A line-search algorithm inspired by the adaptive cubic regularization framework with a worst-case complexity $\mathcal{O}(\epsilon^{-3/2})$

E. Bergou * Y. Diouane† S. Gratton‡

December 4, 2017

Abstract

Adaptive regularized framework using cubics (ARC) has emerged as an alternative to line-search and trust-region for smooth nonconvex optimization, with an optimal complexity amongst second-order methods. In this paper, we propose and analyze the use of a special (iteration dependent) scaled norm in ARC of the form $\|x\|_M = \sqrt{x^\top Mx}$ for $x \in \mathbb{R}^n$, where M is a symmetric positive definite matrix satisfying specific secant equation. Within the proposed norm, ARC behaves as a line-search algorithm along the Newton direction, with a special backtracking strategy and acceptability condition. Under appropriate assumptions, the obtained algorithm enjoys the same convergence and complexity properties as ARC, in particular the complexity for finding an approximate first-order stationary point is $\mathcal{O}(\epsilon^{-3/2})$. Furthermore, using the same scaled norm in the trust-region framework, we have derived a second line-search algorithm. The good potential of the obtained algorithms is showed on a set of large scale optimization problems.

Keywords: Nonlinear optimization, unconstrained optimization, line-search methods, adaptive regularized framework using cubics, trust-region methods, worst-case complexity.

1 Introduction

We consider the following unconstrained optimization problem:

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a given smooth function known as the objective function. Classical iterative methods for solving (1) are trust-region (TR) [8, 19] and line-search (LS) [10] algorithms. Recently, the use of cubic regularization has been first investigated by Nesterov and Polyak [17] and then Cartis *et al* [5] propose a generalization to an adaptive regularized framework using cubics (ARC).

*MaIAGE, INRA, Université Paris-Saclay, 78350 Jouy-en-Josas, France (elhoucine.bergou@inra.fr).

†Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse Cedex 4, France (youssef.diouane@isae.fr).

‡INP-ENSEEIHT, Université de Toulouse, 31071 Toulouse Cedex 7, France (serge.gratton@enseeiht.fr).

The worst-case evaluation complexity of finding an ϵ-approximate first-order critical point using TR or LS methods is shown to be computed in at most $\mathcal{O}(\epsilon^{-2})$ function or gradient evaluations, where $\epsilon \in (0,1)$ is a user-defined accuracy threshold on the gradient norm [16, 14, 7]. ARC takes at most $\mathcal{O}(\epsilon^{-3/2})$ function or gradient evaluations to reduce the gradient norm below ϵ, and thus it is improving substantially the worst-case complexity over the classical TR/LS methods [4]. Such complexity bound can be improved using higher order regularized models, we refer the reader for instance to the references [2, 6].

More recently, a non-standard TR method [9] is proposed with the same worst-case complexity bound as ARC. It is proved also that the same worst-case complexity $\mathcal{O}(\epsilon^{-3/2})$ can be achieved by mean of a specific variable-norm in a TR method [15] or using quadratic-regularization [3]. All previous approaches use a cubic sufficient-descent condition instead of the more usual predicted-reduction based descent. Generally, they need to solve more than one linear system in sequence at each outer iteration, this makes the computational cost per iteration expensive (By outer iteration, one means the sequence of the iterates generated by the the algorithm). In this work, under appropriate assumptions, we will derive an algorithm with the same worst-case complexity of $\mathcal{O}(\epsilon^{-3/2})$ without imposing any cubic sufficient-descent condition on the objective function and requiring only to solve (approximately) one linear system per iteration to get a trial point.

In [1], it has been shown how to use the so-called energy norm in the ARC/TR framework when a symmetric positive definite (SPD) approximation of the objective function Hessian is available. Within the energy norm, ARC/TR methods behave as LS algorithms along the Newton direction, with a special backtracking strategy and an acceptability condition in the spirit of ARC/TR methods. As far as the model of the objective function is convex (i.e., the approximate Hessian is SPD), in [1] we developed a methodology to make ARC behaves as an LS algorithm and showed that the resulting algorithm enjoys the same convergence and complexity analysis properties as ARC, in particular the first-order complexity bound of $\mathcal{O}(\epsilon^{-3/2})$. In the complexity analysis of ARC method [4], it is required that the Hessian approximation has to approximate accurately enough the true Hessian (see [4, Assumption AM.4]), obtaining such convex approximation may be out of reach when handling nonconvex optimization. This paper generalizes the proposed methodology in [1] to handle nonconvex models.

In this paper, we show that by using a special scaled norm the computational cost of the ARC subproblem is nearly the same as solving a linear system. In fact, the proposed approach consists of the use of a scaled norm to define the regularization term in ARC subproblem. The chosen scaled norm is of the form $\|x\|_M = \sqrt{x^\top M x}$ for $x \in \mathbb{R}^n$, where M is an SPD matrix that satisfies a specific secant equation. At each iteration we choose a possible different matrix M. (So, $M = M_k$ where the subscript k is the outer iteration index). Assuming that the Newton direction is not orthogonal with the gradient of f at the current iterate, the specific choice of M renders the ARC subproblem solution collinear with the Newton step and hence leads to an LS algorithm, along the Newton direction. The obtained LS algorithm enjoys the same convergence and worst-case complexity results as ARC method. Additionally, using subspace methods, we also consider a large-scale variant for the case where matrix factorizations are not affordable, implying that only iterative methods for computing a trial step can be used.

Compared to ARC (when using the ℓ_2-norm to define the regularization term in the subproblem), the dominant computational cost (regardless the function evaluation cost) of the resulting algorithm is mainly the cost of successful iterations. In fact, using the proposed scaled norm, the cost of the subproblem solution for unsuccessful iterations is getting inexpensive and requires only an update of a scalar.

An interesting feature of the obtained LS algorithm is the strategy of proceeding line search along the Newton direction which leads to the optimal complexity of ARC (independently from the convexity of the quadratic approximation of f). This is not the typical strategy that one may follow for defining a line-search method, especially when f is concave around the current iterate. The reason of using the Newton direction rather than another descent direction is to keep the worst case complexity bound of order $\epsilon^{-3/2}$. In fact, minimizing along the Newton direction ensures the existence of a scaled norm (which is uniformly equivalent to the Euclidean norm) such that a stationary point of the cubic model can be found along the same direction. Hence, the convergence and complexity analysis can be deduced directly from ARC algorithm. With a different direction, one is not sure about the existence of a scaled norm with the desired properties.

The proposed analysis of the obtained LS algorithm assumes that the Newton direction is not orthogonal with the gradient of f during the minimization process. To satisfy such assumption, we propose to check first if there exists an approximate of the Newton direction (among all the iterates generated using a subspace method) which is not orthogonal with the gradient and that satisfies the desired properties. If the subspace method reaches the exact solution and the assumption is still violated, we avoid such assumption by minimizing the cubic model using the ℓ_2-norm until a successful outer iteration is found. This procedure is included in the final LS algorithm. We note that in our numerical tests the latter restoration procedure was never activated, we believe from the practical point of view that such assumption is not very restrictive.

Similar analysis is applied to the TR framework, where using the same scaled norm (as in ARC) we show that TR method behaves as an LS algorithm. Numerical illustrations (over a test set of large scale optimization problems) are given in order to assess the efficiency of the obtained LS algorithms.

We organize this paper as follows. In Section 2, we introduce the ARC method using a general scaled norm. Section 3 analyses the minimization of the cubic model and discusses the choice of the scaled norm that simplifies solving the ARC subproblem. Section 4 derives the obtained LS algorithm on the base of ARC (when the proposed scaled norm is used) and discusses how the iteration dependent M-norm can be chosen uniformly equivalent to the Euclidean norm. We end the section by stating the necessary stopping criteria that is needed to maintain the complexity of ARC when the subproblem is solved iteratively. Similarly to ARC and using the same scaled norm, an LS algorithm in the spirit of TR algorithm is proposed in Section 5. Numerical tests are illustrated and discussed in Section 6. Conclusions and future improvements are given in Section 7.

Throughout this paper $\|\cdot\|$ will denote the vector or matrix ℓ_2-norm. $\|.\|_M$ will denote the scaled norm which is of the form $\|x\|_M = \sqrt{x^\top M x}$ for $x \in \mathbb{R}^n$ and where M is a given SPD matrix. We denote also by $\mathrm{sgn}(\alpha)$ the sign of a real α.

2 The ARC framework

At a given iterate x_k, we define $m_k^Q : \mathbb{R}^n \to \mathbb{R}$ as an approximate second-order Taylor approximation of the objective function f around x_k, i.e.,

$$m_k^Q(s) \quad = \quad f(x_k) + s^\top g_k + \frac{1}{2} s^\top B_k s, \tag{2}$$

where $g_k = \nabla f(x_k)$ is the gradient of f at the current iterate x_k, and B_k is a symmetric local approximation of the Hessian of f at x_k. For ARC [5], the trial step s_k approximates the global minimizer of the cubic model $m_k^C(s) = m_k^Q(s) + \frac{1}{3}\sigma_k\|s\|_{M_k}^3$, i.e.,

$$s_k^{\text{ARC}} \approx \arg\min_{s\in\mathbb{R}^n} m_k^C(s), \tag{3}$$

where M_k is a positive definite matrix, and $\sigma_k > 0$ is a dynamic positive parameter that might be regarded as the reciprocal of the TR radius in TR algorithms (see [5]). The parameter σ_k is taking into account the agreement between the objective function f and the model m_k^C. To decide whether the trial step is acceptable or not a ratio between the actual reduction and the predicted reduction is computed, as follows:

$$\rho_k \;=\; \frac{f(x_k) - f(x_k + s_k^{\text{ARC}})}{f(x_k) - m_k^Q(s_k^{\text{ARC}})}. \tag{4}$$

For a given scalar $0 < \eta < 1$, the k^{th} outer iteration will be said *successful* if $\rho_k \geq \eta$, and *unsuccessful* otherwise. For all *successful* iterations we set $x_{k+1} = x_k + s_k^{\text{ARC}}$; otherwise the current iterate is kept unchanged $x_{k+1} = x_k$. We note that, unlike the original ARC [5, 4] where the cubic model is used to evaluate the denominator in (4), in the nowadays works related to ARC, only the quadratic approximation $m_k^Q(s_k^{\text{ARC}})$ is used in the comparison with the actual value of f without the regularization parameter (see [2] for instance). Algorithm 1 gives a detailed description of ARC.

