) and dividing by $2h$, which provides us with

$$\nabla^2 f(x_k)s = \frac{\nabla f(x_k + hs) - \nabla f(x_k - hs)}{2h} + \mathcal{O}(h^2),$$

giving

$$\frac{\nabla f(x_k + hs) - \nabla f(x_k - hs)}{2h} \simeq \nabla^2 f(x_k)s.$$

In this case the approximation error becomes $\mathcal{O}(h^2)$, but the two gradient evaluations $\nabla f(x_k + hs)$ and $\nabla f(x_k - hs)$ are needed.

Interestingly, all these matrix-free implementations remain relevant if $\nabla^2 f(x_k)$ is approximated via subsampling (see Section 3.5), proceeding as in Section 3.1 of [15], while back-propagation-like methods in machine learning via artificial neural networks (see Section 7.1) also allow computations of Hessian-vector products at a similar cost [100, 112].

Part II

Adaptive Cubic Regularisation Methods under Inexact Evaluations

Introduction to Part II

Iteration and evaluation complexity of algorithms for nonlinear and possibly nonconvex optimisation problems of the form (1.1) has been the subject of active research in recent years. Until recently, the results had focused on methods using up to second-order derivatives of the objective function and on convergence guarantees to first or second-order stationary points [120, 96, 95, 68, 41].

We recall that the complexity results in [22, 41, 38, 37, 36] are sharp and optimal with respect to steepest descent, Newton's method and Newton's method embedded into a line search or a trust-region strategy [42, 38]. Therefore, ARC methods have generated considerable interest in the research community.

Even more recently, [35] proposed a conceptual unified framework subsuming all the known results for regularisation methods, establishing an upper evaluation complexity bound for arbitrary model degree (in analogy to [22]) and also, for the first time, for arbitrary orders of optimality. This paper additionally covers unconstrained problems and problems involving *"inexpensive" constraints*, that is constraints whose evaluation/enforcement cost is negligible compared to that of evaluating the objective function and its derivatives and, hence, the evaluation complexity is, for such constraints, well captured by the number of evaluations of the objective function and its derivatives. It also allows for a full range of smoothness assumptions on the objective function. Finally, it proves that the complexity results obtained are optimal in the sense that upper and lower evaluation complexity bounds match in order. In [35], all the above mentioned results are established for versions of the regularisation algorithms where it is assumed that objective function values and values of its derivatives (when necessary) can be computed exactly. Nevertheless, in many real-life problems, it may be difficult or even not possible to obtain accurate values of the objective and/or derivatives.

This difficulty has been known for a long time and has generated its own stream of results, among which we mention the Trust-Region method using dynamic accuracy on the objective function and (possibly on) its gradient (see Sections 8.4.1.1 and 10.6 of [52] and [13]), together with the purely probabilistic approaches of [99, 19, 23]. Consequently, suitable variants of existing and well-assessed methods to allow for inexact derivatives and/or function evaluations, though conserving optimal complexity, are getting increasing attraction.

To start with, particularly relevant is the ARC algorithm where exact second derivatives of f are not required [40]. 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 characterises complexity and convergence rate behaviour of the procedure; in [40, 41], the well-known Dennis-Moré condition [57] and slightly stronger agreements are considered.

Recently, ARC and Newton-type methods with inexact or incomplete Hessian information, coupled with possibly inexact function and gradient information, have received large attention (see, e.g., [10, 5, 12, 15, 28, 30, 44, 50, 79, 102, 107, 124, 123, 125, 69]). In particular, ARC methods with probabilistic models have been proposed and studied in [44, 50, 79, 124, 123, 125], while a cubic regularised method incorporating variance reduction techniques has been given in [127]; inexact Hessian information is considered in [50, 9, 124, 123]; approximate gradient and Hessian evaluations are used in [35, 44, 114, 125]; function, gradient and Hessian values are sampled in [84, 18]. The amount of inexactness allowed is controlled dynamically in [35, 44, 84, 50, 9].

The interest in such methods is mainly motivated by problems where the evaluation of f and/or its derivatives is computationally expensive, such as large-scale optimisation problems arising in machine learning and data analysis, modeled as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{N}\sum_{i=1}^{N}\varphi_i(x), \tag{1.108}$$

with N being a positive scalar and $\varphi_i : \mathbb{R}^n \to \mathbb{R}$. That is to say that the objective function f is the mean of the N component functions φ_i in (1.108) and, hence, the evaluation of the exact function and derivatives might be, for larger values of N, computationally expensive.

First-order methods, requiring only up to first-order derivatives of the objective f, together with their stochastic versions, are frequently used by the community within this framework. These methods include Gradient Descent (GD), Stochastic Gradient Descent (SGD), incremental gradient algorithms based on variance reduction, such as Stochastic Variance Reduction Gradient (SVRG), SVRG with Barzilai-Borwein step sizes (SVRG-BB), StochAstic Recursive grAdient algoritHm (SARAH), Stochastic Average Gradient (SAG) and its variant SAGA, gradient methods with adaptive step length selection based on line search or other globalisation strategies (see [11, 54]). Generally, their per-iteration cost is rather low and they result in a quite smart implementation. Anyway, convergence may take a while on large-scale problems (the rate of convergence is generally low), their performance can be seriously hindered by ill-conditioning and, mainly, their success is tightly intertwined with fine-tuning (often many) hyperparameters[†] (think about the step size of a line search scheme, for instance). In addition, the objective function f of (1.1) can be nonconvex, as usual in machine learning applications, so first-order methods are expected to have more difficulty avoiding saddle points, since the make no use of curvature information.

By contrast, second-order strategies, where also second-order derivatives (i.e., curvature information) are considered to guide the optimisation process, have been shown to be highly resilient to ill-conditioned and badly-scaled problems, taking advantage of curvature information to easier escape from saddle points [28, 38, 123] or even local minima (see, e.g. [88]). They also seem to be less sensitive to the choice of hyperparameters and less involved in the parameters tuning. However, the per iteration cost is expected to be higher than first-order schemes, due to the step computation, the Hessian evaluation or the approximation of Hessian-vector products.

This second part of the thesis presents variants of the basic ARC method introduced in Part I, aiming at reducing the per-iteration cost by employing inexact function and/or derivatives information.

This part is the subject of the recent publications [9, 10] and it is argued in three different chapters.

In Chapter 2 we analyse how to compute inexact evaluations for f and its derivatives

[†]The hyperparameters are parameters that cannot be estimated from the data. Their formal definition, quoting [14], is the following.

> *We define a hyperparameter for a learning algorithm $\mathscr{A}$ as a variable to be set prior to the actual application of $\mathscr{A}$ to the data, one that is not directly selected by the learning algorithm itself.*

.

in the finite-sum minimisation setting.

Chapter 3 considers a new ARC method based on inexact Hessian information, dynamically chosen. The theoretical analysis of the proposed procedure is given, in order to show that the key complexity property of ARC framework is still guaranteed. Application to large-scale finite-sum minimisation (1.108) based on subsampled Hessian is discussed and analysed in both a deterministic and probabilistic manner.

Chapter 4 finally proposes an extension of the method considered in Chapter 3 and of the framework [35] when models up to order three are considered, within the range of unconstrained or inexpensively-constrained problems, allowing for inexact evaluations of the objective function and of the required derivatives. This generalisation shares with the method of the previous chapter the advantage of preserving the optimal evaluation complexity of the standard regularisation methods. An interesting, but practically more restrictive, variant of our algorithm is also exploited, deriving an improved evaluation complexity, as well as a probabilistic version of the framework, with a practical application to the context of subsampling methods for machine learning.

Chapter 2

Inexact Evaluations in the Finite-Sum Minimisation Setting

In what follows, we focus on the solution of large-scale instances of the finite-sum problems arising in machine learning and data analysis, modelled by (1.108), that can be conveniently solved by subsampling procedures where $f(x_k)$ and/or $\nabla f(x_k)$ and/or $\nabla f^2(x_k)$ are approximated by randomly sampling component functions from $\{\varphi_i(x_k)\}_{i=1}^N$, $\{\nabla\varphi_i(x_k)\}_{i=1}^N$, $\{\nabla^2\varphi_i(x_k)\}_{i=1}^N$ (see, e.g., [28]).

To this aim, let us consider the generic setting in which at iteration $k \geq 0$ the inexact quantities $\overline{f}_k(x_k)$, $\overline{f}_k(x_k+s_k)$, $\overline{\nabla f}(x_k)$ and $\overline{\nabla^2 f}(x_k)$ have to be computed. Assume that the following requirements have to be satisfied:

$$P\Big[\|\overline{\nabla^j f}(x_k) - \nabla^j f(x_k)\| \leq \tau_{j,k}\Big] \;\geq\; (1-t), \qquad j \in \{1,2\}, \tag{2.1}$$

$$P\Big[|\overline{f}_k(x_k+s_k) - f(x_k+s_k)| \leq \tau_{0,k}\Big] \;\geq\; 1-t, \tag{2.2}$$

$$P\Big[|\overline{f}_k(x_k) - f(x_k)| \leq \tau_{0,k}\Big] \;\geq\; 1-t, \tag{2.3}$$

with $\tau_{j,k}$, $j \in \{1,2\}$ prefixed absolute accuracies for the derivative of order j at iteration k and $t \in (0,1)$ a prescribed probability of failure. We are therefore assuming that the prescribed accuracies on f and its derivative of order $j \in \{1,2\}$ are satisfied with probability at least $1-t$.

The approximations of the objective function value and of first and second derivatives by a subsampling procedures take the form:

$$\overline{f}_k(x_k) = \frac{1}{|\mathcal{D}_{k,1}|}\sum_{i\in\mathcal{D}_{k,1}} \varphi_i(x_k), \qquad \overline{f}_k(x_k+s_k) = \frac{1}{|\mathcal{D}_{k,2}|}\sum_{i\in\mathcal{D}_{k,2}} \varphi_i(x_k+s_k), \tag{2.4}$$

$$\overline{\nabla f}(x_k) = \frac{1}{|\mathcal{G}_k|}\sum_{i\in\mathcal{G}_k} \overline{\nabla \varphi_i}(x_k), \tag{2.5}$$

$$\overline{\nabla^2 f}(x_k) = \frac{1}{|\mathcal{H}_k|}\sum_{i\in\mathcal{H}_k} \overline{\nabla^2 \varphi_i}(x_k), \tag{2.6}$$

where $\mathcal{D}_{k,1}$, $\mathcal{D}_{k,2}$, $\mathcal{G}_k$, $\mathcal{H}_k$ are subsets of $\{1,2,\ldots,N\}$ and $|\mathcal{D}_{k,1}|, |\mathcal{D}_{k,2}|, \mathcal{G}_k|, |\mathcal{H}_k|$ being the so-called sample sizes.

The question then arises of estimating the cardinality of these sample sets in order to ensure that the approximations of the objective function value and its first and second derivatives satisfy (2.1), for $j \in \{1,2\}$, (2.2) and (2.3) (for the case $j=0$). This issue can be addressed using the operator-Bernstein inequality given in [115].

The next theorem provides us with the final result concerning the sample sizes for uniform subsampling the objective function and its derivatives up to order $j = 2$.

Theorem 19. Suppose that there exist nonnegative constants $\{\kappa_{\varphi,j}\}_{j=0}^{2}$ such that, for $x \in \mathbb{R}^n$ and all $j \in \{0, 1, 2\}$,

$$\max_{i\in\{1,\dots,N\}} \|\nabla^j \varphi_i(x)\| \le \kappa_{\varphi,j}(x). \tag{2.7}$$

Let $t \in (0, 1)$ and suppose that a subsample $\mathcal{A}_k$ is chosen randomly and uniformly from $\{1, \dots, N\}$ and that, for some $j \in \{0, 1, 2\}$, one computes

$$\overline{\nabla^j f}(x) = \frac{1}{|\mathcal{A}_k|} \sum_{i\in\mathcal{A}_k} \overline{\nabla^j \varphi_i}(x),$$

with

$$|\mathcal{A}_k| \ge \min\left\{N, \left\lceil \frac{4\kappa_{\varphi,j}(x)}{\tau_j} \left(\frac{2\kappa_{\varphi,j}(x)}{\tau_j} + \frac{1}{3}\right) \log\left(\frac{d_j}{t}\right)\right\rceil\right\}, \tag{2.8}$$

where

$$d_j = \begin{cases} 2, & \text{if } j = 0, \\ n+1, & \text{if } j = 1, \\ 2n, & \text{if } j = 2. \end{cases}$$

Then condition (2.1) holds for $x = x_k$ with probability at least $(1 - t)$ if $j \in \{1, 2\}$, or, if $j = 0$, each of the conditions (2.2) and (2.3) holds with probability at least $(1 - t)$ for $x = x_k + s_k$ and $x = x_k$, respectively.

Proof. Let $\mathcal{A}_k \subseteq \{1, \dots, N\}$ be a sample set of cardinality $|\mathcal{A}_k|$. Consider $j \in \{0, 1, 2\}$ and $|\mathcal{A}_k|$ random tensors $Z_u(x)$ such that,

$$P\left[Z_u(x) = \nabla^j \varphi_i(x)\right] = \frac{1}{N}, \quad (i \in \{1, \dots, N\}).$$

For $u \in \mathcal{A}_k$, let us define

$$X_u \overset{\text{def}}{=} \left(Z_u(x) - \nabla^j f(x)\right), \qquad \overline{\nabla^j f}(x) \overset{\text{def}}{=} \frac{1}{|\mathcal{A}_k|} \sum_{u\in\mathcal{A}_k} Z_u(x)$$

and

$$X \overset{\text{def}}{=} \sum_{u\in\mathcal{A}_k} X_u = |\mathcal{A}_k| \left(\overline{\nabla^j f}(x) - \nabla^j f(x)\right).$$

Since (1.108) gives that

$$\frac{1}{N} \sum_{i=1}^{N} \nabla^j \varphi_i(x) = \nabla^j f(x),$$

we deduce that

$$E(X_u) = \frac{1}{N} \sum_{i=1}^{N} \left(\nabla^j \varphi_i(x) - \nabla^j f(x)\right) = 0, \quad u \in \mathcal{A}_k.$$

Moreover, assuming $Z_u(x) = \nabla^j \varphi_l(x)$ for some $l \in \{1, \dots, N\}$ and using (2.7), we have that

$$\|X_u\| \le \left\| \frac{N-1}{N} \nabla^j \varphi_l(x) - \frac{1}{N} \sum_{i\in\{1,\dots,N\}\setminus\{l\}} \nabla^j \varphi_i(x) \right\| \le 2\,\frac{N-1}{N}\,\kappa_{\varphi,j}(x) \le 2\kappa_{\varphi,j}(x),$$

so that the variance of X can be bounded as follows:

$$\begin{aligned} v(X) &= \max\left[\|E(XX^T)\|, \|E(X^TX)\|\right] \\ &= \max\left[\Big\|\sum_{u\in\mathcal{A}_k} E(X_uX_u^T)\Big\|, \Big\|\sum_{u\in\mathcal{A}_k} E(X_u^TX_u)\Big\|\right] \\ &\le \max\Big[\sum_{u\in\mathcal{A}_k}\|E(X_uX_u^T)\|, \sum_{u\in\mathcal{A}_k}\|E(X_u^TX_u)\|\Big] \\ &\le \max\Big[\sum_{u\in\mathcal{A}_k}E(\|X_uX_u^T\|), \sum_{u\in\mathcal{A}_k}E(\|X_u^TX_u\|)\Big] \\ &\le \sum_{u\in\mathcal{A}_k}E(\|X_u\|^2) \le 4|\mathcal{A}_k|\kappa_{\varphi,j}^2(x), \end{aligned}$$

in which the first and the third inequalities hold because of the triangular inequality, while the second is due to the Jensen's inequality* (note that the spectral norm $\|\cdot\|$ is convex). Therefore, according to the Operator-Bernstein Inequality stated in [115, Theorem 6.1.1], we obtain that

$$P\Big[\|\overline{\nabla_x^j f}(x) - \nabla_x^j f(x)\| \ge \epsilon_j\Big] = P\Big[\|X\| \ge \epsilon_j|\mathcal{A}_k|\Big] \le d_j e^{-\frac{\epsilon_j^2|\mathcal{A}_k|}{4\kappa_{\varphi,j}(x)\left(2\kappa_{\varphi,j}(x)+\frac{1}{3}\epsilon_j\right)}}, \tag{2.9}$$

with $d_j = 2$ if $j = 0$, $d_j = n+1$ if $j = 1$ and $d_j = 2n$ if $j = 2$. Then, bounding the right-hand side of (2.9) by t, taking logarithms and extracting $|\mathcal{A}_k|$ gives (2.8). $\square$

In particular, Theorem 19 gives the lower bounds

$$|\mathcal{D}_{k,\ell}| \ge \min\left\{N, \left\lceil \frac{4\kappa_{\varphi,0}(x)}{\tau_{0,k}}\left(\frac{2\kappa_{\varphi,0}(x)}{\tau_{0,k}} + \frac{1}{3}\right)\log\left(\frac{2}{t}\right)\right\rceil\right\}, \tag{2.10}$$

with

$$x = \begin{cases} x_k, & \text{if } \ell = 1, \\ x_k + s_k & \text{if } \ell = 2, \end{cases}$$

$$|\mathcal{G}_k| \ge \min\left\{N, \left\lceil \frac{4\kappa_{\varphi,1}(x_k)}{\tau_{1,k}}\left(\frac{2\kappa_{\varphi,1}(x_k)}{\tau_{1,k}} + \frac{1}{3}\right)\log\left(\frac{n+1}{t}\right)\right\rceil\right\}, \tag{2.11}$$

$$|\mathcal{H}_k| \ge \min\left\{N, \left\lceil \frac{4\kappa_{\varphi,2}(x_k)}{\tau_{2,k}}\left(\frac{2\kappa_{\varphi,2}(x_k)}{\tau_{2,k}} + \frac{1}{3}\right)\log\left(\frac{2n}{t}\right)\right\rceil\right\}. \tag{2.12}$$

The implementation of rules (2.10)-(2.12) requires the knowledge of the size of the component functions φ_i and their first and second-order derivatives. If only global information is available, the dependence on x may obviously be avoided by choosing a uniform upper bound $\kappa_{\varphi,j}$ for all $x \in \mathbb{R}^n$, at the cost of a lesser adaptivity. Similar bounds on the sample size used to approximate gradients and Hessians up to a prescribed probability have been derived and used in [123], where it has also been observed that there are problems in which estimations of the needed uniform upper bounds can be obtained.

In particular, let $\{(a_i, b_i)\}_{i=1}^N$ denote the pairs forming a data set with $a_i \in \mathbb{R}^n$ being the vector containing the features of the i-th example and b_i being its label.

In [123] the authors considered the minimisation of objective function $(1/N)\sum_{i=1}^N(\Phi(a_i^Tx) - b_ia_i^Tx)$ over a sparsity inducing constraint set, e.g., $\mathcal{X} = \{x \in \mathbb{R}^n \mid \|x\|_1 \le 1\}$, for cumulant generating functions Φ of different forms, and explicitly provided the uniform bound $\kappa_{\varphi,1}$. Taking into account that x belongs to the set $\mathcal{X}$, uniform bounds for the objective function and the Hessian norm can also be derived.

Uniform bounds are available also in the unconstrained setting for binary classification

*The Jensen's inequality describes how averaging interacts with convexity. Let X be a random matrix, and let h be a real-valued function convex on matrices. Then, $h(\mathbb{E}[X]) \le \mathbb{E}[h(X)]$.

problems modelled by the sigmoid function and least-squares loss, i.e. problems of the form (1.108) and

$$\varphi_i(x) = \left(b_i - \frac{1}{1+e^{-a_i^T x}}\right)^2, \quad i = 1\dots,N. \tag{2.13}$$

Let $v_i(x) = (1+e^{-a_i^T x})^{-1}$ and note that $b_i \in \{0,1\}$, $v_i(x) \in (0,1)$ for any $x \in \mathbb{R}^n$. Then, $|\varphi_i(x)| \leq 1$, for any $x \in \mathbb{R}^n$. Moreover, uniform upper bounds $\kappa_{\varphi,j}$ for $\nabla^j \varphi_i(x)$, $j \in \{1,2\}$ can be easily derived and are reported in Table 2.1 along with the expression of the first and second-order derivatives of $\varphi_i(x)$. The computation of these bounds requires a pre-processing phase as the norms of the features vectors $\{a_i\}_{i=1}^N$ of the datasets are needed.

	Derivatives	$\kappa_{\varphi,j}$
$\nabla\varphi_i(x)$	$-2(b_i - v_i(x))(1-v_i(x))v_i(x)a_i$	$\frac{2}{5}\|a_i\|$
$\nabla^2\varphi_i(x)$	$-2v_i(x)(1-v_i(x))(3v_i(x)^2 - 2v_i(x)(1+b_i)+b_i)a_i a_i^\top$	$\frac{1}{2}\|a_i\|^2$

Table 2.1: First and second-order derivatives of (2.13) and corresponding uniform bounds.

Chapter 3

Adaptive Cubic Regularisation Methods under Dynamic Inexact Hessian Information

This chapter 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, wired into the resulting ARC scheme. The agreement between the Hessian of f and its approximation can be loose at the beginning of the iterative process and progressively increases as the norm of step size 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 [50, 124], without impairing optimal complexity results. The new algorithm is theoretically analysed 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 described in Chapter 1.

We also discuss the application of our method to finite-sum minimisation problems (1.108) 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 tests on a set of nonconvex binary classification problems.

The chapter is organised as follows. With reference to the main ARC framework given by Algorithm 2 in Part I, Section 3.1 briefly reviews the classical accuracy requirements for Hessian approximation and introduces our new choice. Then, in Section 3.2, we introduce our variant of the algorithm, based on a dynamic rule for building the inexact Hessian in order to achieve the level of resemblance designed in Section 3.1. The first-order iteration and evaluation complexity bound of the resulting algorithm is studied in Section 3.3 along with the asymptotic behaviour of the generated sequence, while complexity bounds and convergence to second-order points are analysed in Section 3.4. The application of our algorithm to the finite-sum optimisation problem is discussed in Section 3.5. Finally, the main differences of our proposal from the closely related works in the literature is addressed in Section 3.6.

3.1 Overview of the requirements for Hessian approximation

With reference to the basic ARC algorithm (Algorithm 2) introduced in Part I, we have that optimal complexity to reach first-order critical points (see Theorem 8) can be achieved under Assumption 1.2.1 on the objective function, Assumption 1.2.2 on its Hessian and Assumption 1.2.3 about the requirement on the Hessian approximation.

We now recall the classical choices that have been considered in literature, thou preserving the complexity result.
Kohler and Lucchi [79] suggested to achieve (1.31) by imposing

$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \chi \|s_k\|, \tag{3.1}$$

for some $\chi > 0$. It is evident that the agreement between $\nabla^2 f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ depends on the step length which can be determined only after $\overline{\nabla^2 f}(x_k)$ is formed. This issue is circumvented in practice employing the step length at the previous iteration [79].

Xu et al. [124, 125] analysed ARC algorithm making a major modification on the level of resemblance between $\nabla^2 f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ over (1.31), requiring

$$\|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s_k\| \leq \mu \|s_k\|, \tag{3.2}$$

with $\mu \in (0,1)$. In practice, (3.2) is achieved imposing $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \mu$. In order to retain the optimal complexity of the classical ARC method, $\mu = \mathcal{O}(\epsilon_1)$ is assumed. We observe that, given a positive v, the requirement $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq v$ can be enforced approximating $\nabla^2 f(x)$ by finite-differences or interpolating functions [52]. Moreover, for the class of large-scale finite-sum minimisation (1.108), the requirement $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq v$ can be satisfied in probability via subsampling as in (2.6) (see also [10, 28, 79, 124]).

In this work, we propose a variant of Algorithm 2 employing a model of the form (1.14) and a matrix $\overline{\nabla^2 f}(x_k)$ such that

$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq c_k, \tag{3.3}$$

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 norm. We will show that for properly chosen scalars c_k, condition (3.3) is an implementable rule to achieve (1.31). In addition, in the first phase of the procedure, the accuracy imposed on $\overline{\nabla^2 f}(x_k)$ can be less stringent with respect to the proposal made in [50, 124, 125], though preserving the complexity bound $\mathcal{O}(\epsilon_1^{-3/2})$.

The study of our variant of the ARC algorithm is presented in the subsequent sections. We refer to Sections 3.5 and 3.6 for a discussion on the application to the finite-sum optimisation problem of the form (1.108) and the comparison with the above mentioned related works in the literature, that can be addressed in a more direct way once that our new strategy is presented.

3.2 An adaptive choice of the inexact Hessian: the ARC-DH algorithm

We now propose and study a variant of Algorithm 2 which maintains the complexity bound $\mathcal{O}(\epsilon_1^{-3/2})$ to handle first-order critical points.

Our algorithm is based on the use of an approximation $\overline{\nabla^2 f}(x_k)$ of $\nabla^2 f(x_k)$ in the construction of the cubic model and a rule for choosing the level of agreement between $\overline{\nabla^2 f}(x_k)$ and $\nabla^2 f(x_k)$. The accuracy requirements in the approximate minimisation of $m_k(s)$ consist of (1.17) and a condition on $\|\nabla_s m_k(s_k)\|$ which includes the condition (1.18) but it is not limited to it.

Our analysis is carried out under Assumption 1.2.1, Assumption 1.2.2, Assumption 1.2.4 and we suppose that the step s_k computed has the following property.

Assumption 3.2.1. *For all $k \geq 0$ and some $0 \leq \theta_k \leq \theta \in [0, 1)$, s_k satisfies*

$$
\begin{aligned}
& m_k(s_k) < m_k(0), && (3.4)\\
& \|\nabla_s m_k(s_k)\| \leq \theta_k \|\nabla f(x_k)\|. && (3.5)
\end{aligned}
$$

By (1.10) and (1.14) it follows that

$$
m_k^C(s) = \widehat{T}_2(x_k, s) + \frac{1}{2} s^T(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s + \frac{L}{6}\|s\|^3. \tag{3.6}
$$

Then, (1.10) yields

$$
f(x_k + s) \leq \widehat{T}_2(x_k, s) + E_k(s), \tag{3.7}
$$

where

$$
E_k(s) = \frac{1}{2}\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \|s\|^2 + \frac{L}{6}\|s\|^3. \tag{3.8}
$$

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

Assumption 3.2.2. *Let $\overline{\nabla^2 f}(x_k) \in \mathbb{R}^{n \times n}$ satisfy*

$$
\begin{aligned}
\Delta_k &= \nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k), \quad \|\Delta_k\| \leq c_k, && (3.9)\\
c_k &= c, \quad \text{if } \|s_k\| \geq 1, && (3.10)\\
c_k &\leq \alpha(1-\theta)\|\nabla f(x_k)\|, \quad \text{if } \|s_k\| < 1, && (3.11)
\end{aligned}
$$

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 3.2.1.

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

Lemma 20. Let Assumption 1.2.1(ii), Assumption 1.2.2, Assumptions 3.2.1–3.2.2 and Assumption 1.2.4 hold. Let $E_k(s)$ and Δ_k as in (3.8)–(3.9). Then,

$$
\|\Delta_k\| \leq \begin{cases} c\|s_k\|, & \text{if } \|s_k\| \geq 1, \\ \alpha(\kappa_B + \sigma_k)\|s_k\|, & \text{if } \|s_k\| < 1, \end{cases} \tag{3.12}
$$

and

$$
E_k(s_k) \leq \begin{cases} \dfrac{1}{2}\left(c + \dfrac{L}{3}\right)\|s_k\|^3, & \text{if } \|s_k\| \geq 1, \\ \dfrac{1}{2}\left(\alpha(\kappa_B + \sigma_k) + \dfrac{L}{3}\right)\|s_k\|^3, & \text{if } \|s_k\| < 1. \end{cases} \tag{3.13}
$$

Proof. First consider the case $\|s_k\| \geq 1$. Trivially, the inequality in (3.9) gives (3.12) 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 (3.13).
Suppose now that $\|s_k\| < 1$. Using (3.5), Assumptions 1.2.4 and 3.2.1, we obtain

$$\begin{aligned} \theta\|\nabla f(x_k)\| &\geq \|\nabla_s m_k(s_k)\| \\ &= \left\| \nabla f(x_k) + \overline{\nabla^2 f}(x_k)s_k + \sigma_k s_k\|s_k\| \right\| \\ &\geq \|\nabla f(x_k)\| - \|\overline{\nabla^2 f}(x_k)\|\,\|s_k\| - \sigma_k\|s_k\|^2 \\ &\geq \|\nabla f(x_k)\| - \kappa_B\|s_k\| - \sigma_k\|s_k\|, \end{aligned} \tag{3.14}$$

which gives

$$\|s_k\| \geq \frac{(1-\theta)\|\nabla f(x_k)\|}{\kappa_B + \sigma_k}. \tag{3.15}$$

Thus, (3.9) and (3.11) 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, (3.11) implies (3.12) and this along with (3.8) gives (3.13). □

Taking into account the previous result and assuming $\alpha \in [0, 2/3)$, we can establish when the overestimation property $f(x_k+s_k) \leq m_k(s_k)$ is verified. Using (1.14) and (3.6)–(3.7) we see that if $E_k(s_k) \leq \sigma_k\|s_k\|^3/3$, then $m_k^C(s_k) \leq m_k(s_k)$, which implies that $m_k(s_k)$ 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 (3.13) implies $m_k^C(s_k) \leq m_k(s_k)$.
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 (3.13) implies $m_k^C(s_k) \leq m_k(s_k)$.

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

Lemma 21. Let Assumption 1.2.1(ii), Assumption 1.2.2, Assumptions 3.2.1–3.2.2 and Assumption 1.2.4 hold. Suppose that the scalar α in Assumption 3.2.2 is such that $\alpha \in \left[0, \frac{2}{3}\right)$ and that the constant η_2 in Algorithm 2 is such that $\eta_2 \in \left(0, \frac{2-3\alpha}{2}\right)$. It then holds

$$\sigma_k \leq \sigma_{\max} \overset{\text{def}}{=} \max\left\{\sigma_0,\, \gamma_3\frac{3c+L}{2(1-\eta_2)},\, \gamma_3\frac{3\alpha\kappa_B+L}{2-3\alpha-2\eta_2}\right\}, \quad \forall k \geq 0, \tag{3.16}$$

where γ_3 is the constant used in (1.20).

Proof. Let us derive conditions on σ_k ensuring $\rho_k \geq \eta_2$. By (1.14) and (3.4), it follows $\|s_k\| \neq 0$ and by (1.25),

$$\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k) > \frac{\sigma_k}{3}\|s_k\|^3 > 0. \tag{3.17}$$

Moreover, by (3.7) and the fact that $E_k(s_k) > 0$,

$$1 - \rho_k = \frac{f(x_k + s_k) - \widehat{T}_2(x_k, s_k)}{\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)} \leq \frac{E_k(s_k)}{\widehat{T}_2(x_k, 0) - \widehat{T}_2(x_k, s_k)} < \frac{3E_k(s_k)}{\sigma_k \|s_k\|^3}. \tag{3.18}$$

If $\|s_k\| \geq 1$, using (3.13) 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 (3.13) and (3.18) 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 (1.20) implies $\sigma_{k+1} \leq \sigma_k$ in case $\rho_k \geq \eta_2$ and, more generally, the inequality (3.16). □

An important consequence of Lemma 20 and Lemma 21 is that (3.3) implies

$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \max(c, \alpha(\kappa_B + \sigma_{\max}))\|s_k\|, \tag{3.19}$$

for all $k \geq 0$, so that condition (1.31) is satisfied.

With reference to Lemma 21, the value of α in (3.11) determines the accuracy of $\overline{\nabla^2 f}(x_k)$ 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 $2/3$, then the accuracy of the Hessian approximation reduces, η_2 tends to zero and $\sigma_{\max}$ tends to infinity.*

On the basis of the previous analysis we sketch on the following page our version of Algorithm 2, denoted as ARC-DH.

The main feature is the adaptive rule for adjusting the agreement between $\overline{\nabla^2 f}(x_k)$ and $\nabla^2 f(x_k)$, as specified in Assumption 3.2.2. At the beginning of the k-th iteration, the variable flag is equal to either 1 or 0 and determines the value of c_k; specifically $c_k = c$ if $\mathrm{flag} = 1$, $c_k = \alpha(1-\theta)\|\nabla f(x_k)\|$ otherwise, with $\nabla f(x_k)$ being available (at iteration $k = 0$, flag is set equal to 1). Scalars c and α are initialised at Step 0; the choice of α and η_2 is made in accordance to the results presented above. Then, $\overline{\nabla^2 f}(x_k)$ 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 (3.11) is required to hold if $\|s_k\| < 1$, whereas $\|s_k\|$ can be determined only after $\overline{\nabla^2 f}(x_k)$ is formed. Therefore, at the end of a successful iteration, the value of flag is fixed according to the step length at the last step. Successively, once $\overline{\nabla^2 f}(x_k)$ and s_k have been computed, if $\|s_k\| < 1$, $\mathrm{flag} = 1$ 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 $\overline{\nabla^2 f}(x_k)$ is recomputed at the successive iteration. This unsuccessful iteration is ascribed to the choice of the matrix $\overline{\nabla^2 f}(x_k)$, hence the regularisation parameter is

*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.

left unchanged. On the other hand, if the level of accuracy in $\overline{\nabla^2 f}(x_k)$ with respect to $\nabla^2 f(x_k)$ fulfills the requests (3.10)–(3.11), in Step 5 we proceed for acceptance of the trial steps and update of the regularising parameter as in Algorithm 2.

Summarising, by construction, *Assumption 3.2.2 is satisfied at every successful iteration and at any unsuccessful iteration detected in Step 5.*

Algorithm 6 The ARC algorithm with Dynamic Hessian (ARC-DH) accuracy.

```
Step 0: Initialisation. Given an initial point $x_0$, the initial regulariser $\sigma_0 > 0$, the accuracy
level $\epsilon_1$. Given $\theta, \alpha, \eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3, \sigma_{\min}, c$ s.t.

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

Compute $f(x_0)$ and set $k = 0$, $c_0 = c$, $\mathrm{flag} = 1$.
Step 1: Test for termination. If $\|\nabla f(x_k)\| \le \epsilon_1$, terminate with the current solution $\widehat{x} = x_k$.
Step 2: Hessian approximation. Compute $\overline{\nabla^2 f}(x_k)$ satisfying (3.9).
Step 3: Step computation. Choose $\theta_k \le \theta$. Compute the step $s_k$ satisfying (3.4)– (3.5).
Step 4: Check on $\|s_k\|$.
If $\|s_k\| < 1$ and $\mathrm{flag} = 1$ and $c > \alpha(1-\theta)\|\nabla f(x_k)\|$,
    set $x_{k+1} = x_k$, $\sigma_{k+1} = \sigma_k$,   (unsuccessful iteration)
    set $c_{k+1} = \alpha(1-\theta)\|\nabla f(x_k)\|$, $\mathrm{flag} = 0$,
    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 $\rho_k$ in (1.19).
If $\rho_k \ge \eta_1$,
    define $x_{k+1} = x_k + s_k$, set
        $$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k], & \text{if } \rho_k \ge \eta_2, \quad \textit{(very successful iteration)} \\ [\sigma_k, \gamma_2\sigma_k], & \text{if } \rho_k \in [\eta_1, \eta_2), \quad \textit{(successful iteration)} \end{cases}$$
    If $\|s_k\| \ge 1$, set $c_{k+1} = c$, $\mathrm{flag} = 1$;
        else, set $c_{k+1} = \alpha(1-\theta)\|\nabla f(x_{k+1})\|$, $\mathrm{flag} = 0$.
    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)
    $c_{k+1} = c_k$, $\overline{\nabla^2 f}(x_{k+1}) = \overline{\nabla^2 f}(x_k)$.
    Set $k = k+1$ and go to Step 3.
```

Finally, both flag and c_k are updated at Step 5 as follows. If the iteration is *successful*, we update flag and c_k following (3.10)–(3.11) 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 $\overline{\nabla^2 f}(x_k)$.

The classification of successful and unsuccessful iterations of the ARC-DH algorithm between 0 and k can be made introducing the sets:

$$\begin{aligned} \mathcal{S}_k &= \{\, 0 \le j \le k \mid j \text{ successful in the sense of Step 5} \,\}, && (3.20)\\ \mathcal{U}_{k,1} &= \{\, 0 \le j \le k \mid j \text{ unsuccessful in the sense of Step 5} \,\}, && (3.21)\\ \mathcal{U}_{k,2} &= \{\, 0 \le j \le k \mid j \text{ unsuccessful in the sense of Step 4} \,\}. && (3.22) \end{aligned}$$

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

$$c_k = \mathbb{1}_{W_k}\alpha(1-\theta)\|\nabla f(x_k)\| + (1-\mathbb{1}_{W_k})c,$$

where we set $W_k \overset{\text{def}}{=} \{s_k : \|s_k\| < 1\}$. It follows that if

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

for all x in an open convex set X containing $\{x_k\}$ and some positive κ_g, then

$$c_k \le \max\{c,\, \alpha(1-\theta)\kappa_g\}.$$

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, $j \le k$, has the form

$$\begin{aligned} \sigma_{j+1} &\ge \gamma_1\sigma_j, \quad && j \in \mathcal{S}_k, && (3.23)\\ \sigma_{j+1} &\ge \gamma_2\sigma_j, \quad && j \in \mathcal{U}_{k,1}, && (3.24)\\ \sigma_{j+1} &= \sigma_j, \quad && j \in \mathcal{U}_{k,2}. && (3.25) \end{aligned}$$

As a consequence, the upper bound on the scalars σ_k established in Lemma 21 is still valid.

3.3 Complexity and convergence analysis to first-order critical points

To study the iteration complexity of the ARC-DH algorithm we here consider two possible stopping criteria for the approximate minimisation of model (1.14) at Step 3. Given $\theta \in (0,1)$, the first criterion has the form

$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \le \theta \min\left(\|s_k\|^2, \|\nabla f(x_k)\|\right), \qquad (3.26)$$

which amounts to (3.5) with $\theta_k = \theta \min\left(1, \frac{\|s_k\|^2}{\|\nabla f(x_k)\|}\right)$. The second criterion is considered in [40, (3.28)] and takes the form

$$\|\nabla_s m(x_k, s_k, \sigma_k)\| \le \theta \min(1, \|s_k\|)\|\nabla f(x_k)\|. \qquad (3.27)$$

It corresponds to the choice $\theta_k = \theta \min(1, \|s_k\|)$ in (3.5).

Lemma 22. Let Assumption 1.2.1(ii), Assumption1.2.2 and Assumption 1.2.4 hold. Suppose that $\alpha \in \left[0, \frac{2}{3}\right)$ and $\eta_2 \in \left(0, \frac{2-3\alpha}{2}\right)$ in ARC-DH. Then, given $\epsilon_1 > 0$, at iteration $k \in \mathcal{S}_k \cup \mathcal{U}_{k,1}$:

$$\|s_k\| \geq \sqrt{\zeta \|\nabla f(x_k + s_k)\|}, \tag{3.28}$$

for some positive ζ, both when s_k satisfies (3.26) or (3.27) and the norm of the Hessian is bounded above by a constant κ_H on the path of iterates, i.e.

$$\|\nabla^2 f(x_k + \beta s_k)\| \leq \kappa_H, \quad \forall k \geq 0, \quad \beta \in [0, 1]. \tag{3.29}$$

Proof. Since the assumptions of Lemma 6 are satisfied, (3.26) implies (1.18) and by (3.12) it follows that

$$\|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s_k\| \leq \max(c,\, \alpha(\kappa_B + \sigma_{\max}))\|s_k\|^2.$$

Then, from Lemma 6, we have that (3.28) holds, with

$$\zeta \overset{\text{def}}{=} \left(\max\ (c, \alpha(\kappa_B + \sigma_{\max})) + \frac{L}{3} + \theta + \sigma_{\max}\right)^{-1} > 0.$$

We turn now the attention to the case s_k satisfying (3.27). Combining

$$\nabla f(x_k + s_k) = \nabla f(x_k) + \int_0^1 \nabla^2 f(x_k + ts_k)s_k dt$$

and the boundness of the Hessian, we have

$$\|\nabla f(x_k)\| \leq \|\nabla f(x_k + s_k)\| + \kappa_H \|s_k\|$$

and by (3.27):

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

Thus, (1.41) (with $\chi = \max(c,\, \alpha(\kappa_B + \sigma_{\max}))$) and (1.42) give

$$(1 - \theta)\|\nabla f(x_k + s_k)\| \leq \left(\max\left(c, \alpha(\kappa_B + \sigma_{\max})\right) + L/3 + \theta\kappa_H + \sigma_{\max}\right)\|s_k\|^2,$$

so the claim follows with

$$\zeta \overset{\text{def}}{=} \left(\frac{1 - \theta}{\max\left(c, \alpha(\kappa_B + \sigma_{\max})\right) + L/3 + \theta\kappa_H + \sigma_{\max}}\right) > 0.$$

□

The iteration complexity theorem for first-order critical points can then be proved.

Theorem 23. Suppose that Assumption 1.2.1(i) holds together with the assumptions of Lemma 22. Then, the ARC-DH algorithm requires at most

$$\mathcal{I}_S = \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor \tag{3.30}$$

successful iterations and at most

$$\mathcal{I}_T \;=\; \left\lfloor \kappa_s \frac{f(x_0) - f_{low}}{\epsilon_1^{3/2}} \right\rfloor \left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right) + \lfloor \kappa_u (f(x_0) - f_{low}) \rfloor$$

iterations to produce an iterate $x_{\widehat{k}}$ satisfying $\|\nabla f(x_{\widehat{k}})\| \le \epsilon_1$, with $\kappa_s = \frac{3}{\eta_1 \sigma_{\min} \zeta^{3/2}}$, ζ as in Lemma 22 and $\kappa_u = \frac{3}{\eta_1 \sigma_{\min}}$.

Proof. The mechanism of the ARC-DH algorithm for updating σ_k has the form (3.23)–(3.25). An unsuccessful iteration in $\mathcal{U}_{k,2}$ does not affect the value of the regularisation parameter, as $\sigma_{k+1} = \sigma_k$. Moreover, the assumptions of Lemma 21 hold at iterations $k \in \mathcal{S}_k$. Hence, $\sigma_k \le \sigma_{\max}$, for all $k \ge 0$, due to Lemma 21. Then, proceeding as in the proof of Theorem 8 (see (1.47)), it is still true that before termination

$$f(x_0) - f(x_{k+1}) = \sum_{j \in \mathcal{S}_k} (f(x_j) - f(x_j + s_j)) \ge |\mathcal{S}_k| \kappa_s^{-1} \epsilon_1^{3/2},$$

from which (3.30) immediately follows.
The upper bound on $|\mathcal{U}_{k,1}|$ follows from Lemma 7 and is still given by

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

As for $|\mathcal{U}_{k,2}|$, it is less or equal than the number of successful iterations with $\|s_k\| \ge 1$. By construction, an unsuccessful iteration in $\mathcal{U}_{k,2}$ occurs at most once between two successful iterations with the first one such that $\text{flag} = 1$ and it cannot 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 a successful iteration with $\|s_k\| \ge 1$ (see Step 5 of the ARC-DH algorithm), except for the first iteration. If the case $\text{flag} = 1$ 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 initialised to one, at most one additional unsuccessful iteration in $\mathcal{U}_{k,2}$ may occur before the first successful iteration.
Noting that, by (1.47),

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

we have

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

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

As noticed on top of Subsection 1.2.1, the complexity analysis presented implies

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

The lim-type convergence can then be restored as done in Part I for the basic ARC framework. To do that, further characterisations 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

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

The following theorem is stated.

Theorem 24. Suppose that Assumption 1.2.1(i) holds, together with the assumptions of Theorem 23. Then, the steps s_k and the iterates x_k generated by the ARC-DH algorithm satisfy

$$\|s_k\| \to 0, \quad \text{as } k \to \infty, \quad k \in \mathcal{S}, \tag{3.31}$$

and

$$\|\nabla f(x_k)\| \to 0, \quad \text{as } k \to \infty. \tag{3.32}$$

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

Proof. The first two claims are proved proceeding as in the proof of Thereom 9, replacing ζ^* by ζ defined in Lemma 22. Finally, the behaviur of $\{\|s_k\|\}_{k\in\mathcal{S}}$ (see (3.31)) implies that all successful iterations are eventually such that $\|s_k\| < 1$. Thus, the mechanism of the ARC-DH algorithm gives $\text{flag} = 0$ for all k sufficiently large and unsuccessful iterations in the sense of Step 4 cannot occur. □

3.4 Complexity and convergence analysis to second-order critical points

In this section we focus on the convergence of the sequence generated by our procedure to second-order critical points x^*, i.e. points satisfying (1.3), also exploiting the resulting iteration complexity.

As for the basic ARC framework introduced in Part I, we first analyse the asymptotic behaviour of $\{x_k\}$ in the case where the approximate Hessian $\overline{\nabla^2 f}(x_k)$ becomes positive definite along a converging subsequence of $\{x_k\}$. In such a context, we show quadratic local rate of convergence of $\{x_k\}$, under the additional requirement on the step given by the Cauchy condition (1.28)–(1.29).

Second, we consider the case where $\overline{\nabla^2 f}(x_k)$ is not convex and obtain a second-order complexity bound in accordance with the study of Cartis et al. [38]. That said, the result of Theorem 13 still holds for ARC_SO.

Theorem 25. Suppose that Assumption 1.2.1(i) holds together with the assumptions of Theorem 23. Suppose that $\{x_{k_i}\}$ is a subsequence of successful iterates converging to some x^* and that $\overline{\nabla^2 f}(x_{k_i})$ is 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, for all $k \geq 0$, s_k satisfies (1.28)–(1.29), then all the iterations are eventually successful and $x_k \to x^*$ quadratically.

Proof. The proof is the same as the one of Theorem 13, replacing χ by $\alpha(\kappa_B + \sigma_{\max})$ (see the upper bound in (3.12) when $\|s_k\| < 1$). □

Dropping the assumption that $\overline{\nabla^2 f}(x_k)$ is positive definite, convergence to second-order critical points can be studied. Following [38], where a modification of the ARC algorithm in [40] is proposed, we equip the ARC-DH algorithm with a further stopping criterion and impose an additional condition on the step.

- First, the ARC-DH algorithm is stopped when (1.68) holds, so that when the approximate counterpart of the second-order optimality condition (1.3) with the Hessian matrix approximated by $\overline{\nabla^2 f}(x_k)$ is met. 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 [82] may play a significant role; indeed, (1.68) implies closeness to a local minimum for sufficiently small values of the tolerances ϵ_1 and ϵ_2.
- Second, the trial step s_k computed at Step 3 of the ARC-DH algorithm is required to satisfy (1.69)–(1.71), where the minimisation in (1.70) is global, implying that (1.57)–(1.58) are fulfilled.

We refer to the resulting algorithm as ARC Second Order critical points (ARC_SO). The termination criterion here adopted does not affect the mechanism for updating σ_k, then the upper bound $\sigma_{\max}$ on σ_k given in Lemma 21 is still valid. Since the definition of successul iterations in the ARC-DH algorithm is the same as in the basic Algorithm 2, the thesis of Lemma 14 still holds for ARC_SO.

Lemma 26. Suppose that Assumption 1.2.1(i) holds, together with the assumptions of Theorem 23. Suppose that s_k satisfies (1.69)–(1.71). Then, the number of successful iterations of Algorithm ARC_SO with $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) < -\epsilon_2$ is bounded above by

$$\left\lfloor \kappa_e \frac{f(x_0) - f_{low}}{\epsilon_2^3} \right\rfloor,$$

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

Proof. The proof is the same as the one of Lemma 14. □

We thus conclude that Algorithm ARC_SO produces an iterate $x_{\widehat{k}}$ satisfying (1.68) within at most

$$O\left(\max(\epsilon_1^{-3/2}, \epsilon_2^{-3})\right),$$

iterations, in accordance with the complexity result in [38] and of Algorithm 2.

3.5 The finite-sum minimisation setting

In this section we discuss the application of the ARC-DH algorithm to the finite-sum problem (1.108), where the Hessian matrix at iteration k is approximated via the uniform subsampling procedure described in Chapter 2, i.e. by means of (2.6), giving both deterministic and probabilistic results.

The application of the ARC-DH algorithm to problem (1.108) with such an Hessian approximation is supported by the result (2.12) with $\tau_{2,k} = c_k$, which gives the sample size required to obtain an approximation $\overline{\nabla^2 f}(x_k)$ satisfying (3.9) in probability, with probability at least $(1-t)$, being $t \in (0,1)$ a pre-fixed failure probability.

The adaptive nature of (2.12) is thus implicit in its dependance on the iterate and on c_k, which are dynamically updated during the execution of the algorithm. Depending on the size of N, it may clearly be necessary to consider the whole set $\{1,\dots,N\}$ for small values of $\tau_{2,k}$. In particular, whenever N is large enough to ensure that (2.10)–(2.12) do not require the full sample, the size of the sample used to obtain a single approximate Hessian corresponds to $\mathcal{O}(\tau_{2,k}^{-2})$.

A specular approach based on the Inexact Restoration framework is given in [12] where, in the context of finite-sums problems, the sample size rather than the approximation accuracy is adaptively chosen.

The implementation of the rule (2.12) requires the knowledge of the size of second-order derivatives. If only global information is available, the dependence on x may obviously be avoided by choosing a uniform upper bound $\kappa_{\varphi,2}$ for all $x \in \mathbb{R}^n$, at the cost of a lesser adaptivity. We hereafter assume the existence of $\overline{\kappa}_{\varphi,2} \geq 0$ such that

$$\sup_{x\in\mathbb{R}^n} \kappa_{\varphi,2}(x) \leq \overline{\kappa}_{\varphi,2}, \tag{3.33}$$

yielding $\sup_{x\in\mathbb{R}^n} \|\nabla^2 f(x)\| \leq \overline{\kappa}_{\varphi,2}$ and Assumption 1.2.4 with $\kappa_B = \overline{\kappa}_{\varphi,2}$.

We first give deterministic results, namely properties which are valid independently of Assumption 3.2.2 on $\overline{\nabla^2 f}(x_k)$, now guaranteed with probability $(1-t)$ by Theorem 19. In the following theorem the only requirement on $\overline{\nabla^2 f}(x_k)$ is the boundness of its norm, i.e. Assumption 1.2.4; concerning the trial step s_k, the Cauchy condition (1.28)–(1.29) is assumed.[†]

Theorem 27. Let $f \in C^2(\mathbb{R}^n)$. Suppose that Assumption 1.2.1(i) holds, assume also that (2.7) holds for $j = 2$, together with (3.33) and conditions (1.28)–(1.29). Then,

i) given $\epsilon_1 > 0$, the ARC-DH algorithm takes at most $\mathcal{O}(\epsilon_1^{-2})$ successful iterations to satisfy $\|\nabla f(x_k)\| < \epsilon_1$;

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 [41]. In fact, we can rely on the proof of [41, Lemma 3.2] thanks to (1.67) and considering that

$$f(x_k+s_k) - \widehat{T}_2(x_k,s_k) \leq 2\overline{\kappa}_{\varphi,2}\|s_k\|^2, \quad k \geq 0.$$

[†]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 $\{x_k\}_{k\geq 0}$ and Assumption 1.2.4 holds.

$ii)$ 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 the fact that (2.7) holds for $j = 2$, together with (3.33) and [40, Corollary 2.6].

$iii)$ Proceeding as in Theorem 24 we obtain (3.31). Since $\nabla^2 f(x^*)$ in positive definite, x^* is an isolated limit point; consequently, (3.31) and Lemma 11 yield the claim. □

Focusing on the optimal complexity result, we observe that the ARC-DH algorithm requires at most $\mathcal{O}(\epsilon_1^{-3/2})$ iterations to satisfy $\|\nabla f(x_k)\| \le \epsilon_1$ with probability $1-\bar{t}$, $\bar{t} \in (0,1)$, provided that the sample size is chosen according to (2.12) and t is suitable chosen. In fact, let $\mathcal{E}_i$ be the event: "the relation $\|\nabla^2 f(x_i) - \overline{\nabla^2 f}(x_i)\| \le c_i$ holds at iteration i, $1 \le i \le k$", and $\mathcal{E}(k)$ be the event: "the relation $\|\nabla^2 f(x_i) - \overline{\nabla^2 f}(x_i)\| \le c_i$ holds for the entire k iterations". If the events $\mathcal{E}_i$ are independent, it follows, due to (2.1) for $j = 2$ and $\tau_{2,k} = c_k$, that

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

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

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

Taking into account the iteration complexity, we set $k = \mathcal{O}\big(\epsilon_1^{-3/2}\big)$ and deduce the following choice of t:

$$t = \mathcal{O}(\bar{t}\epsilon_1^{3/2}). \tag{3.34}$$

Summarising, choosing, at each iteration, t according to (3.34) and the sample size according to (2.12), the complexity result in Theorem 23 holds with probability of success $1-\bar{t}$. As also noticed in [124], we underline that the resulting per-iteration failure probability t is not too demanding in what concerns the sample size, because it influences only the logarithmic factor of (2.12).

Observe that (2.12) and $c_k = \alpha(1-\theta)\|\nabla f(x_k)\|$ yield $|\mathcal{H}_k| = \mathcal{O}(\|\nabla f(x_k)\|^{-2})$ as long as N is large enough so that full sample size is not reached. Hence, in the general case, $|\mathcal{H}_k|$ is expected to grow along the iteration and reach values of order ϵ_1^{-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 $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) \ge \underline{\lambda}$ for some positive $\underline{\lambda}$, using (1.66) and Lemma 22 we obtain

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

Then $\|\nabla f(x_k)\| \ge \frac{\sqrt{\zeta\epsilon_1}\underline{\lambda}}{1+\theta}$, provided that the algorithm does not terminate at iteration $k+1$; consequently, $|\mathcal{H}_k|$ is expected to grow along such iterations and eventually reach values of order ϵ_1^{-1}. On the other hand, the sample size for Hessian approximation is expected to be small with respect to $\mathcal{O}(\epsilon_1^{-1})$ 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_1 we can claim that, with probability $1-\bar{t}$, at most $\mathcal{O}(\epsilon_1^{-5/2})$ $\nabla^2\varphi_i$-evaluations are required to compute an ϵ-approximate first-order stationary point, provided that $\lambda_{\min}(\overline{\nabla^2 f}(x_k)) \ge \underline{\lambda}$ 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 (see Section 7.1).

For this subclass of problems, Theorem 13, item $ii)$, also ensures that, for k sufficiently large, say $k \ge \bar{k}$, with probability $(1-t)^{k_0}$, there exists $M > 0$ such that

$$\|x_{k+1} - x^*\| \le M\|x_k - x^*\|^2, \quad k = \bar{k}, \dots, \bar{k}+k_0-1,$$

where x^* the unique minimiser. Specifically, proceeding as in [107, Theorem 2] and denoting by $\mathcal{E}_i$ the event: "the relation $\|\nabla^2 f(x_i) - \overline{\nabla^2 f}(x_i)\| \le c_i$ holds at iteration i, $i \ge \bar{k}$", we have that the overall success probability in consecutive k_0 iterations is

$$P\left(\bigcap_{i=\bar{k}}^{\bar{k}+k_0-1} \mathcal{E}_i\right) = (1-t)^{k_0},$$

which concludes our argument.

3.6 Related works

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 works reviewed in this section.

Besides the algorithms in [40, 41, 38], which employs approximations for the Hessian and are suited for a generic nonconvex function f, papers [50, 44, 79, 124, 125] propose variants of the algorithm given in [40] where the gradient and/or the Hessian approximations can be performed via subsampling techniques [15, 28] and are applicable to large-scale finite-sum minimisation (1.108); probabilistic/stochastic complexity and convergence analysis is carried out.

Cartis et al. [40, 41, 38] analyse the ARC framework under varying assumptions on the Hessian approximation $\overline{\nabla^2 f}(x_k)$ and establish optimal and sub-optimal worst-case iteration bounds for first and second-order optimality. First-order complexity was shown to be of $\mathcal{O}(\epsilon_1^{-2})$ iterations under Assumption 1.2.4 and, as mentioned in Section 3.1, and of $\mathcal{O}(\epsilon_1^{-3/2})$ iterations when, in addition, $\overline{\nabla^2 f}(x_k)$ resembles the true Hessian and condition (1.31) is satisfied.

Kohler et al. [79] propose and study a variant of ARC algorithm suited for finite-sum minimisation not necessarily convex. A subsampling scheme for the gradient and the Hessian of f is applied, maintaining the $\mathcal{O}(\epsilon_1^{-3/2})$ result for first-order complexity. The sampling scheme provided guarantees that the subsampled gradient $\nabla f(x_k)$ satisfies

$$\|\nabla f(x_k) - \nabla f(x_k)\| \le M\|s_k\|^2, \quad \forall k \ge 0,\, M > 0, \tag{3.35}$$

with prefixed probability, while the subsampled Hessian $\overline{\nabla^2 f}(x_k)$ satisfies the condition (1.31) with prefixed probability. As specified in Section 3.1, (1.31) is enforced via (3.1) and, since the step length can be determined only after $\nabla f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ are formed, the step length at the previous iteration is taken.

Cartis and Scheinberg [44] analyse a probabilistic cubic regularisation variant where conditions (3.35) and (1.31) 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 the derivatives accuracy may reach order $\mathcal{O}(\epsilon_1)$ at each iteration (see also [10]).

As mentioned in Section 3.1, Xu et al. [124] develop and study a version of ARC where a major modification on the level of resemblance between $\nabla^2 f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ is made over (1.31). Matrix $\overline{\nabla^2 f}(x_k)$ is supposed to satisfy Assumption 1.2.4 and

$$\|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s_k\| \le \mu\|s_k\|, \quad \mu \in (0,1), \tag{3.36}$$

and the latter condition can be enforced building $\overline{\nabla^2 f}(x_k)$ such that

$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \le \mu, \quad \mu = \chi\epsilon_1, \quad \chi > 0. \tag{3.37}$$

Nonconvex finite-sum minimisation is the motivating application for the proposal and uniform/non-uniform sampling strategies are provided to construct matrices $\overline{\nabla^2 f}(x_k)$ satisfying $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \mu$ with prefixed probability. In particular, unlike the criterion in [79], the rule for choosing the sample size at iteration k does not depend on the step s_k which is not available when $\overline{\nabla^2 f}(x_k)$ has to be built. Worst-case iteration count of order $\epsilon_1^{-3/2}$ is shown when $\mu = \mathcal{O}(\epsilon_1)$, while sub-optimal worst-case iteration count of order ϵ_1^{-2} is achieved if $\mu = \mathcal{O}(\sqrt{\epsilon_1})$. Note that the accuracy requirement on $\overline{\nabla^2 f}(x_k)$ is fixed along the iterations and depends on the accuracy requirement on the gradient norm, that is on the gradient norm at the final iteration. Then, when the Hessian of problem (1.108) is approximated via subsampling with accuracy proportional to ϵ_1, as in [124], $\mathcal{O}(\epsilon_1^{-2})$ evaluations of Hessian components $\nabla^2\varphi_i$ are needed at each iteration, assuming N sufficiently large and, hence, $\mathcal{O}(\epsilon_1^{-7/2})$ total evaluations of the component Hessian functions $\nabla^2\varphi_i$, in the worst-case, to reach an ϵ_1-approximate first-order stationary point. At this regard we underline that, as seen at the end of the previous section, $\mathcal{O}(\epsilon_1^{-5/2})$ of such $\nabla^2\varphi_i$-evaluations are needed in the strongly convex case when our adaptive requirement (3.10)–(3.11) for Hessian approximation is used. Additionally, the use of approximate gradient via subsampling is addressed in [125].

Chen et al. [50] propose an ARC procedure for convex optimisation via random sampling. The objective function f is convex and defined as finite-sum (1.108) of possibly nonconvex functions. Semidefinite positive subsampled approximations $\overline{\nabla^2 f}(x_k)$ satisfying $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \mu_k$, $\mu_k \in (0,1)$, are built with a prefixed probability. Iteration complexity of order $\mathcal{O}(\epsilon_1^{-1/3})$ is proved with respect to the fulfillment of condition $f(x_k) - f(x^*) \leq \epsilon_1$, x^* being the global minimum of (1.108); the scalar μ_k is updated as $\mu_{k+1} = \mathcal{O}(\min(\mu_k, \|\nabla f(x_k)\|))$, and the model $m_k(s)$ is minimised on a subspace of $\mathbb{R}^n$ imposing the strict condition $\|\nabla_s m_k(s)\| \leq \theta \min(\|\nabla f(x_k)\|, \|\nabla f(x_k)\|^3, \|s_k\|^2)$, $\theta \in (0,1)$.

Summarising, our proposal differs from the above works in the following respects.
In [79] the upper bound in (3.1) is replaced by a bound computed using information from the previous iteration and no check on the fulfillment of (3.1) is made, while in [44] the error in Hessian approximation is dynamically reduced to fulfill (3.1); on the contrary, our accuracy requirement c_k is computable and the condition (1.31) is satisfied at every successful iteration and at any unsuccessful iteration detected in Step 5, without deteriorating the computational complexity. Our proposal improves upon [124, 125] in the construction of $\overline{\nabla^2 f}(x_k)$, as the level of resemblance between $\nabla^2 f(x_k)$ and $\overline{\nabla^2 f}(x_k)$ is not maintained fixed along iterations but adaptively chosen, remaining less stringent than the first-order ϵ_1 tolerance when the constant accuracy c is selected by the adaptive procedure or, otherwise, whether the current gradient norm is sufficiently high (see, e.g., (3.9)–(3.11)); it improves upon [50] as the prescribed accuracy on $\overline{\nabla^2 f}(x_k)$ (and the sample size) may reduce at some iteration, but the ultimate accuracy on $\|\nabla_s m_k(s_k)\|$ is milder, and our complexity results are optimal for nonconvex problems, while the analysis in [50] is limited to convex problems.

3.7 Chapter conclusion

We proposed an ARC algorithm for solving nonconvex optimisation problems based on a dynamic rule for building inexact Hessian information.

The new algorithm maintains the distinguishing features of the ARC framework, i.e., the optimal worst-case iteration bound for first and second-order critical points.

Application to large-scale finite-sum minimisation is sketched and analysed. 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.

Convergence properties are analysed 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 any indication on the properties of the method when the adaptive accuracy requirement is not satisfied at some iteration. A stochastic analysis, in the spirit of [44], would be handled in Part III of this thesis as a new relevant contribution on the topic, coupled with an extension of the ARC-DH algorithm to the case of inexact gradient evaluations.

Chapter 4

Adaptive Regularisation Methods with Inexact Function and Derivatives Evaluations

This chapter goes further into the analysis of ARC frameworks with inexact function and derivatives computations. A regularisation algorithm using inexact function values and inexact derivatives is proposed and its evaluation complexity carried out.

This algorithm is applicable to unconstrained problems and to problems with inexpensive constraints, that is constraints whose evaluation and enforcement has negligible cost with respect to the objective function evaluation and its derivatives, under the assumption that the derivative of highest degree is globally Lipschitz continuous. It features a very flexible adaptive mechanism for determining the inexactness which is allowed, at each iteration, when computing objective function values and derivatives.

For a better readability and a more practical application, the method is exposed considering convergence up to a q-th order optimality point, with $q \in \{1, 2\}$, using at iteration k a regularised model model of order $p+1$, based on a regularisation term of order $p+1$ and a truncated Taylor's expansion of order p, for $1 \le p \le q$:

$$\begin{aligned} m_k^{(p)}(s) &\overset{\text{def}}{=} \overline{f}(x_k) + \sum_{\ell=1}^{p} \frac{1}{\ell!} \overline{\nabla^\ell f}(x_k)[s]^\ell + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1} \\ &\overset{\text{def}}{=} \overline{T}_p^f(x_k, s) + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1} \qquad (4.1) \\ &\overset{\text{def}}{=} \overline{f}(x_k) - \overline{\Delta T}_p^f(x_k, s) + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1}, \qquad (4.2) \end{aligned}$$

in which $\overline{f}(x_k)$ and $\overline{\nabla^\ell f}(x_k)$ represent approximations of $f(x_k)$ and $\nabla^\ell f(x_k)$, respectively, for $\ell \in \{1, 2\}$. In other words, here, beside the inexact cubic model $m_k^{(2)}(s)$, we are also considering the quadratic model given by $m_k^{(1)}(s)$, i.e. adding the quadratic regularisation term $\sigma_k\|s\|^2/2$ to the first-order truncated Taylor's expansion $\overline{T}_1^f(x_k, s)$. Nevertheless, the complexity analysis presented in the chapter (see [10]) can be extended to arbitrary order q of optimality using an arbitrary model degree of order $p+1$, with $1 \le q \le p$, of available approximate first p derivatives, under the assumption that the derivative of highest degree is globally Hölder continuous[*]. In this sense, it can be seen as an extension of the unifying framework of [35] to the evaluation complexity for the inexact case. In particular, the proposed framework potentially allows all combinations of exact/inexact objective functions and derivatives of any order, including of course models of orders $p+1$ equals to two and three.

Concerning evaluation complexity, if a q-th-order minimiser is sought using approximations to the first p derivatives, with $1 \leq q \leq p \leq 2$, it is proved that a suitable approximate minimiser within ϵ is computed by the proposed algorithm in at most $\mathcal{O}\big(\epsilon^{-\frac{p+1}{p-q+1}}\big)$ iterations and at most $\mathcal{O}\big(|\log(\epsilon)|\epsilon^{-\frac{p+1}{p-q+1}}\big)$ approximate evaluations, as reported in the table below.

q	p	iteration and approximate f evaluations	approximate evaluations of the first p derivatives
1	1	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\lvert\log\epsilon\rvert\epsilon^{-2})$
1	2	$\mathcal{O}(\epsilon^{-3/2})$	$\mathcal{O}(\lvert\log\epsilon\rvert\epsilon^{-3/2})$
2	2	$\mathcal{O}(\epsilon^{-3})$	$\mathcal{O}(\lvert\log\epsilon\rvert\epsilon^{-3})$

Table 4.1: Iteration and evaluation complexity bounds for ARqpDA, given $1 \leq q \leq p \leq 2$.

An algorithmic variant, although more rigid in practice, can be proved to find such an approximate minimiser in $\mathcal{O}\big(|\log(\epsilon)|+\epsilon^{-\frac{p+1}{p-q+1}}\big)$ evaluations. As for the previous chapter, the deterministic complexity results are finally extended to the probabilistic context, yielding adaptive sample size rules for subsampling (see Chapter 2) methods.

The outline of the current chapter is as follows. Section 4.1 recalls the notions of high-order optimality proposed in [35] and introduces the general Adaptive Regularization algorithm with model of order $p+1$ allowing Dynamic Accuracy (ARqpDA). The details of how to obtain the desired relative accuracy levels from known absolute errors are examined in Section 4.2. The evaluation complexity of obtaining approximate minimisers using this algorithm is then analysed in Section 4.3, while an algorithmic variant of the algorithm is discussed in Section 4.4. The general framework is then explicitly specialised to first-order optimisation (case $q=p=1$) in Section 4.5, showing that practical implementation for low orders is simple. The stochastic evaluation complexity and sampling rules for machine learning applications are finally derived in Section 4.6.

4.1 High-order necessary conditions and the ARqpDA algorithm

Given $p \in \{1,2\}$, we consider the set-constrained optimisation problem

$$\min_{x\in\mathcal{X}} f(x), \tag{4.3}$$

where we assume that the values of the objective function f and its derivatives cannot be computed exactly, so that inexact approximations are considered. The feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ is closed and nonempty and represents *inexpensive constraints*, that is constraints such that their evaluation/enforcement has negligible cost compared to that of computing the objective function and its derivative. Such constraints include (but are not limited to)

[*]The p-th derivative tensor of f at x is globally Hölder continuous if there exist constants $L \geq 0$ and $\beta \in (0,1]$ such that, for all $x, y \in \mathbb{R}^n$,

$$\|\nabla_x^p f(x) - \nabla_x^p f(y)\|_{[p]} \leq L\|x-y\|^{\beta}.$$

The more standard case where f is assumed to have Lipschitz-continuous p-th derivative is recovered by setting $\beta = 1$ in the above assumptions (for example, the choices $p=2$ and $\beta=1$ correspond to the assumption that f has a Lipschitz continuous Hessian).

bound constraints and other convex constraints with cheap projections. Unconstrained problems are obviously also covered by this definition.

Similarly to Assumption 1.2.1 and Assumption 1.2.2, let us regroup three fundamental hypotheses on the objective function f in the following assumption.

