Cahier du GERAD G-2015-109

SCALABLE ADAPTIVE CUBIC REGULARIZATION METHODS

JEAN-PIERRE DUSSAULT AND DOMINIQUE ORBAN

Abstract. Adaptive cubic regularization (ARC) methods for unconstrained optimization compute steps from linear systems involving a shifted Hessian in the spirit of the Levenberg-Marquardt and trust-region methods. The standard approach consists in performing an iterative search for the shift akin to solving the secular equation in trust-region methods. Such search requires computing the Cholesky factorization of a tentative shifted Hessian at each iteration, which limits the size of problems that can be reasonably considered. We propose a scalable implementation of ARC named ARC$_q$K in which we solve a set of shifted systems concurrently by way of an appropriate modification of the Lanczos formulation of the conjugate gradient (CG) method. At each iteration of ARC$_q$K to solve a problem with n variables, a range of $m \ll n$ shift parameters is selected. The computational overhead in CG beyond the Lanczos process is thirteen scalar operations to update five vectors of length m and two n-vector updates for each value of the shift. The CG variant only requires one Hessian-vector product and one dot product per iteration, independently of the number of shift parameters. Solves corresponding to inadequate shift parameters are interrupted early. All shifted systems are solved inexactly. Such modest cost makes our implementation scalable and appropriate for large-scale problems. We provide a new analysis of the inexact ARC method including its worst case evaluation complexity, global and asymptotic convergence. We describe our implementation and provide preliminary numerical observations that confirm that for problems of size at least 100, our implementation of ARCqK is more efficient than a classic Steihaug-Toint trust region method. Finally, we generalize our convergence results to inexact Hessians and nonlinear least-squares problems.

1. Introduction

We consider the unconstrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x) \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ and $\nabla^2 f$ is Lipschitz continuous.

Adaptive Cubic Regularization (ARC) algorithms, recently explored by Cartis et al. [1, 2] are closely related to trust region (TR) methods [5] in that steps are computed by solving a sequence of regularized subproblems. A major theoretical appeal of ARC over TR methods is their optimal worst-case complexity property. Whereas the number of function evaluations required to reach a point x for which

2010 *Mathematics Subject Classification.* 65F10, 65F22, 65F25, 65F35, 65F50, 90C06, 90C20, 90C30 .

Key words and phrases. Unconstrained optimization, trust-region algorithms, adaptive cubic regularization.

Research partially supported by an NSERC Discovery Grant.

Research partially supported by an NSERC Discovery Grant.

$\|\nabla f(x)\| \leq \epsilon$ is $O(\epsilon^{-2})$ for TR, which is no better than steepest descent [3], that number is $O(\epsilon^{-3/2})$ for ARC. Dussault [7] develops the ARC$_q$ variant, which uses the usual quadratic model but computes regularized Newton steps satisfying the cubic subproblem's optimality conditions, and obtains simple proofs highlighting the key properties that ensure optimal worst-case complexity.

Both ARC$_q$ and TR algorithms employ the quadratic model

$$q_x(d) = f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T \nabla^2 f(x) d. \tag{1.2}$$

At each iteration, ARC$_q$ minimizes a cubic model [15]

$$c_x^\alpha(d) := q_x(d) + \tfrac{1}{3}\alpha^{-1}\|d\|^3, \tag{1.3}$$

where $\alpha > 0$ plays a role similar to the trust-region radius in TR methods. As in TR methods, minimizing (1.3) involves solving the shifted linear system

$$(\nabla^2 f(x) + \lambda I)d = -\nabla f(x), \tag{1.4}$$

while searching an appropriate value of the shift $\lambda > 0$. Cartis et al. [1] use GLTR [12], which builds upon the truncated conjugate gradient approach of Steihaug [24] and Toint [25] and further explores the boundary of the trust region if the latter is determined to be active. One disadvantage of GLTR resides in its storage and computational requirements as Lanczos vectors must be either stored or re-generated to explore the trust-region boundary. This additional cost, which is incurred from the very first iteration, makes the approach unsuitable for large-scale problems. In addition, numerical experiments reveal that the method of Steihaug [24] and Toint [25] yields an adequate solution most of the time, so that further explorations of the boundary of the trust region may not pay off, as already remarked by the authors of GLTR.

In this paper, we propose ARC$_q$K, an implementation of ARC$_q$ that obtains an approximate minimizer of (1.3) using CG-Lanczos with shifts, a Lanczos implementation of the conjugate gradient algorithm that solves several shifted systems simultaneously proposed by Frommer and Maass [10]. At each iteration of ARC$_q$K, a range of $m \ll n$ shift parameters is selected. The CG variant solves the shifted systems concurrently and interrupts those corresponding to shifts that are too small as soon as negative curvature is detected. The computational overhead beyond the Lanczos process is thirteen scalar operations to update five vectors of length m and two n-vector updates for each value of the shift.

Of course, the overhead of using some 31 shift will not be easily overcomed for small scaled instances. Our results, summarized in performance profile graphs, confirms the efficiency of our approach for problems of dimensions above 100 in the Cutest collection when compared to a Steihaug-Toint approach.

The rest of this paper is organized as follows. We focus the presentation on the worst case function evaluation complexity to achieve first order optimality conditions. We recall the ARC$_q$ algorithm and its worst case evaluation complexity analysis in §2. We introduce ARC$_q$K and analyze its worst case complexity. We then introduce the CG-Lanczos method to solve the shifted systems and analyze its computational complexity. We report on numerical experience on problems from the CUTEst collection [14].

Notation. When the context is clear, we simply write $q(d)$ and $c^\alpha(d)$ instead of $q_x(d)$ and $c_x^\alpha(d)$. In an iterative procedure, we write $q_k(d)$ and $c_k(d)$ instead of

$q_{x_k}(d)$ and $c^{\alpha_k}_{x_k}(d)$. Throughout the text, we use $\|\cdot\|$ to denote the Euclidean norm. For future reference, the gradient of (1.3) is

$$\nabla c^{\alpha}_{x}(d) = \nabla f(x) + \nabla^2 f(x)d + \alpha^{-1}\|d\|d. \tag{1.5}$$

If A is a square symmetric matrix, we write $A \succ 0$ to mean that A is positive definite. If $\{\alpha_k\}$ and $\{\beta_k\}$ are real positive sequences converging to zero, we use the Landau notation $\alpha_k = O(\beta_k)$, $\alpha_k = \Omega(\beta_k)$ and $\alpha_k = \Theta(\beta_k)$ to mean that there exist a constant $C > 0$ and an iteration $k_0 \in \mathbb{N}$ such that $\alpha_k \leq C\beta_k$, $\alpha_k \geq C^{-1}\beta_k$, and $C^{-1}\beta_k \leq \alpha_k \leq C\beta_k$ for all $k \geq k_0$, respectively.

2. The ARC$_q$ algorithm

The basic algorithm, described as Algorithm 2.1, is similar to the basic TR method; ARC$_q$ uses the cubic regularized model to compute the direction d but the quadratic model in the algorithm flow.

The following result characterizes global minimizers of $c_k(d)$.

Theorem 2.1 (1, Theorem 3.1)**.** *Any d_k is a global minimizer of $c_k(d)$ if and only if*

$$\nabla f(x_k) + (\nabla^2 f(x_k) + \lambda_k I)d_k = 0, \tag{2.1a}$$

$$\nabla^2 f(x_k) + \lambda_k I \succeq 0, \tag{2.1b}$$

$$\lambda_k = \|d_k\|/\alpha_k. \tag{2.1c}$$

If $\nabla^2 f(x_k) + \lambda_k I \succ 0$, then d_k is unique.

Theorem 2.1 suggests a computational procedure similar to the Moré and Sorensen [19] approach in trust-region methods. Dussault [7] improves the procedure by using changes of variables to reduce the number of factorizations to one per successful iteration. Algorithm 2.1 summarizes the ARC$_q$ framework.

Algorithm 2.1 ARC$_q$ Framework.

Initialize $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $0 < \eta_1 < \eta_2 < 1$, and $0 < \gamma_1 < 1 < \gamma_2$, $k = 0$
repeat
  compute d_k as a global minimizer of c_k
  compute $\rho_k = \dfrac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}$
  if $\rho_k < \eta_1$ **then**
    $\alpha_{k+1} = \gamma_1 \alpha_k$ *Unsuccessful iteration*
  else
    $x_{k+1} = x_k + d_k$ *Successful iteration*
    if $\rho_k > \eta_2$ **then**
      $\alpha_{k+1} = \gamma_2 \alpha_k$ *Very successful iteration*
    else
      $\alpha_{k+1} = \alpha_k$
    end if
  end if
  $k \leftarrow k + 1$
until termination_criterion

Algorithm 2.1 differs from the original ARC of Cartis et al. [1] only in the use of the quadratic model q_k to compute the ratio ρ instead of the cubic model c_k. The convergence and complexity results are identical and rely on the following assumptions.

