On the oracle complexity of first-order and derivative-free algorithms for smooth nonconvex minimization

C. Cartis[*], N. I. M. Gould[†] and Ph. L. Toint[‡]

22 September 2011

Abstract

The (optimal) function/gradient evaluations worst-case complexity analysis available for the Adaptive Regularizations algorithms with Cubics (ARC) for nonconvex smooth unconstrained optimization is extended to finite-difference versions of this algorithm, yielding complexity bounds for first-order and derivative free methods applied on the same problem class. A comparison with the results obtained for derivative-free methods by Vicente (2010) is also discussed, giving some theoretical insight on the relative merits of various methods in this popular class of algorithms.

Keywords: oracle complexity, worst-case analysis, finite-differences, first-order methods, derivative free optimization, nonconvex optimization.

1 Introduction

We consider algorithms for the solution of the unconstrained (possibly nonconvex) optimization problem

$$\min_{x} f(x) \tag{1.1}$$

where we assume that $f : \mathbb{R}^n \to \mathbb{R}$ is smooth (in a sense to be specified later) and bounded below. All numerical methods for the solution of the general problem (1.1) are iterative and, starting from some initial guess x_0, generate a sequence $\{x_k\}$ of iterates approximating a critical point of f. A variety of algorithms of this form exist, and they are often classified according to their requirements in terms of computing derivatives of the objective function. First-order methods are methods which use $f(x)$ and its gradient $\nabla_x f(x)$, and derivative-free (or zero-th order) methods are those which only use $f(x)$, without any gradient computation. This paper is concerned with estimating worst-case bounds on the number of objective function and/or gradient calls that are necessary for the specific methods in these two classes to compute approximate critical points for (1.1), starting from arbitrary initial guesses x_0. These bounds in turn provide upper bounds on the complexity of solving (1.1) with general algorithms in the first-order or derivative-free classes.

Worst-case complexity analysis for optimization methods probably really started with Nemirovski and Yudin (1983), where the notion of oracle (or black-box) complexity was introduced. Instead of expressing complexity in terms of simple operation counts, the complexity of an algorithm is measured by the number of calls this algorithm makes, in the worst-case, to an oracle (the computation of the objective function or the gradient values, for instance) in order to successfully terminate. Many results of that nature have been derived since, mostly on the convex optimization problem (see, for instance, Nesterov 2004, 2008, Nemirovski, 1994, Agarwal, Bartlett, Ravikummar and Wainwright, 2009), but also for the nonconvex case (see Vavasis 1992*b*, 1992*a*, 1993, Nesterov and Polyak, 2006, Gratton, Sartenaer

[*] School of Mathematics, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JZ, Scotland, UK. Email: coralia.cartis@ed.ac.uk

[†] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK. Email: nick.gould@sftc.ac.uk

[‡] Namur Center for Complex Systems (NAXYS), FUNDP-University of Namur, 61, rue de Bruxelles, B-5000 Namur, Belgium. Email: philippe.toint@fundp.ac.be

and Toint, 2008, Cartis, Gould and Toint 2011*a*, 2010*a*, 2010*b*, 2011*b*, or Vicente, 2010). Of particular interest here is the Adaptive Regularizations with Cubics (ARC) algorithm independently proposed by Griewank (1981), Weiser, Deuflhard and Erdmann (2007) and Nesterov and Polyak (2006), whose worst-case iteration complexity[(1)] was shown in the last of these references to be of $O(\epsilon^{-3/2})$, for finding an approximate solution x_* such that the gradient at x_* is smaller than ϵ in norm. This result was extended by Cartis et al. (2010*a*) to an algorithm no longer requiring the computation of exact second-derivatives, but merely of a suitably accurate approximation[(2)]. Moreover, Cartis et al. (2010*b*, 2011*b*) showed that, when exact second derivatives are used, this complexity bound is tight and is optimal within a large class of second-order methods.

The purpose of the present paper is to use the freedom left in Cartis et al. (2010*a*) to approximate the objective function's Hessian so as to derive complexity bounds for finite-difference methods in exact arithmetic, and thereby establish upper bounds on the oracle complexity of methods for solving unconstrained nonconvex problems, where the oracle consists of evaluating objective-function and/or gradient values. The ARC algorithm and the associated known complexity bounds are recalled in Section 2. Section 3 investigates the case of a first-order variant in which the objective-function's Hessian is approximated by finite differences in gradient values, while Section 4 considers a derivative-free variant where the gradient of f is computed by central differences and its Hessian by forward differences. These results are finally discussed and compared to existing complexity bounds by Vicente (2010) in Section 5.

2 The ARC algorithm and its oracle complexity

The Adaptive Regularization with Cubics (ARC) algorithm is based on the approximate minimization, at iteration k, of (the possibly nonconvex) cubic model

$$m_k(s) = f(x_k) + \langle g_k, s\rangle + \tfrac{1}{2}\langle s, B_k s\rangle + \tfrac{1}{3}\sigma_k\|s\|^3, \tag{2.1}$$

were $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product and $\|\cdot\|$ the Euclidean norm. Here B_k is a symmetric $n\times n$ approximation of $\nabla_{xx}f(x_k)$, $\sigma_k > 0$ is a regularization weight and

$$g_k = \nabla_x m_k(0) = \nabla_x f(x_k). \tag{2.2}$$

By "approximate minimization", we mean that a step s_k is computed that satisfies

$$\langle g_k, s_k\rangle + \langle s_k, B_k s_k\rangle + \sigma_k\|s_k\|^3 \le 0, \tag{2.3}$$

$$\langle s_k, B_k s_k\rangle + \sigma_k\|s_k\|^3 \ge 0 \tag{2.4}$$

$$m_k(s_k) \le m_k(s_k^{\rm C}) \tag{2.5}$$

with

$$s_k^{\rm C} = -\alpha_k^{\rm C} g_k \quad\text{and}\quad \alpha_k^{\rm C} = \arg\min_{\alpha\ge 0} m_k(-\alpha g_k), \tag{2.6}$$

and

$$\|\nabla_x m_k(s_k)\| = \|g_k + B_k s_k + (\sigma_k\|s_k\|)s_k\| \le \kappa_\theta \min[1, \|s_k\|]\,\|g_k\|, \tag{2.7}$$

for some given constant $\kappa_\theta \in (0,1)$.

As noted in Cartis et al. (2010*a*), conditions (2.3) and (2.4) must hold if s_k minimizes the model along the direction $s_k/\|s_k\|$, while (2.7) holds by continuity if s_k is sufficiently close to a first-order critical point of m_k. Moreover, (2.5)-(2.6) are nothing but the familiar Cauchy-point decrease condition. Fortunately, these conditions can be ensured algorithmically. In particular, conditions (2.3)–(2.7) hold if s_k is a (computable) global minimizer of m_k (see Griewank, 1981, Nesterov and Polyak, 2006, see also Cartis, Gould and Toint, 2009). Note that, since $\nabla_x m_k(0) = \nabla_x f(x_k)$, (2.7) may be interpreted as requiring a relative reduction in the norm of the model's gradient at least equal to $\kappa_\theta \min[1, \|s_k\|]$.

The ARC algorithm may then be stated as presented on the following page.

[(1)] That is its oracle complexity for a choice of the oracle corresponding to the computation of the objective function and its first and second derivatives.

[(2)] This method also abandoned global optimization of the underlying cubic model and avoided an *a priori* knowledge of the objective function's Hessian's Lipschitz constant, two assumptions made by Nesterov and Polyak (2006).

Algorithm 2.1: ARC

Step 0: An initial starting point x_0 is given, as well as a user-defined accuracy threshold $\epsilon \in (0,1)$ and constants $\gamma_2 \geq \gamma_1 > 1$, $1 > \eta_2 \geq \eta_1 > 0$ and $\sigma_0 > 0$. Set $k = 0$.

Step 1: If $\|\nabla_x f(x_k)\| \leq \epsilon$, terminate with approximate solution x_k.

Step 2: Compute any Hessian approximation B_k.

Step 3: Compute a step s_k satisfying (2.3)–(2.7).

Step 4: Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}. \tag{2.8}$$

Step 5: Set

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if} \quad \rho_k \geq \eta_1, \\ x_k & \text{otherwise.} \end{cases}$$

Step 6: Set

$$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if} \quad \rho_k > \eta_2, \\ [\sigma_k, \gamma_1 \sigma_k] & \text{if} \quad \eta_1 \leq \rho_k \leq \eta_2, \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise.} \end{cases} \tag{2.9}$$

Step 7: Increment k by one and return to Step 1.