Assumption 4.1.1. *With reference to* (4.3)*, given $p \in \{1,2\}$, we assume that:*

1. *f is p-times continuously differentiable,*
2. *f is bounded below by $f_{\rm low}$, and*
3. *the p-th derivative of f at x is globally Lipschitz continuous, that is, there exist constants $L \geq 0$ such that, for all $x, y \in \mathbb{R}^n$,*

$$\|\nabla_x^p f(x) - \nabla_x^p f(y)\| \leq L\|x-y\|. \tag{4.4}$$

As usual, the p-th degree Taylor's expansion of f around x evaluated at s is still denoted by

$$T_p^f(x,s) \overset{\rm def}{=} f(x) + \sum_{\ell=1}^{p} \frac{1}{\ell!}\nabla^\ell f(x)[s]^\ell = \begin{cases} f(x) + \nabla f(x)^\top s, & \text{if } p=1, \\ f(x) + \nabla f(x)^\top s + \frac{1}{2}s^\top \nabla^2 f(x) s, & \text{if } p=2. \end{cases} \tag{4.5}$$

we may then define the Taylor's increment by

$$\Delta T_p^f(x,s) \overset{\rm def}{=} T_p^f(x,0) - T_p^f(x,s) = -\sum_{\ell=1}^{p} \frac{1}{\ell!}\nabla^\ell f(x)[s]^\ell, \quad p \in \{1,2\}. \tag{4.6}$$

Therefore, for $p \in \{1,2\}$, the model (4.1) (the superscript p is hereafter omitted for simplicity of notation) takes the form:

$$m_k(s) = \overline{T}_p^f(x_k,s) + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1} = \begin{cases} \overline{f}(x) + \overline{\nabla f}(x)^\top s + \frac{\sigma_k}{2}\|s\|^2, & \text{if } p=1, \\ \\ \overline{f}(x) + \overline{\nabla f}(x)^\top s + \frac{1}{2}s^\top \overline{\nabla^2 f}(x) s + \frac{\sigma_k}{3}\|s\|^3, & \text{if } p=2. \end{cases} \tag{4.7}$$

Under the above assumptions, we recall the bounds on differences between f and its derivatives and their Taylor's expansion, proved in [35].

Lemma 28. [35, Lemma 2.1] Let Assumption 4.1.1 hold and let $T_p^f(x,s)$ be the Taylor's approximation of $f(x+s)$ around x given by (4.5). Then for all $x, s \in \mathbb{R}^n$,

$$|f(x+s) - T_p^f(x,s)| \leq \frac{L}{(p+1)!}\|s\|^{p+1}, \tag{4.8}$$

$$\|\nabla_x^j f(x+s) - \nabla_s^j T_p^f(x,s)\| \leq \frac{L}{(p-j+1)!}\|s\|^{p+1-j}. \quad (j=1,\dots,p). \tag{4.9}$$

We underline that (1.10) ensures (4.8) when $p=2$ (third-order model) is considered. We also follow [35] and define a q-th-order necessary minimiser, $q \in \{1,2\}$, as a point $x \in \mathbb{R}^n$ such that, for some $\delta \in (0,1]$,

$$\phi_{f,q}^\delta(x) \overset{\rm def}{=} f(x) - \operatorname*{globmin}_{\substack{x+d \in \mathcal{X} \\ \|d\| \leq \delta}} T_q^f(x,d) = \operatorname*{globmax}_{\substack{x+d \in \mathcal{X} \\ \|d\| \leq \delta}} \Delta T_q^f(x,d) = 0. \tag{4.10}$$

It is worth noting that, in the unconstrained case, $\phi_{f,q}(x)$ represents the maximal decrease in $T_q^f(x,d)$ achievable in a ball of radius δ centered at x and it can be seen as a more compact way of addressing the classical first and second-order necessary optimality conditions, since it subsumes to them.

- If $q = 1$, (4.10) gives that, for any $\delta \in (0, 1]$ (and in particular for $\delta = 1$),

$$\phi^{\delta}_{f,q}(x) = \|\nabla f(x)\|\delta = 0, \tag{4.11}$$

and first-order optimality is thus equivalent to

$$\|\nabla f(x)\| = 0.$$

- Similarly, for $q = 2$, (4.10) gives

$$\phi^{\delta}_{f,q}(x) = \underset{\substack{x+d\in\mathcal{X}\\ \|d\|\leq\delta}}{\text{globmax}} \left(-\nabla f(x)^{\top} d - \frac{1}{2} d^{\top} \nabla^2 f(x) d \right) = 0,$$

being equivalent to $\|\nabla f(x)\| = 0$ and the semi-positiveness of $\nabla^2 f(x)$, i.e.,

$$\|\nabla f(x)\| = 0 \quad \text{and} \quad \lambda_{\min}[\nabla^2 f(x)] \geq 0. \tag{4.12}$$

,

Remark 8 (Higher order optimality)**.** The properties of (4.10) are further discussed in [35], but we emphasise that, for any $q \geq 1$ and in contrast with other known measures, it varies continuously as x varies continuously in $\mathcal{X}$. In the unconstrained case, solving the global optimisation problem involved in its definition is easy for $q = 1$, as the global minimiser is analytically given by $d_* = -\delta\,\nabla f(x)/\|\nabla f(x)\|$, and also for $q = 2$ using a trust-region scheme (whose cost is essentially comparable to that of computing the leftmost eigenvalue in (4.12)).

Let us turn to the case $1 \leq q \leq p \leq 2$. If we now relax the notion of exact minimisers, we may define an (ϵ, δ)-approximate q-th-order necessary minimiser as a point $x \in \mathbb{R}^n$

$$\phi^{\delta}_{f,q}(x) \leq \epsilon \chi_q(\delta), \tag{4.13}$$

where

$$\chi_q(\delta) \stackrel{\text{def}}{=} \sum_{\ell=1}^{q} \frac{\delta^{\ell}}{\ell!} = \begin{cases} \delta, & \text{if } q = 1, \\ \delta + \frac{\delta^2}{2}, & \text{if } q = 2. \end{cases} \tag{4.14}$$

provides a natural scaling. In other words, an (ϵ, δ)-approximate q-th-order necessary minimiser is a point from which no significant decrease of the Taylor's expansion of degree q can be obtained in a ball of optimality radius δ, within the constraint. This notion has the advantage of being well-defined and continuous in x. By definition, the following bounds hold for $\chi_q(\delta)$ in (4.14), $q \in \{1, 2\}$:

$$1 \;\leq\; \chi_q(\delta) \leq \frac{3}{2}\delta^q, \quad \text{if} \quad \delta \geq 1, \tag{4.15}$$

$$\delta \;\leq\; \chi_q(\delta) \leq \frac{3}{2}\delta, \quad \text{if} \quad \delta \in [0, 1]. \tag{4.16}$$

Again, the notion reduces to familiar concepts in the low-order unconstrained cases. For instance, we verify that for unconstrained problems with $q = 2$, (4.13) requires that, if d is the global minimiser in (4.10) (the solution of a trust-region problem),

$$\max\left[0, -\left(\nabla f(x)^T d + \tfrac{1}{2} d^T \nabla^2 f(x) d\right)\right] \leq \epsilon(\delta + \tfrac{1}{2}\delta^2),$$

which automatically holds for any $\delta \in (0,1]$ if $\|\nabla f(x)\| \leq \epsilon$ and $\lambda_{\min}[\nabla^2 f(x)] \geq -\epsilon$. We note that, when assessing whether x is an (ϵ, δ)-approximate q-th-order necessary minimiser, the global maximisation in (4.10) can be stopped as soon as $\Delta T_q^f(x,d)$ exceeds $\epsilon \chi_q(\delta)$, thereby significantly reducing the cost of this assessment.

Having defined what we mean by high-order approximate minimisers, we now turn to describe what we mean by inaccurate objective function and derivatives values. It is important to observe at this point that, in an optimisation problem, the role of the objective function is more central than that of any of its derivatives, since it is the quantity we ultimately wish to decrease. For this reason, we will handle the allowed inexactness in f differently from that in $\nabla^j f$, $j \in \{1,..,p\}$: we will require an (adaptive) absolute accuracy for the first and a relative accuracy for the second. In fact, we can, in a first approach, abstract the relative accuracy requirements for the derivatives $\nabla^j f(x)$ into a requirement on the relative accuracy of $\Delta T_p^f(x,s)$.

Let $\omega \in [0,1)$ represent a relative accuracy level, $p \in \{1,2\}$ and denote inexact quantities by an overbar. For what follows, we will thus require that, if

$$\overline{\Delta T}_p^f(x,s) = \overline{T}_p^f(x_k,0) - \overline{T}_p^f(x_k,s), \tag{4.17}$$

then

$$|\overline{\Delta T}_p^f(x,s) - \Delta T_p^f(x,s)| \leq \omega \overline{\Delta T}_p^f(x,s). \tag{4.18}$$

It may not be obvious at this point how to enforce this relative error bound: this is the object of Section 4.2 below. For now, we simply assume that it can be done in a finite number of evaluations of $\{\overline{\nabla_x^j f(x)}\}_{j=1}^p$ which are inexact approximations of $\{\nabla_x^j f(x)\}_{j=1}^p$.

Given an inexactly computed $\overline{\Delta T}_p^f(x,s)$ satisfying (4.18), we then have to consider to compute our optimality measure inexactly too. Observing that the definition (4.10) is independent of $f(x)$ because of cancellation, we see that, for $1 \leq q \leq p \leq 2$,

$$\overline{\phi}_{f,q}^{\delta}(x) = \max\left[0, \operatorname*{globmax}_{\substack{x+d\in\mathcal{X}\\ \|d\|\leq\delta}} \overline{\Delta T}_q^f(x,d)\right]. \tag{4.19}$$

Under the above assumptions, we now describe an algorithm allowing inexact computation of both the objective function and its derivatives, whose purpose is to find (for given $q \in \{1,2\}$ and a suitable relative accuracy ω) a point x_k satisfying

$$\overline{\phi}_{f,q}^{\delta}(x) \leq \frac{\epsilon}{1+\omega}\chi_q(\delta), \tag{4.20}$$

for some optimality radius $\delta \in (0,1]$. We will show that, at termination, the computed approximation x_ϵ is an (ϵ,δ)-approximate q-th-order necessary minimiser, $q \in \{1,2\}$, i.e. it satisfies (4.13). This algorithm uses a regularised truncated and approximated Taylor's expansion of order $p \in \{1,2\}$, defined at iteration k by (4.1) (or, equivalently, by (4.7)). The superscript p in the model definition is hereafter omitted for simplicity of notation. This model is then approximately minimised and the resulting trial point is then accepted or rejected depending on whether or not it produces a significant decrease. This is detailed in the ARqpDA algorithm on the next page.

Algorithm 7 The Adaptive Regularisation algorithm with Dynamic Accuracy (ARqpDA), $1 \leq q \leq p \leq 2$.

Step 0: Initialisation. An initial point $x_0 \in \mathcal{X}$ and an initial regularisation parameter $\sigma_0 > 0$ are given, as well as an accuracy level $\epsilon \in (0,1)$ and a relative accuracy $\omega > 0$. The constants $\delta_{-1}, \theta, \mu, \eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ and $\sigma_{\min}$ are also given and satisfy $\theta > 0$, $\mu \in (0,1]$, $\delta_{-1} \in (0,1]$, $\sigma_{\min} \in (0, \sigma_0]$,

$$0 < \eta_1 \leq \eta_2 < 1, \;\; 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3, \;\; \alpha \in (0,1), \tag{4.21}$$

$$0 < \omega < \min\left[\frac{1-\eta_2}{3}, \frac{\eta_1}{2} < 1\right]. \tag{4.22}$$

Set $k = 0$.

Step 1: Compute the optimality measure and check for termination. Compute $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$. If (4.20) holds with $\delta = \delta_{k-1}$, terminate with the approximate solution $x_\epsilon = x_k$.

Step 2: Step calculation. Compute a step $s_k \neq 0$ such that $x_k + s_k \in \mathcal{X}$ and an optimality radius $\delta_k \in (0,1]$ by approximately minimising the model $m_k(s)$, in the sense that

$$m_k(s_k) < m_k(0) \tag{4.23}$$

and

$$\|s_k\| \geq \mu \epsilon^{\frac{1}{p-q+1}} \quad \text{or} \quad \overline{\phi}_{m_k,q}^{\delta_k}(s_k) \leq \frac{\theta \|s_k\|^{p-q+1}}{(p-q+1)!}\, \chi_q(\delta_k). \tag{4.24}$$

Step 3: Acceptance of the trial point. Compute $\overline{f}_k(x_k + s_k)$ ensuring that

$$|\overline{f}_k(x_k + s_k) - f(x_k + s_k)| \leq \omega |\overline{\Delta T}_p^f(x_k, s_k)|. \tag{4.25}$$

Also ensure (by setting $\overline{f}_k(x_k) = \overline{f}_{k-1}(x_k)$ or by (re)computing $\overline{f}_k(x_k)$) that

$$|\overline{f}_k(x_k) - f(x_k)| \leq \omega |\overline{\Delta T}_p^f(x_k, s_k)|. \tag{4.26}$$

Then compute

$$\rho_k = \frac{\overline{f}_k(x_k) - \overline{f}_k(x_k + s_k)}{\overline{\Delta T}_p^f(x_k, s_k)}. \tag{4.27}$$

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

Step 4: Regularisation parameter update. Set

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k] & \text{if } \rho_k \geq \eta_2, & (\textit{very successful iteration}) \\ [\sigma_k, \gamma_2 \sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), & (\textit{successful iteration}) \\ [\gamma_2 \sigma_k, \gamma_3 \sigma_k] & \text{if } \rho_k < \eta_1. & (\textit{unsuccessful iteration}) \end{cases} \tag{4.28}$$

Increment k by one and go to Step 1.

Comments on the ARqpDA algorithm

Some comments on this algorithm are useful at this stage.

1. We underline that at each occurrence of Step 2 before termination we have that a nonzero step s_k can be found. In this situation, in fact, $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$ is nonzero (it even does not satisfy (4.20) with $\delta = \delta_{k-1}$), that is letting d the global minimiser of $\overline{T}_q^f(x_k, d)$ (required for evaluating $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$), it follows that the inexact Taylor's series $\overline{T}_q^f(x_k, \beta d)$ decreases monotonically for $\beta \in [0,1]$, thanks to the fact that $q \in \{1,2\}$ and, thus, a decrease of the regularised model (4.7) is possibile for small enough β.

2. Our assumption (4.18) is used three times in the algorithm: at Step 1 for computing $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$, at Step 2 when computing s_k and $\overline{\phi}_{m_k,q}^{\delta_k}(s_k)$.

3. As indicated above, we require a bound on the absolute error in the objective function value: this is the object of (4.25) and (4.26), where we introduced the notation $\overline{f}_k(x_k)$ to denote an inexact approximation of $f(x_k)$. Note that a new value of $\overline{f}_k(x_k)$ should be computed to ensure (4.26) in Step 3 only if $k > 0$ and $\omega\overline{\Delta T}_p^f(x_{k-1}, s_{k-1}) > \omega\overline{\Delta T}_p^f(x_k, s_k)$. If this is the case, the (inexact) function value is computed twice per iteration instead of just once.

4. Contrary to the Trust-Region method with dynamic accuracy of [52, Section 10.6] and [13], we do not recompute approximate values of the objective function at x_k once the computation of s_k is complete (provided we can ensure (4.18), as discussed in Section 4.2).

5. If $\|s_k\| \geq \mu\epsilon^{\frac{1}{p-q+1}}$ in Step 2, then the (potentially costly) calculation of $\overline{\phi}_{m_k,q}^{\delta_k}(s_k)$ is unnecessary and δ_k may be chosen arbitrarily in $(0, 1]$.

6. We call iteration k successful when $\rho_k \geq \eta_1$ and $x_{k+1} = x_k + s_k$. The iteration is called unsuccessful otherwise, setting $x_{k+1} = x_k$ in this case. As usual, we use the notation (1.43) to denote the set $\mathcal{S}_k$ of successful and $\mathcal{U}_k$ of unsuccessful iterations of index at most k, respectively, generated by the ARqpDA algorithm.

7. As indicated above, ensuring (4.18) may require a certain number of (approximate) evaluations of the derivatives of f. For a single iteration of the algorithm, these evaluations are always at the current iterate x_k.

Since the mechanism for updating the regulariser σ_k in (6.8) is the same as (1.20) in Algorithm 2, Lemma 7 still holds, so that the number of unsuccessful iterations $|\mathcal{U}_k|$ until iteration k remains a fixed proportion of that of the successful ones $|\mathcal{S}_k|$.

We now state a lemma ensuring that ARqpDA is well-defined when $s_k \neq 0$, that can be derived without modification from the case where the computation of f and its derivatives are exact.

Lemma 29. [35, Lemma 2.5] Suppose that $s_k^* \neq 0$ is a global minimiser of $m_k(s)$ under the constraint that $x_k + s \in \mathcal{X}$ such that $m_k(s_k^*) < m_k(0)$. Then, there exists a neighbourhood of s_k^* and a range of sufficiently small δ such that (4.23) and the second part of (4.24) hold for any s_k in the intersection of this neighbourhood with $\mathcal{X}$ and any δ_k in this range.

4.2 Enforcing the relative error on Taylor's increments

We now return to the question of enforcing (4.18). For improved readability, we temporarily ignore the iteration index k.

4.2.1 The accuracy checks

While there may be circumstances where (4.18) can be enforced directly, we here consider that the only control the user has on the accuracy of $\overline{\Delta T}_p^f(x, s)$ is by enforcing bounds $\{\varepsilon_j\}_{j=1}^p$ on the absolute errors on the derivatives $\{\nabla^j f(x)\}_{j=1}^p$, $p \in \{1, 2\}$. In other words, we seek to ensure (4.18) by selecting absolute accuracies $\{\varepsilon_j\}_{j=1}^p$ such that, when

$$\|\overline{\nabla^j f}(x) - \nabla^j f(x)\| \leq \varepsilon_j \;\text{ for }\; j \in \{1,\dots,p\}, \quad p \in \{1,2\}, \tag{4.29}$$

the desired accuracy requirement follows.

In all cases described below, the process can be viewed as an iteration with four main steps.

- The first is to compute the relevant approximate derivative satisfying (4.29) for given values of $\{\varepsilon_j\}_{j=1}^p$.
- The second is to use these approximate derivatives to compute the requested Taylor's increment and associated quantities.
- Tests are then performed in the third step to verify the desired accuracy requirements and terminate if they are met.
- If not the case, the absolute accuracies $\{\varepsilon_j\}_{j=1}^p$ are then decreased before a new iteration is started.

We then formalise the resulting accuracy tests in the VERIFY algorithm, stated as Algorithm 8 on the facing page.

Assume that for a vector v_ω, a bound $\delta \geq \|v_\omega\|$, a degree r, the prescribed relative and absolute accuracies ω and $\xi > 0$, the increment $\overline{\Delta T}_r(x, v_\omega)$ are given. We intend to use the algorithm for $\overline{\Delta T}_q^f(x, v_\omega)$, $\overline{\Delta T}_p^f(x, v_\omega)$ and $\overline{\Delta T}_q^{m_k}(x, v_\omega)$, in which, recalling the p-dependent definition of m_k in (4.7),

$$\overline{T}_q^{m_k}(x, v_\omega) = \begin{cases} m_k(x) + \overline{\nabla f}(x_k)^\top v_\omega + \frac{\sigma_k}{2} x^\top v_\omega, & \text{if } q = p = 1, \\ m_k(x) + \overline{\nabla f}(x_k)^\top v_\omega + x^\top \overline{\nabla^2 f}(x_k) v_\omega + \frac{\sigma_k}{2}\|x\| x^\top v_\omega, & \text{if } q = 1,\ p = 2, \\ \overline{T}_1^{m_k}(x, v_\omega) + \frac{1}{2} v_\omega^\top \overline{\nabla^2 f}(x_k) v_\omega + \frac{\sigma_k}{2\|x\|}\|x^\top v_\omega\|^2, & \text{if } q = p = 2. \end{cases} \tag{4.30}$$

$$\overline{\Delta T}_q^{m_k}(x, v_\omega) \overset{\text{def}}{=} m_k(x) - \overline{T}_q^{m_k}(x, v_\omega) = \overline{T}_q^{m_k}(x, 0) - \overline{T}_q^{m_k}(x, v_\omega), \quad q \in \{1,2\}. \tag{4.31}$$

The corresponding quantities using exact evaluations $\nabla^j f(x_k)$, $j \in \{0,1,2\}$, are denoted by $T_q^{m_k}(x, v_\omega)$ and $\Delta T_q^{m_k}(x, v_\omega)$, $q \in \{1,2\}$. For keeping our development general, in Algorithm 8 and Lemma 30 we use the notation $\overline{\Delta T}_r(x, v_\omega)$ and $\Delta T_r(x, v_\omega)$, $r \in \{1,2\}$. Moreover, we assume that the current absolute accuracies $\{\zeta_j\}_{j=1}^r$ of the derivatives of $\overline{T}_r(x, v_\omega)$ with respect to v_ω at $v_\omega = 0$ are given. Because it will be the case below, we assume for simplicity that $\overline{\Delta T}_r(x, v_\omega) \geq 0$.

As can be expected, a suitable relative accuracy requirement will be achievable as long as $\overline{\Delta T}_p^f(x, s)$ remains safely away from zero; but, if exact computations are to be avoided, we may have to accept a simpler absolute accuracy guarantee when $\overline{\Delta T}_p^f(x, s)$ vanishes.

Algorithm 8 Verify the accuracy of $\overline{\Delta T}_r(x, v_\omega)$.

$$\texttt{flag} = \text{VERIFY}\Big(\delta, \overline{\Delta T}_r(x, v_\omega), \{\zeta_j\}_{j=1}^r, \omega, \xi\Big)$$

Set $\texttt{flag} = 0$.

- If
$$\overline{\Delta T}_r(x, v_\omega) = 0 \quad \text{and} \quad \max_{j\in\{1,\dots,r\}} \zeta_j \le \xi, \tag{4.32}$$
set flag = 1.
- Else, if
$$\overline{\Delta T}_r(x, v_\omega) > 0 \quad \text{and} \quad \sum_{j=1}^r \frac{\zeta_j}{j!}\delta^j \le \omega\overline{\Delta T}_r(x, v_\omega), \tag{4.33}$$
set flag = 2.
- Else, if
$$\overline{\Delta T}_r(x, v_\omega) > 0 \quad \text{and} \quad \sum_{j=1}^r \frac{\zeta_j}{j!}\delta^j \le \xi\chi_r(\delta), \tag{4.34}$$
set flag = 3.

Let us now consider what properties are ensured for the various possible values of flag.

Lemma 30. Suppose that

$$\left\| \left[\overline{\nabla^j_{v_\omega} T_r}(x, v_\omega)\right]_{v_\omega=0} - \left[\nabla^j_{v_\omega} T_r(x, v_\omega)\right]_{v_\omega=0} \right\|_{[j]} \le \zeta_j \ \text{ for } \ j \in \{1, \dots, r\} \tag{4.35}$$

and $\omega \in (0,1)$. We then have that:

- if
$$\max_{j\in\{1,\dots,r\}} \zeta_j \le \xi, \tag{4.36}$$
then the VERIFY algorithm returns a nonzero flag;
- if the VERIFY algorithm terminates with $\texttt{flag} = 1$, then $\overline{\Delta T}_r(x, v_\omega) = 0$ and
$$\left|\overline{\Delta T}_r(x, v) - \Delta T_r(x, v)\right| \le \xi\chi_r(\|v\|), \qquad \text{for all } v; \tag{4.37}$$
- if the VERIFY algorithm terminates with $\texttt{flag} = 2$, then $\overline{\Delta T}_r(x, v_\omega) > 0$ and
$$\left|\overline{\Delta T}_r(x, v) - \Delta T_r(x, v)\right| \le \omega\overline{\Delta T}_r(x, v_\omega), \qquad \text{for all } v \text{ with } \|v\| \le \delta; \tag{4.38}$$
- if the VERIFY algorithm terminates with $\texttt{flag} = 3$, then $\overline{\Delta T}_r(x, v_\omega) > 0$ and
$$\max\left[\overline{\Delta T}_r(x, v_\omega), \left|\overline{\Delta T}_r(x, v) - \Delta T_r(x, v)\right|\right] \le \frac{\xi}{\omega}\chi_r(\delta) \ \text{ for all } v \text{ with } \|v\| \le \delta. \tag{4.39}$$

Proof. We first prove the first proposition. If $\overline{\Delta T}_r(x, v_\omega) = 0$ and (4.36), then (4.32) ensures that $\texttt{flag} = 1$ is returned. If $\overline{\Delta T}_r(x, v_\omega) > 0$, from (4.14) and (4.36) we deduce that

$$\sum_{j=1}^r \frac{\zeta_j}{j!}\delta^j \le \Big[\max_{j\in\{1,\dots,r\}} \zeta_j\Big]\chi_r(\delta) \le \xi\chi_r(\delta),$$

also causing termination with $\texttt{flag} = 3$ because of (4.34), if it has not occurred with $\texttt{flag} = 2$ because of (4.33), hence proving the first proposition.

Consider now the three possible termination cases and suppose first that termination occurs with `flag` = 1. Then, using the triangle inequality, (4.35), (4.32) and (4.14), we have that, for any v,

$$\left|\overline{\Delta T}_r(x,v) - \Delta T_r(x,v)\right| \le \sum_{j=1}^{r} \frac{\zeta_j}{j!}\|v\|^j \le \xi\chi_r(\|v\|),$$

yielding (4.37). Suppose now that `flag = 2`. It follows that (4.33) holds and for any v with $\|v\| \le \delta$,

$$\left|\overline{\Delta T}_r(x,v) - \Delta T_r(x,v)\right| \le \sum_{j=1}^{r} \frac{\zeta_j}{j!}\|v\|^j \le \sum_{j=1}^{r} \frac{\zeta_j}{j!}\delta^j \le \omega\overline{\Delta T}_r(x,v_\omega),$$

which is (4.38). Suppose finally that `flag = 3`. Since termination did not occur in (4.33), we have that

$$0 < \omega\overline{\Delta T}_r(x,v_\omega) \le \xi\chi_r(\delta). \tag{4.40}$$

Furthermore, (4.34) implies that, for any v with $\|v\| \le \delta$,

$$\left|\overline{\Delta T}_r(x,v) - \Delta T_r(x,v)\right| \le \sum_{j=1}^{r} \frac{\zeta_j}{j!}\|v\|^j \le \sum_{j=1}^{r} \frac{\zeta_j}{j!}\delta^j \le \frac{\xi}{\omega}\chi_r(\delta).$$

This inequality and (4.40) together imply (4.39). □

Clearly, the outcome corresponding to our initial aim to obtain a relative error of at most ω corresponds to the case where `flag` $= 2$. As we will see in what follows, the two other cases are also useful.

4.2.2 Computing the optimality measure

We now describe, in the subsequent Algorithm 9, how to compute the (inexact) optimality measure $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$ at Step 1 of the ARqpDA algorithm.

Algorithm 9 Modified Step 1 of the ARqpDA algorithm, $1 \le q \le p \le 2$.

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

Step 1.0: The iterate x_k and the radius $\delta_{k-1} \in (0,1]$ are given, as well as the constants $\gamma_\varepsilon \in (0,1)$ and $\kappa_\varepsilon > 0$. Set $i_\varepsilon = 0$.

Step 1.1: Choose an initial set of derivative absolute accuracies $\{\varepsilon_{j,0}\}_{j=1}^p$ such that

$$\varepsilon_{j,0} \le \kappa_\varepsilon, \quad \text{for} \quad j \in \{1,\dots,p\}. \tag{4.41}$$

Step 1.2: If unavailable, compute $\{\overline{\nabla^j f}(x_k)\}_{j=1}^q$ satisfying

$$\|\overline{\nabla^j f}(x) - \nabla^j f(x)\| \le \varepsilon_{j,i_\varepsilon}, \quad \text{for} \quad j \in \{1,\dots,q\}.$$

Step 1.3: Solve

$$\mathop{\text{globmax}}_{\substack{x_k+d\in\mathcal{X}\\ \|d\|\le\delta_{k-1}}} \overline{\Delta T}_q^f(x_k,d),$$

to obtain the global maximiser d_k and the corresponding Taylor's increment $\overline{\Delta T}_q^f(x_k,d_k)$. Compute

$$\texttt{flag} = \text{VERIFY}\Big(\delta_{k-1}, \overline{\Delta T}_q^f(x_k,d_k), \{\varepsilon_j\}_{j=1}^q, \omega, \tfrac{1}{2}\omega\epsilon\Big).$$

Step 1.4: Terminate the ARqpDA algorithm with the approximate solution $x_\epsilon = x_k$ if $\texttt{flag} = 1$, or if $\texttt{flag} = 3$, or if $\texttt{flag} = 2$ and (4.20) holds with $\delta = \delta_{k-1}$. Also go to Step 2 of the ARqpDA algorithm if $\texttt{flag} = 2$ but (4.20) fails.

Step 1.5: Otherwise (i.e. if $\texttt{flag} = 0$), set

$$\varepsilon_{j,i_\varepsilon+1} = \gamma_\varepsilon \varepsilon_{j,i_\varepsilon}, \quad \text{for} \quad j \in \{1,\dots,p\}, \tag{4.42}$$

increment i_ε by one and return to Step 1.1.

We immediately observe that Algorithm 9 terminates in a finite number of iterations, since it does so as soon as $\texttt{flag} > 0$ which, because of the first proposition of Lemma 30, must happen after a finite number of passes in iterations using (4.42). We discuss in Section 4.2.4 exactly how many such decreases might be needed.

We now verify that terminating the ARqpDA algorithm as indicated in this modified version of Step 1 provides the required result.

Lemma 31. If x_k is an isolated feasible point, then

$$\phi_{f,q}^{\delta_{k-1}}(x_k) = 0 = \overline{\phi}_{f,q}^{\delta_{k-1}}(x_k),$$

which means that $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$ is a faithful indicator of optimality at x_k.
Moreover, if the ARqpDA algorithm terminates within Step 1.4, then

$$\phi_{f,q}^{\delta_{k-1}}(x_k) \le \epsilon\chi_q(\delta_{k-1}) \tag{4.43}$$

and x_k is a (ϵ, δ_{k-1})-approximate q-th-order necessary minimiser; otherwise, Algorithm 9 terminates with

$$(1-\omega)\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k) \le \phi_{f,q}^{\delta_{k-1}}(x_k) \le (1+\omega)\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k). \tag{4.44}$$

.

Proof. If x_k is an isolated feasible point (i.e. such that the intersection of any ball of radius $\delta_{k-1} > 0$ centered at x_k with $\mathcal{X}$ is reduced to x_k), then clearly $d_k = 0$ and thus, irrespective of ω and $\delta_{k-1} > 0$ (recall definitions (4.10) and (4.19)),

$$\phi_{f,q}^{\delta_{k-1}}(x_k) = 0 = \overline{\Delta T}_q^f(x_k, d_k) = \overline{\phi}_{f,q}^{\delta_{k-1}}(x_k). \tag{4.45}$$

We then notice that Step 1.2 of Algorithm 9 yields (4.35) with $T_r = T_r^f$, $r = q$ and $\{\zeta_j\}_{j=1}^r = \{\varepsilon_{j,i_\varepsilon}\}_{j=1}^q$. Furthermore, $\omega \in (0,1)$ (see, (4.22)), so that the assumptions of Lemma 30 are satisfied. If x_k is an isolated feasible point, the conclusions of the lemma directly follow from (4.45). Assume therefore that x_k is not an isolated feasible point and first note that, because Step 1.3 finds the global maximum of $\overline{\Delta T}_q^f(x_k, d)$, we have that $\overline{\Delta T}_q^f(x_k, d_k) \ge 0$. Suppose now that, in Step 1.3, the VERIFY algorithm returns `flag = 1` and, thus, that $\overline{\Delta T}_q^f(x_k, d_k) = 0$. This means that x_k is a global minimiser of $\overline{T}_q^f(x_k, d)$ in the intersection of a ball of radius δ_{k-1} and $\mathcal{X}$ and it is such that $\overline{\Delta T}_q^f(x_k, d) \le 0$ for any d in this intersection. Thus, for any such d, we obtain from (4.37) with $\xi = \frac{1}{2}\omega\epsilon$ that

$$\Delta T_q^f(x_k, d) \le \overline{\Delta T}_q^f(x_k, d) + \left|\overline{\Delta T}_q^f(x_k, d) - \Delta T_q^f(x_k, d)\right| \le \tfrac{1}{2}\omega\epsilon\chi_q(\delta_{k-1}),$$

which, since $\omega < 1$, implies (4.43). Suppose next that the VERIFY algorithm returns `flag = 3`. Then $\overline{\Delta T}_q^f(x_k, d_k) > 0$ and thus $d_k \ne 0$. Using the fact that the nature of Step 1.3 ensures that $\overline{\Delta T}_q^f(x_k, d) \le \overline{\Delta T}_q^f(x_k, d_k)$ for d with $\|d\| \le \delta_{k-1}$ we have, using (4.39) with $\xi = \frac{1}{2}\omega\epsilon$, that, for all such d,

$$\begin{aligned}
\Delta T_q^f(x_k, d) &\le \overline{\Delta T}_q^f(x_k, d) + \left|\overline{\Delta T}_q^f(x_k, d) - \Delta T_q^f(x_k, d)\right| \\
&\le \overline{\Delta T}_q^f(x_k, d_k) + \left|\overline{\Delta T}_q^f(x_k, d) - \Delta T_q^f(x_k, d)\right| \\
&\le \epsilon\chi_q(\delta_{k-1}),
\end{aligned}$$

yielding (4.43). If the VERIFY algorithm returns `flag = 2`, then, for any d with $\|d\| \le \delta_{k-1}$,

$$\Delta T_q^f(x_k, d) \le \overline{\Delta T}_q^f(x_k, d) + \left|\overline{\Delta T}_q^f(x_k, d) - \Delta T_q^f(x_k, d)\right| \le (1+\omega)\overline{\Delta T}_q^f(x_k, d_k).$$

Thus, for all d with $\|d\| \le \delta_{k-1}$,

$$\max\left[0, \Delta T_q^f(x_k, d)\right] \le (1+\omega)\max\left[0, \overline{\Delta T}_q^f(x_k, d_k)\right] = (1+\omega)\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k). \tag{4.46}$$

But termination implies that (4.20) holds for $\delta = \delta_{k-1}$ and (4.43) follows with this value of δ. Finally, if the ARqpDA algorithm does not terminates within Step 1.4 but Algorithm 9 for the modifies Step 1 terminates, it must be because the VERIFY algorithm returns `flag` = 2. This implies, as above, that (4.46) holds, which is the rightmost part of (4.44). Similarly, for any d with $\|d\| \leq \delta_{k-1}$,

$$\begin{aligned} \Delta T_q^f(x_k,d) &\geq \overline{\Delta T}_q^f(x_k,d) - \left|\overline{\Delta T}_q^f(x_k,d) - \Delta T_q^f(x_k,d)\right| \\ &\geq \overline{\Delta T}_q^f(x_k,d) - \omega\overline{\Delta T}_q^f(x_k,d_k). \end{aligned}$$

Hence,
$$\begin{aligned} \operatorname*{globmax}_{\substack{x_k+d\in\mathcal{X}\\ \|d\|\leq\delta_{k-1}}} \Delta T_q^f(x_k,d) &\geq \operatorname*{globmax}_{\substack{x_k+d\in\mathcal{X}\\ \|d\|\leq\delta_{k-1}}} \left[\overline{\Delta T}_q^f(x_k,d) - \omega\overline{\Delta T}_q^f(x_k,d_k)\right] \\ &= (1-\omega)\overline{\Delta T}_q^f(x_k,d_k). \end{aligned}$$

Since $\overline{\Delta T}_q^f(x_k,d_k) > 0$ when the VERIFY algorithm returns `flag` = 2, we then obtain that, for all $\|d\| \leq \delta_{k-1}$,

$$\max\left[0, \operatorname*{globmax}_{\substack{x_k+d\in\mathcal{X}\\ \|d\|\leq\delta_{k-1}}} \Delta T_q^f(x_k,d)\right] \geq \max\left[0,(1-\omega)\overline{\Delta T}_q^f(x_k,d_k)\right] = (1-\omega)\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k),$$

which is the leftmost part of (4.44). □

4.2.3 Computing the step

We now consider how to compute s_k at Step 2 of the ARqpDA algorithm. The process is more complicated than for Step 1, as it potentially involves two situations in which one wishes to guarantee a suitable relative error. The first is when minimising the model

$$m_k(s) = \overline{f}(x_k) - \overline{\Delta T}_p^f(x_k,s) + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1}$$

or, equivalently, maximising

$$-m_k(s) = -\overline{f}(x_k) + \overline{\Delta T}_p^f(x_k,s) - \frac{\sigma_k}{(p+1)!}\|s\|^{p+1}, \tag{4.47}$$

and the second is when globally minimising the model Taylor's expansion taken at $x_k + s_k$ in a neighbourhood of diameter δ_k.

The first of these situations can be handled in a way very similar to that used above for computing $\overline{\phi}_{f,q}^{\delta_{k-1}}(x_k)$ at Step 1: given a set of approximate derivatives, a step s_k is computed such that it satisfies (4.23)–(4.24), the relative error of the associated $\overline{\Delta T}_p^f(x_k,s_k)$ is then evaluated; if it is insufficient, the accuracy on the derivative approximations improved and the process restarted. If the relative error on $\overline{\Delta T}_p^f(x_k,s_k)$ is satisfactory and the first test of (4.24) fails, it remains to check that the relative error on $\overline{\phi}_{m_k,q}^{\delta_k}(s_k)$ is also satisfactory. Moreover, as in the original ARqpDA algorithm, we have to take into account the possibility that minimising the model might result in a vanishing decrease.

The resulting somewhat involved process is formalised in Algorithm 10 on the next page.

Algorithm 10 Modified Step 2 of the ARqpDA algorithm.

Step 2: Step calculation.

Step 2.0: The iterate x_k, the radius $\delta_{k-1} \in (0,1]$, the constants $\gamma_\varepsilon \in (0,1)$, $\vartheta \in (0,1)$, the counter i_ε and the absolute accuracies $\{\varepsilon_{j,i_\varepsilon}\}_{j=1}^p$ are given.

Step 2.1: If unavailable, compute $\{\overline{\nabla^j f}(x_k)\}_{j=1}^p$ satisfying (4.29) with $\varepsilon_j = \varepsilon_{j,i_\varepsilon}$ for $j \in \{1,\dots,p\}$.

Step 2.2: Compute a step $s_k \neq 0$ with $x_k + s_k \in \mathcal{X}$ such that (4.23) holds.

- Otherwise, pursue the approximate minimisation of the model $m_k(s)$ for $x_k + s_k \in \mathcal{X}$ in order to satisfy (4.24), yielding a step s_k, a decrease $\overline{\Delta T}_p^f(x_k,s_k)$ and, if the first part of (4.24) fails, the global maximiser $d_k^{m_k}$ of $\overline{\Delta T}_q^{m_k}(s_k,d)$ subject to $\|d\| \le \delta_k$ and $x_k + s_k + d \in \mathcal{X}$, together with the corresponding Taylor's increment $\overline{\Delta T}_q^{m_k}(s_k,d_k^{m_k})$.
- Compute

$$\texttt{flag}_s = \text{VERIFY}\Big(\|s_k\|, \overline{\Delta T}_p^f(x_k,s_k), \{\varepsilon_j\}_{j=1}^p, \tfrac{1}{2}\omega\epsilon\Big).$$

If $\texttt{flag}_s = 0$, go to Step 2.5.

Step 2.3: If $\texttt{flag}_s = 1$ or $\texttt{flag}_s = 3$, compute

$$\operatorname*{globmin}_{x_k+s\in\mathcal{X}} m_k(s),$$

to obtain the minimiser s_k, $\overline{\Delta T}_p^f(x_k,s_k)$.
Set $d_k^{m_k} = 0 = \overline{\Delta T}_q^{m_k}(s_k,d_k^{m_k})$ and compute

$$\texttt{flag}_s = \text{VERIFY}\Big(\|s_k\|, \overline{\Delta T}_p^f(x_k,s_k), \{\varepsilon_j\}_{j=1}^p, \omega, \tfrac{1}{2}\omega\epsilon\Big).$$

If $\texttt{flag}_s = 0$, go to Step 2.5.

Step 2.4: If $\texttt{flag}_s = 1$ or $\texttt{flag}_s = 3$, terminate the ARqpDA algorithm with $x_\epsilon = x_k$. Otherwise, if $\|s_k\| \ge \mu\epsilon^{\frac{1}{p-q+1}}$ or if $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}$ and

$$\texttt{flag}_d = \text{VERIFY}\Big(\delta_k, \overline{\Delta T}_q^{m_k}(s_k,d_k^{m_k}), \Big\{\frac{5}{2}\varepsilon_j\Big\}_{j=1}^q, \omega, \frac{\vartheta(1-\omega)}{(1+\omega)^2}\frac{\omega\epsilon}{2}\Big) > 0,$$

go to Step 3 of the ARqpDA algorithm with the step s_k, the associated $\overline{\Delta T}_p^f(x_k,s_k)$ and δ_k.

Step 2.5: Set (if $\texttt{flag}_s = 0$ or $\texttt{flag}_d = 0$),

$$\varepsilon_{j,i_\epsilon+1} = \gamma_\varepsilon \varepsilon_{j,i_\epsilon}, \quad \text{for} \quad j \in \{1,\dots,p\}, \tag{4.48}$$

increment i_ε by one and go to Step 2.1.

Observe that, in Step 2.2, $d_k^{m_k}$ and $\overline{\Delta T}_q^{m_k}(s_k,d_k^{m_k})$ result from the computation of $\overline{\phi}_{m_k,q}^{\delta_k}(s_k)$ which is necessary to verify the second part of (4.24). Note also that we have specified, in the call to VERIFY in Step 2.4 of Algorithm 10, absolute accuracy values equal to $\left\{\frac{5}{2}\varepsilon_j\right\}_{j=1}^q$. This is because this call aims at checking the accuracy of the Taylor's expansion of the model and the derivatives which are then approximated are not $\{\nabla_x^j f(x_k)\}_{j=1}^q$, but $\{\nabla_d^j T_q^{m_k}(s_k,0)\}_{j=1}^q$. It is easier to derive such (approximate) derivatives for $q=1$ and $q=2$ and to verify that they can be written in a more compact way as

$$\overline{\nabla_d^j T}_q^{m_k}(s_k,0) = \sum_{\ell=j}^p \frac{\overline{\nabla_x^\ell f}(x_k)\|s_k\|^{\ell-j}}{(\ell-j)!} + \left[\nabla_s^j \|s\|^{p+1}\right]_{s=s_k}, \quad 1 \le q \le p \le 2, \tag{4.49}$$

where the last term at the right-hand side is exact. This yields the following error bound.

Lemma 32. Suppose that $\|s_k\| \leq \mu\epsilon^{\frac{1}{p-q+1}}$. Then, for all $j \in \{1, \dots, p\}$, $1 \leq q \leq p \leq 2$,

$$\left|\overline{\nabla_d^j T}_q^{m_k}(s_k, 0) - \nabla_d^j T_q^{m_k}(s_k, 0)\right| \leq \frac{5}{2}\varepsilon_j. \tag{4.50}$$

Proof. Using the triangle inequality, (4.49), the inequality $\|s_k\| \leq \mu\epsilon^{\frac{1}{p-q+1}} \leq \mu$ and (4.14), we have that, for all $j \in \{1, \dots, p\}$,

$$\left|\overline{\nabla_d^j T}_q^{m_k}(s_k, 0) - \nabla_d^j T_q^{m_k}(s_k, 0)\right| \leq \sum_{\ell=j}^{p} \frac{\varepsilon_j \|s_k\|^{\ell-j}}{(\ell-j)!} \leq \varepsilon_j \sum_{\ell=j}^{p} \frac{\mu^{\ell-j}}{(\ell-j)!} \leq \varepsilon_j(1 + \chi_p(\mu)).$$

Inequality (4.50) then follows due to $\chi_p(\mu) \leq \frac{3}{2}\mu$ (recall (4.16)). $\square$

Again, Algorithm 10 must terminate in a finite number of iterations. Indeed, if after finitely many iterations $\texttt{flag}_s = 1$ or $\texttt{flag}_s = 3$ at the start of Step 2.4, the conclusion is obvious. Suppose now that $\texttt{flag}_s = 2$ at all iterations. If $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}$ always hold, the first proposition of Lemma 30 ensures that $\texttt{flag}_d > 0$ after finitely many decreases in (4.48), also causing termination. Termination might of course occur if $\|s_k\| \geq \mu\epsilon^{\frac{1}{p-q+1}}$ before this limit.

The next lemma characterises the outcomes of Algorithm 10.

Lemma 33. Suppose that the modified Step 2 is used in the ARqpDA algorithm. If this algorithm terminates within that step, then either x_k is an (ϵ, δ)-approximate q-th-order necessary minimiser for some $\delta > 0$, or

$$\phi_{f,p}^{\|s_k\|}(x_k) \leq \epsilon\chi_p(\|s_k\|). \tag{4.51}$$

Otherwise we have that (4.23) and

$$\left|\overline{\Delta T}_p^f(x_k, s_k) - \Delta T_p^f(x_k, s_k)\right| \leq \omega\overline{\Delta T}_p^f(x_k, s_k) \tag{4.52}$$

are satisfied. Moreover, either $\|s_k\| \geq \mu\epsilon^{\frac{1}{p-q+1}}$ or

$$\phi_{m_k,q}^{\delta_k}(s_k) \quad\leq\quad (1+\omega)\max\left[\frac{\vartheta(1-\omega)}{(1+\omega)^2}\,\epsilon,\ \frac{\theta\|s_k\|^{p-q+1}}{(p-q+1)!}\right]\chi_q(\delta_k) \tag{4.53}$$

hold.

Proof. We first note that, because of (4.47) and since Step 2.2 imposes (4.23), we have that $\overline{\Delta T}_p^f(x_k, s_k) \geq 0$ at the end of this step. Let us first consider the case where the calls to the VERIFY algorithm in Step 2.2 and in Step 2.3 both return $\texttt{flag}_s = 1$ or $\texttt{flag}_s = 3$ and note that Step 2.1 yields (4.35) with $T_r = T_r^f$, $r = p$ and $\{\zeta_j\}_{j=1}^r = \{\varepsilon_{j,i_\varepsilon}\}_{j=1}^p$. Moreover, (4.22) ensures that $\omega \in (0,1)$, so that we can use Lemma 30 to analyse the outcome of the above calls to the VERIFY Algorithm. Since $s_k \neq 0$, we have that $\overline{\Delta T}_p^f(x_k, s_k) > 0$ because of (4.23). Moreover, using the fact that s_k computed in Step 2.3 is a global minimiser of m_k, we obtain that

$$\overline{\Delta T}_p^f(x_k, s) - \frac{\sigma_k}{(p+1)!}\|s\|^{p+1} \leq \overline{\Delta T}_p^f(x_k, s_k) - \frac{\sigma_k}{(p+1)!}\|s_k\|^{p+1}, \tag{4.54}$$

for all s. Thus, if $\|s\| \leq \|s_k\|$, then $\overline{\Delta T}_p^f(x_k, s) \leq \overline{\Delta T}_p^f(x_k, s_k)$. This implies that

$$\operatorname*{globmax}_{\substack{x_k+s\in\mathcal{X}\\ \|s\|\leq\|s_k\|}} \overline{\Delta T}_p^f(x_k, s) = \overline{\Delta T}_p^f(x_k, s_k).$$

We may now repeat the proof of Lemma 31 for the cases $\texttt{flag}_s \in \{1, 3\}$, with q replaced by p and δ_{k-1} replaced by $\|s_k\|$, and deduce that (4.51) holds.
Assume now that Algorithm 10 terminates at Step 2.4. This means that the VERIFY algorithm invoked in either Step 2.2 or Step 2.3 terminates with $\texttt{flag}_s = 2$ and we deduce from (4.38) that(4.52) holds.
Let us now consider the case $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}$ and note that Lemma 32 ensures that (4.35) is satisfied with $T_r = T_r^{m_k}$, $r = q$ and $\{\zeta_j\}_{j=1}^r = \left\{\frac{5}{2}\varepsilon_{j,i_\varepsilon}\right\}_{j=1}^q$. Moreover, the triangle inequality gives

$$\Delta T_q^{m_k}(s_k, d) \leq \overline{\Delta T}_q^{m_k}(s_k, d) + \left|\overline{\Delta T}_q^{m_k}(s_k, d) - \Delta T_q^{m_k}(s_k, d)\right|. \tag{4.55}$$

First, assume that Algorithm 10 terminates within Step 2.4 because $\texttt{flag}_d = 1$ is returned by VERIFY. Then, $\overline{\Delta T}_q^{m_k}(s_k, d_k^{m_k}) = 0$. Moreover, using (4.55), the definition of $d_k^{m_k}$ given at Step 2.2 of Algorithm 10, (4.37) and recalling that $\omega \leq 1$, we obtain that, for all d with $\|d\| \leq \delta_k$,

$$\begin{aligned}
\Delta T_q^{m_k}(s_k, d) &\leq \overline{\Delta T}_q^{m_k}(s_k, d) + \left|\overline{\Delta T}_q^{m_k}(s_k, d) - \Delta T_q^{m_k}(s_k, d)\right| \\
&\leq \overline{\Delta T}_q^{m_k}(s_k, d_k^{m_k}) + \left|\overline{\Delta T}_q^{m_k}(s_k, d) - \Delta T_q^{m_k}(s_k, d)\right| \\
&= \left|\overline{\Delta T}_q^{m_k}(s_k, d) - \Delta T_q^{m_k}(s_k, d)\right| \\
&\leq \frac{\vartheta(1-\omega)}{2(1+\omega)^2}\,\omega\,\epsilon\,\chi_q(\|d\|) \\
&\leq \frac{\vartheta(1-\omega)}{(1+\omega)^2}\,\epsilon\,\chi_q(\delta_k).
\end{aligned} \tag{4.56}$$

If, instead, termination occurs with VERIFY returning $\texttt{flag}_d = 2$, then we will show that for all d with $\|d\| \leq \delta_k$,

$$\Delta T_q^{m_k}(s_k, d) \leq (1+\omega)\overline{\Delta T}_q^{m_k}(s_k, d_k^{m_k}) \leq (1+\omega)\frac{\theta\|s_k\|^{p-q+1}}{(p-q+1)!}\,\chi_q(\delta_k). \tag{4.57}$$

Indeed, from (4.55), (4.38), (4.22) and the definition of $d_k^{m_k}$ at Step 2.2 of Algorithm 10, we obtain, for all d with $\|d\| \leq \delta_k$,

$$\begin{aligned}
\Delta T_q^{m_k}(s_k, d) &\leq (1+\omega)\overline{\Delta T}_q^{m_k}(s_k, d_k^{m_k}), \\
&\leq (1+\omega)\max\left[0, \operatorname*{globmax}_{\substack{x_k+s_k+d\in\mathcal{X}\\ \|d\|\leq\delta_k}} \overline{\Delta T}_q^{m_k}(s_k, d)\right] \\
&= (1+\omega)\overline{\phi}_{m_k,q}^{\delta_k}(s_k),
\end{aligned}$$

in which the equality follows from the definition (4.19). We can then conclude, using (4.24), that (4.57) holds for all d with $\|d\| \leq \delta_k$.

Finally, if termination occurs instead because VERIFY returns $\texttt{flag}_d = 3$, we deduce from the (4.55) and (4.39) that, for all d with $\|d\| \leq \delta_k$,

$$\Delta T_q^{m_k}(s_k, d) \leq \frac{\vartheta(1-\omega)}{(1+\omega)^2}\,\epsilon\,\chi_q(\delta_k). \tag{4.58}$$

Observe now that (4.22), (4.10) (for m_k at s_k) and each of (4.56), (4.57) or (4.58) ensures (4.53). $\square$

Note that (4.51) may be viewed as a stronger optimality condition than (4.13), since it implies that the p-th (rather than q-th with $q \le p$) order Taylor's expansion of f around x_k is bounded below by a correctly scaled multiple of ϵ and in a possibly larger neighborhood. It is thus acceptable to terminate the ARqpDA algorithm with $x_\epsilon = x_k$, as stated at Step 2.4 of Algorithm 10.

4.2.4 Evaluation complexity of a single ARqpDA iteration

The last part of this section is devoted to bound the evaluation complexity of a single iteration of the ARqpDA algorithm.

The count in (approximate) objective function evaluations is the following: these only occur in Step 3 which requires at most two such evaluations. Now observe that evaluations of $\{\overline{\nabla^j f}\}_{j=1}^p$, $p \in \{1,2\}$, possibly occur in Step 1.2 and Step 2.1. However, it is important to note that, within these steps, the derivatives are evaluated only if the current values of the absolute errors are smaller than that used for the previous evaluations of the same derivative at the same point (x_k). Moreover, these absolute errors are, by construction, linearly decreasing with rate γ_ε within the same iteration of the ARqpDA algorithm. In fact, they are initialised at Step 1.1, decreased each time by a factor γ_ε in (4.42) invoked in Step 1.5, down to values $\{\varepsilon_{j,i_\varepsilon}\}_{j=1}^p$ which are then passed to the modified Step 2 and decreased there further in (4.48) at Step 2.5, again by successive multiplication with γ_ε. Furthermore, we have already argued, both for the modified Step 1 and the modified Step 2, that any of these algorithms terminates as soon as (4.36) holds for the relevant value of ξ, which we therefore need to determine. For Step 1, this value is $\frac{1}{2}\omega\epsilon$, while, for Step 2, it is given by

$$\min\left[\tfrac{1}{2}\omega\epsilon, \frac{\vartheta(1-\omega)}{(1+\omega)^2}\frac{\omega\epsilon}{2}\right] = \frac{\vartheta(1-\omega)}{2(1+\omega)^2}\,\omega\epsilon, \tag{4.59}$$

when $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}$ and by $\frac{1}{2}\omega\epsilon$, when $\|s_k\| \ge \mu\epsilon^{\frac{1}{p-q+1}}$.

The following lemma can thus be proved.

Lemma 34. Recalling the definition of ω in (4.22), each iteration of the ARqpDA algorithm involves at most two (approximate) evaluations of the objective function and at most $1 + \nu_{\max}(\epsilon)$ (approximate) evaluations of its p first derivatives, where

$$\nu_{\max}(\epsilon) = \left\lfloor \frac{1}{\log(\gamma_\varepsilon)}\left\{\log\left(\frac{\vartheta(1-\omega)}{5(1+\omega)^2}\omega\epsilon\right) - \log(\kappa_\varepsilon)\right\}\right\rfloor. \tag{4.60}$$

Proof. The upper bound on the (approximate) function evaluations immediately follows from the observation that, as mentioned at the beginning of the current subsection, these computations occur at most twice at Step 3 of the ARqpDA algorithm. Concerning the second part of the thesis we notice that, from Lemma 32, in Step 2.4 of Algorithm 10 we have to make $\{\overline{\nabla^j f}(x_k)\}_{j=1}^p$ 5/2-times more accurate than the desired accuracy in $\{\overline{\nabla_d^j T}_q^{m_k}(s_k,0)\}_{j=1}^q$, when $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}$ (recall that the input values for the absolute accuracy values in the VERIFY call are $\left\{\frac{5}{2}\varepsilon_j\right\}_{j=1}^q$). Thus, the VERIFY Algorithm stops whenever

$$\max_{j\in\{1,\dots,q\}} \varepsilon_j \le \frac{\vartheta(1-\omega)\omega\epsilon}{5(1+\omega)^2}.$$

We may thus conclude from Lemma 30 that no further reduction in $\{\varepsilon_j\}_{j=1}^p$ (and hence no further approximation of $\{\overline{\nabla^j f}(x_k)\}_{j=1}^p$) will occur once i_ε, the number of decreases in $\{\varepsilon_j\}_{j=1}^p$, is large enough, to ensure that

$$\gamma_\varepsilon^{i_\epsilon}[\max_{j\in\{1,\dots,p\}} \varepsilon_{j,0}] \leq \frac{\vartheta(1-\omega)}{5(1+\omega)^2}\omega\epsilon$$

(note that this inequality could hold for $i_\epsilon = 0$). Since $\omega \in (0,1)$ and because of (4.41), the above inequality is then verified when

$$i_\epsilon \leq \left\lfloor \frac{1}{\log(\gamma_\varepsilon)} \left\{ \log\left(\frac{\vartheta(1-\omega)}{5(1+\omega)^2}\omega\epsilon \right) - \log(\kappa_\varepsilon) \right\} \right\rfloor,$$

which concludes the proof when taking into account that the derivatives must be computed at least once per iteration. □

We conclude this subsection by a comment on what happens whenever exact objective function and derivatives values are used. In that case the (exact) derivatives are computed only once per iteration of the ARqpDA algorithm (in Step 1.2 for the first q and in Step 2.1 for the remaining $(p-q)$) and every other call to VERIFY returns $\texttt{flag} = 1$ or $\texttt{flag} = 2$. Moreover, there is no need to recompute $\overline{f}_k(x_k)$ to obtain (4.26) in Step 3. The evaluation complexity of a single iteration of the ARqpDA algorithm then reduces to a single evaluation of f and its first p derivatives (and $\nu_{\max}(\epsilon) = 1$ for all k), as expected.

4.3 Evaluation complexity of the deterministic ARqpDA

This section is devoted to the evaluation complexity analysis of the ARqpDA algorithm in the deterministic context.

We start by extending the analogous result of Lemma 3 (see, also, [35]) to prove that the decrease of a model employing inexact function and derivatives evaluations remains bounded below and greater than zero, ensuring the well-definitiveness of the ratio ρ_k in (4.27).

Lemma 35. The mechanism of the ARqpDA algorithm guarantees that, for all $k \geq 0$,

$$\overline{\Delta T}_p^f(x_k, s_k) > \frac{\sigma_k}{(p+1)!}\|s_k\|^{p+1}, \tag{4.61}$$

and so (4.27) is well-defined.

Proof. We have that, by the model definition (4.2),

$$0 < m_k(0) - m_k(s_k) = \overline{T}_p(x_k, 0) - \overline{T}_p(x_k, s_k) - \frac{\sigma_k}{(p+1)!}\|s_k\|^{p+1}.$$

□

As done in Lemma 21 for the case of an ARC algorithm with inexact Hessian evaluations (the ARC-DH algorithm), we next show that the regularisation parameter σ_k remains bounded, even in the presence of inexact computation of f and its derivatives. This lemma heavily hinges on (4.18), (4.25) and (4.26).

Lemma 36. Let Assumption 4.1.1 hold. Then, for all $k \geq 0$,

$$\sigma_k \leq \sigma_{\max} \overset{\text{def}}{=} \max\left[\sigma_0, \frac{\gamma_3(L+3)}{1-\eta_2-3\omega}\right], \tag{4.62}$$

with ω defined as in (4.22).

Proof. Assume that

$$\sigma_k \geq \frac{L+3}{1-\eta_2}. \tag{4.63}$$

Also observe that, because of the triangle inequality, (4.52) (as ensured by Lemma 33) and (4.26),

$$\begin{aligned} |\overline{T}_p^f(x_k,s_k) - T_p^f(x_k,s_k)| &\leq |\overline{f}_k(x_k) - f(x_k)| + |\overline{\Delta T}_p^f(x_k,s_k) - \Delta T_p^f(x_k,s_k)| \\ &\leq 2\omega|\overline{\Delta T}_p^f(x_k,s_k)| \end{aligned}$$

and hence, again using the triangle inequality, (4.25), (4.8), (4.61) and (4.63),

$$\begin{aligned} |\rho_k - 1| &\leq \frac{|\overline{f}_k(x_k+s_k) - \overline{T}_p^f(x_k,s_k)|}{\overline{\Delta T}_p^f(x_k,s_k)} \\ &\leq \frac{1}{\overline{\Delta T}_p^f(x_k,s_k)}\Big[|\overline{f}_k(x_k+s_k) - f(x_k+s_k)| + |f(x_k+s_k) - T_p^f(x_k,s_k)| \\ &\qquad + |\overline{T}_p^f(x_k,s_k) - T_p^f(x_k,s_k)|\Big] \\ &\leq \frac{1}{\overline{\Delta T}_p^f(x_k,s_k)}\Big[|f(x_k+s_k) - T_p^f(x_k,s_k)| + 3\omega\overline{\Delta T}_p^f(x_k,s_k)\Big] \\ &\leq \frac{1}{\overline{\Delta T}_p^f(x_k,s_k)}\Big[\frac{L}{(p+1)!}\|s_k\|^{p+1} + 3\omega\overline{\Delta T}_p^f(x_k,s_k)\Big] \\ &< \frac{L}{\sigma_k} + 3\omega \\ &\leq 1-\eta_2 \end{aligned} \tag{4.64}$$

and, thus, that $\rho_k \geq \eta_2$. Then, iteration k is very successful in that $\rho_k \geq \eta_2$ and, because of (6.8), $\sigma_{k+1} \leq \sigma_k$. As a consequence, the mechanism of the algorithm ensures that (4.62) holds. □

We now borrow a technical result from [35].

Lemma 37. [35, Lemma 2.4] Let s be a vector of $\mathbb{R}^n$ and $p \in \{1,2\}$ such that $j \in \{0,\dots,p\}$. Then,

$$\|\nabla_s^j(\|s\|^{p+1})\|_{[j]} \leq \frac{(p+1)!}{(p-j+1)!}\|s\|^{p-j+1}. \tag{4.65}$$

Our next move is to prove a lower bound on the norm of the step. While the proof of this result is clearly inspired from that of [35, Lemma 3.3], it nevertheless crucially differs when approximate values are considered instead of exact ones.

Lemma 38. Let Assumption 4.1.1 hold. Then, for all $k \geq 0$ such that the ARqpDA algorithm does not terminate at iteration $k+1$,

$$\|s_k\| \geq \kappa_s \epsilon^{\frac{1}{p-q+1}}, \tag{4.66}$$

where

$$\kappa_s \stackrel{\rm def}{=} \min\left\{\mu, \left[\frac{(1-\omega)(1-\vartheta)(p-q+1)!}{(1+\omega)(L+\sigma_{\max}+\theta(1+\omega))}\right]^{\frac{1}{p-q+1}}\right\}. \tag{4.67}$$

Proof. If $\|s_k\| \geq \mu\epsilon^{\frac{1}{p-q+1}}$, the result is obvious. Suppose now that

$$\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}}. \tag{4.68}$$

Since the algorithm does not terminate at iteration $k+1$, we have that

$$\overline{\phi}_{f,q}^{\delta_k}(x_{k+1}) > \frac{\epsilon}{1+\omega}\,\chi_q(\delta_k)$$

and therefore, using (4.44), that

$$\phi_{f,q}^{\delta_k}(x_{k+1}) > \frac{1-\omega}{1+\omega}\,\epsilon\,\chi_q(\delta_k). \tag{4.69}$$

Let the global minimum in the definition of $\phi_{f,q}^{\delta_k}(x_{k+1})$ be achieved at d, with $\|d\| \leq \delta_k$. Then, using (4.10), the triangle inequality and (4.65), we deduce that

$$\begin{aligned}
\phi_{f,q}^{\delta_k}(x_{k+1}) &= -\sum_{\ell=1}^{q}\frac{1}{\ell!}\nabla_x^\ell f(x_{k+1})[d]^\ell \\
&\leq \left|\sum_{\ell=1}^{q}\frac{1}{\ell!}\nabla_x^\ell f(x_{k+1})[d]^\ell - \sum_{\ell=1}^{q}\frac{1}{\ell!}\nabla_s^\ell T_p^f(x_k,s_k)[d]^\ell\right| - \sum_{\ell=1}^{q}\frac{1}{\ell!}\nabla_s^\ell T_p^f(x_k,s_k)[d]^\ell \\
&\leq \sum_{\ell=1}^{q}\frac{1}{\ell!}\Big[\|\nabla_x^\ell f(x_{k+1}) - \nabla_s^\ell T_p^f(x_k,s_k)\|_{[\ell]}\Big]\delta_k^\ell \\
&\quad -\sum_{\ell=1}^{q}\frac{1}{\ell!}\left(\nabla_s^\ell\left[T_p^f(x_k,s)+\frac{\sigma_k}{(p+1)!}\|s\|^{p+1}\right]_{s=s_k}\right)[d]^\ell \\
&\quad +\sum_{\ell=1}^{q}\frac{\sigma_k}{\ell!(p-\ell+1)!}\|s_k\|^{p-\ell+1}\delta_k^\ell.
\end{aligned} \tag{4.70}$$

Now, because of (4.1), (4.10) (for m_k at s_k) and the fact that $\|d\| \leq \delta_k$, we have that

$$-\sum_{\ell=1}^{q}\frac{1}{\ell!}\left(\nabla_s^\ell\left[T_p^f(x_k,s)+\frac{\sigma_k}{(p+1)!}\|s\|^{p+1}\right]_{s=s_k}\right)[d]^\ell = \Delta T_q^{m_k}(s_k,d) \leq \phi_{m_k,q}^{\delta_k}(s_k).$$

Then, as $\|s_k\| < \mu\epsilon^{\frac{1}{p-q+1}} < 1$ because of (4.68), we may use (4.53) (ensured by Lemma 33) and (4.9) and distinguish the cases where the maximum in (4.53) is attained in its first or its second argument. In the latter case, we deduce from (4.70) that

$$\begin{aligned}
\phi_{f,q}^{\delta_k}(x_{k+1}) &\leq \sum_{\ell=1}^{q}\frac{L}{\ell!(p-\ell+1)!}\|s_k\|^{p-\ell+1}\delta_k^\ell + (1+\omega)\frac{\theta\,\chi_q(\delta_k)}{(p-q+1)!}\|s_k\|^{p-q+1} \\
&\quad +\sum_{\ell=1}^{q}\frac{\sigma_k}{\ell!(p-\ell+1)!}\|s_k\|^{p-\ell+1}\delta_k^\ell \\
&\leq \frac{\Big[L+\sigma_k+\theta(1+\omega)\Big]\chi_q(\delta_k)}{(p-q+1)!}\|s_k\|^{p-q+1};
\end{aligned} \tag{4.71}$$

otherwise, (4.70) guarantees that

$$\phi_{f,q}^{\delta_k}(x_{k+1}) \quad \leq \quad \frac{(L+\sigma_k)\chi_q(\delta_k)}{(p-q+1)!}\|s_k\|^{p-q+1} + \frac{\vartheta(1-\omega)}{1+\omega}\,\epsilon\chi_q(\delta_k). \tag{4.72}$$

Using now (4.69), (4.22), (4.68), (4.71) and (4.72), we thus have that

$$\begin{aligned}\|s_k\| &\geq \min\left\{\mu\epsilon^{\frac{1}{p-q+1}}, \left[\frac{\epsilon(1-\omega)(p-q+1)!}{(1+\omega)(L+\sigma_k+\theta(1+\omega))}\right]^{\frac{1}{p-q+1}}, \left[\frac{\epsilon(1-\omega)(1-\vartheta)(p-q+1)!}{(1+\omega)(L+\sigma_k)}\right]^{\frac{1}{p-q+1}}\right\}\\ &\geq \min\left\{\mu\epsilon^{\frac{1}{p-q+1}}, \left[\frac{\epsilon(1-\omega)(1-\vartheta)(p-q+1)!}{(1+\omega)(L+\sigma_k+\theta(1+\omega))}\right]^{\frac{1}{p-q+1}}\right\},\end{aligned}$$

where we have used the fact that $\theta \in (0,1)$ to obtain the last inequality. Then, (4.66) follows from (4.62). □

We now combine all the above results to deduce an upper bound on the maximum number of successful iterations, from which a final evaluation (and iteration) complexity bound immediately follows.

