UNIVERSITÀ
DEGLI STUDI
FIRENZE

DIPARTIMENTO
DI INGEGNERIA
INDUSTRIALE

On the use of iterative methods in cubic regularization for unconstrained optimization

T. Bianconcini, G. Liuzzi, B. Morini, M. Sciandrone

Dipartimento di Ingegneria Industriale
Università degli Studi di Firenze
Rapporto Tecnico n. 6/2013

ON THE USE OF ITERATIVE METHODS IN CUBIC REGULARIZATION FOR UNCONSTRAINED OPTIMIZATION

TOMMASO BIANCONCINI[†], G. LIUZZI[‡], BENEDETTA MORINI[§], AND MARCO SCIANDRONE[¶]

Abstract. In this paper we consider the problem of minimizing a smooth function by using the Adaptive Cubic Regularized framework (ARC). We focus on the computation of the trial step as a suitable approximate minimizer of the cubic model and discuss the use of matrix-free iterative methods. Our approach is alternative to the implementation proposed in the original version of ARC, involving a linear algebra phase, but preserves the same worst-case complexity count as ARC. Further we introduce a new stopping criterion in order to properly manage the "over-solving" issue arising whenever the cubic model is not an adequate model of the true objective function. Numerical experiments conducted by using a nonmonotone gradient method as inexact solver are presented. The obtained results clearly show the effectiveness of the new variant of ARC algorithm.

Keywords Unconstrained optimization, cubic regularization, worst-case complexity, matrix-free subproblem solvers.

1. Introduction. Adaptive regularized methods have been recently studied as an alternative to classical globalization techniques for nonlinear constrained and unconstrained optimization [1, 2, 3, 5, 7, 9, 14, 15, 17, 18, 20]. This paper is devoted to the numerical solution of the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function, by means of the Adaptive Regularized framework using Cubics (ARC) [5, 7, 15, 18, 20].

The adaptive cubic regularization of Newton method for (1.1) gives rise to a local cubic overestimator of the objective function f which is employed for computing the step from one iterate to the next. Under mild assumptions, ARC iterates converge to first-order critical points; by strengthening the conditions on the acceptance of the trial step, second-order variants of ARC show global and fast convergence to second-order critical points [5, 15, 18].

Besides the good numerical performance of ARC compared to a standard trust-region approach [5], its distinguishing features from linesearch and trust-region techniques are the results on worst-case iteration and gradient evaluation complexity [7, 18]. Specifically, a worst-case iteration count of order $\epsilon^{-3/2}$ has been established to drive the norm of the gradient of f below the prefixed accuracy ϵ. This bound is sharp and represents a substantial improvement over the Newton's method which may be as slow as the steepest descent method (in the worst case) and require a number of iterations of order ϵ^{-2} [8].

The good complexity bound of ARC can be achieved by minimizing the cubic model approximately within some suitable accuracy requirement and under conditions that can all be ensured if the step taken at each iteration is a global minimizer of the

[†]Dipartimento di Ingegneria dell'Informazione, Via di S. Marta 3, 50139 Firenze, Italia KKT srl, Viale Mazzini 40, 50132 Firenze, Italia, tommaso.bianconcini@unifi.it.

[‡]Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" (IASI) Consiglio Nazionale delle Ricerche (CNR) Viale Manzoni 30, 00185 Roma, Italia, giampaolo.liuzzi@iasi.cnr.it

[§]Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze, viale Morgagni 40, 50134 Firenze, Italia, benedetta.morini@unifi.it.

[¶]Dipartimento di Ingegneria dell'Informazione Via di S. Marta 3, 50139 Firenze, Italia, marco.sciandrone@unifi.it

model in a subspace of $\mathbb{R}^n$ [7]. The second-order variant of ARC, denoted as $\text{ARC}_{(\text{S})}$ in [7], algorithmically ensures the conditions required and one possible implementation relies on linear algebra. In fact, in $\text{ARC}_{(\text{S})}$ the approximate minimization of the cubic model can be performed over evolving subspaces by using Lanczos method until a suitable termination criterion is met.

Despite the good complexity properties, the practical efficiency of ARC depends on the way the cubic model is minimized. Therefore in this work we investigate how to preserve the complexity properties of $\text{ARC}_{(\text{S})}$ framework when procedures alternative to Lanczos method are used for minimizing the cubic model. Our interest is on iterative descent methods, such as gradient methods, limited memory Quasi-Newton methods, conjugates gradients methods, which are matrix-free and either reduce or avoid the cost of the linear algebra phase.

Given an arbitrarily computed approximate minimizer for the model, we introduce a strategy that fits into $\text{ARC}_{(\text{S})}$ framework, thus retaining its worst-case complexity count. To achieve this goal the approximate minimization of the model is combined, if necessary, with the exact optimization of the model in a suitable one-dimensional space. Further, we propose a new stopping criterion (called early stopping) in the minimization process of the model, in order to combine the cubic model and the objective function f. The resulting variant of ARC has been extensively tested using a nonmonotone gradient method as iterative solver for the step and showed to be superior to the implementation of ARC available in the GALAHAD library [12].

The paper is organized as follows. Section 2 reviews the ARC framework, while Section 3 present our variant of ARC preserving the good iteration complexity of $\text{ARC}_{(\text{S})}$. Section 4 discusses the minimization of the cubic model by iterative descent methods and their use in conjunction with the early stopping criterion. Finally, in Section 5 we report the results of extensive experiments comparing our procedure with the version of ARC using the Lanczos solver and implemented in GALAHAD [11, 12].

Notations. The gradient $\nabla_x f(x)$ of f and the Hessian $\nabla_{xx} f(x)$ of f are denoted by $g(x)$ and $H(x)$ respectively. The 2-norm is denoted by $\|x\|$. The identity matrix is indicated by I.

2. Adaptive cubic regularization algorithms. Adaptive cubic regularization of Newton method for unconstrained optimization gives rise to globally convergent procedures recently investigated in many papers, see e.g. [5, 7, 15, 18]. The key feature of this approach is the computation of the step from one iterate to the next by minimizing a cubic overestimator of the objective function f.