Assumption 2.1. *f is twice continuously differentiable.*

Assumption 2.2. *There exists $L > 0$ such that $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|$ for all x, $y \in \mathbb{R}^n$.*

Assumption 2.3. *There exists a value f_{low} such that $f(x_k) \geq f_{low}$ for all k.*

Assumption 2.4. *d is computed as a global minimizer of c_x^α.*

Assumption 2.5. *There exists $\alpha_{\max} > 0$ such that $\alpha_k \leq \alpha_{\max}$ for all k.*

Under the above assumptions, Dussault [7] establishes the following global convergence result.

Theorem 2.2 (7, Corollary 3.3)**.** *Let $\{x_k\}$ be generated by Algorithm 2.1 and let Assumptions 2.1 to 2.5 be satisfied. Any cluster point of $\{x_k\}$ satisfies the second-order necessary optimality conditions.*

A further property related to Newton's method applied to (2.1a)–(2.1c) yields the following complexity result.

Theorem 2.3 (7, Theorem 4.2)**.** *Under Assumptions 2.1 to 2.5, the number of iterations required by Algorithm 2.1 to generate $\bar{x}$ such that $\|\nabla f(\bar{x})\| < \epsilon$ is at most $O(\epsilon^{-3/2})$.*

Theorem 2.2 and Theorem 2.3 rely on exact minimization of the cubic model (1.3). The analysis involving global minimizers of (1.3) ensures that $\Delta q_k(d_k) := q_k(d_k) - q_k(0) \geq \frac{1}{2}\lambda_k\|d_k\|^2$ and that $\|\nabla f(x_{k+1})\| \leq \frac{1}{2}L\|d_k\|^2 + \lambda_k\|d_k\|$. The proof of Theorem 2.3 relies on the fact that both $\lambda_k = \Omega(\|d_k\|)$ and $\lambda_k = O(\|d_k\|)$ hold if α_k is bounded away from zero and d_k is computed as a global minimizer of (1.3), for in that case $\lambda_k = \|d_k\|/\alpha_k$.

In large-scale applications, we must instead consider approximate solutions to (1.3), the analysis of which is the subject of the next section.

2.1. **Approximate model solution.** In this section, we show that the complexity bound of Theorem 2.3 continues to hold for approximate minimizers if we can guarantee that α_k is bounded away from zero and $\lambda_k = \Theta(\|d_k\|)$. Lemma 2.2 establishes the first property. In Section 3, we propose a way to compute λ_k and d_k so that $\lambda_k = \Theta(\|d_k\|)$ as well as $d_k^T(\nabla^2 f(x) + \lambda_k I)d_k \geq 0$.

For the time being, we require λ_k and d_k to satisfy the following approximation to (2.1a)–(2.1c):

$$\nabla f(x_k) + (\nabla^2 f(x_k) + \lambda_k I)d_k = r_k, \tag{2.2a}$$

$$d_k^T(\nabla^2 f(x_k) + \lambda_k I)d_k \geq 0 \tag{2.2b}$$

$$\frac{1}{\beta}\frac{\|d_k\|}{\alpha_k} \leq \lambda_k \leq \beta\frac{\|d_k\|}{\alpha_k}, \tag{2.2c}$$

where r_k is a residual and $\beta \geq 1$ is a sampling parameter.

We modify Assumption 2.4 as follows.

Assumption 2.4b. *At line 3 of Algorithm 2.1, we compute a direction d satisfying (2.2a)–(2.2c).*

In Section 2.2, we state appropriate conditions on $\|r\|$ that ensure convergence and maintain the complexity bound of Theorem 2.3.

2.2. **Convergence analysis and complexity.** We are now able to generalize the basic lemmas from Dussault [7] to approximate minimizers of (1.3).

Lemma 2.1. *Let Assumptions 2.1 and 2.4b be satisfied and assume that $r_k^T d_k = 0$. Then,*

$$\Delta q_k(d_k) = q_k(0) - q_k(d_k) \geq \tfrac{1}{2} \|d_k\|^2 \lambda_k \geq \frac{\|d_k\|^3}{2\beta\alpha_k}.$$

Proof. We multiply (2.2a) by d_k and use the assumption that $r_k^T d_k = 0$ to obtain

$$-\nabla f(x_k)^T d_k = d_k^T (\nabla^2 f(x_k) + \lambda_k I) d_k,$$

so that

$$q_k(0) - q_k(d_k) = -\nabla f(x_k)^T d_k - \tfrac{1}{2} d_k^T \nabla^2 f(x_k) d_k \tag{2.3}$$

$$= \tfrac{1}{2} d_k^T \nabla^2 f(x_k) d_k + \lambda_k \|d_k\|^2, \tag{2.4}$$

and using (2.2b) and (2.2c) yields the result. □

The assumption that $r_k^T d_k = 0$ will naturally be satisfied by the implementation that we propose in Section 3 and corresponds to the requirement [1, (3.11)].

We next observe that α_k is bounded away from zero if $\nabla^2 f$ is Lipschitz continuous.

Lemma 2.2. *If Assumptions 2.1, 2.2 and 2.4b are satisfied and $r_k^T d_k = 0$, then $\alpha_{k+1} \geq \alpha_k$ whenever $\alpha_k < (1-\eta_2)/(2\beta L)$. Thus, $\alpha_k \geq \alpha_{\min} > 0$ for all $k \geq 0$ where $\alpha_{\min} := \min(\alpha_0, \gamma_1(1-\eta_2)/(2\beta L))$.*

Proof. Assumption 2.2 ensures that $f(x_k + d_k) - q_k(d_k) \leq L\|d_k\|^3$. Lemma 2.1 implies

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)} = 1 + \frac{q_k(d_k) - f(x_k + d_k)}{\Delta q_k(d_k)} \geq 1 - 2L\beta\alpha_k.$$

Therefore, $\rho_k > \eta_2$ whenever $\alpha_k < (1-\eta_2)/(2\beta L)$, and $\alpha_{k+1} \geq \alpha_k$. The smallest possible value to which α_k may decrease is then $\gamma_1(1-\eta_2)/(2\beta L)$. Should the initial value α_0 be smaller than this threshold, the first iterations will increase α_k so that the overall lower bound is the value $\alpha_{\min}$ stated. □

The next requirement states the accuracy on the residual of the Newton equations.

Assumption 2.6. *In (2.2a), $\|r_k\| \leq \xi \min(\|\nabla f(x_k)\|, \|d_k\|)^2$ for some $\xi > 0$.*

Lemma 2.3. *Let Assumptions 2.1, 2.2, 2.4b and 2.6 be satisfied and assume that $r_k^T d_k = 0$. Then $\|\nabla f(x_{k+1})\| \leq \kappa_g^{-1} \|d_k\|^2$ for each successful iteration k, where $\kappa_g := (\frac{1}{2}L + \beta/\alpha_{\min} + \xi)^{-\frac{1}{2}}$.*

Proof. Using a generalization of the fundamental theorem of integral calculus [21, §8.1.2], (2.2a) and Assumptions 2.1 and 2.2, we may write

$$\begin{aligned}\|\nabla f(x_{k+1})\| &= \left\|\nabla f(x_k) + \int_0^1 \nabla^2 f(x_k + \tau d_k) d_k \,\mathrm{d}\tau\right\| \\ &= \left\|\int_0^1 \nabla^2 f(x_k + \tau d_k) d_k \,\mathrm{d}\tau - (\nabla^2 f(x_k) + \lambda_k I) d_k + r_k\right\| \\ &= \left\|\int_0^1 (\nabla^2 f(x_k + \tau d_k) - \nabla^2 f(x_k)) d_k \,\mathrm{d}\tau - \lambda_k d_k + r_k\right\| \\ &\leq \|d_k\| \left\|\int_0^1 L\tau d_k \,\mathrm{d}\tau\right\| + \lambda_k \|d_k\| + \|r_k\|.\end{aligned}$$

Now, (2.2c), Assumption 2.6 and Lemma 2.2 combine with the above to yield

$$\|\nabla f(x_{k+1})\| \leq (\tfrac{1}{2}L + \beta/\alpha_k + \xi) \|d_k\|^2 \leq \kappa_g^{-2} \|d_k\|^2. \qquad \square$$

For a given $\epsilon > 0$, we now wish to bound the total work required to reach a first iteration $k(\epsilon)$ such that $\|\nabla f(x_{k(\epsilon)})\| < \epsilon$ and $\|\nabla f(x_j)\| \geq \epsilon$ for $j < k(\epsilon)$.