Theorem 39. Let Assumption 4.1.1 hold and $\epsilon \in (0,1)$ be given. Then, the ARqpDA algorithm using the modified Steps 1 (on page 75) and 2 (on page 78) produces an iterate x_ϵ such that (4.13) or (4.51) holds in at most

$$\left\lfloor \kappa_p(f(x_0)-f_{\rm low})\left(\epsilon^{-\frac{p+1}{p-q+1}}\right)\right\rfloor + 1 \tag{4.73}$$

successful iterations,

$$\tau(\epsilon) \stackrel{\rm def}{=} \left\lfloor\left\{\left\lfloor\kappa_p(f(x_0)-f_{\rm low})\left(\epsilon^{-\frac{p+1}{p-q+1}}\right)+1\right\rfloor\left(1+\frac{|\log\gamma_1|}{\log\gamma_2}\right)+\frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right)\right\}\right\rfloor \tag{4.74}$$

iterations in total, $2\tau(\epsilon)$ (approximate) evaluations of f and $(1+\nu_{\max}(\epsilon))\tau(\epsilon)$ approximate evaluations of $\{\nabla_x^j f\}_{j=1}^p$, where $\sigma_{\max}$ is given by (4.62), ω by (4.22), $\nu_{\max}(\epsilon)$ by (4.60) and where

$$\kappa_p \stackrel{\rm def}{=} \frac{(p+1)!}{(\eta_1-2\omega)\sigma_{\min}}\max\left\{\frac{1}{\mu^{p+1}}, \left[\frac{(1+\omega)(L+\sigma_{\max}+\theta(1+\omega))}{(1-\omega)(1-\vartheta)(p-q+1)!}\right]^{\frac{p+1}{p-q+1}}\right\}. \tag{4.75}$$

Proof. At each successful iteration k before termination the algorithm guarantees the decrease

$$\begin{aligned} f(x_k)-f(x_{k+1}) &\geq [\overline{f}_k(x_k)-\overline{f}_k(x_{k+1})] - 2\omega\overline{\Delta T}_p^f(x_k,s_k)\\ &\geq \eta_1\overline{\Delta T}_p^f(x_k,s_k) - 2\omega\overline{\Delta T}_p^f(x_k,s_k) \qquad (4.76)\\ &> \frac{(\eta_1-2\omega)\sigma_{\min}}{(p+1)!}\|s_k\|^{p+1},\end{aligned}$$

where we used (4.22), (4.25), (4.26), (4.27), (4.61) and (6.8). Moreover, we deduce from (4.76) and (4.66) that

$$f(x_k)-f(x_{k+1}) \geq \kappa_p^{-1}\epsilon^{\frac{p+1}{p-q+1}}, \quad \text{where} \quad \kappa_p^{-1} \stackrel{\rm def}{=} \frac{(\eta_1-2\omega)\sigma_{\min}\kappa_s^{p+1}}{(p+1)!}, \tag{4.77}$$

with κ_s as in (4.67). Thus, since $\{f(x_k)\}_{k\geq 0}$ decreases monotonically,

$$f(x_0) - f(x_{k+1}) \geq \kappa_p^{-1} \, \epsilon^{\frac{p+1}{p-q+1}} \, |\mathcal{S}_k|.$$

Using that f is bounded below by $f_{\rm low}$, we conclude that

$$|\mathcal{S}_k| \leq \kappa_p (f(x_0) - f_{\rm low}) \epsilon^{-\frac{p+1}{p-q+1}} \tag{4.78}$$

until termination, and the desired bound on the number of successful iterations follows. Lemma 7 is then invoked to compute the upper bound on the total number of iterations, and Lemma 34 to deduce the upper bounds on the number of evaluations of f and its derivatives. □

We emphasise that (4.73) was shown in [35] to be optimal for a quite wide class of minimisation algorithms. The slightly weaker bound $(1+\nu_{\max}(\epsilon))\tau(\epsilon)$ may be seen as the (very modest) price to pay for allowing inexact evaluations.

Focusing on the order in ϵ and using (4.74), we therefore obtain the following condensed result on evaluation (and iteration) complexity for nonconvex optimisation.

Theorem 40. Let Assumption 4.1.1 hold. Then, given $\epsilon \in (0,1)$, the ARqpDA algorithm using the modified Steps 1 (on page 75) and 2 (on page 78) needs at most

$$O\left(\epsilon^{-\frac{p+1}{p-q+1}}\right) \quad \text{iterations and (approximate) evaluations of } f$$

and at most

$$O\left(|\log(\epsilon)|\epsilon^{-\frac{p+1}{p-q+1}}\right) \quad \text{(approximate) evaluations of the } p \text{ first derivatives}$$

to compute an (ϵ,δ)-approximate q-th-order necessary minimiser for the set-constrained problem (4.3).

As indicated in the comment at the end of Section 4.2, all $\mathcal{O}(|\log(\epsilon)|)$ terms reduce to a constant independent of ϵ if exact evaluations of f and its derivatives are used, so that the above results recover the optimal complexity bounds of [35]. Furthermore, we notice that this term is completely negligible with respect to $\epsilon^{-\frac{p+1}{p-q+1}}$, concluding that the use of inexact function and derivatives evaluations has a very low impact in terms of complexity.

We conclude this section by commenting on the special case where the objective function evaluations are exact and that of the derivatives inexact. We first note that this case is already covered by the theory presented above (since (4.25) and (4.26) automatically holds as their left-hand side is identically zero), but in this case the ARqpDA algorithm can be simplified by replacing the computation of $\overline{f}(x_k+s_k)$ by that of $f(x_k+s_k)$ and by entirely skipping the verification and possible recomputation of $\overline{f}(x_k)$. As consequence, the ARqpDA algorithm only evaluates the exact objective function f once per iteration and the maximum number of such evaluations is therefore given by $\tau(\epsilon)$ instead of $2\tau(\epsilon)$, while the maximum number of (inexact) derivatives evaluations is still given by $(1+\nu_{\max}(\epsilon))\tau(\epsilon)$.

4.4 A variant of the ARqpDA algorithm

We now describe a variant of the ARqpDA algorithm for which an even better evaluation complexity can be proved, but at the price of a more restrictive dynamic accuracy strategy.

In the Step 1.0 of the ARqpDA algorithm, we allow the choice of an arbitrary set of $\{\varepsilon_{j,0}\}_{j=1}^{p}$ with the constraint that $\varepsilon_{j,0} \le \kappa_\varepsilon$ for $j \in \{1, \dots, p\}$, $p \in \{1,2\}$. This allows such accuracy thresholds to vary non-monotonically from iteration to iteration, providing considerable flexibility and allowing large inaccuracies, even if these thresholds were made small in past iterations due to local nonlinearity.

A different, more rigid, strategy is also possible: suppose that the thresholds $\{\varepsilon_{j,0}\}_{j=1}^{p}$ are not reset at each iteration, namely that

$$\text{Step 1.1 is only executed for } k = 0. \tag{4.79}$$

This clearly results in a monotonic decrease of each ε_j across all iterations. As a consequence, $\nu_{\max}(\epsilon)$ in (4.60) now bounds the total number of reductions of the ε_j over all iterations, i.e. on the entire run of the algorithm.

We then deduce that the total number of approximate evaluations of the derivatives is then bounded by $\nu_{\max}(\epsilon) + \tau(\epsilon)$ (instead of $(1 + \nu_{\max}(\epsilon))\tau(\epsilon)$) and we may establish the worst-case complexity of the resulting "monotonic" variant as follows.

Theorem 41. Let Assumption 4.1.1 hold and $\epsilon \in (0,1)$ be given. Then, the ARqpDA algorithm using the modified Steps 1 (on page 75) and 2 (on page 78) as well as the modified rule (4.79) produces an iterate x_ϵ such that (4.13) or (4.51) holds in at most (4.73) successful iterations, $\tau(\epsilon)$ iterations in total, $2\tau(\epsilon)$ (approximate) evaluations of f and $\nu_{\max}(\epsilon) + \tau(\epsilon)$ approximate evaluations of $\{\nabla_x^j f\}_{j=1}^{p}$, where $\tau(\epsilon)$ is given by (4.74), κ_p by (4.75), $\sigma_{\max}$ by (4.62), ω by (4.22) and $\nu_{\max}(\epsilon)$ by (4.60).

As above, this complexity bound can be condensed to

$$\begin{aligned} &O\left(\epsilon^{-\frac{p+1}{p-q+1}}\right) \text{ iterations and (approximate) evaluations of } f \\ &O\left(|\log(\epsilon)| + \epsilon^{-\frac{p+1}{p-q+1}}\right) \text{ (approximate) evaluations of the first } p \text{ derivatives,} \end{aligned} \tag{4.80}$$

typically improving on Theorem 40. When $p = 2$ and $q = 1$, the ARqpDA variant using the more restrictive accuracy strategy (4.79) requires at most

$$\mathcal{O}\left(|\log(\epsilon)| + \epsilon^{-3/2}\right)$$

(approximate) evaluations of the gradient, which corresponds to the bound derived for the ARC-DFO algorithm of [39]. This is not surprising, as this latter algorithm uses a monotonically decreasing sequence of finite-difference step sizes, implying monotonically decreasing gradient-accuracy thresholds.

One should however notice that the improved bound (4.80) comes at the price of asking, for potentially many iterations, an accuracy on $\{\overline{\nabla^j f}\}_{j=1}^{p}$ which is tighter than what is needed to ensure progress of the minimisation. In practice, this might be a significant drawback. We will thus restrict our attention, in what follows, to the original ARqpDA algorithm, but similar developments are obviously possible for the "monotonic" variant just discussed.

4.5 Application to unconstrained and bound-constrained first and second-order nonconvex inexact optimisation

As already mentioned, the ARqpDA algorithm, here defined for better readability and more practical application with $1 \leq q \leq p \leq 2$, has been formally written using variables q and p, since it can be in theory defined also when $1 \leq q \leq p$ ($p > 2$) is considered. This means that a p-th order Taylor's expansion is employed in the model definition (4.1), with a regularisation term of order $(p+1)$ to search for an approximate (ϵ-δ) q-th order necessary minimiser in the sense of the measure (4.20). In such a wide-ranging applicability context, the framework discussed may appear somewhat daunting in its generality. Moreover, the fact that it involves (possibly constrained) global optimisation subproblems in several of its steps may suggest that it has to remain conceptual.

We show in this section that this is not the case and stress that it is much simpler when specialised to small values of p and q (which are, for now, the most practical ones) and that our approach leads to elegant and implementable numerical algorithms. To illustrate this point, we now specify what happens for each of the three cases associated with the range $1 \leq q \leq p \leq 2$.

4.5.1 Quadratic regularisation for first-order optimality (case $p = q = 1$)

We first discuss the case where one seeks to compute a first-order critical point for an unconstrained optimisation problem using approximate function values as well as approximate gradients. This corresponds to the case $q = 1$ (first-order optimality), $p = 1$ (model with first-order Taylor's expansion and quadratic regularisation) and $\mathcal{X} = \mathbb{R}^n$.

We first note that, as pointed out in (4.11),

$$\phi^{\delta}_{f,1}(x) = \|\nabla f(x)\|\delta \quad \text{and} \quad \overline{\phi}^{\delta}_{f,1} = \|\overline{\nabla f}(x)\|\delta, \quad \text{irrespective of} \quad \delta \in (0,1], \tag{4.81}$$

which means that, since we can choose $\delta = 1$, Step 1 of the ARqpDA algorithm reduces to the computation of an approximate gradient $\overline{\nabla f}(x_k)$ with relative error ω and the verification of ϵ-approximate optimality is not yet achieved. If that is the case, computing s_k at Step 2 is also simple, since in this case it can che chosen as the global minimiser s_k^* of the model, which takes the form:

$$s_k = s_k^* = -\frac{1}{\sigma_k}\overline{\nabla f}(x_k).$$

Lemma 29 then ensures that this step is acceptable for some $\delta_k \in (0,1]$, the value of which being irrelevant since it is not used in Step 1 of the next iteration. Moreover, if the relative error on $\overline{\nabla f}(x_k)$ is bounded by ω, then

$$\begin{aligned}
|\overline{\Delta T}^f_1(x_k, s_k) - \Delta T^f_1(x_k, s_k)| &\leq \|\overline{\nabla f}(x_k) - \nabla f(x_k)\|\frac{\|\overline{\nabla f}(x_k)\|}{\sigma_k} \\
&\leq \omega\frac{\|\overline{\nabla f}(x_k)\|^2}{\sigma_k} \\
&= \omega\overline{\Delta T}^f_1(x_k, s_k)
\end{aligned}$$

and (4.18) automatically holds, so that no iteration is needed in Algorithm 10.

The resulting algorithm, where we have made the modified Step 1 explicit, is given as Algorithm 11 (AR1DA) on the facing page.

Algorithm 11 The AR1DA Algorithm (ARqpDA with $q = p = 1$).

Step 0: Initialisation. An initial point $x_0 \in \mathbb{R}^n$ and an initial regularisation parameter $\sigma_0 > 0$ are given, as well as an accuracy level $\epsilon \in (0,1)$ and a relative accuracy $\omega > 0$. The constants α, ω, κ_ε, η_1, η_2, γ_1, γ_2, γ_3 and $\sigma_{\min}$ are also given, satisfying

$$\sigma_{\min} \in (0, \sigma_0], \quad 0 < \eta_1 \le \eta_2 < 1, \quad 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3,$$
$$\kappa_\varepsilon \in (0,1], \quad \gamma_\varepsilon \in (0,1), \quad \alpha \in (0,1), \quad 0 < \omega < \min\left[\frac{1-\eta_2}{3}, \frac{\eta_1}{2}\right].$$

Set $k = 0$.

Step 1: Compute the optimality measure and check for termination. Initialise $\varepsilon_{1,0} = \kappa_\varepsilon$ and set $i = 0$. Do

1. compute $\overline{\nabla f}(x_k)$ with $\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \le \varepsilon_{1,i}$ and increment i by one.
2. if $\varepsilon_{1,i} \le \omega \|\overline{\nabla f}(x_k)\|$ and $\|\overline{\nabla f}(x_k)\| \le \epsilon/(1+\omega)$, then terminate with $x_\epsilon = x_k$; else, if $\varepsilon_{1,i} \le \omega\|\overline{\nabla f}(x_k)\|$, go to Step 2;
3. set $\varepsilon_{1,i+1} = \gamma_\varepsilon \varepsilon_{1,i}$ and return to item 1 in this enumeration.

Step 2: Step calculation. Set

$$s_k = -\overline{\nabla f}(x_k)/\sigma_k \quad \text{and} \quad \overline{\Delta T}_1^f(x_k, s_k) = \|\overline{\nabla f}(x_k)\|^2/\sigma_k.$$

Step 3: Acceptance of the trial point.
Compute $\overline{f}_k(x_k + s_k)$ ensuring that

$$|\overline{f}_k(x_k + s_k) - f(x_k + s_k)| \le \omega |\overline{\Delta T}_1^f(x_k, s_k)|. \tag{4.82}$$

Also ensure (by setting $\overline{f}_k(x_k) = \overline{f}_{k-1}(x_k)$ or by (re)computing $\overline{f}_k(x_k)$) that

$$|\overline{f}_k(x_k) - f(x_k)| \le \omega |\overline{\Delta T}_1^f(x_k, s_k)| \tag{4.83}$$

Then, compute

$$\rho_k = \frac{\overline{f}_k(x_k) - \overline{f}_k(x_k + s_k)}{\overline{\Delta T}_1^f(x_k, s_k)}. \tag{4.84}$$

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

Step 4: Regularisation parameter update. Set

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k] & \text{if } \rho_k \ge \eta_2, \\ [\sigma_k, \gamma_2 \sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_2 \sigma_k, \gamma_3 \sigma_k] & \text{if } \rho_k < \eta_1. \end{cases} \tag{4.85}$$

Increment k by one and go to Step 1.

Theorem 40 then guarantees that the AR1DA Algorithm will find an ϵ-approximate first-order minimiser for the unconstrained version of problem (4.3) in at most $\mathcal{O}(\epsilon^{-2})$ iterations and approximate evaluations of the objective function (which is proved in [35] to be optimal) and at most $\mathcal{O}(|\log(\epsilon)|\epsilon^{-2})$ approximate evaluations of the gradient.

Comments on the AR1DA algorithm

The following comments can be useful for a better contextualisation of the algorithm.

1. The accuracy requirement is truly adaptive and the absolute accuracy $\varepsilon_{1,i}$ may remain quite large as long as $\|\overline{\nabla f}(x_k)\|$ itself remains large, as shown by item 3 in

Step 1. Note that, in the worst case, Step 1 terminates whenever $\varepsilon_{1,i} \leq \epsilon\,\omega/(1+\omega)$ as one of the following situation occurs within Step 1.2: either $\|\overline{\nabla f}(x_k)\| \leq \epsilon/(1+\omega)$ or $\|\overline{\nabla f}(x_k)\| > \epsilon/(1+\omega) > \varepsilon_{1,i}/\omega$.

2. The accuracy requirement for computing $\overline{f}$ does not depend on the absolute accuracy of the gradient, but only on its (squared) norm. At initial iterations, this may be quite large.

3. The AR1DA Algorithm is very close in spirit to the Trust-Region with Dynamic Accuracy of [52 Sections 8.4.1.1 and 10.6] and, when values of f are computed exactly, essentially recovers a proposal in [26]. It is also close to the proposal of [99], which is based on an Armijo-like line search and has similar accuracy requirements.

4. Due to the definition of s_k at Step 2, we have that for any successful iteration k:

$$x_{k+1} = x_k + s_k = x_k - \frac{1}{\sigma_k}\overline{\nabla f}(x_k), \tag{4.86}$$

that can be viewed as the general iteration of a gradient descent method under inexact gradient evaluations with an adaptive choice of the so-called learning rate (it corresponds to the opposite of the coefficient of $\overline{\nabla f}(x_k)$ in (4.86)), taken as the inverse of the quadratic regulariser $\sigma_k > 0$ (see, e.g., [106]).

4.5.2 Cubic regularisation for first-order and/or second-order optimality (case $1 \leq q \leq p = 2$)

We now turn to the case where one seeks a first-order critical point ($q = 1$) for an unconstrained problem using approximate gradients and Hessians ($p = 2$).

As for the case $p = q = 1$, we have that (4.81) holds, making the verification of optimality in Step 1 relatively easy. Computing s_k is now more complicated but still practical, as it now implies minimising the regularised model m_k starting from x_k until a step s_k is found, such that

$$\|s_k\| \geq \mu\epsilon^{\frac{1}{2}} \quad \text{or} \quad \overline{\phi}^{\delta}_{m_k,1}(s_k) = \|\nabla_s m_k(s_k)\| \leq \tfrac{1}{2}\theta\|s_k\|^2$$

(as proposed in [41], see also [70, 95, 43, 60]), with the additional constraint that, for $s_k \neq 0$,

$$\max(\varepsilon_{1,i}, \varepsilon_{2,i}) \leq \omega \frac{\overline{\Delta T}^f_2(x_k, s_k)}{\chi_2(\|s_k\|)}, \tag{4.87}$$

where

$$\overline{\Delta T}^f_2(x_k, s_k) = -\overline{\nabla f}(x_k)^\top s_k - \tfrac{1}{2}s_k^\top \overline{\nabla^2 f}(x_k)s_k.$$

The resulting specification of ARqpDA for AR$q2$DA is quite similar to AR1DA and is omitted for brevity. We note what follows.

1. Algorithm AR$q2$DA is guaranteed by Theorem 40 to find an ϵ-approximate first-order minimiser for the unconstrained version of problem (4.3) in at most $\mathcal{O}(\epsilon^{-3/2})$ iterations and approximate evaluations of the objective function (which is proved in [35] to be optimal) and at most $\mathcal{O}(|\log(\epsilon)|\epsilon^{-3/2})$ approximate evaluations of the gradient and Hessian.

2. As for AR1DA, the absolute accuracies required by AR$q2$DA on the approximate function, gradient and Hessian only depend on the magnitude of the Taylor's increment, which is typically quite large in early iterations. The relative errors on the latter two remain bounded away from zero.

3. The absolute accuracies required on the approximate gradient and Hessian are comparable in magnitude, although (4.87) could be exploited to favour one with respect to the other.

Finally, The case where $p = 2$ (model employing second-order Taylor's expansion and cubic regularisation) and $q = 2$ (second-order optimality) is also computationally quite accessible: calculating the optimality measure $\overline{\phi}_{f,1}^{\delta_k}(x_k)$ or $\overline{\phi}_{m_k,1}^{\delta_k}(s_k)$ now involve a standard trust-region subproblem, for which both exact and approximate numerical solvers are known (see [52, Chapter 7] for instance), but the rest of the algorithm — in particular its adaptive accuracy requirement — is very similar to what we have just discussed (see also [35]). Theorem 40 then ensures that the resulting method converges to an ϵ-approximate second-order necessary minimiser for the unconstrained version of problem (4.3) in at most $\mathcal{O}\big(\epsilon^{-3}\big)$ iterations and approximate evaluations of the objective function and at most $\mathcal{O}\big(|\log(\epsilon)|\epsilon^{-3}\big)$ approximate evaluations of the gradient and the Hessian.

We conclude this section with a brief discussion of the case where $q = 1$ and $p \in \{1, 2\}$ as before, but where $\mathcal{X}$ is now defined by bound constraints. It is clear that evaluating and enforcing such constraints (by projection, say) has negligible cost and therefore falls in our framework. In this case, the calculations of $\overline{\phi}_{f,1}^{\delta_k}(x_k)$ or $\overline{\phi}_{m_k,1}^{\delta_k}(s_k)$ involve linear optimisation problems[†], which are computationally quite tractable. If $p = 1$, the Step 2.2 and 2.3 involve convex quadratic optimisation, while they involve minimising a regularised quadratic model if $p = 2$. All results remain the same and the ARqpDA algorithm is then guaranteed to find a bound-constrained approximate first-order approximate minimiser in at most $\mathcal{O}\big(\epsilon^{-2}\big)$ or $\mathcal{O}\big(\epsilon^{-3/2}\big)$ iterations and approximate evaluations of the objective function (which is proved in [35] to be optimal) and at most $\mathcal{O}\big(|\log(\epsilon)|\epsilon^{-2}\big)$ or $\mathcal{O}\big(|\log(\epsilon)|\epsilon^{-3/2}\big)$ approximate evaluations of the gradient and the Hessian. The same algorithms and results obviously extend to the case where $\mathcal{X}$ is a convex polyhedral set or any closed non-empty convex set, provided the cost of the projection on this set remains negligible compared to that of (approximately) evaluating the objective function and its derivatives.

4.6 A probabilistic viewpoint on ARqpDA

In this section we consider the case where the bounds $\{\varepsilon_j\}_{j=1}^p$ on the absolute errors on the derivatives $\{\nabla^j f(x)\}_{j=1}^p$ are satisfied with probability at least $(1 - t)$, with arbitrary $t \in (0, 1)$.

As already pointed out in Chapter 2, this may occur in the finite-sum setting if the approximate derivatives are obtained by some random sampling scheme. We therefore assume that (2.1) for $j \in \{1, ..., p\}$, (2.2) and (2.3) hold, with $\tau_{0,k} \stackrel{\rm def}{=} \omega|\overline{\Delta T}_p^f(x_k, s_k)|$ and $\tau_{j,k} = \varepsilon_{j,i_k}$, $j \in \{1, 2\}$, with i_k the last index considered in the inner loop for i at Step 1 of the ARqpDA algorithm. Clearly, different values for t could be chosen in (2.1), one for each index (order of the derivative) $j \in \{1, ..., p\}$, $p \in \{1, 2\}$. Similarly, different values of t in (2.2) and (2.3) could be considered. However, for the sake of simplicity, we here assume that all the inequalities involved in (2.1)–(2.3) hold with the same fixed lower bound $(1 - t)$ on the probability of success. We also assume that the events in (2.1)–(2.3) are independent.

Among the stochastic variants of trust-region and adaptive cubic regularisation methods analysed in literature, complexity results are given in expectation in [23, 44], i.e. giving an upper bound on the expected number of iterations needed to reach a first-order critical point per the first time, while in [2, 124, 125] the analysis is carried out in probability. The latter point shows that the complexity result given by the deterministic analysis holds with a certain probability, assuming that the accuracies on the objective function and/or

[†]Formerly known as linear programming problems (LPs).

derivatives are satisfied ad each iteration of the method within a certain pre-fixed (iteration independent) probability. In this section we join this second line of research, following the high-probability approach of [124, 125], where an overall and cumulative success of (2.1)–(2.3) is assumed along all the iterations up to termination.

We stress that Algorithms 9 and 10 terminate independently of the satisfaction of the accuracy requirements on the derivatives. This is due to the fact that termination relies on the inequality (4.36). Moreover, during the iterations of either of these algorithms before the last, it may happen that the accuracy on the derivatives fails to be achieved, but this has no impact on the worst-case complexity. Satisfying the accuracy requirement is only crucial at the last iteration of Algorithm 9 or 10 (that is in Steps 1.2 and 2.2).

Let $\mathcal{E}_r(S)$ be the event: "the relations

$$\|\overline{\nabla^j f}(x_k) - \nabla^j f(x_k)\| \le \tau_{j,k} \quad \text{for all } \; j \in \{1,\dots,r\}, \quad r \in \{1,2\},$$

hold for some j at Step S of the last iteration of the relevant algorithm". In Step 1.2, inexact values are computed for the first q derivatives and the probability that the event $\mathcal{E}_q(1.2)$ occurs is therefore at least $(1-t)^q$. Similarly, the probability that event $\mathcal{E}_p(2.1)$ occurs is at least $(1-t)^p$. Finally, at Step 3 of the ARqpDA algorithm, the probability that both (4.25) and (4.26) hold is at least $(1-t)^2$. Then, letting for $i \in \{1,\dots,k\}$, $\mathcal{E}_{[i]}$ be the event: "inequalities (4.18), (4.25) and (4.26) hold at iteration i of the ARqpDA algorithm", the probability that $\mathcal{E}_{[i]}$ occurs is thus at least $(1-t)^{p+q+2}$. Finally, letting $\mathcal{E}(k)$ be the event: "$\mathcal{E}_{[i]}$ occurs for all iterations $i \in \{1,\dots,k\}$ of the ARqpDA algorithm", we deduce that

$$P\Big[\mathcal{E}(k)\Big] = P\left[\bigcap_{i=1}^{k} \mathcal{E}_{[i]}\right] \ge (1-t)^{k(p+q+2)}.$$

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

$$P\Big[\mathcal{E}(k)\Big] \ge (1-t)^{k(p+q+2)} = 1-\bar{t}, \;\; \text{i.e. } \; t = 1-(1-\bar{t})^{\frac{1}{k(p+q+2)}} = \mathcal{O}\left(\frac{\bar{t}}{k(p+q+2)}\right).$$

Taking into account that, when (4.18), (4.25) and (4.26) hold, the ARqpDA algorithm terminates in at most $k = \mathcal{O}\big(\epsilon^{-\frac{p+1}{p-q+1}}\big)$ iterations (as stated by Theorem 40), we deduce the following result.

Theorem 42. Let Assumption 4.1.1 hold. Suppose that the probabilistic assumptions of this section hold and that, at each of iteration of the ARqpDA algorithm, the probability t satisfies

$$t = \mathcal{O}\left(\frac{\bar{t}\,\epsilon^{\frac{p+1}{p-q+1}}}{(p+q+2)}\right). \tag{4.88}$$

Then, given $\epsilon \in (0,1)$, the conclusions of Theorem 40 hold with probability at least $(1-\bar{t})$.

As a consequence, when $p = q = 2$ we have to choose $t = \mathcal{O}\left(\frac{1}{6}\,\bar{t}\,\epsilon^3\right)$, while, when $p = q = \beta = 1$, we have to choose $t = \mathcal{O}\left(\frac{1}{4}\,\bar{t}\,\epsilon^2\right)$.

We stress that the above analysis is unduly pessimistic in the case where $p = q = 1$. Indeed, as already noticed in Section 4.5, no reduction in $\{\varepsilon_j\}_{j=1}^p$ is necessary at Step 2, as (4.18) is automatically enforced whenever the relative error on the first derivative $\overline{\nabla f}(x)$ is bounded by ω. Noting that this last event has probability at least $(1-t)$, we can conclude that $P(\mathcal{E}_{[i]}) \ge (1-t)^3$ and to get the optimal complexity $\mathcal{O}(\epsilon^{-2})$ with probability at least $1-\bar{t}$ we need to choose $t = \mathcal{O}\left(\frac{1}{3}\,\bar{t}\,\epsilon^2\right)$.

We also emphasise that the purpose of Theorem 42 is limited to offer guidance on desirable value of t and not to prescribe an algorithmically binding bound. Indeed, some of the constants involved in the bound of Theorem 40 (and thus of Theorem 42) are typically unknown a priori (which is why we have not been more specific in (4.88)). The above argument naturally applies to (1.108), where the approximation of the objective function values and of its first and second derivatives is obtained by the uniform random subsampling procedure introduced in Section 3.5. In particular, at iteration k these approximations are still given by (2.4)–(2.6), (2.10)–(2.12) by virtue of the operator-Bernstein inequality, under the assumptions of Theorem 19.

We stress that the adaptive nature of these sample sizes is apparent in formulae (2.10)–(2.12), because they depend on x_k and $\tau_{j,k}$, which are themselves dynamically updated in the course of the ARqpDA algorithm. Depending on the size of N, it may clearly be necessary to consider the whole set $\{1, \dots, N\}$ for small values of $\{\varepsilon_j\}_{j=0}^{2}$. If the cost of evaluating functions ψ_i, for $1 \le i \le N$, is comparable for all i, the cost of evaluating $\overline{f}_k(x_k)$ amounts to the fraction $|\mathcal{D}_{k,1}|/N$ of the effort for computing the exact value $f(x_k)$. Analogous considerations hold for the objective function derivatives.

In particular, whenever N is large enough to ensure that (2.10)–(2.12) do not require the full sample, the size of the sample used to obtain a single approximate objective function value is $\mathcal{O}(\varepsilon_0^{-2})$. Analogously, gradient and Hessian values are approximated by averaging over samples of size $\mathcal{O}(\varepsilon_1^{-2})$ and $\mathcal{O}(\varepsilon_2^{-2})$, respectively. In Step 3 of the AR$1$DA algorithm, the choice $\varepsilon_0 \in \left[\gamma_\epsilon \omega \|\overline{\nabla^j f}(x_k)\|^2/\sigma_k, \omega \|\overline{\nabla^j f}(x_k)\|^2/\sigma_k\right]$ is required to ensure that (4.82)-(4.83) are satisfied. With this choice, the iteration k of the AR$1$DA algorithm requires $\mathcal{O}(\|\overline{\nabla^j f}(x_k)\|^{-4})$ ψ_i-evaluations ($\mathcal{O}(\epsilon^{-4})$ ψ_i-evaluations in the worst-case). Similarly, $\varepsilon_0 = \mathcal{O}(\omega \|\overline{\Delta T}_2^E(x_k, s_k)\|)$ is needed at iteration k of the AR$q2$DA algorithm. As a consequence, and if the algorithm does not terminate at iteration $k+1$, it follows from Lemma 35 and 38 that $\mathcal{O}(\epsilon^{-(3/(3-q))^2})$ ψ_i-evaluations may be required in the worst case.

Finally, using Lemma 34 and (4.59), we claim that each iteration of the AR$1$DA and AR$q2$DA algorithms requires at most $\mathcal{O}((1+\nu_{\max}(\epsilon))\epsilon^{-2})$ evaluations of component gradients and component Hessians, where $\nu_{\max}(\epsilon)$ has been defined in (4.60). These bounds turn out to be better or the same as those derived in [23, 44, 125]. Although they may appear discouraging, it should be kept in mind that they are valid only if N is truly large compared with $1/\epsilon$ (for instance, it has to exceed $\mathcal{O}(\epsilon^{-4})$ to allow for approximate functions in the AR$1$DA Algorithm). In other words, the sampling schemes (2.10)–(2.12) are most relevant when $1/\epsilon$ remains modest compared with N.

4.7 Chapter conclusion

We have provided a general regularisation algorithm using inexact function and derivatives values, featuring a flexible adaptive mechanism for specifying the amount of inexactness acceptable at each iteration.

This algorithm, inspired by the unifying framework proposed in [35], is applicable to unconstrained and inexpensively-constrained nonconvex optimisation problems and is here proved to gain up to second-order optimality using regularisation up to order 3, under a set of usual smoothness assumptions on the objective function f (standard in such a context) and the global Lipschitz continuity of the objective function highest derivative employed in the model definition.

Optimal evaluation (and iteration) complexity is achieved if the accuracy requirements for function and derivatives evaluations are assumed and restored in probability, when such requirements hold with a pre-fixed iteration independent probability. Nevertheless, all the results of this chapter holds for arbitrary degree of available derivatives, arbitrary order of optimality and assuming the global Hölder continuity of the objective function highest derivative (see [10]).

We have finally provided a probabilistic version of the complexity result and derived associated lower bounds on sample size in the context of subsampling.

Part III

Stochastic Complexity Analysis

Introduction to Part III

Solving optimisation problems involving inexact evaluations is a recent attractive topic, that has been investigated in three different frameworks.

- The first is that of deterministic dynamic accuracy, where it is assumed that the accuracy of f and its derivatives can be specified and fullfilled by the algorithm (see [52, Section 10.6] and [88, 66, 10], for example). In this context, adaptive conditions are derived that guarantee convergence to approximate solutions and evaluation complexity of the resulting algorithms can be analysed [10], indicating a very modest degradation of the worst-case performance compared with the case where evaluations are exact [35, 34].

- To deal with the nondeterministic aspects of these algorithms, high probability results are given in [9, 10, 50, 124, 125] and it is shown that the optimal complexity result is restored in probability, in the spirit of Subsection 3.5 and Subsection 4.6 of this thesis.

- A drawback of this approach is that nothing is said for the case where the requested accuracy requirement cannot be met or, as is often the case, cannot even be measured. In fact, the techniques listed at the previous bullet do not provide algorithms which are robust against failures to satisfy the adaptive accuracy requirements. This is in contrast with the interesting analysis of unconstrained first-order methods in [99, 127, 23]. Combining the generality of our approach with the robustness of the proposal is thus desirable.

This part of the thesis investigates this third point, motivated by the fact that inexactness in the function (and possible derivatives) values can be often seen as caused by some random noise, in which case the algorithm/user is not able to specify an accuracy level and poor accuracy might result.

The available analysis for this case differs by the assumptions made on the distribution of this noise.

In [99], the authors consider a line search method for the unbiased case and estimate its evaluation complexity for finding approximate first-order critical points. It is assumed, instead, that the function values of f, as well as those of its derivatives, can be approximated within a prescribed accuracy with a fixed and sufficiently high probability, conditioned to the past.

A similar context is considered in [17], where the objective function values are inexact but computed with accuracy guaranteed with probability one.

A trust-region variant is also proposed in [49], where the authors proved almost-surely convergence to first-order critical points. The approach of [44] includes the use of random first-order models and directions within the line search method as well as probabilistic second-order models in the ARC framework. In both cases, the authors employ exact function evaluations. A general theory for global convergence rate analysis is also provided.

More recently, [23] proposed an evaluation complexity analysis for a trust-region method (covering convergence to second-order points) using elegant properties of supermartingales, and making no assumption on bias. A recent overview is proposed in [55].

The structure of this part is as follows.

In Chapter 5 we adapt an extended version of the adaptive cubic regularisation method with dynamic inexact Hessian information for unconstrained nonconvex optimisation (problem (1.1)) presented in Chapter 3 to the stochastic optimisation setting. While exact function evaluations are still considered, this novel variant inherits the innovative use of adaptive accuracy requirements for Hessian approximations and additionally employs inexact computations of the gradient. Without restrictions on the variance of the errors, we assume that these approximations are available within a sufficiently large, but fixed, probability and we extend, in the spirit of [44], the deterministic analysis of the framework to its stochastic counterpart, showing that the expected number of iterations to reach a first-order stationary point matches the well known worst-case optimal complexity. The chapter presents the work in [6], just accepted for publication.

Chapter 6 is the core of [8], recently submitted for publication and of publication [7]. Its focus is to generalise the framework of Chapter 4 allowing random noise in derivatives and inexact function values for computing approximate minimisers of problem (4.3), preserving optimal evaluation complexity in the order of the accuracy tolerances and is here presented up to second-order optimality for inexpensively constrained smooth optimisation problems.

Chapter 5

Stochastic Analysis of an Adaptive Cubic Regularisation Method under Inexact Gradient Evaluations and Dynamic Hessian Accuracy

The cubic regularisation variant proposed in Chapter 3 employs exact gradient and ensures condition (1.31) by requiring (3.3), i.e.

$$\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\| \le c_k,$$

where we recall that the guideline for choosing c_k is the following:

$$c_k \le \begin{cases} c, \quad c > 0, & \text{if} \quad \|s_k\| \ge 1, \\ \alpha(1-\theta)\|\nabla f(x_k)\|, & \text{if} \quad \|s_k\| < 1, \end{cases} \tag{5.1}$$

with $0 \le \alpha < \frac{2}{3}$ and $0 < \theta < 1$.

Regarding the gradient approximation, in Chapter 3 we assumed to use exact gradient computations, while in [39, 79] the accuracy requirement has the following form:

$$\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \le \mu \|s_k\|^2, \tag{5.2}$$

in which $\overline{\nabla f}(x_k)$ denotes the gradient approximation and μ is a positive constant. It thus depends on the norm of the step.

In [125], as for the Hessian approximation, in order to get rid of the norm of the step, a very tight accuracy requirement in used as the absolute error has to be of the order of ϵ_1^2 at each iteration, i.e.

$$\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \le \mu \epsilon_1^2. \tag{5.3}$$

As already noticed, in [124, 125], a complexity analysis in high probability is carried out in order to cover the situation where accuracy requirements (3.37) and (5.3) for Hessian and gradient estimation are satisfied only with a sufficiently large probability.

At variance, the behaviour of cubic regularisation approaches employing approximated derivatives is analysed in expectation in [44], assuming that (1.31) and (5.2) are satisfied with high probability. In the finite-sum minimisation context, accuracy requirements (1.31), (3.1) and (5.2) can be enforced with high probability by subsampling via an inner iterative process. Namely, the approximated derivative is computed using a predicted accuracy, the step s_k is computed and, if the predicted accuracy is larger than the required accuracy, the predicted accuracy is progressively decreased (the sample size progressively increased), until the accuracy requirement is satisfied.

In this chapter we generalise the method given in Chapter 3, keeping the practical adaptive criterion (3.3), which is implemented without including an inner loop, but allowing inexactness in the gradient as well. More specifically, we require that the gradient approximation satisfies the following relative implicit condition:

$$\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \leq \zeta_k \|\overline{\nabla f}(x_k)\|^2, \tag{5.4}$$

where ζ_k is an iteration-dependent nonnegative parameter.

Unlike [44, 79] (see (5.2)), this latter condition does not depend on the norm of the step. As a consequence, its practical implementation calls for an inner loop that can be performed before the step computation and, remarkably, extra-computations of the step are not needed.

In the upcoming analysis, we assume that the accuracy requirements (3.3) and (5.4) are satisfied with high probability and we perform, in the spirit of [44], the stochastic analysis of the resulting method, showing that the expected number of iterations needed to reach an ϵ_1-approximate first-order critical point is, in the worst-case, of the order of $\epsilon_1^{-3/2}$. Of course, this analysis also applies to the method given in Chapter 3, that can be seen as a particular case (exact gradient computations).

The rest of this chapter is organised as follows. In Section 5.1 we introduce a stochastic ARC algorithm with inexact gradients and dynamic Hessian accuracy, stating the main assumptions on the stochastic process induced by the algorithm. Relying on several existing results and deriving some additional outcomes, Section 5.2 is then devoted to perform the iteration complexity analysis of the framework, while Section 5.3 proposes a practical guideline to apply the method for solving finite-sum minimisation problems.

5.1 A stochastic cubic regularisation algorithm with inexact derivatives evaluations: the SARC-IGDH algorithm

With reference to problem (1.1), we still consider the set of smoothness assumptions on f given by Assumption 1.2.1, but we also need to assume that the Hessian matrix of problem (1.1) is global Lipschitz continuous, as stated below.

Assumption 5.1.1. *The Hessian of the objective function in* (1.1) *is assumed to be globally Lipschitz continuous with Lipschitz constant $L > 0$, i.e.*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|, \tag{5.5}$$

for all $x, y \in \mathbb{R}^n$.

The iterative method we are going to introduce is, basically, the stochastic counterpart of the ARC-DH algorithm, adding inexactness for first-order information. We here list the main differences.

- At iteration k, given the trial step s, the value of the objective function at $x_k + s$ is predicted by mean of a cubic model $m_k(s)$ defined in terms of an approximate Taylor's expansion of f centered at x_k with increment s, truncated to the second order, namely

$$m_k(s) = f(x_k) + \overline{\nabla f}(x_k)^T s + \frac{1}{2} s^T \overline{\nabla^2 f}(x_k) s + \frac{\sigma_k}{3}\|s\|^3 \overset{\text{def}}{=} \widetilde{T}_2(x_k, s) + \frac{\sigma_k}{3}\|s\|^3, \tag{5.6}$$

in which the gradient $\overline{\nabla f}(x_k)$ and the Hessian matrix $\overline{\nabla^2 f}(x_k)$ represent approximations of $\nabla f(x_k)$ and $\nabla^2 f(x_k)$, respectively.

- Given this new model definition, the minimisation of the cubic model (5.6) at each iteration is still done following the strategy in ARC-DH, in the sense that the minimiser s_k satisfies

$$m_k(x_k, s_k, \sigma_k) < m_k(x_k, 0, \sigma_k), \tag{5.7}$$

$$\|\nabla_s m_k(x_k, s_k, \sigma_k)\| \le \theta_k \|\overline{\nabla f}(x_k)\|, \tag{5.8}$$

  for all $k \ge 0$ and some $0 \le \theta_k \le \theta$, $\theta \in [0, 1)$. Practical choices for θ_k are, for instance, $\theta_k = \theta \min\left(1, \frac{\|s_k\|^2}{\|\overline{\nabla f}(x_k)\|}\right)$ or $\theta_k = \theta \min(1, \|s_k\|)$ (see, e.g., [9]), leading to

$$\|\nabla_s m_k(x_k, s_k, \sigma_k)\| \le \theta \min\left(\|s_k\|^2, \|\overline{\nabla f}(x_k)\|\right), \tag{5.9}$$

  and

$$\|\nabla_s m_k(x_k, s_k, \sigma_k)\| \le \theta \min(1, \|s_k\|)\|\overline{\nabla f}(x_k)\|, \tag{5.10}$$

  respectively. Note that, differently from (3.5), (3.26) and (3.27) in Chapter 3, conditions (5.8), (5.9) and (5.10) employ inexact gradient evaluations.

- The relative decrease is defined by

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\widetilde{T}_2(x_k, 0) - \widetilde{T}_2(x_k, s_k)}, \tag{5.11}$$

  with $\widetilde{T}_2$ as in (5.6).

- If $\rho_k \ge \eta$, with $\eta \in (0, 1)$ a prescribed decrease fraction, then the trial point is accepted, the iteration is declared successful, the regularisation parameter is decreased by a factor γ (until a lower bound) and we go on recomputing the approximate model at the updated iterate; otherwise, an unsuccessful iteration occurs: the point $x_k + s_k$ is rejected, the regulariser is increased by a factor γ, a new approximate model at x_k is computed and a new trial step s_k is recomputed.

- Being within the range of a scheme inducing a stochastic process, no termination criterion is employed.

At each iteration, the model $m_k(s)$ involved relies on inexact quantities (see in (5.6)), that can be considered as realisations of random variables. Hereafter, all random quantities are denoted by capital letters, while the use of small letters is reserved for their realisations (that, for instance, appear in the ARC algorithms considered in the first two parts of this thesis). In particular, let us denote a random model at iteration k as M_k, while we use the notation m_k for its realisation. Moreover, we denote by $\overline{\nabla f}(X_k)$ and $\overline{\nabla^2 f}(X_k)$ the random variables for $\overline{\nabla f}(x_k)$ and $\overline{\nabla^2 f}(x_k)$, respectively. Consequently, the iterates X_k, as well as the regularisers Σ_k and the steps S_k are the random variables with realisations x_k, σ_k and s_k, respectively.

The core of the analysis exploited in this chapter is to derive the expected worst-case iteration complexity bound to approach an ϵ_1-approximate first-order optimality point; that is, given a first-order tolerance $\epsilon_1 \in (0, 1)$, the number of steps $\overline{k}$ (in the worst-case) such that an iterate $x_{\overline{k}}$ satisfying

$$\|\nabla f(x_{\overline{k}})\| \le \epsilon_1$$

is reached, among all possible realisations of the algorithm.

To this purpose, after the description of the algorithm, we state the main definitions and hypotheses needed to carry on with the analysis up to the complexity result.

Our algorithm is reported below.

Algorithm 12 The Stochastic ARC algorithm with Inexact Gradient and Dynamic Hessian (SARC-IGDH) accuracy.

```
Step 0: Initialisation. An initial point $x_0 \in \mathbb{R}^n$ and an initial regularisation parameter $\sigma_0 > 0$
    are given. The constants $\theta$, $\alpha$, $\eta$, $\gamma$, $\sigma_{\min}$ and $c$ are also given, such that
        $0 < \theta < 1,\ \alpha \in \left[0, \frac{2}{3}\right),\ \sigma_{\min} \in (0, \sigma_0],\ 0 < \eta < \frac{2-3\alpha}{2},\ \gamma > 1,\ c > 0.$   (5.12)
    Compute $f(x_0)$ and set $k = 0$, $\mathrm{flag} = 1$.
Step 1: Gradient approximation. Compute an approximate gradient $\overline{\nabla f}(x_k)$
Step 2: Hessian approximation (model costruction). If $\mathrm{flag} = 1$, set $c_k = c$; else, set $c_k =$
    $\alpha(1-\theta)\|\overline{\nabla f}(x_k)\|$.
    Compute an approximate Hessian $\overline{\nabla^2 f}(x_k)$ that satisfies the condition (3.3) with a
    prefixed probability. Form the model $m_k(s)$ defined in (5.6).
Step 3: Step calculation. Choose $\theta_k \le \theta$. Compute the step $s_k$ satisfying (5.7)–(5.8).
Step 4: Check on the norm of the trial step. If $\|s_k\| < 1$ and $\mathrm{flag} = 1$ and
    $c > \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|$,
        set $x_{k+1} = x_k$, $\sigma_{k+1} = \sigma_k$, $\mathrm{flag} = 0$   (unsuccessful iteration)
        set $k = k + 1$ and go to Step 1.
Step 5: Acceptance of the trial point and parameters update. Compute $f(x_k + s_k)$ and
    the relative decrease $\rho_k$ defined in (5.11).
    If $\rho_k \ge \eta$,
        set $x_{k+1} = x_k + s_k$, set $\sigma_{k+1} = \max[\sigma_{\min}, \frac{1}{\gamma}\sigma_k]$.   (successful iteration)
        If $\|s_k\| \ge 1$, set $\mathrm{flag} = 1$; otherwise, set $\mathrm{flag} = 0$.
    Else,
        set $x_{k+1} = x_k$, $\sigma_{k+1} = \gamma\sigma_k$.   (unsuccessful iteration)
    Set $k = k + 1$ and go to Step 1.
```

We note that the SARC-IGDH algorithm generates a random process

$$\{X_k, S_k, M_k, \Sigma_k, C_k\}, \tag{5.13}$$

($X_0 = x_0$ and $\Sigma_0 = \sigma_0$ are deterministic), where C_k refers to the random variable for the dynamic Hessian accuracy c_k, that is adaptively defined in Step 2 of the SARC-IGDH algorithm. In fact, since its definition relies on random quantities, C_k constitutes a random variable too. We recall that, in the deterministic counterpart given by the ARC-DH algorithm, the Hessian approximation $\overline{\nabla^2 f}(x_k)$ computed at iteration k has to satisfy the absolute accuracy requirement (3.3). Here, this condition is assumed to be satisfied only with a certain probability (see Assumption 5.1.2).

The main goal is thus to prove that, if M_k is sufficiently accurate with a sufficiently high probability conditioned to the "past", then the stochastic process preserves optimal iteration complexity in expectation, to reach an ϵ_1-approximate first-order critical point. To this scope, the next section is devoted to state the basic probabilistic accuracy assumptions and definitions.

5.1.1 Main assumptions on the algorithm

For $k \geq 0$, to formalise the conditioning on the past, let $\mathcal{A}_{k-1}^M$ denote the $\widehat{\sigma}$-algebra induced by the random variables M_0, M_1,..., M_{k-1}, with $\mathcal{A}_{-1}^M = \widehat{\sigma}(x_0)$.

We first consider the following definitions for measuring the accuracy of the model estimates.

Definition 5.1.1 (Accurate model)**.** *A sequence of random models $\{M_k\}$ is said to be p_*-probabilistically sufficiently accurate for the SARC-IGDH algorithm, $p_* \in (0,1]$, with respect to the corresponding sequence $\{X_k, S_k, \Sigma_k, C_k\}$, if the event $\mathcal{M}_k = \mathcal{M}_k^{(1)} \cap \mathcal{M}_k^{(2)} \cap \mathcal{M}_k^{(3)}$, with*

$$\begin{aligned} \mathcal{M}_k^{(1)} &= \left\{ \|\overline{\nabla f}(X_k) - \nabla f(X_k)\| \leq \kappa(1-\theta)^2 \left(\frac{\|\overline{\nabla f}(X_k)\|}{\Sigma_k} \right)^2, \quad \kappa > 0 \right\}, && (5.14) \\ \mathcal{M}_k^{(2)} &= \left\{ \|\overline{\nabla^2 f}(X_k) - \nabla^2 f(X_k)\| \leq C_k \right\}, && (5.15) \\ \mathcal{M}_k^{(3)} &= \left\{ \|\overline{\nabla f}(x_k)\| \leq \kappa_g, \quad \|\overline{\nabla^2 f}(x_k)\| \leq \kappa_B, \qquad \kappa_g > 0,\ \kappa_B > 0 \right\}, && (5.16) \end{aligned}$$

satisfies

$$P(\mathcal{M}_k | \mathcal{A}_{k-1}^M) = \mathbb{E}[\mathbb{1}_{\mathcal{M}_k} | \mathcal{A}_{k-1}^M] \geq p_*. \tag{5.17}$$

What follows is an assumption regarding the nature of the stochastic information used by the SARC-IGDH algorithm.

Assumption 5.1.2. *We assume that the sequence of random models $\{M_k\}$, generated by the SARC-IGDH algorithm, is p_*-probabilistically sufficiently accurate for some sufficiently high probability $p_* \in \left(\frac{2}{3}, 1\right]$.*

5.2 Complexity analysis of the algorithm

For a given level of tolerance ϵ_1, the aim of this section is to derive a bound on the expected number of iterations $\mathbb{E}[N_\epsilon]$ which is needed, in the worst-case, to reach an ϵ_1-approximate first-order stationary point.

Specifically, N_ϵ denotes a random variable corresponding to the number of steps required by the process until $\|\nabla f(X_k)\| \leq \epsilon_1$ occurs for the first time, namely

$$N_\epsilon = \inf\{k \geq 0 \mid \|\nabla f(X_k)\| \leq \epsilon_1\}; \tag{5.18}$$

indeed, N_ϵ can be seen as a stopping time for the stochastic process generated by the SARC-IGDH algorithm (see [23, Definition 2.1]).

The analysis follows the path of [44], but some results need to be proved as for the adopted accuracy requirements for gradient and Hessian and failures in the sense of Step 4. It is preliminarly useful to sum up a series of existing lemmas from [44, 9] and to derive some of their suitable extensions, which will be of paramount importance to perform the complexity analysis of our stochastic method. These lemmas are recalled in the following subsection.

5.2.1 Existing and preliminary results

We observe that each iteration k of the SARC-IGDH algorithm with $\mathbb{1}_{\mathcal{M}_k} = 1$ corresponds to an iteration of the ARC-DH algorithm, before termination, except for the fact that in the SARC-IGDH algorithm the model (5.6) is defined not only using inexact Hessian information, but also considering an approximate gradient.

In particular, the nature of the accuracy requirement for the gradient approximation given by (5.14) is different from the one for the Hessian approximation, namely (5.15). In fact, a realisation c_k of the upper bound C_k in (5.15), needed to obtain an approximate Hessian $\overline{\nabla^2 f}(x_k)$, is determined by the mechanism of the algorithm and is available when forming the Hessian approximation $\overline{\nabla^2 f}(x_k)$. On the other hand, (5.14) is an implicit condition and can be practically gained computing the gradient approximation within a prescribed absolute accuracy level, that is eventually reduced to recompute the inexact gradient $\overline{\nabla f}(x_k)$; but, in contrast with [44, Algorithm 4.1], without additional step computations, which is performed only once per iteration at Step 3 of the SARC-IGDH algorithm.

We will see that, for any realisation of the algorithm, if the model is accurate (i.e., $\mathbb{1}_{\mathcal{M}_k} = 1$), then there exist $\delta \geq 0$ and $\xi_k > 0$ such that

$$\|(\overline{\nabla f}(x_k) - \nabla f(x_k))s_k\| \leq \delta \|s_k\|^3, \qquad \|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\| \leq \xi_k \|s_k\|^2,$$

which will be fundamental to recover optimal complexity. At this regard, let us consider the following definitions and state the lemma below.

With reference to the SARC-IGDH algorithm, for all $0 \leq k \leq l$, $l \in \{0, ..., N_\epsilon - 1\}$, we define the events

$$\begin{aligned} \mathcal{S}_k &= \{ \text{ iteration } k \text{ is successful} \}, \\ \mathcal{U}_{k,1} &= \{ \text{ iteration } k \text{ is unsuccessful in the sense of Step 5} \}, \\ \mathcal{U}_{k,2} &= \{ \text{ iteration } k \text{ is unsuccessful in the sense of Step 4} \}. \end{aligned} \tag{5.19}$$

We underline that if $k \in \mathcal{U}_{k,1}$ then $\rho_k < \eta$, while $k \in \mathcal{U}_{k,2}$ if and only if $\|s_k\| < 1$, $\text{flag} = 1$ and $c > \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|$. Moreover, if $\rho_k < \eta$ and a failure in Step 4 does not occur, then $k \in \mathcal{U}_{k,1}$.

Lemma 43. Consider any realisation of the SARC-IGDH algorithm. Then, at each iteration k such that $\mathbb{1}_{\mathcal{M}_k^{(1)} \cap \mathcal{M}_k^{(3)}} = 1$ (accurate gradient and bounded inexact derivatives), we have

$$\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \leq \delta \|s_k\|^2, \qquad \delta \overset{\text{def}}{=} \kappa \left(\frac{\kappa_B}{\sigma_{\min}} + 1 \right) \max\left[\frac{\kappa_g}{\sigma_{\min}}, \frac{\kappa_B}{\sigma_{\min}} + 1 \right], \tag{5.20}$$

and, thus,

$$\|(\overline{\nabla f}(x_k) - \nabla f(x_k))s_k\| \leq \delta \|s_k\|^3. \tag{5.21}$$

Proof. Let us consider k such that $\mathbb{1}_{\mathcal{M}_k^{(1)} \cap \mathcal{M}_k^{(3)}} = 1$. Using (5.8) we obtain

$$\begin{aligned} \theta \|\overline{\nabla f}(x_k)\| &\geq \|\nabla_s m(x_k, s_k, \sigma_k)\| = \left\| \overline{\nabla f}(x_k) + \overline{\nabla^2 f}(x_k)s_k + \sigma_k s_k \|s_k\| \right\| \\ &\geq \|\overline{\nabla f}(x_k)\| - \|\overline{\nabla^2 f}(x_k)\| \, \|s_k\| - \sigma_k \|s_k\|^2. \end{aligned} \tag{5.22}$$

We can then distinguish between two different cases. If $\|s_k\| \geq 1$, from (5.22) and (5.16) we have that

$$\theta\|\overline{\nabla f}(x_k)\| \geq \|\overline{\nabla f}(x_k)\| - \|\overline{\nabla^2 f}(x_k)\|\,\|s_k\|^2 - \sigma_k\|s_k\|^2 \geq \|\overline{\nabla f}(x_k)\| - (\kappa_B + \sigma_k)\|s_k\|^2,$$

which is equivalent to

$$\|s_k\|^2 \geq \frac{(1-\theta)\|\overline{\nabla f}(x_k)\|}{\kappa_B + \sigma_k}.$$

Consequently, by (5.14) and (5.16),

$$\begin{aligned}\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| &\leq \kappa\left(\frac{1-\theta}{\sigma_k}\right)^2\|\overline{\nabla f}(x_k)\|^2 \leq \frac{\kappa\kappa_g(1-\theta)^2\|\overline{\nabla f}(x_k)\|}{\sigma_k^2\|s_k\|^2}\|s_k\|^2\\ &\leq \kappa\kappa_g(1-\theta)\frac{\kappa_B+\sigma_k}{\sigma_k^2}\|s_k\|^2 \leq \kappa\frac{\kappa_g}{\sigma_{\min}}\left(\frac{\kappa_B}{\sigma_{\min}}+1\right)\|s_k\|^2, \qquad (5.23)\end{aligned}$$

where in the last inequality we have used that $\theta \in (0,1)$ and $\sigma_k \geq \sigma_{min}$.
If, instead, $\|s_k\| < 1$, the inequalities (5.22) and (5.16) lead to

$$\theta\|\overline{\nabla f}(x_k)\| \geq \|\overline{\nabla f}(x_k)\| - \|\overline{\nabla^2 f}(x_k)\|\,\|s_k\| - \sigma_k\|s_k\| \geq \|\overline{\nabla f}(x_k)\| - (\kappa_B + \sigma_k)\|s_k\|,$$

obtaining that

$$\|s_k\| \geq \frac{(1-\theta)\|\overline{\nabla f}(x_k)\|}{\kappa_B + \sigma_k}. \qquad (5.24)$$

Hence, by squaring both sides in the above inequality and using (5.14), $\theta \in (0,1)$ and $\sigma_k \geq \sigma_{min}$, we obtain

$$\begin{aligned}\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| &\leq \kappa\left(\frac{1-\theta}{\sigma_k}\right)^2\|\overline{\nabla f}(x_k)\|^2 = \frac{\kappa(1-\theta)^2\|\overline{\nabla f}(x_k)\|^2}{\sigma_k^2\|s_k\|^2}\|s_k\|^2\\ &\leq \kappa\left(\frac{\kappa_B+\sigma_k}{\sigma_k}\right)^2\|s_k\|^2 \leq \kappa\left(\frac{\kappa_B}{\sigma_{\min}}+1\right)^2\|s_k\|^2. \qquad (5.25)\end{aligned}$$

Inequality (5.20) then follows by virtue of (5.23) and (5.25), while (5.21) stems from (5.20), by means of the triangle inequality. □

The following Lemma is a slight modification of Lemma 20.

Lemma 44. Consider any realisation of the SARC-IGDH algorithm and assume that $c \geq \alpha(1-\theta)\kappa_g$. Then, at each iteration k such that $\mathbb{1}_{\mathcal{M}_k^{(2)}\cap\mathcal{M}_k^{(3)}}(1-\mathbb{1}_{\mathcal{U}_{k,2}}) = 1$ (successful or unsuccessful in the sense of Step 5, with accurate Hessian and bounded inexact derivatives), we have

$$\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\| \leq c_k \leq \xi_k\|s_k\|, \qquad \xi_k \overset{\text{def}}{=} \max[c, \alpha(\kappa_B + \sigma_k)], \qquad (5.26)$$

and, thus,

$$\|(\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k))s_k\| \leq \xi_k\|s_k\|^2. \qquad (5.27)$$

Proof. Let us consider k such that $\mathbb{1}_{\mathcal{M}_k^{(2)}\cap\mathcal{M}_k^{(3)}}(1-\mathbb{1}_{\mathcal{U}_{k,2}}) = 1$. The SARC-IGDH algorithm ensures that, if $\|s_k\| \geq 1$, then $c_k = c$ or

$$c_k = \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|. \qquad (5.28)$$

Therefore, (5.28), $\|s_k\| \geq 1$ and (5.16) give

$$\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\| \leq c_k \leq \max[c, \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|] \leq \max[c, \alpha(1-\theta)\kappa_g] \leq c\|s_k\|, \quad (5.29)$$

where we have considered the assumption $c \geq \alpha(1-\theta)\kappa_g$. On the other hand, Step 4 guarantees the choice

$$c_k \leq \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|, \quad (5.30)$$

when $\|s_k\| < 1$. In this case, the inequality (5.24) still holds. Thus,

$$\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\| \leq c_k = \frac{c_k}{\|s_k\|}\|s_k\| \leq \frac{c_k(\kappa_B + \sigma_k)}{(1-\theta)\|\overline{\nabla f}(x_k)\|}\|s_k\| \leq \alpha(\kappa_B + \sigma_k)\|s_k\|, \quad (5.31)$$

where the last inequality is due to (5.30). Finally, (5.29) and (5.31) imply (5.26), while (5.27) follows by (5.26) via the triangle inequality. □

The next lemma bounds the decrease of the objective function on successful iterations, irrespectively of the satisfaction of the accuracy requirements for gradient and Hessian approximations. Its proof is very close to the one of Lemma 35 (see, also, [22, Lemma 2.1]).

Lemma 45. Consider any realisation of the SARC-IGDH algorithm. At each iteration k we have

$$\widetilde{T}_2(x_k, 0) - \widetilde{T}_2(x_k, s_k) > \frac{\sigma_k}{3}\|s_k\|^3 \geq \frac{\sigma_{\min}}{3}\|s_k\|^3 > 0. \quad (5.32)$$

Hence, on every successful iteration k:

$$f(x_k) - f(x_{k+1}) > \eta\frac{\sigma_k}{3}\|s_k\|^3 \geq \eta\frac{\sigma_{\min}}{3}\|s_k\|^3 > 0. \quad (5.33)$$

Proof. We first notice that, by (5.7), we have that $\|s_k\| \neq 0$. Inequality (5.32) indeed directly follows from (5.6) and (5.7). The second part of the thesis is easily proved taking into account that, if k is successful, then (5.32) implies

$$f(x_k) - f(x_{k+1}) \geq \eta(\widetilde{T}_2(x_k, 0) - \widetilde{T}_2(x_k, s_k)) > \eta\frac{\sigma_k}{3}\|s_k\|^3.$$

□

As a corollary, because of the fact that $x_{k+1} = x_k$ on each unsuccessful iteration k, for any realisation of the SARC-IGDH algorithm we have that

$$f(x_k) - f(x_{k+1}) \geq 0.$$

We now show that, if the model is accurate, there exists a constant $\overline{\sigma} > 0$ such that an iteration is successful or unsuccessful in the sense of Step 4 ($\mathbb{1}_{\mathcal{M}_k}(1 - \mathbb{1}_{\mathcal{U}_{k,1}}) = 1$), whenever $\sigma_k \geq \overline{\sigma}$. In other words, it is an iteration at which the regulariser is not increased.

Lemma 46. Let Assumption 1.2.1 (ii) and Assumption 5.1.1 hold. Let δ be given in (5.20), $\beta > 1$ and assume $c \geq \alpha(1-\theta)\kappa_g$. For any realisation of the SARC-IGDH algorithm, if the model is accurate and

$$\sigma_k \geq \overline{\sigma} \overset{\text{def}}{=} \max\left[\frac{6\delta + 3\alpha\kappa_B + L}{2(1-\eta) - 3\alpha}, \frac{6\delta + 3c + L}{2(1-\eta)}, \beta\sigma_0\right] > 0, \tag{5.34}$$

then the iteration k is successful or a failure in the sense of Step 4 occurs.

Proof. The proof is inspired by the one of Lemma 21. Let us consider an iteration k such that $\mathbb{1}_{\mathcal{M}_k}(1 - \mathbb{1}_{\mathcal{U}_{k,1}}) = 1$ and the definition of ρ_k in (5.11). Assume that a failure in the sense of Step 4 does not occur. If $\rho_k - 1 \geq 0$, then iteration k is successful by definition. We can thus focus on the case in which $\rho_k - 1 < 0$. In this situation, the iteration k is successful provided that $1 - \rho_k \leq 1 - \eta$. From (5.5) and the Taylor's expansion of f centered at x_k with increment s it first follows that

$$f(x_k + s) \leq f(x_k) + \nabla f(x_k)^\top s + \frac{1}{2} s^\top \nabla^2 f(x_k) s + \frac{L}{6}\|s\|^3. \tag{5.35}$$

Therefore, since $\mathbb{1}_{\mathcal{M}_k} = 1$,

$$\begin{aligned} f(x_k + s_k) - \overline{T}_2(x_k, s_k) &\leq (\nabla f(x_k) - \overline{\nabla f}(x_k))^\top s_k + \frac{1}{2} s_k^\top (\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)) s_k + \frac{L}{6}\|s_k\|^3 \\ &\leq \|\overline{\nabla f}(x_k) - \nabla f(x_k)\|\|s_k\| + \frac{1}{2}\|\overline{\nabla^2 f}(x_k) - \nabla^2 f(x_k)\|\|s_k\|^2 + \frac{L}{6}\|s_k\|^3 \\ &\leq \left(\delta + \frac{L}{6} + \frac{\xi_k}{2}\right)\|s_k\|^3, \end{aligned} \tag{5.36}$$

where we have used (5.20) and (5.26). Thus, by (5.36) and (5.32),

$$1 - \rho_k = \frac{f(x_k + s_k) - \widetilde{T}_2(x_k, s_k)}{\widetilde{T}_2(x_k, 0) - \widetilde{T}_2(x_k, s_k)} < \frac{(6\delta + 3\xi_k + L)\,\|s_k\|^3}{2\sigma_k\|s_k\|^3} = \frac{6\delta + 3\xi_k + L}{2\sigma_k}.$$

Depending on the maximum in the definition of ξ_k in (5.26), two different cases can then occur. If $\xi_k = c$, then $1 - \rho_k \leq 1 - \eta$, provided that

$$\sigma_k \geq \frac{6\delta + 3c + L}{2(1-\eta)}.$$

Otherwise, if $c < \alpha(\kappa_B + \sigma_k)$, so that $\xi_k = \alpha(\kappa_B + \sigma_k)$, then

$$1 - \rho_k < \frac{6\delta + 3\alpha(\kappa_B + \sigma_k) + L}{2\sigma_k} \leq 1 - \eta,$$

provided that

$$\sigma_k \geq \frac{6\delta + 3\alpha\kappa_B + L}{2(1-\eta) - 3\alpha}.$$

In conclusion, the iteration k is successful if (5.34) holds. Note that $\overline{\sigma}$ is a positive lower bound on σ_k because of the ranges for the values of η and α in (5.12). $\square$

As done in Lemma 6, we can now use some of the results from the proof of the previous lemma to prove the following, giving a crucial relation between the step length $\|s_k\|$ and the true gradient norm $\|\nabla f(x_k + s_k)\|$ at the next iteration.

Lemma 47. Let Assumption 1.2.1 (ii) and Assumption 5.1.1 hold. Assume also that $c \geq \alpha(1-\theta)\kappa_g$. For any realisation of the SARC-IGDH algorithm, at each iteration k such that $\mathbb{1}_{\mathcal{M}_k}(1-\mathbb{1}_{\mathcal{U}_{k,2}}) = 1$ (accurate in which the iteration is successful or a failure in the sense of Step 5 occurs), we have

$$\|s_k\| \geq \sqrt{\zeta_k \|\nabla f(x_k+s_k)\|}, \tag{5.37}$$

for some positive ζ_k, whenever s_k satisfies (5.9). Moreover, (5.37) holds even in case s_k satisfies (5.10), provided that there exists $L_g > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L_g \|x-y\|, \tag{5.38}$$

for all $x, y \in \mathbb{R}^n$.

