Les Cahiers du GERAD ISSN: 0711–2440

Scalable adaptative cubic regularization methods

J.-P. Dussault
D. Orban

G–2015–109

October 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Scalable adaptative cubic regularization methods

Jean-Pierre Dussault [a]
Dominique Orban [b]

[a] *GERAD & Department of Mathematics, Université de Sherbrooke, Sherbrooke (Québec) Canada, J1K 2R1*

[b] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

jean-pierre.dussault@usherbrooke.ca
dominique.orban@gerad.ca

October 2015

Les Cahiers du GERAD
G–2015–109

Abstract: Adaptative cubic regularization (ARC) methods for unconstrained optimization compute steps from linear systems with a shifted Hessian in the spirit of the modified Newton method. In the simplest case, the shift is a multiple of the identity, which is typically identified by trial and error. We propose a scalable implementation of ARC in which we solve a set of shifted systems concurrently by way of an appropriate Krylov solver.

Key Words: Unconstrained optimization, trust-region algorithms, adaptative cubic regularization.

Acknowledgments: Research partially supported by NSERC Discovery Grants of both authors.

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 $\mathcal{C}^2$. Adaptive Cubic Regularization (ARC) algorithms, recently explored by Cartis, Gould, and Toint (2011a,b) are closely related to trust region (TR) methods (Conn, Gould, and Toint, 2000) 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.

Both ARC and TR algorithms make use of the quadratic model

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

where we adopt the convention that x, d are column vectors of $\mathbb{R}^n$ and $\nabla f(x)$ is a line vector (dual of $\mathbb{R}^n$).

At each iteration, ARC minimizes a cubic model (Griewank, 1981)

$$c_x^\alpha(d) := q_x(d) + \tfrac{1}{3\alpha}\|d\|^3, \tag{1.2}$$

where $\alpha > 0$ plays a role similar to the trust-region radius in TR methods.[1]

As in trust-region methods, solving (1.2) involves the solution of a shifted linear system

$$(\nabla^2 f(x) + \lambda I)d = -\nabla f(x). \tag{1.3}$$

In particular, building on the Steihaug-Toint approach, GLTR continues to explore the boundary of the ball once attained. At some matrix storage cost, an enhanced approximation becomes then available. For large scale applications, most of the subproblems may well be approximately solved without the refinment and thus reduce to the Steihaug-Toint solution. However, as developed in Cartis et al. (2011a), to apply this strategy for the ARC method requires matrix storage and computations from the very first iteration, which makes the use of the approach unlikely for large scale problems.

Dussault (2015) develops the ARC$_q$ variant, which uses the usual quadratic model but retains the cubic subproblem to compute regularized Newton steps and obtains simple proofs highlighting the key properties that ensure worst-case complexity.

In this paper, we propose ARC$_q$K, an implementation of ARC$_q$ that obtains approximate solutions to (1.2) using the shifted CG-Lanczos iterative method, a Lanczos implementation of the conjugate gradient algorithm that solves several shifted systems simultaneously proposed by Frommer and Maass (1999). The shifted CG-Lanczos uses the same number of matrix-vector products as the classical CG-Lanczos method. The only additional cost resides in a few scalar and vector operations for each value of the shift.

The rest of this paper is organized as follows. We first recall the ARC$_q$ algorithm and its 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. Before concluding, we report on numerical experience on problems with $n = 10,000$ and $100,000$ variables.

2 The ARC$_q$ algorithm

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

Theorem 2.1 (Dussault (2015)) *Let $\{x_k\}$ be the sequence generated by Algorithm 2.1. If $\{f(x_k)\}$ is bounded below and there exists a constant $L > 0$ such that $\|\nabla^2 f(x_k)\| \le L$ for all k, then every cluster point of $\{x_k\}$ satisfies the second order necessary optimality conditions.*

[1] α corresponds to $1/\sigma$ in the notation of Cartis et al. (2011b)