Algorithm 1: ARC algorithm.

Data: select an initial point x_0 and the constant $0 < \eta < 1$. Set the initial regularization $\sigma_0 > 0$, and the constants $0 < \nu_1 \leq 1 < \nu_2$.

for $k = 1, 2, \ldots$ **do**

Step 1: compute the gradient g_k and an approximated Hessian B_k;

Step 2: compute the step s_k^{ARC} as an approximate solution of (3) such that

$$m_k^C(s_k^{\text{ARC}}) \;\leq\; m_k^C(s_k^{\text{Cauchy}}) \tag{5}$$

where $s_k^{\text{Cauchy}} = -\delta_k^{\text{Cauchy}} g_k$ and $\delta_k^{\text{Cauchy}} = \arg\min_{t>0} m_k^C(-tg_k)$;

Step 3: if $\rho_k \geq \eta$ **then**

set $x_{k+1} = x_k + s_k^{\text{ARC}}$ and $\sigma_{k+1} = \nu_1\sigma_k$;

else

set $x_{k+1} = x_k$ and $\sigma_{k+1} = \nu_2\sigma_k$;

end

end

The Cauchy step s_k^{Cauchy}, defined in Step 2 of Algorithm 1, is computationally inexpensive compared to the computational cost of the global minimizer of m_k^C. The condition (5) on s_k^{ARC} is sufficient for ensuring global convergence of ARC to first-order critical points. Under appropriate assumptions, convergence results of Algorithm 1 can be found in [5, 2]. Moreover, in this case, the algorithm ensures a function-evaluation complexity bound of order ϵ^{-2} to guarantee

$$\|g_k\| \;\leq\; \epsilon, \tag{6}$$

where $\epsilon > 0$ is a pre-defined constant.

If B_k is set to be equal to the exact Hessian of the problem in Step 1 of Algorithm 1, one can improve the function-evaluation complexity to be of the order of $\epsilon^{-3/2}$ for ARC algorithm by imposing, in addition to the Cauchy decrease, another termination condition on the computation of the trial step s_k^{ARC} (see [4, 2]). Such a termination condition is of the form

$$\|\nabla m_k^C(s_k^{\text{ARC}})\| \quad \leq \quad \zeta \|s_k^{\text{ARC}}\|^2, \tag{7}$$

where $\zeta > 0$ is a given constant chosen at the start of the algorithm.

When only an approximation of the Hessian is available in Step 1 of Algorithm 1, an additional condition has to be imposed on the Hessian approximation B_k in order to ensure an optimal complexity of order $\epsilon^{-3/2}$. Such condition is often considered as (see [4, Assumption AM.4]):

$$\|(\nabla^2 f(x_k) - B_k) s_k^{\text{ARC}}\| \leq C \|s_k^{\text{ARC}}\|^2 \tag{8}$$

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

From now on, we will assume that first-order stationarity is not reached yet, meaning that the gradient of the objective function is non null at the current iteration k (i.e., $g_k \neq 0$).

3 On the cubic model minimization

In this section, we will mostly focus on the solution of the subproblem (3) for a given outer iteration k. Thus the subscript k will be dropped to keep the notations simple. In a precedent work [1], when the matrix B is assumed to be positive definite, we showed that the minimizer s^{ARC} of the cubic model defined in (3) is getting collinear with the Quasi-Newton direction $(-B^{-1}g)$ when the matrix M is set to be equal to B. In this section we generalize our proposed approach to cover the case where the linear system $Bs = -g$ admits a solution and B is not necessarily SPD. We will explicit the condition to impose on the matrix M in order to get the solution of the ARC subproblem at a modest computational cost.

3.1 Exact solution of the ARC subproblem

For the purpose of this subsection, the vector s^Q will denote an exact solution of $Bs = -g$ when it exists (i.e., s^Q corresponds to a stationary point of m^Q). Moreover, assuming that s^Q is not orthogonal with the gradient g, let d^{N} denotes a scaled Quasi-Newton direction, i.e., $d^{\text{N}} = -\frac{s^Q}{g^\top s^Q}$. For the sake of simplicity, in what comes next, we shall refer to d^{N} as the Newton direction. Note that such direction is a descent direction for the objective function f (i.e. $g^\top d^{\text{N}} < 0$). In this subsection, we state our precise assumption formally as follows:

Assumption 3.1 *The model m^Q admits a stationary point s^Q such as $|g^\top s^Q| \geq \epsilon_d \|g\| \|s^Q\|$ where $\epsilon_d > 0$ is a pre-defined positive constant.*

When such assumption is violated, in Section 4, we explain our proposed approach to restore such assumption in the final algorithm (see Algorithm 2).

In what comes next, we define s_N^{ARC} as the Newton-like step associated with the minimization of the cubic model m^C:

$$s_N^{\text{ARC}} = \alpha_N d^{\text{N}}, \quad \text{where} \quad \alpha_N = \arg\min_{t>0} m_k^C(t d^{\text{N}}). \tag{9}$$

The next theorem gives the explicit form of s_N^{ARC}.

Theorem 3.1 *Let Assumption 3.1 hold. The Newton-like step (9) is given by:*

$$s_N^{\mathrm{ARC}} \;=\; \delta^{\mathrm{N}} s^Q, \quad \textit{where} \quad \delta^{\mathrm{N}} = \frac{2}{1-\mathrm{sgn}(g^\top s^Q)\sqrt{1+4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|}}}. \tag{10}$$

Proof. Indeed, for all $t>0$, one has

$$m^C(td^{\mathrm{N}}) - m^C(0) \;=\; tg^\top d^{\mathrm{N}} + \frac{t^2}{2}[d^{\mathrm{N}}]^\top [Bd^{\mathrm{N}}]^\top + \frac{\sigma t^3}{3}\|d^{\mathrm{N}}\|_M^3 = -t - \frac{1}{g^\top s^Q}\frac{t^2}{2} + \frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}\frac{t^3}{3}.$$

We compute the value of the parameter t at which the unique minimizer of the above function is attained. Let α_N denotes this optimal parameter. Taking the derivative of (11) with respect to t and equating the result to zero, one gets

$$0 \;=\; -1 - \frac{1}{g^\top s^Q}\alpha_N + \frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}(\alpha_N)^2, \tag{11}$$

and thus, since $\alpha_N > 0$,

$$\alpha_N \;=\; \frac{\frac{1}{g^\top s^Q} + \sqrt{\left(\frac{1}{g^\top s^Q}\right)^2 + 4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}}}{2\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}} = \frac{2g^\top s^Q}{-1+\mathrm{sgn}(g^\top s^Q)\sqrt{1+4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|}}}.$$

Hence, $s_N^{\mathrm{ARC}} = \delta^{\mathrm{N}} s^Q$, where $\delta_N^{\mathrm{ARC}} = \frac{2}{1-\mathrm{sgn}(g^\top s^Q)\sqrt{1+4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|}}}$. ■

In general, the matrix M can be chosen arbitrarily as long as it is an SPD matrix. Our goal, in this paper, is to determine how we can choose the matrix M so that the Newton-like step s_N^{ARC} becomes a stationary point of subproblem (3). The following theorem gives explicitly the necessary and sufficient condition on the matrix M to reach this aim.

Theorem 3.2 *Let Assumption 3.1 hold. The step s_N^{ARC} is a stationary point for the subproblem (3) if and only if there exists $\theta>0$ such that $Ms^Q = \frac{\theta}{g^\top s^Q}g$. Note that $\theta = \|s^Q\|_M^2$.*

Proof. Indeed, if we suppose that the step s_N^{ARC} is a stationary point of the subproblem (3), this means that

$$\nabla_s m^C(s_N^{\mathrm{ARC}}) \;=\; g + Bs_N^{\mathrm{ARC}} + \sigma\|s_N^{\mathrm{ARC}}\|_M M s_N^{\mathrm{ARC}} = 0, \tag{12}$$

In another hand, $s_N^{\mathrm{ARC}} = \alpha_N d^{\mathrm{N}}$ where $d^{\mathrm{N}} = -\frac{s^Q}{g^\top s^Q}$ and $\alpha_N > 0$ solution of $-1 - \frac{1}{g^\top s^Q}\alpha_N + \frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}(\alpha_N)^2 = 0$. Hence, we obtain that

$$\begin{aligned} 0 \;&=\; \nabla_s m^C(s_N^{\mathrm{ARC}}) = g + \frac{\alpha_N}{g^\top s^Q}g - \sigma(\alpha_N)^2\frac{\|s^Q\|_M}{|g^\top s^Q|}\frac{Ms^Q}{g^\top s^Q} \\ &=\; \left(1+\frac{\alpha_N}{g^\top s^Q}\right)g - \left(\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}(\alpha_N)^2\right)\left(\frac{g^\top s^Q}{\|s^Q\|_M^2}Ms^Q\right) \\ &=\; \left(\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|^3}(\alpha_N)^2\right)\left(g - \frac{g^\top s^Q}{\|s^Q\|_M^2}Ms^Q\right). \end{aligned}$$

Equivalently, we conclude that $Ms^Q = \frac{\theta}{g^\top s^Q} g$ where $\theta = \|s^Q\|_M^2 > 0$. ■

The key condition to ensure that the ARC subproblem stationary point is equal to the Newton-like step $s_N^{\rm ARC}$, is the choice of the matrix M which satisfies the following secant-like equation $Ms^Q = \frac{\theta}{g^\top s^Q} g$ for a given $\theta > 0$. The existence of such matrix M is not problematic as far as Assumption 3.1 holds. In fact, Theorem 4.1 explicits a range of $\theta > 0$ for which the matrix M exists. Note that in the formula of $s_N^{\rm ARC}$ such matrix is used only through the computation of the M-norm of s^Q. Therefore an explicit formula of the matrix M is not needed, and only the value of $\theta = \|s^Q\|_M^2$ suffices for the computations.

When the matrix M satisfies the desired properties (as in Theorem 3.2), one is ensured that $s_N^{\rm ARC}$ is a stationary point for the model m^C and hence condition (7) is automatically satisfied. However, in addition to such condition, ARC algorithm imposes on the approximate step also to satisfy the Cauchy decrease condition (5) for a given σ. The latter condition is not guaranteed by $s_N^{\rm ARC}$ as the model m^C may be non-convex. In the next theorem, we show that for a sufficiently large σ, $s_N^{\rm ARC}$ is getting the global minimizer of m^C and thus satisfying the Cauchy decrease is not an issue anymore.