Proof. Let us consider an iteration k such that $\mathbb{1}_{\mathcal{M}_k}(1-\mathbb{1}_{\mathcal{U}_{k,2}}) = 1$. From the Taylor's series of $\nabla f(x)$ centered at x_k with increment s and the definition of the model (5.6), proceeding as in the proof of Lemma 6, we obtain

$$\begin{aligned} \|\nabla f(x_k+s_k) - \nabla_s \widetilde{T}_2(x_k+s_k)\| &\leq \|\nabla f(x_k) - \overline{\nabla f}(x_k)\| + \|(\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k))s_k\| \\ &\quad + \int_0^1 \|\nabla^2 f(x_k+\tau s) - \nabla^2 f(x_k)\|\|s_k\|\, d\tau \\ &\leq \left(\delta + \xi_k + \frac{L}{2}\right)\|s_k\|^2, \end{aligned} \tag{5.39}$$

where we have used (5.20), (5.27) and (5.5). Moreover, since $\nabla_s m(s_k) = \nabla_s \widetilde{T}_2(x_k, s_k) + \sigma_k \|s_k\| s_k$, it follows:

$$\|\nabla f(x_k+s_k)\| \leq \|\nabla f(x_k+s_k) - \nabla_s \widetilde{T}_2(x_k, s_k)\| + \|\nabla_s m(s_k)\| + \sigma_k \|s_k\|^2. \tag{5.40}$$

As a consequence, the thesis follows from (5.39)–(5.40) with

$$\zeta_k^{-1} \overset{\text{def}}{=} \left(\delta + \xi_k + \frac{L}{2} + \theta + \sigma_k\right) > 0, \tag{5.41}$$

when the stopping criterion (5.9) is considered.
Assume now that (5.10) is used for Step 3 of the SARC-IGDH algorithm. Inequalities (5.20) and (5.38) imply that

$$\begin{aligned} \|\overline{\nabla f}(x_k)\| &\leq \|\overline{\nabla f}(x_k) - \nabla f(x_k)\| + \|\nabla f(x_k) - \nabla f(x_k+s_k)\| + \|\nabla f(x_k+s_k)\| \\ &\leq \delta\|s_k\|^2 + L_g\|s_k\| + \|\nabla f(x_k+s_k)\|. \end{aligned} \tag{5.42}$$

By using (5.39)–(5.40) and plugging (5.42) into (5.10), we finally have

$$\|\nabla f(x_k+s_k)\|(1-\theta) \leq \left[(1+\theta)\delta + \xi_k + \frac{L}{2} + \theta L_g + \sigma_k\right]\|s_k\|^2,$$

which is equivalent to (5.37), with

$$\zeta_k \overset{\text{def}}{=} \frac{1-\theta}{(1+\theta)\delta + \xi_k + L/2 + \theta L_g + \sigma_k} > 0. \tag{5.43}$$

□

Remark 9. It is worth noticing that the global Lipschitz continuity of the gradient, namely (5.38), is needed only when condition (5.10) is used at Step 3 of the SARC-IGDH algorithm.

We finally report a result from [44] that will be central to carry out the complexity analysis addressed in the following two subsections.

Lemma 48. [44, Lemma 2.1] Let N_ϵ be the hitting time defined as in (5.18). For all $k < N_\epsilon$, let $\{\mathcal{M}_k\}$ be the sequence of events in Definition 5.1.1 so that (5.17) holds. Let $\mathbb{1}_{W_k}$ be a nonnegative stochastic process such that $\widehat{\sigma}(\mathbb{1}_{W_k}) \subseteq \mathcal{A}_{k-1}^M$, for any $k \geq 0$. Then,

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}\mathbb{1}_{\mathcal{M}_k}\right] \geq p_*\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}\right].$$

Similarly,

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}(1-\mathbb{1}_{\mathcal{M}_k})\right] \leq (1-p_*)\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}\right].$$

Proof. [44, Lemma 2.1] The proof is a simple consequence of properties of expectations, see for example [113, Property H^* on page 216].

$$\mathbb{E}[\mathbb{1}_{\mathcal{M}_k}|\mathbb{1}_{W_k}] = \mathbb{E}\left[\mathbb{E}\left[\mathbb{1}_{\mathcal{M}_k}|\mathcal{A}_{k-1}^M\right]|\mathbb{1}_{W_k}\right] \geq \mathbb{E}[p_*|\mathbb{1}_{W_k}] = p_*,$$

where we also used that $\widehat{\sigma}(\mathbb{1}_{W_k}) \subseteq \mathcal{A}_{k-1}^M$. Hence by the law of total expectation, we have $\mathbb{E}[\mathbb{1}_{W_k}\mathbb{1}_{\mathcal{M}_k}] = \mathbb{E}[\mathbb{1}_{W_k}\mathbb{E}[\mathbb{1}_{\mathcal{M}_k}|\mathbb{1}_{W_k}]] \geq p_*\mathbb{E}[\mathbb{1}_{W_k}]$. Similarly, we can derive $\mathbb{E}[\mathbb{1}_{\{k<N_\epsilon\}}\mathbb{1}_{W_k}\mathbb{1}_{\mathcal{M}_k}] \geq p_*\mathbb{E}[\mathbb{1}_{\{k<N_\epsilon\}}\mathbb{1}_{W_k}]$, because $\mathbb{1}_{\{k<N_\epsilon\}}$ is also determined by $\mathcal{A}_{k-1}^M$. Finally,

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}\mathbb{1}_{\mathcal{M}_k}\right] = \mathbb{E}\left[\sum_{k=0}^{\infty} \mathbb{1}_{\{k<N_\epsilon\}}\mathbb{1}_{W_k}\mathbb{1}_{\mathcal{M}_k}\right] \geq p_*\mathbb{E}\left[\sum_{k=0}^{\infty} \mathbb{1}_{\{k<N_\epsilon\}}\mathbb{1}_{W_k}\right] = p_*\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{W_k}\right].$$

The second inequality is proved analogously. □

5.2.2 Bounding the expected number of steps with $\Sigma_k \geq \overline{\sigma}$

In this section we derive an upper bound for the expected number of steps in the process generated by the SARC-IGDH algorithm with $\Sigma_k \geq \overline{\sigma}$.

Given $l \in \{0, ..., N_\epsilon - 1\}$, for all $0 \leq k \leq l$, let us define the event

$$\Lambda_k = \{\text{iteration } k \text{ is such that } \Sigma_k < \overline{\sigma}\} \tag{5.44}$$

and let

$$N_{\Lambda^c} \overset{\text{def}}{=} \sum_{k=0}^{N_\epsilon-1}(1-\mathbb{1}_{\Lambda_k}), \qquad N_\Lambda \overset{\text{def}}{=} \sum_{k=0}^{N_\epsilon-1}\mathbb{1}_{\Lambda_k}, \tag{5.45}$$

be the number of steps, in the stochastic process induced by the SARC-IGDH algorithm, with $\Sigma_k \geq \overline{\sigma}$ and $\Sigma_k < \overline{\sigma}$, before N_ϵ is met, respectively.

In what follows we consider the validity of Assumption 1.2.1, Assumption 5.1.1, Assumption 5.1.2 and the following assumption on Σ_0.

By referring to Lemma 48 and some additional lemmas from [44], we can first obtain an upper bound on $\mathbb{E}[N_{\Lambda^c}]$.

In particular, rearranging [44, Lemma 2.2], given a generic iteration l, we derive a bound on the number of iterations that are successful or unsuccessful in the sense of Step 4, with $\Sigma_k \geq \overline{\sigma}$, in terms of the overall number of iterations $l+1$. At this regard, we underline that in case of unsuccessful iterations at Step 4, the value of Σ_k is not modified and such an iteration occurs at most once between two successful iterations (not necessary consecutive) with the first one having the norm of the step not smaller than one or once before the first successful iteration of the process (since flag is initially 1). In fact, a failure in the sense of Step 4 may occur only if $\text{flag}{=}1$; except for the first step, 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 the SARC-IGDH algorithm), except for the first iteration. If the case $\text{flag} = 1$ and $\|s_k\| < 1$ occurs, then flag is set to zero, preventing a failure in Step 4 at the subsequent iteration and it is not further changed until a subsequent successful iteration.

Lemma 49. Given $l \in \{0, ..., N_\epsilon - 1\}$, for all realisations of the SARC-IGDH algorithm,

$$\sum_{k=0}^{l} (1 - \mathbb{1}_{\Lambda_k})\, \mathbb{1}_{\mathcal{S}_k \cup \mathcal{U}_{k,2}} \leq \frac{2}{3}(l+1).$$

Proof. Each iteration k such that $(1 - \mathbb{1}_{\Lambda_k})\mathbb{1}_{\mathcal{S}_k \cup \mathcal{U}_{k,2}} = 1$ is an iteration with $\Sigma_k \geq \overline{\sigma}$ that can be a successful iteration, leading to $\Sigma_{k+1} = \max[\sigma_{\min}, \frac{1}{\gamma}\Sigma_k]$ (Σ_k is decreased), or an unsuccessful iteration in the sense of Step 4. In the latter case, Σ_k is left unchanged ($\Sigma_{k+1} = \Sigma_k$). Moreover, Σ_k in successful/unsuccessful in the sense of Step 5 iterations is decreased/increased by the same factor γ. More in depth, since $\Sigma_0 < \overline{\sigma}$ (recall (5.34) and the fact that $\Sigma_0 = \sigma_0$), we have two possible scenarios. In the first one $\Sigma_k < \overline{\sigma}$, $k = 0, \ldots, l$, and the thesis obviously follows. In the second scenario there exists at least one index k such that $\Sigma_k \geq \overline{\sigma}$ and at least one unsuccessful iteration $j \in \{0, \ldots, k-1\}$ at which Σ_k has been increased by the factor γ. In case $\mathbb{1}_{\mathcal{U}_{k,2}} = 1$, Σ_k is left unchanged, flag is set to 0 and $\mathbb{1}_{\mathcal{U}_{k+1,2}} = 0$. Then, at any iteration j such that $\mathbb{1}_{\mathcal{U}_{j,1}} = 1$ corresponds at most one successful iteration and one unsuccessful iteration in the sense of Step 4, with $\Sigma_k \geq \overline{\sigma}$, and this yields the thesis. □

We note that in the stochastic ARC method in [44] each iteration can be successful or unsuccessful according to the satisfaction of the decrease condition $\rho_k \geq \eta$. On the contrary, in the SARC-IGDH algorithm also failures in Step 4 may occur and this yields the bound $2(l+1)/3$ in Lemma 49, while the corresponding bound in [44] is $(l+1)/2$.

As in [44], we note that $\widehat{\sigma}(\mathbb{1}_{\Lambda_k}) \subseteq \mathcal{A}_{k-1}^M$, that is the variable Λ_k is fully determined by the first $k-1$ iterations of the SARC-IGDH algorithm. Then, setting $l = N_\epsilon - 1$, we can rely on Lemma 48 (with $W_k = \Lambda_k^c$) to deduce that

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon - 1} (1 - \mathbb{1}_{\Lambda_k})\mathbb{1}_{\mathcal{M}_k}\right] \geq p_* \mathbb{E}\left[\sum_{k=0}^{N_\epsilon - 1} (1 - \mathbb{1}_{\Lambda_k})\right]. \tag{5.46}$$

Considering the bound in Lemma 49, together with the fact that Lemma 46 and the mechanism of Step 4 in the SARC-IGDH algorithm ensure that each iteration k such that $\mathbb{1}_{\mathcal{M}_k} = 1$ with $\sigma_k \geq \overline{\sigma}$ can be successful or unsuccessful in the sense of Step 4 (i.e., $\mathbb{1}_{\mathcal{S}_k \cup \mathcal{U}_{k,2}} = 1$), we have that

$$\sum_{k=0}^{N_\epsilon - 1} (1 - \mathbb{1}_{\Lambda_k})\mathbb{1}_{\mathcal{M}_k} \leq \sum_{k=0}^{N_\epsilon - 1} (1 - \mathbb{1}_{\Lambda_k})\mathbb{1}_{\mathcal{S}_k \cup \mathcal{U}_{k,2}} \leq \frac{2}{3} N_\epsilon.$$

Taking expectation in the above inequality and recalling the definition of N_{Λ^c} in (5.45), from (5.46) we conclude that:

$$\mathbb{E}[N_{\Lambda^c}] \leq \frac{2}{3p_*}\mathbb{E}[N_\epsilon]. \tag{5.47}$$

The remaining bound for $\mathbb{E}\left[N_\Lambda\right]$ will be derived in the next section.

5.2.3 Bounding the expected number of steps with $\Sigma_k < \overline{\sigma}$

Let us now obtain an upper bound for $\mathbb{E}[N_\Lambda]$, with N_Λ defined in (5.45). To this purpose, the following additional definitions are needed.

Definition 5.2.1. *Let $\mathcal{U}_{k,1}$, $\mathcal{U}_{k,2}$ and $\mathcal{S}_k$ be as defined in* (5.19). *With reference to the process* (5.13) *generated by the SARC-IGDH algorithm let us define:*

- *the event $\overline{\Lambda}_k = \{$iteration k is such that $\Sigma_k \leq \overline{\sigma}\}$, i.e. $\overline{\Lambda}_k$ is the closure of Λ_k.*
- *$N_I = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}(1 - \mathbb{1}_{\mathcal{M}_k})$: number of inaccurate iterations with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_A = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}$: number of accurate iterations with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_{AS} = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\mathcal{S}_k}$: number of accurate successful iterations with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_{AH} = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\mathcal{U}_{k,2}}$: number of accurate unsuccessful iterations, in the sense of Step 4, with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_{AU} = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\Lambda_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\mathcal{U}_{k,1}}$: number of accurate unsuccessful iterations, in the sense of Step 5, with $\Sigma_k < \overline{\sigma}$;*
- *$N_{IS} = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}(1 - \mathbb{1}_{\mathcal{M}_k})\mathbb{1}_{\mathcal{S}_k}$: number of inaccurate successful iterations, with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_S = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{S}_k}$: number of successful iterations, with $\Sigma_k \leq \overline{\sigma}$;*
- *$N_H = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\mathcal{U}_{k,2}}$: number of unsuccessful iterations in the sense of Step 4;*
- *$N_U = \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\Lambda_k}\mathbb{1}_{\mathcal{U}_{k,1}}$: number of unsuccessful iterations, in the sense of Step 5, with $\Sigma_k < \overline{\sigma}$.*

It is worth noting that an upper bound on $\mathbb{E}[N_\Lambda]$ is given, once an upper bound on $\mathbb{E}[N_I] + \mathbb{E}[N_A]$ is provided, since

$$\mathbb{E}[N_\Lambda] \leq \mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\right] = \mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}(1 - \mathbb{1}_{\mathcal{M}_k}) + \sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\right] = \mathbb{E}[N_I] + \mathbb{E}[N_A]. \quad (5.48)$$

Following [44], to bound $\mathbb{E}[N_I]$ we can still refer to the central Lemma 48 (with $W_k = \overline{\Lambda}_k$), of which the result stated below is a direct consequence.

Lemma 50. [44, Lemma 2.6] With reference to the stochastic process (5.13) generated by the SARC-IGDH algorithm and the definitions of N_I, N_A in Definition 5.2.1,

$$\mathbb{E}[N_I] \leq \frac{1 - p_*}{p_*}\mathbb{E}[N_A]. \quad (5.49)$$

Proof. [44, Lemma 2.6] By applying subsequently both inequalities in Lemma 48 with $W_k = \overline{\Lambda}_k$, we obtain

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\right] \geq p_*\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\right]$$

and

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}(1-\mathbb{1}_{\mathcal{M}_k})\right] \le (1-p_*)\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\right],$$

giving

$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}(1-\mathbb{1}_{\mathcal{M}_k})\right] \le \frac{1-p_*}{p_*}\mathbb{E}\left[\sum_{k=0}^{N_\epsilon-1} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\right],$$

which is equivalent to (5.49) because of the definitions of N_I and N_A in Definition 5.2.1. □

Concerning the upper bound for $\mathbb{E}[N_A]$, we observe that

$$\mathbb{E}[N_A] = \mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_{AU}] \le \mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_U]. \tag{5.50}$$

In the following Lemma we provide upper bounds for N_{AS} and N_{AH}, given in Definition 5.2.1.

Lemma 51. Let Assumption 1.2.1 and Assumption 5.1.1 hold. Assume also that the stopping criterion (5.9) is used to perform each Step 3 of the SARC-IGDH algorithm. With reference to the stochastic process (5.13) induced by the SARC-IGDH algorithm, there exists $\kappa_s > 0$ such that

$$N_{AS} \le \kappa_s(f_0 - f_{low})\epsilon_1^{-3/2} + 1. \tag{5.51}$$

Moreover, in case the stopping criterion (5.10) is used in Step 3, (5.51) still holds provided that there exists $L_g > 0$ such that (5.38) is satisfied for all $x, y \in \mathbb{R}^n$.
Finally, let Assumption 1.2.1 hold, independently of the stopping criterion used to perform Step 3, there exists $\kappa_u > 0$ such that

$$N_{AH} \le \kappa_u(f_0 - f_{low}). \tag{5.52}$$

Proof. Taking into account that (5.33) holds for each realisation of the SARC-IGDH algorithm, (5.37) is valid for each realisation of the SARC-IGDH algorithm with $\mathbb{1}_{\mathcal{M}_k}(1-\mathbb{1}_{\mathcal{U}_{k,2}}) = 1$, recalling that $f(X_k) = f(X_{k+1})$ for all $k \in \mathcal{U}_{k,1} \cup \mathcal{U}_{k,2}$ and setting $f_0 \overset{\text{def}}{=} f(X_0)$, it follows:

$$\begin{aligned}
f_0 - f_{low} &\ge f_0 - f(X_{N_\epsilon}) = \sum_{k=0}^{N_\epsilon-1}(f(X_k) - f(X_{k+1}))\mathbb{1}_{\mathcal{S}_k} \ge \sum_{k=0}^{N_\epsilon-1} \overbrace{\eta\frac{\sigma_{\min}}{3}\|S_k\|^3}^{>0}\mathbb{1}_{\mathcal{S}_k} \\
&\ge \sum_{k=0}^{N_\epsilon-2} \eta\frac{\sigma_{\min}}{3}\|S_k\|^3\mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k} \ge \sum_{k=0}^{N_\epsilon-2} \eta\frac{\sigma_{\min}}{3}\zeta_k^{3/2}\|\nabla f(X_{k+1})\|^{3/2}\mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k} \\
&\ge \sum_{k=0}^{N_\epsilon-2} \eta\frac{\sigma_{\min}}{3}\zeta^{3/2}\|\nabla f(X_{k+1})\|^{3/2}\mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\overline{\Lambda}_k} \\
&\ge (N_{AS}-1)\kappa_s^{-1}\epsilon^{3/2},
\end{aligned}$$

in which ζ_k is defined in (5.41), when s_k satisfies (5.9), and in (5.43), when s_k satisfies (5.10), and

$$\kappa_s^{-1} \overset{\text{def}}{=} \eta\frac{\sigma_{\min}}{3}\zeta^{3/2}, \tag{5.53}$$

where

$$\zeta \overset{\text{def}}{=} \frac{1}{\delta + \max[c, \alpha(\kappa_B + \overline{\sigma})] + L/2 + \theta + \overline{\sigma}} > 0,$$

in case (5.9) is used and

$$\zeta \overset{\text{def}}{=} \frac{1-\theta}{(1+\theta)\delta + \max[c, \alpha(\kappa_B + \overline{\sigma})] + L/2 + \theta L_g + \overline{\sigma}} > 0,$$

whenever (5.10) is adopted. Hence, (5.51) holds.
Moreover, an upper bound for N_{AH} can be obtained taking into account that, as already noticed, an iteration $k \geq 1$ in the process such that $\mathbb{1}_{\mathcal{U}_{k,2}} = 1$ occurs at most once between two successful iterations with the first one having the norm of the trial step not smaller than 1, plus at most once before the first successful iteration in the process (since in the SARC-IGDH algorithm flag is initialised at 1). Therefore, by means of (5.33),

$$f_0 - f_{low} \geq f_0 - f(X_{N_\epsilon}) = \sum_{k=0}^{N_\epsilon - 1} (f(X_k) - f(X_{k+1}))\mathbb{1}_{\mathcal{S}_k} \geq \sum_{\substack{k=0 \\ \|S_k\| \geq 1}}^{N_\epsilon - 1} (f(X_k) - f(X_k + S_k))\mathbb{1}_{\mathcal{S}_k}$$

$$\geq \eta \frac{\sigma_{\min}}{3} \sum_{\substack{k=0 \\ \|S_k\| \geq 1}}^{N_\epsilon - 1} \mathbb{1}_{\mathcal{S}_k} \|S_k\|^3 \geq \kappa_u^{-1} N_H,$$

where we set $\kappa_u^{-1} \overset{\text{def}}{=} \eta\sigma_{\min}/3$ and N_H denotes (see Definition 5.2.1) the number of unsuccessful iterations in the sense of Step 4. Then, since $N_H \geq N_{AH}$, (5.52) follows.

□

An upper bound for N_U can be still derived using [44, Lemma 2.5].

Lemma 52. [44, Lemma 2.5] For any $l \in \{0, ..., N_\epsilon - 1\}$ and for all realisations of the SARC-IGDH algorithm, we have that

$$\sum_{k=0}^{l} \mathbb{1}_{\Lambda_k} \mathbb{1}_{\mathcal{U}_{k,1}} \leq \sum_{k=0}^{l} \mathbb{1}_{\overline{\Lambda}_k} \mathbb{1}_{\mathcal{S}_k} + \left\lceil \log_\gamma \left(\frac{\overline{\sigma}}{\sigma_0} \right) \right\rceil.$$

Proof. The proof of [44, Lemma 2.5] can still be applied since the process induced by the SARC-IGDH algorithm ensures that Σ_k is decreased by a factor γ on successful steps, increased by the same factor on unsuccessful ones in the sense of Step 5 and *left unchanged* if an unsuccessful iteration in the sense of Step 4 occurs. Therefore, the total number of iterations at which $\Sigma_k < \overline{\sigma}$ and Σ_k is increased is bounded by the total number of iterations where $\Sigma_k \leq \overline{\sigma}$ is decreased plus the number of iterations needed to increase Σ_k from its initial value Σ_0 to $\overline{\sigma}$, given by $\left\lceil \log_\gamma \left(\frac{\overline{\sigma}}{\sigma_0} \right) \right\rceil$. □

Consequently, considering $l = N_\epsilon - 1$ and Definition 5.2.1,

$$N_U \leq N_S + \left\lceil \log_\gamma \left(\frac{\overline{\sigma}}{\sigma_0} \right) \right\rceil = N_{AS} + N_{IS} + \left\lceil \log_\gamma \left(\frac{\overline{\sigma}}{\sigma_0} \right) \right\rceil. \tag{5.54}$$

We underline that the right-hand side in (5.54) involves N_{IS}, that has not been bounded yet. To this aim we can proceed as in [44], obtaining that

$$\mathbb{E}[N_{IS}] \leq \frac{1-p_*}{2p_* - 1} \left(2\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \left\lceil \log_\gamma \left(\frac{\overline{\sigma}}{\sigma_0} \right) \right\rceil \right). \tag{5.55}$$

In fact, recalling the definition of N_{IS} and (5.50), the inequality (5.49) implies that

$$\mathbb{E}[N_{IS}] \le \mathbb{E}[N_I] \le \frac{1-p_*}{p_*}\mathbb{E}[N_A] \le \frac{1-p_*}{p_*}\left(\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_U]\right). \tag{5.56}$$

Indeed, taking expectation in (5.54) and plugging it into (5.56),

$$\mathbb{E}[N_{IS}] \le \frac{1-p_*}{p_*}\left(2\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_{IS}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil\right),$$

which yields (5.55).

The upper bound on $\mathbb{E}[N_A]$ then follows:

$$\begin{aligned}
\mathbb{E}[N_A] &\le \mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_U] \le 2\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \mathbb{E}[N_{IS}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil \\
&\le \left(\frac{1-p_*}{2p_*-1}+1\right)\left(2\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil\right) \\
&= \frac{p_*}{2p_*-1}\left(2\mathbb{E}[N_{AS}] + \mathbb{E}[N_{AH}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil\right) \\
&\le \frac{p_*}{2p_*-1}\left[(f_0 - f_{low})\left(2\kappa_s\epsilon^{-3/2} + \kappa_u\right) + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right],
\end{aligned} \tag{5.57}$$

in which we have used (5.50), (5.52), (5.51), (5.54) and (5.55).

Therefore, recalling (5.48) and (5.49), we obtain that

$$\mathbb{E}\big[N_\Lambda\big] \le \frac{1}{p_*}\mathbb{E}[N_A] \le \frac{1}{2p_*-1}\left[(f_0 - f_{low})\left(2\kappa_s\epsilon_1^{-3/2} + \kappa_u\right) + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right], \tag{5.58}$$

where the last inequality follows from (5.57).

We are now in the position to state our final result, providing the complexity of the stochastic method associated with the SARC-IGDH algorithm, in accordance with the complexity bounds given by the deterministic analysis of an ARC framework with exact [22] and inexact [41, 38, 37, 9 10] function and/or derivatives evaluations.

Theorem 53. Let Assumptions 1.2.1 and Assumption 5.1.1 hold. Assume that Assumption 5.1.2 holds and that the stopping criterion (5.9) is used to perform each Step 3 of the SARC-IGDH algorithm. Then, the hitting time N_ϵ for the stochastic process generated by the SARC-IGDH algorithm satisfies

$$\mathbb{E}[N_\epsilon] \le \frac{3p_*}{(3p_*-2)(2p_*-1)}\left[(f_0 - f_{low})\left(2\kappa_s\epsilon_1^{-3/2} + \kappa_u\right) + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right]. \tag{5.59}$$

Moreover, in case the stopping criterion (5.10) is used to perform Step 3, (5.59) still holds, provided that there exists $L_g > 0$ such that (5.38) is satisfied for all $x, y \in \mathbb{R}^n$.

Proof. By definition (see (5.45)), $\mathbb{E}[N_\epsilon] = \mathbb{E}\big[N_{\Lambda^c}\big] + \mathbb{E}\big[N_\Lambda\big]$. Thus, considering (5.47),

$$\mathbb{E}[N_\epsilon] \le \frac{2}{3p_*}\mathbb{E}[N_\epsilon] + \mathbb{E}\big[N_\Lambda\big],$$

and, hence, by (5.58),

$$\mathbb{E}[N_\epsilon] \le \frac{3p_*}{3p_*-2}\mathbb{E}\big[N_\Lambda\big] = \frac{3p_*}{(3p_*-2)(2p_*-1)}\left[(f_0 - f_{low})\left(2\kappa_s\epsilon_1^{-3/2} + \kappa_u\right) + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right],$$

which concludes the proof. □

5.3 Subsampling scheme for finite-sum minimisation

Similarly to Section 3.5, we now consider the solution of large-scale instances of finite-sum minimisation problems of the form (1.108). In this context, the approximations $\overline{\nabla f}(x_k)$ and $\overline{\nabla^2 f}(x_k)$ to the gradient and the Hessian used at Step 1 and Step 2 of the SARC-IGDH algorithm, respectively, are obtained via (2.5)–(2.6) by using random and uniform subsets selected as prescribed by (2.11)–(2.12). The aim is still to achieve the probabilistic accuracy requirements in (2.1) ($j \in \{1,2\}$), in which the absolute accuracies $\tau_{1,k}$ and $\tau_{2,k}$, used for gradient and Hessian approximations at iteration k, respectively, correspond to the right-hand sides in (5.14) and (5.15).

We underline that assuming (2.7) in Theorem 19 is crucial to derive the lower bounds (2.11)–(2.12). A way of enforcing (2.7) for the cases of interest ($j \in \{1,2\}$) is assuming the existence of $\kappa_g > 0$ and $\kappa_B > 0$ such that $\kappa_{\varphi,1}(x) \le \kappa_g$ and $\kappa_{\varphi,2}(x) \le \kappa_B$, for any $x \in \mathbb{R}^n$. We will consider such an assumption in the remainder of this chapter.

Since the subsampling procedures used at iteration k to get $\mathcal{G}_k$ and $\mathcal{H}_k$ in (2.11)–(2.12) are independent, it follows that when $\{\tau_{j,k}\}_{j=1}^2$ are chosen as the right-hand sides in (5.14) and (5.15), respectively, the builded model (5.6) is p_*-probabilistically sufficiently accurate with[*] $p_* = (1-t)^2$. Therefore, a practical version of the SARC-IGDH algorithm is for instance given by adding a suitable termination criterion and modifying the first three steps of the SARC-IGDH algorithm as reported in Algorithm 13 below.

Algorithm 13 Modified Steps 0–2 of the SARC-IGDH algorithm.

Step 0: Initialisation. An initial point $x_0 \in \mathbb{R}^n$ and an initial regularisation parameter $\sigma_0 > 0$ are given, as well as an accuracy level $\epsilon_1 \in (0,1)$. The constants θ, α, η, γ, $\sigma_{\min}$, κ, τ_0, κ_τ and c are also given such that

$$0 < \theta, \kappa_\tau < 1, \quad 0 \le \alpha < \tfrac{2}{3}, \quad \sigma_{\min} \in (0, \sigma_0],$$
$$0 < \eta < \tfrac{2-3\alpha}{2}, \quad \gamma > 1, \quad \kappa \in [0,1), \quad \tau_0 > 0, \quad c > 0.$$

Compute $f(x_0)$ and set $k = 0$, $\text{flag} = 1$.

Step 1: Gradient approximation. Set $i = 0$ and initialise $\tau_{1,k}^{(i)} = \tau_0$.
Do:

1.1 compute $\overline{\nabla f}(x_k)$ such that (2.5) and (2.11) are satisfied with $\tau_{1,k} = \tau_{1,k}^{(i)}$;

1.2 if $\tau_{1,k}^{(i)} \le \kappa(1-\theta)^2 \left(\frac{\|\overline{\nabla f}(x_k)\|}{\sigma_k}\right)^2$, go to Step 2;
else, set $\tau_{1,k}^{(i+1)} = \kappa_\tau \tau_{1,k}^{(i)}$, increment i by one and go to Step 1.1;

Step 2: Hessian approximation (model costruction). If $\text{flag} = 1$, set $c_k = c$; else, set $c_k = \alpha(1-\theta)\|\overline{\nabla f}(x_k)\|$.
Compute $\overline{\nabla^2 f}(x_k)$ using (2.6) and (2.12) with $\tau_{2,k} = c_k$ and form the model $m_k(s)$ defined in (5.6).

[*]Of course, different failure probabilities t_j, $j \in \{1,2\}$, can be considered in (2.1), resulting in replacing t by t_j in (2.11)–(2.12).

Concerning the gradient estimate, the scheme computes (Step 1) an approximation $\overline{\nabla f}(x_k)$ satisfying the accuracy criterion

$$\|\overline{\nabla f}(x_k) - \nabla f(x_k)\| \leq \kappa(1-\theta)^2 \left(\frac{\|\overline{\nabla f}(x_k)\|}{\sigma_k}\right)^2, \tag{5.60}$$

which is independent of the step computation and based on the knowable quantities κ, θ and σ_k. This is done by reducing the accuracy $\tau_{1,k}^{(i)}$ and repeating the inner loop at Step 1, until the fulfillment of the inequality at Step 1.2.

We underline that the condition (5.60) is guaranteed by the algorithm, since (2.11) is a continuous and increasing function with respect to $\tau_{j,k}$, for fixed k, t and N; hence, there exists a sufficiently small $\overline{\tau}_{1,k}$ such that the right-hand side term in (2.11) will reach, in the worst-case, the full sample size N, yielding $\overline{\nabla f}(x_k) = \nabla f(x_k)$. Moreover, if the stopping criterion $\|\overline{\nabla f}(x_k)\| \leq \epsilon_1$ is used, the loop is ensured to terminate also whenever the predicted accuracy requirement $\tau_{1,k}^{(i)}$ becomes smaller than $\kappa(\frac{1-\theta}{\sigma_k})^2\epsilon_1^2$. On the other hand, we expect in practice to use a small number of samples in the early stage of the iterative process, when the norm of the approximated gradient is not yet small.

To summarise, if (without loss of generality) we assume that $\overline{\tau}_{1,k} \geq \hat{\tau}$ at each iteration k, we conclude that, in the worst case, Step 1 will lead to at most $\lfloor \log(\hat{\tau})/\log(\kappa_\tau \tau_0) \rfloor + 1$ computations of $\overline{\nabla f}(x_k)$. The Hessian approximation $\overline{\nabla^2 f}(x_k)$ is, instead, defined at Step 2 and its computation is based on the reliable (adaptive) value of c_k. We remark that at iteration k we have that:

- $\overline{\nabla^2 f}(x_k)$ is computed only once, irrespectively of the approximate gradient computation considered at Step 1;
- a finite loop is considered at Step 1 to obtain a gradient approximation satisfying (5.60), where the right-hand side is independent of the step length $\|s_k\|$, thou implying (5.20)–(5.21). Hence, the gradient approximation is fully determined at the end of Step 1 and further recomputations due to the step calculation (see the SARC-IGDH algorithm, Step 3) are not required.

We conclude this section by noticing that each iteration k of the SARC-IGDH algorithm with the modified steps introduced in Algorithm 13 can indeed be seen as an iteration of the SARC-IGDH algorithm where the sequence of random models $\{M_k\}_{k\geq 0}$ is p_*-probabilistically sufficiently accurate in the sense of Definition 5.1.1, with $p_* = (1-t)^2$, and an iteration of the ARC-DH algorithm, when $\kappa = 0$ is considered in (5.14) (exact gradient evaluations).

5.4 Chapter conclusion

We have proposed the stochastic iteration complexity analysis of the process generated by an ARC algorithm for solving unconstrained, nonconvex, optimisation problems under inexact derivatives information.

The algorithm is an extension of the ARC-DH algorithm in Chapter 3, since it employs approximated evaluations of the gradient with the main feature of maintaining the dynamic rule for building Hessian approximations, introduced and numerically tested in [9].

This kind of accuracy requirement is always reliable and computable when an approximation of the exact Hessian is needed by the scheme and, in contrast to other strategies such that the one in [44], does not require the inclusion of additional inner loops to be satisfied.

With respect to the framework in Chapter 3, where in the finite-sum setting optimal complexity is restored with high probability, we have here provided properties of the method when the adaptive accuracy requirements of the derivatives involved in the model definition are not accomplished, with a view to search for the number of expected steps that the process takes to reach the prescribed accuracy level.

The stochastic analysis is thus performed exploiting the theoretical framework given in [44], showing that the expected iteration complexity bound matches the worst-case optimal complexity of the ARC framework. The possible lack of accuracy of the model has "just" the effect of scaling the optimal complexity we would derive from the deterministic analysis of the framework (see, e.g., Theorem 23), by a factor which depends on the probability p_* of the model being sufficiently accurate.

Chapter 6

Stochastic Analysis of an Adaptive Regularisation Methods with Inexact Function and Derivatives Evaluations

This chapter investigates the stochastic evaluation complexity of the adaptive regularisation algorithm ARqpDA for computing approximate local minimisers of the possibly constrained minimisation problem (4.3), whose main characteristic is the following.

- The values of the objective function j-th derivatives $\nabla^j f$ are subject to random noise and can only be computed inexactly, this is to say that only an approximation $\overline{\nabla^j f}$, $j \in \{1, 2\}$, can be calculated. Inexact values of the objective function are also allowed, but are assumed to follow a "dynamic accuracy" framework in which the accuracy of these evaluations is deterministically controlled in response to the inexact values of the derivatives.

The analysis here presented, in which the objective function values can be computed within a prescribed accuracy allowing randomly inexact evaluations of its derivatives, enhances the study in [9, 10, 124] for adaptive regularisation algorithm, where it is carried out under the assumption that the estimates are sufficiently accurate at every iteration. Our choice of imposing the dynamic accuracy framework for the objective function complements that of [6, 44], allowing for more inaccuracy, but in a deterministic context. A truly stochastic approach for adaptive regularisation algorithms with random function evaluations remains a challenging open problem, as discussed in [55, page 41], but considering dynamic accuracy on the objective function is nevertheless a realistic request in applications such as those where the objective function value is approximated by using smoothing operators and the derivatives are estimated by randomised finite-differences [17, 16, 86, 93].

As in Chapter 4 (see, also, [10, 35, 34]), we propose a regularisation algorithm for the solution of the problem, which is based on (regularised) polynomial models that can be extended to the case of arbitrary degree (see [10]). This allows us to seek for first and second-order critical points, but also apply for critical points of arbitrary order. In this respect we improve upon the algorithms with stochastic models, such as [6, 10, 23, 35].

We establish, in expectation, sharp worst-case bounds on the evaluation complexity of computing these (possibly high-order) approximate critical points, depending on the order and on the degree of the polynomial model used. These bounds correspond in order to the best known bounds for regularisation algorithms using exact evaluations. That said, all the main results are presented, according to Chapter 4, for the case $1 \le q \le p \le 2$ (model with up to cubic regularisation for first or second-order optimality), for a simpler presentation and in light of a more practical application. These results are obtained by building on the probabilistic framework of [44], and by merging approximation results in [10] with techniques of [34], indeed proceeding as in Section 5.2.

The outline of the chapter is as follows. Section 6.1 introduces the regularisation algorithm and the associated stochastic assumptions. Its evaluation complexity is then studied in Section 6.2.

6.1 A stochastic regularisation algorithm with inexact evaluations: the SARqp algorithm

We first make our framework more formal by describing our assumptions on problem (4.3).

Assumption 6.1.1. *With reference to problem* (1.1)*, given $p \in \{1, 2\}$, the objective function f is assumed to be:*

(i) bounded below in $\mathcal{X}$, that is there exists a constant $f_{\rm low}$ such that $f(x) \ge f_{\rm low}$ for all $x \in \mathcal{X}$;

(ii) p-times continuously differentiable in a convex neighbourhood of $\mathcal{X}$; moreover, its j-th order derivative is Lipschitz continuous for $j \in \{1, \dots, p\}$ in the sense that there exist constants $L_{f,j} \ge 0$ such that, for all $j \in \{1, \dots, p\}$ and all x, y in that neighbourhood,

$$\|\nabla_x^j f(x) - \nabla_x^j f(y)\| \le L_{f,j} \|x - y\|. \tag{6.1}$$

Note that, if $\mathcal{X}$ is convex, then Assumption 6.1.1(ii) can be restricted to hold in an open neighbourhood of $\mathcal{X}$.

Under Assumption 6.1.1(ii) , the p-th order Taylor's series $T_p^f(x, s)$ of f taken at a point x and evaluated for a step s is well-defined and given by (4.5), so we still refer to the Taylor's increment $\Delta T_p^f(x, s)$ defined by (4.6).

We will also rely on Lemma 28 (with $L = L_{f,p}$) from [35, Lemma 2.1], which is an important consequence of Assumption 6.1.1(ii).

In this context, given $1 \le q \le p$, the "accuracy requests" $\epsilon = (\epsilon_1, \dots, \epsilon_q)$ and "optimality radiuses" $\delta = (\delta_1, \dots, \delta_q)$, with

$$\epsilon_j \in (0, 1] \text{ and } \delta_j \in (0, 1] \text{ for } j \in \{1, \dots, q\},$$

we say that $x \in \mathcal{X}$ is a *strong* q-th order (ϵ, δ)-approximate minimiser for problem (4.3) if

$$\phi_{f,j}^{\delta_j}(x) \le \epsilon_j \frac{\delta_j^j}{j!}, \quad \text{for } j \in \{1, \dots, q\}, \tag{6.2}$$

where $\phi_{f,j}^{\delta_j}(x)$ is defined as in (4.19). It means that no significant decrease of the Taylor's expansions of degree 1 to q can be obtained in a ball of optimality radius δ_j, within the constraint. We immediately notice that (6.2) is a slightly different requirement compared to the criterion (4.13) adopted in Chapter 4, where the condition $\phi_{f,q}^{\delta}(x) \le \epsilon_\chi \chi_q(\delta)$, with ϵ_χ a scalar in $(0, 1)$, replaces (6.2).

The optimality measure $\phi_{f,j}^{\delta}(x)$, $\delta > 0$, is a nonnegative (continuous) function that can be used as necessary condition to measure closeness to stationary points of order $q \in \{1, 2\}$ (see, e.g., [34]).

In this respect we observe that:

- for $q = 1$ the condition (6.2) is equivalent to (4.13), since it reduces, by (4.14), to
$$\phi^{\delta_1}_{f,1}(x) \le \epsilon_1\delta_1 = \epsilon_1\chi_1(\delta_1);$$
- for $q = 2$, $\|\nabla f(x)\| = 0$ and $\lambda_{\min}(\nabla^2 f(x)) \ge \epsilon_2$ further imply $\phi^{\delta_2}_{f,2}(x) \le \epsilon_2\delta_2^2/2$.

Moreover, in analogy with (4.10), the two points below show that (6.2) reduces to the known first and second-order optimality conditions for unconstrained minimisation when $\epsilon_1 = \epsilon_2 = 0$ and $\mathcal{X} = \mathbb{R}^n$ are considered.

- Assuming that $q = 1$, we have from (4.10) and (6.2) with $j = 1$ that
$$\phi^{\delta_1}_{f,1}(x) = \|\nabla_x f(x)\|\delta_1 = 0,$$
for any $\delta_1 \in (0, 1]$ (as in (4.11)).
- If $q = 2$, (4.10) and (6.2) further imply that
$$\phi^{\delta_2}_{f,2}(x) = \operatorname*{globmax}_{\|d\|\le\delta_2}\left(-\frac{1}{2}d^\top\nabla^2 f(x)d\right) = \frac{1}{2}\max\left[0, -\lambda_{\min}(\nabla^2 f(x))\right]\delta_2^2,$$
which is the same as requiring the semi-positive definitiveness of $\nabla^2 f(x)$.

The requirement (6.2) has firstly been introduced in [34] and it is extended here to the constrained case. When thinking about generalisations in which $q > 2$, it is an extension of the notion of (*weak*) approximate minimisers discussed in [10, 35] and given in (4.13) (if we allow (4.13) to hold also for $q > 2$). That this is weaker than (6.2) for the case $q \ge 2$ is seen if one considers that $\chi_q(\delta) \in [\delta, 2\delta)$ for $q \ge 1$ and $\delta \in [0, 1]$ (see [8, bound (2.8)]). As a consequence, $\chi_q(\delta)$ is typically significantly larger than $\delta_j^j/j!$, for $j \in \{1, \ldots, q\}$ and $q > 1$.

We are now in position to describe our stochastic adaptive regularisation SARqp algorithm, defined for $1 \le q \le p \le 2$, whose purpose is to compute a (strong) q-th order (ϵ, δ)-approximate minimiser for problem (4.3).

The algorithm is basically the stochastic counterpart of the ARqpDA algorithm. Given the vector of accuracies ϵ and $1 \le q \le p \le 2$, we briefly summarise here the little differences with the ARqpDA algorithm.

- At iteration k, the local model takes the form:
$$m_k(s) = -\overline{\Delta T}^f_p(x_k, s) + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1} \stackrel{\rm def}{=} -\sum_{\ell=1}^{p}\frac{1}{\ell!}\overline{\nabla^\ell f}(x)[s]^\ell + \frac{\sigma_k}{(p+1)!}\|s\|^{p+1}, \qquad (6.3)$$
still using inexact derivatives $\overline{\nabla^\ell f}(x)$ in place of the exact ones $\nabla^\ell f(x)$, $\ell \in \{1, ..., p\}$. At variance with (4.1), the above definition of m_k no longer includes $f(x_k)$.
- A step s_k would next be computed by approximately minimising $m_k(s)$, finding a feasible step s_k in the sense that the trial point $x_k + s_k \in \mathcal{X}$ satifies
$$m_k(s_k) \le m_k(0) = 0 \qquad (6.4)$$
and
$$\overline{\phi}^{\delta_{k,j}}_{m_k,j}(s_k) = \operatorname*{globmax}_{x_k+s_k+d\in\mathcal{X}, \|d\|\le\delta_{k,j}} \overline{\Delta T}^{m_k}_j(s_k, d) \le \theta\epsilon_j\frac{\delta^j_{k,j}}{j!}, \qquad (6.5)$$
for $j \in \{1, \ldots, q\}$, $1 \le q \le p \le 2$, some $\theta \in (0, \frac{1}{2})$ and $\delta_k \in (0, 1]^q$. We recall that, for $j \in \{1, 2\}$, $\overline{T}^{m_k}_j(s_k, d)$ and $\overline{\Delta T}^{m_k}_j(s_k, d)$ take the form (4.30)–(4.31) in which the

model definition (6.3) is considered and, as usual, the corresponding quantities using exact evaluations $\nabla^j f(x_k)$, $j \in \{0,1,2\}$, are denoted by $T_q^{m_k}(s_k,d)$ and $\Delta T_q^{m_k}(s_k,d)$, $q \in \{1,2\}$. We thus notice that (6.5) replaces the right-hand side criterion (4.24) considered in the ARqpDA algorithm.

- The estimates $\overline{f}(x_k)$, $\overline{f}(x_k+s_k)$ and $\overline{\Delta T}_p^f(x_k,s_k)$ are then used to compute the ratio ρ_k, the value of which decides of the acceptance of the trial point x_k+s_k, and are required to satisfy the accuracy conditions (4.25)–(4.26), with

$$0 < \omega < \min\left[\frac{1-\eta}{3}, \frac{\eta}{2}\right], \quad \eta \in (0,1). \tag{6.6}$$

- The updating rule for the regulariser σ_k follows the scheme at Step 4 of the ARqpDA algorithm, with the main difference of distinguishing just between unsuccessful and very successful iterations, at which σ_k is respectively increased or decreased by the same factor.

The SARqp algorithm is detailed below as Algorithm 14.

Algorithm 14 The SARqp Algorithm, $1 \le q \le p \le 2$.

Step 0: Initialisation. An initial point $x_0 \in \mathcal{X}$ and an initial regularisation parameter $\sigma_0 > 0$ are given, as well as a vector of accuracies $\epsilon \in (0,1]^q$. The constants $\theta \in (0,\frac{1}{2})$, $\eta \in (0,1)$, $\gamma > 1$, $\alpha \in (0,1)$, $0 < \omega < \min\left[\frac{1-\eta}{3}, \frac{\eta}{2}\right]$ and $\sigma_{\min} \in (0,\sigma_0)$ are also given. Set $k=0$.

Step 1: Model construction. Compute approximate derivatives $\{\overline{\nabla_x^\ell f}(x_k)\}_{\ell\in\{1,\dots,p\}}$ and form the model $m_k(s)$ defined in (6.3).

Step 2: Step calculation. Compute a step s_k satisfying $x_k+s_k \in \mathcal{X}$, (6.4), (6.5) for $j \in \{1,\dots,q\}$ and some $\delta_k \in (0,1]^q$. If $\overline{\Delta T}_p^f(x_k,s_k)=0$, go to Step 4.

Step 3: Function estimates computation. Compute the approximations $\overline{f}(x_k)$ and $\overline{f}(x_k+s_k)$ of $f(x_k)$ and $f(x_k+s_k)$, respectively, such that (4.25)–(4.26) are satisfied.

Step 4: Acceptance test. Set

$$\rho_k = \begin{cases} \dfrac{\overline{f}(x_k)-\overline{f}(x_k+s_k)}{\overline{\Delta T}_p^f(x_k,s_k)}, & \text{if } \overline{\Delta T}_p^f(x_k,s_k) > 0,\\ -\infty, & \text{otherwise.}\end{cases} \tag{6.7}$$

If $\rho_k \ge \eta$ (*successful iteration*), then define $x_{k+1} = x_k + s_k$; otherwise (*unsuccessful iteration*), define $x_{k+1} = x_k$.

Step 5: Regularisation parameter update. Set

$$\sigma_{k+1} = \begin{cases} \max\left[\sigma_{\min}, \frac{1}{\gamma}\sigma_k\right], & \text{if } \rho_k \ge \eta,\\ \gamma\sigma_k, & \text{if } \rho_k < \eta.\end{cases} \tag{6.8}$$

Increment k by one and go to Step 1.

We first verify that the algorithm is well-defined.

Lemma 54. A step s_k satisfying (6.4) and (6.5) for $j \in \{1, \dots, q\}$ and some $\delta_k \in (0,1]^q$ always exists.

Proof. The proof is a direct extension of that of [34, Lemma 4.4] using inexact models. For completeness, it is given in the Appendix (Section A.2). □

Comments on SARqp

1. In what follows we assume that, once the model $m_k(s)$ in (6.3) is determined, then the computation of the pair (s_k, δ_k) (and thus of the trial point $x_k + s_k$) is deterministic. Moreover, we assume that the mechanism which ensures (4.25)-(4.26) at Step 3 of the algorithm is also deterministic, so that ρ_k and the fact that iteration k is successful are deterministic outcomes of the realisation of the model.
2. Observe that, because we have chosen m_k to be a model of the local variation in f rather than a model of f itself, $\overline{f}(x_k)$ is not needed (and not computed) at Steps 1 and 2 of the algorithm. This distinguishes the SARqp algorithm from the approaches of [23, 49].

As in the previous chapter, all random quantities are denoted by capital letters, while the use of small letters is reserved for their realisation. In particular, let us denote a random model at iteration k as M_k, while we use the notation m_k for its realisations. Given x_k, the source of randomness in m_k comes from the random approximation of the derivatives. Similarly, the iterates X_k, as well as the regularisation parameters Σ_k and the steps S_k are random variables and x_k, σ_k, s_k denote their realisations. Moreover, δ_k denotes a realisation of the random vector Δ_k arising in (6.5). Hence, the SARqp Algorithm generates a random process

$$\{X_k, S_k, M_k, \Sigma_k, \Delta_k\} \tag{6.9}$$

($X_0 = x_0$ and $\Sigma_0 = \sigma_0$ are deterministic).

6.1.1 The stochastic setting

In view of our last comment, we now make our probabilistic assumptions on the SARqp algorithm explicit.

As in Subsection 5.1.1, our assumption on the past is formalised, for $k \geq 0$, considering $\mathcal{A}^M_{k-1}$ the $\widehat{\sigma}$-algebra induced by the random variables M_0, M_1,..., M_{k-1}, with $\mathcal{A}^M_{-1} = \widehat{\sigma}(x_0)$.

In order to formalise our probabilistic assumptions we need few more definitions. We define, at iteration k of an arbitrary realisation,

$$d_{k,j} = \arg \underset{x_k+s_k+d\in\mathcal{X}, \|d\|\leq\delta_{k,j}}{\text{globmax}} \Delta T_j^{m_k}(s_k, d), \tag{6.10}$$

the argument of the maximum in the definition of $\phi^{\delta_{k,j}}_{m_k,j}(x_k)$, and

$$\overline{d}_{k,j} = \arg \underset{x_k+s_k+d\in\mathcal{X}, \|d\|\leq\delta_{k,j}}{\text{globmax}} \overline{\Delta T}_j^{m_k}(s_k, d), \tag{6.11}$$

that in the definition of $\overline{\phi}^{\delta_{k,j}}_{m_k,j}(s_k)$.

We also define, at the end of Step 2 of iteration k, the events

$$\mathcal{M}_k = \begin{cases} \mathcal{M}_k^{(1)} \cap \bigcap_{j=1}^q \left(\mathcal{M}_{k,j}^{(2)} \cap \mathcal{M}_{k,j}^{(3)}\right), & \text{if } q=1 \text{ and } \mathcal{X} \text{ is convex, or} \\ & \text{if } q=2 \text{ and } \mathcal{X}=\mathbb{R}^n, \\ \mathcal{M}_k^{(1)} \cap \mathcal{M}_k^{(4)} \cap \bigcap_{j=1}^q \left(\mathcal{M}_{k,j}^{(2)} \cap \mathcal{M}_{k,j}^{(3)}\right), & \text{otherwise,} \end{cases} \tag{6.12}$$

with

$$\begin{array}{rcl}\mathcal{M}_k^{(1)} & = & \left\{|\overline{\Delta T}_p^f(X_k,S_k) - \Delta T_p^f(X_k,S_k)| \le \omega \overline{\Delta T}_p^f(X_k,S_k)\right\},\\ \mathcal{M}_{k,j}^{(2)} & = & \{|\overline{\Delta T}_j^{m_k}(S_k,D_{k,j}) - \Delta T_j^{m_k}(S_k,D_{k,j})| \le \omega \overline{\Delta T}_j^{m_k}(S_k,D_{k,j}),\\ \mathcal{M}_{k,j}^{(3)} & = & \{|\overline{\Delta T}_j^{m_k}(S_k,\overline{D}_{k,j}) - \Delta T_j^{m_k}(S_k,\overline{D}_{k,j})| \le \omega \overline{\Delta T}_j^{m_k}(S_k,\overline{D}_{k,j}),\\ \mathcal{M}_k^{(4)} & = & \{\max_{\ell\in\{2,\ldots,p\}} \|\overline{\nabla_x^\ell f}(X_k)\| \le \Theta\},\end{array}$$

for some $\Theta > 0$. Note that Θ is independent of k and does not need to be known explicitly. Moreover, $\mathcal{M}_k^{(4)}$ is not involved in the definition of $\mathcal{M}_k$ if $q=1$ and $\mathcal{X}$ is convex, nor if $q=2$ and the problem is unconstrained. In what follows, we will say that iteration k is *accurate*, if $\mathbb{1}_{\mathcal{M}_k} = 1$, and iteration k is *inaccurate*, if $\mathbb{1}_{\mathcal{M}_k} = 0$.

The conditions defining $\mathcal{M}_k$ may seem abstract at a first sight, but we now motivate them by looking at what kind of accuracy on each derivative $\overline{\nabla_x^\ell f}(x_k)$ ensures that they hold.

Lemma 55. Consider $1 \le q \le p \le 2$. For each $k \ge 0$, we have the following.

1. Let
$$\tau_k \stackrel{\rm def}{=} \max\left[\|S_k\|, \max_{j\in\{1,\ldots,q\}}[\|D_{k,j}\|, \|\overline{D}_{k,j}\|]\right] \tag{6.13}$$
and
$$\overline{\Delta T}_{k,\min} \stackrel{\rm def}{=} \min\left[\overline{\Delta T}_p^f(X_k,S_k), \min_{j\in\{1,\ldots,q\}}\left[\overline{\Delta T}_j^{m_k}(S_k,D_{k,j}), \overline{\Delta T}_j^{m_k}(S_k,\overline{D}_{k,j})\right]\right]. \tag{6.14}$$
Then $\mathcal{M}_k^{(1)}$, $\{\mathcal{M}_{k,j}^{(2)}\}_{j=1}^q$ and $\{\mathcal{M}_{k,j}^{(3)}\}_{j=1}^q$ occur if
$$\|\overline{\nabla_x^\ell f}(X_k) - \nabla_x^\ell f(X_k)\| \le \omega \frac{\overline{\Delta T}_{k,\min}}{6\tau_k^\ell}, \quad \text{for} \quad \ell \in \{1,\ldots,p\}. \tag{6.15}$$

2. Suppose that Assumption 6.1.1(ii) holds. Then $\mathcal{M}_k^{(4)}$ occurs if
$$\|\overline{\nabla_x^\ell f}(X_k) - \nabla_x^\ell f(X_k)\| \le \Theta_0, \quad \text{for} \quad \ell \in \{2,\ldots,p\} \tag{6.16}$$
and some constant $\Theta_0 \ge 0$ independent of k and ℓ.

Proof. Consider the first assertion. That $\mathcal{M}_k^{(1)}$ occurs follows from the inequalities

$$\begin{array}{rcl}|\overline{\Delta T}_p^f(X_k,S_k) - \Delta T_p^f(X_k,S_k)| & \le & \displaystyle\sum_{\ell=1}^p \frac{\|S_k\|^\ell}{\ell!}\|\overline{\nabla_x^\ell f}(X_k) - \nabla_x^\ell f(X_k)\|\\ & \le & \displaystyle\sum_{\ell=1}^p \frac{\tau_k^\ell}{\ell!}\|\overline{\nabla_x^\ell f}(X_k) - \nabla_x^\ell f(X_k)\|\\ & \le & \displaystyle\sum_{\ell=1}^p \frac{\omega}{6\ell!}\overline{\Delta T}_{k,\min}\\ & \le & \displaystyle\sum_{\ell=1}^p \frac{\omega}{6\ell!}\overline{\Delta T}_p^f(X_k,S_k)\\ & \le & \frac{1}{6}\chi_p(1)\,\omega\,\overline{\Delta T}_p^f(X_k,S_k)\\ & < & \omega\,\overline{\Delta T}_p^f(X_k,S_k),\end{array}$$

where we have used (6.13), (6.15), (6.14) and the fact that $\chi_p(1) \le 2$. The verification that

$\{\mathcal{M}_{k,j}^{(2)}\}_{j=1}^{q}$ and $\{\mathcal{M}_{k,j}^{(3)}\}_{j=1}^{q}$ also occur uses a very similar argument, with one additional ingredient: employing the triangle inequality and recalling (6.3), we have that, for all $\ell \in \{1,\ldots,p\}$,

$$\left\|\overline{\nabla_d^\ell T}_j^{m_k}(S_k,0) - \nabla_d^\ell T_j^{m_k}(S_k,0)\right\| \le \sum_{t=\ell}^{p}\left\|\overline{\nabla_x^t f}(X_k) - \nabla_x^t f(X_k)\right\| \frac{\|S_k\|^{t-\ell}}{(t-\ell)!}.$$

Considering now $D = D_{k,j}$ or $D = \overline{D}_{k,j}$ and using the above inequality, (6.13), (6.15), (6.14) and the facts that $\chi_j(1) \le 2$ and $\chi_{p-\ell}(1) \le 2$, we have that

$$\begin{aligned}
|\overline{\Delta T}_j^{m_k}(S_k,D) - \Delta T_j^{m_k}(S_k,D)| &\le \sum_{\ell=1}^{j}\frac{\|D\|^\ell}{\ell!}\|\overline{\nabla_d^\ell T}_j^{m_k}(S_k,0) - \nabla_d^\ell T_j^{m_k}(S_k,0)\| \\
&\le \sum_{\ell=1}^{j}\frac{\|D\|^\ell}{\ell!}\sum_{t=\ell}^{p}\left\|\overline{\nabla_x^t f}(X_k) - \nabla_x^t f(X_k)\right\|\frac{\|S_k\|^{t-\ell}}{(t-\ell)!} \\
&\le \sum_{\ell=1}^{j}\frac{1}{\ell!}\sum_{t=\ell}^{p}\left\|\overline{\nabla_x^t f}(X_k) - \nabla_x^t f(X_k)\right\|\frac{\tau_k^t}{(t-\ell)!} \\
&\le \sum_{\ell=1}^{j}\frac{1}{\ell!}\sum_{t=\ell}^{p}\frac{1}{(t-\ell)!}\,\omega\frac{\overline{\Delta T}_{k,\min}}{6} \\
&\le \frac{1}{6}\omega\,\overline{\Delta T}_{k,\min}\sum_{\ell=1}^{j}\frac{1}{\ell!}(1+\chi_{p-\ell}(1)) \\
&\le \omega\,\overline{\Delta T}_{m_k,j}(S_k,D),
\end{aligned}$$

as desired. To prove the second assertion, observe that Assumption 6.1.1(ii) implies that $\|\nabla^\ell f(X_k)\| \le L_{f,\ell-1}$, for $j \in \{2,\ldots,p\}$, and thus, using (6.16), that

$$\begin{aligned}
\|\overline{\nabla_x^\ell f}(X_k)\| &\le \|\nabla_x^\ell f(X_k)\| + \|\overline{\nabla_x^\ell f}(X_k) - \nabla_x^\ell f(X_k)\| \\
&\le L_{f,\ell-1} + \Theta_0,
\end{aligned}$$

for $\ell \in \{2,\ldots,p\}$.
This gives the desired conclusion with the choice $\Theta = \max_{\ell\in\{2,\ldots,p\}} L_{f,\ell-1} + \Theta_0$. □

Of course, the conditions stated in Lemma 55 are sufficient but by no means necessary to ensure $\mathcal{M}_k$. In particular, they make no attempt to exploit a possible favourable balance between the errors made on derivatives at different degrees, nor do they take into account that $\mathcal{M}_k^{(1)}$, $\mathcal{M}_{k,j}^{(2)}$ and $\mathcal{M}_{k,j}^{(3)}$ only specify conditions on model accuracy in a finite, dimension-independent, subset of directions. Despite these limitations, (6.15) and (6.16) allow the crucial conclusion that $\mathcal{M}_k$ does occur if the derivatives $\overline{\nabla_x^j f}(X_k)$ are sufficiently accurate compared to the model decrease. Moreover, since one would expect that, as an approximate minimiser is approached, $\|S_k\|$, $\|D_{k,j}\|$ and $\|\overline{D}_{k,j}\|$ (and thus τ_k) become small, they also show the accuracy requirement becomes looser for derivatives of higher degree.

We now formalise our assumption on the stochastic process generated by the SARqp algorithm.

Assumption 6.1.2. *For all $k \ge 0$, the event $\mathcal{M}_k$ satisfies the condition*

$$p_{\mathcal{M}_k} = P(\mathcal{M}_k|\mathcal{A}_{k-1}^M) = \mathbb{E}[\mathbb{1}_{\mathcal{M}_k}|\mathcal{A}_{k-1}^M] \ge p_* \tag{6.17}$$

for some $p_* \in (\frac{1}{2},1]$ independent of k.

We observe that, in contrast with [23, 49], the definition of $\mathcal{M}_k$ does not require the model to be "linearly/quadratically" accurate everywhere in a ball around x_k of radius at least $\|s_k\|$, but merely that its variation is accurate enough along s_k (as specified in $\mathcal{M}_k^{(1)}$) and along $d_{k,j}$ and $\overline{d}_{k,j}$ (as specified in $\mathcal{M}_{k,j}^{(2)}$ and $\mathcal{M}_{k,j}^{(3)}$)* for all $j \in \{1, \ldots, q\}$. The need to consider $\mathcal{M}_{k,j}^{(2)}$ and $\mathcal{M}_{k,j}^{(3)}$, for $j \in \{1, \ldots, q\}$, in the definition of $\mathcal{M}_k$ results from our insistence that q-th order approximate optimality must include j-th order approximate optimality for all such j. The Assumption 6.1.2 also parallels assumptions in [23, 44, 49, 99], where the accuracy in function values is measured using the guaranteed model decrease or proxies given by the $(p+1)$-st power of the trust-region radius or the step length. Finally, the conditions imposed by $\mathcal{M}_{k,j}^{(2)}$ and $\mathcal{M}_{k,j}^{(3)}$ are only used whenever considering the value of $\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k)$, that is in Lemma 56, itself only called upon in Lemma 58 in the case where $\|S_k\| \leq 1$. As a consequence, they are irrelevant when long steps are taken ($\|S_k\| > 1$).

6.2 Worst-case evaluation complexity

Having set the stage and stated our assumptions, we may now consider the worst-case evaluation complexity of the SARqp algorithm. Our aim is to derive a bound on the expected number of iterations $\mathbb{E}(N_\epsilon)$ which is needed, in the worst-case, to reach an (ϵ, δ)-approximate q-th-order necessary minimiser.

Specifically, given $q \in \{1, 2\}$, N_ϵ is the number of iterations required until (6.2) holds for the first time, i.e.,

$$N_\epsilon = \inf\left\{ k \geq 0 \mid \phi_{f,j}^{\Delta_{k-1,j}}(X_k) \leq \epsilon_j \frac{\Delta_{k-1,j}^j}{j!} \text{ for } j \in \{1, \ldots, q\} \right\}. \tag{6.18}$$

Note that $\phi_{f,j}^{\Delta_{k-1,j}}(X_k)$, the j-th order optimality measure at iteration k, uses the optimality radius $\Delta_{k-1,j}$ resulting from the step computation at iteration $k-1$, as is the case in [10, 35]. Now recall that the trial point $X_{k-1} + S_{k-1}$ and the vector of radii Δ_{k-1} are deterministic once the inexact model at iteration $k-1$ is known. Therefore, these variables are measurable for $\mathcal{A}_{k-1}^M$ and because of our deterministic assumptions on the accuracy of f, the event $\{X_k = X_{k-1} + S_{k-1}\}$ (which occur when iteration $k-1$ is successful) is also measurable for $\mathcal{A}_{k-1}^M$. As a consequence and since $\phi_{f,j}^{\Delta_{k-1,j}}(X_k)$ uses exact derivatives of f, the event $\{N_\epsilon = k\}$ is measurable with respect to $\mathcal{A}_{k-1}^M$. The definition (6.18) can thus be viewed as that of a family of ϵ-dependent hitting times for the stochastic process generated by the SARqp algorithm (see, e.g., [44, Section 2.3]).

For completeness, we report the standard notion of hitting time for stochastic processes as stated in [44].

Definition 6.2.1 (Hitting time). *For a give discrete time stochastic process Z_t, the hitting time for an event $\{Z_t \in S\}$ is a random variable, defined as $T_S = \min\{t \geq 0 \mid Z_t \in S\}$, corresponding to the first time t at which the event $\{Z_t \in S\}$ occurs. In our context S will be a set of real numbers smaller than some given value.*

6.2.1 General properties of the algorithm

We first consider properties of "accurate" iterations, in the sense that $\mathcal{M}_k$ occurs, and start with the relation between $\phi_{m_k,j}^{\delta_{k,j}}(s_k)$ and its approximation.

The next lemma is inspired by Lemma 31, but significantly differs in that it now requires considering both directions $d_{k,j}$ and $\overline{d}_{k,j}$.

*A slightly stronger assumption would be to require a sufficient relative accuracy along s_k and in a (typically small) neighbourhood of s_k.

Lemma 56. Consider any realisation of the SARqp algorithm and assume that $\mathcal{M}_k$ occurs. Then, for $j \in \{1, \ldots, q\}$,

$$(1-\omega)\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k) \le \phi_{m_k,j}^{\delta_{k,j}}(s_k) \le (1+\omega)\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k). \tag{6.19}$$

Proof. Let $j \in \{1, \ldots, q\}$. Consider $d_{k,j}$ defined in (6.10). From (6.5), we have that

$$\begin{aligned}
\Delta T_j^{m_k}(s_k, d_{k,j}) &\le \overline{\Delta T}_j^{m_k}(s_k, d_{k,j}) + |\Delta T_j^{m_k}(s_k, d_{k,j}) - \overline{\Delta T}_{m_k,j}(s_k, d_{k,j})| \\
&\le (1+\omega)\overline{\Delta T}_{m_k,j}(s_k, d_{k,j}) \\
&\le (1+\omega) \underset{\|d\|\le\delta_{k,j}, x_k+s_k+d\in\mathcal{X}}{\text{globmax}} \overline{\Delta T}_{m_k,j}(s_k, d) \\
&= (1+\omega)\overline{\Delta T}_{m_k,j}(s_k, \overline{d}_{k,j}),
\end{aligned}$$

where we used the fact that $\mathcal{M}_k$ occurs to derive the second inequality and considered $\overline{d}_{k,j}$ defined in (6.11). Therefore,

$$\phi_{m_k,j}^{\delta_{k,j}}(s_k) = \Delta T_j^{m_k}(s_k, d_{k,j}) \le (1+\omega)\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k).$$

This proves the rightmost inequality of (6.19). Similarly, using our assumption that $\mathcal{M}_k$ occurs, we obtain that

$$\begin{aligned}
\Delta T_j^{m_k}(s_k, \overline{d}_{k,j}) &\ge \overline{\Delta T}_{m_k,j}(s_k, \overline{d}_{k,j}) - |\Delta T_j^{m_k}(s_k, \overline{d}_{k,j}) - \overline{\Delta T}_{m_k,j}(s_k, \overline{d}_{k,j})| \\
&\ge (1-\omega)\overline{\Delta T}_{m_k,j}(s_k, \overline{d}_{k,j})
\end{aligned}$$

and, hence, from (4.10) and (6.5), that

$$(1-\omega)\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k) \le \underset{\|d\|\le\delta_{k,j}, x_k+s_k+d\in\mathcal{X}}{\text{globmax}} \Delta T_j^{m_k}(s_k, d) = \phi_{m_k,j}^{\delta_{k,j}}(s_k),$$

which concludes the proof of (6.19). □

The next step is to adapt an important property of $\Delta_{k,j}$ in the exact case to our inexact framework.

Lemma 57. Suppose that Assumption 6.1.1(ii) holds. Then, for any $j \in \{1, \ldots, q\}$, $1 \le q \le 2$:

1. if $j = 1$ and $\mathcal{X}$ is convex or if $j = 2$ and $\mathcal{X} = \mathbb{R}^n$, $\Delta_{k,j}$ can always be chosen equal to one;
2. in the other cases, and assuming that $\mathcal{M}_k$ occurs, then, either $\|s_k\| > 1$ or $\Delta_{k,j} \le 1$ can be chosen such that
$$\Delta_{k,j} \ge \kappa_\delta(\sigma_k)\epsilon_j, \tag{6.20}$$
where $\kappa_\delta(\sigma) \in (0,1)$ is independent of ϵ and decreasing with σ.

Proof. The proof broadly follows the developments of [34, Lemmas 4.3 and 4.4], except that it now uses the model involving approximate derivatives and that L_f, the upper bound of the derivatives of f at x_k derived from Assumption 6.1.1(ii), is now replaced by Θ, as guaranteed by $\mathcal{M}_k^{(4)}$. The details are provided in the Appendix (Section A.2). □

In what follows, we will assume that, whenever $q > 2$ or $\mathcal{X}$ is nonconvex or $q = 2$ and $\mathcal{X} \subseteq \mathbb{R}^n$, the SAR$qp$ algorithm computes a pair (s_k, δ_k) such that, for each $j \in \{1, \dots, q\}$, $\delta_{k,j}$ is always within a fraction of its maximal value, thereby ensuring (6.20).

We now prove a crucial inequality relating the step length to the accuracy requirements.