Following Cartis et al. [2], we define the index sets

$$\begin{aligned}\mathcal{S} &:= \{k \geq 0 \mid \text{iteration } k \text{ is successful or very successful}\} \\ \mathcal{S}(\epsilon) &:= \{k \in \mathcal{S} \mid \|\nabla f(x_{k+1})\| \geq \epsilon\} \\ \mathcal{U} &:= \{k \geq 0 \mid \text{iteration } k \text{ is unsuccessful}\}.\end{aligned}$$

For sets such as $\mathcal{S}$, we denote their cardinality as $|\mathcal{S}|$.

Theorem 2.4 (Complexity bound of ARC$_q$)**.** *Let Assumptions 2.1 to 2.3 and 2.5 to 2.6 be satisfied and assume that $r_k^T d_k = 0$. Then,*

$$|\mathcal{S}(\epsilon)| \leq \frac{f(x_0) - f_{low}}{C} \epsilon^{-\frac{3}{2}} = O(\epsilon^{-\frac{3}{2}}), \qquad C := \eta_1 \frac{\kappa_g^3}{2\beta\alpha_{\max}} > 0,$$

where κ_g is defined in Lemma 2.3. In addition,

$$|\mathcal{S}(\epsilon)| + |\mathcal{U}| \leq \frac{f(x_0) - f_{low}}{C} \left(1 + \log\left(\frac{\alpha_{\min}}{\alpha_{\max}}\right) / \log \gamma_1\right) \epsilon^{-\frac{3}{2}}.$$

Proof. Let $k \in \mathcal{S}(\epsilon)$. Lemmas 2.1 and 2.3 combine with Assumption 2.5 to give

$$f(x_k) - q_k(d_k) \geq \frac{\kappa_g^3}{2\beta\alpha_k} \|\nabla f(x_{k+1})\|^{\frac{3}{2}} \geq \frac{\kappa_g^3}{2\beta\alpha_{\max}} \epsilon^{\frac{3}{2}}. \tag{2.5}$$

Because k is a successful iteration,

$$f(x_k) - f(x_{k+1}) \geq \eta_1 (f(x_k) - q_k(d_k)) \geq \eta_1 \frac{\kappa_g^3}{2\beta\alpha_{\max}} \epsilon^{\frac{3}{2}} = C\, \epsilon^{\frac{3}{2}}.$$

We have shown that at every iteration $k \in \mathcal{S}(\epsilon)$, f decreases by at least $C\epsilon^{\frac{3}{2}} > 0$ and Assumption 2.3 ensures that $\mathcal{S}(\epsilon)$ must be finite. Let $\mathcal{S}(\epsilon) = \{k_1, \ldots, k_{\ell(\epsilon)}\}$ with $k_i < k_{i+1}$ for $i = 1, \ldots, \ell(\epsilon)$. Because intermediate unsuccessful iterations keep f constant, the above combines with Assumption 2.3 to yield

$$f(x_0) - f_{\text{low}} \geq f(x_0) - f(x_{k_{\ell(\epsilon)}}) \geq \ell(\epsilon)\, C\, \epsilon^{\frac{3}{2}}.$$

Therefore,

$$|\mathcal{S}(\epsilon)| = \ell(\epsilon) \leq \frac{f(x_0) - f_{\text{low}}}{C} \epsilon^{-\frac{3}{2}}.$$

In the most pessimistic scenario, a sequence of unsuccessful iterations reduces α_k from $\alpha_{\max}$ down to $\alpha_{\min}$ after each successful iteration. Each such sequence has at most $\log(\alpha_{\min}/\alpha_{\max})/\log\gamma_1$ iterations since each unsuccessful iteration reduces α_k by a factor γ_1. The total number of unsuccessful iterations across all such sequences is thus bounded above by

$$|\mathcal{U}| \leq \ell(\epsilon) \log\left(\frac{\alpha_{\min}}{\alpha_{\max}}\right) / \log\gamma_1.$$

Finally, the total number of iterations is bounded above by

$$|\mathcal{S}(\epsilon)| + |\mathcal{U}| \leq \frac{f(x_0) - f_{\text{low}}}{C} \left(1 + \log\left(\frac{\alpha_{\min}}{\alpha_{\max}}\right) / \log\gamma_1\right) \epsilon^{-\frac{3}{2}}. \qquad \square$$

Theorem 2.4 also establishes global convergence of ARC$_q$ to first order stationary points. Indeed, for any $\epsilon > 0$, the number of iterations with $\|\nabla f(x_k)\| > \epsilon$ is finite, so that $\liminf \|\nabla f(x_k)\| = 0$. It relies on the strong assumption that we identify approximate second-order solutions of the regularized subproblems as specified by (2.2a)–(2.2c). Because our implementation, detailed in Section 3 satisfies that assumption, Theorem 2.4 is sufficient for our purposes. However, our implementation is intended for large-scale problems and Assumption 2.6 may be too stringent. Fortunately, it can be relaxed as follows.

Assumption 2.6b. *In (2.2a), $\|r_k\| \leq \xi \min(\|\nabla f(x_k)\|, \|d_k\|)^{1+\zeta}$ for some $\xi > 0$ and $0 < \zeta \leq 1$.*

Assumption 2.6b changes Lemma 2.3 as follows.

Lemma 2.4. *Let Assumptions 2.1, 2.2, 2.4b and 2.6b be satisfied and assume that $r_k^T d_k = 0$. Then $\|\nabla f(x_{k+1})\| \leq \kappa_g^{-1}(\|d_k\|^2 + \|d_k\|^{1+\zeta})$ for each successful iteration k, where $\kappa_g := 2\max(\frac{1}{2}L + \beta/\alpha_{\min}, \xi)$.*

Proof. Using the same arguments as in the proof of Lemma 2.3,

$$\begin{aligned}\|\nabla f(x_{k+1})\| &\leq \|d_k\| \left\| \int_0^1 L\tau d_k \, \mathrm{d}\tau \right\| + \lambda_k \|d_k\| + \|r_k\| \\ &\leq (\tfrac{1}{2}L + \beta/\alpha_{\min}) \|d_k\|^2 + \xi \|d_k\|^{1+\zeta}. \end{aligned} \qquad \square$$

Accordingly, Theorem 2.4 is modified as follows.

Corollary 2.1 (Complexity bound of approximate ARC$_q$)**.** *Under the same assumptions as Theorem 2.4 with Assumption 2.6 replaced with Assumption 2.6b, we have*

$$|\mathcal{S}(\epsilon)| \leq \frac{f(x_0) - f_{low}}{C} \max(\epsilon^{-3/(1+\zeta)}, \epsilon^{-3/2}), \qquad C := \eta_1 \frac{\kappa_g^3}{16\beta\alpha_{\max}} > 0,$$

where κ_g is defined in Lemma 2.4. In addition,

$$|\mathcal{S}(\epsilon)| + |\mathcal{U}| \leq \frac{f(x_0) - f_{low}}{C} \left(1 + \log\left(\frac{\alpha_{\min}}{\alpha_{\max}}\right) / \log\gamma_1\right) \max(\epsilon^{-3/(1+\zeta)}, \epsilon^{-3/2}).$$

Proof. Proceeding as in Theorem 2.4, for any $k \in \mathcal{S}(\epsilon)$, Lemma 2.1 and Assumption 2.5 imply $f(x_k) - q_k(d_k) \geq (2\beta\alpha_k)^{-1}\|d_k\|^3 \geq (2\beta\alpha_{\max})^{-1}\|d_k\|^3$. There are now two cases to consider. Consider first the possibility that at this iteration k, $\|d_k\| \leq 1$ so that $\|d_k\|^2 \leq \|d_k\|^{1+\zeta}$. We obtain from Lemma 2.4 that $\|\nabla f(x_{k+1})\| \leq 2\kappa_g^{-1}\|d_k\|^{1+\zeta}$. We now simply follow the logic of the proof of Theorem 2.4 and conclude that for this iteration,

$$f(x_k) - f(x_{k+1}) \geq \frac{\eta_1 \kappa_g^3}{16\beta\alpha_{\max}} \epsilon^{3/(1+\zeta)}.$$

Consider next the case $\|d_k\| > 1$. The same reasoning leads us to conclude that for this iteration,

$$f(x_k) - f(x_{k+1}) \geq \frac{\eta_1 \kappa_g^3}{16\beta\alpha_{\max}} \epsilon^{3/2}.$$

Thus, for any $k \in \mathcal{S}(\epsilon)$,

$$f(x_k) - f(x_{k+1}) \geq C \min(\epsilon^{3/(1+\zeta)}, \epsilon^{3/2}).$$

Continuing with the logic of the proof of Theorem 2.4, we obtain

$$|\mathcal{S}(\epsilon)| \leq \frac{f(x_0) - f_{\text{low}}}{C} \max(\epsilon^{-3/(1+\zeta)}, \epsilon^{-3/2}).$$

The remainder of the proof is unchanged. □

In a typical scenario, the tolerance $\epsilon < 1$ and Corollary 2.1 yields a complexity of $O(\epsilon^{-3/(1+\zeta)})$ iterations. However, if a loose tolerance $\epsilon \geq 1$ is of interest, the complexity is instead of $O(\epsilon^{-3/2})$ iterations. In the next section, we confirm that in the local regime, superlinear convergence occurs at a rate $1 + \zeta$.