We denote by

$$\mathcal{S} = \{k \geq 0 \mid \rho_k \geq \eta_1\}$$

the set of *successful iterations*, and

$$\mathcal{S}_j = \{k \in \mathcal{S} \mid k \leq j\} \quad \text{and} \quad \mathcal{U}_j = \{0, \ldots, j\} \setminus \mathcal{S}_j, \tag{2.10}$$

the sets of successful and unsuccessful iterations up to iteration j.

It is not the purpose of the present paper to discuss implementation issues or convergence theory for the ARC algorithm, but we need to recall from Cartis et al. (2010*a*) the main complexity results for this method, as well as the assumptions under which these hold.

We first restate our assumptions.

A.1: The objective function f is twice continuously differentiable on $\mathbb{R}^n$ and its gradient and Hessian are Lipschitz continuous on the path of iterates with Lipschitz constants L_g and L_H, respectively, i.e., for all $k \geq 0$ and all $\alpha \in [0,1]$,

$$\|\nabla_x f(x_k) - \nabla_x f(x_k + \alpha s_k)\| \leq L_g \alpha \|s_k\| \tag{2.11}$$

and

$$\|\nabla_{xx} f(x_k) - \nabla_{xx} f(x_k + \alpha s_k)\| \leq L_H \alpha \|s_k\|. \tag{2.12}$$

A.2: The objective function f is bounded below, that is there exists a constant $f_{\rm low} > -\infty$ such that

$$f(x) \geq f_{\rm low}$$

for all $x \in \mathbb{R}^n$

A.3: For all $k \geq 0$, the Hessian approximation B_k satisfies

$$\|B_k\| \leq \kappa_{\rm B} \tag{2.13}$$

and

$$\|(\nabla_{xx} f(x_k) - B_k)s_k\| \leq \kappa_{\text{BH}} \|s_k\|^2 \tag{2.14}$$

for some constants $\kappa_{\text{B}} > 0$ and $\kappa_{\text{BH}} > 0$.

We start by noting that the form of the cubic model (2.1) ensures a crucial bound on the step norm and model decrease.

Lemma 2.1 *Suppose that we apply the ARC algorithm to problem (1.1), and also that (2.3), (2.4) and (2.5) hold. Then*

$$\|s_k\| \leq \frac{3}{\sigma_k} \max\left[\|B_k\|, \sqrt{\sigma_k \|g_k\|}\right] \tag{2.15}$$

and

$$m_k(s_k) \leq f(x_k) - \tfrac{1}{6}\sigma_k \|s_k\|^3. \tag{2.16}$$

Proof. See Lemma 2.2 in Cartis et al. (2011*a*) for the proof of (2.15) and Lemma 4.2 in Cartis et al. (2010*a*) for that of (2.16). □

For our purposes it is also useful to consider the following bounds on the value of the regularization parameter.

Lemma 2.2 *Suppose that we apply the ARC algorithm to problem (1.1), and also that* ***A.1*** *and (2.13) hold. Then there exists a constant $\kappa_\sigma > 0$ independent of n such that, for all $k \geq 0$*

$$\sigma_k \leq \max\left[\sigma_0, \frac{\kappa_\sigma}{\epsilon}\right]. \tag{2.17}$$

If, in addition, (2.14) also holds, then there exists a constant $\sigma_{\max} > 0$ independent of n and ϵ such that, for all $k \geq 0$

$$\sigma_k \leq \sigma_{\max}. \tag{2.18}$$

Proof. See Lemmas 3.2 and 3.3 in Cartis et al. (2010*a*) for the proof of (2.17) and Lemma 5.2 in Cartis et al. (2011*a*) for that of (2.18). Note that both of these proofs crucially depend on the identity (2.2), which means they have to be revisited if this equality fails. □

Without loss of generality, we assume in what follows that ϵ is small enough for the second term in the max of (2.17) to dominate, and thus that (2.17) may be rewritten to state that, for all $k \geq 0$

$$\sigma_k \leq \frac{\kappa_\sigma}{\epsilon}. \tag{2.19}$$

If (2.18) holds, then, crucially, the step s_k can then be proved to be sufficiently long compared to the gradient's norm at iteration $k+1$.

Lemma 2.3 *Suppose that we apply the ARC algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.3*** *hold. Then, for all $k \geq 0$, one has that, for some $\kappa_g > 0$ independent of n,*

$$\|s_k\| \geq \kappa_g \sqrt{\|\nabla_x f(x_k + s_k)\|}. \tag{2.20}$$

Proof. See Lemma 5.2 in Cartis et al. (2010*a*). □

The final important observation in the complexity analysis is that the total number of iterations required by the ARC algorithm to terminate may be bounded in terms of the number of successful iterations needed.

Lemma 2.4 *Suppose that we apply the ARC algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.3*** *hold and, for any fixed $j \geq 0$, let $\mathcal{S}_j$ and $\mathcal{U}_j$ be defined in (2.10). Assume also that*

$$\sigma_k \geq \sigma_{\min} \tag{2.21}$$

for all $k \leq j$ and some $\sigma_{\min} > 0$. Then one has that

$$|\mathcal{U}_j| \leq \left\lceil (|\mathcal{S}_j| + 1) \frac{1}{\log \gamma_1} \log\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right) \right\rceil. \tag{2.22}$$

Proof. See Theorem 2.1 in Cartis et al. (2010*a*). Observe that this proof uniquely depends on the mechanism used in the algorithm for updating σ_k, and it is independent of the values of g_k or B_k. □

Combining those results and using **A.2** then yields the following oracle complexity theorem.

Theorem 2.5 *Suppose that we apply the ARC algorithm to problem (1.1), and also that **A.1**–**A.3** hold, that $\epsilon \in (0,1)$ is given and that (2.21) holds. Then the algorithm terminates after at most*

$$\mathrm{N}_1^s \stackrel{\rm def}{=} 1 + \left\lceil \kappa_{\rm S}^s \epsilon^{-3/2} \right\rceil, \tag{2.23}$$

successful iterations and at most

$$\mathrm{N}_1 \stackrel{\rm def}{=} \left\lceil \kappa_{\rm S} \epsilon^{-3/2} \right\rceil \tag{2.24}$$

iterations in total, where

$$\kappa_{\rm S}^s \stackrel{\rm def}{=} (f(x_0) - f_{\rm low})/(\eta_1 \alpha_{\rm S}), \quad \alpha_{\rm S} \stackrel{\rm def}{=} (\sigma_{\min} \kappa_g^3)/6 \tag{2.25}$$

and

$$\kappa_{\rm S} \stackrel{\rm def}{=} (1 + \kappa_{\rm S}^u)(2 + \kappa_{\rm S}^s), \quad \kappa_{\rm S}^u \stackrel{\rm def}{=} \log(\sigma_{\max}/\sigma_{\min})/\log\gamma_1, \tag{2.26}$$

with κ_g and $\sigma_{\max}$ defined in (2.20) and (2.18), respectively. As a consequence, the algorithm terminates after at most N_1^s gradient evaluations and at most N_1 objective function evaluations.

Proof. See Corollary 5.3 in Cartis et al. (2010*a*). □

The bound given by (2.23) is known to be qualitatively[(3)] tight and optimal for a wide class of second-order methods (see Cartis et al. 2010*b*, 2011*b*).

3 A first-order finite-difference ARC variant

The objective of this section is to extend the ARC algorithm to a version using finite differences in gradients to compute the Hessian approximation B_k. If the accuracy of the finite-difference scheme is high enough to ensure that (2.14) holds, then one might expect that a worst-case iteration complexity similar to (2.23)-(2.24) would hold, thereby providing a first worst-case oracle complexity estimate for first-order methods applied on nonconvex unconstrained problems.

For defining this algorithm, which we will refer to as the ARC-FDH algorithm, we only need to specify the details of the estimation of B_k. We consider computing this latter matrix by first using n forward gradient differences at x_k with stepsize h_k, and then symmetrizing the result, that is

$$[A_k]_{i,j} = \left[\frac{\nabla_x f(x_k) - \nabla_x f(x_k + h_k e_j)}{h_k}\right]_i \quad \text{and} \quad B_k = \tfrac{1}{2}(A_k + A_k^T), \tag{3.1}$$

(where e_j is the j-th vector of the canonical basis). It is well known (see Nocedal and Wright, 1999, Section 7.1) that

$$\|\nabla_{xx} f(x_k) - B_k\| \le \kappa_{\rm eHg} h_k \tag{3.2}$$

for some constant $\kappa_{\rm eHg} \in [0, L_H]$. The only remaining issue is therefore to define a procedure guaranteeing that

$$h_k \le \kappa_{\rm hs} \|s_k\|. \tag{3.3}$$

for some $\kappa_{\rm hs} > 0$ and all $k \ge 0$. As we show below, this can be achieved if we consider the ARC-FDH algorithm on the next page, where $\kappa_{\rm hs} \ge 1$.