Lemma 58. Consider any realisation of the SARqp algorithm. Assume that $\mathcal{M}_k$ occurs, that iteration k is successful and that, for some $j \in \{1, \dots, q\}$, (6.2) fails for $(x_{k+1}, \delta_{k,j})$. Then, either $\|s_k\| > 1$ or

$$(1-2\theta)\epsilon_j \frac{\delta_{k,j}^j}{j!} \le \frac{L_{f,p} + \sigma_k}{(p-q+1)!} \sum_{\ell=1}^{j} \frac{\delta_{k,j}^\ell}{\ell!} \|s_k\|^{p-\ell+1}. \tag{6.21}$$

Proof. See [34, Lemma 5.3] for the composite unconstrained Lipschitz continuous case. Suppose that $\|s_k\| \le 1$. Since (6.2) fails at $(x_{k+1}, \delta_{k,j})$, we must have that

$$\phi_{f,j}^{\delta_{k,j}}(x_{k+1}) > \epsilon_j \frac{\delta_{k,j}^j}{j!} > 0, \tag{6.22}$$

for some $j \in \{1, \dots, q\}$. Define d to be the argument of the global minimum in the definition of $\phi_{f,j}^{\delta_{k,j}}(x_{k+1})$. Hence,

$$0 < \|d\| \le \delta_{k,j} \tag{6.23}$$

and $x_k + d \in \mathcal{X}$. Using (6.22), (4.10) and the triangle inequality, we thus obtain that

$$\phi_{f,j}^{\delta_{k,j}}(x_{k+1}) = \Delta T_j^f(x_{k+1}, d) \le \left| \Delta T_j^f(x_{k+1}, d) - \Delta T_j^{m_k}(s_k, d) \right| + \Delta T_j^{m_k}(s_k, d). \tag{6.24}$$

Recalling from [35, Lemma 2.4]) that

$$\|\nabla_s^\ell \|s_k\|^{p+1}\| = \frac{(p+1)!}{(p-\ell+1)!} \|s_k\|^{p-\ell+1},$$

we may now use the fact that $x_{k+1} = x_k + s_k$, since the iteration k is successful, (4.9) (with $L = L_{f,p}$ in Lemma 28), the definition of $T_j^{m_k}(s_k, d)$ (see the lines below (4.30)–(4.31)), (6.23) and the triangle inequality to obtain that

$$\begin{aligned} \left| \Delta T_j^f(x_{k+1}, d) - \Delta T_j^{m_k}(s_k, d) \right| \quad &\le \quad \sum_{\ell=1}^{j} \frac{\delta_{k,j}^\ell}{\ell!} \|\nabla_x^\ell f(x_{k+1}) - \nabla_s^\ell T_p^f(x_k, s_k)\| \\ &\quad + \frac{\sigma_k}{(p+1)!} \sum_{\ell=1}^{j} \frac{\delta_{k,j}^\ell}{\ell!} \|\nabla_s^\ell \|s_k\|^{p+1}\| \\ &\le \quad \frac{L_{f,p} + \sigma_k}{(p-q+1)!} \sum_{\ell=1}^{j} \frac{\delta_{k,j}^\ell}{\ell!} \|s_k\|^{p-\ell+1}. \end{aligned} \tag{6.25}$$

Moreover, using (6.5), (6.19) and the fact that $\omega < 1$ (see (6.6)), we deduce that

$$\Delta T_j^{m_k}(s_k, d) \le \phi_{m_k,j}^{\delta_{k,j}}(s_k) \le (1+\omega) \overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k) \le 2\theta \epsilon_j \frac{\delta_{k,j}^j}{j!}. \tag{6.26}$$

Substituting (6.25) and (6.26) into (6.24) and using (6.23) and (6.22), we obtain (6.21). □

Lemma 59. Suppose that Assumption 6.1.1(ii) holds and consider any realisation of the SARqp algorithm. Suppose also that $\mathcal{M}_k$ occurs, that iteration k is successful and that, for some $j \in \{1, \ldots, q\}$, (6.2) fails for $(x_{k+1}, \delta_{k,j})$. Then,

$$\|s_k\|^{p+1} \geq \psi(\sigma_k)\epsilon_j^{\pi}, \tag{6.27}$$

where

$$\pi = \begin{cases} \dfrac{p+1}{p-q+1}, & \text{if } q = 1 \text{ and } \mathcal{X} \text{ is convex or if } q = 2 \text{ and } \mathcal{X} = \mathbb{R}^n, \\ \dfrac{q(p+1)}{p}, & \text{otherwise,} \end{cases} \tag{6.28}$$

and

$$\psi(\sigma) = \begin{cases} \min\left[1, \left(\dfrac{2(1-2\theta)(p-q+1)!}{3q!(L_{f,p}+\sigma)}\right)^{\pi}\right], & \begin{array}{l}\text{if } q = 1 \text{ and } \mathcal{X} \text{ is convex, or} \\ \text{if } q = 2 \text{ and } \mathcal{X} = \mathbb{R}^n,\end{array} \\ \min\left[1, \left(\dfrac{2(1-2\theta)(p-q+1)!\,\kappa_\delta(\sigma)^{q-1}}{3q!(L_{f,p}+\sigma)}\right)^{\pi}\right], & \text{otherwise.} \end{cases} \tag{6.29}$$

Proof. See [34, Lemma 5.4] for the unconstrained case. If $\|s_k\| > 1$, the conclusion immediately follows. Suppose therefore that $\|s_k\| \leq 1$ and consider j such that (6.21) holds. Recalling the definition of χ_j in (4.14), (6.21) can be rewritten as

$$\alpha_k\, \epsilon_j\, \delta_{k,j}^j \leq \|s_k\|^{p+1} \chi_j\left(\frac{\delta_{k,j}}{\|s_k\|}\right), \tag{6.30}$$

where we have set

$$\alpha_k = \frac{(1-2\theta)(p-q+1)!}{q!(L_{f,p}+\sigma_k)}.$$

In particular, from (4.15) we have that, when $\|s_k\| \leq \delta_{k,j}$,

$$\alpha_k\, \epsilon_j \leq \frac{3}{2}\|s_k\|^{p+1}\left(\frac{1}{\|s_k\|}\right)^j = \frac{3}{2}\|s_k\|^{p-j+1}. \tag{6.31}$$

Suppose first that $q = 1$ and $\mathcal{X}$ is convex or $q = 2$ and $\mathcal{X} = \mathbb{R}^n$. Then, from our assumptions and Lemma 57, $\delta_{k,j} = 1$ and $\|s_k\| \leq 1 = \delta_{k,j}$. Thus (6.31) yields the first case of (6.28)–(6.29). Suppose now that $q > 2$ or that $\mathcal{X}$ is not convex or that $q = 2$ and $\mathcal{X} \subseteq \mathbb{R}^n$. Then, our assumptions imply that (6.20) holds. If $\|s_k\| \leq \delta_{k,j}$, we may again deduce from (6.31) that the first case of (6.28)–(6.29) holds, which implies, because $\kappa_\delta(\sigma) < 1$ and $1/(p-j+1) \leq j/p$, that the second case also holds. Consider therefore the case where $\|s_k\| > \delta_{k,j}$. Then, (6.30) and (4.16) give that

$$\alpha_k\, \epsilon_j\, \delta_{k,j}^j \leq \frac{3}{2}\|s_k\|^{p+1}\left(\frac{\delta_{k,j}}{\|s_k\|}\right),$$

which, with (6.20), implies the second case of (6.28)–(6.29), as requested. □

Note that $\psi(\sigma)$ is decreasing as a function of σ in both cases of (6.29). We now investigate the decrease of the exact objective function values at successful iterations.

Lemma 60. Suppose that Assumption 6.1.1(ii) holds and consider any realisation of the SARqp algorithm. Then,

$$\overline{\Delta T}_p^f(x_k,s_k) \geq \frac{\sigma_k}{(p+1)!}\|s_k\|^{p+1} \geq \frac{\sigma_{\min}}{(p+1)!}\|s_k\|^{p+1} \geq 0. \tag{6.32}$$

Moreover, if iteration k is successful,

$$f(x_k) - f(x_{k+1}) \geq \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\|s_k\|^{p+1} > 0. \tag{6.33}$$

Proof. The inequality (6.32) immediately follows from (6.3), (6.4) and (6.8). Now the fact that iteration k is successful, together with (6.6) and (4.25)–(4.26), imply that

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq \overline{f}(x_k) - \overline{f}(x_{k+1}) - 2\omega\overline{\Delta T}_p^f(x_k,s_k) \\ &\geq \eta\overline{\Delta T}_p^f(x_k,s_k) - 2\omega\overline{\Delta T}_p^f(x_k,s_k), \end{aligned}$$

yielding (6.33) by virtue of (6.32) and (6.6). □

We finally conclude our analysis of "accurate" iterations by proving a standard achievement in the analysis of adaptive regularisation methods. A similar version of this result was presented in [10, Lemma 4.2] for the case where both function values and models are sufficiently accurate.

Lemma 61. Suppose that Assumption 6.1.1(ii) holds and that $\beta > 1$ is given. For any realisation of the SARqp algorithm, if iteration k is such that $\mathcal{M}_k$ occurs and

$$\sigma_k \geq \overline{\sigma} \overset{\text{def}}{=} \max\left[\beta\sigma_0, \frac{L_{f,p}}{1-\eta-3\omega}\right], \tag{6.34}$$

then iteration k is successful.

Proof. The proof parallels the one of Lemma 36. Specifically, it follows from (4.64) with $L = L_{f,p}$ and $\eta_2 = \eta$, assuming (6.34). □

6.2.2 Bounding the expected number of steps with $\Sigma_k \geq \overline{\sigma}$

We now return to the general stochastic process generated by the SARqp algorithm, aiming at bounding from above the expected number of steps in the process generated by the algorithm with $\Sigma_k \geq \overline{\sigma}$.

To this purpose, for all $0 \leq k \leq \ell$, given $\ell \in \{0,\ldots,N_\epsilon - 1\}$, we define the events Λ_k as in (5.44), $\mathcal{S}_k$ as in (5.19) and we again make use of N_Λ and N_{Λ^c} in (5.45), being the number of steps in the stochastic process induced by the SARqp algorithm with $\Sigma_k < \overline{\sigma}$ and $\Sigma_k \geq \overline{\sigma}$, before N_ϵ in (6.18) is met, respectively.

In what follows we suppose that Assumption 6.1.1 and Assumption 6.1.2 hold.

In this case, a bound on the number of successful iterations with $\Sigma_k \geq \overline{\sigma}$ in terms of the overall number of iterations $l+1$ can be obtained considering [44, Lemma 2.2]. In fact, by the definitions of Λ_k and $\mathcal{S}_k$ we know that when $\mathbb{1}_{\Lambda_k^c}\mathbb{1}_{\mathcal{S}_k} = 1$, then we have a successful iteration and $\Sigma_k \geq \overline{\sigma}$. In this case, $\Sigma_{k+1} = \max\left[\sigma_{\min}, \gamma^{-1}\Sigma_k\right]$. It follows that, among all iterations, at most half can be successful and have $\Sigma_k \geq \overline{\sigma}$, because for each such iteration, when Σ_k gets reduces by a factor of γ^{-1} , there has to be at least one iteration when Σ_k is increased by a factor γ, being $\Sigma_0 < \overline{\sigma}$ (recall (6.34) and that $\Sigma_0 = \sigma_0$).

This result is summarised by the lemma below, which is basically [44, Lemma 2.2] written for the mechanism of the SARqp algorithm.

Lemma 62. Given $l \in \{0, ..., N_\epsilon - 1\}$, for all realisations of the SARqp algorithm,

$$\sum_{k=0}^{l} \mathbb{1}_{\Lambda_k^c} \mathbb{1}_{\mathcal{S}_k} \leq \frac{l+1}{2}. \tag{6.35}$$

We may now proceed as in Subsection 5.2.2, using [44, Lemma 2.2], to derive an upper bound on $\mathbb{E}[N_{\Lambda^c}]$. In particular, the argument unfolds as follows.

- As in [44], we note that both $\widehat{\sigma}(\mathbb{1}_{\Lambda_k})$ and $\widehat{\sigma}(\mathbb{1}_{\Lambda_k^c})$ belong to $\mathcal{A}_{k-1}^M$, since the random variable Λ_k is fully determined by the first $k-1$ iterations of the SARqp algorithm. Then, setting $\ell = N_\epsilon - 1$, we can rely on Lemma 48 (with $W_k = \mathbb{1}_{\Lambda_k^c}$) and (6.17) to deduce that
$$\mathbb{E}\left[\sum_{k=0}^{N_\epsilon - 1} \mathbb{1}_{\Lambda_k^c} \mathbb{1}_{\mathcal{M}_k}\right] \geq \mathbb{E}\left[\sum_{k=0}^{N_\epsilon - 1} p_{\mathcal{M}_k} \mathbb{1}_{\Lambda_k^c}\right] \geq p_* \, \mathbb{E}\left[\sum_{k=0}^{N_\epsilon - 1} \mathbb{1}_{\Lambda_k^c}\right]. \tag{6.36}$$
- As a consequence, given that Lemma 61 ensures that each iteration k where $\mathcal{M}_k$ occurs and $\sigma_k \geq \overline{\sigma}$ is successful, we have that
$$\sum_{k=0}^{N_\epsilon - 1} \mathbb{1}_{\Lambda_k^c} \mathbb{1}_{\mathcal{M}_k} \leq \sum_{k=0}^{N_\epsilon - 1} \mathbb{1}_{\Lambda_k^c} \mathbb{1}_{\mathcal{S}_k} \leq \frac{N_\epsilon}{2},$$
in which the last inequality follows from (6.35), with $\ell = N_\epsilon - 1$. Taking expectation in the above inequality, using (6.36) and recalling the rightmost definition in (5.45), we obtain that, for any realisation,
$$\mathbb{E}[N_{\Lambda^c}] \leq \frac{1}{2p_*} \mathbb{E}[N_\epsilon], \tag{6.37}$$
as in [44, Lemma 2.3].

The remaining upper bound on $\mathbb{E}[N_\Lambda]$ will be the focus of the next section.

6.2.3 Bounding the expected number of steps with $\Sigma_k < \overline{\sigma}$

With reference to the SARqp algorithm, the analysis of this section again considers the set of events in Definition 5.2.1 and parallels the one of Subsection 5.2.3.

The following crucial lemma is proved according to the properties of the SARqp algorithm, in order to derive an upper bound for the expectation $\mathbb{E}[N_{AS}]$ of the number of accurate successful iterations with $\Sigma_k < \overline{\sigma}$.

Lemma 63. Let Assumption 6.1.1 hold. For all realisations of the SARqp algorithm we have that

$$\mathbb{E}[N_{AS}] \leq \frac{(f_0 - f_{\text{low}})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\overline{\sigma})} \left(\min_{j \in \{1,\dots,q\}} \epsilon_j\right)^{-\pi} + 1, \tag{6.38}$$

where π, $\psi(\sigma)$ and $\overline{\sigma}$ are defined in (6.28), (6.29) and (6.34), respectively.

Proof. For all realisations of the SARqp algorithm we have that:

- if iteration k is successful, then (6.33) holds;

- if iteration k is successful and accurate (i.e., $\mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k} = 1$) and (6.2) fails for $(x_{k+1}, \delta_{k,j})$, then (6.27) holds;
- if iteration k is unsuccessful, the mechanism of the SARqp algorithm guarantees that $x_k = x_{k+1}$ and, hence, that $f(x_{k+1}) = f(x_k)$.

Therefore, for any $\ell \in \{0, ..., N_\epsilon - 1\}$,

$$\begin{aligned} f_0 - f_{\rm low} &\geq f_0 - f(X_{\ell+1}) = \sum_{k=0}^{\ell} \mathbb{1}_{\mathcal{S}_k}(f(X_k) - f(X_{k+1})) \geq \sum_{k=0}^{\ell} \mathbb{1}_{\mathcal{S}_k} \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\|S_k\|^{p+1} \\ &\geq \sum_{k=0}^{\ell-1} \mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k} \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\|S_k\|^{p+1} \qquad (6.39) \\ &\geq \sum_{k=0}^{\ell-1} \mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k} \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\psi(\Sigma_k)\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{\pi} \\ &\geq \sum_{k=0}^{\ell-1} \mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\overline{\Lambda}_k} \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\psi(\Sigma_k)\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{\pi} \\ &\geq \frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\psi(\overline{\sigma})\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{\pi}\left(\sum_{k=0}^{\ell-1} \mathbb{1}_{\mathcal{S}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\overline{\Lambda}_k}\right), \qquad (6.40) \end{aligned}$$

having set $f_0 \stackrel{\rm def}{=} f(X_0)$ and where the last inequality is due to fact that $\psi(\sigma)$ is a decreasing function. We now notice that, by Definition 5.2.1,

$$N_{AS} - 1 \leq \sum_{k=0}^{N_\epsilon - 2} \mathbb{1}_{\overline{\Lambda}_k}\mathbb{1}_{\mathcal{M}_k}\mathbb{1}_{\mathcal{S}_k}.$$

Hence, letting $\ell = N_\epsilon - 1$ and taking expectations in (6.40), we conclude that

$$f_0 - f_{\rm low} \geq (\mathbb{E}[N_{AS}] - 1)\frac{(\eta - 2\omega)\sigma_{\min}}{(p+1)!}\psi(\overline{\sigma})\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{\pi},$$

which is equivalent to (6.38). □

In light of this result, considering that the thesis of Lemma 52 and of Lemma 50 still hold for the SARqp algorithm, we can proceed as in Subsection 5.2.3 obtaining the bounds below.

- (5.48): $\mathbb{E}[N_\Lambda] \leq \mathbb{E}[N_I] + \mathbb{E}[N_A]$;
- (5.49): $\mathbb{E}[N_I] \leq \frac{1-p_*}{p_*}\mathbb{E}[N_A]$;
- (5.50): $\mathbb{E}[N_A] \leq \mathbb{E}[N_{AS}] + \mathbb{E}[N_U]$;
- (5.54): $\mathbb{E}[N_U] \leq \mathbb{E}[N_{AS}] + \mathbb{E}[N_{IS}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil$;
- (5.56): $\mathbb{E}[N_{IS}] \leq \frac{1-p_*}{p_*}(\mathbb{E}[N_{AS}] + \mathbb{E}[N_U])$, giving (5.55): $\mathbb{E}[N_{IS}] \leq \frac{1-p_*}{2p_*-1}\left(2\mathbb{E}[N_{AS}] + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil\right)$ from (5.54);
- the bound:

$$\mathbb{E}[N_A] \leq \frac{p_*}{2p_*-1}\left[\frac{2(f_0 - f_{\rm low})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\overline{\sigma})}\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{-\pi} + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right], \qquad (6.41)$$

  obtained starting from (5.50) and applying (5.54), (5.55), (6.38);

- the bound:

$$\mathbb{E}[N_\Lambda] \le \frac{1}{2p_* - 1}\left[\frac{2(f_0 - f_{\rm low})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\overline{\sigma})}\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{-\pi} + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right], \qquad (6.42)$$

  given by (5.48), (5.49) and (6.41).

The final evaluation complexity result is then a direct consequence of the results above.

Theorem 64. Suppose that Assumption 6.1.1 and Assumption 6.1.2 hold. Then, the following conclusions also hold.

1. If $q = 1$ and $\mathcal{X}$ is convex or if $q = 2$ and $\mathcal{X} = \mathbb{R}^n$, then

$$\mathbb{E}[N_\epsilon] \le \kappa(p_*)\left(\frac{2(f_0 - f_{\rm low})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\sigma_s)}\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{-\frac{p+1}{p-q+1}} + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right);$$

2. If $q > 2$ or $\mathcal{X}$ is nonconvex or $q = 2$ and $\mathcal{X} \subseteq \mathbb{R}^n$, then

$$\mathbb{E}[N_\epsilon] \le \kappa(p_*)\left(\frac{2(f_0 - f_{\rm low})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\sigma_s)}\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{-\frac{q(p+1)}{p}} + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right),$$

with $\kappa(p_*) \overset{\rm def}{=} \frac{2p_*}{(2p_*-1)^2}$ and N_ϵ, $\psi(\sigma)$, $\overline{\sigma}$ defined as in (6.18), (6.29), (6.34), respectively.

Proof. Recalling the definitions (5.45) and the bound (6.37), we obtain that

$$\mathbb{E}[N_\epsilon] = \mathbb{E}[N_\Lambda^c] + \mathbb{E}[N_\Lambda] \le \frac{\mathbb{E}[N_\epsilon]}{2p_*} + \mathbb{E}[N_\Lambda],$$

which implies, using (6.42), that

$$\frac{2p_* - 1}{2p_*}\mathbb{E}[N_\epsilon] \le \frac{1}{2p_* - 1}\left(\frac{2(f_0 - f_{\rm low})(p+1)!}{(\eta - 2\omega)\sigma_{\min}\psi(\overline{\sigma})}\left(\min_{j\in\{1,\ldots,q\}} \epsilon_j\right)^{-\pi} + \left\lceil \log_\gamma\left(\frac{\overline{\sigma}}{\sigma_0}\right)\right\rceil + 2\right).$$

This bound and the inequality $\frac{1}{2} < p_* \le 1$ (see Assumption 6.1.2) yield the desired result. □

Since the SARqp algorithm requires at most two function evaluations and one evaluation of the derivatives of orders 1 to p per iteration, the bounds stated in the above theorem effectively provide an upper bound on the average evaluation complexity of finding (ϵ, δ)-approximate q-th order minimisers, $1 \le q \le p \le 2$.

Theorem 64 generalises the complexity bounds stated in [34, Theorem 5.5] to the case where evaluations of f and its derivatives are inexact, under probabilistic assumptions on the accuracies of the latter. As it was shown in [34, Theorems 6.1 and 6.4] that the evaluation complexity bounds are sharp in order of the tolerance for exact evaluations and Lipschitz continuous derivatives of f, this is also the case for the bounds of Theorem 64.

6.3 Chapter conclusion

We have shown that the SARqp algorithm, a stochastic inexact adaptive regularisation algorithm using derivatives of order up to p, computes an (ϵ, δ)-approximate q-th order minimiser of problem (4.3) in at most $\mathcal{O}(\epsilon^{-\frac{p+1}{p-q+1}})$ iterations in expectation if q is either one and $\mathcal{X}$ is convex, or two and the problem is unconstrained, while it may need $\mathcal{O}(\epsilon^{-\frac{q(p+1)}{p}})$ iterations in expectation in the other cases[†]. The results have been fully proved for the case $1 \leq q \leq p \leq 2$, but they apply in the same way for generic $p > 2$ and $1 \leq q \leq p$, as done in [8].

The proved evaluation complexity bounds are sharp in the order of ϵ (see [34]). We therefore conclude that, if the probabilities $p_{\mathcal{M}_k}$ in Assumption 6.1.2 are suitably large, the evaluation complexity of the SARqp algorithm is identical (in order) to that of the exact algorithm in [34].

[†]These simplified order bounds assume that $\epsilon_j = \epsilon$, for $j \in \{1, \dots, q\}$.

Part IV

Supervised Learning and Numerical Tests

Chapter 7

Machine Learning and Real-Life Applications

This final part is devoted to the numerical performance of the proposed methods, after setting the context from which the problems to be solved arise.

Within the framework of supervised learning, our experimentation mainly covers non-convex binary classification tasks on real and synthetic datasets, ill-conditioned problems and a real-life engineering application related to the parametric design of centrifugal pumps.

7.1 Basics of supervised learning

The aim of machine learning techniques is that of building models, usually called meta-models, that are able to learn a task, for example to approximate a function of performance or classify data from given examples arising from the considered application, to take decisions.

The examples are usually N couples $\{(a_i, y_i)\}_{i=1}^N$, called samples. Each couple contains a d_a-dimensional vector $a_i \in \mathbb{R}^{d_a}$ (the so-called feature vector), whose components are called features (or attributes), and a target vector $y_i \in \mathbb{R}^{d_y}$ (it can be known or unknown) that corresponds to the outcome of the process the model is expected to perform.

To start with the basic concept of "learning a task", let us refer to the seminal definition by T. Mitchell in [89].

> *A computer program (or machine) is said to learn from experience $\mathscr{E}$ with respect to some class of tasks $\mathscr{T}$ and performance measure $\mathscr{P}$, if its performance at tasks in $\mathscr{T}$, as measured by $\mathscr{P}$, improves with experience $\mathscr{E}$.*

These entities $\mathscr{T}$, $\mathscr{P}$ and $\mathscr{E}$ could be at a first look something abstract, so let us go a little bit inside their meaning.

Classes of problems $\mathscr{T}$

The main classes of problems addressed by machine learning strategies are the following.

- *Classification.* In this case, given $m \geq 2$ classes (or categories), the algorithm has to learn model which is a function
$$h : \mathbb{R}^{d_a} \to \mathcal{Y}, \tag{7.1}$$

to predict the class value $y \in \mathcal{Y}$ of the considered input vector $a \in \mathbb{R}^{d_a}$. Classification tasks are characterised by $\mathcal{Y} = \mathbb{Z}$, i.e. the elements in $\mathcal{Y}$ are discrete, corresponding to disjoint classes, mutually exclusive and each feature vector $a \in \mathbb{R}^{d_a}$ belongs to one and only one of these classes. In this context, the target value y is called label. If $|\mathcal{Y}| = 2$ (consider for instance $\mathcal{Y} = \{0, 1\}$) we are in the range of binary classification. A classical example of this task is given by the spam/anti-spam filtering for emails. When $|\mathcal{Y}| > 2$ (a common choice is $\mathcal{Y} = \{1, 2, ..., m\}$, $m > 2$) we speak about multi-class (or multi-labeled) classification.

- *Regression.* This class is characterised by $\mathcal{Y} = \mathbb{R}$, since the machine has to learn a function $h : \mathbb{R}^{d_a} \to \mathbb{R}$ to predict a performance value $y \in \mathbb{R}$ for the input $a \in \mathbb{R}^{d_a}$ of interest. Classical examples are given by vocal or facial recognition, automatic translations of data into a discrete text strings, imputation of missing data from a dataset, reconstruction of the original signal given an occurrence of it affected by some source of noise.

Measure of performance $\mathscr{P}$

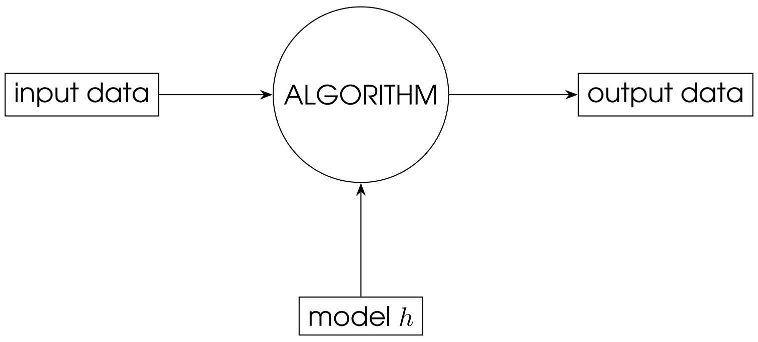
It is a crucial quantitative measure to understand the capability of the machine learning algorithm at hand in solving the given task. $\mathscr{P}$ is not a universal measure, but it is usually $\mathscr{T}$-dependent and corresponds to a measure of the losses from inaccurate predictions. For instance, it can be chosen as the percentage of feature vectors uncorrectly classified (measuring the error made by the learned model), when validating a classification task in which the true labels of the considered feature vectors are available. Different performance measures can also be considered in the resolution of the same problem (this will be the case of Subsection 7.3), depending on the aspects of the phenomenon that are more relevant for the use of the performance prediction given by the learning algorithm.

The experience $\mathscr{E}$

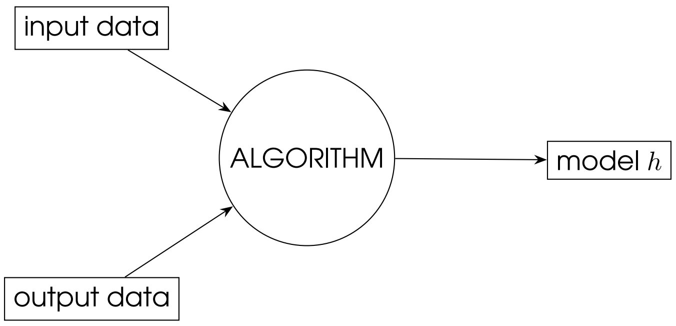
Machine learning approaches are usually split in the following three classes.

- *Supervised learning.* This class is characterised by a known correspondence between inputs data and their outputs. Specifically, a set of inputs (feature vectors) $\{a_i \in \mathbb{R}^{d_a}\}_{i=1}^{N}$ are given along with their outputs $\{y_i \in \mathcal{Y}\}_{i=1}^{N}$, corresponding to the labels if a classification task is considered or, more in general, to target vectors. These couples define the so-called training set. The main feature is that those outputs $y_i \in \mathcal{Y}$ are known for each input $a_i \in \mathbb{R}^{d_a}$, $i \in \{1, ..., N\}$. The goal is thus to use these data to learn a function h defined as in (7.1), to (hopefully) accurately associate each input $a_i \in \mathbb{R}^{d_a}$ to its output $y_i \in \mathcal{Y}$. As we will see in the next subsection, such a function is obtained minimising the chosen performance measure $\mathscr{P}$, when such a $\mathscr{P}$ is a measure of the error made predicting outputs with the learned model h. This is with the aim of using the computed model h to predict targets of new untested data, i.e. of feature vectors $a \in \mathbb{R}^{d_a}$ whose target vectors $y \in \mathcal{Y}$ are unknown.
  Usually, before using the model h for predictions, the predictive ability of the computed model h obtained by the learning algorithm is tested computing the performance measure on a set $\{(\overline{a}_i, \overline{y}_i)\}_{i=1}^{N_T}$ of testing samples, that form the so-called testing set. Such examples can be given or they can be obtained by considering a randomisation of the available dataset $\{(a_i, y_i)\}_{i=1}^{N_{tot}=N+N_T}$, using the first N examples as the training data forming the training set and the remaining N_T as the testing data in the testing set.

  The scheme is indeed specular with respect to standard programming. In fact, a standard programming algorithm (see the figure below) provides output data once input data and a given function h to link the input data to the output data are given.



By contrast, the supervised machine learning framework takes input data (feature vectors) *and output data* (target vectors) *as its input*, returning a model function h to be use to link the input data to the output data, as represented below.



- *Unsupervised learning.* At variance with the supervised case, there is now no information about the correlation between the input data and their outputs in the training set. The learning algorithm has here to find intrinsic similarities or structures within the input data, in order to split them into classes. Examples are given by clustering, in which data are grouped basing on classes of resemblance, or anomaly detection.
- *Reinforcement (or semi-supervised) learning.* This third case stems from the two above and is a sort of hybrid approach. Similar to human learning models, a small amount of input data with the corresponding outputs are given for training, but along with a large amount of input data with unknown outputs. The goal is now to combine this information and avoid either discarding the untested data and doing supervised learning or discarding the input data with known outputs and doing unsupervised learning.

In the following, we will introduce the formal machine learning procedure and the underlying optimisation problem, focusing on supervised binary (nonconvex) classification, that is the framework to which our numerical tests belong.

7.1.1 The underlying optimisation problem

Our goal is to determine a prediction function (model)

$$h : \mathcal{A} \to \mathcal{Y},$$

from an input space $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ to an output space $\mathcal{Y} \subseteq \mathbb{R}^{d_y}$ (with d_a and d_y positive integers) such that, given $a \in \mathcal{A}$, the value $h(a)$ offers an accurate prediction of the true output y. That is, choosing a prediction function h that avoids rote memorisation and instead generalises the concepts that can be learned from a given set of training examples. To do this, one should choose the prediction function h by attempting to minimise a risk measure, i.e. a measure of the errors made by h in the predictions, over a suitably selected family $\mathcal{H}$ of prediction functions.

To formalise this idea, suppose that the examples are sampled from a joint probability distribution function $P(a, y)$, $P : \mathcal{A} \times \mathcal{Y} \to [0, 1]$, that simultaneously represents the distribution $P(a)$ of inputs as well as the conditional probability $P(y|a)$ of the label y being appropriate for an input a. One should seek to find h that yields a small expected risk $\mathbb{E}[\mathbb{1}_{h(a)\neq y}]$ of misclassification over all possible inputs. In other words, an $h \in \mathcal{H}$ that minimises the probability of having a wrong prediction, by minimising the so-called expected risk

$$R^*(h) \overset{\text{def}}{=} P(\{h(a) \neq y\}) = \int_{\mathcal{A}\times\mathcal{Y}} \mathbb{1}_{\{h(a)\neq y\}} dP(a, y) = \mathbb{E}[\mathbb{1}_{\{h(a)\neq y\}}], \tag{7.2}$$

over $h \in \mathcal{H}$. Such a framework has the double difficulty of being variational, since we are optimising over a set of functions $h \in \mathcal{H}$, and stochastic, since the objective function involves an expectation. Moreover, the minimisation of (7.2) is attempted to be done without the explicit knowledge of the joint probability P. The only tractable option is thus to construct a surrogate problem that relies solely on the known training examples $\{(a_i, y_i)\}_{i=1}^N$, $(a_i, y_i) \in \mathcal{A} \times \mathcal{Y}$. Overall, there are two main issues that must be addressed:

- how to choose the parameterised family of prediction functions $\mathcal{H}$;
- how to find the particular prediction function $h^* \in \mathcal{H}$ that is optimal.

Theoretical guidelines for choosing the class of functions $\mathcal{H}$

As suggested in [28], the family of functions $\mathcal{H}$ should be determined with the following three potentially competing goals in mind.

i) The class $\mathcal{H}$ should contain prediction functions that are able to achieve a low value of the so-called empirical risk

$$R_N^*(h) = \frac{1}{N}\sum_{i=1}^N \mathbb{1}_{\{h(a_i)\neq y_i\}},$$

defined over the training set, so as to avoid both rote memorisation, that can lead to overfitting, and underfitting the data. For rote memorisation we mean a function h that simply memorises the training samples, such as

$$h(a) = \begin{cases} y_i, & \text{if } a = a_i,\ i \in \{1, ..., N\}, \\ 0 \text{ or } 1 \text{ (arbitrarily), otherwise.} \end{cases}$$

In this case h is thus extremely precise in associating the feature vectors a_i of the training set to the right labels y_i, but totally inefficient in doing the same on the testing set. On the other hand, training data could be too redundant or insufficient to let the computed model h able to properly predict labels on the testing set. This can be in principle achieved by selecting a rich family of functions or by using a priori knowledge to select a well-targeted family.

ii) The gap $|R^*(h) - R_N^*(h)|$ should be small over all $h \in \mathcal{H}$. Generally, this gap decreases if the number of training examples increases; but, due to potential overfitting, it increases when one uses richer families of functions (see [28]). This latter fact puts the second goal at odds with the first.

iii) The family of functions $\mathcal{H}$ should be selected so that one can efficiently solve the corresponding optimisation problem, the difficulty of which may increase when one employs a richer family of functions and/or a larger training set.

The observation about the gap between expected and empirical risk can be understood by recalling certain laws of large numbers. For instance, the Hoeffding inequality (in [74]) guarantees that, with probability at least $(1-t)$, one has

$$|R^*(h) - R^*_N(h)| \leq \sqrt{\frac{1}{2N}\log\left(\frac{2}{t}\right)},$$

for a given $h \in \mathcal{H}$. We note that the number N of training samples is at the denominator and t is fixed, so this bound offers the intuitive explanation that the gap decreases as one uses more training examples. However, this view is not enough in the context of machine learning, since h is not a fixed function, but the functional variable over which we are optimising. For this reason, one often turns to uniform laws of large numbers and the concept of the Vapnik-Chervonenkis (VC) dimension of $\mathcal{H}$, a measure of the capacity of such a family of functions (see [119]), in order to get an upper bound on $|R^*(h) - R^*_N(h)|$ that is uniform among the class function $\mathcal{H}$.
Resorting to the VC dimension to measure capacity, one of the most important results in learning theory can be established: with $d_{\mathcal{H}}$ defined as the VC dimension of $\mathcal{H}$, one has with probability at least $(1-t)$ that

$$\sup_{h\in\mathcal{H}} |R^*(h) - R^*_N(h)| \leq \mathcal{O}\left(\sqrt{\frac{1}{2N}\log\left(\frac{2}{t}\right) + \frac{d_{\mathcal{H}}}{N}\log\left(\frac{N}{d_{\mathcal{H}}}\right)}\right). \qquad (7.3)$$

We highlight that for a fixed $d_{\mathcal{H}}$,

$$\sqrt{\frac{1}{2N}\log\left(\frac{2}{t}\right) + \frac{d_{\mathcal{H}}}{N}\log\left(\frac{N}{d_{\mathcal{H}}}\right)} \to 0, \quad \text{as } N \to \infty,$$

providing uniform convergence of $R^*_N(h)$ to $R^*(h)$, when $h \in \mathcal{H}$ with fixed VC dimension $d_{\mathcal{H}}$, as the number of training samples N increases. However, it also shows that, for a fixed N, the gap can widen for larger $d_{\mathcal{H}}$. Indeed, to maintain the same gap, one must increase N at the same rate if $d_{\mathcal{H}}$ is increased.
The uniform convergence embodied in this result is crucial in machine learning, since one wants to ensure that the prediction system performs well with any new data provided to it. Interestingly, (7.3) is independent of the number of parameters that distinguish a particular member function h of the family $\mathcal{H}$. In some settings, such as logistic regression (see, e.g. [24]), this number is essentially the same as $d_{\mathcal{H}}$, which might suggest that the task of optimising over $h \in \mathcal{H}$ is more cumbersome as $d_{\mathcal{H}}$ increases. However, this is not always the case. Certain families of functions are amenable to minimisation despite having a very large or even infinite number of parameters [117, Section 4.11]. For example, support vector machines (see [53]) were designed to take advantage of this fact [117, Theorem 10.3].
While the bound in (7.3) is theoretically interesting and provides useful insight, it is rarely directly used in practice, since it is typically easier to estimate the gap between empirical and expected risk with cross-validation experiments. We refer to [28] for an overview on this approach, known as *structural risk minimisation* and *early stopping*, that has proved to be widely successful in [118, 119].

Parametrising the class of functions $\mathcal{H}$

Rather than consider a variational optimisation problem over a generic family of prediction functions, as the minimisation of (7.2), we assume that the class of prediction functions $\mathcal{H}$ is formed by functions h parametrised by a vector $x \in \mathbb{R}^n$, over which the optimisation of (7.2) has to be performed.

Formally, for some given $h(\cdot;x) : \mathcal{A} \times \mathbb{R}^n \to \mathcal{Y}$, we consider the parametric family of prediction functions

$$\mathcal{H} \overset{\text{def}}{=} \{h(\cdot;x) \mid x \in \mathbb{R}^n\}.$$

Therefore, the minimisation of (7.2) can be replaced by the minimisation of

$$\int_{\mathcal{A}\times\mathcal{Y}} \mathbb{1}_{\{h(a;x)\neq y\}} dP(a,y), \tag{7.4}$$

over $x \in \mathbb{R}^n$, which is now a stochastic minimisation problem.

Expected risk

We note that the argument of the integral in (7.4) depends on the indicator function $\mathbb{1}_{\{h(a;x)\neq y\}}$, leading to an optimisation problem with discontinuous objective function. This issue can be overcome finding a prediction function $h(\cdot;x)$, $x \in \mathbb{R}^n$, in the parametric family $\mathcal{H}$ that minimises the losses incurred from inaccurate predictions.

For this purpose, we assume to have a given continuous loss function (also called cost function) $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that, given a pair $(a, y) \in \mathcal{A} \times \mathcal{Y}$, yields the loss $\ell(h(a;x), y)$ when $h(a;x)$ and y are the predicted and true targets, respectively. The minimisation of (7.4) can indeed be replaced by the minimisation of

$$R(x) \overset{\text{def}}{=} \int_{\mathcal{A}\times\mathcal{Y}} \ell(h(a;x), y) dP(a,y) = \mathbb{E}[\ell(h(a;x), y)], \tag{7.5}$$

over $x \in \mathbb{R}^n$, which is still a stochastic optimisation problem but with continuous objective function. We will hereafter refer to $R : \mathbb{R}^n \to \mathbb{R}$ as the expected risk (i.e. expected loss), given a parameter vector $x \in \mathbb{R}^n$, with respect to the probability distribution P.

Empirical risk

Furthermore, we highlight that while it may be desirable to minimise (7.5), such a goal is untenable when one does not have complete information about P. In fact, only an occurrence of such a joint probability distribution producing the training data $\{(a_i, y_i)\}_{i=1}^N$ is known and it is given by the sequence of available training data itself. Thus, in practice, one can only seek the solution of a problem that involves an estimate of the expected risk R in (7.5).

Given the training data $\{(a_i, y_i)\}_{i=1}^N$, the idea is thus to consider the empirical risk, redefined as

$$R_N(x) \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \ell(h(a_i;x), y_i), \tag{7.6}$$

in place of the expected risk $R(x)$ in (7.5), obtaining a (possibly nonconvex) optimisation problem with continuous and theoretically computable objective function. Therefore,

$$\min_{x\in\mathbb{R}^n} R_N(x) = \min_{x\in\mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \ell(h(a_i;x), y_i) \tag{7.7}$$

may be considered the practical optimisation problem to be solved.

Last, to simplify notations, let us define

$$\varphi_i(x) \overset{\text{def}}{=} \ell(h(a_i;x), y_i), \quad i \in \{1, .., N\},$$

representing the loss of the i-th sample using the parameter vector $x \in \mathbb{R}^n$. Consequently, the minimisation problem (7.7) takes the well-known finite-sum form (1.108).

Usual choices for the loss function are the following.

- Square loss: $\ell(h(a_i;x),y_i) = (y_i - h(a_i;x))^2$, that is one of the most common.
- ε-insensitive loss: $\ell(h(a_i;x),y_i) = \max[0, |h(a_i;x) - y_i| - \varepsilon]$, that has the problem of being not derivable.
- Huber loss: $\ell(h(a_i;x),y_i) = \begin{cases} \frac{1}{2}(y_i - h(a_i;x))^2, & \text{if } |y_i - h(a_i;x)| < 1, \\ |y_i - h(a_i;x)| - \frac{1}{2} & \text{otherwise,} \end{cases}$, that is a continuous and derivable combination of the previous two.
- log loss: $\ell(h(a_i;x),y_i) = \log\left(1 + e^{-\hat{y}_i x^\top a_i}\right)$, in which $\hat{y}_i = \begin{cases} 1, & \text{if } y_i = 1, \\ -1, & \text{if } y_i = 0. \end{cases}$
  The log loss induces the minimisation problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{-\hat{y}_i x^\top a_i}\right), \tag{7.8}$$

  known as logistic regression (training phase), that comes out with an optimal $x^* \in \mathbb{R}^d$. The label prediction $y_{pred}(a)$ on a new feature vector $a \in \mathbb{R}^{d_a}$ is as follows:

$$y_{pred}(a) = \begin{cases} +1, & \text{if } \sigma(a^\top x^*) \geq 0.5, \\ -1, & \text{if } \sigma(a^\top x^*) < 0.5, \end{cases}$$

  in which $\sigma : \mathbb{R} \to (0,1)$ is the so-called sigmoid function, defined by

$$\sigma(a^\top x) = \frac{1}{1 + e^{-a^\top x}}, \qquad a, x \in \mathbb{R}^n. \tag{7.9}$$

Finally, setting

$$x^* = \arg\min_{x \in \mathbb{R}^n} R_N(x), \qquad \widehat{x} = \arg\min_{x \in \mathbb{R}^n} R(x),$$

we report from [116, Section 3.7] a bound on $|R(\widehat{x}) - R_N(x^*)|$ that holds with probability at least $(1-2t)$, providing that R and R_N are bounded functions of x. In fact, under this assumption, the triangle inequality gives

$$\begin{aligned} |R(\widehat{x}) - R_N(x^*)| &\leq |R(\widehat{x}) - R(x^*)| + |R(x^*) - R_N(x^*)| \\ &\leq \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{H}} + \log\left(\frac{1}{t}\right)}{N}}\right) + \mathcal{O}\left(\sqrt{\frac{\log\left(\frac{1}{t}\right)}{N}}\right), \end{aligned} \tag{7.10}$$

since (see [116, Section 3.7])

$$|R(\widehat{x}) - R(x^*)| \leq \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{H}} + \log\left(\frac{1}{t}\right)}{N}}\right) + \mathcal{O}\left(\sqrt{\frac{\log\left(\frac{1}{t}\right)}{2N}}\right)$$

with probability at least $(1-2t)$, for all $\widehat{x} \in \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$ and

$$|R(x^*) - R_N(x^*) \leq \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{H}} + \log\left(\frac{1}{t}\right)}{N}}\right),$$

with probability at least $(1-t)$, for all $x^* \in \mathbb{R}^n$.

We conclude this paragraph saying that in the context of binary classification, the VC dimension of a class of predictive functions $h(\cdot;x)$, $x \in \mathbb{R}^n$, is the largest integer value $d_{\mathcal{H}}$ for which, given a set of feature vectors $\{a_1, ..., a_{d_{\mathcal{H}}}\}$ and labels $\{y_1, ..., y_{d_{\mathcal{H}}}\}$, it exists a function $h(\cdot;x)$ able to predict all the given labels without errors.

Neural networks meta-models

We now briefly consider the case in which the function h involved with the definition of the empirical risk (7.6) is represented by an Artificial Neural Network (ANN), formalised by the following parametric application

$$a_i \in \mathbb{R}^{d_a} \mapsto net(a_i;x) \overset{\text{def}}{=} h(a_i;x) \in \mathbb{R}^{d_y},$$

given the parameter $x \in \mathbb{R}^n$, as a model to compute the prediction $net(a_i;x)$ of the true performance $y_i \in \mathbb{R}^{d_y}$, for each feature vector a_i in the training set.

The $net(\cdot;x)$ function belongs to a class of functions identified by the choice of the parameter vector x and is applied to each a_i as follows. Given $x \in \mathbb{R}^n$ and the i-th training feature vector a_i, $i \in \{1,...,N\}$, the value of the prediction function $net(a_i;x)$ is computed applying successive transformations, made in layers:

$$a_i^{(0)} \overset{\text{def}}{=} a_i \mapsto a_i^{(1)} \mapsto \cdots \mapsto a_i^{(j)} \mapsto \cdots \mapsto a_i^{(J)} \overset{\text{def}}{=} net(a_i;x).$$

In particular, a canonical feed-forward fully-connected layer (see, e.g., [28]) performs the computation as

$$a_i^{(j)} \overset{\text{def}}{=} \varsigma^{(j)}(W^{(j)} a_i^{(j-1)} + b^{(j)}) \in \mathbb{R}^{d_j}, \tag{7.11}$$

for $j \in \{1,...,J\}$, in which:

- $a_i^{(0)} \overset{\text{def}}{=} a_i \in \mathbb{R}^{d_a}$ represents the input layer, thus $d_0 = d_a$, which is considered as layer 0 and not counted in the total number of layers;
- $J \in \mathbb{N}$ is the number of layers and $(J-1)$ the number of the so-called hidden layers, representing the depth of the network;
- $net(a_i;x) \overset{\text{def}}{=} a_i^{(J)} \in \mathbb{R}^{d_y}$ is the so-called output layer and, hence, $d_J = d_y$;
- the matrix $W^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ (weights), whose (k,r) component is denoted by $W^{(j)}(k,r)$, and the vector $b^{(j)} \in \mathbb{R}^{d_j}$ (bias), whose k-th component is denoted by $b^{(j)}(k)$, contain the parameters of the j-th layer;
- the components $\{a_i^{(j)}(k)\}_{k=1}^{d_j} \subseteq \mathbb{R}$ of $a_i^{(j)} \in \mathbb{R}^{d_j}$ constitute the neurons of the j-th layer;
- $d_j \in \mathbb{N}$ is the number of neurons of the j-th layer;
- $\varsigma^{(j)} : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ is a component-wise nonlinear (otherwise the composition would be linear) activation function associated with the layer j; $\varsigma^{(j)} = (\varsigma_1^{(j)}, \cdots, \varsigma_{d_j}^{(j)})$ with $\varsigma_k^{(j)} : \mathbb{R} \to \mathbb{R}$, $k \in \{1,...,d_j\}$, the activation function of the k-th neuron of layer j.

Classical choices for the activation functions, for $z \in \mathbb{R}$, are:

- the sigmoid (also called logistic) function in (7.9): $\varsigma(z) = \sigma(z)$, commonly used to get outputs in $(0,1)$;
- the hyperbolic tangent function: $\varsigma(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, ofter chosen as the activation function for the neurons of the output layer when the output range is $(-1,1)$ and, more in general, throughout the hidden layers;
- the (non differentiable) Rectified Linear Unit (ReLU) function: $\varsigma(z) = \max\{z,0\}$, usually considered as the activation function for the neurons of the hidden layers because of its simple expression;

- the Exponential Linear Unit (ELU) function: $\varsigma(z) = z\mathbb{1}_{\{z\geq 0\}} + (e^z - 1)\mathbb{1}_{\{z\geq 0\}}$, considered as the differentiable version of ReLU;

- the linear function: $\varsigma(z) = z$, typically used just for the output layer, in case of regression;

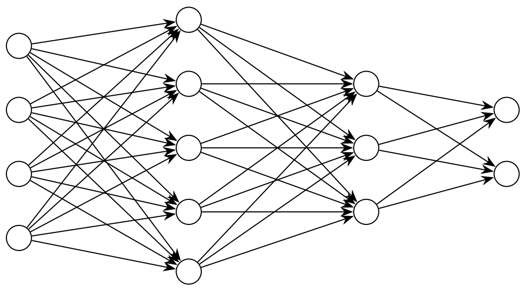
- the softmax (also called normalised exponential) function

$$\varsigma : \{\, v \in \mathbb{R}^k \,\} \to \Big\{ w \in \mathbb{R}^k \mid w(i) > 0,\ \sum_{i=1}^{k} w(i) = 1 \Big\},$$

  such that $e_i^\top \varsigma(v) = w(i) = \frac{e^{v(i)}}{\sum_{r=1}^{k} e^{v(r)}}$. This function is very popular for multi-class classification, since the i-th component $w(i)$ of $\varsigma(v)$ gives the probability of belonging to the class $i \in \{1, ..., k\}$.

We have formally shown how a feed-forward fully-connected network works. For a visual understanding, it is helpful to mention that this kind of network can be seen as an acyclic directed weighted graph, where each neuron corresponds to a node and the $W^{(j)}(k, r)$ represents the weight of the edge connecting the r-th neuron $a_i^{(j-1)}(r)$ of layer $(j-1)$ to the k-th neuron $a_i^{(j)}(k)$ of layer j, for $r \in \{1, ..., d_{j-1}\}$, $k \in \{1, ..., d_j\}$, $j \in \{1, ..., J\}$. We say that a network is fully-connected whether all the neurons in one layer are connected to the neurons in the next layer, while a network is said to be feed-forward wherein connections between the nodes do not form a cycle. In particular, shallow networks are networks with a single hidden layer.

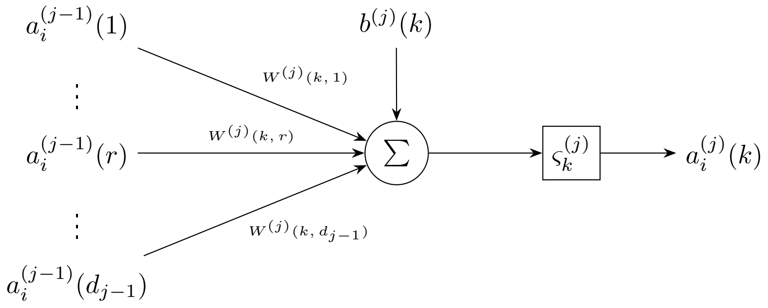
We report below a graphical representation of a feed-forward neural network with $J = 3$ layers, 2 hidden layers, input layer in $\mathbb{R}^4$ ($d_a = d_0 = 4$), first hidden layer in $\mathbb{R}^5$ (5 neurons), second hidden layer in $\mathbb{R}^3$ (3 neurons), output layer in $\mathbb{R}^2$ ($d_y = 2$) and a set of $q = 51$ parameters.



The value of the k-th neuron $a_i^{(j)}(k)$ in the j-th level is determined by

$$a_i^{(j)}(k) = \varsigma_k^{(j)} \left(\sum_{r=1}^{d_{j-1}} W^{(j)}(k, r) a_i^{(j-1)}(r) + b^{(j)}(k) \right) \in \mathbb{R}, \tag{7.12}$$

corresponding to the k-th component of $a_i^{(j)}$ in (7.11), for $k \in \{1, ..., d_j\}$, $j \in \{1, ..., J\}$. According to (7.12), the diagram below gives a visual representation of how the k-th neuron $a_i^{(j)}(k)$ of the layer j works.



We recall that $net(a_i;x) \overset{\text{def}}{=} a_i^{(J)}$, where $a_i^{(J)} \in \mathbb{R}^{d_y}$ ($d_J = d_y = 1$ in the case of a single performance output or label) is the ultimate output vector and the parameters $\{(W^{(1)}, b^{(1)}), ..., (W^{(J)}, b^{(J)})\}$ are compactly collected* in the parameter vector $x \in \mathbb{R}^d$, $d = \prod_{j=1}^J d_j(1 + d_{j-1})$. As an example, if the network has no hidden layers (i.e., $J = 1$) and $h(\{a_i\}_{i=1}^N) \subseteq (0,1)$ (so $d_1 = d_y = 1$), setting $b^{(1)} = 0$ (no bias), denoting $x^\top \overset{\text{def}}{=} W^{(1)} \in \mathbb{R}^{1\times d_a}$ (where $d = d_a$) and choosing $s^{(1)}$ as the sigmoid function (7.9), the scheme reduces to $net(a_i;x) = \sigma(x^\top a_i) \in \mathbb{R}$.

The value of the parameter vector x^* needed to select the particular function $net(\cdot;x^*)$ among the class of functions $\{net(\cdot;x) | x \in \mathbb{R}^n\}$ is thus chosen in order to (approximately) minimise the prediction error on the training set, i.e., problem (7.7), that takes the form of the following unconstrained optimisation problem:

$$\min_{x\in\mathbb{R}^n} f(x) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N \varphi_i(x) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N (y_i - net(a_i;x))^2, \tag{7.13}$$

if one for instance chooses the square loss as a measure of the prediction accuracy, where the objective function to minimise can result in a possibly nonconvex overall objective f (*training phase*).

The optimal parameter vector x^* obtained by the resolution of (7.13) can in turn be used to predict performances on a the testing set $\{\overline{a}_i, \overline{y}_i\}_{i=1}^{N_T}$, on which the values of the true performances $\overline{y}_i$, $i \in \{1, ..., N_T\}$, are known, with the aim of testing the accuracy of the derived predictive model $net(\cdot;x^*)$ for its validation (*testing phase*).

The computed model $net(\cdot;x^*)$ can finally be used to get the prediction $net(a;x^*)$ of the performance on a new feature vector $a \in \mathbb{R}^{d_a}$, $a \notin \{a_i\}_{i=1}^N \cup \{\overline{a}_i\}_{i=1}^{N_T}$, for which the value of performance is untested (*execution phase*).

ANN are widely used as meta-models within machine learning applications since the evaluation of $net(a_i;x)$ and its derivative $\frac{\partial}{\partial x} net(a_i;x)$, usually needed to solve the optimisation problem (7.13) numerically, can be obtained quite efficiently using the forward and backward propagation algorithms (see, e.g., [100]), that are based on the well-known chain rule with Leibniz's notations. We remind that each evaluation of the network requires a forward propagation. Once the output $net(a_i;x)$ of the network is evaluated, its gradient $\frac{\partial}{\partial x} net(a_i;x)$ can be computed with an additional backward propagation.

Furthermore, the use of ANN also finds a theoretical foundation. This is due to a result given by the authors of [75] in 1989, known as *universal approximation theorem for neural networks*, stating that a feed-forward network with a single hidden layer, which contains a finite number of neurons, is a universal approximator among continuous functions on compact subsets of $\mathbb{R}^{d_a}$, under mild assumptions of the activation function (non constant, monotonically-increasing, continuous). We notice that the sigmoid function (7.9) satisfies

*This is easily done considering x as the block vector $x = (x_1^\top, \cdots, x_J^\top)^\top$, with

$$x_j^\top = (W^{(j)}(1,1), \cdots, W^{(j)}(d_j,1), \cdots, W^{(j)}(1,d_{j-1}), \cdots, W^{(j)}(d_j,d_{j-1}), b^{(j)}(1), \cdots, b^{(j)}(d_j)) \in \mathbb{R}^{1\times d_j(1+d_{j-1})}$$

the row vector containing the parameters of the j-th layer, $j \in \{1, \cdots, J\}$.

such assumptions. Nevertheless, the authors in [3] have proved that, in the worst-case, the number of neurons of the hidden layer is exponential in the number of features d_a, leading to challenging implementations. In addition, we have to keep in mind that the optimisation procedure related to the training phase could fail anyway or lead to poor predictions on the testing data. For such reasons, users have often opted for deeper neural networks (i.e., with more hidden layers), but at a lower number of neurons per layer, in compliance with the cost budget and the processing power.

7.2 Numerical tests for nonconvex binary classification

We now consider binary classification problems and focus on the two macro-steps, the training and the testing phase, we adopt for our numerical experimentation.

- *The training phase.* Given a training set $\{(a_i, y_i)\}_{i=1}^N$ of N features $a_i \in \mathbb{R}^n$ and corresponding binary labels $y_i \in \{0,1\}$, we solve the following minimisation problem:

$$\min_{x\in\mathbb{R}^n} f(x) = \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N \varphi_i(x) = \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N \left(y_i - \sigma\left(a_i^\top x\right)\right)^2. \tag{7.14}$$

  That is we use the sigmoid function $\sigma : \mathbb{R} \to (0,1)$ in (7.9) as the model for predicting the values of the labels, that can be seen as the continuous counterpart of the Heaviside step function:

$$H(z) = \begin{cases} 1, \text{ if } z \geq 0, \\ 0, \text{ if } z < 0, \end{cases}$$

  representing the values of the labels. Furthermore, the square loss is considered as a measure of the error on such predictions, that has to be minimised by approximately solving (7.14) in order to come out with the parameter vector x^*, to be used for label predictions on the available testing set.
  Therefore, (7.14) represents the formal minimisation problem (7.7) with:

$$\mathcal{X} = \mathbb{R}^n, \quad \mathcal{Y} = \{0,1\}, \quad h(a_i; x) = \sigma(a_i^\top x), \quad \ell(h(a_i;x), y_i) = (h(a_i;x) - y_i)^2.$$

  Trivially, it has the form (1.108) and can be seen as a neural network without hidden layers and no bias (recall, e.g., Subsection 7.1.1). Moreover, at variance with the logistic regression (7.8), the problem (7.14) is nonconvex. We refer to f as the training loss, which corresponds to the empirical risk $R_N(x)$ in (7.7).

- *The testing phase.* Once the training phase is completed, a number N_T of new data, the testing data $\{\overline{a}_i, \overline{y}_i\}_{i=1}^{N_T}$, is used to validate the computed model. The values $\sigma(\overline{a}_i^\top x^*)$ are used to predict the testing labels $\overline{y}_i$, $i \in \{1, ..., N_T\}$, according to

$$\sigma(\overline{a}_i^\top x^*) = \begin{cases} 1, \text{ if } \overline{a}_i^\top x^* \geq 0, \\ 0, \text{ if } \overline{a}_i^\top x^* < 0, \end{cases}$$

  since

$$\sigma(\overline{a}_i^\top x^*) \geq \frac{1}{2} \quad \Leftrightarrow \quad \overline{a}_i^\top x^* \geq 0.$$

  The corresponding error, measured by $\frac{1}{N_T}\sum_{i=1}^{N_T}\left(\overline{y}_i - \sigma\left(\overline{a}_i^\top x^*\right)\right)^2$, is then computed.

7.2.1 Implementation issues

In this subsection we analyse the behaviour of the ARC frameworks given by the ARC-DH algorithm and its stochastic adaptation described by the SARC-IGDH algorithm within the context of nonconvex finite-sum binary classification.

Concerning the algorithm ARC-DH, we will show that it can be computationally more convenient than ARC variants in literature. Our numerical validation is based on inexact Hessians built via uniform subsampling and rule (2.12) for choosing the sample size. The results obtained indicate that suitable levels of accuracy in the Hessian approximation and careful adaptations of the rule (2.12) improve efficiency of existing procedures exploiting subsampled Hessians.

Turning to the SARC-IGDH algorithm, inexact gradient and Hessian evaluations are performed as sketched in modified Steps 0–2 of Algorithm 13. The performance of the proposed algorithm is compared with that of the corresponding version given by ARC-DH, employing exact gradient, with the aim to provide numerical evidence that adding a further source of inexactness in the gradient computation is beneficial in terms of computational cost saving.

Implementation issues concerning the ARC-DH algorithm, the ARC variants to be compared (see Subsection 7.2.2) and the SARC-IGDH algorithm (see Subsection 7.2.3) are the object of this subsection, while statistics of our runs are discussed in Subsection 7.2.2 and Subsection 7.2.3 for what concerns the numerical tests and comparisons on the ARC-DH algorithm and the SARC-IGDH algorithm, respectively.

The common implementation issues of the main phases of ARC variants considered in the three coming subsections is as follows.

Common parameters tuning for algorithms in Subsections 7.2.2–7.2.3–7.2.4.

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

$$\eta_1 = 0.1, \quad \eta = \eta_2 = 0.8, \quad \gamma_1 = 0.5, \quad \gamma_2 = 1.5, \quad \gamma = \gamma_3 = 2, \quad \alpha = 0.5. \tag{7.15}$$

The initial guess is the null vector $x_0 = (0, ..., 0)^\top \in \mathbb{R}^n$ in all runs.

The minimisation of the cubic model at Step 3 of the ARC-DH algorithm is performed employing a Barzilai-Borwein procedure [103, 4]. To be more precise, we follow [20] and employ the method proposed in [71, NMS Algorithm 2]. As for the procedure in Section 1.3.3 using the Lanczos method, the major per-iteration cost of such Barzilai-Borwein process is one Hessian-vector product for each iteration, needed to compute the gradient of the cubic model at each iteration. The threshold used in the termination criterion (3.5) is $\theta_k = 0.5$, $k \geq 0$.

We impose a maximum of 500 iterations and we declare a successful termination when one of the two following conditions are met:

$$\|\nabla f(x_k)\| \leq \epsilon_1, \tag{7.16}$$

$$|f(x_k) - f(x_{k-1})| \leq 10^{-6}|f(x_k)|,$$

when considering the ARC-DH algorithm and the ARC variants in Subsection 7.2.2, and

$$\|\overline{\nabla f}(x_k)\| \leq \epsilon_1, \tag{7.17}$$

when considering the SARC-IGDH algorithm and the comparisons in Subsection 7.2.3.

The values of the first-order tolerance are $\epsilon_1 = 10^{-3}$, for the numerical tests in Subsection 7.2.2, and $\epsilon_1 = 5 \cdot 10^{-3}$, for the numerical tests in 7.2.3.

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

7.2.2 Numerical tests on the ARC-DH algorithm

In this subsection we test different ARC variants and rules for choosing the sample size of Inexact Hessians and we perform two sets of experiments.

First, in Subsection 7.2.2.1, we compare ARC variants with optimal complexity on a set of synthetic datasets from [15].
The ARC-DH algorithm is compared with versions of ARC employing:

- exact Hessians;
- inexact Hessians $\overline{\nabla^2 f}(x_k)$ with accuracy requirement (3.3) and $c_k = \epsilon_1$, for all $k \geq 0$ [124, 123];
- inexact Hessians $\overline{\nabla^2 f}(x_k)$ with accuracy requirement (3.1) implemented as suggested in [79], i.e. the unavailable information $\|s_k\|$ on the right-hand side is replaced by $\|s_{k-1}\|$, for $k > 0$.

Second, in Subsection 7.2.2.2, 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 (2.12) yields to full sample when high accuracy is imposed, i.e. when $\tau_{2,k} = c_k$ is sufficiently small, and sample sizes $|\mathcal{H}_k|$ equal to a prefixed fraction of N are often employed in literature, even though first-order complexity becomes $\mathcal{O}(\epsilon_1^{-2})$ (see, [5, 12, 15, 123]).

In order to measure the computational cost of the considered algorithms, we use as in [15] 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 $x \in \mathbb{R}^n$ consists in the computation of the scalar product $a_i^\top x$. Once evaluated, this scalar product can be reused for obtaining $\nabla\varphi_i(x)$, while the computation of $\nabla^2\varphi_i(x)$ times a vector $v \in \mathbb{R}^n$ requires the scalar product $a_i^\top v$ and it is as expensive as one $\varphi_i(x)$ evaluation (see Table 2.1). Consequently, each full Hessian-vector product costs as one function or gradient evaluation. When $|\mathcal{H}_k|$ samples are used for the Hessian approximation $\overline{\nabla^2 f}(x_k)$, the cost of one matrix-vector product of the form $\overline{\nabla^2 f}(x_k)v$ is counted as $|\mathcal{H}_k|/N$ EGE.

7.2.2.1 Synthetic datasets

The first class of databases we consider is a set of synthetic datasets from [15], firstly proposed in [92]. These datasets have been constructed so that Hessians have condition numbers of order up to 10^7, a wide spectrum of 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

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

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 $j = 1, \dots, d$. Then, the matrix D is scaled as

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

The computation of the matrix $\overline{\nabla^2 f}(x_k)$ according to (2.12) involves the constant

$$\kappa_{\varphi,2}(x_k) = \max_{i\in\{1,\dots,N\}} \left\{ 2e^{-a_i^T x_k}\left(1+e^{-a_i^T x_k}\right)^{-4} \left| y_i \left(\left(e^{-a_i^T x_k}\right)^2 - 1 \right) + 1 - 2e^{-a_i^T x_k} \right| \|a_i\|^2 \right\}. \tag{7.18}$$

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

In our implementation of the ARC-DH algorithm, the value of c used in (3.10) whenever $\|s_k\| \ge 1$ is such that $|\mathcal{D}_0|$ computed via (2.12) with $\tau_{2,0} = c_0 = c$ satisfies $|\mathcal{H}_0|/N = 0.1$. The failure probability t in (2.12) is set equal to 0.2.

We shall hereafter refer to the implementation of the ARC-DH algorithm as *ARC-Dynamic*.

The numerical tests in this section compare *ARC-Dynamic* with the following variants.

- *ARC-Full*: the ARC-DH algorithm employing exact Hessians;
- *ARC-Sub*: the ARC-DH algorithm with inexact Hessian $\overline{\nabla^2 f}(x_k)$ and accuracy $c_k = \epsilon_1$, for all $k \ge 0$, i.e.
$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \le \epsilon_1, \quad \forall k \ge 0, \tag{7.19}$$
as suggested in [124, 123];
- *ARC-KL*: the ARC-DH algorithm employing inexact Hessian $\overline{\nabla^2 f}(x_k)$ and accuracy $c_k = \chi\|s_{k-1}\|$, for all $k \ge 1$. In other words, we use the accuracy requirement (3.1) replacing, as suggested in [79], the unavailable information $\|s_k\|$ at the righthand side of (3.1) with the norm of the step s_{k-1} at the previous iteration, i.e.,
$$\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \le \chi\|s_{k-1}\|, \quad \forall k \ge 1. \tag{7.20}$$
To make a fair comparison with *ARC-Dynamic*, the sample size $|\mathcal{H}_0|$ 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 $|\mathcal{H}_1|$ resulting from (2.12) with $\tau_{2,1} = c_1 = \chi\|s_0\|$ is 10% of the total number of samples.

The synthetic datasets are listed in Table 7.1. For each dataset, the number N of training samples, the feature dimension d_a 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 $\epsilon_1 = 10^{-3}$) and the value of the scalar c selected.

Dataset	Training N	d_a	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 7.1: Synthetic datasets. Number of training samples (N), feature dimension (d_a), number of testing samples (N_T), 2-norm condition number of the Hessian matrix at the computed solution ($cond$), scalar c used in forming Hessian estimates (c).

In Table 7.2 we report the results on all the synthetic datasets obtained with *ARC-Dynamic* and values c as in Table 7.1. Since the selection of the subset $\mathcal{H}_k$ 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 7.3 and 7.4 we report statistics for these problems solved with our algorithm and constant c different from the default value in Table 7.1; we refer to such runs as *ARC-Dynamic(c)*. We duplicate the results given in Table 7.2 for sake of readibility.