2.3. **Asymptotic analysis.** We now address the behavior of the Algorithm 2.1 in the vicinity of an isolated local minimizer, i.e., we make the following assumption.

Assumption 2.7. *Assume that* $\{x_k\} \to x^*$ *with* $\nabla f(x^*) = 0$ *and* $\nabla^2 f(x^*) \succ 0$.

Our next result states that $\{\nabla f(x_k)\}$ converges to zero superlinearly with rate $1 + \zeta$, where ζ is stated in Assumption 2.6b.

Theorem 2.5 (Superlinear convergence of ARC$_q$)**.** *Let Assumptions 2.1 to 2.3 and 2.5 to 2.7 be satisfied and assume that* $r_k^T d_k = 0$ *for all sufficiently large* k. *Then there exists an iteration* K *such that for all* $k \geq K$,

$$\|\nabla f(x_{k+1})\| = O(\|\nabla f(x_k)\|^{1+\zeta}). \tag{2.6}$$

Proof. Let $\lambda_{\min}^* > 0$ denote the smallest eigenvalue of $\nabla^2 f(x^*)$. Assumption 2.2 ensures that there is $K \in \mathbb{N}$ such that the smallest eigenvalue of $\nabla^2 f(x_k)$ is $\lambda_{\min}^k \geq \frac{1}{2}\lambda_{\min}^*$ for all $k \geq K$.

Proceeding as in the proof of Lemma 2.1, (2.4) yields $\Delta q_k(d_k) \geq \frac{1}{2}\lambda_{\min}^k \|d_k\|^2$. A similar analysis to that of the proof of Lemma 2.2 then implies

$$\rho_k \geq 1 - 2L\frac{\|d_k\|}{\lambda_{\min}^k} \geq 1 - 4L\frac{\|d_k\|}{\lambda_{\min}^*}. \tag{2.7}$$

Because $\{x_k\} \to x^*$, we can assume that K is large enough that for all $k \geq K$,

$$\|d_k\| \leq \frac{1-\eta_1}{4L}\lambda^*_{\min},$$

which combines with (2.7) to show that $\rho_k \geq \eta_1$ for all $k \geq K$, and thus, only successful iterations are performed.

Now, using (2.2a) and Assumption 2.6b,

$$\begin{aligned}\|d_k\| &\leq \|(\nabla^2 f(x_k) + \lambda_k I)^{-1}\| \, \|r_k - \nabla f(x_k)\| \\ &\leq \frac{\|r_k\| + \|\nabla f(x_k)\|}{\lambda^k_{\min}} \leq 2\|\nabla f(x_k)\| \, \frac{1 + \|\nabla f(x_k)\|^\zeta}{\lambda^*_{\min}},\end{aligned}$$

for all $k \geq K$. Finally, we can also assume that $\|d_k\| \leq 1$ so that Lemma 2.4 yields (2.6). □

3. CG-Lanczos implementation

We now present an implementation that takes advantage of the approximation criteria developed above. Our implementation is based on the conjugate-gradient method (CG) of Hestenes and Stiefel [16]. We first explain how we use shifted systems to implement ARC$_q$K and then describe the details of our implementation of CG based on the Lanczos [18] process to solve all the shifted systems simultaneously. Although the Lanczos form of CG differs from that initially stated by Hestenes and Stiefel [16], it is equivalent to it in exact arithmetic, can be derived from the Lanczos tridiagonalization process, and is amenable to the simultaneous solution of shifted symmetric systems.

The key idea is to discretize the half line $0 < \lambda < \infty$ into values $0 < \lambda_0 < \cdots < \lambda_m < \infty$. For reasons motivated by double precision arithmetic, we impose $10^{-15} \leq \lambda_i \leq 10^{15}$ for $i = 0, \ldots, m$. From the formulation (2.2c), a simple choice consists in setting $\beta\lambda_i = \lambda_{i+1}/\beta$, for some $\beta > 1$. For instance, $\beta = \sqrt{10}$ yields the 31 values $\lambda_i = 10^i$ for $i = -15, \ldots, 15$. The computational feasibility of such a procedure is detailed below and comes from the fact that an appropriate shifted CG-Lanczos implementation obtains all $m+1$ (approximate) solutions of $(\nabla^2 f(x) + \lambda_i I)d(\lambda_i) \approx -\nabla f(x)$, or establishes that $\nabla^2 f(x) + \lambda_i I \not\succeq 0$, at modest expense beyond that of solving a single linear system.

We now describe how we solve a sequence of shifted linear systems simultaneously in a way that is consistent with the minimization of (1.3). Our implementation is an adaptation of [10, Algorithm 6].

Algorithm 3.1 describes the CG-Lanczos with shifts implementation for a generic symmetric system $Mx = b$ with shifts λ_i, i.e.,

$$(M + \lambda_i I)x = b, \quad i = 0, \ldots, m. \tag{3.1}$$

In Algorithm 3.1, boldface quantities are block quantities with one component per shift parameter. Initialization statements initialize all $m+1$ values of a given block variable to identical copies of the right hand side value. For instance, the statement $\boldsymbol{p} = b$ means that the $n \times (m+1)$ array $\boldsymbol{p}$ is initialized to $m+1$ copies of b. The statement $\boldsymbol{\sigma} = \beta$ means that all $m+1$ elements of the array $\boldsymbol{\sigma}$ are initialized to β. For conciseness, the shifts are gathered in the array $\boldsymbol{\lambda}$.

A few observations about Algorithm 3.1 are in order. Firstly, note that a single operator-vector product is required per iteration, and takes place in the Lanczos

Algorithm 3.1 Lanczos-CG with shifts for (3.1)

1: Set $\boldsymbol{x}_0 = 0$, $\beta_0 v_0 = b$, $\boldsymbol{p}_0 = b$ — β_0 *such that* $\|v_0\| = 1$
2: set $v_{-1} = 0$, $\boldsymbol{\sigma}_0 = \beta_0$, $\boldsymbol{\omega}_{-1} = 0$, $\boldsymbol{\gamma}_{-1} = 1$, $\boldsymbol{\pi}_0 = \beta_0^2$ — $\boldsymbol{\pi}_0 = \|\boldsymbol{p}_0\|^2$
3: **for** $j \leftarrow 0, 1, 2, \ldots$ **do**
4: $\quad \delta_j = v_j^T M v_j$ — *Lanczos part of the iteration*
5: $\quad \beta_{j+1} v_{j+1} = M v_j - \delta_j v_j - \beta_j v_{j-1}$ — β_{j+1} *such that* $\|v_{j+1}\| = 1$
6: $\quad \boldsymbol{\delta}_j = \delta_j + \boldsymbol{\lambda}$ — *CG part of the iteration in block form*
7: $\quad \boldsymbol{\gamma}_j = 1/(\boldsymbol{\delta}_j - \boldsymbol{\omega}_{j-1}/\boldsymbol{\gamma}_{j-1})$
8: $\quad \boldsymbol{\omega}_j = (\beta_{j+1}\boldsymbol{\gamma}_j)^2$
9: $\quad \boldsymbol{\sigma}_{j+1} = -\beta_{j+1}\boldsymbol{\gamma}_j \boldsymbol{\sigma}_j$ — $\boldsymbol{\sigma}_{j+1} = \|\boldsymbol{r}_{j+1}\|$
10: $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \boldsymbol{\gamma}_j \boldsymbol{p}_j$
11: $\quad \boldsymbol{p}_{j+1} = \boldsymbol{\sigma}_{j+1} v_{j+1} + \boldsymbol{\omega}_j \boldsymbol{p}_j$
12: $\quad \boldsymbol{\pi}_{j+1} = \boldsymbol{\sigma}_{j+1}^2 + \boldsymbol{\omega}_j^2 \boldsymbol{\pi}_j$ — $\boldsymbol{\pi}_{j+1} = \|\boldsymbol{p}_{j+1}\|^2$
13: **end for**

part of the iteration, which is independent of the shifts. The extra cost incurred by requesting the solution of multiple shifted systems is confined to the CG part of the iteration, which only performs scalar and vector operations.