[(3)] The constants may not be optimal.

Algorithm 3.1: ARC-FDH

Step 0: An initial starting point x_0 is given, as well as a user-defined accuracy threshold $\epsilon \in (0,1)$ and constants $\gamma_2 \geq \gamma_1 > 1$, $\gamma_3 \in (0,1)$, $1 > \eta_2 \geq \eta_1 > 0$ and $\sigma_0 > 0$. If $\|\nabla_x f(x_0)\| \leq \epsilon$, terminate. Otherwise, set $k = 0$, $j = 0$ and choose an initial stepsize $h_{0,0} \in (0,1]$.

Step 1: Estimate $B_{k,j}$ using (3.1) with stepsize $h_{k,j}$.

Step 2: Compute a step $s_{k,j}$ satisfying (2.3)–(2.7).

Step 3: Compute $\nabla_x f(x_k + s_{k,j})$. If $\|\nabla_x f(x_k + s_{k,j})\| \leq \epsilon$, terminate with approximate solution $x_k + s_{k,j}$.

Step 4: If

$$h_{k,j} > \kappa_{\rm hs}\|s_{k,j}\|, \tag{3.4}$$

set $h_{k,j+1} = \gamma_3 h_{k,j}$, increment j by one and return to Step 1. Otherwise, set $s_k = s_{k,j}$ and $h_k = h_{k,j}$.

Step 5: Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}. \tag{3.5}$$

Step 6: Set

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if} \quad \rho_k \geq \eta_1, \\ x_k & \text{otherwise.} \end{cases}$$

Step 7: Set

$$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if} \quad \rho_k > \eta_2, \\ [\sigma_k, \gamma_1 \sigma_k] & \text{if} \quad \eta_1 \leq \rho_k \leq \eta_2, \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise.} \end{cases} \tag{3.6}$$

Step 8: Set $h_{k+1,0} = h_k$ and $j = 0$. Increment k by one and return to Step 1 if $\rho_k \geq \eta_1$, or to Step 2 otherwise.

By convention and analogously to our notation for s_k and h_k, we denote by B_k the approximation $B_{k,j}$ obtained at the end of the loop between Steps 1 and 4. Clearly, the test (3.4) in Step 4 ensures that (3.3) holds, as requested. Observe that, because the norm of the step is a monotonically decreasing function as a function of σ_k (see Lemma 3.1 in Cartis et al., 2009), it decreases at an unsuccessful iteration, which might then possibly require a new evaluation of the approximate Hessian in order to preserve (3.3). Observe also that the mechanism of the algorithm implies that the positive sequence $\{h_k\}$ is non-increasing and bounded above by $h_{0,0} \leq 1$.

It now remains to show that this algorithm is well defined, which we do under the additional assumption that the (true) gradients remain bounded at all iterates. Since the sequence $\{f(x_k\}$ is monotonically decreasing, this condition can for instance be ensured by assuming bounded gradients of the level set $\{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$.

A.4: There exists a constant $\kappa_{\rm ubg} \geq 0$ such that, for all $k \geq 0$

$$\|\nabla_x f(x_k)\| \leq \kappa_{\rm ubg}.$$

Lemma 3.1 *Suppose that we apply the ARC-FDH algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.4*** *hold. Then (2.13) holds with*

$$\kappa_{\rm B} \stackrel{\rm def}{=} \max[\,\kappa_{\rm eHg} + L_g,\, \sqrt{\kappa_\sigma \kappa_{\rm ubg}}\,] \geq \sqrt{\kappa_\sigma \kappa_{\rm ubg}} \tag{3.7}$$

and, for all $k \geq 0$ and all $j \geq 0$,

$$\|s_{k,j}\| \geq \frac{(1-\kappa_\theta)\,\epsilon}{\max\left[\,4\kappa_{\rm B}, \kappa_{\rm B} + 3\sqrt{\sigma_k \kappa_{\rm ubg}}\,\right]} \tag{3.8}$$

Proof. We first note that (2.11) ensures that $\|\nabla_{xx} f(x_k)\| \leq L_g$ for all $k \geq 0$ and therefore that

$$\|B_{k,j}\| \leq \|B_{k,j} - \nabla_{xx} f(x_k)\| + \|\nabla_{xx} f(x_k)\| \leq \kappa_{\rm eHg} + L_g \leq \max[\,\kappa_{\rm eHg} + L_g,\ \sqrt{\kappa_\sigma \kappa_{\rm ubg}}\,], \tag{3.9}$$

where we used the triangle inequality, the bound $h_{k,j} \leq h_{0,0} \leq 1$ and (3.2). Hence (2.13) holds with (3.7). Observe now that (2.2) and the mechanism of the algorithm then implies that, as long as the algorithm has not terminated,

$$\|g_k\| > \epsilon. \tag{3.10}$$

We know from (2.7) and (2.2) that, for all $k \geq 0$,

$$\kappa_\theta \min[1, \|s_{k,j}\|]\, \|g_k\| \geq \|\nabla_x m_k(0) + B_{k,j}s_{k,j} + (\sigma_k\|s_{k,j}\|)s_{k,j}\| \geq \|g_k\| - \|B_{k,j}s_{k,j} + (\sigma_k\|s_{k,j}\|)s_{k,j}\|,$$

and thus, using (3.10), that

$$\|B_{k,j}s_{k,j} + (\sigma_k\|s_{k,j}\|)s_{k,j}\| \geq (1-\kappa_\theta)\|g_k\| > (1-\kappa_\theta)\epsilon.$$

Taking this bound, (2.13) with (3.7), (2.15), (2.2) and **A.4** into account, we deduce that

$$\begin{array}{rcl}(1-\kappa_\theta)\epsilon & < & \kappa_{\rm B}\|s_{k,j}\| + \sigma_k\|s_{k,j}\|^2 \\ & \leq & \left\{\kappa_{\rm B} + 3\max\left[\|B_{k,j}\|, \sqrt{\sigma_k\|g_k\|}\,\right]\right\}\|s_{k,j}\| \\ & \leq & \left\{\,\kappa_{\rm B} + 3\max\left[\,\kappa_{\rm B}, \sqrt{\sigma_k\kappa_{\rm ubg}}\,\right]\,\right\}\|s_{k,j}\|,\end{array}$$

proving (3.8). □

We are now able to deduce that the inner loop of the ARC-FDH algorithm terminates in a bounded number of iterations and hence that the desired accuracy on the Hessian approximation is obtained.

Lemma 3.2 *Suppose that we apply the ARC-FDH algorithm to problem (1.1), and also that **A.1**, **A.4** and (2.21) hold. Then the total number of times where a return from Step 4 to Step 1 is executed in the algorithm is bounded above by*

$$\left\lceil \frac{\log\kappa_h + \frac{3}{2}\log\epsilon}{\log\gamma_3}\right\rceil_+ \tag{3.11}$$

*where $\kappa_h > 0$ is a constant independent of n and where $\lceil\alpha\rceil_+$ denotes the maximum of zero and the first integer larger than or equal to α. Moreover **A.3** holds.*

Proof. The inequality (3.8) and (2.19) give that, for $j \geq 0$,

$$(1-\kappa_\theta)\epsilon \leq \max\left[4\kappa_{\rm B}, \kappa_{\rm B} + 3\sqrt{\frac{\kappa_\sigma\kappa_{\rm ubg}}{\epsilon}}\right]\|s_{k,j}\| \leq \frac{4\kappa_{\rm B}}{\epsilon^{1/2}}\|s_{k,j}\|, \tag{3.12}$$

where we have used the bound $\kappa_{\rm B} \geq \sqrt{\kappa_\sigma\kappa_{\rm ubg}}$ and the inclusion $\epsilon \in (0,1)$ to deduce the last inequality. Now the loop between Steps 1 and 4 of the ARC-FDH algorithm terminates as soon as (3.4) is violated, which must happen if j is large enough to ensure that

$$h_{k,j} = \gamma_3^j h_{k,0} \leq \gamma_3^j \leq \frac{\kappa_{\rm hs}(1-\kappa_\theta)}{4\kappa_{\rm B}}\epsilon^{3/2} \leq \kappa_{\rm hs}\|s_{k,j}\|, \tag{3.13}$$

where we have successively used the mechanism of the algorithm, and (3.12). The second inequality in (3.13) and the decreasing nature of the sequence $\{h_k\}$ then ensures that (3.3) must hold for all j after at most (3.11) (with $\kappa_h = \kappa_{\rm hs}(1-\kappa_\theta)/4\kappa_{\rm B}$) reductions of the stepsize by γ_3, which proves the first part of the lemma. Finally, (3.3) and (3.2) imply also that (2.14) holds for B_k. This with (2.13) ensures that **A.3** is satisfied. □

We may then conclude with our main result for this section.

