# A fully adaptive trust-region method

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Adaptive second-order methods, such as Cartis, Gould, and Toint's adaptive cubic regularized Newton method (ARC), attempt to maintain strong convergence guarantees without depending on conservative estimates of problem properties such as Lipschitz constants. However, on close inspection, one can show existing 'adaptive methods' have theoretical guarantees with severely suboptimal dependence on problem properties such as the Lipschitz constant of the Hessian. For example, Cartis, Gould, and Toint's bound the number of iterations of their algorithm by $O(\Delta_f L^{3/2}\sigma_{\min}^{-1}\epsilon^{-3/2})$ where $\sigma_{\min}$ is the smallest observed regularization parameter. However, as $\sigma_{\min}$ could be arbitrarily small, this bound can be vacuous. Unfortunately, subsequent adaptive second-order methods are built on their ideas, and therefore also suffer from this issue. We present the first adaptive second-order method which circumvents this issue and requires at most $O(\Delta_f L^{1/2}\epsilon^{-3/2})+\tilde{O}(1)$ iterations to find an $\epsilon$-approximate stationary point, matching the optimal iteration bound up to an additive logarithmic term. Our method is a simple variant of a classic trust-region method and in our experiments performs competitively with both ARC and a classical trust-region method.

## 1 Introduction

Second-order methods are known for to quickly and accurately solve sparse nonconvex optimization problems that, for example, arise in optimal control [1], truss design [2], AC optimal power flow [3], and PDE constrained optimization [4]. Recently, there has also been a large push to extend second-order methods to tackle machine learning problems by coupling them with carefully designed subproblem solvers [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

Much of the early theory for second-order methods focused on showing fast local convergence and (eventual) global convergence [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]. These proofs of global convergence, unsatisfactorily, rested on showing at each iteration second-order methods reduced the function value almost as much as gradient descent [22, Theorem 4.5] [30, Theorem 3.2.], this is despite the fact that in practice second-order methods require far fewer iterations. In 2006, Nesterov and Polyak [31] partially resolved this inconsistency by introducing a new second-order method, cubic regularized Newton's method (CRN). Their method can be used to find stationary points of multivariate and possibly nonconvex functions $f : \mathbf{R}^n \to \mathbf{R}$. Their convergence results assumes the optimality gap

$$\Delta_f := f(x_1) - \inf_{x\in\mathbf{R}^n} f(x)$$

is finite and that the Hessian of $f$ is $L$-Lipschitz:

$$\|\boldsymbol{\nabla}^2 f(x) - \boldsymbol{\nabla}^2 f(x')\| \leq L\|x - x'\| \quad \forall x, x' \in \mathbf{R}^n \tag{1}$$

where $\|\cdot\|$ is the spectral norm for matricies and the Euclidean norm for vectors. If $L$ is known they guarantee their algorithm terminates with an $\epsilon$-approximate stationary point:

$$\|\boldsymbol{\nabla} f(x)\| \leq \epsilon$$

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

after at most

$$O(\Delta_f L^{1/2}\epsilon^{-3/2}) \tag{2}$$

iterations. For sufficiently small $\epsilon$, this improves on the classic guarantee that gradient descent terminates after at most $O(\Delta_f S\epsilon^{-2})$ iterations for $S$-smooth functions, thereby partially resolving this inconsistency of theory and practice.

However, CRN only achieves (2) if the Lipschitz constant of the Hessian is known. In practice, we rarely know the Lipschitz constant of the Hessian, and if we do it is likely to be a conservative estimate. With this in mind, many authors have studied how to build practical algorithms that achieve the convergence guarantees of CRN without needing to know the Lipschitz constant of the Hessian. For example, Cartis, Gould and Toint [32] develop adaptive cubic regularized Newton method (ARC), and guarantee to find an $\epsilon$-approximate stationary point after at most

$$O(\Delta_f L^{3/2}\sigma_{\min}^{-1}\epsilon^{-3/2}) \tag{3}$$

iterations[1], where $\sigma_{\min} \in (0, \infty)$ is the smallest regularization parameter used by their algorithm. This matches the $\epsilon$-dependence of (2). However, there are two issues with this bound. One is that the dependence on $L$ is suboptimal (note (2) has an $L^{1/2}$ term whereas (3) has an $L^{3/2}$). Moreover, one can construct examples where $\sigma_{\min}$ is arbitrarily small and therefore (3) is vacuous[2]. An ideal algorithm wouldn't incur this cost for adaptivity. This motivates the following definition.

**Definition 1.** *A method is **fully adaptive** on a problem class if, without knowing problem parameters, it achieves the same worst-case complexity bound as one obtains if problem parameters were known, up to a **problem-independent** constant-factor and additive lower order terms in the target accuracy $\epsilon$.*

Unfortunately, despite the enormous quantity of research on adaptive methods in the past decade [33, 32, 34, 35, 36, 37], they all use similar ideas to Cartis, Gould and Toint [32] and consequently all fail to be fully adaptive[3].

**Our contributions:**

1. We presents the first fully adaptive second-order method for finding stationary points of nonconvex functions with $L$-Lipschitz Hessians and bounded optimality gap. In particular, we prove our method finds an $\epsilon$-approximate stationary point after at most $O(\Delta_f L^{1/2}\epsilon^{-3/2}) + \tilde{O}(1)$ iterations.
2. Our method is a simple variant of the classic trust-region method [22]. It also accepts approximate solutions to the trust-region subproblems.
3. We show our trust-region method has quadratic convergence when it enters a region around a point satisfying the second-order sufficient conditions for local optimality.
4. Our algorithm appears promising in experiments. We test our method on the CUTEst test set [38] against other methods including ARC and a classic trust-region method. Our method appears be competitive in terms of number of iterations but uses slightly more gradient evaluations than ARC.

**Paper outline** The paper is structured as follows. Section 2 presents our trust-region method and contrasts it with existing trust-region methods. Section 3 presents our main result: a convergence bound for finding $\epsilon$-approximate stationary points that is fully adaptive to problems with Lipschitz continuous Hessian. Section 4 shows quadratic convergence of the method. Section 5 discusses the experimental results.