Theorem 3.3 *Let Assumption 3.1 hold. Let M be an SPD matrix which satisfies $Ms^Q = \frac{\theta}{g^\top s^Q} g$ for a fixed $\theta > 0$. If the matrix $B + \sigma\|s_N^{\rm ARC}\|_M M$ is positive definite, then the step $s_N^{\rm ARC}$ is the unique minimizer for the subproblem (3).*

Proof. Indeed, using [8, Theorem 3.1], we have that, for a given vector s_*, it is a global minimizer of m^C if and only if it satisfies

$$(B + \lambda_* M)s_* = -g$$

where $B + \lambda_* M$ is positive semidefinite matrix and $\lambda_* = \sigma\|s_*\|_M$. Moreover, if $B + \lambda_* M$ is positive definite, s_* is unique.

Since $Ms^Q = \frac{\theta}{g^\top s^Q} g$, by applying Theorem 3.2, we see that

$$(B + \lambda_N M)s_N^{\rm ARC} = -g$$

with $\lambda_N = \sigma\|s_N^{\rm ARC}\|_M$. Thus, if we assume that $B + \lambda_N M$ is positive definite matrix, then $s_N^{\rm ARC}$ is the unique global minimizer of the subproblem (3). ■

Theorem 3.3 states that the step $s_N^{\rm ARC}$ is the global minimizer of the cubic model m^C as far as the matrix $B + \lambda_N M$ is positive definite, where $\lambda_N = \sigma\|s_N^{\rm ARC}\|_M$. Note that

$$\lambda_N \;=\; \sigma\|s_N^{\rm ARC}\|_M = \frac{2\sigma\|s^Q\|_M}{\left|1 - {\rm sgn}(g^\top s^Q)\sqrt{1 + 4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|}}\right|} \to +\infty \quad \text{as } \sigma \to \infty.$$

Thus, since M is an SPD matrix and the regularization parameter σ is increased for unsuccessful iterations in Algorithm 1, the positive definiteness of matrix $B+\lambda_N M$ is guaranteed after finitely many unsuccessful iterations. In other words, one would have insurance that $s_N^{\rm ARC}$ will satisfy the Cauchy decrease after a certain number of unsuccessful iterations.

Another important practical consequence of Theorem 3.3 is that solving exactly the ARC subproblem amounts to computing the solution of the linear system $Bs = -g$ and evaluating the scalar in (10). This means that solving the subproblem, while varying the value of σ,

is inexpensive once the solution of $Bs = -g$ is found. This remark will be essential when considering unsuccessful steps in the overall optimization algorithm. Another consequence is that the subproblem solution can be obtained using a direct method if the matrix B is not too large. Typically, one can use the LDL^T factorization to solve this linear system. In the next section we consider the case where $Bs = -g$ is solved iteratively.

3.2 Approximate solution using subspace methods

For large scale optimization problems, computing s^Q can be prohibitively computationally expensive. We will show that it will be possible to relax this requirement by letting the step s^Q be only an approximation of the exact solution.

For an approximate solution, as far as the global convergence of Algorithm 1 is concerned, all what we need is that the solution of the subproblem (3) yields a decrease in the cubic model which is as good as a the Cauchy decrease as emphasized in (5). In practice, a version of Algorithm 1 solely based on the Cauchy step would suffer from the same drawbacks as the steepest descent algorithm on ill-conditioned problems and faster convergence can be expected if the matrix B influences also the minimization direction. The main idea consists of achieving a further decrease on the cubic model (better that the Cauchy decrease) by projection onto a sequence of embedded Krylov subspaces. We now show how to use a similar idea to compute a solution of the subproblem that is computationally cheap and yields the global convergence of Algorithm 1.

A classical way to approximate the exact solution s^Q is by using subspace methods (typically a Krylov subspace method). For that, let $\mathcal{L}_i$ be a subspace of $\mathbb{R}^n$ and l its dimension. Let Q_i denotes an $n \times l$ matrix whose columns form a basis of $\mathcal{L}_i$. Thus for all $s \in \mathcal{L}_i$, we have $s = Q_i z$, for some $z \in \mathbb{R}^l$. We will denote by $[s^Q]_i$ an exact stationary point of the model function m^Q over the subspace $\mathcal{L}_i$ (when it exists)). We define $[s_N^{\rm ARC}]_i$ as the projection, onto the subspace $\mathcal{L}_i$, of the step $s_N^{\rm ARC}$ associated with the subproblem (3), that is $[s_N^{\rm ARC}]_i = Q_i z_N$ where z_N is the Newton-like step associated to the following reduced subproblem:

$$\min_{z \in \mathbb{R}^l} f(x) + z^\top Q_i^\top g + \frac{1}{2} z^\top Q_i^\top B Q_i z + \frac{1}{3}\sigma \|z\|^3_{Q_i^\top M Q_i}. \tag{13}$$

We update Assumption 3.1 in this subsection as follows:

Assumption 3.2 *The model m^Q admits a stationary point $[s^Q]_i$ over the subspace $\mathcal{L}_i$ such as $|g^\top [s^Q]_i| \geq \epsilon_d \|g\| \|[s^Q]_i\|$ where $\epsilon_d > 0$ is a pre-defined positive constant.*

As already mentioned in the exact case, a procedure to restore the violation of such assumption will be included in the final algorithm (see Algorithm 2).

We will now give the extension of the results shown in the previous subsection (the exact case). The next two theorems are the extension of Theorems 3.1 and 3.2 in the inexact case.

Theorem 3.4 *Let Assumption 3.2 hold. Then one has*

$$[s_N^{\rm ARC}]_i \;=\; [\delta^{\rm N}]_i [s^Q]_i \quad \textit{where} \quad [\delta^{\rm N}]_i = \frac{2}{1 - {\rm sgn}(g^\top [s^Q]_i)\sqrt{1 + 4\frac{\sigma \|[s^Q]_i\|^3_M}{|g^\top [s^Q]_i|}}}. \tag{14}$$

Proof. Indeed, one has $[s_N^{\rm ARC}]_i = Q_i z_N$, where z_N is the Newton-like step associated to the reduced subproblem (13). Hence by applying Theorem 3.1 to the reduced subproblem (13), it follows that

$$z_N = [\delta^{\rm N}]_i z^Q \quad \text{where} \quad [\delta^{\rm N}]_i = \frac{2}{1 - {\rm sgn}((Q_i^\top g)^\top z^Q)\sqrt{1 + 4\frac{\sigma\|z^Q\|^3_{Q_i^\top M Q_i}}{|(Q_i^\top g)^\top z^Q|}}},$$

where z^Q is a stationary point of the quadratic part of the minimized model in (13). Thus, by substituting z_N in the formula $[s_N^{\rm ARC}]_i = Q_i z_N$, one gets

$$\begin{aligned}
[s_N^{\rm ARC}]_i &= Q_i \left(\frac{2}{1 - {\rm sgn}((Q_i^\top g)^\top z^Q)\sqrt{1 + 4\frac{\sigma\|z^Q\|^3_{Q_i^\top M Q_i}}{|Q_i^\top g)^\top z^Q|}}} z^Q \right) \\
&= \frac{2}{1 - {\rm sgn}(g^\top Q_i z^Q)\sqrt{1 + 4\frac{\sigma\|Q_i z^Q\|^3_M}{|g^\top Q_i z^Q|}}} Q_i z^Q = \frac{2}{1 - {\rm sgn}(g^\top [s^Q]_i)\sqrt{1 + 4\frac{\sigma\|[s^Q]_i\|^3_M}{|g^\top [s^Q]_i|}}} [s^Q]_i.
\end{aligned}$$

■

Theorem 3.5 *Let Assumption 3.2 hold. The step $[s_N^{\rm ARC}]_i$ is a stationary point for the subproblem (3) over the subspace $\mathcal{L}_i$ if and only if there exists $\theta > 0$ such that M satisfies $M[s^Q]_i = \frac{\theta}{g^\top [s^Q]_i} g$.*

Proof. The proof is similar to the proof of Theorem 3.2 by considering $[s^Q]_i$ instead of s^Q. ■

As in the exact case, the key assumption to ensure that, the step $[s_N^{\rm ARC}]_i$ is a stationary point of (13), is the use of the M-norm where M is satisfying the following condition: $M[s^Q]_i = \frac{\theta}{g^\top [s^Q]_i} g$, for a given $\theta > 0$. Under Assumption 3.2, the existence of such matrix M is addressed in Theorem 4.1. Again we note that only θ is needed for the computation of the desired step.

Similarly to the exact case one can deduce the following result on the optimality of $[s_N^{\rm ARC}]_i$ over the subspace $\mathcal{L}_i$.

Theorem 3.6 *Let Assumption 3.2 hold. Let M be an SPD matrix which satisfies $M[s^Q]_i = \frac{\theta}{g^\top [s^Q]_i} g$ for a fixed $\theta > 0$. If the matrix $Q_i^\top (B + \sigma\|[s_N^{\rm ARC}]_i\|_M M) Q_i$ is positive definite, then the step $[s_N^{\rm ARC}]_i$ is the unique minimizer of the subproblem (3) over the subspace $\mathcal{L}_i$.*

Proof. Indeed, one has $M[s^Q]_i = \frac{\theta}{g^\top [s^Q]_i} g$ meaning that $Q_i^\top M Q_i z^Q = \frac{\theta}{(Q_i^\top g)^\top z^Q} Q_i^\top g$. Hence, if we suppose that the matrix $Q_i^\top (B + \lambda_N M) Q_i$ is positive definite, by applying Theorem 3.2 to the reduced subproblem (13), we see that the step z_N is the unique global minimizer of the subproblem (13). We conclude that $[s_N^{\rm ARC}]_i = Q_i z_N$ is the global minimizer of the subproblem (3) over the subspace $\mathcal{L}_i$. ■

Note that, when the growing subspaces $\mathcal{L}_i$ reach the whole space $\mathbb{R}^n$, the subspace approximation (in exact arithmetics) gives the exact solution of the linear system. On the base of this remark, from now on, we will consider that s^Q is computed using a subspace method as described in Subsection 3.2.