Theorem 3.3 *Suppose that we apply the ARC-FDH algorithm to problem (1.1), and also that* ***A.1***, ***A.2*** *and* ***A.4*** *hold, that $\epsilon \in (0,1)$ is given and that (2.21) holds. Then the algorithm terminates after at most*

$$\mathrm{N}_1^s \stackrel{\rm def}{=} 1 + \left\lceil \kappa_{\rm S}^s \epsilon^{-3/2} \right\rceil, \tag{3.14}$$

successful iterations and at most

$$\mathrm{N}_1 \stackrel{\rm def}{=} \left\lceil \kappa_{\rm S} \epsilon^{-3/2} \right\rceil \tag{3.15}$$

iterations in total, where $\kappa_{\rm S}^s$ and $\kappa_{\rm S}$ are given by (2.25) and (2.26), respectively. As a consequence, the algorithm terminates after at most

$$(n+1)\mathrm{N}_1^s + n \left\lceil \frac{\log \kappa_h + \frac{3}{2}\log\epsilon}{\log \gamma_3} \right\rceil_+ \tag{3.16}$$

gradient evaluations and at most N_1 objective function evaluations.

Proof. Lemma 3.2 ensures that **A.3** holds. Theorem 2.5 is thus applicable and the number of successful iterations is therefore bounded by (2.23), while the total number of iterations is bounded by (2.24). The bound (3.16) and the bound of the number of function evaluations then follows from Lemma 3.2 and the observation that, in addition to the computation of $\nabla_x f(x_k)$ (at successful iterations only) and $f(x_k)$, each successful iteration involves an estimation of the Hessian by finite differences, each of which requires n gradient evaluations, plus possibly at most (3.11) additional Hessian estimations at the same cost. □

Very broadly speaking, we therefore require at most

$$O\left(n\left[\left\lceil \frac{1}{\epsilon^{3/2}}\right\rceil + \lceil |\log\epsilon|\rceil\right]\right) \tag{3.17}$$

gradient and

$$O\left(\left\lceil \frac{1}{\epsilon^{3/2}}\right\rceil\right)$$

function evaluations in the worst-case. Both bounds are qualitatively very similar to the bound (2.24) for the original ARC algorithm.

We close this section by observing that better bounds may be obtained by reconsidering the technique used to decrease h_k. The technique described in Algorithm ARC-DFH is based on an linear decrease, specifically by the choice $h_{k,j+1} = \gamma_3 h_{k,j}$, leading, as explained in the proof of Lemma 3.2, to a factor $\log\epsilon$ (see (3.11)). We could equally choose a faster exponential decrease, with $h_{k,j+1} = h_{k,j}^\alpha$ for any $\alpha > 1$, and $h_{k,0} < 1$, leading to a bound of the form

$$\left\lceil \frac{\log[\log\kappa_h + \frac{3}{2}\log\epsilon] - \log\log h_{k,0}}{\log\alpha} \right\rceil_+$$

instead of (3.11). In fact, an arbitrarily slow increase in ϵ for the latter bound can be achieved by selecting a suitably fast decreasing scheme for h_k. However, the significance of such improvements is limited when one measures their impact on the overall complexity of the algorithm. Indeed, for values of ϵ sufficiently small to be of interest, $|\log\epsilon| < \epsilon^{-3/2}$ and the term $(n+1)N_1^s$ completely dominates the second term in the bound (3.16). Decreasing the second term, even significantly, therefore results in a very marginal theoretical improvement.

Better bounds can also be obtained if we assume that the Hessian has a known sparsity pattern. The finite-difference scheme my then be adapted (see Powell and Toint, 1979, or Goldfarb and Toint, 1984) to require much fewer than n gradient differences to obtain a Hessian approximation, in which case the factor n in (3.17) may often be replaced by a small constant. Similar gains can be obtained if f is partially separable (Griewank and Toint, 1982). Finally, parallel evaluations of the gradient in Step 1 may also result in substantial computational savings.

4 A derivative-free ARC variant

We are now interested in pursuing the same idea further and considering a derivative-free variant of the ARC algorithm, where both gradients and Hessians are approximated by finite differences. However, this introduces two additional difficulties: the approximation techniques used for the gradient and Hessian should be clarified, and some results we relied on in the previous section (in particular Lemmas 2.2 and 2.3) have to be revisited because they depend on the true gradient of the objective function, which is no longer available.

Consider the approximation of gradients and Hessians first. From the discussion above, we see that preserving (2.14) is necessary for using results for the original ARC algorithm. It is then natural to seek a higher degree of accuracy for the gradient itself, since this is the quantity that the algorithm drives to zero. We therefore suggest using a central difference scheme for the gradient, approximating the i-th component of the gradient at x_k by

$$[g_k]_i = \frac{f(x_k + t_k e_i) - f(x_k - t_k e_i)}{2t_k} \tag{4.1}$$

for some stepsize $t_k > 0$. It is well-known (see Nocedal and Wright, 1999, Section 7.1) that such a scheme ensures the bound

$$\|\nabla_x f(x_k) - g_k\| \le \kappa_{\rm egt} t_k^2 \tag{4.2}$$

for some constant $\kappa_{\rm egt} \in [0, L_H]$, where g_k is now the vector *approximating* $\nabla_x f(x_k)$, i.e. whose i-th component is given by (4.1). Similarly, we may approximate the (i,j)-th entry of the Hessian at x_k by a difference quotient and symmetrize the result, yielding

$$[A_k]_{i,j} = \frac{f(x_k + t_k e_i + t_k e_j) - f(x_k + t_k e_i) - f(x_k + t_k e_j) + f(x_k)}{t_k^2} \quad \text{and} \quad B_k = \tfrac{1}{2}(A_k + A_k^T) \tag{4.3}$$

(see Nocedal and Wright, 1999, Section 7.1). This implies the error bound

$$\|\nabla_{xx} f(x_k) - B_k\| \le \kappa_{\rm eHt} t_k \tag{4.4}$$

for some constant $\kappa_{\rm eHt} \in [0, L_H]$. Note that (4.4) gives the same type of error bound as (3.2) above, and we are again interested in an algorithm which guarantees (2.14) from (4.4), i.e. such that

$$t_k \le \kappa_{\rm ts} \|s_k\| \tag{4.5}$$

for all $k \ge 0$ and some constant $\kappa_{\rm ts} > 0$.

The gradient approximation scheme also raises the question of proper termination of any algorithm using g_k rather than $\nabla_x f(x_k)$. Since this latter quantity is unavailable by assumption, it is impossible to test its norm against the threshold ϵ. The next best thing is to test $\|g_k\|$ for a sufficiently small difference stepsize t_k. More specifically, if

$$\|g_k\| \le \tfrac{1}{2}\epsilon \quad \text{and} \quad t_k \le \sqrt{\frac{\epsilon}{2\kappa_{\rm egt}}} \stackrel{\rm def}{=} t_\epsilon \tag{4.6}$$

then (4.2) and the triangle inequality ensure that $\|\nabla_x f(x_k)\| \le \epsilon$, as requested. In what follows, we assume that we know a suitable value for $\kappa_{\rm egt}$ or, equivalently, of t_ϵ, and then use (4.6) for detecting an approximate first-order critical point. *The worst-case complexity is therefore to be understood as the maximum number of function evaluations necessary for the test (4.6) to hold.*

Using these ideas, we may now state the ARC-DFO variant of the ARC algorithm on the following page, where $\gamma_3 \in (0,1)$.

As was the convention for the ARC-FDH algorithm above, we denote by B_k, g_k and g_k^+ the quantities $B_{k,j}$, $g_{k,j}$ and $g_{k,j}^+$ obtained at the end of the loop between Steps 3 and 7 (we show below that this loop terminates finitely). It is also clear that the stepsizes t_k are monotonically decreasing. We also see that Step 7 ensures (4.5). We next verify that the Hessian approximations remains bounded and that loop between Steps 3 and 7 always terminates after a finite number of iterations.

Algorithm 4.1: ARC-DFO

Step 0: An initial starting point x_0 is given, as well as a user-defined accuracy threshold $\epsilon \in (0,1)$ and constants $\gamma_2 \geq \gamma_1 > 1$, $1 > \eta_2 \geq \eta_1 > 0$ and $\sigma_0 > 0$. Choose a stepsize $t_{0,0} \leq t_\epsilon$. Set $k = 0$ and $j = 0$.