[1] Obtaining this bound does require carefully inspection of Cartis, Gould and Toint [32] (who highlighted only on the $\epsilon$-dependence of their bound). For simplicity of discussion we assume the ARC subproblems are solved exactly (i.e., $C = 0$, $\kappa_\theta = 0$), and that the initial regularization parameter satisfies $\sigma_0 = O(L + \sigma_{\min})$ (the bound only gets worse otherwise). We also consider only the bound on the number of Hessian evaluations, inclusion of the unsuccessful iterations (where cubic regularized subproblems are still solved) makes this bound even worse. Finally, we ignore problem-independent parameters $\gamma_1, \gamma_2, \gamma_3$, and $\eta_1$.

[2] One can get rid of the assumption that the regularization parameter is less than $\sigma_{\min}$ by forcing it to be above $\sigma_{\min}$ but this just makes the iteration bound $O\left(\left(\frac{L^{3/2}}{\sigma_{\min}} + \sigma_{\min}^{3/2}\right)\Delta_f\epsilon^{-3/2}\right)$ which is also not fully adaptive.

[3] It is often overlooked that Nesterov and Polyak [31] includes an adaptive method, but for similar reasons to Cartis, Gould and Toint [32] it is also not fully adaptive.

**Notation** Let $\mathbf{N}$ be the set of natural numbers (starting from one), $\mathbf{I}$ be the identity matrix, and $\mathbf{R}$ the set of real numbers. Throughout this paper we assume that $n \in \mathbf{N}$ and $f : \mathbf{R}^n \to \mathbf{R}$ is bounded below and twice-differentiable. We define $f_\star := \inf_{x \in \mathbf{R}^n} f(x)$ and $\Delta_f := f(x_1) - f_\star$.

# 2 Our trust-region method

## 2.1 Trust-region subproblems

As is standard for trust-region methods [22] at each iteration $k$ of our algorithm we build a second-order Taylor series approximation at the current iterate $x_k$:

$$M_k(d) := \frac{1}{2} d^T \boldsymbol{\nabla}^2 f(x_k) d + \boldsymbol{\nabla} f(x_k)^T d \tag{4}$$

and minimize that approximation over a ball with radius $r_k > 0$:

$$\min_{d \in \mathbf{R}^n} M_k(d) \text{ s.t. } \|d\| \leq r_k \tag{5}$$

to generate a search direction $d_k$. One important practical question is given a candidate search direction $d_k$, how can we verify that solves (5). For this one can use the following well-known Fact 1.

**Fact 1** (Theorem 4.1 [39])**.** *The direction $d_k$ exactly solves* (5) *if and only there exists $\delta_k \in [0, \infty)$ such that:*

$$\boldsymbol{\nabla} M_k(d_k) + \delta_k d_k = 0 \tag{6a}$$

$$\delta_k r_k \leq \delta_k \|d_k\| \tag{6b}$$

$$\|d_k\| \leq r_k \tag{6c}$$

$$\boldsymbol{\nabla}^2 f(x_k) + \delta_k \mathbf{I} \succeq 0 \tag{6d}$$

*which solves* (5)*.*

In practice, it is not possible to exactly solve the trust-region subproblem defined in (5), instead we only require that the trust-region subproblem is approximately solved. For our method, it will suffice to find a direction $d_k$ satisfying:

$$\|\boldsymbol{\nabla} M_k(d_k) + \delta_k d_k\| \leq \gamma_1 \|\boldsymbol{\nabla} f(x_k + d_k)\| \tag{7a}$$

$$\gamma_2 \delta_k r_k \leq \delta_k \|d_k\| \tag{7b}$$

$$\|d_k\| \leq r_k \tag{7c}$$

$$M_k(d_k) \leq -\gamma_3 \frac{\delta_k}{2} \|d_k\|^2 \tag{7d}$$

where $\delta_k$ denotes the solution for the above system and $\gamma_1 \in [0, 1)$, $\gamma_2 \in (1/\omega, 1]$, $\gamma_3 \in (0, 1]$. Setting $\gamma_1 = 0, \gamma_2 = 1, \gamma_3 = 1$ represents the exact version of these conditions. As Lemma 1 shows, exactly solving the trust-region subproblem gives a solution to the system (7). However, the converse it not true, an exact solution to (7) does not necessarily solve the trust-region subproblem. Nonetheless, these conditions are all we need to prove our results, and are easier to verify than a relaxation of (5) that includes a requirement like (6d) which needs a computationally expensive eigenvalue calculation.

**Lemma 1.** *Any solution to* (6) *is a solution to* (7) *with $\gamma_1 = 0, \gamma_2 = 1, \gamma_3 = 1$.*

*Proof.* The only tricky part is proving (7d). However, this can be shown using standard arguments: $M_k(d_k) = \frac{1}{2} d_k^T \boldsymbol{\nabla}^2 f(x_k) d_k + \boldsymbol{\nabla} f(x_k)^T d_k = -\frac{1}{2} d_k^T (\boldsymbol{\nabla}^2 f(x_k) + 2\delta_k \mathbf{I}) d_k \leq -\frac{\delta_k}{2} \|d_k\|^2$ where the second equality uses (6a) and the inequality (6d). □

## 2.2 Our trust-region method

An important component of a trust-region method is the decision for computing the radius $r_k$ at each iteration. This choice based on whether the observed function value reduction is comparable to the predicted reduction from the second-order Taylor series expansion $M_k$. In particular, given a search direction $d_k$ existing trust-region methods compute the ratio

$$\rho_k := \frac{f(x_k) - f(x_k + d_k)}{-M_k(d_k)} \tag{8}$$

and then increase $r_k$ if $\rho_k \geq \beta$ or decrease $r_k$ if $\rho_k < \beta$. Unfortunately, while intuitive, this criteria is provably bad, in the sense that one can construct examples of functions with Lipschitz continuous Hessians where any trust-region method that uses this criteria will have a convergence rate proportional to $\epsilon^{-2}$ [40, Section 3].