Algorithm 2.1 ARC$_q$ algorithm.

```
Model_Algorithm(x, α, f)
  { Given: x; }
  {   objective function f; }
  {   initial value for α. }
  repeat
    d ← Solve_Model(c, x, α)
    Δf  ←  f(x) − f(x + d)
    Δq  ←  q(0) − q(d)
    ρ  ←  Δf/Δq
    if (ρ < 0.25 ) then  α ← α/2   {Unsuccessful}
    else
      x ← x + d                     {Successful}
      if (ρ > 0.75 ) then
        α ← 2 ∗ α                   {Very successful}
  until ( termination_criterion)
  Result ← x
```

2.1 ARC$_q$ complexity bounds

A global minimizer d_k of $c_{x_k}^{\alpha_k}(d)$ satisfies (Cartis et al., 2011a, Theorem 3.1)

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

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

From (Cartis et al., 2011a, Theorem 3.1), we have $\lambda_k = \|d_k\|/\alpha_k$, which yields the following result.

Lemma 2.2 *Assume d_k is a global minimizer of $c_{x_k}^{\alpha_k}(d)$. Then,*

$$\Delta q_{x_k}(d_k) = f(x_k) - q_{x_k}(d_k) \geq \frac{1}{2\alpha_k}\|d_k\|^3.$$

The next result states that whenever $\nabla^2 f$ is Lipschitz continuous (with constant L_H), α_k is bounded away from zero.

Lemma 2.3 *If $\alpha_k < 4/L_H$, then $\alpha_{k+1} \geq \alpha_k$. Thus, $\alpha_k \geq 1/L_0 := \min(\alpha_0, 1/(8L_H))$ for all $k \geq 0$.*

The next result states how accurately (1.3) should be solved.

Lemma 2.4 $\|d_k\| \geq \kappa_g \sqrt{\|\nabla f(x_{k+1})\|}$ *for all successful iterations k, where*

$$\kappa_g := \sqrt{\frac{1}{\frac{1}{2}L_H + L_0}}.$$

With those three properties, we may obtain the worst case complexity bound.

Theorem 2.5 (ARC$_q$ complexity bound) *The maximum number of successful iterations of ARC$_q$ is $|\mathcal{S}_j| \leq \frac{4L_0}{\kappa_g^3 \epsilon^{\frac{3}{2}}}\left(f(x_0) - f(x_{low})\right) = L^s \epsilon^{-\frac{3}{2}}$. The maximum number of successful and unsuccessful iterations is*

$$|\mathcal{S}_j| + |\mathcal{U}_j| \leq \epsilon^{-\frac{3}{2}}\left(2L^s + \log(\alpha_0/\bar{\alpha})\right).$$

Proof. For any $k < k(\epsilon)$, $\nabla f(x_k) > \epsilon$. The lemmas 2.2 and 2.4 combine to obtain

$$f(x_k) - q_k(d_k) \geq \frac{\kappa_g^3}{L_0}\epsilon^{\frac{3}{2}}.$$

For successful iterates, $\frac{f(x_k)-f(x_{k+1})}{f(x_k)-q_k(d_k)} \geq \frac{1}{4}$ so that

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{4}(f(x_k) - q_k(d_k)) \geq \frac{\kappa_g}{4L_0}\epsilon^{\frac{3}{2}}.$$

Summing over all successful iterates before $k(\epsilon)$, and assuming that the monotonically decreasing sequence $\{f(x_k)\}$ is bounded below by f_{low}, we get (for $j = k(\epsilon)$)

$$f(x_0) - f_{\text{low}} \geq \sum_{k\in\mathcal{S}_j} (f(x_k) - f(x_{k+1})) \geq |\mathcal{S}_j| \frac{\kappa_g^3}{4L_0}\epsilon^{\frac{3}{2}},$$

which we use to bound $|\mathcal{S}_j| \leq \frac{4L_0}{\kappa_g^3\epsilon^{\frac{3}{2}}}(f(x_0) - f(x_{\text{low}})) = L^s\epsilon^{-\frac{3}{2}}$.