In Figures 7.1–7.2 we additionally show the decrease of the training loss and the testing loss among all the synthetic datasets against the number of EGE and in Figure 7.3 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 Figures 7.1–7.2 we do not plot *ARC-Dynamic*(1.00) as it overlaps with *ARC-Dynamic*.

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 7.2: 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 7.3: 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*.

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 7.4: 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*.

Some comments are in order:

- The condition (7.19) 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 the convergence rate.
- Focusing on the two adaptive strategies *ARC-Dynamic* and *ARC-KL*, Table 7.2 shows that on average the former is less expensive than the latter. Figures 7.1–7.2 show that *ARC-KL* is fast in the first stage of the convergence history, becoming progressively slower as the norm of the step starts changing significantly from an iteration to the other (see Figure 7.4). 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 7.4 shows that the norm of the step changes slowly initially, yet 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 7.3. We observe that such a norm, depicted against EGE, oscillates in *ARC-KL*, while this is not the case for *ARC-Dynamic* and *ARC-Dynamic(c)*.
- Focusing on our proposed adaptive strategy, Figures 7.5–7.6 show that *ARC-Dynamic* uses sets $\mathcal{H}_k$ whose cardinality varies adaptively through iterations and it is considerably smaller than N in most of them. Moreover, the performance of *ARC-Dynamic* appears to be quite insensitive to the choice of the scalar c. In fact, computational savings of *ARC-Dynamic* over *ARC-Sub* are achieved with various values of c, including those reported in Table 7.1.

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 7.5: 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.

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 7.5, 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 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 accuracies dynamically chosen along iterations work 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.

7.2.2.2 Real datasets

In this subsection we present our second set of numerical results, performed on the machine learning datasets using subsampled ARC variants with deterministic suboptimal complexity of order $\mathcal{O}(\epsilon_1^{-2})$. More in depth, we compare our adaptive strategy with the version of ARC considered in [123] where, at each iteration, the Hessian is approximated via subsampling on a set with prefixed and small cardinality.

Our adaptive choice of $|\mathcal{H}_k|$ is implemented by introducing safeguards in (2.12) ($\tau_{2,k} = c_k$). Whenever $\|s_k\| < 1$, we choose the cardinality of $\mathcal{H}_k$ according the following rule:

$$|\mathcal{H}_k| = \max\left\{\lceil 0.05N\rceil, \min\left\{\lceil 0.1N\rceil, \left\lceil \frac{4\rho}{c_k}\left(\frac{2\rho}{c_k}+\frac{1}{3}\right)\log\left(\frac{2n}{t}\right)\right\rceil\right\}\right\}, \quad k \geq 0, \tag{7.21}$$

with $\rho > 0$. Clearly, $|\mathcal{H}_k|/N$ 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 failure probability $t = 0.2$ is still considered. The scalar ρ is chosen so that $|\mathcal{H}_k| = \lceil 0.1N\rceil$ when $c_k = \alpha(1-\theta)\epsilon_1^{2/3}$, i.e. the value of c_k corresponding to $\|\nabla f(x_k)\| = \epsilon_1^{2/3}$, and $\|s_k\| < 1$. Whenever $\|s_k\| \geq 1$, the scalar c used in (3.10) is fixed so that $|\mathcal{H}_k| = \lceil 0.05N\rceil$.

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

We compare *ARC-Dynamic* with the above choice of $|\mathcal{H}_k|$ against its variant using Hessian approximations obtained by subsampling on 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 7.6 we list the specifics of the used datasets and, for sake of completeness, the value of the ratio ρ/c determining the Hessian sample size whenever $\|s_k\| \geq 1$ is used. In the same table, in the column with header ϵ_1, we report the used stopping tolerance.

We report below a brief description of each dataset below and the applicative context from which it arises.

- Mushroom. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the *Agaricus* and *Lepiota* family. Each species is identified as edible or poisonous on the basis of a certain number of characteristics, ranging from the color and the odor to the habitat or the cap shape.

- HTRU2. It is a data set which describes a sample of pulsar stars candidates. Pulsars are a rare type of Neutron stars that produce radio emission detectable here on Earth. They are of considerable interest as probes of space-time, the interstellar medium, and states of matter. Since their detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find, machine learning tools are welcome to automatically detect pulsar candidates from possible outliners.

- Cina0 and a9a. In both cases, the causal discovery task is to uncover the socio-economic factors affecting high income (the target value indicates whether the income exceeds 50K). Among all, attributes include age, workclass, marital status and education.

- Gisette. The aim is to detect the highly confusable handwritten digits "4" and "9".

- MNIST. The MNIST dataset, usually considered for classifying the 10 handwritten digits, is here used to discard even digits (labelled by 1) from odd digits (labelled by 0).
- Ijcnn1. It has been introduced in 2001, for a challenge during the International Joint Conference on Neural Networks (IJCNN) conference. The attributes relate to the physical aspects of an internal combustion engine and aim at forecasting whether an regular ignition will occur or not.
- Reged0. It is a genomics dataset to find genes which could be responsible of lung cancer. From the causal discovery point of view, it is in fact very important to separate genes whose activity cause lung cancer from those whose activity is a consequence of the disease.

All test problems have been solved with $\epsilon_1 = 10^{-3}$, except for Cina0 and HTRU2, where the tolerance has been increased to 10^{-2}, since for lower values of ϵ_1 we had no longer improvements on the decrease of the training and the testing loss, regardless of the method in use. By contrast, Mushroom was also solved using the tighter tolerance $\epsilon_1 = 10^{-5}$, as below the threshold $\epsilon_1 = 10^{-3}$ further reduction in the training and testing loss was observed and the percentage of failures in the classification on the testing set dropped from 1% to zero. This can be observed in Table 7.7, where we report the average percentage of correctly classified testing data.

We underline that the gap between the percentages reported in each column of Table 7.7 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.

Dataset	Training N	d_a	Testing N_T	ϵ_1	ρ/c
Mushroom (83)	6503	112	1621	10^{-3}	2.3241
				10^{-5}	2.3241
HTRU2 (83)	10000	8	7898	10^{-2}	3.6942
Cina0 (45)	10000	132	6033	10^{-2}	2.8671
Gisette (83)	5000	5000	1000	10^{-3}	1.6182
MNIST (81)	60000	784	10000	10^{-3}	6.3841
a9a (83)	22793	123	9768	10^{-3}	4.3922
Ijcnn1 (47)	49990	22	91701	10^{-3}	7.5283
Reged0 (47)	400	999	100	10^{-3}	0.4443

Table 7.6: Real datasets. Size of the training set (N), problem dimension (d_a), size of the testing set (N_T), tolerance ϵ for approximate optimality (ϵ_1) and the ratio ρ/c used for sample sizes computations.

Method	Mushroom		HTRU2	Cina0	Gisette	MNIST	a9a	Ijcnn1	Reged0
	$\epsilon_1 = 10^{-3}$	$\epsilon_1 = 10^{-5}$							
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 7.7: 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.

In Table 7.8 we report, for each considered test problem and for each method under comparison, the average number of EGE performed over 20 runs. We compare the performance of *ARC-Dynamic* with that of *ARC-Full* and *ARC-Fix(p)*, $p \in \{0.01, 0.05, 0.1, 0.2\}$.

Method	Mushroom		HTRU2	Cina0	Gisette	MNIST	a9a	Ijcnn1	Reged0
	$\epsilon_1 = 10^{-3}$	$\epsilon_1 = 10^{-5}$							
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 7.8: 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.

Focusing on the strategies employing a prefixed sample size, Table 7.8 shows that there is not a clear winner, as their performances 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 within these applications. On the other hand, *ARC-Dynamic* always terminates with the lowest number of EGE and the 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 7.7. Denoting by T the set of test problems in Table 7.6, 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 $p_t \in T$ by the solver $s \in S$, the performance profile [59] for each $s \in S$ is defined as the fraction

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

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 the 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 $\tau = 1.9$ 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 the runs we observe 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 7.8–7.11 where the training and testing loss are plotted versus the number of EGE; representative runs reported concern datasets MNIST and Gisette.

7.2.3 Numerical tests on the SARC-IGDH algorithm

Before presenting our numerical results on the SARC-IGDH algorithm, let us go a little bit further in the actual implementation of gradient and Hessian approximations to be computed at Step 1 and Step 2 of Algorithm 13 and describe the cost measure in use for comparisons.

Preliminary, we note that if (7.17) holds and the model is accurate (recall Definition 5.1.1), then, by (5.14),

$$\|\nabla f(x_k)\| \le \|\overline{\nabla f}(x_k)\| + \|\nabla f(x_k) - \overline{\nabla f}(x_k)\| \le \overline{\epsilon}_1 := \epsilon_1 + \kappa[(1-\theta)/\sigma_{min}]^2\epsilon_1^2$$

and, hence, x_k is an $\overline{\epsilon}_1$-approximate first-order optimality point. Since the model is accurate with probability at least p_*, the iterate x_k is an $\overline{\epsilon}_1$-approximate first-order optimality point with probability at least p_*. For the numerical tests of this subsection the failure probability t in (2.11)–(2.12) is set equal to 0.18, therefore the probability of success p_* in Definition 5.1.1 corresponds to $(1-t)^2 \simeq 0.67 > 2/3$.

We further note that the exact gradient and the Hessian of the component functions $\varphi_i(x)$, $i \in \{1, ..., N\}$, are given in Table 2.1. Then, the gradient and the Hessian approximations $\overline{\nabla^j f}(x_k)$, $j \in \{1,2\}$, computed at Step 1 and Step 2 of Algorithm 13 according to (2.5)–(2.6), (2.11)–(2.12), involve the constant $\kappa_{\varphi,2}(x_k)$ in (7.18) and

$$\kappa_{\varphi,1}(x_k) \quad = \quad \max_{i\in\{1,\dots,N\}} \left\{ 2e^{-a_i^\top x_k}\left(1+e^{-a_i^\top x_k}\right)^{-2}\left|y_i - \left(1+e^{-a_i^\top x_k}\right)^{-1}\right|\|a_i\|\right\},$$

whose computations can indeed be an issue in theirselves. Nevertheless, reasoning as in Subsection 7.2.2.1 for the ARC-DH algorithm, the exactness and the specific form (see (7.14)) of the function evaluation $f(x_k)$ imply that the values $a_i^\top x_k$, $1 \le i \le N$, are available at iteration k and, hence, $\kappa_{\varphi,j}(x_k)$, $j \in \{1,2\}$, can be determined at the (offline) extra cost of computing $\|a_i\|^j$, $j \in \{1,2\}$, for $1 \le i \le N$.

As in Subsection 7.2.2.1, the value of c used in (5.1), in order to reduce the iteration computational cost whenever $\|s_k\| \ge 1$, is such that $|\mathcal{H}_0|$ computed via (2.12), with $\tau_{2,0} = c$ (first approximation of the Hessian), satisfies $|\mathcal{H}_0|/N = 0.1$. We indeed start using the 10% of the examples to approximate the Hessian.

Concerning the gradient approximation performed at Step 1 of Algorithm 13, the value of τ_0 is chosen in order to use a prescribed percentage of the number of training samples N to obtain $\overline{\nabla f}(x_0)$. In all runs, such a percentage has been set to 0.4. Then, we proceeded as follows. We computed $\overline{\nabla f}(x_0)$ via (2.5), with $|\mathcal{G}_0|/N = 0.4$. Then, we compute τ_0 so that (2.11), with $\tau_{1,0} = \tau_0$, is satisfied as an equality. Finally, the value of κ at Step 1.2 of Algorithm 13 has been correspondingly set to $4\tau_{1,0}^{(0)}\left(\sigma_0/\|\overline{\nabla f}(x_0)\|\right)^2$, with $\tau_{1,0}^{(0)} = \tau_0$. This way, the acceptance criterion of Step 1.2 is satisfied without further inner iterations (i.e., for $i = 0$), when $k = 0$, and τ_0 is indeed considered as the starting accuracy level for the gradient approximation at each execution of Step 1 of Algorithm 13.

We will hereafter refer to such implementation of the SARC-IGDH algorithm coupled with Algorithm 13 as $SARC$.

The numerical tests of this section compare $SARC$ with *ARC-Dynamic*, that employs exact gradient evaluations, with $\gamma_1 = 1/\gamma$, $\gamma_2 = \gamma_3 = \gamma$ and $\eta_1 = \eta_2 = \eta$.

We have already noticed that the problem (7.14) arises in the training of an artificial neural network with no hidden layers and no bias. Nevertheless, to cover the general situation where $SARC$ algorithm is applied to more complex neural networks, we have followed the approach in [123] for what concerns the cost measure. Going into more details, at the generic iteration k, we count the N forward propagations needed to evaluate the objective in (7.14) at x_k as a unit Cost Measure (CM), while the evaluation of the approximated gradient at the same point requires $|\mathcal{G}_k|$ additional backward propagations at the weighed cost $|\mathcal{G}_k|/N$ CM. Moreover, each vector-product $\overline{\nabla^2 f}(x_k)v$, $v \in \mathbb{R}^n$, needed at each iteration of the Barzilai-Borwein method used to minimise the cubic model at Step 3 of the SARC-IGDH algorithm, is performed via finite-differences (see Subsection 1.3.3). Then it has to be computed $\overline{\nabla f}(x_k + hv)$, ($h \in \mathbb{R}^+$), leading to additional $|\mathcal{H}_k|$ forward and backward propagations, at the price of the weighted cost $2|\mathcal{H}_k|/N$ CM, and a potential extra-cost of $|\mathcal{H}_k \setminus (\mathcal{G}_k \cap \mathcal{H}_k)|/N$ CM to compute $\nabla \varphi_i(x_k)$, for all $i \in \mathcal{H}_k \setminus (\mathcal{G}_k \cap \mathcal{H}_k)$. This latter approximation is computed once at the beginning of the Barzilai-Borwein procedure. Denoting by r the number of Barzilai-Borwein iterations at iteration k, the increase of the CM at the k-th iteration of *ARC-Dynamic* and *SARC* related to the derivatives computation is reported in Table 7.9.

ARC-Dynamic	*SARC*
$1 + 2|\mathcal{H}_k|r/N$	$(|\mathcal{G}_k| + 2|\mathcal{H}_k|r + |\mathcal{H}_k \setminus (\mathcal{G}_k \cap \mathcal{H}_k)|)/N$

Table 7.9: Increase of the CM at the k-th iteration of *ARC-Dynamic* and *SARC* related to the derivatives computation; r denotes the number of performed Barzilai-Borwein iterations.

We will refer to the Cost Measure at Termination (CMT) as the main parameter to evaluate the efficiency of the method within the numerical tests.

We finally report statistics of the numerical tests performed by *SARC* and *ARC-Dynamic* on the set of synthetic datasets from Subsection 7.2.2.1, listed in Table 7.1, together with the corresponding value of c used to build the Hessian approximation.

In Table 7.10 we report, for both $SARC$ and *ARC-Dynamic*, the total number of iterations (n-iter), the value of Cost Measure at Termination (CMT) and the mean percentage of saving (Save-M) obtained by *SARC* with respect to *ARC-Dynamic*. Since the selection of the subsets $\mathcal{G}_k$, $\mathcal{H}_k$ in (2.11)–(2.12) is uniformly and randomly made at each iteration of the method, statistics in the forthcoming tables are averaged over 20 runs.

Dataset	*ARC-Dynamic*		*SARC*		
	n-iter	CMT	n-ter	CMT	Save-M
Synthetic1	11.1	130.84	10.0	95.27	27%
Synthetic2	10.6	109.56	10.2	93.08	15%
Synthetic3	11.2	109.64	10.0	97.52	11%
Synthetic4	11.0	124.07	10.4	100.48	19%
Synthetic6	10.0	84.18	10.1	106.31	−26%

Table 7.10: Synthetic datasets. The columns are divided in two different groups. *ARC-Dynamic*: average number of iterations (n-iter) and CMT. *SARC*: average number of iterations (n-iter), CMT and mean percentage of saving (Save-M) obtained by *SARC* over *ARC-Dynamic*. Mean values over 20 runs.

Method	Synthetic1	Synthetic2	Synthetic3	Synthetic4	Synthetic6
ARC-Dynamic	94.34%	92.68%	94.64%	95.52%	93.82%
SARC	93.18%	92.44%	93.62%	94.61%	93.70%

Table 7.11: Synthetic datasets. Binary classification rate at termination on the testing set employed by *ARC-Dynamic* and *SARC*; mean values over 20 runs.

Table 7.10 shows that the novel adaptive strategy employed by $SARC$ results more efficient than *ARC-Dynamic*, reaching an ϵ_1-approximate first-order stationary point at a lower CMT, in all cases except from Synthetic6.

This is obtained without affecting the classification accuracy on the testing sets, as it is shown in Table 7.11, where the average binary accuracy on the testing sets achieved by the methods under comparison is reported. Note that the resolution via $SARC$ is cheaper for the Synthetic6 (see Table 7.10), but the classification rate at termination is slightly worst (see Table 7.11).

To give more evidence of the gain in terms of CMT provided by $SARC$ on the synthethic datasets along the iterative process, we display in Figures 7.12–7.13 the decrease of the training and the testing loss versus the adopted cost measure CM, while Figure 7.14 is reserved to the plot of the computed gradient norm versus CM. For such figures, a representative plot is considered among each series of 20 runs obtained by $SARC$ and *ARC-Dynamic* on each of the considered dataset.

In all cases, except from Synthetic6, Figures 7.12–7.13 show the savings gained by $SARC$ in terms of the overall computational cost, as well as the improvements in the training phase and the testing accuracy under the same cost measure.

More in general, we stress that second-order methods show their strength on these ill-conditioned datasets, since all the tested procedures manage to reduce the norm of the gradient and reach high accuracies in the classification rate. Even if we believe that reporting binary classifications accuracy obtained by each of the considered methods at termination is relevant in itself, we remark that the higher accuracy obtained at termination by *ARC-Dynamic* (see Table 7.11) is just due to the fact the $SARC$ stops earlier. This should not be confused with a better performance of *ARC-Dynamic*, since Figures 7.12–7.13 for Synthetic1–Synthetic4 highlights that, along all datasets, when $SARC$ stops its testing loss is sensibly below the corresponding one performed by *ARC-Dynamic* at the same CMT value.

We finally analyse the adaptive choices of the sample sizes $|\mathcal{G}_k|$ and $|\mathcal{H}_k|$ given by (2.11)–(2.12) in Figure 7.15. As expected, the two strategies are more or less comparable when selecting the sample sizes for Hessian approximations, while the number of samples used to compute gradient approximations by $SARC$ oscillates across all iterations, always remaining far below the full sample size.

In so doing, we outline that too small values of τ_0 seem to have a bad influence on the performance of $SARC$, while as τ_0 increases it generally produces frequent saving in the CMT, once that it is above a certain threshold value. In support of this observation, we report in Figure 7.16 the variation of CMT against τ_0 on Synthetic1 and Synthetic4.

We finally notice that, except for few iterations at the first stage of the iterative process, the sample size for the Hessian approximation is lower than that used for the gradient approximation. This is in line with the theoretical expectations, as the gradient is eventually required to be more accurate than the Hessian. In fact, the error in the gradient approximation has to be of the order of $\|s_k\|^2$, while that in the Hessian approximation has to be of the order of $\|s_k\|$ (recall Lemma 43 and 44).

7.2.4 Preliminary numerical tests on the AR1DA algorithm

In this section we consider some preliminary tests on algorithm AR1DA, defined by Algorithm 11. Specifically, in Subsection 7.2.4.1, we report the numerical results given by AR1DA on the Mushroom dataset, with and without an ANN models.

For the numerical tests of this section the algorithm has been run until CM=80. Therefore, differently from the previous numerical experimentation, we also consider models arising from ANN and compare the quality of the approximation obtained with a fixed amount of work corresponding to 80 CM.

In addition to the specifications listed in the box on page 148, the following settings are considered:

$$\kappa_\varepsilon = \gamma_\varepsilon = 5 \cdot 10^{-1}, \qquad \omega = \frac{\eta_1}{4} = 2.5 \cdot 10^{-2}.$$

Moreover, the estimates for the objective function in (7.14) and its gradient needed by the AR1DA algorithm at iteration k are provided using (2.10) with $\tau_{0,k} = \omega|\overline{\Delta T}(x_k, s_k)|$ and (2.11) with $\tau_{1,k} = \varepsilon_{1,i_k}$, being i_k the last index considered in the inner loop for i at Step 1 of the ARqpDA algorithm. In both (2.10)–(2.11) the failure probability t is set equal to 0.2 and a constant κ_φ is considered in place of $\kappa_{\varphi,1}(x_k)$ and $\kappa_{\varphi,2}(x_k)$, for all $k \geq 0$. The theoretical guideline for this is that if

$$\kappa_\varphi \geq \max\left[\sup_{x\in\mathbb{R}^n}\left[\max_{i\in\{1,\dots,N\}}|\varphi_i(x)|\right], \sup_{x\in\mathbb{R}^n}\left[\max_{i\in\{1,\dots,N\}}\|\nabla\varphi_i(x)\|\right]\right],$$

then Theorem 19 holds with $\kappa_{\varphi,j}(x_k) = \kappa_\varphi$, for $j \in \{1,2\}$ and $k \geq 0$.

The value of κ_φ has been chosen experimentally on each of the considered datasets, in order to get reasonable cardinalities $|\mathcal{D}_{k,\ell}|$, $\ell \in \{1,2\}$, and $\mathcal{G}_k$ throughout the running of the algorithm. Subsection 7.2.4.1 considers $\kappa_\varphi = 3 \cdot 10^{-3}$, while $\kappa_\varphi = 10^{-6}$ is used for the numerical tests of Section 7.3.

Also for the tests of this section, the CM used in Subsection 7.2.3 is kept as the cost measure. At this regard we note that each computation of the approximate objective function at x_k (i.e. $\overline{f}(x_k)$) via the left-hand side equation in (2.4) requires the weighted cost $|\mathcal{D}_{k,1}|/N$ CM, while each evaluation of the inexact gradient at the same point using (2.5) with a sample $\mathcal{G}_k \subseteq \mathcal{D}_{k,1}$ needs the extra cost of $|\mathcal{G}_k|/N$ CM. More in general, once $\overline{f}(x_k)$ is computed, the evaluation of $\overline{\nabla f}(x_k)$ costs $(2|\mathcal{G}_k \setminus (\mathcal{G}_k \cap \mathcal{D}_{k,1})| + |\mathcal{G}_k \cap \mathcal{D}_{k,1}|)/N$ CM.

7.2.4.1 Binary classification of the Mushroom dataset via neural networks

We now solve the binary classification task on the Mushroom and the MNIST datasets.

The training phase considers the numerical optimisation of problem (7.13) on the training data $\{(a_i, y_i)\}_{i=1}^N$ of each dataset, with meta-models $net(\cdot; x)$, $x \in \mathbb{R}^q$, given by feed-forward fully-connected ANN, formally given by the string

$$(d_0 = d_a, d_1, \cdots, d_J = d_y),$$

describing the number of neurons of each of the layers, where the notations are inherited by Section 7.1.

We report in the first two columns of Table 7.12 the considered ANN and the respective total number of parameters n. For all layers, the sigmoid activation function (7.9) has been adopted when dealing with the Mushroom dataset, while MNIST is solved using the hyperbolic tangent function ($tanh$) and the sigmoid as the activations for all the hidden layers and the the output layer, respectively.

Thinking about the basic networks $(112, 1)$ and $(784, 1)$, we stress that $q = 112 = d_a$, for the Mushroom, and $q = 784$, for the MNIST (and not $q = d_a d_y + d_y = 113$, $q = 785$, respectively), since in these cases a zero bias ($b^{(1)} = 0$) is considered, so that (7.13) reduces to (7.14). Moreover, for a better understanding, let us focus on the shallow network $(112, 10, 1)$ considered within the Mushroom dataset. We notice that the problem (7.13) takes the specific form:

$$\min_{x\in\mathbb{R}^n} f(x) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N \varphi_i(x) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^N (z_i - net(a_i; x))^2,$$

Dataset	ANN	n	ACC
Mushroom	$(112,1)$	112	97.41%
	$(112,10,1)$	1141	95.37%
	$(112,10,5,1)$	1191	99.07%
	$(112,10,10,1)$	1251	96.55%
	$(112,50,20,1)$	6691	31.28%
MNIST	$(784,1)$	784	87.37%
	$(784,15,1)$	11791	88.42%
	$(784,15,2,1)$	11810	89.23%

Table 7.12: Mushroom and MNIST datasets. List of the considered architectures (ANN) with the related number of parameters (n) and the binary classification rate (ACC) on the testing set via AR1DA; mean values over 20 runs.

in which, given the weights $W^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ and the bias $b^{(j)} \in \mathbb{R}^{d_j}$, $j \in \{1,2\}$,

$$net(a_i;x) = a_i^{(2)} = \sigma\left(\sum_{r=1}^{10} W^{(2)}(1,r) \underbrace{\left(\sigma\left(\sum_{s=1}^{112} W^{(1)}(r,s)a_i(s) + b^{(1)}(r)\right)\right)}_{=a_i^{(1)}(r)} + b^{(2)}\right) \in (0,1),$$

$$x \in \mathbb{R}^n, \qquad n = \sum_{j=1}^{2} d_j(1+d_{j-1}) = 1141,$$

for $i \in \{1,...,N\}$, where we make use of the notations introduced in Subsection 7.1.1 and σ is here component-wise applied.

For each of the two datasets under consideration, each architecture in Table 7.12, coupled with the corresponding optimal parameter vector $x^* \in \mathbb{R}^q$ coming from the training phase, is then used as the predictive model for the labels $\overline{y}_i$ of the feature vectors $\overline{a}_i$, $i \in \{1,...,N_T\}$, of the corresponding testing set. The performance in terms of the binary classification rate (ACC) on the testing set is reported in the fourth column of Table 7.12, considering the mean value over 20 runs.

These values immediately reveal two important features of ANN.

- The first is that using an ANN is not synonym of improving performances, since there are no universal rules to decide the particular structure of the network, establishing the number of hidden layers and of neurons per layers, but this has to be calibrated depending on the specific problem to solve. For instance, the shallow network considered for the Mushroom dataset seems to be a useless complication of the simpler model $(112,1)$ with no hidden layers, since the classification rate undergoes a slight. This does not happen for the MNIST dataset, in which the accuracy increases with the growth of the number of layers within the considered ANN structures. On the other hand, adding a second hidden layer with a small amount of neurons seems to be particularly effective for the Mushroom dataset, giving the best performance of AR1DA in terms of classification on the testing set. This result is also competitive with the rates achieved by the ARC-DH algorithm in Table 7.7.

- The second is that one has to be careful in increasing the number of parameters of the model. In fact, when the number of parameters approaches the number $N = 6503$ (recall Table 7.6) of training data, overfitting can easily occur. This is because in such a situation the model is close to interpolate the data of the training set, resulting in a low ability of generalisation on the testing set. We have shown this phenomenon by progressively increasing the number of neurons in the second hidden layer. The classification rate starts letting down even with $(112, 10, 10, 1)$, but it drops drastically to 31.28% when the network $(112, 50, 20, 1)$ with 6691 parameters is considered. As one may easily verify by looking at Tables 7.12 and 7.6, this is not the case of the MNIST dataset, where the number of parameters n related to the ANN structures in use is always far away from the number $N = 60000$ of training samples.

To get more insight, Figures 7.17–7.19 on pages 188–190 show that the decrease of the training and testing loss against CM goes down to more or less the same values at termination, meaning that the training phase has been particularly effective and thus the method is able of generalising the ability of predicting labels learnt on the training phase when the testing data are considered. The only one exception is, as expected, the ANN $(112, 50, 20, 1)$ (Mushroom dataset), for which the training loss gets down to a higher value and the behaviour of the corresponding testing loss is more erratic.

Finally, we report in Figures 7.20–7.21, pages 191–192, the sample sizes (in percentage) for computed function and gradient against CM.

For what concerns the Mushroom dataset (Figure 7.20), the graphs show that sample size for estimating the objective function oscillates for a while, flattening on the full sample size eventually in $(112, 1)$ and $(112, 10, 1)$. The same is true for the corresponding gradients estimations, even if the growth toward the full sample is slower. The trends are even more fluctuating with $(112, 10, 5, 1)$, $(112, 50, 20, 1)$ and $(112, 10, 10, 1)$, where full sample can still be touched for both function and gradient approximations, but they are often below this limit. In particular, the sample size for computing the gradient always remains below 30% of the total samples with respect to $(112, 10, 10, 1)$, with a huge fractions of the computations under 10%, while this threshold is about 25% for $(112, 10, 5, 1)$ and $(112, 50, 20, 1)$. An analogous behaviour is reflected by Figure 7.21 for MNIST.

Transversely, we note that at the same value of CM, the sample size needed for function approximation is not smaller than the corresponding sample size for computing the gradient, according with the tighter accuracy required on the function with respect to its gradient, suggesting a sort of hierarchy between function and its first-order derivative information. At this regard, we remind that an absolute accuracy requirement is considered for function estimates, while gradient approximations are requested to fulfill a relative accuracy condition.

7.3 A real-life machine learning application: the parametric design of centrifugal pumps

In recent years, the approach of designers to the aerodynamic and mechanical redesign of turbomachinery components has changed with respect to some decades ago [73]. Often, customer requirements lead to analyse the performance of a component with complex geometry and to extend this investigation to different operating conditions simultaneously, with the aim of optimising the performance under tight constraints.

The design process considered in this section is described in [48, 105]. That is an extension of the scheme commonly used for the design of a component of a single pump. It is based on the fact that the customer is usually provided with a catalogue of sample pumps and the one that is closest to his requirements is individuated among them. Then, starting from this baseline configuration the new pump is built, optimising the parameters describing its geometry. When a whole family of pumps is considered, the designer provides the customer the possibility of choosing a pump with characteristics that are intermediate among the ones of the pumps already in the catalogue. Then, a baseline configuration is not available and the design starts from scratch. During the design, some objective functions are associated to the different pumps and correspond to measures of performance of the pump or of the manufactural constraints for the pump to be built. Their evaluations are indeed crucial.

Nowadays, coping with that is in principle possible evaluating the performance objective functions through Computational Fluid Dynamics (CFD) analysis (see [101]), due to the exponential increase of computational power, but these calculations are based on the solution of partial differential equations and can result in a computationally expensive process, that can take more time than the limit allowed by the production needs. In fact, even if reliability is still the most important aspect that guides the choice of the final geometry, the business competitiveness requires the design process to be as short and less expensive as possible, creating the need of meeting all customer requirements by accepting a compromise between reliability, low-cost manufacturing and high aerodynamic efficiency.

The research in this field is currently active, providing a wide range of different strategies for the end user to handle the optimisation procedure of a machine component. The most common techniques in this field are: gradient methods, methods based on the response surface approximation, e.g. Artificial Neural Networks (ANN), Support Vector Machines (SVM), Design of Experiment (DOE) methods, exploratory techniques, e.g. Genetic Algorithms (GA), simulated annealing, particle swarm optimisation algorithms, adjoint methods or a combination of these and have been investigated in a series of research papers, among others [101, 109, 25, 46, 61, 85, 121].

The challenge of all these frameworks and future developments is to make the resolution of the underlying dynamics needed for performance evaluation compatible with the production needs, designing a set of mathematical tools which can help evaluating the performance in a faster way, requiring less CFD evaluations compared to usual simulations. The aim is to gain a possible increase (or at least a no significant loss) of the prediction accuracy and no significant increase (in the worst-case) of the overall computational cost.

A key point is then to preserve a data-driven approach, in order to let the scheme rely only on the system output, without the necessity of any information regarding the model or equations used. In this light, an idea could be employing machine learning frameworks in the crucial phases of the parametric design, aimed at predicting the value of performance every time that such a pipeline requires an approximate evaluation of the performance on a new point of the design space (i.e., on an untested parametric pump). In this context, methods based on the response surface have reached a good level of maturity and represent a good compromise in terms of time consumption and prediction accuracy. Regression meta-models are employed to predict values of the functions describing the component performance and to build the response surface, in order to reduce the number of CFD computations required and speed up the overall optimisation process (see, [48, 110, 109]).

Generally (see [90]), a new design starts from scratch and relies on a quick and flexible design tool, capable of describing in a continuous manner the parametrisation of the pumps of a given family. The design space investigated to meet customer requirements becomes really vast, so that a large number of points is required to cover it adequately, and a very large number of computations is needed to accurately train the meta-model. Moreover, it is difficult to estimate the required number of computations, as it will be higher than what could be expected by the designer. In fact, no matter how robust the tool is, it is extremely difficult to take into account a priori all the manufacturing or geometrical constraints.

Thus, a designer could experience that many of the analysed geometries result in unfeasible solutions from a manufacturing point of view, leading to pumps that cannot be produced in practice. In the following, these geometries will be called unfeasible, while all the others will be called feasible. As an outcome, the resulting training set will be strongly unbalanced, in the sense that the unfeasible geometries are generally expected to be many more than the feasible ones. Consequently, the use of such a database to train the meta-model could lead to poor accuracy, or even to failure, of the meta-model training. As a result, the number of computations necessary to generate a suitably performing database, with enough feasible features, can increase exponentially, and the time needed to perform the computations could thus be too much high compared to the production needs.

Before outlining the specific design process, following the framework in [104], let us consider a brief subsection about the historical background on centrifugal pumps, their composition and main characterisation.

7.3.1 Centrifugal pumps: historical background and characterisation

The first centrifugal pump, in the modern sense of the word, dates back to the late 17-th century, when Thomas Savery, a British engineer, Denis Papin, a French physicist, and Thomas Newcomen, a British blacksmith and inventor, developed a water pump that used the steam to power the pump piston. This pump was firstly used to pump the water out of mines.

The first centrifugal impeller, whose origin is attributed to Denis Papin (1689), was characterised by straight vanes. Only after 150 years, following Papin's theory, Combs published a paper in which he presented curved vanes and the effect of the curvature in the design of impellers. The introduction of proper volute casing is then due to W. H. Andrews in 1839, who also was the first to make use of a fully shrouded impeller. In the same year, W. H. Johnson constructed the first three-stage centrifugal pump (see [62]).

The most important feature of these pumps is that the compression derives from the centrifugal effect of the impeller. In fact, the impeller moves the fluid transforming the mechanical energy from an external motor into kinetic energy and, then, into pressure. Such a motor can be electrical, steam powered or an internal combustion engine (ICE).

We now list the essential components of a centrifugal pump.

- Impeller: it is the mobile part, with the function of transforming energy into pressure. There are different types of impellers, especially based on the presence or the absence of the covers and shroud. We distinguish the following three different types.
    - Closed: the impeller has radial vanes that are enclosed by two discs, named "shrouds".
    - Semi-open: this type of impeller is generally more efficient due to the elimination of the disk friction from the front shroud and a preferred choice when the liquid used may contain suspended particles or fibres.
    - Open: in this case the impeller can have three different types of back shrouded. The first one is the fully scalloped open impeller, whose reduction of the back shroud decreases massively the axial thrust, caused by the hydraulic pressure. The second is the partially open impeller that has higher efficiency and head characteristics, and the last is the fully back shrouded impeller.
- Volute (or pump casing): it is the channel at the impeller outlet that guides the uncompressible flow, firstly in the suction and then in the delivery. Even this component can have different shapes, as listed below.
    - Vaneless guide ring: it consists of two smooth discs and it is used in pumps where the liquid speed is low, as well as its heads. For large heads, the outlet diameter of the ring would become too larger and thus it may be unpractical.

  - Concentric casing: this structure is used with single stage pumps or the last stage of multistage pumps. Since the casing and impeller centers are the same, the flow pressure will not increase in the casing but only in the volutes.
  - Volute casing: the main peculiarity of this casing is that it is eccentric. In fact, as mentioned above, a part of the kinetic energy is converted into pressure head inside the casing. This kind of volute can be used as single or double volute casing.
  - Vaned diffuser ring: it consists of the symmetrical installation of several vanes, gradually widening, in the pump casing. The number of diffuser vanes must be higher than the impeller vanes, otherwise pressure increase will not take place. At the same time, if the diffuser vanes are too many, this will cause a pressure drop.

- Shaft: it is the key element of the rotary movement of the impeller, through which it is connected to the motor.

To be specific, this section considers the parametric design of the components of a whole family of pumps with horizontal suction duct, single-shaft centrifugal impeller, vertical discharge diffuser and volute, at a wide range of specific speeds, similar to that depicted on the left-hand side of Figure 7.22 on page 192.

Following [109], each centrifugal pump can be identified by a set of d_a features (also addressed as the degree of freedoms of the pump), related to the geometrical shape and to some performance parameters of the pump. These components are compactly stored in a vector (the feature vector). The choice of these features depends on the specific problem at hand. We here report the main of them for the datasets we are going to consider in Subsection 7.3.3. As suggested by [109], a three-dimensional cylindrical coordinate system (x, r, θ) is used for the blade description and parametrisation, whose x-axis corresponds to the pump shaft, while the r-axis and θ-axis refer to the radial and angular position, respectively.

- Characterisation of the impeller.
  - Main dimensions: the exit diameter, the maximum axial size, the outlet width, the inlet shroud and hub diameters.
  - Meridional channel layout: the shroud and hub contours are described in the (x, r) plane.
  - Blade turning angle: the camber line of a single section is parametrised in terms of the turning angle along the meridional abscissa and takes back to azimuthal angle for building the blade in three dimensions.
  - Blade thickness: the thickness distribution is described along the curvilinear abscissa along the blade.
  - Stacking law: it is determined by a B-spline that describes the curve of the azimuthal angle along the span.
- Characterisation of the volute.
  - Main dimensions: the cut water diameter, the inlet width and the tongue thickness.
  - Volute type: the section type along the (x, r) plane; it can be trapezoidal, rectangular or circular.
  - Area: the distribution area from the inlet to the outlet of the given volute.

These are the main parameters, used to define the geometry of the centrifugal pump components.

In addition, a fluid dynamical parameter that strongly affects the pumps efficiency is considered: the back pressure. It is defined as the resistance or the force opposing the desired flow through pipes, due to friction loss and pressure drop. This effect is caused also by different high or low pressure regions than intended, which implicate a reduction of the discharge.

7.3.2 Pipelines for the parametric design: an industrial application

The industrial application considered in this section is here presented. It concerns the parametric design of a whole family of centrifugal pumps following the pipeline in [104, 48], which is based on coupling a geometry parameterisation tool, CFD computations for solving the Reynolds Averaged Navier-Stokes (RANS) equations and (feed-forward) ANN as a regression meta-model (see, [72]). It is assumed that the machine performance is evaluated through h scalar performance functions: $pf_1, ..., pf_h$ and $pf = (pf_1, ..., pf_h)^\top \in \mathbb{R}^h$.

The first phase of the procedure regards the ANN training and is described below. In Phase 1, h ANN models, one for each performance function, are trained and will be used in Phase 2 to build the response surface.

Pipeline 7.3.1: Parametric design of a family of turbomachinery components, coupling CFD and ANN.
Phase 1: ANN training.

Step 1: Geometry parametrisation. Choose n parameters (degrees of freedom) to describe the machine geometry, such that the i-th machine will be identified by a vector $a_i = (a_i(1), ..., a_i(n))^\top \in \mathbb{R}^n$ (feature or sample vector).

Step 2: Sampling of the design space. Taking into account the range of variation of each parameter, the design space is built. Assuming that for each machine i, $a_i^{\min}(k) \leq a_i(k) \leq a_i^{\max}(k)$, for $k \in \{1, ..., n\}$, the resulting space is defined as

$$\mathscr{L} = [a_i^{\min}(1), a_i^{\max}(1)] \times \cdots \times [a_i^{\min}(n), a_i^{\max}(n)] \subseteq \mathbb{R}^n.$$

The design space is randomly sampled to generate a dataset $\mathscr{D}_0 = \{a_1, ..., a_m\}$.

Step 3: CFD simulations. CFD computations are performed on $\mathscr{D}_0$ to split the features in the sets $\mathscr{F}'$ of feasible samples and $\mathscr{U}'$ of unfeasible samples, where usually $|\mathscr{F}'| \ll |\mathscr{U}'|$. The machine performance functions of feasible samples are evaluated: $pf(a_i) = (pf_1(a_i), ..., pf_h(a_i))^\top \in \mathbb{R}^h$ for $a_i \in \mathscr{F}'$ and a performance database $\mathscr{D}_{\mathscr{F}'}$ to be used as a training set in the next stage is built, which can be thought of as a set of pairs: $\mathscr{D}_{\mathscr{F}'} = \{(a_i, pf(a_i)) \mid a_i \in \mathscr{F}'\}$.

Step 4: ANN training. The performance database is used to train the ANN models that, learning from the examples in $\mathscr{D}_{\mathscr{F}'}$, build their own functions $\widetilde{pf}_1, ..., \widetilde{pf}_h$ that approximate the true measures of performance $pf_1, ..., pf_h$.

In Phase 2, the meta-models are then used to predict performance functions of new geometries, aiming at reducing the amount of CFD computations required for the design. On the response surface, a multi-objective algorithm is then run to find the set of optimal configurations.

Pipeline 7.3.2: Parametric design of a family of turbomachinery components, coupling CFD and ANN.
Phase 2: Research of an optimal solution set.

Step 1: Sampling of the design space. The design space is sampled again producing a new dataset $\mathscr{D}_1$.

Step 2: ANN execution. ANN models are used to predict the function $pf = (pf_1, ..., pf_h)^\top$ on all the new samples in $\mathscr{D}_1$ through the function $\widetilde{pf} = (\widetilde{pf}_1, ..., \widetilde{pf}_h)^\top$ built at Step 4 of Phase 1 (Pipeline 7.3.1), thus producing the response surface.

Step 3: Multi-objective algorithm. A multi-objective algorithm is run to find the set of optimal solutions $\mathscr{D}^*$.

Step 4: CFD validation of the optimal solutions set. The found solutions set $\mathscr{D}^*$ is validated through CFD computations to discard the unfeasible samples, arising from the sampling at Step 1.

Comments on Pipelines 7.3.1–7.3.2

There are mainly two drawbacks arising in the outlined procedure.

- The first drawback arises at Steps 2–3 of Phase 1 (Pipeline 7.3.1). Even if particular attention is dedicated to the implementation of the parameterisation tool and to correlate the tens of degrees of freedom, when sampling the design space a wide percentage of the samples in $\mathscr{D}_0$ (usually more than 70%) turn out to be unfeasible. This is an intrinsic aspect of the nature of a parametric design, in which the same geometrical parameters and ranges of variation are applied to pump geometries with very different characteristics, such as a wide range of specific speeds and manufacturing constraints. As a consequence, most of the CFD computations performed to build the training set $\mathscr{D}_0$ are useless. Therefore, to obtain an accurate prediction of the performance functions related to feasible pumps, the meta-model has to be trained on a set $\mathscr{F}' \subseteq \mathscr{D}_0$ of just feasible samples, but this has the disadvantage of requiring a too large number of CFD computations to discard the feasible from the unfeasible ones, as most of the samples will yield unfeasible geometries. Moreover, even admitting that this discarding process would be sustainable, the number of the feasible pumps in the training set $\mathscr{F}'$ could be too small for the meta-model to be effective and the number of computations necessary to regenerate a training set with enough feasible features generally increase exponentially (together with the time needed to perform CFD computations).
- Similarly, in Phase 2 (Pipeline 7.3.2), the trained meta-model is used to predict the objective functions of new randomly selected geometries. The sampling is performed within the whole design space, so the most part of the geometries provided as meta-model input $\mathscr{D}_1$ are again expected to be unfeasible. The predicted performance functions values will then be meaningless and it cannot be excluded that Step 3 of Phase 2 yields a set of optimal solutions consisting of just unfeasible samples, leading to the need of restarting the whole procedure. This outcome is generally independent of the chosen meta-model, leading to the conclusion that the considered approach may not be effective for a parametric design.

In this light, authors in [104] made the approach given in Pipelines 7.3.1–7.3.2 more practical by including the use of a binary classifier with the aim of discarding the unfeasible pumps in $\mathscr{D}_0$ and in $\mathscr{D}_1$, obtained at Step 2 of Phase 1 and at Step 1 of Phase 2, in a cheaper and faster way. This would help both to avoid a useless large amount of expensive CFD calculations to form the ANN training set and to restrict the regression process to a set mainly composed by feasible geometries, to reduce the presence of unfeasible ones in the optimal solution set, also making the gaining of the response surface at Step 3 of Phase 2 less expensive. In so doing, due to the fact that the employment of binary classification can lead to false positive and false negative predictions, the use of CFD computations is still considered to be sure of removing the possible outliners (unfeasible pumps) among the pumps classified as feasible. For such a reason, we will refer to this novel approach as "the hybrid approach".

The tool chosen as a classifier in [104] was SVM (Support Vector Machine, see [111, 53] for details), but the framework can in principle be stated independently of the specification of the classifier. Hence, the novel Pipeline 7.3.3–7.3.4 are reported addressing the classifier as BC (Binary Classifier).

Pipeline 7.3.3: Hybrid approach: Parametric design of a family of turbomachinery components, coupling CFD, ANN and BC.
Phase 1: ANN training.

Step 1: Geometry parametrisation. Choose n parameters (degrees of freedom) to describe the machine geometry, such that the i-th machine will be identified by a vector $a_i = (a_i(1), ..., a_i(n))^\top \in \mathbb{R}^n$ (feature or a sample vector).

Step 2: Sampling of the design space. Taking into account the range of variation of each parameter, the design space is built. Assuming that for each machine i, $a_i^{\min}(k) \leq a_i(k) \leq a_i^{\max}(k)$ for $k \in \{1, ..., n\}$, the resulting space is defined as

$$\mathscr{L} = [a_i^{\min}(1), a_i^{\max}(1)] \times \cdots \times [a_i^{\min}(n), a_i^{\max}(n)] \subseteq \mathbb{R}^n.$$

The design space is randomly sampled to generate a dataset $\mathscr{D}_0 = \{a_1, ..., a_m\}$.

Step 3: Binary classification. The samples in $\mathscr{D}_0$ are given in input to the binary classifier which divides them into the two classes: $\mathscr{F}$ (classified as feasible) and $\mathscr{U}$ (classified as unfeasible). Features in $\mathscr{U}$ are no longer taken into account and just those in $\mathscr{F}$ are considered in the next steps.

Step 4: CFD simulations. CFD computations are performed on the samples in $\mathscr{F}$. The outliers are eliminated obtaining a subset $\mathscr{F}'$ of just feasible features for which the machine performance functions are evaluated: $pf(a_i) = (pf_1(a_i), ..., pf_h(a_i))^\top \in \mathbb{R}^h$ and $a_i \in \mathscr{F}'$. The performance database $\mathscr{D}_{\mathscr{F}'}$ to be used as a training set in the next stage is built, which is a set of pairs: $\mathscr{D}_{\mathscr{F}'} = \{(a_i, pf(a_i)) \mid a_i \in \mathscr{F}'\}$.

Step 5: ANN training. The performance database is used to train the ANN models that, learning from the examples in $\mathscr{D}_{\mathscr{F}'}$, build their own functions $\widetilde{pf}_1, ..., \widetilde{pf}_h$ that approximate the true measures of performance $pf_1, ..., pf_h$.

Pipeline 7.3.4: Hybrid approach: Parametric design of a family of turbomachinery components, coupling CFD, ANN and BC.
Phase 2: Research of an optimal solution set.

Step 1: Sampling of the design space. The design space is sampled again producing a new dataset $\mathscr{D}_1$.

Step 2: Binary classification. Samples in $\mathscr{D}_1$ are given in input to the binary classifier which divides them into the two classes $\mathscr{F}''$ (classified as fasible) and $\mathscr{U}''$ (classified as unfeasible). Features in $\mathscr{U}''$ are no longer taken into account and just those in $\mathscr{F}''$ are considered in the next step.

Step 3: ANN execution. The ANN model is used to predict $pf = (pf_1, ..., pf_h)^\top$ of the samples in $\mathscr{F}''$.

Step 4: Multi-objective algorithm. A multi-objective algorithm is run to find the set of optimal solutions $\mathscr{D}^*$.

Step 5: CFD validation of the optimal solutions set. The found solutions set $\mathscr{D}^*$ is validated through CFD computations to eliminate possible outliers (unfeasible samples).

Comments on Pipelines 7.3.3–7.3.4

The classification procedures added at Step 3 of Phase 1 and at Step 2 of Phase 2 are intended to mitigate the drawbacks sketched in the comments after Pipelines 7.3.1–7.3.2. Its benefit is summarised below.

- The first classification allows the CFD computations performed at Step 4 of Phase 1 to be restricted to only the set $\mathscr{F}$ of features classified as feasible by the classifier in use, with the aim of eliminating outliers (i.e. false positives). This produces a great saving in CFD computations, as usually $|\mathscr{F}| \ll |\mathscr{D}_0|$ and CFD performed on samples in $\mathscr{U}$ would be of no use.
- The second process allows function values to be predicted on only samples in $\mathscr{F}''$, collecting pumps classified as feasible and, hopefully, mainly composed by feasible features. However, some outliers could still be present in the set, so Step 5 of Phase 2 is still necessary, but, due to the detection already performed by the classifier, it is expected to be much less expensive than the corresponding Step 4 of Phase 2 (Pipeline 7.3.2).

7.3.3 Numerical resolution of the classification task

In this last subsection we consider the practical training of a classifier to be used at Step 3 of Phase 1 and at Step 2 of Phase 2 (Pipelines 7.3.3–7.3.4) to discard the unfeasible (unmanufacturable) configurations from the feasible (manufacturable) ones. This issue has been successfully addressed in [104] on a set of three different datasets. We here consider the two of them listed in Table 7.13 (page 169), gently provided by the TRAF-group of the Department of Industrial Engineering of the "Università degli Studi di Firenze". In addition to the number of features d_a, the number of training samples N, the number of testing samples N_T, also the approximate ratio ρ_u between unfeasible and feasible samples in the training set are reported.

As it can be easily noticed by looking at Table 7.13, the two datasets are characterised by a different proportion between unfeasible and feasible samples in the training set.

Dataset	Training N	d_a	Testing N_T	ρ_u
PUMPSU	50000	40	32000	$3{:}1$
PUMPSVU	61600	44	15400	$7{:}1$

Table 7.13: Pumps datasets. Size of the training set (N), problem dimension d_a (number of features), size of the testing set (N_T) and the ratio ρ_u between unfeasible and feasible pumps in the training set.

We will hereafter refer to datasets in which the number of unfeasible and feasible pumps is almost the same ($\rho_u \simeq 1$) as *balanced*, while datasets where the number of unfeasible pumps is much larger than the number of feasible ones as *unbalanced*. Hence, the two datasets in Table 7.13 can be both considered as unbalanced (PUMPSU stands for PUMPS Unbalanced, PUMPSVU stands for PUMPS Very Unbalanced).

In presence of unbalanced training datasets, the classifier can have very little information about the minority class and this is expected to result into major difficulties in learning the recognition of such features on the testing set, making predictions on feasible configurations more uncertain. It is thus easy to have many feasible features misclassified. For the testing dataset $\{(\overline{a}_i, \overline{y}_i)\}_{i=1}^{N_T}$ the exact labels $\overline{y}_i$ are known, so the errors in such predictions can be computed. Let $\mathscr{C} \in \mathbb{R}^{N_T}$ be the vector with the correct features classification, i.e. $\mathscr{C}(i) = 1$ if the i-th feature $\overline{a}_i$ in the testing set is feasible ($\overline{y}_i = 1$) and $\mathscr{C}(i) = 0$ if $\overline{a}_i$ is unfeasible ($\overline{y}_i = 0$), $i \in \{1, ..., N_T\}$, and $P\mathscr{C} \in \mathbb{R}^{N_T}$ the result of the classification process, i.e. $P\mathscr{C}(i)$ is the predicted value of $\mathscr{C}(i)$, $i \in \{1, ..., N_T\}$.

For each feature, the following four different situations can occur.

- TP = *True Positive*: $\mathscr{C}(i) = 1$, $P\mathscr{C}(i) = 1$, the feature is feasible and is correctly classified;
- FP = *False Positive*: $\mathscr{C}(i) = 0$, $P\mathscr{C}(i) = 1$, the feature is unfeasible but is misinterpreted, since it is classified as feasible;
- TN = *True Negative*: $\mathscr{C}(i) = 0$, $P\mathscr{C}(i) = 0$, the feature is unfeasible and is correctly classified;
- FN = *False Negative*: $\mathscr{C}(i) = 1$, $P\mathscr{C}(i) = 0$, the feature is feasible but is misinterpreted, since it is classified as unfeasible.

It is easy to note that when the training dataset is balanced, the probability of misinterpreting a feature is the same for the two classes (the probability of a false positive is expected to be more or less the same as that of a false negative), then the fraction of features correctly classified (binary classification rate),

$$ACC = \frac{TP + TN}{TN + TP + FN + FP} \tag{7.22}$$

represents a good measure of accuracy. On the other hand, when dealing with training datasets dominated by the fraction of unfeasible features, the probability of getting a false negative can be much higher than the probability of a false positive. Therefore, the ratio (7.22) is not so meaningful, being

$$ACC \simeq \frac{TN}{TN + FN}$$

and thus it is possible to achieve a high accuracy level even if all the features in the minority class are misinterpreted.

For this reason, a proper choice of the parameters to evaluate the performance of the classification process has to be made, taking into account that the datasets can be unbalanced, as it is the case in the parametric design of centrifugal pumps. The two sets of parameters reported below and considered for the numerical tests are the one in [104, 98]. The first set are the TP, FP, TN and FN in percentage form and are listed below.

- $TPR = \frac{TP}{TP+FN}$ *True Positive Rate* or sensitivity or recall, fraction of positive samples correctly classified over all positive samples available in the test;
- $FPR = \frac{FP}{TN+FP}$ *False Positive Rate*, fraction of unfeasible features misinterpreted over all negative samples available in the test;
- $TNR = \frac{TN}{TN+FP}$ *True Negative Rate* or specificity, fraction of negative samples correctly classified over all negative samples available in the test;
- $FNR = \frac{FN}{TP+FN}$ *False Negative Rate*, fraction of feasible features misinterpreted over all positive samples available in the test.

The second set of parameters that is of interest for a designer is as follows.

- $PPV = \frac{TP}{TP+FP}$ *Positive Predictive Value*, the fraction of true positives in the set of features classified as positive;
- $FDR = \frac{FP}{TP+FP}$ *False Discovery Rate*, the fraction of false positives in the set of features classified as positive;
- $NPV = \frac{TN}{TN+FN}$ *Negative Predictive Value*, the fraction of true negatives in the set of features classified as negative;
- $FOR = \frac{FN}{TN+FN}$ *False Omission Rate*, the fraction of false negatives in the set of features classified as negative.

By definition, the couples (TPR, FNR), (TNR, FPR), (PPV, FDR) and (NPV, FOR) sum up to one, so only one parameter for each couple can be shown.

We highlight that, in addition to ACC in (7.22), TPR and FPR, we will concentrate on the following to interpret the numerical resolution of the classification problem, due to their significant meaning in the context of parametric pumps design.

- PPV: it interests the designer as it gives a measure of the quality of set $\mathscr{F}$, telling how many features are actually feasible, meaning that $FDR = 1 - PPV$ indicates the percentage of useless CFD computations performed at Step 4 of Phase 1 (Pipeline 7.3.3) and how many outliers could be part of the optimal solutions set at Steps 4–5 of Phase 2 (Pipeline 7.3.4).
- FOR: it gives the percentage of feasible features that are lost owing to the classification process, as they are incorrectly considered as unfeasible.

We will refer to the sequence of ratios TPR, FPR, PPV, FDR, FOR on the testing set as the *performance rates*.

As already mentioned, the classifier used in Pipelines 7.3.3–7.3.4 by the authors in [104] was the SVM. In this case, by making strongly use of a careful setting of the method free parameters, their nonlinear classification task on the two datasets in Table 7.13 gives, in the best case, the results in Table 7.14 on the next page (see [104], Tables 2–3).

As one may easily see, a large percentage of true positive is achieved in correspondence of a relatively small percentage of false positive (see the values of TPR and FPR). The values of FDR, in turn, indicate that the fraction of false positives within the set of of features classified as positive is still not neglectable, specially in the PUMPSVU dataset,

Dataset	TPR	FPR	PPV	FDR	FOR
PUMPSU	78.3%	23.0%	50.8%	49.2%	7.9%
PUMPSVU	83.0%	19.8%	39.4%	60.6%	3.1%

Table 7.14: Pumps datasets. Performance rates (testing set) via SVM in [104].

where only 39.4% of the features classified as positive corresponds to true positives. This will lead to a remaining effort in terms of CFD evaluations when considering Step 4 in Phase 1 and Steps 4–5 in Phase 2 of Pipelines 7.3.3–7.3.4, to be sure of eliminating the remaining false positives. Last, the FOR rate is particularly low on the PUMPSVU dataset, meaning that just 3.1% of the feasible features are lost during the classification process, due to their wrong classification, while this percentage is more than doubled for the PUMPSU dataset.

To sum up, the SVM strategy seems to be particularly capable of detecting true positives and this feature appears to be quite robust to the unbalance of the training set. To our knowledge, this is currently a characteristic that has not been overcome by other tested methods on this specific class of centrifugal pumps datasets. On the other hand, some drawbacks are still in place. For instance, one could wonder whether a larger gain in terms of CFD savings measured by FDR, hopefully with a lower loss of feasible feature among the classification process, i.e. with lower values of FOR. Moreover, the implementation of the code considered in [104] is far from being smart and its execution is generally expensive in terms of computational time. This could be a limit when dealing with the needs of the overall industrial process. Therefore, alternative resolutions based on innovative methods could be of interest, at least when dealing with moderately unbalanced datasets as the PUMPSU.

A first effort in this direction was made by the first author in [104], using two second-order strategies based on the Levenberg-Marquardt method, namely the FLM and the LMN method in [105]. We report the resulting set of performance rates on the testing set in Table 7.15 below.

Method	TPR	FPR	PPV	FDR	FOR
FLM [105]	76%	20%	36%	64%	5%
LMN [105]	77%	23%	34%	66%	4%

Table 7.15: PUMPSU dataset. Performance rates (testing set) via the FLM and the LMN methods in [105].

The outcomes are comparable with the one in Table 7.14 for the PUMPSU dataset for what concerns the TPR and the FPR and a little reduction of FOR is achieved, but show a degradation in terms of PPV and FDR. Hence, no improvements are obtained when looking at the use of the classifier in Pipelines 7.3.3-7.3.4 in terms of possible gain in CFD computations.

Our contribution attempts to take a step further about this practical aspect. To this purpose, we solve the nonconvex minimisation problem (7.14) (with σ defined in (7.9)), consisting in a least-squares minimisation with sigmoid prediction in which labels 1 correspond to feasible pumps, while labels 0 denote unfeasible configurations. For the training and testing phase, the datasets in Table 7.13 have been considered. In order to give an innovative investigation of the problem resolution, the AR1DA algorithm (see Algorithm 11) and the *ARC-Dynamic* (see Subsection 7.2.2) have been tested. The implementation follows the specifications in Subsection 7.2.4, with the only difference of considering $\kappa = 10^{-6}$, the stopping criterion (7.17) for AR1DA and (7.16) for *ARC-Dynamic* with

$\epsilon_1 = 10^{-4}$, in order to compare with the results in Table 7.15.

To present the results we first consider the PUMPSU dataset and plot in Figure 7.23 on page 193 the testing losses computed by AR$1$DA and *ARC-Dynamic* with respect to the cost measure CM defined in Subsection 7.2.3 and in Figure 7.24 the sample size (percentage) used by AR$1$DA for estimating function and gradient against CM, while the performance rates on the testing set are reported in Table 7.16 below (mean values over 20 runs are considered).

Method	TPR	FPR	PPV	FDR	FOR	ACC
ARC-Dynamic	73.2%	30.9%	64.4%	35.6%	22.9%	70.8%
AR$1$DA	72.8%	31.3%	64.1%	35.9%	23.3%	70.5%

Table 7.16: PUMPSU dataset. Performance rates and binary classification rate (ACC) on the testing set via *ARC-Dynamic* and AR$1$DA; mean values over 20 runs.

We highlight that the accuracy in the overall binary classification is around 71%, with TPR values close to the ones obtained by the methods in Table 7.15. The Levenberg-Marquardt methods considered in Table 7.15 produce preferable FPR and FOR rates, but our experimentation significantly lowers the percentage of useless CFD evaluations involved in the design process, producing a cost saving of at least 44% (compare the FDR values in Tables 7.15–7.16). Moreover, the computational time switches from minutes to seconds, since both the *ARC-Dynamic* and AR$1$DA algorithm terminate in at most 10 seconds, while the elapsed time reported in [105] for the results in Table 7.15 via the FLM method is 15 minutes and all the algorithms in Tables 7.15–7.16 result in a very smart implementation when comparing to SVM.

Interestingly, Figure 7.24 shows that the sample size for gradient approximation is almost always below 8% of the full sample size, without never reaching it, and it remains much lower than the corresponding sample size for evaluating the function estimate, according to the behaviour of the method in the numerical tests of Subsection 7.2.4.1.

We also notice that the performances of AR$1$DA and *ARC-Dynamic* are very close, even thought *ARC-Dynamic* outperforms AR$1$DA as it can be also seen from Figure 7.23, comparing the decrease of the testing losses generated by the two methods.

Analogous, but less favorable, performance and accuracy rates are obtained when solving the problem with other two well-known first-order methods, the Mini-Batch Gradient Descent (MBGD) method and the Stochastic Variance Reduced Gradient (SVRG) method. The null vector is still considered as the starting point and, following [27, 78, 76], the step size $5 \cdot 10^{-3}$ and the mini-batch size 2^7 is considered for MBGD, while the implementation of SVRG considers a step size of 10^{-1} and $5N$ inner iterations for each outer iteration. The performance rates and the accuracy in binary classification when (7.17) is reached are reported in Table 7.17.

Method	TPR	FPR	PPV	FDR	FOR	ACC
MBGD	71.7%	31.4%	63.6%	36.4%	24.0%	69.9%
SVRG	72.5%	32.1%	63.4%	36.6%	23.7%	69.9%

Table 7.17: PUMPSU dataset. Performance rates and binary accuracy (ACC) on the testing set via MBGD and SVRG; mean values over 20 runs.

Interestingly, the best performing of *ARC-Dynamic* with respect to first-order methods is even much more evident when dealing with the very unbalanced dataset PUMPSVU, as it can bee seen below from Table 7.18.

Method	TPR	FPR	PPV	FDR	FOR	ACC
ARC-Dynamic	20.6%	2.0%	59.3%	40.7%	10.0%	88.7%
AR1DA, MBGD, SVR	0%	0%	undefined	undefined	12.1%	87.9%

Table 7.18: PUMPSVU dataset. Performance rates and binary accuracy (ACC) on the testing set via *ARC-Dynamic* and the set of first-order methods: AR1DA, MBGD and SVR; mean values over 20 runs.

With this dataset, the main difficulty in the classification process is that the set of data the *ARC-Dynamic* algorithm has to be trained on, and that have to be classified, is very unbalanced, owing to the strong predominance of unfeasible geometries over feasible ones. Consequently, as might be expected, the performance rates in Table 7.18 computed by *ARC-Dynamic* show that the method is very affordable in the recognition of unfeasible pumps (the FPR drops to 2% improving also the corresponding value given in Table 7.14 by the SVM), but suffers from an intrinsic deficit in learning the classification of feasible pumps.

This characteristic is also evident looking at the decrease of the training and the testing loss in Figure 7.23, since at termination both the curves are totally flat and, hence, the learning process do not manage to go further. Accordingly, this is an example in which the high accuracy (ACC value) in the overall binary classification is not so representative, being just a reflection of the fact that the number of unfeasible pumps in the training set, that are correctly classified by *ARC-Dynamic*, is much larger than the number of feasible one (recall that $\rho_u \simeq 7$).

Remark 10 (Effectiveness of second-order curvature information)**.** The difficulty of detecting feasible pumps is extremely more marked when solving the problem with the first-order methods in Table 7.18, where all the pumps in the testing set are classifies as unfeasible (thus PPV and FDR are undefined). Remarkably, this feature does not change also allowing full gradient and function evaluations in AR1DA and exact gradient evaluations in MBGD, SVRG, so *this seems a real-life example in which second-order information play a crucial role.*

Nevertheless, even if the use of *ARC-Dynamic* on the PUMPSVU dataset allows a wide percentage of false positives in the classified set, the general advantage gained by the use of the classification procedure is that it lowers the ratio between unfeasible and feasible pumps in the testing set, corresponding to $\mathscr{F}'$ and $\mathscr{F}''$ in Pipelines 7.3.3–7.3.4, that has to be checked by expensive CFD computations. This new ratio is given by FDR/PPV and it is definitely lower using *ARC-Dynamic* and AR1DA with respects to the other reported methods, among both the pumps dataset at hand, as evident from Table 7.19 on the next page. In particular, the original ratio $\rho_u = 7$ between unfeasible and feasible pumps in the training set is reduced by a factor 10 using *ARC-Dynamic* on the PUMPSVU dataset, giving $FDR/PPV = 0.69$.

Dataset	ρ_u	FDR/PPV						
		SVM	FLM	LMN	MBGD	SVRG	AR1DA	*ARC-Dynamic*
PUMPSU	3	0.97	1.78	1.94	0.57	0.58	0.56	0.55
PUMPSVU	7	1.54	n.a.	n.a.	und.	und.	und.	0.69

Table 7.19: Pumps datasets. Ratio FDR/PPV computed by the considered methods on the PUMPSU and the PUMPSVU datasets. The values of ρ_u on both datasets are also reported. The notation "n.a." and "und." stand for "not available" and "undefined", respectively.

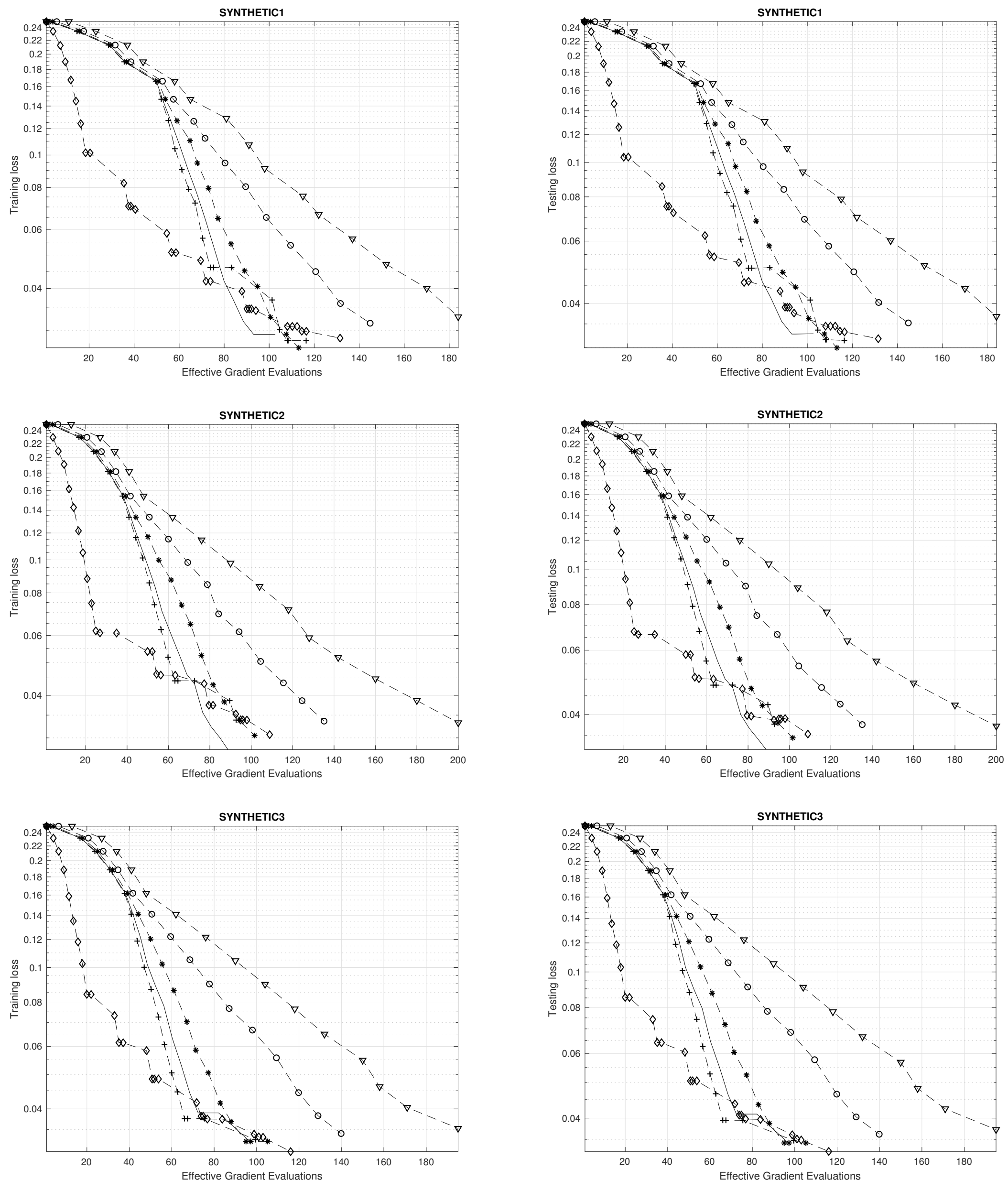


Figure 7.1: Comparison of *ARC-Dynamic* (continuous line), *ARC-Dynamic(c)* with $c = 0.5$ (dashed line with circles), $c = 0.75$ (dashed line with asterisks), $c = 1.25$ (dashed line with plus symbols), ARC-KL (dashed line with diamonds) and *ARC-Sub* (dashed line with triangles) against EGE. Each row corresponds to a different synthetic dataset. Training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis.