Instead of (8), we introduce a variant of the ratio by adding the term $\frac{\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|$ to the predicted reduction where $\theta \in (0, \infty)$ is a problem independent hyperparameter (we use $\theta = 0.1$ in our implementation). This requires the algorithm to reduce the function value more if the gradient norm at the candidate solution $x_k + d_k$, and search direction norm are big. In particular, we define our new ratio as:

$$\hat{\rho}_k := \frac{f(x_k) - f(x_k + d_k)}{-M_k(d_k) + \frac{\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|} \tag{9}$$

Our trust-region method is presented in Algorithm 1. The algorithm includes some other minor modification of classic trust-region methods [22]: we accept all search directions that reduce the function value, and update the $r_{k+1}$ using $\|d_k\|$ instead of $r_k$. We recommend contrasting our algorithm with [36, Algorithm 1] which is trust-region method with an iteration bound proportional to $\epsilon^{-3/2}$ but is more complex and not fully adaptive.

For the remainder of this paper $x_k$ and $d_k$ refer to the iterates of Algorithm 1.

**Function** `FLAT`$(f, x_1, r_1; \theta, \beta, \omega, \gamma)$**:**
  **Input requirements:** $r_1 \in (0, \infty)$, $x_1 \in \mathbf{R}^n$ ;
  **Problem-independent parameter requirements:** $\theta \in (0, 1), \beta \in (0, 1), \omega \in (1, \infty)$, $\frac{\beta\theta}{\gamma_3(1-\beta)} + \gamma_1 < 1$, $\gamma_1 \in [0, 1)$, $\gamma_2 \in (1/\omega, 1]$, $\gamma_3 \in (0, 1]$;
  **for** $k = 1, \ldots, \infty$ **do**
    Approximately solve the trust-region subproblem, i.e., obtain $d_k$ that satisfies (7) ;
    **if** $f(x_k + d_k) \leq f(x_k)$ **then**
      $x_{k+1} \leftarrow x_k + d_k$
    **else**
      $x_{k+1} \leftarrow x_k$
    **end**
    Compute $\hat{\rho}_k$ using (9) ;
    **if** $\hat{\rho}_k \geq \beta$ **then**
      $r_{k+1} \leftarrow \omega\|d_k\|$
    **else**
      $r_{k+1} \leftarrow \|d_k\|/\omega$
    **end**
  **end**
**end**

**Algorithm 1: F**ully **A**daptive **T**rust Region Method (FLAT)

## 3 Proof of full adaptivity on Lipschitz continuous functions

This section proves that our method is fully adaptive for finding approximate stationary points on $L$-Lipschitz functions. The core idea behind our proof is to get a handle on the size of $\|d_k\|$. In particular, if we can bound $\|d_k\|$ from below and $\hat{\rho}_k \geq \beta$ then the $\frac{\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|$ term guarantees that at iteration $k$ the function value is reduced by a large amount relative to the gradient norm $\|\nabla f(x_k + d_k)\|$.

Lemma 2 guarantees the norm of the gradient for the candidate solution $x_k + d_k$ lower bounds the size of $\|d_k\|$ under certain conditions. Note this bound on the gradient, i.e., (12) holds without us needing to know the Lipschitz constant of the Hessian $L$. The proof of Lemma 2 appears in Section A.1 and heavily leverages Fact 2.

**Fact 2** (Nesterov & Polyak 2006, Lemma 1 [31]). *If $\nabla^2 f$ is L-Lipschitz,*

$$\|\nabla f(x_k + d_k)\| \leq \|\nabla M_k(d_k)\| + \frac{L}{2}\|d_k\|^2 \tag{10}$$

$$f(x_k + d_k) \leq f(x_k) + M_k(d_k) + \frac{L}{6}\|d_k\|^3. \tag{11}$$

**Lemma 2.** *Suppose $\nabla^2 f$ is L-Lipschitz. If $\|d_k\| < \gamma_2 r_k$ or $\hat{\rho}_k \leq \beta$ then*

$$\|\nabla f(x_k + d_k)\| \leq c_1 L \|d_k\|^2 \tag{12}$$

*where $c_1 > 0$ is a problem-independent constant:*

$$c_1 := \max\left\{\frac{5 - 3\beta}{6(\gamma_3(1-\gamma_1)(1-\beta) - \beta\theta)}, \frac{1}{2(1-\gamma_2)}\right\}.$$

For the remainder of this section we will find the following quantities useful,

$$\underline{d}_\epsilon := \gamma_2 \omega^{-1} c_1^{-1/2} L^{-1/2} \epsilon^{1/2}$$
$$\bar{d}_\epsilon := \frac{2\omega}{\beta\theta} \cdot \frac{\Delta_f}{\epsilon}.$$

As we will show shortly in Lemma 4, after a short warm up period $\underline{d}_\epsilon$ and $\bar{d}_\epsilon$ represent lower and upper bound on $\|d_k\|$ (i.e., $\underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon$) as long as $\|\nabla f(x_k + d_k)\| \geq \epsilon$. But before presenting and proving Lemma 4 we develop Lemma 3 which is a stepping stone to proving Lemma 4. Lemma 3 shows that if $\|d_k\|$ is almost above $\bar{d}_\epsilon$ then the trust-region radius will shrink, and if $\|d_k\|$ is almost below $\underline{d}_\epsilon$ then the trust-region radius will grow (recall from Algorithm 1 that $\omega \in (1, \infty)$).

**Lemma 3.** *Suppose $\nabla^2 f$ is L-Lipschitz. Let $\epsilon \in (0, \infty)$ and $\|\nabla f(x_k + d_k)\| \geq \epsilon$ then*

1. *If $\|d_k\| > \bar{d}_\epsilon/\omega$ then $\|d_k\|/\omega = r_{k+1}$.*
2. *If $\|d_k\| < \omega\gamma_2^{-1}\underline{d}_\epsilon$ then $\gamma_2 r_k \leq \|d_k\| \leq r_k$ & $\omega\|d_k\| = r_{k+1}$.*

*Proof.* Proof for 1. We have

$$\|d_k\| > \bar{d}_\epsilon/\omega = 2\frac{\Delta_f}{\beta\theta\epsilon} \geq 2\frac{f(x_k) - f(x_{k+1})}{\beta\theta\epsilon} \tag{13}$$

where the first equality uses the definition of $\bar{d}_\epsilon$ and the second inequality uses $f(x_{k+1}) \geq \inf_{x \in \mathbf{R}^n} f(x)$ and $f(x_1) \geq f(x_k)$. Furthermore,

$$\hat{\rho}_k = \frac{f(x_k) - f(x_k + d_k)}{-M_k(d_k) + \frac{\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|} \leq \frac{f(x_k) - f(x_{k+1})}{\frac{\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|} \leq 2\frac{f(x_k) - f(x_{k+1})}{\theta\epsilon\|d_k\|} < \beta$$

where the first inequality follows from $-M_k(d_k) \geq 0$ and $f(x_{k+1}) \leq f(x_k + d_k)$, the second inequality follows from the fact that $\|\nabla f(x_k + d_k)\| \geq \epsilon$, and the third inequality uses (13). By inspection of Algorithm 1, if $\hat{\rho}_k < \beta$, then $\|d_k\|/\omega = r_{k+1}$.