Secondly, recall that the vectors v_j are orthonormal in exact arithmetic while the search directions $\boldsymbol{p}_j$ are $(M + \boldsymbol{\lambda} I)$-conjugate so long as negative curvature is not detected.

Thirdly, Algorithm 3.1 neither forms nor recurs the residual $\boldsymbol{r}_j = b - M\boldsymbol{x}_j$. A recursion argument shows that $\boldsymbol{r}_j = \boldsymbol{\sigma}_j v_j$, and by orthogonality, $\|\boldsymbol{r}_j\| = \boldsymbol{\sigma}_j$ is available at no extra cost and may be used in a stopping criterion.

Finally, Algorithm 3.1 uses $\boldsymbol{\pi}_j$ to recur $\|\boldsymbol{p}_j\|^2$, i.e., the squared Euclidean norm of each column of $\boldsymbol{p}_j$. Initially, each component of $\boldsymbol{\pi}_0$ is set to $\beta_0^2 = \|b\|^2$. Because the conjugate-gradient method minimizes a quadratic whose gradient at $\boldsymbol{x}_j$ is $-\boldsymbol{r}_j$ along the direction $\boldsymbol{p}_j$ using an exact linesearch, we must have $\boldsymbol{r}_{j+1}^T \boldsymbol{p}_j = \boldsymbol{\sigma}_{j+1} v_{j+1}^T \boldsymbol{p}_j = 0$. Thus,

$$\|\boldsymbol{p}\|_{j+1}^2 = \boldsymbol{\sigma}_{j+1}^2 \|v_{j+1}\|^2 + 2\boldsymbol{\sigma}_{j+1}\boldsymbol{\omega}_j v_{j+1}^T \boldsymbol{p}_j + \boldsymbol{\omega}^2 \|\boldsymbol{p}_j\|^2,$$

which yields the update on line 12.

Not all shifted systems will require the same number of iterations to converge and we interrupt iterations corresponding to values of the shift for which either the required tolerance is reached, or negative curvature is detected.

We now describe how negative curvature may be detected during the iterations of Algorithm 3.1. Because the argument is independent of the shift, we assume that $m = 1$ and $\lambda_1 = 0$, i.e., we solve the system $Mx = b$ with the Lanczos variant of CG. At iteration j,

$$\delta_j = v_j^T M v_j,$$

where the vectors $\{v_j\}$ are orthonormal. If negative curvature is present, δ_j may never reveal so, but $p_j^T M p_j$ will. We seek a cheap expression to check the sign of $p_j^T M p_j$ where the vectors $\{p_j\}$ are M-conjugate. At iteration j, $p_j = r_j + \omega_{j-1} p_{j-1}$, where $r_j = b - Mx_j$ is updated cheaply via $r_j = \sigma_j v_j$, and ω_{j-1} and σ_j are scalars.

Thus

$$\begin{aligned} p_j^T M p_j &= p_j^T M r_j + \omega_{j-1} p_j^T M p_{j-1} \\ &= p_j^T M r_j \\ &= r_j^T M r_j + \omega_{j-1} p_{j-1}^T M r_j \\ &= \sigma_j^2 \delta_j + \omega_{j-1} p_{j-1}^T M r_j. \end{aligned} \tag{3.2}$$

The iterates are updated according to $x_j = x_{j-1} + \gamma_{j-1} p_{j-1}$, so that

$$r_j = b - M x_j = b - M x_{j-1} - \gamma_{j-1} M p_{j-1} = r_{j-1} - \gamma_{j-1} M p_{j-1}.$$

By orthogonality,

$$\sigma_j^2 = r_j^T r_j = r_j^T r_{j-1} - \gamma_{j-1} r_j^T M p_{j-1} = -\gamma_{j-1} r_j^T M p_{j-1},$$

and therefore,

$$p_{j-1}^T M r_j = -\frac{1}{\gamma_{j-1}} r_j^T r_j = -\frac{1}{\gamma_{j-1}} \sigma_j^2. \tag{3.3}$$

Finally, using the update formula for γ_j, (3.2) and (3.3) combine to yield

$$p_j^T M p_j = \sigma_j^2 (\delta_j - \omega_{j-1}/\gamma_{j-1}) = \sigma_j^2 / \gamma_j.$$

Therefore the sign of γ_j is the same as that of $p_j^T M p_j$.

4. Implementation and numerical experiments

We implemented Algorithm 4.1 in the Julia language version 1.2 and used the following packages from the JuliaSmoothOptimizers organization [20]:

- NLPModels.jl provides a consistent optimization problem interface on which solvers can rely regardless of where problems come from;
- CUTEst.jl gives solvers access to the CUTEst collection [14] by bridging it with NLPModels.jl;
- Krylov.jl provides an implementation of Algorithm 3.1 alongside other Krylov methods;
- SolverBenchmark.jl provides data structures to gather problem statistics, and utilities to generate tables of results and performance profiles;
- SolverTools.jl implements building blocks for optimization algorithms, including trust region management and linesearch procedures, and facilities for benchmarking solvers on collections of test problems.

During the course of Algorithm 3.1, the solution of the ith system $(\nabla^2 f(x_k) + \lambda_i I)d = -\nabla f(x_k)$, $i = 1, \ldots, m$, is interrupted as soon as there is an iteration j for which the ith component of $\boldsymbol{\gamma}_j$ is negative, which indicates that $\nabla^2 f(x) + \lambda_i I \not\succeq 0$.

If $\boldsymbol{\gamma}_j \geq 0$ for all j, the solution of the ith system is terminated as soon as there is an iteration j for which the residual norm $\|r_j\| = \|\nabla f(x_k) + (\nabla^2 f(x_k) + \lambda_i I) d_j\|$ satisfies Assumption 2.6b.

The complete procedure is summarized as Algorithm 4.1. At line 5, we let i^+ be the index of the smallest shift for which no negative curvature was detected. Thus, no negative curvature was detected for any $i^+ \leq i \leq m$. If there exists no such i^+, the algorithm terminates because $\nabla^2 f(x_k)$ has negative eigenvalues larger than the largest allowed shift in magnitude. In our implementation, that means negative eigenvalues smaller than -10^{15}. It is always possible to increase the number and/or

the maximum value of the sampled λ if such occurrences happen often. Such a situation never happened in our numerical experiments.

If $d_i^* \approx d(\lambda_i)$ denotes the solution thus identified by Algorithm 3.1 corresponding to λ_i, we retain the one that most closely satisfies $\alpha_k \lambda_i = \|d_i^*\|$ at line 6.

When an iteration is unsuccessful, ARC$_q$ must compute a minimizer of (1.3) with the updated regularization parameter $\alpha_{k+1} = \gamma_1 \alpha_k$. By contrast, ARC$_q$K simply turns its attention to the next shift value and the associated search direction, which was already computed. For a given λ, an exact solution $d(\lambda)$ would satisfy (2.1c). If λ_k is the shift used at iteration k, and the iteration is unsuccessful, we search for the smallest $\lambda_j > \lambda_k$, i.e., the smallest $j > k$ such that $\alpha(\lambda_j) := \|d_j\|/\lambda_j \leq \gamma_1 \alpha_k$. If this search were to fail, it would indicate that our sample of shifts does not contain sufficiently large values.

Algorithm 4.1 ARC$_q$K.