To bound the number of unsuccessful iterations, note that the algorithm flow ensures that

$$2\alpha_k \geq \alpha_{k+1}, \quad \forall k \in \mathcal{S}_j$$

and

$$\frac{1}{2}\alpha_i \geq \alpha_{i+1}, \forall i \in \mathcal{U}_j.$$

Therefore, $\alpha_0 2^{|\mathcal{S}_j|-|\mathcal{U}_j|} \geq \alpha_j$ so that $|\mathcal{S}_j| - |\mathcal{U}_j| \geq \log\left(\frac{\bar{\alpha}}{\alpha_0}\right)$ which yields

$$|\mathcal{U}_j| \leq \left\lceil |\mathcal{S}_j| + \log(\frac{\alpha_0}{\bar{\alpha}})\right\rceil \leq \left\lceil \epsilon^{-\frac{3}{2}}(L^s + \epsilon^{\frac{3}{2}}\log(\frac{\alpha_0}{\bar{\alpha}}))\right\rceil \tag{2.2}$$

so that for $\epsilon < 1$, the total number of iterations, both successful and unsuccessful

$$|\mathcal{S}_j| + |\mathcal{U}_j| \leq \epsilon^{-\frac{3}{2}}\left(2L^s + \log\left(\frac{\alpha_0}{\bar{\alpha}}\right)\right)$$

□

3 Shifted-systems formulation

Lemma 2.2 ensures $\Delta q_{x_k}(d_k) \geq \frac{1}{2}\lambda_k\|d_k\|^2$. In addition, Lemma 2.4 ensures that $\|g_{k+1}\| \leq \frac{1}{2}L_H\|d_k\|^2 + \lambda_k\|d_k\|$. The proof of Theorem 2.5 relies on the fact that $\lambda_k = \Omega(\|d_k\|)$ in Lemma 2.2 and that $\lambda_k = O(\|d_k\|)$ in Lemma 2.4. Thus, the complexity bound holds if $\lambda_k = \Theta(\|d_k\|)$, which occurs if d_k is computed as a global minimiser of the cubic model, for in that case $\lambda_k = \|d_k\|/\alpha_k$.

We now propose a way to compute values of λ_k and d_k that satisfy $\lambda_k = \Theta(\|d_k\|)$ as well as $\nabla^2 f(x)+\lambda_k I \succeq 0$. 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 finite precision arithmetic, we impose $10^{-15} \leq \lambda_i \leq 10^{15}$ for $i = 0, \ldots, m$. A simple choice consists in setting $\lambda_{i+1} = \beta\lambda_i$, for some $\beta > 0$. For instance, $\beta = 10$ yields the 31 values $\lambda_i = 10^i$ for $i = -15, \ldots, 15$. The computational feasibility of such a procedure is detailed in §4 and comes from the fact that an appropriate shifted CG-Lanczos implementation obtains all $m + 1$ (approximate) solutions of $d(\lambda_i)^T(\nabla^2 f(x) + \lambda_i I) \approx -\nabla f(x)^T$, or establishes that $\nabla^2 f(x) + \lambda_i I \not\succeq 0$, at modest extra cost compared to that of a single linear system.

During the simultaneous solution of the shifted systems, parameter values λ_i for which $\nabla^2 f(x) + \lambda_i I \not\succeq 0$ are eliminated. The solution of each remaining system is interrupted as soon as the residual $r_i := \nabla f(x_k) +$

$d_k^T(\nabla^2 f(x_k) + \lambda_i I)$ is sufficiently small. Among those remaining parameters, we select one that most closely satisfies $\alpha\lambda_i = \|d(\lambda_i)\|$. We select values λ_k and d_k so

$$\nabla f(x_k) + d_k^T(\nabla^2 f(x_k) + \lambda_k I) = r_k, \tag{3.1a}$$

$$d_k^T(\nabla^2 f(x_k) + \lambda_k I)d_k \geq 0 \tag{3.1b}$$

$$\frac{1}{\beta}\frac{\|d_k\|^\tau}{\alpha_k} \leq \lambda_k \leq \beta\frac{\|d_k\|^\tau}{\alpha_k}. \tag{3.1c}$$

As explained below, the parameter $0 < \tau \leq 1$ allows for inexact solves.

We now have all the ingredients to state the ARC$_q$K algorithm, keeping in mind that the step $d(\lambda)$ is computed by the shifted CG-Lanczos algorithm described and analyzed in §4.

One important difference between ARC$_q$ and ARC$_q$K is that in the latter, almost nothing is computed during unsuccessful iterations.