If the Hessian H of f is Lipschitz continuous (with constant $2L$), the Taylor expansion of f around x_k gives

$$\begin{aligned} f(x_k+p) &= f(x_k) + p^T g(x_k) + \frac{1}{2} p^T H(x_k) p + \int_0^1 (1-\tau) p^T (H(x_k+\tau p) - H(x_k)) p \, d\tau \\ &\leq f(x_k) + p^T g(x_k) + \frac{1}{2} p^T H(x_k) p + \frac{1}{3} L ||p||^3 \stackrel{\text{def}}{=} m_k^C(p), \end{aligned}$$

for all $p \in \mathbb{R}^n$. Thus, for every step p such that $m_k^C(p) \leq m_k^C(0) = f(x_k)$, the point $x_k + p$ improves f.

In order to define a model of practical interest, the constant L may be replaced by a dynamic positive parameter σ_k and $H(x_k)$ may be approximated by a symmetric

matrix B_k. This gives rise to the model

$$m_k(p) = f(x_k) + p^T g(x_k) + \frac{1}{2} p^T B_k p + \frac{1}{3}\sigma_k \|p\|^3, \tag{2.1}$$

which is employed in the Adaptive Regularization algorithm using Cubics (ARC) proposed by Cartis et al. in [5, 7].

The k-th iteration of ARC method is sketched in Algorithm 2.1. In Step 1 the trial step p_k from x_k to x_{k+1} is computed as an approximate minimizer of the model m_k and provides a decrease in m_k greater than or equal to the reduction attained by the Cauchy point (2.3). Then, in Step 3 p_k is accepted and the new iterate x_{k+1} is set to $x_k + p_k$ if a sufficient decrease in the objective is achieved; otherwise, the step is rejected and x_{k+1} is set to x_k. Since the denominator in (2.4) is strictly positive whenever the current iterate is not a first-order critical point, then ARC algorithm is well defined and the sequence $\{f(x_k)\}$ generated is monotonically nonincreasing. The rules in Step 4 for updating the parameter σ_k take into account the agreement between f and m_k and parallel those for updating the trust-region radius in trust-region methods [19].

Algorithm 2.1: k-th iteration of ARC

Given x_k and the scalars $\sigma_k > 0$, $1 > \eta_2 \geq \eta_1 > 0$, $\gamma_2 \geq \gamma_1 > 1$.

1. Compute an approximate minimizer p_k of m_k such that

$$m_k(p_k) \leq m_k(p_k^c), \tag{2.2}$$

where p_k^c is the Cauchy point

$$p_k^c = -\alpha_k g(x_k), \qquad \alpha_k = \operatorname*{argmin}_{\alpha \geq 0} m_k(-\alpha g(x_k)). \tag{2.3}$$

2. Compute

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - m_k(p_k)}. \tag{2.4}$$

3. Set

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

4. Set

$$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if } \rho_k \geq \eta_2 & \text{(very successful iteration),} \\ [\sigma_k, \gamma_1 \sigma_k) & \text{if } \eta_1 \leq \rho_k \leq \eta_2 & \text{(successful iteration),} \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise} & \text{(unsuccessful iteration).} \end{cases}$$

Condition (2.2) on p_k imposes at least as much decrease in the model as that obtained by the Cauchy point p_k^c. A lower bound on the decrease achieved by p_k^c with respect to $m_k(0) = f(x_k)$ is given below.

LEMMA 2.1. [5, Lemma 2.1] *Suppose that the step p_k satisfies (2.2). Then for $k \geq 0$, we have that*

$$f(x_k) - m_k(p_k) \geq f(x_k) - m_k(p_k^c) \geq \frac{\|g(x_k)\|}{6\sqrt{2}} \min\left(\frac{\|g(x_k)\|}{1+\|B_k\|}, \frac{1}{2}\sqrt{\frac{\|g(x_k)\|}{\sigma_k}}\right) \tag{2.5}$$

By imposing (2.2), global convergence to stationary points of problem (1.1) can be enforced as stated by the proposition below.

PROPOSITION 2.2. [5, Corollary 2.6] *Let $f \in C^1(\mathbb{R}^n)$ and $\{x_k\}$ be the sequence generated by the ARC Algorithm. Suppose that $\|B_k\|$ is uniformly bounded for all $k \geq 0$ and that the gradient g is uniformly continuous on the sequence $\{x_k\}$. Then,*

$$\lim_{k\to\infty} \|g(x_k)\| = 0. \tag{2.6}$$

In order to enforce local fast convergence, more model reduction than (2.2) is sought and this requires to approximately minimize m_k. We now discuss the stationary points and values of m_k analyzed in [5, 15]. Any stationary point $\hat{p}$ of m_k satisfies

$$(B_k + \hat{\lambda} I)\hat{p} = -g(x_k), \tag{2.7}$$

$$\hat{\lambda} = \sigma_k \|\hat{p}\|. \tag{2.8}$$

A stationary point p_k^* is a global minimizer of m_k over $\mathbb{R}^n$ if and only if there exists a positive λ_k^* such that the pair (p_k^*, λ_k^*) satisfies (2.7) and (2.8) and $B_k + \lambda_k^* I$ is positive semidefinite. Clearly, if $B_k + \lambda_k^* I$ is positive definite then p_k^* is unique.

The stationary values $m_k(\hat{p})$ are strictly decreasing in $\|\hat{p}\|$ and they are up to $2\ell + 1$ values if B_k has ℓ negative eigenvalues. If B_k is indefinite then the stationary values include one global minimum and possibly a second local minimum.

When the model is convex, equations (2.7) and (2.8) characterize any global minimizer. This occurrence is verified for k sufficiently large when the iterates converge to a point with positive definite Hessian and the approximate Hessian B_k becomes positive definite asymptotically. Another occurrence is when B_k is a L-BFGS Hessian approximation. Finally, a convex model can be obtained from the application of ARC method to nonlinear least-squares problems [2, 14]. In this case $f(x) = \|F(x)\|^2$ for some vector-valued function F and the model used is a variant of (2.1) of practical interest for computing zero or small-residual solutions of the problem. Specifically, it consists of the Gauss-Newton model regularized by a cubic term

$$m_k(p) = \frac{1}{2}\|F(x_k) + J(x_k)p\|^2 + \frac{1}{3}\sigma_k \|p\|^3, \tag{2.9}$$

where J is the Jacobian matrix of F. For any positive σ_k the model (2.9) is strictly convex.