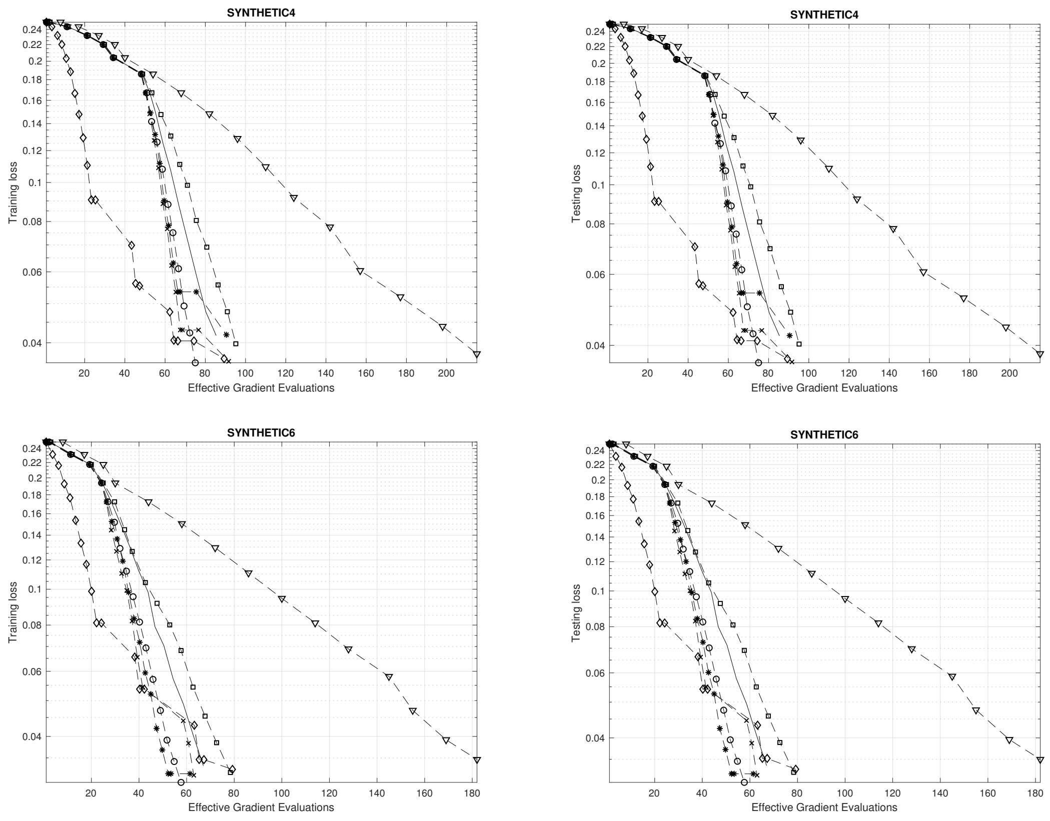
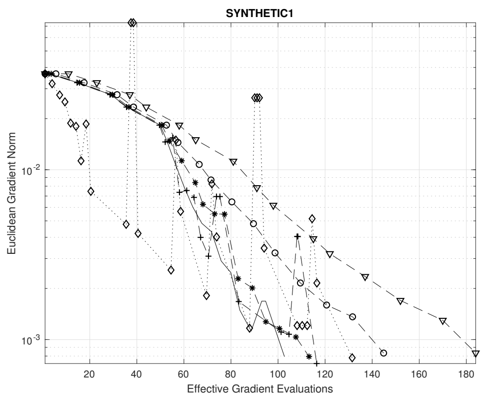
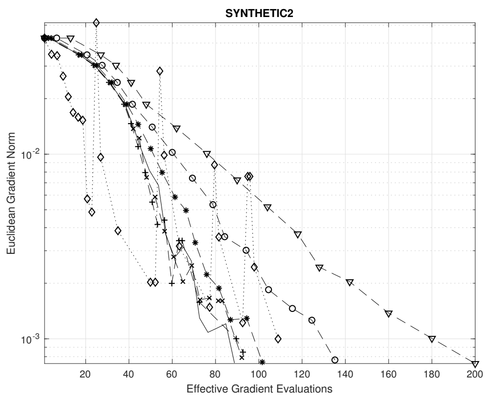
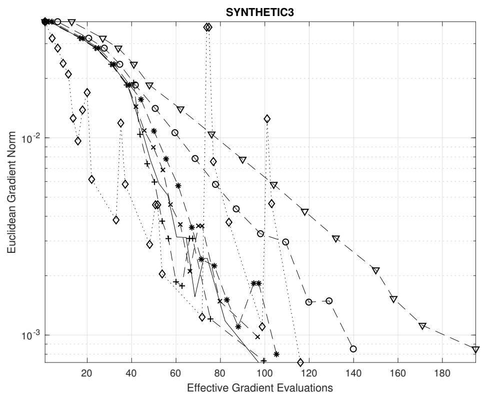
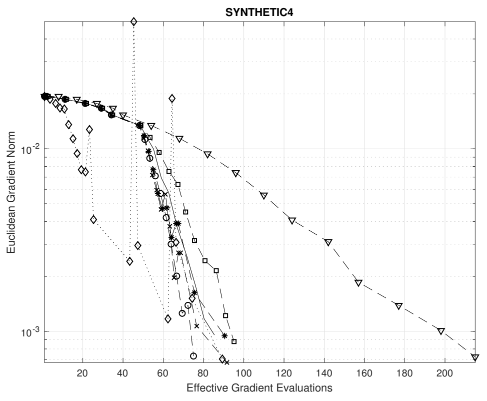


Figure 7.2: Comparison of *ARC-Dynamic* (continuous line), *ARC-Dynamic(c)* with $c = 0.25$ (dashed line with squares), $c = 0.5$ (dashed line with circles), $c = 0.75$ (dashed line with asterisks), $c = 1$ (dashed line with crosses), ARC-KL (dashed line with diamonds) and *ARC-Sub* (dashed line with triangles) against EGE. Each row corresponds to a different synthetic dataset. Training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis.









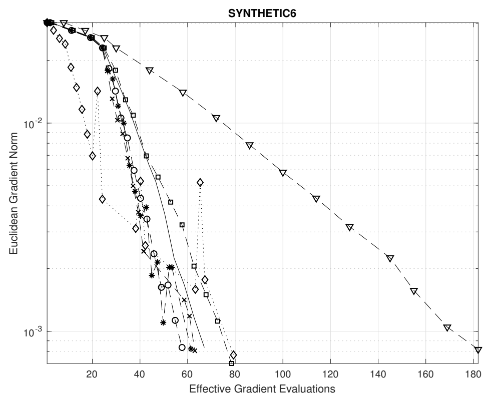


Figure 7.3: Synthetic datasets, Euclidean norm of the gradient against EGE (training set), logarithmic scale on the y axis. *ARC-Dynamic* (continuous line), *ARC-Dynamic(c)* with $c = 0.25$ (dashed line with squares), $c = 0.5$ (dashed line with circles), $c = 0.75$ (dashed line with asterisks), $c = 1$ (dashed line with crosses), $c = 1.25$ (dashed line with plus symbols), *ARC-KL* (dot line with diamonds) and *ARC-Sub* (dashed line with triangles).

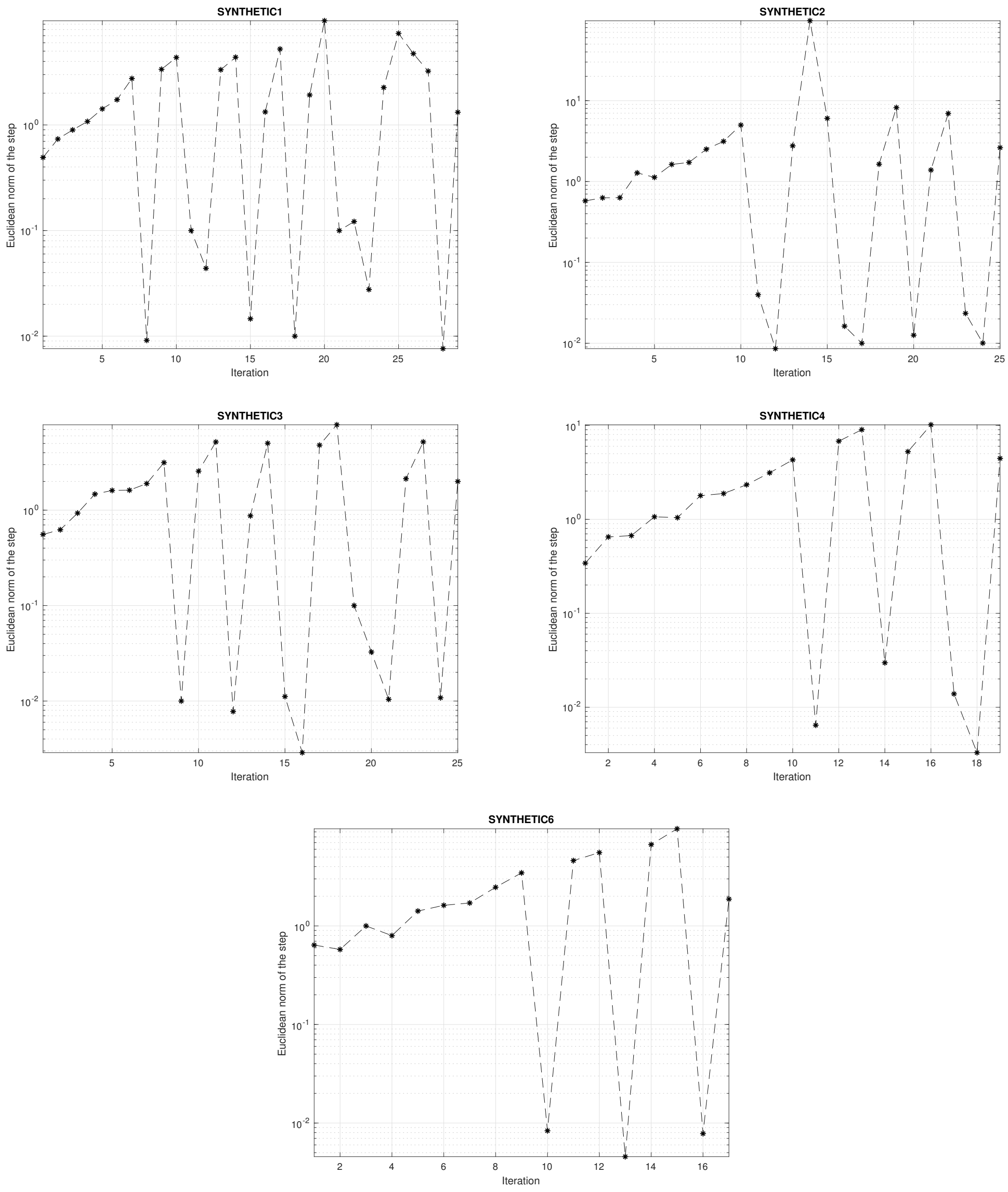


Figure 7.4: Synthetic datasets, 2-norm of the step against iterations via *ARC-KL*. Logarithmic scale on the y-axis.

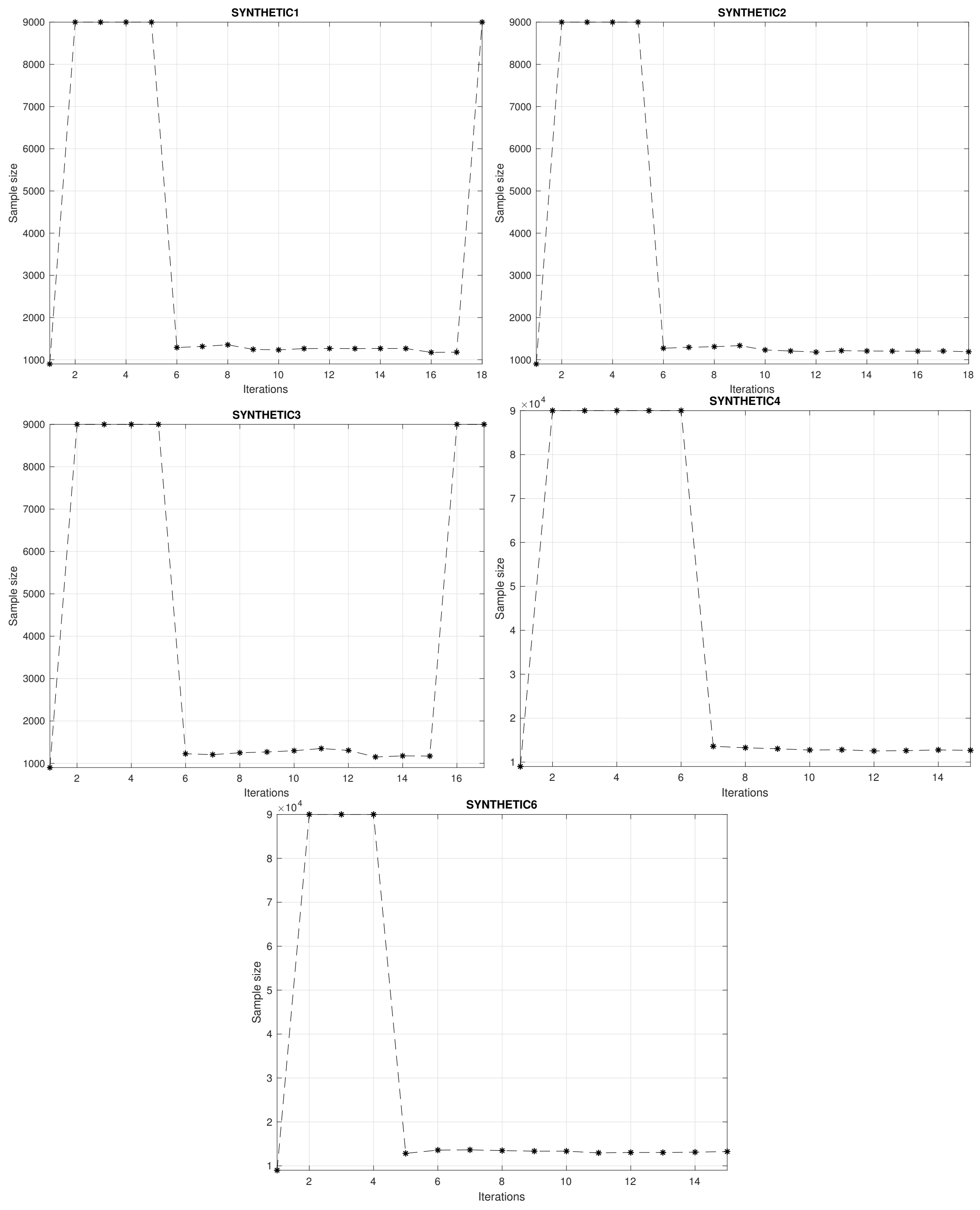


Figure 7.5: Synthetic datasets. Sample size for Hessian approximations against iterations.

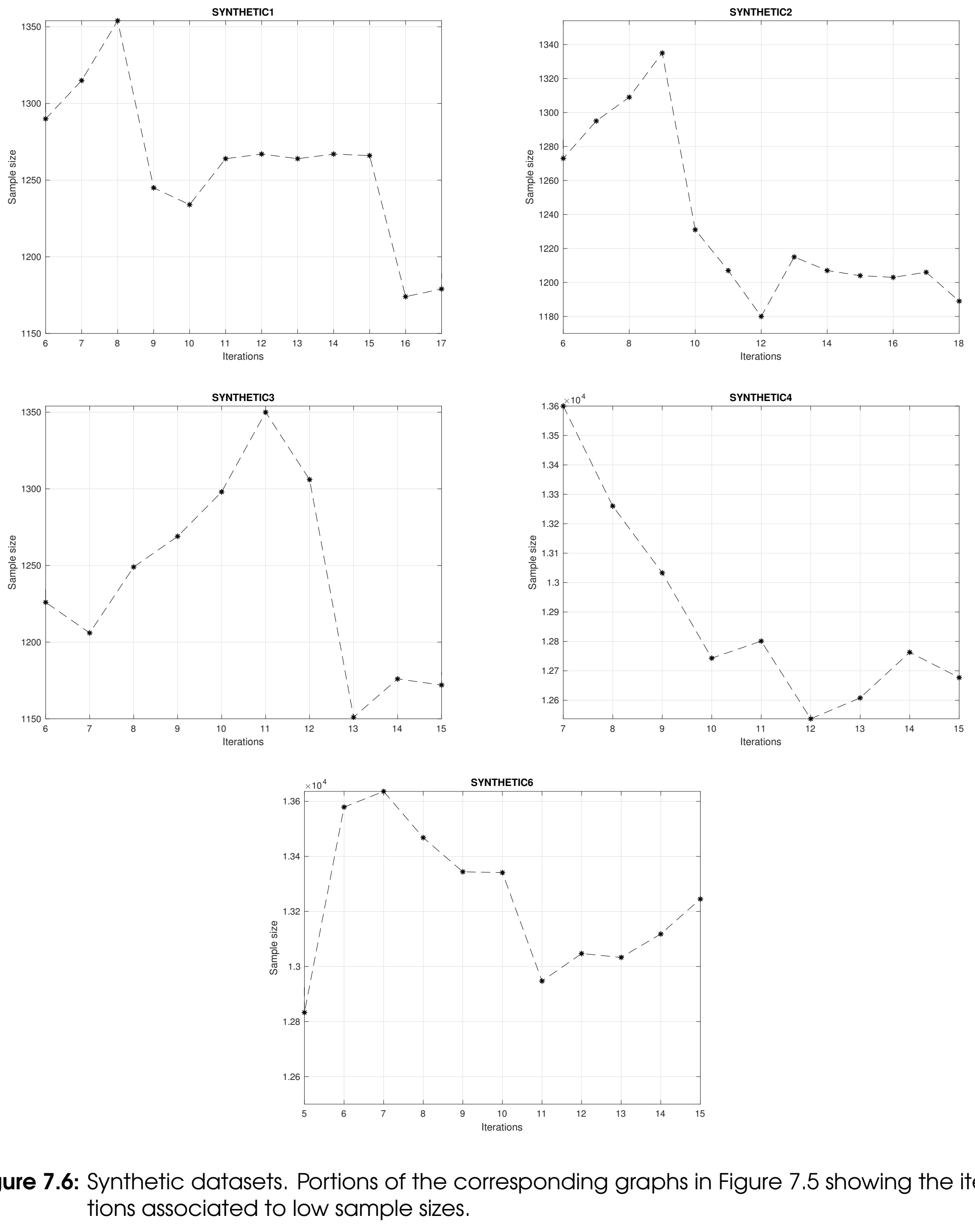


Figure 7.6: Synthetic datasets. Portions of the corresponding graphs in Figure 7.5 showing the iterations associated to low sample sizes.

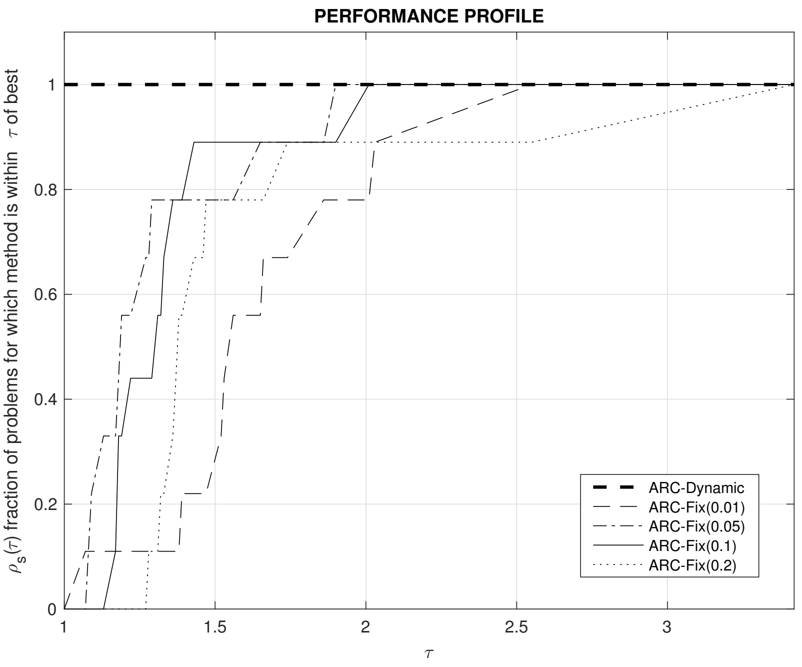


Figure 7.7: Performance profile (EGE count) on $[1, 3.4]$ for real datasets.

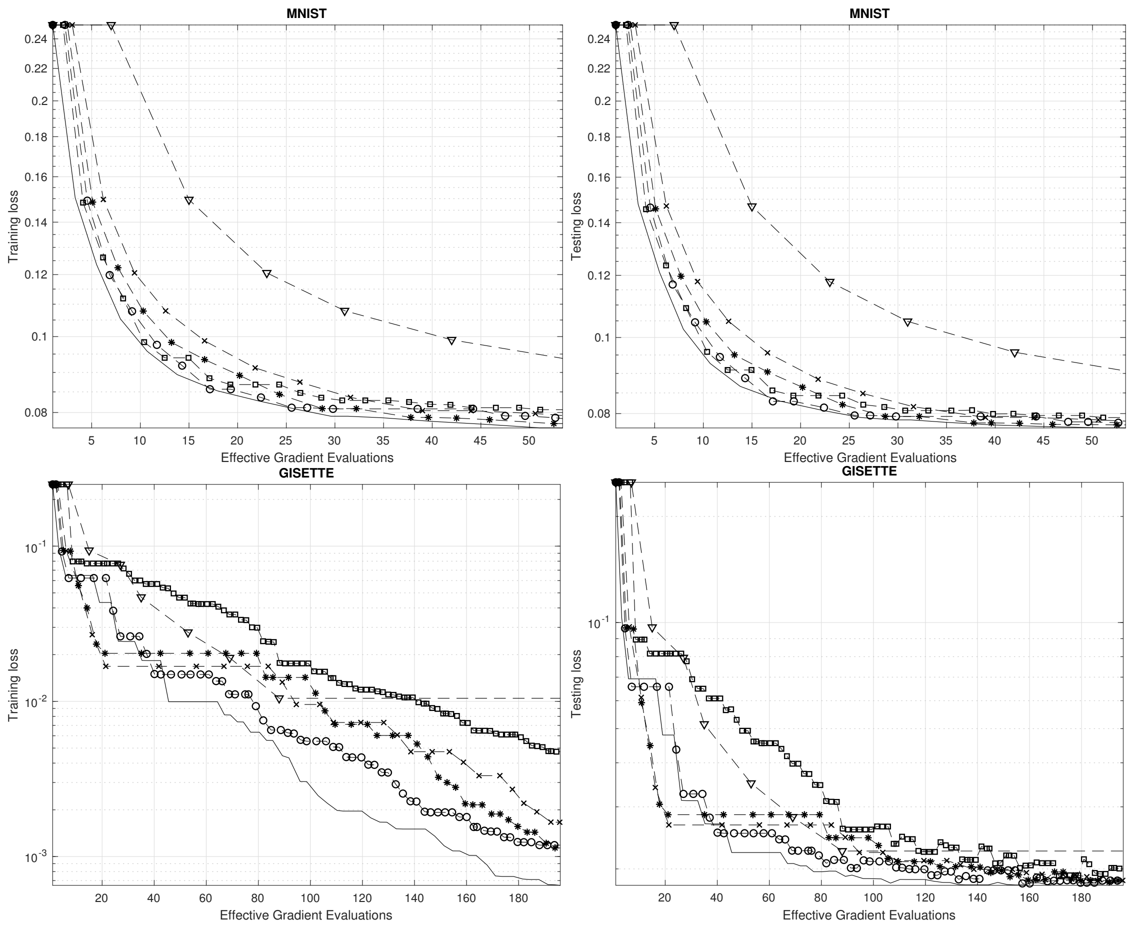


Figure 7.8: MNIST dataset (top), Gisette dataset (bottom), training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis. *ARC-Dynamic* (continuous line), *ARC-Fix(p)* with $p = 0.2$ (dashed line with crosses), $p = 0.1$ (dashed line with asterisks), $p = 0.05$ (dashed line with circles), $p = 0.01$ (dashed line with squares) and *ARC-Full* (dashed line with triangles).

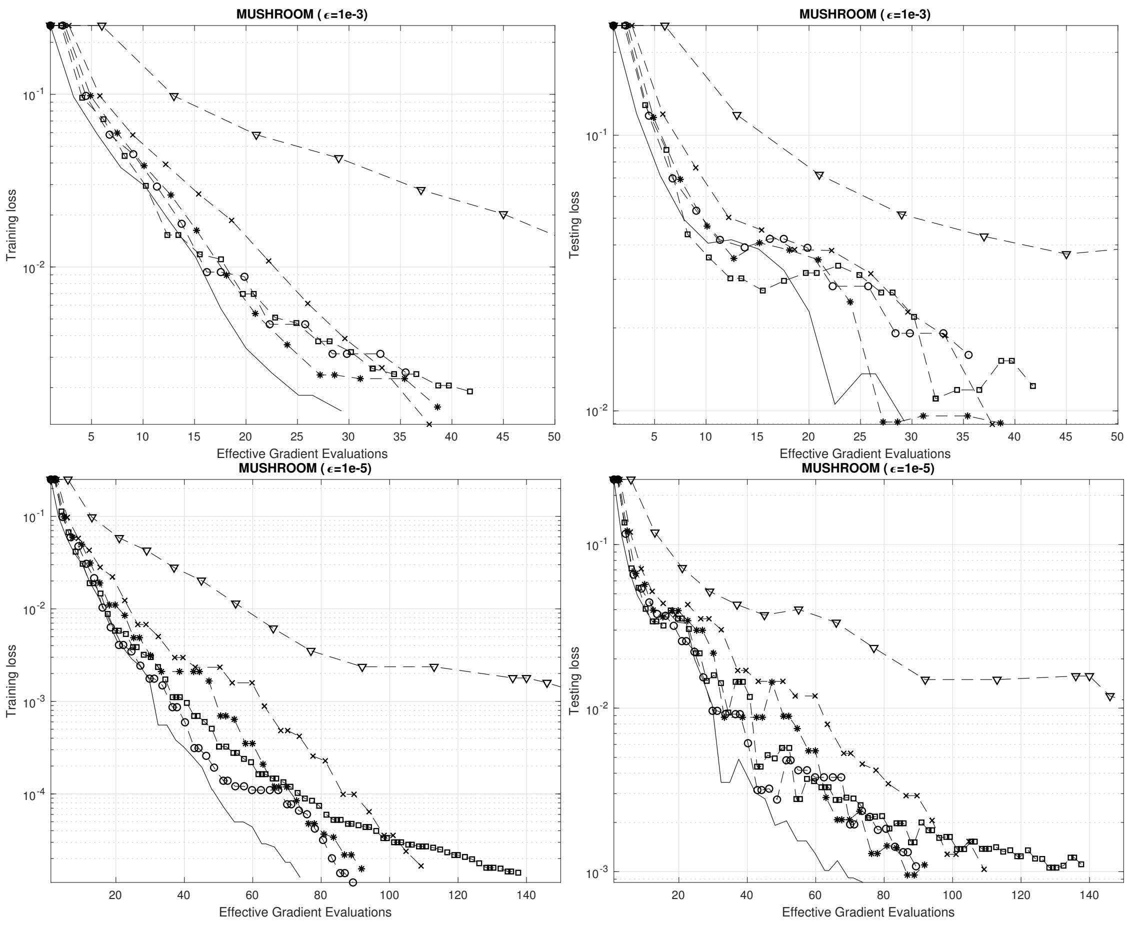


Figure 7.9: Mushroom dataset, training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis. *ARC-Dynamic* (continuous line), *ARC-Fix(p)* with $p = 0.2$ (dashed line with crosses), $p = 0.1$ (dashed line with asterisks), $p = 0.05$ (dashed line with circles), $p = 0.01$ (dashed line with squares) and *ARC-Full* (dashed line with triangles).

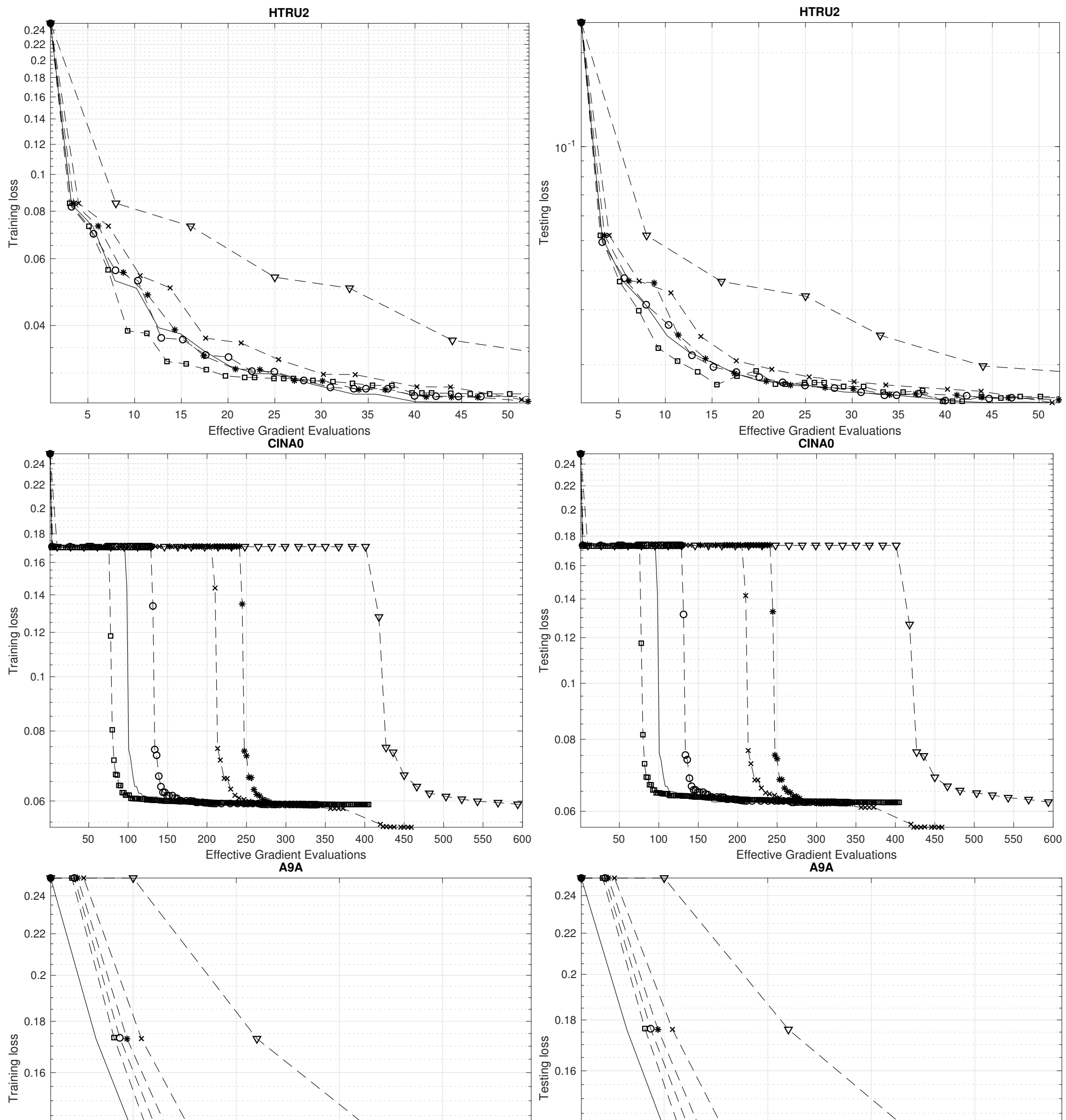


Figure 7.10: HTRU2 dataset (top), Cina0 dataset (mid), a9a dataset (bottom), training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis. *ARC-Dynamic* (continuous line), *ARC-Fix(p)* with $p = 0.2$ (dashed line with crosses), $p = 0.1$ (dashed line with asterisks), $p = 0.05$ (dashed line with circles), $p = 0.01$ (dashed line with squares) and *ARC-Full* (dashed line with triangles).

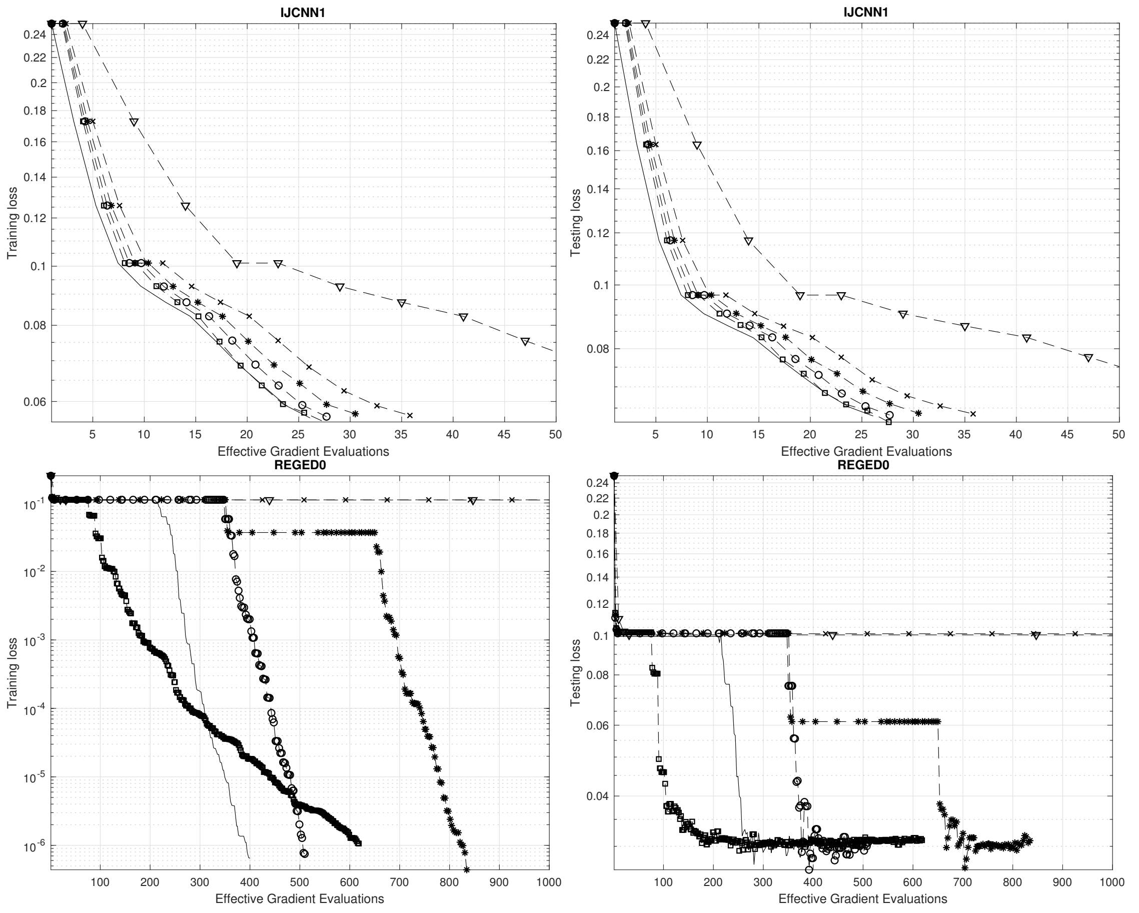


Figure 7.11: Ijcnn1 dataset (top), Reged0 dataset (bottom), training loss (left) and testing loss (right) against EGE, logarithmic scale on the y axis. *ARC-Dynamic* (continuous line), *ARC-Fix(p)* with $p = 0.2$ (dashed line with crosses), $p = 0.1$ (dashed line with asterisks), $p = 0.05$ (dashed line with circles), $p = 0.01$ (dashed line with squares) and *ARC-Full* (dashed line with triangles).

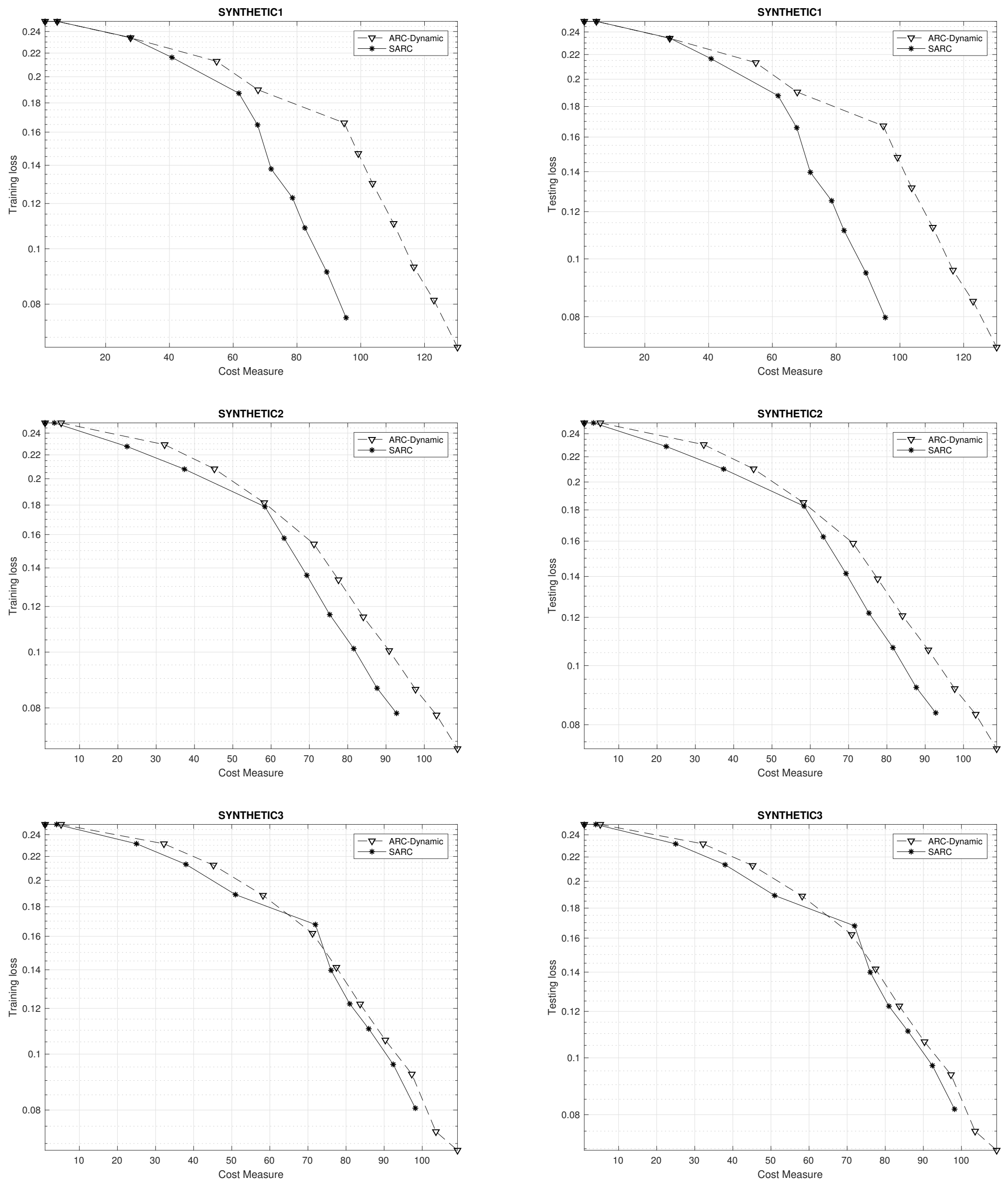


Figure 7.12: Synthetic datasets. Comparison of *SARC* (continuous line with asterisks) and *ARC-Dynamic* (dashed line with triangles) against the considered cost measure CM. Each row corresponds to a different synthetic dataset. Training loss (left) and testing loss (right) against CM with logarithmic scale on the y axis.

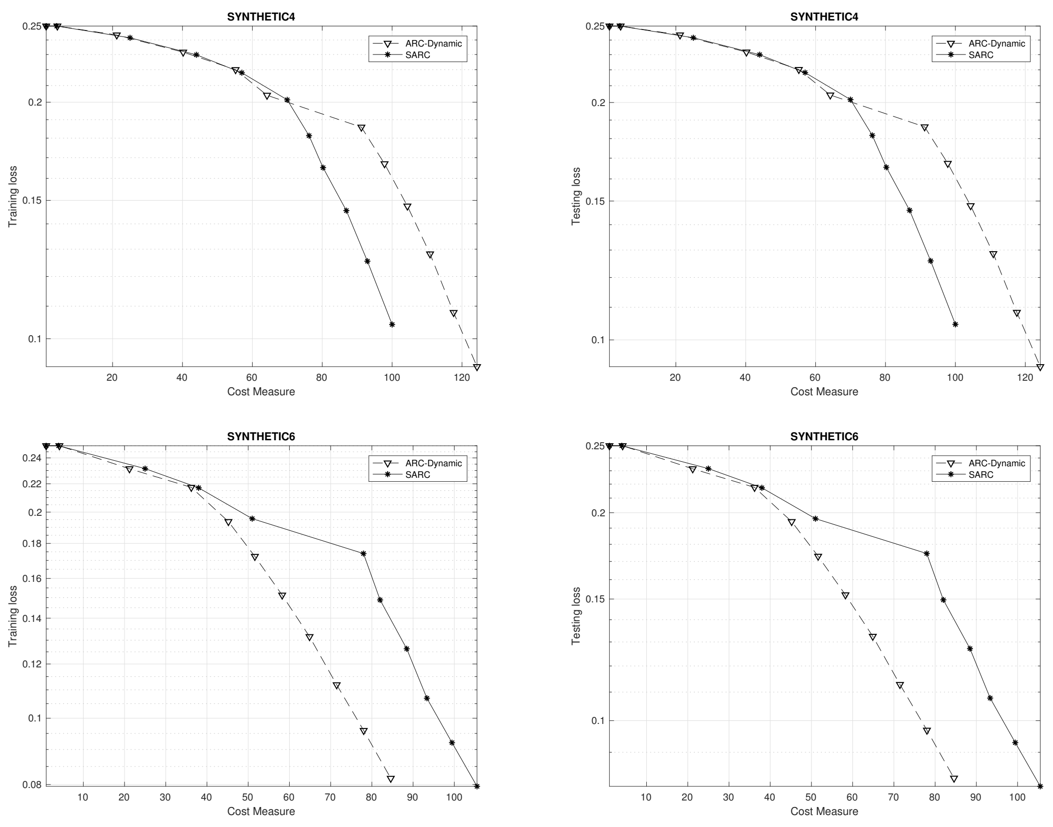


Figure 7.13: Synthetic datasets. Comparison of *SARC* (continuous line with asterisks) and *ARC-Dynamic* (dashed line with triangles) against the considered cost measure CM. Each row corresponds to a different synthetic dataset. Training loss (left) and testing loss (right) against CM with logarithmic scale on the y axis.

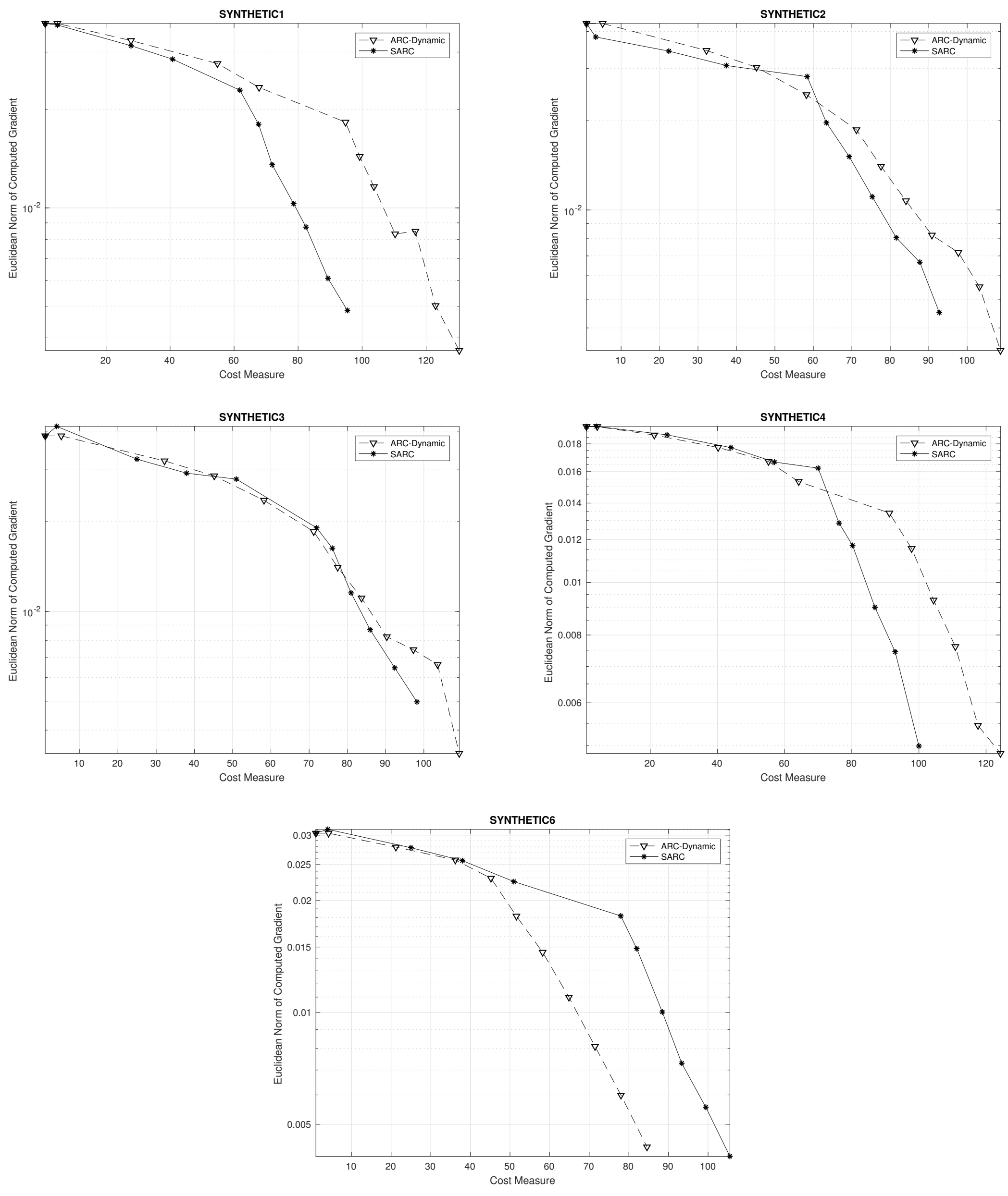


Figure 7.14: Synthetic datasets. Euclidean norm of the computed gradient against CM (training set) with logarithmic scale on the y axis. *SARC* (continuous line with asterisks), *ARC-Dynamic* (dashed line with triangles).

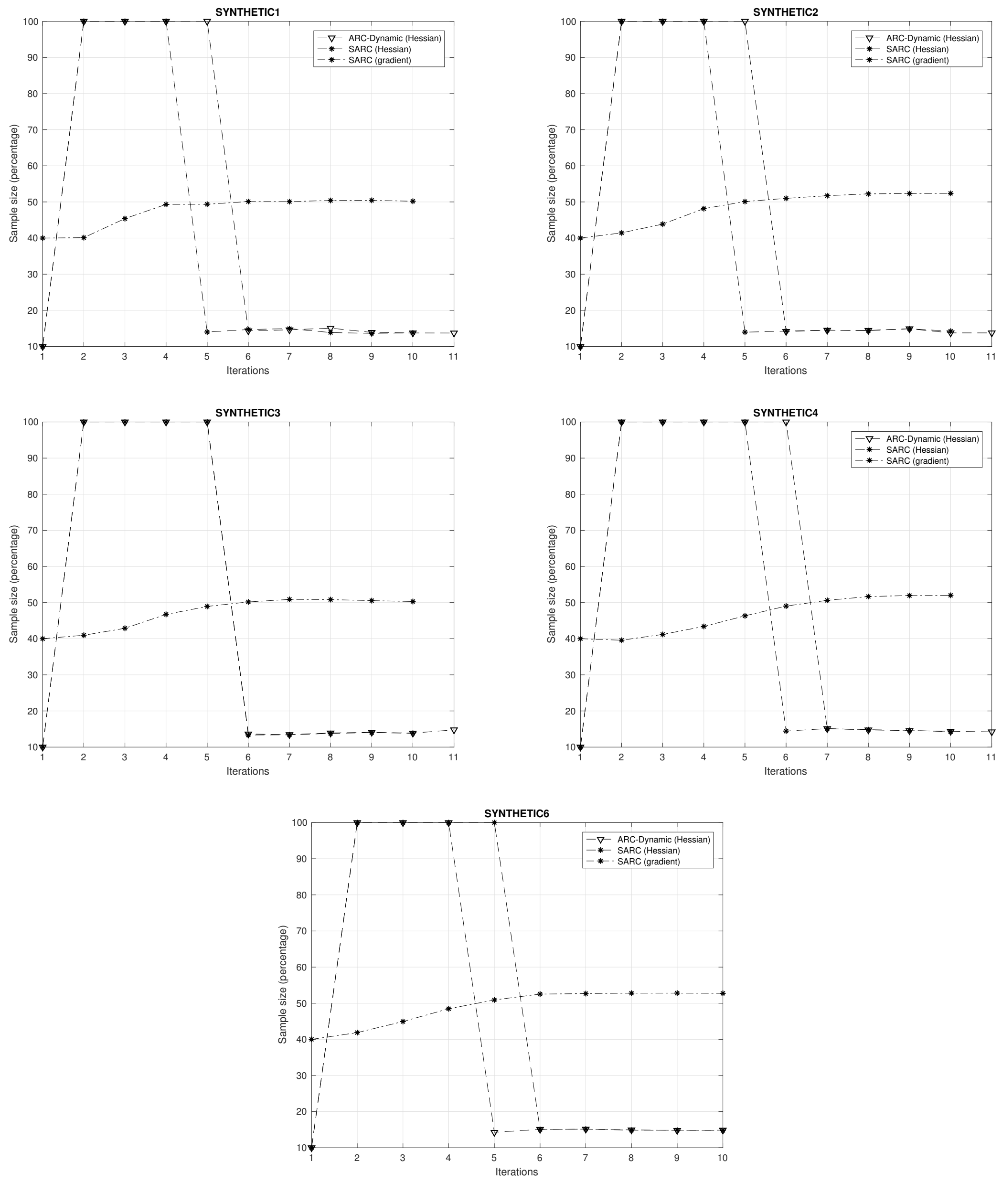


Figure 7.15: Synthetic datasets. Sample size (percentage) for Hessian approximations employed by *ARC-Dynamic* (dashed line with triangles) and $SARC$ (dashed line with asterisks), together with the sample size (percentage) for gradient approximations considered by $SARC$ (dotted dashed line with asterisks) against iterations.

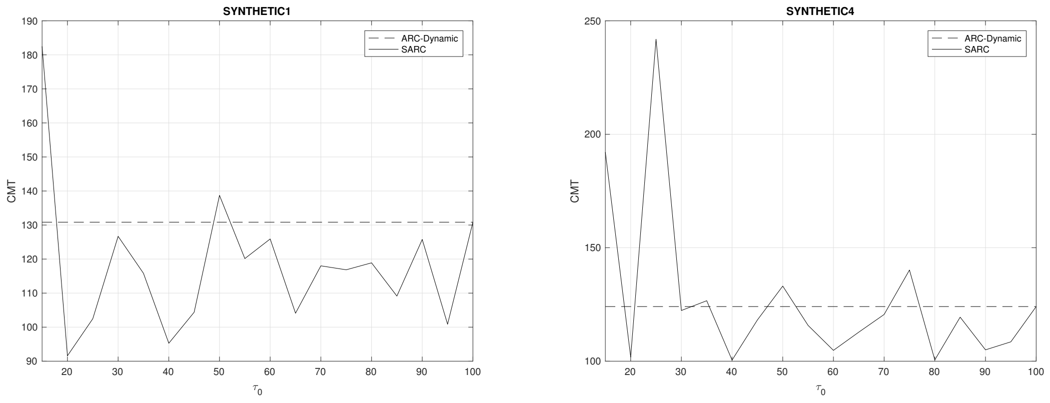


Figure 7.16: Cost Measure at Termination (CMT) against τ_0 among $SARC$ (continuous line) and *ARC-Dynamic* (dashed line) on Synthetic1 and Synthetic4.

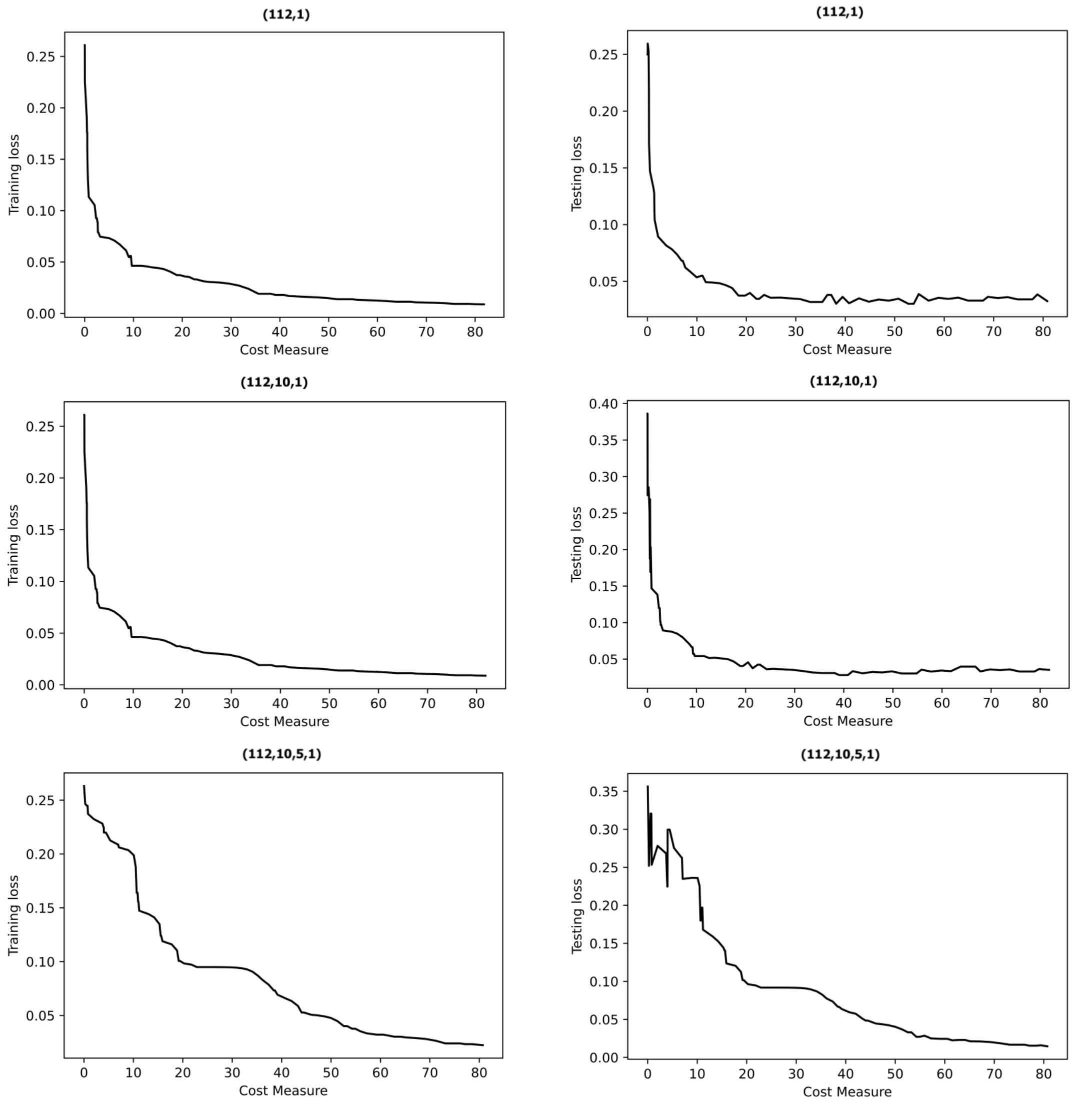


Figure 7.17: Mushroom dataset. Each row corresponds to a different ANN architecture within the AR1DA method. Training loss (left) and testing loss (right) against CM.

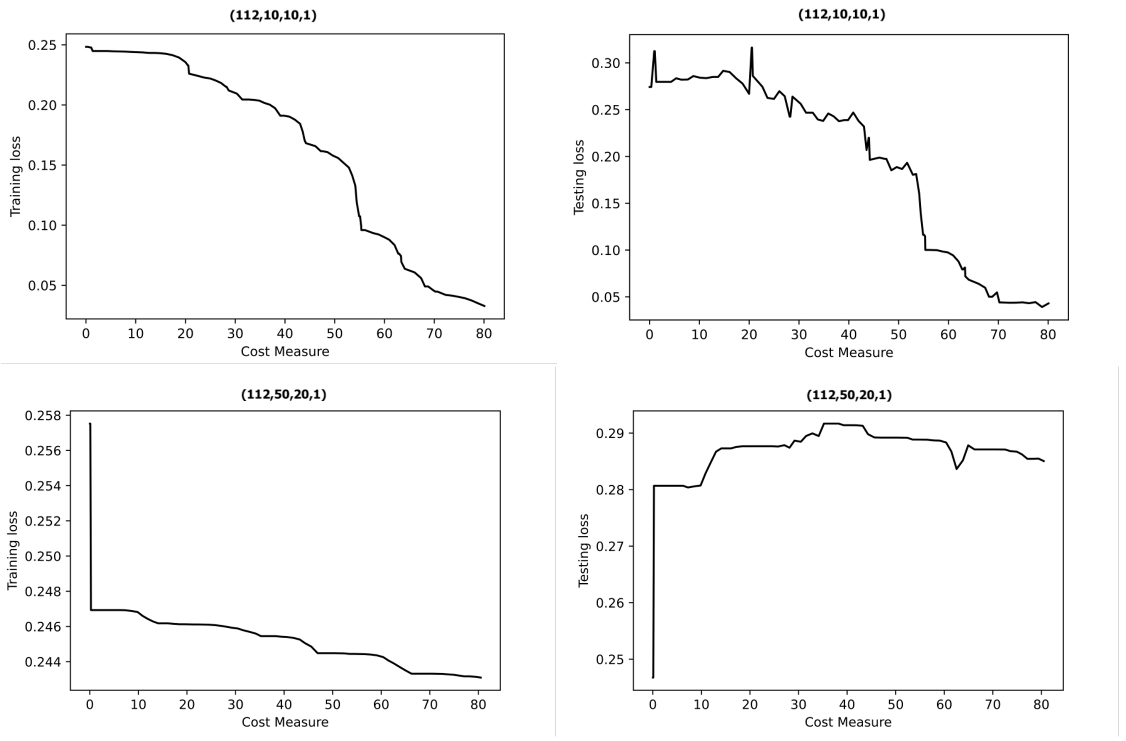


Figure 7.18: Mushroom dataset. Each row corresponds to a different ANN architecture within the AR1DA method. Training loss (left) and testing loss (right) against CM.

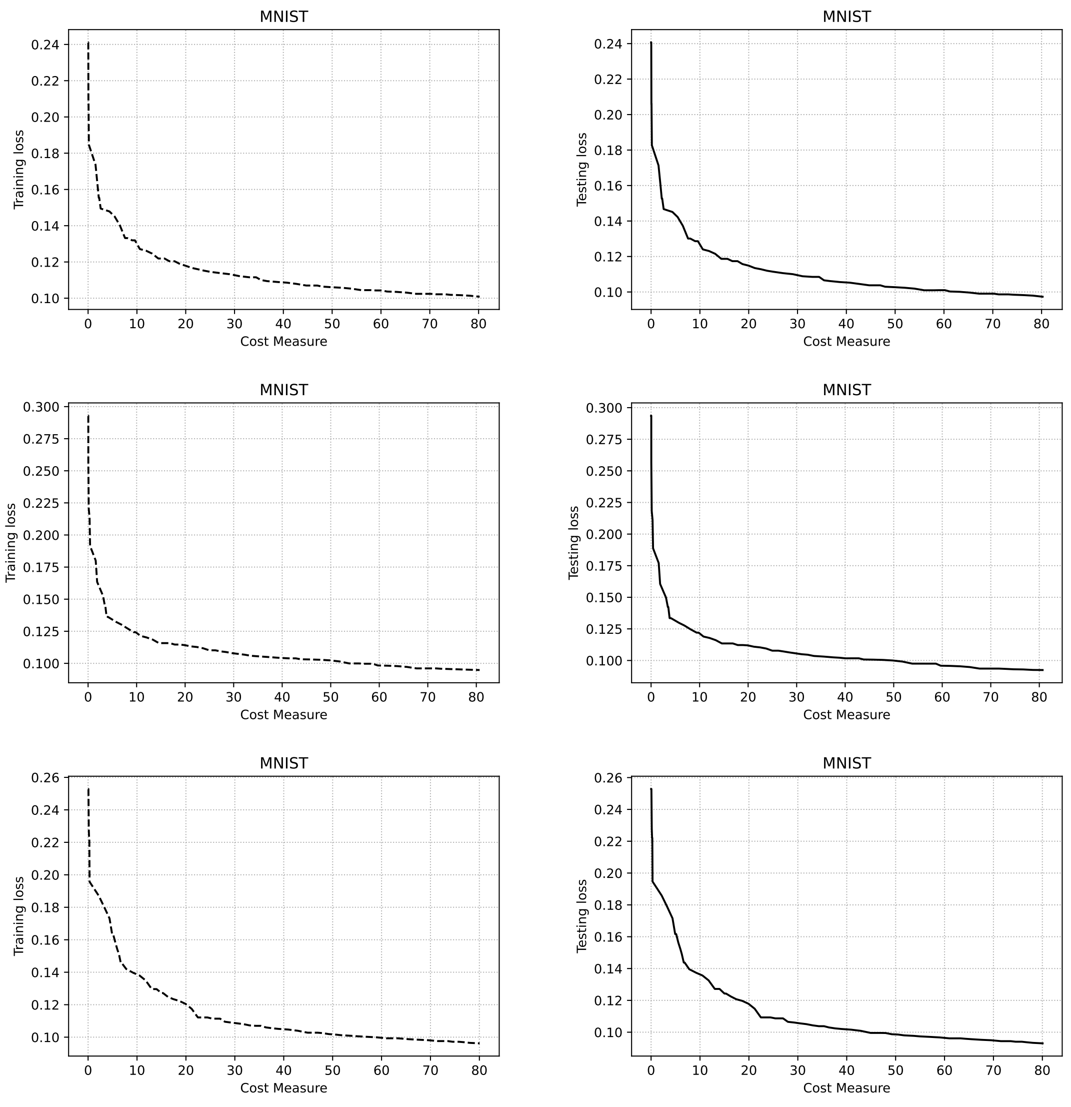


Figure 7.19: MNIST dataset. Each row corresponds to the performance of AR1DA with a different ANN architecture: $(784, 1)$ (first row), $(784, 15, 1)$ (second row) and $(784, 15, 2, 1)$ (third row). Training loss (left) and testing loss (right) against CM.

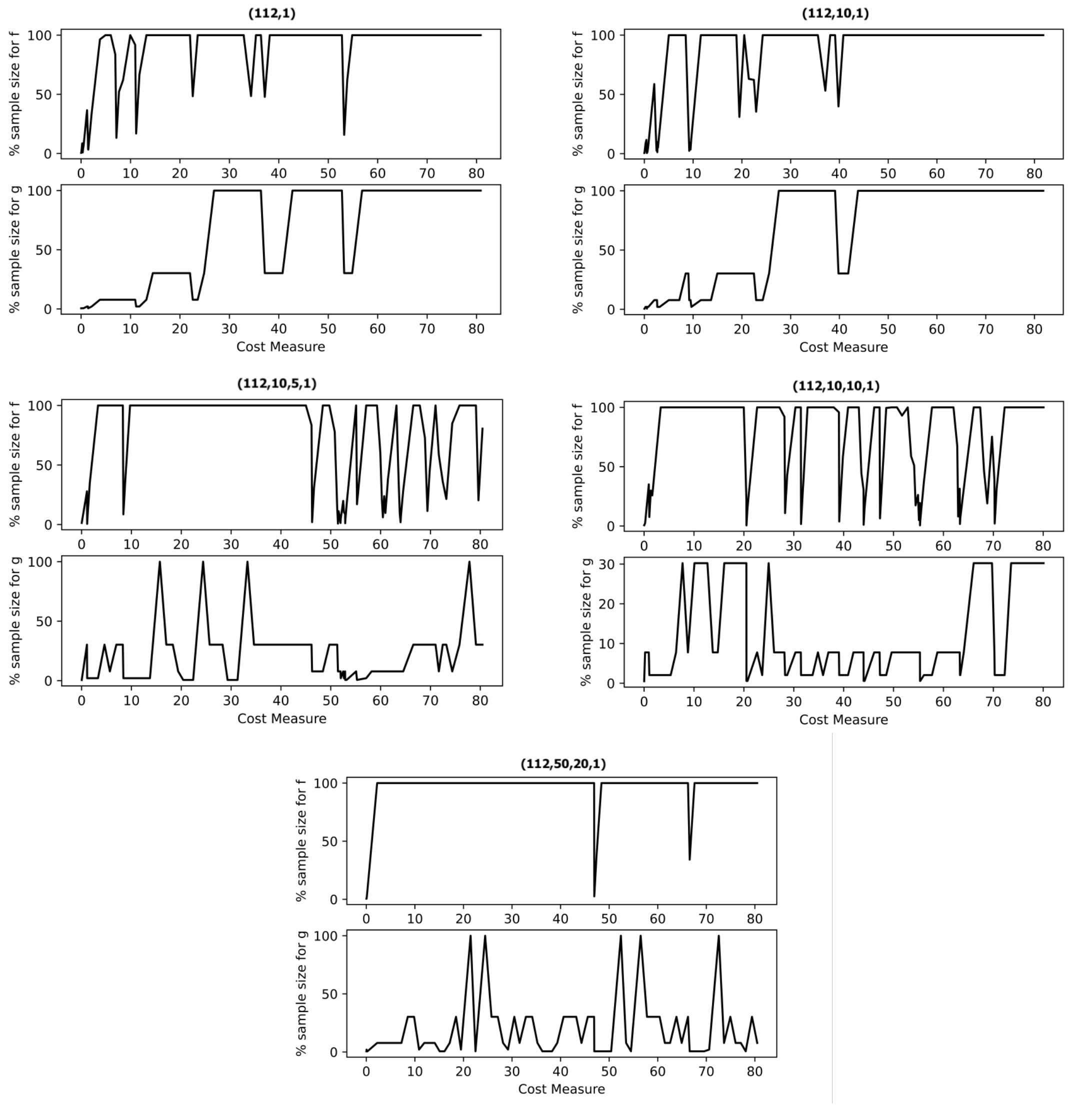
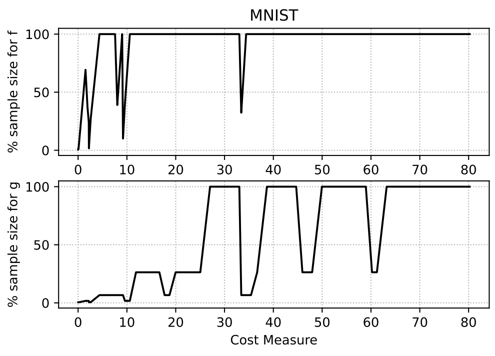
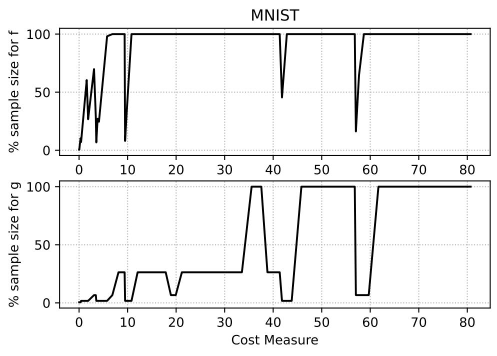


Figure 7.20: Mushroom dataset. Each row corresponds to a different ANN architecture within the AR1DA method. Sample size (percentage) for computed objective and gradient against CM.