4 ARC algorithm using a specific M-norm

In this section, using the proposed M-norm, we show that ARC framework (Algorithm 1) behaves as an LS algorithm. For the sake of clarity, the subscript k is reused to indicate the outer iteration index all over our analysis. In the sequel, s_k^Q will denote an approximate solution of the linear system $B_k s = -g_k$ when it exists.

Under Assumption 3.2 and given an SPD matrix M_k such that $M_k s_k^Q = \frac{\theta_k}{g_k^\top s_k^Q} g_k$. By using the M_k-norm in the definition of the cubic model m_k^C, an approximate stationary point of the subproblem (3) is of the form

$$s_k^{\text{ARC}} \;=\; \delta_k s_k^Q, \;\; \text{where} \;\; \delta_k = \frac{2}{1 - \text{sgn}(g_k^\top s_k^Q)\sqrt{1 + 4\frac{\sigma_k \|s_k^Q\|_{M_k}^3}{|g_k^\top s_k^Q|}}}. \tag{15}$$

Moreover, on the base of Theorem 3.6 and using the update mechanism of σ_k in the algorithm, one can guarantee that, after a finite number of unsuccessful iterations, the approximate solution s_k^{ARC} will satisfy the Cauchy decrease condition (5) as it will approach the global minimum of the cubic model m_k^C.

For *unsuccessful* iterations in Algorithm 1, it is required only to update the value of σ_k while the current step direction is kept unchanged. Hence, the approximate solutions of the subproblem given by (15) can be obtained only by updating the step-size δ_k. This means that the computational cost of *unsuccessful* iterations is getting cheap compared to solving subproblem as required by ARC when the Euclidean norm is used (see e.g. [5]). As a consequence, the use of the specific proposed M_k-norm in Algorithm 1 leads to a new formulation of ARC algorithm where the dominant computational cost is the cost of *successful* iterations. With such choice of the matrix M_k, ARC algorithm behaves as an LS method with a specific backtracking strategy and an acceptance criteria in the sprite of ARC algorithm. The trial step is of the form $s_k^{\text{ARC}} = \delta_k s_k^Q$ where the step length $\delta_k > 0$ is chosen such as

$$\rho_k = \frac{f(x_k) - f(x_k + s_k^{\text{ARC}})}{f(x_k) - m_k^Q(s_k^{\text{ARC}})} \geq \eta \quad \text{and} \quad m_k^C(s_k^{\text{ARC}}) \leq m_k^C(s_k^{\text{Cauchy}}), \tag{16}$$

or, equivalently,

$$f(x_k + \delta_k s_k^Q) \leq f(x_k) + \eta\left(\delta_k g_k^\top s_k^Q + \frac{\delta_k^2}{2}[s_k^Q]^\top B_k s_k^Q\right) \quad \text{and} \quad m_k^C(\delta_k s_k^Q) \leq m_k^C(-\delta_k^{\text{Cauchy}} g_k).$$

The step lengths δ_k and δ_k^{Cauchy} are computed respectively as follows:

$$\delta_k \;=\; \frac{2}{1 - \text{sgn}(g_k^\top s_k^Q)\sqrt{1 + 4\frac{\sigma_k \|s_k^Q\|_{M_k}^3}{|g_k^\top s_k^Q|}}}, \tag{17}$$

and

$$\delta_k^{\text{Cauchy}} \;=\; \frac{2}{\frac{g_k^\top B_k g_k}{\|g_k\|^2} + \sqrt{\left(\frac{g_k^\top B_k g_k}{\|g_k\|^2}\right)^2 + 4\frac{\sigma_k \|g_k\|_{M_k}^3}{\|g_k\|^2}}} \tag{18}$$

where σ_k is initially equals to the current value of the regularization parameter (as in the original ARC algorithm). For large values of δ_k the sufficient decrease condition (16) may not be satisfied. In this case, the value of σ_k is enlarged using an expansion factor $\nu_2 > 1$. Iteratively, the value of δ_k is updated (when σ_k increases, the absolute value of δ_k decreases) and the acceptance condition (16) is checked again, until its satisfaction. We denote ARC algorithm, when the proposed scaled M_k-norm is used, by LS-ARC (as it behaves as an LS type method in this case).

The computation of δ_k and δ_k^{Cauchy} using (17) and (18) requires the computation of the M_k-norm of the vectors s_k^Q and g_k respectively. One can notice that $\|s_k^Q\|_{M_k}$ is trivially given by $\theta_k^{1/2}$ but the computation of $\|g_k\|_{M_k}$ is not straightforward for a given choice of θ_k. In Theorem 4.1, we give explicitly the expression of $\|g_k\|_{M_k}$ for a range of choices for the parameter θ_k. By doing so, in the final algorithm LS-ARC, there will be no need to compute explicitly the matrix M_k and only a choice of the parameter θ_k has to be fixed.

To derive the convergence and complexity analyse of the LS-ARC algorithm from ARC, it suffices to show that the M_k-norm is uniformly equivalent to the Euclidean norm. The next theorem gives a range of choices for the parameter θ_k to ensure the existence of an SPD matrix M_k such as $M_k s_k^Q = \frac{\theta_k}{g_k^\top s_k^Q} g_k$ and the M_k-norm is uniformly equivalent to the ℓ_2-norm.

Theorem 4.1 *Let Assumption 3.2 hold. For all*

$$\theta_k = \beta_k \|s_k^Q\|^2 \quad \textit{where } \beta_k \in (\beta_{\min}, \beta_{\max}) \quad \textit{and} \quad \beta_{\max} > \beta_{\min} > 0, \tag{19}$$

there exists an SPD matrix M_k such as $M_k s_k^Q = \frac{\theta_k}{g_k^\top s_k^Q} g_k$, the M_k-norm is uniformly equivalent to the ℓ_2-norm on $\mathbb{R}^n$. Moreover, one has

$$\|g_k\|_{M_k}^2 = \beta_k \|g_k\|^2 \left(\frac{5}{2} - \frac{3}{2}\cos(\alpha_k)^2 + 2\left(\frac{1-\cos(\alpha_k)^2}{\cos(\alpha_k)} \right)^2 \right) \quad \textit{where } \cos(\alpha_k) = \frac{g_k^\top s_k^Q}{\|g_k\|\|s_k^Q\|}. \tag{20}$$

Proof. Let $\bar{s}_k^Q = \frac{s_k^Q}{\|s_k^Q\|}$ and $\bar{g}_k$ be an orthonormal vector to $\bar{s}_k^Q$ (i.e., $\|\bar{g}_k\| = 1$ and $\bar{g}_k^\top \bar{s}_k^Q = 0$) such that

$$\frac{g_k}{\|g_k\|} = \cos(\alpha_k)\bar{s}_k^Q + \sin(\alpha_k)\bar{g}_k. \tag{21}$$

For a given $\theta_k = \beta_k \|s_k^Q\|^2$ where $\beta_k \in (\beta_{\min}, \beta_{\max})$ and $\beta_{\max} > \beta_{\min} > 0$, one would like to construct an SPD matrix M_k such as $M_k s_k^Q = \frac{\theta_k g_k}{g_k^\top s_k^Q}$, hence

$$M_k \bar{s}_k^Q = \frac{\theta_k g_k}{g_k^\top s_k^Q \|s_k^Q\|} = \frac{\theta_k \|g_k\|}{g_k^\top s_k^Q \|s_k^Q\|} \left(\cos(\alpha_k)\bar{s}_k^Q + \sin(\alpha_k)\bar{g}_k \right) = \beta_k \bar{s}_k^Q + \beta_k \tan(\alpha_k)\bar{g}_k.$$

Using the symmetric structure of the matrix M_k, let γ_k be a positive parameter such as

$$M_k = \left[\bar{s}_k^Q, \bar{g}_k \right] N_k \left[\bar{s}_k^Q, \bar{g}_k \right]^\top \text{ where } N_k = \begin{bmatrix} \beta_k & \beta_k \tan(\alpha_k) \\ \beta_k \tan(\alpha_k) & \gamma_k \end{bmatrix}.$$

The eigenvalues $\lambda_k^{\min}$ and $\lambda_k^{\max}$ of the matrix N_k are the roots of

$$\lambda^2 - (\beta_k + \gamma_k)\lambda + \beta_k\gamma_k - (\beta_k \tan(\alpha_k))^2 = 0,$$

hence

$$\lambda_k^{\min} = \frac{(\beta_k + \gamma_k) - \sqrt{\omega_k}}{2} \quad \text{and} \quad \lambda_k^{\max} = \frac{(\beta_k + \gamma_k) + \sqrt{\omega_k}}{2},$$

where $\omega_k = (\beta_k - \gamma_k)^2 + 4\left(\beta_k \tan(\alpha_k)\right)^2$. Note that both eigenvalues are monotonically increasing as functions of γ_k.

One may choose $\lambda_k^{\min}$ to be equal to $\frac{1}{2}\beta_k = \frac{1}{2}\frac{\theta_k}{\|s_k^Q\|^2}$, therefore $\lambda_k^{\min} > \frac{1}{2}\beta_{\min}$ is uniformly bounded away from zero. Moreover we deduce from the expression of $\lambda_k^{\min}$ the following:

$$\gamma_k \;=\; 2\beta_k \tan(\alpha_k)^2 + \beta_k/2. \tag{22}$$

From Assumption 3.2, i.e., $|g_k^\top s_k^Q| \geq \epsilon_d \|g_k\| \|s_k^Q\|$ where $\epsilon_d > 0$, one has $\tan(\alpha_k)^2 \leq \tau_d \triangleq \frac{1-\epsilon_d^2}{\epsilon_d^2}$. Hence,

$$\gamma_k \leq 2\beta_k \tau_d + \beta_k/2 \leq 2\beta_{\max}\tau_d + \beta_{\max}/2.$$

Therefore γ_k is uniformly bounded from above, and by monotonicity, $\lambda_k^{\max}$ is also uniformly bounded from above.