The use of an approximate global minimizer is considered in [5] and a viable strategy for its approximation is derived using (2.7) and (2.8). We briefly review the main issues for approximating the optimal pair (p_k^*, λ_k^*). Let $p(\lambda)$ solve

$$(B_k + \lambda I)p(\lambda) = -g(x_k), \tag{2.10}$$

λ_k be an approximation to the optimal value λ_k^*, and $p_k = p(\lambda_k)$. The scalar λ_k can be obtained applying a root-finding solver to the so-called secular equation, i.e. the scalar nonlinear equation

$$\lambda - \sigma_k \|p(\lambda)\| = 0,$$

which can be reformulated as

$$\psi(\lambda) = \frac{1}{\|p(\lambda)\|} - \frac{\sigma_k}{\lambda} = 0. \tag{2.11}$$

Letting $\lambda_{min}(B_k)$ be the smallest eigenvalue of B_k and $\zeta = \max\{0, -\lambda_{min}(B_k)\}$, the function $\psi(\lambda)$ is concave and strictly increasing when $\lambda > \zeta$. Hence, either the Newton or the secant method applied to (2.11) converges globally and monotonically to the positive root λ_k^* for any initial guess in the open interval (ζ, λ_k^*) [5, Theorem 6.3]. Clearly, the application of these methods requires the solution of system (2.10) for various λ and the use of a Krylov method represents a relevant alternative.

Using the Lanczos method m_k can be minimized over evolving subspaces of $\mathbb{R}^n$. Specifically, the Lanczos method can be used to build an orthogonal basis $\{q_1, \ldots, q_j\}$ for the Krylov space $\mathcal{K}_j = \{g(x_k), B_k g(x_k), \ldots, B_k^{j-1} g(x_k)\}$. Then, letting $Q_j \in \mathbb{R}^{n \times j}$ be the matrix $Q_j = (q_1, \ldots, q_j)$, the minimizer $p_{k,j}$ of m_k over $\mathcal{K}_j$ is the vector

$$p_{k,j} = Q_j y_j \quad \text{such that} \quad y_j = \operatorname*{argmin}_{y \in \mathbb{R}^j} m_k(Q_j y). \tag{2.12}$$

Solving the problem on each expanding subspace $\mathcal{K}_j$ is computationally convenient and the minimization process is carried out on evolving subspaces until a specified accuracy requirement is met. In particular, the procedure is repeated until a vector y_{j^*} satisfying

$$\|\nabla m_k(p_{k,j^*})\| = \|\nabla m_k(Q_{j^*} y_{j^*})\| \leq \eta_k \|\nabla m_k(0)\|, \tag{2.13}$$

is computed for a given $\eta_k \in [0, 1)$. Once y_{j^*} is computed, the approximate minimizer p_{k,j^*} can be evaluated either by recomputing the vectors q_j, $1 \leq j \leq j^*$, or by recovering them from memory. It has been found advantageous to store a small number t of the first q_j, $1 \leq j \leq t$, and to start from $j = t$, if necessary, the so-called *second-pass iteration* to determine p_{k,j^*}. Further economies can be made recording all the generated values $m_k(p_{k,j})$, $1 \leq j \leq j^*$, picking an iteration $h \leq j^*$ which gives a specified fraction of the best value obtained, and then accepting $Q_h y_h$ as the required approximation [5, 11].

Under suitable assumptions, the sequence $\{x_k\}$ generated by ARC shows superlinear or quadratic convergence rate if the steps used satisfy (2.13) and $\eta_k \to 0$ as $k \to \infty$ [5, Corollary 4.8, 4.10]. For instance, superlinear convergence rate can be ensured provided that

$$\|\nabla m_k(p_k)\| \leq \min\{\theta, \|g(x_k)\|^{1/2}\} \|g(x_k)\|, \tag{2.14}$$

and quadratic convergence rate can be achieved if

$$\|\nabla m_k(p_k)\| \leq \min\{\theta, \|p_k\|\} \|g(x_k)\|, \tag{2.15}$$

with $\theta > 0$.

3. Worst-case iteration complexity bounds for ARC. In this section, following [7], we preliminary recall the necessary ingredients to obtain the complexity bound in ARC. Then, we present a new procedure that uses a simple gradient method with backtracking and generates a global minimizer of of m_k over a one-dimensional space. Thus, the step obtained can be employed in connection with ARC framework attaining its complexity bound. The procedure is matrix-free and does not require to

store vectors or to recover them from memory, as in a Krylov method, whose number can be, in principle, equal to the dimension of the problem.

Global convergence properties and worst-case complexity of the ARC algorithm have been established in [7]. In particular, for the ARC algorithm it is possible to bound the number of successful iterations (i.e. number of gradient evaluations) required to drive the norm of the gradient g below a prefixed accuracy provided that, at successful iterates, the step p_k yields a sufficient predicted reduction, as shown in the proposition below.

PROPOSITION 3.1. *[7, Theorem 2.2] Let $\{f(x_k)\}$ be bounded from below. Assume that the successful iterates x_k generated by* ARC *algorithm have the property*

$$f(x_k) - m_k(p_k) \geq \rho\epsilon^{3/2}, \tag{3.1}$$

where ρ is a positive constant independent of k and $\epsilon > 0$. Then, ARC *algorithm requires at most $\mathcal{O}(\epsilon^{-3/2})$ iterations to attain*

$$\|g(x_k)\| \leq \epsilon.$$

The worst case complexity bound of order $\epsilon^{-3/2}$ was shown to be sharp and represents a substantial improvement over the Newton's method which may be as slow as the steepest descent method (in the worst case) and requires a number of iterations of order ϵ^{-2} [8].

The key point in Proposition 3.1 is condition (3.1) which can be accomplished, for instance, by requiring that, for all $k \geq 0$, p_k satisfies (2.15) and

$$g(x_k)^T p_k + p_k^T B_k p_k + \sigma_k \|p_k\|^3 = 0, \tag{3.2}$$

$$p_k^T B_k p_k + \sigma_k \|p_k\|^3 \geq 0. \tag{3.3}$$

The variant $\text{ARC}_{(\text{S})}$ introduced in [7, Algorithm 4.1] for all $k \geq 0$ uses a step such that (2.15), (3.2) and (3.3) are met and (2.2) remains satisfied. A situation in which such conditions hold is when the step p_k is computed by approximately minimizing m_k over nested subspaces as in (2.12) and termination criterion (2.15) is adopted. This feature is a consequence of the following result.