Step 1: Estimate $g_{0,0}$ using (4.1) with stepsize $t_{0,j}$.

Step 2: If $\|g_{0,j}\| \leq \frac{1}{2}\epsilon$, terminate with approximate solution x_0.

Step 3: Estimate $B_{k,j}$ using (4.3) with stepsize $t_{k,j}$.

Step 4: Compute a step $s_{k,j}$ satisfying (2.3)–(2.7).

Step 5: Estimate $g^+_{k,j}$ using (4.1) with x_k replaced by $x_k + s_{k,j}$ and the stepsize $t_{k,j}$.

Step 6: If $\|g^+_{k,j}\| \leq \frac{1}{2}\epsilon$, terminate with approximate solution $x_k + s_{k,j}$.

Step 7: If

$$t_{k,j} > \kappa_{\text{ts}} \min[\|s_{k,j}\|, \|g_{k,j}\|] \tag{4.7}$$

set $t_{k,j+1} = \gamma_3 t_{k,j}$, increment j by one and return to Step 3. Otherwise, set $s_k = s_{k,j}$ and $t_k = t_{k,j}$.

Step 8: Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}. \tag{4.8}$$

Step 9: Set

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if} \quad \rho_k \geq \eta_1, \\ x_k & \text{otherwise,} \end{cases} \quad \text{and} \quad g_{k+1,0} = \begin{cases} g^+_{k,j} & \text{if} \quad \rho_k \geq \eta_1, \\ g_{k,j} & \text{otherwise.} \end{cases}$$

Step 10: Set

$$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if} \quad \rho_k > \eta_2, \\ [\sigma_k, \gamma_1 \sigma_k] & \text{if} \quad \eta_1 \leq \rho_k \leq \eta_2, \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise.} \end{cases} \tag{4.9}$$

Step 11: Set $t_{k+1,0} = t_k$ and $j = 0$. Increment k by one and return to Step 3 if $\rho_k \geq \eta_1$ or to Step 4 otherwise.

Lemma 4.1 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.4*** *hold. Then there exist constants $\kappa_{\text{B}} > 1$ and $\kappa_{\text{ng}} > 0$ such that, if $B_{k,j}$ is estimated at Step 3, then*

$$\|g_k\| \leq \kappa_{\text{ng}} \quad \textit{and} \quad \|B_k\| \leq \kappa_{\text{B}}. \tag{4.10}$$

for all $k \geq 0$. Moreover, we have that, for all $j \geq 0$,

$$\|s_{k,j}\| \geq \frac{(1 - \kappa_\theta)\,\epsilon}{\max\left[\,4\kappa_{\text{B}}, \kappa_{\text{B}} + 3\sqrt{\sigma_k \kappa_{\text{ubg}}}\,\right]} \tag{4.11}$$

and there exists a $\kappa(\sigma_k) > 0$ such that, at iteration k of the algorithm, the loop between Steps 3 and 7 terminates in at most

$$\left\lceil \frac{\log \kappa(\sigma_k) + \log \epsilon}{\log \gamma_3} \right\rceil_+ \tag{4.12}$$

iterations. Finally, the inequalities

$$\|g_k - \nabla_x f(x_k)\| \le \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2, \tag{4.13}$$

$$\|g_k^+ - \nabla_x f(x_k+s_k)\| \le \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 \tag{4.14}$$

and

$$\|B_k - \nabla_{xx} f(x_k)\| \le \kappa_{\rm eHt}\kappa_{\rm ts}\|s_k\| \tag{4.15}$$

hold for each $k \ge 0$.

Proof. Consider iteration k. As in Lemma 3.1, we obtain that $\|B_{k,j}\| \le \kappa_{\rm B}$ and therefore that the second inequality in (4.10) holds. The proof of the first is similar in spirit:

$$\|g_k\| \le \|g_k - \nabla_x f(x_k)\| + \|\nabla_x f(x_k)\| \le \kappa_{\rm egt} + \kappa_{\rm ubg} \stackrel{\rm def}{=} \kappa_{\rm ng},$$

where we used (4.2), the inequality $t_{k,j} \le t_{0,0} \le 1$ and **A.4**. Observe now that the mechanism of the algorithm implies that, as long as the algorithm has not terminated,

$$\|g_k\| \ge \tfrac{1}{2}\epsilon. \tag{4.16}$$

As in the proof of Lemma 3.1 (using (4.16) instead of (3.10)), we may now derive that (4.11) holds for all k and all $j \ge 0$. Defining

$$\mu(\sigma_k) \stackrel{\rm def}{=} \frac{1-\kappa_\theta}{\max\left[\,4\kappa_{\rm B}, \kappa_{\rm B} + 3\sqrt{\sigma_k\kappa_{\rm ubg}}\,\right]}$$

this lower bound may then be used to deduce that the loop between Steps 3 and 7 terminates as soon as (4.7) is violated, which must happen if j is large enough to ensure that

$$t_{k,j} = \gamma_3^j t_{k,0} \le \gamma_3^j \le \kappa_{\rm ts}\min\left[\mu(\sigma_k), \tfrac{1}{2}\right]\epsilon \le \kappa_{\rm ts}\min[\|s_{k,j}\|, \|g_k\|], \tag{4.17}$$

where we used (4.16) to derive the last inequality. This implies that j never exceeds

$$\left\lceil \frac{\log\{[\kappa_{\rm ts}\min[\mu(\sigma_k), \frac{1}{2}]\} + \log\epsilon}{\log\gamma_3} \right\rceil_+,$$

which in turn yields (4.12) with $\kappa(\sigma_k) \stackrel{\rm def}{=} \kappa_{\rm ts}\min[\mu(\sigma_k), \frac{1}{2}]$. Since the loop between Steps 3 and 7 always terminates finitely, (4.5) holds for all $k \ge 0$ and the inequalities (4.13)–(4.15) then follow from (4.2) and (4.4). □

Unfortunately, several of the basic properties of the ARC algorithm mentioned in Section 2 can no longer be extended here. This is the case of (2.19), (2.18) and (2.20), which we thus need to reconsider.

The proof of (2.19) is involved and needs to be restarted from the Cauchy condition (2.5)-(2.6). This condition is known to imply the inequality

$$f(x_k) - m_k(s_k) \ge \kappa_{\rm C}\|g_k\|\min\left[\frac{\|g_k\|}{1+\|B_k\|}, \sqrt{\frac{\|g_k\|}{\sigma_k}}\right] \tag{4.18}$$

for some constant $\kappa_{\rm C} \in (0,1)$ (see Lemma 1.1 in Cartis et al., 2011*a*). We may then build on this relation in the next two useful lemmas inspired from Cartis et al. (2011*a*).

Lemma 4.2 *[See Lemma 3.2 in Cartis et al., 2011*a*] Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that **A.1** and **A.4** hold, and that*

$$\sqrt{\sigma_k}\|s_k\| \ge \frac{108\sqrt{2}}{1-\eta_2}(L_g + \kappa_{\rm egt}\kappa_{\rm ts}^2(\kappa_{\rm ubg}+\kappa_{\rm egt}) + \kappa_{\rm B}) \stackrel{\rm def}{=} \kappa_{\rm HB}. \tag{4.19}$$

Then iteration k of the algorithm is successful with ($\rho_k \ge \eta_2$) and

$$\sigma_{k+1} \le \sigma_k. \tag{4.20}$$

Proof. From (4.19), we have that $g_k \neq 0$, since otherwise the algorithm would have stopped. Thus (4.18) implies that $f(x_k) > m_k(s_k)$. It then follows from (4.8) that

$$\rho_k > \eta_2 \Leftrightarrow \nu_k \stackrel{\rm def}{=} f(x_k+s_k) - f(x_k) - \eta_2[m_k(s_k) - f(x_k)] < 0.$$