A possible choice for the matrix M_k uniformly bounded can be obtained by completing the vectors family $\{\bar{s}_k^Q, \bar{g}_k\}$ to an orthonormal basis $\{\bar{s}_k^Q, \bar{g}_k, q_3, q_4, \ldots, q_n\}$ of $\mathbb{R}^n$ as follows:

$$M_k = [\bar{s}_k^Q, \bar{g}_k, q_3, \ldots, q_n] \begin{bmatrix} N_k & 0 \\ 0 & D \end{bmatrix} [\bar{s}_k^Q, \bar{g}_k, q_3, \ldots, q_n]^\top,$$

where $D = \operatorname{diag}(d_3, \ldots, d_n) \in \mathbb{R}^{(n-2)\times(n-2)}$ with positive diagonal entrees independent from k. One concludes that for all $\theta_k = \beta_k \|s_k^Q\|^2$ where $\beta_k \in (\beta_{\min}, \beta_{\max})$ and $\beta_{\max} > \beta_{\min} > 0$, the eigenvalue of the constructed M_k are uniformly bounded away from zero and from above, hence the scaled M_k-norm is uniformly equivalent to the ℓ_2-norm on $\mathbb{R}^n$.

Moreover, using (21) and (22), one has

$$\begin{aligned}
\|g_k\|_{M_k}^2 &= \|g_k\|^2 \left(\cos(\alpha_k)\bar{s}_k^Q + \sin(\alpha_k)\bar{g}_k\right)^\top \left(\cos(\alpha_k) M_k \bar{s}_k^Q + \sin(\alpha_k) M_k \bar{g}_k\right) \\
&= \|g_k\|^2 \left(\frac{\theta_k \cos(\alpha_k)^2}{\|s_k^Q\|^2} + \gamma_k \sin(\alpha_k)^2 + 2\sin(\alpha_k)\cos(\alpha_k)\frac{\theta_k \tan(\alpha_k)^2}{\|s_k^Q\|^2}\right) \\
&= \beta_k \|g_k\|^2 \left(\cos(\alpha_k)^2 + \frac{5}{2}\sin(\alpha_k)^2 + 2\sin(\alpha_k)^2\tan(\alpha_k)^2\right) \\
&= \beta_k \|g_k\|^2 \left(\frac{5}{2} - \frac{3}{2}\cos(\alpha_k)^2 + 2\left(\frac{1-\cos(\alpha_k)^2}{\cos(\alpha_k)}\right)^2\right).
\end{aligned}$$

■

In the next lemma, we show that when our proposed M_k-norm is used, the condition (7) imposed on the cubic model m_k^C can be expressed only in terms of s_k^Q and ∇m_k^Q. The latter condition is required in the algorithm at each iteration to ensure that it takes at most $\mathcal{O}(\epsilon^{-3/2})$ function-evaluation (and iteration) to reduce the gradient norm below $\epsilon \in (0, 1)$.

Lemma 4.1 *Let Assumption 3.2 hold. Let $\theta_k = \beta_k \|s_k^Q\|^2$ as in (19). Then imposing the condition (7) is equivalent to the following condition*

$$\|\nabla m_k^Q(s_k^Q)\| \quad \leq \quad \frac{2\,\mathrm{sgn}(g_k^\top s_k^Q)\zeta}{-1+\mathrm{sgn}(g_k^\top s_k^Q)\sqrt{1+4\frac{\sigma_k\theta_k^{3/2}}{|g_k^\top s_k^Q|}}}\|s_k^Q\|^2. \tag{23}$$

Proof. In the following proof we choose to omit the outer iteration index k. Since Assumption 3.2 holds and $\theta_k = \beta_k\|s_k^Q\|^2$ as in (19), Theorem 4.1 implies the existence of an SPD matrix M such that $Ms^Q = \frac{\theta}{g^\top s^Q} g$. Using such M-norm, an approximate solution of the cubic model m^C is of the form $s^{\mathrm{ARC}} = \alpha_N d^{\mathrm{N}} = -\alpha_N \frac{s^Q}{g^\top s^Q}$ where $\alpha_N > 0$ satisfies (11). Hence,

$$\nabla m^C(s^{\mathrm{ARC}}) \quad = \quad g + Bs^{\mathrm{ARC}} + \sigma\|s^{\mathrm{ARC}}\|_M Ms^{\mathrm{ARC}} = g - \frac{\alpha_N}{g^\top s^Q}Bs^Q - \frac{\sigma\alpha_N^2\|s^Q\|_M}{|g^\top s^Q|}\frac{Ms^Q}{g^\top s^Q}.$$

Since $Ms^Q = \frac{\theta}{g^\top s^Q} g$ with $\theta = \|s^Q\|_M^2$, one has

$$\nabla m^C(s^{\mathrm{ARC}}) \quad = \quad g - \frac{\alpha_N}{g^\top s^Q}Bs^Q - \frac{\sigma\alpha_N^2\|s^Q\|_M^3}{|g^\top s^Q|^3}g = \left(1 - \frac{\sigma\alpha_N^2\|s^Q\|_M^3}{|g^\top s^Q|^3}\right)g - \frac{\alpha_N}{g^\top s^Q}Bs^Q.$$

By definition of α_N (see (11)), one has $1 - \frac{\sigma\alpha_N^2\|s^Q\|_M^3}{|g^\top s^Q|^3} = -\frac{\alpha_N}{g^\top s^Q}$, which means

$$\nabla m^C(s^{\mathrm{ARC}}) \quad = \quad -\frac{\alpha_N}{g^\top s^Q}\left(g + Bs^Q\right) = -\frac{\alpha_N}{g^\top s^Q}\nabla m^Q(s^Q).$$

Hence, the condition (7) is being equivalent to

$$\|\nabla m^Q(s^Q)\| \quad \leq \quad \zeta\frac{|g^\top s^Q|}{\alpha_N}\|s\|^2 = \frac{2\,\mathrm{sgn}(g^\top s^Q)\zeta}{-1+\mathrm{sgn}(g^\top s^Q)\sqrt{1+4\frac{\sigma\|s^Q\|_M^3}{|g^\top s^Q|}}}\|s^Q\|^2.$$

■

Remark 4.1 *The use of an exact solver to compute s_k^Q implies that $\|\nabla m_k^Q(s_k^Q)\| = 0$ for all iterations. Thus the condition (23) is automatically satisfied for a such iteration.*

Remark 4.2 *When a subspace method is used to approximate the step s_k^Q, we note the important freedom to add an additional preconditioner to the problem. In this case, one would solve the quadratic problem with preconditioning until the criterion (23) is met. This is expected to happen early along the Krylov iterations when the preconditioner for the linear system $B_k s = -g_k$ is good enough.*

To derive the final algorithm, one needs to address the case where Assumption 3.2 is not satisfied. In fact, in this case, the step s_k^Q either does not exist or could be approximately orthogonal with the gradient g_k. A possible way to overcome this issue can be ensured by modifying the matrix B_k using regularization techniques. By doing so, the global convergence will still hold as well as a function-evaluation complexity bound of order ϵ^{-2} (as no assumption on the matrix B_k is required in this case). To get the $\epsilon^{-3/2}$ complexity bound, in addition to

the criterion (23), one needs the Hessian approximation matrix B_k to satisfy the condition (8). Finding a new Hessian approximation B_k so that the new s_k^Q satisfies Assumption 3.2 is not an easy task.

A practical way to satisfy Assumption 3.2 without modifying B_k can be as follow: using a subspace method to solve the linear system $B_k s = g_k$, let $[s_k^Q]_i$, if it exists, be the first iterate to satisfy (23). Then if $[s_k^Q]_i$ does not satisfy Assumption 3.2, run further iteration of the subspace method until the satisfaction of Assumption 3.2. If the subspace method ends and Assumption 3.2 is still violated, one would restore such assumption by minimizing the cubic model using the ℓ_2-norm until a successful outer iteration is found. The described procedure is detailed in Algorithm 3. We note that in our numerical tests, the restoration procedure was not entered. In fact, for all the tested problems, there was always an iterate $[s_k^Q]_i$ of the subspace method (`MINRES` Matlab solver in our tests) that satisfies both Assumption 3.2 with $\epsilon_d = 10^{-12}$ and the condition (23).

Algorithm 2 details the final algorithm. We note that LS-ARC is nothing but ARC algorithm using a specific M_k-norm. As mentioned earlier, for each outer iteration k in Algorithm 2, the matrix M_k is not used explicitly in the computations. In fact, only the value of the parameter β_k is required in our proposed algorithm.

Algorithm 2: LS-ARC (Main).

Data: select an initial point x_0, the constants $0 < \eta < 1$, $\zeta > 0$ and a small $0 < \epsilon_d < 1$.
Set the initial regularization $\sigma_0 > 0$, and the constants $0 < \nu_1 \leq 1 < \nu_2$.

for $k = 1, 2, \ldots$ **do**

- **Step 1:** compute the gradient g_k and an approximated Hessian B_k satisfying (8);
- **Step 2:** choose a parameter $\beta_k > 0$ uniformly bounded;
- **if** *there exists an approximated step* s_k^Q *satisfying* $|g_k^\top s_k^Q| \geq \epsilon_d \|g_k\| \|s_k^Q\|$ *and (23)* **then**
  - Go to **Step 3**;
- **else**
  - Go to **Algorithm 3** (with $k_r = k$);
- **end**
- **Step 3:** set δ_k and δ_k^{Cauchy} respectively using (17) and (18);
- **repeat**
  - set $\sigma_k \leftarrow \nu_2 \sigma_k$ and update δ_k and δ_k^{Cauchy} respectively using (17) and (18);
  - **if** $\|\nabla m_k^Q(s_k^Q)\| > \zeta |\delta_k| \|s_k^Q\|^2$ **then**
    - Go to **Step 2**;
  - **end**
- **until** *condition (16) will be satisfied*;
- set $s_k^{\text{ARC}} = \delta_k s_k^Q$, $x_{k+1} = x_k + s_k^{\text{ARC}}$ and $\sigma_{k+1} = \nu_1 \sigma_k$;

end

Using the fact that the proposed M_k-norm is uniformly equivalent to the l_2 one along all the iterations, all the convergence results of ARC algorithm [5, 4, 2] apply as well for Algorithm 2. In particular, the LS-ARC algorithm is guaranteed to have an improved function-evaluation worst-case complexity of order $\epsilon^{-3/2}$ to ensure (6). For completeness we state this result in the next theorem.

Algorithm 3: LS-ARC (Restoration with ℓ_2-norm).