LEMMA 3.2. *[7, Lemma 4.1] Suppose that p_k is the global minimizer of $m_k(p)$, for $p \in \mathcal{L}_k$, where $\mathcal{L}_k$ is a subspace of $\mathbb{R}^n$. Then p_k satisfies* (3.2) *and* (3.3).

On the other hand, if the step p_k is computed by a procedure other than the minimization of m_k over evolving subspaces, conditions (3.2) and (3.3) may not hold thus making the analysis in [7] unuseful and, possibly, loosing the complexity property of the ARC algorithm. For this reason, in the following Algorithm 3.1 we introduce a strategy that produces an approximate minimizer for m_k satisfying conditions (2.2), (2.15), (3.2) and (3.3). Thus, condition (3.1) holds and the properties in terms of worst-case complexity bound are ensured.

Note that in Step c.1 $p_{k,j+1}$ is the global minimizer of m_k over the subspace generated by $d_{k,j}$, so that it satisfies (3.2) and (3.3) and (2.3) remains satisfied. Since it is accepted and the algorithm terminates when the accuracy requirement (2.15) is met, by construction $p_{k,j+1}$ satisfies also condition (3.1). In Step c, for $j \geq 1$ the vectors

Algorithm 3.1: A variant of Step 1 of ARC.

At each iteration of ARC algorithm, perform Step 1 as follows

a. Compute an approximate minimizer $p_{k,0}$ of $m_k(p)$, such that

$$m_k(p_{k,0}) \leq m_k(p_k^c). \tag{3.4}$$

b. Set $d_{k,0} = p_{k,0}$.
c. For $j = 0, 1, \ldots$
c.1 Compute

$$p_{k,j+1} = \beta d_{k,j}, \qquad \beta = \operatorname*{argmin}_{\beta \in R} m_k(\beta d_{k,j}).$$

c.2 If $p_{k,j+1}$ satisfies (2.15)
set $p_k = p_{k,j+1}$ and stop.
Else
apply a backtracking linesearch and find

$$z_{k,j+1} = p_{k,j+1} - \zeta_{k,j+1} \nabla m_k(p_{k,j+1}), \tag{3.5}$$

such that

$$m_k(z_{k,j+1}) \leq m_k(p_{k,j+1}) - \mu \zeta_{k,j+1} \|\nabla m_k(p_{k,j+1})\|^2.$$

c.3 Set $d_{k,j+1} = z_{k,j+1}$, $j = j + 1$.

$d_{k,j}$ are generated by applying an Armijo-type line search. Taking into account that m_k is coercive and the properties of the Armijo-type line search, it follows that Step c terminates in a finite number of iterations. Formally, we can state the next result.

PROPOSITION 3.3. *Assume $f \in C^2(\mathbb{R}^n)$, $g(x_k) \neq 0$. Then, Algorithm 3.1 terminates in a finite number of iterations producing a step p_k satisfying conditions (2.15), (3.2) and (3.3).*

Proof. Let J be the set of iterations j executed at Step c of Algorithm 3.1. We first show that J is finite, that is the test at Step c.2 is satisfied in a finite number of iterations. By contradiction, let us suppose that the test at Step c.2 is never satisfied so that the set J of iterations j of Algorithm 3.1 is infinite. Since m_k is coercive, the properties of the backtracking procedure ensure that

$$\lim_{j \to \infty} \|\nabla m_k(p_{k,j+1})\| = 0. \tag{3.6}$$

Considering the gradient of the model function $m_k(p)$,

$$\nabla m_k(p) = g(x_k) + B_k p + \sigma_k \|p\| p,$$

and $g(x_k) \neq 0$, from (3.6) we get that $\|p_{k,j+1}\| \geq \eta > 0$ for all j. Hence, we can write $\min\{\theta, \|p_{k,j+1}\|\}\|g(x_k)\| \geq \min\{\theta, \eta\}\|g(x_k)\|$ which is a constant. Thus, using again (3.6), it follows that, for j sufficiently large

$$\|\nabla m_k(p_{k,j+1})\| \leq \min\{\theta, \eta\}\|g(x_k)\| \leq \min\{\theta, \|p_{k,j+1}\|\}\|g(x_k)\|,$$

that is, $p_{k,j+1}$ satisfies condition (2.15). Then, Algorithm 3.1 would stop at Step c.2 thus contradicting the assumption that J is infinite.

Now, let $\bar{j}$ be the iteration such that $p_{k,\bar{j}+1}$ satisfies (2.15). Then, by the instruction of the algorithm, we also have that p_k satisfies (2.15).

Furthermore, since $p_{k,j+1}$, for all $j \geq 0$ is the global minimizer of m_k over the subspace generated by $d_{k,j}$, Lemma 3.2 implies that $p_{k,j+1}$, and in particular p_k, satisfies (3.2) and (3.3), which concludes the proof. □

Now we can prove the main result of this section, namely that, at every successful iteration k, condition (3.1) holds.

PROPOSITION 3.4. *Let*

(i) $f \in C^2(\mathbb{R}^n)$;

(ii) $\|g(x) - g(y)\| \leq \mathcal{K}\|x - y\|$, *for all* $x, y \in X$ *an open convex set containing the iterates and* $\mathcal{K} \geq 1$;

(iii) $\|H(x) - H(x_k)\| \leq L\|x - x_k\|$, *for all* $x \in [x_k, x_k + p_k]$ *and all* $k \geq 0$;

(iv) $\|(H(x_k) - B_k)p_k\| \leq C\|p_k\|^2$, *for all* $k \geq 0$ *and some positive constant* C;

(v) $\sigma_k \geq \sigma_{min}$ *for all* $k \geq 0$ *and some positive* σ_{min}.