Proof for 2. We will prove the result by contrapositive. In particular, suppose that $\neg(\gamma_2 r_k \leq \|d_k\| \leq r_k)$ or $\|d_k\|\omega \neq r_{k+1}$. Let us consider these two cases. If $\neg(\gamma_2 r_k \leq \|d_k\| \leq r_k)$ then as $\|d_k\| \leq r_k$ we have $\gamma_2 r_k > \|d_k\|$. If $\|d_k\|\omega \neq r_{k+1}$ then by inspection of Algorithm 1 we have $\hat{\rho}_k \leq \beta$. Therefore, in both these cases the premise of Lemma 2 holds. Now, by $\|\nabla f(x_k + d_k)\| \geq \epsilon$ and Lemma 2 we get

$$\epsilon \leq \|\nabla f(x_k + d_k)\| \leq c_1 L \|d_k\|^2$$

which implies $\|d_k\| \geq c_1^{-1/2} L^{-1/2} \epsilon^{1/2} = \gamma_2^{-1}\omega(\gamma_2\omega^{-1}c_1^{-1/2}L^{-1/2}\epsilon^{1/2}) = \gamma_2^{-1}\omega\underline{d}_\epsilon$. □

Now we show that the norm of the direction $d_k$, after some finite iteration $\underline{k}_\epsilon$, will be bounded below and above by $\underline{d}_\epsilon$ and $\bar{d}_\epsilon$ respectively. For that we first define:

$$K_\epsilon := \min\{\{k \in \mathbf{N} : \|\nabla f(x_k + d_k)\| \leq \epsilon\} \cup \{\infty\}\}$$

as the first iteration for which $\|\nabla f(x_k + d_k)\| \leq \epsilon$, and we also define:

$$\underline{k}_\epsilon := \min\{\{k \in \mathbf{N} : \underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon\} \cup \{K_\epsilon - 1\}\}$$

as the first iteration for which $\underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon$. An illustration of Lemma 4 is given in Figure 1. In particular, after a certain warm up period the direction norms can no longer rise above $\bar{d}_\epsilon$ or below $\underline{d}_\epsilon$. Broadly speaking, the idea behind the proof is that if $\|d_k\|$ is above $\bar{d}_\epsilon/\omega$ then at the next iteration $\|d_k\|$ decreases and on the other hand if $\|d_k\|$ is bellow $\underline{d}_\epsilon \omega \gamma_2^{-1}$ then at the next iteration it must increase.

**Lemma 4.** *Suppose $\nabla^2 f$ is L-Lipschitz and let $\epsilon \in (0, \infty)$. If $\underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon$ then $\underline{d}_\epsilon \leq \|d_j\| \leq \bar{d}_\epsilon$ for all $j \in [k, K_\epsilon) \cap \mathbf{N}$. Furthermore, $\underline{k}_\epsilon \leq 1 + \log_{\gamma_2 \omega}(\max\{1, \underline{d}_\epsilon/r_1, r_1/\bar{d}_\epsilon\})$.*

*Proof.* We begin by proving $\underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon$ then $\underline{d}_\epsilon \leq \|d_j\| \leq \bar{d}_\epsilon$ for $j \in [k, K_\epsilon) \cap \mathbf{N}$. We assume that $k < K_\epsilon$ otherwise our desired conclusion clearly holds. We split this proof into two claims.

Our first claim is that $\|d_k\| \leq \bar{d}_\epsilon$ implies $\|d_{k+1}\| \leq \bar{d}_\epsilon$. We split $\|d_k\| \leq \bar{d}_\epsilon$ into two subcases. If $\|d_k\| \leq \bar{d}_\epsilon/\omega$, then inspection of Algorithm 1 shows that $\|d_{k+1}\| \leq r_{k+1} \leq \|d_k\|\omega \leq \bar{d}_\epsilon$. If $\bar{d}_\epsilon/\omega \leq \|d_k\| \leq \bar{d}_\epsilon$, then Lemma 3.1 implies that $\|d_{k+1}\| \leq r_{k+1} \leq \|d_k\|/\omega \leq \bar{d}_\epsilon$.

Our second claim is that $\underline{d}_\epsilon \leq \|d_k\|$ implies $\underline{d}_\epsilon \leq \|d_{k+1}\|$. We split $\underline{d}_\epsilon \leq \|d_k\|$ into three subcases. If $\|d_{k+1}\| < \gamma_2 r_{k+1}$, then the contrapositive of Lemma 3.2 implies that $\|d_{k+1}\| \geq \underline{d}_\epsilon$. If $\gamma_2 r_{k+1} \leq \|d_{k+1}\| \leq r_{k+1}$ and $\underline{d}_\epsilon \leq \|d_k\| < \gamma_2^{-1}\omega \underline{d}_\epsilon$, then $\underline{d}_\epsilon < \gamma_2 \omega \underline{d}_\epsilon \leq \gamma_2 \omega \|d_k\| =_\star \gamma_2 r_{k+1} \leq \|d_{k+1}\|$ where $\star$ uses Lemma 3.2. If $\gamma_2 r_{k+1} \leq \|d_{k+1}\| \leq r_{k+1}$ and $\underline{d}_\epsilon \omega \gamma_2^{-1} \leq \|d_k\|$, then $\underline{d}_\epsilon \leq \frac{\gamma_2 \|d_k\|}{\omega} \leq_\star \gamma_2 r_{k+1} \leq \|d_{k+1}\|$ where $\star$ is from the update rule for $r_{k+1}$ in Algorithm 1.