Data: Starting from x_{k_r}, σ_{k_r}, g_{k_r}, and B_{k_r} from the main Algorithm (i.e., Algorithm 2) and consider the same parameters as in there.

for $k = k_r, k_r + 1, \ldots$ **do**

  Step 1: compute the step s_k^{ARC} as an approximate solution of (3) with M_k set to the identity matrix, such that $\|\nabla m_k^C(s_k^{\text{ARC}})\| \leq \zeta \|s_k^{\text{ARC}}\|^2$ and $m_k^C(s_k^{\text{ARC}}) \leq m_k^C(s_k^{\text{Cauchy}})$;

  Step 2: if $\rho_k \geq \eta$ **then**

    set $x_{k+1} = x_k + s_k^{\text{ARC}}$ and $\sigma_{k+1} = \nu_1 \sigma_k$;

    Leave Restoration and return to **Step 1** of the Main algorithm (starting at a new (k+1)-th iteration using x_{k+1} and σ_{k+1});

  else

    set $x_{k+1} = x_k$, $g_{k+1} = g_k$, $B_{k+1} = B_k$ and $\sigma_{k+1} = \nu_2 \sigma_k$;

  end

end

Theorem 4.2 *Under regularity assumptions on f (as in [2, 4]), given an $\epsilon > 0$, Algorithm 2 needs at most*

$$\left\lfloor \kappa_S \frac{f(x_0) - f_{low}}{\epsilon^{3/2}} \right\rfloor$$

iterations and at most

$$\left\lfloor \kappa_S \frac{f(x_0) - f_{low}}{\epsilon^{3/2}} \right\rfloor \left(1 + \frac{|\log(\nu_1)|}{\log(\nu_2)}\right) + \frac{1}{\log(\nu_2)} \log\left(\frac{\sigma_0}{\sigma_{\max}}\right)$$

evaluation of f to produce an iterate x_ϵ such that $\|\nabla f(x_\epsilon)\| \leq \epsilon$ where f_{low} is a lower bound on f, $\sigma_{\max}$ and κ_S are positive constants.

Proof. Using Theorem 4.1 and Lemma 4.1, the proof is the same as [2, Theorem 2.5]. ■

5 TR algorithm using a specific M-norm

Similarly to ARC algorithm, it is possible to render TR algorithm behaves as an LS algorithm using the same scaled norm to define the trust-region neighborhood. As a reminder in a basic TR algorithm [8], one computes a trial step s_k^{TR} by approximately solving

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & m_k^Q(s) \\ \text{s. t.} \quad & \|s\|_{M_k} \leq \Delta_k, \end{aligned} \tag{24}$$

where $\Delta_k > 0$ is known as the TR radius. As in ARC algorithms, the scaled norm $\|.\|_{M_k}$ may vary along the iterations and M_k is an SPD matrix.

Once the trial step s_k^{TR} is determined, the objective function is computed at $x_k + s_k^{\text{TR}}$ and compared with the value predicted by the model at this point. If the model value predicts sufficiently well the objective function (i.e., the iteration is *successful*), the trial point $x_k + s_k^{\text{TR}}$ will be accepted and the TR radius is eventually expanded (i.e., $\Delta_{k+1} = \tau_2 \Delta_k$ with $\tau_2 \geq 1$). If

the model turns out to predict poorly the objective function (i.e., the iteration is *unsuccessful*), the trial point is rejected and the TR radius is contracted (i.e., $\Delta_{k+1} = \tau_1 \Delta_k$ with $\tau_1 < 1$). The ratio between the actual reduction and the predicted reduction for the TR algorithms is defined as in ARC (see (4)). For a given scalar $0 < \eta < 1$, the iteration will be said *successful* if $\rho \geq \eta$, and *unsuccessful* otherwise. Algorithm 4 gives a detailed description of a basic TR algorithm.

Algorithm 4: TR algorithm.

Data: select an initial point x_0 and $0 < \eta < 1$. Set the initial TR radius $\Delta_0 > 0$ and the constants $0 \leq \tau_1 < 1 \leq \tau_2$.

for $k = 1, 2, \ldots$ **do**

Step 1: compute the gradient g_k and an approximated Hessian B_k;

Step 2: compute the step s_k^{TR} as an approximate solution of (24) such that

$$m_k^Q(s_k^{\mathrm{TR}}) \quad \leq \quad m_k^C(s_k^{\mathrm{TR,Cauchy}}) \tag{25}$$

where $s_k^{\mathrm{TR,Cauchy}} = -\delta_k^{\mathrm{TR,Cauchy}} g_k$ and $\delta_k^{\mathrm{TR,Cauchy}} = \arg \min_{0<t\leq \frac{\Delta_k}{\|g_k\|_{M_k}}} m_k^Q(-t g_k)$;

Step 3: if $\rho_k \geq \eta$ **then**

set $x_{k+1} = x_k + s_k^{\mathrm{TR}}$ and $\Delta_{k+1} = \tau_2 \Delta_k$

else

set $x_{k+1} = x_k$ and $\Delta_{k+1} = \tau_1 \Delta_k$;

end

end

We will focus on the TR subproblem minimization for a given iteration k. The outer iteration subscript k will be dropped to keep the notations simple. As for ARC algorithm, we will call the Newton-like step associated with the TR subproblem the vector of the following form

$$s_N^{\mathrm{TR}} = \alpha_N d^{\mathrm{N}}, \quad \text{where} \quad \alpha_N = \arg \min_{t>0; \|t d^{\mathrm{N}}\|_M \leq \Delta} m^Q(t d^{\mathrm{N}}). \tag{26}$$

Note that for the TR subproblem, the solution we are looking for lies either interior to the trust region, that is $\|s\|_M < \Delta$, or on the boundary, $\|s\|_M = \Delta$. If the solution is interior, the solution s^{TR} is the unconstrained minimizer of the quadratic model m^Q. Such scenario can only happen if m^Q is convex. In the non convex case a solution lies on the boundary of the trust region, while in the convex case a solution may or may not do so. Consequently in practice, the TR algorithm finds first the unconstrained minimizer of the model m^Q. If the model is unbounded from below, or if the unconstrained minimizer lies outside the trust region, the minimizer then occurs on the boundary of the trust region. Similarly to ARC algorithm, one has the following results:

Theorem 5.1 *Let Assumption 3.1 holds.*

1. *The Newton-like step (26) is of the following form:*

$$s_N^{\mathrm{TR}} \quad = \quad \delta_N^{\mathrm{TR}} s^Q, \quad \textit{where} \quad \delta_N^{\mathrm{TR}} = \min\left(1, -\operatorname{sgn}(g^\top s^Q) \frac{\Delta}{\|s^Q\|_M}\right). \tag{27}$$

2. *When it lies on the border of the trust region $s_N^{\rm TR}$ is a stationary point of the subproblem (24) if and only if $Ms^Q = \frac{\theta}{g^\top s^Q} g$ where $\theta = \|s^Q\|_M^2$.*

3. *Moreover, when $s_N^{\rm TR}$ lies on the border, if the matrix $B+\lambda_N^{\rm TR} M$ is positive definite, then $s_N^{\rm TR}$ is the unique minimizer of the subproblem (24) where $\lambda_N^{\rm TR} = \frac{g^\top s^Q}{\theta}\left(1 + \mathrm{sgn}(g^\top s^Q)\frac{\|s^Q\|_M}{\Delta}\right)$.*

Proof.

1. To calculate the Newton step $s_N^{\rm TR}$, we first note, for all $t > 0$

$$m_Q(td^{\rm N}) \;=\; m^Q(0) - t - \frac{1}{g^\top s^Q}\frac{t^2}{2}. \tag{28}$$

Consider the case where the curvature model along the Newton direction is positive, that is when $g^\top s^Q < 0$, and compute the value of the parameter t at which the unique minimizer of (28) is attained. Let t^* denotes this optimal parameter. Taking the derivative of (28) with respect to t and equating the result to zero, one has $t^* = -g^\top s^Q$. Two sub-cases may then occur. The first is when this minimizer lies within the trust region (i.e. $t^*\|d^{\rm N}\|_M \leq \Delta$), then

$$\alpha_N \;=\; t^* = -g^\top s^Q.$$

If on the other hand, $t^*\|d^{\rm N}\|_M > \Delta$, then the line minimizer is outside the trust region and we have that

$$\alpha_N \;=\; \frac{\Delta}{\|d^{\rm N}\|_M} = -g^\top s^Q \frac{\Delta}{\|s^Q\|_M}.$$

Finally, we consider the case where the curvature of the model along the Newton-like step is negative, that is, when $g^\top s^Q > 0$. In that case, the minimizer lies on the boundary of the trust region, and thus

$$\alpha_N \;=\; \frac{\Delta}{\|d^{\rm N}\|_M} = g^\top s^Q \frac{\Delta}{\|s^Q\|_M}.$$

By combining all cases, one concludes that

$$s_N^{\rm TR} \;=\; \delta_N^{\rm TR} s^Q, \quad \text{where} \quad \delta_N^{\rm TR} = \min\left(1, -\mathrm{sgn}(g^\top s^Q)\frac{\Delta}{\|s^Q\|_M}\right).$$

2. Suppose that the Newton-like step lies on the border of the trust region i.e., $s_N^{\rm TR} = \delta_N^{\rm TR} s^Q = -\mathrm{sgn}(g^\top s^Q)\frac{\Delta}{\|s^Q\|_M} s^Q$. The latter step is a stationary point of the subproblem (24) if and only if there exists a Lagrange multiplier $\lambda \geq 0$ such that

$$(B + \lambda M) s_N^{\rm TR} \;=\; -g.$$

Substituting $s_N^{\rm TR} = \delta_N^{\rm TR} s^Q$ in the latter equation, one has

$$\lambda M s^Q \;=\; \left(1 - \frac{1}{\delta_N^{\rm TR}}\right) g. \tag{29}$$

By multiplying it from left by $(s^Q)^\top$, we deduce that

$$\lambda \;=\; \left(1 - \frac{1}{\delta_N^{\rm TR}}\right)\frac{g^\top s^Q}{\|s^Q\|_M^2} = \frac{g^\top s^Q}{\theta}\left(1 + \mathrm{sgn}(g^\top s^Q)\frac{\|s^Q\|_M}{\Delta}\right).$$

By replacing the value of λ in (29), we obtain that $Ms^Q = \frac{\theta}{g^\top s^Q} g$ where $\theta = \|s^Q\|_M^2 > 0$.