Then, Algorithm 3.1 is such that condition (3.1) *holds at every successful iteration* k *of ARC.*

Proof. Since Algorithm 3.1 ensures that (3.2) and (3.3) hold, from [7, Lemma 4.2] it follows that

$$f(x_k) - m_k(p_k) \geq \frac{1}{6}\sigma_k\|p_k\|^3. \tag{3.7}$$

Then, by [7, Lemma 5.2] and considering that (2.15) is satisfied, the step p_k is such that, for all successful iterations k and some positive ν,

$$\|p_k\| \geq \nu\sqrt{\|g(x_{k+1})\|}.$$

The above lower bound along with (3.7) and $\min\{\|g(x_k)\|, \|g(x_{k+1})\|\} > \epsilon$ imply

$$f(x_k) - m_k(p_k) \geq \frac{1}{6}\nu\sigma_{min}\epsilon^{3/2},$$

which concludes the proof. □

4. An ARC algorithm using iterative methods and early stopping. As discussed in the preceding sections, a key issue of ARC algorithm concerns the computation of the trial step p_k as a suitable (approximate) unconstrained minimizer of the cubic model at each iteration. Therefore we focus on the approximate minimization of m_k and on the employment of a stopping criterion, the "early stopping", which advantageously combine the model and the "true" objective function f.

From a theoretical point of view we may observe that:

- in order to guarantee global convergence, it is sufficient that the trial step is such that

$$m_k(p_k) \leq m_k(p_k^c),$$

  where p_k^c is the Cauchy point;
- a complexity bound is guaranteed provided that condition (3.1) holds.

Hence, as regards the convergence properties, any globally convergent iterative method, employing at the first iteration an exact line search along the steepest descent direction, can be employed to solve the problem

$$\min_p \; m_k(p) = f(x_k) + g(x_k)^T p + \frac{1}{2} p^T B_k p + \frac{1}{3}\sigma_k \|p\|^3, \tag{4.1}$$

and compute the step p_k. Concerning the stopping criteria, condition (2.14) can be used to solve (4.1) while condition (2.15) has to be imposed within Algorithm 3.1 to guarantee the worst-case complexity property. In the following we present a further stopping criterion that takes into account the objective function f during the minimization process of the model.

The early stopping criterion

As said before, in order to suitably combine the cubic model and the objective function f, we adopt a new stopping criterion, that we call *early stopping*. This kind of stopping criterion is a technique widely adopted in machine learning (see, e.g., [4]). When a machine learning model is trained, a descent algorithm is applied to minimize the error on the training data. In order to prevent the overfitting phenomenon (which happens when the model over-learns the training data and the generalization capability deteriorates), the performances of the model in terms of generalization capability (which is the true objective of the learning) are periodically evaluated during the optimization process using a different data set, the so-called validation set. The optimization is stopped when the error on the validation set starts to increase.

By drawing inspiration from the above early stopping strategy, during the minimization process of the cubic model $m_k(p)$, the true objective function $f(x)$ is periodically evaluated (say every N iterations of the adopted iterative method) and the minimization method is stopped when f starts to increase.

As an example, with reference to the CUTEr test function CHAINWOO, Figure 4.1 shows the behavior of both $m_k(p(j))$ and $f(x_k + p(j))$ as functions of the inner iteration counter j, where x_k is the current iterate and $\{p(j)\}$ denotes the sequence generated by a descent method applied to the minimization of the model function $m_k(p)$. As it can be seen, after a few inner iterations, there is no agreement between the cubic model and the true objective function, so that the algorithm could take advantage of the early stopping condition just described.

Formally, we denote by $\{p(j)\}$ the sequence generated by an iterative method applied to minimize $m_k(p)$, where $p(0)$ is set equal to the Cauchy point p_k^c.

We propose to terminate the method whenever either criterion (2.14) is satisfied, or

$$f(x_k + p(j)) \geq f(x_k + p(j-N)), \qquad \text{with} \qquad \text{mod}(j, N) = 0, \tag{4.2}$$

for some $N \in \mathbb{N}, N \geq 1$ and to consider respectively $p(j)$ or $p(j-N)$ as trial step. Specifically, in order to guarantee the bound on the global worst-case iteration complexity, a further test (related to condition (3.1)) on the reduction of the model must be considered. When this test is satisfied, the tentative step can be accepted as p_k, otherwise Algorithm 3.1 must be applied to compute p_k.

The minimization scheme of the cubic model

We observe that, in principle, the early stopping criterion could be used in connection with the Krylov-type algorithms described in Section 2. In this case the adoption of

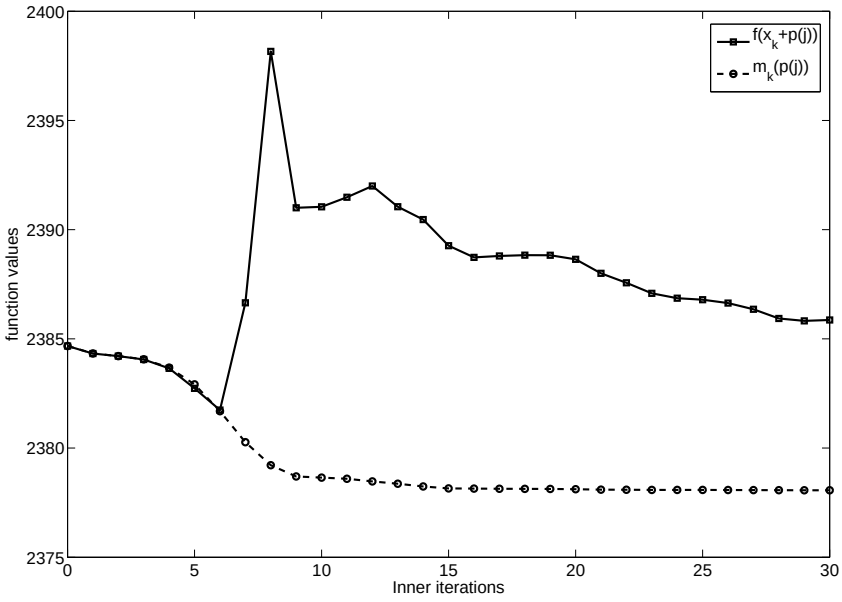

FIG. 4.1. *Example of early stopping*

the early stopping, which requires the periodic evaluation of the true objective function, involves some critical issues that may drastically influence the computational cost. In fact the minimization in nested Krylov subspaces requires to store the elements of the basis of the subspace, which will be used to reconstruct the trial step p_k. Since in general this operation may require a large amount of memory, it is a common practice to keep in memory only the first few elements of the basis. In this case, the so-called *second-pass iteration* (see Section 2) is required to rebuilt the approximate minimizer $p_{k,j}$, which is used for computing the objective function value in the updated point, i.e., $f(x_k + p_{k,j})$. We may expect that the second-pass iteration, to be performed several times during each major iteration, determines a drastic increase of the computational cost.