By induction on the previous two claims we deduce if $\underline{d}_\epsilon \leq \|d_k\| \leq \bar{d}_\epsilon$ then $\underline{d}_\epsilon \leq \|d_j\| \leq \bar{d}_\epsilon$ for $j \in [k, K_\epsilon) \cap \mathbf{N}$.

Next, we prove that if

$$k \geq 1 + \log_{\omega\gamma_2}(\underline{d}_\epsilon/r_1)$$

then $\|d_k\| \geq \underline{d}_\epsilon$. If $\underline{d}_\epsilon \leq \|d_j\|$ for some $j \leq k$, then the result holds because as we already established $\underline{d}_\epsilon \leq \|d_k\| \Rightarrow \underline{d}_\epsilon \leq \|d_{k+1}\|$. On the other hand, if $\|d_j\| < \underline{d}_\epsilon$ for all $j \leq k$, then Lemma 3.2 implies that $r_{j+1} = \omega\|d_j\| \geq \omega\gamma_2 r_j$ which by induction gives $\|d_k\| \geq (\omega\gamma_2)^{k-1} r_1 \geq (\omega\gamma_2)^{\log_{\omega\gamma_2}(\underline{d}_\epsilon/r_1)} r_1 = \underline{d}_\epsilon$.

Finally, we prove that if

$$k \geq 1 + \log_\omega(r_1/\bar{d}_\epsilon)$$

then $\|d_k\| \leq \bar{d}_\epsilon$. If $\|d_j\| \leq \bar{d}_\epsilon$ for some $j \leq k$, then the result holds because as we already established $\|d_k\| \leq \bar{d}_\epsilon \Rightarrow \|d_{k+1}\| \leq \bar{d}_\epsilon$. On the other hand, if $\|d_j\| > \bar{d}_\epsilon$ for all $j \leq k$, then Lemma 3.1 implies that $r_{j+1} = \|d_j\|/\omega \leq r_j/\omega$ which by induction gives $\|d_k\| \leq \omega^{1-k} r_1 \leq \omega^{-\log_\omega(r_1/\bar{d}_\epsilon)} r_1 = \bar{d}_\epsilon$. □

Let

$$\mathcal{P}_\epsilon := \{k \in \mathbf{N} : \hat{\rho}_k \geq \beta, \underline{k}_\epsilon \leq k < K_\epsilon\}$$

which represents the set of iterations, before we find an $\epsilon$-approximate stationary point, where the function value is reduced a large amount compared with our target reduction, i.e., $\hat{\rho}_k \geq \beta$. Lemma 5 shows that there is a finite number of these iterations until the gradient drops below the target threshold $\epsilon$. The proof of Lemma 5 appears in Appendix A.2. Roughly, the idea of the proof is to use that, due our definition of $\hat{\rho}_k$, when $\hat{\rho}_k \geq \beta$ we always reduce the function value by at least $\frac{\beta\theta}{2}\|\nabla f(x_k + d_k)\|\|d_k\|$ and $\|d_k\|$ can be lower bounded by $\underline{d}_\epsilon$ using Lemma 4. As we cannot reduce the function value by a constant value indefinitely, we must eventually have $\|\nabla f(x_k + d_k)\| \leq \epsilon$.

**Lemma 5.** *Suppose $\nabla^2 f$ is L-Lipschitz and $\epsilon \in (0, \infty)$ then $|\mathcal{P}_\epsilon| \leq \frac{\bar{d}_\epsilon}{\underline{d}_\epsilon \omega} + 1 = \frac{2\omega c_1^{1/2}}{\beta\theta} \cdot \frac{\Delta_f L^{1/2}}{\epsilon^{-3/2}} + 1$.*

With Lemma 5 in hand we are now ready to prove our main result, Theorem 1. We have already provided a bound on the length of the warm up period, $\underline{k}_\epsilon$ (Lemma 4) and on the number of points with $\hat{\rho}_k \geq \beta$. Therefore, the only obstacle is to bound the number of points with $\hat{\rho}_k < \beta$. However, on these iterations we always decrease the radius by at least $\omega$ (see update rules in Algorithm 1), and therefore as $\|d_k\|$ is bounded below by $\underline{d}_\epsilon$, what goes up must come down — so we must have

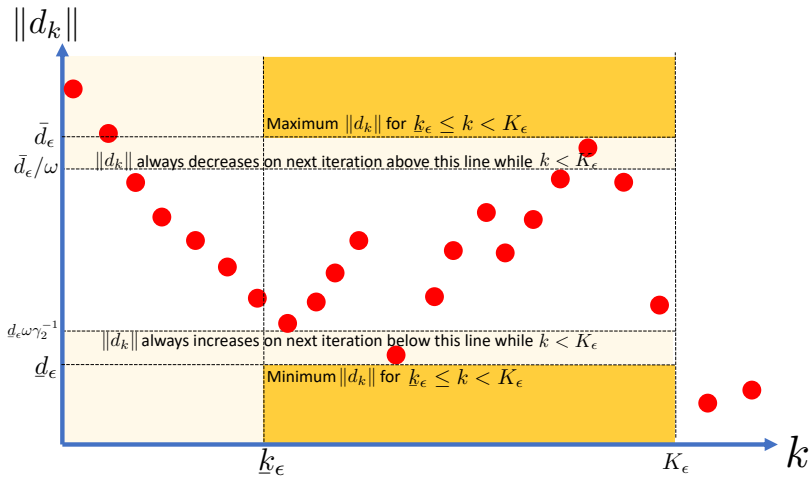



Figure 1: An example of a plausible sequence of iterates and the norms of their directions. Each red dot represents an iterate and its search direction norm. This illustrates Lemma 4.

a similar number of iterations where we increase the radius $r_k$, which by definition of Algorithm 1 only occurs if $\beta \ge \hat{\rho}_k$. Consequently, the number of iterations where $\beta < \hat{\rho}_k$ can be bounded by the number of iterations where $\beta \ge \hat{\rho}_k$ plus a $\tilde{O}(1)$ term. This is the crux of the proof of Theorem 1 which appears in Section A.3.