1: Initialize $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $0 < \eta_1 < \eta_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$, $\boldsymbol{\lambda}$, $k = 0$
2: **repeat**
3: solve $(\nabla^2 f(x_k) + \boldsymbol{\lambda} I) d(\boldsymbol{\lambda}) = -\nabla f(x_k)$ for $d(\boldsymbol{\lambda})$ *use Algorithm 3.1*
4: success = **false**
5: $i^+ = \min\{0 \leq i \leq m \mid \nabla^2 f(x) + \lambda_i I \succ 0\}$
6: $j = \arg\min\{|\alpha\lambda_i - \|d(\lambda_i)\|| \mid i^+ \leq i \leq m\}$
7: **repeat**
8: $d_k = d(\lambda_j)$
9: compute $\rho_k = \dfrac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}$
10: **if** $\rho_k < \eta_1$ **then** *unsuccessful*
11: $\alpha_{k+1} = \alpha_k$
12: **while** $\alpha_{k+1} > \gamma_1 \alpha_k$ **do**
13: $\alpha_{k+1} = \|d_{j+1}\|/\lambda_{j+1}$ *consider next shifts*
14: $j = j + 1$
15: **end while**
16: **else** *successful*
17: success = **true**
18: $x_{k+1} = x_k + d_k$
19: **if** $\rho_k > \eta_2$ **then** *very successful*
20: $\alpha_{k+1} = \gamma_2 \alpha_k$ *increase α*
21: **else**
22: $\alpha_{k+1} = \alpha_k$
23: **end if**
24: **end if**
25: **until** success
26: $k \leftarrow k + 1$
27: **until** termination_criterion

In order to assess the scalability of our implementation, we perform numerical experiments using all 240 unconstrained problems from the CUTEst collection. We compare ARC$_q$K with an implementation of the truncated conjugate-gradient trust-region method [24, 25]. Solving a trust-region subproblem

$$\underset{d}{\text{minimize}} \; q_x(d) \quad \text{subject to} \; \|d\| \leq \Delta,$$

where $\Delta > 0$ is the trust-region radius, involves identifying a shift $\lambda \geq 0$ satisfying (2.1a)–(2.1b) but with (2.1c) replaced with the complementarity condition

$$\lambda(\Delta - \|d\|) = 0.$$

We refer the interested reader to [5, 24] for more details. For our comparison, we use a classical Steihaug-Toint approach in which we use a standard non shifted conjugate gradient algorithm which is interrupted either when reaching the boundary of the trust region or when satisfying the required accuracy.

Our experiments ran on a 2.4GHz four-core Intel Xeon E5530 with hyperthreading and 46Gb of memory. We limit the computing time to 3,600 seconds and declare stationarity as soon as $\|\nabla f(x)\| \leq \epsilon_a + \epsilon_r \|\nabla f(x_0)\|$ with $\epsilon_a = 10^{-5}$ and $\epsilon_r = 10^{-6}$. We do not set limits on the number of function evaluations.

The complete Julia implementation of Algorithm 4.1 and the Steihaug-Toint variant are described by Dussault [8]. The constants used are $\zeta = 0.5$, $\gamma_1 = 0.1$, $\gamma_2 = 5.0$, $\eta_1 = 0.1$, $\eta_2 = 0.75$. The required accuracy for both the Steihaug-Toint trust region variant and for ARC$_q$K is set to $\|r\| \leq \|\nabla f(x)\|^{1+\zeta}$, see Assumption 2.6.

ARC$_q$K solved 223 problems successfully, exceeded the time budget on 14 problems, declared problem `hielow` unbounded below,[1] and exited for an unknown reason on problem `nelsonls`. Our Steihaug-Toint implementation solved 226 problems, and exceeded the time budget on 14 problems. Both solvers exceeded the time budget on `ba_l16ls`, `ba_l21ls`, `ba_l49ls`, `ba_l52ls`, `ba_l73ls`, `eigenbls`, `fletcbv3`, `fletchbv`, `indef`, `jimack`, `nonsmsqrt`, `sbrybnd` and `scosine`. In addition, ARC$_q$K exceeded the time budget on `eigencls` and `mnists5ls`, and the Steihaug-Toint implementation exceeded the time budget on `yatp2ls`. Note that most of the above problems are large nonlinear least-squares problems for which Newton's method is probably not appropriate.

Complete numerical results for both algorithms are reported in the electronic supplement accompanying this paper. Figure 1 reports aggregated results in the form of time and evaluation Dolan and Moré [6] performance profiles.

While our preliminary implementation is slower than the Steihaug-Toint method, and requires more objective value and gradient evaluations, it saves a substantial amount of Hessian-vector products. Moreover, for a restricted set of the CUTEst collection of all (113) the problems of size at least 100, ARC$_q$K matches or outperforms the Steihaug-Toint method. This is to be expected, the overhead of solving the shifted systems being relatively high for smaller instances.

5. Extensions

5.1. **Inexact Hessians.** In many practical situations, the exact Hessian $\nabla^2 f(x_k)$ is either not available or too costly to evaluate, and we may wish to use an approximation. Thus, instead of (1.2), our quadratic model at iteration k becomes

$$q_k(d) = f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T B_k d,$$

where $B_k = B_k^T$. We update Assumption 2.4b as follows.

Assumption 2.4c. *At line 3 of Algorithm 2.1, we compute a direction d satisfying (2.2a)–(2.2c) with B_k in place of $\nabla^2 f(x_k)$ in (2.2a) and (2.2b).*

[1] An iterate was generated where the objective evaluated to $-\infty$.

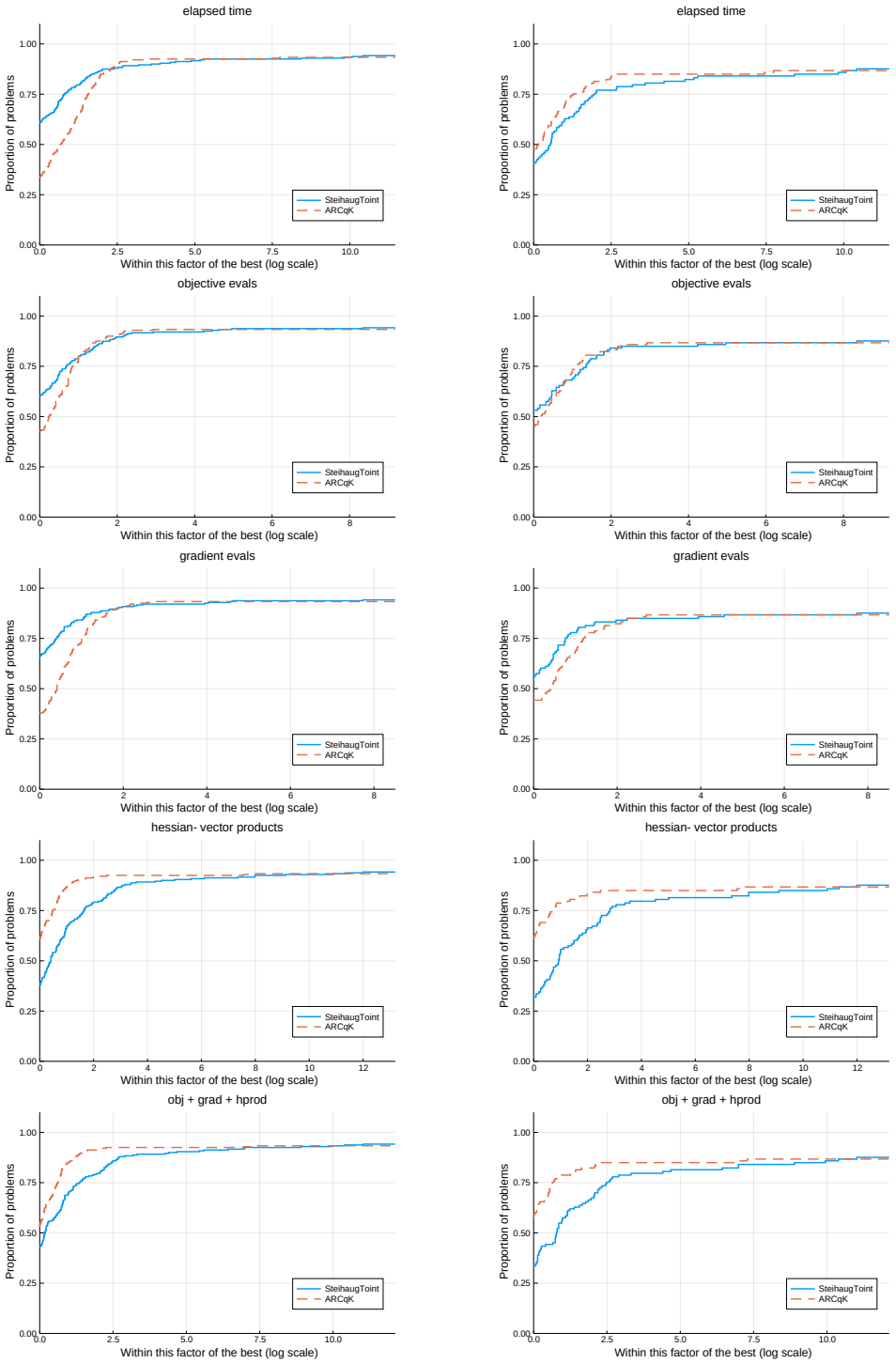

FIGURE 1. Performance profiles comparing ARC$_q$K with a standard truncated conjugate-gradient approach on 240 unconstrained CUTEst problems (left) and the 113 problems with $n \geq 100$ (right).