We immediately note that, for $k \geq 0$,

$$\nu_k = f(x_k+s_k) - m_k(s_k) + (1-\eta_2)[m_k(s_k) - f(x_k)].$$

We then develop the first term in the right-hand side of this expression using a Taylor expansion of $f(x_k+s_k)$, giving that, for $k \geq 0$,

$$f(x_k+s_k) - m_k(s_k) = \langle \nabla_x f(\xi_k) - g_k, s_k \rangle - \tfrac{1}{2}\langle s_k, B_k s_k \rangle - \tfrac{1}{3}\sigma_k \|s_k\|^3 \tag{4.21}$$

for some ξ_k in the segment (x_k, x_k+s_k). But we observe that

$$\begin{array}{rcl}
\|\nabla_x f(\xi_k) - g_k\| & \leq & \|\nabla_x f(\xi_k) - \nabla_x f(x_k)\| + \|\nabla_x f(x_k) - g_k\| \\
& \leq & L_g\|s_k\| + \kappa_{\rm egt} t_k^2 \\
& \leq & L_g\|s_k\| + \kappa_{\rm egt}\kappa_{\rm ts}^2 \|s_k\|\,\|g_k\| \\
& \leq & [L_g + \kappa_{\rm egt}\kappa_{\rm ts}^2(\|\nabla_x f(x_k)\| + \|\nabla_x f(x_k) - g_k\|)]\|s_k\| \\
& \leq & [L_g + \kappa_{\rm egt}\kappa_{\rm ts}^2(\kappa_{\rm ubg} + \kappa_{\rm egt})]\|s_k\|,
\end{array}$$

where we successively used the triangle inequality, (2.11), (4.2), the negation of (4.7), **A.4** and the inequality $t_k \leq 1$. Thus the Cauchy-Schwartz inequality, (4.21) and the second inequality of (4.10) give that, for $k \geq 0$,

$$f(x_k+s_k) - m_k(s_k) \leq [L_g + \kappa_{\rm egt}\kappa_{\rm ts}^2(\kappa_{\rm ubg} + \kappa_{\rm egt}) + \kappa_{\rm B}]\,\|s_k\|^2. \tag{4.22}$$

The proof of the lemma then follows exactly as in Lemma 3.2 in Cartis et al. (2010*a*), using (4.18), with (4.22) playing the role of inequality (3.9) and $L_g + \kappa_{\rm egt}\kappa_{\rm ts}(\kappa_{\rm ubg} + \kappa_{\rm egt})$ playing the role of $\kappa_{\rm H}$. □

We may then recover boundedness of the regularization parameters.

Lemma 4.3 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that **A.1** and **A.4** hold. Then there exists a $\kappa_\sigma > 0$ such that (2.17) holds for all $k \geq 0$.*

Proof. The proof is identical to that of Lemma 3.3 in Cartis et al. (2011*a*), giving $\kappa_\sigma \stackrel{\rm def}{=} \gamma_2 \kappa_{\rm HB}^2$. □

Again, we replace (2.17) by (2.19) and, since κ_σ does not depend on $\kappa_{\rm B}$, possibly increase $\kappa_{\rm B}$ to ensure that $\kappa_{\rm B} \geq \kappa_\sigma \kappa_{\rm ubg}$ without loss of generality. Armed with these results, we may return to Lemma 4.1 above and obtain stronger conclusions.

Lemma 4.4 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that **A.1** and **A.4** hold. Then there exists a constant $\kappa_t > 0$ such that the return from Step 7 to Step 3 of the algorithm can only be executed at most*

$$\left\lceil \frac{\log \kappa_t + \frac{3}{2}\log \epsilon}{\log \gamma_3} \right\rceil_+ \tag{4.23}$$

times during the entire run of the algorithm.

Proof. Replacing (2.17) into (4.11) and using the fact that s_k is just the last $s_{k,j}$, we obtain that, for all $k \geq 0$

$$\|s_k\| \geq \frac{(1-\kappa_\theta)\,\epsilon}{\max\left[\,4\kappa_{\rm B}, \kappa_{\rm B} + 3\sqrt{\kappa_\sigma \kappa_{\rm ubg}/\epsilon}\,\right]} \geq \frac{(1-\kappa_\theta)\,\epsilon^{3/2}}{4\kappa_{\rm B}} \stackrel{\rm def}{=} \kappa_{s\epsilon}\epsilon^{3/2}.$$

Thus no return from Step 7 to Step 3 of the ARC-DFO algorithm is possible from the point where $j \geq 0$, the total number of times this return is executed, is large enough to ensure that

$$t_{k,j} = \gamma_3^j t_{0,0} \leq \gamma_3^j \leq \kappa_{\rm ts} \min\left[\kappa_{s\epsilon}\epsilon^{3/2}, \tfrac{1}{2}\epsilon\right] \leq \kappa_{\rm ts} \min\left[\,\|s_{k,j}\|, \|g_{k,j}\|\,\right],$$

where we have derived the last inequality using the fact that $\|g_{k,j}\| \geq \frac{1}{2}\epsilon$ as long as the algorithm has not terminated. This imposes that

$$j \leq \frac{1}{\log \gamma_3} \min \left[\log\left(\kappa_{\rm ts}\kappa_{{\rm s}\epsilon}\right) + \tfrac{3}{2}\log\epsilon,\ \log\left(\tfrac{1}{2}\kappa_{\rm ts}\right) + \log\epsilon \right],$$

and the desired bound on j follows with $\kappa_t = \kappa_{\rm ts}\min[\kappa_{{\rm s}\epsilon}, \frac{1}{2}]$. □

We may also revisit the second part of Lemma 2.2 in the derivative-free context. Our proof is directly inspired by Lemma 5.2 in Cartis et al. (2011*a*).

Lemma 4.5 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.4*** *hold. Then there exists a* $\sigma_{\max} > 0$ *independent of* ϵ *such that (2.18) holds for all* $k \geq 0$*.*

Proof. Using (2.1), the Cauchy-Schwarz and the triangle inequalities, (4.13), (2.12) and (4.15), we know that

$$\begin{array}{rcl} |f(x_k+s_k) - m_k(s_k)| & \leq & \|\nabla_x f(x_k) - g_k\|\,\|s_k\| \\ & & +\frac{1}{2}\left[\,\|\nabla_{xx} f(\xi_k) - \nabla_{xx} f(x_k)\| + \|\nabla_{xx} f(x_k) - B_k\|\,\right]\|s_k\|^2 \\ & & -\frac{1}{3}\sigma_k\|s_k\|^3 \\ & \leq & \left[\kappa_{\rm egt}\kappa_{\rm ts} + \frac{1}{2}(L_H + \kappa_{\rm eHt}\kappa_{\rm ts}) - \frac{1}{3}\sigma_k\right]\|s_k\|^3 \end{array}$$

for some $\xi_k \in [x_k, x_k + s_k]$. Thus, using (4.8) and (2.16),

$$|\rho_k - 1| = \left|\frac{f(x_k+s_k) - m_k(s_k)}{f(x_k) - m_k(s_k)}\right| \leq \frac{\kappa_{\rm egt}\kappa_{\rm ts} + \frac{1}{2}(L_H + \kappa_{\rm eHt}\kappa_{\rm ts}) - \frac{1}{3}\sigma_k}{\frac{1}{6}\sigma_k} \leq 1 - \eta_2$$

as soon as

$$\sigma_k \geq \frac{2\kappa_{\rm egt}\kappa_{\rm ts} + L_H + \kappa_{\rm eHt}\kappa_{\rm ts}}{1 - \frac{1}{3}\eta_2}.$$

As a consequence, iteration k is then successful, $\rho_k \geq \eta_2$ and $\sigma_{k+1} \leq \sigma_k$. It then follows that (2.18) holds with

$$\sigma_{\max} = \max\left[\sigma_0, \frac{\gamma_2(2\kappa_{\rm egt}\kappa_{\rm ts} + L_H + \kappa_{\rm eHt}\kappa_{\rm ts})}{1 - \frac{1}{3}\eta_2}\right].$$

□

It then remains to show that, under (4.13)–(4.15), an analog of Lemma 2.3 holds for the derivative-free case.