**Theorem 1.** *Suppose that $f$ is $L$-Lipschitz and bounded below with $\Delta_f = f(x_1) - f_\star$, then for all $\epsilon \in (0, \infty)$ there exists some iteration $k$ of Algorithm 1 with $\|\nabla f(x_k + d_k)\| \le \epsilon$ and*

$$k \le O\left(\frac{\Delta_f L^{\frac{1}{2}}}{\epsilon^{\frac{3}{2}}} + \log\left(\frac{\epsilon^{\frac{1}{2}}}{L^{\frac{1}{2}} r_1} + \frac{r_1 \epsilon}{\Delta_f} + 1\right) + 1\right)$$

*where $O(\cdot)$ hides problem-independent constant factors and $r_1$ is the initial trust-region radius.*

One drawback of Theorem 1 is that it only bounds the number of iterations to find a first-order stationary point. Many second-order methods in the literature show convergence to points satisfying the second-order optimality conditions [41, 32, 36, 18]. Of course, these methods are not fully adaptive. Therefore, in the future, it would be interesting to develop a method that provides a fully adaptive convergence guarantee for finding second-order stationary points.

## 4 Quadratic convergence when sufficient conditions for local optimality hold

**Theorem 2.** *Suppose $f$ is twice differentiable and for some $x_\star \in R^n$ the second-order sufficient conditions for local optimality hold ($\nabla f(x_\star) = 0$ and $\nabla^2 f(x_\star) \succ 0$). Under these conditions there exists a neighborhood $N$ around $x_\star$ and a constant $c > 0$ such that if $x_i \in N$ then there exist $\hat{k} \in N$ such that for all $k \ge \hat{k}$ we have $\|x_{k+1} - x_\star\| \le c\|x_k - x_\star\|^2 \le \frac{1}{2}\|x_k - x_\star\|$.*

The proof of Theorem 2 appears in Section B. It is a little tricker than typical quadratic convergence proofs for trust-region methods because in our method we have $\lim_{k\to 0} r_k \to 0$ whereas classical trust-region methods have $r_k$ bounded away from zero [22, Proof of Theorem 4.14]. Fortunately, one can show that asymptotically $r_k \ge \omega\|x_k - x_\star\|$ so the decaying radius does not interfere with quadratic convergence. In particular, the crux of proving Theorem 2 is proving the premise of Lemma 6 (as the conclusion of Lemma 6 is $r_k \ge \omega\|x_k - x_\star\|$). For this Lemma we define $\mathbf{diam}(X) := \sup_{x,x' \in X} \|x - x'\|$.

**Lemma 6.** *Let $N$ be a bounded set such that for all $x_k \in N$ we have $x_{k+1} \in N$, $\hat{\rho}_k \ge \beta$, and $\min\{\gamma_2 r_k, \|x_{k+1} - x_\star\|\} \le \|d_k\| \le \omega\gamma_2\|x_k - x_\star\|$. Suppose there exists $x_k \in N$ and let $i$ be the smallest index with $x_i \in N$, then $r_k \ge \omega\|x_k - x_\star\|$ for all $k \ge 2 + i + \log_{\gamma_2\omega}(\frac{\mathbf{diam}(N)}{\|d_i\|})$.*

*Proof.* Let $k \ge i$. By induction $x_k \in N$. By $\hat{\rho}_k \ge \beta$ and inspection of Algorithm 1 we have $r_{k+1} = \omega\|d_k\|$. Suppose that $\|d_k\| \ge \gamma_2 r_k$ for all $i \le k \le i + \ell$ then $\mathbf{diam}(N) \ge \|d_{k+1}\| \ge \gamma_2 r_{k+1} =$

Table 1: Geometric mean for total number of iterations and function and gradient evaluations per solver on 60 randomly generated instances with $\epsilon = 10^{-5}$ termination tolerance. Failures counted as the maximum number of iterations (10000) when computing the geometric mean.

| | **#iter** | **#function** | **#gradient** |
|---|---|---|---|
| **Newton trust-region [44]** | 480.1 | 482.3 | 482.3 |
| **Our method** | 308.1 | 309.6 | 309.6 |
| 95% **CI for ratio** | [1.24, 1.95] | [1.24, 1.94] | [1.24, 1.94] |

$\gamma_2\omega\|d_k\| = \gamma_2^\ell\omega^\ell\|d_i\|$. Rearranging gives $\ell \leq \log_{\gamma_2\omega}(\frac{\mathbf{diam}(N)}{\|d_i\|})$. Next observe that if $\|d_k\| < \gamma_2 r_k$ then $\|d_{k+1}\| \leq \omega\gamma_2\|x_{k+1} - x_\star\| \leq \omega\gamma_2\|d_k\| = (\omega\gamma_2/\omega)r_{k+1} = r_{k+1}$. By induction $\|d_k\| < \gamma_2 r_k$ for all $k > \ell$. Finally, observe that if $\|d_k\| < \gamma_2 r_k$ then $r_{k+1} = \omega\|d_k\| \geq \omega\|x_{k+1} - x_\star\|$. □

# 5 Experimental results

We test our algorithm on learning linear dynamical systems [42] and the CUTEst test set [43]. Appendix D contains the complete set of results from our experiments and additional instructions for reproducing the experiments. Our method is implemented in the attached open-source Julia module. The implementation uses a factorization and eigendecomposition approach to solve the trust-region subproblems (i.e., satisfy (7)). We perform our experiments using Julia 1.6 on a Linux virtual machine that has 8 CPUs and 16 GB RAM.

For these experiments, the selection of the parameters (unless otherwise specified) is as follow: $r_1 = 1.0$, $\beta = 0.1$, $\theta = 0.1$, $\omega = 8.0$, $\gamma_1 = 0.0$, $\gamma_2 = 0.8$, and $\gamma_3 = 1.0$. When implementing Algorithm 1 with some target tolerance $\epsilon$, we immediately terminate when we observe a point $x_k$ with $\|\boldsymbol{\nabla} f(x_k + d_k)\| \leq \epsilon$. This also includes the case when we check the inner termination criteria for the trust-region subproblem. The full details of the implementation are described in Appendix C.