Therefore, we focus on iterative descent methods that:

(i) "directly" generate the approximate minimizers $p(j)$ of the cubic model, so that the evaluation of the true objective function can be performed without additional costs or the need of memory requirements;

(ii) are matrix-free methods (as, for instance, gradient methods, limited memory Quasi-Newton methods, conjugate gradient methods) in order to reduce the computational cost of linear algebra operations.

On these bases, in Algorithm 4.1 we report the conceptual scheme of an iterative method, using the analyzed stopping criteria, for the minimization of the model $m_k(p)$.

Finally, for the sake of completeness in Algorithm 4.2 we report the k-iteration of the proposed version of ARC which is formally described below.

Note that the trial point $\tilde{p}_k$ must satisfy condition (4.4) for the successful iterations. This condition aims to ensure the bound on the global worst-case iteration complexity.

Algorithm 4.1: Sketch of an iterative method

Given $p(0) = p_k^c$, $\epsilon > 0$ and integers $N \geq 1$, $j_{max} \in [1, +\infty]$. Set $j = 0$.
While (2.14) and (4.2) are not satisfied and $j \leq j_{max}$
Set $p(j+1) = p(j) + \alpha(j)d(j)$
where $d(j)$ is a descent direction and $\alpha(j)$ is computed by means of a line search.
Set $j = j + 1$.
End While
If (2.14) holds, **then** set $\tilde{p}_k = p(j)$.
Else set $\tilde{p}_k = p(j - N)$.

The global convergence of Algorithm 4.2 follows from Proposition 2.2 while its complexity bound is stated below.

PROPOSITION 4.1. *Let $\{x_k\}$ be the sequence generated by Algorithm* 4.2*, and let $\{f(x_k)\}$ be bounded below. Assume that:*

(i) *$f \in C^2(\mathbb{R}^n)$;*
(ii) *$\|g(x) - g(y)\| \leq \mathcal{K}\|x - y\|$, for all $x, y \in X$ an open convex set containing the iterates and $\mathcal{K} \geq 1$;*
(iii) *$\|H(x) - H(x_k)\| \leq L\|x - x_k\|$, for all $x \in [x_k, x_k + p_k]$ and all $k \geq 0$;*
(iv) *$\|(H(x_k) - B_k)p_k\| \leq C\|p_k\|^2$, for all $k \geq 0$ and some positive constant C;*
(v) *$\sigma_k \geq \sigma_{min}$ for all $k \geq 0$ and some positive σ_{min}.*

Then Algorithm 2.1 *requires at most $\mathcal{O}(\epsilon^{-3/2})$ iterations to attain*

$$\|g(x_k)\| \leq \epsilon.$$

Proof. For all successful iterations k the step p_k satisfies (3.1) either because $\tilde{p}_k$ does or because p_k is computed by Algorithm 3.1 and by virtue of Proposition 3.4. Then, the result follows from Proposition 3.1. □

5. Numerical experiments. In this section we report the results of the computational experiments performed in order to assess the effectiveness of ARC algorithms using iterative methods and early stopping.

Iterative cubic subproblem solvers

As iterative descent method applied to the cubic model $m_k(p)$ (see Algorithm 4.1) for computing the trial step p_k we employ the Barzilai-Borwein nonmonotone gradient method (NMGRAD) defined in [16]. The proposed version of ARC algorithm, called ARC-NMGRAD, has bee compared with the original ARC version (ARC-GLTR) proposed in [5], and using the Lanczos-based inexact solver implemented in GALAHAD-GLTR [11, 12].

Test problems

We selected all the CUTEr [13] variable dimension nonlinear and unconstrained problems, thus coming up with a set of 52 CUTEr problems. Then, we use them to define two subsets of, respectively, medium-sized ($n \in [1000, 2000]$) and large-sized ($n \in [4000, 5000]$) problems, as reported in Table 5.1.

Algorithm 4.2: k-th iteration of the proposed version of ARC

Given x_k, the scalars $\sigma_k > 0$, $1 > \eta_2 \geq \eta_1 > 0$, $\gamma_2 \geq \gamma_1 > 1$, $\epsilon > 0$, $\alpha > 0$, and the integer $N \geq 1$.

1. Compute the Cauchy point p_k^c, and apply Algorithm 4.1 to compute $\tilde{p}_k$.
   If
   $$\frac{f(x_k) - f(x_k + \tilde{p}_k)}{f(x_k) - m_k(\tilde{p}_k)} \geq \eta_1 \tag{4.3}$$
   then, if
   $$f(x_k) - m_k(\tilde{p}_k) \geq \alpha \epsilon^{3/2}, \tag{4.4}$$
   then set $p_k = \tilde{p}_k$ and go to Step 3; otherwise, apply Algorithm 3.1 to compute p_k.
2. Compute
   $$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{f(x_k) - m_k(p_k)}. \tag{4.5}$$
3. Set
   $$x_{k+1} = \begin{cases} x_k + p_k & \text{if } \rho_k \geq \eta_1, \\ x_k & \text{otherwise.} \end{cases}$$
4. Set
   $$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if } \rho_k \geq \eta_2 & \text{(very successful iteration),} \\ [\sigma_k, \gamma_1 \sigma_k) & \text{if } \eta_1 \leq \rho_k \leq \eta_2 & \text{(successful iteration),} \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise} & \text{(unsuccessful iteration).} \end{cases}$$

ARC implementation details

We have implemented the ARC algorithms as Matlab m-files and interfaced with CUTEr, with B_k set to the true Hessian $H(x_k)$. The two cubic subproblem solvers, GALAHAD-GLTR and NMGRAD, respectively, are implemented in Fortran and interfaced with Matlab and CUTEr.

The parameters defining the original ARC method (Algorithm 2.1) and the proposed ARC algorithm (Algorithm 4.2) have been chosen as described in [5]. The parameter α of condition (4.4) in Algorithm 4.2 has been set equal to 10^{-6}.

For all algorithms, the maximum number of outer iterations has been fixed equal to 10000, furthermore, a limit of two hours on the computing time has been imposed.