Algorithm 3.1 ARC$_q$K.

```
ARC_qK (x, α, f)
  { Given: x; }
  {    objective function f; }
  {    initial value for α; }
  {    shifts λ :   0 < λ_0 < ... λ_m < ∞. }
  repeat
    d(λ)  ←  solve(∇²f(x), ∇f(x), λ)
    success  ←  false
    i⁺  ←  min_{0≤i≤m} : (∇²f(x) + λ_i I) ≻ 0
    {  target value should be close to satisfy αλ = ‖d‖. }
    j  ←  arg min_{i⁺≤i≤m}(target(i) = |αλ_i − ‖d(λ_i)‖|)
    repeat
      d ← d(λ_j)
      Δf  ←  f(x) − f(x + d)
      Δq  ←  q(0) − q(d)
      ρ  ←  Δf/Δq
      if (ρ < 0.25 ) then
        {  Go to next value of the shift λ. }
        α  ←  ‖d_{j+1}‖/λ_{j+1}              {Unsuccessful}
        j  ←  j + 1
      else
        success  ←  true
        x  ←  x + d                          {Successful}
        if (ρ > 0.75 ) then
          α  ←  2 ∗ α                        {Very successful}
    until ( success)
  until ( termination_criterion)
  Result  ←  x
```

3.1 Worst-case complexity analysis

The complexity analysis follows the same pattern as the analysis of ARC$_q$. We obtain bounds similar to lemmas 2.2–2.4 from which the result will follow.

Lemma 3.1 *Assume λ_k and d_k satisfy (3.1a)–(3.1c). Then,*

$$\Delta q_{x_k}(d_k) = f(x_k) - q_{x_k}(d_k) \geq \frac{\|d_k\|^{1+\tau}\lambda_k}{2} - r_k d_k \geq \frac{\|d_k\|^{2+\tau}}{2\beta\alpha_k} - r_k d_k.$$

Proof. $f(x_k) - q_{x_k}(d_k) = -(\nabla f(x_k)d_k + \frac{1}{2}d_k^T\nabla^2 f(x_k)d_k)$ and using (3.1a) and (3.1c), $-(\nabla f(x_k)d_k + d_k^T\nabla^2 f(x_k)d_k) = \frac{\lambda_k\|d_k\|^{1+\tau}}{2} - r_k d_k$, which using (3.1b) and (3.1c) combines to $f(x_k) - q_{x_k}(d_k) \geq \frac{\|d_k\|^{2+\tau}}{2\beta\alpha_k} - r_k d_k$. □

By imposing the stopping tolerance on the residual r_k to ensure $r_k d_k \leq \frac{\|d_k\|^{1+\tau}\lambda_k}{4}$ or $r_k d_k \leq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}$, we get the bound $\Delta q_{x_k}(d_k) \geq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}$.

Corollary 3.2 *Under the same assumptions as lemma 3.1, assume further that*

$$r_k d_k \leq \frac{\|d_k\|^{1+\tau}\lambda_k}{4} \qquad \textit{or} \tag{3.2}$$

$$r_k d_k \leq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}; \tag{3.3}$$

then, $\Delta q_{x_k}(d_k) \geq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}$.

We next observe that whenever $\nabla^2 f$ is Lipschitz continuous (with constant L_H), α_k is actually bounded away from zero.

Lemma 3.3 *If $r_k d_k \leq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}$ and $\alpha_k < \frac{1}{16\beta L_H}$, then $\alpha_{k+1} \geq \alpha_k$. Thus, $\alpha_k \geq \min(\alpha_0, \frac{1}{32\beta L_H}) := \frac{1}{L_0^K}$ for all $k \geq 0$.*

Proof. By rewriting the expression $\rho_k = \frac{f(x_k)-f(x_k+d_k)}{f(x_k)-q_{x_k}(d_k)}$ as

$$\rho_k = 1 + \frac{q_{x_k}(d_k) - f(x_k + d_k)}{\Delta q_{x_k}(d_k)}$$

and noting from corollary 3.2 that $\Delta q_{x_k}(d_k) \geq \frac{\|d_k\|^{2+\tau}}{4\beta\alpha_k}$ while $q_{x_k}(d_k) - f(x_k + d_k) \leq L_H\|d_k\|^3$, we consider separately the cases $\|d_k\| \geq 1$ and $\|d_k\| \geq 1$. When $\|d_k\| \geq 1$, we get that $L_H\|d_k\|^3 \leq L_H$