3. Indeed, If the step s_N^{TR} lies on the boundary of the trust-region (i.e.$\|s_N^{\mathrm{TR}}\|_M = \Delta$), One has $Ms^Q = \frac{\theta}{g^\top s^Q} g$. Then applying (2) of Theorem 5.1, we see that

$$(B + \lambda_N^{\mathrm{TR}} M) s_N^{\mathrm{TR}} = -g$$

with $\lambda_N^{\mathrm{TR}} = \frac{g^\top s^Q}{\theta} \left(1 + \mathrm{sgn}(g^\top s^Q) \frac{\|s^Q\|_M}{\Delta}\right) > 0$. Applying [8, Theorem 7.4.1], we see that if we assume that the matrix $B + \lambda_N^{\mathrm{TR}} M$ is positive definite, then s_N^{TR} is the unique minimizer of the subproblem (24). ■

Given an SPD matrix M that satisfies the secant equation $Ms^Q = \frac{\theta}{g^\top s^Q} g$, item (3) of Theorem 5.1 states that the step s_N^{TR} is the global minimizer of the associated subproblems as far as the matrix $B + \lambda_N^{\mathrm{TR}} M$ is SPD. We note that λ_N^{TR} goes to infinity as the trust region radius Δ goes to zero, meanning that the matrix $B + \lambda_N^{\mathrm{TR}} M$ will be SPD as far as Δ is chosen to be sufficiently small. Since the TR update mechanism allows to shrink the value of Δ (when the iteration is declared as unsuccessful), satisfying the targeted condition will be geared automatically by the TR algorithm.

In the case where the linear system $Bs = -g$ is solved iteratively (by projection onto a sequence embedded subspaces). Then using similar arguments as for ARC algorithm (as far as Assumption 3.2 holds), one can extend the results in Theorem 5.1 when a truncated Newton-like step is used in the TR algorithm.

Again, when Assumption 3.2 hold, we note that *unsuccessful* iterations in the TR Algorithm require only updating the value of the TR radius Δ and the current step direction is kept unchanged. For such iterations, as far as there exists a matrix M satisfies such as $Ms^Q = \frac{\theta}{g^\top s^Q} g$, the approximate solution of the TR subproblem is obtained only by updating the step-size δ. This means that the computational cost of unsuccessful iterations do not requires solving any extra subproblem. As a consequence, in this setting TR behaves as an LS method with a specific backtracking strategy. In fact, at the k^{th} iteration, the trial step is of the form $s_k = \delta_k s_k^Q$ where s_k^Q is the (approximate) solution of the linear system $B_k s = -g_k$. The step length $\delta_k > 0$ is chosen such as

$$f(x_k + s_k) \leq f(x_k) + \eta \left(s_k^\top g_k + \frac{1}{2} s_k^\top B_k s_k\right) \quad \text{and} \quad m_k^Q(s_k) \leq m_k^Q(-\delta_k^{\mathrm{TR,Cauchy}} g_k). \tag{30}$$

The values of δ_k and $\delta_k^{\mathrm{TR,Cauchy}}$ are computed respectively as follows:

$$\delta_k \;=\; \min\left(1, -\mathrm{sgn}(g_k^\top s_k^Q) \frac{\Delta_k}{\|s_k^Q\|_{M_k}}\right), \tag{31}$$

$$\delta_k^{\mathrm{TR,Cauchy}} = \begin{cases} \frac{\Delta_k}{\|g_k\|_{M_k}} & \text{if } g_k^\top B_k g_k \leq 0 \text{ or } \frac{g_k^\top B_k g_k}{\|g_k\|^2} \geq \frac{\Delta_k}{\|g_k\|_{M_k}}, \\ \frac{g_k^\top B_k g_k}{\|g_k\|^2} & \text{else.} \end{cases} \tag{32}$$

where Δ_k is initially equals to the current value of the TR radius (as in the original TR algorithm). For large values of δ_k the sufficient decrease condition (30) may not be satisfied, in this case, the value of Δ_k is contracted using a contraction factor τ_1. Iteratively, the value of δ_k is

Algorithm 5: The LS-TR algorithm.

Data: select an initial point x_0 and $0 < \eta < 1$ and a small $0 < \epsilon_d \leq 1$. Set the initial TR radius $\Delta_0 > 0$ and the constants $0 \leq \tau_1 < 1 \leq \tau_2$.

for $k = 1, 2, \ldots$ **do**

Step 1: compute the gradient g_k and an approximated Hessian B_k;

Step 2: choose a parameter $\beta_k > 0$ uniformly bounded and an approximate stationary point s_k^Q of m_k^Q satisfying $|g_k^\top s_k^Q| \geq \epsilon_d \|g_k\| \|s_k^Q\|$;

Step 3: set δ_k and $\delta_k^{\text{TR,Cauchy}}$ using (27) and (32);

repeat

set $\Delta_k \leftarrow \tau_1 \Delta_k$, and update δ_k and $\delta_k^{\text{TR,Cauchy}}$ using (27) and (32);

until *condition (30) will be satisfied*;

set $s_k = \delta_k s_k^Q$, $x_{k+1} = x_k + s_k$ and $\Delta_{k+1} = \tau_2 \Delta_k$;

end

updated and the acceptance condition (30) is checked again until its satisfaction. Algorithm 5 details the adaptation of the classical TR algorithm when our proposed M_k-norm is used. We denote the modified TR algorithm by LS-TR (Algorithm 5).

As LS-ARC algorithm, for each outer iteration k in Algorithm 5, the matrix M_k is not used explicitly in the computations but only through the value of $\theta_k > 0$. To extend the same convergence and complexity results of the classical TR algorithm, the choice of the parameter β_k has to satisfy the requirement that makes the proposed M_k-norm uniformly equivalent to the l_2 one along the iterations (see Theorem 4.1).

Note that using a subspace method, for each iteration, there exist at least one iterate that satisfies Assumption 3.2 (since the Cauchy step can be regarded as the first iterate of subspace method applied to solve $B_k s = -g_k$). Thus in the frame of LS-TR method, satisfying Assumption 3.2 is not an issue.

6 Numerical experiments

In this section we report the results of experiments performed in order to assess the efficiency and the robustness of the proposed algorithms (`LS-ARC` and `LS-TR`) compared with the classical LS algorithm using the standard Armijo rule. In the latter approach, the trial step is of the form $s_k = \delta_k d_k$ where $d_k = s_k^Q$ if $-g_k^\top s_k^Q \geq \epsilon_d \|g_k\| \|s_k^Q\|$ (s_k^Q is being an approximate stationary point of m_k^Q) otherwise $d_k = -g_k$, and the step length $\delta_k > 0$ is chosen such as

$$f(x_k + s_k) \leq f(x_k) + \eta s_k^\top g_k, \tag{33}$$

where $\eta \in (0, 1)$. The appropriate value of δ_k is estimated using a backtracking approach with a contraction factor set to $\tau \in (0, 1)$ and where the step length is initially chosen to be 1. This LS method will be called `LS-ARMIJO`. We implement all the the algorithms as Matlab m-files and for all the tested algorithms B_k is set to the true Hessian $\nabla^2 f(x_k)$, and $\epsilon_d = 10^{-12}$ (for `LS-ARC` and `LS-ARMIJO`).

By way of comparison, we have also implemented the standard ARC and TR algorithms (see Algorithms 1 and 4) using the Lanczos-based solver `GLTR`/`GLRT` implemented in `GALAHAD`

[12].The two subproblem solvers, `GLTR/GLRT` are implemented in Fortran and interfaced with Matlab using the default parameters. For the subproblem formulation we used the ℓ_2-norm (i.e., for all iterations the matrix M_k is set to identity). We shall refer to the ARC/TR methods based on `GLRT/GLTR` as `GLRT-ARC/GLTR-TR`.

The other parameters defining the implemented algorithms are set as follows, for `GLRT-ARC` and `LS-ARC`

$$\eta = 0.1,\ \nu_1 = 0.5,\ \nu_2 = 2,\ \text{and}\ \sigma_0 = 1;$$

for `GLTR-TR` and `LS-TR`

$$\eta = 0.1,\ \tau_1 = 0.5,\ \tau_2 = 2,\ \text{and}\ \Delta_0 = 1;$$

and last for `LS-ARMIJO`

$$\eta = 0.1,\ \text{and}\ \ \tau = 0.5.$$

In all algorithms the maximum number of iterations is set to 10000 and the algorithms stop when

$$\|g_k\| \leq 10^{-5}.$$

A crucial ingredient in `LS-ARC` and `LS-TR` is the management of the parameter θ_k. A possible choice for θ_k is $|g_k^\top s_k^Q|$. This choice is inspired from the fact that, when the Hessian matrix B_k is SPD, this update corresponds to use the energy nom (i.e., set the matrix M_k equals to B_k) (see [1] for more details). However, this choice did not lead to good performance of the algorithms `LS-ARC` and `LS-TR`. In our implementation, for the `LS-ARC` algorithm, we set the value of θ_k as follows:

$$\theta_k = \beta_k \|s_k^Q\|^2,$$

where $\beta_k = \kappa\sigma_k^{-2/3}$ and $\kappa = 10^{-4}$ if $g_k^\top s_k^Q < 0$ and 2 otherwise. By this choice, we are willing to allow `LS-ARC`, at the start of the backtracking procedure, to take a step that is (approximately) the Newton step. Similarly, for the `LS-TR` method, we set $\theta_k = \|s_k^Q\|^2$ in fact as such value is of the same order as Δ_k^2 this allows `LS-TR` to use the Newton step at the start of the backtracking strategy (as in `LS-ARMIJO` method). Within this choice of θ_k, we ensure that the M_k-norm will be uniformly equivalent to the ℓ_2-norm for all iterations (see Theorem 4.1).

All the Algorithms are evaluated on a set of unconstrained optimization problems from the CUTEst collection [13]. The test set contains 62 large-scale ($1000 \leq n \leq 10000$) CUTest test problem with their default parameters. Regarding the algorithms `LS-TR`, `LS-ARC`, and `LS-ARMIJO`, we approximate the solution of the linear system $B_k s = -g_k$ using the `MINRES` Matlab solver. The latter method is a Krylov subspace method designed to solve symmetric linear systems [18]. We run the algorithms with the `MINRES` default parameters except the relative tolerance error which is set to 10^{-4}. We note that on the tested problems, `LS-ARC` satisfied Assumption 3.2 (with $\epsilon_d = 10^{-12}$) for all outer iterations, thus the restoration procedure of such assumption using the ℓ_2-norm was not activated.