The algorithms stop when
$$\|g(x_k)\| \leq \epsilon, \qquad \text{with } \epsilon = 10^{-5}.$$

Implementation details on subproblem solution

The GALAHAD-GLTR solver of the original ARC method has been run with a memory parameter $m = 10$ and requiring 90% accuracy in solution reconstruction [14].

	medium size	large size		medium size	large size
ARWHEAD	1000	5000	FLETCBV2	1000	4000
BDQRTIC	1000	4000	FLETCBV3	1000	4000
BROWNBS	1000	4000	FLETCHBV	1000	4000
BROYDN7D	1000	5000	FLETCHCR	1000	4000
BRYBND	1000	4000	FMINSRF2	1024	4096
CHAINWOO	1000	4000	FREUROTH	1000	4000
CRAGGLVY	1000	5000	GENHUMPS	1000	4000
CURLY10	1000	4000	GENROSE	1000	4000
CURLY20	1000	4000	LIARWHD	1000	4000
CURLY30	1000	4000	MOREBV	1000	4000
DIXMAANA	1500	4500	NONCVXU2	1000	4000
DIXMAANB	1500	4500	NONCVXUN	1000	4000
DIXMAANC	1500	4500	NONMSQRT	1024	3600
DIXMAAND	1500	4500	NONDIA	1000	4000
DIXMAANE	1500	4500	NONDQUAR	1000	4000
DIXMAANF	1500	4500	OSCIPATH	1000	4000
DIXMAANG	1500	4500	POWELLSG	1000	4000
DIXMAANH	1500	4500	QUARTC	1000	3000
DIXMAANI	1500	4500	SINQUAD	1000	4000
DIXMAANJ	1500	4500	SPARSINE	1000	4000
DIXMAANK	1500	4500	SPARSQUR	1000	4000
DIXMAANL	1500	4500	SPMSRTLS	1000	3997
DQRTIC	1000	4000	SROSENBR	1000	4000
EDENSCH	2000	4000	TOINTGSS	1000	4000
ENGVAL1	1000	4000	TQUARTIC	1000	4000
EXTROSNB	1000	4000	WOODS	1000	4000

TABLE 5.1
Test problem dimensions

Concerning the early stopping parameter N of Algorithm 4.1, we conducted an experimentation to understand its influence on the overall algorithm. In particular, we compared three versions of the algorithm with the early stopping parameter $N = 5, 10, \infty$, where the symbol ∞ indicates that the parameter N in Algorithm 4.1 has been set equal to "infinity" to turn off the early stopping criterion. Below, the version of the algorithm ARC-NMGRAD with early stopping parameter N will be indicated as ARC-NMGRAD(N).

The subproblem solvers have been invoked specifying a limit of 1000 iterations (i.e., $j_{max} = 1000$ in Algorithm 4.1), and using 10^{-8} as value of θ in the relative stopping criterion (2.14).

Numerical results

All the methods have been compared using the performance profiles [10]; the profiles displayed refer to the run on the set of 104 problems including both the medium-sized and the large-sized problems listed above.

First we compare the relative efficiency of ARC-NMGRAD(5), ARC-NMGRAD(10) and ARC-NMGRAD(∞) to show the effectiveness of the introduced early stopping criterion. The results are plotted in Fig. 5.1. The benefits deriving from the adoption of the early stopping are evident. Indeed, both the versions using the early stopping criterion clearly outperform the version with $N = \infty$ in terms both of efficiency and robustness. Furthermore, it can be noted that the best performance is obtained with the version with $N = 5$. Hence, in the following we compare ARC-GLTR against ARC-NMGRAD with $N = 5$.

We remark that ARC-GLTR and ARC-NMGRAD(5) solve the same number of problems, that is 76 over 104 problems. Figures 5.2 and 5.3 display the iteration-count and CPU time performance profiles for the two methods. We may observe that

ARC-NMGRAD(5) requires a lower number of iteration in most runs and results to be much faster than ARC-GLTR.

On the basis of results that we do not report in the paper, we can further state that the ARC-NMGRAD(10) outperforms ARC-GLTR in terms of cpu time, whereas the performance ARC-NMGRAD(∞) is similar to that of ARC-GLTR.

Summarizing, the results of the computational experiments show that the employment of a gradient-based method as inexact solver is a valid alternative to ARC-GLTR, and the adoption of the early stopping criterion enhances the performance of ARC-NMGRAD and leads to a very efficient variant of ARC. Indeed, it appears that the early stopping strategy is a useful tool to properly manage the "over-solving" issue arising whenever the cubic model is not an adequate model of the true objective function.

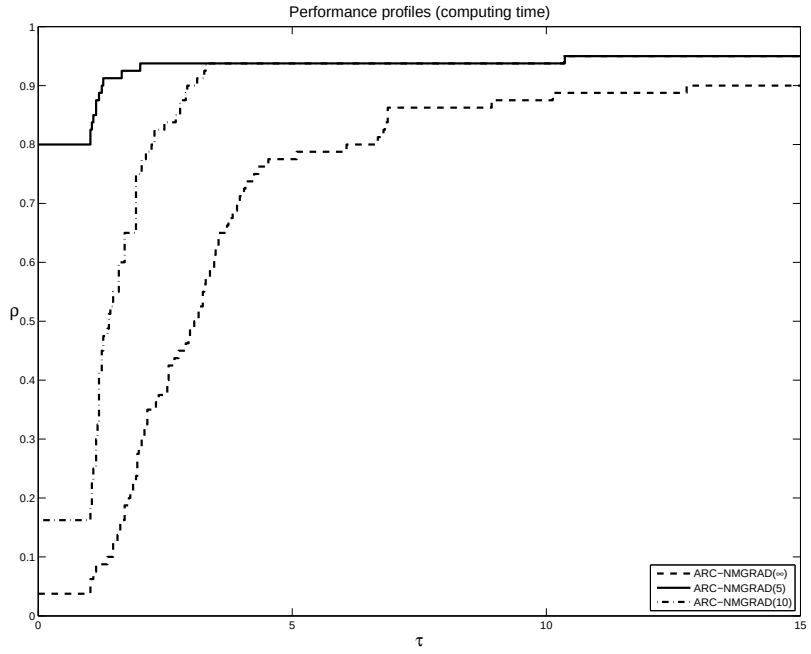

FIG. 5.1. *Performance profiles, $\rho(\tau)$: computing time*

6. Acknowledgments. The Authors are indebted with Prof. Philippe L. Toint for his valuable insights and suggestions on the worst-case complexity of ARC algorithms that led to significant improvements of the paper.