When $\|d_k\| \leq 1$, $\rho_k > 0.75$ whenever $\alpha_k < \frac{1}{16\beta L_H}$ and thus $\alpha_k \geq \frac{1}{32\beta L_H}$. □

Next is a requirement to solve sufficiently precisely the Newton equation. We need to further restrict r_k such that for some $\xi > 0$,

$$\|r_k\| \leq \xi\|d_k\|^{1+\tau} \tag{3.4}$$

This does not readily follow from (3.2) or (3.3) since $r_k d_k$ could be negative, or close to zero while r_k would be large.

Lemma 3.4 *If $\|r_k\| \leq \xi\|d_k\|^{1+\tau}$, $\|d_k\| \geq \kappa_g^K\sqrt{\|\nabla f(x_{k+1})\|}$ for all successful iterations k, where*

$$\kappa_g^K := \sqrt{\frac{1}{\frac{1}{2}L_H + 2\beta L_0^K + \xi}}.$$

Proof. Denoting $g_{k+1} = g(x_k + d_k) = \nabla f(x_k + d_k)$ and g_k accordingly, we use a generalization of the fundamental theorem of integral calculus (Ortega, 1990, §8.1.2) to write

$$g_{k+1} = g_k + \int_0^1 d_k^T H(x_k + \tau d_k) d\tau$$

On the other hand, d_k satisfies (3.1a)

$$\nabla c_{x_k}^{\alpha_k}(d_k) = g_k + d_k(H(x_k) + \lambda_k I) = r_k$$

so that

$$\begin{aligned}
\|g_{k+1}\| = \|g_{k+1} - \nabla c_{x_k}^{\alpha_k}(d_k)\| &= \left\| \left(\int_0^1 d_k^T H(x_k + \tau d_k) d\tau \right) - d_k^T (H(x_k) + \lambda_k I) + r_k \right\| \\
&= \left\| \left(\int_0^1 d_k^T (H(x_k + \tau d_k) - H(x_k)) d\tau \right) - \lambda_k d_k^T + r_k \right\| \\
&\leq \|d_k\| \left\| \left(\int_0^1 L_H \tau d_k d\tau \right) \right\| + \|\lambda_k d_k\| + \|r_k\| \\
&\leq \left(\left(\frac{L_H}{2} + \frac{\beta}{\alpha_k} \right) \|d\|^{1-\tau} + \xi \right) \|d_k\|^{1+\tau} \\
&\leq \left(\left(\frac{L_H}{2} + \beta L_0^K \right) \|d\|^{1-\tau} + \xi \right) \|d_k\|^{1+\tau}
\end{aligned}$$

□

The complexity result follows directly from the lemmas above.

Theorem 3.5 (Complexity bound of ARC$_q$ K) *The maximum number of successful iterations of ARC$_q$ K is* $|\mathcal{S}_j| \leq \frac{4L_0^K}{{\kappa_g^K}^3 \epsilon^{\frac{\tau+2}{\tau+1}}} (f(x_0) - f(x_{low})) = L^s \epsilon^{-\frac{\tau+2}{\tau+1}}$. *The maximum number of successful and unsuccessful iterations is*

$$|\mathcal{S}_j| + |\mathcal{U}_j| \leq \epsilon^{-\frac{\tau+2}{\tau+1}} \left(2L^s + \log\left(\frac{\alpha_0}{\bar{\alpha}}\right) \right)$$

Remark 1 *The above result is optimal. For very large problems, it may happen that the tolerances required on r_k are impractical. Then, we may sacrifice optimality to develop a workable implementation.*

Remark 2 *The conditions on the residual r_k combine to $\|r_k\| \leq \frac{\lambda_k \|d_k\|^\tau}{4} \leq \frac{\beta \|d_k\|^{1+\tau}}{4\alpha_k}$. From an asymptotic point of view, this is coherent with usual truncated Newton criteria since close to a strong second order point, $\|d_k\| = \|\nabla f(x_k)\|$.*

3.2 Asymptotic analysis

The asymptotic analysis follows from the fact that close to a strong second order point, $\|d_k\| = \|\nabla f(x_k)\| = \|x_k - x^*\|$. For our computations, $\|d_k\| = \|\nabla f(x_k)\|$ is always true. Lemma 3.4 ensures $\|\nabla f(x_{k+1})\| = \|d_k\|^{1+\tau}$. Close to the second order point, $\|\nabla f(x_k)\| = \|x_k - x^*\|$ yields quadratic local convergence order.