## 5.1 Learning linear dynamical systems

We test our method on learning linear dynamical systems [42] to see how efficient our method compared to a trust-region solver. We synthetically generate an example with noise both in the observations and also the evolution of the system, and then recover the parameters using maximum likelihood estimation. Details are provided in Appendix D.1.

On this problem we compare our algorithm with the 'Newton trust-region method' that is available through the Optim.jl package [44]. The comparisons are summarized in Table 1.

## 5.2 Results on CUTEst test set

The CUTEst test set [43] is a standard test set for nonlinear optimization algorithms. We will be comparing our results with the results for ARC reported in [33, Table 1]. Catris et.al [33] report the results for three different implementations of their ARC algorithm. We limit our comparison to the ARC g-rule algorithm since it performs better than the other ARC approaches. We also run the Newton trust-region algorithm from the Optim.jl package [44].

We give the complete set of results in Appendix D.2. Our algorithm is stopped as soon $\|\boldsymbol{\nabla} f(x_k+d_k)\|$ is smaller than $10^{-5}$. For the Newton trust-region method [44] we also used as a stopping criteria a value of $10^{-5}$ for the gradient termination tolerance. We used 10000 as an iteration limit and any run exceeding this is considered a failure. This choice of parameters is in order to be consistent with [33].

As we can see from Table 2, our algorithm offers significant promise, requiring similar function evaluations (and therefore subproblem solves) to converge than Newton trust-region [44] and ARC, although the number of gradient evaluations is slightly higher than ARC. In addition, the comparison between these algorithms in term of total number of iterations and total number of gradient evaluations is summarized in Figure 2.

Table 2: Number of failures, geometric mean for total number of iterations and function and gradient evaluations per solver on 67 unconstrained instance from the CUTEst benchmark instances. Failures counted as the maximum number of iterations (10000) when computing the geometric mean.

| | **#failures** | **#iter** | **#function** | **#gradient** |
|---|---|---|---|---|
| **Our method** | 3 | 41.5 | 44.4 | 44.4 |
| **ARC with the g-rule [33]** | 1 | 38.1 | 38.1 | 26.6 |
| **Newton trust-region [44]** | 4 | 44.5 | 47.2 | 47.2 |

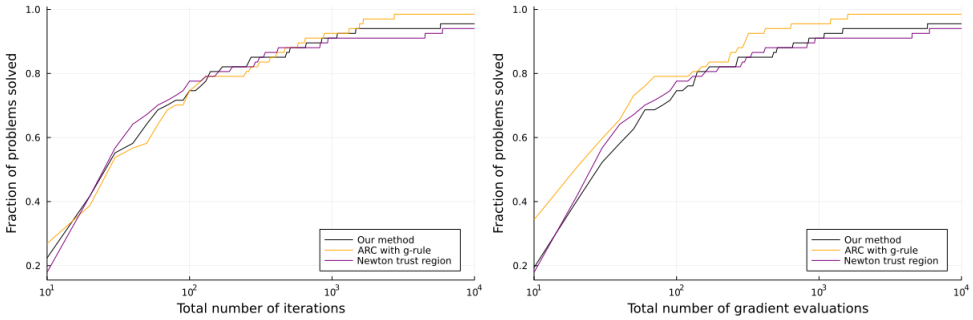


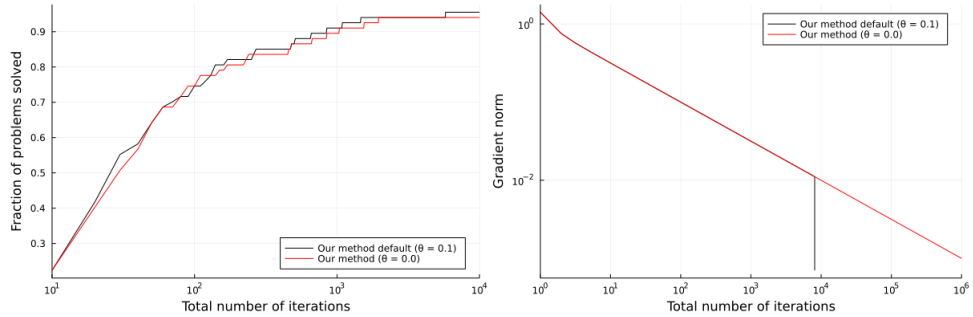

Figure 2: Fraction of problems solved on the CUTEst benchmark.

### 5.3 Convergence of our method with different $\theta$ values

To demonstrate the role of the additional $\frac{\theta}{2}\|d_k\|\|\nabla f(x_k + d_k)\|$ term we run experiments with different $\theta$ values. In particular, we contrast our default value of $\theta = 0.1$ with $\theta = 0$ which corresponds to not adding the $\frac{\theta}{2}\|d_k\|\|\nabla f(x_k + d_k)\|$ term to $\hat{\rho}_k$ (recall the discussion in Section 2.2).

In Figure 3a we rerun on the CUTEst test set (as per Section 5.2) and compare these two options. One can see the algorithm performs similarly with either $\theta = 0$ or $\theta = 0.1$.

In Figure 3b we test on the hard example from [40] which is designed to exhibit the poor worst-case complexity of trust-region methods (i.e., a convergence rate proportional to $\epsilon^{-2}$) if the initial radius $r_1$ is chosen sufficiently large (to achieve this we set $r_1 = 1.5$). We run this example with $\epsilon = 10^{-3}$. One can see that while for the first $\approx 10^4$ iterations the methods follow identical trajectories, thereafter $\theta = 0.1$ rapidly finds a stationary point whereas $\theta = 0.0$ requires two orders of magnitude more iterations to terminate. This crystallizes the importance of $\theta$ in circumventing this worst-case performance.




(a) Fraction of problems solved versus total number of iterations on the CUTEst test set

(b) Gradient norm versus total number of iterations on the example of [40] with $r_1 = 1.5$

Figure 3: Performance of our algorithm based on different $\theta$ values.

# References


[1] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[2] Klaus Schittkowski, Christian Zillober, and Rainer Zotemantel. Numerical comparison of nonlinear programming algorithms for structural optimization. *Structural Optimization*, 7(1):1–19, 1994.

[3] Stephen Frank, Steffen Rebennack, et al. A primer on optimal power flow: Theory, formulation, and practical examples. *Colorado School of Mines, Tech. Rep*, 2012.