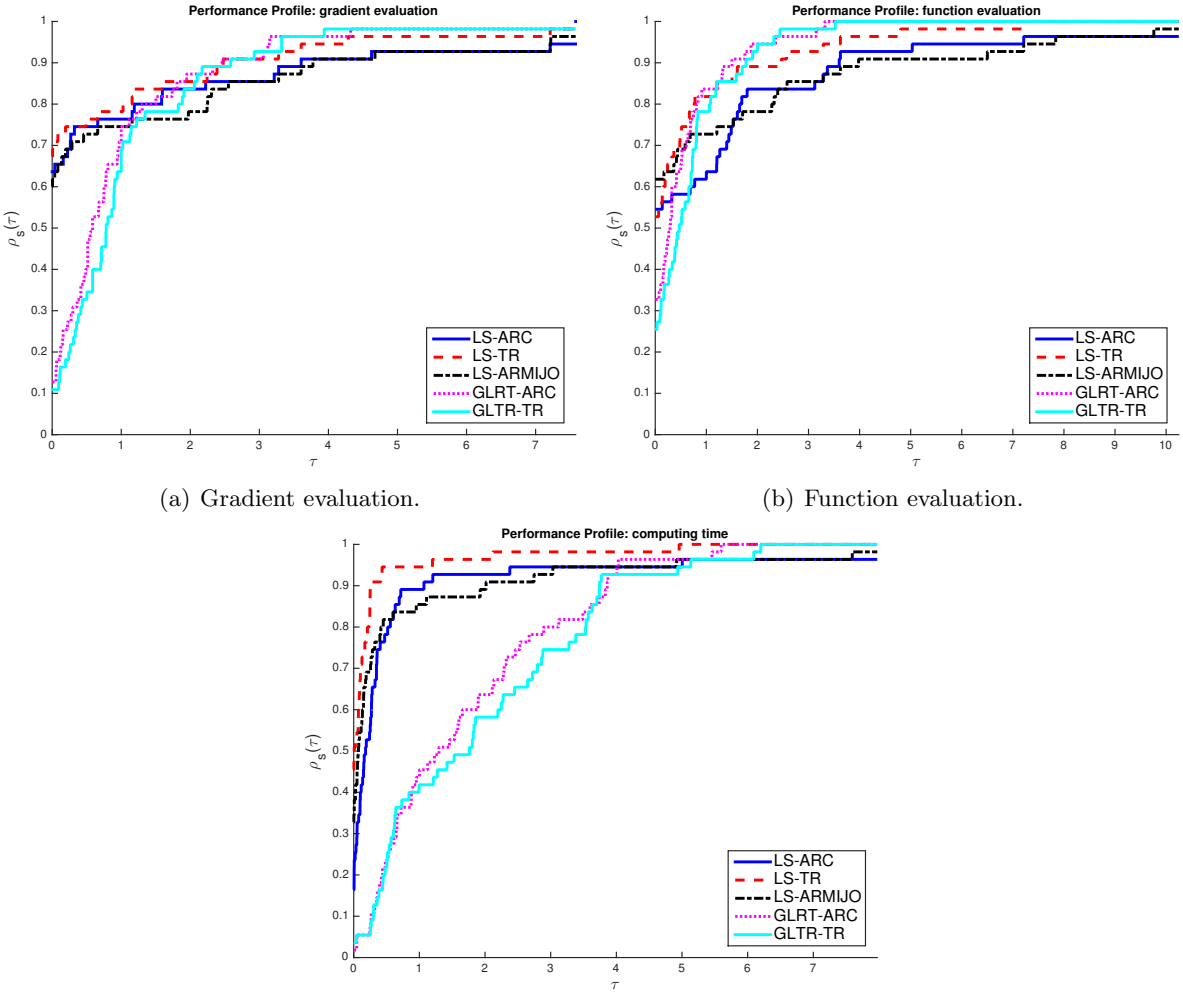
To compare the performance of the algorithms we use performance profiles proposed by Dolan and Moré [11] over a variety of problems. Given a set of problems $\mathcal{P}$ (of cardinality $|\mathcal{P}|$) and a set of solvers $\mathcal{S}$, the performance profile $\rho_s(\tau)$ of a solver s is defined as the fraction of problems where the performance ratio $r_{p,s}$ is at most τ

$$\rho_s(\tau) \ = \ \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}.$$

The performance ratio $r_{p,s}$ is in turn defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}},$$

where $t_{p,s} > 0$ measures the performance of the solver s when solving problem p (seen here as the function evaluation, the gradient evaluation, and the CPU time). Better performance of the solver s, relatively to the other solvers on the set of problems, is indicated by higher values of $\rho_s(\tau)$. In particular, efficiency is measured by $\rho_s(1)$ (the fraction of problems for which solver s performs the best) and robustness is measured by $\rho_s(\tau)$ for τ sufficiently large (the fraction of problems solved by s). Following what is suggested in [11] for a better visualization, we will plot the performance profiles in a $\log_2$-scale (for which $\tau = 1$ will correspond to $\tau = 0$).



(a) Gradient evaluation.

(b) Function evaluation.

(c) CPU time.

Figure 1: Performance profiles for 62 large scale optimization problems (i.e., $1000 \leq n \leq 10000$).

We present the obtained performance profiles in Figure 1. Regarding the gradient evaluation (i.e., outer iteration) performance profile, see Figure 1(a), LS approaches are the most efficient among all the tested solvers (in more 60% of the tested problems LS methods perform the best,

while GLRT-ARC and GLTR-TR are performing better only in less than 15%). When it comes to robustness, all the tested approaches exhibit good performance (GLRT-ARC and GLTR-TR are slightly better).

For function evaluation performance profile (Figure 1(b)), GLRT-ARC, GLTR-TR show a better efficiency but not as good as LS methods (in more than 50% of the tested problems LS methods performs the best while GLRT-ARC, GLTR-TR are better only in less than 35%). The robustness of the tested algorithms is the same as in the outer iteration performance profile.

In terms of the demanded computing time, see Figure 1(c), as one can expect, GLRT-ARC and GLTR-TR are turned to be very consuming compared to the LS approaches. In fact, unlike the LS methods where only an approximate solution of one linear system is needed, the GLRT/GLTR approaches may require (approximately) solving multiple linear systems in sequence.

For the LS approaches, one can see that LS-TR displays better performance compared to LS-ARMIJO on the tested problems. The main difference between the two LS algorithms is the strategy of choosing the search direction whenever $g_k^\top s_k^Q > 0$. In our tested problems, the obtained performance using LS-TR suggests that going exactly in the opposite direction $-s_k^Q$ (whenever s_k^Q is not a descent direction) can be seen as a good strategy compared to LS-ARMIJO.

7 Conclusion

In this paper, we have investigated the use of a specific scaled norm to define the cubic regularization term in the ARC subproblem. With this norm choice, we have shown that the trial step of ARC is getting collinear to the Newton like step. The obtained ARC algorithm behaves as an LS approach with a specific backtracking strategy. Under mild assumptions, the proposed scaled norm was shown to be uniformly equivalent to the Euclidean norm, in this case, the obtained LS algorithm enjoys the same convergence and complexity properties as ARC. Similarly to ARC algorithm and using the same scaled norm to define the TR neighborhood, the proposed approach was generalized to cover the TR framework as well. Unlike classical LS methods based on Newton direction (wherever the latter is not a descent direction, one generally proceeds by taking $-g_k$ or modifying the diagonal of B_k) our proposed LS algorithms proceed line search only along the Newton direction independently from the convexity of the quadratic approximation of f. Our numerical experiments showed encouraging performance of the proposed LS algorithms.

A number of issues need further investigation, in particular the best choice and the impact of the parameter θ_k on the performance of the proposed LS approaches. Also, the analysis in this paper suggests that taking the Newton direction is suitable for defining a line-search method with an optimal worst case complexity bound of order $\epsilon^{-3/2}$ to derive the gradient norm below ϵ. It would be interesting to confirm the potential of the proposed line search strategy compared to the classical LS approaches using extensive numerical tests.

References

[1] E. Bergou, Y. Diouane, and S. Gratton. On the use of the energy norm in trust-region and adaptive cubic regularization subproblems. *Comput. Optim. Appl.*, 68(3):533–554, 2017.

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

[3] E. G. Birgin and J. M. Martínez. The use of quadratic regularization with a cubic descent condition for unconstrained optimization. *SIAM J. Optim.*, 27(2):1049–1074, 2017.

[4] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic overestimation methods for unconstrained optimization. Part II: worst-case function- and derivative-evaluation complexity. *Math. Program.*, 130(2):295–319, 2011.

[5] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: Motivation, convergence and numerical results. *Math. Program.*, 127(2):245–295, 2011.

[6] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Optimality of orders one to three and beyond: characterization and evaluation complexity in constrained nonconvex optimization. Technical report, 2017.

[7] C. Cartis, Ph. R. Sampaio, and Ph. L. Toint. Worst-case evaluation complexity of non-monotone gradient-related algorithms for unconstrained optimization. *Optimization*, 64(5):1349–1361, 2015.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, PA, USA, 2000.

[9] F. E. Curtis, D. P. Robinson, and M. Samadi. A trust region algorithm with a worst-case iteration complexity of $O(\epsilon^{-3/2})$ for nonconvex optimization. *Math. Program.*, 162(1):1–32, 2017.

[10] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall Inc, Englewood Cliffs, NJ, 1983.

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

[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 Trans. Math. Softw.*, 29(4):353–372, 2003.

[13] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60(3):545–557, 2015.

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

[15] J. M. Martínez and M. Raydan. Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization. *J. Global Optim.*, 68(2):367–385, 2017.

[16] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic Publishers, 2004.

[17] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton's method and its global performance. *Math. Program.*, 108(1):177–205, 2006.

[18] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[19] Y. Yuan. Recent advances in trust region algorithms. *Math. Program.*, 151(1):249–281, 2015.