Lemma 4.6 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that* ***A.1*** *and* ***A.4*** *hold. Then there exists a constant* $\kappa_g > 0$ *such that, for all* $k \geq 0$*,*

$$\|s_k\| \geq \kappa_g\sqrt{\|g_k^+\|}. \tag{4.24}$$

Proof. We first observe, using the triangle inequality, (4.14) and (2.7), that

$$\begin{array}{rcl} \|g_k^+\| & \leq & \|g_k^+ - \nabla_x f(x_k+s_k)\| + \|\nabla_x f(x_k+s_k) - \nabla_x m_k(s_k)\| + \|\nabla_x m_k(s_k)\| \\ & \leq & \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 + \|\nabla_x f(x_k+s_k) - \nabla_x m_k(s_k)\| + \kappa_\theta \min[1, \|s_k\|]\,\|g_k\| \end{array} \tag{4.25}$$

for all $k \geq 0$. The second term on this last right-hand side may then be bounded for all $k \geq 0$ by

$$\begin{array}{rcl} \|\nabla_x f(x_k+s_k) - \nabla_x m_k(s_k)\| & \leq & \|\nabla_x f(x_k) - g_k\| + \|\displaystyle\int_0^1 [\nabla_{xx} f(x_k + \alpha s_k) - B_k]\, s_k \, d\alpha\| + \sigma_k \|s_k\|^2 \\ & \leq & \|\displaystyle\int_0^1 \{[\nabla_{xx} f(x_k+\alpha s_k) - \nabla_{xx} f(x_k)] + [\nabla_{xx} f(x_k) - B_k]\}\, s_k \, d\alpha\| \\ & & +\|\nabla_x f(x_k) - g_k\| + \sigma_k \|s_k\|^2 \\ & \leq & \max_{\alpha\in[0,1]} \|\nabla_{xx} f(x_k+\alpha s_k) - \nabla_{xx} f(x_k)\| \, \|s_k\| \\ & & +(\kappa_{\rm eHt} + \kappa_{\rm egt})\kappa_{\rm ts}\|s_k\|^2 + \sigma_{\max}\|s_k\|^2 \\ & \leq & [L_H + (\kappa_{\rm eHt} + \kappa_{\rm egt})\kappa_{\rm ts} + \sigma_{\max}]\|s_k\|^2, \end{array} \tag{4.26}$$

where we successively used the mean-value theorem, (2.1), the triangle inequality, (2.12), (4.13), (4.15) and (2.18). We also have, using the triangle inequality, (4.13), (2.11) and (4.14), that

$$\begin{array}{rcl} \|g_k\| & \leq & \|g_k - \nabla_x f(x_k)\| + \|\nabla_x f(x_k)\| \\ & \leq & \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 + \|\nabla_x f(x_k+s_k)\| + L_g\|s_k\| \\ & \leq & \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 + \|\nabla_x f(x_k+s_k) - g_k^+\| + \|g_k^+\| + L_g\|s_k\| \\ & \leq & 2\kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 + \|g_k^+\| + L_g\|s_k\|, \end{array}$$

which implies that, for all $k \geq 0$,

$$\kappa_\theta \min[1, \|s_k\|]\, \|g_k\| \leq (2\kappa_\theta\kappa_{\rm egt}\kappa_{\rm ts} + \kappa_\theta L_g)\|s_k\|^2 + \kappa_\theta\|g_k^+\|. \tag{4.27}$$

Therefore, substituting (4.26) and (4.27) into (4.25), we obtain that, for all $k \geq 0$,

$$\|g_k^+\| \leq \kappa_{\rm egt}\kappa_{\rm ts}\|s_k\|^2 + [L_H + (\kappa_{\rm eHt} + \kappa_{\rm egt})\kappa_{\rm ts} + \sigma_{\max}]\|s_k\|^2 + (2\kappa_\theta\kappa_{\rm egt}\kappa_{\rm ts} + \kappa_\theta L_g)\|s_k\|^2 + \kappa_\theta\|g_k^+\|.$$

Thus

$$(1-\kappa_\theta)\|g_k^+\| \leq [\kappa_\theta L_g + L_H + \kappa_{\rm ts}(\kappa_{\rm eHt} + 2\kappa_{\rm egt}(1+\kappa_\theta)) + \sigma_{\max}]\, \|s_k\|^2$$

for all $k \geq 0$. This gives (4.24) with

$$\kappa_g \stackrel{\rm def}{=} \sqrt{\frac{1-\kappa_\theta}{\kappa_\theta L_g + L_H + \kappa_{\rm ts}(\kappa_{\rm eHt} + 2\kappa_{\rm egt}(1+\kappa_\theta)) + \sigma_{\max}}}.$$

□

We are thus in principle again in position to apply the oracle complexity results for the ARC algorithm. Unfortunately, Theorem 2.5 may no longer be applied as such (as it requires the true gradient of the objective function), but our final theorem is derived in a very similar manner.

Theorem 4.7 *Suppose that we apply the ARC-DFO algorithm to problem (1.1), and also that* ***A.1***, ***A.2*** *and* ***A.4*** *hold, that $\epsilon \in (0,1)$ is given and that (2.21) holds. Then the algorithm terminates after at most*

$$N_1^s \stackrel{\rm def}{=} 1 + \left\lceil \kappa_S^s \epsilon^{-3/2} \right\rceil, \tag{4.28}$$

successful iterations and at most

$$N_1 \stackrel{\rm def}{=} \left\lceil \kappa_S \epsilon^{-3/2} \right\rceil \tag{4.29}$$

iterations in total, where κ_S^s and κ_S are given by (2.25) and (2.26), respectively. As a consequence, the algorithm terminates after at most

$$(N_1 - N_1^s)(1+2n) + N_1^s \left\lceil \frac{n^2+5n+2}{2} \right\rceil + \left\lceil \frac{n^2+3n}{2} \right\rceil \left\lceil \frac{\log\kappa_t + \frac{3}{2}\log\epsilon}{\log\gamma_3} \right\rceil_+. \tag{4.30}$$

objective function evaluations.

Proof. If the ARC-DFO algorithm does not terminate before or at iteration k, we know that

$$\min[\,\|g_j\|, \|g_{j+1}\|\,] \geq \tfrac{1}{2}\epsilon$$

for $j = 1, \dots, k$. As a consequence, we deduce from the definition of successful iterations, (2.16) and (4.24) that

$$f(x_k) - f(x_{k+1}) \geq \eta_1[f(x_k) - m_k(s_k)] \geq \frac{1}{48}\sigma_{\min}\eta_1\kappa_g^3\epsilon^{3/2} \;\text{ for all }\; k \in \mathcal{S}_k.$$

Since the mechanism of the ARC-DFO algorithm ensures that the iterates remain unchanged at unsuccessful iterations, summing up to iteration k, we therefore obtain that

$$f(x_0) - f(x_{k+1}) = \sum_{i\in\mathcal{S}_k}[f(x_i) - f(x_{i+1})] \geq \frac{1}{48}\sigma_{\min}\eta_1\kappa_g^3\epsilon^{3/2}|\mathcal{S}_k|.$$

Using now **A.2**, we conclude that

$$|\mathcal{S}_k| \leq \frac{48(f(x_0) - f_{\rm low})}{\sigma_{\min}\eta_1\kappa_g^3\epsilon^{3/2}},$$

from which (4.28) follows with

$$\kappa_{\rm S}^s = \frac{48(f(x_0) - f_{\rm low})}{\sigma_{\min}\eta_1\kappa_g^3}.$$

We then use Lemma 2.4 to deduce (4.29). If we ignore the estimations of $B_{k,j}$ in Step 3 after a return from Step 7, we now observe that each successful iteration involves up to

$$1 + 2n + \left(\frac{n(n+1)}{2}\right)$$

function evaluations, while unsuccessful iterations involve $1+2n$ evaluations. Adding the two, we obtain a number of

$$({\rm N}_1 - {\rm N}_1^s)(1+2n) + {\rm N}_1^s\left[1 + 2n + \frac{n(n+1)}{2}\right]$$

evaluations at most, to which we have to add those needed in the loop between Steps 3 and 7, whose number does not exceed

$$\left[n + \frac{n(n+1)}{2}\right]\left\lceil\frac{\log\kappa_t + \frac{3}{2}\log\epsilon}{\log\gamma_3}\right\rceil_+.$$

The resulting grand total is then given by (4.30). □

We may again considerably simplify this result (at the cost of a weaker bound). If we assume that the terms in n^2 and n dominate the constants, we obtain that, in the worst case, at most

$$O\left(\frac{n^2+5n}{2}\left[1 + \lceil|\log\epsilon|\rceil_+ + \left\lceil\frac{1}{\epsilon^{3/2}}\right\rceil_+\right]\right) \tag{4.31}$$

function evaluations are needed by the ARC-DFO algorithm to achieve approximate criticality in the sense of (4.6). Again, known sparsity of the Hessian or partial separability may reduce the factor n^2 in (4.31) to (typically) a small multiple of n or a small constant, thereby bridging the gap between ARC-DFO and ARC itself. The potential benefits of using parallel evaluations of the objective function are even more obvious here that for the ARC-FDH algorithm. Finally notice that automatic differentiation may often be an alternative to derivative-free technology when the source code for the evaluation of f is available, in which case the ARC-FDH algorithm is the natural choice.

We conclude this section by noting that, as was the case for Algorithm ARC-FDH, the bound (4.30) can be (marginally) improved by increasing the speed at which t_k decreases to zero in Step 7 of Algorithm ARC-DFO: the last term in (4.30) then decreases correspondingly, but remains dominated by the first two for all values of ϵ of interest.