[4] Lorenz T Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders. Large-scale pde-constrained optimization: an introduction. In *Large-Scale PDE-Constrained Optimization*, pages 3–13. Springer, 2003.

[5] Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, pages 1895–1904. PMLR, 2017.

[6] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic variance-reduced cubic regularized Newton methods. In *International Conference on Machine Learning*, pages 5990–5999. PMLR, 2018.

[7] Ching-pei Lee, Cong Han Lim, and Stephen J Wright. A distributed quasi-Newton algorithm for empirical risk minimization with nonsmooth regularization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1646–1655, 2018.

[8] Frank E Curtis, Michael J O'Neill, and Daniel P Robinson. Worst-case complexity of an sqp method for nonlinear equality constrained stochastic optimization. *arXiv preprint arXiv:2112.14799*, 2021.

[9] Frank E Curtis and Rui Shi. A fully stochastic second-order trust region method. *Optimization Methods and Software*, pages 1–34, 2020.

[10] Frank E Curtis, Katya Scheinberg, and Rui Shi. A stochastic trust region algorithm based on careful step normalization. *Informs Journal on Optimization*, 1(3):200–220, 2019.

[11] Vipul Gupta, Avishek Ghosh, Michał Dereziński, Rajiv Khanna, Kannan Ramchandran, and Michael W Mahoney. Localnewton: Reducing communication rounds for distributed learning. In *Uncertainty in Artificial Intelligence*, pages 632–642. PMLR, 2021.

[12] Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1):35–70, 2020.

[13] Chih-Hao Fang, Sudhir B Kylasa, Fred Roosta, Michael W Mahoney, and Ananth Grama. Newton-admm: A distributed gpu-accelerated optimizer for multiclass classification problems. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2020.

[14] Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate Newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] Sen Na, Michał Dereziński, and Michael W Mahoney. Hessian averaging in stochastic Newton methods achieves superlinear convergence. *arXiv preprint arXiv:2204.09266*, 2022.

[16] Peng Xu, Jiyan Yang, Fred Roosta, Christopher Ré, and Michael W Mahoney. Sub-sampled Newton methods with non-uniform sampling. *Advances in Neural Information Processing Systems*, 29, 2016.

[17] Rixon Crane and Fred Roosta. DINO: Distributed Newton-type optimization method. In *International Conference on Machine Learning*, pages 2174–2184. PMLR, 2020.

[18] Junyu Zhang, Lin Xiao, and Shuzhong Zhang. Adaptive stochastic variance reduction for subsampled Newton method with cubic regularization. *INFORMS Journal on Optimization*, 4(1):45–64, 2022.

[19] Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtárik. Stochastic newton and cubic newton methods with simple local linear-quadratic rates. *arXiv preprint arXiv:1912.01597*, 2019.

[20] Philip E Gill and Walter Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, 7(1):311–350, 1974.

[21] Jorge J Moré and Danny C Sorensen. On the use of directions of negative curvature in a modified newton method. *Mathematical Programming*, 16(1):1–20, 1979.

[22] Danny C Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

[23] Stefan Ulbrich. On the superlinear local convergence of a filter-SQP method. *Mathematical Programming*, 100(1):217–245, 2004.

[24] Luís N Vicente and Stephen J Wright. Local convergence of a primal-dual method for degenerate nonlinear programming. *Computational Optimization and Applications*, 22(3):311–328, 2002.

[25] Richard H Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[26] Lifeng Chen and Donald Goldfarb. Interior-point $l_2$-penalty methods for nonlinear programming with strong global convergence properties. *Mathematical Programming*, 108(1):1–36, 2006.

[27] Andrew R Conn, Nicholas IM Gould, Dominique Orban, and Philippe L Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical programming*, 87(2):215–249, 2000.

[28] Nicholas IM Gould, Dominique Orban, and PHL Toint. *An interior-point l1-penalty method for nonlinear optimization*. Citeseer, 2002.

[29] Andreas Wächter and Lorenz T Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[30] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science and Media*, 2006.

[31] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[32] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function-and derivative-evaluation complexity. *Mathematical programming*, 130(2):295–319, 2011.

[33] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[34] Coralia Cartis, Nicholas IM Gould, and Ph L Toint. An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. *IMA Journal of Numerical Analysis*, 32(4):1662–1695, 2012.

[35] Frank E Curtis, Daniel P Robinson, Clément W Royer, and Stephen J Wright. Trust-region newton-cg with strong second-order complexity guarantees for nonconvex optimization. *SIAM Journal on Optimization*, 31(1):518–544, 2021.

[36] Frank E Curtis, Daniel P Robinson, and Mohammadreza Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Mathematical Programming*, 162(1-2):1–32, 2017.

[37] Jean-Pierre Dussault. Simple unified convergence proofs for the trust-region and a new arc variant. Technical report, Technical report, University of Sherbrooke, Sherbrooke, Canada, 2015.

[38] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational optimization and applications*, 60(3):545–557, 2015.

[39] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[40] Coralia Cartis, Nicholas IM Gould, and Ph L Toint. On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems. *SIAM journal on optimization*, 20(6):2833–2852, 2010.

[41] Ernesto G Birgin and José Mario Martínez. The use of quadratic regularization with a cubic descent condition for unconstrained optimization. *SIAM Journal on Optimization*, 27(2):1049–1074, 2017.

[42] Moritz Hardt, Tengyu Ma, and Benjamin Recht. Gradient descent learns linear dynamical systems. *arXiv preprint arXiv:1609.05191*, 2016.

[43] D. Orban, A. S. Siqueira, and contributors. CUTEst.jl: Julia's CUTEst interface. `https://github.com/JuliaSmoothOptimizers/CUTEst.jl`, October 2020.

[44] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, 3(24), 2018.

[45] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and trends in machine learning*, 2015.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The comparisons with other algorithms show the limitations of our work.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sections 3 and 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sections 3 and 4
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and instructions to reproduce the results are attached in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We use CUTEst.jl and Optim.jl. For both we cite the creators.

(b) Did you mention the license of the assets? [Yes] See Appendixes D.1 and D.2

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is attached in the supplementary materials.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]