With this change, we may verify that the results of Section 2.2 can be adapted to accommodate inexact Hessians. Under Assumption 2.4c, Lemmas 2.1 and 2.2 are unchanged. We now make the following additional assumption on the quality of the approximation, which was also used with $\zeta = 1$ by Cartis et al. [1, 2].

Assumption 5.1. *There exists a constant $C_H > 0$ such that for all k,*

$$\|(\nabla^2 f(x_k) - B_k)d_k\| \leq C_H \|d_k\|^{1+\zeta},$$

where ζ is as in Assumption 2.6b.

We now verify that if an inexact step d_k is computed so that the residual r_k satisfies Assumption 2.6b, Lemma 2.4 continues to apply.

Lemma 5.1. *Let Assumptions 2.1, 2.2, 2.6b and 2.4c be satisfied and assume that $r_k^T d_k = 0$. Then $\|\nabla f(x_{k+1})\| \leq \kappa_g^{-1}(\|d_k\|^2 + \|d_k\|^{1+\zeta})$ for each successful iteration k, where $\kappa_g := 2\max(\frac{1}{2}L + \beta/\alpha_{\min}, C_H + \xi)$.*

Proof. Proceeding as in the proof of Lemma 2.3, we bound $\|\nabla f(x_{k+1})\|$ above with

$$\begin{aligned}
&\left\|\nabla f(x_k) + \int_0^1 \nabla^2 f(x_k + \tau d_k)d_k \,\mathrm{d}\tau\right\| \\
&= \left\|\int_0^1 \nabla^2 f(x_k + \tau d_k)d_k \,\mathrm{d}\tau - (B_k + \lambda_k I)d_k + r_k\right\| \\
&= \left\|\int_0^1 (\nabla^2 f(x_k + \tau d_k) - \nabla^2 f(x_k))d_k \,\mathrm{d}\tau + (\nabla^2 f(x_k) - B_k)d_k - \lambda_k d_k + r_k\right\| \\
&\leq \|d_k\| \left\|\int_0^1 L\tau d_k \,\mathrm{d}\tau\right\| + C_H\|d_k\|^{1+\zeta} + \lambda_k\|d_k\| + \|r_k\|.
\end{aligned}$$

Now, (2.2c), Assumptions 2.6b and 5.1, and Lemma 2.2 yield

$$\|\nabla f(x_{k+1})\| \leq (\tfrac{1}{2}L + \beta/\alpha_{\min})\,\|d_k\|^2 + (C_H + \xi)\,\|d_k\|^{1+\zeta}.$$

We may now conclude as in the proof of Lemma 2.4. □

The complexity bound of Corollary 2.1 continues to apply. Finally, the local rate given in Theorem 2.5 also continues to apply with a slight update to the proof. We state it for completeness.

Theorem 5.1 (Superlinear convergence of ARC$_q$)**.** *Let Assumptions 2.1 to 2.3, 2.5, 2.6b, 2.7, 2.4c and 5.1 be satisfied and assume that $r_k^T d_k = 0$ for all sufficiently large k. Then there exists an iteration K such that for all $k \geq K$, (2.6) holds.*

Proof. We proceed as in the proof of Theorem 2.5 with the difference that we bound $\|d_k\|$ as follows. Assumptions 2.4c and 5.1 allow us to write

$$\begin{aligned}
\|r_k - \nabla f(x_k)\| &\leq \|(\nabla^2 f(x_k) + \lambda_k I)d_k\| + \|(\nabla^2 f(x_k) - B_k)d_k\| \\
&\leq \lambda_{\min}^k \|d_k\| + C_H\|d_k\|^{1+\zeta} \\
&\leq \tfrac{1}{2}\lambda_{\min}^* \|d_k\| + C_H\|d_k\|^{1+\zeta}.
\end{aligned}$$

Because $\{d_k\} \to 0$, we can choose K large enough that $C_H\|d_k\|^{1+\zeta} \leq \frac{1}{2}\lambda^*_{\min}\|d_k\|$ for all $k \geq K$. Assumption 2.6b now yields

$$\|d_k\| \leq \frac{\|r_k\| + \|\nabla f(x_k)\|}{\lambda^*_{\min}},$$

and the proof concludes as in Theorem 2.5. □

5.2. **Nonlinear Least-Squares Problems.** A prime application of the extension to inexact Hessians occurs in the context of nonlinear least-squares problems, where

$$f(x) = \tfrac{1}{2}\|F(x)\|^2, \qquad F(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix},$$

and each $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$. We begin with the following assumption.

Assumption 5.2. *Each f_i, $i = 1, \ldots, m$, is twice continuously differentiable.*

Evidently, Assumption 5.2 implies Assumption 2.1.

For reference, we note that

$$\nabla f(x) = J(x)^T F(x) \quad \text{where} \quad J(x) := \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{bmatrix},$$

and

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^m f_i(x)\nabla^2 f_i(x).$$

We follow Cartis et al. [4] and make the following additional assumption.

Assumption 5.3. *Each f_i, ∇f_i and $\nabla^2 f_i$, $i = 1, \ldots, m$ is globally Lipschitz continuous.*

Under Assumption 5.3, it is possible to show that Assumption 2.2 is satisfied.

Assumption 2.3 is not necessary in the least-squares context as we may simply set $f_{\text{low}} = 0$.

In practice, it is common to use the Gauss-Newton approximation $J(x)^T J(x)$ instead of $\nabla^2 f(x)$, which is justified if the residuals f_i are all either nearly zero or nearly linear around a local minimizer. Thus we consider the following variant of Assumption 5.1.

Assumption 5.4. *There exists a constant $C_H > 0$ such that for all k,*

$$\|\sum_{i=1}^m f_i(x_k)\nabla^2 f_i(x_k)d_k\| \leq C_H \|d_k\|^{1+\zeta},$$

where ζ is as in Assumption 2.6b.

Under the above assumption, the convergence and complexity properties of the previous sections apply to the nonlinear least-squares problem. Obviously, should Assumption 5.4 be too restrictive, a different Hessian approximation satisfying Assumption 5.1 can be used. In the sequel, we focus on the Gauss-Newton approximation because of its ubiquity.

The solution of the shifted system

$$(J(x_k)^T J(x_k) + \lambda_k I)d_k = -J(x_k)^T F(x_k), \tag{5.1}$$

known as the *regularized normal equations*, can be interpreted as the solution of the regularized linear least-squares problem

$$\underset{d}{\text{minimize}} \ \tfrac{1}{2} \left\| \begin{bmatrix} J(x_k) \\ \sqrt{\lambda_k} I \end{bmatrix} d + \begin{bmatrix} F(x_k) \\ 0 \end{bmatrix} \right\|^2 . \tag{5.2}$$

It is well-known that applying CG (or CG-Lanczos) directly to (5.1) is prone to accumulation of rounding errors because $J(x_k)^T J(x_k)$ can be substantially more ill conditioned than $J(x_k)$. Several iterative methods exist for (5.2), but a straightforward approach consists in modifying Algorithm 3.1 to accommodate the structure of (5.1) and decouple the product with $J(x_k)$ from that with $J(x_k)^T$. Such a modification of CG is known as CGLS and several implementations appear in the literature, the first of which was stated by Hestenes and Stiefel [16]. As it turns out, Frommer and Maass [10] were also interested in (5.1). We use their formulation, stated as Algorithm 5.1 for the generic regularized normal equations

$$(A^T A + \lambda I)x = A^T b, \tag{5.3}$$

where A can be rectangular.

Algorithm 5.1 Lanczos-CGLS with shifts for (5.3)