5 Discussion and conclusions

Comparing algorithms on the basis of their worst-case complexity is always an exercise whose interest is mostly theoretical, but this is especially the case for what we have presented above. Indeed, several factors limit the predictive nature of these results on the practical behaviour of the considered minimization methods. The first is obviously the worst-case nature of the efficiency estimates, which (fortunately) can be quite pessimistic in view of expected or observed efficiency. The second, which is specific to the results presented here, is the intrinsic limitation induced by the use of finite-precision arithmetic. In the context of actual computation, not only it is unrealistic to consider vanishingly small values of ϵ, but the choice of arbitrarily small finite-differences stepsizes is also very questionable(4), even if difficulties caused by finite precision may be attenuated by using multiple-precision packages. The following comments should therefore be considered as interesting theoretical considerations throwing some light on the fundamental differences between algorithms, even if their practical relevance to actual numerical performance is potentially remote. Designing and studying worst-case analysis in the presence of round-off errors remains an interesting challenge.

We first note that the gap in worst-case performance between second-order (ARC), first-order (ARC-FDH) and derivative-free (ARC-DFO) methods is remarkably small if one consider the associated bounds in the asymptotic regime where ϵ tends to zero. The effect of finite-difference schemes is, up to constants, limited to the occurrence of an multiplicative factor of size $1 + |\log \epsilon|$, which may be considered as modest. The most significant effect is not depending on the ϵ-asymptotics, but rather depending on the dimension n of the problem: as expected, derivative-free methods suffer most in this respect, with bounds depending on n^2 rather than n for first-order methods or a constant for second-order ones. The result may seem unsurprising when considering the mechanism of finite-difference schemes only, but the interaction between the differencing stepsize and the user-specified accuracy makes them nontrivial, as can be seem from the technicality of the proofs presented.

The bounds for derivative-free methods are also interesting to compare with those derived by Vicente (2010), where direct-search type methods are shown to require at most $O(\epsilon^{-2})$ iterations to find a point x_k satisfying $\|\nabla_x f(x_k)\| \leq \epsilon$ when applied to function with Lipschitz continuous gradients(5). At iteration k, such methods compute the function values $\{f(x_k + \alpha_k d) \mid d \in \mathcal{D}_k\}$, where $\mathcal{D}_k$ is a positive spanning set for $\mathbb{R}^n$ and α_k an iteration-dependent stepsize. If one of these value is (sufficiently) lower than $f(x_k)$ the corresponding $x_k + \alpha_k d$ is chosen as the next iterate and a new iteration started. In the worst-case, an algorithm of this type therefore requires $n+1$(6) function evaluations, and thus its function-evaluation complexity is

$$O\left(n \left\lceil \frac{1}{\epsilon^2} \right\rceil\right)$$

Thus the ARC-DFO algorithm is more advantageous than such direct-search methods (in the worst-case and up to a constant factor) when the worst-case oracle complexity of the former is better than that of the latter, namely when

$$(n^2 + 5n) \left\lceil \frac{1 + |\log \epsilon|}{\epsilon^{3/2}} \right\rceil = O\left(n \left\lceil \frac{1}{\epsilon^2} \right\rceil\right),$$

which, taking into account just the leading coefficients, simplifies to

$$n = O\left(\frac{1}{[\,1 + |\log \epsilon|\,]\sqrt{\epsilon}}\right).$$

It is interesting to note that this relation only holds for relatively small n, especially for values of ϵ that are only moderately small, and for a more restrictive class of functions (**A.1** is required here, while Vicente (2010) only requires Lipschitz continuous gradients). Direct-search methods are thus very often

(4) Recommended values for these stepsizes are bounded below by adequate roots of machine precision (see Section 8.4.3 in Conn, Gould and Toint, 2000 or Sections 5.4 and 5.6 in Dennis and Schnabel, 1983, for instance).

(5) Note that the use of this inequality as a stopping criterion is not explicitly covered in Vicente (2010), but may nevertheless be constructed by using the stepsizes at unsuccessful iterations. The complexity result in this paper may therefore be interpreted as an indication of how many iterations will be performed by the algorithm before a stopping criterion in the spirit of (4.6) is activated. Vicente also proposes a surrogate stopping rule that avoids the need to know $\|\nabla_x f(x_k)\|$, but notes that this too may be impractical unless L_g is known.

(6) The minimal size of a positive spanning set in $\mathbb{R}^n$.

more efficient (in this theoretical sense) than the ARC-DFO algorithm, even if the latter dominates for small values of ϵ. These results could of course be used to select an optimal methods for given n and ϵ, to define a method with best *theoretical* complexity bounds.

Finally notice that the central properties needed for proving the complexity result for the ARC-DFO algorithm are the bounds (4.13)–(4.15). These could as well be guaranteed by more sophisticated derivative-free techniques where multivariate interpolation is used to construct Hessian approximation from past points in a suitable neighbourhood of the current iterate (see Conn, Scheinberg and Vicente, 2009, Fasano, Nocedal and Morales, 2009, or Scheinberg and Toint, 2010, for instance). This suggests that a worst-case analysis of these methods might be quite close to that of Algorithm ARC-DFO. Indeed, if gains in the number of function evaluations might be possible by the re-use of these past points compared to using fresh evaluations for establishing a local quadratic model at every iteration, it is not clear that these gains can always be obtained in practice, in particular if every step is large compared the necessary finite-difference stepsize.

Acknowledgments

The work of the second author is funded by EPSRC Grant EP/E053351/1. All three authors are grateful to the Royal Society for its support through the International Joint Project 14265, and for the helpful comments of three anonymous referees.

References

A. Agarwal, P. L. Bartlett, P. Ravikummar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. *in* 'Proceedings of the 23rd Annual Conference on Neural Information Processing Systems', 2009.

C. Cartis, N. I. M. Gould, and Ph. L. Toint. Trust-region and other regularisation of linear least-squares problems. *BIT*, **49**(1), 21–53, 2009.

C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic overestimation methods for unconstrained optimization. Part II: worst-case function-evaluation complexity. *Mathematical Programming, Series A*, 2010*a*. DOI: 10.1007/s10107-009-0337-y.

C. Cartis, N. I. M. Gould, and Ph. L. Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization. *SIAM Journal on Optimization*, **20**(6), 2833–2852, 2010*b*.

C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic overestimation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming, Series A*, **127**(2), 245–295, 2011*a*.

C. Cartis, N. I. M. Gould, and Ph. L. Toint. Complexity bounds for second-order optimality in unconstrained optimization. *Journal of Complexity*, **(to appear)**, 2011*b*.

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

A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-free Optimization*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2009.

J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1983. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.

G. Fasano, J. Nocedal, and J.-L. Morales. On the geometry phase in model-based algorithms for derivative-free optimization. *Optimization Methods and Software*, **24**(1), 145–154, 2009.

D. Goldfarb and Ph. L. Toint. Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations. *Mathematics of Computation*, **43**(167), 69–88, 1984.

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

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, Cambridge, United Kingdom, 1981.

A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. *in* M. J. D. Powell, ed., 'Nonlinear Optimization 1981', pp. 301–312, London, 1982. Academic Press.

A. S. Nemirovski. Efficient methods in convex programming. Lectures notes (online) available on http://www2.isye.gatech.edu/˜nemirovs/OPTI_LectureNotes.pdf, 1994.

A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* J. Wiley and Sons, Chichester, England, 1983.

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

Yu. Nesterov. Accelerating the cubic regularization of Newton's method on convex problems. *Mathematical Programming, Series A*, **112**(1), 159–181, 2008.

Yu. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming, Series A*, **108**(1), 177–205, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Series in Operations Research. Springer Verlag, Heidelberg, Berlin, New York, 1999.

M. J. D. Powell and Ph. L. Toint. On the estimation of sparse Hessian matrices. *SIAM Journal on Numerical Analysis*, **16**(6), 1060–1074, 1979.

K. Scheinberg and Ph. L. Toint. Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization. *SIAM Journal on Optimization*, **20**(6), 3512–3532, 2010.

S. A. Vavasis. Approximation algorithms for indefinite quadratic programming. *Mathematical Programming*, **57**(2), 279–311, 1992*a*.

S. A. Vavasis. *Nonlinear Optimization: Complexity Issues.* International Series of Monographs on Computer Science. Oxford University Press, Oxford, England, 1992*b*.

S. A. Vavasis. Black-box complexity of local minimization. *SIAM Journal on Optimization*, **3**(1), 60–80, 1993.

L. N. Vicente. Worst case complexity of direct search. Technical report, Department of Mathematics, University of Coimbra, Coimbra, Portugal, May 2010. Preprint 10-17, revised 2011.

M. Weiser, P. Deuflhard, and B. Erdmann. Affine conjugate adaptive Newton methods for nonlinear elastomechanics. *Optimization Methods and Software*, **22**(3), 413–431, 2007.