REFERENCES

[1] S. Bellavia, C. Cartis, N.I.M. Gould, B. Morini, Ph.L. Toint: Convergence of a Regularized Euclidean Residual Algorithm for Nonlinear Least-Squares. SIAM J. Numer. Anal. **48**, 1–29 (2010)

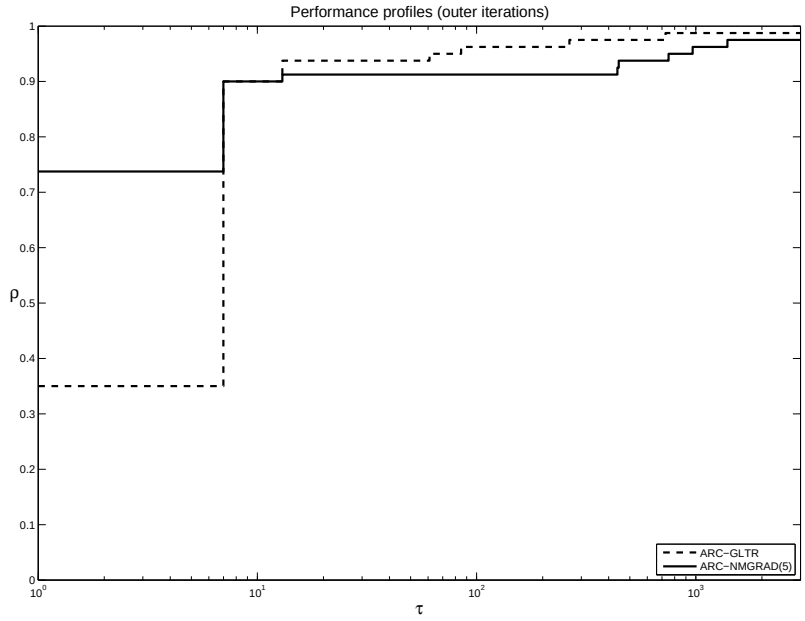

FIG. 5.2. *Performance profiles, $\rho(\tau)$: outer iterations*

[2] S. Bellavia, B. Morini: Strong local convergence properties of adaptive regularized methods for nonlinear least-squares. Technical Report n. 1/2013. Dipartimento di Ingegneria Industriale (2013)

[3] H.Y. Benson, D.F. Shanno: Interior-Point methods for nonconvex nonlinear programming: cubic regularization. Oprimization Online (2012)

[4] C.M.Bishop: Pattern Recognition and Machine Learning. Series: Information Science and Statistics. Springer (2006)

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

[6] C.Cartis, N.I.M. Gould, Ph.L. Toint: Trust-region and other regularizations of linear least-squares problems. BIT Numerical Mathematics **49**, 21–53 (2009)

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

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

[9] C. Cartis, N.I.M. Gould, Ph. L. Toint: An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. IMA J. Numer. Anal. to appear (2013)

[10] E.D. Dolan and J.J. Moré: Benchmarking Optimization Software with Performance Profiles. Math. Program. Ser. A **91**, 201–213 (2002)

[11] N.I.M. Gould, S. Lucidi, M. Roma,Ph.L. Toint: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. **9**, 504–525 (1999)

[12] N.I.M. Gould, D. Orban, and Ph.L. Toint: GALAHAD—a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. ACM T. Math. Software. **29**, 353–372 (2003)

[13] N.I.M. Gould, D. Orban, and Ph.L. Toint: CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited. ACM T. Math. Software **29**, 373–394 (2003)

[14] N.I.M. Gould, M. Porcelli, Ph.L. Toint: Updating the regularization parameter in the adaptive

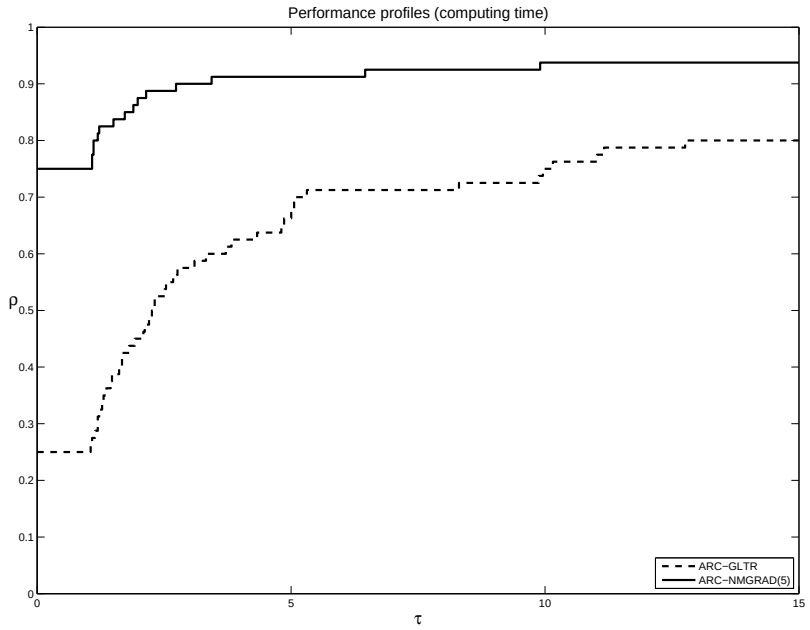

FIG. 5.3. *Performance profiles, $\rho(\tau)$: computing time*

cubic regularization algorithm. Comput. Optim. Appl. **53**, 1–22 (2012).

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

[16] L. Grippo, M. Sciandrone: Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. Comput. Optim. Appl. **23**, 143–169 (2002)

[17] Yu. Nesterov: Modified Gauss-Newton scheme with worst-case guarantees for global performance. Optim. Methods Softw. **22**, 469–483 (2007)

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

[19] Ph.L. Toint: Nonlinear Stepsize Control, Trust Regions and Regularizations for Unconstrained Optimization. Optim. Methods Soft. **28**, 82–95 (2013)

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