1: Set $\boldsymbol{x}_0 = 0$, $u_0 = b$, $\beta_0 v_0 = A^T u_0$, $u_0 = u_0/\beta_0$, $\boldsymbol{p}_0 = \beta_0 v_0$ *β_0 such that $\|v_0\| = 1$*
2: set $u_{-1} = 0$, $\boldsymbol{\sigma}_0 = \beta_0$, $\boldsymbol{\omega}_{-1} = 0$, $\boldsymbol{\gamma}_{-1} = 1$, $\boldsymbol{\pi}_0 = \beta_0^2$ *$\pi_0 = \|\boldsymbol{p}_0\|^2$*
3: **for** $j \leftarrow 0, 1, 2, \ldots$ **do**
4: $\quad \tilde{u}_j = A v_j$ *Lanczos part of the iteration*
5: $\quad \delta_j = \tilde{u}_j^T \tilde{u}_j$ *$\delta_j = v_j^T A^T A v_j$*
6: $\quad u_{j+1} = \tilde{u}_j - \delta_j u_j - \beta_j u_{j-1}$
7: $\quad \beta_{j+1} v_{j+1} = A^T u_{j+1}$ *β_{j+1} such that $\|v_{j+1}\| = 1$*
8: $\quad u_{j+1} = u_{j+1}/\beta_{j+1}$
9: $\quad \boldsymbol{\delta}_j = \delta_j + \boldsymbol{\lambda}$ *CGLS part of the iteration in block form*
10: $\quad \boldsymbol{\gamma}_j = 1/(\boldsymbol{\delta}_j - \boldsymbol{\omega}_{j-1}/\boldsymbol{\gamma}_{j-1})$
11: $\quad \boldsymbol{\omega}_j = (\beta_{j+1}\boldsymbol{\gamma}_j)^2$
12: $\quad \boldsymbol{\sigma}_{j+1} = -\beta_{j+1}\boldsymbol{\gamma}_j \boldsymbol{\sigma}_j$ *$\boldsymbol{\sigma}_{j+1} = \|\boldsymbol{r}_{j+1}\|$*
13: $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \boldsymbol{\gamma}_j \boldsymbol{p}_j$
14: $\quad \boldsymbol{p}_{j+1} = \boldsymbol{\sigma}_{j+1} v_{j+1} + \boldsymbol{\omega}_j \boldsymbol{p}_j$
15: $\quad \boldsymbol{\pi}_{j+1} = \boldsymbol{\sigma}_{j+1}^2 + \boldsymbol{\omega}_j^2 \boldsymbol{\pi}_j$ *$\boldsymbol{\pi}_{j+1} = \|\boldsymbol{p}_{j+1}\|^2$*
16: **end for**

If A has size $m \times n$, each $v_j \in \mathbb{R}^n$ and we see that Algorithm 5.1 also updates auxiliary vectors $u_j \in \mathbb{R}^m$. Typical least-squares problems are over-determined, i.e., $m > n$, although the algorithm remains well defined for $m \leq n$. The vector $\tilde{u}_j$ at line 4 can share storage with u_{j+1}.

Note that the residual vector whose norm is updated as $\boldsymbol{\sigma}_{j+1}$ now represents the optimality residual $\boldsymbol{r}_j := A^T(b - A\boldsymbol{x}_j)$. Algorithm 5.1 can be further improved by monitoring the least-squares residual $\boldsymbol{s}_j := b - A\boldsymbol{x}_j$. The latter would be initialized as $\boldsymbol{s}_0 = u_0 = b$ and simply updated as $\boldsymbol{s}_{j+1} = \boldsymbol{\sigma}_j u_j$ after u_j has been normalized. Contrary to the v_j, the u_j are not orthogonal so that $\|\boldsymbol{s}_j\|$ must be explicitly computed if it is of interest.

A final difference with Algorithm 3.1 is that Algorithm 5.1 need not be concerned with detection of negative curvature.

Discussion

We introduced a new scalable factorization-free implementation of the ARC_q variant of the adaptive regularization by cubics. An inexact search direction is computed based on the simultaneous solution of a set of shifted linear systems. The resulting method, named ARC_qK, enjoys a complexity bound of $O(\epsilon^{-3/(1+\zeta)})$ where $\zeta \in (0, 1]$ guides the accuracy of the system solves, and converges superlinearly. Our implementation is competitive with the method of Steihaug [24] and Toint [25] on the unconstrained CUTEst problems in terms of time, and number of objective and gradient evaluations. ARC_qK has a noticeable advantage in terms of number of Hessian-vector products.

We also discussed a variant of ARC_qK for nonlinear least-squares problems. Our complexity bound continues to apply if the Gauss-Newton Hessian approximation is accurate along the step.

Certainly, much work remains to be done to develop a reliable and efficient version of Algorithm 4.1, but the numerical results of this section speak in favor of our approach. Among possible improvements, we note that our current implementation solves all the shifted systems up front. Although this could be performed in parallel, there may be virtue to only solving a system when it is needed. The CG iterations could be put on hold when a shifted system has been solved to the desired accuracy, and resumed only if it is determined that a system corresponding to a larger value of the shift should be solved.

In practice, preconditioning CG is important if one is to develop a robust implementation. Of course, preconditioning is strongly application dependent, but generic (e.g., diagonal or banded) preconditioners can prove generally useful.

The numerical stability of the so-called *multishift* CGLS has been studied for some time in the literature and several variants have been proposed, each with its own numerical properties. In practice, implementations tend to depend on the condition number of $A^T A$ rather than that of A even though it is not the case when applying CGLS to solve a single shifted system. van den Eshof and Sleijpen [26] provide a review, numerous references, and an implementation of the multishift CGLS with favorable stability properties, although it is based on the standard CG rather than the Lanczos variant of CG. In this paper, we used the implementation of Frommer and Maass [10] and its stability properties should be properly evaluated.

A possible improvement over CGLS is to use a variant of LSQR [23] adapted to solve regularized linear least-squares problems with multiple values of the regularization parameter. However a disadvantage of CG and CGLS is that they do not reduce the residual norm $\|r_k\|$ monotonically. In the inexact-Newton context of the present research, an iterative method that reduces the residual norm monotonically could be preferable. Such methods include the method of conjugate residuals, also introduced by Hestenes and Stiefel [16] for symmetric positive-definite systems, or its MINRES variant of Paige and Saunders [22] for general symmetric systems, and LSMR of Fong and Saunders [9] for least squares. Variants of some of those methods to accommodate multiple shifts are described by Glässner et al. [11] and Jegerlehner [17].

On the optimization side, Gould et al. [13] suggest more sophisticated parameter update strategies for least-squares problems that could prove beneficial to ARC_qK.

References

[1] 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.

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

[3] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Complexity bounds for second-order optimality in unconstrained optimization. *J. Complexity*, 28(1):93–108, 2012.

[4] C. Cartis, N. I. M. Gould, and Ph. L. Toint. On the evaluation complexity of cubic regularization methods for potentially rank-deficient nonlinear least-squares problems and its relevance to constrained nonlinear optimization. *SIAM J. Optim.*, 23(3): 1553–1574, 2013.

[5] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*, volume 1 of *MPS/SIAM Series on Optimization*. SIAM, Philadelphia, USA, 2000.

[6] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program., Series B*, 29(2):201–213, 2002.

[7] J.-P. Dussault. ARCq: a new adaptive regularization by cubics. *Optim. Method Softw.*, 33(2):322–335, 2018.

[8] J.-P. Dussault. A unified efficient implementation of trust-region type algorithms for unconstrained optimization. *INFOR: Information Systems and Operational Research*, 0(0):1–20, 2019.

[9] D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput.*, 33(5):2950–2971, 2011.

[10] A. Frommer and P. Maass. Fast CG-based methods for Tikhonov-Phillips regularization. *SIAM J. Sci. Comput.*, 20(5):1831–1850, 1999.

[11] U. Glässner, S. Güsken, T. Lippert, G. Ritzenhöfer, K. Schilling, and A. Frommer. How to compute Green's functions for entire mass trajectories within Krylov solvers. *Int J Mod Phys C*, 07(05):635–644, 1996.

[12] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM J. Optim.*, 9(2):504–525, 1999.

[13] N. I. M. Gould, M. Porcelli, and Ph. L. Toint. Updating the regularization parameter in the adaptive cubic regularization algorithm. *Comput. Optim. Appl.*, 53(1):1–22, 2012.

[14] 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.*, 60:545–557, 2015.

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

[16] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49(6):409–436, 1952.

[17] B. Jegerlehner. Krylov space solvers for shifted linear systems. Technical Report IUHET-353, Indiana University, Department of Physics, Bloomington, IN, 1996. https://arxiv.org/abs/hep-lat/9612014.

[18] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.*, 45(4):255–282, 1950.

[19] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. and Statist. Comput.*, 4(3):553–572, 1983.

[20] D. Orban and A. S. Siqueira. JuliaSmoothOptimizers: Infrastructure and solvers for continuous optimization in Julia, 2019. juliasmoothoptimizers.github.io.

[21] J. M. Ortega. *Numerical analysis: A second course*. Number 3 in Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

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

[23] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[24] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983.

[25] Ph. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88. Academic press, 1981.

[26] J. van den Eshof and G. L. Sleijpen. Accurate conjugate gradient methods for families of shifted systems. *Appl. Numer. Math.*, 49(1):17–37, 2004. Numerical Algorithms, Parallelism and Applications.

Contents

GERAD and Département d'Informatique, Université de Sherbrooke, Sherbrooke, QC, Canada.
URL: www.dmi.usherb.ca/~dussault
Email address: Jean-Pierre.Dussault@USherbrooke.CA

GERAD and Mathematics and Industrial Engineering Department, Polytechnique Montréal, Montréal, QC, Canada.
URL: www.gerad.ca/~orban
Email address: dominique.orban@gerad.ca