4 CG-Lanczos implementation

We now describe how we solve a sequence of shifted linear systems simultaneously in a way that is consistent with the minimization of (1.2). Our implementation is an adaptation of Frommer and Maass (1999).

Algorithm 4.1 describes the CG-Lanczos with shifts implementation for a generic symmetric system $Mx = b$ with shifts λ_i, i.e.,

$$(M + \lambda_i I)x = b, \quad i = 1, \ldots, m. \tag{4.1}$$

In Algorithm 4.1, 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 $n \times (m+1)$ 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 $\boldsymbol{\lambda}$.

Algorithm 4.1 Lanczos-CG with shifts for (4.1)

Set $\boldsymbol{x}_0 = 0$, $\beta_0 v_0 = b$, $\boldsymbol{p}_0 = b$,
set $v_{-1} = 0$, $\boldsymbol{\sigma}_0 = \beta_0$, $\boldsymbol{\omega}_{-1} = 0$, $\boldsymbol{\gamma}_{-1} = 1$,
for $j = 0, 1, 2, \dots$ **do**
  $\delta_j = v_j^T M v_j$ *// Lanczos part of the iteration*
  $\beta_{j+1} v_{j+1} = M v_j - \delta_j v_j - \beta_j v_{j-1}$
  $\boldsymbol{\delta}_j = \delta_j + \boldsymbol{\lambda}$ *// CG part of the iteration in block form*
  $\boldsymbol{\gamma}_j = 1/(\boldsymbol{\delta}_j - \boldsymbol{\omega}_{j-1}/\boldsymbol{\gamma}_{j-1})$
  $\boldsymbol{\omega}_j = (\beta_{j+1}\boldsymbol{\gamma}_j)^2$
  $\boldsymbol{\sigma}_{j+1} = -\beta_{j+1}\boldsymbol{\gamma}_j\boldsymbol{\sigma}_j$
  $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \boldsymbol{\gamma}_j \boldsymbol{p}_j$
  $\boldsymbol{p}_{j+1} = \boldsymbol{\sigma}_{j+1} v_{j+1} + \boldsymbol{\omega}_j \boldsymbol{p}_j$

A few observations about Algorithm 4.1 are in order. Firstly, note that a single operator-vector product is required per iteration, and takes place in the Lanczos 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 for as long as negative curvature is not detected.

Finally, Algorithm 4.1 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.

Because we use adaptative stopping tolerances, not all systems will require the same number of iterations and we terminate 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 4.1. 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 $r_j = b - M x_j$ is updated cheaply via $r_j = \sigma_j v_j$, and ω_{j-1} and σ_j are scalars. Thus

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

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,

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

and therefore

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

Finally, using the update formula for γ_j, we may also write

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

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

5 Large scale numerical examples

In order to assess the scalability of our implementation, we performed some numerical experiments using adaptations and modifications of some CUTE problems Lukšan, Matonoha, and Vlček (2010). The Figure 5.1 illustrates the relative merit of our scalable implementation ARCqK with the L-BFGS-B solver.

In order to illustrate the really large scale applicability we picked an example, cragglvy and boosted its dimension to $n = 10\ 000\ 000$.

The statistics for ARC$_q$K suggest good scalability properties, at least on such an instance.

Table 5.1: ARC$_q$K shows remarkable consistency, probably due to a well conditioned Hessian

ARC$_q$K

n	m	#functions	#gradients	#hessian vector products
10 000 000	31	39	39	172
1 000 000	31	39	39	179
10 000 000	6	39	39	172

The statistics for L-BFGS-B show more severe increase of number of evaluations when the number of pairs of vectors is kept low.