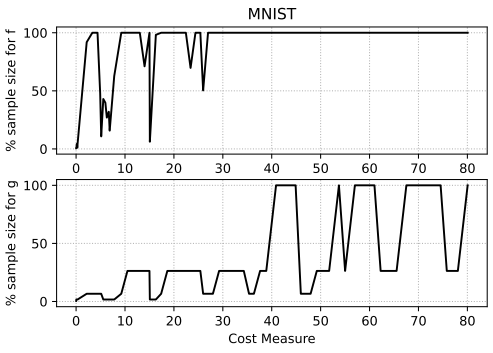


Figure 7.21: MNIST dataset. Sample sizes performed by AR1DA with different ANN architectures against CM: $(784, 1)$ (top-left), $(784, 15, 1)$ (top-right) and $(784, 15, 2, 1)$ (bottom).

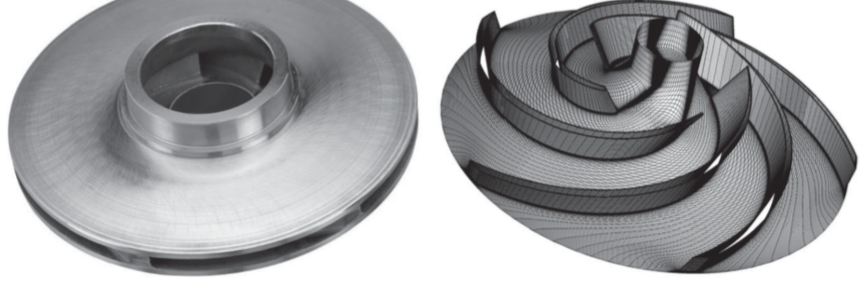
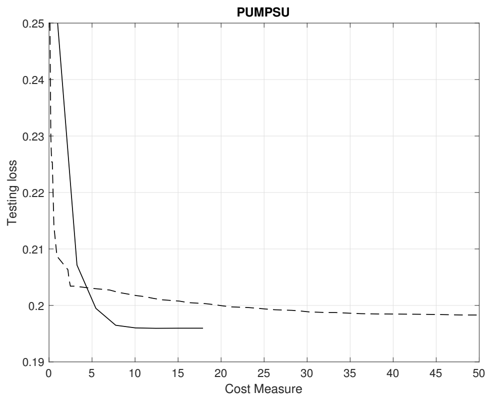


Figure 7.22: Single-shaft centrifugal impeller (left) and 3D view of impeller grid (right).



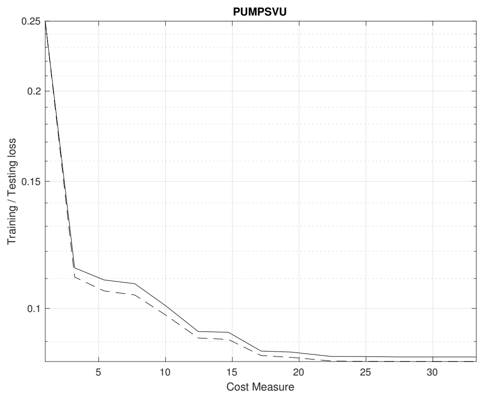


Figure 7.23: Left: PUMPSU dataset, comparison of *ARC-Dynamic* (continuous line) and AR1DA (dashed line) against Cost Measure. Right-hand side.
Right: PUMPSVU dataset, training loss (dashed line) and testing loss (continuous line) against Cost Measure via *ARC-Dynamic*.

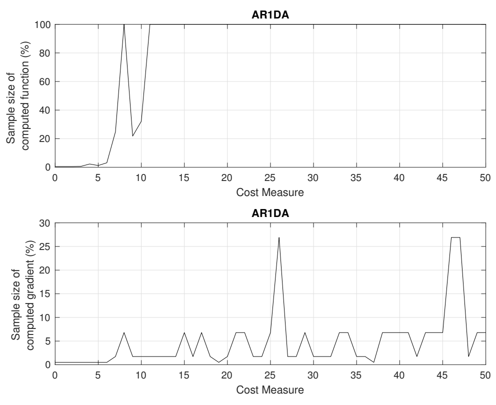


Figure 7.24: PUMPSU dataset. Sample size (percentage) for gradient and Hessian approximations employed by AR1DA against Cost Measure.

Conclusions and Perspectives

In this thesis we have ideated, analysed and tested novel variants of adaptive regularisation methods, allowing for inexactness in function and/or derivatives evaluations, able to reduce the per-iteration cost of their exact counterparts while preserving optimal complexity. This latter result has been the common thread to perform a deterministic, probabilistic and stochastic complexity analysis.

We conclude summarising our main research contributions and highlighting future perspectives.

We have started ideating an ARC algorithm with inexact Hessian computations in which the accuracy requirement (3.3) is totally reliable and implementable, without requiring inner loops subject to the step computation, that is the ARC-DH algorithm. This is a relevant improvement in itself, since to our knowledge all the methods based on the classical (implicit) agreement (1.31) (see, e.g. [44]) call for inner loops to approximate the Hessian, since the level of resemblance $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\|$ at iteration k depends on the norm of the step $\|s_k\|$ that is unknown when computing $\overline{\nabla^2 f}(x_k)$. In fact, just consider the basic ARC algorithm (Algorithm 2), we notice that $\|s_k\|$ can be determined only once the step computation (Step 2) is performed, while the Hessian approximation is required for the model definition already at Step 1. In this respect, the trick of using $\|s_{k-1}\|$ in place of $\|s_k\|$ to fulfill (1.31) before the step computation, used in [79], is not always licit, as highlighted in the numerical tests of Subsection 7.2.2.1. On this we notice that the vicious circle related to (1.31) is eluded by our accuracy requirement (3.3) for the Hessian approximation. That allows us to get rid off inner loops for Hessian estimation and only results in at most an unsuccessful iteration in the sense of Step 4 of the ARC-DH algorithm.

The approach also enhances on the ones that use an absolute accuracy requirement for Hessian approximation of the form $\|\nabla^2 f(x_k) - \overline{\nabla^2 f}(x_k)\| \leq \mu$, with $\mu = \mathcal{O}(\epsilon_1)$, as the main advantage of our adaptive strategy in Chapter 3 is that the accuracy requirement c_k for Hessian approximation (recall (3.3)) can be arbitrary large to reduce, when $\|s_k\| \geq 1$, the per-iteration cost and it approaches the first-order tolerance ϵ_1 only when $\|\nabla f(x_k)\| \simeq \epsilon_1$ and $\|s_k\| < 1$. This adaptive selection formalises the intuition that the accuracy on the inexact Hessian does not need to be stringent from the beginning, but it can be loose when starting the iterative process, it is allowed to oscillate adaptively during the execution, becoming more stringent only when the norm of the step falls below the threshold 1 and a first-order critical point is approached. A quantification of the criticality that could be reached if the Hessian accuracy is quantised, with a finite number of accuracy levels, could also be an interesting point for future investigation.

We have also showed that in the case of sums of strictly convex functions the complexity bound is improved in terms of component Hessian evaluations over schemes that do not employ adaptive rules.

We tested the new algorithm in Subsection 7.2.2 on a large number of problems and compared its outcome with the performance of its exact counterpart, ARC variants with optimal complexity and of ARC variants employing a prefixed small Hessian sample size with suboptimal complexity properties. The former comparison was carried out on synthetic datasets coming from moderately ill-conditioned problems, while the latter comparison considers machine learning datasets coming from the literature.

Numerical results within the context of supervised learning highlight that adaptiveness allows to reduce the overall computational effort without affecting the resolution of the nonconvex binary classification tasks and that the performance 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.

We remark that the real-life application to the parametric design of centrifugal pumps in Section 7.3 reveals the occurrence of a concrete problem in which second-order information seems to give a crucial contribution, since the resolutions with the main classical first-order methods, also allowing exact function and gradient computations, are totally unsuccessful.

The optimal complexity bounds derived by the deterministic analysis and retrieved in probability have then been investigated stochastically in Chapter 5, introducing inexactness in gradient computations.

For what concerns Hessian estimations, the adaptive criterion (3.3) is maintained. On the other hand, to avoid linking the accuracy requirement for gradient approximation to the norm of the step (as done in [44, 79]), the sort of relative criterion (5.4) (recall also (5.14)) for gradient estimations is introduced. We underline that the criterion is completely free from the knowledge on the norm of the step, yet its implementation requires the introduction of an inner loop, since (5.4) is an implicit requirement. Anyway, its practical implementation remains independent of the step computation and the desired accuracy can be always fulfilled in the context of nonconvex finite-sum minimisation subsampling schemes, as stated in Subsection 5.3.

The stochastic complexity analysis of the resulting SARC-IGDH algorithm is thus carried on assuming that the accuracy conditions for gradient and Hessian approximations are fulfiled within a pre-fixed (iteration independent) probability of success, building on some useful results from [44]. The analysis of course covers the particular case of the ARC-DH algorithm. From this perspective, we remark that the eventuality of having iterations in which the desired accuracies in the inexact gradient and/or Hessian computations are not satisfied results only in scaling the optimal complexity bound derived from the deterministic and probabilistic analysis by a factor involving the probability p of the model being accurate (see (5.59)).

The numerical tests of Subsection 7.2.3 are in continuity with the ones for the ARC-DH algorithm and confirm the theoretical achievements, highlighting the improvements of the novel strategy on the computational cost in most of the tests with no worsening of the accuracy rates. We underline that the gain in terms of the computational savings with respect to the considered methods in literature and the exact version of the method is less pronounced compared to the savings we had when passing from the exact version to the variant of the ARC-DH algorithm with just inexact Hessian information, as one would have expected due to the more significant cost of Hessian-vector products than gradient evaluations.

Future developments on the algorithmic aspects could include the employment of variance reduction techniques, as well as extensions of the SARC-IGDH algorithm to the case where the objective function is also evaluated with adaptive accuracy, avoiding (hopefully) the inclusion of inner loops to check for the accuracy requirements themselves, that is currently not an obvious issue.

Meanwhile, we have proposed in Chapter 4 an extension of the framework in [35] for unconstrained or inexpensively-constrained problems (see (4.3)) under approximate function and derivatives evaluations, in which the approximation of the model (4.2) at iteration k is done via an absolute accuracy requirement for function estimations (recall (4.25)–(4.26)), while the accuracy request on the Taylor's increment $\overline{\Delta T}_p^f$ appearing in (4.2) and containing the approximate derivatives up to order p is relative and ensured providing that the derivatives satisfy the absolute accuracy conditions (4.29).

The extended framework is optimal with respect to complexity and the evaluation complexity results, presented for a model of order $p+1$ to reach an optimality point of order q with $1 \le q \le p \le 2$, can be generalised to the case $1 \le q \le p$, $p > 2$, i.e. to arbitrary order model degree for arbitrary order optimality, supporting all possible combinations of exact and inexact objective functions and derivatives. At this regard we have to say that the given specialisations for lower orders ($1 \le q \le p \in \{1, 2\}$) of the AR$qp$DA algorithm are reliable in practical implementations of the method, while the case $q > 2$ remains at the moment more interesting from a theoretical/investigative perspective. An improved complexity has even been derived at the price of more stringent accuracy requirements on $\{\overline{\nabla^j f}\}_{j=1}^p$, while, as for the framework in Chapter 3, sample size rules for addressing the context of subsampling methods for machine learning have been derived. We also note that the full power of Assumption 6.1.1(ii) is only required for Lemma 57, while Lipschitz continuity of $\nabla^p f(x)$ is sufficient for all subsequent derivations of Chapter 6.

While the complexity bounds are restored in probability, the stochastic complexity analysis is performed under randomly inexact derivatives evaluations, but prescribing a deterministically controlled accuracy requirement on function approximations (from here the informal name of semi-stochastic approach). Nevertheless, the imposed dynamic accuracy framework for the objective function complements that of [17], while unbiased estimates are not assumed, extending the probabilistic assumptions with respect to [23]. The established sharp worst-case expected bounds on the evaluation complexity of computing these (possibly high-order) approximate critical points correspond in order to the best known bounds for regularisation algorithms using exact evaluations and, as for the stochastic analysis of the ARC-DH algorithm, they are scaled by a term related to the probability of success of having an accurate model.

Preliminary numerical tests for the AR$1$DA algorithm (case $q = p = 1$) have been performed in Subsection 7.2.4, also employing ANN models, and seem to be quite promising in light of the use of the method for dealing with machine learning applications. It would be interesting to further explore the numerical experimentation, also including the implementation and the testing of AR$q2$DA ($q \in \{1, 2\}$). At this regard, a nontrivial question to answer is whether experiment could currently be made using ARpqDA with $p > 2$, starting from the $p = 3$ case, and whether the consequent increasing cost (per nonlinear iteration) might pay off.

That said, there are of course many ways in which Algorithm ARqpDA might be improved. For instance, the central calculation of the relatively accurate Taylor's increments may possibly be made more efficient by updating the absolute accuracies for different degrees separately. Further techniques to avoid unnecessary derivatives computations, without affecting optimal complexity, could also be investigated. Another interesting avenue is the application of the new results to multi-precision optimisation in the context of very high performance computing. In this context, it is of paramount importance to limit energy dissipation in the course of an accurate calculation; this may be obtained by varying the accuracy of the most crucially expensive of its parts, see e.g. [66] for unconstrained quadratic optimisation and [67] for unconstrained nonconvex optimisation. The discussion above again provides guidance at what level of arithmetic accuracy is needed to achieve overall performance while maintaining optimal complexity.

For future work, we should say that the current study on these frameworks does not cover the case of noisy functions (see [51, 49, 99]), as well as the second-order stochastic complexity analysis. The stochastic second-order complexity analysis of ARC methods with derivatives and function estimates will be a challenging line of investigation. Concerning the latter point, we remark that a recent advance in (10), based on properties of supermartingales, has tackled with the second-order convergence rate analysis of a stochastic trust-region method.

Moreover, the performed analysis deliberately considers evaluation complexity (sometimes called oracle complexity), which measures the number of calls to user-supplied outer procedures for computing approximate function and derivatives values, irrespective of internal computations within the algorithm itself. This is motivated by the observation that these often dominate the total cost of the computation in practical applications. We are fully aware that total computational iteration or evaluation complexity remains a challenge for upcoming investigation. Fortunately, the difference is well understood at least when searching for first or second-order approximate minimisers, in that moderately costly methods are available for handling the algorithm internal calculations (see [33, 37, 108]).

All in all, we go forward trusting that some of the hints embedded in the proposed strategies could play their own role to effectively reinforce the simulation of a real application, with the compass given by the theoretical foundation of innovative research numerical methods.

Appendix A

Appendix

A.1 Computing the global minimiser of the model (1.14)

As done in the proof of Theorem 15, throughout this subsection we omit for readability purposes the iteration subscript k. With reference to (1.84)–(1.86), we notice that the solution $s(\lambda)$ we are looking for depends upon the nonlinear equality $\|s(\lambda)\| = \lambda/\sigma$. To examine $\|s(\lambda)\|$ in details, let us define $\psi(\lambda) \overset{\text{def}}{=} \|s(\lambda)\|^2$, satisfying

$$\psi(\lambda) = \|U^\top(\Lambda + \lambda I_n)^{-1}U\nabla f(x)\|^2 = \|(\Lambda + \lambda I_n)^{-1}U\nabla f(x)\|^2 = \sum_{i=1}^{n} \frac{(e_i^\top U\nabla f(x))^2}{(\lambda_i + \lambda)^2}.$$

We can thus distinguish three different cases.

- $\overline{\nabla^2 f}(x)$ is positive semidefinite. In this case, the solution is given by the single positive root to either of the equivalent univariate nonlinear equations

$$\theta_2(\lambda) \overset{\text{def}}{=} \psi(\lambda) - \frac{\lambda^2}{\sigma^2} = 0 \quad \text{or} \quad \theta_1(\lambda) \overset{\text{def}}{=} \sqrt{\psi(\lambda)} - \frac{\lambda}{\sigma} = 0, \tag{A.1}$$

  since the left-hand sides in (A.1) are strictly decreasing functions of λ, for $\lambda \geq \max[0, -\lambda_1] = 0$ and range between some positive value and $-\infty$. See Figure A.1 below for an illustration of the graphs of $\theta_1(\lambda)$ and $\theta_2(\lambda)$.

- $\overline{\nabla^2 f}(x)$ is indefinite and $e_1^\top U\nabla f(x) \neq 0$. The solution is likewise the root larger than $-\lambda_1$ of the same equations and as in the previous point the model $m_k(s)$ has a unique global minimiser.

- $\overline{\nabla^2 f}(x)$ is indefinite and $e_1^\top U\nabla f(x) = 0$.

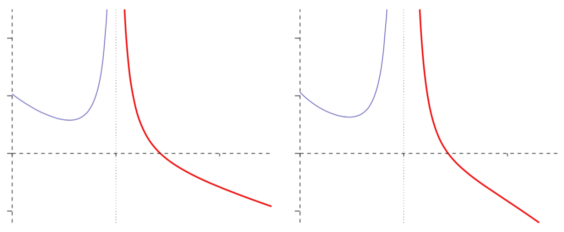


Figure A.1: Graphs of the functions $\theta_1(\lambda)$ (left) and $\theta_2(\lambda)$ (right) from (A.1) when $\nabla f(x) = (0.25, 1)^\top$, $\overline{\nabla^2 f}(x) = \operatorname{diag}(-1, 1)$ and $\sigma = 2$.

The latter case is analogous to those for the hard case in [52, Section 7.3.1.3] for the trust-region subproblem in which the required solution s_* is made up as a suitable linear combination of u_1 and $\lim_{s \to -\lambda_1} s(\lambda)$.

To determine the values of the coefficients of this linear combination, in place of the trust-region constraint, we employ the constrains in (1.84) and find a value of $\alpha \in \mathbb{R}$ such that

$$-\lambda_1 = \sigma\|s(-\lambda_1) + \alpha u_1\|.$$

As for the trust-region case, it is in practice preferable to solve one of the following equations

$$\begin{aligned} \varphi_2(\lambda) &\overset{\text{def}}{=} \frac{1}{\psi(\lambda)} - \frac{\sigma^2}{\lambda^2} = 0, \qquad & \varphi_1(\lambda) &\overset{\text{def}}{=} \frac{1}{\sqrt{\psi(\lambda)}} - \frac{\sigma}{\lambda} = 0, \\ \beta_2(\lambda) &\overset{\text{def}}{=} \frac{\lambda^2}{\psi(\lambda)} - \sigma^2 = 0 \quad \text{or} & \beta_1(\lambda) &\overset{\text{def}}{=} \frac{\lambda}{\sqrt{\psi(\lambda)}} - \sigma = 0, \end{aligned} \tag{A.2}$$

instead of (A.1). In Figures A.2–A.3 on this page we report their graphical representations.

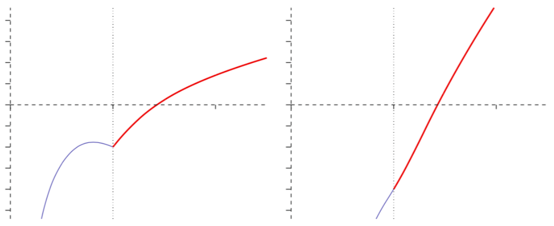


Figure A.2: Graphs of the functions $\varphi_1(\lambda)$ (left) and $\varphi_2(\lambda)$ (right) from (A.2) when $\nabla f(x) = (0.25, 1)^\top$, $\overline{\nabla^2 f}(x) = \text{diag}(-1, 1)$ and $\sigma = 2$.

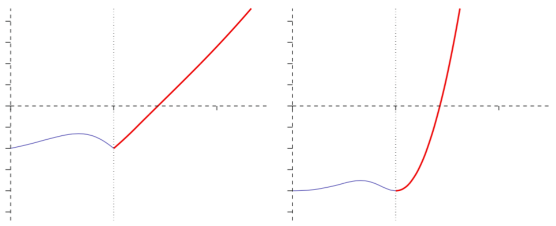


Figure A.3: Graphs of the functions $\beta_1(\lambda)$ (left) and $\beta_2(\lambda)$ (right) from (A.2) when $\nabla f(x) = (0.25, 1)^\top$, $\overline{\nabla^2 f}(x) = \text{diag}(-1, 1)$ and $\sigma = 2$.

To find the required root of whichever of the functions in (A.2), a safeguarded univariated Newton's iteration can be considered. This, in turn, leads to the resolution of a sequence of linear systems

$$B(\lambda^{(j)})s = (\overline{\nabla^2 f}(x) + \lambda^{(j)} I_n)s = -\nabla f(x),$$

for selected $\lambda^{(j)} > \max[0, -\lambda_1]$, in which the use of the derivatives of $\psi(\lambda)$ is obtained once a factorisation of $\overline{\nabla^2 f}(x) + \lambda^{(j)} I_n$ is known. To this aim, we report the following useful results, proved in [52, Lemma 7.3.1].

Lemma 65. [40, Lemma 6.1] Suppose that $s(\lambda)$ satisfies (1.86), $\psi(\lambda) \overset{\text{def}}{=} \|s(\lambda)\|^2$ and $\lambda > \lambda_1$. Then,

$$\psi'(\lambda) = 2s(\lambda)^\top \nabla_\lambda s(\lambda) \quad \text{and} \quad \psi''(\lambda) = 6\|\nabla_\lambda s(\lambda)\|^2,$$

where, recalling the definition (1.85),

$$\nabla_\lambda s(\lambda) = -B(\lambda)^{-1} s(\lambda).$$

Moreover, given the Cholesky factorisation $B(\lambda) = L(\lambda)L(\lambda)^\top$ (L lower triangular with positive diagonal components), it follows that

$$s(\lambda)^\top \nabla_\lambda s(\lambda) = -\|\omega(\lambda)\|^2,$$

in which $L(\lambda)L(\lambda)^\top s(\lambda) = -\nabla f(x)$ and $L(\lambda)\omega(\lambda) = s(\lambda)$.

Corollary 66. Suppose that $\nabla f(x) \neq 0$. The function $\varphi_1(\lambda)$ in (A.2) is strictly increasing, when $\lambda > \max[0, -\lambda_1] = 0$, and concave. Its first two derivatives are

$$\varphi_1'(\lambda) = \frac{-s(\lambda)^\top \nabla_\lambda s(\lambda)}{\|s(\lambda)\|^3} + \frac{\sigma}{\lambda^2} > 0$$

and

$$\varphi_1''(\lambda) = \frac{3\left(s(\lambda)^\top \nabla_\lambda s(\lambda)^2 - \|s(\lambda)\|^2 \|\nabla_\lambda s(\lambda)\|^2\right)}{\|s(\lambda)\|^5} - \frac{2\sigma}{\lambda^3} < 0.$$

The basic Newton's iteration to numerically solve $\varphi_1(\lambda) = 0$ is as follows.

Algorithm 15 [40, Algorithm 6.1].

```
Let $\lambda > \max[0, -\lambda_1]$ be given.
Step 1. Factorise $B(\lambda) = LL^\top$.
Step 2. Solve $LL^\top s = -\nabla f(x)$.
Step 3. Solve $L\omega = s$.
Step 4. Compute the Newton correction $\Delta\lambda^N = \frac{\lambda\left(\|s\| - \frac{\lambda}{\sigma}\right)}{\|s\| + \frac{\lambda}{\sigma}\left(\frac{\lambda\|\omega\|^2}{\|s\|^2}\right)}$.
Step 5. Replace $\lambda$ by $\lambda + \Delta\lambda^N$.
```

As in the case of the trust-region method, various safeguards should be added to gain and speed up convergence of Algorithm 15 from an arbitrary λ (see, e.g., [52, Sections 7.3.4–7.3.8]). The idea in [40, Section 6.1] for improving the speed of global convergence is to only linearise the term $\omega(\lambda) \overset{\text{def}}{=} 1/\sqrt{\psi(\lambda)}$ of φ_1 in (A.2) instead of the σ/λ term, as done in the Newton's method when computing a correction $\Delta\lambda^c$ to the estimate λ of the required root of φ_1. The resulting correction satisfies the equation

$$\omega(\lambda) + \omega'(\lambda)\Delta\lambda^c = \frac{1}{\sqrt{\psi(\lambda)}} - \frac{1}{2}\frac{\psi'(\lambda)}{\sqrt{\psi(\lambda)^3}}\Delta\lambda^c = \frac{\sigma}{\lambda + \Delta\lambda^c}, \tag{A.3}$$

which corresponds to a quadratic equation for the correction $\Delta\lambda^c$. In this context, we report the result from [40] showing a case in which Algorithm 15 with the variant (A.3) is globally convergent.

Theorem 67. [40, Theorem 6.3] Suppose $\lambda > -\lambda_1$ and $\varphi_1(\lambda) < 0$. Then, both the Newton iterate $\lambda+\Delta\lambda^N$ and alternative $\lambda+\Delta\lambda^c$ inherits these properties and converge monotonically towards the required root λ_*. The convergence is globally linear with factor at least $1 - \varphi_1'(\lambda_*)/\varphi_1'(\lambda) < 1$ and is ultimately quadratic. Moreover, $\lambda + \Delta\lambda^N \leq \lambda + \Delta\lambda^c \leq \lambda_*$.

Proof. The proof follows the one in [40, Theorem 6.3] which is, in turn, inherited from [52, Lemma 7.3.2]. Since $\omega(\lambda)$ is a concave function of λ, (A.2) and (A.3) give that

$$\varphi_1(\lambda+\Delta\lambda^c) = \omega(\lambda+\Delta\lambda^c) - \frac{\sigma}{\lambda+\Delta\lambda^c} \leq \omega(\lambda) + \omega'(\lambda)\Delta\lambda^c - \frac{\sigma}{\lambda+\Delta\lambda^c} = 0.$$

The Newton's iteration then satisfies the linearised equation

$$\omega(\lambda) + \omega'(\lambda)\Delta\lambda^N = \frac{\sigma}{\lambda} - \frac{\sigma}{\lambda^2}\Delta\lambda^N. \tag{A.4}$$

But, as σ/λ is a convex function of λ,

$$\frac{\sigma}{\lambda+\Delta\lambda^c} \geq \frac{\sigma}{\lambda} - \frac{\sigma}{\lambda^2}\Delta\lambda^c,$$

and, hence,

$$\omega(\lambda) + \omega'(\lambda)\Delta\lambda^c \geq \frac{\sigma}{\lambda} - \frac{\sigma}{\lambda^2}\Delta\lambda^c,$$

from (A.3). Combining this with (A.4) we obtain that

$$\varphi_1'(\lambda)(\Delta\lambda^c - \Delta\lambda^N) = \left(\omega'(\lambda) + \frac{\sigma}{\lambda^2}\right)(\Delta\lambda^c - \Delta\lambda^N) \geq 0$$

and, hence, $\Delta\lambda^c \geq \Delta\lambda^N > 0$, since Corollary 66 gives $\varphi_1'(\lambda) > 0$. The alternative iterate indeed improves the Newton's one. $\square$

It is worth noting that similar results can be derived for the other roots in (A.1)–(A.2) that we will not report here. Let us instead conclude with the following theorem, concerning the global minimiser of $s(\sigma)$ of the cubic model $m_k(s)$ in (1.14).

Theorem 68. [40, Theorem 6.4] Suppose that $s(\sigma) \neq 0$ is a global minimiser of the model $m_k(s)$ in (1.14). Then, the length of the minimiser $\nu(\sigma) \overset{\text{def}}{=} \|s(\sigma)\|$ is a non-increasing function of σ.

Proof. We consider the proof in [40], instead of considering the more complicated proof given by Griewank in [70]. From Theorem 15 we have that

$$(\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n)s(\sigma) = -\nabla f(x) \tag{A.5}$$

and that $\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n$ and thus $\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n + \sigma\|s(\sigma)\|^{-1}s(\sigma)s(\sigma)^\top$ are positive semidefinite. We consider the derivative $\nu'(\sigma) = \|s(\sigma)\|^{-1}s(\sigma)^\top\nabla_\sigma s(\sigma)$. Differentiating (A.5) with respect to σ reveals that

$$(\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n)\nabla_\sigma s(\sigma) + \|s(\sigma)\|s(\sigma) + \sigma\|s(\sigma)\|^{-1}s(\sigma)s(\sigma)^\top\nabla_\sigma s(\sigma) = 0$$

and thus that

$$\left(\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n + \sigma\|s(\sigma)\|^{-1}s(\sigma)s(\sigma)^\top\right)\nabla_\sigma s(\sigma) = -\|s(\sigma)\|s(\sigma).$$

Left-multiplication by $s(\sigma)^\top$ and dividing by $\|s(\sigma)\|$ give that

$$\nu'(\sigma) = -\frac{\nabla_\sigma s(\sigma)^\top \left(\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n + \sigma\|s(\sigma)\|^{-1}s(\sigma)s(\sigma)^\top\right)\nabla_\sigma s(\sigma)}{\|s(\sigma)\|^2} \leq 0,$$

since we have seen that $\overline{\nabla^2 f}(x) + \sigma\|s(\sigma)\|I_n + \sigma\|s(\sigma)\|^{-1}s(\sigma)s(\sigma)^\top$ is positive semidefinite. Consequently, $\nu'(\sigma)$ is a non-increasing function of σ. $\square$

A.2 Proofs

We here report the proof of Lemma 18 in Chapter 1, Lemma 54 and of Lemma 57 considered in Chapter 6.

Proof of Lemma 18
[52, Proof of Lemma 5.2.1] Firstly, by definition of the Krylov's spaces and assumption (1.93),

$$\begin{aligned}\mathrm{span}\{q_0, ..., q_i, q_{i+1}\} &= \mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, i+1)\\ &= \mathrm{span}\{\mathcal{K}(\overline{\nabla^2 f}(x_k), q_0, i), (\overline{\nabla^2 f}(x_k))^{i+1}q_0\}\\ &= \mathrm{span}\{q_0, ..., q_i, (\overline{\nabla^2 f}(x_k))^{i+1}q_0\},\end{aligned}$$

for $i \in \{0, ..., t\}$. Thus,

$$q_{i+1} \in \mathrm{span}\{q_0, q_1, ..., q_i, (\overline{\nabla^2 f}(x_k))^{i+1}q_0\} \tag{A.6}$$

and

$$(\overline{\nabla^2 f}(x_k))^{i+1}q_0 \in \mathrm{span}\{q_0, q_1, ..., q_{i+1}\}, \tag{A.7}$$

for $i \in \{0, ..., t\}$. The relationship (1.94) is trivially true when $i = 0$. Now assume, inductively, that

$$q_i \in \mathrm{span}\{q_0, q_1, ..., q_{i-1}, \overline{\nabla^2 f}(x_k)q_{i-1}\}, \tag{A.8}$$

for $i \in \{0, ..., r\}$. It then follows that

$$\overline{\nabla^2 f}(x_k)q_{i-1} \in \mathrm{span}\{q_0, q_1, ..., q_i\}, \tag{A.9}$$

for $i \in \{0, ..., r\}$, as the q_i are mutually orthonormal. Then, following (A.6),

$$\begin{aligned}q_{r+1} &\in \mathrm{span}\{q_0, q_1, ..., q_r, (\overline{\nabla^2 f}(x_k))^{r+1}q_0\}\\ &= \mathrm{span}\{q_0, q_1, ..., q_r, \overline{\nabla^2 f}(x_k)[(\overline{\nabla^2 f}(x_k))^{r}q_0]\}\\ &= \mathrm{span}\{q_0, q_1, ..., q_r, \overline{\nabla^2 f}(x_k)q_0, ..., \overline{\nabla^2 f}(x_k)q_{r-1}, \overline{\nabla^2 f}(x_k)q_r\}\\ &= \mathrm{span}\{q_0, q_1, ..., q_r, \overline{\nabla^2 f}(x_k)q_r\},\end{aligned}$$

in which the last two equalities follow from (A.7) and (A.9), respectively. Thus, (A.8) holds for $r \in \{0, ..., t\}$, which concludes the proof.

Proof of Lemma 54
Let s_k^* be a global minimiser of $m_k(s)$ in $\mathcal{X}$. By Taylor's theorem, we have that, for all d such

that $x_k + s_k^* + d \in \mathcal{X}$,

$$\begin{aligned} 0 \quad \leq \quad & m_k(s_k^*+d) - m_k(s_k^*) = \sum_{\ell=1}^{p} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell \\ & + \frac{\sigma_k}{(p+1)!} \left[\sum_{\ell=1}^{p} \frac{1}{\ell!} \nabla_s^\ell \left(\|s_k^*\|^{p+1} \right) [d]^\ell + \frac{1}{(p+1)!} \nabla_s^{p+1} \left(\|s_k^* + \tau d\|^{p+1} \right) [d]^{p+1} \right], \end{aligned} \tag{A.1}$$

for some $\tau \in (0,1)$. We may now use the expression of $\nabla_s^\ell \left(\|s_k^*\|^{p+1} \right)$ given by [35, Lemma 2.4] in (A.1) and deduce that, for any $j \in \{1, \ldots, q\}$ and all d such that $x_k + s_k^* + d \in \mathcal{X}$,

$$\begin{aligned} & - \sum_{\ell=1}^{j} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell - \frac{\sigma_k}{(p+1)!} \sum_{\ell=1}^{j} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell \\ & \qquad \leq \sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell + \frac{\sigma_k}{(p+1)!} \left[\sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell + \|d\|^{p+1} \right]. \end{aligned} \tag{A.2}$$

It is now possible to choose $\delta_{k,j} \in (0,1]$ such that, for every d with $\|d\| \leq \delta_{k,j}$,

$$\sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell + \frac{\sigma_k}{(p+1)!} \left[\sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell + \|d\|^{p+1} \right] \leq \frac{1}{2} \theta \epsilon_j \frac{\delta_{k,j}^j}{j!}. \tag{A.3}$$

We therefore obtain that if $\delta_{k,j}$ is small enough to ensure (A.3), then (A.2) implies that

$$- \sum_{\ell=1}^{j} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell - \frac{\sigma_k}{(p+1)!} \sum_{\ell=1}^{j} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell \leq \frac{1}{2} \theta \epsilon_j \frac{\delta_{k,j}^j}{j!}. \tag{A.4}$$

and therefore that, for all $j \in \{1, \ldots, q\}$,

$$\max_{x_k + s_k^* + d \in \mathcal{X}, \|d\| \leq \delta_{k,j}} \overline{\Delta T}_{m_k,j}(s_k^*, d) \leq \frac{1}{2} \theta \epsilon_j \frac{\delta_{k,j}^j}{j!}.$$

Hence, the pair (s_k^*, δ_k) is acceptable for Step 2 of the algorithm. If we now assume that $x_k + s_k^*$ is not an isolated feasible point, the above inequality and the continuity of $\overline{T}_p^f(x_k, s)$ and its derivatives with respect to s then ensure the existence of a feasible neighbourhood $\mathcal{N}_k^*$ of s_k^* in which

$$\operatorname*{globmax}_{x_k + s + d \in \mathcal{X}, \|d\| \leq \delta_{k,j}} \overline{\Delta T}_j^{m_k}(s, d) \leq \theta \epsilon_j \frac{\delta_{k,j}^j}{j!}, \tag{A.5}$$

for all $s \in \mathcal{N}_k^*$. We may then choose any s_k in $\mathcal{N}_k^*$ such that, in addition to satisfy (A.5) and being such that $x_k + s_k$ is feasible, (6.4) also holds. Thus, the definition of $\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k)$ in (6.5) gives that

$$\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k) \leq \theta \epsilon_j \frac{\delta_{k,j}^j}{j!} \tag{A.6}$$

and every such (s_k, δ_k) is also acceptable for Step 2 of the algorithm.

Proof of Lemma 57 Let s_k^* be a global minimiser of $m_k(s)$. We first consider the case where $q = 1$ and $\mathcal{X}$ is convex or $q = 2$ and $\mathcal{X} = \mathbb{R}^n$. Then, it can be verified that, for each $j \in \{1, \ldots, q\}$, the optimisation problem involved in the definition of $\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k^*)$ (in (6.5)) is convex and therefore that $\delta_{k,j}$ can be chosen arbitrarily in $(0,1]$. The first case of Lemma 57 then follows from the continuity of $\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s)$ with respect to s.

In order to prove the second case, we now pursue the reasoning of the proof of Lemma 54. We start by supposing that $\|s_k^*\| > 1$. We may then reduce the neighbourhood of s_k^* in which s_k can be chosen to guarantee that $\|s_k\| \geq 1$, which then gives the desired result because of (A.5). Suppose therefore that $\|s_k^*\| \leq 1$. The triangle inequality then implies that

$$\|\nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)\| \leq \sum_{i=\ell}^{p} \frac{1}{(i-\ell)!} \|\overline{\nabla_x^i f}(x_k)\| \, \|s_k^*\|^{i-\ell},$$

for $\ell \in \{q+1, \ldots, p\}$, and thus, using, AS.1 and [35, Lemma 2.4], we deduce that

$$\begin{aligned}\sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell + \frac{\sigma_k}{(p+1)!} \left[\sum_{\ell=j+1}^{p} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell \right] \\ \leq \sum_{\ell=j+1}^{p} \frac{\|d\|^\ell}{\ell!} \left[\sum_{i=\ell}^{p} \frac{\|s_k^*\|^{i-\ell}}{(i-\ell)!} \|\overline{\nabla_x^i f}(x_k)\| + \frac{\sigma_k \|s_k^*\|^{p-\ell+1}}{(p-\ell+1)!} \right].\end{aligned}$$

We now call upon the fact that, since $q \geq 3$ or $\mathcal{X}$ is not convex or $q = 2$ and $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{M}_k$ occurs by assumption, $\mathcal{M}_k^{(4)}$ also occurs. Thus,

$$\begin{aligned}\sum_{\ell=j+1}^{p} \frac{1}{\ell!} \nabla_s^\ell \overline{T}_p^f(x_k, s_k^*)[d]^\ell + \frac{\sigma_k}{(p+1)!} \left[\sum_{\ell=j+1}^{p} \nabla_s^\ell \|s_k^*\|^{p+1} [d]^\ell \right] \\ \leq \sum_{\ell=j+1}^{p} \frac{\|d\|^\ell}{\ell!} \left[\Theta \sum_{i=\ell}^{p} \frac{\|s_k^*\|^{i-\ell}}{(i-\ell)!} + \frac{\sigma_k \|s_k^*\|^{p-\ell+1}}{(p-\ell+1)!} \right].\end{aligned}$$

We therefore obtain from (A.3) that any pair $(s_k^*, \delta_{s,j})$ satisfies (A.4) for $\|d\| \leq \delta_{s,j}$ if

$$\sum_{\ell=j+1}^{p} \frac{\delta_{s,j}^\ell}{\ell!} \left[\Theta \sum_{i=\ell}^{p} \frac{1}{(i-\ell)!} \|s_k^*\|^{i-\ell} + \frac{\sigma_k \|s_k^*\|^{p-\ell+1}}{(p-\ell+1)!} \right] + \sigma_k \frac{\delta_{s,j}^{p+1}}{(p+1)!} \leq \frac{1}{2} \theta \epsilon_j \frac{\delta_{s,j}^j}{j!}; \tag{A.7}$$

which, because $\|s_k^*\| \leq 1$, is in turn ensured by the inequality

$$\sum_{\ell=j+1}^{p} \frac{\delta_{s,j}^\ell}{\ell!} \left[\Theta \sum_{i=\ell}^{p} \frac{1}{(i-\ell)!} + \sigma_k \right] + \sigma_k \frac{\delta_{s,j}^{p+1}}{(p+1)!} \leq \frac{1}{2} \theta \epsilon_j \frac{\delta_{s,j}^j}{j!}. \tag{A.8}$$

Observe now that, since $\delta_{s,j} \in [0,1]$, $\delta_{s,j}^\ell \leq \delta_{s,j}^{j+1}$ for $\ell \in \{j+1, \ldots, p\}$. Moreover, we have that,

$$\sum_{i=\ell}^{p} \frac{1}{(i-\ell)!} \leq e < 3, \quad (\ell \in \{j+1, \ldots, p+1\}), \qquad \sum_{\ell=j+1}^{p+1} \frac{1}{\ell!} \leq e - 1 < 2$$

and therefore (A.8) is guaranteed by the condition

$$j!(6\Theta + 2\sigma_k)\, \delta_{s,j} \leq \frac{1}{2} \theta \epsilon_j, \tag{A.9}$$

which means that the pair (s_k^*, δ_s) satisfies (A.4) for all $j \in \{1, \ldots, q\}$, whenever

$$\delta_{s,j} \leq \frac{1}{2} \delta_{\min,k} \stackrel{\rm def}{=} \frac{\theta \epsilon_j}{2q!(6\Theta + 2\sigma_k)}.$$

As in the proof of Lemma 54, we may invoke the continuity of the derivatives of $m_k(s)$ with respect to s to deduce that there exists a neighbourhood $\mathcal{N}_k^*$ of s_k^* such that (A.5) holds for every $s \in \mathcal{N}_k^*$ and every $\delta_{k,j} \leq \delta_{\min,k}$. Choosing now s_k to ensure (6.4) and $x_k + s_k \in \mathcal{X}$

in addition to (A.5), we obtain that the pair $(s_k, \delta_{k,j})$ satisfies both (6.4) and

$$\overline{\phi}_{m_k,j}^{\delta_{k,j}}(s_k) \le \theta\epsilon_j \, \frac{\delta_{k,j}^j}{j!}.$$

The desired conclusion then follows with

$$\kappa_\delta(\sigma) = \frac{\nu\theta}{q!(6\Theta + 2\sigma)}$$

for any constant $\nu \in (0,1)$. Moreover, $\kappa_\delta(\sigma)$ is clearly a decreasing function of σ.

Coauthorship of publications related to the thesis

Much of the work presented in this thesis has been the object of recent advances in research papers and communications to the scientific community, as reported below.

Submitted papers

- S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *Adaptive Regularization for Nonconvex Optimization Using Inexact Function Values and Randomly Perturbed Derivatives.* Preprint on arXiv, 2020 (`https://arxiv.org/abs/2005.04639`).

Published papers

- S. Bellavia, G. Gurioli. *Stochastic Analysis of an Adaptive Cubic Regularisation Method under Inexact Gradient Evaluations and Dynamic Hessian Accuracy.* Optimization, 2021 (`https://doi.org/10.1080/02331934.2021.1892104`).
- S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *A Stochastic Cubic Regularisation Method with Inexact Function Evaluations and Random Derivatives for Finite Sum Minimisation.* International Conference on Machine Learning (ICML), Conference Paper, 2020 (`https://drive.google.com/file/d/1HpfInwNMv72B2cu0zNW_FILKhxtL3RpJ/view`).
- S. Bellavia, G. Gurioli, B. Morini. *Adaptive Cubic Regularization Methods with Dynamic Inexact Hessian Information and Applications to Finite-Sum minimisation.* IMA Journal of Numerical Analysis **41**(1), 764–799, 2021.
- S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *Adaptive Regularization Algorithms with Inexact Evaluations for Nonconvex Optimization.* SIAM Journal on Optimization **29**(4), 2281–2915, 2019.

Related talks at conferences and during visiting

- *Adaptive Cubic Regularisation Methods under Dynamic Inexact Hessian Information for Nonconvex Optimisation.* University of Oxford (UK), February 11, 2020. Talk given within my visiting research period at the University of Oxford ("Numerical Analysis Group Internal Seminar") under the guidance of Prof. Coralia Cartis[*].

[*]Prof. Coralia Cartis, Associate Professor in Numerical Optimisation, Mathematical Institute and Balliol College, University of Oxford (UK), email: coralia.cartis@maths.ox.ac.uk.

- *Finite-Sum Minimisation via Adaptive Cubic Regularisation Methods*. University of Novi Sad, (Serbia), December 3, 2019. Talk given during my one week visit at the University of Novi Sad, within the mobility project: "Second order methods for optimization problems in machine learning" - Executive program of scientific and technologic cooperation between Italy and Serbia for the years 2019-2021, Ministry of Foreign Affairs.
- *Cubic Regularisation Methods for Optimisation Problems in Machine Learning*. Talk given at the 9-th International Congress of Industrial and Applied Mathematics (ICIAM 2019), July 15, 2019, Universitat de València (ES).
- *Cubic Regularisation Methods with Dynamic Inexact Hessian Information*. Talk given at the conference: "2nd IMA OR Society Conference on Mathematics of Operational Research", April 26, 2019, Aston University, Birmingham (UK).

Other published papers

Last, but not least, I would like to report a series of published papers on the research topic I started after graduation, thanks to a valuable collaboration with Prof. Luigi Brugnano[†], concerning the design, analysis and development of energy-preserving numerical methods for solving Hamiltonian PDEs based on the discrete line integral framework.

- L. Brugnano, G. Gurioli, C. Zhang. *Spectrally Accurate Energy-preserving Methods for the Numerical Solution of the "Good" Boussinesq Equation*. Numerical Methods for Partial Differential Equations **35**(4), 1343–1362, 2019.
- L. Brugnano, G. Gurioli, Y. Sun. *Energy-conserving Hamiltonian Boundary Value Methods for the numerical solution of the Korteweg-de Vries equation*. Journal of Computational and Applied Mathematics **351**, 117–135, 2019.
- L. Brugnano, G. Gurioli, F. Iavernaro. *Predictor-corrector implementation of EQUIP methods*. AIP Conference Proceedings **1978**(1),120004, 2018.
- L. Brugnano, G. Gurioli, F. Iavernaro. *Analysis of Energy and QUadratic Invariant Preserving (EQUIP) methods*. Journal of Computational and Applied Mathematics **335**, 51–73, 2018.
- L. Brugnano, G. Gurioli, F. Iavernaro, E. B. Weinmüller. *Line Integral Solution of Hamiltonian Systems with Holonomic Constraints*. Applied Numerical Mathematics **127**, 56–77, 2018.

[†]Prof. Luigi Brugnano, Full Professor in Numerical Analysis, Dipartimento di Matematica e Informatica (DIMAI) "Ulisse Dini", Università degli Studi di Firenze (Italy), email: luigi.brugnano@unifi.it.

Bibliography

(1) N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, T. Ma. *Finding approximate local minima faster than gradient descent.* Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, 1195–1199, 2017.

(2) A. Bandeira, K. Scheinberg, L. Vicente. *Convergence of trust-region methods based on probabilistic models.* SIAM Journal on Optimization **24**, 1238–1264, 2014.

(3) A. R. Barron. *Universal approximation bounds for superpositions of a sigmoidal function.* IEEE Transactions on Information Theory **39**(3), 930–945, 1993.

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

(5) S. Bellavia, N. Krejic, N. Krklec Jerinkic. *Subsampled Inexact Newton methods for minimizing large sums of convex functions.* IMA Journal of Numerical Analysis **40**(4), 2309–2341, 2020.

(6) S. Bellavia, G. Gurioli. *Stochastic Analysis of an Adaptive Cubic Regularisation Method under Inexact Gradient Evaluations and Dynamic Hessian Accuracy.* Optimization, 2021 (`https://doi.org/10.1080/02331934.2021.1892104`).

(7) S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *A Stochastic Cubic Regularisation Method with Inexact Function Evaluations and Random Derivatives for Finite Sum Minimisation.* International Conference on Machine Learning (ICML), Conference Paper, 2020 (`https://drive.google.com/file/d/1HpfInwNMv72B2cu0zNW_FILKhxtL3RpJ/view`).

(8) S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *Adaptive Regularization for Nonconvex Optimization Using Inexact Function Values and Randomly Perturbed Derivatives.* Submitted for publication, 2020 (`https://arxiv.org/abs/2005.04639`).

(9) S. Bellavia, G. Gurioli, B. Morini. *Adaptive Cubic Regularization Methods with Dynamic Inexact Hessian Information and Applications to Finite-Sum minimisation.* IMA Journal of Numerical Analysis **41**(1), 764–799, 2021.

(10) S. Bellavia, G. Gurioli, B. Morini, Ph. L. Toint. *Adaptive Regularization Algorithms with Inexact Evaluations for Nonconvex Optimization.* SIAM Journal on Optimization **29**(4), 2281–2915, 2019.

(11) S. Bellavia, T. Bianconcini, N. Krejić, B. Morini. *Subsampled first-order optimization methods with applications in imaging.* Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging. Springer, to appear.

(12) S. Bellavia, N. Krejic, B. Morini. *Inexact restoration with subsampled trust-region methods for finite-sum minimization.* Computational Optimization and Applications **73**, 701–736, 2020.

(13) S. Bellavia, S. Gratton, E. Riccietti. *A Levenberg-Marquardt method for large nonlinear least-squares problems with dynamic accuracy in functions and gradients.* Numerische Mathematik **140**, 791–825, 2018.

(14) Y. Bengio. *Practical Recommendations for Gradient-based Training of Deep Architectures. Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 437–478, 2012.

(15) A. S. Berahas, R. Bollapragada, J. Nocedal. *An investigation of Newton-sketch and subsampled Newton methods.* Optimization Methods and Software **35**(4), 661–680, 2020.

(16) A. Berahas, L. Cao, K. Choromanski, K. Scheinberg. *A theoretical and empirical comparison of gradient approximations in derivative-free optimization.* Preprint on arXiv, 2020 (`https://arxiv.org/abs/1905.01332`).

(17) A. Berahas, L. Cao, K. Scheinberg. *Global convergence rate analysis of a generic line search algorithm with noise.* Preprint on arXiv, 2019 (`https://arxiv.org/abs/1910.04055`).

(18) E. Bergou, Y. Diouane, V. Kungurtsev, C. W. Royer. *A subsampling line-search method with second-order results.* Preprint on arXiv, 2018 (`https://arxiv.org/abs/1810.07211`).

(19) E. Bergou, S. Gratton, L. N. Vicente. *Levenberg-Marquardt Methods Based on Probabilistic Gradient Models and Inexact Subproblem Solution, with Application to Data Assimilation.* SIAM/ASA Journal on Uncertainty Quantification **4**(1), 924–951, 2016.

(20) T. Bianconcini, G. Liuzzi, B. Morini, M. Sciandrone. *On the use of iterative methods in cubic regularization for unconstrained optimization.* Computational Optimization and Applications **60**(1), 35–57, 2015.

(21) E. G. Birgin, J. M. Martínez. *A Newton-like method with mixed factorizations and cubic regularization for unconstrained minimization.* Computational Optimization and Applications **73**(3), 707–753, 2019.

(22) E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, Ph. L. Toint. *Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models.* Mathematical Programming, Ser. A, **163**(1-2), 359-368, 2017.

(23) J. Blanchet, C. Cartis, M. Menickelly, K. Scheinberg. *Convergence rate analysis of a stochastic trust region method via supermartingales.* INFORMS Journal on Optimization **1**(2), 92–119, 2019.

(24) R. Bollapragada, R. Byrd, J. Nocedal. *Exact and inexact subsampled Newton methods for optimization.* IMA Journal of Numerical Analysis **39**(2), 545–578, 2019.

(25) D. Bonaiuti, A. Arnone, M. Ermini, L. Baldassarre. *Analysis and Optimization of Transonic Centrifugal Compressor Impellers Using the Design of Experiments Technique.* Journal of Turbomachinery **128**(4), 786–797, 2006.

(26) T. Bonniot. *Convergence and complexity of unconstrained optimization methods with inexact gradients.* Master's thesis, ENSEEIHT, Toulouse, France, 2018 (supervised by S. Gratton, D. Orban and Ph. L. Toint).

(27) L. Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent.* Proceedings of COMPSTAT'2010, 177–186, 2010.

(28) L. Bottou, F. E. Curtis, J. Nocedal. *Optimization Methods for Large-Scale Machine Learning.* SIAM Review **60**(2), 223-311, 2018.

(29) C. P. Brás, J. M. Martínez, M. Raydan. *Large-scale unconstrained optimization using separable cubic modeling and matrix-free subspace minimization.* Computational Optimization and Applications **75**(1), 169–205, 2020.

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

(31) Y. Carmon, J. C. Duchi, O. Hinder, A. Sidford. *Lower bounds for finding stationary points I.* Mathematical Programming, Series A, **184** 71–120, 2020.

(32) Y. Carmon, J. C. Duchi, O. Hinder, O. A. Sidford. *Accelerated methods for nonconvex optimization.* SIAM Journal on Optimization, **28**(2), 1751–1772, 2018.

(33) Y. Carmon, J. C. Duchi. *Gradient descent efficiently finds the cubic-regularized nonconvex Newton step.* Preprint on arXiv, 2016 (`https://arxiv.org/abs/1612.00547`).

(34) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Strong evaluation complexity bounds for arbitrary-order optimization of nonconvex nonsmooth composite functions.* arXiv:2001.10802, 2020.

(35) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints.* SIAM Journal on Optimization **30**(1), 513–541, 2020.

(36) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Worst-case evaluation complexity and optimality of second-order methods for nonconvex smooth optimization.* Proceedings of the 2018 International Conference of Mathematicians (ICM 2018), 3711–3750, 2019.

(37) C. Cartis, N. I. M. Gould, Ph. L. Toint. *An adaptive cubic regularisation algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity.* IMA Journal of Numerical Analysis **32**, 1662–1695, 2012.

(38) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Complexity bounds for second-order optimality in unconstrained optimization.* Journal of Complexity **28**, 93–108, 2012.

(39) C. Cartis, N. I. M. Gould, Ph. L. Toint. *On the oracle complexity of first-order and derivative-free algorithms for smooth nonconvex minimization.* SIAM Journal on Optimization **22**(1), 66–86, 2012.

(40) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Adaptive cubic overestimation methods for unconstrained optimization. Part I: motivation, convergence and numerical results.* Mathematical Programming, Ser. A, **127**, 245–295, 2011.

(41) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Adaptive cubic overestimation methods for unconstrained optimization. Part II: worst-case function and derivative-evaluation complexity.* Mathematical Programming, Ser. A, **130**(2), 295–319, 2011.

(42) C. Cartis, N. I. M. Gould, Ph. L. Toint. *On the complexity of steepest descent, Newton's and regularized Newton's method for nonconvex unconstrained optimization.* SIAM Journal on Optimization **20**(6), 2833–2852, 2010.

(43) C. Cartis, N. I. M. Gould, Ph. L. Toint. *Trust-region and other regularization of linear least-squares problems.* BIT Numerical Mathematics **49**(1), 21–53, 2009.

(44) C. Cartis, K. Scheinberg. *Global convergence rate analysis of unconstrained optimization methods based on probabilistic models.* Mathematical Programming, Ser. A, **169**, 337–375, 2018.

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

(46) C. Chahine, J. R. Seume, T. Verstraete. *The Influence of Metamodeling Techniques on the Multidisciplinary Design Optimization of a Radial Compressor Impeller.* Proceedings of ASME Turbo Expo 2012: Turbine Technical Conference and Exposition, 1951–1964, 2012.

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

(48) M. Checcucci, A. Schneider, M. Marconcini, F. Rubechini, A. Arnone, L. De Franco, M. Coneri. *A novel approach to parametric design of centrifugal pumps for a wide range of specific speeds.* Proceedings of 12th International Symposium on Experimental and Computational Aerother-modynamics of Internal Flows, ISAIF 12 paper nr.121, 2015.

(49) R. Chen, M. Menickelly, K. Scheinberg. *Stochastic optimization using a trust-region method and random models.* Mathematical Programming, Series A, **169**(2), 447–487, 2018.

(50) X. Chen, B. Jiang, T. Lin, S. Zhang. *Adaptively Accelerating Cubic Regularized Newton's Methods for Convex Optimization via Random Sampling.* Preprint on arXiv, 2019
(`https://arxiv.org/abs/1802.05426`).

(51) R. Chen. *Stochastic Derivative-Free Optimization of Noisy Functions.* Theses and Dissertations, Lehigh University, 2015.

(52) A. R. Conn, N. I. M. Gould, Ph. L. Toint. *Trust-Region Methods.* SIAM, Philadelphia, 2000.

(53) C. Cortes, V. N. Vapnik. *Support-Vector Networks.* Machine Learning **20**(3), 273–297, 1995.

(54) F. E. Curtis, K. Scheinberg. *Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning.* Leading Developments from INFORMS Communities, 89–114, 2017.

(55) F. E. Curtis, K. Scheinberg. *Adaptive stochastic optimization: a framework for analyzing stochastic optimization algorithms.* IEEE Signal Processing Magazine **37**(5), 32–42, 2000.

(56) J. E. Dennis, R. B., Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1983. Reprinted as Classics in Applied Mathematics SIAM **16**, Philadelphia, USA, 1996.

(57) J. E. Dennis, J. J. Moré. *A characterization of superlinear convergence and its application to quasi-Newton methods.* Mathematics of Computation **28**, 549–560, 1974.

(58) P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms.* Springer Series in Computational Mathematics **35**, Springer, Berlin, 2004.

(59) E. D. Dolan, J. J. Moré. *Benchmarking optimization software with performance profiles.* Mathematical Programming **91**(2), 201–213, 2002.

(60) J. P. Dussault. *Simple unified convergence proofs for the trust-region and a new ARC variant.* Technical report, University of Sherbrooke, Sherbrooke, Canada, 2015.

(61) L. Ellbrant, L. Eriksson, H. Martensson. *Design of Compressor Blades Considering Efficiency and Stability using CFD Based Optimization.* Proceedings of ASME Turbo Expo 2012: Turbine Technical Conference and Exposition, 371–382, 2012.

(62) P. Girdhar, O. Moniz Octo. *Practical centrifugal pumps: design, operation and maintenance.* Newnes Oxford, 2005.

(63) G. H. Golub, C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press,Baltimore, 3rd ed., 1996.

(64) G. H. Golub, D. P. O'Leary. *Some history of the conjugate gradient methods and the Lanczos algorithms: 1948–1976.* SIAM Review **31**(1), 50–100, 1989.

(65) N. I. M. Gould, S. Lucidi, M. Roma, Ph. L. Toint. *Solving the trust-region subproblem using the Lanczos method.* SIAM Jouenal on Optimization **9**(2), 504–525, 1999.

(66) S. Gratton, E. Simon, Ph. L. Toint. *An algorithm for the minimization of nonsmooth nonconvex functions using inexact evaluations and its worst-case complexity.* Mathematical Programming, Series A, 2020
(`https://link.springer.com/article/10.1007/s10107-020-01466-5`).

(67) S. Gratton, Ph. L. Toint. *A note on solving nonlinear optimization problems in variable precision.* Computational Optimization and Applications **76**, 917–933, 2020.

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

(69) S. Gratton. *Second-order convergence properties of trust-region methods using incomplete curvature information, with an application to multigrid optimization.* Journal of Computational Mathematics **24**(6), 676–692, 2006.

(70) A. Griewank. *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, 1981.

(71) L. Grippo, M. Sciandrone. *Nonmonotone Globalization Techniques for the Barzilai-Borwein Gradient Method.* Computational Optimization and Applications **23**(2), 143–169, 2002.

(72) S. Haykin. *Neural Networks: A Comprehensive Foundation.* 2nd edition. Macmillan, New York, 1998.

(73) P. Hergt. *Pump Research and Development: Past, Present, and Future.* Journal of Fluids Engineering **121**(2), 248–253, 1999.

(74) W. Hoeffding. *Probability Inequalities for Sums of Bounded Random Variables.* Journal of the American Statistical Association **58**(301), 13–30, 1963.

(75) K. Hornik, M. Stinchcombe, H.White. *Multilayer feed- forward networks are universal approximators.* Neural Networks **2**(5), 359–366, 1989.

(76) R. Johnson, Z. Tong. *Accelerating Stochastic Gradient Descent using Predictive Variance Reduction.* Advances in Neural Information Processing Systems, 315–323, 2013.

(77) C. T. Kelley. *Iterative methods for linear and nonlinear equations.* Society for Industrial and Applied Mathematics, 1995.

(78) N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. T. P. Tang. *On large-batch training for deep learning: generalization gap and sharp minima.* Preprint on arXiv, 2016 (`https://arxiv.org/abs/1609.04836v2`).

(79) J. M. Kohler, A. Lucchi. *Subsampled cubic regularization for nonconvex optimization.* 34th International Conference on Machine Learning (ICML 2017), 1895–1904, 2017.

(80) D. P. Kouri, M. Heinkenscloss, D. Rizdal, B. G. van Bloemen-Waanders. *Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty.* SIAM Journal on Scientific Computing **36**(6), A3011–A3029, 2014.

(81) Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE **8**6, 2278–2324, 1998.

(82) J. D. Lee, M. Simchowitz, M. I. Jordan, B. Recht. *Gradient Descent Only Converges to minimizers.* Journal of Machine Learning Research: Workshop and Conference Proceedings **49**, 1–12, 2016.

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

(84) L. Liu, X. Liu, C. J. Hsieh, D. Tao. *Stochastic second-order methods for nonconvex optimization with inexact Hessian and gradient.* Preprint on arXiv, 2018 (`https://arxiv.org/abs/1809.09853`).

(85) H. Liu, K. Wang, S. Yuan, M. Tan, Y. Wang, L. Dong. *Multicondition Optimization and Experimental Measurements of a Double-blade Centrifugal Pump Impeller.* Journal of Fluids Engineering **135**(1), 2013.

(86) A. Maggiary, A. Wachter, I. Dolinskaya, J. Staumz. *A derivative-free trust-region algorithm for the optimization of functions smmothed via Gaussian convolution using adaptive multiple importance sampling.* SIAM Journal on Optimization **28**(2), 1478–1507, 2018.

(87) J. M. Martínez, M. Raydan. *Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization.* Journal of Global Optimization **68**(2), 367–385, 2017.

(88) J. M. Martínez, M. Raydan. *Separable cubic modeling and a trust-region strategy for unconstrained minimization with impact in global optimization.* Journal of Global Optimization **63**(2), 319–342, 2015.

(89) T. Mitchell. *Machine Learning.* New York: McGraw-Hill, 1997.

(90) J. Monodero. *Parametric Design: A Review and Some Experiences.* Automation in Construction **9**(4), 369–377, 2000.

(91) J. J. Moré, D. C. Sorensen. *Computing a trust region step.* SIAM Journal on Scientific and Statistical Computing **4**, 553–572, 1983.

(92) I. Mukherjee, K. Canini, R. Frongillo, Y. Singer. *Parallel Boosting with Momentum.* Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 17–32, 2013.

(93) Y. Nesterov, V. Spokoiny. *Random Gradient-Free Minimization of Convex Functions.* Foundations of Computational Mathematics **17**, 527–566, 2017

(94) Y. Nesterov. *Accelerating the cubic regularization of Newton's method on convex problems.* Mathematical Programming **112**(1), 159–181, 2008.

(95) Y. Nesterov, B. T. Polyak. *Cubic regularization of Newton method and its global performance.* Mathematical Programming, Ser. A, **108**, 177–205, 2006.

(96) Y. Nesterov. *Introductory Lectures on Convex Optimization.* Kluwer Academic Publishers, Dordrecht, 2004.

(97) J. Nocedal, S. Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

(98) D. L. Olson, D. Delen. *Advanced data mining techniques.* Berlin: Springer-Verlag, 2008.

(99) C. Paquette, K. Scheinberg. *A Stochastic Line Search Method with Expected Complexity Analysis.* SIAM Journal on Optimization **30**(1), 349–376, 2020.

(100) B. A. Pearlmutter. *Fast exact multiplication by the Hessian.* Neural computation **6**(1), 147–1660, 1994.

(101) S. Pierret. *Turbomachinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Network.* ASME Journal of Turbomachinery **121** (3), 326–332,1999.

(102) M. Pilanci, M. J. Wainwright. *Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence.* SIAM Journal on Optimization **27**, 205–245, 2017.

(103) M. Raydan. *The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem.* SIAM Journal on Optimization **7**(1), 26–33, 1997.

(104) E. Riccietti, J. Bellucci, M. Checcucci, M. Marconcini, A. Arnone. *Support Vector Machine classification applied to the parametric design of centrifugal pumps.* Engineering Optimization **50**(8), 1304–1324, 2017.

(105) E. Riccietti. *Levenberg-Marquardt methods for the solution of noisy nonlinear least squares problems.* PhD thesis, Università degli Studi di Firenze, Italy, 2017 (supervised by Prof. S. Bellavia).

(106) H. Robbins, S. Monro. *A Stochastic Approximation Mehod.* Tha Annals of Mathematical Statistics **22**(3), 400–407, 1951.

(107) F. Roosta-Khorasani, M.W. Mahoney. *Sub-Sampled Newton Methods.* Mathematical Programming **174**, 293–326, 2019.

(108) C. W. Royer, M. O'Neill, S. J. Wright. *A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization.* Mathematical Programming **180**, 451–488, 2020.

(109) F. Rubechini, A. Schneider, A. Arnone, F. Daccá, C. Canelli, P. Garibaldi. *Aerodynamic redesigning of an industrial gas turbine.* Proceedings of ASME Turbo Expo 2011, 1387–1394, 2011.

(110) F. Rubechini, A. Schneider, A. Arnone, S. Cecchi, and F. A. Malavasi. *A redesign strategy to improve the efficiency of a 17-stage steam turbine.* Proceedings of ASME Turbo Expo 2009, 1463–1470, 2009.

(111) B. Schölkopf, A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* Cambridge, MA: MIT Press, 2001.

(112) N. N. Schraudolph. *Fast curvature matrix-vector products for second-order gradient descent.* Neural computation **14**(7), 1723–1738, 2002.

(113) A. Shiryaev. *Probability, Graduate Texts on Mathematics.* Springer, New York, 1995.

(114) N. Tripuraneni, M. Stern, J. Regier, M. I. Jordan. *Stochastic cubic regularization for fast nonconvex optimization.* Advances in neural information processing systems, 2899–2908, 2018.

(115) J. Tropp. *An Introduction to Matrix Concentration Inequalities.* Number 8,1-2 in Foundations and Trends in Machine Learning. Now Publishing, Boston, USA, 2015.

(116) V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer New York, 2000

(117) V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

(118) V. N. Vapnik, A. Y. Chervonenkis. *Theory of Pattern Recognition.* Nauka, Moscow, 1974.

(119) V. N. Vapnik, A. Y. Chervonenkis. *On the uniform convergence of relative frequencies of events to their probabilities.* Proceedings of the USSR Academy of Sciences **181**(4), 781–783, 1968.

(120) S. A. Vavasis. *Black-box complexity of local minimization.* SIAM Journal on Optimization, **3**(1), 60–80, 1993.

(121) A. Veress, R. Van den Braembussche. *Inverse design and optimization of a return channel for a multistage centrifugal compressor.* Journal of Fluids Engineering **126**(5), 799–806, 2004.

(122) M. Weiser, P. Deuflhard, B. Erdmann. *Affine conjugate adaptive Newton methods for nonlinear elastomechanics.* Optimization Methods and Software **22**(3), 413–431, 2007.

(123) P. Xu, F. Roosta-Khorasani, M. W. Mahoney. *Second-Order Optimization for nonconvex Machine Learning: An Empirical Study.* Proceedings of the 2020 SIAM International Conference on Data Mining, 199-207, 2020.

(124) P. Xu, F. Roosta-Khorasani, M. W. Mahoney. *Newton-Type Methods for nonconvex Optimization Under Inexact Hessian Information.* Mathematical Programming **184**, 35–70, 2020.

(125) Z. Yao, P. Xu, F. Roosta-Khorasani, M. W. Mahoney. *Inexact nonconvex Newton-type methods. (arXiv:1802.06925), 2018.*

(126) X. Zhang, C. Ling, L. Qi. *The best rank-1 approximation of a symmetric tensor and related spherical optimization problems.* SIAM Journal on Matrix Analysis **33**(3), 806–821, 2012.

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