Table 5.2: L-BFGS-B for this instance benefits from using more pairs in the limited memory strategy

L-BFGS-B

n	m	#functions	#gradients
10 000 000	6	352	355
1 000 000	6	303	306
10 000 000	31	145	148

Conclusion

We have introduced a new scalable implementation of the variant ARC$_q$ of the adaptative regularization by cubics. Our implementation is based on Levenberg-Marquardt shifted linear systems of equations which we solve all at once.

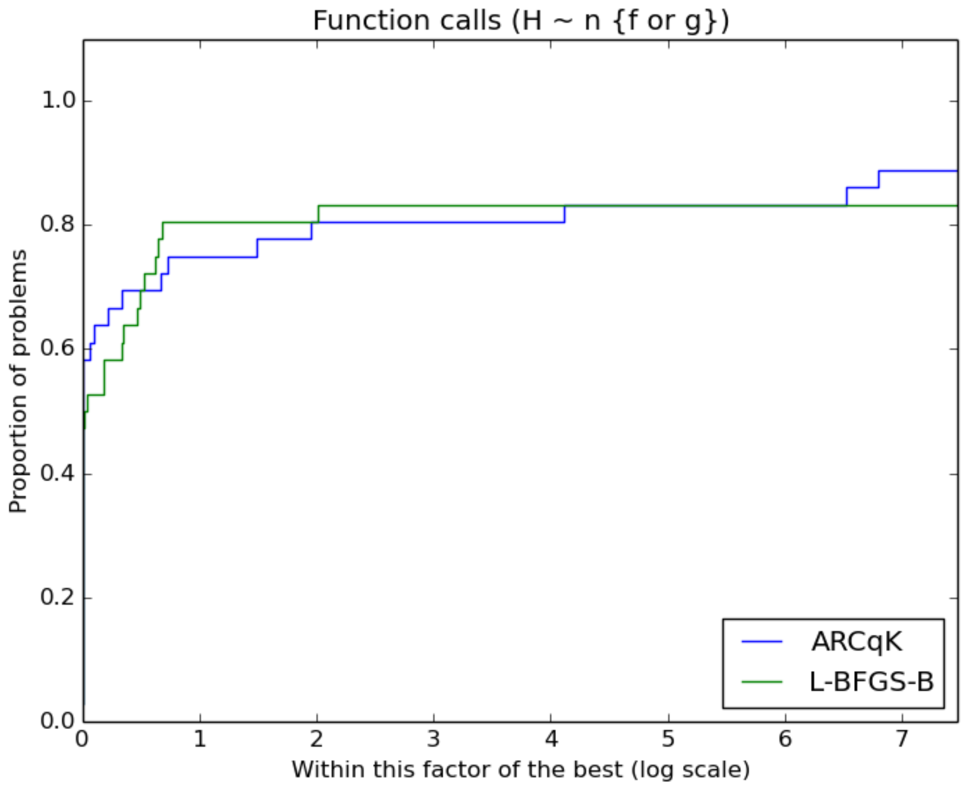

Figure 5.1: Comparison with L-BFGS-B using at most 20000 evaluations (functions, gradients, hessian vector products) with stopping criterion $\|\nabla f(x^*)\|_\infty \leq \max(10^{-10}\|\nabla f(x_0)\|_\infty, 10^{-6})$

References

C. Cartis, N.I.M. Gould, and Ph.L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. Mathematical Programming, 127(2):245–295, 2011a. DOI: 10.1007/s10107-009-0286-5.

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. Mathematical Programming, 130(2):295–319, 2011b. DOI: 10.1007/s10107-009-0337-y.

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. DOI: 10.1137/1.9780898719857.

J.-P. Dussault. Simple unified convergence proofs for trust region and a new ARC variant. Technical report, Université de Sherbrooke, 2015. www.optimization-online.org/DB_FILE/2015/06/4939.html.

A. Frommer and P. Maass. Fast CG-based methods for Tikhonov-Phillips regularization. SIAM Journal on Scientific Computing, 20(5):1831–1850, 1999. DOI: 10.1137/S1064827596313310.

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.

L. Lukšan, C. Matonoha, and J. Vlček. Modified CUTE problems for sparse unconstrained optimization. Technical Report 1081, Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží, 18207 Prague 8, Czech Republic, 2010.

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